FRAUD DETECTION IN HIGHLY IMBALANCED BIG DATA WITH NOVEL AND EFFICIENT DATA REDUCTION TECHNIQUES

by

John T. Hancock III

A Dissertation Submitted to the Faculty of

The College of Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Florida Atlantic University

Boca Raton, FL

May 2024

FRAUD DETECTION IN HIGHLY IMBALANCED BIG DATA WITH NOVEL AND EFFICIENT DATA REDUCTION TECHNIQUES

by

John T. Hancock III

This dissertation was prepared under the direction of the candidate's dissertation advisor, Dr. Taghi M. Khoshgoftaar, Department of Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SUPERVISORY COMMITTEE:

Taghi Khoshgoftaar
Taghi Khoshgoftaar (Apr 10, 2024 11:40 EDT)

Taghi M. Khoshgoftaar, Ph.D.
Dissertation Advisor

Mehrdad Nojoumian (Apr 10, 2024 12:01 EDT)

Mehrdad Nojoumian, Ph.D.

Hari Kalva

Hari Kalva, Ph.D.
Chair, Department of Electrical Engineering and Computer Science

Dingding Wang
Dingding Wang (Apr 10, 2024 12:08 EDT)

Dingding Wang, Ph.D.

Stella Batalama

Stella N. Batalama, Ph.D.
Dean, The College of Engineering and Computer Science

Man Chon U (Apr 15, 2024 13:40 EDT)

Man Chon U, Ph.D.

Robert W. Stackman Jr., Ph.D.
Dean, Graduate College

April 16, 2024

Date

ACKNOWLEDGEMENTS

My heartfelt thanks go to my Advisor, Dr. Taghi M. Khoshgoftaar, for his pivotal guidance and abundant learning opportunities during my graduate journey. His invaluable support has significantly contributed to my success in academia and my professional life. I am also grateful to Dr. Mehrdad Nojoumian, Dr. Dingding Wang, and Dr. Man Chon U for their roles on my supervisory committee.

I'd like to extend my gratitude to the FAU Data Mining and Machine Learning Research Laboratory team for their unwavering support and insightful feedback during my doctoral studies. Dr. Richard Bauder, Dr. Joffrey Leevy, Dr. Karl Weiss, Dr. Naeem Seliya, Dr. Justin Johnson, Dr. Huanjing Wang, Dr. Richard Zuech, along with Robby Kennedy and Qianxin Liang, have been exceptional mentors and played a significant role in shaping my journey as a research scholar.

ABSTRACT

Author:	John T. Hancock III
Title:	Fraud Detection in Highly Imbalanced Big Data with Novel and Efficient Data Reduction Techniques
Institution:	Florida Atlantic University
Dissertation Advisor:	Dr. Taghi M. Khoshgoftaar
Degree:	Doctor of Philosophy
Year:	2024

The rapid growth of digital transactions and the increasing sophistication of fraudulent activities have necessitated the development of robust and efficient fraud detection techniques, particularly in the financial and healthcare sectors. This dissertation focuses on the use of novel data reduction techniques for addressing the unique challenges associated with detecting fraud in highly imbalanced Big Data, with a specific emphasis on credit card transactions and Medicare claims. The highly imbalanced nature of these datasets, where fraudulent instances constitute less than one percent of the data, poses significant challenges for traditional machine learning algorithms. This dissertation explores novel data reduction techniques tailored for fraud detection in highly imbalanced Big Data. The primary objectives include developing efficient data preprocessing and feature selection methods to reduce data dimensionality while preserving the most informative features, investigating various machine learning algorithms for their effectiveness in handling imbalanced data, and evaluating the proposed techniques on real-world credit card and Medicare fraud datasets.

This dissertation covers a comprehensive examination of datasets, learners, experimental methodology, sampling techniques, feature selection techniques, and hybrid techniques. Key contributions include the analysis of performance metrics in the context

of newly available Big Medicare Data, experiments using Big Medicare data, application of a novel ensemble supervised feature selection technique, and the combined application of data sampling and feature selection. The research demonstrates that, across both domains, the combined application of random undersampling and ensemble feature selection significantly improves classification performance.

The contributions presented advance the field of fraud detection. They add to the development of fraud detection systems. The proposed data reduction techniques offer practical solutions for tackling the challenges associated with imbalanced and high-dimensional data, offering a means to reduced financial losses, improved patient care, and increased trust in critical financial and healthcare domains. This dissertation offers practical implications for the financial and healthcare sectors by improving the detection of fraudulent transactions and Medicare Insurance claims. The proposed techniques may also be considered for general machine learning applications.

FRAUD DETECTION IN HIGHLY IMBALANCED BIG DATA WITH NOVEL AND EFFICIENT DATA REDUCTION TECHNIQUES

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

Fraud detection in highly imbalanced Big Data has emerged as a critical challenge across various industries, particularly in the financial and healthcare sectors. The rapid growth of digital transactions, coupled with the increasing sophistication of fraudulent activities, has necessitated the development of robust and efficient fraud detection techniques. This dissertation focuses on addressing the unique challenges associated with detecting fraud in highly imbalanced Big Data datasets, with a specific emphasis on credit card transactions and Medicare claims.

Credit card fraud has become a pervasive issue worldwide, with fraudsters employing advanced methods to exploit vulnerabilities in payment systems. In recent years, credit card transaction fraud is on the rise. In 2022, the FTC estimated consumer losses increased by 30% to $8.8 billion [135]. The growth of e-commerce and the increasing reliance on digital payments have further exacerbated the problem. Fraudulent activities not only result in substantial financial losses for banks and credit card companies but also erode consumer trust and confidence in the financial system.

Similarly, healthcare fraud, particularly in the Medicare program, has become a major concern in the United States [11,18]. Medicare, a federal health insurance program primarily serving individuals aged 65 and above, as well as some younger people with disabilities, is a prime target for fraudsters due to its vast size and complexity. In the year 2019, the United States Department of Justice (DoJ) succeeded in recovering approximately $3 billion from such fraudulent practices [31]. However, the exact magnitude of losses incurred is somewhat obscured by the Centers for Medicare and Medicaid Services' (CMS) categorization of "improper payments" which encompasses both fraudulent and non-fraudulent erroneous

payments [27]. Notably, the CMS reported improper payments in 2019 that exceeded the recovered amount by the DoJ by more than five-fold. Hence, there is a need for automated fraud detection to assist in identifying and prosecuting fraud [4]. Fraudulent activities include billing for services not provided, misrepresenting diagnoses to justify unnecessary tests or procedures, and falsifying patient records, among others.

Detecting fraud in credit card transactions and Medicare claims presents several challenges, primarily due to the highly imbalanced and noisy nature of the data [12,96]. In most cases, fraudulent transactions or claims constitute a small fraction of the total data points, often less than 1%. This imbalance makes it difficult for traditional machine learning algorithms to effectively learn and identify fraudulent patterns, as they tend to be biased towards the majority class [8]. In this dissertation, we identify legitimate transactions or claims as the majority class. Moreover, the sheer volume of data generated in these domains, often in the order of millions or billions of records, poses computational challenges and requires scalable solutions. For solutions to the additional challenges class noise presents in the classification of imbalanced big data please see [80, 102].

To illustrate the scale and imbalance of the data, consider the Credit Card Fraud Detection dataset used in this dissertation [99]. The dataset contains 284,807 credit card transactions, out of which only 492 (0.17%) are fraudulent. Similarly, the Medicare Part D dataset [158], which consists of prescription drug claims, has 174,357,611 records, with 679,946 (0.39%) labeled as fraudulent. These examples highlight the need for specialized techniques to handle the imbalance and the volume of the data effectively.

The imbalanced nature of credit card transaction and Medicare claims data poses significant challenges for machine learning algorithms. Most standard algorithms assume a balanced class distribution and aim to optimize overall accuracy. However, in the presence of imbalanced data, these algorithms tend to favor the majority class, resulting in high overall accuracy but poor detection of the minority class [67]. This bias towards the majority class can lead to a high number of false negatives, where fraudulent transactions or claims

are misclassified as legitimate, resulting in significant financial losses and potential legal consequences.

Furthermore, the imbalanced nature of the data can impact the training process of machine learning models [109, 166]. With a limited number of fraudulent examples, the models may struggle to learn the patterns and characteristics associated with fraudulent behavior. The lack of sufficient representative examples of the minority class can hinder the model's ability to generalize and accurately identify fraudulent instances in unseen data.

In addition to the challenges posed by imbalanced data [178, 179], the high dimensionality of credit card and Medicare datasets adds another layer of complexity. These datasets often contain numerous features, including categorical variables with high cardinality, such as provider types, procedure codes, and diagnosis codes. Traditional encoding techniques, such as one-hot encoding, can lead to a significant increase in the dimensionality of the data, making it computationally expensive and prone to overfitting [88].

High-dimensional data also introduces challenges in feature selection and interpretation. Identifying the most relevant features for fraud detection becomes a crucial task, as irrelevant or redundant features can introduce noise and reduce the performance of machine learning models [68, 108]. Moreover, interpreting the results and understanding the factors contributing to fraudulent behavior becomes more complex in high-dimensional spaces.

To address these challenges, this dissertation proposes novel data reduction techniques tailored for fraud detection in highly imbalanced Big Data. Therefore, we discuss our primary objectives here. Our first objective is to develop efficient data preprocessing and feature selection methods to reduce the dimensionality of the data while preserving the most informative features for fraud detection. This involves exploring techniques such as ensemble feature selection, and SHapley Additive exPlanations (SHAP) [39].

The second objective in our research is to investigate and compare various machine learning algorithms, including ensemble methods, for their effectiveness in handling imbalanced data and detecting fraudulent patterns [52]. This includes evaluating the performance

of ensemble techniques and one-class classifiers in the context of fraud detection [116].

In order to show the utility of accomplishing these objectives, we evaluate the proposed techniques on real-world credit card and Medicare fraud datasets and provide insights into their performance, scalability, and interpretability [57]. This involves conducting extensive experiments, analyzing the results using appropriate evaluation metrics, and providing practical recommendations for implementing these techniques in real-world fraud detection systems.

1.1 CONTRIBUTIONS

This dissertation encompasses a comprehensive exploration of machine learning methodology, yielding significant advancements in data reduction techniques including, feature selection, data sampling, hybrid data reduction techniques and evaluation of model performance. The key contributions of this research are listed here.

1. **Analysis of Performance Metrics:** We conduct experiments to demonstrate that the Area Under the Precision-Recall Curve (AUPRC) [20] metric provides more insightful information on the impact of Random Undersampling (RUS) on the classification of highly imbalanced Big Data than the Area Under the Receiver Operating Characteristic Curve (AUC) [19]. We present results comparing AUPRC and AUC metrics to illustrate how RUS impacts the classification performance in the context of highly imbalanced Big Data. This dissertation provides a comprehensive analysis of metrics for evaluating classifier performance in highly imbalanced Big Data contexts, showing that the choice of metric, AUPRC vs. AUC significantly influences the outcome of statistical tests and conclusions one might draw from them. These results provide actionable insights for practitioners, especially in fields where false positives have significant implications, suggesting the prioritization of AUPRC as a more informative metric in such contexts.

2. **Large-scale Data Utilization:** Over the course of our research, we have reported results from experiments involving a dataset with approximately 175 million instances, notably using Medicare Part D data which became available in 2021, highlighting the novelty and scale of the data employed. Moreover, we present results from experiments with other large datasets. Due to the availability of a high performance computing environment, we are able to conduct experiments with Big Data that are not feasible on smaller systems [76].

3. **Contribution to Medicare Fraud Detection:** By focusing on Medicare insurance claims data, the research contributes valuable insights into the domain of automated Medicare fraud detection, potentially enabling more effective identification of fraudulent activities and thereby facilitating substantial savings.

4. **Novel Ensemble Supervised Feature Selection Technique:** Our research group developed an ensemble supervised feature selection method for building explainable machine learning models, applied to Medicare fraud detection. This line of research led to the introduction of an innovative ensemble supervised feature selection method, leveraging six different machine learning algorithms to generate a comprehensive feature ranking, optimizing the performance of models for Medicare fraud detection.

5. **Novel Application of SHAP for Data Reduction:** The work we document in this dissertation shows how we developed an implementation of SHAP as a feature selection technique with for classifiers such as One-Class Support Vector Machine, specifically for fraudulent credit card transaction, and Medicare insurance fraud identification.

6. **Apply One-Class GMM:** For practitioners facing a one-class scenario – where only data of one class is available for experimentation – we show that One-Class Gaussian Mixture Models (GMM) built with SHAP-selected features perform significantly better than those without feature selection. To the best of our knowledge, this work is

the first to apply SHAP-based data reduction to improve One-Class Gaussian Mixture Models for credit card fraud detection and Medicare fraud detection.

7. **Synergistic Application of Data Sampling and Feature Selection:** Over the course of the studies conducted and documented in this dissertation, we performed an extensive evaluation of the impact of RUS and a novel ensemble feature selection method. We evaluate these methods both individually and in combination, and assess their impact on the classification performance of models built to classify imbalanced Big Medicare Data. Cognizant of our contribution regarding the efficacy of AUPRC as a robust metric for imbalanced data scenarios, we utilize AUPRC as the primary performance metric. We demonstrate that the combined application of RUS and the ensemble feature selection technique significantly improves the classification performance compared to using all available data and features, leading to more efficient and effective Medicare fraud detection models.

8. **Reduction in Model Training Time and Computational Resources:** An added benefit of the data reduction techniques we develop and apply in our dissertation is that they yield reductions in model training time and computational resource consumption, offering practical benefits for large-scale Medicare fraud detection tasks.

9. **Adaptability Across Different Label Availability Scenarios:** Our research also makes a contribution in application of SHAP-based feature analysis techniques in three distinct label availability scenarios: unlabeled, one-class, and binary-class. We provide a detailed comparative analysis of the features, by SHAP feature importance, in the different label availability scenarios, offering insights into the importance and relevance of specific features across various models and scenarios. Moreover, the analysis technique we use applies to the wide range of datasets that SHAP is applicable to.

1.2 DISSERTATION STRUCTURE

The dissertation is organized as follows:

Chapter two provides an overview of the datasets used in this study, including the Credit Card Fraud Detection dataset [99], and the Medicare Part B [162], Part D [158], and DMEPOS [156] datasets. It describes the data compilation process, feature engineering, and the challenges associated with handling highly imbalanced and high-dimensional data. This chapter lays the foundation for understanding the characteristics and complexities of the datasets used throughout the dissertation.

Chapter three introduces the machine learning algorithms employed in this dissertation, including ensemble methods such as Random Forest [22], Extremely Randomized Trees [35], and Gradient Boosting. In Chapter three, we cover the strengths and weaknesses of each algorithm in the context of fraud detection and imbalanced data. This chapter also introduces the evaluation metrics used to assess the performance of these algorithms, taking into account the specific requirements of imbalanced classification tasks.

Chapter four presents the experimental methodology, including cross-validation techniques, performance metrics, and statistical tests used to evaluate the effectiveness of the proposed data reduction techniques. It describes the experimental setup, and the statistical methods employed to assess the significance of the results. This chapter provides a rigorous framework for evaluating the performance of the proposed techniques and ensuring the reliability of the findings.

Chapter five focuses on sampling techniques for handling imbalanced data, specifically RUS and One-Class Sampling. It investigates the impact of these techniques on the performance of various classifiers and provides recommendations for their application in fraud detection tasks. This chapter aims to provide insights into the trade-offs and considerations involved in employing sampling techniques for imbalanced data.

Chapter six explores feature selection techniques, including ensemble feature selection and SHAP, to identify the most informative features for fraud detection. It discusses the

importance of feature selection in improving model interpretability and reducing computational complexity. This chapter presents a comprehensive analysis of the selected features and their contributions to the fraud detection models.

Chapter seven introduces hybrid techniques that combine sampling and feature selection methods to further enhance the performance of fraud detection models. It presents a comparative analysis of different hybrid data reduction approaches and their effectiveness in handling highly imbalanced Big Data. This chapter aims to provide a holistic view of the synergistic effects of combining multiple data reduction techniques for improved fraud detection performance.

Chapter eight concludes the dissertation by summarizing the key findings, contributions, and future research directions in the field of fraud detection using machine learning techniques. It highlights the practical implications of the proposed data reduction techniques for the financial and healthcare industries, as well as other domains facing similar challenges with imbalanced Big Data. This chapter also discusses the limitations of the current work and provides recommendations for future research endeavors.

The research presented in this dissertation has significant implications for the financial and healthcare industries, as well as other domains facing similar challenges with imbalanced Big Data. By developing effective data reduction techniques and machine learning models for fraud detection, this work aims to contribute to the ongoing efforts to combat fraudulent activities, protect consumers, and maintain the integrity of financial and healthcare systems. The proposed techniques have the potential to enhance the efficiency and accuracy of fraud detection systems, ultimately leading to reduced financial losses, improved patient care, and increased trust in these critical domains.

In conclusion, this dissertation addresses the pressing issue of fraud detection in highly imbalanced Big Data, with a focus on credit card transactions and Medicare insurance claims. Through the development of novel data reduction techniques and the investigation of advanced machine learning algorithms, this work aims to provide practical solutions for

tackling the challenges associated with imbalanced and high-dimensional data. The insights and contributions presented in this dissertation have the potential to significantly advance the field of fraud detection and contribute to the development of more robust and effective fraud detection systems.

CHAPTER 2

DATASETS AND COMPILATION

2.1 INTRODUCTION

In this chapter we discuss the highly imbalanced datasets used in this dissertation. We report summary data on the datasets used in this study in Table 2.1. The ratio of the size of the minority class to the size of the majority class is less than one percent in all cases. The class imbalance presents a challenge for Machine Learning algorithms, since it is easy for a model to become biased to the majority class. We cover research work done with seven datasets. Six of the datasets pertain to Medicare fraud detection, and one pertains to Credit Card fraud detection. For each part of the Medicare program, Part D, Part B, and DMEPOS, we have two datasets: aggregated and non-aggregated. Therefore, we describe data preprocessing required for the experiments with non-aggregated and aggregated data covered in this dissertation. Earlier experiments with non-aggregated data were time and computing resource intensive. Therefore, we conducted later experiments with aggregated data. The Credit Card Fraud dataset described here is used as it is provided from the Kaggle.com website. No preprocessing is required.

The Medicare datasets described in this dissertation are compiled from information provided by the CMS and the United States Office of Inspector General (OIG). We utilize three sources for data from the CMS. They are all Medicare Health insurance plans. One plan is known as Part D, the second is known as Part B, and the third is known as an acronym, DMEPOS, which stands for Durable Medical Equipment, Prosthetics, Orthotics and Supplies. In the context of health insurance, a plan is simply the agreement between the insurer and the insured as to what things are covered under the insurance policy.

Part D covers prescription medications, and Part B covers treatments and procedures. DMEPOS covers equipment procedures related to use of the equipment. The CMS makes different raw data available for all programs, however, we use the same technique to compile all sources into datasets suitable for supervised machine learning. This technique also involves data from the OIG which we use for labeling. For more on techniques for labeling insurance claims data please see [1, 7, 16]. The data from the OIG is known as the List of Excluded Individuals and Entities (LEIE). We provide a section that describes the LEIE. After compilation, the same process is used for labeling data for each Medicare plan. We describe that process as well here. First we describe the Kaggle Credit Card Fraud Detection dataset, then we move on to describe the Medicare datasets. Finally, we discuss the CatBoost [141] encoding technique for handling categorical data [184].

Table 2.1: Summary of Datasets

Dataset	Instance Count	# Minority Class	Ratio
Credit Card Fraud	284,807	492	0.1730%
Part D	174,357,611	679,946	0.3931%
Part B	67,856,547	128,141	0.1892%
DMEPOS	12,268,795	53,425	0.4374%
Part D Aggregated	5,344,106	3,700	0.0693%
Part B Aggregated	8,669,497	3,954	0.0456%
DMEPOS Aggregated	2,056,075	1,809	0.0880%

2.2 CREDIT CARD FRAUD DATASET

One dataset that is the subject of research in this dissertation is the Kaggle Credit Card Fraud Detection dataset [99], hereafter referred to as the Credit Card Fraud dataset. The dataset is the result of a collaborative effort between the machine learning Group at the Universitè Libre de Bruxelles and Worldline. It is publicly available for download from the Kaggle.com website.

The Credit Card Fraud data has a total of 31 attributes. We consider 29 of the attributes

to be independent variables that are useful for machine learning. We define two terms to help explain the attributes of the Credit Card Fraud data: *concrete*, and *abstract*. Concrete attributes are attributes that are clearly defined and directly relatable to something in everyday human experience. For example, transaction amount, is a concrete attribute. Transaction amount is directly relatable to something in human experience. It is some amount of money used in the exchange of goods or services. Abstract attributes are not defined in terms of things directly relatable to human experience. For example, an attribute of a dataset that is the result of applying PCA to some unknown data is abstract. Two of the attributes of the Credit Card Fraud dataset are concrete. One of the concrete attributes, Time, is not used since it is equivalent to a unique identifier. Unique identifiers pose a problem for many Machine Learning algorithms because algorithms tend to memorize them as perfect predictors of the training data, and fail to generalize and perform well on the test data. The Time attribute holds the number of seconds elapsed since the first transaction in the dataset. The dataset has one other concrete attribute, Amount, which is simply the transaction amount. It ranges in value from 0.0 to 25,691.16. We do not know the units for the Amount attribute. The remaining 28 attributes are abstract. They are the result of applying principal component analysis (PCA) [175] to an unknown source of numeric data. Since there are no easily understandable, natural language descriptions of the features V1 through V28, descriptive statistics of these attributes are listed in Table 2.2. An inspection of the descriptive statistics of features V1 through V28 shows that their mean values are all approximately zero, and they range in values from approximately -100 to 100. However, their 25% to 75% percentile ranges are much smaller, generally in the interval from -1 to 1.

The Credit Card Fraud dataset is highly imbalanced; 492 out of 284,807 transactions are labeled as fraudulent (positive), and the remaining transactions are labeled as not-fraudulent (negative). The dataset has a Class attribute, which we treat as the dependent variable in the dataset. Instances of the negative class have a Class value of 0, and instances of the positive class have a Class value of 1.

In summary, the Credit Card Fraud dataset is a valuable resource for research on techniques for working with highly imbalanced data. Since the Credit Card Fraud dataset is derived from actual transactions, models tested on this dataset may offer practical solutions to real-world problems in the financial industry. Hence, we find this dataset to be a good choice for our experiments.

2.3 MEDICARE DATASETS

Another application domain of our dissertation is automated Medicare insurance fraud detection. Medicare is the United States public health insurance program, primarily dedicated to individuals aged 65 and over. The organization responsible for the Medicare program is the Centers for Medicare and Medicaid Services (CMS). To foster research, the CMS maintains a repository of publicly available Medicare insurance claims data. We use data from three different sources in this study. Data from each source has unique attributes. The smallest dataset we use is from a section of the CMS website titled "Medicare Durable Medical Equipment, Devices & Supplies – by Referring Provider and Service" (DMEPOS) [156]. The CMS provides a dataset of approximately 12 million instances from this data. The next largest data we use is from a section of the CMS website "Medicare Physician & Other Practitioners – by Provider and Service" (Part B) [162]. The CMS provides a dataset with approximately 68 million instances from the Part B data. Finally, the largest data we use is from a section of the CMS website titled "Medicare Provider Utilization and Payment Data: Part D Prescriber" (Part D) [158]. The CMS provides a dataset of nearly 175 million instances from the Part D data. The CMS regularly adds to each of these data sources, which lends them the aspects of volume and velocity that characterizes Big Data [32]. The volume and velocity of the Part B, Part D, and DMEPOS data also reflects the quantity and speed at which the CMS receives insurance claims from healthcare providers.

The most recent CMS data we use in constructing the Medicare datasets first became available in 2021. The latest Part B, Part D, and DMEPOS data all spans the years 2013

through 2019. We obtain the Part D, Part B and DMEPOS data from on-line sources. The CMS website offers a user interface as well as an application programming interface for examining these datasets, and to carry out rudimentary exploration of data. We acquired the Medicare datasets from the CMS site in comma-separated format files. These files form the basis of our datasets. The files are available to the public, for download. We use data spanning the years 2013–2019. The CMS provides supplementary documentation that explains the Medicare data. These are the CMS-provided data dictionaries [154, 155, 157, 159, 161, 164]. The datasets used in this study are compiled in the manner described by Johnson and Khoshgoftaar in [97]. A key enhancement made to the data compiled by Johnson and Khoshgoftaar is that categorical features are encoded with CatBoost Encoding or LightGBM's built-in encoding [101] for categorical features.

2.3.1 Non-aggregated Part B Data

The Part B data is the second-largest dataset used in this dissertation. The Part B data describes treatments and procedures that a provider performs for patients. One record of the Part B data represents a summary of all the times a provider rendered a particular treatment or procedure for their patients for the year. The treatment or procedure is identified by the Healthcare Common Procedure Coding System (HCPCS) code listed in the record. HCPCS codes are instrumental in the categorization and standardization of healthcare procedures, particularly focusing on the identification of medical products, supplies, and ancillary services. For extensive research on techniques for encoding HCPCS values for Medicare data classification see [86, 87, 94]. Table 2.3 contains names and descriptions of elements of the Part B data we use in this study.

The definitions from the Part B data dictionary are as given in [163]. We add information about the number of possible values for categorical features. We use attributes of the Part B data based on whether they could be used as unique identifiers. We eschew unique identifiers as features for machine learning models since there is a risk that the model will

memorize these identifiers and fail to generalize. We discard the NPI and location data since they would serve as the equivalent of a unique identifier for the provider. The final number of instances we obtain for the Part B data is 67,856,547. The fraction of instances of the minority class in the Part B data is 0.0019. This concludes our discussion of the Part B non-aggregated data. Next we cover the aggregated Part B data.

2.3.2 Aggregated Part-B Data

We use two sources to compile the aggregated Part B data, that differ in specificity. The first source of the Part B data, which we refer to as the "provider-service-level Part B data" is Medicare Physician & Other Practitioners – by Provider and Service [162]. This is the same source used to compile the non-aggregated Part B data. The second source, which we refer to as the "provider-level Part B data" is Medicare Physician & Other Practitioners – by Provider [161]. The provider-service-level part B data contains one record, for each year, for each specific treatment or procedure that the provider submits claims to Medicare for. The provider-level part B data has records that contain information about the provider's activity over all claims for all treatments and procedures for a year. For research on the provider level data for Medicare data classification please see [95].

The provider-service-level Part B data has 29 features, but as is the case with the Part D data, many of the attributes are provider demographic data, which we do not use for modeling purposes. It also has categorical data that describes the treatment or procedure rendered to the patient, which we do not include in the final datasets, since we aggregate at the provider level. We utilize categorical features at the provider level for place of service, provider gender, and provider type. Provider type has proven to be a strong predictor for fraud detection [72, 74]. The provider-service-level Part B data also includes numeric data on claims submitted for treatments and procedures. There are features for the total number of times the service is provided, the total number of patients the service is provided to, the daily number of beneficiaries treated, the average amount the provider charges for the service,

and data on the average amounts Medicare pays for the service. The provider-service-level Part B data has approximately 68 million records.

The provider-level Part B data has 47 features. They are listed in Tables 2.5 and 2.6. There is a group of attributes in the provider-level Part B data for the total number of patients treated, by age. Therefore, there are four features, one for patients under 65, one for patients aged 65-75, and one for patients aged 75-84, and one for patients aged 85 and over. There is a final age-related feature that has the average number of patients in each age group. There are seven features which pertain to claims the provider sends to Medicare for all treatments and procedures that the provider renders to their patients for the year. These features contain data on the total number of distinct procedures, total number of patients treated, and the sum of money the provider has billed Medicare for, for the year. The seven features also include the amount Medicare is allowed to pay a provider, and the amount Medicare actually paid the provider. Finally, there is a feature for a standardized total payment amount. The standardized total Medicare payment amount is calculated by adjusting for geographic differences in costs, making it easier to compare prices across regions. The provider-level Part B data has another 14 features that are the result of splitting these seven features into two groups: one group for medication-related services, and the second for all other services. There are features for the total number of male, and female patients. The provider-level Part B data has another 18 features for the percentages of patients with certain chronic conditions. Alzheimer's, kidney disease, and asthma are all examples of chronic conditions. There is also an average beneficiary risk score. The score is calculated as described in the section on aggregated Part B data above. With a model that adjusts risk based on hierarchical condition categories (HCC). As per the CMS's definition, beneficiaries possessing risk scores higher than the average of HCC score of 1.08 are projected to have Medicare spending that exceeds the average.

Table 2.2: Credit Card Fraud Data Descriptive Statistics

	mean	std	min	25%	50%	75%	max
V1	0.0000	1.9587	-56.4075	-0.9204	0.0181	1.3156	2.4549
V2	0.0000	1.6513	-72.7157	-0.5985	0.0655	0.8037	22.0577
V3	-0.0000	1.5163	-48.3256	-0.8904	0.1798	1.0272	9.3826
V4	0.0000	1.4159	-5.6832	-0.8486	-0.0198	0.7433	16.8753
V5	0.0000	1.3802	-113.7433	-0.6916	-0.0543	0.6119	34.8017
V6	0.0000	1.3323	-26.1605	-0.7683	-0.2742	0.3986	73.3016
V7	-0.0000	1.2371	-43.5572	-0.5541	0.0401	0.5704	120.5895
V8	0.0000	1.1944	-73.2167	-0.2086	0.0224	0.3273	20.0072
V9	-0.0000	1.0986	-13.4341	-0.6431	-0.0514	0.5971	15.5950
V10	0.0000	1.0888	-24.5883	-0.5354	-0.0929	0.4539	23.7451
V11	0.0000	1.0207	-4.7975	-0.7625	-0.0328	0.7396	12.0189
V12	-0.0000	0.9992	-18.6837	-0.4056	0.1400	0.6182	7.8484
V13	0.0000	0.9953	-5.7919	-0.6485	-0.0136	0.6625	7.1269
V14	0.0000	0.9586	-19.2143	-0.4256	0.0506	0.4931	10.5268
V15	0.0000	0.9153	-4.4989	-0.5829	0.0481	0.6488	8.8777
V16	0.0000	0.8763	-14.1299	-0.4680	0.0664	0.5233	17.3151
V17	-0.0000	0.8493	-25.1628	-0.4837	-0.0657	0.3997	9.2535
V18	0.0000	0.8382	-9.4987	-0.4988	-0.0036	0.5008	5.0411
V19	0.0000	0.8140	-7.2135	-0.4563	0.0037	0.4589	5.5920
V20	0.0000	0.7709	-54.4977	-0.2117	-0.0625	0.1330	39.4209
V21	0.0000	0.7345	-34.8304	-0.2284	-0.0295	0.1864	27.2028
V22	-0.0000	0.7257	-10.9331	-0.5424	0.0068	0.5286	10.5031
V23	0.0000	0.6245	-44.8077	-0.1618	-0.0112	0.1476	22.5284
V24	0.0000	0.6056	-2.8366	-0.3546	0.0410	0.4395	4.5845
V25	0.0000	0.5213	-10.2954	-0.3171	0.0166	0.3507	7.5196
V26	0.0000	0.4822	-2.6046	-0.3270	-0.0521	0.2410	3.5173
V27	-0.0000	0.4036	-22.5657	-0.0708	0.0013	0.0910	31.6122
V28	-0.0000	0.3301	-15.4301	-0.0530	0.0112	0.0783	33.8478
Amount	88.3496	250.1201	0.0000	5.6000	22.0000	77.1650	25691.1600

Table 2.3: Features of the Part B Dataset

Feature Name	Description
Rndrng_Prvdr_Crdntls	The referring provider's credentials [example: MD]; categorical, 23,672 distinct values
Rndrng_Prvdr_Gndr	The referring provider's gender; categorical, three distinct values
Rndrng_Prvdr_Ent_Cd	Type of entity [individual or organization] reported in NPPES; categorical, 2 distinct values
Rndrng_Prvdr_Type	Derived from the Medicare provider/supplier specialty code reported on all the NPI's Part B non-institutional claims (DMEPOS and non-DMEPOS); categorical, 204 distinct values
Rndrng_Prvdr_Mdcr_Prtcptg_Ind	Identifies whether the provider participates in Medicare and/or accepts assignment of Medicare allowed amounts; categorical two distinct values
HCPCS_Cd	HCPCS code used to identify the specific medical service furnished by the provider; categorical 7,738 distinct values
HCPCS_Desc	Description of the HCPCS code for the specific medical service furnished by the provider; categorical, 8,252 distinct values
HCPCS_Drug_Ind	Identifies whether the HCPCS code for the specific service furnished by the provider is a HCPCS listed on the Medicare Part B Drug Average Sales Price (ASP) File; categorical, two distinct values
Place_Of_Srvc	Identifies whether the place of service submitted on the claims is a facility (value of 'F') or non-facility (value of 'O'); categorical two distinct values
Tot_Benes	Number of distinct Medicare beneficiaries receiving the service for each Rndrng_NPI, HCPCS_Cd, and Place_Of_Srvc
Tot_Srvcs	Number of services provided; note that the metrics used to count the number provided can vary from service to service
Tot_Bene_Day_Srvcs	Number of distinct Medicare beneficiary/per day services
Avg_Sbmtd_Chrg	Average of the charges that the provider submitted for the service
Avg_Mdcr_Alowd_Amt	Average of the Medicare allowed amount for the service
Avg_Mdcr_Pymt_Amt	Average amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item service
Avg_Mdcr_Stdzd_Amt	Average amount that Medicare paid after beneficiary deductible and coinsurance amounts have been deducted for the line item service and after standardization of the Medicare payment has been applied

2.3.3 Non-aggregated Part D Data

The Part D data comprises the largest of the three datasets that we work with. The Part D data pertains to medications that providers prescribe for their patients. A record of the Part D data relates to one particular medication that a provider prescribed for patients for one year. As with the DMEPOS and Part B data, we discard the NPI and location data that could interfere with a Machine Learning Model's ability to generalize. Table 2.4 summarizes all the Part D data elements we use in this study. Similar to Table 2.3, we use data definitions from the Part D data dictionary [159], and augment them with information on distinct values of categorical features. Our finalized version of the Part D data contains 174,357,611 records. The fraction of instances in the minority class in the Part D data is 0.0039.

The Part D data contains records of the medications healthcare providers prescribe for their patients. One record of the Part D data relates to one medication one provider prescribes for his, her, or their patients for one year. There are both categorical and numeric features in the Part D data. Please see Table 2.4 for details on each of the features. The definitions of the features in Table 2.4 are provided by the Medicare Part D Data Dictionary [159]. For the categorical features in Table 2.4, we add counts of the distinct values next to the feature description.

2.3.4 Aggregated Part D Data

We use two sources for the Part D data. The first is Medicare Part D Prescribers – by Provider and Drug [158], and the second is Medicare Part D Prescribers – by Provider [160]. The key difference between the two sources is the level of specificity of the data. The first source, Medicare Part D Prescribers – by Provider and Drug has a record for every combination of health care provider, medication that the health care provider prescribes, and year. This is the same dataset used to compile the non-aggregated Part D data. To avoid confusion, we refer to this as the "provider-drug-level Part D data". The second source, Medicare Part D

Prescribers – by Provider is less specific. It contains a record for each provider for each year. We refer to this as the "provider-level Part D data".

The provider-drug-level Part D data is the same data we use to compile the non-aggregated Part D data. Not all of these attributes are relevant for machine learning. These are attributes related to the provider's name and address. We eschew these attributes since they could form a unique identifier that a machine learning model could memorize instead of properly generalizing the data. The features are summarized in Table 2.4. We retain one identifier, the provider's national provider identifier (NPI), which we use later for labeling purposes. The provider-drug-level Part D data has two categorical features, that identify the type of medication prescribed. During the aggregation phase of our dataset compilation, we discard these categorical features. Another feature which is ultimately discarded, but useful for processing is the year in which the claim was made. We use this feature for aggregating the Part D provider-drug-level data by year, but we do not use it as an attribute for supervised machine learning. Since it is at the provider level, when we aggregate records, we retain a categorical feature for the provider type. Numeric features in the provider-drug-level part D data are readily useful for supervised machine learning. These include data on the total volume and frequency of prescriptions a provider submits claims for, the number of patients, as well as the total cost of the claims. Furthermore, there are similar, additional features for patients aged 65 and over. There are approximately 174 million records in the collection of provider-drug-level Part D data files.

The provider-level Part D data contains 51 additional attributes pertaining to claim the provider submits to Medicare, across all the medications the provider prescribes for the year. They are listed in Tables 2.7 and 2.8. The feature descriptions we provide here are from the provider-level Part D data dictionary [154]. The provider-level Part D data has ten features of summary statistics about the beneficiaries of the claims the provider submits. There is also an average beneficiary risk score. The score is calculated as described in the section on aggregated Part B data above. The provider-level Part D data also has features

for the total number of claims, total number of 30-day prescription orders, total drug cost, total day's supply dispensed, and the total number of beneficiaries seen, in the form of subtotals within various categories of claims. The categories are Low-Income Subsidy (LIS) claim, Medicare Advantage Prescription Drug Plan (MAPD) coverage claims, and Medicare Prescription Drug Plan (PDP) claims. The statistics are also divided by several drug categories, including claims for opiate drugs, long-acting (LA) opiate drugs, antibiotic drugs, and anti-psychotic drugs.

Table 2.4: Features of the Part D Dataset

Feature Name	Description
Prscrbr_Type	Derived from the Medicare provider/supplier specialty code; categorical, 249 distinct values
Prscrbr_Type_Src	Source of the Medicare provider/supplier specialty code; categorical, 2 distinct values
Brnd_Name	Brand name (trademarked name) of the drug filled; categorical, 3,907 distinct values
Gnrc_Name	A term referring to the chemical ingredient of a drug rather than the trademarked brand name under which the drug is sold; categorical, 2,272 distinct values
Tot_Clms	The number of Medicare Part D claims
Tot_30day_Fills	The aggregate number of Medicare Part D standardized 30-day fills
Tot_Day_Suply	The aggregate number of day's supply for which this drug was dispensed
Tot_Drug_Cst	The aggregate drug cost paid for all associated claims
Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one claim for the drug

Table 2.5: Provider Level Part B Features, Descriptions as Presented in [164]

Feature	Description
Tot_HCPCS_Cds	Total number of unique HCPCS codes
Tot_Benes	Total Medicare beneficiaries receiving services from the provider
Tot_Srvcs	Total provider services
Tot_Sbmtd_Chrg	The total charges that the provider submitted for all services
Tot_Mdcr_Alowd_Amt	The Medicare allowed amount for all provider services
Tot_Mdcr_Pymt_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for all the provider's line item service
Tot_Mdcr_Stdzd_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item service and after standardization of the Medicare payment has been applied
Drug_Tot_HCPCS_Cds	Total number of HCPCS codes for drug services
Drug_Tot_Benes	Total Medicare beneficiaries receiving drug services
Drug_Tot_Srvcs	Total drug services
Drug_Sbmtd_Chrg	The total charges that the provider submitted for drug services
Drug_Mdcr_Alowd_Amt	The Medicare allowed amount for drug services
Drug_Mdcr_Pymt_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for all the provider's line item drug services
Drug_Mdcr_Stdzd_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item drug service
Med_Tot_HCPCS_Cds	Total number of HCPCS codes associated with medical services
Med_Tot_Benes	Total Medicare beneficiaries receiving medical services
Med_Tot_Srvcs	Total medical services
Med_Sbmtd_Chrg	The total charges that the provider submitted for medical services
Med_Mdcr_Alowd_Amt	The Medicare allowed amount for medical services
Med_Mdcr_Pymt_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for all the provider's line item medical services
Med_Mdcr_Stdzd_Amt	Total amount that Medicare paid after deductible and coinsurance amounts have been deducted for the line item medical service
Bene_Avg_Age	Average age of beneficiaries. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Age_LT_65_Cnt	Number of beneficiaries under the age of 65. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Age_65_74_Cnt	Number of beneficiaries between the ages of 65 and 74. Beneficiary age is calculated at the end of the calendar year or at the time of death
Bene_Age_75_84_Cnt	Number of beneficiaries between the ages of 75 and 84
Bene_Age_GT_84_Cnt	Number of beneficiaries over the age of 84. Beneficiary age is calculated at the end of the calendar year or at the time of death

Table 2.6: (*continued* Provider Level Part B Features, Descriptions as Presented in [164]

Feature	Description
Bene_Feml_Cnt	Number of female beneficiaries
Bene_Male_Cnt	Number of male beneficiaries
Bene_Dual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare and Medicaid benefits
Bene_Ndual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare only benefits
Bene_CC_AF_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for atrial fibrillation
Bene_CC_Alzhmr_Pct	Percent of beneficiaries meeting the Chronic Conditions Data Warehouse (CCW) chronic condition algorithm for Alzheimer's, related disorders, or dementia
Bene_CC_Asthma_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for Asthma
Bene_CC_Cncr_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithms for cancer
Bene_CC_CHF_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for heart failure
Bene_CC_CKD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for chronic kidney disease
Bene_CC_COPD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for chronic obstructive pulmonary disease
Bene_CC_Dprssn_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for depression
Bene_CC_Dbts_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for diabetes
Bene_CC_Hyplpdma_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for hyperlipidemia
Bene_CC_Hyprtnsn_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for hypertension
Bene_CC_IHD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for ischemic heart disease
Bene_CC_Opo_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for osteoporosis
Bene_CC_RAOA_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for rheumatoid arthritis/osteoarthritis
Bene_CC_Sz_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for schizophrenia and other psychotic disorders
Bene_cc_strok_pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for stroke
Bene_Avg_Risk_Scre	Average Hierarchical Condition Category (HCC) risk score of beneficiaries

Table 2.7: Provider-level Part D Features, Descriptions as Presented in [154]

Feature	Description
GE65_Tot_Clms	The number of Medicare Part D claims for beneficiaries age 65 and older
GE65_Tot_30day_Fills	The number of Medicare Part D standardized 30-day fills for beneficiaries age 65 and older
GE65_Tot_Drug_Cst	The aggregate total drug cost paid for all associated claims for beneficiaries age 65 and older
GE65_Tot_Day_Suply	The aggregate number of day's supply for which this drug was dispensed, for beneficiaries age 65 and older
GE65_Tot_Benes	The total number of unique Medicare Part D beneficiaries age 65 and older with at least one claim for the drug
Brnd_Tot_Clms	Total claims of brand-name drugs, including refills
Brnd_Tot_Drug_Cst	Aggregate drug cost paid for brand-name drugs
Gnrc_Tot_Clms	Total claims of generic drugs, including refills
Gnrc_Tot_Drug_Cst	Aggregate cost paid for generic drugs
Othr_Tot_Clms	Total claims of other drugs, including refills. A drug is classified as "other" using any FDA approval categories not included in the brand or generic definitions
Othr_Tot_Drug_Cst	Aggregate cost paid for all other drugs not classified as brand or generic
MAPD_Tot_Clms	The number of claims for beneficiaries covered by (Medicare Advantage plan that includes Medicare Part (MAPD)
MAPD_Tot_Drug_Cst	Aggregate cost paid for claims filled by beneficiaries in MAPD plans
PDP_Tot_Clms	The number of claims for beneficiaries covered by standalone Prescription Drug Plans (PDPs)
PDP_Tot_Drug_Cst	Aggregate drug cost paid for claims filled by beneficiaries in standalone PDPs
LIS_Tot_Clms	Total number of claims from this prescriber, including refills, for beneficiaries with a Part D low-income subsidy (LIS)
LIS_Drug_Cst	Aggregate drug cost paid for claims for beneficiaries with a Part D low-income subsidy
NonLIS_Tot_Clms	Total number of claims from this prescriber, including refills, for beneficiaries without a Part D low-income subsidy
NonLIS_Drug_Cst	Aggregate drug cost paid for claims for beneficiaries without a Part D low-income subsidy
Opioid_Tot_Clms	Total claims of opioid drugs, including refills
Opioid_Tot_Drug_Cst	Aggregate cost paid for opioid drugs
Opioid_Tot_Suply	The aggregate number of day's supply for opioid drugs
Opioid_Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one opioid claim
Opioid_Prscrbr_Rate	The percent of the Tot_Clms represented by the Opioid_Tot_Clms
Opioid_LA_Tot_Clms	The aggregate number of day's supply for long-acting (LA) opioid drugs
Opioid_LA_Tot_Drug_Cst	Aggregate cost paid for long-acting opioid drugs

Table 2.8: (*continued)* Provider-level Part D Features, Descriptions as Presented in [154]

Feature	Description
Opioid_LA_Tot_Suply	The aggregate number of day's supply for long-acting opioid drugs
Opioid_LA_Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one long-acting opioid claim
Opioid_LA_Prscrbr_Rate	The percent of the Opioid_Tot_Clms represented by the Opioid_LA_Tot_Clms
Antbtc_Tot_Clms	Total claims of antibiotic drugs, including refills
Antbtc_Tot_Drug_Cst	Aggregate cost paid for antibiotic drugs
Antbtc_Tot_Benes	The total number of unique Medicare Part D beneficiaries with at least one antibiotic claim
Antpsyct_GE65_Tot_Clms	Total claims of antipsychotic drugs, including refills, for beneficiaries age 65 and older
Antpsyct_GE65_Tot_Drug_Cst	Aggregate cost paid for antipsychotic drugs for beneficiaries age 65 and older
Antpsyct_GE65_Bene_Suprsn_Flag	A flag indicating the reason the Antpsyct_GE65_Tot_Benes variable is suppressed
Antpsyct_GE65_Tot_Benes	The total number of unique Medicare Part D beneficiaries age 65 and older with at least one antipsychotic claim
Bene_Avg_Age	Average age of beneficiaries
Bene_Age_LT_65_Cnt	Number of beneficiaries under the age of 65
Bene_Age_65_74_Cnt	Number of beneficiaries between the ages of 65 and 74
Bene_Age_75_84_Cnt	Number of beneficiaries between the ages of 75 and 84
Bene_Age_GT_84_Cnt	Number of beneficiaries over the age of 84
Bene_Feml_Cnt	Number of female beneficiaries
Bene_Male_Cnt	Number of male beneficiaries
Bene_Dual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare and Medicaid benefits
Bene_Ndual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare only benefits
Bene_Avg_Risk_Scre	Average Hierarchical Condition Category (HCC) risk score of beneficiaries

2.3.5 Non-aggregated DMEPOS Data

The third type of Medicare claims data we use in our study is the DMEPOS data. Records in the non-aggregated DMEPOS data are similar to the records in the Part D data in that they summarize a provider's activity for a year. However, instead of data about prescription medication, the DMEPOS data has data on durable medical equipment, prosthetics, orthotics, and nutrition related products. Each record of the DMEPOS data contains a Healthcare Common Procedure Coding System (HCPCS) code for the item the provider renders to the patient. This HCPCS code plays a role similar to the brand and generic

names of medications in the Part D data. The non-aggregated DMEPOS data also contains a rental indicator. This rental indicator serves as a binary variable, delineating whether the claim submitted for DMEPOS product or service pertains to a rental. Numeric attributes of the DMEPOS data encapsulate the billing activities of providers in relation to each HCPCS code and the rental indicator. These attributes are: the count of suppliers providing products to the referring provider, the aggregate number of claims made, and the total number services rendered. Table 2.9 contains the details on the features of the non-aggregated DMEPOS data. All the feature descriptions in Table 2.9 are given in the DMEPOS data dictionary [157]. Similar to Table 2.4, we add counts of the distinct values of categorical features to the feature descriptions. The ratio of the minority class to the majority class in the DMEPOS data is approximately 0.0044. Once we complete the process for labeling the DMEPOS data by joining it to the LEIE data, we have a dataset with 12,215,370 instances.

As stated previously, any feature of the Medicare data that we document as categorical is encoded with CatBoost encoding during experiments, except in cases where LightGBM is used since LightGBM has a built-in function for encoding categorical data [48, 122]. In the DMEPOS data, as well as data for other parts of the program, there are several attributes that are not suitable for Machine Learning. These are attributes that pertain to the identity of providers, but provide no description of their activities or practices. These are the National Provider ID and other attributes related to the providers' address. As mentioned previously, we do not use these features because they are unique identifiers that may cause models to memorize rather than generalize.

Table 2.9: Features of the DMEPOS Dataset

Feature Name	Description
Rfrg_Prvdr_Crdntls	the referring provider's credentials [example: MD]; categorical, 7,315 distinct values
Rfrg_Prvdr_Gndr	the referring provider's gender; categorical, three distinct values
Rfrg_Prvdr_Ent_Cd	Type of entity [individual or organization] reported in NPPES; categorical, 2 distinct values
Rfrg_Prvdr_Type	Derived from the Medicare provider/supplier specialty code reported on all the NPI's Part B non-institutional claims (DMEPOS and non-DMEPOS); categorical, 204 distinct values
Rfrg_Prvdr_Type_Flag	A flag variable that indicates the source of the Referring Provider Type; categorical two distinct values
BETOS_Lvl	High level grouping of the Berenson-Eggers Type of Service (BETOS) Classifications into three groups including Durable Medical Equipment, Prosthetic and Orthotic Devices, and Drugs and Nutritional Products; categorical three distinct values
BETOS_Cd	The BETOS classification code assigned to the HCPCS Healthcare Common Procedure Coding System code; categorical, 12 distinct values
BETOS_Desc	Description of the HCPCS code for the specific product or service furnished by the DMEPOS supplier; categorical, 12 distinct values
HCPCS_Cd	HCPCS code for the specific product or service furnished by the DMEPOS supplier; categorical, 1337 distinct values
HCPCS_Desc	Description of the HCPCS code for the specific product or service furnished by the DMEPOS supplier; categorical, 1,618 distinct values
Suplr_Rentl_Ind	Identifies whether the DMEPOS product/service submitted on the supplier's claim is rental or non-rental; categorical 2 distinct values
Tot_Suplrs	Number of suppliers rendering DMEPOS products/services ordered by the referring provider
Tot_Suplr_Benes	Number of beneficiaries associated with the supplier DMEPOS products/services ordered by the referring provider
Tot_Suplr_Clms	Number of DMEPOS claims submitted by the supplier, reflecting products/services ordered by the referring provider
Tot_Suplr_Srvcs	Number of DMEPOS products/services rendered by the supplier
Avg_Suplr_Sbmtd_Chrg	Average of the charges that suppliers submit for DMEPOS products/services
Avg_Suplr_Mdcr_Alowd_Amt	Average Medicare allowed amounts for the DMEPOS product/service rendered by suppliers
Avg_Suplr_Mdcr_Pymt_Amt	Average amount that Medicare paid suppliers after deductible and coinsurance amounts have been deducted for the line item DMEPOS product/service

Table 2.10: Provider Level DMEPOS Features, Descriptions as Presented in [155]

Feature	Description
DME_Tot_Suplrs	Number of suppliers rendering durable medical equipment products/services.
DME_Tot_Suplr_Benes	Total number of unique beneficiaries associated with durable medical equipment claims submitted by suppliers and ordered by the referring provider.
DME_Tot_Suplr_Clms	Total number of durable medical equipment claims submitted by suppliers, reflecting services ordered by the referring provider
DME_Tot_Suplr_Srvcs	Total durable medical equipment products/services rendered by suppliers and ordered by the referring provider.
DME_Suplr_Sbmtd_Chrgs	The total charges that suppliers submitted for all durable medical equipment products/services ordered by the referring provider.
DME_Suplr_Mdcr_Alowd_Amt	The Medicare allowed amount for all durable medical equipment products/services ordered by the referring provider
DME_Suplr_Mdcr_Pymt_Amt	Amount that Medicare paid after deductible and coinsurance amounts have been deducted for all supplier's durable medical equipment line item products/services ordered by the referring provider.
POS_Tot_Suplrs	Number of suppliers rendering prosthetic and orthotic products/services.
POS_Tot_Suplr_Benes	Total number of unique beneficiaries associated with prosthetic and orthotic claims submitted by suppliers and ordered by the referring provider.
POS_Tot_Suplr_Clms	otal number of prosthetic and orthotic claims submitted by suppliers, reflecting products/services ordered by the referring provider.
POS_Tot_Suplr_Srvcs	Total prosthetic and orthotic products/services rendered by suppliers and ordered by the referring provider.
POS_Suplr_Sbmtd_Chrgs	The total charges that suppliers submitted for all prosthetic and orthotic products/services ordered by the referring provider.
POS_Suplr_Mdcr_Alowd_Amt	The Medicare allowed amount for all prosthetic and orthotic products/services ordered by the referring provider.
POS_Suplr_Mdcr_Pymt_Amt	Amount that Medicare paid after deductible and coinsurance amounts have been deducted for all supplier's prosthetic and orthotic line item products/services ordered by the referring provider.
Drug_Tot_Suplrs	Number of suppliers rendering drug and nutritional products/services.
Drug_Tot_Suplr_Benes	Total number of unique beneficiaries associated with drug and nutritional product claims submitted by suppliers and ordered by the referring provider.
Drug_Tot_Suplr_Clms	Total number of drug and nutritional product claims submitted by suppliers, reflecting services ordered by the referring provider.
Drug_Tot_Suplr_Srvcs	Total drug and nutritional products/services rendered by suppliers and ordered by the referring provider.

Table 2.11: (*continued*) Provider Level DMEPOS Features, Descriptions as Presented in [155]

Feature	Description
Drug_Suplr_Sbmtd_Chrgs	The total charges that suppliers submitted for drug and nutritional products/services ordered by the referring provider.
Drug_Suplr_Mdcr_Alowd_Amt	The Medicare allowed amount for drug and nutritional products/services ordered by the referring provider.
Drug_Suplr_Mdcr_Pymt_Amt	Amount that Medicare paid suppliers after deductible and coinsurance amounts have been deducted for drug and nutritional line item products/services ordered by the referring provider.
Bene_Avg_Age	Average age of beneficiaries. Beneficiary age is calculated at the end of the calendar year or at the time of death.
Bene_Age_LT_65_Cnt	Number of beneficiaries under the age of 65. Beneficiary age is calculated at the end of the calendar year or at the time of death.
Bene_Age_65_74_Cnt	Number of beneficiaries between the ages of 65 and 74. Beneficiary age is calculated at the end of the calendar year or at the time of death.
Bene_Age_75_84_Cnt	Number of beneficiaries between the ages of 75 and 84. Beneficiary age is calculated at the end of the calendar year or at the time of death.
Bene_Age_GT_84_Cnt	Number of beneficiaries over the age of 84. Beneficiary age is calculated at the end of the calendar year or at the time of death.
Bene_Feml_Cnt	Number of female beneficiaries.
Bene_Male_Cnt	Number of male beneficiaries.
Bene_Dual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare only benefits.
Bene_Ndual_Cnt	Number of Medicare beneficiaries qualified to receive Medicare and Medicaid benefits.
Bene_CC_AF_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for atrial fibrillation.
Bene_CC_Alzhmr_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for Alzheimer's, related disorders, or dementia.
Bene_CC_Asthma_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for Asthma.
Bene_CC_Cncr_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithms for cancer. Includes breast cancer, colorectal cancer, lung cancer and prostate cancer.
Bene_CC_CHF_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for heart failure.
Bene_CC_CKD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for chronic kidney disease.
Bene_CC_COPD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for chronic obstructive pulmonary disease.

Table 2.12: (*continued*) Provider Level DMEPOS Features, Descriptions as Presented in [155]

Feature	Description
Bene_CC_Dprssn_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for depression.
Bene_CC_Dbts_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for diabetes.
Bene_CC_Hyplpdma_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for hyperlipidemia.
Bene_CC_Hyprtnsn_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for hypertension.
Bene_CC_IHD_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for ischemic heart disease.
Bene_CC_Opo_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for osteoporosis.
Bene_CC_RAOA_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for rheumatoid arthritis/osteoarthritis.
Bene_CC_Sz_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for schizophrenia and other psychotic disorders.
Bene_CC_Strok_Pct	Percent of beneficiaries meeting the CCW chronic condition algorithm for stroke.
Bene_Avg_Risk_Scre	Average Hierarchical Condition Category (HCC) risk score of beneficiaries.

Table 2.13: LEIE Exclusion Codes and Rules

Rule number	Description
1128(a)(1)	Conviction of program-related crimes
1128(a)(2)	Conviction relating to patient abuse or neglect
1128(a)(3)	Felony conviction relating to health care fraud
1128(b)(4)	License revocation or suspension
1128(b)(7)	Fraud, kickbacks and other prohibited activities
1128(c)(3)(g)(i)	Conviction of two mandatory exclusion offenses 10 years
1128(c)(3)(g)(ii)	Conviction of 3 mandatory exclusion offenses indefinite

2.3.6 Aggregated DMEPOS data

Similar to the Part B and Part D data, we have two sources for the aggregated DMEPOS data. We refer to the first source as DMEPOS provider-service-level DMEPOS data, and

the second source as the provider-level DMEPOS data. The provider-service-level data is the same data that is used to compile the non-aggregated DMEPOS data, summarized in Table 2.9.

The second source of DMEPOS data is the provider-level DMEPOS data. This data contains beneficiary demographic data. The demographic data includes attributes that pertain to patients' eligibility for Medicare and Medicaid. The data also includes an average HCC risk score for beneficiaries, as well as 16 Chronic Condition (CC) features indicating the proportion of treated beneficiaries diagnosed with a CC. Additionally, eight statistical measures related to claims data are provided, detailing the number of suppliers, HCPCS codes, beneficiaries, claims, services, and the financial transactions involved, including submissions to Medicare, payments made by Medicare, and the permissible Medicare payments. These statistics are subsequently split into specific categories such as durable medical equipment (DME), prosthetic and orthotic services (POS), and drug-related services. The provider-level features are summarized in Tables 2.10, 2.11, and 2.12.

2.3.7 LEIE

To label the Medicare data used in this dissertation, we utilize data from another source within the United States Federal Government, the Office of Inspector General (OIG). The data from the OIG is the "List of Excluded Individuals and Entities" (LEIE) [130]. The LEIE contains data on healthcare providers that have been convicted of Medicare insurance fraud. Therefore, we combine either the Part B, Part D, or the DMEPOS data with the LEIE to form a labeled dataset. This enables us to perform Medicare fraud detection as a supervised Machine Learning task.

Healthcare providers that are prohibited from participating in the United States government's healthcare programs are cataloged in the LEIE. The three key pieces of information we use in the LEIE are the healthcare provider's National Provider Identifier (NPI), the term of the exclusion period, and the exclusion rule code. We filter the LEIE for a subset of

exclusion rule codes. We retain only the records of the LEIE that have the exclusion rule codes Herland *et al*. use in [75]. We provide a listing of the exclusion codes used in Table 2.13. Not every record in the LEIE has an NPI. We discard these records since we have no way to relate them to the Medicare data. The NPI is necessary for joining the LEIE data to the Part B, Part D, and DMEPOS data, so we further filter the LEIE for records where the NPI is present.

2.4 LABELING AND JOINING

For the non-aggregated Part B, Part D, and DMEPOS data, labeling is straightforward. We adopt a uniform approach to labeling the three non-aggregated datasets with the LEIE. The LEIE is released on a monthly schedule by the OIG. When a healthcare provider is listed in the LEIE, it may denote a conviction for actions that prohibits them from filing insurance claims with Medicare. The LEIE, DMEPOS, Part B, and Part D data sources all share a common element, the NPI. There are different types of exclusions that apply to providers that are on the LEIE. When a provider is listed in the LEIE for any of these types of exclusions, we mark all associated records dated before the end of the exclusion period as fraudulent.

The data from DMEPOS, Part B and Part D covers entire calendar years, while exclusion periods end at specific months. For this reason, we approximate the end of an exclusion period to the closest year. When an exclusion period concludes, the healthcare provider in question is expunged from the LEIE, allowing them to once again file claims with Medicare. Consequently, any data related to claims made by the provider after the expiration of the exclusion period is considered non-fraudulent. Compiling a dataset encompassing all available years will not include records of providers previously listed in the LEIE, as the current LEIE will not include such records. As such, tools like the Internet Archive Tool[1] should be utilized to access earlier versions of the LEIE and label the older records in the

[1]http://archive.org/web

CMS data accordingly.

For the aggregated Part B, DMEPOS and Part D data, our approach is to aggregate data from the first source at the drug/service level, and then enrich the aggregated data with the data from the second source that is at the provider level. For the provider-service-level Part B data, we retain the features for the provider type, place of service, and provider gender, since they are provider-level features. We discard features related to specific services, such as the Healthcare Common Procedure Coding System (HCPCS) code. We then take the remaining numeric features and for each of them, we calculate six summary statistics: sum, mean, median, minimum, maximum, and standard deviation, using all records for the provider, for the year. The provider-drug-level Part D data does not have attributes for the provider gender, or place of service. However, it has a feature for provider type, so we retain that since it is at the provider level. We discard the codes for medications in the provider-drug-level Part D data as part of the aggregation process. For the DMEPOS data we retain the provider gender and provider type. Similar to Part B and Part D data, we discard the HCPCS code in records of the provider-service level DMEPOS data. Aggregation reduces the size of the DMEPOS, Part B and Part D datasets.

Next, we enrich both the Part D, DMEPOS, and the Part B data by joining them to the provider level datasets. This means we join the aggregated data to the provider-level data by NPI. The joining process yields an unlabeled dataset. Joining data by NPI means that we retain a record in our dataset if, and only if, a record exists in the aggregated data and the provider-level data with the same NPI and year values. The unlabeled Part B and DMEPOS datasets are exactly the same size as the aggregated Part B dataset. However, the unlabeled Part D dataset has about one million fewer records than the aggregated Part D dataset. This is due to a lack of provider level Part D records for some providers and some years.

As a last step in compiling the aggregated DMEPOS, Part B and Part D datasets, we label them with records from the LEIE. We use the same labeling process as described above for the non-aggregated data, joining records by NPI.

2.5 HANDLING CATEGORICAL FEATURES

As stated previously, we use CatBoost encoding [141] for all learners discussed in this dissertation except LightGBM since LightGBM has a built-in function for encoding categorical data. The CatBoost Encoder is another publicly available, open source library one can easily install with a Python package manager [136]. We opted for a general purpose encoding method, since the focus of our dissertation is on data reduction techniques. For a study on special purpose encoding techniques in the Medicare fraud detection application domain, please see [89]. The function of the CatBoost Encoder is to convert categorical features to floating point numbers, so they are suitable for use with the learners. The CatBoost encoder must be fit to data before it can be used to encode features. One must take care to fit the encoder to the training data only. Otherwise, the encoded features will contain information about the test data that may cause learners to overfit to the dataset, and yield unrealistically high performance metrics. This is commonly known as target leakage.

CatBoost encoding offers protections against target leakage is through its Ordered Target Statistics method of encoding categorical features. In simple target encoding, a categorical feature is assigned the mean value of the dependent variable that the feature is observed to co-occur with. This strategy for encoding may lead to information leakage in the sense that if the encoded feature co-occurs with different values of the dependent variable in the test data, the encoded feature will not be a useful predictor of the dependent variable. To avoid this issue, Ordered Target Statistics is a technique for ensuring that the encoded value for a categorical feature of a given instance is derived from other instances. Put another way, the encoded value of a categorical feature is not allowed to be calculated from the label it appears with. This makes it impossible for the encoded feature value of the instance to be directly related to the value of the dependent variable.

2.6 CHAPTER SUMMARY

This chapter has presented a comprehensive examination of the seven datasets utilized in the study. We emphasize their imbalanced nature, which is critical for developing data reduction techniques in Big Data fraud detection. The datasets are comprised of Medicare data, except for one dataset which is composed of Credit Card transaction data. All the datasets exhibit class imbalance. In all the datasets, the minority class is less than one percent of the total instances. The Credit Card Fraud dataset is characterized by its high imbalance, with only 492 fraudulent transactions out of 284,807 total transactions. The majority of its attributes are the result of principal component analysis (PCA), making them abstract and not directly relatable to human experience, except for the 'Amount' attribute which represents the transaction amount.

The Medicare datasets encompass data from three sources, each with unique attributes: Part D, Part B, and DMEPOS, with the Part D dataset being the largest at nearly 175 million instances. These datasets reflect the volume and velocity aspects of Big Data and are instrumental in exploring automated Medicare insurance fraud detection. The datasets were labeled using the List of Excluded Individuals and Entities (LEIE), which contains information on healthcare providers convicted of Medicare insurance fraud, enabling supervised machine learning tasks for fraud detection. For each of the Part B, Part D, and DMEPOS datasets, we have aggregated and non-aggregated versions of the datasets. Therefore, we also describe the aggregation process for the Part B, Part D, and DMEPOS Medicare datasets used in this dissertation. Due to the larger size, experiments with non-aggregated data are more time-consuming. Hence, further experiments were conducted with aggregated data. At a high level, the aggregation process is the same for the three Medicare datasets. Each Medicare program has a provider level data source, which summarizes the providers' activity for the year, and a second, more specific data source which contains a record for particular medications (Part D), equipment (DMEPOS), or services (Part B) rendered to the patients.

CHAPTER 3

LEARNERS

3.1 INTRODUCTION

Since the focus of our dissertation is data reduction techniques, we must review the classifiers used in the development of these techniques. This chapter provides details on the classifiers employed in the research for this dissertation. Most of the classifiers fall into three broad categories: Boosting Learners, Bagging Learners, and One-Class Classifiers (OCCs). We refer to any classifier that is not specifically defined as an OCC as a Binary-Class Classifier (BCC).

We begin by explaining Bagging techniques. The Bagging classifiers used for experiments in this dissertation are Random Forest, and Extremely Randomized Trees. We then cover the Boosting technique and the Boosting classifiers XGBoost, CatBoost, and LightGBM. We also include a section on the Logistic Regression classifier, since it is utilized in some experiments. Logistic Regression does not fall into any of the three previously mentioned categories. To complete our explanations of classifiers, we provide details on One-Class Gaussian Mixture Model (GMM), and One-Class Support Vector Machine (SVM). Furthermore, we present results of preliminary experiments that were influential in our experimental designs for evaluating BCCs versus OCCs, and data reduction techniques.

3.1.1 Bagging

Breiman coined the term Bagging in a study published in 1996 [21]. One may apply Bagging to both classification and regression tasks. Since our dissertation involves classification, we focus on how bagging may be applied for classification. Bagging relies on a

sampling technique known as bootstrap sampling. A bootstrap sample is a sample taken with replacement. In order to apply Bagging, one trains an ensemble of the same classifiers on different bootstrap samples of the training data. After the classifiers are trained, one may use them for classification. Bagging specifies that we treat the classifiers' output as votes for the classification outcome. The class assigned to an instance is the class that the majority of the members of the ensemble assign the instance to. Some informal reasoning about the probability of a correct result explains the appeal of Bagging. Suppose the probability of a correct classification of one of the learners is greater than one half. Then, as the size of the ensemble of learners increases, the probability that more than half of the learners will agree on the correct class of an instance will increase in proportion with the size of the ensemble. In this dissertation we report on the results of research conducted with two learners that utilize the bagging technique: Random Forest [22], and Extremely Randomized Trees [35]. In both of these techniques, the weak learners are Decision Trees.

3.1.2 Random Forest

The Random Forest implementations we use in this dissertation use decision trees for the weak learners. In his seminal work on Random Forest, [22], Breiman describes an additional aspect that is not a part of the Bagging framework. This aspect is the method Random Forest uses to select splits in its decision trees. A split in a decision tree is the numeric value in a node that is used to decide which edge to follow. Splits pertain to specific features in a dataset, and if the instance's value of a feature is less than the value of the split, one edge is followed out of the node, otherwise, the other edge is followed from the node. Therefore, splits are important components of decision trees, since they determine the path that is taken through the decision tree to reach a classification. Random Forest selects the feature for the split randomly from the set of features of the dataset. We utilize Random Forest in research conducted in this dissertation since previous research shows its strong performance in classifying imbalanced Big Data [9, 107].

3.1.3 Extremely Randomized Trees

ET is the second Bagging-based classifier we use here. Guerts *et al*. introduced ET in the seminal paper [35]. ET functions similarly to Random Forest. However, in addition to the random choice of features to use for the splits in decision trees, ET also chooses the values of splits randomly. The random selection of splits sets ET apart from the other classifiers in our dissertation, since the others use optimization techniques for determining the values of splits. For example, ET's closest relative in this dissertation, Random Forest, selects values for the splits in decision trees to optimize for Gini impurity [145]. The appeal of selecting the value of a split randomly is that it should be faster than an optimization technique which requires many iterations over small increments of the value being optimized.

3.2 BOOSTING

The Gradient Boosted Machine algorithm was introduced by Friedman [34]. Our research shows this technique works well in the Medicare fraud detection application domain that is central to this dissertation [46]. are The Gradient Boosting Machine technique is an ensemble technique, but the way in which the constituent learners are combined is different from how it is accomplished with the Bagging technique. The Gradient Boosting Machine technique begins with a single learner that makes an initial set of estimates $\hat{\mathbf{y}}$ of the dependent variable $\mathbf{y}$. The differences (residuals) in the estimates $\hat{\mathbf{y}}$ and $\mathbf{y}$ forms a vector $\mathbf{y} - \hat{\mathbf{y}}$ that we can think of as a new dependent variable that we can estimate with the original independent variables and a second learner. Then, the sum of the output values of the two models will be a more accurate estimate of the dependent variable than the output values of the first model. We can continue to add learners to the ensemble similarly, where each new learner is trained to predict the residuals of the current ensemble. Therefore, each learner we add to the ensemble provides a better estimate of the dependent variable. The Gradient Boosting Machine implementations we use are all enhancements to Friedman's initial proposal. They all involve a specific type of learner, Decision Tree, so we refer to them as Gradient Boosted

Decision Trees (GBDTs).

Of the three GBDT implementations we use, XGBoost was the first to be released. Chen and Guestrin released XGBoost in 2016. XGBoost offers several enhancements to the GBDT technique. The first enhancement is an improved loss function used during the training phase. The loss function contains an additional term for regularization to prevent overfitting. Another enhancement XGBoost makes to GBDTs is one that has to do with calculating splits in the constituent decision trees of the GBDT ensemble. Chen and Guestrin introduce the so called "approximate algorithm" which is a technique for estimating optimal values of splits. The approximate algorithm is suitable for distributed environments, as well as applications where the entire dataset does not fit in main memory. A third enhancement in XGBoost is another algorithm for finding splits that works well with sparse data. Sparse data is the type of data that is nearly constant in value with infrequently occurring aberrations. XGBoost can take advantage of sparse data with its "sparsity aware split finding" feature.

Ke *et al*. released the seminal paper on LightGBM in 2017 [101]. Their goal was to offer a GBDT implementation that yields performance equivalent to XGBoost, while consuming fewer resources. In order to achieve their goal, Ke *et al*. make two key enhancements to the GBDT technique. The first is Exclusive Feature Bundling (EFB). EFB is a technique for reducing the dimensions of a dataset by combining two features (attributes) of a dataset into a single feature. EFB is an effective technique for sparse data. When two attributes of a dataset exhibit sparsity, and the infrequently occurring values of both attributes are mutually exclusive, they may be safely combined into a single feature without the loss of information. EFB reduces the number of dimensions of a dataset, which helps reduce training time. LightGBM's second enhancement to the GBDT technique is called Gradient-based One-Side Sampling (GOSS). GOSS is a technique for intelligently reducing the number training instances used. GOSS selects instances for training based on their contribution to the loss function that is calculated as part of fitting the GBDT ensemble to the training data. If the

instance contributes more than a configurable threshold value to the loss of the model, then it is retained for further iterations of the fitting process. Likewise, instances that contribute less than the threshold amount are set aside. Via GOSS and EFB Ke *et al*. deliver a GBDT implementation that consumes fewer computing resources, but is capable of performing on a par with other GBDTs.

The third GBDT implementation we use is CatBoost [141]. Prokhorenkova *et al*. introduced CatBoost in 2018. One may find more information on applications of CatBoost in various domains in [42]. Their motivation for developing CatBoost was to prevent overfitting. The first protection against overfitting that CatBoost makes is Ordered Boosting. Ordered Boosting is a technique for selecting training instances. There are two steps for adding a decision tree to the GBDT ensemble. The first is to fit the decision tree to the dependent variable in the training data. Multiple decision trees are fit to different samples of the training data. After the trees have been fit, they must be evaluated in order to select the tree that best enhances the overall performance of the ensemble. Under Ordered Boosting one can be sure that training instances used to fit the decision tree will not be used to evaluate it for inclusion into the ensemble. This helps prevent the ensemble from being overfit to the training data. The second kind of overfitting CatBoost offers protections against is through its Ordered Target Statistics method of encoding categorical features. Results of experiments we have conducted show CatBoost's overfitting protections make it an effective classifier for imbalanced data [43]. In simple target encoding, a categorical feature is assigned the mean value of the dependent variable that the feature is observed to co-occur with. This strategy for encoding may lead to information leakage in the sense that if the encoded feature co-occurs with different values of the dependent variable in the test data the encoded feature will not be a useful predictor of the dependent variable. To avoid this issue, Ordered Target Statistics is a technique for ensuring that the encoded value for a categorical feature of a given instance is derived from other instances. Put another way, the encoded value of a categorical feature is not allowed to be calculated from the label it

appears with. This makes it impossible for the encoded feature value of the instance to be directly related to the value of the dependent variable. This is a protection against what Prokhorenkova *et al.* call "target leakage."

3.3 LOGISTIC REGRESSION

Given its inherent simplicity relative to other classifiers, Logistic Regression, [113] serves as an apt starting point. At its core, Logistic Regression revolves around tailoring a sigmoid function to a dataset. This is accomplished by setting the sigmoid function's parameters to the maximum likelihood estimate of their value, given the training data. A Logistic Regression model is characterized by one parameter for each independent variable. This attribute renders it considerably simpler than many ensemble methods which have a larger number of parameters due to their being composed of instances of other models. Our rationale behind employing Logistic Regression in our dissertation is to ensure we check for the potential efficacy of a simpler model before recommending more sophisticated algorithms that consume far more computing resources.

3.4 ONE CLASS CLASSIFIERS

Throughout this dissertation, we address the challenge of data imbalance. It may even be the case that one only has data on instances of one class even though one expects to find instances of another class in future data . For example, one may only have transaction or claims data classified as legitimate, yet wish to develop a technique for detecting the positive (fraudulent) class in future data. In such a scenario, one-class classifiers are appropriate. Here we discuss the two one-class classifiers used in this dissertation, One-Class Support Vector Machine (SVM), and One-Class Gaussian Mixture Model (GMM).

3.4.1 One-Class SVM

The scikit-learn implementation of One-Class SVM we use functions as described by Schölkopf *et al*. [149]. One may view One-Class SVM as a specialization of SVM which can be trained on data where all instances have the same label. The key concept of SVM is the maximum margin-separating hyperplane. In a binary classification setting, this is the boundary that is maximally distant from the borders of two regions containing instances of either class. SVM fits a function to approximate the maximum margin-separating hyperplane of the training sample of a dataset. Classification of test samples is performed by assigning an instance to the same class as the other instances that lie on the same side of the maximum margin-separating hyperplane. Different kernel functions may be employed with SVM to project the features of a dataset into a space where its points are separable by a hyperplane. One-Class SVM does not have two classes of points in the feature space of a dataset. Therefore, it finds the maximum margin-separating hyperplane between its training data and the origin. The origin is the point in the feature space that has the value zero in all dimensions. The scikit-learn implementation of SVM OCC we utilize here allows one to select a radial basis function (RBF) [148] kernel. This is the default kernel, and used in experiments covered in this Dissertation. The use of this kernel transforms the procedure for calculating the maximum margin-separating hyperplane to one for finding a surrounding hypersphere that contains most of the instances of the training data. One-Class SVM with an RBF kernel can then be conceptualized as classifying instances of the test data according to whether the instance is contained in a hypersphere of the feature space after it is transformed by the radial basis function.

3.4.2 One-Class GMM

We employ the One-Class GMM implemented in the Python scikit-learn library [139]. GMM models utilize the expectation-maximization (EM) algorithm to fit data to a series of Gaussian distributions for the purpose of classification [33]. GMM models are an approach

where we conceptualize the training data as a composite of multiple multivariate Gaussian distributions, each potentially encapsulating distinct subsets of instances within the training data. The EM algorithm, when applied to GMMs, is an iterative process of estimating the expected value of the log-likelihood of the data given the current estimates of the GMM's parameters, and then maximizing the value of the log-likelihood by updating the GMM's parameters. The process of estimating and maximizing proceeds iteratively until the values of the GMM's parameters change by less than a predetermined threshold amount on successive iterations, or a predetermined number of iterations has been reached.

The training process for One-Class GMMs involves fitting several Gaussian components to the target data, which in this context, comprises instances from only a single class. Upon completion of training on instances belonging to a solitary class, the One-Class GMM undertakes the task of classification by evaluating the likelihood that a new test instance originates from the distributions modeled on the training data. Instances are classified as belonging to the class not encountered during training, if the likelihood computed for them falls beneath a predefined threshold. The one-class classification scenario is different from the unsupervised scenario, where labels are completely unknown. For readers interested in delving deeper into the broader topic of unsupervised techniques for anomaly detection, please see the work by Kennedy [105].

3.4.3 Calibration

We employ calibration as part of evaluating model performance metrics of OCCs. One may employ calibration to convert the output of a model to probablities, or to smooth a classifier's output to obtain more meaningful output probabilities for computing classification metrics such as precision and recall. During our our experiments with OCCs we learned of two readily available techniques for calibration, Sigmoid Calibration, and Isotonic Regression.

Sigmoid Calibration refers to the technique of smoothing the output of a classification model using a logistic function. Platt describes Sigmoid Calibration as a technique

for converting the output of Support Vector Machine to probabilities [140]. The scikit-learn implementation of One-Class SVM that we use does not return a class membership probability. Therefore, we found it necessary to use Platt's technique.

In our research concerning One-Class GMM, we discovered that Sigmoid Calibration significantly enhances the calculation of AUPRC for One-Class GMMs as well. The implementation of the One-Class GMM utilized in our research is from scikit-learn version 1.2.0. The package name is `sklearn.mixture.GaussianMixture`. During initial experimentation, it was observed that the `predict_proba` method of this particular version of One-Class GMM tended to produce class membership probabilities that were predominantly near 0.0 or 1.0. This concentration of probabilities near the extremes led to a lack of variation in precision and recall across most threshold values, which in turn resulted in a precision-recall curve with insufficient data points to yield meaningful insights.

This limitation prompted us to conclude that the `predict_proba` function of `sklearn.mixture.GaussianMixture` lacked the necessary robustness for our analysis. To address this issue, we implemented sigmoid calibration to transform the output probabilities from the GMM into more well-calibrated probabilities.

Further research on the general subject of calibration is available in works by Morrison [138] or Kull *et al*. [110]. In experiments covered in this dissertation, we employ a second calibration technique, Isotonic Regression, which is described by Kull *et al*.. Isotonic regression is a non-parametric approach. It entails the approximation of a piecewise-constant function to the classifier's output scores, thereby enabling a monotonic transformation of these scores into probabilities. The primary advantage of this method lies in its computational efficiency and its model-independent nature, allowing for its application across various classifier models. This concludes our discussion on calibration. Next we move on to discuss situations where one-class classifiers are an appropriate choice.

3.4.4 Choice of One-Class Versus Binary Class Classifiers

In conducting the research that constitutes this dissertation we observe that Binary-Class classifiers (BCCs) outperform One-Class classifiers (OCCs) [121]. Therefore, when instances of both classes are available, our results imply practitioners should employ BCCs. As stated in the introduction, one should only be interested in using OCCs as classifiers when there is only data from one class available.

One study relevant to this dissertation that demonstrates the improvement in performance that BCCs yield over OCCs is "Assessing One-Class and Binary Classification Approaches for Identifying Medicare Fraud", by Leevy *et al*. In their study Leevy *et al*. compare the performance of OCCs and BCCs in classifying the non-aggregated Medicare Part D data described in Chapter 2. The results in Tables 3.1 and 3.2 show the clear superiority of BCCs in classifying the Medicare Part D data. In these tables we report Area Under the Receiver Operating Characteristic Curve (AUC) [19], and Area Under Precision Recall Curve (AUPRC) [20] scores that the OCCs and BCCs discussed in this chapter yield when classifying the aggregated Medicare Part D Data. These results are also reported by Leevy *et al*. The AUC and AUPRC scores are the mean values of ten iterations of five-fold cross validation. If calibration is used to convert or smooth model output probabilities, we report the type of calibration used after the name of the classifier. If Sigmoid Calibration is used "Sigmoid" appears after the classifier name, and if Isotonic Regression is used, "Isotonic" appears after the classifier name.

Table 3.1: OCCs' performance in Classifying Medicare Part D Data

Classifier	AUC	AUPRC
One-Class GMM Sigmoid	0.7289	0.1481
One-Class SVM Isotonic	0.5127	0.0031

Table 3.2: BCCs' performance in Classifying Medicare Part D Data

Classifier	AUC	AUPRC
Catboost	0.9693	0.8124
Extremely Randomized Trees	0.9348	0.6480
Random Forest	0.9627	0.7859
XGBoost	0.9682	0.7803
Logistic Regression	0.8600	0.2901

3.4.5 Training One-Class Classifiers on Majority versus Minority

As part of the investigation of data reduction techniques in this dissertation, we explore the possibility of training OCCs on the minority class. These experiments were part of our initial foray into the use of one-class classifiers. While the approach yields a huge reduction in data, we found that training OCCs on the minority class does not yield performance on a par with training OCCs on the majority class. These experiments led to the later investigations on how much of the majority class we could discard for training and maintain the performance of training with the entire training data. Here we present results of experiments performed with OCCs trained on the majority and minority classes of the Credit Card Fraud and Medicare Part D data. We evaluate the performance of two OCCs. These are: One-Class SVM and One-Class GMM.

We evaluate the performance of the classifiers with AUC and AUPRC. Due to poor results in preliminary experiments, we use only Sigmoid Calibration to convert the output of GMM to probabilities. Unless stated otherwise, all results reported are mean values of ten iterations of five-fold cross validation.

First we present results for the OCCs trained on the minority class of the Credit Card

Fraud dataset discussed in Chapter 2 . These results are reported in Table 3.3. The next set of results presented in Table 3.4 are for OCCs trained on the majority class of the Credit Card Fraud data. The results clearly show better performance when OCCs are trained on the majority class.

Table 3.3: Credit Card Fraud Mean AUC and AUPRC scores by classifier; The results are for models trained on the minority class only.

Classifier	AUC	AUPRC
One-Class GMM Sigmoid	0.7382	0.2205
One-Class SVM Isotonic	0.5603	0.0334
One-Class SVM Sigmoid	0.5718	0.0184

Table 3.4: Credit Card Fraud Mean AUC and AUPRC scores by classifier; The results are for models trained on the majority class only.

Classifier	AUC	AUPRC
One-Class GMM Sigmoid	0.9496	0.4971
One-Class SVM Isotonic	0.9091	0.4165
One-Class SVM Sigmoid	0.9084	0.3775

We report results for similar experiments for classifying the aggregated Medicare Part D data. Classification results are in Tables 3.5 and 3.6. The results in Table 3.5 are from experiments where all classifiers are trained on the minority class. For comparison, we include results of experiments where learners are trained on the majority class in Table 3.6. Due to long-running times, the results for One-Class SVM Isotonic are the mean values of one iteration of five-fold cross validation. Furthermore, results for One-Class SVM Sigmoid are not available, also due to long-running times.

Table 3.5: Part D Mean AUC and AUPRC scores by classifier; the results are for models trained on the minority class only.

Classifier	AUC	AUPRC
One-Class GMM Sigmoid	0.4739	0.0007
One-Class SVM Isotonic	0.6082	0.0042
One-Class SVM Sigmoid	0.6072	0.0017

Table 3.6: Part D Mean AUC and AUPRC scores by classifier; the results are for models trained on the majority class only; scores for One-Class SVM Isotonic are the mean value of one iteration of five-fold cross validation.

Classifier	AUC	AUPRC
One-Class GMM Sigmoid	0.7289	0.1481
One-Class SVM Isotonic	0.5127	0.0031

The results in Tables 3.5 and 3.6 for the classification of the Medicare Part D data are consistent with the results for the classification of the Credit Card Fraud data. OCCs trained on the majority class yield better classification scores than OCCs trained on the minority class. Moreover we find One-Class GMM Sigmoid yields the best performance in terms of AUPRC in all cases. The results of these experiments influenced the direction of further experiments. These results led us to focus OCCs trained on the majority class in later experiments. Another key take-away from the results is that GMM yields consistently higher performance. In later experiments, we found GMM to be the best performing OCC. One more important finding is that, although performance is poor, we found that OCCs

appeared to be capable of a degree of learning with a small number of instances. This is apparent in the results in Table 3.3 for One-Class GMM. The ability for OCCs to learn on a small number of instances implies that OCCs have the potential for yielding strong results when trained on a sample smaller than the size of the entire training data. For a study on the impact of the size of the training data on Medicare claims classification, please see [143, 144].

3.5 CHAPTER SUMMARY

This chapter explores classifiers that are used in experiments covered in this dissertation. From the Bagging family we provide details on Random Forest, and Extremely Randomized Trees. From the Boosting Family we provide details on CatBoost, XGBoost, and LightGBM. Moreover, we delve into Logistic Regression and OCCs. As part of our discussion on OCCs we cover calibration of OCC output.

Through empirical analysis, the chapter compares classifier performance. We provide guidance for the selection of OCCs versus BCCs. We provide results from preliminary experiments to explain the scenarios where OCCs versus BCCs are appropriate. We provide additional results from preliminary experiments to show that while OCCs should be trained on the Majority class, they exhibit the potential to be trained on smaller samples and retain the ability to provide meaningful classification results. This is a key result that influenced further experimentation in data reduction techniques.

CHAPTER 4

EXPERIMENTAL METHODOLOGY

4.1 INTRODUCTION

This chapter explains the methodology employed to ensure the robustness of the research findings presented in this dissertation. The key elements of the methodology are: k-fold cross-validation, Area Under the Receiver Operating Characteristic curve (AUC), Area Under the Precision Recall Curve (AUPRC), Analysis of Variance (ANOVA), and Tukey's Honestly Significant Difference (HSD) Test. We provide sections on each element. Moreover, the order in which we cover the elements is important, and it mirrors the order in which we conduct our experiments. We employ k-fold cross-validation, resulting in k experimental outcomes, in terms of AUC or AUPRC. We provide in-depth explanations of how AUC and AUPRC are calculated since these metrics are the focus of most of the experiments covered in this dissertation. Once the experimental outcomes are recorded, we perform statistical analyses on them. The statistical analyses are in the form of ANOVA and HSD tests. Understanding these key elements provides an understanding of the method used to conduct experiments covered in this dissertation.

4.2 CROSS-VALIDATION

Throughout the course of our research, we employ a technique for separating our data for training models and testing models, known as cross-validation. This is the first step in experiments we conduct. Witten provides a detailed justification for cross-validation in [174]. We agree with Witten's justification and utilize the scikit-learn `sklearn.model_selection.StratifiedKFold` library to execute cross-validation in

practice. Nearly every experiment discussed in our dissertation involves ten iterations of five-fold cross-validation. Hence, cross-validation is an important aspect of the research we report on here.

Though we rely on 5-fold cross-validation for most of our experiments, here, we explain the more general approach of k-fold cross-validation, since it is trivial to apply it to the special case where $k = 5$. Stratified k-fold cross-validation is characterized by its systematic approach to partitioning the dataset into k distinct subsets to facilitate the evaluation of the model's performance. Each of the k subsets is known as a fold. This technique entails the iterative use of each data instance as test data once. The same data instance is incorporated into the training data k-1 times, thereby ensuring exhaustive participation of all instances in both model training and evaluation. Put another way, in each iteration of k-fold cross-validation, k-1 folds are used to train a model, and one of the k folds is used to evaluate the model's performance in terms of the desired metrics. Retraining models is a crucial task in Machine Learning applications since performance can degrade as new data for the same application domain becomes available [125, 126]. The k-fold cross-validation framework supports k train/test iterations. The iterations can be run in parallel if sufficient computing resources are available.

Stratified k-fold cross-validation is an important variation on cross-validation that is important for experimenting with imbalanced datasets [150, 167]. Stratification makes the additional requirement to maintain a uniform representation of each class within every fold, thereby mitigating the potential biases that may emanate from disproportionate class distributions. This stratification is critical, as it ensures the integrity and reliability of the cross-validation process, particularly in scenarios involving imbalanced datasets where certain classes are underrepresented. Stratification is built in to the `sklearn.model_selection.StratifiedKFold` library we use for the experiments discussed in this dissertation.

Another important feature of the cross-validation implementation used in our experi-

ments is shuffling. Before `sklearn.model_selection.StratifiedKFold` divides data into folds, it randomly reorders the data. This is important because data may be stored in an order that can influence the outcome of Machine Learning experiments. For example, for some reason, the Medicare Part D data we discuss in this dissertation could be ordered by state, in which case splitting the data into k-folds may result in each fold having data from distinct subsets of states. To combat this possibility,
`sklearn.model_selection.StratifiedKFold` shuffles the data. If the data is not in any particular order, shuffling has no effect. Therefore, it is prudent to employ a cross-validation technique that includes shuffling. Even if cross-validation is not employed, and data is simply split into train and test sets, shuffling can still be beneficial.

To further mitigate the possibility of random chance influencing the outcome of our experiments, we perform ten iterations of k-fold cross-validation. Therefore, most of the metrics we report in this dissertation are the mean values of 50 experimental outcomes. The experimental outcome is a performance metric that is recorded when applying the trained model to the test fold.

We apply treatments, or vary the levels (values) of experimental factors in order to test hypotheses or perform case studies. Applying a treatment or varying the value of an experimental factor are synonymous in the context of this dissertation. In order to do so, we will conduct ten iterations of k-fold cross-validation for every combination of treatments in the scope of the case study. For example, if we wish to investigate the effect of the choice of classifier on experimental outcomes, for each classifier designated, we would perform ten iterations of k-fold cross-validation. Now that we have explained cross-validation, we move on to explain the metrics that we record as experimental outcomes.

4.3 PERFORMANCE METRICS

For the experiments covered in this dissertation, we employ Area Under the Precision Recall Curve (AUPRC), and Area Under the Receiver Operating Characteristic Curve (AUC).

Previous research indicates these metrics provide insight for the evaluation of Big Data classification tasks [64, 152]. These metrics are useful for assessing the overall performance of a model without the need to decide on a classification threshold. For extensive research on optimizing thresholds see [90, 92]. For a study on threshold optimization in the presence of class noise, please see [91]. We record these metrics on the outcome of every test round of k-fold cross-validation. Later we use them in statistical analysis.

4.3.1 Area Under the Precision Recall Curve

AUPRC is computed from two threshold-dependent classification performance metrics, precision and recall. Since AUPRC is ultimately computed over many thresholds, ultimately it is not threshold dependent. However, the concept of classification threshold is still important. Therefore, we review the concept of threshold-dependent metrics. To compute the value of a threshold-dependent performance metric of a classifier, a classification threshold must be selected. Moreover, one must employ a machine learning model that assigns a classification probability output value for instances of the dataset. The class an instance is assigned to depends on whether the model's output probability is less than or greater than the threshold value. After the instances of a labeled dataset are assigned to classes, one may evaluate the classification results by comparing the assigned classes to the labels of the dataset [123]. It is important to note that the classification result depends on the threshold selected.

The formula for precision is:

$$\frac{\text{true positives}}{\text{true positives + false positives,}} \tag{4.1}$$

and recall is calculated as the ratio:

$$\frac{\text{true positives}}{\text{true positives + false negatives.}} \tag{4.2}$$

In order to calculate the AUPRC score of a model that has been trained to classify a dataset, we compute many precision and recall scores for small increments of the threshold value ranging from zero to one. For every threshold value, we record a point on a curve such that the x-coordinate of the point is the recall score, and the y-coordinate of the point is the precision score. After the points have been computed, numeric integration techniques are used to compute the area under the curve formed by the points. Models that yield high precision and recall scores for many thresholds will have an AUPRC score close to 1.0, and models that have low precision or recall scores for many thresholds will have an AUPRC score close to zero. A threshold can be selected to make precision or recall high, but only a model with robust performance will have high precision and recall values over a wide range of classification threshold values. We consider AUPRC to be a value aggregated over all possible threshold values. Therefore, we consider AUPRC to be a threshold agnostic classification performance metric.

4.3.2 Area Under the Receiver Operating Characteristic Curve

Area Under the Receiver Operating Characteristic Curve (AUC) is a metric for measuring the performance of a classifier [19]. It is calculated by varying the classification decision threshold from zero to one in small increments and plotting the false positive rate versus the true positive rate that the classifier yields for each value of the decision threshold. Once the points are plotted, a curve is formed, and AUC is the area under this curve. A perfect classifier yields an AUC score of 1.0, and a classifier that assigns instances to classes randomly yields an AUC score of approximately 0.5.

Over the course of our research, we found AUPRC to be a more informative metric than AUC for classification problems involving imbalanced data. An analysis of the components of AUC calculation explains why this is so. First we examine the definitions of the true positive rate and false positive rate. The true positive rate is

$$\frac{\text{true positives}}{\text{true positives + false negatives}} \tag{4.3}$$

and false positive rate equivalent to

$$\frac{\text{false positives}}{\text{true negatives + false positives}}. \tag{4.4}$$

Since true positive rate and recall are actually the same quantity, the difference in AUC and AUPRC must come from the difference between precision and the false positive rate. We see that the false positive rate involves true negatives, whereas precision does not. In highly imbalanced Big Data, where the positive class is the minority class, the true positives in the formula for precision should be small numbers, so that when the number of false positives starts to grow, it can quickly dominate the value of precision. Hence, precision can easily reflect the number of false positives in classifying imbalanced Big Data. A similar analysis of the terms involved in calculating the false positive rate shows that false positives get drowned out due to the size of the negative class. Since the denominator in the definition of the false positive rate is the size of the negative class, and the size of the negative class is large in imbalanced Big Data, a change in the number of false positives may be difficult to perceive.

To solidify the argument, we give an example with some hypothetical numbers. Let us assume we have a dataset where the size of the negative class is two million. Furthermore, let us assume we have done a classification of the data for some output probability threshold, and we have 1,800 true positives and 2,000 false positives. Moreover, the sample has 2,000 positive instances. From these numbers, we can calculate precision and false positive rate. The precision is

$$\frac{1,800}{1,800 + 2,000} \approx 0.47,$$

and the false positive rate is

$$\frac{2,000}{2,000,000} = 0.001$$

Now let us assume that for a different output probability threshold the number of false positives has increased to 4,000. Then the new value of precision is

$$\frac{1,800}{1,800 + 4,000)} \approx 0.31,$$

and the false positive rate is

$$\frac{4,000}{2,000,000} = 0.002$$

In this example, doubling the total number of false positives decreases the precision score by 0.16, but only increases the false positive rate by 0.001. Since both values are used directly as values of coordinates on curves that occupy the same square with area 1×1 in the x-y coordinate plane, we can compare them directly. Therefore, for this example, we conclude that the increase in false positives has a bigger impact on precision and AUPRC, than on the false positive rate and AUC. Therefore, it is possible that if a factor in some experiments causes a larger number of false positives, AUC will not reflect the impact, but AUPRC will. Due to the more informative nature of AUPRC in the classification of large imbalanced datasets, Over the course of our research we took greater interest in AUPRC.

4.4 STATISTICAL TESTS

In the second phase of experiments covered in this dissertation we conduct statistical analyses. These analyses are important because they aid in interpreting the experimental outcomes recorded in the first phase of k-fold cross-validation. In order to answer research questions, and test hypotheses, we vary the values of experimental factors. For example, to determine which classifier is the best for a given dataset, we may experiment with several classifiers. Experimental outcomes, in terms of AUC or AUPRC may be very close.

Variation may be meaningful, or due to random chance. In order to determine whether the differences in experimental outcomes are meaningful, we employ statistical tests. We employ two statistical tests which are described in the following sections: Analysis of Variance, and Tukey's Honestly Significant Difference test.

4.4.1 Analysis of Variance

Analysis of variance (ANOVA) [78] is a statistical method to compare the mean values of experimental outcomes. One may employ ANOVA to assess the variance among the mean values of groups of experimental outcomes. ANOVA is a method for decomposing the total variance observed in a set of experimental outcomes into components attributable to different sources. This enables researchers to discern whether the means of several groups are significantly different from each other, which is crucial in testing hypotheses concerning group differences. ANOVA involves a null hypothesis significance test, where the null hypothesis is that the mean values of different groups of experimental outcomes are not significantly different.

To conduct ANOVA we partition the total variance in the data into components. These components typically include the variance within groups (error variance) and the variance between groups (treatment variance). This is depicted in the SSE term in the ANOVA table, Table 4.1 below. We have reproduced Tables 4.1 and 4.2 from Jain [79]. The within-group variance serves as an estimate of the population error variance. The between-group variance, on the other hand, reflects differences among group means. This is captured in the SSA and SSB terms in Table 4.1. If the treatment variance significantly exceeds the error variance, it suggests that the group means differ more than would be expected by chance alone. SSA and SSB represent groups of experimental outcomes by applying hypothetical treatments 'A' and 'B' in a series of experiments.

Table 4.1: ANOVA Table Part 1: Components and Variation, as specified in Jain [79]

Component	Sum of Squares	Percentage of Variation
y	$SSY = \sum y_{ij}^2$	
$\bar{y}$	$SS0 = ab\bar{y}^2$	
$y - \bar{y}$	$SST = SSY - SS0$	100
A	$SSA = b\sum \alpha_j^2$	$100\left(\frac{SSA}{SST}\right)$
B	$SSB = a\sum \beta_i^2$	$100\left(\frac{SSB}{SST}\right)$
e	$SSE = SST - (SSA + SSB)$	$100\left(\frac{SSE}{SST}\right)$

Table 4.2: ANOVA Table Part 2: Degrees of Freedom, Mean Square, and F-Values, as specified in Jain [79]

Component	Degrees of Freedom	Mean Square	F-Computed	F-Table
y	ab			
$\bar{y}$	1			
$y - \bar{y}$	$ab - 1$			
A	$a - 1$	$MSA = \frac{SSA}{a-1}$	$\frac{MSA}{MSE}$	$F_{1-\alpha,a-1,(a-1)(b-1)}$
B	$b - 1$	$MSB = \frac{SSB}{b-1}$	$\frac{MSB}{MSE}$	$F_{1-\alpha,b-1,(a-1)(b-1)}$
e	$(a-1)(b-1)$	$MSE = \frac{SSE}{(a-1)(b-1)}$		

The statistical significance in ANOVA is determined by the F-test. The F-test compares the ratio of treatment variance to error variance. The ratio is known as the test statistic. A significant F-test indicates that the variances of the group means are sufficiently large relative to the variance within the groups, suggesting that at least one of the group means is significantly different from the others. A significance level must be decided upon in order to determine whether the test statistic indicates a statistically significant outcome. For the experiments covered in this dissertation, we select a significance level of $\alpha = 0.01$.

The R programming language [142] provides a library for automating the computations

outlined in Tables 4.1 and 4.2. We rely on this library function for computing ANOVA in the statistical analyses for our experiments. The library provides a probability for the value of the test statistic that we then compare to our selected significance level. If that value is less than, or equal to the value of our significance level, we reject the null hypothesis that the treatment has no effect on experimental outcomes, and we accept the alternative hypothesis that the treatment has an effect.

In conclusion, ANOVA is a tool that allows one to make an objective decision about the effect of a treatment on experimental outcomes. It can be difficult for a human to look at tables of experimental results grouped by treatments and determine whether the treatment causes a significant change in the outcome. In these circumstances one may quickly employ an automated ANOVA test to assist in making the decision whether the treatment has a significant effect.

4.4.2 Tukey's Honestly Significant Difference Test

Tukey's Honestly Significant Difference (HSD) Test [165] is a post-hoc analysis. Here, "post-hoc" means "formulated after the event." In the context of our methodology, the event is an ANOVA test. When an ANOVA test indicates that the treatment's impact on experimental outcomes is statistically significant, it does not specify which levels of the treatment make significant differences. The HSD test addresses this gap by performing comparisons between pairs of group means. The HSD test refines the output of the ANOVA test.

We utilize the R programming language `agricolae` library's `HSD.test` function to conduct HSD tests. The input to `HSD.test` is the output of the R language ANOVA function mentioned in the previous section. The result of the HSD test is a grouping of levels of factors where group 'a' is linked with the largest value of the metric recorded as the experimental outcome, group 'b' is commensurate with the second-highest values recorded as the experimental outcome, and so on. A group may have one or more members.

The experimental outcomes yielded by members of the same group are not significantly different. Experimental outcomes yielded by members of different groups are significantly different

4.5 METHODOLOGY EXAMPLE

Here we present an example of our experimental methodology. The AUC scores listed in Table 4.3 are reproduced from a study by Hancock and Khoshgoftaar [50]. In their study they wished to determine which maximum tree depth yielded the best performance in Terms of AUC. The AUC scores listed in Table 4.3 are from classifying the non-aggregated Medicare Part D data discussed in Chapter 2. The experimental outcomes are AUC scores. Table 4.3 holds the mean values of 50 experimental outcomes for each level of maximum tree depth. Put another way, 5-fold cross-validation was employed to generate the experimental outcomes summarized in Table 4.3.

Table 4.3: Performance of Classifiers with varying levels of maximum tree depth; Mean AUC Scores (10 iterations of 5-fold cross-validation)

Depth	6	16	24	32	48
RF-GPU	N/A	0.80821	0.95195	0.96996	0.96846
	N/A	0.00091	0.00078	0.00031	0.00039
XGB	0.75633	0.92450	0.97273	N/A	N/A
	0.00071	0.00141	0.00041	N/A	N/A

Table as presented in [50]; XGBoost (XGB) and Random Forest GPU implementation (RF-GPU) mean AUC scores, and standard deviations by maximum tree depth; mean AUC scores are on the line with the classifier name, and standard deviations are directly beneath the AUC scores.

The AUC scores are close. In the context of the experiments, varying the maximum tree depth factor is equivalent to applying a treatment. In order to determine if varying the maximum depth factor yields a significant impact on AUC scores, Hancock and Khosh-

goftaar apply an ANOVA test. The result of the ANOVA test is listed in Table 4.4. The F statistic value in the ANOVA test result is 710,768.46, which is associated with a very small probability, less than 10^{-5}. Therefore, the maximum depth factor has a statistically significant impact on experimental outcomes.

Table 4.4: ANOVA for Depth as a factor of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Depth	3	0.91380	0.30460	710768.46	*
Residuals	196	0.00010	*		

* indicates the value is less than 10^{-5}.

Since the effect of maximum tree depth has a significant impact on experimental outcomes, the Tukey HSD test will determine which groups of mean values are significantly different, which groups are associated with the highest AUC scores, the second-highest AUC scores and so on. The result of the HSD test is in Table 4.5. In this case each level of the factor is in a group by itself. Interestingly the HSD result shows that the second-largest value for maximum tree depth yields the highest AUC scores.

Table 4.5: HSD test groupings after ANOVA of AUC for the Depth factor

Group a consists of: Depth 32
Group b consists of: Depth 48
Group c consists of: Depth 24
Group d consists of: Depth 16

4.6 CHAPTER SUMMARY

This chapter covers the four key components of the experimental methodology we employ in the experiments covered in this dissertation. The first component is k-fold cross-validation.

In the test phases of k-fold cross-validation, we record experimental outcomes as AUC or AUPRC scores. We then use these scores as input to an ANOVA test to determine whether experimental treatments have a statistically significant impact on the experimental outcomes. When the ANOVA test indicates the treatments have a significant impact, we employ the HSD test to rank the levels of the factors in terms of their impact on experimental outcomes.

CHAPTER 5

SAMPLING TECHNIQUES

5.1 INTRODUCTION

Since the theme of this dissertation is data reduction techniques, sampling is a natural subject for us to include. By definition sampling implies using fewer instances of a dataset, and is therefore a data reduction technique. In the research covered, we use two sampling techniques: random undersampling (RUS), and a variation on RUS which we call one-class sampling. Sampling techniques have been employed successfully with a range of small class ratios, from what is considered severely imbalanced [63] all the way to what we identify as rarity [66].

In the context of highly imbalanced datasets, conventional classification algorithms tend to exhibit a bias towards the majority class, often leading to suboptimal classification performance, especially in terms of sensitivity or recall for the minority class [117]. This imbalance poses no issue for One-Class Classifiers (OCCs), which, by design, are trained on instances of one-class. However, results of experiments conducted indicate that sampling to reduce training data size is still beneficial in the one-class scenario. In this chapter we discuss RUS and one-class sampling, and then provide examples of research where these techniques are used. We show that RUS as a data reduction technique for Binary Class Classifiers (BCCs) is not as attractive as our sampling technique for data reduction with OCCs. Our results show that RUS is an effective data reduction technique if only the Area Under the Reciever Operating Characteristic Curve (AUC) [19] score is important, but not as effective when the Area under the Precision Recall (AUPRC) [20] score is also important. However, our results in the one-class scenario show that OCCs are robust to

one-class sampling in terms of AUC and AUPRC.

OCCs and BCCs are used in different scenarios, so their performance should not be directly compared. In the OCC scenario, data of only one class is available; hence it is not possible to use BCCs. Therefore, results about data reduction techniques that apply for BCCs do not apply to OCCs. Hence, it would be erroneous to combine results from studies on OCCs and BCCs to draw overall conclusions about the effect of a data reduction technique. Therefore, we treat RUS, a technique defined for BCCs, separately from one-class sampling, which is a technique defined for OCCs.

5.2 RANDOM UNDERSAMPLING

The classifiers discussed in this section of the dissertation are all BCCs. Hence, the research here applies to the BCC scenario, where two classes of instances are available for model training and testing. In [56] we define RUS as the process where, we randomly discard instances of the majority class until the class ratio reaches a specified level. For example, in many studies we use ratios 1:1, 1:3, 1:9, 1:27, or 1:81. In practice, we use the Imblearn [131] library's RandomUnderSampler module to control class ratios for all experiments. For a study on performance with extremely imbalanced class ratios please see [13]. Cost-sensitive learning is alternative to undersampling successfully employed by our research group [93].

There is a large body of work that supports RUS as the best technique for working with highly imbalanced data. For examples, see [180, 182]. Since Bauder and Khoshgoftaar find supervised learners to be best suited to the task of identifying fraud in Medicare data in [2,4], a logical next step is to focus on supervised learners and undersampling techniques. In [10], Bauder and Khoshgoftaar study Random Forest's ability to detect anomalous behavior in Medicare data with various class imbalance ratios. They control the imbalance ratios with RUS. We see that in [10], Bauder and Khoshgoftaar report the best performance of Random Forest with a mean AUC value of 0.87302 with a class distribution of 90% of instances labeled non-fraudulent to 10% of instances labeled as fraudulent. Building on the

success of [10], in [81], and [82], Johnson and Khoshgoftaar investigate the performance of Deep Learning algorithms on the Medicare fraud detection task. The application of Deep Learning to Medicare fraud detection is informed by the survey on Deep Learning to classify imbalanced data [84]. In [82], Johnson and Khoshgoftaar expand the scope of sampling techniques, in conjunction with varying the classification threshold of learners. They show that a hybrid technique of RUS and Random Oversampling have an impact on Deep Learning algorithms' performance in terms of AUC.

In 2018, Bauder *et al*. released a study on sampling techniques for classifying imbalanced Big Data [14]. Their application domain is the same as ours, Medicare Fraud detection. However, the authors perform experiments on a smaller dataset, based on a combination of data from Medicare Parts A, B, and DMEPOS. Their combined dataset contains fewer than one million instances. In their study, Bauder *et al*. use six data sampling techniques: Random Oversampling (ROS), Adaptive Synthetic (ADASYN) [71], Synthetic Minority Oversampling Technique (SMOTE) [37], two variations of Borderline SMOTE (also covered in [37]), and RUS. They use each sampling technique to induce class ratios of 50:50, 65:35, 25:75, 10:90, and 1:99. In addition, they conduct experiments with models trained on the dataset in its initial state, which has a class ratio of 473:759,267. All their experiments are conducted as programs running on the Apache Spark [176] platform. Therefore, they use the Apache Spark implementations of Gradient Boosted Trees [137], Logistic Regression [113], and Random Forest. Bauder *et al*. evaluate experimental outcomes in terms of AUC only. Their conclusion is that classifiers fitted to data that is pre-processed with RUS yield the best performance. Hence, RUS is the only sampling technique we use in experiments covered in this dissertation. However, in addition to AUC, we also evaluate experimental outcomes in terms of the AURPC metric. Like Bauder *et al*., we find that classifiers trained on data that is treated with RUS yield higher AUC scores, but we go on to show that RUS can be a detriment to AUPRC scores. This is an important aspect of our study that sets our research apart from Bauder *et al*.'s.

5.3 ONE-CLASS SAMPLING

One-class sampling is something more unique to our research. In [51] we introduce our technique for applying sampling for data reduction in the one-class scenario. We propose and demonstrate a data reduction technique to improve the performance of one-class classifiers for highly imbalanced Big Data. We focus on insurance fraud detection in the aggregated Medicare Part D prescription drug claims data discussed in Chapter 2. The novelty of our methodology with regard to this data reduction technique is that, to the best of our knowledge, we are the first to document such a technique in conjunction with One-Class GMM's. For an in-depth study on popular sampling techniques, including RUS please see [81]. We would like to draw a distinction between this data reduction technique and RUS. Our definition of RUS matches the definition of RUS carried out by the RUS function of the Python imbalanced-learn library [132]. In RUS instances of the majority class are discarded until the minority to majority class ratio reaches a desired level. Hence, instances of the minority class are retained. In the experiments documented here, we first discard all the elements of the minority class, because we are doing one-class classification, then we take a sample of the majority class.

To prove the novelty of our one-class sampling approach for One-Class SVM and One-Class GMM, we conducted a literature review and were unable to find studies that involve applying OCCs and data reduction techniques to Big Data [51]. Of the research we could find, the studies propose novel sampling techniques, or ensembles of classifiers. We argue that our methods are easier to reproduce since we use publicly available software for classifiers and data reduction. Nevertheless, the tools we use are combined in a novel manner. Our search for related work revealed that the experiments documented in this dissertation are the first on a data reduction technique for OCCs applied to Big Data.

5.4 MEDICARE DATASETS

In this section we cover experiments where RUS and one-class sampling are applied in experiments involving the Medicare datasets discussed in Chapter 2. RUS is the simplest data reduction technique. It was an element of our earliest research which we continued to expand and refine. The factor that sets our initial research with RUS apart from previous research with RUS and Big Medicare data is that we were the first to apply RUS to the latest publicly available Medicare data. In the research documented in this chapter, we show that RUS has a negative impact on the classification of Medicare datasets that is not reflected in AUC scores, but is reflected in AUPRC scores. Hence, RUS is an effective data reduction technique when AUC is important, but if AUPRC is also important, RUS is a less effective data reduction technique.

5.4.1 Initial Result

The first study of ours which involves RUS and Medicare datasets is "Performance of CatBoost and XGBoost in Medicare Fraud Detection" [44]. This study took inspiration from a previous study by Herland *et al*. This is another study which indicated that the best technique for addressing class imbalance for the task of fraud detection in Medicare data is RUS to induce a class ratio of 1:1 [73]. Therefore, in [44], we use a 1:1 class ratio as well. This is the only sampling ratio used in the study. However, we were able to build models that yield strong performance in terms of AUC with the 1:1 class ratio. In [44], we use the CatBoost and XGBoost learners that were described in Chapter 3. We conduct experiments with aggregated Medicare Part B data. The mean AUC score for ten iterations of five-fold cross validation for CatBoost is 0.9080. The similar AUC score for XGBoost is 0.8616. These results inspired continued research to determine whether data reduction could yield better performance.

5.4.2 RUS as a Treatment for Data Reduction

This lead to research such as documented in "The Effects of Random Undersampling for Big Data Medicare Fraud Detection" [55], where RUS yields an improvement in AUC scores. This is the first research where we apply RUS as a treatment for data reduction. We show it is possible to obtain better classification performance for experiments involving highly imbalanced Big Data with the application of data sampling techniques. We apply RUS to the non-aggregated Part D data described in Chapter 3. We find that RUS significantly improves the classification performance of Extremely Randomized Trees and XGBoost learners in a Medicare Big Data fraud classification task. We employ RUS to evaluate performance at multiple minority:majority class ratios. According to the outcome of a Tukey's Honestly Significant Difference test, we find RUS to the 1:9 or 1:27 class ratios yields the best performance, providing AUC scores of over 0.97. Models built with undersampled Big Data require significantly less time to train. In [55], our contribution is to prove the effectiveness of RUS in classifying Medicare Part D data. Our review of related work shows this study is the first to apply RUS to data on the scale of the non-aggregated Part D data in order to prove one can obtain better performance. The non-aggregated Medicare Part D data has approximately 175 million instances.

In [55], we report results for the XGBoost and Extremely Randomized Trees classifiers. Please see Chapter 3, on learners for details about these classifiers. We report the mean AUC scores XGBoost and ET yield for ten iterations of five-fold cross validation. We see the highest mean AUC score for ET in Table 5.6 is associated with a class ratio of 1:1. On the other hand, for XGBoost we see the highest mean AUC score is associated with a class ratio of 1:27. This appeared to be wonderful news. We could apply a simple data reduction technique, RUS, and build models that yield better performance. Unfortunately, this result does not hold for the important AUPRC metric, as we found in subsequent studies.

Our results show that RUS has a significant, positive impact on the performance of ET and XGBoost in terms of AUC. If AUC happens to be the only metric that is important

for one's application, then that is an ideal result, since we obtain better performance with a smaller investment of computing time. In [55], we see that over all combinations of classifier and RUS level, ET yields the best performance with RUS applied to make class ratios 1:9 or 1:27. This study demonstrates the positive impact RUS has on classification results of the non-aggregated Part D data. However, subsequent work discussed below shows RUS does not have a similar effect on AUPRC scores.

5.4.3 RUS impact on AUPRC versus AUC

In a subsequent study, "Informative Evaluation Metrics for Highly Imbalanced Big Data Classification" [57], we took a more in-depth look at RUS, and its impact as reflected by the AUC and AUPRC performance metrics. We conduct experiments with ET and XGBoost that show the AUPRC metric provides a more meaningful insight into the impact of RUS than AUC. Evaluating experiments with multiple metrics is a robust method for overcoming challenges in Machine Learning, such as class imbalance. In the research conducted in [57], we report RUS may provide an improvement to AUC scores; however, at the same time, RUS may be detrimental to AUPRC scores. AUPRC is a metric that involves precision, whereas AUC does not. As discussed in the section on metrics in Chapter 4, on methodology, in the classification of imbalanced Big Data, an increase in false positive counts has a more noticeable drop in precision scores. Therefore, in application domains where false positives are undesirable, optimizing models for AUPRC is a wise choice. In [57], our contribution is to compare the performance of models in terms of AUPRC and AUC to show the impact of RUS on the classification of imbalanced Big Data. We compare the performance via experiments in the classification of highly imbalanced Big Data. Models are built with data in its original class ratio, and with data undersampled into 5 distinct class ratios. We report the results of 600 experiments where we apply RUS to the non-aggregated Part D data.

Analysis of Variance (ANOVA) [78] tests indicated that RUS has a significant impact on AUC and AUPRC scores. Here we summarize HSD results in the study, since a subsequent

study has results for the expanded set of experiments. In HSD tests for this study, we see the 1:9 and 1:27 RUS levels associated with the best performance when averaged across results for ET and XGBoost. Moreover, we see the HSD group 'ab' contains the 1:3 and 1:81 levels of the RUS factor. The 'ab' label stems from the fact that the range of AUC scores in this group overlap with in the ranges of scores in groups 'a' and 'b'. Furthermore, we see applying RUS to induce the 1:1 class ratio is associated with the worst performance. Finally, the initial class ratio, where no RUS is applied, is the group with the third-best performance. We averaged AUC scores across both ET and XGB-24 as part of the HSD

Similar to the analysis we do for AUC scores, in [57] we conducted a second ANOVA test for the impact of Classifier and RUS on AUPRC scores. The ANOVA test results show both RUS and choice of classifier have a significant impact on experimental outcomes. The HSD test result for the RUS factor reinforce the key finding at this stage of our research: for the XGBoost and ET classifiers, RUS can be used to build models which yield performance in terms of AUPRC, which is not significantly different from using all data. The 1:27 and 1:81 class ratios induced by RUS yield performance similar to training with data at the original class ratio. In subsequent research, we had hoped to find that the RUS data reduction technique would yield better performance with other classifiers.

5.4.4 Expansion of experiments to assess impact of RUS

In order to confirm whether models built with training data treated with RUS yield performance at least as good as models built with the original training data, we expanded the number of datasets and classifiers used in experiments [58, 59]. Here we include results of these experiments.

We evaluate the performance of five ensemble learners in the Machine Learning task of Medicare fraud detection. RUS is applied to induce five class ratios. The classifiers are evaluated with both AUC and AUPRC metrics. We show that AUPRC provides a better insight into classification performance. Our findings reveal that the AUC metric hides

the performance impact of RUS in the expanded experimental methodology. However, classification results in terms of AUPRC show RUS has a detrimental effect. We show that, for highly imbalanced Big Data, the AUC metric fails to capture information about precision scores and false positive counts that the AUPRC metric reveals. Our contribution in [59] and [58] is to show AUPRC is a more effective metric for evaluating the effect of data reduction on performance when working with highly imbalanced Big Data.

The use of a single metric to draw conclusions on the impact of a factor in classification experiments may lead to mistakes that we wish to help our fellow researchers avoid. RUS is an appealing strategy for mitigating class imbalance in Big Data. It can drastically reduce the size of the training data used during the model training phase of Machine Learning. Less training data translates into faster training times for many Machine Learning algorithms. Therefore, applying RUS may save a researcher time when conducting experiments. Our contribution is to reveal that there is a trade-off for applying RUS that one should consider before concluding that applying RUS is the best choice. We show that RUS may have a positive impact on AUC [19] scores. At the same time, we show RUS may have a clear negative impact on AUPRC [20] scores.

This underscores the importance of evaluating results in terms of more than one metric. Therefore, if performance in terms of AUPRC is important, applying RUS may not be a viable option. We validate these findings using three distinct data sets and five popular ensemble learners in the task of Medicare fraud detection. In our experiments, we apply RUS to induce five different levels of minority:majority class ratios, and classify datasets of varying sizes. The smallest dataset we work with has approximately 12 million instances. We also perform experiments with a dataset that has approximately 68 million instances, and another dataset that has approximately 175 million instances. For each dataset we find the same pattern holds: AUC scores are either not affected, or improved by RUS, and AUPRC scores are degraded, except for the Extremely Randomized Trees classifier and RUS to reduce data to the 1:81 minority:majority class ratio.

We report AUC and AUPRC scores for classification results. The values for AUC and AUPRC reported here are mean values computed by averaging 50 experimental outcomes. One round of five-fold cross validation yields one experimental outcome consisting of one AUC and AUPRC score. Since we do 10 iterations of five-fold cross validation, we obtain 50 instances of each metric. We report the same data in graphical and tabular form. The graphical form shows relative performance and the trend in results as the size of the majority class is increased. In addition, we provide the data in tabular form along with standard deviations to give a sense of the spread of AUC and AUPRC scores over the ten iterations of five-fold cross validation.

As stated previously we apply RUS to induce a class ratio in the training data. We do not alter the class ratio in the test data. Therefore, scores one sees reported in this section are results for classifying data sampled from its original class ratio. Put another way, models are trained on data with RUS applied, and only evaluated on test data with the original class ratio.

Related Works

"Data sampling approaches with severely imbalanced big data for medicare fraud detection" is a 2018 study by Bauder *et al* [14]. In their study, the authors combine Part B, Part D, and DMEPOS Medicare claims data to form a dataset for Medicare fraud detection via classification. Hence, their study is in the same application domain as ours, albeit with fewer data than we use, since we use data was not available at the time their study was written. The data they work with has under one million instances. Bauder *et al*. employ six data sampling techniques in their experiments. The six techniques are RUS, Random Oversampling (ROS), Synthetic Minority Oversampling Technique (SMOTE) [37], two variations of Borderline SMOTE (also covered in [37]), and Adaptive Synthetic Sampling Technique (ADASYN) [71]. The sampling techniques are used to induce minority:majority class ratios of 1:99, 10:90, 25:75, 65:35, and 50:50. Experiments with the original class

ratio of 473:759,267 (approximately 0.00062) are performed as well. For classification experiments, they use Apache Spark [176] implementations of Random Forest, Logistic Regression [113] and Gradient Boosted Trees [137]. To evaluate the performance of the combinations of classifiers and data sampling techniques, the authors use AUC. Bauder *et al*. conclude that classifiers trained on data with RUS applied to it yield significantly better performance, in terms of AUC, than classifiers trained on data with the original class ratio. Since Bauder *et al.* prefer RUS to other sampling techniques, we employ only RUS. However, we measure classification results in terms of the AUPRC metric as well. Our results are unique and meaningful, since, on one hand we confirm Bauder *et al*.'s results that show an improvement in AUC scores when RUS is applied, but on the other hand, we show that AUPRC scores decline when RUS is used.

Hasanin *et al*. [65] study the effect of RUS on Geometric Mean [112] and AUC scores. They find RUS has a positive impact on AUC and Geometric Mean scores in the classification of imbalanced Big Data. Here, we investigate the performance of RUS on AUC and AUPRC scores. One advantage AUC and AUPRC have over the Geometric Mean is that they reflect performance over a range of model output probability threshold values, whereas in order to calculate Geometric Mean, one must select a specific output probability threshold. Therefore, a model's Geometric Mean score can only tell us about the performance of the model for one particular threshold value. Another issue that sets our study apart from Hasanin *et al*. is that the largest dataset used in their study has under 1.7 million instances. The data we work with here is orders of magnitude larger. Hasanin *et al*. report that they use one-hot encoding for all categorical features. Here we use CatBoost encoding [141], a technique that is more scalable than one-hot encoding since it does not require the introduction of additional attributes to the dataset. For example, to one-hot encode a categorical value that has thousands of possible values, we would need to add thousands of attributes to our dataset. However, CatBoost encoding requires no additional space consumption.

In "Threshold based optimization of performance metrics with severely imbalanced big security data" Calvert and Khoshgoftaar use multiple metrics to evaluate multiple classifiers [24]. The application domain for their study is information systems network security. Hence, their results reveal the ability of Machine Learning algorithms to detect malicious network traffic. Since most of the traffic in their dataset is benign, the classification task is an exercise in the classification of imbalanced data. The data they use in their experiments has approximately 1.7 million instances. To give a sense of the level of class imbalance, the dataset Calvert and Khoshgoftaar use has a minority to majority class ratio of approximately 0.0014. Therefore, their data resembles ours, however, we find our dataset is more realistic, since Calvert and Khoshgoftaar generate the malicious traffic in the raw network traffic data they use in their experiments. In a sense our raw data is more natural, since we do not play a role in generating it. Their key finding is that one classifier yields the best performance in terms of AUC, but significantly worse in terms of other metrics. Furthermore, they find one classifier yields the best performance in multiple metrics other than AUC. They claim this result indicates AUC alone cannot identify the best performing model. The principal item that differentiates our study from Calvert and Khoshgoftaar's is that they do not use RUS as a factor in any of their experiments.

Johnson and Khoshgoftaar document experiments with Neural Network-based classifiers and RUS in [83]. The performance metrics they use to evaluate classification results are AUC, Geometric Mean, True Positive Rate, and True Negative Rate. The application domain for their study is Medicare fraud detection. One thing that sets research covered in this dissertation apart from Johnson and Khoshgoftaar's is our use of the AUPRC metric. Our studies have use of the AUC metric in common. Johnson and Khoshgoftaar find that when RUS is applied to their data to induce a class ratio with the minority class occupying more than 20% of the data, AUC scores begin to deteriorate. In a further demonstration on the effect the RUS technique, they show that all metrics reflect worsening performance as RUS is used to grow the proportion of the minority class in the training data. Johnson and

Khoshgoftaar work with data similar to ours, however, they apply an aggregation step to prepare their data for experiments. The aggregation reduces the size of their dataset to under five million instances. The aggregation also eliminates the highest cardinality categorical features. They use one-hot encoding for the remainder of the categorical features. For a study on many available options for encoding categorical features, please see [45]. For experiments reported in this section of the dissertation, we selected CatBoost encoding, which has the advantage of supporting much higher cardinality categorical features than can be practical with one-hot encoding. Due to the differences we have listed here, our study represents a contribution that is separate from what Johnson and Khoshgoftaar have to offer.

In a later work, Johnson and Khoshgoftaar use Geometric Mean and AUC to evaluate the performance of Deep Learning algorithms to classify imbalanced Big Medicare Data [85]. They apply RUS to have the minority class constitute larger percentages of the training data. The dataset used in their experiments has approximately five million instances. Results in this study show metrics are even more sensitive to RUS than their previous study. In this study, they find performance, in terms of both metrics, begins to suffer when the minority class becomes more than one percent of the training data. One major difference between our study and Johnson and Khoshgoftaar's is that we use AUPRC, whereas they use Geometric Mean. Also, as in their previous study, Johnson and Khoshgoftaar use an aggregation technique which eliminates high-cardinality categorical features, and then use one-hot encoding to encode remaining categorical features. For reasons mentioned previously, we prefer to apply CatBoost encoding to our categorical features. Hence, our study also differs significantly from this second study by Johnson and Khoshgoftaar.

Related works cover concepts that overlap with research reported in this section of our dissertation. We find research where RUS is a factor in experiments with highly imbalanced Big Data. However, we do not find a study that reveals insights into the divergent effect of RUS on AUC and AUPRC scores in the classification of highly imbalanced Big Data.

We feel our contribution is an important one since it shows that focus on AUC alone can cause one to overlook the negative impact of RUS, and therefore possibly other factors, on classification performance.

Results

The first results we report are the AUC and AUPRC scores the five classifiers yield for classifying the DMEPOS data. We provide plots of AUC and AUPRC scores in Figure 5.1. Due to space limitations, we use the following abbreviations for classifier names in the tables and figures in this section. Classifier names are abbreviated as follows: CatBoost with maximum tree depth set to 16: CB-16, LightGBM: LGB, Extremely Randomized Trees: ET, and the GPU implementation of Random Forest with maximum tree depth set to 32 RF-GPU-32. For studies on the impact of GPUs on the classification of Big Data, please see [103, 104]. In Figure 5.1 one may notice some facts that hold for other data as well. Mean AUC scores are high, and show little impact of RUS as the size of the majority class increases. However, there is more variance in results in terms of the mean AUPRC metric. ET yields consistently strong performance as RUS is applied to make the majority class larger in the training data. Random Forest and XGBoost yield better performance in terms of AUPRC when we apply RUS to make the size of the majority class larger. CatBoost and LightGBM yield relatively poor performance in terms of AUPRC regardless of how RUS changes.

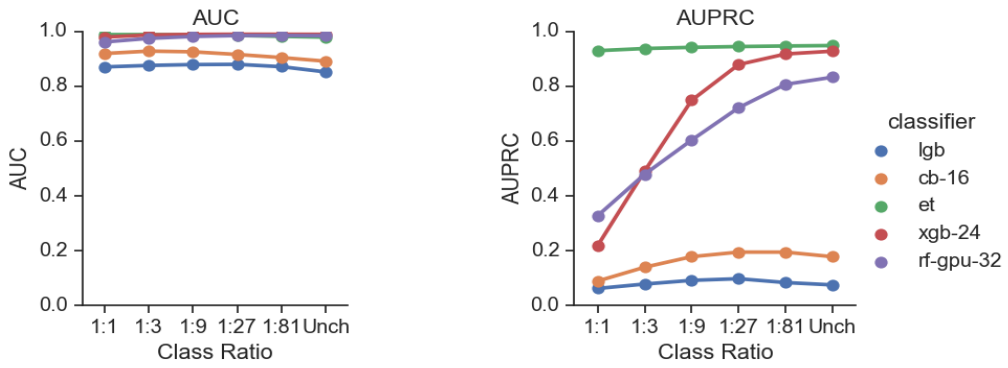

Figure 5.1: DMEPOS Data: AUC Scores (left) and AUPRC scores (right)

Table 5.1: DMEPOS Mean and Standard Deviation of AUC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	LGB	CB-16	ET	XGB-24	RF-GPU-32
1:1	0.87070	0.91952	0.98862	0.97944	0.96163
	(0.00145)	(0.00134)	(0.00059)	(0.00077)	(0.00082)
1:3	0.87610	0.92829	0.98874	0.98732	0.97472
	(0.00161)	(0.00108)	(0.00072)	(0.00067)	(0.00082)
1:9	0.87922	0.92575	0.98774	0.98933	0.98175
	(0.00169)	(0.00152)	(0.00095)	(0.00069)	(0.00060)
1:27	0.88011	0.91607	0.98536	0.98992	0.98511
	(0.00165)	(0.00174)	(0.00111)	(0.00069)	(0.00066)
1:81	0.87140	0.90471	0.98207	0.98977	0.98549
	(0.00241)	(0.00192)	(0.00124)	(0.00063)	(0.00074)
Unchanged	0.85223	0.89143	0.97917	0.98959	0.98453
	(0.00492)	(0.00207)	(0.00113)	(0.00069)	(0.00104)

Standard deviations are below AUC scores in parentheses.

Table 5.2: DMEPOS Mean and Standard Deviation of AUPRC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	LGB	CB-16	ET	XGB-24	RF-GPU-32
1:1	0.06233	0.08935	0.92987	0.21863	0.32763
	(0.00319)	(0.00453)	(0.00267)	(0.00823)	(0.01293)
1:3	0.07842	0.13984	0.93779	0.49056	0.47900
	(0.00294)	(0.00422)	(0.00258)	(0.00875)	(0.01033)
1:9	0.09192	0.17826	0.94208	0.74880	0.60368
	(0.00305)	(0.00497)	(0.00265)	(0.00968)	(0.00678)
1:27	0.09788	0.19499	0.94493	0.87928	0.72180
	(0.00404)	(0.00510)	(0.00267)	(0.00534)	(0.00474)
1:81	0.08387	0.19489	0.94710	0.91788	0.80715
	(0.00566)	(0.00508)	(0.00301)	(0.00324)	(0.00387)
Unchanged	0.07531	0.17815	0.94866	0.92811	0.83319
	(0.00752)	(0.00474)	(0.00213)	(0.00316)	(0.00547)

Standard deviations are below AUPRC scores in parentheses.

Next, we report results for classifying the Part B data, starting with Figure 5.2. Results are similar to those we obtain for the DMEPOS data. Again, all classifiers show strong performance in terms of mean AUC. However, the mean AUPRC scores reveal a different picture altogether. Consistent with results for DMEPOS, we see ET yields consistently strong AUPRC scores. XGBoost and Random Forest AUPRC scores appear to be in some direct proportion with the size of the majority class in the training data. Also in keeping with their performance in classifying the DMEPOS data, LightGBM and CatBoost yield low AUPRC scores for any level of RUS.

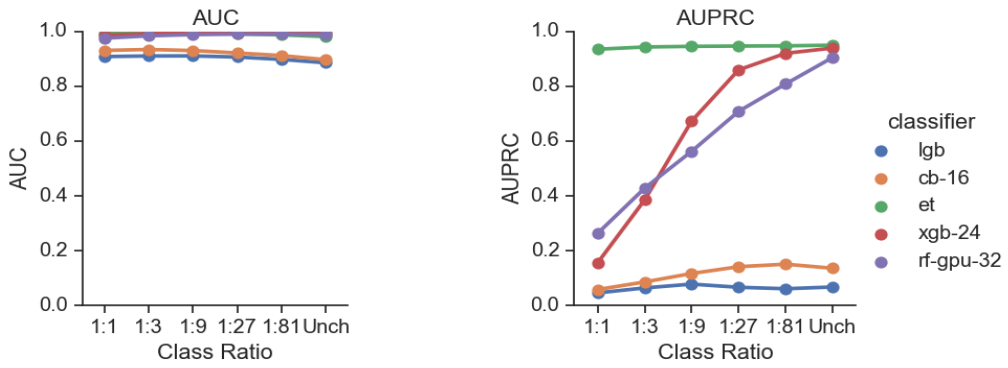

Figure 5.2: Part B Data: AUC scores (left) and AUPRC scores (right)

Table 5.3: Part-B Mean and Standard Deviation of AUC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	LGB	CB-16	ET	XGB-24	RF-GPU-32
1:1	0.90846	0.93072	0.99316	0.98621	0.97534
	(0.00086)	(0.00087)	(0.00039)	(0.00039)	(0.00039)
1:3	0.91054	0.93427	0.99323	0.99219	0.98404
	(0.00078)	(0.00089)	(0.00037)	(0.00028)	(0.00038)
1:9	0.91087	0.93028	0.99259	0.99394	0.98849
	(0.00102)	(0.00101)	(0.00041)	(0.00025)	(0.00043)
1:27	0.90664	0.92148	0.99085	0.99443	0.99050
	(0.00241)	(0.00127)	(0.00041)	(0.00032)	(0.00041)
1:81	0.89803	0.91200	0.98764	0.99458	0.99069
	(0.00530)	(0.00104)	(0.00051)	(0.00028)	(0.00048)
Unchanged	0.88554	0.89698	0.98118	0.99436	0.98862
	(0.00542)	(0.00121)	(0.00080)	(0.00033)	(0.00051)

Standard deviations are below AUC scores in parentheses.

Table 5.4: Part-B Mean and Standard Deviation of AUPRC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	LGB	CB-16	ET	XGB-24	RF-GPU-32
1:1	0.04621	0.05853	0.93517	0.15584	0.26342
	(0.00253)	(0.00238)	(0.00209)	(0.00419)	(0.01037)
1:3	0.06426	0.08606	0.94335	0.38531	0.42815
	(0.00217)	(0.00236)	(0.00160)	(0.00786)	(0.00815)
1:9	0.07818	0.11695	0.94563	0.67304	0.56271
	(0.00513)	(0.00224)	(0.00129)	(0.00795)	(0.00532)
1:27	0.06664	0.14166	0.94682	0.85988	0.70862
	(0.00761)	(0.00309)	(0.00155)	(0.00376)	(0.00746)
1:81	0.06128	0.15084	0.94758	0.92030	0.80848
	(0.01261)	(0.00299)	(0.00150)	(0.00167)	(0.00270)
Unchanged	0.06768	0.13607	0.94990	0.93947	0.90510
	(0.01071)	(0.00254)	(0.00162)	(0.00156)	(0.00182)

Standard deviations are below AUPRC scores in parentheses.

Finally, we report classification results for Part D data. We notice in Figure 5.3 that while mean AUC scores are high, with the larger dataset there is a bifurcation of performance in terms of AUC. XGBoost, ET, and Random Forest yield AUC scores that appear noticeably higher than those of LightGBM and CatBoost. For classifying the Part D data, mean AUPRC scores are generally lower than those we see for the Part B or DMEPOS data. However, the same pattern we saw in the results for the Part B and DMEPOS data appears again in the Part D data; ET yields consistently high results, Random Forest and XGBoost scores improve as we apply RUS to increase the size of the majority class, and LightGBM and CatBoost yield low scores relative to the other classifiers.

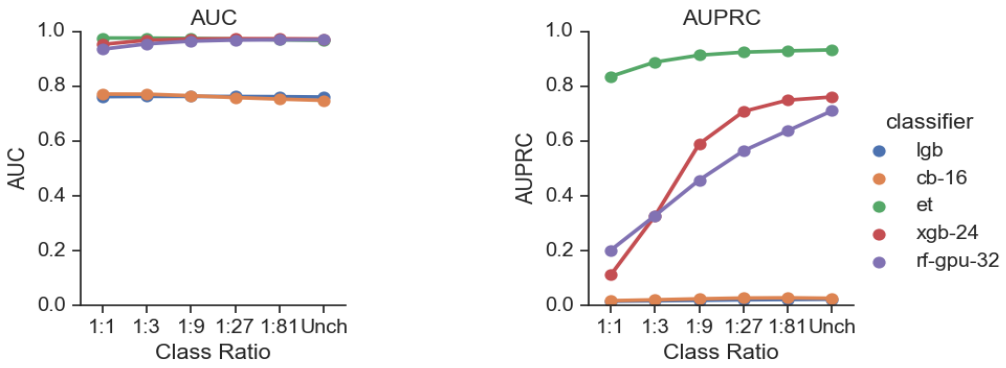

Figure 5.3: Part D Data: AUC scores (left) and AUPRC scores (right)

Table 5.5: Part-D Mean and Standard Deviation of AUC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	LGB	CB-16	ET	XGB-24	RF-GPU-32
1:1	0.76230	0.77169	0.97625	0.95292	0.93523
	(0.00067)	(0.00075)	(0.00033)	(0.00059)	(0.00046)
1:3	0.76326	0.77125	0.97586	0.96809	0.95465
	(0.00060)	(0.00074)	(0.00038)	(0.00045)	(0.00024)
1:9	0.76317	0.76571	0.97464	0.97256	0.96476
	(0.00055)	(0.00080)	(0.00038)	(0.00037)	(0.00040)
1:27	0.76279	0.75888	0.97244	0.97363	0.96868
	(0.00060)	(0.00062)	(0.00037)	(0.00037)	(0.00030)
1:81	0.76212	0.75329	0.96966	0.97348	0.96963
	(0.00064)	(0.00089)	(0.00035)	(0.00039)	(0.00037)
Unchanged	0.76085	0.74847	0.96746	0.97273	0.96996
	(0.00081)	(0.00076)	(0.00038)	(0.00042)	(0.00031)

Standard deviations are below AUC scores in parentheses.

Table 5.6: Part-D Mean and Standard Deviation of AUPRC with varying levels of RUS (10 iterations of 5-fold cross-validation)

Class Ratio	LGB	CB-16	ET	XGB-24	RF-GPU-32
1:1	0.01567	0.01766	0.83614	0.11168	0.20205
	(0.00015)	(0.00020)	(0.00307)	(0.00158)	(0.00322)
1:3	0.01724	0.02096	0.88826	0.32721	0.32806
	(0.00016)	(0.00021)	(0.00181)	(0.00572)	(0.00322)
1:9	0.01910	0.02451	0.91341	0.58986	0.45801
	(0.00025)	(0.00035)	(0.00103)	(0.00571)	(0.00312)
1:27	0.02095	0.02736	0.92451	0.70815	0.56448
	(0.00031)	(0.00039)	(0.00080)	(0.00375)	(0.00195)
1:81	0.02203	0.02818	0.92925	0.74950	0.63787
	(0.00030)	(0.00044)	(0.00073)	(0.00335)	(0.00283)
Unchanged	0.02294	0.02651	0.93288	0.76105	0.71047
	(0.00037)	(0.00054)	(0.00073)	(0.00406)	(0.00169)

Standard deviations are below AUPRC scores in parentheses

Statistical Analysis

In order to make an informed decision on the results of the previous section, we apply statistical analysis in the form of ANOVA [78] and HSD [165] tests. The ANOVA tests tell us whether a factor has a significant effect on experimental outcomes. When the ANOVA test identifies that a factor has a significant impact, we can then perform an HSD test to rank the levels of the factor in terms of its impact on experimental outcomes. Here, the outcome is either an AUC score or an AUPRC score. For all statistical tests, we use a significance level of $\alpha = 0.01$.

The first outcome we perform analysis for is the AUC score. This is analysis of the variance in AUC scores for all classifiers, all datasets, and all levels of RUS. In the following

ANOVA tables, CLF indicates the classifier factor, RUS indicates the RUS factor, and Size indicates the number of instances in the dataset before RUS is applied. Please see Table 5.7 for the ANOVA test results. Since the Pr(>F), or p-values associated with each factor are practically zero, we conclude that all factors have a significant effect on experimental outcomes. Therefore, we can conduct HSD tests to rank the levels of each factor in terms of its impact on AUC score.

Table 5.7: ANOVA for RUS, CLF, and Size as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
RUS	5	0.06	0.01	14.21	*
CLF	4	16.97	4.24	4832.58	*
Size	2	4.70	2.35	2673.69	*
Residuals	4488	3.94	†		

* indicates value is less than 1×10^{-4}, †indicates value is less than 1×10^{-2}

HSD test results separate factors into groups such that levels of a factor that have a similar impact on performance are placed into the same group. The group that is associated with the highest value of the experimental outcome is labeled as group 'a', lower ranked groups are labeled with letters that follow in alphabetical order. If there is overlap in the performance associated with two groups, then overlapping groups will be labeled with letters in common. For example, in Table 5.8, the results for the 1:1 level of the RUS factor are placed in group 'ab', and results for the 1:81 level of the RUS factor are placed in group 'bc'. This implies the intersection of confidence intervals for AUC scores associated with these levels of RUS overlap.

The HSD test results in Table 5.8 show that applying RUS to induce class ratios of 1:9, 1:3, or 1:27 yields the best performance. This is important to bear in mind, since the HSD tests for the impact of RUS on AUPRC scores will show there is a negative impact on performance.

Table 5.8: HSD test groupings after ANOVA of AUC for the RUS factor

Group a consists of: 1:9, 1:3, 1:27
Group ab consists of: 1:81
Group bc consists of: 1:1
Group c consists of: Unchanged

The next HSD test we undertake is to determine which classifier yields the best performance. The results here are processed for all datasets, and all levels of RUS. Here we see ET and XGBoost yield the best performance.

Table 5.9: HSD test groupings after ANOVA of AUC for the CLF factor

Group a consists of: XGB-24, ET
Group b consists of: RF-GPU-32
Group c consists of: CB-16
Group d consists of: LGB

The final HSD test we can conduct is for the size factor. We have three datasets of varying size. The HSD test results do not reveal a trend on the impact of size. We see that the best AUC scores are associated with the medium size, Part B, dataset. However, the second-best AUC scores are associated with the small size, DMEPOS dataset. Finally, the lowest AUC scores are coupled with the largest, Part D dataset.

Table 5.10: HSD test groupings after ANOVA of AUC for the Size factor

Group a consists of: Medium
Group b consists of: Small
Group c consists of: Large

Here we begin a series of statistical tests similar to the previous section, only now the experimental outcome we are analyzing is AUPRC instead of AUC. As is the case with the previous analysis of AUC scores, results here are obtained by processing experimental outcome data over all factors: data set size, RUS and classifier, for the Part B, Part D, and DMEPOS data. The factors in Table 5.11 are the same as in Table 5.7. Similar to Table 5.7, the Pr(>F) values for all factors in Table 5.11 are practically zero, which means each factor has a significant impact on AUPRC scores.

Table 5.11: ANOVA for RUS, CLF, and Size as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
RUS	5	44.65	8.93	713.68	*
CLF	4	501.08	125.27	10011.73	*
Size	2	9.62	4.81	384.50	*
Residuals	4488	56.16	0.01		

* indicates value is less than 1×10^{-4}

Since all factors have a significant impact on performance, we can conduct a Tukey HSD test to rank the levels of the factors in terms of their impact on performance. Here we see a clear relationship between the class ratio and AUPRC scores. The HSD results in Table 5.12 show that models built with larger number of majority class instances in the training data are associated with higher AUPRC scores. Since the results here are for AUPRC scores averaged across all datasets and learners, they imply that in general, RUS to induce a class

ratio any larger than 1:81 yields worse performance in terms of AUPRC. This implies RUS is a viable data reduction technique, but only to induce the 1:81 class ratio.

Table 5.12: HSD test groupings after ANOVA of AUPRC for the RUS factor

Group a consists of: Unchanged, 1:81
Group b consists of: 1:27
Group c consists of: 1:9
Group d consists of: 1:3
Group e consists of: 1:1

Next we report the results of the HSD test to rank the classifier factor. In keeping with the previous results for performance in terms of AUC, ET and XGBoost are associated with the best performance. However, we see in Table 5.13 that for performance in terms of AUPRC, ET is in a class by itself. ET and XGBoost are grouped together for best performance in terms of AUC.

Table 5.13: HSD test groupings after ANOVA of AUPRC for the CLF factor

Group a consists of: ET
Group b consists of: XGB-24
Group c consists of: RF-GPU-32
Group d consists of: CB-16
Group e consists of: LGB

Interestingly, for performance in terms of AUPRC, we see there is a trend in the size of the dataset and performance in terms of AUPRC. The smallest dataset is associated with

the best performance in terms of AUPRC, as Table 5.14 reveals. Intuitively, the smallest dataset has the smallest negative class size. This could make it so there are fewer false positives to bring down the precision score. It is an interesting question for future work whether we might see less of an impact of RUS on performance in terms of AUPRC for small imbalanced datasets than large imbalanced datasets.

Table 5.14: HSD test groupings after ANOVA of AUPRC for the Size factor

Group a consists of: Small
Group b consists of: Medium
Group c consists of: Large

Conclusions from Case Studies [55, 58, 59]

We have presented a thorough review of experiments for Medicare fraud detection with the three highly imbalanced non-aggregated Medicare datasets. The three datasets, DMEPOS, Part B, and Part D, range in size from about 12 million to 175 million instances. To the best of our knowledge, [55, 58, 59], covered in this chapter are the first to contain a study on the latest versions of all three datasets . The original class ratio of the DMEPOS data is 0.0044, Part B data is 0.0019, and the Part D data is 0.0039. Our primary goal in compiling this study is to show that the AUC metric does not give a clear signal on the negative impact of RUS in the classification of highly imbalanced Big Data. RUS is a tempting technique for addressing class imbalance with Big Data since it lowers resource consumption. However, our results and statistical analyses show that applying RUS to change class ratios appears to have a positive impact on AUC scores. At the same time, for some classifiers, we see a significant drop in AUPRC scores as we apply RUS to make the majority class closer in size to the minority class. Moreover, we find that although other classifiers yield relatively high AUC scores, they yield relatively low AUPRC scores, regardless of the level of RUS used.

For highly imbalanced Big Data, we find that the size of the majority class overwhelms other terms in the calculation of the false positive rate, and therefore hides the detrimental effect of RUS. This in turn hides the effect of RUS in the calculation of AUC as well. Since the AUPRC metric involves recall, and not the false positive rate, the impact of RUS is easier to detect in AUPRC scores. Moreover, as Table 5.14 illustrates, the size of the majority class magnifies the impact on AUPRC scores. Therefore, the size of the imbalanced dataset is an important consideration when selecting a performance metric.

The choice of classifier has a significant effect on our experimental outcomes. XGBoost and ET are consistently the best classifiers, both for performance in terms of AUC and AUPRC. It is interesting to note that were we to select XGBoost and ET on the basis of their performance in terms of AUC, we would find they both give strong performance for all levels of RUS. However, the AUPRC scores would more accurately inform which classifier performs better. Overall, we find ET is the more robust of the two classifiers, since it is the least impacted by RUS. To the best of our knowledge, [55,58,59] are the first studies where ET is applied in the domain of Medicare fraud detection. All classifiers appear to do well when we look at AUC scores. However, when we look at AUPRC scores we see differences in classifier performance. XGBoost and Random Forest show improved performance as the class ratio reduces from 1:1 to its original highly imbalanced ratio. CatBoost and LightGBM yield relatively poor performance in terms of AUPRC. The ensemble technique for classifiers does not appear to have an effect on outcome, since LightGBM, CatBoost, and XGBoost are all GBDT implementations, and Random Forest and ET are applications of a bagging technique. Moreover, we see the performance of CatBoost and LightGBM, in terms of AUPRC, appears to diminish also as the size of the dataset increases. Overall, we see RUS has the smallest effect on ET. We conjecture that ET's random split selection makes it robust to the random deletion of instances of the majority class that we perform when doing RUS. Our results can be summarized as follows: in the classification of the highly imbalanced non-aggregated Part B, Part D, and DMEPOS Big Data, AUPRC shows

that RUS can have a detrimental effect on performance, whereas AUC does not show this detrimental effect, and ET's performance in terms of AUPRC is more robust to RUS than other learners. In terms of data reduction, for the non-aggregated Medicare Data, we conclude that is a viable data reduction technique for the ET classifier, and with a 1:81 class ratio.

5.4.5 One Class Sampling and Medicare Data

The research covered up to this point involves RUS for data reduction in the binary-class classification scenario. Now we move on to discuss data sampling in the one-class scenario. This research also addresses the issue of fraud within the Medicare program, which results in losses amounting to billions of dollars annually. By leveraging One-Class Classifiers (OCCs), we demonstrate a data reduction technique in the task of fraud detection utilizing the aggregated Medicare Part D and Part B insurance claims data [51, 53]. Our research introduces a novel data reduction technique that allows for the training of One-Class Gaussian Mixture Models (GMMs) on a significantly reduced subset of the full dataset. This approach maintains high performance in fraud identification while optimizing computational efficiency. We demonstrate that models trained on as little as 20 percent of the original data yield performance, in terms of AUC and AUPRC, comparable to models trained on the complete training data. Our findings suggest that it is feasible to efficiently classify highly imbalanced datasets, such as those encountered in Medicare fraud detection, using One-Class GMMs with a substantial data reduction. This research contributes to the field by presenting a scalable and effective methodology for fraud detection in large-scale healthcare datasets, showcasing the potential of Machine Learning in enhancing the integrity of public health insurance programs.

Preliminary experiments with the same datasets used in this study revealed that One-Class GMMs exhibit superior performance when compared to One-Class Support Vector Machines (SVMs). This significant difference in performance led us to postpone further

experimentation with One-Class SVMs, primarily due to their relatively poor results. Our decision to focus on One-Class GMMs is further supported by recent studies, which also report the superior efficacy of One-Class GMMs over One-Class SVMs [114, 115]. For those interested in a broader understanding of the applications of One-Class classifiers in Big Data application domains, we recommend the thorough review of the literature provided in [151]. Generative Adversarial Networks (GANs) have been employed to address class imbalance, however we find One-Class GMM also outperformed GANs in preliminary experiments [146, 147, 177].

In this section of the dissertation, we cover research on a data reduction technique to enhance the performance of One-Class Gaussian Mixture Models (GMMs) [100] in identifying fraudulent activities within Medicare Part D and Medicare Part B insurance claims data. This technique is designed to enable the training of One-Class GMMs on a significantly reduced subset of the full dataset, thereby optimizing computational efficiency without compromising the effectiveness of fraud detection. In the case study covered here, we report findings from experiments with the Medicare Part D data and the Medicare Part B data.

Our findings are pivotal in demonstrating that, with the datasets used in this study, it is feasible to train One-Class Gaussian Mixture Models (GMMs) on 80 percent less data without compromising performance, as measured by the AUC, and AUPRC metrics in the classification of the Medicare Part B and Part D datasets. This significant data reduction translates to accelerated training times on large datasets, which is particularly beneficial in the context of Big Data. The methodologies we propose highlight the potential of One-Class GMMs in efficiently classifying highly imbalanced datasets, such as those encountered in Medicare fraud detection. For further insights into existing techniques addressing class imbalance in Big Data, we refer readers to the comprehensive survey by Leevy *et al*. [124].

Related Works

A recent study involving Medicare Fraud detection is by Herland *et al* [75]. Similar to our study, they investigate Medicare fraud detection utilizing datasets derived from the CMS's publicly available data. Their research encompasses several datasets: Medicare Physician & Other Practitioners (Part B) [162] for the years 2012–2015, Medicare Part D Prescribers (Part D) [158] for the years 2013–2015, and the Medicare Durable Medical Equipment, Prosthetics, Orthotics, and Supplies (DMEPOS) [156] for the same period. Furthermore, Herland *et al*. construct an integrated dataset, termed the Combined dataset, through the amalgamation of the Part B, Part D, and DMEPOS datasets. Similar to the data used in our study, Herland *et al*. have high imbalance in all four datasets. In terms of model development, Herland *et al*. explore the efficacy of Logistic Regression, Random Forest, and Gradient Boosting classifiers across these datasets. Their findings indicate that models built with the Combined dataset, particularly Logistic Regression, yield the best performance in fraud detection. Our study, however, diverges from Herland *et al*.'s approach in several respects. Notably, their research does not incorporate OCCs, and it does not cover the sampling technique we propose here.

"Data Reduction to Improve the Performance of One-Class Classifiers on Highly Imbalanced Big Data", [51] by Hancock and Khoshgoftaar, takes the initial step of investigating the effect of applying sampling to highly imbalanced Medicare big data for use with One-Class GMM. We only used one dataset, the Medicare Part D data. We found that sampling as little as five percent of the original data yielded performance, in terms of AUC, that was not significantly different from using the entire dataset. By "significantly different" we mean that, at the $\alpha = 0.01$ significance level, there was no statistically significant difference in AUC scores of models built with the entire dataset, versus models built with a sample. For performance in terms of AUPRC, we found that models built with samples as small as 20 percent of the entire training data yielded AUPRC scores that were not significantly different from using the entire training data. In [53], which we report on here, we expand

the experiments performed in [51] to encompass another highly imbalanced Big Medicare Data dataset.

Our comprehensive literature review highlights a notable gap in extant studies. Among the few related studies we have identified, we found one with the notable limitation of the necessity for expert intervention in data sampling [98]. We found other studies which we posit do not conform to our more rigorous interpretation of one-class methodology [70]. Furthermore, we noted additional studies in the healthcare fraud detection application domain, but they do not investigate the use of OCCs in conjunction with sampling techniques. We have developed and applied a unique data reduction technique that is especially adept at managing the challenges posed by severely imbalanced Big Data. We tailor OCC methodologies to the nuanced demands of Big Data applications, marking a significant advancement in the field by expanding our previous study.

Results

Before delving into results, we explain notation used in Tables 5.15 through 5.18 in this section. We represent the sample size by the number that is the result of dividing the sample size by the size of the entire dataset. For example, in the first row of Table 5.15, the figure 0.05 means that the result pertains to experiments where the sample that is five percent of the size of the entire dataset is used.

The first set of results we wish to present are for One-Class GMM's performance in terms of AUPRC and AUC for various levels of the sampling factor applied to the Part D data. In Table 5.15, we observe that samples of sizes twenty and twenty-five percent of the entire training data yield performance that is higher than using the entire dataset.

Table 5.16 contains the AUC scores the One-Class GMMs yield when classifying the Part D data. The trend of similar or better scores with smaller sample sizes is also apparent in Table 5.16.

The second set of results we wish to present are for classification of the Part B data. The

results in Table 5.17 follow a pattern that is similar to what we see in the results for the Part D data. The AUPRC scores for fractions of sizes 20 and 25 percent of the data are higher than the AUPRC score One-Class GMM yields when the entire dataset is used. Moreover, the mean AUPRC scores of models built with the other sample sizes of five, ten, and 15 percent are also higher than the AUPRC score of the One-Class GMM built with the entire dataset.

Table 5.15: Mean AUPRC values by fraction for 10 iterations of five-fold cross validation, for classifying the Part D dataset

Classifier Fraction	GMM Sigmoid
0.05	0.1343
0.10	0.1405
0.15	0.1401
0.2	0.1442
0.25	0.1443
1.0	0.1437

Table 5.16: Mean AUC values by fraction for 10 iterations of five-fold cross validation for classifying the Part D dataset

Classifier	GMM Sigmoid
Fraction	
0.05	0.7280
0.10	0.7298
0.15	0.7285
0.2	0.7291
0.25	0.7275
1.0	0.7257

Table 5.17: Mean AUPRC values by fraction for 10 iterations of five-fold cross validation, for classifying the Part B dataset

Classifier	GMM Sigmoid
Fraction	
0.05	0.0019
0.10	0.0018
0.15	0.0017
0.2	0.0018
0.25	0.0018
1.0	0.0016

Finally, we present the mean AUC scores the One-Class GMMs yield when classifying the Part B data. The similarity of the AUC scores in Tables 5.16 and 5.18, and the discrepancies of AUPRC scores in Tables 5.15 and 5.17 highlight how AUPRC can be

a more informative metric in the classification of imbalanced big data. Both AUC and AUPRC involve the true positive rate. The key difference between the AUC and AUPRC metrics is that AUC involves the false positive rate, whereas AUPRC involves precision. For more details on why AUPRC is a more informative metric for analyzing classification results of imbalanced Big Data Please see the discussion on AUC versus AUPRC in the Chapter 4 on Methodology.

Table 5.18: Mean AUC values by fraction for 10 iterations of five-fold cross validation, for classifying the Part B Dataset

Classifier	GMM Sigmoid
Fraction	
0.05	0.7042
0.10	0.7214
0.15	0.7121
0.2	0.7031
0.25	0.7210
1.0	0.6178

The results in Tables 5.15 through 5.18 provide evidence that we can apply data sampling to One-Class GMM's training data, and One-Class GMM will yield performance similar to, or better than One-Class GMM trained on the entire dataset. This is because we find models trained on a fraction of the training data yield similar or better AUC and AUPRC scores than models trained with all the available training data.

Another result we wish to present is on the running time of One-Class GMM when trained on sampled data versus the entire dataset. The data presented in Tables 5.15 through 5.18 indicate favorable variation in AUC or AUPRC scores when smaller samples of training data are used. This observation underscores the advantage of training models with the least

amount of data required. The lower running times, coupled with the similar or better AUC and AUPRC scores yielded by models trained with a fraction of the training data, are what determines the advantage. Typically, such a practice results in shorter training times for most classifiers. For example, training the One-Class GMM with a 25 percent sample of the Part D data averaged around five minutes, compared to approximately thirteen minutes when using the full dataset. We found similar speed-up for classifying the part B data as well. This significant reduction in training time suggests that as dataset sizes increase, employing our sampling technique could become essential. Moreover, in scenarios where the dataset size precludes the use of all available data, our findings support the strategy of training GMM on a subset of the data.

The presentation of AUC, AUPRC, and the discussion of running times concludes our presentation of results. In the next section, we move on to statistical analysis of the AUC and AUPRC scores reported in this section.

Statistical Analysis

This section contains a statistical analysis of the experimental outcomes depicted in Tables 5.15 through 5.18. This analysis aims to conclusively identify the smallest fraction of data that is sufficient for classifying Part D or Part B data in model building, ensuring performance on par with or superior to One-Class GMM models trained on the entire dataset.

An initial step involves conducting an ANOVA test to evaluate the impact of training data size on the AUC scores for One-Class GMMs classifying Part D data. According to the results presented in Table 5.19, the proportion of majority class instances utilized in training the One-Class GMM does not significantly influence the AUC performance at the $\alpha = 0.01$ significance level. Consequently, it is feasible to train the One-Class GMM with as little as a five percent sample of the Part D training data, achieving AUC performance equivalent to that obtained using the full set of majority class instances in the training data.

Table 5.19: ANOVA for Percentage as a factor of One-Class GMM's Performance in terms of AUC in Classifying the Part D Dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Percentage	5	0.00	0.00	0.74	0.5912
Residuals	294	0.04	0.00		

Next, we report the results of analysis of the AUPRC scores the One-Class GMMs yield when classifying the Part D data. The results of the ANOVA test, as indicated in Table 5.20, reveal that the portion of majority class instances used in training the One-Class GMM significantly affects the AUPRC scores at an $\alpha = 0.01$ significance level. Consequently, to identify the optimal percentage of instances that leads to the best performance, we proceed to conduct an HSD test [165], which is essential for determining the most effective sample size for model training. This test categorizes factors into groups, assigning them alphabetical labels that indicate their rank. Factors designated as 'a' are correlated with the highest AUPRC scores, and this ranking proceeds in reverse order alphabetically with subsequent labels reflecting groups with decreasing AUPRC scores.

Table 5.20: ANOVA for Fraction as a factor of One-Class GMM's performance in terms of AUPRC in Classifying the Part D Dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Fraction	5	0.00	0.00	3.95	0.0017
Residuals	294	0.05	0.00		

The outcome of the HSD test, detailed in Table 5.21, reveals that utilizing merely 20 percent of the training data instances can achieve performance comparable to that achieved using the entire training dataset. Furthermore, it is observed that training the One-Class GMM with 25 percent of the data positions it first in Group 'a', indicating that this fraction

of data results in the highest mean AUPRC score for the model.

Table 5.21: HSD test groupings after ANOVA of AUPRC for the Fraction factor for the Part D Dataset

Group a consists of: 0.25, 0.20, 1.00
Group ab consists of: 0.10, 0.15
Group b consists of: 0.5

Our statistical analysis indicates that using only a fraction of the Part D dataset can result in models achieving performance comparable to those trained on the entire dataset, as measured by two different metrics. This finding is significant as it suggests the possibility of constructing models that require less time for training due to the reduced volume of training data. Additionally, conducting the analysis across two distinct metrics adds a layer of robustness to our study, reinforcing the reliability of our conclusions.

Now we move on to perform a similar statistical analysis for the results of classifying the Part B data. The next analysis we wish to present is the result of the ANOVA test for the impact of the sample size on the One-Class GMMs performance in terms of AUC. The result of the ANOVA test is reported in Table 5.22. The Pr(>F) value in the test result implies that the sample size has a statistically significant impact on AUC scores.

Table 5.22: ANOVA for Fraction as a factor of performance in terms of AUC in Classifying the Part B Dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Fraction	5	0.39	0.08	8.34	*
Residuals	294	2.73	0.01		

* indicates the value is less than 1×10^{-4}

Since the sample size has a statistically significant impact on the AUC score the One-Class GMMs yield in classifying the Part B data, we conduct an HSD test to group the levels of the sample size factor. The HSD test result in Table 5.23 is that any sample size yields a better AUC score than using the entire dataset.

Table 5.23: HSD test groupings after ANOVA of AUC for the Fraction factor for the Part B Dataset

Group a consists of: 0.10, 0.25, 0.15, 0.05, 0.2
Group b consists of: 1.0

Next, we move on to test for the impact of the sample size on the One-Class GMM's ability to classify the Part B data in terms of AUPRC scores. In order to determine whether the sample size has a statistically significant impact on AUPRC scores, we conduct an ANOVA test. The result of the ANOVA test is in table 5.24. The Pr(>F) value reported in Table 5.24 means that the sample size does not have a significant influence over the AUPRC scores in One-Class GMM's classification of the Part B data.

Table 5.24: ANOVA for Fraction as a factor of performance in terms of AUPRC in Classifying the Part B Dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Fraction	5	0.00	0.00	2.06	0.0699
Residuals	294	0.00	0.00		

Since the result of the ANOVA test implies that the sample size does not have a statistically significant impact on AUPRC scores, this ends our statistical analysis. In conclusion, we find that applying sampling to the test data that we use to train One-Class GMM to classify the Part D or Part B data either has no effect on AUC or AUPRC scores,

or it yields an improvement. The HSD result for classifying the Part D data in Table 5.21 is the most restrictive. It implies that we can train One-Class GMM on a sample of the training data as small as 20 percent of the original training data, and One-Class GMM will yield performance similar to using all the training data. The key take-away from the statistical analysis is proof that one can save time by training One-Class GMM on samples of the training data. It is acceptable to take advantage of the time savings because the performance will be equivalent to or better than using the entire dataset.

Conclusions from OCC studies

In [51,53] we evaluate a novel data reduction technique aimed at enhancing the performance of One-Class GMMs in the context of Medicare Fraud detection. To our knowledge, these are the first studies on data reduction methods specifically tailored to address the challenges posed by severely imbalanced Big Data in conjunction with OCCs. Our findings demonstrate that reducing the training dataset to as little as 20 percent of its original size does not significantly affect the performance of One-Class GMMs, as measured by both AUC and AUPRC scores. Notably, models built on 20 percent of the training data exhibit optimal performance. We make this claim based on the fact that, according to our statistical analysis, 20 percent is the smallest fraction of data that can be used to build models that yield performance that is not significantly different from models built with all the training data. Moreover, since the models are built with a fraction of the data, there is a reduction in training time. Our results show this statement holds for two highly imbalanced Big Data datasets: Medicare Part D and Medicare Part B. These conclusions are bolstered by statistical tests, affirming the relationship between the size of the training data and experimental outcomes.

Our results for the Part B and Part D datasets show that sampling may have a significant impact on experimental outcomes in terms of AUC scores, but not have a significant impact on AUPRC scores, or vice-versa. The absence of an impact is nevertheless positive news.

It implies we can train models with the smallest-sized sample of the training data used in experiments, resulting in the shortest training time, and still enjoy performance equivalent to training models with all the training data.

We would like to point out that our methodology can be used as a general technique in large scale experiments with One-Class GMMs to accelerate the pace of experimentation. One can apply our methodology to discover an optimal sample size of the training data to use, in terms of the metric one is interested in. In subsequent experiments One-Class GMMs may be trained on the same sample size of the training data, in less time, hence speeding up the pace of the subsequent experiments. Looking forward, we plan to extend the application of our data reduction approach for anomaly detection in a broader range of Big Data datasets.

5.5 CHAPTER SUMMARY

This chapter delves into the application and efficacy of sampling techniques for data reduction, specifically RUS and One-Class Sampling, in the application domain of Medicare fraud detection in highly imbalanced datasets. We elucidate the challenges posed by such datasets to conventional classification algorithms. We show data reduction has may have detrimental impact on BCCs. ET appears to be the most robust BCC when it comes to data reduction. We found we could apply RUS to reduce data to the 1:81 class ratio and maintain strong performance in terms of AUC and AUPRC.

The chapter introduces a novel data reduction technique tailored for one-class classification scenarios. We highlight its novelty in the context of Big Data and its application to Medicare fraud detection. This technique, distinct from RUS, involves discarding all elements of the minority class and sampling from the majority class, which is a departure from traditional sampling methodologies. Through a series of experiments, we show that reducing the training dataset to a fraction of its original size (as low as 20%) does not substantially compromise the performance of One-Class Gaussian Mixture Models (GMMs),

as evidenced by AUC and AUPRC scores. This finding underscores the potential for significant data reduction without sacrificing model accuracy, thus facilitating more efficient training processes and resource utilization. Overall, we find sampling as a data reduction technique to work better in OCC scenarios than BCC scenarios.

CHAPTER 6

FEATURE SELECTION TECHNIQUES

6.1 INTRODUCTION

We define feature selection as a data reduction technique where data attributes are removed with three motivations: 1) shorter model training time, 2) avoiding overfitting, and 3) improved performance. When a dataset has n attributes, removing one feature reduces the size of the data by a factor on the order of $\frac{1}{n}$. Hence, feature selection usually yields shorter model training times and protects against overfitting. When applying feature selection also results in similar or better performance than using the entire training data, we have a compelling reason to apply feature selection in subsequent work with the same dataset. Hence, feature selection is an interesting subject for researchers. Here we cover both binary-class classification (BCC) scenarios and one-class classification (OCC) scenarios. The ensemble feature selection techniques we cover are applicable in the BCC setting, since they leverage the built-in feature importance functionality of binary-class classifiers. SHapley Additive Explanations (SHAP) is applicable for data reduction by feature selection in the OCC and BCC setting [38]. Data reduction by feature selection has an additional benefit. Feature selection can be used to build more easily explainable models. Moreover, we show how the results of SHAP-based feature selection conducted with multiple models can be applied to do feature analysis.

6.2 ENSEMBLE FEATURE SELECTION

Our research group has published many studies on ensemble feature selection techniques which can be used for data reduction. The first studies starting with a technique that

we initially termed "Supervised Feature Selection" which we rename to "Voting Feature Selection" (VFS) in this dissertation to avoid confusion with an order-preserving technique, which is also referred to as "Supervised Feature Selection" (SFS) in our published work. Here, ensemble feature selection refers to the concept of combining two or more feature selection techniques to do feature selection as defined above.

6.2.1 Voting Feature Selection (VFS)

Here we describe the VFS data reduction technique for the purposes of contrast with SFS. VFS was designed by our research group, and is the predecessor to the more robust, order-preserving SFS method. In studies where VFS is used [119, 120, 129], VFS was implemented as an advanced preprocessing phase. The methodology for VFS is a two phase framework. The initial phase was characterized by the application of three distinct filter-based feature ranking methodologies: Information Gain Ratio, Information Gain, and Chi-squared (Chi^2) tests. Concurrently, feature ranking was also derived from the built-in feature importance list, ascertained through four supervised learning algorithms: CatBoost [141], XGBoost [28], LightGBM [101], and Random Forest [22]. This makes for a total of seven feature ranking strategies used in phase one.

Subsequently, in the second phase, the focus was on the extraction of the top a attributes as determined by their rankings. The selection of a value for a depends on the total number of features in the dataset. In our research we selected $a = 20$ since this was approximately half of the available features. The criterion for feature selection was predicated on the identification of attributes that were consistently ranked in the top a attributes for a minimum of n different feature ranking methodologies. In these studies, n took the values of 4 through 7. This selection criterion yielded four distinct datasets of selected features: 4-Agree, 5-Agree, 6-Agree, and 7-Agree.

One shortcoming of VFS is information loss. When we select features based on their membership in the top a elements of n rankings, we discard information about their position

in the ranking. This shortcoming is overcome by SFS.

6.2.2 Feature Popularity

One variation that VFS explored in our research is feature popularity [181, 183]. Feature popularity is another data reduction technique that involves VFS applied to similar datatsets. We include the discussion of Feature popularity since it may be applied to SFS as a data reduction technique in future work with multiply labeled datasets. Feature Popularity's utility extends to any domain characterized by datasets with multiple class labels.

Feature popularity is applied as follows: initially, we generate feature importance lists by applying VFS to k datasets. We then perform a second iteration of VFS. We select a number $n \leq k$ of feature importance lists that a feature must appear in, in order to be selected. This technique is interesting since it selects features that simultaneously predict multiple labels. Feature popularity could be applied in future work where SFS is applied to the datasets used in [181, 183]. To the best of our knowledge feature popularity is the first data reduction technique that leverages multiply labeled datasets. Since current work where feature popularity is applied depends on VFS, it succumbs to the same shortcoming as other work that relies on VFS. We lose the information on the order of features by their feature importance.

6.2.3 Supervised Feature Selection (SFS)

Here we describe the novel ensemble feature selection technique, SFS. A synopsis of our technique is that it is an efficient means to intelligently leverage the built-in feature importance functionality of multiple machine learning algorithms for data reduction. As we shall see, at the completion of SFS, it is still possible to have an ordering of features by feature importance, something which is not possible with VFS. Hence, SFS is a superior data reduction technique.

Methodology of the SFS technique

SFS begins with the selection of a collection of supervised machine learning algorithms. In the context of our technique for ensemble feature selection, we call a supervised machine learning algorithm a *ranker*. For our study, we employ six rankers: CatBoost, XGBoost, Random Forest, ET, LightGBM, and Decision Tree [23]. These algorithms are a convenient choice since they all maintain a built-in list of feature importance values during the model fitting phase of supervised machine learning. Each algorithm's implementation may have a unique method for calculating feature importance. For example, a Decision Tree implementation may assign a feature its importance value based on the change in Gini Impurity [69], or Shannon Entropy [153], that is obtained by splitting the training data into subsets based on some criteria related to the feature [118]. This is because a Decision Tree implementation could employ either Gini Impurity change or Shannon Entropy change as a splitting criterion during model training. Therefore, these statistics are readily available to return to the user as feature importance values. Once fitting is completed, one may access the ordered list of features, ranked according to their importance to the algorithm. Since we use six rankers, we have six ordered lists of features. We refer to an ordered list of features as a *ranking*.

The novelty of SFS is the method by which we combine the rankings. Our approach is to assign each feature the median value of its rank in each ranking. We use the median rank for several reasons. First, the median robust to outliers. Therefore, if one ranker assigns a rank r to a feature that is much different from the rank the other rankers assign, the median value will be the same if r were above or below ranks assigned by other rankers, regardless of how extreme it is. Therefore, using the median rank of a feature helps prevent a single ranker from exerting too much influence on the final rank assigned by the ensemble. Another reason we use the median value of a feature's rank is that it preserves information. If all rankers agree that a feature has a particular ranking, then the median rank will also be that value. This is an advantage over the majority rules approach to combining rankings in VFS,

where we select features based on the number of rankers that agree that the feature is among the highest n ranked features. Such an approach loses the relative position of features in the final ranking. Since selecting the median rank of a feature can preserve information about the ranks assigned to it, the outcome leads to more explainable models. Moreover, in preliminary experiments with the same learners and datasets, we found SFS yielded higher AUPRC scores than VFS. We conjecture this is due to the general robustness of median values to outliers.

Assigning each feature the median value of its positions in all the rankings results in an ordered list of features. In order to complete the feature selection process, one decides on a number of features to retain by selecting the first n features in the ordered list, where n is an integer between one and the number of features in the dataset. Algorithm 1 below is a

formal description of our ensemble feature selection technique.

input :

- D // labeled dataset
- $F^j, j = 1, \ldots, m,$ // F features of D
- $y^j, j = 1, \ldots, m,$ // y labels of D
- $\omega_1 \ldots \omega_k$ // a set of k supervised machine learning algorithms that maintian feature importance lists (rankers)

output:

$\mathbb{R}$; // Ensemble output, a ranking of features F^j of dataset D

R $\leftarrow \emptyset$ **for** $i = 1, \ldots, k$ **do**
  Fit ω_i to F^j, y^j;
  $\mathbf{R}_i \leftarrow$ feature ranking list from ranker ω_i;
  $\mathbf{R} \leftarrow \mathbf{R} \cup \mathbf{R}_i$;
end
for $F^j \in F$ **do**
  $l \leftarrow \emptyset$; // list of feature rankings ;
  for $\mathbf{R}_i \in \mathbf{R}$ **do**
    $l \leftarrow \mathbf{R}_i(F^j)$; // Get a list of all the ranks of feature F^j
  end
  $\mathbb{R}(F^j) \leftarrow$ median(l) ; // The rank of feature F^j is the median of all of its rank values in **R**.
end
return $\mathbb{R}$

Algorithm 1: Ensemble Supervised Feature Selection Technique

We choose to make the return value of Algorithm 1 the ordered list of features $\mathbb{R}$ to emphasize that one may then use first n elements of $\mathbb{R}$ to select n features. We do this to make it clear that our ensemble feature selection technique has an application for building explainable models. We recommend using statistical analysis such as Analysis of Variance (ANOVA) [78] tests, and Tukey's Honestly Significant Difference (HSD) [165] tests, to find the minimum value of the number of features n that we can build a model with, that yields

performance similar to, or better than using all features. Then this minimal subset is the one we use to explain a model's behavior.

We refer to the output of the feature selection technique by the number of features selected. We define a "feature set" as a group of features identified by a feature selection technique.

Studies Involving SFS

We employ SFS in our most recent work. In "Enhancing credit card fraud detection through a novel ensemble feature selection technique" [172], we apply SFS to the credit card fraud detection application domain. Classifiers are employed to classify the Credit Card Fraud data discussed in Chapter two. In this study, we investigate two of feature ranking methodologies: one comprising an ensemble of twelve Threshold-Based Feature Selection (TBFS) techniques [168], and SFS techniques. As a control, we explore scenarios where all available features are utilized without selection. In [172] we employ two Decision Tree-based classifiers, specifically CatBoost and XGBoost. The performance of these models is assessed through Area Under the Receiver Operating Characteristic (AUC) [19] and Area Under the Precision Recall Curve (AUPRC) [20]. Given that AUPRC offers a more precise quantification of false positive rates, particularly in the context of datasets characterized by high levels of imbalance, it is a suitable metric for model evaluation. Please see Chapter four on methodology, and Chapter five on sampling techniques for an in-depth analysis on the advantages of using AUPRC over AUC for the evaluation of the impact of data reduction techniques in the classification of highly imbalanced Big Data. The findings resulting from experiments conducted in [172] reveal that the combined use of SFS techniques and the incorporation of all features yields performance that is comparable, if not superior, to that achieved through the ensemble of TBFS techniques.

In [172], ANOVA Tests for the impact of the feature selection technique on classification outcomes were conducted. The feature selection technique did not have a significant impact

on AUC scores. This means that one is free to employ the feature selection technique for data reduction that is most convenient, since there is no significant effect on experimental outcomes. However, an ANOVA test for the impact of the feature selection technique on AUPRC scores indicates the feature selection technique has a significant effect on experimental outcomes. The subsequent HSD test based on the result of the ANOVA test indicates that SFS is associated with higher AUPRC scores. Therefore, the conclusion relevant to data reduction techniques we can make from results presented in [172] is that SFS is the preferred data reduction technique for classifying the highly imbalanced Credit Card Fraud data. Therefore, future work should seek to confirm the result holds for the classification other highly imbalanced datasets.

The second, recent work in data reduction where SFS is employed is "Explainable machine learning models for Medicare fraud detection" [41]. We apply our feature selection technique to build machine learning models for the automated detection of Medicare insurance fraud. The principal sources of data for this investigation is are the aggregated Medicare Part B, and Medicare Part D datasets discussed in Chapter two. Not only is the Medicare data highly dimensional, in the sense that it has many attributes, but also it contains many records, since on the order of millions of records are added annually. An effective feature selection technique is therefore highly desirable, since that reduces the overall size of the data that must be processed.

The learners we utilize in [41] are five well-known, open-source ensemble learning methods for our classification tasks, and one linear algorithm. The ensemble algorithms are XGBoost [28], LightGBM [101], Extremely Randomized Trees (ET) [35], Random Forest [22], and CatBoost [141]. The linear algorithm is Logistic Regression [113]. For feature selection, we use the five ensemble methods, as well as Decision Tree [23].

Hyperparameter Settings

Since learners are employed in SFS, we feel it is important to mention them here. Since our focus in this dissertation is data reduction techniques, and SFS is a data reduction technique, hyperparameter settings are important to report. Previous research shows hyperparameter settings have a statistically significant impact on outcomes of experiments with imbalanced data [47, 49, 60]. In our experiments, we have adopted the identical hyperparameter configurations that were used in the tests detailed in [54] for the classifiers. Preliminary trials for the aforementioned study indicated that these hyperparameter configurations exhibit robust performance in classifying highly imbalanced data, without showing tendencies of overfitting when determining the best classification thresholds based on multiple metrics. These metrics include f-Measure [36], geometric mean of true positive rate and true negative rate [112], Matthews Correlation Coefficient (MCC) [29], and Precision [69].

Table 6.1 includes hyperparameter settings that were adjusted from their default values. Besides the settings outlined in Table 6.1, we have also fixed random number generator seeds for all classifiers to ensure the reproducibility of the results. All remaining settings are kept at their default values. Furthermore, we have not made any changes to the hyperparameter settings for Logistic Regression and Decision Tree.

Table 6.1: Hyperparameter settings used in experiments

Classifier	Parameter Name	Parameter Setting
CatBoost	task_type	'GPU'*
	max_ctr_complexity	1
	max_depth	5
ET	max_depth	8
XGBoost	max_depth	3
	tree_method	'gpu_hist'*
LightGBM	max_depth	4
Random Forest	max_depth	4

*Setting selects Graphics Processing Unit (GPU) implementation of the classifier

This concludes the discussion of the seven types of machine learning algorithms used in [41]. Most of the models discussed are used in both feature selection, and classification. These are Random Forest, ET, XGBoost, CatBoost, and LightGBM. We use Logistic Regression for classification only, and we use Decision Tree for feature selection, only. Next we briefly review the experimental methodology employed in the study.

Results of Experiments where SFS is Used

For every pair of classifier and feature set, we conduct five-fold cross validation [174]. This yields five Area Under the Receiver Operating Characteristic Curve (AUC) [19] and five Area Under the Precision Recall Curve (AUPRC) [20] values, which we record as experimental outcomes. Since we are dealing with imbalanced data, we consider AUPRC to be a more robust metric when it comes to evaluating the performance of classifiers. However, we include AUC as well, since it is a secondary, threshold-agnostic classification

metric. Since we have six feature sets of Part B data, and six classifiers, we have a total of 1,800 AUC and AUPRC values in the set of experimental outcomes. For experiments with the Part D data, we have ten feature sets, and therefore, 3,000 AUC and AUPRC values in the set of experimental outcomes. This concludes the summary of the methodology we employed to use SFS as a data reduction technique in [41]. Next we provide results of experiments carried out during the study.

Our primary metric for gauging performance is AUPRC. In their study [24], Calvert and Khoshgoftaar show that AUC on its own may be a misleading metric to gauge experimental outcomes, due to the fact that it does not align with results in terms of several other metrics in their study. Here we present experimental outcomes in tabular form. Tables 6.2 and 6.3 contain the mean AUPRC scores for each combination of classifier and feature set. For each combination of classifier and feature set, we perform ten iterations of five-fold cross validation. Therefore, each AUPRC score is the mean value of 50 recorded AUPRC scores.

Table 6.2: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for classifying the Medicare Part D data (Part 1)

Features Classifier	7a	7b	8	9	10
CatBoost	0.7575	0.7582	0.7570	0.7558	0.7585
ET	0.5941	0.5171	0.5585	0.6006	0.5878
LightGBM	0.4548	0.4533	0.4689	0.5116	0.5529
Logistic Regression	0.3468	0.3368	0.3497	0.3482	0.3537
Random Forest	0.5873	0.4937	0.5927	0.5393	0.5903
XGBoost	0.7533	0.7539	0.7533	0.7514	0.7571

Table 6.3: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for classifying the Medicare Part D data (Part 2)

Features Classifier	15	20	25	30	82
CatBoost	0.8016	0.7953	0.7962	0.7949	0.7797
ET	0.4954	0.4647	0.4605	0.4391	0.3275
LightGBM	0.4447	0.4661	0.4603	0.4841	0.4982
Logistic Regression	0.3669	0.3536	0.3519	0.2939	0.3047
Random Forest	0.6097	0.5398	0.5519	0.5249	0.2429
XGBoost	0.7889	0.7448	0.7589	0.7548	0.7376

Tables 6.4 and 6.5 are similar to Tables 6.2 and 6.3, but they contain AUC scores instead of AUPRC scores. In our opinion, it is more difficult to see a trend in the data in Tables 6.4 and 6.5 than Tables 6.2 and 6.3. Furthermore, we conjecture the absence of the trend is due to the shortcomings of the AUC metric for classifying large imbalanced datasets. This shortcoming is discussed in Chapter four on methodology where we provide a detailed discussion of the AUC and AUPRC metrics. The AUPRC metric does not suffer this drawback. Therefore, the AUPRC metric may reveal the effect of a factor in a set of experiments on the classification of imbalanced Big Data. The visibility of the trend in AUPRC scores in Tables 6.2 and 6.3 is a testament to the advantage of using AUPRC to evaluate results in the classification of imbalanced Big Data.

Table 6.4: Mean AUC values by classifier and number of features for ten iterations of five-fold cross validation, for classifying the Medicare Part D data (Part 1)

Features Classifier	7a	7b	8	9	10
CatBoost	0.9223	0.9231	0.9228	0.9293	0.9383
ET	0.8541	0.8415	0.8448	0.8614	0.8574
LightGBM	0.7541	0.7449	0.7730	0.7904	0.8298
Logistic Regression	0.8850	0.8698	0.8816	0.8832	0.8869
Random Forest	0.8340	0.8222	0.8384	0.8244	0.8332
XGBoost	0.9276	0.9282	0.9278	0.9346	0.9408

Table 6.5: Mean AUC values by classifier and number of features for ten iterations of five-fold cross validation, for classifying the Medicare Part D data (Part 2)

Features Classifier	15	20	25	30	82
CatBoost	0.9436	0.9560	0.9567	0.9588	0.9587
ET	0.8294	0.8323	0.8352	0.8429	0.8116
LightGBM	0.7505	0.7929	0.7913	0.8311	0.8455
Logistic Regression	0.9006	0.9143	0.9133	0.8620	0.8536
Random Forest	0.8315	0.8288	0.8316	0.8496	0.7909
XGBoost	0.9472	0.9390	0.9447	0.9449	0.9426

Table 7.6 is similar to Tables 6.2 and 6.3, but for the Part B data. We detect a trend in Table 7.6, similar to the trend we see in Tables 6.2 and 6.3. AUPRC scores are generally higher for models built with fewer features than for models built with the maximum number

of features.

Table 6.6: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for classifying the Medicare Part B data

Features Classifier	10	15	20	25	30	80
CatBoost	0.6581	0.6792	0.7069	0.7009	0.7016	0.6817
ET	0.0400	0.0462	0.0443	0.0524	0.0424	0.0433
LightGBM	0.3939	0.3830	0.4261	0.4589	0.4293	0.4146
Logistic Regression	0.0093	0.0326	0.0338	0.0065	0.0064	0.0103
Random Forest	0.4356	0.3990	0.3736	0.3800	0.3395	0.2462
XGBoost	0.6611	0.6715	0.6995	0.6956	0.6955	0.6886

Also, as the case with the Part D Data, the AUC scores in Table 6.7 do not show as clear of a trend in the effect of features used. We find it difficult to distinguish which feature set out of 25, 30, and 80 yields the best performance. In our opinion, the results in Tables 6.4, 6.5, and 6.7 show that AUC is a less suitable metric for evaluating the classification of imbalanced Big Data, since trends are more difficult to detect. Put another way, we find that the results in terms of AUC are ambiguous.

Table 6.7: Mean AUC values by classifier and number of features for ten iterations of five-fold cross validation, for classifying the Medicare Part B data

Features Classifier	10	15	20	25	30	80
CatBoost	0.9346	0.9409	0.9493	0.9537	0.9539	0.9569
ET	0.7907	0.8080	0.8122	0.8255	0.8214	0.8409
LightGBM	0.8300	0.8176	0.8412	0.8679	0.8643	0.8477
Logistic Regression	0.8349	0.8207	0.8240	0.7874	0.7896	0.8166
Random Forest	0.8096	0.8245	0.8374	0.8525	0.8476	0.8643
XGBoost	0.9375	0.9399	0.9493	0.9521	0.9529	0.9561

Statistical Analysis

Here we move on to use statistical methods to process the data presented in the results in a way that allows us to make definitive conclusions as to the effect of the feature selection, and classifier factors in our experiments.

Two factor ANOVA for Feature Selection Experiments with Part D Data Analysis of Results in Terms of AUPRC

The first statistical method we use is an ANOVA test to determine whether the classifier and number of features factors have a significant impact on experiments involving the Part D data. The result of the ANOVA test is in Table 6.8. The Pr(> F), or p-values, in Table 6.8 are practically 0, so we conclude that both factors have a significant effect. In the ANOVA tables that follow, the term "Features" indicates the treatment of the number of features in the dataset as a factor in the experimental design. For example, in Table 6.8, there are ten possible values for the number of features used in the datasets, therefore, nine degrees of freedom (Df) for the "Features" factor.

Table 6.8: ANOVA for Features and Classifier as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	9	2.97	0.33	47.15	*
Classifier	5	71.64	14.33	2050.15	*
Residuals	2985	20.86	0.01		

indicates the value is less than 1×10^{-4}

Since we conclude that both the classifier and number of features have a have significant effect on experimental outcomes, we conduct an HSD test to rank the levels of the factors in terms of their impact. The result of the HSD test is the assignment of levels of the factor to groups. The alphabetical order of the group label corresponds to the level of performance, or rank, of the group.

Table 6.9 contains the result of the HSD test that we use to rank the impact of feature selection. Averaged across the performance of all the classifiers, 10 features yields the best performance. In Table 6.9, group 'a' is associated with the highest AUPRC scores, group 'ab' with scores in a range overlapping groups 'a' and 'ab', group 'ab' is associated with scores overlapping both groups 'a' and 'bc', and so on. Put another way, groups in the HSD test result correspond with their rank in alphabetical order.

Table 6.9: HSD test groupings after ANOVA of AUPRC for the Features factor

Group a consists of: 10
Group ab consists of: 15, 9, 7a, 8
Group bc consists of: 25, 20
Group c consists of: 7b, 30
Group d consists of: 82

Table 6.10 contains the HSD result for the effect of the choice classifier on AUPRC scores in our experiments with the Part D data. The pattern apparent in the test result is that the Gradient Boosted Decision Tree algorithms CatBoost and XGBoost do the best, followed by the Bagging algorithms, Random Forest and ET, followed by the linear method of Logistic Regression. The rank of LightGBM is an exception to the pattern.

Table 6.10: HSD test groupings after ANOVA of AUPRC for the Classifier factor

Group a consists of: CatBoost
Group b consists of: XGBoost
Group c consists of: Random Forest
Group d consists of: ET
Group e consists of: LightGBM
Group f consists of: Logistic Regression

Two factor ANOVA for Feature Selection Experiments with Part D data Analysis of Results in Terms of AUC

Next, we include ANOVA and HSD test results similar to those in Tables 6.8, 6.9, and 6.10. The difference is that, for the results in Tables 6.11, 6.12, and 6.13, we use the AUC scores recorded in experimental outcomes as opposed to the AUPRC scores. The ANOVA test result recorded in Table 6.11 shows that the Pr(>F) values are practically zero, which means that both the classifier and the number of features used have a significant impact on the AUC scores we record as experimental outcomes.

Table 6.11: ANOVA for Features and Classifier as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	9	0.21	0.02	10.43	*
Classifier	5	9.38	1.88	833.75	*
Residuals	2985	6.71	0.00		

** indicates the value is less than 1×10^{-4}

Although the ANOVA test result in Table 6.11 implies that both the classifier and number of features have a significant impact on experimental outcomes, the HSD test result in Table 6.12 accentuates the difficulty one may encounter when using AUC to gauge the performance of models when working with imbalanced Big Data. It is more difficult to identify the best choice since all factors are members of some interval that overlaps with other intervals.

Table 6.12: HSD test groupings after ANOVA of AUC for the Features factor

Group a consists of: 30
Group ab consists of: 10
Group abc consists of: 25, 20
Group abcd consists of: 9
Group bcde consists of: 82, 15
Group cde consists of: 8
Group de consists of: 7a
Group e consists of: 7b

The HSD results for the effect of the choice of classifier on AUC scores are more

interpretable. However, they serve to further illustrate difficulties one may encounter when using AUC to gauge the performance of a model. Here we see that the more biased Logistic Regression model is ranked above Random Forest and ET. The HSD test result for performance of the classifier in terms of AUPRC in Table 6.10 disagrees with this ranking. We speculate this is due to class imbalance. In particular, we conjecture that the higher rank of Logistic Regression in Table 6.13 versus Table 6.10 is due to the overwhelming number of instances in the negative class in the Part D data, and how the false positive rate is used to calculate the AUC score versus how Precision is used to calculate the AUPRC score.

Table 6.13: HSD test groupings after ANOVA of AUC for the Classifier factor

Group a consists of: CatBoost, XGBoost
Group b consists of: Logistic Regression
Group c consists of: ET
Group d consists of: Random Forest
Group e consists of: LightGBM

Two factor ANOVA for Supervised Feature Selection Experiments with Part B data Analysis of Results in Terms of AUPRC

Now we move on to present a statistical analysis of results of experiments conducted with the Part B data. The result of the ANOVA test in Table 6.14 shows that both the classifier and the number of features have a significant impact on AUPRC scores. Therefore, we conduct further HSD tests to rank levels of the factors according to their impact on AUPRC scores.

Table 6.14: ANOVA for Features and Classifier as factors of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	0.24	0.05	13.16	*
Classifier	5	130.10	26.02	7242.04	*
Residuals	1789	6.43	0.00		

* indicates the value is less than 1×10^{-4}

The HSD test result in Table 6.15 indicates a different effect of supervised feature selection than what we found previously in the Part D data. In Table 6.15, it is clear that applying supervised feature selection yields better performance in terms of AUPRC, than not using supervised feature selection. However, the number of features used is not significant. Therefore, we are free to select the number of features that is most convenient. Since feature set 10 is the smallest, in terms of the size of the data, we would select that. This is because in our experience machine learning algorithms run to completion more quickly with smaller datasets.

Table 6.15: HSD test groupings after ANOVA of AUPRC for the Features factor

Group a consists of: 25, 20, 30, 15, 10
Group b consists of: 80

The trend of classifier performance we observed for the Part B data in Table 6.16 is slightly more clear in the outcome of the HSD test recorded in Table 6.12, since LightGBM is ranked directly under CatBoost and XGBoost. Thus, the ranking in Table 6.16 shows that the Gradient Boosted Decision Tree algorithms CatBoost, LightGBM, and XGBoost yield the best performance, followed by the Bagging technique algorithms Random Forest and ET, and finally the linear technique, Logistic Regression.

Table 6.16: HSD test groupings after ANOVA of AUPRC for the Classifier factor

Group a consists of: CatBoost, XGBoost
Group b consists of: LightGBM
Group c consists of: Random Forest
Group d consists of: ET
Group e consists of: Logistic Regression

Two factor ANOVA for Supervised Feature Selection Experiments with Part B data Analysis of Results in Terms of AUC

Just as we do for the Part D data, here we conduct an analysis of the performance of the AUC score results of experiments involving the Part B data. As in the other cases, the outcome of the ANOVA test documented in Table 6.17, for the significance of the effect of the classifier and number of features factors on experimental outcomes, shows that both factors have a significant impact. Therefore, we do additional HSD tests to rank the factors in terms of their impact.

Table 6.17: ANOVA for Features and Classifier as factors of performance in terms of AUC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	0.13	0.03	23.51	*
Classifier	5	5.99	1.20	1111.70	*
Residuals	1789	1.93	0.00		

* indicates the value is less than 1×10^{-4}

The HSD test result in Table 6.18 highlights the misleading nature of the AUC score. The outcome of the HSD test implies that feature sets 25, 30, and 80 all yield overlapping

AUC scores. This is true because the AUC score involves the false positive rate, as opposed to the AUPRC score, which involves precision, and therefore the fraction of the positive class correctly identified, instead. We conjecture that for many threshold values, models built with feature set 80 have lower false positive rates, and lower precision scores, and the differences in rankings of feature sets is evidence of that.

Table 6.18: HSD test groupings after ANOVA of AUC for the Features factor

Group a consists of: 80
Group ab consists of: 25, 30
Group b consists of: 20
Group c consists of: 15, 10

The HSD results in Table 6.19 are a case where we find AUC not to be so misleading. The results align with the trend we see in other HSD results. The trend is that the GBDT algorithms are ranked highest, followed by the Bagging methods, and finally the linear technique.

Table 6.19: HSD test groupings after ANOVA of AUC for the Classifier factor

Group a consists of: CatBoost, XGBoost
Group b consists of: LightGBM, Random Forest
Group c consists of: ET, Logistic Regression

Conclusions from Studies Involving SFS

The results documented in [172] show SFS outperforms TBFS as a data reduction technique. Results from [41] indicate that SFS is a novel feature selection technique which can be employed to significantly decrease the size of a highly imbalanced Big Data dataset without necessarily compromising classification performance. Statistical analysis is essential in

order to make the determination as to what the minimum number of features are that one can eliminate from a dataset and build models that yield acceptable performance. Therefore, we would like to point out that one should employ statistical tests similar to the tests we conduct as a part of the data reduction technique.

In [41], we employ an ensemble supervised feature selection technique to rank features of two Big Data datasets. The datasets are derived from Medicare Part D and Medicare Part B insurance claims data, and labeled with data from the LEIE. They qualify as Big Data, and they are highly imbalanced. Therefore, our results demonstrate that our feature selection technique is a viable technique that can work for reducing the dimensionality of other highly imbalanced Big Data for supervised machine learning tasks. Taking our results for the Part D dataset as an example, our feature selection technique reduces a dataset from 82 to ten features. Yet, models built on the reduced set of features yield classification results that are significantly better than using all 82 features.

Another important advantage of SFS is its running time. Our technique runs in the time that it takes to fit a collection of six classifiers to the dataset. If the fitting process for each of the classifiers is run in parallel, then the running time of the feature selection technique is bounded from above by the time it takes to fit one classifier. This is a far better running time than the brute-force approach of trying all possible combinations of features. The brute-force approach requires exponential resources. Since our technique reduces the size of the dataset, this also improves the running time of many learners that one can fit to it, to train models for Medicare Fraud detection. Moreover, the reduction in size of the dataset also helps control overfitting. We conjecture this is why we observe better performance in terms of AUPRC for models built with fewer features

Explainable models are an additional result of our feature selection technique. When one can show that models built with a reduced number of features yield performance equivalent to using all features, one can make the case that the reduced number of features contains all the information that the algorithm requires in order to successfully carry out the machine

learning task. When fewer features are required, it is easier to understand and explain what sort of information a model uses. Future work should consider applying the techniques discussed here to other Big Data in other application domains.

6.3 SHAP

Another important feature selection technique we can use for data reduction is SHapley Additive exPlanations (SHAP) [134]. The learners used in our application of SFS are BCCs; hence a binary-class setting is implicit in [41, 172]. SHAP is applicable in both OCC and BCC settings. If instances of both classes are available so that BCCs are applicable, techniques such as SFS may outperform SHAP [171]. Therefore, it deserves attention, since it is a versatile data reduction technique.

6.3.1 SHAP Methodology

Here we explain how SHAP may be applied as a data reduction technique. We use SHAP in conjunction with the Machine Learning algorithms to determine feature importance. Lundberg and Lee introduced SHAP (SHapley Additive exPlanations) in their 2017 paper, aiming to bridge the gap between simplicity and accuracy in model predictions [134]. They sought to provide an interpretable method, using a game-theoretic approach from Shapley's 1953 work for determining individual contributions in a coalition. Applying this to machine learning, they used SHAP to assess feature importance, showing that many existing methods fall under a broader class that includes their technique.

This class of methods builds secondary models to determine feature importance for specific input values of the main model. SHAP, in particular, stands out for its three properties: local accuracy (the secondary model's output matches the main model's output for the same input), missingness (a feature's absence in the original dataset means it doesn't influence the secondary model's approximation), and consistency (the effect of a feature on the secondary models should mirror its impact on the main models). Lundberg and Lee

demonstrated that SHAP is unique in the class of methods to which it belongs; it is the only technique with these three properties.

In their paper, Lundberg and Lee also discuss several SHAP variants, with Kernel SHAP being the primary model-agnostic version, applicable to any model. Kernel SHAP is a technique to calculate how much each attribute of a particular instance contributes to the model's output value for that instance. Thus, SHAP values may be interpreted as representing feature importance. SHAP values, can be either positive or negative and are interpreted based on their absolute magnitude. Averaging these absolute SHAP values across multiple instances is a technique for calculating feature importance. Moreover, we apply SHAP as a stand-alone library which provides an object for calculating SHAP values. SHAP is passed a sample of data and a trained model, and with these, SHAP calculates feature importance.

SHAP is available as an open-source Python package, providing tools for visualizing feature importance. These tools include plots showing attribute contributions to model output, the impact of input changes on outputs, and local importance plots for individual instances. This availability and theoretical backing make SHAP a valuable, model-agnostic resource for understanding and explaining feature importance in machine learning models.

Since SHAP is used in conjunction with a classifier, for each classifier, we first train a model on all the viable attributes of the dataset. Then we apply the SHAP kernel explainer to the trained model and a sample of the data. We then call the kernel explainer's `shap_values` function, passing it another sample of the data. The output of the `shap_values` function is a list of each attribute and value that indicates how much the attribute contributes to the output value of the classifier, for each instance in the sample passed to `shap_values`. These values are known as the SHAP values of the attributes. SHAP values may be positive or negative, and their magnitude indicates the effect the attribute has on the model's output value for a particular instance of the dataset. Therefore, the absolute value of a SHAP value is an indicator of the effect of an attribute on the model's output. We then use the

mean SHAP value of every attribute to put an order on the attributes of the data. In the SHAP library documentation, Lundberg, who co-authored the original paper on SHAP, notes his method of calculating attribute importance is to use the average of the absolute SHAP scores for each feature.[1] We find that this technique for computing the attributes' importance places different importance on data attributes depending on the classifier we use. This aligns with the expectation that different models will leverage different features to learn how to classify a dataset. Note that in one-class and unsupervised learning settings, labels are not required for training the model, or the calculation of SHAP feature importance. For a study on unsupervised techniques for treating fraud detection datasets, please see [106]. Hence, SHAP is a data reduction technique that is applicable in more scenarios than SFS.

6.3.2 Applications of SHAP in Data Reduction

In "Feature selection strategies: A comparative analysis of shap-value and importance-based methods" [173], we present a comparison between SHAP (SHapley Additive exPlanations) values, and the inherent feature importance rankings provided by models. The investigation employs a systematic approach to discern the most influential features as determined by these two methods for data reduction. Subsequently, the efficacy of models built with reduced data is assessed. The research conducted using five classifiers: XGBoost, Decision Tree, CatBoost, Extremely Randomized Trees, and Random Forest. The effectiveness of the feature selection techniques is reported in terms of AUPRC. The study is carried out on the Credit Card Fraud Detection dataset, discussed in Chapter two. Each learner used in the study has a built-in feature importance function. Therefore, features were ranked by their SHAP feature importance, and their built-in feature importance. Put another way, we obtained two feature importance lists for each learner. Next models were built with the top 3, 5, 7, 10, and 15 features for each method.

The results reported in [173], corroborated by statistical analyses, reveal a consistent

[1] https://github.com/shap/shap/issues/632#issuecomment-503778940

superiority in performance of models that employ feature selection based on inherent importance values over those that rely on SHAP values. This holds across different classifiers and varying sizes of feature subsets. Notably, the research advocates for the preference of the model's built-in feature importance ranking as the principal method for feature selection. Consequently, the adoption of built-in feature importance rankings is posited as a more efficient and pragmatic strategy. However, the findings in [173] are only applicable for data reduction efforts in scenarios where classifiers with built-in feature importance functionality. In scenarios where built-in feature importance is not available, SHAP remains a workable approach.

SHAP for Data Reduction with One Class Classifiers

In "A model-agnostic feature selection technique to improve the performance of one-class classifiers", [39] we explore SHAP (SHapley Additive exPlanations) for data reduction with one-class classifiers on a credit card fraud dataset. We apply SHAP to select key features and evaluate One-Class Gaussian mixture models and One-Class Support Vector Machines. Statistical analysis tests show Gaussian mixture models built with SHAP-selected features perform significantly better than Gaussian mixture models built without feature selection. To the best of our knowledge, we are the first to show the benefit of SHAP-based feature selection to One-Class Gaussian Mixture Models (GMMs) [77]. Moreover, we show that robust performance with features of the full dataset may be a prerequisite in order for SHAP feature selection to impart further gains. Our results provide novel evidence that SHAP can identify informative features for one-class classifiers.

To the best of our knowledge, in [39] we are the first to apply SHAP [134] in a feature selection technique for One-Class Classifiers (OCCs). Here, we provide a solution to the issue Bauder *et al.* demonstrate with diminishing performance due to class rarity [15], since our approach shows an improvement in performance via a feature selection technique. We show that the feature selection technique improves the performance of One-Class GMM

in the credit card fraud detection application domain. Moreover, the results we report for One-Class SVM show that some degree of robustness in models built with all features is necessary for SHAP feature selection to be effective. Two recent studies which show similar results of One-Class GMMs out-performing One-Class SVMs are in alignment with this finding [115], [114].

The results of our experiments in [39] show that our novel technique of selecting features based on SHAP feature importance for OCCs is effective. We make this conclusion on the basis of the performance of One-Class GMM with features selected according to SHAP feature importance. One-Class GMM models built with the 15 most important features according to SHAP show significantly better performance than One-Class GMM models built with all features of the Credit Card dataset.

A second conclusion we draw from the results in [39] is that some models benefit from SHAP feature selection, while others do not. We suspect that the initial, worse performance of One-Class SVM impacts the quality of the feature ranking that SHAP provides. Put another way, since One-Class SVM does not yield strong performance to begin with, it is not a good source of information about which features are useful for the credit card fraud detection machine learning task. The subject of future research will be to apply the SHAP feature selection technique to other application domains where one-class classifiers are an appropriate choice.

"A model-agnostic feature selection technique to improve the performance of one-class classifiers" [39], gave rise to "SHAP as a Data Reduction Technique for Highly Imbalanced Big Data" [40]. In [40], we investigate the efficacy of SHapley Additive exPlanations (SHAP) as a feature selection technique for One-Class classification tasks. In this study we utilize the Credit Card Fraud and aggregated Medicare data. Our contribution is to show that researchers can use SHAP in conjunction with One-Class Classifiers to do feature selection on highly imbalanced datasets, and then build models, with the selected features, that yield performance similar to, or better than, models built using all features. Our results

in Big Medicare data fraud detection show that an over 90 percent data reduction through feature selection can nevertheless coincide with the best performance in terms of Area under the Precision Recall Curve.

To the best of our knowledge, [40] represents the second instance of applying SHAP as a data reduction method for One-Class Classifiers (OCCs). Since [40] is an expansion on that [39], we focus on applying the SHAP data reduction technique to a wider variety of data, and another application domain. This improves our confidence in the general applicability of SHAP for feature selection. We demonstrate that employing the SHAP data reduction technique can significantly enhance the performance of One-Class Gaussian Mixture Models (GMM) in the contexts of credit card and Medicare fraud detection. The datasets used in the study are the Credit Card Fraud and aggregated Medicare Part D datasets discussed in Chapter two.

In [40], we conduct feature selection experiments where we build models using variously sized subsets of the most important features. A review of related literature reveals a gap in current research. We find a lack of studies where SHAP is utilized for feature selection in OCC models, which research covered in this dissertation aims to address. In light of the identified research gap, we explore the efficacy of SHAP in selecting the most informative features for OCCs in the tasks of Medicare fraud detection, and credit card fraud detection. Our results reveal SHAP's capability to isolate a subset of features that enhance the accuracy of models, effectively disregarding inputs that are either noisy or of minimal utility. This insight is pivotal in demonstrating how the interpretability and performance of OCCs can be significantly improved through feature selection. Moreover, we show feature selection is an effective data reduction technique, since we build models that yield the best performance with a fraction of the features, and therefore a fraction of the training data. Our results in credit card fraud detection show data can be reduced by half, and our results in Medicare insurance fraud detection show data can be reduced by over 90 percent. Our research contributes to the body of knowledge by demonstrating the effectiveness of SHAP in feature

selection techniques for OCCs, an area previously unaddressed in existing literature.

Methodology Employed in OCC Data Reduction Experiments

Our experimental methodology is structured into two phases. Each phase is conducted by the execution of distinct Python programs. The first phase is dedicated to feature selection. The second phase focuses on the evaluation of the model. In [40], both the feature selection and model evaluation phases are conducted using the Credit Card Fraud dataset and Part D datasets. Prior to their use in training or inference by classifiers, we scale all features to the $[0, 1]$ range. This ensures the magnitude of feature values do not cause features to exert undue influence on model training.

We use default values for most of the One-Class GMM classifier's hyperparameter settings. The one exception is the `n_components` hyperparameter, which we set to 2. Hyperparameter values were selected with a grid search. The grid search found default values were optimal, except for the `n_components` parameter. We use the same hyperparameter values for classifiers during SHAP feature selection and classification experiments.

In the process of feature selection, we utilize the Python library `shap`, which is an implementation of the SHAP concepts as formulated by Lundberg and Lee. To avoid confusion whether we are referring to the theoretical work versus the software library, it is important to distinguish between the theoretical framework of SHAP (denoted in uppercase) and its implementation in Python, referred to as `shap` (in lowercase). For the purposes of our study, we employ version 0.41.0 of the `shap` library [133].

Our methodology also involves our standard ten iterations of five-fold cross-validation. In the initial phase of our methodology, we focus on training a One-Class GMM model using the complete dataset, in order to use the One-Class GMM with SHAP. After fitting the One-Class GMM to the desired dataset, we provide the trained model, and a sample of the dataset as input to the `shap` library's `KernelExplainer` object. SHAP feature importance is calculated as explained in the SHAP methodology section above.

In the second phase of the experiments documented in [40], we utilize the lists of feature importance that were generated in the first phase. During this stage, we construct One-Class GMM models incorporating the top n features identified as most important by SHAP. Different sets of values of n are used for the different datasets because the datasets have different numbers of features, and the Part D data has significantly more features than the Credit Card Fraud data. For the Credit Card Fraud data n is three, five, seven, ten, 15, or 29 features. For the Part D data, n is five, ten, 15, 20, 30, 40, or 82. In both cases, we selected the numbers of features to use by starting with a few features and gradually increasing this number to encompass 50 percent of the features. Since our results show diminishing returns when more than 50 percent of features are used, we do not perform any further gradual increase in the number of features in our experiments. Instead, we build models with all features. We use all features as a means of thoroughly assessing the efficacy of our feature selection strategy.

AUPRC is employed in [40] as the primary metric for evaluating the balance between Precision and Recall in classification models. Prior research has established AUPRC as a more appropriate metric compared to other measures [114, 127]. This research shows that AUPRC more clearly reflects the impact of experimental factors. Moreover, since it is threshold-agnostic, we do not need to find optimal threshold values in order to evaluate experimental outcomes.

In [40], we use six or seven distinct values for the number of features. Therefore, the cumulative count of AUPRC scores for each amounts to $6 \times 50 = 300$. For the Part D data, we have seven distinct values for the number of features factor, so we record a total of $7 \times 50 = 350$ AUPRC scores. This comprehensive dataset of AUPRC scores facilitates a robust analysis of model performance across varying feature sets. Having 50 AUPRC scores for each level of the number-of-features factor mitigates the risk of an improbably lucky test/train split, and provides a sufficient number of data points for statistical tests.

Results Reported for OCC Data Reduction Experiments

Our first result from phase one is the feature importance lists we obtain by applying SHAP to the One-Class GMM model and the Credit Card Fraud data and the Part D data. Table 6.20 contains the 15 most important features when SHAP is used in conjunction with One-Class GMM and the Credit Card Fraud data. The features are in order by their mean SHAP importance. Since we use SHAP and GMM together to obtain the feature importance list, we refer to the feature importance as SHAP-GMM feature importance. SHAP leverages the trained model to compute the significance of each feature. Therefore, SHAP engenders a distinct ranking of features that is specific to each individual model. Due to space limitations, we report the 15 most important features for each dataset.

Table 6.20: Top 15 Credit Card Fraud Data features

Rank	Feature Name
1	V11
2	V2
3	V26
4	V12
5	V8
6	V5
7	V9
8	V7
9	V18
10	V3
11	V10
12	V16
13	V17
14	V21
15	V14

features selected by their SHAP-GMM feature importance values; the features are listed in order of their SHAP feature importance values; a rank of one corresponds to the most important feature, and so on.

As mentioned previously, to further corroborate the impact of SHAP feature selection on experimental outcomes, we apply the same treatment to a larger dataset with more features, the Part D dataset. Table 6.21 contains the result of the SHAP-GMM ranking of the Part D dataset. To illustrate how feature selection improves explainability, let us consider the One-Class GMM model built with the five highest-ranked features from Table 6.21. These features pertain to the number of prescriptions the provider's patients go to the

pharmacy to have filled, the total number of claims the provider makes, and the total supply of medications the provider prescribes for patients. It is straightforward to make the case that this model detects fraud as a function of the amount of medication the providers are prescribing for their patients.

Table 6.21: Top 15 Part D features

Rank	Feature Name
1	Tot_30day_Fills
2	Tot_30day_Fills_sum
3	Tot_Clms_sum
4	Tot_Clms
5	Tot_Day_Suply_sum
6	Tot_Drug_Cst
7	PDP_Tot_Clms
8	PDP_Tot_Drug_Cst
9	Tot_Day_Suply
10	GE65_Tot_Drug_Cst
11	Tot_Clms_std
12	GE65_Tot_Clms
13	GE65_Tot_Day_Suply
14	Tot_Day_Suply_max
15	GE65_Tot_30day_Fills

features selected by their SHAP-GMM feature importance values; a rank of one corresponds to the most important feature, and so on

Here we move on to present the results of experiments where increasing numbers of features are used to classify the Credit Card Fraud and Part D Data. We report results for the Credit Card Fraud data first. Table 6.22 presents the average Area Under the Precision-

Recall Curve (AUPRC) scores for the One-Class GMM for classification of the Credit Card Fraud data. The values represented in this table are the mean AUPRC scores the trained One-Class GMM yields when applied to the test fold of the dataset. Notably, Table 6.22 illustrates an ascending trend in AUPRC scores, up to 15 features. However, the utilization of all 29 features results in a lower AUPRC score for the One-Class GMM. This suggests that the application of our feature selection methodology notably enhances the One-Class GMM's efficacy in detecting fraudulent activities.

Table 6.22: Mean One-Class GMM AUPRC scores by the number of features selected

Features	AUPRC
3	0.2515
5	0.3669
7	0.3411
10	0.4071
15	0.6420
29	0.4971

Table 6.23 contains the AUPRC scores One-Class GMM yields when used to build models with increasing numbers of features of the Part D data. Similar to Table 6.22, the values in Table 6.23 are calculated as the mean of 50 AUPRC scores. Notably, Table 6.23 illustrates an descending trend in AUPRC scores. We can see that the use of fewer features is associated with higher AUPRC scores. This underscores the utility of SHAP data reduction for the Part D data.

Table 6.23: Mean AUC and AUPRC Scores by the Number of Features Selected

Features	AUPRC
5	0.3782
10	0.3590
15	0.3268
20	0.3028
30	0.2677
40	0.2483
82	0.1437

To informally quantify the speedup obtained by feature selection by SHAP-GMM data reduction we did an analysis of log files where training times are recorded. During the course of ten iterations of five-fold cross validation, we fit the One-Class GMM 50 times. To train a One-Class GMM with all features, the mean time to fit the One-Class GMM model is approximately 818.0265 seconds, with a standard deviation of 75.2190. The mean time to fit One-Class GMM with the five highest SHAP-GMM feature importance scores is 131.2791 seconds, with a standard deviation of 24.0600. The ratio of the mean training times is 6.2312. Therefore applying feature selection for data reduction would provide a significant speedup in experiments and application development.

Statistical Analysis of OCC Data Reduction Experiments

First, we perform a detailed statistical analysis of the results produced in experiments with One-Class GMM and the Credit Card Fraud data, presented in Table 6.22. Our statistical methodology begins with a one-factor ANOVA. The number of features used to build the model is the factor. Hereafter, we refer to this factor as the "number-of-features" factor. For this analysis, we wish to determine whether the factor significantly affects the experimental

outcome. The outcome is the AUPRC score produced by the One-Class GMM model. We conduct ANOVA, and the following HSD test when appropriate, in the manner described in Chapter four on experimental methodology.

Table 6.24 displays the results of the ANOVA test for the One-Class GMM experiments using SHAP and varying numbers of features. As the p-value obtained from this test is practically zero, the HSD test is applicable.

Table 6.24: ANOVA for the Number-of-Features factor of One Class GMM's performance in terms of AUPRC in Classifying Credit Card Fraud Data

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	4.64	0.93	264.16	*
Residuals	294	1.03	0.00		

* indicates the value is less than 1×10^{-4}

Table 6.25 provides the results obtained from the HSD test. The table relates the relative impact of the 'number of features' factor in the experiments where SHAP feature importance values were utilized for feature selection. These HSD test findings reinforce the previously noted trend concerning the Area Under the Precision-Recall Curve (AUPRC) scores, as detailed in Table 6.22. Notably, the One-Class GMM models, when constructed using the 15 most influential features identified by the SHAP analysis, demonstrate significantly higher AUPRC scores compared to the models that incorporate the entire feature set. This implies we can reduce data by 50 percent and nevertheless build One-Class GMMs that yield the best performance.

Table 6.25: HSD test Result for Credit Card Data Experiments, groupings after ANOVA of One Class-GMM AUPRC Scores for the Number-of-Features factor

Group a consists of: 15
Group b consists of: 29
Group c consists of: 10
Group cd consists of: 5
Group d consists of: 7
Group e consists of: 3

Next, we move on to conduct a statistical analysis of the impact of the number-of-features factor on the AUPRC scores of One-Class GMM models built to classify the Part D data. First, we perform the ANOVA test, to determine whether the number-of-features factor has a statistically significant impact. The result of the ANOVA test is in Table 6.27.

Table 6.26: ANOVA for the Number-of-Features factor of One Class GMM's performance in terms of AUPRC in classifying Medicare Part D Data

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	6	1.89	0.31	1050.03	*
Residuals	343	0.10	0.00		

Table 6.27: * indicates the value is less than 1×10^{-4}

Since the p-value reported in Table 6.27 is practically zero, it is possible to rank the levels of the number-of-features factor by conducting an HSD test. The result of the HSD test is reported in Table 6.28. Since models built with five features are in HSD group 'a', we can reduce data by over 90 percent and continue to build models that yield the best

performance.

Table 6.28: HSD test Result for Part D Data Experiments, groupings after ANOVA of One Class-GMM AUPRC Scores for the Number-of-Features factor

Group a consists of: 5
Group b consists of: 10
Group c consists of: 15
Group d consists of: 20
Group e consists of: 30
Group f consists of: 40
Group g consists of: 82

The outcome of the HSD test results in Table 6.28 is straightforward to interpret; fewer features, and therefore reduced data results in a higher AUPRC score. This concludes our discussion of SHAP for data reduction. We now move on to discuss an additional benefit of data reduction by feature selection, improved model data explainability and feature analysis.

6.4 EXPLAINABILITY

An additional benefit of using SHAP for data reduction is that it also provides a means of explaining models via the SHAP feature importance. In this section we cover research done where SHAP is leveraged to explain models and feature analysis.

6.4.1 Feature Analysis

Another research paper [39] inspired is "A problem-agnostic approach to feature selection and analysis using SHAP" [61]. This study shows the SHAP feature importance ranking

in a feature selection technique by selecting the k highest ranking features. Furthermore, this SHAP-based feature selection technique is applicable regardless of the availability of labels for data. We use the Credit Card Fraud detection dataset discussed in Chapter two to simulate three label availability scenarios. When no labeled data is available, unsupervised learners should be used. For a study on unsupervised techniques for anomaly detection, please see [105]. Feature analysis may also be used for anomaly detection, for example [17]. We explore feature selection for data reduction with Isolation Forest and SHAP for this case. When data of one class is available, a one-class classifier, such as Gaussian Mixture Model (GMM) can be used in combination with SHAP for determining feature importance, and for feature selection. Finally, if labeled data from both classes is available a binary-class classifier can be used in conjunction with SHAP for data reduction. Our contribution is to provide a comparative analysis of features selected in the three label availability scenarios. Our primary conclusion is that feature sets may be reduced with SHAP without compromising performance. To the best of our knowledge, [61] is the first study to explore this feature analysis technique, applicable in the three label availability scenarios.

In the methodology section of [61], we define a feature analysis technique applicable in all three data label availability scenarios. We are able to accomplish this by showing that our technique works in all three scenarios with a combination of SHAP and algorithms that are applicable in each scenario. To illustrate the technique, we utilize the Credit Card Fraud Detection dataset. Since the focus of [61] study is the exposition of a technique for working with data in the three data label availability scenarios, in [61], we use one dataset. We simulate the no-class and one-class scenarios with the Credit Card Fraud data. The Credit Card Fraud data is labeled. We use labeled data to measure the classifiers' performance in order to validate SHAP feature selection for feature analysis. We use unmodified, publicly available open-source software in our study; therefore the techniques covered here are easily transferable to other datasets.

Methodology For Feature Analsis

Our motivation for selecting Isolation Forest, GMM, and XGBoost is to use one learner for each scenario of label availability. Each classifier is appropriate depending on how much information we have about class membership. We chose SHAP as the method for determining feature importance for each model because it is applicable to each model in each scenario. Furthermore, in order to utilize SHAP, one does not need expert knowledge about a dataset's attributes. We have very little information about the origins of many of the Credit Card Fraud dataset's attributes because they are the result of PCA. Therefore, SHAP is an appropriate technique for determining feature importance. Once we have a ranking of feature importance, we confirm the feature importance by building models with increasing numbers of important features. The expected result is that models built with some number of the most important features should yield performance equivalent to models built with all features. Such a result would confirm that SHAP is capable of identifying the important attributes of a dataset.

Therefore, for each classifier, Isolation Forest, GMM, and XGBoost, we first train a model on all the viable attributes of the Credit Card Fraud dataset, that is, Amount, and features V1–V28. Then we apply the SHAP kernel explainer and calculate the SHAP feature importance as described above.

After computing the SHAP feature importance for each classifier we select the k most important attributes, according to their mean absolute SHAP value, and use the model in ten iterations of five-fold cross validation. k takes the values 3, 5, 7, 10, 15, and 29. We use the AUC and AUPRC scores resulting from five-fold cross validation in statistical analysis. The object of the statistical analysis is to determine whether there is any merit to the ranking provided by the mean absolute SHAP values of the attributes. If the statistical analysis indicates that models built with a subset of the attributes with higher mean absolute SHAP values yield performance that is similar to, or better than models built with all attributes, then we have confirmation that the mean absolute SHAP value is an indication of

the attribute's importance. As mentioned previously, we record one AUC and one AUPRC score during each fold of five-fold cross validation.

Once we have confirmation that our technique for using SHAP to compute feature importance is viable, we can do feature analysis. The SHAP feature importance, combined with knowledge of the functioning of the classifier enables us to make conjectures on the nature of the important features. In order to devise a methodology, we conduct feature analysis first by considering each list of the most important features individually. Then we analyze the features in the intersection of the most important features for each of the three possible pairs of classifiers. There is a similarity metric, the Kuncheva index, which can be used to quantify the similarity of two sets. The index was introduced by Kuncheva [111]. The Kuncheva index takes a value from -1 to 1, where identical sets have a Kuncheva index closer to 1, and disjoint sets have a Kuncheva index closer to -1. The formula for the Kuncheva similarity index, I_c, of two sets, T_i and T_j is

$$I_c(T_i, T_j) = \frac{dp - k^2}{k(p - k)}, \tag{6.1}$$

where k is equal to the sizes of the sets T_i and T_j, d is the size of the intersection of the sets, and p is the total number of features in the dataset. In addition to using the Kuncheva index to compare pairs of feature sets, we consider the group of attributes which appear in all three groups of important features.

Feature Analysis Results

We present two forms of results. The first form of results are classification performance results. These motivate the second form of results, which are presented as a feature analysis. The feature analysis covers the features selected by the combination of machine learning models and the SHAP feature importance ranking. The results we present cover all three label availability scenarios: no-class, one-class, and binary-class. We report classification results in terms of the threshold agnostic metrics AUC and AUPRC. However,

we do statistical analysis of the AUPRC scores because previous research in experiments in classifying highly imbalanced data shows that AUPRC is the better metric for revealing the impact of experimental factors [57].

Prerequisite Performance Analysis

As mentioned previously, performance analysis is a prerequisite for feature analysis. Experiments were conducted with three learners and six levels of features selected: 3, 5, 7, 10, 15, and 29. We conduct ten iterations of five-fold cross-validation, so for every learner, we have 300 experimental outcomes. We present the results for Isolation Forest first. This is the first, or, no-class label availability scenario. Table 6.29 contains the mean AUC and AUPRC scores for ten rounds of five-fold cross validation. Hence, the scores in Table 6.29 are the mean values of 50 scores, where each score is obtained as the outcome of one round of five-fold cross validation. We observe that AUC and AUPRC scores usually rise as the number of features increases. There is one exception in Table 6.29, which is that Isolation Forest yields better performance with 15 features than with 29 features. Moreover, we note in Table 6.29 that the change in AUC scores is not as dramatic. This is attributed to AUC being an inappropriate metric for imbalanced data.

Table 6.29: Mean AUC and AUPRC Scores by the Number of Features Selected for Isolation Forest

features	AUC	AUPRC
3	0.5827	0.0023
5	0.7791	0.0151
7	0.8274	0.0615
10	0.9261	0.2162
15	0.9430	0.2636
29	0.9520	0.2448

The next performance results we present are for GMM. This is the one-class label availability scenario. In Table 6.30, we see that both AUC and AUPRC scores peak for models built with 15 features.

Table 6.30: Mean AUC and AUPRC Scores by the Number of Features Selected for GMM

features	AUC	AUPRC
3	0.8971	0.2515
5	0.9260	0.3669
7	0.9280	0.3411
10	0.9397	0.4071
15	0.9584	0.6420
29	0.9447	0.5825

One factor ANOVA for GMM Feature Selection Experiments with Credit Data Analysis of Results in Terms of AUPRC

Next, we do a statistical analysis of GMM's AUPRC scores for varying levels of features used. First, we confirm that the number of features used to build a model has a significant impact on experimental outcomes with an ANOVA test. The ANOVA test result in Table 6.31 has a Pr(>F) value which is practically zero. This means that differences in the mean AUPRC scores of GMM for models built with different numbers of features is not due to random chance. Therefore, a Tukey HSD test will inform us as to which models built with which numbers of features yield the best performance. The results in Table 6.32 show that GMM models built with 15 features yield statistically similar performance.

Table 6.31: ANOVA for features as a factor of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
features	5	5.62	1.12	197.77	0.0000
Residuals	294	1.67	0.01		

Table 6.32: HSD test groupings after ANOVA of AUPRC for the features factor

Group a consists of: 15
Group b consists of: 29
Group c consists of: 10
Group cd consists of: 5
Group d consists of: 7
Group e consists of: 3

Now we move on to consider the performance of XGBoost as we increase the number

of features in the order of their SHAP ranking. This is the binary-class scenario. In Table 6.33, we find a pattern similar to the AUC and AUPRC scores we recorded for Isolation Forest and GMM. Scores increase with the number of features, however, the increase slows for the largest numbers of features.

Table 6.33: Mean AUC and AUPRC Scores by the Number of Features Selected for XGBoost

features	AUC	AUPRC
3	0.9699	0.7247
5	0.9722	0.8165
7	0.9730	0.8302
10	0.9739	0.8446
15	0.9783	0.8535
29	0.9793	0.8570

Finally, we conduct a statistical analysis of XGBoost's AUPRC scores. In order to determine if an HSD test is applicable, we conduct an ANOVA test. The ANOVA test results are listed in Table 6.34. Since the Pr(>F) value associated with the number of features is practically zero, an HSD test is applicable. The HSD test result in Table 6.35 shows that, as is the case with GMM and Isolation Forest, there is no statistically significant difference between the AUPRC scores of models built with 15 or 29 features.

One factor ANOVA for XGBoost Feature Selection Experiments with Credit Data Analysis of Results in Terms of AUPRC

Table 6.34: ANOVA for features as a factor of performance in terms of AUPRC

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
features	5	0.61	0.12	152.06	0.0000
Residuals	294	0.24	0.00		

Table 6.35: HSD test groupings after ANOVA of AUPRC for the features factor

Group a consists of: 29, 15
Group ab consists of: 10
Group bc consists of: 7
Group c consists of: 5
Group d consists of: 3

The key take-away from our performance analysis is regarding the number of features that we can build models with reduced data by applying SHAP as a feature selection technique. The HSD test results for XGBoost, Isolation Forest, and GMM all indicate that, with the Credit Card Fraud Data, one may build models with the top 15 features, and they yield performance similar to, or better than models built using all features. It is also important to note that we obtain these results for models that are appropriate to use in all three scenarios of label data availability. Put another way, the HSD results show that no matter how much information we have about the class membership of instances of a dataset, we can apply SHAP to rank the attributes of the dataset, select the top 50% of features, and build models that yield performance equivalent to using all the features. This confirms the relevance of

Table 6.36: Part I, Features selected by applying SHAP to Isolation Forest (IF), GMM, and XGBoost (XGB)

Classifier, Amount, Features V1-V13												
IF	Amount	V2	V3	V4					V10		V12	V13
GMM		V2	V3		V5	V7	V8	V9	V10	V11	V12	
XGB	Amount		V3	V4		V7	V8	V9	V10	V11	V12	V13

Table 6.37: Part II, Features selected by applying SHAP to Isolation Forest (IF), GMM, and XGBoost (XGB)

Classifier, Features V14-V28													
IF			V16		V18	V19		V22		V24	V25	V26	V27
GMM	V14		V16	V17	V18		V21					V26	
XGB	V14	V15		V17		V19		V22					

SHAP feature importance.

Feature Analysis

The second results we present are in the form of a feature analysis. In Tables 6.36 and 6.37, we list the 15 highest-ranked features, where the rank of a feature is determined by applying SHAP to the classifier and the Credit Card Fraud data. Arranging the features in this way enables us to observe which important features are in common for different combinations of classifiers. If a cell in Table 6.36 or 6.37 is blank, it means the feature's SHAP rank is not in the top 15 for the classifier.

There are three forms of feature analysis that we can do based on the results from ranking the features with SHAP. We can analyze each feature set individually [128], we can compare pairs of feature sets, and we can look at features that all three feature sets have in common. Therefore, we proceed with analysis in this order. To begin the analysis of individual feature sets, we remind the reader that the Credit Card Fraud dataset has two features with a clear meaning, Time and Amount. As stated previously, we do not use the

Time feature because it is equivalent to a unique identifier and would cause memorization. Therefore, the only feature fed to our models which relates directly to something in experience is the Amount feature, which holds the value of the transaction amount. As we explained above, the remaining features named V1–V28 are mixtures of other values of features that are computed by PCA. Since it is possible to train machine learning models such as XGBoost that can classify the Credit Card Fraud data with high AUPRC scores, we know the values V1–V28 carry meaningful information that is useful in identifying fraudulent transactions. Of the three models we use, GMM and Isolation Forest do anomaly detection, so important features for these models pertain to anomaly detection. XGBoost does supervised learning, so features important for XGBoost must carry information that relates to the Class (fraudulent/not fraudulent) label of the Credit Card Fraud dataset.

We begin the analysis by inspecting the 15 most important features for Isolation Forest, in the no-class scenario. Since Isolation Forest is an unsupervised learning algorithm, it operates by detecting anomalous instances. In the context of credit card transactions, these anomalies can potentially represent fraudulent activities. When SHAP is applied to Isolation Forest, to rank the features of the Credit Card Fraud data, the top 15 features are: Amount, V2, V3, V4, V10, V12, V13, V16, V18, V19, V22, V24, V25, V26, and V27. Therefore, anomalies in the Credit Card Fraud data must be more apparent in these features. The inclusion of Amount indicates the transaction size's significant role in Isolation Forest's anomaly detection process.

Next, we consider the feature set which results from applying SHAP to GMM and selecting the 15 features with the highest SHAP ranking. This is the one-class scenario. GMM the fits the data to one or more multivariate Gaussian distributions. This lends GMM a natural capability to identify clusters or groups within the data. The fifteen most important features identified by SHAP are V2, V3, V5, V7, V8, V9, V10, V11, V12, V14, V16, V17, V18, V21, V26. We make a note that Amount is missing from GMM's feature set. This indicates GMM detects anomalies with a focus on inherent transaction characteristics rather

than transaction size.

The third feature set we cover in this analysis is the one obtained by in the binary-class label availability scenario by applying SHAP to XGBoost, and selecting the 15 most important features. The features are Amount, V3, V4, V7, V8, V9, V10, V11, V12, V13, V14, V15, V17, V19, V22. The combination of XGBoost and SHAP demonstrates a balanced emphasis on both the transaction amount and PCA derived features. Since XGBoost yields the best performance, it is tempting to conclude that XGBoost and SHAP is the best feature selection technique. However, we wish to point out that in true no-class and one-class scenarios, one will not have the luxury of a labeled dataset to employ XGBoost with.

A second form of analysis we can do is to compare features selected between the pairs of classifiers. This is an important analysis to do. In order to explain why the analysis is important, we must remind the reader that we do not have information about what 28 out of the 30 attributes of the Credit Card Fraud data represent. These are the features named V1–V28. We only know that they are the outcome of applying PCA to some other data, which we no nothing about. However, SHAP uses the classifier, and the independent features of the data to determine feature importance. If SHAP indicates the same features are important for two classifiers, and we know something about how the two classifiers learn to classify the data, we can learn something about the feature. We can deduce that the feature is useful in distinct processes simultaneously. Such deductions can provide information about the nature of a feature. We cannot deduce, for example, that V3 is a person's age, but we can deduce that it influences the outcome of GMM and XGBoost at the same time. We believe this method of analysis is novel. Even though we do not know anything about the origins of the features V1–V28, we can nevertheless learn something about them by observing how important they are to pairs of classifiers.

Since we have three classifiers, there are three possible pairs, Isolation Forest and GMM, Isolation Forest and XGBoost, and GMM and XGBoost. Isolation Forest and GMM have

seven features in common: V2, V3, V10, V12, V16, V18, and V26. Therefore, in Table 6.38, we see the Kuncheva index is -0.0667. Since Isolation Forest and GMM take different approaches to anomaly detection, we surmise these features are useful in both approaches. That is to say, since these features are important to Isolation Forest, we know certain combinations of values of these features occur infrequently in the data. Moreover, since these features are important to GMM, we also know that these features tend to cause data points to lie outside high probability regions of the multivariate Gaussian distributions that GMM fits to the Credit Card Fraud data. Hence, we conjecture the instances of the Credit Card Fraud data that correspond to fraudulent transactions are in isolated regions of the feature space of the dataset.

Table 6.38: XGBoost+SHAP vs GMM+SHAP vs Isolation Forest+SHAP

Feature Set 2 Feature Set 1	GMM+SHAP	Isolation Forest+SHAP	XGBoost+SHAP
GMM+SHAP	1.0000	-0.0667	0.2000
Isolation Forest+SHAP	-0.0667	1.0000	-0.0667
XGBoost+SHAP	0.2000	-0.0667	1.0000

We move on to compare the nine features in common between the GMM and XGBoost feature sets. These are V3, V7, V8, V9, V10, V11, V12, V14, and V17. We note that XGBoost and GMM have a larger number of features in common. Table 6.38 shows that these two feature sets have the largest Kuncheva index value of 0.200. XGBoost and GMM are dissimilar techniques. As stated previously, GMM approximates the majority class with a summation of multivariate Gaussian distributions. XGBoost adds the outputs of decision trees to yield a probability that an instance is a member of a class. Hence, on one hand, we find no argument in terms of the classifiers functioning that explains the feature overlap in these two scenarios. On the other hand, we note that XGBoost and GMM yield stronger

performance. From that perspective, it would make sense that we see a larger overlap in the 15 most important features, as compared to Isolation Forest. XGBoost and GMM are also allowed more information about the class labels, and this is another factor that contributes to the larger overlap in the sets of features selected in both scenarios.

The final pair of classifiers we compare are XGBoost and Isolation Forest. Of the 15 most important features that SHAP selects for both classifiers, the features in common are: Amount, V3, V10, V12, V13, V19, V22. Similar to Isolation Forest and GMM, there are seven features in common. Hence, in Table 6.38 the Kuncheva index for these two feature sets is also -0.0667. Though the Kuncheva index value for this pair of feature sets is in the same as Isolation Forest and GMM, the features are different. This is not surprising since we are comparing the outcome of feature selection in different label availability scenarios. Interestingly, transaction amount is selected for both classifiers. This is interesting since both Isolation Forest and XGBoost are decision tree based learners. Therefore, we surmise that Amount is a useful variable for separating the sets of transactions into classes, since the nodes in decision trees are rules for dividing instances of the dataset into classes based on the instances' relation to a specific value of an attribute. We also know that Isolation Forest has no information about the class label, and therefore separates the instances of a dataset by anomaly detection. For a related study on a statistical method for outlier detection in Medicare data please see [3,5,6]. We conclude that transaction amount carries information about how unusual (anomalous) an instance of the credit card data is, and it also carries information about class membership. Hence, we have evidence of how one feature can be important in different label availability scenarios.

Another comparison of feature sets is to look at what things are in common for all three feature sets. First, we look at the features listed in Tables 6.36 and 6.37 that are selected in all three scenarios. These are V3, V10, and V12. We conclude that these features are both strong indicators of anomalies, and they must also be effective in determining the label. The features are strong indicators of anomalies because they are important for both GMM

and Isolation Forest. The features must also be effective in determining the label because they are important for XGBoost. Such a conclusion about features of the Credit Card Fraud data can only be made by doing this type of analysis involving the feature importance for classifiers that may be used in the different label availability scenarios.

The second interesting feature of Tables 6.36 and 6.37 that we take note of is the features that are not important for any of the classifiers. We find V1, V6, V23, and V28 are not in the 15 most important features for any scenario. Hence, we conclude that these features do not indicate anomalies, and they do not form patterns that XGBoost might use to predict the fraud label. This concludes the second set of results we provide in [61].

Conclusions from Feature Analsis

In [61], we demonstrate a technique for analyzing a dataset that is applicable in all label availability scenarios. SHAP is used in combination with the Isolation Forest, GMM, and XGBoost algorithms for each label availability scenario. This enables us to do the analysis that determines which features are important, regardless of whether labels are available for the data. Our results corroborate that, when labels are available, the feature importance assigned by SHAP is consistent with performance. Put another way, as we add features to models in order of their mean absolute SHAP values, the models' performance improves until we have added 10 or 15 features, depending on the classifier.

Our research in [61], and covered here, shows a treatment that can be applied to data in all data labeling scenarios. Whether there are no labels on the data, or only instances of one class are available, or the data is labeled, we show researchers can use SHAP to determine feature importance. Performance analysis of experiments that utilize SHAP feature importance to build models confirms that SHAP feature importance is meaningful in terms of its ability to identify features that contribute to the performance of a model. SHAP values also place an order on the features in terms of their impact on performance. Moreover, even though we are only given that most of the Credit Card Fraud data's features

are derived from principal components analysis of unknown data, our feature analysis technique is nevertheless applicable. We have demonstrated a new technique for doing feature analysis in all label availability scenarios. Future work includes application of our feature analysis to new application domains.

6.5 CHAPTER SUMMARY

In this chapter we covered many methods of data reduction via feature selection. The techniques covered in this chapter fall in two broad categories: ensemble feature selection techniques, and SHAP. We cover three variants of ensemble feature selection techniques: Voting Feature Selection, Feature Popularity, and Supervised Feature Selection. Voting Feature selection is similar to Supervised Feature Selection, in that it uses the built-in feature importance functionality of multiple learners for classification. Unlike Supervised Feature Selection, Voting Feature Selection uses filter-based feature selection techniques. Moreover, Voting Feature Selection looses the order of features it selects. Our section on Feature Popularity explains that Feature Popularity is implemented as second iteration of Voting Feature Selection. Supervised Feature Selection also leverages built-in feature importance from multiple learners. However, since it uses the technique of selecting the median feature importance rank to construct the final list of selected features, it preserves an order of features selected, and is therefore more useful in further feature analysis. Furthermore, in preliminary experiments we found Supervised Feature Selection outperformed Voting Feature Selection the same learners and datasets.

The second major area of research reviewed in this chapter is SHAP. We show how SHAP is leveraged as a feature selection technique which can be used for data reduction. We review research where we found built-in feature selection importance can outperform SHAP. However, many classifiers, such as One-Class SVM and One-Class GMM do not have built-in feature importance functionality, and SHAP is a viable means to determine feature importance for data reduction with One-Class classifiers. To the best of our knowledge, we

are the first to apply SHAP for feature selection with One-Class classifiers, and we use SHAP for data reduction with Credit Card Fraud detection, and Medicare Fraud detection datasets simulated one-class scenario experiments. We consider our experiments with SHAP and the one-class classifiers a success, since we show that, models built with reduced training data, yield performance that is similar to, or better than the performance of models built with all the available training data. In conclusion, in this Chapter we have shown multiple ways in which feature selection can be used as robust data reduction technique.

CHAPTER 7

HYBRID TECHNIQUES

7.1 INTRODUCTION

In this chapter, we cover hybrid data reduction techniques. These are techniques which involve a combination of the data reduction techniques discussed in this dissertation in previous chapters. The techniques are feature selection, and data sampling. The research discussed here is originally documented in "Improving medicare fraud detection through big data size reduction techniques" [169], and the subsequent work "Data Reduction Techniques for Highly Imbalanced Medicare Big Data" [62].

The motivation for [169] is to address the challenges of imbalanced Big Data and high dimensionality. Feature selection and data sampling are often utilized as initial data preparation steps. On one hand, data sampling is adopted to tackle the issue of class imbalance. It entails adjusting the training dataset by adding or subtracting examples to ensure a more even balance between fraudulent and non-fraudulent entries. On the other hand, feature selection, which addresses high dimensionality, focuses on choosing a specific group of attributes from the training data, and only these chosen attributes are used to construct the final model. This not only streamlines the learning process but can also enhance classification accuracy by discarding less relevant attributes.

7.2 FEATURE SELECTION, FOLLOWED BY RANDOM UNDERSAMPLING

The initial research we conducted in [169] we attempted the hybrid approach of feature selection followed by random undersampling. We refrain from presenting tables of results and statistical analysis because the same data is presented in the next section which covers an

expansion of this research. In [169], we conduct experiments with the aggregated Medicare Part D Big Data discussed in Chapter two. Moreover, we employ the Supervised Feature Selection (SFS) technique discussed in Chapter six. We employ RUS to generate five distinct class ratios. In this research, we assess six machine learning classifiers in detecting Medicare insurance fraud. The learners are: XGBoost [28], Logistic Regression (LR) [30], Extremely Randomized Trees (ET) [35], LightGBM [101], Random Forest (RF) [22], and CatBoost [141]. The metrics we use to evaluate performance are Area Under the Receiver Operating Characteristic Curve (AUC) [19] and Area Under the Precision-Recall Curve (AUPRC) [20]. Furthermore, we conduct experiments with models built with the original dataset for comparison with those obtained through our proposed methodology to underscore the enhancement in performance. The outcomes of the experiments conducted in [169] indicate that it is possible to achieve a substantial reduction in the size of the training data without a need to compromise performance in terms of AUPRC. This reduction in the size of the training data also contributes to expedited model training times.

There are three phases of experiments carried out in [169]. In the initial phase of our investigation, we apply Random Undersampling (RUS) to generate datasets with the following induced class ratios: 1:1, 1:3, 1:9, 1:27, and 1:81, Furthermore, we conduct experiments with the original dataset which exhibits a class ratio of 1:1,429. Therefore, in the first phase of experiments, for each classifier we build six distinct models. We build one model for every combination of classifier and induced class ratio. The objective of this experimental setup is to determine the effect of RUS on model performance, in order to investigate how different class ratios influence the outcomes of classification models.

Here, we summarize the outcomes of experiments involving RUS. In terms of AUC, CatBoost consistently attains the highest scores across various class ratios. Its peak performance observed at a 1:27 ratio. Similarly, XGBoost and LightGBM algorithms exhibit relatively strong performance. These two learners also sustain high AUC scores across a broad spectrum of class ratios. In terms of AUPRC, CatBoost emerges as the superior

model, since it yields the highest AUPRC scores across all class ratios. XGBoost and LightGBM also demonstrate competitive prowess in this regard. Conversely, Logistic Regression (LR), Random Forest (RF), and Extra Trees (ET) algorithms yield lower AUPRC scores. While the mean AUC scores are consistently high, indicating minimal influence of RUS, the mean AUPRC scores exhibit greater variability. As mentioned previously in this dissertation, we find AUPRC to be more informative in the evaluation of classification models built with large imbalanced datasets. We find AUPRC gives a clearer insight into the effect of RUS. Statistical analysis consisting of Analysis of Variance (ANOVA) [78] and Tukey's Honestly Significant Difference [165] tests were conducted to determine the effect of the classifier and class ratio factors on AUPRC scores. The results of the tests confirm that models built with training data having class ratios of 1:81, the Original class ratio and 1:27 yield the best performance. These levels of the RUS factor belong to HSD test result group 'a'. Furthermore, an intra-group analysis of group 'a' reveals that models built with training data with a 1:81 class ratio outperform models built with training data at the original class ratio, and models built with training data with a 1:27 class ratio.

In the next phase of experiments conducted in [169], we integrate the SFS ensemble feature selection technique. For a comprehensive understanding of ensemble feature selection methods please see [170]. We use SFS to derive feature ranking lists. The feature ranking lists are used to select subsets of the top n features, where n is equal to 7, 8, 9, 10, 15, 20, 25 or 30. In this phase of experiments, we build classification models for every combination of list of features and learner. ANOVA and subsequent HSD tests lead to the following conclusions about the impact of feature selection in the experiments: models built with feature selection applied to their training data yield better performance with respect to AUPRC. The statistical tests show a significant improvement over the performance of models built on training data without feature selection applied. Among the models assessed, the HSD test results show that models built with the top 10 features selected by SFS yield the highest AUPRC scores.

In the last phase of experiments conducted in [169], we use a hybrid methodology that combines feature selection with RUS. The approach is to apply feature selection first, followed by RUS. The process yields a pronounced reduction in data volume. The performance of models constructed using this considerably diminished dataset is then compared with the performance models built with the original dataset. We use the optimal results from the previously mentioned experiments to select the levels of factors used in the hybrid experiments. Therefore, we use the list of the top ten features as determined by SFS for feature selection and RUS to induce a class ratio of 1:81 in the training data.

Statistical analysis was not conducted to compare the outcomes of experiments where feature selection and RUS are applied with the outcomes of experiments where data is left at its original class ratio. Instead, mean AUC and AUPRC scores are compared to show that in most cases, models built with feature selection and RUS applied to their training data yield higher AUC and AUPRC scores.

The results of this initial study inspired an expanded study with additional datasets, and an expanded methodology as well. In addition to investigating the effect of feature selection and RUS, in [62] we also investigate the effect of applying RUS first, then feature selection. Since [62] is an expansion of [169], we defer reporting tables and statistical analyses to the next section.

7.3 RANDOM UNDERSAMPLING, FOLLOWED BY FEATURE SELECTION

In [62], we perform experiments on two imbalanced Big Medicare Datasets. In the experiments, the data reduction techniques are applied alone, and in combination. The statistical analysis of the experimental outcomes indicates that the data reduction techniques, in combination, yield the best performance in terms of Area Under the Precision Recall Curve (AUPRC) [20]. Furthermore, that performance is significantly better than using all features.

Related Work

In their investigation into the classification of imbalanced Medicare Big Data, Johnson and Khoshgoftaar apply Deep Learning algorithms and evaluate performance using Geometric Mean and Area Under the Curve (AUC) metrics [85]. Employing a dataset of approximately five million instances and increasing levels of RUS, their study finds that performance metrics deteriorate when the minority class exceeds one percent of the training data. Our research diverges from Johnson and Khoshgoftaar's in several significant ways. Firstly, we use AUPRC for evaluation of experimental outcomes, in contrast to their use of Geometric Mean And AUC. Secondly, Johnson and Khoshgoftaar do not employ feature selection techniques as we do here. Hence, their study does not include a detailed exposition of the methodologies available for combining feature selection and RUS. A third major difference in our studies is the learners we employ. Johnson and Khoshgoftaar employ neural networks, one type of Bagging classifier, Random Forest, and one type of Gradient Boosting Decision Tree classifier. Here, we use three instances of Gradient Boosting Decision Tree classifiers, and two types of Bagging classifiers. These methodological choices create a distinct difference between our work in [62] and the research conducted by Johnson and Khoshgoftaar.

Discussion of Methodologies

Here we discuss the five experimental scenarios covered in [62]. Four of the five scenarios are data reduction techniques. The fifth scenario is the control scenario, where we leave data unchanged in order to assess the efficacy of the data reduction techniques. We provide the discussion here to aid in understanding which scenario is ultimately found to yield the best performance. Consistent with other experiments documented in this dissertation, all experiments involve ten iterations of five-fold cross validation. Now, we proceed to discuss the aspects that are unique for each scenario. The scenarios exhaust the ways one could apply RUS and Feature Selection as data preprocessing steps. The first scenario, which

we call "Scenario One" is the scenario where we apply RUS only. While other sampling techniques are available, results documented in previous research on sampling techniques applied to highly imbalanced Big Data show that RUS yields the best performance [65]. Since RUS is only applied to the training data, it must be applied during each fold of five-fold cross validation. In order to apply RUS, we select a target class ratio of minority to majority instances, then we randomly remove instances of the majority class until the target class ratio is reached. For the experiments in this study, we use target class ratios of 1:1, 1:3, 1:27, 1:81. Moreover, we include experiments where the data is left at its original class ratio to validate the effect of RUS. The stage in Scenario One experiments where RUS is applied is depicted in Figure 7.1.

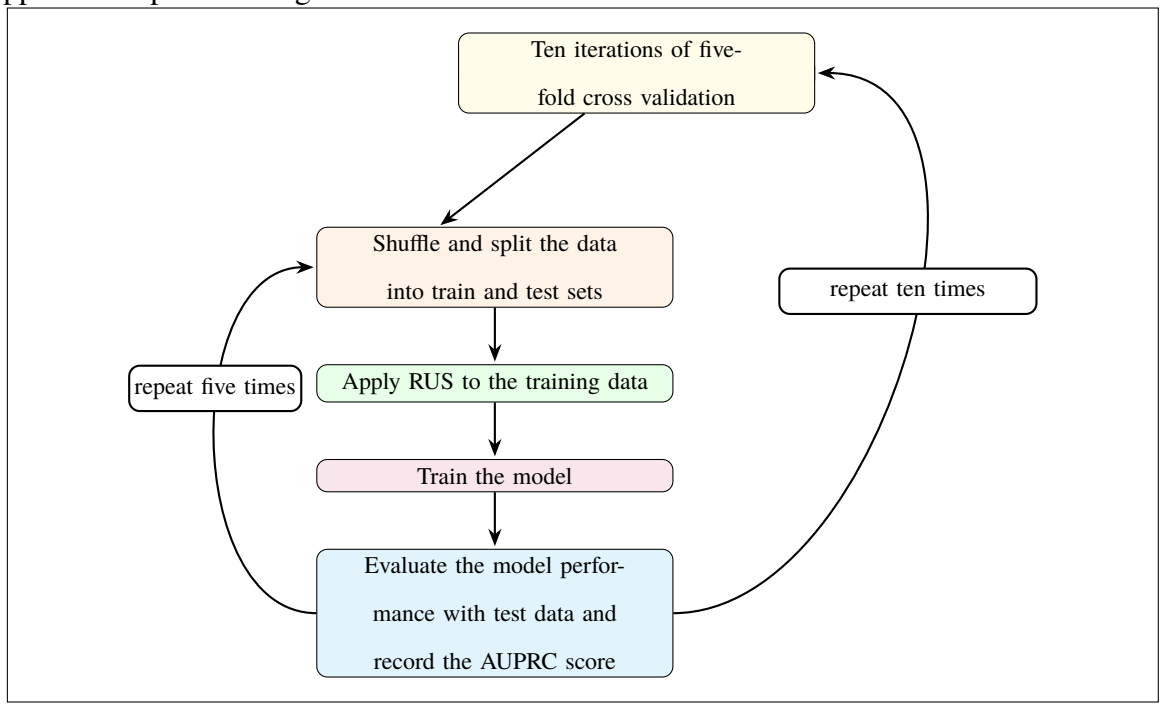

Figure 7.1: Procedure for Scenario One experiments, RUS only

This concludes our discussion of Scenario One, now we move on to discuss Scenario Two. Regarding Scenario Two, it is important to note that supervised feature selection takes place prior to the start of the ten iterations of five-fold cross validation. The crux of the feature selection process is a feature ranking technique. Our feature ranking technique

leverages six learners to generate an ordered list of the features of a dataset. The six learners are CatBoost, XGBoost, LightGBM, Decision Tree, Random Forest, and ET. Each of these learners builds a feature importance list as a side effect of the training process. Therefore, we train the six learners to obtain six feature importance lists. To apply our feature selection technique, we merge the six lists according to the following logic: we assign each feature the median value of its rank in each of the six feature importance lists. This places an order on the features in the dataset, and we can select a number of features in this order. For further details on SFS please see Chapter six on feature selection.

Scenario Two experiments are experiments where we apply SFS as the only data pre-processing step. The feature ranking process must be performed only once per dataset. In Figure 7.3, we show how feature selection fits into Scenario Two. At the beginning of Scenario Two we select the top k features from the list that the feature ranking process produces. For Scenario Two with Part D data, we use this technique to build feature sets of sizes 7, 10, 15, 20, 25, and 30 features. We include experiments where all features are used as a check for the effectiveness of the feature selection technique. One may notice in our results for experiments with Part D data that we have feature sets named “7a” and “7b”. This is because it is possible for two features to be assigned the same rank, because they have the same median rank. Hence, we build datasets with seven features with one of each feature that has the same rank at position 7.

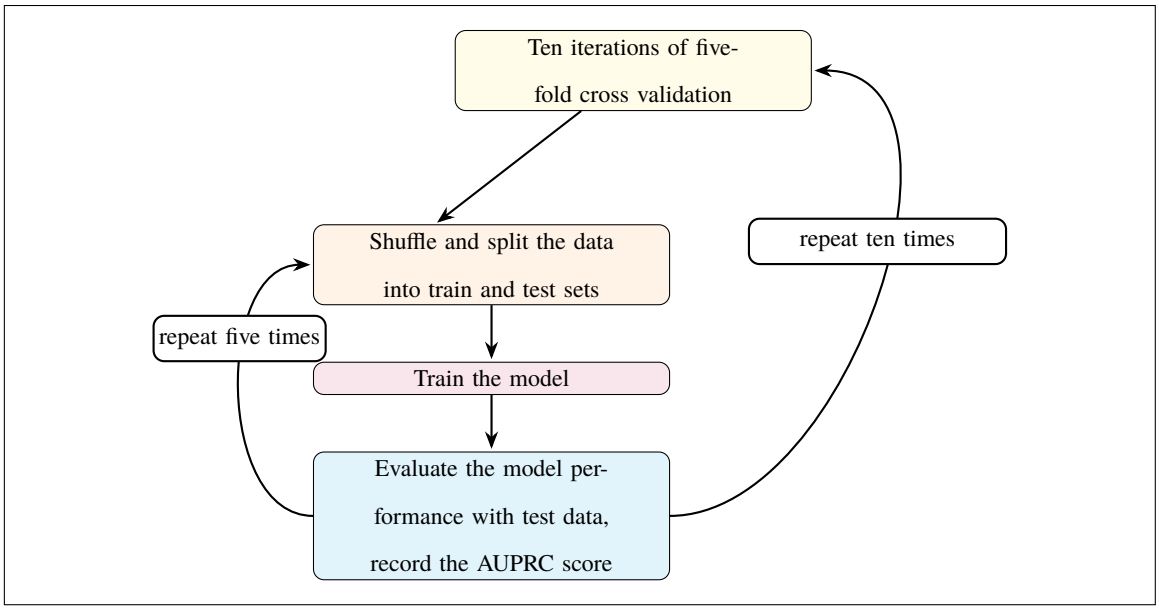

Figure 7.2: Five-fold cross validation

For experiments with Part B data, we use feature sets of size 10, 15, 20, 25, 30 as well as the set of all features. Experiments with Part D data showed that models built with ten features yielded performance that is significantly better than the performance of models built with any other number of features, hence we conducted the experiments with seven features for the Part D data. For experiments with Part B data, models built with ten features did not significantly outperform other models in terms of AUPRC scores, so it was not necessary to conduct experiments with fewer features.

The next scenario, Scenario Three is the first scenario where we use a combination of RUS and feature selection. In Scenario Three, we do feature selection first, then apply RUS. Before the start of the ten iterations of five-fold cross validation, the ensemble feature selection technique is applied to the dataset to rank the features. At the start of the experiment, a decision is made on how many of the highest ranking features will be used. Since our results from the Scenario Two experiments with the Part D data show that models built with seven features do not outperform models built with ten features, for our Scenario Three experiments we use feature set sizes of 10, 15, 20, 25, 30, and all features. The same

numbers of features are used for experiments with the Part D and the Part B data. Scenario Three experiments are similar to Scenario One experiments, since we apply RUS to the training data before the training step of each fold of five-fold cross validation. However, since the Scenario One experiments with the Part D and Part B data both show that the 1:81 class ratio yields the best performance, we only apply RUS to induce class ratios of 1:81 in the Scenario Three experiments. Scenario Three is depicted in Figure 7.4.

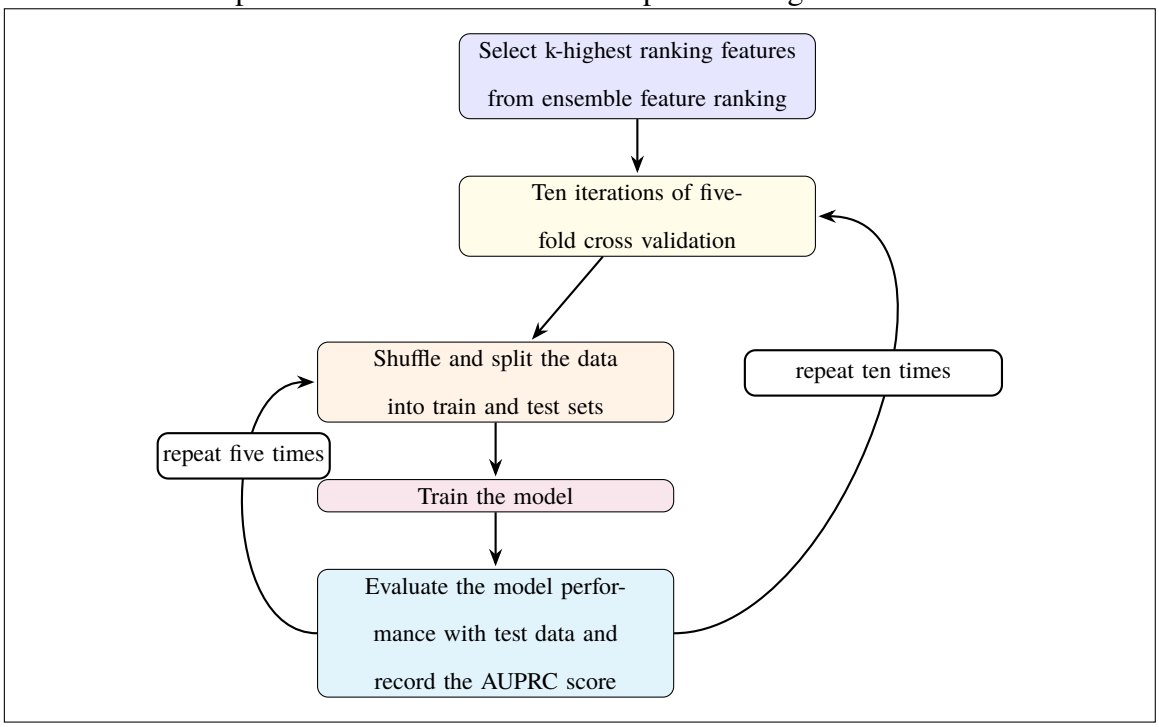

Figure 7.3: Methodology for Scenario Two experiments, feature selection only

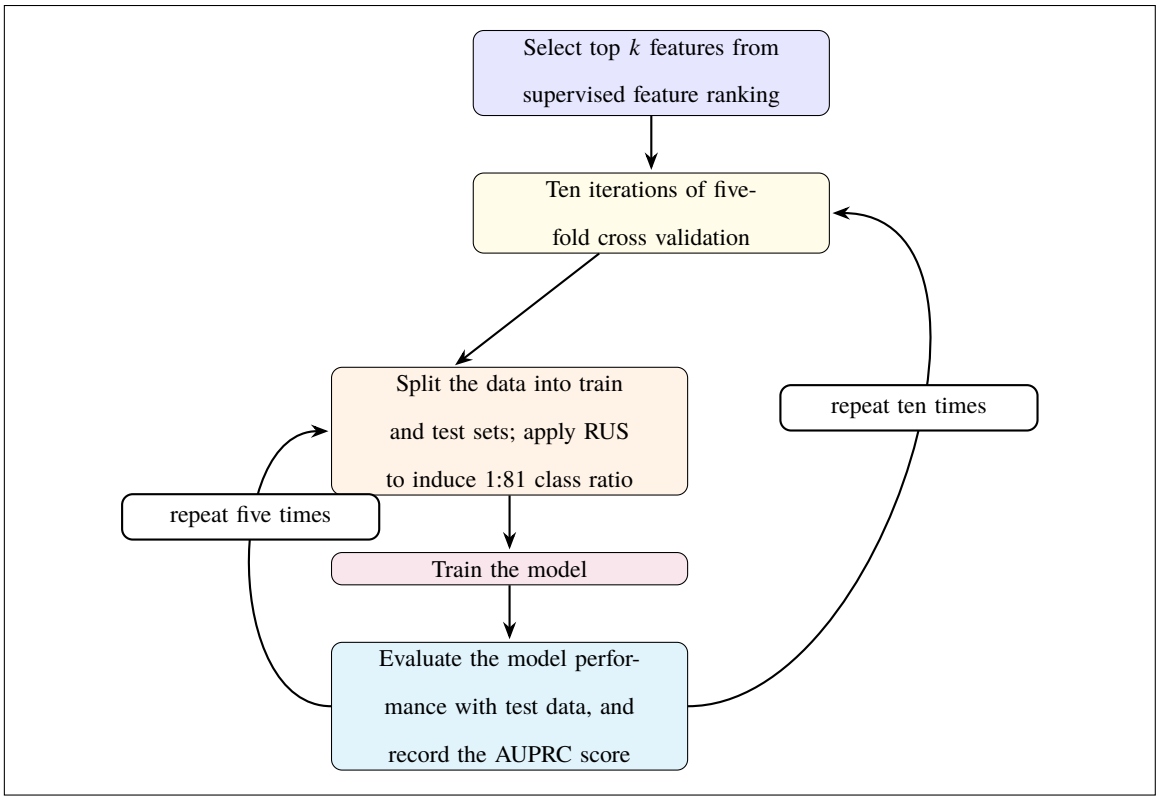

Figure 7.4: Methodology Scenario Three Experiments, for feature selection, then RUS

In Scenario Four, we apply RUS prior to feature ranking. We refer to the modified feature ranking process as RUS prior to feature ranking. RUS is applied to the data prior to the start of the ensemble feature ranking technique. We only apply RUS to induce a 1:81 class ratio in the data prior to feeding the data to the classifiers to obtain the feature importance lists. As stated previously, we use the 1:81 class ratio because we found that the 1:81 ratio yields the best performance in the Scenario One experiments. In order to perform RUS prior to feature selection, we make a modification to the ensemble feature selection technique. We make the modification in order to mitigate the impact of RUS on the ensemble feature ranking process. The modification is that for each learner, we repeat the feature ranking process ten times. That is, we apply RUS to the data, to generate six ranked lists of all features, ten times. Then, in the merge step we merge all the ranked lists. In our study we use six learners, so we generate sixty ranked lists of features. We then

merge the sixty ranked lists according to the same logic as the feature selection technique as it is described in figure 7.5.

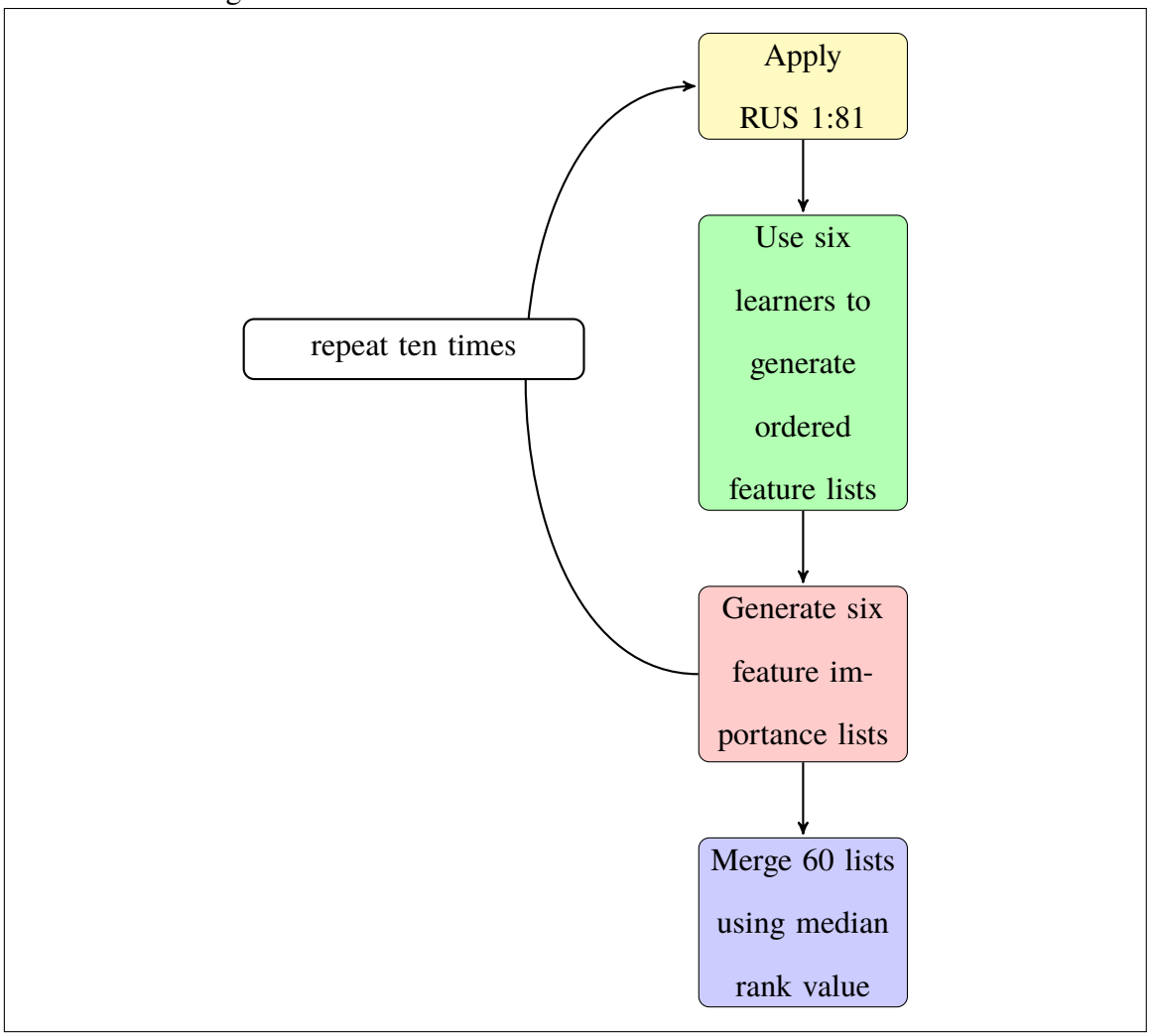

Figure 7.5: Methodology for RUS followed by Supervised Feature Selection

After generating the ranked list of features from the RUS then feature selection process depicted in Figure 7.5, we are ready to conduct the Scenario Four experiments. We use the same numbers of features, 10, 15, 20, 25, 30, and all features, for the Scenario Four experiments as used in the Scenario Three experiments. In these experiments, we perform RUS during cross validation. It is important to note that the features in each subset selected in Scenario Four may be different from the features selected in Scenario Two or

Three because the feature ranking process for Scenario Four is different from that used in Scenarios Two and Three. Please see Figure 7.6 for a graphical description of Scenario Four. In Scenario Four, RUS to induce a class ratio of 1:81 is also applied during the training phase as well. We induce the 1:81 class ratio since this ratio yields the best results in Scenario One experiments.

We have Scenario Five as a control so that we can be certain there is some benefit to our data reduction techniques. In Scenario Five, we apply no preprocessing to the data before using it to train the classifiers. Therefore, in Scenario Five, we do ten iterations of five-fold cross validation, with all features and the data at its original class ratio. Hence, Scenario Five is depicted in Figure 7.2. This concludes our discussion of methodology, now we move on to present our results.

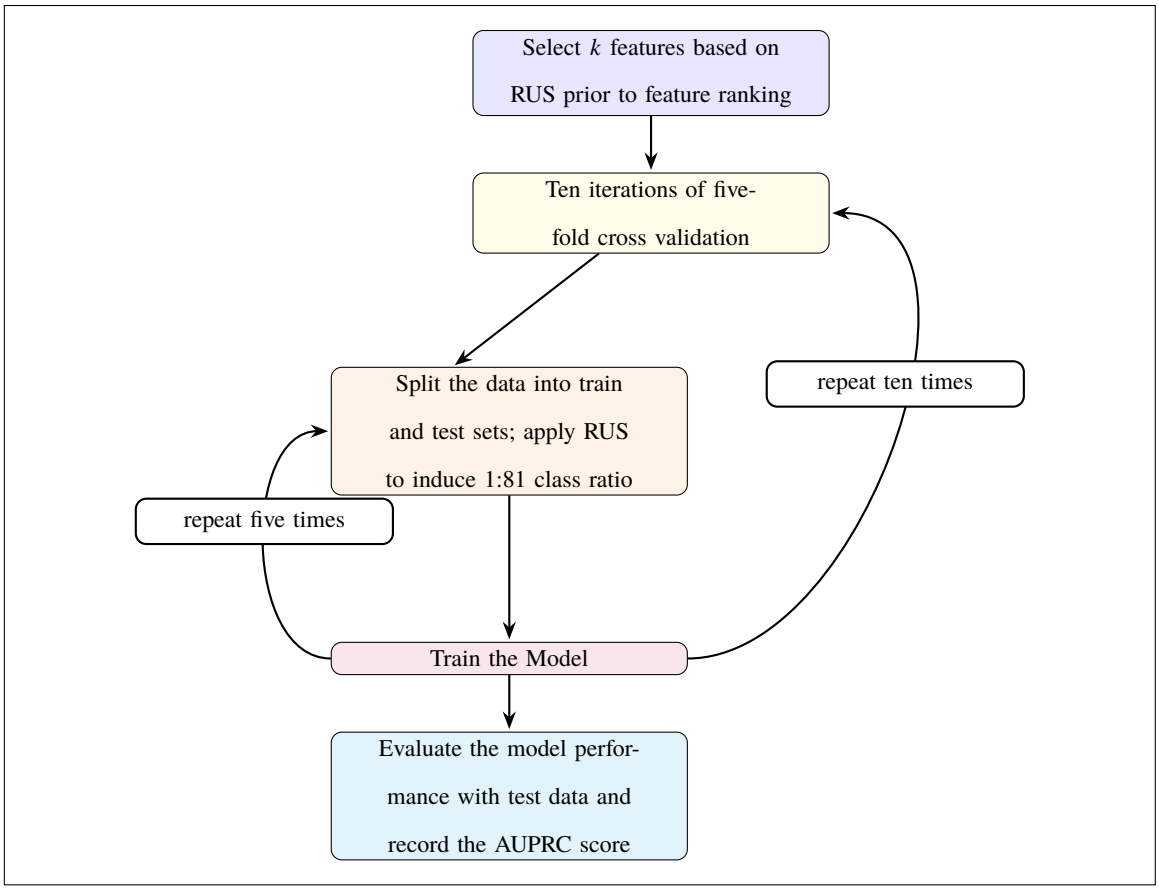

Figure 7.6: Methodology Scenario Four Experiments, for RUS then feature selection

Results of Experimental Scenarios

In this section, we present results for classification experiments conducted in [62]. All the numeric figures in tables in this section are the mean values of ten iterations of five-fold cross validation. We present classification results first for experiments on the Part D data, and then for experiments with the Part B data. We have subsections for Scenarios One through Four. Results for Scenario Five are included as the columns in the tables for Scenario One where the original class ratios are used. In the tables of this section, the figures in bold text indicate the maximum value for the combination of classifier and factor.

Part D Scenario One: RUS only

Table 7.1 holds results from experiments where we vary the induced class ratio with RUS. In Table 7.1, we notice an upward trend in AUPRC scores as the class ratio approaches the original ratio. However, there are some cases, such as that of LightGBM's, when experiments where RUS is applied yield higher AUPRC scores than cases where RUS is not applied. As can be seen in Table 7.1 for LightGBM, this is the 1:27 ratio. The substantial improvement in LightGBM's performance is significant, and highlights the importance of including RUS in experiments with highly imbalanced Big Data for evaluating classifiers.

Table 7.1: Mean AUPRC values by classifier and induced class ratio for ten iterations of five-fold cross validation, for Part D Scenario One

Ratio Classifier	1:1	1:3	1:9	1:27	1:81	1:1,429
CatBoost	0.6304	0.7047	0.7498	0.7662	**0.7798**	0.7793
ET	0.0937	0.1199	0.1635	0.2029	0.2401	**0.3254**
LightGBM	0.5786	0.6588	0.7025	**0.7183**	0.6783	0.5132
Logistic Regression	0.1189	0.1701	0.2173	0.2455	0.2700	**0.3060**
Random Forest	0.1784	0.2208	0.2212	0.2240	0.2199	**0.2469**
XGBoost	0.6065	0.6768	0.7164	0.7377	0.7351	**0.7372**

Part D Scenario Two: Feature Selection Only

The experimental outcomes in terms of AUPRC are listed in Tables 7.2 and 7.3. Interestingly, the classifiers yield higher AUPRC scores when feature selection is applied. CatBoost, Random Forest, Logistic Regression, and XGBoost yield the highest AUPRC scores with 15 features. ET yields the highest AUPRC score with nine features. LightGBM yields the best performance with ten features. Therefore, the results in Tables 7.2 and 7.3 are strong

evidence that feature selection can improve the performance of classification results with Medicare Part D data.

Table 7.2: Mean AUPRC values by classifier and number of features (Part 1) for ten iterations of five-fold cross validation, for Part D Scenario Two

Features Classifier	7a	7b	8	9	10
CatBoost	0.7575	0.7582	0.7570	0.7558	0.7585
ET	0.5941	0.5171	0.5585	**0.6006**	0.5878
LightGBM	0.4548	0.4533	0.4689	0.5116	**0.5529**
Logistic Regression	0.3468	0.3368	0.3497	0.3482	0.3537
Random Forest	0.5873	0.4937	0.5927	0.5393	0.5903
XGBoost	0.7533	0.7539	0.7533	0.7514	0.7571

Table 7.3: Mean AUPRC values by classifier and number of features (Part 2) for ten iterations of five-fold cross validation, for Part D Scenario Two

Features Classifier	15	20	25	30	82
CatBoost	**0.8016**	0.7953	0.7962	0.7949	0.7797
ET	0.4954	0.4647	0.4605	0.4391	0.3275
LightGBM	0.4447	0.4661	0.4603	0.4841	0.4982
Logistic Regression	**0.3669**	0.3536	0.3519	0.2939	0.3047
Random Forest	**0.6097**	0.5398	0.5519	0.5249	0.2429
XGBoost	**0.7889**	0.7448	0.7589	0.7548	0.7376

Part D Scenario 3: Feature Selection, then RUS 1:81

Table 7.4 contains the mean AUPRC scores for the same experiments where we perform feature selection, then sample the training data to induce a 1:81 class ratio. It is interesting to note that all classifiers trained on preprocessed data yield higher AUPRC scores when classifying data in the test set than classifiers trained on the original data. In both the Scenario two and Scenario Three results, we see better performance with fewer features. Models with fewer features are easier to explain because they are simpler, and there is a reduced chance for complex interactions.

Table 7.4: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for Part D Scenario Three

Features Classifier	10	15	20	25	30	82
CatBoost	0.7589	**0.8051**	0.7973	0.7984	0.7938	0.7798
ET	**0.4986**	0.4151	0.3933	0.4009	0.3668	0.2401
LightGBM	0.7070	**0.7400**	0.7100	0.7019	0.6937	0.6783
Logistic Regression	0.3117	**0.3189**	0.3117	0.3170	0.2486	0.2700
Random Forest	**0.4820**	0.4753	0.3983	0.4120	0.3545	0.2199
XGBoost	0.7473	**0.7860**	0.7588	0.7554	0.7491	0.7351

Part D Scenario Four: RUS 1:81 then feature selection

The last results we report are for experiments performed with another hybrid approach. The experiments in this scenario use RUS and feature selection in a different combination that in Scenario Three. In Scenario Four, data is sampled to a 1:81 level, then supervised feature selection is applied to rank features. A clear impact of the preprocessing treatment is apparent in terms of the AUPRC scores. In Table 7.5, we see that all classifiers exhibit

a response to the preprocessing treatment. Models trained on the preprocessed data yield higher scores than models trained on the original dataset.

Table 7.5: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for Part D Scenario Four

Features Classifier	10	15	20	25	30	82
CatBoost	0.7546	**0.7992**	0.7914	0.7965	0.7926	0.7798
ET	**0.4765**	0.4228	0.3857	0.3967	0.3639	0.2401
LightGBM	0.7073	**0.7268**	0.6971	0.6974	0.6857	0.6783
Logistic Regression	0.2609	**0.2785**	0.2358	0.2461	0.2613	0.2700
Random Forest	0.4209	**0.4639**	0.3311	0.3553	0.3392	0.2199
XGBoost	0.7471	**0.7743**	0.7550	0.7524	0.7476	0.7351

Part B Scenario One: RUS only

Here, we report the results of experiments in which we repeated the scenarios we conducted in experiments with the Part D data, with the Part B data. Table 7.6 holds the AUPRC scores that are the outcomes of experiments where we induce various class ratios in the training data. Here the impact of the treatment is not as clear, however, there is an improvement in the performance of LightGBM when trained on the preprocessed data over LightGBM when trained with data at its original class ratio.

Table 7.6: Mean AUPRC values by classifier and induced class ratio for ten iterations of five-fold cross validation, for Part B Scenario One

Ratio Classifier	1:1	1:3	1:9	1:27	1:81	1:2,500
CatBoost	0.4428	0.5320	0.6228	0.6569	0.6812	**0.6817**
ET	0.0125	0.0135	0.0184	0.0272	0.0336	**0.0433**
LightGBM	0.4001	0.4859	0.5563	**0.5967**	0.5766	0.4146
Logistic Regression	0.0058	0.0069	0.0076	0.0086	0.0099	**0.0103**
Random Forest	0.0791	0.1210	0.1596	0.1829	0.2017	**0.2462**
XGBoost	0.4240	0.5104	0.5783	0.6234	0.6536	**0.6886**

Part B Scenario Two: feature selection only

Table 7.7 contains the AUPRC scores that are the result of training models on the Part B data when it is preprocessed with feature selection. We find it is important to point out that all classifiers yield better performance when trained on fewer than all features.

Table 7.7: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for Part B Scenario Two

Features Classifier	10	15	20	25	30	80
CatBoost	0.6581	0.6792	**0.7069**	0.7009	0.7016	0.6817
ET	0.0400	0.0462	0.0443	**0.0524**	0.0424	0.0433
LightGBM	0.3939	0.3830	0.4261	**0.4589**	0.4293	0.4146
Logistic Regression	0.0093	0.0326	**0.0338**	0.0065	0.0064	0.0103
Random Forest	**0.4356**	0.3990	0.3736	0.3800	0.3395	0.2462
XGBoost	0.6611	0.6715	**0.6995**	0.6956	0.6955	0.6886

Part B Scenario Three: feature selection then RUS 1:81

Table 7.8 contains the AUPRC scores from the results of experiments in Scenario Three. In Scenario Three, Part B data that is preprocessed by feature selection, followed by RUS. The results in Table 7.8 show that all models respond well to being trained with data that is preprocessed with the Scenario Three approach. That is to say, the maximum scores each classifier yields is when the classifier is trained with the preprocessed data.

Table 7.8: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for Part B Scenario Three

Features Classifier	10	15	20	25	30	80
CatBoost	0.6600	0.6850	**0.7143**	0.7047	0.7023	0.6812
ET	0.0233	0.0261	0.0317	**0.0429**	0.0358	0.0336
LightGBM	0.5756	0.5904	**0.6185**	0.6009	0.6030	0.5766
Logistic Regression	0.0076	0.0205	**0.0217**	0.0074	0.0076	0.0099
Random Forest	0.2820	0.2758	0.3168	**0.3225**	0.2990	0.2017
XGBoost	0.6436	0.6535	**0.6798**	0.6708	0.6681	0.6536

Part B Scenario Four: RUS 1:81 then feature selection

Lastly for the Part B data, we review the AUPRC scores of classifiers trained on data which is first undersampled to the 1:81 class ratio, and then feature selection is applied. An inspection of these scores in Table 7.9 reveals that all models yield higher AUPRC scores in the classification of the training data when they are trained on data preprocessed with the Scenario Four technique. We find this result is noteworthy and should be considered for further analysis.

Table 7.9: Mean AUPRC values by classifier and number of features for ten iterations of five-fold cross validation, for Part B Scenario Four

Features / Classifier	10	15	20	25	30	80
CatBoost	0.6787	0.6725	**0.6994**	0.6978	0.6975	0.6812
ET	0.0289	0.0352	0.0495	**0.0512**	0.0462	0.0336
LightGBM	0.5968	0.5803	**0.6063**	0.5935	0.5938	0.5766
Logistic Regression	0.0078	0.0065	0.0067	0.0069	0.0090	**0.0099**
Random Forest	0.3313	0.3036	**0.3161**	0.3120	0.2892	0.2017
XGBoost	0.6560	0.6406	0.6644	0.6630	**0.6662**	0.6536

Statistical Analysis

In the Statistical analysis that follows, we first present an analysis of results for scenarios three and four separately to determine which levels of experimental factors yield the best performance. We forgo the analyses for scenarios one and two since these are applications of RUS and feature selection separately. Chapters five and six cover the application of RUS or feature selection separately, so we felt it would be redundant to repeat similar analyses here. For the Part D data, ANOVA and HSD tests revealed that models trained on data with RUS to induce a 1:81 class ratio yield the best performance, and models built SFS applied to select the top 10 features also yield the best performance. After presenting the analysis for scenarios one and two, we do a combined analysis of the outcomes of each scenario to determine which scenario(s) yield the best performance. We coin the term "inter-scenario analysis" for the combined analysis.

Part D Scenario 3: Feature Selection, then RUS 1:81

Now we move on to the statistical analysis of the Scenario Three experiments with the Part D data, where we do feature selection, then apply RUS to induce a 1:81 class ratio in the training data. The ANOVA test result in Table 7.10 is similar to the results for Scenarios One and Two in that the Pr(>) F values both factors is practically zero. Since we use only one sampling ratio in Scenario Three, only the choice of classifier and number of features are treated as factors in the ANOVA test.

Table 7.10: ANOVA for Features and Classifier as factors of performance in terms of AUPRC, for Part D Scenario Three

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	2.17	0.43	178.58	*
Classifier	5	72.01	14.40	5933.71	*
Residuals	1789	4.34	0.00		

* indicates the value is less than 1×10^{-4}

Since the ANOVA test shows that both the choice of classifier, and the number of features selected have a significant impact on AUPRC scores, we do the HSD tests to determine which levels of the factors yield the best performance. Table 7.11 contains the HSD result for the number of features factor. In it we see that, similar to the result in Scenario Two, models built with ten features are in the group that yields the best performance. This is a noteworthy result, since it demonstrates that a data reduction technique can yield better performance. In the results of both Scenario Two and Scenario Three, models with fewer features demonstrated superior performance. Such models are more straightforward, making them easier to interpret due to their inherent simplicity and decreased potential for intricate interactions.

Table 7.11: HSD test groupings after ANOVA of AUPRC for the Features factor, for Part D Scenario Three

Group a consists of: 15, 10
Group b consists of: 25, 20
Group c consists of: 30
Group d consists of: 82

Next, we move on to the HSD result for the classifier factor. Here we see that the three GBDT techniques, CatBoost, XGBoost, and LightGBM yield the best performance. It is interesting to note that this is similar to the HSD test result for the classifier factor in Scenario One, and that in the experiments in Scenario One and Scenario Three, RUS is applied to the training data.

Table 7.12: HSD test groupings after ANOVA of AUPRC for the Classifier factor, for Part D Scenario Three

Group a consists of: CatBoost
Group b consists of: XGBoost
Group c consists of: LightGBM
Group d consists of: Random Forest, ET
Group e consists of: Logistic Regression

Part D Scenario Four: RUS 1:81 then feature selection

Finally, we come to the last scenario for the experiments with Part D data. Table 7.13 contains the result of the ANOVA test for the Scenario Four experiments. Similar to

Scenario Three, we only use one level of undersampling, 1:81. Therefore, we treat the number of features, and the choice of classifiers as experimental factors. The Pr(>F) values indicate that both factors have a significant effect on AUPRC scores.

Table 7.13: ANOVA for Features and Classifier as factors of performance in terms of AUPRC, for Part D Scenario Four

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	1.43	0.29	127.51	*
Classifier	5	80.72	16.14	7209.10	*
Residuals	1789	4.01	0.00		

* indicates the value is less than 1×10^{-4}

Since the ANOVA test for Scenario Four indicates that the number of features used to train the model has a significant effect on experimental outcomes, we use the HSD test to determine which number of features yields the best performance. For the Scenario Four experiments, we see in Table 7.14, that 15 features yields the best performance.

Table 7.14: HSD test groupings after ANOVA of AUPRC for the Features factor for, Part D Scenario Four

Group a consists of: 15
Group b consists of: 10
Group c consists of: 25, 20, 30
Group d consists of: 82

The HSD test for experiments with Part D data is documented in Table 7.15. Again, we have the result that CatBoost yields the best performance, and the three GBDT techniques yield the top three mean AUPRC scores.

Table 7.15: HSD test groupings after ANOVA of AUPRC for the Classifier factor, for Part D Scenario Four

Group a consists of: CatBoost
Group b consists of: XGBoost
Group c consists of: LightGBM
Group d consists of: ET
Group e consists of: Random Forest
Group f consists of: Logistic Regression

PART D, INTER-SCENARIO ANALYSIS

Now that we have documented the outcomes of the individual scenarios, we do a further analysis to determine which scenario(s) yield the best performance over all. To get started we take note of which levels of factors yield the best performance in a scenario. We then select data from experiments to include for this analysis based on which experiments included factors set at the levels that yield the best performance. For Scenario One, RUS only, we select RUS 1:81. For Scenario Two, feature selection only, we select experiments where ten features are used. For Scenario Three, feature selection, then RUS, we select experiments where 10 features are used, and RUS 1:81 is used. For Scenario Four, RUS, then feature selection, we select experiments where RUS 1:81 is used, and 10 features are used. We then treat each scenario, and the classifier as an experimental factor and conduct an ANOVA test, followed by HSD tests to determine which scenario(s) yield the best performance, and which classifiers yield the best performance. We consider the case where no preprocessing is done to the data to be the fifth scenario. Table 7.16 summarizes the levels of factors used in all scenarios.

Table 7.16: Summary of Part D Scenarios, and optimal levels of features selected from each scenario

Scenario	**Description**
Scenario One	RUS only: RUS 1:81 selected
Scenario Two	Feature selection only: 10 features selected
Scenario Three	Feature selection, then RUS: 10 features and RUS 1:81 selected
Scenario Four	RUS, then feature selection: RUS 1:81 and 10 features selected
Scenario Five	No preprocessing done on the data

Table 7.17 contains the result of the ANOVA test for the scenario and classifier factors. The Pr(>F) values indicate that both the scenario, and the classifier have a significant impact on experimental outcomes.

Table 7.17: ANOVA for Scenario and Classifier as factors of performance in terms of AUPRC, for Part D Experiments

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Scenario	4	3.65	0.91	125.08	*
Classifier	5	48.41	9.68	1328.40	*
Residuals	1490	10.86	0.01		

* indicates the value is less than 1×10^{-4}

The ANOVA test result in Table 7.17 indicates that the scenario has a significant impact on experimental outcomes. Therefore, we can conduct a Tukey HSD test to determine which scenario yields the highest AUPRC scores. The HSD test result in Table 7.18 indicates that either feature selection, or feature selection followed by RUS yield the best performance.

Table 7.18: HSD test groupings after ANOVA of AUPRC for the Scenario factor in Part D Experiments

Group a consists of: Scenario Two, Scenario Three
Group b consists of: Scenario Four
Group c consists of: Scenario One, Scenario Five

Table 7.19 contains the HSD test result for the classifier factor. Since we find that for the majority of the scenarios individually that the GBDT classifiers yield the best performance, it is not surprising that the three GBDT classifiers are members of the best-performing groups.

Table 7.19: HSD test groupings after ANOVA of AUPRC for the Classifier factor in Part D Experiments

Group a consists of: CatBoost, XGBoost
Group b consists of: LightGBM
Group c consists of: ET
Group d consists of: Random Forest
Group e consists of: Logistic Regression

Part B Statistical Analysis

Here we report individual scenario results for the experiments with Part B data. We do the analysis of scenarios in the same order for experiments with Part B data as we do for the Part D data. For the same reasons to avoid redundancy with Chapters five and six, here we report the analysis for scenarios three and four only.

Part B Scenario Three, Feature Selection, Then RUS

We continue to the first hybrid approach for experiments with Part B data, where we apply feature selection, then RUS. Although we apply RUS to make the class ratio 1:81 in Scenario Three, RUS is not an experimental factor since it does not change throughout the course of the Part B Scenario Three experiments. Since we build models with different numbers of features and different classifiers, these are experimental factors. The ANOVA test results in Table 7.20 show that both the choice of classifier and the number of features selected have a significant impact on experimental outcomes.

Table 7.20: ANOVA for Features and Classifier as factors of performance in terms of AUPRC, for Part B Scenario Three

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	0.34	0.07	61.20	*
Classifier	5	146.24	29.25	26678.81	*
Residuals	1789	1.96	0.00		

* indicates the value is less than 1×10^{-4}

Since the ANOVA test result in Table 7.20 shows that the number of features selected has a significant impact on the AUPRC scores recorded in the Part B Scenario Three experiments, we perform an HSD test to determine which number of features can be used to build models that yield the best performance. The HSD test result in Table 7.21 indicates that models built with 20 features yield the best performance, since this set of experiments is in the HSD group 'a'.

Table 7.21: HSD test groupings after ANOVA of AUPRC for the Features factor, for Part B Scenario Three

Group a consists of: 20
Group ab consists of: 25
Group b consists of: 30
Group c consists of: 15
Group d consists of: 10, 80

Next, we turn to the result of the HSD test for the effect of the classifier on experimental outcomes in the Part B Scenario Three experiments. The result confirms the pattern we have seen deviated from only once. That pattern is that the GBDT methods yield the best AUPRC scores, with CatBoost yielding the best AUPRC scores.

Table 7.22: HSD test groupings after ANOVA of AUPRC for the Classifier factor, for Part B Scenario Three

Group a consists of: CatBoost
Group b consists of: XGBoost
Group c consists of: LightGBM
Group d consists of: Random Forest
Group e consists of: ET
Group f consists of: Logistic Regression

Part B, Scenario Four, RUS 1:81, then Feature Selection

Here, we proceed to the analysis of the last of the Scenario Four experimental results with Part B data. This is the scenario where we apply RUS to induce a class ratio of 1:81 prior to doing feature selection.

Following our procedure for statistical analysis, we conduct an ANOVA test to determine which experimental factors have a significant effect on experimental outcomes. In Table 7.20, we confirm that both the choice of classifier and the number of features used in the experiment have a significant impact on experimental outcomes. Although RUS is employed prior to feature selection, it is not an experimental factor since we only apply RUS to induce a single class ratio.

Table 7.23: ANOVA for Features and Classifier as factors of performance in terms of AUPRC, for Part B Scenario Four

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Features	5	0.20	0.04	33.72	*
Classifier	5	143.27	28.65	24577.10	*
Residuals	1789	2.09	0.00		

* indicates the value is less than 1×10^{-4}

The ANOVA test result in Table 7.23 indicates that the number of features used has a significant impact on experimental outcomes, so we conduct an HSD test. The test result is in Table 7.24. The test result indicates that RUS followed by feature selection with any number of features, except 15, yields the best performance, and also that applying RUS and then feature selection yields better performance than not applying feature selection.

Table 7.24: HSD test groupings after ANOVA of AUPRC for the Features factor, for Part B Scenario Four

Group a consists of: 20, 25, 30, 10
Group b consists of: 15
Group c consists of: 80

The second HSD test we conduct for the Part B Scenario Four experiments is for the effect of the classifier on experimental outcomes. In Table 7.25 we confirm the clear pattern in the HSD results we have observed in all but one scenario, and that is that the GBDT techniques outperform all others, and CatBoost yields the best performance.

Table 7.25: HSD test groupings after ANOVA of AUPRC for the Classifier factor, for Part B Scenario Four

Group a consists of: CatBoost
Group b consists of: XGBoost
Group c consists of: LightGBM
Group d consists of: Random Forest
Group e consists of: ET
Group f consists of: Logistic Regression

PART B INTER-SCENARIO ANALYSIS

Here we begin the analysis of results between scenarios. We do the same inter-scenario analysis for the Part B scenarios that we did for the Part D scenarios. As stated previously in the context of the Part D data inter-scenario analysis, though we report results for four scenarios, there is a latent fifth scenario, which is the case where do not do any preprocessing

to the data. Also, similar to the Part D inter-scenario analysis, we select the levels of factors that yield the best performance in each scenario. Table 7.26 contains a summary of the levels of factors selected for each scenario.

Table 7.26: Summary of Part B Scenarios, and optimal levels of features selected from each scenario

Scenario	**Description**
Scenario One	RUS only: RUS 1:81 selected
Scenario Two	Feature selection only: 10 features selected
Scenario Three	Feature selection, then RUS: 20 features and RUS 1:81 selected
Scenario Four	RUS, then feature selection: RUS 1:81 and 10 features selected
Scenario Five	No preprocessing done on the data

First, we conduct an ANOVA test to confirm that the scenario and the choice of classifier have a significant impact on experimental outcomes. The Pr(>F) values for both the scenario and classifier factors in Table 7.27 imply that both factors have a significant impact on AUPRC values.

Table 7.27: ANOVA for Scenario and Classifier as factors of performance in terms of AUPRC, for Part B Experiments

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Scenario	4	0.46	0.12	28.70	*
Classifier	5	113.33	22.67	5621.43	*
Residuals	1490	6.01	0.00		

* indicates the value is less than 1×10^{-4}

Since we have confirmed that both the scenario and the classifier have a significant impact on experimental outcomes, we conduct an HSD test to determine which levels of

the factors yield the best performance. The HSD test result in Table 7.28 implies that the two hybrid approaches yield the best performance. This is noteworthy since it means the two techniques that yield the largest data reduction also yield the strongest performance.

Table 7.28: HSD test groupings after ANOVA of AUPRC for the Scenario factor, for Part B Experiments

Group a consists of: Scenario Three, Scenario Four
Group b consists of: Scenario Two
Group bc consists of: Scenario One
Group c consists of: Scenario Five

Finally, we take a look at the classifiers that perform the best over all scenarios. Here, we find CatBoost and XGBoost yield the best performance, followed by LightGBM. The consistent pattern we find in the results for the individual scenarios in Part B carries over into the results for the Part B inter-scenario analysis.

Table 7.29: HSD test groupings after ANOVA of AUPRC for the Classifier factor, for Part B Experiments

Group a consists of: CatBoost, XGBoost
Group b consists of: LightGBM
Group c consists of: Random Forest
Group d consists of: ET
Group e consists of: Logistic Regression

Conclusions from Experiments with Hybrid Techniques

We presented an in-depth statistical analysis of the experimental outcomes in terms of AUPRC for results for Medicare insurance fraud detection in Big Medicare Data datasets. For both Medicare Part B and Part D datasets, we carry out experiments in five scenarios that exhaust the possible ways to utilize, or omit, the RUS and feature selection data reduction techniques. For both datasets, we found that data reduction techniques also improve classification results. We also found that the three GBDT classifiers, LightGBM, XGBoost, and CatBoost yield the best performance in all experiments, with one exception. In any case, CatBoost consistently yields the best performance.

In experiments with the Part D data, we conducted separate statistical analyses of the four scenarios where at least one data reduction technique was used, to determine which technique yields the best performance in each scenario. We then conducted a statistical analysis of results between all scenarios. The result of the analysis shows that our supervised feature selection technique alone, or our feature selection technique, followed by RUS, yields the best performance.

We performed a similar statistical analysis for experiments involving the Medicare Part B data. After doing the analysis of the individual scenarios, we again selected the techniques which yielded the best results, and performed an analysis of results between all scenarios. We find that either combination of using our feature selection technique, followed by RUS, or RUS followed by our feature selection technique both yield the best performance.

Therefore, in the classification of either dataset, we find that a technique with the largest amount of data reduction also yields the best performance. That is the technique of doing feature selection, then applying RUS. The same statistical analysis shows that the GBDT classification techniques, XGBoost, CatBoost and LightGBM outperform the other three learners we use in our experiments. These other learners are Logistic Regression, Random Forest, and Extremely Randomized trees. An added benefit of our feature selection technique is model explainability. It is easier to reason about how a model performs

classifications when it is built with fewer features. Overall, the key conclusion one should draw from our results is that intelligent data reduction techniques, applied in combination, may improve the results in classifying highly imbalanced, Big Data.

7.4 CHAPTER SUMMARY

In this chapter, we cover the hybrid data reduction techniques of combining RUS and SFS. We report results from an initial study [169], where we found an indication that a hybrid technique could yield better results than the data reduction techniques applied individually. We show how the initial study was expanded [62], and an exhaustive investigation into methods of combining RUS and SFS was undertaken to find which technique yields the best performance. Looking at results of all studies covered in this chapter, the key take-away is that SFS, followed by RUS yields the best performance. Since both SFS and RUS are data reduction techniques, the results of studies documented in this chapter show that data reduction is worthwhile since it yields improved performance and faster training times.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 CONCLUSIONS

Throughout this dissertation, we have explored various data reduction techniques to address the challenges posed by highly imbalanced Big Data in the context of fraud detection. The primary focus has been on credit card transactions and Medicare insurance claims, two domains that are particularly susceptible to fraudulent activities. By leveraging machine learning algorithms and developing novel data reduction approaches, this research aims to contribute to the ongoing efforts in combating fraud and enhancing the integrity of financial and healthcare systems. In this chapter, we present conclusions drawn as the result of the research covered in previous chapters.

8.1.1 Datasets and Compilation

The second chapter delves into the datasets utilized in this research, focusing on the challenges posed by highly imbalanced data in fraud detection. The datasets include the Credit Card Fraud dataset and several Medicare datasets (Part D, Part B, DMEPOS, both aggregated and non-aggregated versions), highlighting the extreme imbalance present in such datasets where fraudulent transactions or claims constitute less than 1% of the data.

The Credit Card Fraud dataset, sourced from a collaboration between the machine learning group at Université Libre de Bruxelles and Worldline, features transactions with 29 attributes derived from PCA, making most of the features abstract. The Medicare datasets, compiled from CMS and OIG sources, cover various aspects of Medicare programs. The discussion in this part of the dissertation covers the magnitude of the data, and the complexity

involved in preprocessing and labeling to prepare the data for machine learning applications.

We feel it is important to focus on the datasets used in this dissertation, and how they are compiled. Therefore, we provide an overview of the datasets in this chapter, highlighting their imbalanced nature and the preprocessing required for the experiments. The conclusions researchers can draw from our coverage of the datasets, and their compilation process, lead to further research. The process we report here can be used to compile new Medicare datasets as data becomes available. Moreover, we come to the conclusion that the same general technique for combining cross-agency government data may be applied in other application domains. The technique is joining data by a common identifier, published in different sources of data from separate government entities. If data from one source can be characterized as explaining data from the other source, then these could be good candidates for raw data for a Machine Learning dataset.

The documentation of the data compilation process, feature engineering, and the application of CatBoost encoding for categorical features laid the foundation for the subsequent chapters. The second chapter also emphasized the importance of handling unique identifiers and high-cardinality categorical features to prevent models from memorizing patterns and failing to generalize. The detailed description of the datasets and the preprocessing steps ensures the reproducibility of the experiments and provides a clear understanding of the data characteristics.

8.1.2 Learners

This third chapter introduces the machine learning algorithms employed, including ensemble methods like Random Forest and Extremely Randomized Trees, Logistic Regression, and one-class classifiers like One-Class SVM and One-Class GMM. The discussion extends to the calibration and training of these one-class classifiers on majority versus minority classes. The chapter discusses the nuances of choosing and training algorithms in the context of imbalanced data sets.

In Chapter three, we also provide background information on the families of Machine Learning algorithms employed in this dissertation, including ensemble boosting methods such as Random Forest, Extremely Randomized Trees, and Gradient Boosting methods such as LightGBM, XGBoost and CatBoost. The strengths and weaknesses of each algorithm in the context of fraud detection and imbalanced data were discussed. The conclusion one may draw from our discussions on learners is that it is possible to combine decision-trees in two different ways to build ensemble classifiers. One way results in the Boosting family of classifiers, and the second way results in the Bagging family of classifiers. A further conclusion is that within the families of classifiers, researchers have discovered novel means of optimizing the ensemble techniques. Furthermore, the chapter introduced the evaluation metrics, Area Under the Receiver Operating Characteristic Curve (AUC) and Area Under the Precision-Recall Curve (AUPRC), which were used to assess the performance of these algorithms.

The comparative analysis of the circumstances where Binary-Class Classifiers (BCCs) versus One-Class Classifiers (OCCs) should be used provides insight into their suitability for different scenarios. An extremely important conclusion one should draw from the chapter is that OCCs are appropriate when only data from one class is available. The chapter also highlighted the importance of hyperparameter tuning and the selection of appropriate evaluation metrics for imbalanced classification tasks. The discussion on the calibration of OCC output probabilities further emphasized the need for careful consideration when working with these classifiers. This is another important conclusion to draw from Chapter three: some models' output values are not class membership probabilities, and require calibration to convert the output values to probabilities before use in calculation of values such as AUPRC.

8.1.3 Experimental Methodology

The methodology chapter outlines the experimental framework, including cross-validation, how performance metrics are calculated during cross-validation, and how statistical tests like ANOVA and Tukey's HSD test are conducted. This rigorous approach ensures a comprehensive evaluation of the proposed techniques, addressing the challenges of imbalanced data and high dimensionality.

Hence, in Chapter four we present how the elements of cross-validation, performance metrics, and statistical analysis come together to evaluate the effectiveness of the proposed data reduction techniques. The rigorous framework ensures the reliability of the findings, and allows for objective decision-making regarding the impact of various factors on experimental outcomes. The chapter also provides a detailed explanation of the AUPRC metric, highlighting its superiority over AUC in capturing the impact of data reduction techniques on the classification of highly imbalanced Big Data. The use of stratified k-fold cross-validation and multiple iterations of the experiments demonstrates the robustness of the methodology, and the reliability of the results. The inclusion of Analysis of Variance (ANOVA) and Tukey's Honestly Significant Difference (HSD) tests further strengthens the methodology. The ANOVA tests show whether the experimental factors have a statistically significant impact on experimental outcomes, and the HSD tests provide a clear understanding of how the experimental factors can be put into groups having similar effects on experimental outcomes.

The principal conclusion that we make from the material presented in Chapter four is that there is an experimental methodology one may apply that works well for evaluating the impact of data reduction techniques for highly imbalanced Big Data. We present the template used to make decisions about which experimental factors yield the best, or worst results. This template can be used in Machine Learning experiments over a wide range of application domains.

8.1.4 Sampling Techniques

In Chapter four, sampling techniques are explored in depth, with a focus on Random Undersampling (RUS) and One-Class Sampling. The chapter provides a critical analysis of the impact of these techniques on classifier performance, particularly in the context of Medicare data, illustrating the benefits and detriments we observed from employing the sampling strategies.

The investigation of these techniques on various classifiers reveals that RUS has a negative impact on the classification of Medicare datasets, which is not reflected in AUC scores but is evident in AUPRC scores. This finding emphasizes the importance of using AUPRC as a more informative metric when dealing with highly imbalanced Big Data.

Additionally, the chapter introduces a novel one-class sampling technique tailored for OCCs, demonstrating that reducing the training dataset to as little as 20% of its original size does not significantly compromise the performance of One-Class Gaussian Mixture Models (GMMs). The extensive experiments conducted on the Medicare Part D and Part B datasets, along with the statistical analysis, provide strong evidence for the effectiveness of the proposed one-class sampling technique. The chapter also highlights the potential of this technique in accelerating the pace of experimentation with OCCs on large-scale datasets.

The most important inferences one should make from this chapter are that AUC can sometimes hide the effect of a factor on experimental outcomes. We provide evidence of how the effect of RUS on experimental outcomes is apparent in terms of AUPRC, but not AUC. The second inference is that data sampling should be employed in conjunction with use of One-Class GMM.

8.1.5 Feature Selection Techniques

In our chapter on feature selection techniques, we examine ensemble methods and SHAP for their effectiveness in reducing dimensionality, while retaining informative features for fraud detection. The chapter presents a thorough analysis of feature selection's role in enhancing

model interpretability and performance.

We use these selection techniques, to identify the most informative features for fraud detection. The chapter demonstrates that the Supervised Feature Selection (SFS) technique outperforms Threshold-Based Feature Selection (TBFS) in the credit card fraud detection domain. Furthermore, the application of SHAP as a feature selection technique for OCCs was investigated. Our investigation shows that SHAP can effectively identify informative features, and yield similar or better performance when used for building models with One-Class GMMs. We found this in both credit card and Medicare fraud detection tasks. The chapter also highlights the potential of feature selection techniques in enhancing model interpretability and reducing computational complexity. The detailed analysis of the selected features provided valuable insights into the factors contributing to fraudulent behavior and demonstrates the effectiveness of the proposed techniques in building explainable models. The comparative analysis of features selected by SHAP in different label availability scenarios further emphasizes the versatility and robustness of the approach.

The outcome of research covered in Chapter six is that feature selection serves many purposes. As a data reduction technique, it can improve the performance of models both in terms of training time and performance metrics such as AUPRC. Moreover, it can be used as an aid for model explainability. SHAP is especially useful when used comparatively with multiple models in analysis of the Credit Card data, which is difficult to do in other ways since most of the features of the Credit Card data are defined as the result of applying Principal Components Analysis to some other data. Perhaps most significantly, feature selection is shown to be an effective data reduction technique for working with Big Data, and therefore can make some classification tasks feasible when they would be otherwise infeasible due to the size of the data.

8.1.6 Hybrid Techniques

In the context of this dissertation, hybrid techniques are techniques that combine feature selection and sampling methods. They are presented as a means to further improve fraud detection models. In Chapter seven, the synergistic effects of these combined approaches are analyzed, demonstrating their potential in addressing the complexities of imbalanced Big Data.

In this chapter, we show how the hybrid techniques further enhance the performance of fraud detection models. The exhaustive investigation of various scenarios of combinations of RUS and SFS reveal that applying SFS followed by RUS yields the best performance in terms of AUPRC scores. This finding underscores the effectiveness of data reduction techniques in improving classification results and reducing training times for highly imbalanced Big Data. The chapter also provides a comprehensive statistical analysis of the experimental outcomes, confirming the superiority of the hybrid approach over individual techniques. Additionally, there is an inter-scenario analysis, which further solidifies the conclusions and provides a clear understanding of the optimal levels of factors for each scenario. The consistently strong performance of Gradient Boosted Decision Tree classifiers, particularly CatBoost, across all scenarios highlights their suitability for fraud detection tasks in highly imbalanced Big Data.

We position the research covered in Chapter seven last since it is the culmination of the research conducted in previous chapters. The verdict of our chapter on hybrid techniques is that one may employ both sampling and feature selection for substantial data reduction in Big Medicare Data fraud detection tasks. This results in accelerated model training and enables a faster pace of research.

8.2 FUTURE WORK

In this final section, we propose directions for future research, emphasizing the need for continued exploration of data reduction techniques, machine learning algorithms, and

their application to new domains. Potential areas include the integration of deep learning approaches, the expanded application of our sampling and novel feature selection methods, and the extension of the current framework to other forms of fraud detection beyond the financial and healthcare sectors.

The research presented in this dissertation opens up several avenues for future work. One potential direction is to explore the applicability of the proposed data reduction techniques to other domains facing similar challenges with imbalanced Big Data, such as intrusion detection, anomaly detection in sensor networks, and rare event prediction in various industries. Adapting and refining these techniques for domain-specific requirements could lead to the development of more effective and efficient solutions. For example, investigating the performance of the hybrid SFS-RUS approach in network intrusion detection or industrial equipment failure prediction could yield valuable insights and potentially lead to the creation of more robust and reliable systems in these domains.

Another area for future research is the investigation of other advanced machine learning algorithms, such as deep learning architectures in conjunction with the proposed data reduction techniques [25, 26]. Exploring the synergies between these algorithms and data reduction methods could potentially yield further improvements in fraud detection performance and scalability.

Furthermore, the interpretability of fraud detection models is a crucial aspect that warrants further attention. Developing more sophisticated techniques for feature analysis and model explanation, building upon the foundation laid by the SHAP-based approach presented in this dissertation, could enhance the transparency and trustworthiness of fraud detection systems. This is particularly important in the context of regulatory compliance and user acceptance. Future research could focus on integrating the proposed data reduction techniques with state-of-the-art explainable methods to provide more comprehensive and intuitive explanations for the decision-making process of fraud detection models. This could not only improve the interpretability of the models but also facilitate the identification

of potential biases and ensure fairness in the detection process.

In summary, the research presented in this dissertation has laid a solid foundation for the development of effective data reduction techniques for fraud detection in highly imbalanced Big Data. The future work outlined above has the potential to build upon these contributions, further advancing the state-of-the-art in this critical domain and contributing to the creation of more secure and trustworthy financial and healthcare systems. By exploring the applicability of the proposed techniques to other domains, investigating their integration with advanced machine learning algorithms, enhancing model interpretability, and extending them to real-time scenarios, future research can significantly expand the impact and practical utility of this work. Ultimately, the continued development and refinement of data reduction techniques for imbalanced Big Data will play a crucial role in combating fraud and ensuring the integrity of various industries in the face of evolving threats and challenges.

This dissertation makes significant contributions to the field of fraud detection using machine learning data reduction techniques in the context of highly imbalanced Big Data. The proposed data reduction techniques, including sampling, feature selection, and hybrid approaches, have demonstrated their effectiveness in improving the performance of fraud detection models while reducing computational complexity and enhancing model interpretability. The insights gained from this research have the potential to significantly advance the development of more robust and efficient fraud detection systems in the financial and healthcare domains. The rigorous experimental methodology, extensive statistical analysis, and comprehensive discussion of the results ensure the reliability and reproducibility of the findings. The dissertation also provides valuable guidance for practitioners and researchers working with highly imbalanced Big Data, emphasizing the importance of selecting appropriate evaluation metrics, employing effective data reduction techniques, and considering the specific requirements of the application domain.

BIBLIOGRAPHY

[1] R. A. Bauder, R. da Rosa, and T. M. Khoshgoftaar. Identifying medicare provider fraud with unsupervised machine learning. In *2018 IEEE international conference on information Reuse and integration (IRI)*, pages 285–292. IEEE, 2018.

[2] R. A. Bauder, M. Herland, and T. M. Khoshgoftaar. Evaluating model predictive performance: A medicare fraud detection case study. In *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 9–14. IEEE, 2019.

[3] R. A. Bauder and T. M. Khoshgoftaar. A probabilistic programming approach for outlier detection in healthcare claims. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 347–354. IEEE, 2016.

[4] R. A. Bauder and T. M. Khoshgoftaar. Medicare fraud detection using machine learning methods. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 858–865, Dec 2017.

[5] R. A. Bauder and T. M. Khoshgoftaar. Multivariate anomaly detection in medicare using model residuals and probabilistic programming. In *The Thirtieth International Flairs Conference*, 2017.

[6] R. A. Bauder and T. M. Khoshgoftaar. Multivariate outlier detection in medicare claims payments applying probabilistic programming methods. *Health Services and Outcomes Research Methodology*, 17:256–289, 2017.

[7] R. A. Bauder and T. M. Khoshgoftaar. The detection of medicare fraud using machine learning methods with excluded provider labels. In *The Thirty-First International Flairs Conference*, 2018.

[8] R. A. Bauder and T. M. Khoshgoftaar. The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data. *Health information science and systems*, 6:1–14, 2018.

[9] R. A. Bauder and T. M. Khoshgoftaar. Medicare fraud detection using random forest with class imbalanced big data. In *2018 IEEE international conference on information reuse and integration (IRI)*, pages 80–87. IEEE, 2018.

[10] R. A. Bauder and T. M. Khoshgoftaar. Medicare fraud detection using random forest with class imbalanced big data. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 80–87, July 2018.

[11] R. A. Bauder and T. M. Khoshgoftaar. A survey of medicare data processing and integration for fraud detection. In *2018 IEEE international conference on information reuse and integration (IRI)*, pages 9–14. IEEE, 2018.

[12] R. A. Bauder and T. M. Khoshgoftaar. Big data and class imbalance in medicare fraud detection. pages 62–86, 2020.

[13] R. A. Bauder and T. M. Khoshgoftaar. A study on rare fraud predictions with big medicare claims fraud data. *Intelligent Data Analysis*, 24(1):141–161, 2020.

[14] R. A. Bauder, T. M. Khoshgoftaar, and T. Hasanin. Data sampling approaches with severely imbalanced big data for medicare fraud detection. *2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI)*, pages 137–142, 2018.

[15] R. A. Bauder, T. M. Khoshgoftaar, and T. Hasanin. An empirical study on class rarity in big data. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 785–790. IEEE, 2018.

[16] R. A. Bauder, T. M. Khoshgoftaar, and A. Napolitano. Fraud detection with a limited number of known fraudulent medicare providers. In *The Thirty-First International Flairs Conference*, 2018.

[17] R. A. Bauder, T. M. Khoshgoftaar, A. Richter, and M. Herland. Predicting medical provider specialties to detect anomalous insurance claims. In *2016 IEEE 28th international conference on tools with artificial intelligence (ICTAI)*, pages 784–790. IEEE, 2016.

[18] R. A. Bauder, T. M. Khoshgoftaar, and N. Seliya. A survey on the state of healthcare upcoding fraud analysis and detection. *Health Services and Outcomes Research Methodology*, 17:31–55, 2017.

[19] M. Bekkar, H. K. Djemaa, and T. A. Alitouche. Evaluation measures for models assessment over imbalanced data sets. *J Inf Eng Appl*, 3(10), 2013.

[20] K. Boyd, K. H. Eng, and C. D. Page. Area under the precision-recall curve: point estimates and confidence intervals. *Joint European conference on machine learning and knowledge discovery in databases*, pages 451–466, 2013.

[21] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[22] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[23] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[24] C. L. Calvert and T. M. Khoshgoftaar. Threshold based optimization of performance metrics with severely imbalanced big security data. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1328–1334. IEEE, 2019.

[25] G. Castaneda, P. Morris, and T. M. Khoshgoftaar. Evaluation of maxout activations in deep learning across several big data domains. *Journal of Big Data*, 6:1–35, 2019.

[26] G. Castaneda, P. Morris, and T. M. Khoshgoftaar. Maxout neural network for big data medical fraud detection. In *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 357–362. IEEE, 2019.

[27] Centers for Medicare and Medicaid Services. 2019 estimated improper payment rates for centers for medicare & medicaid services (cms) programs. https://www.cms.gov/newsroom/fact-sheets/2019-estimated-improper-payment-rates-centers-medicare-medicaid-services-cms-programs, 2019.

[28] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016.

[29] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.

[30] R. Chiramdasu, G. Srivastava, S. Bhattacharya, P. K. Reddy, and T. Reddy Gadekallu. Malicious url detection using logistic regression. In *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*, pages 1–6, 2021.

[31] Civil Division, U.S. Department of Justice. Fraud statistics, overview. https://www.justice.gov/opa/press-release/file/1354316/download, 2020.

[32] A. De Mauro, M. Greco, and M. Grimaldi. A formal definition of big data based on its essential features. *Library Review*, 2016.

[33] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

[34] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[35] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[36] Q. Gu, Z. Cai, L. Zhu, and B. Huang. Data mining on imbalanced data sets. In *2008 International Conference on advanced computer theory and engineering*, pages 1020–1024. IEEE, 2008.

[37] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[38] J. T. Hancock, R. A. Bauder, and T. M. Khoshgoftaar. Shap as a data reduction technique for highly imbalanced big data. *International Journal on Artificial Intelligence Tools*. (status: under review).

[39] J. T. Hancock, R. A. Bauder, and T. M. Khoshgoftaar. A model-agnostic feature selection technique to improve the performance of one-class classifiers. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2023.

[40] J. T. Hancock, R. A. Bauder, and T. M. Khoshgoftaar. Shap as a data reduction technique for highly imbalanced big data. *International Journal on Artificial Intelligence Tools*, 2024. (status: under review).

[41] J. T. Hancock, R. A. Bauder, H. Wang, and T. M. Khoshgoftaar. Explainable machine learning models for medicare fraud detection. *Journal of Big Data*, 10(1):154, 2023.

[42] J. T. Hancock and T. M. Khoshgoftaar. Catboost for big data: an interdisciplinary review. *Journal of big data*, 7(1):1–45, 2020.

[43] J. T. Hancock and T. M. Khoshgoftaar. Medicare fraud detection using catboost. In *2020 IEEE 21st international conference on information reuse and integration for data science (IRI)*, pages 97–103. IEEE, 2020.

[44] J. T. Hancock and T. M. Khoshgoftaar. Performance of catboost and xgboost in medicare fraud detection. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 572–579. IEEE, 2020.

[45] J. T. Hancock and T. M. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7:1–41, 2020.

[46] J. T. Hancock and T. M. Khoshgoftaar. Gradient boosted decision tree algorithms for medicare fraud detection. *SN Computer Science*, 2(4):1–12, 2021.

[47] J. T. Hancock and T. M. Khoshgoftaar. Impact of hyperparameter tuning in classifying highly imbalanced big data. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 348–354. IEEE, 2021.

[48] J. T. Hancock and T. M. Khoshgoftaar. Leveraging lightgbm for categorical big data. In *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 149–154. IEEE, 2021.

[49] J. T. Hancock and T. M. Khoshgoftaar. Hyperparameter tuning for medicare fraud detection in big data. *SN Computer Science*, 3(6):1–13, 2022.

[50] J. T. Hancock and T. M. Khoshgoftaar. Optimizing ensemble trees for big data healthcare fraud detection. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 243–249. IEEE, 2022.

[51] J. T. Hancock and T. M. Khoshgoftaar. Data reduction to improve the performance of one-class classifiers on highly imbalanced big data. In *2023 22nd IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023.

[52] J. T. Hancock and T. M. Khoshgoftaar. Exploring maximum tree depth and random undersampling in ensemble trees to optimize the classification of imbalanced big data. *SN Computer Science*, 4(5):462, 2023.

[53] J. T. Hancock and T. M. Khoshgoftaar. A novel data reduction technique for medicare fraud detection with one-class classifiers. *Journal of Reliability Quality and Safety Engineering*, 2024. (status: book chapter under review).

[54] J. T. Hancock, T. M. Khoshgoftaar, and J. M. Johnson. A comparative approach to threshold optimization for classifying imbalanced data. In *The International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2022.

[55] J. T. Hancock, T. M. Khoshgoftaar, and J. M. Johnson. The effects of random undersampling for big data medicare fraud detection. In *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2022*. IEEE, 2022.

[56] J. T. Hancock, T. M. Khoshgoftaar, and J. M. Johnson. Exploring area under the precision recall curve and random undersampling to classify imbalanced big data. *Proceedings of the 27th ISSAT International Conference on Reliability and Quality in Design*, page 111–116, 2022.

[57] J. T. Hancock, T. M. Khoshgoftaar, and J. M. Johnson. Informative evaluation metrics for highly imbalanced big data classification. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1419–1426. IEEE, 2022.

[58] J. T. Hancock, T. M. Khoshgoftaar, and J. M. Johnson. Evaluating classifier performance with highly imbalanced big data. *Journal of Big Data*, 10(1):42, 2023.

[59] J. T. Hancock, T. M. Khoshgoftaar, and J. M. Johnson. Using area under the precision recall curve to assess the effect of random undersampling in the classification of imbalanced medicare big data. *International Journal of Reliability, Quality and Safety Engineering*, 2023.

[60] J. T. Hancock, T. M. Khoshgoftaar, and S. Landset. Statistical significance of hyperparameter tuning for varying levels of class imbalance. In *Proceedings of the 26th ISSAT International Conference on Reliability and Quality in Design*, pages 155–158. IEEE, 2021.

[61] J. T. Hancock, Q. Liang, and T. M. Khoshgoftaar. A problem-agnostic approach to feature selection and analysis using shap. *Journal of Big Data*, 2024. (status: under review).

[62] J. T. Hancock, H. Wang, T. M. Khoshgoftaar, and Q. Liang. Data reduction techniques for highly imbalanced medicare big data. *Journal of Big Data*, 11(1):8, 2024.

[63] T. Hasanin, T. M. Khoshgoftaar, and R. A. Bauder. Experimental studies on the impact of data sampling with severely imbalanced big data. In *Reuse in intelligent systems*, pages 1–32. CRC Press, 2020.

[64] T. Hasanin, T. M. Khoshgoftaar, and J. L. Leevy. A comparison of performance metrics with severely imbalanced network security big data. In *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 83–88. IEEE, 2019.

[65] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and R. A. Bauder. Severely imbalanced big data challenges: investigating data sampling approaches. *Journal of Big Data*, 6(1):1–25, 2019.

[66] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and R. A. Bauder. Investigating class rarity in big data. *Journal of Big Data*, 7:1–17, 2020.

[67] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and N. Seliya. Examining characteristics of predictive models with imbalanced big data. *Journal of Big Data*, 6:1–21, 2019.

[68] T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and N. Seliya. Investigating random undersampling and feature selection on bioinformatics big data. In *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 346–356. IEEE, 2019.

[69] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[70] T. Hayashi and H. Fujita. One-class ensemble classifier for data imbalance problems. *Applied Intelligence*, 52(15):17073–17089, 2022.

[71] H. He, Y. Bai, E. A. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[72] M. Herland, R. A. Bauder, and T. M. Khoshgoftaar. Medical provider specialty predictions for the detection of anomalous medicare insurance claims. In *2017 IEEE international conference on information reuse and integration (IRI)*, pages 579–588. IEEE, 2017.

[73] M. Herland, R. A. Bauder, and T. M. Khoshgoftaar. The effects of class rarity on the evaluation of supervised healthcare fraud detection models. *Journal of Big Data*, 6(1):1, 2019.

[74] M. Herland, R. A. Bauder, and T. M. Khoshgoftaar. Approaches for identifying us medicare fraud in provider claims data. *Health care management science*, 23:2–19, 2020.

[75] M. Herland, T. M. Khoshgoftaar, and R. A. Bauder. Big data fraud detection using multiple medicare data sources. *Journal of Big Data*, 5(1):1–21, 2018.

[76] M. Herland, T. M. Khoshgoftaar, and R. Wald. A review of data mining using big data in health informatics. *Journal of Big data*, 1:1–35, 2014.

[77] J. Ilonen, P. Paalanen, J.-K. Kamarainen, and H. Kalviainen. Gaussian mixture pdf in one-class classification: computing and utilizing confidence values. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 2, pages 577–580. IEEE, 2006.

[78] G. R. Iversen and H. Norpoth. *Analysis of variance*. Number 1 in Quantitative Applications in the Social Sciences. Sage, Newbury Park, 1987.

[79] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*, volume 1. Wiley New York, 1991.

[80] J. M. Johnson, R. K. Kennedy, and T. M. Khoshgoftaar. Learning from highly imbalanced big data with label noise. *International Journal on Artificial Intelligence Tools*, 32(5), 2023.

[81] J. M. Johnson and T. M. Khoshgoftaar. Deep learning and data sampling with imbalanced big data. In *2019 IEEE 20th international conference on information reuse and integration for data science (IRI)*, pages 175–183. IEEE, 2019.

[82] J. M. Johnson and T. M. Khoshgoftaar. Deep learning and thresholding with class-imbalanced big data. In *2019 18th IEEE international conference on machine learning and applications (ICMLA)*, pages 755–762. IEEE, 2019.

[83] J. M. Johnson and T. M. Khoshgoftaar. Medicare fraud detection using neural networks. *Journal of Big Data*, 6(1):1–35, 2019.

[84] J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

[85] J. M. Johnson and T. M. Khoshgoftaar. The effects of data sampling with deep learning and highly imbalanced big data. *Information Systems Frontiers*, 22(5):1113–1131, 2020.

[86] J. M. Johnson and T. M. Khoshgoftaar. Hcpcs2vec: Healthcare procedure embeddings for medicare fraud prediction. In *2020 IEEE 6th international conference on collaboration and internet computing (CIC)*, pages 145–152. IEEE, 2020.

[87] J. M. Johnson and T. M. Khoshgoftaar. Semantic embeddings for medical providers and fraud detection. In *2020 IEEE 21st International conference on information reuse and integration for data science (IRI)*, pages 224–230. IEEE, 2020.

[88] J. M. Johnson and T. M. Khoshgoftaar. Encoding techniques for high-cardinality features and ensemble learners. In *2021 IEEE 22nd international conference on information reuse and integration for data science (IRI)*, pages 355–361. IEEE, 2021.

[89] J. M. Johnson and T. M. Khoshgoftaar. Medical provider embeddings for healthcare fraud detection. *SN Computer Science*, 2(4):276, 2021.

[90] J. M. Johnson and T. M. Khoshgoftaar. Output thresholding for ensemble learners and imbalanced big data. In *2021 IEEE 33rd international conference on tools with artificial intelligence (ICTAI)*, pages 1449–1454. IEEE, 2021.

[91] J. M. Johnson and T. M. Khoshgoftaar. Robust thresholding strategies for highly imbalanced and noisy data. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1182–1188. IEEE, 2021.

[92] J. M. Johnson and T. M. Khoshgoftaar. Thresholding strategies for deep learning with highly imbalanced big data. *Deep Learning Applications, Volume 2*, pages 199–227, 2021.

[93] J. M. Johnson and T. M. Khoshgoftaar. Cost-sensitive ensemble learning for highly imbalanced classification. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1427–1434. IEEE, 2022.

[94] J. M. Johnson and T. M. Khoshgoftaar. Encoding high-dimensional procedure codes for healthcare fraud detection. *SN Computer Science*, 3(5):362, 2022.

[95] J. M. Johnson and T. M. Khoshgoftaar. Healthcare provider summary data for fraud classification. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 236–242. IEEE, 2022.

[96] J. M. Johnson and T. M. Khoshgoftaar. A survey on classifying big data with label noise. *ACM Journal of Data and Information Quality*, 14(4):1–43, 2022.

[97] J. M. Johnson and T. M. Khoshgoftaar. Data-centric ai for healthcare fraud detection. *SN Computer Science*, 4(4):389, 2023.

[98] P. Juszczak and R. P. Duin. Uncertainty sampling methods for one-class classifiers. In *Proceedings of ICML-03, Workshop on Learning with Imbalanced Data Sets II*, pages 81–88. Citeseer, 2003.

[99] Kaggle. Credit card fraud detection. *https://www.kaggle.com/mlg-ulb/creditcardfraud*, 2018.

[100] L. Kalantari, P. Gader, S. Graves, and S. A. Bohlman. One-class gaussian process for possibilistic classification using imaging spectroscopy. *IEEE Geoscience and Remote Sensing Letters*, 13(7):967–971, 2016.

[101] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[102] R. K. Kennedy, J. M. Johnson, and T. M. Khoshgoftaar. The effects of class label noise on highly-imbalanced big data. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1427–1433. IEEE, 2021.

[103] R. K. Kennedy and T. M. Khoshgoftaar. Accelerated deep learning on hpcc systems. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 847–852. IEEE, 2020.

[104] R. K. Kennedy and T. M. Khoshgoftaar. An examination of neural networks on cluster computers. In *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 155–160. IEEE, 2021.

[105] R. K. Kennedy, Z. Salekshahrezaee, and T. M. Khoshgoftaar. Unsupervised anomaly detection of class imbalanced cognition data using an iterative cleaning method. In *2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 303–308. IEEE, 2023.

[106] R. K. Kennedy, Z. Salekshahrezaee, F. Villanustre, and T. M. Khoshgoftaar. Iterative cleaning and learning of big highly-imbalanced fraud data using unsupervised learning. *Journal of Big Data*, 10(1):106, 2023.

[107] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse. An empirical study of learning from imbalanced data using random forest. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 310–317. IEEE, 2007.

[108] T. M. Khoshgoftaar and P. Rebours. Improving software quality prediction by noise filtering techniques. *Journal of Computer Science and Technology*, 22:387–396, 2007.

[109] T. M. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, and A. Folleco. Learning with limited minority class data. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 348–353. IEEE, 2007.

[110] M. Kull, T. M. Silva Filho, and P. Flach. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. *Electronic Journal of Statistics*, 11(2):5052–5080, 2017.

[111] L. I. Kuncheva. A stability index for feature selection. In *Artificial intelligence and applications*, pages 421–427. Citeseer, 2007.

[112] L. I. Kuncheva, A. Arnaiz-Gonzalez, J.-F. Díez-Pastor, and I. A. Gunn. Instance selection improves geometric mean accuracy: a study on imbalanced data classification. *Progress in Artificial Intelligence*, 8(2):215–228, 2019.

[113] S. Le Cessie and J. C. Van Houwelingen. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 41(1):191–201, 1992.

[114] J. L. Leevy, J. T. Hancock, and T. M. Khoshgoftaar. Assessing one-class and binary classification approaches for identifying medicare fraud. In *2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 267–272. IEEE, 2023.

[115] J. L. Leevy, J. T. Hancock, and T. M. Khoshgoftaar. Comparative analysis of binary and one-class classification techniques for credit card fraud data. *Journal of Big Data*, 10(1):118, 2023.

[116] J. L. Leevy, J. T. Hancock, T. M. Khoshgoftaar, and A. Abdollah Zadeh. Investigating the effectiveness of one-class and binary classification for fraud detection. *Journal of Big Data*, 10(1):157, 2023.

[117] J. L. Leevy, J. T. Hancock, T. M. Khoshgoftaar, and J. Peterson. Detecting information theft attacks in the bot-iot dataset. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 807–812. IEEE, 2021.

[118] J. L. Leevy, J. T. Hancock, T. M. Khoshgoftaar, and J. M. Peterson. An easy-to-classify approach for the bot-iot dataset. In *2021 IEEE third international conference on cognitive machine intelligence (CogMI)*, pages 172–179. IEEE, 2021.

[119] J. L. Leevy, J. T. Hancock, T. M. Khoshgoftaar, and J. M. Peterson. Iot information theft prediction using ensemble feature selection. *Journal of Big Data*, 9(1):1–48, 2022.

[120] J. L. Leevy, J. T. Hancock, T. M. Khoshgoftaar, and N. Seliya. Iot reconnaissance attack classification with random undersampling and ensemble feature selection. In *2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC)*, pages 41–49. IEEE, 2021.

[121] J. L. Leevy, J. T. Hancock, T. M. Khoshgoftaar, and A. A. Zadeh. One-class classifier performance: Comparing majority versus minority class training. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 86–91. IEEE, 2023.

[122] J. L. Leevy, J. T. Hancock, R. Zuech, and T. M. Khoshgoftaar. Detecting cybersecurity attacks using different network features with lightgbm and xgboost learners. In *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*, pages 190–197. IEEE, 2020.

[123] J. L. Leevy, J. M. Johnson, J. T. Hancock, and T. M. Khoshgoftaar. Threshold optimization and random undersampling for imbalanced credit card data. *Journal of Big Data*, 10(1):58, 2023.

[124] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1):1–30, 2018.

[125] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya. The effect of time on the maintenance of a predictive model. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1891–1896. IEEE, 2019.

[126] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya. Investigating the relationship between time and predictive model maintenance. *Journal of Big Data*, 7:1–19, 2020.

[127] J. L. Leevy, T. M. Khoshgoftaar, and J. T. Hancock. Evaluating performance metrics for credit card fraud classification. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1336–1341. IEEE, 2022.

[128] J. L. Leevy, T. M. Khoshgoftaar, and J. T. Hancock. Feature evaluation for iot botnet traffic classification. *International Journal of Internet of Things and Cyber-Assurance*, 2(1):87–102, 2022.

[129] J. L. Leevy, T. M. Khoshgoftaar, and J. T. Hancock. Using random undersampling and ensemble feature selection for iot attack prediction. *International Journal of Reliability, Quality and Safety Engineering*, 2023.

[130] LEIE. Office of inspector general leie downloadable databases., 2022. [Online]. Available: `https://oig.hhs.gov/exclusions/index.asp`.

[131] G. Lemaître, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563, 2017.

[132] G. Lemaître, F. Nogueira, and C. K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[133] S. Lundberg and Others. shap. https://github.com/slundberg/shap/tree/v0.41.0. Accessed: 2023-07-09.

[134] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[135] J. Mayfield. New ftc data show consumers reported losing nearly $8.8 billion to scams in 2022. https://www.ftc.gov/news-events/news/press-releases/2023/02/new-ftc-data-show-consumers-reported-losing-nearly-88-billion-scams-2022, 2023. Accessed: 2024-03-10.

[136] W. McGinnis. Category encoders python library, 2022. https://contrib.scikit-learn.org/category_encoders/.

[137] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

[138] G. S. Morrison. Tutorial on logistic-regression calibration and fusion: converting a score to a likelihood ratio. *Australian Journal of Forensic Sciences*, 45(2):173–197, 2013.

[139] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[140] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[141] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[142] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[143] A. N. Richter and T. M. Khoshgoftaar. Learning curve estimation with large imbalanced datasets. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 763–768. IEEE, 2019.

[144] A. N. Richter and T. M. Khoshgoftaar. Sample size determination for biomedical big data with limited labels. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 9:1–13, 2020.

[145] L. Rokach and O. Maimon. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

[146] R. Sauber-Cole and T. M. Khoshgoftaar. The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey. *Journal of Big Data*, 9(1):98, 2022.

[147] R. Sauber-Cole, T. M. Khoshgoftaar, and J. M. Johnson. Gans for class-imbalanced data: A meta-analysis of github projects. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1419–1424. IEEE, 2022.

[148] B. Schölkopf, K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE transactions on Signal Processing*, 45(11):2758–2765, 1997.

[149] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.

[150] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Mining data with rare events: a case study. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 132–139. IEEE, 2007.

[151] N. Seliya, A. Abdollah Zadeh, and T. M. Khoshgoftaar. A literature review on one-class classification and its potential applications in big data. *Journal of Big Data*, 8(1):1–31, 2021.

[152] N. Seliya, T. M. Khoshgoftaar, and J. Van Hulse. A study on the relationships of classifier performance metrics. In *2009 21st IEEE international conference on tools with artificial intelligence*, pages 59–66. IEEE, 2009.

[153] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[154] The Centers for Medicare and Medicaid Services. Medicare part d prescribers – by provider data dictionary. https://data.cms.gov/resources/medicare-part-d-prescribers-by-provider-data-dictionary, 2020.

[155] The Centers for Medicare and Medicaid Services. Medicare durable medical equipment, devices & supplies – by referring provider. https://data.cms.gov/provider-summary-by-type-of-service/medicare-durable-medical-equipment-devices-supplies/medicare-durable-medical-equipment-devices-supplies-by-referring-provider, 2021.

[156] The Centers for Medicare and Medicaid Services. Medicare durable medical equipment, devices & supplies – by referring provider and service. https://data.cms.gov/provider-summary-by-type-of-service/medicare-durable-medical-equipment-devices-supplies/medicare-durable-medical-equipment-devices-supplies-by-referring-provider-and-service, 2021.

[157] The Centers for Medicare and Medicaid Services. Medicare durable medical equipment, devices & supplies – by referring provider and service data dictionary. https://data.cms.gov/resources/medicare-durable-medical-equipment-devices-supplies-by-referring-provider-and-service-data-dictionary, 2021.

[158] The Centers for Medicare and Medicaid Services. Medicare part d prescribers – by provider and drug. https://data.cms.gov/provider-summary-by-type-of-service/medicare-part-d-prescribers/medicare-part-d-prescribers-by-provider-and-drug, 2021.

[159] The Centers for Medicare and Medicaid Services. Medicare part d prescribers – by provider and drug data dictionary. https://data.cms.gov/resources/medicare-part-d-prescribers-by-provider-and-drug-data-dictionary, 2021.

[160] The Centers for Medicare and Medicaid Services. Medicare part d prescribers - by provider. https://data.cms.gov/provider-summary-by-type-of-service/medicare-part-d-prescribers/medicare-part-d-prescribers-by-provider, 2021.

[161] The Centers for Medicare and Medicaid Services. Medicare physician & other practitioners – by provider. https://data.cms.gov/provider-summary-by-type-of-service/medicare-physician-other-practitioners/medicare-physician-other-practitioners-by-provider, 2021.

[162] The Centers for Medicare and Medicaid Services. Medicare physician & other practitioners – by provider and service. https://data.cms.gov/provider-summary-by-type-of-service/medicare-physician-other-practitioners/medicare-physician-other-practitioners-by-provider-and-service, 2021.

[163] The Centers for Medicare and Medicaid Services. Medicare physician & other practitioners – by provider and service data dictionary. https://data.cms.gov/resources/medicare-physician-other-practitioners-by-provider-and-service-data-dictionary, 2021.

[164] The Centers for Medicare and Medicaid Services. Medicare physician & other practitioners – by provider data dictionary. https://data.cms.gov/resources/medicare-physician-other-practitioners-by-provider-data-dictionary, 2021.

[165] J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.

[166] J. Van Hulse and T. M. Khoshgoftaar. Knowledge discovery from imbalanced and noisy data. *Data & Knowledge Engineering*, 68(12):1513–1542, 2009.

[167] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942, 2007.

[168] J. Van Hulse, T. M. Khoshgoftaar, A. Napolitano, and R. Wald. Threshold-based feature selection techniques for high-dimensional bioinformatics data. *Network modeling analysis in health informatics and bioinformatics*, 1:47–61, 2012.

[169] H. Wang, J. T. Hancock, and T. M. Khoshgoftaar. Improving medicare fraud detection through big data size reduction techniques. In *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 208–217. IEEE, 2023.

[170] H. Wang, T. M. Khoshgoftaar, and A. Napolitano. A comparative study of ensemble feature selection techniques for software defect prediction. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 135–140. IEEE, 2010.

[171] H. Wang, Q. Liang, J. T. Hancock, and T. M. Khoshgoftaar. A comparative study of model-agnostic and importance-based feature selection approaches. In *2023 IEEE*

5th International Conference on Cognitive Machine Intelligence (CogMI), pages 75–82. IEEE Computer Society, 2023.

[172] H. Wang, Q. Liang, J. T. Hancock, and T. M. Khoshgoftaar. Enhancing credit card fraud detection through a novel ensemble feature selection technique. In *2023 IEEE 24th International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 121–126. IEEE, 2023.

[173] H. Wang, Q. Liang, J. T. Hancock, and T. M. Khoshgoftaar. Feature selection strategies: A comparative analysis of shap-value and importance-based methods. *Journal of Big Data*, 2024.

[174] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[175] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[176] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[177] P. Zheng, S. Yuan, X. Wu, J. Li, and A. Lu. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1286–1293, 2019.

[178] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. Detecting sql injection web attacks using ensemble learners and data sampling. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 27–34. IEEE, 2021.

[179] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. Detecting web attacks in severely imbalanced network traffic data. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 267–273. IEEE, 2021.

[180] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. Detecting web attacks using random undersampling and ensemble learners. *Journal of Big Data*, 8(1):1–20, 2021.

[181] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. Feature popularity between different web attacks with supervised feature selection rankers. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 30–37. IEEE, 2021.

[182] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. Investigating rarity in web attacks with ensemble learners. *Journal of Big Data*, 8(1):1–27, 2021.

[183] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. A new feature popularity framework for detecting cyberattacks using popular features. *Journal of Big Data*, 9(1):119, 2022.

[184] R. Zuech, J. T. Hancock, and T. M. Khoshgoftaar. Predicting cyberattacks with destination port through various input feature scenario. *International Journal of Reliability, Quality and Safety Engineering*, 29(03):2250003, 2022.