Improving Machine Learning Methods for Handling Data Imbalance Problem

Abdullah-All-Tanvir

Department of Computer Science and Engineering

United International University

A thesis submitted for the degree of

MSc in Computer Science & Engineering

June 2022

Declaration

I, Abdullah-All-Tanvir, declare that this thesis titled, Improving Machine Learning Methods for Handling Data Imbalance Problem and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while candidature for an MSc degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or qualification at United International University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is my own work.
- I have acknowledged all primary sources of help.
- Where the thesis is based on my work jointly with others, I have clarified exactly what others did and what I have contributed myself.

Signed: Abdullah-All-Tanvir

Date: 24-07-2022

Abdullah-All-Tanvir

Certificate

I do hereby declare that the research works embodied in this thesis entitled Improving Machine Learning Methods for Handling Data Imbalance Problem is the outcome of an original work carried out by Abdullah-All-Tanvir under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of MSc in Computer Science and Engineering.

Signed: _____

Date: _____

(Dr. Swakkhar Shatabda)

Department of Computer Science and Engineering,

United International University,

Dhaka-1209, Bangladesh.

Abstract

Class unbalanced datasets are widespread in various fields, including health, security, and banking. When dealing with imbalanced datasets, a standard supervised learning algorithm is biased toward the dominant class. In real-life applications, however, the minority class instances are more interested in reflecting the notion than the majority class instances. For categorizing unbalanced datasets, numerous strategies based on sampling methods (under-sampling of the majority class and oversampling of the minority class), cost-sensitive learning methods, and ensemble learning have recently been employed in the literature. However, deleting the majority of samples at random using a uniform distribution may result in needless data loss. In this paper, we proposed 3 different cluster-based undersampling models to prevent unnecessary data loss. First, we inject test data into training data for clustering. Then we select 25% close to the centroid and 25% from the boundary line. For the last method, we clean 50% majority data around minority data. We experiment with our methods over 49 datasets and calculate auROC, auPR, F1-Score, and MCC for evaluation. According to the experimental results, our methods are promising and successful strategies for dealing with severely unbalanced datasets.

Acknowledgements

I would like to thank my advisor Dr. Swakkhar Shatabda. His guidance and direction throughout the thesis have been invaluable, and I greatly appreciate the new perspective he provided. His energized mints, visionaries, attentive remarks and proposals, and exceptional help at each phase of my MSc were acknowledging and fundamental. His capacity to muddle me enough to at last answer my very own inquiry effectively is something profitable that I have realized, and I would endeavor to imitate it if at any time I get the chance.

I also thank Dr. Nurul Huda, MSCSE Director, for his constructive feedback on coursework and for being an affable minor advisor. I want to thank Chowdhury Mofizur Rahman, Vice Chancellor, for giving me the opportunity. And, finally, I want to acknowledge my friends and family for bearing with me and having a sympathetic ear whenever I faced struggles.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Problem Statement

Classification of samples from the imbalanced dataset is a very tedious task. Mostly, the datasets from real-life problems have high dimensions with more than one class. The presence of an imbalanced course significantly affects the classification accuracy as it often miss-classified the minority class. However, there are some problems where predicting the minority class is the most critical case. Customer purchasing intention from online shopping, for instance, most of the time, customers refuse to buy products from an e-commerce website. So the number of times buying kinds of stuff will be significantly less, and it is a high possibility that if we want to predict if a customer will buy a product under some circumstances, the class will be miss-classified. Again, for detecting fraud transactions on credit cards arises the same problem. Most of cases, the transactions are solid. Only a few times that fraud happens. As a result, it is difficult to detect credit card fraud as the ratio of the fraudulent and non-fraudulent transaction are high. To handle such problems, researchers came up with various methods to make the model more efficient and differentiate all the classes.

1.2 Motivation

When it comes to a real-life problem, sometimes there is a high chance that one class may have many instances compared to another. In the imbalanced dataset, the class with an immense number of instances is called majority data, and the class with small-scale is called minority data. The traditional classification models ignore the minority data in most cases due to the higher number of majority data. Those models cannot handle imbalanced data properly as their primary focus is on improving the accuracy. However, some studies in the recent past are capable of handling this issue a little bit. The common methods which are introduced so far are the sampling technique, cost-sensitive method, an ensemble method. There are two ways to solve this issue in the sampling technique: undersampling and oversampling; If the majority data has been reduced from an imbalanced dataset, it is called undersampling, [2, 3] and if the minority data has been increased, it is called oversampling [4, 5]. For undersampling, there is a high possibility of losing important information. And for oversampling, minority data are increased randomly, some data can be duplicated multiple times, whereas others can be none. The cost-sensitive method's goal is to minimize a model's cost during training [6]. For each

class, a separate cost of misclassification mistakes is assigned. Normally the lower cost is assigned for minority data and the higher for majority data. As there are no specific rules for assigning those costs, sometimes it can be misclassified and produces different output for a different kind of cost. Ensemble approaches like Bagging and Boosting are also commonly utilized [7] for identifying imbalanced data. In each iteration, ensemble approaches essentially employ sampling techniques.

1.3 Objectives

The main objectives of this paper are as follows:

- Develop a new method for under-sampling based on the limitations and open issues in CUSBoost [8].
- Performing the model on real-life data.

1.4 Brief Methodology

In this paper, we propose an improvement of the cluster-based undersampling method. As mentioned earlier, in the undersampling method, the majority of data are reduced; we propose several techniques to lower the majority of data. First, we have selected 46 open source datasets with different class imbalance ratios and normalized those data. Then we use StratifiedKFold cross-validation for the train-test split. After that, we applied our hypothesis to training data and created a model with an AdaBoost classifier. Later on, we test them with test data. Lastly, we evaluate our model with real-life data [9].

1.5 Outcome

From the simulation results, we observed that the proposed techniques could balance any imbalance dataset and significantly improve the performance in terms of auROC, are, MCC and F1-Score. The proposed techniques can solve the open issues of existing work and provide better outcomes.

1.6 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 provides related works.

Chapter 3 Cluster Based Sampling.

Chapter 4 Customer Purchase Intention Prediction for Online Shopping

Chapter 5 Presents the conclusions, summarizes the thesis contributions, and discusses the future works.

Chapter 2

Background

As class imbalance is a major problem in machine learning, many methods are introduced to overcome this problem. Previously, there have been so many researches trying to figure out how can imbalanced data can be handled. Researchers have often used sampling, cost-sensitive methods, or ensemble techniques. Some of them have used deep neural networks and reinforcement learning as well.

2.1 Preliminary

Undersampling: Undersampling is a strategy for balancing unequal datasets by retaining all of the data in the minority class while reducing the size of the majority class. It's one of the methods data scientists may employ to obtain more accurate information from previously unbalanced datasets.

Oversampling: Oversampling is the process of replicating minority class instances at random and adding them to the training dataset. Replacement is used to choose examples from the training dataset at random. This means that examples from the minority class can be selected and added to the new "more balanced" training dataset multiple times; they are chosen from the original training dataset, added to the new training dataset, and then returned or "replaced" in the original dataset, allowing them to be chosen again.

SMOTE: SMOTE stands for Synthetic Minority Oversampling Technique. It is a data augmentation method that generates synthetic data points from the original data. SMOTE may be considered a more advanced kind of oversampling or a data augmentation method. SMOTE has the benefit of not producing duplicate data points but synthetic data points that are somewhat different from the original data points.

AdaBoost: An AdaBoost classifier is a meta-estimator that starts by fitting a classifier on the original dataset and then fits successive copies of the classifier on the same dataset but adjusts the weights of poorly classified instances such that future classifiers focus more on difficult situations.

Stratified Cross Validation: Stratified k-fold cross-validation is the same as regular k-fold cross-validation, except that it uses stratified sampling rather than random sampling.

2.2 Related Work

2.2.1 Class Imbalance

There is a lot of research is conducted to solve the class imbalance issue.

A. *Sampling Technique*

i. **Undersampling**

Rayhan et al. have created a new algorithm named CUSTBoost to cite rayhan2017cusboost. CUSBoost is employed for successful imbalanced classification where the research conducted AdaBoost to compare for best output. The proposed CUSBoost complements the RUSBoost and SMOTEBoost methods; was compared against state-of-the-art ensemble learning methods such as AdaBoost, RUSBoost, and SMOTEBoost. The method was implemented on 13 unstable multi datasets with varying imbalance proportions. It is considered an effective method for dealing with extremely unbalanced datasets. CUSBoost first groups the majority class occurrences before performing random under-sampling, allowing the Adaboost to employ examples from all areas of the set of data. It has been demonstrated to outperform the other techniques in a specific range of unbalanced ratios.

Cluster-based under-sampling is done by Rekha et al [10]. In this research, the majority class instances are chosen based on their average distance from the cluster's minority class samples.

Another research based on the cluster-based under-sampling technique was conducted by Lin et al [11]. In this research, they have given two hypotheses for selecting the center of each cluster and the cluster number. They used 44 different datasets, and after that, they applied their methods to two large datasets to find out the improvement.

Yen et al. discuss selecting training data for imbalanced data to improve performance [12]. At first, they divided their samples into k number of clusters. They have selected a certain ratio of majority and minority data for each cluster.

Peng et al. try to solve the problem using reinforcement learning [13]. Reinforcement learning has been innovated in this article to diminish the non-differentiable optimization problem. Evaluation metric optimization was incorporated to demonstrate the effectiveness of datasets. The study empirically analyzed the time complexity and the under-sampled datasets and outperformed them. The method extracted the best outcome and performed it consistently to give privileged data sampling.

ZHANG et al. solved this case by EEU algorithm, which is evaluation-based undersampling used in this research to resolve the bias issue for imbalanced datasets [14]. As minority data is ignored in the implementation, the algorithm works with them and improves the accuracy to make a balanced sample. 5 UCI datasets have been used in the experiment, and it has been observed that the EEU algorithm improved the minority data and outperformed the samplings.

Huang et al. solve the problem of class imbalance for face recognition using a deep convolutional neural network [15]. They have shown that cluster-based large margin local embedding with the combination of k-nearest cluster-based algorithm higher the accuracy compared to the existing methods for both face recognition and face attribute prediction of the imbalanced class distribution.

A diversified sensitivity-based undersampling was implemented by Ng et al. to make a boundary between minor and major data [16]. Clustering and sampling of undersampling method applied to make a balanced iteration which consumes the training and makes a good generalization. The research proposed the DSUS algorithm, which showed 80% accuracy with 14 different UCI datasets.

Using a KNN-based overlapping samples filter methodology, Nwe et al. offer an effective under-sampling strategy for classifying unbalanced and overlapping data [17]. They have also examined three classifiers of ensemble learning to analyze the performance. They calculated the area under the curve (AUC) for evaluation purposes, G-mean and F-measure.

Devi et al. proposed a boosting aided adaptive cluster-based method to reduce the high ratio of majority data in an imbalanced dataset [18]. Adaboost ensemble classifier is recommended for this study. The proposed method is compared with seven different models to validate.

Zhang et al. address an issue regarding undersampling [19]. Randomly reduced majority instances might cause an unnecessary information loss. To overcome this issue, their study proposed a method named Undersampling using Sensitivity (USS), where they refer to low sensitivity data as noisy and high sensitivity as borderline examples. The examples with high sensitivity are selected in USS. This study has applied over 20 different datasets from the KEEL repository.

Le et al. proposed a novel approach to exclude a portion of the majority of data. First, they calculate the Instance Hardness Threshold (IHT) value for the majority of instances [20]. The data with the highest value denotes noisy data, and then those data are removed to balance with minority examples. Their study also investigated cluster-based boosting algorithms that calculate the distance from the centroid to the data point and mention them as weight. Diverse iterations are performed by this algorithm to select the weak classifier and then by joining them to make a strong classifier.

Vuttipittayamongkol et al. handled the class imbalance situation by introducing a neighborhood-based undersampling method [1]. In this approach, they eliminate potential majority data points from the overlapping area, which makes the minority data more visible and increases the ability to distinguish the class label.

ii. **Oversampling**

One of the research focuses on imbalanced data classification and suggests a bi-directional algorithm mentioned BDSK by Song et al [21]. The recommended algorithm was implemented with the help of K-means and successfully resolved the noise and shortage of sampling issues. The implemented algorithm improved the class imbalance problem, and

tan corpus data shows the efficiency rather than another data set. Bi-directional BDSK progressed the imbalances and nicely turned them into a rapid class sampling.

Another paper focused by Shangguan et al. on improving the data quality by using the D-SMOTE algorithm [22]. To detect the abnormal data samples, 2D CNN has been implemented and based on the extracted result, different Machine learning approaches are compared and contrasted. The study clearly explored that oversampling technique accurately detects abnormal data and improves the data quality and accuracy.

EGWAN is an algorithm proposed by Ren et al. that has been applied to generate data samplings for the minority classifiers [23]. In this study, an entropy-weighted level has been constructed to make balanced data factors. As two benchmark algorithms have been compared, it is successfully portrayed that the proposed EGWAN improved performance of this study.

One of the articles proposed by Wang et al. is entropy and confidence-based undersampling techniques named ECUBoost [24]. This method handles the missing informative data by considering both confidence and entropy as benchmarks. It guarantees the validity and structural distribution of the majority of samples during undersampling.

Peng et al. show the best adaptive classifiers to resolve the imbalance issue [25]. The over-sampling method of this study made a packaged use of trained data and reduce the information loss. 16 UCI datasets are experimented with through 3 evaluation matrices and successfully extracted the 12 state-of-art methods comparisons for a better result.

The research worked with some medical datasets where the C4.5 decision tree was implemented by Xu et al [26]. The paper compared KNSMOTE and SMOTE to make high generalized data through given imbalanced data. All 6 UCI datasets of this study went through the oversampling algorithms and secured 97.58% accuracy based on the Random Forest algorithm.

SVM classification has been implemented by Shi et al. in the research to avoid the faults of the vehicle diagnosis [27]. Linear discriminant analysis is the solution of this study, where it has been assured to generate quality data with the threshold adjustment. There used different datasets to make a comparison. Comparisons show that the proposed algorithm is more effective with the best quality of fault diagnosis.

MAHAKIL theory has been proposed in this research by Bennin et al. that generates two instances from one parent [28]. 20 defect datasets are performed through the MAHAKIL and compared with SMOTE, ADASYN, and Borderline-SMOTE to clearly visualize accuracy and performance. the proposed algorithm is highly appreciable to make a high-quality generation through an imbalanced dataset.

Tao et al. proposed novel adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristic filtering [29]. Unlike the other clustering-based oversampling methods, the proposed approach clusters the minority instances using modified density peaks clustering rather than traditional k-means clustering techniques, which allows the proposed method to accurately identify sub-clusters of various sizes and densities, which

is beneficial for the proposed method to simultaneously accommodate for between-class and within-class imbalance issues caused by various reasons.

B. *Ensemble Techniques*

Chen et al. discussed in their research that performance degradation is the root cause of some imbalanced data slit [30]. The paper recognized the issue and proposed an algorithm to resolve it. The undersampling method was implemented to cut off the unrepresentative instances, while oversampling was for diverse instances. The two leveled approaches separate data into two folds and generate quality samples to extract the best quality outcomes.

One of the researchers, Zhang et al., used the inverse undersampling method to generate a huge number of data subsets [31]. Different classifiers are used to make a decision boundary in this research. The major and minor datasets are separated by using the undersampling classifiers. The uncommon thing that has been used is the stacking model. The model was applied on 17 UCI datasets, and AUC, G-mean, and F-1 were measured with the best effectiveness and accuracy.

In one research, Yang et al. addressed that the new minority sample is expressed as the manifold distance, not the Euclidean [32]. Overlapping majority-minority samples tend to spread from the view into completely disjoint subspaces of diverse learning. In addition, imbalanced data can avoid creating samples between minority data that are far away in the manifold place. Experiments with 23 UCI datasets have shown that the recommended method is more accurate in over-sampling classification. The study implements other optimizers with MDOTE to improve the outcome of imbalanced portions.

Li et al. introduced novel ensemble techniques that integrate the edges of both ensembles learning for biasing classifiers and the newly developed undersampling method named Binary PSO instance selection [33]. This method is significant for improving the performance of imbalanced data and producing the best possible outcome of the original data.

C. *Cost Sensitive Techniques*

Ahmed, at el, points out a major issue is in the supervised learning process class imbalance along with overlapping [34]. To resolve this problem, a complement of RUSBoost as LUIBoost has been incorporated in this study. The implementation reduces information losses and generates a balanced sample per iteration. LUIBoost is considered the most cost-effective and the best among all RUSBoost as this research implements it on 18 imbalanced datasets.

Tao et al. solve the imbalance situation by a novel self-adaptive cost weights-based support vector machine cost-sensitive ensemble [35]. To ensure compatibility of optimization targets between weak learners and boosting schemes in the proposed strategy, we use cost-sensitive SVMs as the fundamental weak leaner while also modifying the usual boosting scheme to cost-sensitive ones. We also provide a self-adaptive sequential misclassification cost weights determination approach to ensure more training minority examples for succeeding classifiers, especially borderline minority instances.

D. *Gan Based Techniques*

Lee et al. conduct research that creates new virtual data that is comparable to the data that already exists by using the Generative Adversarial Networks (GAN) model [36]. To classify data after balancing, Random Forest Classifier has been used by authors. The whole procedure is performed over CICIDS 2017 datasets.

A deep learning-based approach is used by Zhou et al. to balance the imbalance of data [37]. Generative Adversarial Networks (GAN) are used for increasing the minority class where they extracted features from a few minority samples via Auto Encoders instead of using all minority samples.

2.2.2 Online Shoppers

There are several works in the literature that study the problem of predicting customers' purchasing intention for online shopping or e-commerce sites. Most of the studies are based on real-time data that were collected from online shopping websites. Several of them have also used session and log data available from the users. A few of the studies focus on finding the key features for online purchase and thus determine why a buyer abandons online shopping.

Esmeli et al.[38] have proposed a machine learning model that can predict a customer's intention in a running session. One of their findings is using session, and log data are not very effective for predicting online purchasing intention. To evaluate their model, they have used the utility scoring method, which can predict a customer's intention of buying a product by creating features dynamically in a particular session. They use Decision Tree, Random Forest, Naive Bayes, K-Nearest Neighbor, and Bagging classifiers. Among them, the Decision Tree performs better with 97% area under the receiver operating characteristic curve (AUC) score.

Houda et al.[39] investigate the choice of customers behind rejecting shopping online. They performed an empirical study with 147 students and concluded that cognitive variables of perceived usefulness and perceived ease of use have a crucial impact on the attitude toward Internet usage. Chung et al. [40] apply different approaches to explore the underlying connection between online shoppers' decision-making process and the type of input device the shoppers use in terms of affect-driven information processing. The results reflect that shoppers using a touch interface to view products demonstrate significantly higher engagement with their shopping experience in a low involvement setting. Using a touch interface also increases the likelihood that consumers will choose a hedonic over a utilitarian option to make an immediate purchase decision.

Mudaa et al.[41] have analyzed the purchasing behavior of a generation of Malaysia and concluded that most people buy products from online retailers operating via Facebook and Instagram. The summary of their study is to find a positive relationship between the perceived trust and the perceived reputation with the online purchase intention of Generation Y shoppers. Baga et al. [42] have studied the level of attributes of durable goods for predicting the customer's purchase intention. Social perception scores of all brands collected from social media and reviews, the polarity calculated by using sentiment analysis, are used to build the prediction model. After that, all the satisfactory instances are selected for each attribute with significant regression analysis to predict the proper product attributes.

By combining the technology acceptance model with the additional determinants and incorporating habitual online usage as a new mediator, a new online purchase intention model is suggested by Law et al. [43]. This model demonstrates the constant use of online connectivity to the intention of purchase. The impact of this research will help online marketers and retailers understand the situation of the middle-aged market segment, which will help them to develop better strategies. Suchacka et al. [44] use Web server log data to understand the attitude of a customer who buys products from e-commerce websites. As there is a problem with customer behavior characterized by the web server log data, the authors analyze user sessions based on session features. The main goal is to point out the features producing a high probability of making a purchase for innovative and traditional customers. An online bookstore website is used for this study, which is hosted on an Apache HTTP Server on Linux with PHP and MYSQL support. They use Association Rules Mining, also called as Apriori Algorithm, for this purpose. The same authors demonstrate a classification problem in another study to recast online purchase predictions [45]. Here, in the session feature space of an online shop, each user session is represented by a 23-element vector. The authors propose a Support Vector Machine (SVM) classification model dividing the user sessions into binary classes. The data are collected from the online bookshop's history. All the data are classified by the browsing sessions and the buying sessions. 99% accuracy is achieved by this model, along with an almost 95% probability of predicting a purchasing session. In another study [46], they introduced a binary classification over user sessions in an online store, which are two types: buying sessions and browsing sessions. Their proposed method uses the traditional K-NN algorithm. They also use historical data collected from an e-commerce bookstore's log file. After implementing the K-NN classifier with different neighbor sizes, $K = 11$ turns out to be the best possible number of neighbors to be considered for classification. The model achieves 87.5% sensitivity and 99.85% accuracy.

Rita et al. [47] introduce the four characteristics of the e-service quality model that better predicts the consumer behaviors that are the subject of this research. This study focuses not only on a customer's intention but also on the impact of customer trust. The findings reveal that three aspects of e-service quality, including website design, security or privacy, and fulfillment, impact total e-service quality. On the other hand, customer service has no bearing on the overall quality of an e-service. Dabbous et al. [48] address the gap between online and offline shopping. In light of this expanding tendency, this study proposes an empirical model that is evaluated using structural equation modeling to explain the impact of content quality and brand interaction inside social media on consumers' brand awareness and purchase inclinations. The study also looks at whether hedonic motivation, consumer involvement, and brand awareness play a role in the relationship between social media stimuli and offline purchase intent. According to the findings, Millennials place a high value on the quality of information companies give on social media and corporate users' involvement.

Martins et al. [49] show that smartphone advertisements can play a vital role in convincing customers to online shopping. Ducoffe's web advertising model and flow experience theory are combined in their proposed model, and they use 303 Portuguese respondents for data collection. The study concludes that the value of advertising, interactive web design, flow experience, and brand recognition can define purchase intention. Xiao et al. [50] describe four cues supporting this consuming behavior that influence customers' buying decisions in cross-border e-commerce: online promotion, content

marketing, personalized recommendation, and social review. They find that these four cues play a crucial role in a customer's purchasing behavior and negatively impact brand familiarity. In recent work [9], the authors attempt to predict the purchase intention of a customer by tracking them during the visit of a session. They use Random Forest, Multilayer Perceptron, and Support Vector Machine with oversampling and feature selection methods. They conclude that multilayer perception using backpropagation and weight backtracking gives the best accuracy and $F1$-Score compared to another classifier. In another work [51], the authors use the same dataset and apply Naive Bayes, C4.5, and Random Forest Classifiers. They also use oversampling techniques to balance the dataset. Their results show that the accuracy and $F1$-Score of Random Forest are higher than the earlier work [9].

The standard benchmark dataset that was proposed by Sakar et al. [9] has been used extensively in the literature [9, 51–55]. However, we notice that most of these methods are performing well only in terms of accuracy and other metrics that are dependent on the confusion matrix and thus prone to threshold or bias set by the classification methods. Often this is not recommended for imbalanced datasets. We have observed that these classifiers proposed in the literature do poorly regarding robust metrics like MCC, F1, auROC and auPR. This paper addresses this gap in the literature and performs ablation using several classifiers to come to a conclusive method.

Chapter 3

Cluster-Based Undersampling

CUSTBoost Algorithms: CUSTBoost uses sampling done by clustering to balance the dataset and then uses Adaboost as a classifier. For sampling the dataset, first, it differentiates between majority and minority data. Then it used k-means clustering on majority data where k is the cluster number which is a hyperparameter. After that, 50% majority instances are selected by performing random under-sampling and the rest of the data are removed. Here, 50% is a parameter where it can be tuned based on the dataset. Finally, the selected majority data are combined with minority data and then the Adaboost classifier is performed to classify the label.

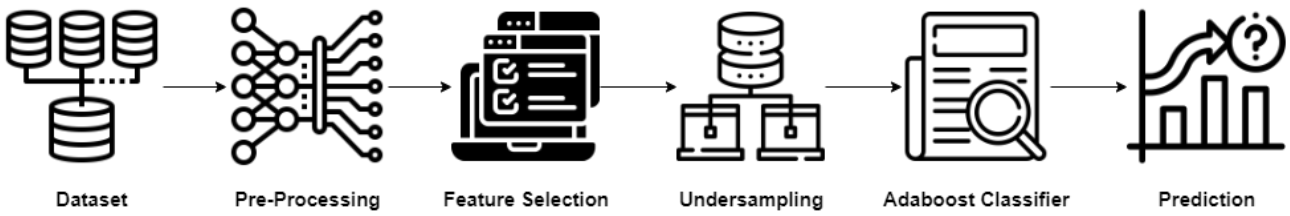


Figure 3.1: Workflow for our proposed model for cluster-based sampling.

3.0.1 Proposed Method

In this study, we proposed some noble cluster-based approaches to overcome the limitations of CUSBoost. The main limitation is that it reduces the majority of data randomly where some important information might be missing. Again, reducing the majority of data to 50% might affect the model as well. So far, we are able to increase our performance compared to previous studies. The methodology that we have followed in this paper is depicted in Figure 4.1. We have in total 3 experiments to overcome the problem of CUSTBoost. We are focusing on the problem of CUSTBoost where it took the 50% majority class randomly. The hypothesizes are-

3.0.1.1 Injecting Test Data

In the clustering, CUSTBoost applied k-means clustering on the majority training data where we use both training and testing data to select the centroid of each cluster. However, we don't put the label of test data as it will cause overfitting rather, we use only features for clustering. Once we get the centroids of each cluster, we remove test data and do rest of the operations as same as CUSTBoost.

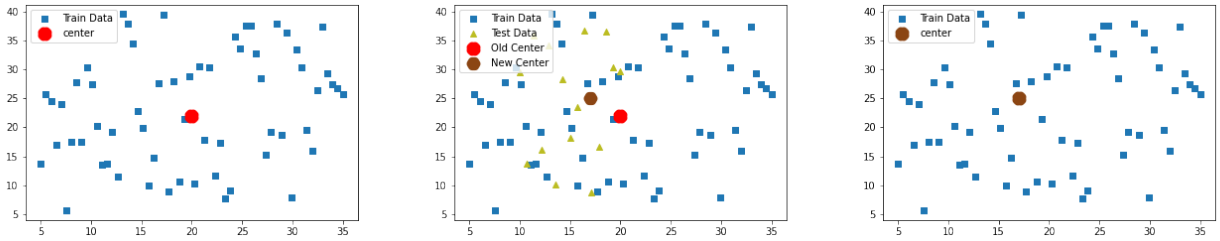


Figure 3.2: Before adding test majority data, the position of the centroid (Left), after adding test data, the centroid is shifted to another position (Middle) and removing test data and keep the new centroid position (Right) for Method A.

In Figure 3.2, we have shown how our 1st methods are working. The pseudo code for this hypothesis are explained below

Algorithm 1 Algorithm for First Hypothesis

0: $X \longleftarrow append(XTrain, XTest)$
0: Applying K-Means Clustering for X
0: Predict only XTrain data
0: $points\_under\_each\_cluster \longleftarrow PointsInEachClusterForXTrain$ =0

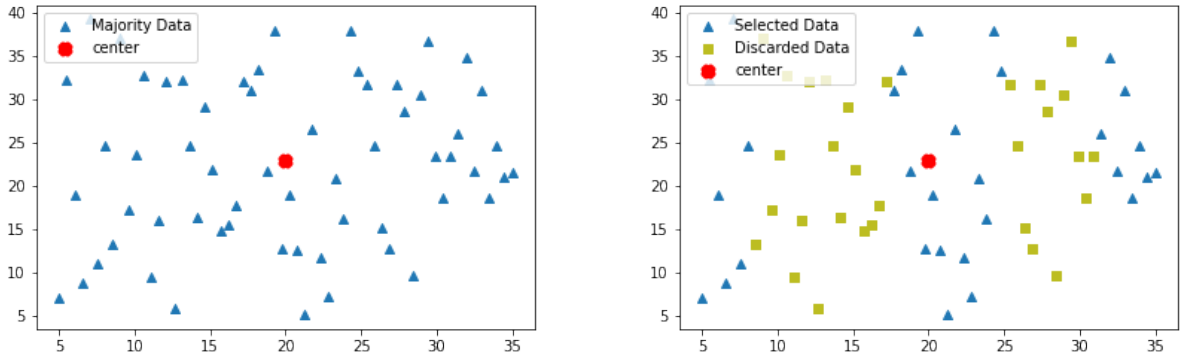


Figure 3.3: After Clustering, how does the majority of data look (Left) and which data should be selected, and which data should be discarded (Right) for Method B

3.0.1.2 Choosing Data Point Based on Centroid

In this method we are focusing on selecting the majority class. In CUSTBoost, they have selected 50% majority data randomly from each cluster. And in this method, we select 25% data near to the center and 25% far from the center. We calculate the Euclidian distance from each data point to the center. Then we sort all the data points based on Euclidian distance. After that, we take 1st 25% data and the last 25% majority data. In Figure 3.3, we can see how the majority of data are reduced in this procedure. The pseudo-code for this hypothesis is given below-

Algorithm 2 Algorithm for Second Hypothesis

0: **for** $cluster \in points\_under\_each\_cluster$ **do**
0: $points\_under\_this\_cluster \longleftarrow AllPointsUnderThisCluster$
0: $majority\_distance[majority\_data] \longleftarrow Euclidian\_Distance(centroid, majority\_data)$
0: $sorted\_point \longleftarrow sort(distance\_from\_cluster\_to\_each\_point)$
0: $points\_close\_to\_center \longleftarrow distance\_from\_cluster\_to\_each\_point \times 0.5$
0: $points\_close\_to\_boundary \longleftarrow distance\_from\_cluster\_to\_each\_point - points\_close\_to\_center$
0: $selected\_majority\_instances \longleftarrow sorted\_point[: points\_close\_to\_center] + sorted\_point[-points\_close\_to\_boundary :]$
=0

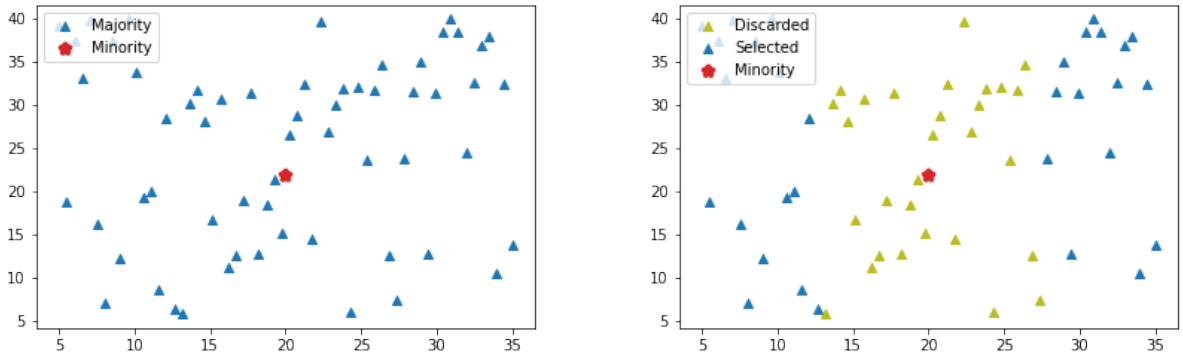


Figure 3.4: Before neighborhood cleaning (Left) and after neighborhood cleaning (Right) for Method C.

3.0.1.3 Neighborhood Cleaning

Here, we use both majority and minority data for clustering. Then in each cluster, 1st we take minority data and remove 50% majority data around minority data which is named neighborhood cleaning. Figure 3.4 shows how neighborhood cleaning works. The pseudo-code for this hypothesis is given below-

3.0.2 Benchmark Dataset

As we are handling the imbalanced dataset, we take 46 different kinds of datasets from the Keel data repository with different data imbalance ratios. Table 1 has shown the characteristics of this dataset. After that, we used a real-life dataset [9] to evaluate our model which is based on purchasing intention of a customer from an e-commerce website. In this dataset, there are about 10422 instances that are negative class from in total of 12330 records, and the rest of them are positive class. The properties of the dataset have been described in Table 1.

3.0.3 Performance Evaluation

As we are dealing with an imbalanced dataset, we cannot rely on accuracy. For this purpose, we use Area Under the Receiver Operator Curve (AUROC) and Area Under Precision-Recall Curve (AUPR) based True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) from the confusion matrix.

dataset_name	number_of_features	number_of_records	IR
abalone19.txt	8	4174	129.44
abalone9-18.txt	8	731	16.40
ecoli-0-1-3-7_vs_2-6.txt	7	281	39.14
ecoli-0_vs_1.txt	7	220	1.86
ecoli1.txt	7	336	3.36
ecoli2.txt	7	336	5.46
ecoli3.txt	7	336	8.60
ecoli4.txt	7	336	15.80
glass-0-1-2-3_vs_4-5-6.txt	9	214	3.20
glass-0-1-6_vs_2.txt	9	192	10.29
glass-0-1-6_vs_5.txt	9	184	19.44
glass0.txt	9	214	2.06
glass1.txt	9	214	1.82
glass2.txt	9	214	11.59
glass4.txt	9	214	15.47
glass5.txt	9	214	22.78
glass6.txt	9	214	6.38
haberman.txt	3	306	2.78
iris0.txt	4	150	2.00
new-thyroid1.txt	5	215	5.14
newthyroid2.txt	5	215	5.14
page-blocks-1-3_vs_4.txt	10	472	15.86
page-blocks0.txt	10	5472	8.79
pima.txt	8	768	1.87
segment0.txt	19	2308	6.02
shuttle-c0-vs-c4.txt	9	1829	13.87
shuttle-c2-vs-c4.txt	9	129	20.50
vehicle0.txt	18	846	3.25
vehicle1.txt	18	846	2.90
vehicle2.txt	18	846	2.88
vehicle3.txt	18	846	2.99
vowel0.txt	13	988	9.98
wisconsin.txt	9	683	1.86
yeast-0-5-6-7-9_vs_4.txt	8	528	9.35
yeast-1-2-8-9_vs_7.txt	8	947	30.57
yeast-1-4-5-8_vs_7.txt	8	693	22.10
yeast-1_vs_7.txt	7	459	14.30
yeast-2_vs_4.txt	8	514	9.08
yeast-2_vs_8.txt	8	482	23.10
yeast1.txt	8	1484	2.46
yeast3.txt	8	1484	8.10
yeast4.txt	8	1484	28.10
yeast5.txt	8	1484	32.73
yeast6.txt	8	1484	41.40
ecoli-0-3-4_vs_5	7	200	9.00
glass-0-1-4-6_vs_2	9	172	9.12
glass-0-1-5_vs_2	9	214	2.06
led7digit-0-2-4-5-6-7-8-9_vs_1	7	443	10.97
yeast-0-2-5-7-9_vs_3-6-8	8	1004	9.14

Table 3.1: Data Description

Algorithm 3 Algorithm for Third Hypothesis

```
for cluster ∈ points_under_each_cluster do
    points_under_this_cluster ⟵ AllPointsUnderThisCluster
    minority_point_under_this_cluster ⟵ ϕ
    majority_point_under_this_cluster ⟵ ϕ
    for i ∈ points_under_this_cluster do
        if i ∈ idx_min then
            minority_point_under_this_cluster.append(i)
        else
            majority_point_under_this_cluster.append(i)
    number_of_points_to_choose_from_this_cluster ⟵ len(majority_point_under_this_cluster) ×
PercentageForMajority
    majority_distance ⟵ ϕ
    if length_of_minority_data = 0 then
        selected_majority_idx.extend(majority_point_under_this_cluster)
    else
        for minority_data ∈ minority_point_under_this_cluster do
            for majority_data ∈ majority_point_under_this_cluster do
                majority_distance[majority_data] ⟵ Euclidian_Distance(minority_data, majority_data)
            sorted_majority_distance ⟵ sort(majority_distance)
            neighborhood_majority_instances ⟵ sorted_majority_distance[number_of_selected_neighbor :
]
            for instances ∈ neighborhood_majority_instances do
                if instances ∉ neighborhood_majority_idx then
                    neighborhood_majority_idx.append(instances)
=0
```

True Positive: For correctly predicted Positive class.

False Positive: For incorrectly predicted Positive class

False Negative: For incorrectly predicted Negative class

True Negative: For correctly predicted Negative class

(a) **F1-Score:** F1-Score is the weighted average of Precision and Recall. The range of F1-Score is 0 to 1.

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{3.1}$$

(b) **Matthews correlation coefficient:** Matthews correlation coefficient (MCC) is arguably the most elegant way to find out the quality of a binary classification problem. The high value (close to 1) means that both classes are predicted well.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.2}$$

(c) **Area Under Receiver Operator Curve:** This matrix can measure the performance of a classification problem in terms of a threshold value. It interprets how the model is able to differentiate between classes. The highest value is 1 which indicates that the model can perfectly distinguish all the classes and the lowest value is 0 which means the model has no capability to

separate the classes. This curve can be plotted with True Positive Rate (TPR) against False Positive Rate (FPR) where TPR is on the y-axis and FPR is on the x-axis.

$$TPR : \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.3}$$

$$FPR : \frac{FalsePositive}{FalsePositive + TrueNegative} \tag{3.4}$$

(d) **Area Under Precision-Recall:** This metric is used for evaluating binary classification for imbalanced data. It is a graphical representation of how the model separates all the classes. A threshold has been selected between 0 to 1 and if the probability is greater than the threshold then it is a positive class, otherwise, it is a negative class. This curve is created in terms of precision which is on the y-axis and recall which is on the x-axis.

$$Precision : \frac{TruePositive}{TruePositive + FalsePositive} \tag{3.5}$$

$$Recall : \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.6}$$

3.0.4 Experimental Analysis

We use the Normalization method to create a vector. Then we used Stratified K Fold Cross Validation to split data into train set and test set where the value of K is equal to 5. After that, we sample all data based on our hypothesis. The cluster numbers we select for clustering are 5,10,15,20 and 23. And finally, we use the Adaboost classifier to develop our model. The parameter we use for Adaboost is n_estimators = 65, learning_rate= 0.1. We calculate F1-Score, MCC, auROC, and auPR which are shown in table 3. We take the highest value for each metric from all clusters. The summary of the ablation is given below:

Table 3.2: Summary of the experiments conducted in the ablation study.

experiment	hypothesis-1	hypothesis-2	hypothesis-3
method-1	✓		
method-2		✓	
method-3			✓
method-4	✓	✓	
method-5	✓		✓
method-6		✓	✓
method-7	✓	✓	✓

3.0.5 Baseline with Method 1, 2, 3

In this experiment, we compare our three main hypotheses with the baseline method named the Custboost algorithm. We apply all of our hypotheses individually to all datasets and then compare them with Custboost in terms of F1-Score, MCC, AUPR, and AUROC. The comparison show that

all of our hypothesis performs better than Custboost and particularly our second hypothesis is better than the others. The results are shown in Table 3.3 and Table 3.4

3.0.6 Baseline with the Hybrid method

In this set of experiments, first, we combine our all hypotheses according to Table 4.3. Then we compare them with the baseline model in terms of F1-Score, MCC, AUPR, and AUROC. The results are shown in Table 3.5 and Table 3.6.

3.0.6.1 Comparison With Other Existing Work

In this section, we compare our results with other existing works discussed in the literature. We compare our methods with four different methods which are Rayhan et al [8], Zhang et al [19], Lin et al [11] and Vuttipittayamongkol et al [1]. We experiment with our technique with all the datasets these authors have used. We have shown the comparison result below. We find that our methods give us the best result for almost all datasets. Here, Vuttipittayamongkol et al compare the result in terms of F-1 Score which is shown in Table ,3.7 and the rest of them did their experiments in terms of auPR which is described in Table 3.8.

3.0.7 Performance on real-life dataset

In this section, we performed our model with a real-life dataset collected [9]. We used MLP and SVM as classifiers and calculate accuracy, F-1 score, True Positive Rate (TPR), and True Negative Rate (TNR). The table shows the performance comparison where we can see our method outperformed the existing method.

3.1 Graphs

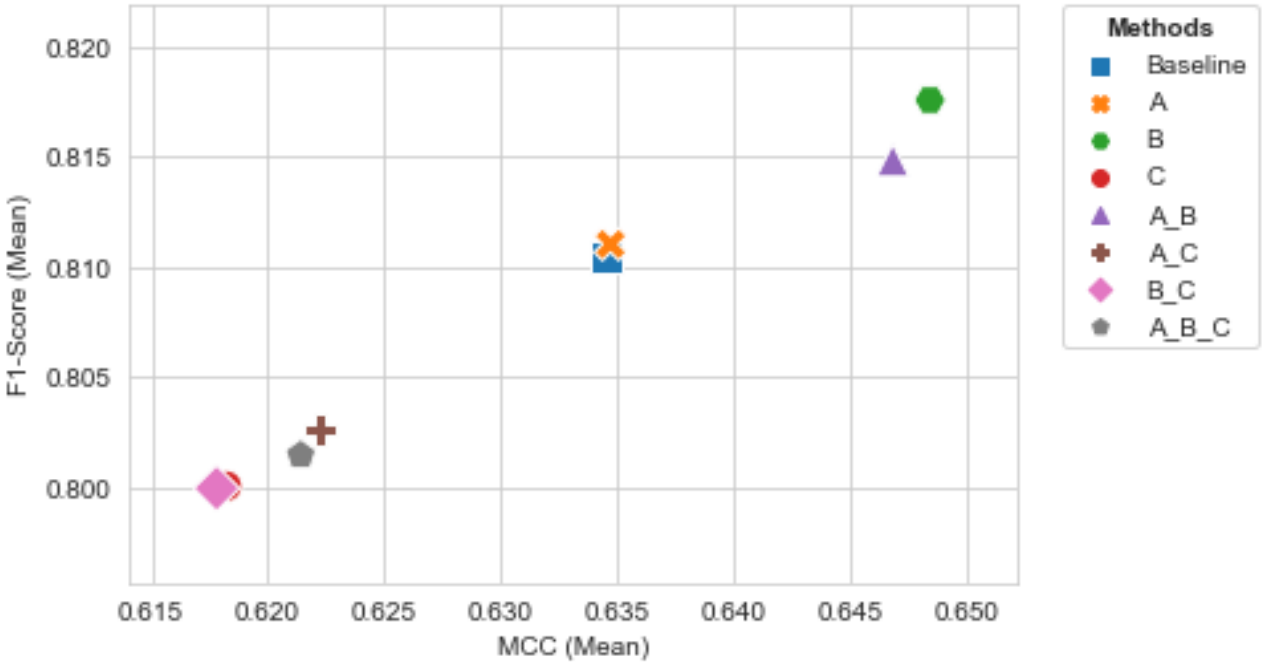


Figure 3.5: All Method's Performance Comparison (F1-Score Vs MCC).

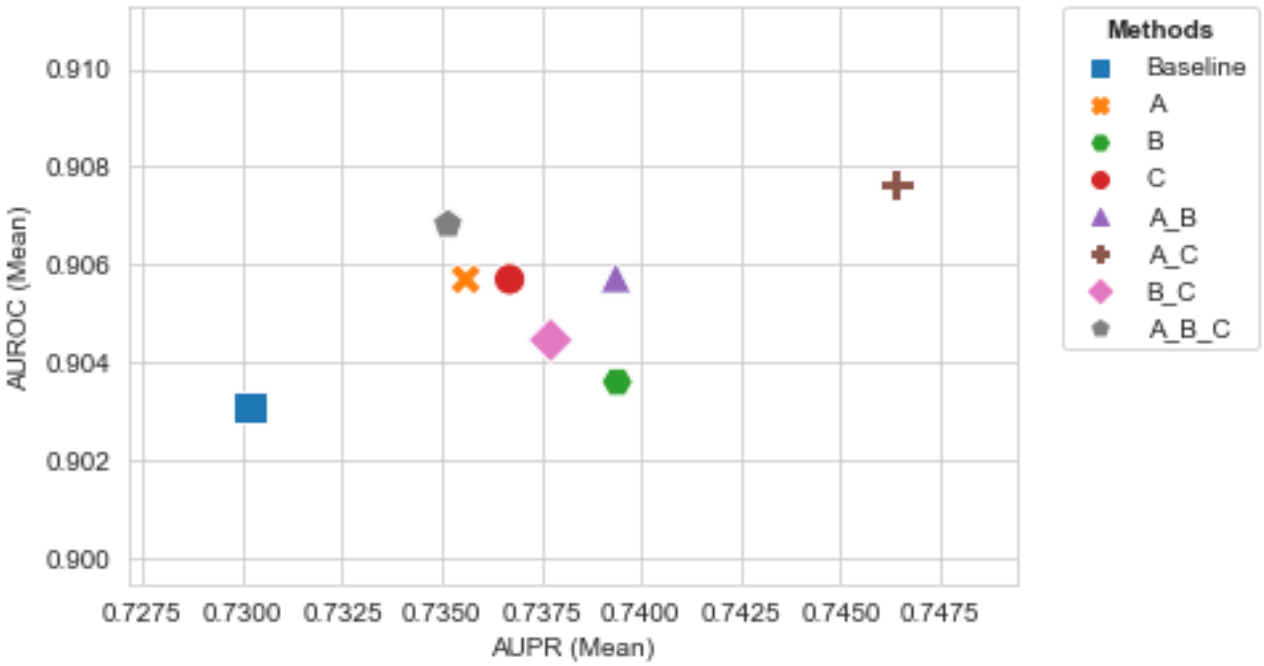


Figure 3.6: All Method's Performance Comparison (AUROC Vs AUPR).

	F1_Score				MCC			
dataset_name	baseline	a	b	c	baseline	a	b	c
abalone19.txt	0.4981	0.4981	0.4981	0.4981	0	0	0	0
abalone9-18.txt	0.6698	0.667	0.6417	0.6218	0.3909	0.4237	0.3673	0.3655
ecoli-0-1-3-7_vs_2-6.txt	0.8097	0.7788	0.7973	0.7964	0.6421	0.5766	0.6	0.6471
ecoli-0-3-4_vs_5.txt	0.9006	0.9311	0.908	**0.9315**	0.807	0.8677	0.8296	0.8712
ecoli-0_vs_1.txt	0.9849	0.9802	0.9844	0.9755	0.9705	0.9609	0.9701	0.9534
ecoli1.txt	0.8721	0.8592	0.8879	0.8912	0.754	0.7357	0.7803	0.7856
ecoli2.txt	0.8868	0.906	0.8998	0.9076	0.7786	0.8138	0.8038	0.8223
ecoli3.txt	0.8106	0.8076	0.7926	0.8153	0.6372	0.6238	0.5989	0.6356
ecoli4.txt	0.8794	0.8969	0.8929	0.8983	0.7648	0.8047	0.7923	0.8155
glass-0-1-2-3_vs_4-5-6.txt	0.9246	0.9208	0.9164	0.931	0.8629	0.8488	0.8444	0.8669
glass-0-1-4-6_vs_2.txt	0.556	0.5943	0.6869	0.529	0.1222	0.1975	0.3819	0.066
glass-0-1-5_vs_2.txt	0.5248	0.5652	0.6051	0.5642	0.0709	0.139	0.229	0.1362
glass-0-1-6_vs_2.txt	0.5843	0.574	0.665	0.6215	0.1996	0.1502	0.3453	0.2761
glass-0-1-6_vs_5.txt	0.9075	0.8562	0.8942	0.8943	0.8377	0.7341	0.8162	0.7886
glass0.txt	0.7737	0.7785	0.788	0.7711	0.5749	0.5867	0.6009	0.554
glass1.txt	0.7251	0.7151	0.7295	0.7434	0.4648	0.447	0.4702	0.4974
glass2.txt	0.6109	0.581	0.6562	0.5497	0.2522	0.1743	0.3286	0.1021
glass4.txt	0.8963	0.923	0.8224	0.832	0.7978	0.8623	0.6684	0.6765
glass5.txt	0.8593	0.8297	0.9296	0.8672	0.7282	0.6831	0.8777	0.7541
glass6.txt	0.9294	0.9306	0.9492	0.9374	0.8613	0.8642	0.9011	0.8787
haberman.txt	0.6331	0.6406	0.6325	0.613	0.2703	0.2847	0.2815	0.266
iris0.txt	1	1	1	0.9923	1	1	1	0.9852
led7digit-0-2-4-5-6-7-8-9_vs_1.txt	0.881	0.873	0.8813	0.8667	0.7655	0.7473	0.7657	0.7421
new-thyroid1.txt	0.9909	1	1	0.9909	0.9826	1	1	0.9826
newthyroid2.txt	0.9919	0.9919	0.9909	1	0.9845	0.9845	0.9826	1
page-blocks-1-3_vs_4.txt	0.9763	0.9821	0.9739	0.9835	0.9553	0.9657	0.9514	0.9682
page-blocks0.txt	0.8806	0.8815	0.8828	0.8705	0.7619	0.7634	0.766	0.747
pima.txt	0.6222	0.6256	0.6204	0.5653	0.2892	0.2995	0.2967	0.2046
segment0.txt	0.9893	0.9911	0.9893	0.9883	0.9788	0.9823	0.9787	0.9767
shuttle-c0-vs-c4.txt	0.9957	0.9979	0.9979	0.9957	0.9915	0.9958	0.9958	0.9915
shuttle-c2-vs-c4.txt	1	1	1	0.9646	1	1	1	0.9386
vehicle0.txt	0.9352	0.928	0.9316	0.9253	0.8724	0.8577	0.865	0.852
vehicle1.txt	0.7037	0.7145	0.7082	0.6534	0.4291	0.455	0.4532	0.3341
vehicle2.txt	0.9416	0.9345	0.9344	0.9359	0.8862	0.8723	0.8726	0.8732
vehicle3.txt	0.6889	0.687	0.6794	0.6349	0.3859	0.3866	0.373	0.349
vowel0.txt	0.9371	0.9436	0.9394	0.9418	0.879	0.8877	0.8806	0.8863
wisconsin.txt	0.9309	0.923	0.926	0.9141	0.8625	0.8476	0.8535	0.8292
yeast-0-2-5-7-9_vs_3-6-8.txt	0.899	0.8894	0.8927	0.894	0.8006	0.7813	0.7873	0.7929
yeast-0-5-6-7-9_vs_4.txt	0.7944	0.7808	0.7837	0.7665	0.5991	0.5698	0.5762	0.5724
yeast-1-2-8-9_vs_7.txt	0.5782	0.5672	0.5419	0.5422	0.239	0.1742	0.1105	0.1116
yeast-1-4-5-8_vs_7.txt	0.4889	0.5175	0.5179	0.4889	0	0.0765	0.0802	0
yeast-1_vs_7.txt	0.6611	0.6745	0.6768	0.6596	0.3646	0.3779	0.3793	0.3601
yeast-2_vs_4.txt	0.8894	0.8822	0.889	0.8862	0.7809	0.7685	0.7812	0.7864
yeast-2_vs_8.txt	0.8015	0.7955	0.7907	0.8091	0.6569	0.6307	0.6305	0.666
yeast1.txt	0.7183	0.7245	0.7131	0.6972	0.4415	0.453	0.4286	0.412
yeast3.txt	0.8852	0.88	0.8894	0.8689	0.7714	0.7624	0.7812	0.741
yeast4.txt	0.6949	0.6817	0.6941	0.563	0.4036	0.3877	0.3957	0.1722
yeast5.txt	0.8438	0.8739	0.851	0.852	0.6989	0.7539	0.7101	0.7074
yeast6.txt	0.7553	0.7654	0.7868	0.7624	0.5215	0.5415	0.5915	0.551
Average	0.8104	0.8110	**0.8175**	0.8000	0.6344	0.6347	**0.6484**	0.6182

Table 3.3: Comparison between Baseline and Method 1,2,3 in terms of F1-Score and MCC

	auPR				auROC			
dataset_name	baseline	a	b	c	baseline	a	b	c
abalone19.txt	0.9973	0.9975	0.9976	0.9971	0.7627	0.7715	0.7913	0.7603
abalone9-18.txt	0.4463	0.4557	0.4351	0.4518	0.8402	0.8424	0.8249	0.8345
ecoli-0-1-3-7_vs_2-6.txt	0.7083	0.811	0.7278	0.8322	0.9197	0.9488	0.9445	0.9622
ecoli-0-3-4_vs_5.txt	0.8439	0.9298	0.925	0.9025	0.9819	0.975	0.9826	0.9799
ecoli-0_vs_1.txt	0.9979	0.9985	0.9985	0.9986	0.9964	0.9975	0.9975	0.9975
ecoli1.txt	0.9117	0.8911	0.9049	0.9048	0.961	0.9628	0.9601	0.9664
ecoli2.txt	0.8738	0.8876	0.8442	0.8654	0.9451	0.9501	0.9447	0.9537
ecoli3.txt	0.7333	0.7037	0.6549	0.7306	0.9534	0.9443	0.9299	0.9379
ecoli4.txt	0.8835	0.9195	0.9007	0.9195	0.9755	0.9794	0.9814	0.9865
glass-0-1-2-3_vs_4-5-6.txt	0.952	0.9325	0.9519	0.9652	0.9762	0.9752	0.9761	0.9814
glass-0-1-4-6_vs_2.txt	0.2869	0.2646	0.3065	0.2175	0.8101	0.7771	0.7964	0.7802
glass-0-1-5_vs_2.txt	0.2331	0.2666	0.2947	0.2752	0.7487	0.7554	0.7478	0.7659
glass-0-1-6_vs_2.txt	0.2556	0.3406	0.4034	0.2683	0.7431	0.796	0.7543	0.7774
glass-0-1-6_vs_5.txt	0.8917	0.8833	0.9333	0.8917	0.9943	0.9943	0.9957	0.9943
glass0.txt	0.7618	0.7994	0.8049	0.8104	0.8681	0.8838	0.8838	0.8787
glass1.txt	0.7118	0.7434	0.6971	0.7269	0.7942	0.7994	0.7932	0.8196
glass2.txt	0.3162	0.2972	0.329	0.2737	0.8191	0.7967	0.7949	0.835
glass4.txt	0.8111	0.826	0.8111	0.8312	0.9229	0.9401	0.935	0.9393
glass5.txt	0.8575	0.8617	0.9583	1	0.9902	0.9902	0.9976	1
glass6.txt	0.9371	0.883	0.9088	0.9276	0.9694	0.9584	0.9653	0.9715
haberman.txt	0.4524	0.4322	0.4519	0.4541	0.6681	0.6691	0.6704	0.7049
iris0.txt	1	1	1	0.9933	1	1	1	0.99
led7digit-0-2-4-5-6-7-8-9_vs_1.txt	0.7842	0.7928	0.7833	0.7966	0.9609	0.966	0.9607	0.9572
new-thyroid1.txt	1	1	1	0.9982	1	1	1	0.9996
newthyroid2.txt	1	1	1	1	1	1	1	1
page-blocks-1-3_vs_4.txt	0.9857	0.9857	0.9857	0.9857	0.9989	0.9989	0.9989	0.9991
page-blocks0.txt	0.8788	0.8757	0.8734	0.8763	0.9807	0.981	0.9801	0.9812
pima.txt	0.5363	0.5136	0.5089	0.5039	0.6972	0.6918	0.6861	0.6898
segment0.txt	0.9955	0.9952	0.995	0.9953	0.9992	0.9991	0.9991	0.9992
shuttle-c0-vs-c4.txt	0.9962	0.9962	0.9962	0.9962	0.9997	0.9997	0.9997	0.9997
shuttle-c2-vs-c4.txt	1	1	1	0.9538	1	1	1	0.95
vehicle0.txt	0.9693	0.9597	0.9645	0.9629	0.9909	0.9883	0.9894	0.9896
vehicle1.txt	0.5751	0.5839	0.5876	0.5771	0.819	0.8225	0.8233	0.814
vehicle2.txt	0.9288	0.915	0.928	0.9303	0.9809	0.9785	0.9804	0.9817
vehicle3.txt	0.567	0.5858	0.5768	0.5955	0.8007	0.8079	0.8014	0.806
vowel0.txt	0.963	0.9589	0.9595	0.9517	0.9962	0.9951	0.9955	0.9942
wisconsin.txt	0.9583	0.9536	0.9541	0.955	0.9776	0.977	0.976	0.9765
yeast-0-2-5-7-9_vs_3-6-8.txt	0.8228	0.8254	0.8393	0.7919	0.9252	0.9213	0.9319	0.9226
yeast-0-5-6-7-9_vs_4.txt	0.5824	0.5785	0.5736	0.524	0.8291	0.8466	0.8525	0.8337
yeast-1-2-8-9_vs_7.txt	0.2551	0.2316	0.2271	0.2589	0.7652	0.7341	0.7594	0.7633
yeast-1-4-5-8_vs_7.txt	0.1394	0.1393	0.1462	0.1477	0.6642	0.6769	0.667	0.6692
yeast-1_vs_7.txt	0.3883	0.4051	0.412	0.3948	0.8158	0.807	0.8114	0.8149
yeast-2_vs_4.txt	0.8688	0.8954	0.8784	0.8629	0.9736	0.9783	0.9698	0.969
yeast-2_vs_8.txt	0.5628	0.5625	0.5187	0.5941	0.8416	0.8953	0.8284	0.8669
yeast1.txt	0.6146	0.6221	0.6204	0.627	0.8025	0.8035	0.8011	0.8026
yeast3.txt	0.8256	0.8393	0.832	0.8304	0.9657	0.97	0.9694	0.9693
yeast4.txt	0.4017	0.4259	0.4228	0.4049	0.895	0.8972	0.8892	0.8841
yeast5.txt	0.737	0.7002	0.7707	0.7658	0.9868	0.9873	0.987	0.9885
yeast6.txt	0.5787	0.579	0.6353	0.5774	0.9432	0.9481	0.9519	0.9404
Average	0.7301	0.7356	0.7393	0.7366	0.9030	0.9056	0.9086	0.9057

Table 3.4: Comparison between Baseline and Method 1,2,3 in terms of auPR and auROC

	F1-Score					MCC				
dataset_name	baseline	a_b	a_c	b_c	a_b_c	baseline	a_b	a_c	b_c	a_b_c
abalone19.txt	0.4981	0.4981	0.4981	0.4981	0.4981	0	0	0	0	0
abalone9-18.txt	0.6698	0.6641	0.6191	0.6182	0.6198	0.3909	0.3987	0.3188	0.3278	0.3437
ecoli-0-1-3-7_vs_2-6.txt	0.8097	0.8973	0.8306	0.7788	0.764	0.6421	0.8204	0.6803	0.5936	0.5402
ecoli-0-3-4_vs_5.txt	0.9006	0.9201	0.9037	0.9127	0.9284	0.807	0.8557	0.8125	0.8334	0.8625
ecoli-0_vs_1.txt	0.9849	0.985	0.9752	0.9752	0.9801	0.9705	0.9708	0.9521	0.9521	0.9611
ecoli1.txt	0.8721	0.8748	0.8953	0.9004	0.9004	0.754	0.7591	0.7945	0.8036	0.8078
ecoli2.txt	0.8868	0.9088	0.9004	0.9013	0.9009	0.7786	0.8209	0.8078	0.804	0.8052
ecoli3.txt	0.8106	0.8052	0.8006	0.8181	0.7887	0.6372	0.6237	0.6087	0.6389	0.6099
ecoli4.txt	0.8794	0.8579	0.8758	0.8798	0.8663	0.7648	0.7449	0.7649	0.7796	0.7463
glass-0-1-2-3_vs_4-5-6.txt	0.9246	0.9116	0.9235	0.9245	0.9183	0.8629	0.8294	0.856	0.8578	0.841
glass-0-1-4-6_vs_2.txt	0.556	0.5798	0.5455	0.5612	0.5703	0.1222	0.1899	0.0974	0.1347	0.1514
glass-0-1-5_vs_2.txt	0.5248	0.6569	0.5911	0.6	0.6188	0.0709	0.3611	0.2078	0.2229	0.2405
glass-0-1-6_vs_2.txt	0.5843	0.5987	0.551	0.5489	0.5541	0.1996	0.2074	0.1116	0.1163	0.1312
glass-0-1-6_vs_5.txt	0.9075	0.9075	0.9124	0.9091	0.9091	0.8377	0.8377	0.8337	0.8398	0.8398
glass0.txt	0.7737	0.7642	0.7802	0.7756	0.8005	0.5749	0.5604	0.5748	0.5611	0.6111
glass1.txt	0.7251	0.7516	0.7637	0.7624	0.7726	0.4648	0.517	0.5351	0.5343	0.5616
glass2.txt	0.6109	0.5942	0.5641	0.5891	0.5721	0.2522	0.2272	0.1495	0.1863	0.1815
glass4.txt	0.8963	0.8427	0.8333	0.8372	0.8317	0.7978	0.7381	0.6933	0.6955	0.6797
glass5.txt	0.8593	0.8785	0.8631	0.9018	0.8784	0.7282	0.7746	0.7397	0.8192	0.7921
glass6.txt	0.9294	0.9403	0.9336	0.9499	0.9356	0.8613	0.886	0.8734	0.9077	0.8774
haberman.txt	0.6331	0.6066	0.6109	0.5875	0.5991	0.2703	0.2182	0.262	0.2143	0.2433
iris0.txt	1	1	0.9923	0.9923	0.9923	1	1	0.9852	0.9852	0.9852
led7digit-0-2-4-5-6-7-8-9_vs_1.txt	0.881	0.8794	0.8806	0.8726	0.8703	0.7655	0.7634	0.7676	0.7522	0.7464
new-thyroid1.txt	0.9909	0.9909	0.9909	0.9909	0.9909	0.9826	0.9826	0.9826	0.9826	0.9826
newthyroid2.txt	0.9919	0.9919	0.9909	0.9829	0.9909	0.9845	0.9845	0.9826	0.9671	0.9826
page-blocks-1-3_vs_4.txt	0.9763	0.9846	0.9835	0.9835	0.9787	0.9553	0.9712	0.9682	0.9682	0.9594
page-blocks0.txt	0.8806	0.8825	0.8729	0.873	0.87	0.7619	0.7651	0.7525	0.7527	0.7465
pima.txt	0.6222	0.6262	0.5824	0.5603	0.5764	0.2892	0.3173	0.2268	0.19	0.2074
segment0.txt	0.9893	0.9902	0.9884	0.9884	0.9875	0.9788	0.9805	0.9769	0.9768	0.9751
shuttle-c0-vs-c4.txt	0.9957	0.9956	0.9979	0.9955	0.9957	0.9915	0.9913	0.9958	0.9912	0.9915
shuttle-c2-vs-c4.txt	1	1	1	1	1	1	1	1	1	1
vehicle0.txt	0.9352	0.9357	0.9303	0.9308	0.933	0.8724	0.873	0.8621	0.8628	0.8669
vehicle1.txt	0.7037	0.7067	0.677	0.6774	0.6514	0.4291	0.4484	0.3646	0.3695	0.3346
vehicle2.txt	0.9416	0.9375	0.9364	0.9381	0.9403	0.8862	0.8779	0.8734	0.8787	0.8818
vehicle3.txt	0.6889	0.6859	0.6318	0.6427	0.6345	0.3859	0.3812	0.3411	0.3542	0.3513
vowel0.txt	0.9371	0.9443	0.9584	0.9325	0.9437	0.879	0.889	0.9175	0.8688	0.8918
wisconsin.txt	0.9309	0.9244	0.9234	0.9101	0.914	0.8625	0.849	0.8493	0.8224	0.8292
yeast-0-2-5-7-9_vs_3-6-8.txt	0.899	0.8865	0.8891	0.8888	0.8929	0.8006	0.7779	0.7886	0.7813	0.7946
yeast-0-5-6-7-9_vs_4.txt	0.7944	0.777	0.7663	0.7542	0.7333	0.5991	0.5784	0.5654	0.5362	0.5075
yeast-1-2-8-9_vs_7.txt	0.5782	0.5491	0.5458	0.5452	0.5642	0.239	0.1574	0.1358	0.1321	0.1553
yeast-1-4-5-8_vs_7.txt	0.4889	0.516	0.4889	0.4889	0.4889	0	0.0662	0	0	0
yeast-1_vs_7.txt	0.6611	0.6547	0.6582	0.6673	0.6584	0.3646	0.3542	0.3747	0.3558	0.3594
yeast-2_vs_4.txt	0.8894	0.8772	0.8865	0.872	0.8773	0.7809	0.7629	0.7873	0.7652	0.7629
yeast-2_vs_8.txt	0.8015	0.8019	0.8103	0.7831	0.773	0.6569	0.6568	0.6629	0.6214	0.5985
yeast1.txt	0.7183	0.7173	0.6963	0.6871	0.6897	0.4415	0.44	0.4133	0.4051	0.4015
yeast3.txt	0.8852	0.8831	0.8729	0.8728	0.8787	0.7714	0.7688	0.7469	0.7495	0.7601
yeast4.txt	0.6949	0.7226	0.6026	0.5572	0.6084	0.4036	0.4598	0.2409	0.1757	0.2631
yeast5.txt	0.8438	0.8423	0.8495	0.8288	0.8387	0.6989	0.6965	0.7168	0.6612	0.6873
yeast6.txt	0.7553	0.7792	0.7522	0.7518	0.7913	0.5215	0.5618	0.535	0.5428	0.5991
Average	0.8104	0.8148	0.8025	0.7999	0.8014	0.6344	0.6468	0.6221	0.6177	0.6214

Table 3.5: Comparison between Baseline and Method 4,5,6,7 in terms of F1-Score and MCC

	auPR					auROC				
dataset_name	**baseline**	**a_b**	**a_c**	**b_c**	**a_b_c**	**baseline**	**a_b**	**a_c**	**b_c**	**a_b_c**
abalone19.txt	0.9973	0.9971	0.9974	0.997	0.9973	0.7627	0.7529	0.773	0.7473	0.7899
abalone9-18.txt	0.4463	0.4619	0.4554	0.4538	0.4309	0.8402	0.8339	0.84	0.8384	0.8408
ecoli-0-1-3-7_vs_2-6.txt	0.7083	0.8286	0.8261	0.7312	0.7111	0.9197	0.9379	0.9633	0.9536	0.9514
ecoli-0-3-4_vs_5.txt	0.8439	0.9405	0.9081	0.9219	0.9132	0.9819	0.9896	0.9792	0.9736	0.9653
ecoli-0_vs_1.txt	0.9979	0.9982	0.9989	0.9982	0.999	0.9964	0.9971	0.998	0.9968	0.9982
ecoli1.txt	0.9117	0.9066	0.9088	0.9111	0.9281	0.961	0.9619	0.9654	0.9651	0.9685
ecoli2.txt	0.8738	0.8605	0.8931	0.8833	0.894	0.9451	0.9455	0.9469	0.9529	0.9527
ecoli3.txt	0.7333	0.7459	0.7474	0.7538	0.6973	0.9534	0.9513	0.946	0.9398	0.9468
ecoli4.txt	0.8835	0.8632	0.8895	0.9054	0.8649	0.9755	0.9806	0.987	0.9802	0.9802
glass-0-1-2-3_vs_4-5-6.txt	0.952	0.933	0.9571	0.9586	0.9481	0.9762	0.9734	0.9761	0.9788	0.9731
glass-0-1-4-6_vs_2.txt	0.2869	0.2944	0.2879	0.2717	0.2529	0.8101	0.8067	0.789	0.7972	0.7775
glass-0-1-5_vs_2.txt	0.2331	0.3825	0.3038	0.3708	0.2811	0.7487	0.7871	0.7548	0.7589	0.764
glass-0-1-6_vs_2.txt	0.2556	0.2943	0.346	0.2932	0.2483	0.7431	0.7919	0.8286	0.7736	0.7967
glass-0-1-6_vs_5.txt	0.8917	0.9333	0.9167	0.9083	0.9667	0.9943	0.9957	0.9929	0.9943	0.9971
glass0.txt	0.7618	0.7634	0.8235	0.8094	0.8485	0.8681	0.873	0.8933	0.8833	0.9011
glass1.txt	0.7118	0.7321	0.7767	0.7562	0.7632	0.7942	0.7984	0.8439	0.8163	0.8226
glass2.txt	0.3162	0.3511	0.3212	0.3147	0.2914	0.8191	0.8224	0.8056	0.8166	0.82
glass4.txt	0.8111	0.877	0.8383	0.804	0.7942	0.9229	0.9417	0.941	0.9336	0.946
glass5.txt	0.8575	0.8917	0.9583	1	0.9583	0.9902	0.9939	0.9976	1	0.9976
glass6.txt	0.9371	0.9231	0.9244	0.9043	0.9171	0.9694	0.9594	0.9631	0.9577	0.9642
haberman.txt	0.4524	0.4266	0.4468	0.4263	0.4266	0.6681	0.6613	0.6705	0.6813	0.6724
iris0.txt	1	1	0.9933	0.9933	0.9933	1	1	0.99	0.99	0.99
led7digit-0-2-4-5-6-7-8-9_vs_1.txt	0.7842	0.7957	0.7961	0.7816	0.7973	0.9609	0.9579	0.9598	0.959	0.9627
new-thyroid1.txt	1	1	1	0.9964	0.9982	1	1	1	0.9992	0.9996
newthyroid2.txt	1	1	0.9968	0.9982	1	1	1	0.9992	0.9996	1
page-blocks-1-3_vs_4.txt	0.9857	0.9857	0.9881	0.9857	0.9857	0.9989	0.9989	0.9991	0.9989	0.9989
page-blocks0.txt	0.8788	0.8755	0.8773	0.8802	0.8755	0.9807	0.9803	0.981	0.9809	0.9805
pima.txt	0.5363	0.5096	0.5189	0.5199	0.5097	0.6972	0.6888	0.7013	0.6983	0.6956
segment0.txt	0.9955	0.996	0.9943	0.9944	0.9945	0.9992	0.9993	0.999	0.999	0.999
shuttle-c0-vs-c4.txt	0.9962	0.9962	0.9962	0.9962	0.9962	0.9997	0.9997	0.9997	0.9997	0.9997
shuttle-c2-vs-c4.txt	1	1	1	1	1	1	1	1	1	1
vehicle0.txt	0.9693	0.9588	0.9653	0.9635	0.9684	0.9909	0.9875	0.9897	0.9892	0.9901
vehicle1.txt	0.5751	0.5818	0.5981	0.5698	0.5741	0.819	0.8165	0.8243	0.8191	0.8185
vehicle2.txt	0.9288	0.9318	0.936	0.9295	0.9392	0.9809	0.9806	0.9835	0.9796	0.9808
vehicle3.txt	0.567	0.5729	0.5857	0.5828	0.5877	0.8007	0.8053	0.8122	0.8055	0.8097
vowel0.txt	0.963	0.9627	0.9555	0.9628	0.9606	0.9962	0.9948	0.9949	0.9955	0.9953
wisconsin.txt	0.9583	0.956	0.9559	0.9485	0.9568	0.9776	0.9797	0.9766	0.9742	0.9768
yeast-0-2-5-7-9_vs_3-6-8.txt	0.8228	0.8121	0.8101	0.8095	0.8125	0.9252	0.9249	0.9254	0.9214	0.9247
yeast-0-5-6-7-9_vs_4.txt	0.5824	0.5838	0.6041	0.5605	0.5565	0.8291	0.8412	0.8448	0.8655	0.8429
yeast-1-2-8-9_vs_7.txt	0.2551	0.204	0.2554	0.2119	0.2241	0.7652	0.7316	0.7442	0.7412	0.7655
yeast-1-4-5-8_vs_7.txt	0.1394	0.1085	0.1341	0.1412	0.1165	0.6642	0.6988	0.6628	0.6542	0.6447
yeast-1_vs_7.txt	0.3883	0.3605	0.395	0.3837	0.3899	0.8158	0.8262	0.8117	0.807	0.817
yeast-2_vs_4.txt	0.8688	0.8595	0.8747	0.864	0.8629	0.9736	0.9704	0.9717	0.9725	0.9636
yeast-2_vs_8.txt	0.5628	0.5789	0.5875	0.5362	0.5407	0.8416	0.8407	0.8601	0.8333	0.8549
yeast1.txt	0.6146	0.6196	0.6292	0.6267	0.6273	0.8025	0.8028	0.8068	0.8026	0.809
yeast3.txt	0.8256	0.8376	0.8467	0.8468	0.8549	0.9657	0.9697	0.9703	0.9691	0.9721
yeast4.txt	0.4017	0.4607	0.4185	0.419	0.412	0.895	0.8944	0.8801	0.8923	0.8873
yeast5.txt	0.737	0.6832	0.7957	0.7212	0.7595	0.9868	0.9865	0.9882	0.9878	0.987
yeast6.txt	0.5787	0.5928	0.5396	0.5522	0.5963	0.9432	0.9492	0.9421	0.9444	0.9424
Average	0.7301	0.7393	0.7463	0.7377	0.7351	0.9030	0.9057	0.9076	0.9044	0.9068

Table 3.6: Comparison between Baseline and Method 4,5,6,7 in terms of auPR and auROC

dataset_name	Method_1	Method_2	Method_3	Method_4	Our_Method
abalone9-18.txt	12.90	13.04	22.86	16.00	**66.47**
ecoli-0-1-3-7_vs_2-6.txt	50.00	50.00	**100.00**	66.67	86.39
ecoli1.txt	78.95	75.68	73.68	66.67	**90.75**
ecoli2.txt	90.00	90.00	90.00	85.71	**91.60**
ecoli3.txt	60.87	63.64	70.00	63.64	**82.79**
ecoli4.txt	88.89	88.89	**100.00**	80.00	90.16
glass0.txt	61.54	63.16	64.52	64.86	**77.92**
glass2.txt	33.33	40.00	33.33	28.57	**61.01**
glass4.txt	66.67	66.67	66.67	23.53	**87.72**
glass5.txt	0.00	0.00	0.00	0.00	**90.97**
new-thyroid1.txt	71.43	71.43	82.35	77.78	**100.00**
newthyroid2.txt	82.35	82.35	82.35	63.64	**100.00**
page-blocks-1-3_vs_4.txt	71.43	71.43	71.43	35.71	**99.17**
pima.txt	57.14	58.29	57.99	51.46	**62.63**
segment0.txt	58.56	70.65	73.86	57.78	**99.02**
vehicle0.txt	69.64	77.89	71.56	74.00	**93.57**
vehicle1.txt	45.03	46.74	48.81	47.13	**70.96**
vowel0.txt	94.12	94.12	**97.30**	85.71	94.71
wisconsin.txt	**94.95**	**94.95**	**94.95**	92.16	92.65
yeast-1_vs_7.txt	16.22	8.70	16.00	17.54	**67.56**
yeast-2_vs_4.txt	69.57	70.00	66.67	42.86	**89.02**
yeast3.txt	60.19	66.67	58.72	34.78	**89.11**
yeast4.txt	23.88	30.19	32.00	17.58	**70.49**

Table 3.7: Comparison with Vuttipittayamongkol et al [1] in terms of F1-Score

dataset_name	Rayhan er al [8]	Zhang et al [19]	Lin et al [11]_1	Lin et al [11]_2	Our_Method
abalone19.txt	0.7471	-	0.6840	0.7280	**0.7685**
abalone9-18.txt	0.7918	-	0.8080	0.8310	**0.8400**
ecoli-0-1-3-7_vs_2-6.txt	-	-	0.8380	0.8040	**0.9607**
ecoli-0_vs_1.txt	-	67.74	0.9820	0.9820	**0.9984**
ecoli1.txt	-	-	0.9400	0.9270	**0.9729**
ecoli2.txt	-	-	**0.9560**	0.9470	0.9491
ecoli3.txt	-	-	0.9090	0.9260	**0.9417**
ecoli4.txt	-	-	0.9490	0.9500	**0.9865**
glass-0-1-2-3_vs_4-5-6.txt	0.9704	-	0.9820	0.9700	**0.9852**
glass-0-1-6_vs_2.txt	-	-	0.7160	**0.7900**	0.7788
glass-0-1-6_vs_5.txt	-	-	0.9430	0.9640	**0.9971**
glass0.txt	0.8545	70.67	**0.8900**	0.8730	**0.8900**
glass1.txt	-	-	**0.8340**	0.8240	0.8272
glass2.txt	0.7398	-	0.7150	0.7600	**0.8390**
glass4.txt	-	60.63	0.8130	0.8530	**0.9550**
glass5.txt	0.9963	-	0.8880	0.9490	**1.0000**
glass6.txt	0.9457	-	0.9170	0.9050	**0.9721**
haberman.txt	-	-	0.6410	0.6030	**0.6852**
iris0.txt	-	-	0.9900	0.9900	**1.0000**
new-thyroid1.txt	-	-	0.9380	0.9730	**1.0000**
newthyroid2.txt	-	-	0.9560	0.9240	**1.0000**
page-blocks-1-3_vs_4.txt	0.9985	-	0.9370	0.9920	**0.9989**
page-blocks0.txt	-	-	0.9840	**0.9860**	0.9814
pima.txt	0.6803	-	**0.7700**	0.7580	0.6997
segment0.txt	-	-	0.9950	0.9960	**0.9993**
shuttle-c0-vs-c4.txt	-	-	**1.0000**	**1.0000**	0.9997
shuttle-c2-vs-c4.txt	-	67.9	**1.0000**	0.9880	**1.0000**
vehicle0.txt	-	-	0.9740	0.9900	**0.9909**
vehicle1.txt	-	-	0.7780	0.8320	**0.8374**
vehicle2.txt	-	-	0.9940	**0.9950**	0.9840
vehicle3.txt	-	-	0.8480	0.8270	**0.8544**
vowel0.txt	-	-	0.9550	0.9870	**0.9957**
wisconsin.txt	-	-	**0.9900**	**0.9900**	0.9820
yeast-0-5-6-7-9_vs_4.txt	-	66.44	0.8020	0.8690	**0.8844**
yeast-1-2-8-9_vs_7.txt	-	-	0.6870	0.6920	**0.7634**
yeast-1-4-5-8_vs_7.txt	-	-	0.6150	0.6270	**0.6837**
yeast-1_vs_7.txt	0.7896	**97.32**	0.7450	0.7680	0.8334
yeast-2_vs_4.txt	0.9615	-	0.9420	0.9770	**0.9783**
yeast-2_vs_8.txt	-	-	0.7550	**0.8680**	0.8432
yeast1.txt	-	84.31	0.7400	0.7470	**0.8668**
yeast3.txt	-	-	0.9580	0.9670	**0.9721**
yeast4.txt	0.8651	-	0.8260	0.8740	**0.8949**
yeast5.txt	0.9876	-	0.9820	0.9870	**0.9892**
yeast6.txt	0.9333	91.03	0.8410	0.9090	**0.9559**
Average	-	-	0.8728	0.8886	**0.9137**

Table 3.8: Comparison in terms of auROC

	Existing Method[9](MLP)	Our Method(MLP)	Existing Method[9](SVM)	Our Method(SVM)
Accuracy	87.24	91.32	84.88	89.87
F1-Score	0.86	0.87	.82	0.85
TPR	0.84	0.86	0.75	0.79
TNR	0.92	0.96	0.94	0.97

Table 3.9: Comparison With Real Life Dataset

Chapter 4

Customer Purchase Intention Prediction for Online Shopping

The methodology that we have followed in this paper is depicted in Figure 4.1. It starts by collecting a dataset from an e-commerce site followed by a pre-processing step where the categorical data are converted into numeric and also normalized. After that feature selection is performed, which is an optional phase in the machine learning methodology and depends on the performance of the features selected. It follows a balancing technique to address the imbalance problem. After that classifiers are designed for the selection of the final prediction model after performance evaluation. The rest of this section describes these steps in detail.

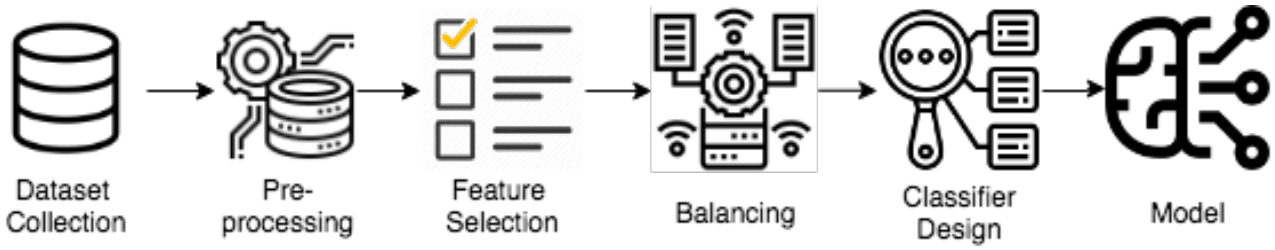


Figure 4.1: Workflow for our proposed model for Online Shopping.

4.0.1 Benchmark Dataset

In our study, we formulated the problem as a binary classification and collected the dataset that was previously used [9]. In this dataset, there are both categorical and numerical features. The summary of the features is given in Table 4.1 and Table 4.2. There are 12,330 records of different customers in a one-year period with different behavior. Among these 12,300 records, 10,422 (i.e., 84.5%) are with negative samples where customers did not complete their shopping, and the rest of the 1,908 records are with positive samples where customers completed their transaction.

4.0.2 Pre-processing

One-hot encoding converts categorical data into a numeric form and is provided to any ML algorithms for better predictive work. As mentioned earlier, we have some categorical features and we have used this approach to convert those data into numerical features. As we are also dealing with numeric

Table 4.1: Summary of numerical features.

Feature	Feature description	Min value	Max value	σ
Administrative	Number of pages visited by the visitor about account management	0	27	3.32
Administrative duration	Total amount of time (in seconds) spent by the visitor on account management-related pages	0	3398	176.70
Informational	Number of pages visited by the visitor about a Web site, communication and address information of the shopping site	0	24	1.26
Informational duration	Total amount of time (in seconds) spent by the visitor on informational pages	0	2549	140.64
Product related	Number of pages visited by a visitor about product related pages	0	705	44.45
Product-related duration	Total amount of time (in seconds) spent by the visitor on product related pages	0	63,973	1912.25
Bounce rate	Average bounce rate value of the pages visited by the visitor	0	0.2	0.04
Exit rate	Average exit rate value of the pages visited by the visitor	0	0.2	0.05
Page value	Average page value of the pages visited by the visitor	0	361	18.55
Special day	Closeness of the site visiting time to a special day	0	1.0	0.19

Table 4.2: Summary of categorical features.

Feature	**Feature description**	**Levels**
OperatingSystems	Operating system of the visitor	8
Browser	Browser of the visitor	13
Region	Geographic region from which the session has been started by the visitor	9
TrafficType	Traffic source by which the visitor has arrived at the Web site (e.g., banner, SMS, direct)	20
VisitorType	Visitor type as "New Visitor," "Returning Visitor," and "Other"	3
Weekend	Boolean value indicating whether the date of the visit is weekend	2
Month	Month value of the visit date	12
Revenue	Class label indicating whether the visit has been finalized with a transaction	2

features, we need to scale the data for better performance. Standard Scalar converts statistical data where 0 is the mean value and 1 is the standard variance. As the ML models are initialized to a starting value and gradually update themselves on the basis of error estimation, unscaled varying input ranges can make the model learn slowly and target variables having unscaled can affect the gradients causing the whole learning model to get into an ultimate failure. The class label was converted to two distinct labels 0 and 1 using a simple label encoding method.

4.0.3 Feature Selection

There are two types of features in the benchmark dataset that is used in this work. A summary of the features is given in Table 4.1 and Table 4.2. We have used the χ^2 based feature selection technique to select the best features. The feature selection reveals that out of 28 features 20 are significant. They are 'VisitorType_New_Visitor', 'SpecialDay', 'Month_May', 'Month_Mar', 'BounceRates', 'Month_Oct', 'ExitRates', 'Month_Feb', 'VisitorType_Returning_Visitor', 'Month_Dec', 'Month_Sep', 'Weekend', 'Month_June', 'Region', 'Browser', 'OperatingSystems', 'VisitorType_Other', 'TrafficType', 'Month_Jul' and 'Month_Aug'.

4.0.4 Handling Data Imbalance

As discussed earlier, out of 12,300 records, 10,422 records belong to the negative class, thus it creates a class imbalance problem. For this problem, the prediction would be biased as it tends to predict the negative class. Therefore, we need to resample the positive class proportion following the negative class proportion. For undersampling and oversampling, we have applied Synthetic Minority Oversampling Technique (SMOTE) [56] and random undersampling [57] methods.

4.0.5 Classification Algorithms

We have employed five classification algorithms in our experiments: Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), Multilayer perceptron (MLP), and XGBoost Classifier. In this section, we provide brief preliminaries on these algorithms.

i. **Decision Tree (DT):** Decision tree [58] is an explainable classification model that takes decisions based on features set as nodes in the decision tree. Decisions are taken on the nodes based on the values of the attributes. Different types of metrics such as entropy, information gain, Gini index, etc. are used for attribute selection in Decision trees. In our experiments, we have used entropy as the attribute selector and set a pre-pruning parameter max_depth = 5.

ii. **Support Vector Machine (SVM):** SVM [59] works by an optimal hyperplane separating the datasets which are defined as an optimization problem.

$$min\ \frac{1}{2}\ ||\ w^2\ ||\ +\ C\ \sum_{i=1}^{k} \epsilon_i\ subject\ to\ r^t\ (w^T x^t + w_0) \geq 1 - \epsilon_i$$

Here, w is a discriminant weight vector, C is the regularization parameter which is the control complexity of the model that is fitted to the data, $\epsilon = (\epsilon_1, \epsilon_2, \cdots, \epsilon_k)$ is the vector of the slack variables, and r^t is the actual value of sample t. In our experiments with SVM, we use the "rbf" kernel and set the C value equals to 7.

iii. **Multilayer perceptron (MLP):** Multilayer perceptron is an artificial neural network model [60] with feed forwarding connections. Our MLP model consists of three hidden layers with 28, 56 and 28 neurons in each, respectively. We have used stochastic gradient descent as the optimizer with learning rate set to 0.00005 and relu activation function [61].

iv. **Random Forest:** Random forest [62] learns a large number of individual decision tree to form an ensemble. It performs bootstrapping on the feature space and the decision is taken based on a voting mechanism. In our experiments, we have set the max_depth of the decision tree to 20 and the number of estimators to 100.

v. **XGBoost:** XGBoost [63] approximates a function by optimizing a specific loss function where different regularization methods are applied. The objective function (loss function and regularization), at iteration t that we need to minimize, is the following:

$$L^{(t)} = \sum_{(} i = 1)^{n} \; l(y_i, y_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

4.0.6 Performance Evaluation

We have used a train-test split on the dataset as mentioned in the original paper where the selected dataset was proposed [9]. This is done to enable a fair comparison among the methods. Note that all the experimental results that are reported in the next section represents performance on the test data after the dataset was split by 70% (train) to 30% (test).

In a typical binary classification task often the metrics are defined on confusion matrix for each model. In a confusion matrix, true positives (TP) are the correctly predicted positive instances, true negatives (TN) are the correctly predicted negative instances, false positives (FP) are the incorrectly predicted negative instances and the false negatives (FN) are the incorrectly predicted positive instances. We have employed the following metrics based on these terms.

The percentage of the correct prediction is denoted by accuracy. The model will perform better if the accuracy is high (close to 100%)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precision is the value of ratio of correctly predicted positive instances to the total predicted positive instances. Here, 1 is the highest value of precision and 0 is the lowest.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall or sensitivity or True Positive Rate (TPR) is the true positive rate defined as the ratio of true positive to the total positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

Specificity or True Negative Rate (TNR) is the probability that an actual negative will test negative. The value of TNR lies in between 0 to 1 where 1 is the highest value.

$$TNR = \frac{TP}{TP + FP} \tag{4.4}$$

F1-Score is the weighted average of Precision and Recall. The range of F1-Score is 0 to 1.

$$F1 - Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{4.5}$$

Matthews correlation coefficient (MCC) is arguably the most elegant way to find out the quality of a binary classification problem. The high value (close to 1) means that both classes are predicted well.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.6}$$

There are two metrics that are not dependent on the confusion metrics. They are the area under receiver operating characteristic curve (auROC) and the area under precision recall curve (auPR). ROC curve is a plot of TPR against FPR. Both of these curves are not dependent on the threshold chosen by the classifier and they are highly suitable for imbalanced data classification in addition to MCC and F1-score.

4.1 Experimental Analysis

We have used Python 3.7 version and scikit-learn library [64] for implementation of all the algorithms. Algorithms were run using the Jupyter notebook environment. All the experiments are run on the dataset after a 70-30 split to generate train and test data respectively. Methods like feature selection, oversampling and undersampling are only done on the train set to avoid any possible overfitting.

We have started with a baseline experiment and added the components to assess the strengths. Following sections present results from each of the experiments. A summary of the ablation is given in Table 4.3.

Table 4.3: Summary of the experiments conducted in ablation study.

experiment	baseline	feature selection	oversampling	undersampling
baseline-1	✓			
data sampling-1	✓		✓	
data sampling-2	✓			✓
feature selection-1	✓	✓		
combined-1	✓	✓	✓	
combined-2	✓	✓		✓

Table 4.4: Evaluation of classification with baseline methods.

Model	**Accuracy**	**Precision**	**F1 Score**	**TPR**	**TNR**	**auROC**	**auPR**	**MCC**
SVM	89.59	88.76	0.8887	0.35	0.92	0.878	0.691	0.5591
RF	90.51	89.93	0.9007	0.74	**0.93**	0.932	0.748	**0.6089**
MLP	88.05	88.08	0.8806	0.61	**0.93**	0.886	0.632	0.543
DT	**90.54**	**89.9**	**0.8998**	0.75	0.92	0.922	0.717	0.6049
XGBoost	89.94	89.14	0.8905	**0.76**	0.91	**0.937**	**0.749**	0.5672

4.1.1 Baseline Methods

The first set of experiments are carried out on the selected classifier with all features present. In this setting, we do not perform any data sampling to handle the imbalance. The results are reported in Table 4.4. The results show that Decision Tree provides the higher accuracy, precision, recall and F1-score while XGBoost provides better True Positive Rate, AUC Score and AUC-PR Score. However, Random Forest gives the highest MCC, which is 0.6089.

4.1.2 Baseline with Data Sampling

For the next set of experiments we employ data sampling techniques to handle the imbalance in our data. Firstly, we have applied oversampling techniques in our dataset. We have used SMOTE for oversampling of the data. The results are reported in Table 4.5. Here, we note that Random Forest performs better than other classifiers in terms of TNS, auORC auPR and MCC. However, XGBoost has similar performances and slightly better performance in terms of accuracy, precision, recall and f1-score. Note that overall TPR has lowered in this experiments compared to the baseline experiments.

Table 4.5: Classification Report with Oversampling

Model	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
SVM	88.94	88.62	0.8876	0.65	0.93	0.878	0.666	0.5564
RF	89.65	90.05	0.8982	0.65	**0.95**	**0.933**	**0.731**	**0.625**
MLP	87.21	87.2	0.8721	0.59	0.92	0.9	0.631	0.4931
DT	89.4	89.96	0.8964	0.65	**0.95**	0.919	0.689	0.6127
XGBoost	**90.21**	**90.11**	**0.9016**	**0.69**	0.94	0.93	0.724	0.6243

When we use the random under-sampling technique to balance our dataset, we get overall better performance from XGBoost except for the auPR Score, which is better for Random Forest. The results for this set of experiments are reported in Table 4.6. Here too, we note the downgrade in TPR. However, both oversampling and undersampling show the relative superiority of XGBoost as a classifier over the other methods.

Table 4.6: Classification Report with Under sampling

Model	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
SVM	87.51	88.73	0.88	0.58	0.94	0.897	0.663	0.564
RF	88.21	**90.4**	0.8893	0.58	**0.96**	0.93	**0.734**	0.6196
MLP	84.1	88.83	0.8555	0.49	**0.96**	0.9	0.66	0.5445
DT	87.02	89.88	0.8794	0.55	**0.96**	0.919	0.696	0.5961
XGBoost	**89.32**	90.27	**0.8969**	**0.64**	0.95	**0.93**	0.718	**0.6235**

4.1.3 Feature Selection

We have used χ^2 based on feature selection to select the best features. We have selected top k features according to the ranking keeping values of k in the set $\{10, 15, 20\}$. The results are reported in Table 4.7. Here, we note that the best accuracy is found using 20 top features selected and using the XGBoost classifier, which also shows higher precision, TPR, TNR, and auPR values. The decision

tree classifier, however, shows good performance when trained with 15 and 20 features in terms of F1-Score and MCC.

Table 4.7: Classification performance evaluation of feature selection with different top k features selected.

Model	k	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
SVM	20	89.56	88.79	0.8895	0.72	0.92	0.88	0.694	0.5675
	15	89.78	89.03	0.8918	0.72	0.92	0.877	0.701	0.5699
	10	89.51	88.76	0.8895	0.71	0.92	0.85	0.663	0.5619
RF	20	89.1	88.11	0.8787	0.75	0.91	0.932	0.743	0.5898
	15	89.13	88.14	0.8814	0.73	0.91	0.928	0.736	0.6025
	10	89.51	89.09	0.8926	0.68	0.93	0.923	0.707	0.5802
MLP	20	88.97	88.37	0.8859	0.67	0.92	0.902	0.669	0.5516
	15	88.4	87.44	0.877	0.67	0.91	0.898	0.622	0.5113
	10	88.23	87.73	0.8786	0.62	0.92	0.861	0.587	0.5291
DT	20	90.54	89.9	**0.8998**	0.76	0.93	0.923	0.731	**0.6049**
	15	90.54	89.9	**0.8998**	0.76	0.93	0.923	0.731	**0.6049**
	10	89.89	89.64	0.8975	0.68	0.94	0.926	0.7014	0.6037
XGBoost	20	**90.65**	**90.01**	0.898	**0.801**	**0.96**	**0.937**	**0.749**	0.5992
	15	90.19	89.45	0.8928	0.7804	0.92	0.936	0.742	0.5775
	10	89.83	89.01	0.8889	0.76	0.91	0.935	0.731	0.5611

To demonstrate the effectiveness of feature selection we have created two bubble graphs as shown in Figure 4.2. The first plot on the left of Figure 4.2 shows an MCC vs F1-Score plot and the second graph on the right shows an auPR vs auROC plot using bubbles of different shapes denoting different sizes of the features selected and the colors indicating the classifiers. The graphs reflect two different scenarios: i) how the classifiers are acting in terms of metrics that are dependent on the confusion matrix and independent of the confusion matrix, and ii) based on different features selected. It is evident that XGBoost and Decision tree classifiers outperform other classifiers.

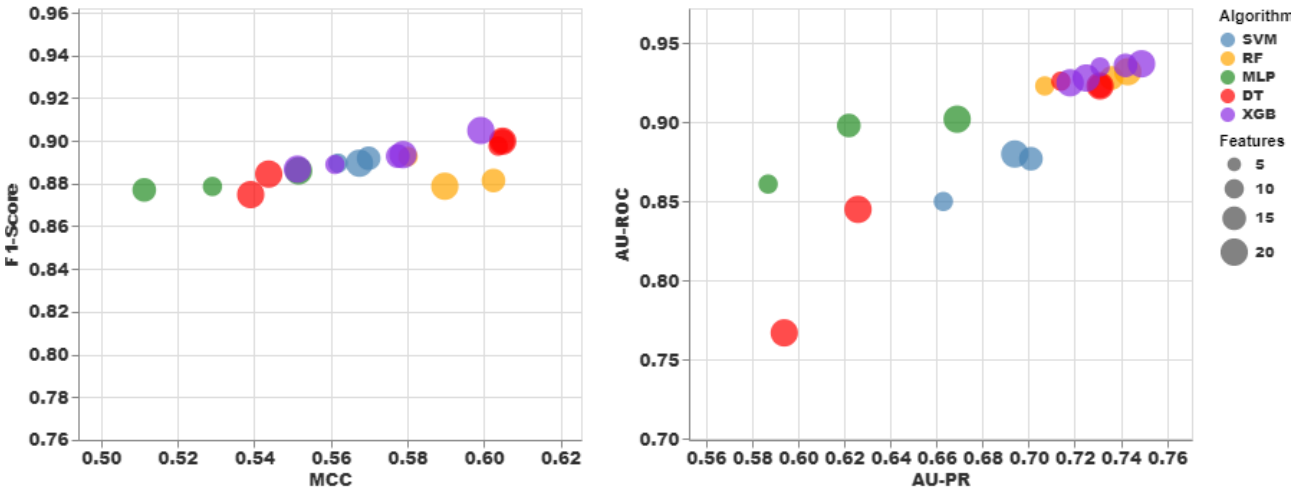


Figure 4.2: Plot of MCC vs F1-Score (left) and auPR vs auROC (right) for all algorithms using baseline with feature selection.

In Figure 4.3, we have shown a lolly-pop chart with all the feature importance from χ^2 test. Here, the feature importance values are scaled to show the relative importance.

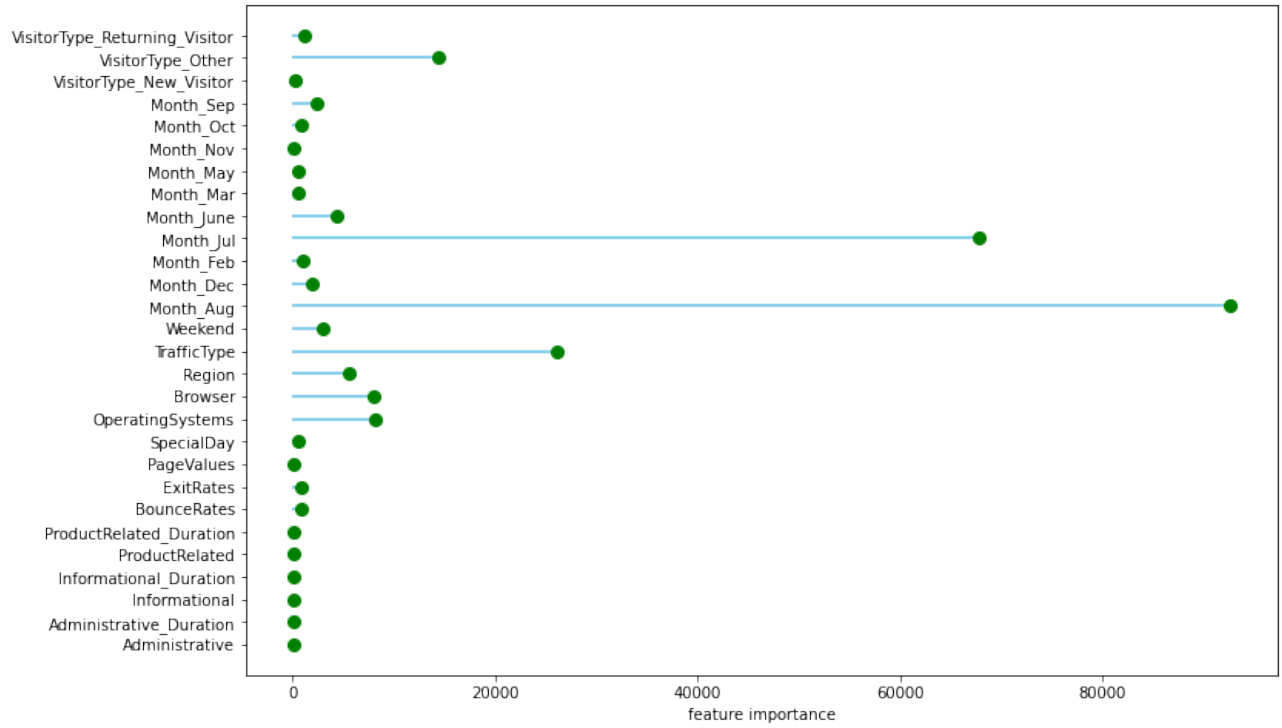


Figure 4.3: Plot of feature ranks from χ^2 test (importance values scaled).

4.1.4 Combined Evaluation

In the combined evaluation, we have combined feature selection technique with oversampling and undersampling respectively. In Table 4.8, we have reported the classification metrics from the experiments with different classifiers using feature selection and oversampling.

Table 4.8: Classification Report of Feature Selection with different k values and Oversampling

Model	k	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
SVM	20	88.86	88.75	0.888	0.64	0.93	0.877	0.665	0.5747
	15	88.61	89.07	0.8882	0.61	0.94	0.869	0.67	0.5769
	10	88.21	88.88	0.885	0.6	0.94	0.854	0.641	0.5756
RF	20	89.48	89.72	0.8959	0.65	0.94	0.933	0.737	0.6069
	15	89.13	89.67	0.8936	0.63	0.94	0.93	0.722	0.6072
	10	88.51	89.31	0.8884	0.61	0.94	0.923	0.701	0.6069
MLP	20	87.92	88.28	0.8808	0.6	0.93	0.881	0.611	0.5347
	15	86.32	88.68	0.8715	0.54	0.95	0.892	0.624	0.5542
	10	87.32	88.27	0.8766	0.47	0.94	0.863	0.647	0.576
DT	20	87.29	90.15	0.882	0.56	**0.96**	0.919	0.705	0.6009
	15	88.38	89.68	0.8887	0.6	0.95	0.921	0.699	0.6059
	10	89.16	**90.32**	0.8959	0.62	0.95	0.922	0.693	0.6046
XGBoost	20	**90.32**	89.82	0.8998	**0.72**	0.93	**0.937**	**0.754**	**0.6269**
	15	90.21	89.97	**0.9008**	0.7	0.94	0.935	**0.754**	0.6069
	10	89.86	89.72	0.8979	0.68	0.94	0.933	0.734	0.6029

We have also made two bubble plots similar to the previous section to show the relative performance of the classifiers. The plots are given in Figure 4.4. From the data reported in the table and presented via the plot, it is clearly evident that the variants of XGBoost are outperforming the rest of the classifiers in both cases.

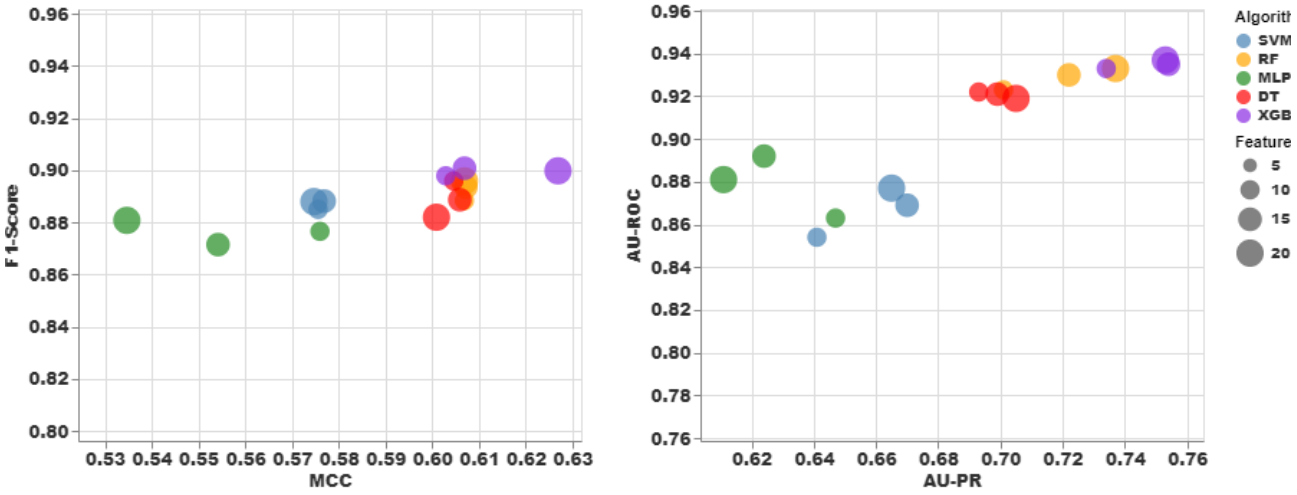


Figure 4.4: Plot of MCC vs F1-Score (left) and auPR vs auROC (right) for all algorithms using baseline with feature selection and oversampling.

The next set of experiments are done using feature selection and undersampling combined. The results are reported in Table 4.9. One of the interesting findings in case of undersampling is TNR for all the classifiers have improved compared to the previous experiments. However, MCC is not that high due to poor TPR. The overall comparison is shows using two sets of parameters in Figure 4.5. Again we note that the XGBoost classifiers are performing better than all other classifiers for both sets of parameters. However, feature selection with oversampling has a higher auPR compared to feature selection with undersampling.

Table 4.9: Classification Report of Feature Selection with different k values and Undersampling

Model	k	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
SVM	20	87.81	88.91	0.8825	0.59	0.94	0.892	0.665	0.5677
	15	87.62	88.83	0.881	0.58	0.94	0.893	0.671	0.5693
	10	87.37	88.84	0.8793	0.57	0.95	0.883	0.651	0.5693
RF	20	87.86	90.11	0.8861	0.58	**0.96**	0.925	0.71	0.6088
	15	87.62	89.9	0.8838	0.57	**0.96**	0.924	0.715	0.6009
	10	87.4	89.87	0.8822	0.56	**0.96**	0.918	0.675	0.5986
MLP	20	86.51	88.41	0.8722	0.55	0.95	0.899	0.66	0.5482
	15	85.78	89.35	0.8691	0.53	**0.96**	0.911	0.661	0.5717
	10	85.1	88.8	0.863	0.51	**0.96**	0.892	0.637	0.5508
DT	20	87.13	90.1	0.8807	0.56	**0.96**	0.921	0.697	0.6033
	15	87.13	90.1	0.8807	0.56	**0.96**	0.921	0.697	0.60339
	10	87.32	**90.16**	0.8822	0.56	**0.96**	0.917	0.687	0.6063
XGBoost	20	**88.94**	90.04	**0.8936**	**0.62**	**0.96**	**0.931**	0.712	**0.6138**
	15	88.81	89.76	0.8918	0.61	0.95	0.93	**0.718**	0.6042
	10	88.86	89.83	0.8924	**0.62**	0.95	0.927	0.698	0.6068

From the overall experiments it is evident that XGBoost is the best performing classifier. However, we also note that the second best performance is given by the very simple Decision Tree Classifier. We have done short experiments on parameter tuning for these two classifiers only. The results are presented in Table 4.10 and Table 4.11. We have used different types of max_depth $\{5, 10, 15\}$ for finding out the best outcome for Decision tree classifier. We note from the results in Table 4.10 where

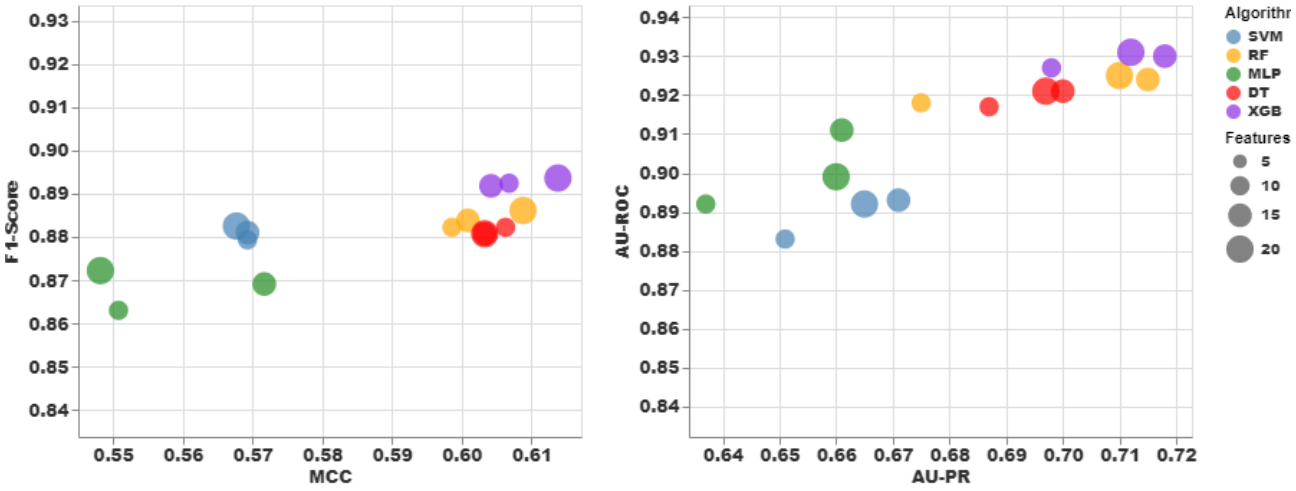


Figure 4.5: Plot of MCC vs F1-Score (left) and auPR vs auROC (right) for all algorithms using baseline with feature selection and undersampling.

max_depth=5 has the best results. We have selected this value for the explainability analysis part as well.

Table 4.10: Performance of Decision Tree with Different Max-Depth Value

max_depth	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
5	**90.54**	**89.9**	**0.8998**	**0.76**	**0.93**	**0.923**	**0.731**	**0.6049**
10	88.97	88.21	0.8844	0.68	0.92	0.845	0.626	0.5438
15	87.72	87.29	0.8748	0.61	0.92	0.767	0.594	0.5391

We have also tuned the hyper-parameters for XGBoost with different max_depth values from $\{2, 5, 10\}$. The results reported in Table 4.11 show that max_depth=2 has the best outcome.

Table 4.11: Performance of XGboost with Different Max-Depth Value

max_depth	Accuracy	Precision	F1 Score	TPR	TNR	auROC	auPR	MCC
2	**90.65**	**90.01**	**0.898**	**0.801**	**0.96**	**0.937**	**0.749**	**0.5992**
5	90.08	89.32	0.8935	0.75	0.92	0.928	0.725	0.5789
10	89.27	88.47	0.8867	0.7	0.92	0.925	0.718	0.5513

To show the overall performance comparison, we show two scatter plots in Figure 4.6 and Figure 4.7. Here, different classifiers with their best version are shown for six different ablation experiments. In Figure 4.6, the scatter plots of MCC vs F1-Score have been shown. Here, we note that XGBoost is the best performing classifier. Random Forest classifier is in the next position, however, it does not dominate the other classifiers in any of the metric.

Figure 4.7 shows the comparison for all classifiers as scatter plots of auROC vs auPR for different types of ablation experiments. Here too, we note that the best performing classifier is XGBoost followed by Random Forest. In both of these plots shown in Figure 4.6 and Figure 4.7, we note that the best performing variant of XGBoost is the one where we have employed feature selection and oversampling along with the baseline. However, only in terms on F1-Score, the baseline and feature selection variant are slightly better, but note that it has relatively poor performance in terms of the

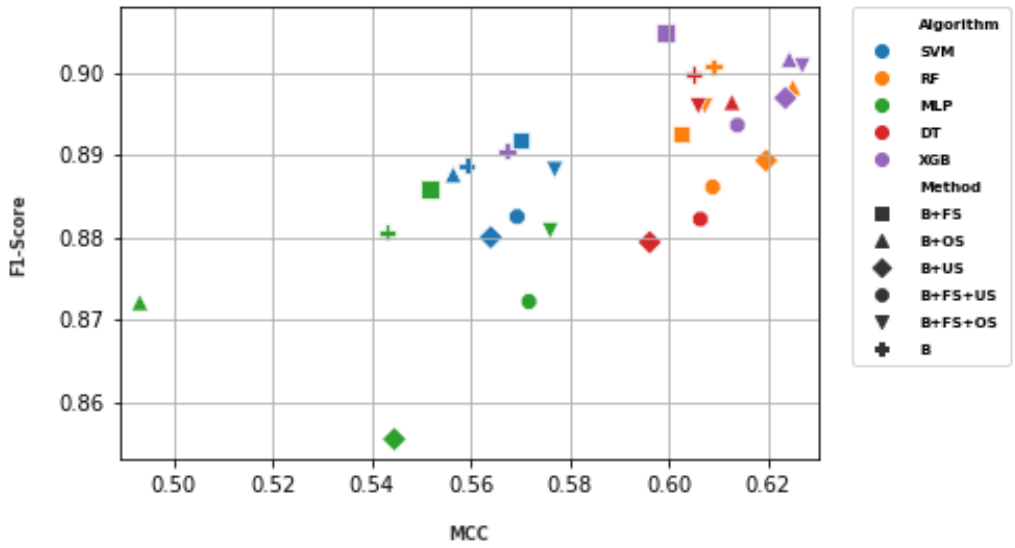


Figure 4.6: Comparison of MCC vs F1-Score for all algorithms. Here all six ablation experiments are shown: B=baseline, FS=Feature Selection, OS=Oversampling and US=Undersampling.

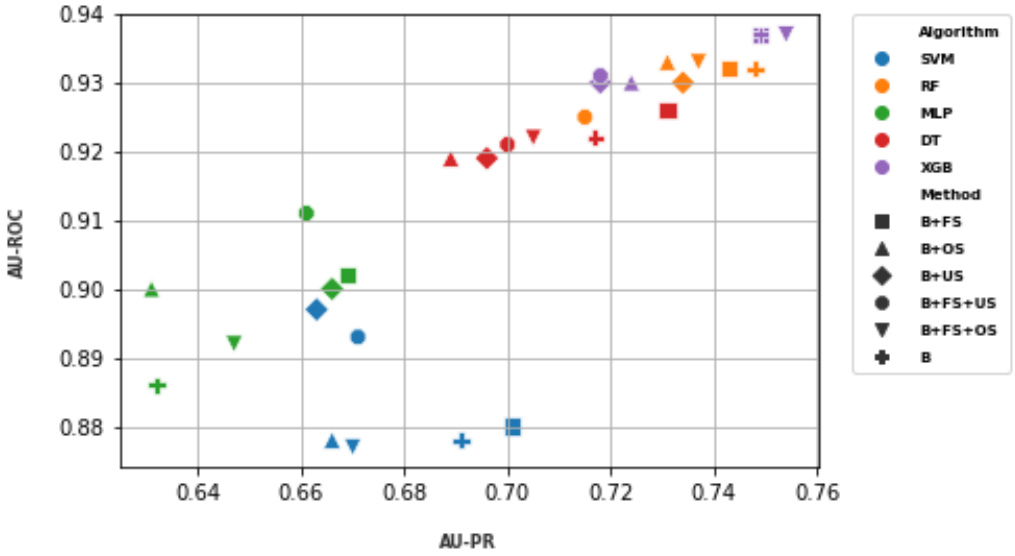
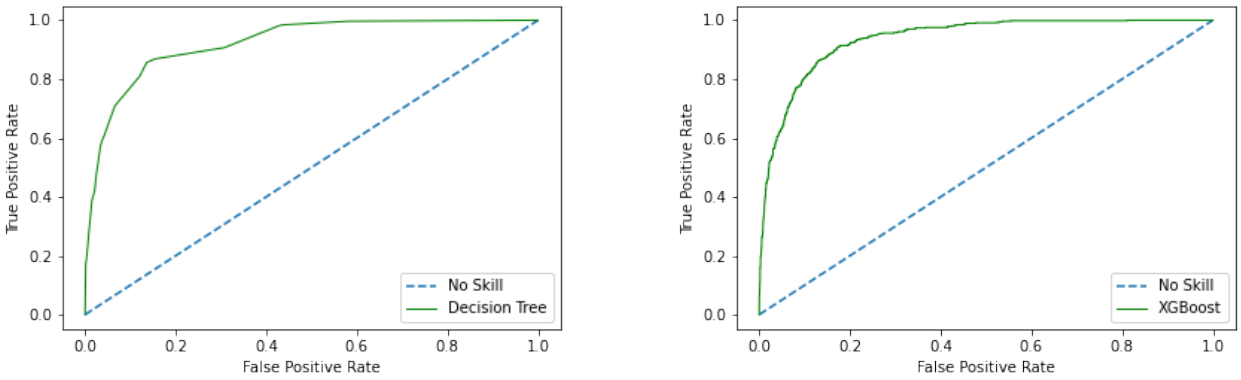


Figure 4.7: Comparison of auPR vs auROC for all algorithms. Here all six ablation experiments are shown: B=baseline, FS=Feature Selection, OS=Oversampling and US=Undersampling.

other metrics as shown in both figures. We have depicted the Receiver Operating characteristics (ROC) curves for two best classifiers: decision tree and XGBoost in Figure 4.8.

4.1.5 Comparison with the Existing Works

In this section, we compare the results from our experiments with those from the literature. We have chosen three other existing works for comparison: Sakar et al. [9], Baati et al. [51] and Song et al. [52]. All of these works have been applied on the same benchmark dataset that we have used in this paper. The comparison is shown in the results of Table 4.12. From the results reported in the table, we note the our method achieves best performance among all the methods in terms of accuracy, TNR and F1-Score. The other methods in the literature actually did not report auROC, auPR or MCC. Therefore, we have not included those metrics in this table. Sakar et al. [9] achieves higher TPR,



(a) (b)

Figure 4.8: Receiver Operating characteristics curves for (a) Decision Tree and (b) XGBoost Classifier.

however, the other metrics achieved by their methods show that their proposed method might be biased as compared to our proposed method. Also note that the method proposed by Song et al. [52] performs very much similar in terms of accuracy, but fails to do well in terms of F1-Score and TPR.

Table 4.12: Comparison with Existing Works

Matrics	Sakar et al [9]	Baati et al [51]	Song et al. [52]	This paper
Accuracy	87.24	86.78	90.15	**90.65**
True Positive Rate	**0.84**	0.62	0.73	0.801
True Negative Rate	0.92	0.91	0.93	**0.96**
F1-Score	0.86	0.60	0.65	**0.91**

4.1.6 Explainability Analysis

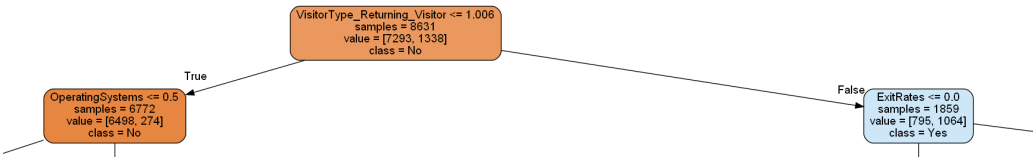
For the explainability analysis we have chosen the best model found by the decision tree with max_depth=5 using feature selection. The performance of this model is very similar to the best model XGBoost and yet this model provides us with the opportunity to understand or explain the nature of the classifier. The rest of the classifiers are black-box in nature and are not much suitable for such analysis.

The best decision tree with entropy as attribute selector and max_depth=5 is shown in parts of Figure 4.9. The root and its two children are shown in Figure 4.9(a). Note that, the values on which the decision tree is learned are normalized. We see that the features used here are PageValues, Month_November and Bounce_rates. The two subtrees situated at the two children of the root node are elaborated in Figure 4.9(b) and Figure 4.9(c). Part of the subtree rooted and shown in Figure 4.9(c) is shown fully in Figure 4.9(d). Such trees give us idea about the importance of the features or attributes and also help us to take decisions. The business rules could be generated from these trees by traversing from the root to a leaf. For example, two such rules are shown in the following.

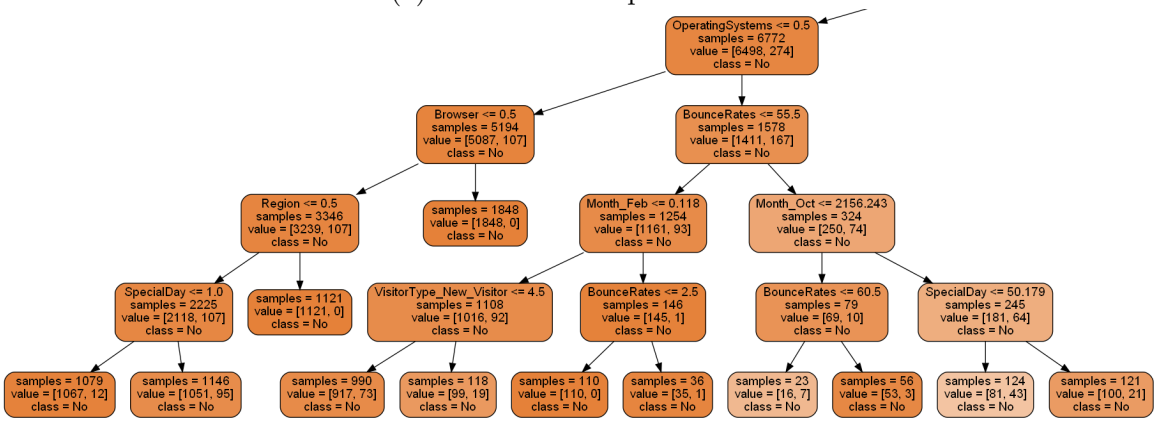
i. VisitorType_ReturningVisitor = True && Exit_Rates $> \tau$ && BounceRates $> \beta$ && Page_Values $> \alpha$ && administrative $> \gamma$ && Region $> \delta \Rightarrow$ **yes**

ii. VisitorType_ReturningVisitor = True && Exit_Rates $> \tau$ && BounceRates $> \beta$ && Page_Values $> \alpha$ && administrative $\leq \kappa$ && Page_Values $> \eta \Rightarrow$ **no**

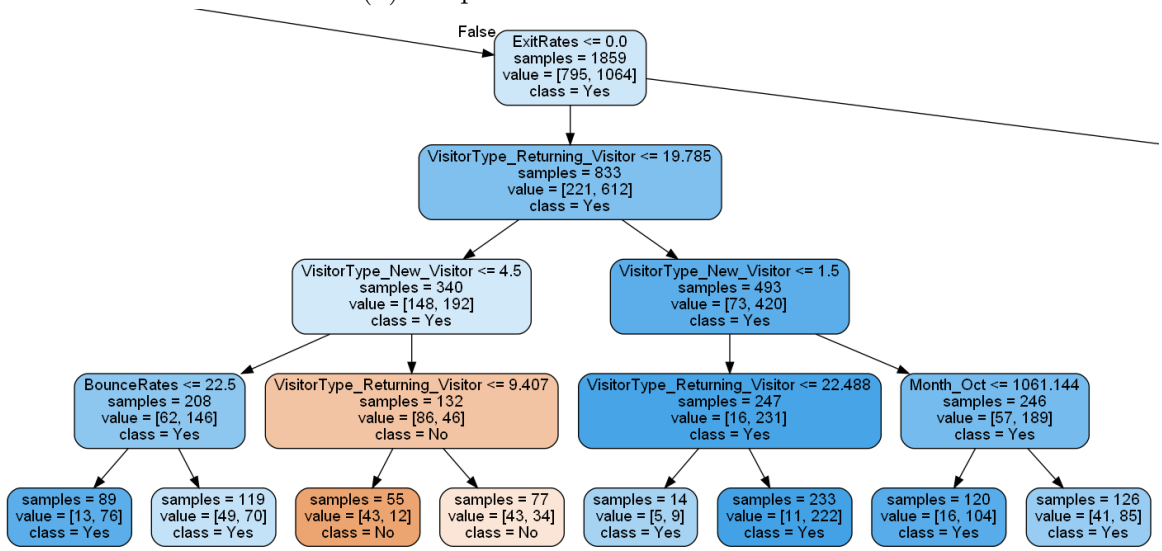
Note that we have masked the values for the attributes. Similar rules are possible to generate from trees of this kind. Also note that we have not done any association rule mining for this dataset, which also shows the itemsets with high confidence and may provide explainable rules similar to the decision tree classifier.



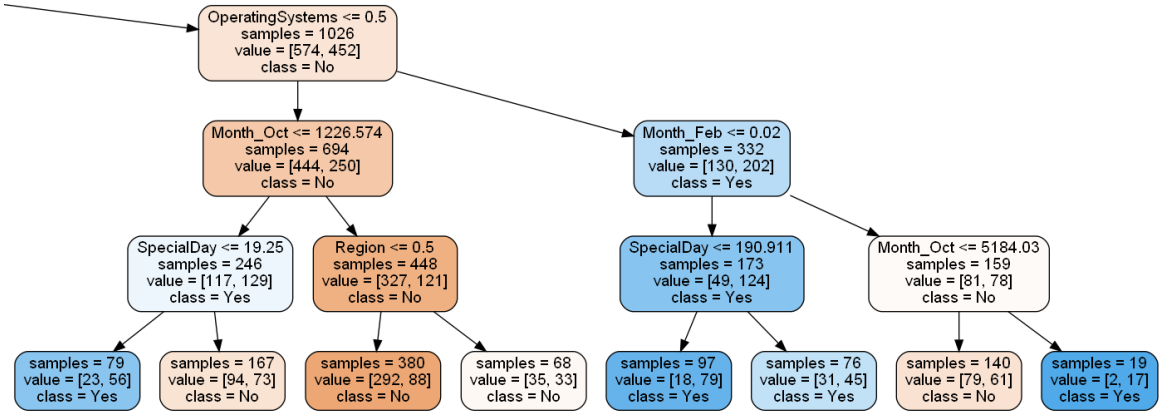
(a) Decision tree up to level 1



(b) left part of the tree elaborated



(c) right-middle part of the tree



(d) the right part of the tree

Figure 4.9: Best decision tree in parts.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Designing an efficient and standard machine learning algorithm is a challenging task owing to the imbalance of data present in the dataset. To replicate the overall data distribution present in a dataset, sampling techniques are widely used to deal with this problem. In this paper, we investigated three cluster-based under-sampling methods to tackle the data loss problem. First, we inject test data into training data for clustering. Then we select 25% close to the centroid and 25% from the boundary line. For the last method, we clean 50% majority data around minority data. According to the simulation results, our proposed method is better than the existing techniques with respect to auPR and auROC. Moreover, we also implemented F1-Score and MCC to find out how our methods are performing.

5.2 Future Work

In our future work, we are going to consider some other sampling methods to design the under-sampling technique. In addition, balancing data for oversampling using clustering will be another interesting future work.

Bibliography

[1] Pattaramon Vuttipittayamongkol and Eyad Elyan. Neighbourhood-based undersampling approach for handling imbalanced and overlapped data. *Information Sciences*, 509:47–70, 2020. ix, 5, 17, 23

[2] Michał Koziarski. Radial-based undersampling for imbalanced data classification. *Pattern Recognition*, 102:107262, 2020. 1

[3] Chih-Fong Tsai, Wei-Chao Lin, Ya-Han Hu, and Guan-Ting Yao. Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. *Information Sciences*, 477:47–54, 2019. 1

[4] Bartosz Krawczyk, Michał Koziarski, and Michał Woźniak. Radial-based oversampling for multiclass imbalanced data classification. *IEEE transactions on neural networks and learning systems*, 31(8):2818–2831, 2019. 1

[5] Shigang Liu, Guanjun Lin, Qing-Long Han, Sheng Wen, Jun Zhang, and Yang Xiang. Deepbalance: Deep-learning and fuzzy oversampling for vulnerability detection. *IEEE Transactions on Fuzzy Systems*, 28(7):1329–1343, 2019. 1

[6] Kyung Hye Kim and So Young Sohn. Hybrid neural network with cost-sensitive support vector machine for class-imbalanced multimodal data. *Neural Networks*, 130:176–184, 2020. 1

[7] Roozbeh Razavi-Far, Maryam Farajzadeh-Zanjani, Boyu Wang, Mehrdad Saif, and Shiladitya Chakrabarti. Imputation-based ensemble techniques for class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 2019. 2

[8] Farshid Rayhan, Sajid Ahmed, Asif Mahbub, Rafsan Jani, Swakkhar Shatabda, and Dewan Md Farid. Cusboost: Cluster-based under-sampling with boosting for imbalanced classification. In *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, pages 1–5. IEEE, 2017. 2, 17, 24

[9] C Okan Sakar, S Olcay Polat, Mete Katircioglu, and Yomi Kastro. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks. *Neural Computing and Applications*, 31(10):6893–6908, 2019. 2, 10, 13, 17, 24, 25, 28, 35, 36

[10] G Rekha and Amit Kumar Tyagi. Cluster-based under-sampling using farthest neighbour technique for imbalanced datasets. In *International Conference on Innovations in Bio-Inspired Computing and Applications*, pages 35–44. Springer, 2019. 4

[11] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409:17–26, 2017. 4, 17, 24

[12] Show-Jane Yen and Yue-Shi Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36(3):5718–5727, 2009. 4

[13] Minlong Peng, Qi Zhang, Xiaoyu Xing, Tao Gui, Xuanjing Huang, Yu-Gang Jiang, Keyu Ding, and Zhigang Chen. Trainable undersampling for class-imbalance learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4707–4714, 2019. 4

[14] YONGQING ZHANG, RONGZHAO LU, JI HUANG, and DONGRUI GAO. Evolutionary-based ensemble under-sampling for imbalanced data. In *2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing*, pages 212–216. IEEE, 2019. 4

[15] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *IEEE transactions on pattern analysis and machine intelligence*, 42(11):2781–2794, 2019. 5

[16] Wing WY Ng, Junjie Hu, Daniel S Yeung, Shaohua Yin, and Fabio Roli. Diversified sensitivity-based undersampling for imbalance classification problems. *IEEE transactions on cybernetics*, 45(11):2402–2412, 2014. 5

[17] Mar Mar Nwe and Khin Thidar Lynn. Knn-based overlapping samples filter approach for classification of imbalanced data. In *International Conference on Software Engineering Research, Management and Applications*, pages 55–73. Springer, 2019. 5

[18] Debashree Devi, Suyel Namasudra, and Seifedine Kadry. A boosting-aided adaptive cluster-based undersampling approach for treatment of class imbalance problem. *International Journal of Data Warehousing and Mining (IJDWM)*, 16(3):60–86, 2020. 5

[19] Jianjun Zhang, Ting Wang, Wing WY Ng, Shuai Zhang, and Chris D Nugent. Undersampling near decision boundary for imbalance problems. In *2019 International conference on machine learning and cybernetics (ICMLC)*, pages 1–8. IEEE, 2019. 5, 17, 24

[20] Tuong Le, Hoang Le Son, Minh Thanh Vo, Mi Young Lee, Sung Wook Baik, et al. A cluster-based boosting algorithm for bankruptcy prediction in a highly imbalanced dataset. *Symmetry*, 10(7):250, 2018. 5

[21] Jia Song, Xianglin Huang, Sijun Qin, and Qing Song. A bi-directional sampling based on k-means method for imbalance text classification. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–5. IEEE, 2016. 5

[22] Anqi Shangguan, Guo Xie, Lingxia Mu, Rong Fei, and Xinhong Hei. Abnormal samples oversampling for anomaly detection based on uniform scale strategy and closed area. *IEEE Transactions on Knowledge and Data Engineering*, 2021. 6

[23] Jinfu Ren, Yang Liu, and Jiming Liu. Ewgan: Entropy-based wasserstein gan for imbalanced learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10011–10012, 2019. 6

[24] Zhe Wang, Chenjie Cao, and Yujin Zhu. Entropy and confidence-based undersampling boosting random forests for imbalanced problems. *IEEE transactions on neural networks and learning systems*, 31(12):5178–5191, 2020. 6

[25] Yuxin Peng. Adaptive sampling with optimal cost for class-imbalance learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. 6

[26] Zhaozhao Xu, Derong Shen, Tiezheng Nie, Yue Kou, Nan Yin, and Xi Han. A cluster-based oversampling algorithm combining smote and k-means for imbalanced medical data. *Information Sciences*, 572:574–589, 2021. 6

[27] Qian Shi and Hui Zhang. Fault diagnosis of an autonomous vehicle with an improved svm algorithm subject to unbalanced datasets. *IEEE Transactions on Industrial Electronics*, 68(7):6248–6256, 2020. 6

[28] Kwabena Ebo Bennin, Jacky Keung, Passakorn Phannachitta, Akito Monden, and Solomon Mensah. Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Transactions on Software Engineering*, 44(6):534–550, 2017. 6

[29] Xinmin Tao, Qing Li, Wenjie Guo, Chao Ren, Qing He, Rui Liu, and JunRong Zou. Adaptive weighted over-sampling for imbalanced datasets based on density peaks clustering with heuristic filtering. *Information Sciences*, 519:43–73, 2020. 6

[30] Zhi Chen, Jiang Duan, Li Kang, and Guoping Qiu. A hybrid data-level ensemble to enable learning from highly imbalanced dataset. *Information Sciences*, 554:157–176, 2021. 7

[31] Yuwei Zhang, Guanjun Liu, Wenjing Luan, Chungang Yan, and Changjun Jiang. An approach to class imbalance problem based on stacking and inverse random under sampling methods. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6. IEEE, 2018. 7

[32] Lingkai Yang, Yinan Guo, and Jian Cheng. Manifold distance-based over-sampling technique for class imbalance learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10071–10072, 2019. 7

[33] Jinyan Li, Yaoyang Wu, Simon Fong, Antonio J Tallón-Ballesteros, Xin-she Yang, Sabah Mohammed, and Feng Wu. A binary pso-based ensemble under-sampling model for rebalancing imbalanced training data. *The Journal of Supercomputing*, pages 1–36, 2021. 7

[34] Sajid Ahmed, Farshid Rayhan, Asif Mahbub, Md Rafsan Jani, Swakkhar Shatabda, and Dewan Md Farid. Liuboost: locality informed under-boosting for imbalanced data classification. In *Emerging Technologies in Data Mining and Information Security*, pages 133–144. Springer, 2019. 7

[35] Xinmin Tao, Qing Li, Wenjie Guo, Chao Ren, Chenxi Li, Rui Liu, and Junrong Zou. Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. *Information Sciences*, 487:31–56, 2019. 7

[36] JooHwa Lee and KeeHyun Park. Gan-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, 25(1):121–128, 2021. 8

[37] Funa Zhou, Shuai Yang, Hamido Fujita, Danmin Chen, and Chenglin Wen. Deep learning fault diagnosis method based on global optimization gan for unbalanced data. *Knowledge-Based Systems*, 187:104837, 2020. 8

[38] Ramazan Esmeli, Mohamed Bader-El-Den, and Hassana Abdullahi. Towards early purchase intention prediction in online session based retailing systems. *Electronic Markets*, pages 1–19, 2020. 8

[39] Houda Zarrad and Mohsen Debabi. Online purchasing intention: Factors and effects. *International Business and Management*, 4(1):37–47, 2012. 8

[40] Sorim Chung, Thomas Kramer, and Elaine M Wong. Do touch interface users feel more engaged? the impact of input device type on online shoppers' engagement, affect, and purchase decisions. *Psychology & Marketing*, 35(11):795–806, 2018. 8

[41] Mazzini Muda, Rohani Mohd, and Salwana Hassan. Online purchase behavior of generation y in malaysia. *Procedia Economics and Finance*, 37:292–298, 2016. 8

[42] Sujoy Bag, Manoj Kumar Tiwari, and Felix TS Chan. Predicting the consumer's purchase intention of durable goods: An attribute-level analysis. *Journal of Business Research*, 94:408–419, 2019. 8

[43] Monica Law, Ron Chi-Wai Kwok, and Mark Ng. An extended online purchase intention model for middle-aged online users. *Electronic Commerce Research and Applications*, 20:132–146, 2016. 9

[44] Grażyna Suchacka and Grzegorz Chodak. Using association rules to assess purchase probability in online stores. *Information Systems and e-Business Management*, 15(3):751–780, 2017. 9

[45] Grazyna Suchacka, Magdalena Skolimowska-Kulig, and Aneta Potempa. Classification of e-customer sessions based on support vector machine. *ECMS*, 15:594–600, 2015. 9

[46] Grażyna Suchacka, Magdalena Skolimowska-Kulig, and Aneta Potempa. A k-nearest neighbors method for classifying user sessions in e-commerce scenario. *Journal of Telecommunications and Information Technology*, 2015. 9

[47] Paulo Rita, Tiago Oliveira, and Almira Farisa. The impact of e-service quality and customer satisfaction on customer behavior in online shopping. *Heliyon*, 5(10):e02690, 2019. 9

[48] Amal Dabbous and Karine Aoun Barakat. Bridging the online offline gap: Assessing the impact of brands' social network content quality on brand awareness and purchase intention. *Journal of Retailing and Consumer Services*, 53:101966, 2020. 9

[49] José Martins, Catarina Costa, Tiago Oliveira, Ramiro Gonçalves, and Frederico Branco. How smartphone advertising influences consumers' purchase intention. *Journal of Business Research*, 94:378–387, 2019. 9

[50] Liang Xiao, Feipeng Guo, Fumao Yu, and Shengnan Liu. The effects of online shopping context cues on consumers' purchase intention for cross-border e-commerce sustainability. *Sustainability*, 11(10):2777, 2019. 9

[51] Karim Baati and Mouad Mohsil. Real-time prediction of online shoppers' purchasing intention using random forest. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 43–51. Springer, 2020. 10, 35, 36

[52] Peiyi Song and Yutong Liu. An xgboost algorithm for predicting purchasing behaviour on e-commerce platforms. *Tehnički vjesnik*, 27(5):1467–1471, 2020. 35, 36

[53] Faqih Hamami and Ahmad Muzakki. Machine learning pipeline for online shopper intention classification. In *AIP Conference Proceedings*, volume 2329:1, page 050014. AIP Publishing LLC, 2021.

[54] Rizal Dwi Prayogo and Siti Amatullah Karimah. Feature selection and adaptive synthetic sampling approach for optimizing online shopper purchase intent prediction. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–5. IEEE, 2021.

[55] Md Rayhan Kabir, Faisal Bin Ashraf, and Rasif Ajwad. Analysis of different predicting model for online shoppers' purchase intention from empirical data. In *2019 22nd International Conference on Computer and Information Technology (ICCIT)*, pages 1–6. IEEE, 2019. 10

[56] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 27

[57] Tawfiq Hasanin and Taghi Khoshgoftaar. The effects of random undersampling with simulated class imbalance for big data. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 70–79. IEEE, 2018. 27

[58] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. 27

[59] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 27

[60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 27

[61] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. 27

[62] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 28

[63] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001. 28

[64] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 29