9-30-2019

Click Fraud Detection in Online and In-app Advertisements: A Learning Based Approach

Thejas Gubbi Sadashiva
Florida International University, tgs001@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

CLICK FRAUD DETECTION IN ONLINE AND IN-APP ADVERTISEMENTS: A LEARNING BASED APPROACH

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Thejas Gubbi Sadashiva

2019

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Thejas Gubbi Sadashiva, and entitled Click Fraud Detection in Online and In-app Advertisements: A Learning Based Approach, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Niki Pissinou

Jean Andrian

N.R. Sunitha

Deng Pan

Leonardo Bobadilla

S.S. Iyengar, Major Professor

Date of Defense: September 30, 2019

The dissertation of Thejas Gubbi Sadashiva is approved.

Dean John L. Volakis
College of Engineering and Computing

Andres G. Gil
Vice President for Research and Economic Development and
Dean of the University Graduate School

Florida International University, 2019

DEDICATION

To my parents, my wife, my younger brother, my parents-in law and family.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Professor Sundaraja Sitharama Iyengar for his kind supervision during my Ph.D. Without his encouragement and guidance, I would not have been able to complete my Ph.D., and this dissertation would not have existed. I would also like to thank Professor N.R. Sunitha for mentoring, supporting, and helping me in all the stages. I would also like to thank Professor Niki Pissinou, Professor Jean Andrian, Professor N.R. Sunitha, Professor Deng Pan, and Professor Leonardo Bobadilla for being my doctoral committee. They have provided me many valuable questions and useful suggestions for my dissertation. I would also like to thank all of my coauthors and lab-mates with whom I have collaborated. It was my great honor to work with them. I am also thankful to our school's administrative and advising staff Professor Jason Liu, Professor Shu-Ching Chen, Professor Tao Li, Professor Nagarajan Prabakar, Jill Weiss, Olga Carbonell, Vanessa Cornwall, Carlos Cabrera, Rebeca Arocha, Ariana Taglioretti, Col. Jerry Miller, Donaley Dorsett, Jose Oquendo, Steven Luis and entire IT Support team and Luis Rivera as well as Partner institutions from India Siddaganga Institute of Technology and R.V. College of Engineering. Parts of this dissertation were originally published and submitted in ACMSE 2019, IEEE SoutheastCon 2019, IEEE CSITSS 2019, IEEE Access 2019, IEEE TKDE 2019, and ACM TIST. I am also thankful to all the reviewers, editors, and conference organizers who provided valuable feedback and facilitated in the publication of my research. Finally, I would like to thank my parents G. C. Sadashiva and Vyjayanthi Mala, my wife Shruthi B, my younger brother Benak Shreyas G.S., and my parents-in law and family for their motivation and support.

ABSTRACT OF THE DISSERTATION

CLICK FRAUD DETECTION IN ONLINE AND IN-APP ADVERTISEMENTS: A LEARNING BASED APPROACH

by

Thejas Gubbi Sadashiva

Florida International University, 2019

Miami, Florida

Professor S.S. Iyengar, Major Professor

Click Fraud is the fraudulent act of clicking on pay-per-click advertisements to increase a site's revenue, to drain revenue from the advertiser, or to inflate the popularity of content on social media platforms. In-app advertisements on mobile platforms are among the most common targets for click fraud, which makes companies hesitant to advertise their products. Fraudulent clicks are supposed to be caught by ad providers as part of their service to advertisers, which is commonly done using machine learning methods. However: (1) there is a lack of research in current literature addressing and evaluating the different techniques of click fraud detection and prevention, (2) threat models composed of active learning systems (smart attackers) can mislead the training process of the fraud detection model by polluting the training data, (3) current deep learning models have significant computational overhead, (4) training data is often in an imbalanced state, and balancing it still results in noisy data that can train the classifier incorrectly, and (5) datasets with high dimensionality cause increased computational overhead and decreased classifier correctness – while existing feature selection techniques address this issue, they have their own performance limitations. By extending the state-of-the-art techniques in the field of machine learning, this dissertation provides the following solutions: (i) To address (1) and (2), we propose a hybrid deep-learning-based model which con-

sists of an artificial neural network, auto-encoder and semi-supervised generative adversarial network. (ii) As a solution for (3), we present Cascaded Forest and Extreme Gradient Boosting with less hyperparameter tuning. (iii) To overcome (4), we propose a row-wise data reduction method, KSMOTE, which filters out noisy data samples both in the raw data and the synthetically generated samples. (iv) For (5), we propose different column-reduction methods such as multi-time-scale Time Series analysis for fraud forecasting, using binary labeled imbalanced datasets and hybrid filter-wrapper feature selection approaches.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

Due to the growth in web technologies and media, advertising companies have shifted focus from conventional newspaper and television advertisements to browser and in-app advertisements. Internet giants such as Google, Yahoo, and Facebook are fundamentally advertising networks that act as brokers between advertisers and content publishers, with ad services constituting the largest portion of their yearly revenues. These ad networks are provided with advertisements and agree on a fee for every user action, such as each click. According to [RSF17], the number of smartphone users worldwide will reach an estimated 2.87 billion by 2020 – a jump of 1.3 billion users in a span of 6 years. These developments have caused the advertising industry to shift its focus to digitally active platforms like the Android and iOS app domains, prompting them to create a business model that leverages the high footfall of users to promote advertisements [NW16]. This business model includes various parameters to expand the opportunities of success for the advertising party and to the payment for the advertisement publisher. One such parameter is "click and impression", which is used to calculate the number of times the ad is displayed to the user, and the number of times doing so resulted in the user clicking the ad [HIRR18]; this is regarded as a direct measure of the advertisement's success, which correlates the ad publisher's efforts and positively contributes to the advertising party's intentions.

There are many vulnerabilities related to how "click and impression" is calculated, and click fraud is one of the most challenging ones to address. Click Fraud refers to the practice of generating random clicks in order to extract illegitimate revenue from the advertising party. In the case of a pay-per-click advertisement, the expectation is that each click is from a genuine user; but in 2017 about 1 in 5

clicks were fraudulent – on smartphones this number doubled over the course of 4 months [ppc19]. A simple scenario: a publisher receives a link from the advertiser in order to host it on their web pages, and a pay-per-click contract is agreed upon by each party. In order to inflate the number of clicks for the ad receives, a script that generates automated random clicks is created, resulting in a misleading report to be presented to the advertising party. Some of the other important parameters that decide the remuneration to the publishers include conversion rate, cost-per-click, cost per mile, average ads position, and click through rate.

One might assume that the beneficiaries of click fraud are the publishers who are taking the money from the advertiser for each pay-per-click transaction, but it could also be the third party websites that host these ads, and who can in turn demand extra money from the publishers to account for the same. It is imperative to distinguish between clicks produced by genuine users and fraudulent clicks produced by parties that illicitly benefit from it [TKC+19]. In this work, we look at the three most common offenders: Competitors, Webmasters and Fraud rings:

- Competitors are responsible for the vast majority of click fraud. Click fraud rewards them with a competitive advantage by wasting the competitor's pay-per-click budget.
- Malicious webmasters gain unjustified income from displaying ads on their website when they commit click fraud. Instead of building and developing their website to entice traffic, they may click on these ads to increase revenue.
- Fraud rings are groups of criminals that specifically target ad networks to exploit the maximum amount of money possible in a short duration. A Russian criminal fraud group dubbed 'The Methbot Operation' unlawfully earned $3 - $5 million per day from fraudulent activity [cli19].

There are many ways click fraud can be committed, some of which we describe below [fru19]:

- Crowdsourcing is a method used by publishers to increase clicks on the ads. They use the visitors on the website to click on their ads intentionally or unintentionally, such as by putting up fake icons (like a play button for a video) or links that specify a discount on prices or in-game items that redirect the visitors to the advertiser's site.
- Click Farm is a method to commit click fraud by persuading people to click on ads all day in exchange for money. In comparison with automated scripts, using actual human beings proves to be more advantageous since the persuaders can instruct the ones who perform the clicks on how to click on the advertisements more naturally.
- Hit Inflation Attack is another method of click fraud wherein legitimate users are redirected to a website by going to the advertisement first and then to the page they intended to browse. The visitor does not see the advertisement but notices a slightly longer load time than usual.
- Botnets are malware that infects a large number of computers. The malware controls the computers as a group, instructing them to visit different sites and click on advertisements without the knowledge of the owners of those computers. They generate many seemingly unrelated clicks from multiple IP addresses.

A more advanced form of click fraud attacks is Adversarial Machine Learning which involves smartly subduing the intelligence of machine learning algorithms. These attacks involve modifying the data by including malicious input to cause a malfunction in the standard machine learning model [BFR14]. The two main types

of Machine Learning attacks are Evasion attacks and Poisoning attacks. In an Evasion attack, the attackers modify inputs at the testing time, in order to avoid getting detected, and thus classifying illegitimate data as acceptable in the later stages. In a Poisoning Attack, the training data is corrupted by injecting malicious data in order to put the whole learning process at stake [BFR14, BCN+14, BNJT10, NRH+10, BNL11]. In addition to ad networks, Online Social Network (OSN) platforms such as Instagram, Facebook, Twitter, etc. can also be subject to the Click fraud problem primarily based on fake like clicks on posts with the intention of achieving popularity [JL08, ins18]. There are numerous means of creating such fake likes such as (a) influencing friends and relatives to make the fake likes, (b) utilizing botnets to artificially enhance popularity, (c) having paid services to purchase fake likes, (d) having click spammers provide free fake likes or (e) entering and becoming a part of the collusion network.

1.1 Challenges

Although the fraud clicks are supposed to be caught by the ad providers as part of their service to the advertisers, there are some challenges as follows:

1. *There is a lack of research in the current literature for addressing and evaluating different techniques of click fraud detection and prevention.* Additionally, the attack model can itself be an active learning system (smart attacker) to actively mislead the training process of fraud detection model via polluting the training data.
2. *Currently, deep learning models are employed for click fraud detection. However, due to certain obstacles, significant computational overhead is incurred in search for the best model.*

The deep learning models [TKC+19, HAH+18] employed in the existing click fraud approaches need extensive hyper-parameter tuning if they are to provide appreciable results. However, the real-time advertisement datasets are usually huge which in-turn, result in an increase in the feature space dimension. Hyper-parameter tuning is a tedious repetitive process whenever the model has to be retrained periodically. Since the number of parameters to be tuned are many in the case of neural networks, a large amount of time and resources is essential. As an existing example, Google AdWords [fru19], an advertising network, has a three-tier system to detect click fraud. This three-tier system is composed of automated filters to detect suspicious clicks in real time followed by an offline analysis and finally an in-depth investigation. From this example, it is evident that the click fraud detection process is not entirely automated.

3. *Often, the training data could be in an imbalanced state (row-wise). However, even after balancing the dataset, noisy data may exist in both the states which may train the classifier incorrectly.*

   Most of the real-time datasets associated with medicine [TDK10], text classification [LLS09], intrusion detection [KTPA12], and click fraud detection [FP97] are imbalanced. A binary dataset having 95% positive samples may obtain an extremely high classification accuracy. However, this accuracy may be incorrect due to over-fitting. In the recent past, several studies have proposed various solutions to address this issue [HG08]. Among them, there are several re-sampling techniques available [BSL11, CBHK02, KM+97, SW08] to balance the imbalanced dataset such as Synthetic Minority Over-sampling Technique (SMOTE) [CBHK02] which is one of the popular re-sampling techniques. 85 variants of SMOTE have been proposed in the recent past with the

intention of improving the quality of the training data which in-turn enhances the performance of the classifiers. Additionally, studies show that noise and borderline samples could also affect the performance of the classifier which are generated by applying the re-sampling techniques (SMOTE) and may also exist in the original data [GSM07, Jap03, NSW10]. Applying SMOTE on imbalanced datasets gives better results, but it generates synthetic samples, resulting in a notable increase in the size of the data. Presently, the data collected in real-time is extremely large (BIG DATA) [ZE+11]. SMOTE can be considerably improved by performing certain modifications (Borderline-SMOTE V1,V2 [HWM05], Safe-level SMOTE [BSL09]) or by adding some extensions (SMOTE-IPF [SLSH15], ENN or TL [BPM04]).

4. *Also, datasets with high dimensionality cause an increase in the computational overhead and a decrease in the classifier correctness. Although existing Feature Selection techniques address this issue to provide an optimal set of features, they have their own performance limitations.*

   Determining the important classifier features is a key phase in the classification process and as the number of features in the ever increasing number of data samples is proving to be a significant problem. Therefore a reduction in dimensionality by having a small set of features is necessary to achieve higher accuracy, shorter computation time and to reduce overfitting. There are 2 ways in which dimensionality can be reduced i.e., Feature Extraction (FE) and Feature Selection (FS). In FE, a liner or non-linear combination of features is employed to achieve lesser dimensionality where the actual data is transformed and thereby, there is scope for distortion. In FS, a certain criteria is taken into account for the selection of the feature's subset since the dataset could contain a number of irrelevant attributes with respect to the class which

could result in a fall in accuracy of the induced classifier [JKP94]. Identifying and removing such attributes reduces dimensionality which in-turn results in a drop in computation time while improving accuracy. In [Kus99], they state that the overabundance of features rendered the nearest neighbor approach on Internet Advertisement dataset. FS has many applications in various fields like image processing, natural language processing, bioinformatics, data mining, and machine learning (ML) [HBK14]. The selection method is divided into two standard categories based on their working modules, classifier independent 'filter' technique, and classifier dependent 'wrapper' and 'embedded' technique. The filter method carries out feature selection on the basis of certain metrics such as distance, correlation, consistency measure, and mutual information (MI) while being independent of the classifier. This method increases the computational efficiency and reduces data dimensionality [SIL07]. A key weakness however, is the uncertainty that exists regarding the relationship between the feature attributes and target class. The classifier dependent systems rely upon the classifier for the selection process, the output of which is used to obtain the subset of features in the Wrapper method, making it biased to a classifier. Additionally, the wrapper method is vulnerable to overfitting, usually when the dataset is of a smaller size less [BPZL12]. The embedded method makes use of the classifier in the training phase and selects the optimal features like a learning procedure. The embedded method in comparison with the wrapper method, proves to be less vulnerable to overfitting and the computation is much faster [LPd$^+$12]. These existent FS algorithms are useful but do not always prove to be extremely helpful when we use certain machine learning algorithms like Random Forests.

1.2 Objective of this Dissertation

To address the above said challenges, in this dissertation, we propose ML based solutions to improve the performance of the existing click fraud detection methods. Following serve as the objectives of this dissertation:

- Application of learning based approach as a hybrid solution for click fraud detection.
- Proposing an alternative to Multi-layered Neural network by enhancing the hybrid cascaded forest model with lesser hyper-parameter tuning.
- To improve the existing state-of-the-art row and column based data reduction techniques.

1.3 Contributions

This dissertation presents solutions to the click fraud problem by extending the state-of-the-art techniques in the area of supervised ML.

1.3.1 Deep Learning-based Model to Fight Against Ad Click Fraud

To address challenge (1), we propose a way to handle the imbalanced dataset [Kag18] based on our pre-analysis on attributes like ip and app id. We simulated learning models like logistic regression, random forest, naive bayes, and support vector machine from which it concludes that logistic regression and random forest competitively perform well. We developed a hybrid model consisting of Semi-supervised Generative Adversarial Network (GAN), Auto Encoder and Neural Network to solve

the problem of fake clicks in adversarial environment. We used the semi-supervised GAN to create adversarial attacks in order to improve the accuracy of the Neural Network. To the best of our knowledge, we are the first to use $Cos(\theta)$, as a supervised loss in a semi-supervised GAN and the hybrid model performs better than other models.

1.3.2 CFXGB: An Optimized and Effective Learning Approach for Click Fraud Detection

As a solution for challenge (2), we extend the work done by [ZF17] Zhou et al. by proposing a two-phase model consisting of feature transformation by Cascaded Forests and classification by Extreme Gradient Boosting. Our approach requires only minimal hyper-parameter tuning i.e., the number of parameters that vary are few. Use of forests in the form of layers is more intuitive in understanding the model's learning method as compared to the use of neural nets. Accuracy, Area-Under-Curve (AUC), Recall, F1-score and Precision are used to evaluate the superiority of the proposed model against recent works. We have evaluated our model on five benchmark datasets. Three of them are click fraud datasets and two of them are intrusion detection datasets.

1.3.3 KSMOTE: An Extension of Synthetic Minority Oversampling Technique for Imbalanced Datasets

With respect to challenge (3), prior works show that filters have not been commonly used in combination with SMOTE. Hence, we propose an extension called KSMOTE which employs the Kalman Filter[BW$^+$01] to remove noisy data samples. The use

of the Kalman filter as a data-reduction method improves the efficacy of the classifier and reduces the processing overhead. We use three Click-Fraud datasets, an Intrusion Detection dataset and a few UCI[BM98] datasets that are considered by previous researchers, to evaluate our work. Furthermore, we make comparisons with several, existing variants of SMOTE. Metrics such as Recall, Accuracy, F1-Score and Precision have been used to provide comparisons. AUC is another very good metric to verify overfitting caused by SMOTE.[Bra97].

1.3.4 A Multi-time-scale Time Series Analysis for Click Fraud Forecasting using Binary Labeled Imbalanced Dataset

To address challenge (4), We present a generalized model for modeling temporal click fraud data. The proposed model consists of four stages: Pre-analysis and preprocessing, Probabilistic or learning-based data smoothing, fraudulent pattern identification, and time-series model fitting. The objectives of the proposed work are: firstly, model multi-time-scale time series data on Auto-Regression (AR)/Moving Average (MA) by relying only on time and the label, without the need of too many attributes. Secondly, to model different time scales separately on AR and MA models. Then, we Evaluate the models by tuning forecasting errors and also with minimizing Akaike information criterion (AIC) and Bayesian information criterion (BIC) to obtain a best fit model for all time scale data. Choosing AIC or BIC as a criterion mainly depends on our requirement where AIC is chosen to select more efficient models in terms of accuracy and small forecasting errors whereas on other hand, BIC is chosen if we want to select a model that fits for different training data

without becoming progressively worse in terms of forecasting performance. We extend Box-Jenkins [NB13, BJRL15] and Boroojeni et al. methodology [BAB+17] for modelling of ad clicking activity forecasting to show the future possible fraudulent behavior. In our proposed approach, we model multi-time-scale seasonality to forecast the fraudulent behavior in terms of minutes and hours interval. Our proposed approach can be considered as an extension of Seasonal Auto-regressive Integrated Moving (SARIMA) model [Tay10]. AIC, BIC, and residual errors are used to fine tune the model.

1.3.5 Learning-Based Model to Fight against Fake Like Clicks on Instagram Posts

To address (1), we carried out a series of steps: Collection of data based on different parameters and attacks, identification of important features to train and test the models, pre-analysis to show the relationship between the follower and following participation in valid and invalid like clicks, we then developed an automated learning model to detect fake liking behavior on the Instagram post, and finally, we also examined autoencoder loss function to differentiate bots and human clicks.

1.3.6 MRFI and ARFI: Hybrids of Filter and Wrapper Feature Selection Approaches

With respect to challenge (4), we gathered that the existent FS algorithms are useful but do not always prove to be extremely helpful when we use certain machine learning algorithms like Random Forests. Therefore, in our work, we propose two new FS techniques, Metric Ranked Feature Inclusion (MRFI) and Accuracy Ranked

Feature Inclusion (ARFI), which can be used effectively across a variety of learning models. Our proposed algorithms are hybrids of the filter and wrapper methods and follow a two phase process. The first phase takes inspiration from the filter technique, and we assign scores for the features to rank them. For the first proposed algorithm, the score is assigned to each feature after clustering the data with the help of that feature alone. We use K-Means to cluster the data and then apply a clustering metric by the name of V-Measure. ARFI involves scoring each feature based on the accuracy of a classifier (Random Forest), which is evaluated with only that particular feature. Ranking the features using these techniques truly brings out their importance to the label. The next stage of the algorithm, i.e., the feature subset selection phase, avoids redundancy. Here, the variables are iteratively added to the optimal subset one by one, and each time, the learning model is evaluated. The recently added feature is retained or dropped depending on the calculated accuracy. The second stage behaves as the wrapper part. Both MRFI and ARFI share the same feature subset selection technique. We validated our models with the various datasets and compared our results with another standard FS technique, Recursive Feature Elimination (RFE). Our models outperformed RFE with every dataset and gave us positive results.

1.3.7 Mini-Batch Normalized Mutual Information: A Hybrid Feature Selection Method

As a solution to challenge (4), we propose a combination of the filter and wrapper methods, which has the advantage of both the techniques. It is fast and general like the filter method. At the same time, it accounts to learning algorithm obtaining the best set of features without the need for the user to input the feature number

unlike most of other established algorithms such as RFE. We also take advantage of mini-batch Kmeans which perform better as compared to K-Means in terms of computation time for lager datasets. Here, we cluster the data using mini-batch Kmeans clustering and rank them using normalized mutual information (NMI), a measure to calculate the relevance and the redundancy between the candidates' attribute and the class. We apply a greedy search method by using Random Forest to get the optimal set of features. However, our method is flexible in terms of the learning algorithm that can be incorporated.

CHAPTER 2

DEEP LEARNING-BASED MODEL TO FIGHT AGAINST AD CLICK FRAUD

Click fraud is a fast-growing cyber-criminal activity with the aim of deceptively clicking on the advertisements to make the profit to the publisher or cause loss to the advertiser. Due to the popularity of smartphones since the last decade, most of the modern-day advertisement businesses have been shifting their focus toward mobile platforms. Nowadays, in-app advertisement on mobile platforms is the most targeted victim of click fraud. Malicious entities launch attacks by clicking ads to artificially increase the click rates of specific ads without the intention of using them for legitimate purposes. The fraud clicks are supposed to be caught by the ad providers as part of their service to the advertisers; however, there is a lack of research in the current literature for addressing and evaluating different techniques of click fraud detection and prevention. Another challenge toward click fraud detection is that the attack model can itself be an active learning system (smart attacker) with the aim of actively misleading the training process of fraud detection model via polluting the training data.

In this chapter, we propose a deep-learning based model to address the challenges as mentioned above. The model is a hybrid of artificial neural network (ANN), auto-encoder and semi-supervised GAN. Our proposed approach triumphs excellent accuracy than other models.

2.1 Introduction

With the invent of smart-phones, people have started using it addictively in their daily life. On the other hands, as the digital advertising industry business model rely on the people usage of computing devices, even they have shifted their concentration on the mobile platform with the transformation of promoting ads through in-app advertisement [NW16]. In their business model, "click and impression" plays a significant role as it is one of the parameters to calculate the payment for the publisher by the ad network. Whenever a person clicks an ad, then it will be recorded by the ad network as a count that the number of times the ad was clicked. In the same manner, whenever the ad loaded in the user devices, then it will be recorded by the ad network as a count that the number of times the ad was loaded [HIRR18]. Most common parameters that are considered by the ad network to calculate the payment for the publisher are: pay per click, cost per click, click through rate, average ads position, cost per mile, conversion rate and cost per conversion.

However, in the business revenue model there exists a security problem, i.e., click fraud. Of course, the typical way of interacting with the computing device is through clicks. For example, we click an app to open it, we click a hyperlink to redirect to the destination page, or we click on like icon on social network posts or YouTube videos to express our emotion, we click on exciting ads to explore them, etc. [TSC+19]. In all the above-said example the click operation plays a vital role in the expectation of genuine purpose. Because too many clicks on a social network post say that the post is becoming prevalent and too many clicks on the ad, show the demand on the product that is promoted through ads. It is not apparent to believe that all clicks are valid (legitimate) or fraudulent clicks. There is no protocol defined to authenticate these clicks based on trust [TPIS18]. Click fraud is nothing but the

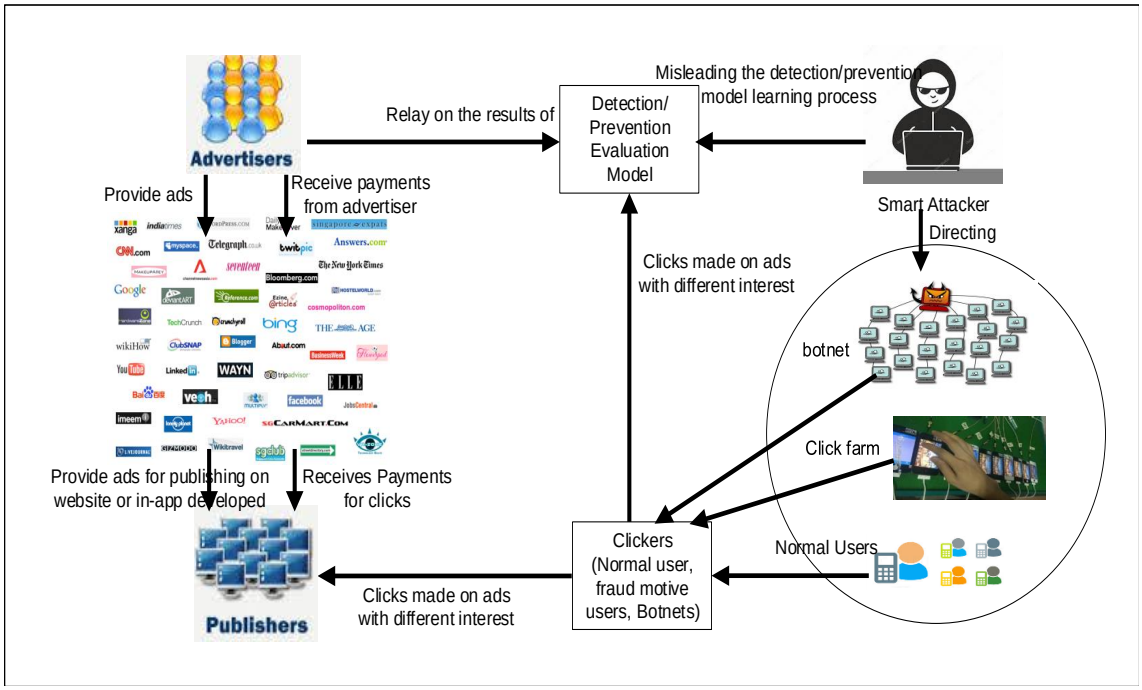


Figure 2.1: Architecture of Proposed System with Ad Network, Advertiser, Publisher, Clickers

action that happens by fraudulent clicks where the click made on an ad without the intention of using them for legitimate purposes.

Problem Figure 2.1 depicts the proposed architecture of click fraud. Here, an advertiser is the one who wants to make a profit from his/her business or product by mean of promoting it to the world. These promotions are done in the form of advertisement on websites or in-app platforms. Hence, advertisers pay ad networks to promote their ads on the websites or in-app. The publisher is the one who develops the website or mobile application and deploys ads on them. Ad Network provides a platform for advertiser and publisher. They are responsible for receiving an advertisement request and required payment form the advertiser for promotion. They are also responsible for approving the ad publishing request from the publisher, allowing them to deploy ads on their website or in-app. With these, the ad network is also responsible for evaluating the clicks made on ads and making payment to all

valid clicks. Detection/Prevention models or the learning-based models or manual analysis are used by ad networks to evaluate the clicks as either valid or invalid. Here clicks are made by Clickers who can be the regular users, i.e., clicks with genuine interest, or fraud motive users like click farms, botnets or smart attacker. The smart attacker is the one who misleads the learning models to perform wrongly via polluting the training model and directing the other real-time attacks [BFR14, BCN+14, BNJT10, NRH+10, BNL11].

2.1.1 Summary of Contribution

We propose a way to handle the imbalanced dataset based on our pre-analysis on attributes like ip and app id. We simulated learning models like logistic regression, random forest, naive bayes, and support vector machine from which it concludes that logistic regression and random forest competitively performs well. We developed a hybrid model consisting of Semi-supervised GAN, Auto Encoder and Neural Network to solve the problem of fake clicks in adversarial environment. We used the semi-supervised GAN to create adversarial attacks in order to improve the accuracy of the Neural Network. To the best of our knowledge, we are the first to use $Cos(\theta)$, as a supervised loss in a semi-supervised GAN and the hybrid model performs better than other models.

2.1.2 Organization of the Chapter

in Section 2.2 we review the related work on click fraud in ad networks, in Sect. 2.3 we describe the dataset characteristics, challenges and experimental setup, in Section 2.4 we discuss and present the preliminary results obtained on Logistic Regression, Support Vector Machine, Random Forest, and Naive Bayes, in Section

2.5 we propose our deep learning approach and provide details about the hybrid model, building blocks of the hybrid model, attacker model GAN and discussions on the presented results, and in Section 8.6 we conclude our discussion.

2.2 Related Work

Research shows that the more asymmetric in quality the ad networks are, the more asymmetric their equilibrium prices will be[DM14]. A study showed that one of the largest click fraud botnets, called ZeroAccess, induces advertising losses on the order of $100,000 per day [LNGL14].

Non-statistical Models: In [Had10], they take the average clicks on bluff ads as a way of discrimination; In [CdQC12], they use CAPTCHAs to make sure that the user is real. In [FLDH+16a], they used Social Network Analysis to find top three ad networks that were being used to spread the fraud click malware. In [APS+11], they use Splay trees to store the IPs via which fraud clicks occur based on a burst. In [LLCX15], they first find the eigenvalues of displayed ad images, if the ad is shown it is attested, based on if the eigenvalues of the image match those stored in their server, the user is certified as honest, if not then it is analysed. Researchers build on several published theoretical results to devise the Similarity-Seeker algorithm that discovers coalitions made by pairs of fraudsters [MAA07].

Statistical Models: These use sophisticated statistical models to find out which IP addresses are behind fraud attacks [KKL+14] or analyse periodic activity of DNS to find out nefarious activities. In [CSC14], they take certain features of the Android app and use machine learning to flag fake ads and in [DGZ12], they find gold standard users to find probability of fraud. In [KNB08], they use reverse ad algorithm which checks whether a system is a robot or not. In [**?**], the research

deals in the earlier stages of working with click fraud, it deals with the selection of appropriate features for best results. In [QZL18], deals with the usage of ensemble model cascading gradient boosting model. By comparing all the related works and to the best of our knowledge, our work is the very first attempt done with deep learning hybrid model.

By comparing all the related works and to the best of our knowledge, our work is the very first attempt done with deep learning hybrid model consisting of ANN, auto-encoder, and semi-supervised GAN.

2.3 Dataset Characteristics, Challenges and Experimental Setup

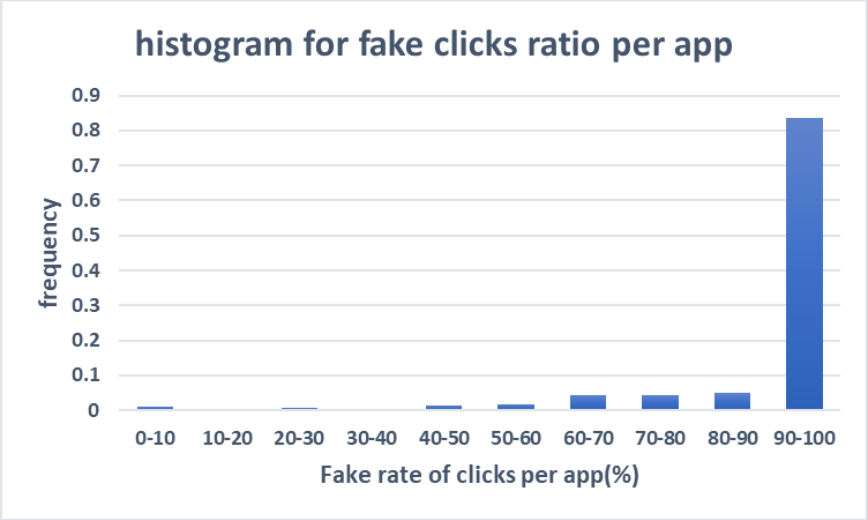
Dataset Characteristics: For our experiment, we have used real-time dataset provided by kaggle [Kag18]. The data were collected for four days (2017/11/06 to 2017/11/09). The dataset contains 184903890 number of real-time ad clicks observations collected on a mobile platform. It contains eight attributes in which seven are features (independent attributes), and one is a label (dependent attribute). The overall size of the dataset is around 7 GB. Table 2.1 describes the characteristics of the dataset. The dataset consists of 277396 unique ip's, 706 unique app id's, 202 unique channel id's, 3475 unique device id's, and 800 unique os version id's. All these are encoded due to the privacy factor. The Figure 2.2 shows the histogram plots on unique ip's and app id's. The plot in Figure 2.2a shows fake clicks ratio per app id where 90 percent of the clicks generated are suspicious, the plot in Figure 2.2b shows the number of observation per famous app id's where there are 5 app id's which has 5.1 (+ or -) 1.7 millions of observations and the plot in Figure 2.2c shows ip's participation in fake and valid clicks. Figure 2.2d shows the heat map

Table 2.1: Characteristics of the Ad Click Dataset

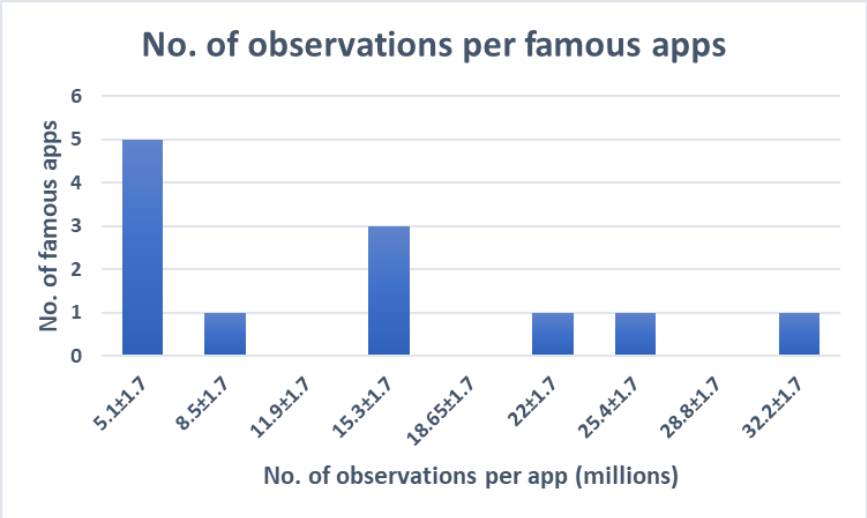
Attributes	**Description**	**Attributes**	**Description**
ip	Ip address of click	app	App id for marketing
device	Device type id of user mobile phone (example: iPhone 6, iPhone 6 plus, Samsung Galaxy 8, etc.)	attributed_time	If user download the app for after clicking an ad, this is the time of the app download
channel	Channel id of mobile ad publisher	os	Os version id of user mobile phone
Is_attributed	The target that is to be predicted, indicating the app was downloaded	click_time	The time stamp of click (UTC)

generated for each attribute compiling the null values in the dataset. We see areas with lighter blank spaces which consist of null values. Hence, we drop attributes with maximum null values to have a clean dataset. We see that attributed time has maximum null values. Hence we drop the column and achieve a clean dataset.

Dataset Challenges: The vast dataset is not a problem, actually very beneficial, so long as it is evenly distributed. However, our dataset was overwhelmingly slanted towards negative or fake clicks, which created a problem in determining accuracy, i.e. even if we do not predict a single positive or real click, it will still give approximately 100% accuracy. However, we did not get 100% accuracy in the case of deep learning model implementation when we took an evenly distributed dataset. For this kind of distribution, we were able to achieve an accuracy of 94-95%. Based on ip address of click and app id for marketing we divide the dataset into six classes as as follows: *Class 1: Hypo-active users of non-suspicious apps:* Combination of observations with unique ip participation count less than 20 times



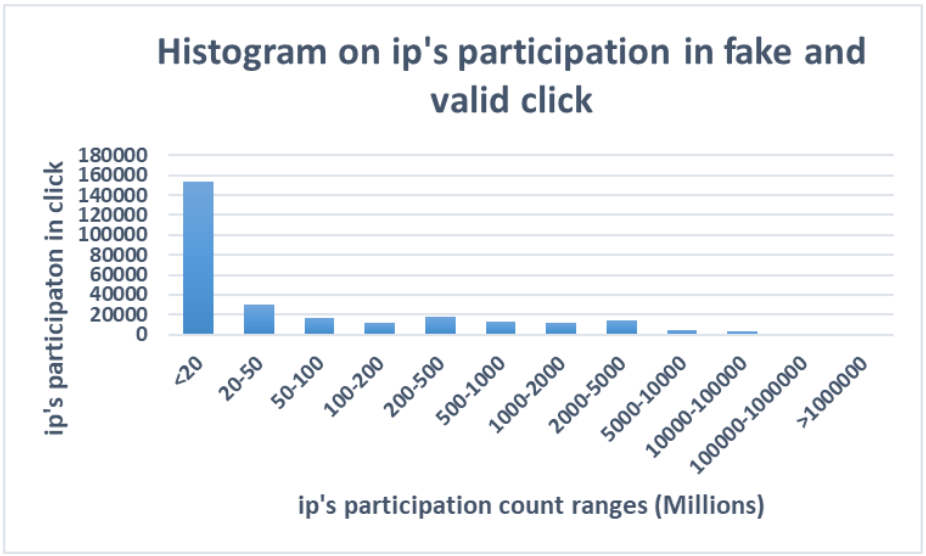
(a)



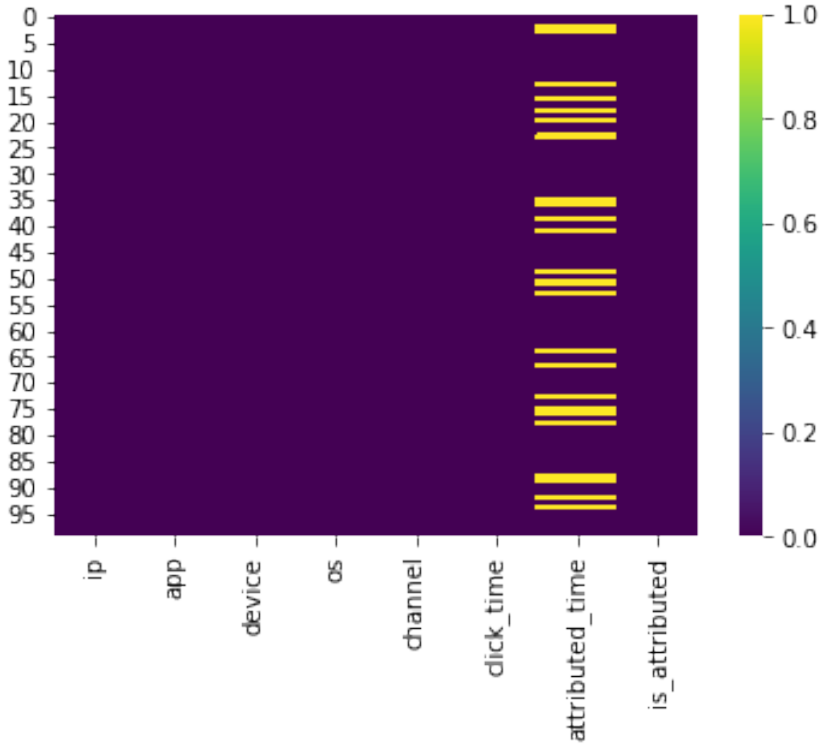
(b)

Figure 2.2: Pre-Analysis on the Dataset with Respect to Ip's and App Id's. (a) Histogram for Fake Clicks Ratio Per App, and (b) No. of Observations Per Famous App's [TKC+19]

and app id frequency of participation less than 70% with 25974 rows. *Class 2: Active users of non-suspicious apps:* Combination of observations with unique ip participation count in the range (greater than or equal to 20 times and less than 1000 times) and app id frequency of participation less than 70% with 27174 rows. *Class 3: Hyperactive users of non-suspicious apps:* Combination of observations



(c)



(d)

Figure 2.2: (continued) Pre-Analysis on the Dataset with Respect to Ip's and App Id's. (c) Histogram on Ip's Participation in Fake and Valid Clicks, and (d) Heat Map Portraying Null Values [TKC+19]

with unique ip participation count greater than or equal to 1000 times and app id frequency of participation less than 70% with 112790 rows. *Class 4: Hypo-active users of suspicious apps:* Combination of observations with unique ip participation count less than 20 times and app id frequency of participation greater than or equal to 70% with 784964 rows. *Class 5: Active users of suspicious apps:* Combination of observations with unique ip participation count in the range (greater than or equal to 20 times and less than 1000 times) and app id frequency of participation greater than or equal to 70% with 19914810 rows. *Class 6: Hyperactive users of suspicious apps:* Combination of observations with unique ip participation count greater than or equal to 1000 times and app id frequency of participation greater than or equal to 70% with 164038178 rows.

Experimental Setup: All methods and models are experimented on Intel Xeon 8 cores CPU with 32 GB RAM, 100 GB SSD and Tesla K80 GPU with 12 GB memory. Implementation is done in python where train to test data ratio is 3:2 by randomly selecting the rows in the dataset.

2.4 Preliminary Experiments

In order to analyze the relation between the dependent and independent attributes we train and test the following prediction models: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF) and Multinomial Naive Bayes (NB_M). *Performance matrices:* To evaluate the performance of each model we have considered Precision, Recall Accuracy as shown in equation 2.1-2.3 where t_{posi}: True Positive, t_{negi}: True Negative, f_{posi}: False Positive and f_{negi}: False Negative .

$$Precision(P) = \frac{t_{posi}}{t_{posi} + f_{posi}} \tag{2.1}$$

Table 2.2: Preliminary Results

Models	**LR**			**SVM**			**RF**			NB_M		
Performance Metric	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**
Hypo-active users of non-suspicious apps	0.18	0.70	71.33	0.13	0.75	70.87	0.46	0.61	73.43	0.22	0.57	69.74
Active users of non-suspicious apps	0.42	0.57	64.23	0.13	0.87	64.60	0.64	0.69	74.26	0.25	0.61	64.00
Hyperactive users of non-suspicious apps	0.65	0.78	74.19	0.06	0.90	54.57	0.84	0.81	82.83	0.33	0.60	57.36
Hypo-active users of suspicious apps	0.99	0.82	81.45	0.91	0.81	80.22	0.91	0.94	87.28	0.54	0.87	56.23
Active users of suspicious apps	0.99	1.00	99.59	0.99	1.00	99.60	0.99	0.99	99.59	0.66	1.00	66.44
Hyperactive users of suspicious apps	0.99	0.99	99.91	0.99	1.00	99.70	0.99	1.00	99.71	0.93	1.00	92.60

Note: P: Precision, R: Recall, A: Accuracy (%), LR: Logistic Regression, SVM: Support Vector Machine, RF: Random Forest, NB_m: Multinomial Naive Bayes

$$Recall(R) = \frac{t_{posi}}{t_{posi} + f_{negi}} \tag{2.2}$$

$$Accuracy(A) = \frac{t_{posi} + t_{negi}}{t_{posi} + t_{negi} + f_{posi} + f_{negi}} \tag{2.3}$$

Observations: From Table 2.2 and Figure 2.3, 2.4, and 2.5 we can state that RF performs better with minimal error rate and gives us the highest accuracy rate.

2.5 Deep Learning Approach

After having some degree of success in the above approaches, we wanted to develop an algorithm to detect and discard fake clicks in real time. For that, we used Deep

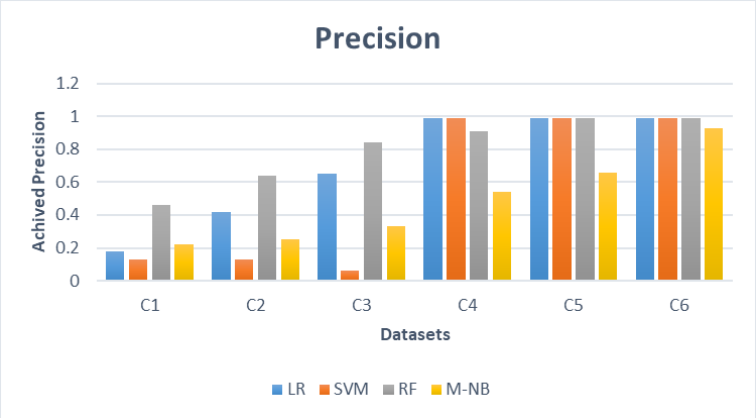


Figure 2.3: Precision Comparison

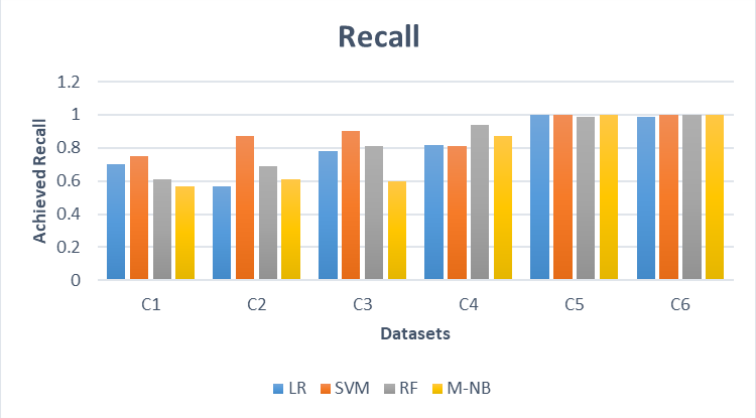


Figure 2.4: Recall Comparison.

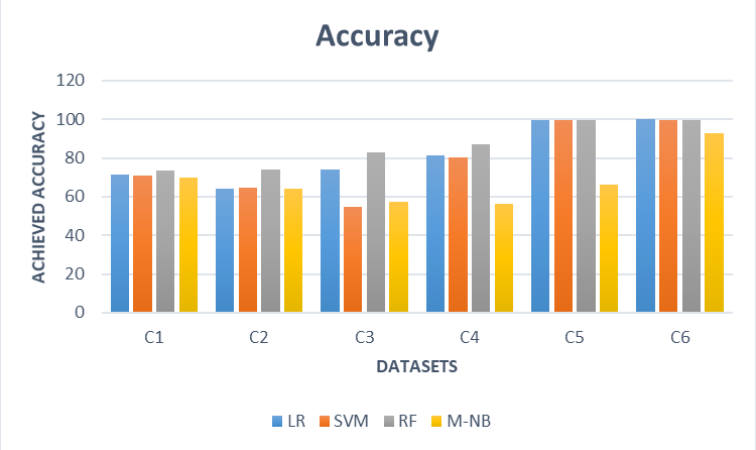


Figure 2.5: Accuracy Comparison.
Note: C1: Hypo-active users of non-suspicious apps, C2: Active users of non-suspicious apps, C3: Hyperactive users of non-suspicious apps, C4: Hypo-active users of suspicious apps, C5: Active users of suspicious apps, C6: Hyperactive users of suspicious apps

Learning methods, which also substantially improved our accuracy. We built a multi-layered Neural Network with an attached autoencoder using Keras backend Tensorflow [Cho15, RHW86]. We developed an approach based on Replicator Neural Network [VAK+16] for detecting bots. Due to a heavily imbalanced dataset, we used a semi-supervised GAN to generate fake data in order to act as an attack and also increase the accuracy of our Neural Network. Figure 2.6 depicts our hybrid deep learning model.

Work Flow: (1) User clicks on the link; the info is saved and sent to an autoencoder, (2) A trained Autoencoder regenerates the data after adding some noise to it, (3) If regeneration loss is more than the threshold, the user is discarded as a bot, (4) If not, the user is considered human and his/her info is passed to a neural network, and (5) The Neural Network predicts whether the user has an intention of downloading the app or not.

Below the concepts behind our scheme are explained. We use the dataset to train both the supervised Neural Network and Unsupervised Autoencoder. Since the data used is based on human clicks, it is easy to train the autoencoder to recognize human behavior and discard the bots. Our model follows the pattern of Batch Normalization after every Dense layer. To stop the model from overfitting, we put a dropout after every layer with the probability of 0.2 [SHK+14]. We initialize the parameters of the hidden layers using Xavier[GB10a] and and trained the network using adam[KB14]. Table 3 shows the results. In Receiver Operating Characteristic (ROC) graph, the True Positive rate is plotted in a function of False positive Rate. Each (x, y) pair represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under this curve perfectly summarizes the Specificity (False positive rate) and Recall (True Positive rate). We use ROC curves to understand how well the network is working. Table 2.3 shows the results and Fig.

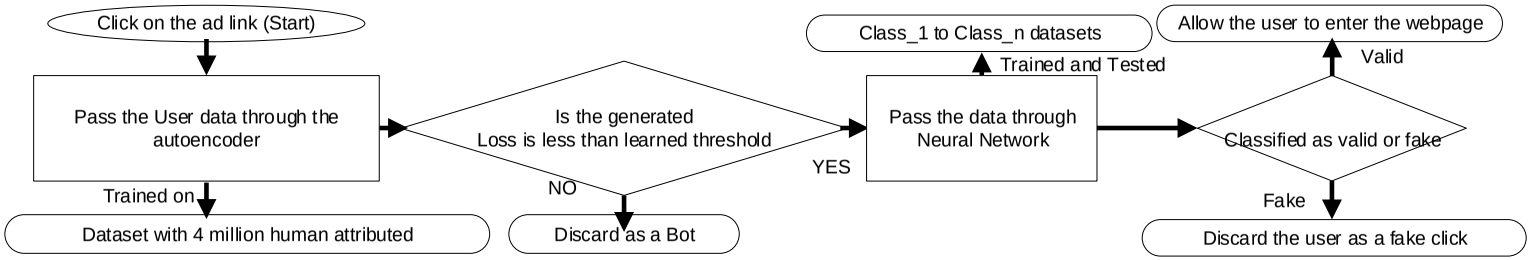


Figure 2.6: Multi-layered Neural Network with an Attached Autoencoder Model

2.7-2.13 shows the plots of ROC for all class of datasets.

Table 2.3: Results from Neural Network

Dataset	**P**	**R**	**A(%)**	**AUC**
class 0	0.95	0.94	94.00	0.979
class 1	0.71	0.73	73.53	0.899
class 2	0.74	0.75	74.86	0.877
class 3	0.81	0.81	81.40	0.897
class 4	0.89	0.88	90.01	0.802
class 5	0.98	0.97	99.50	0.947
class 6	0.99	0.99	99.80	0.713

Note: Class 0: Equal distribution of valid and fake, P: Precision, R: Recall, A: Accuracy, AUC: Area under the ROC Curve

2.5.1 Auto Encoder

In an autoencoder, there are two parts [VLBM08].

Encoder Here noise is added to the real data and is passed through a neural network. Let $x_{data} = OriginalData$, we add noise using a random neural network which is not trained as

$$x_{noise} = neural\_net(x_{data}) \tag{2.4}$$

Now we use the encoder's neural network to compress the data

$$x_{compressed} = encoder(x_{noise}) \tag{2.5}$$

Decoder Here the output of encoder travels through another Neural network.

$$y_{pred} = decoder(x_{compressed}) \tag{2.6}$$

The Decoder output $y_{pred} = decoder(x_{compressed})$ is compared with real input using root mean squared and based on the error $RegenerationLoss = E = Error =$

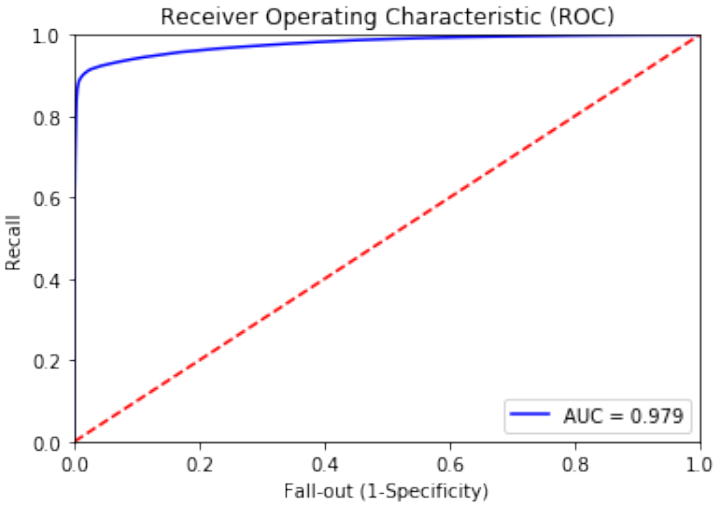


Figure 2.7: Equal Distribution of Valid and Fake.

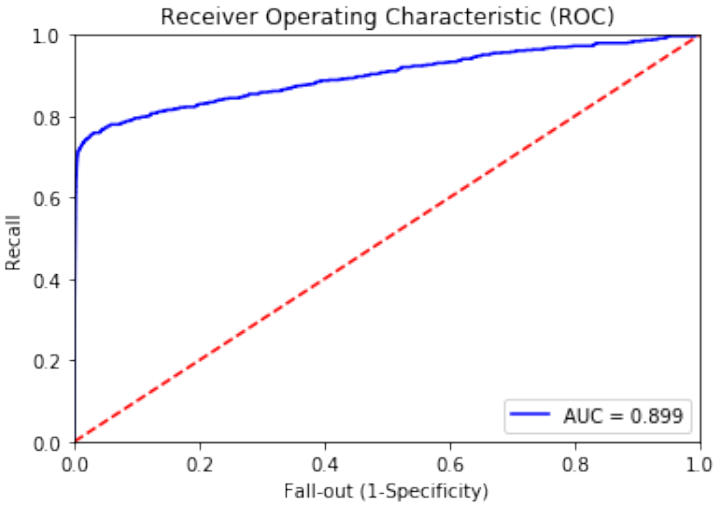


Figure 2.8: Hypo-active Users of Non-suspicious Apps.

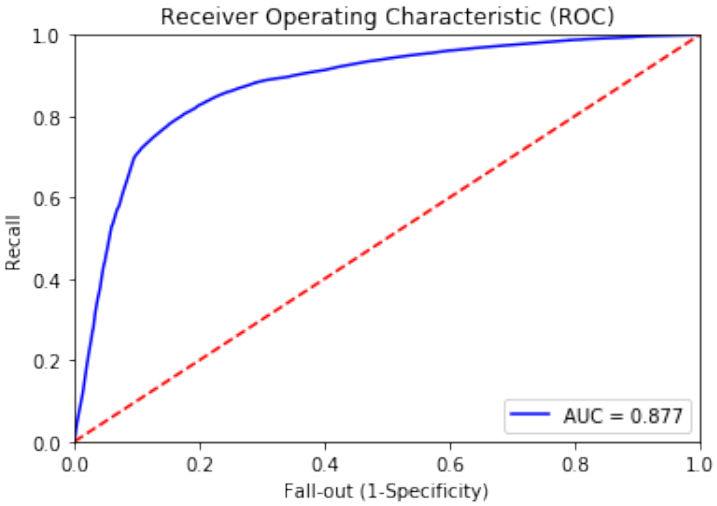


Figure 2.9: Active Users of Non-suspicious Apps.

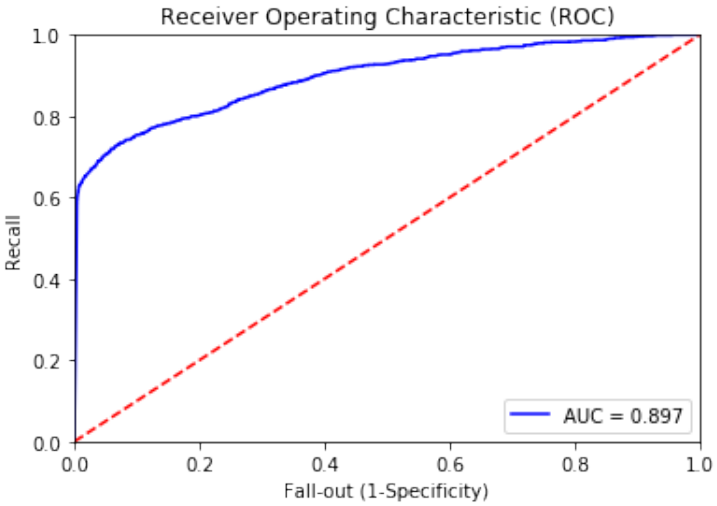


Figure 2.10: Hyperactive Users of Non-suspicious Apps.

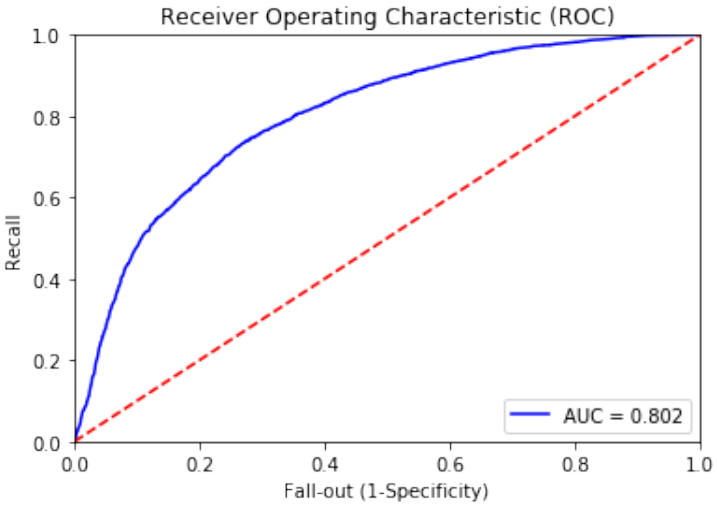


Figure 2.11: Hypo-active Users of Suspicious Apps.

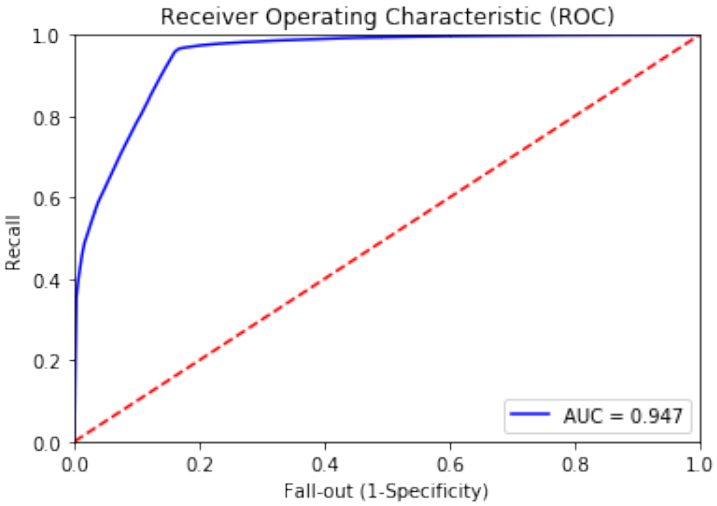


Figure 2.12: Active Users of Suspicious Apps.

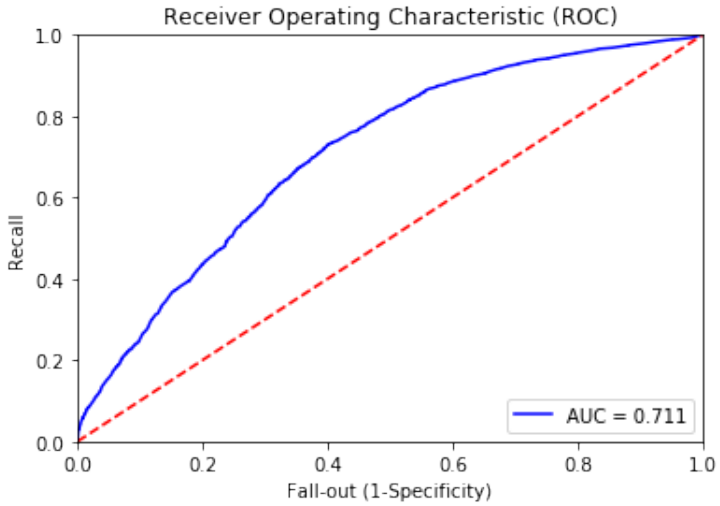


Figure 2.13: Hyperactive Users of Suspicious Apps.

$sqrt(x_{data} - y_{pred})^2$, the auto encoder is then trained using Adam [KB14].

$$RegenerationLoss = E = Error = sqrt(x_{data} - y_{pred})^2 \tag{2.7}$$

We trained the autoencoder [VLBM08] on a dataset of 3 million humans. Since the autoencoder is unsupervised, it cannot predict whether the user is real or fake. However, the autoencoder was trained to regenerate the data. It learned the distribution of data for which it was trained. For example, if an autoencoder that has only been trained on the human dataset and the Regeneration loss for click is higher than a decided threshold, then we can discard that as a bot. Since we did not have the data classified as bots and humans, therefore we could not show how well this method will work. However, a similar technique was used by Veeramachaneni et al. [VAK+16] with their Replicator Neural Network. They got some great results for detecting frauds based on regeneration error.

2.5.2 Semi-supervised GAN

We develop a semi-supervised GAN [GPAM+14, KMRW14] to generate fake samples as a smart attacker for the Neural Network.

Generator

In Figure 2.14, we show how we take care of generating different attributes.

$$x_{noise} = uniform\_distribution(-1, 1) \tag{2.8}$$

$$x_{generated}, x_{generated\_logits} = generator(x_{noise}) \tag{2.9}$$

After the $softmax$ function is applied to each attribute vector, $max_one_hot_encode$ to take the highest value of vector and make it 1, while others are converted to zero. We also concatenate the vectors with $softmax$ activation: $x_{generated\_logits}$.

Discriminator

It gives two outputs, $discriminator\_1$ value denotes whether the given data is real or generated, $discriminator\_2$ value denotes whether the given data indicates a valid click or not.

$$x_{g\_out} = discriminator(x_{generated}) \tag{2.10}$$

$$x_{r\_out1} = discriminator\_1(x_{real\_data}) \tag{2.11}$$

$$x_{r\_out2} = discriminator\_2(x_{real\_data}) \tag{2.12}$$

$$x_{classified\_1} = \frac{1}{1 + e^{-x_{g\_out}}} \tag{2.13}$$

$$x_{classified\_2} = \frac{1}{1 + e^{-x_{r\_out1}}} \tag{2.14}$$

$$x_{classified\_3} = \frac{1}{1 + e^{-x_{r\_out2}}} \tag{2.15}$$

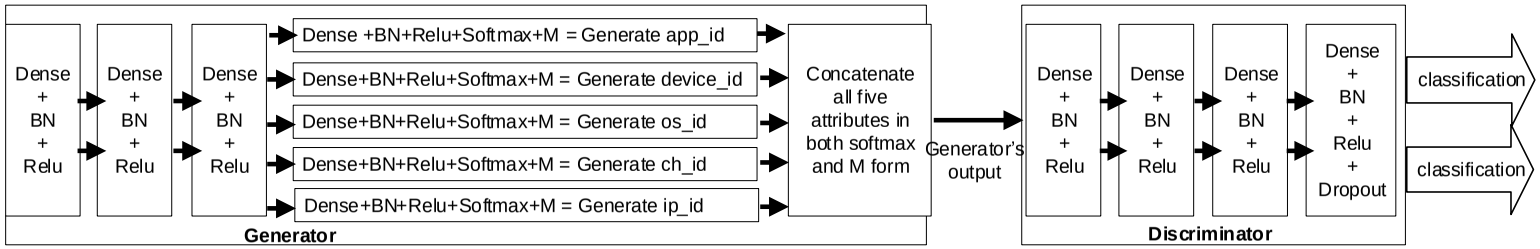


Figure 2.14: GAN Structure

Note: $(BN : BatchNormalization, M : max\_one\_hot\_encode, Dense : Singlelayerneuralnet, Relu : max(0.2 * x, x), x : OutputofBatchNormalization)$

$x_{classified\_1}$: Probability that the generated data is real. $x_{classified\_2}$: Probability that the real data is real. $x_{classified\_3}$: Probability that the click in real data is valid.

Loss

Discriminator and Generator loss are as follows.

Discriminator Loss let *supervised* be *sp* and *unsupervised* be *usp*

$$D_{usp\_gloss} = -\log\left(1 - x_{classified\_1}\right) \tag{2.16}$$

$$D_{usp\_dloss} = -\log\left(x_{classified\_2}\right) \tag{2.17}$$

$$D_{sp\_loss} = -y * log x_{classified\_3} + (-(1-y)) * log(1 - x_{classified\_3}) \tag{2.18}$$

$$D_{loss} = D_{usp\_gloss} + D_{usp\_dloss} + D_{sp\_loss} \tag{2.19}$$

Generator Loss

$$G_{usp\_loss} = -\log\left(x_{classified\_1}\right) \tag{2.20}$$

$$G_{sp\_loss} = -(log(Cos(\theta))) = -log\left(Cos\left(\frac{(x_{real\_data} * x_{generated\_logits})}{(|x_{real\_data}| * |x_{generated\_logits}|)}\right)\right) \tag{2.21}$$

$$G_{loss} = G_{usp\_loss} + G_{sp\_loss} \tag{2.22}$$

Both these losses are used to train the Generator and discriminator separately.

Results

We trained our GAN on a dataset with equally valid and fake clicks; the discriminator got an accuracy of 89.7%. Based on discriminator's classification, the generator created 63000 valid clicks, given 100000 randomized inputs. To make sure that we were getting the correct results, we did a dot product of randomly selected real valid clicks and the ones generated by the GAN, we got an average result of 0.65, while the average results for the dot products of the same real valid clicks with each

other was 0.85, showing that there is a great variance even within the real valid clicks. On the other hand, the dot product of the generated valid clicks and real fake clicks was 0.24. We even added this data to the equally distributed training dataset to train the Neural Network, and saw an increase of 0.6-1%, from 94% to 94.6-95% on the same test cases as used in Table 3, depending on the structure of the Neural Network. The increase may be a minute for the given dataset since the neural network was already performing well, but this technique can be used in other one-sided sparse data sets too. To the best of our knowledge, this is the first work to use $Cos(\theta)$, as a supervised loss function in a semi-supervised GAN.

2.6 Conclusion

As shown by the above results, we have achieved good high accuracy even on small datasets, showing the capability of the neural network to understand the probability distribution of fake and real users. The reason for the success of the neural network in this form of fraud detection is because of multiple layers, with each layer learning the distribution to a certain extent and passing that knowledge to the next layers. The Auto-Encoder can understand the distribution of human clicks since the data available belongs to human beings. This allows it to encode and then decode the data; it is a type of decryption and encryption. Since it has only been trained to encrypt and decrypt human data, it can be said that the loss error will be high for bots. However due to lack of bot clicks available, the auto-encoder in current form will not be able to detect bot with a distribution similar to humans, but when put in practice it will be able to learn botnet distribution too. It is different from a standard neural network since the loss generated is ambiguous, which means, we will have to change the loss threshold from time to time depending on the success and

failure of auto-encoder. Neural Network will fail to detect bots if they maliciously download, but the autoencoder will discard them.

CHAPTER 3

CFXGB: AN OPTIMIZED AND EFFECTIVE LEARNING APPROACH FOR CLICK FRAUD DETECTION

Click Fraud is a fraudulent act of clicking on pay-per-click advertisements to increase the site's revenue or to drain revenue from the advertiser. This illegal act has been putting commercial industries in a dilemma for quite some time. These industries think twice before advertising their products on websites, as many parties try to exploit them. To safely promote their products, there must be an efficient system to detect click fraud. Currently, deep learning models are employed for click fraud detection. However, due to certain obstacles, a significant computational overhead is incurred in search for the best model.

To address this problem, we propose a model called CFXGB (Cascaded Forest and eXtreme Gradient Boosting). The proposed model, classified under supervised machine learning, is a combination of two learning models used for feature transformation and classification. We showcase its superior performance compared to other related models. We make the comparison with multiple click fraud datasets with varying sizes. Several intrusion detection datasets were also used to validate its efficacy against deep learning models.

3.1 Introduction

The recent approaches to detect click fraud use deep learning models [TKC+19, HAH+18]. These models require intense hyperparameter tuning for excellent performance. The data is enormous since there are multiple features and the number of clicks is growing by the minute. In real-time systems and business applications, advertisement data is usually of gigantic volumes resulting in the feature space di-

mension to drastically increase. This data is then used to re-train these real-time models which includes hyperparameter tuning on the new dataset. This re-training must be done periodically. As we all know, neural networks are black-box models and the number of parameters to be tuned are many. This tuning would require time and a powerful machine to process the data.

Google AdWords [fru19] is an advertising network that has a system in place to detect click fraud. This system is a three-tier system in which they first use automated filters that detect suspicious clicks in real-time. Then, offline analysis by automated algorithms and human analysts occurs. Finally, Google performs an in-depth investigation of the complaints from advertisers. As seen by these steps, the whole process of detecting click fraud is not entirely automated. Also, the current literature indicates that the existing detection models are less effective and require more human intervention.

3.1.1 Summary of Contribution

To address these problems :

- We extend the work done by [ZF17] Zhou et al. by proposing a two-phase model consisting of feature transformation by Cascaded Forests and classification by eXtreme Gradient Boosting (CFXGB).
- Our approach requires only minimal hyperparameter tuning i.e., The number of parameters that vary are few.
- Use of forests in the form of layers is more intuitive in understanding the model's learning method as compared to the use of neural nets.
- Accuracy, Area-Under-Curve (AUC), Recall, F1-score and Precision are used to evaluate the superiority of the proposed model against recent works.

- We have evaluated our model on five benchmark datasets. Three of them are click fraud datasets and two of them are intrusion detection datasets.

3.1.2 Organization of the Chapter

In section 8.2, we discuss the related work regarding Click Fraud, Cascaded Forest and eXtreme Gradient Boosting (XGBoost). In section 3.3, we present our proposed approach i.e., CFXGB. We analyze the total pipeline of the proposed approach and then we list all the parameters given to the model. In section 7.4, we initially present the pre-processing performed on each dataset in detail followed by the results obtained by running these processed datasets on the model and finally provide a comparison with prior works. In section 7.5, we discuss the evaluation metrics, results and justify the model parameters.

3.2 Related Work

3.2.1 Click Fraud

Various methods have been designed to predict click fraud. A statistical model has been created to identify the IPs responsible for the fraud attacks [KKL+14] and based on this information, classification is done. In [APS+11], Splay trees are used for the storage of IP addresses through which burst of fraud clicks occur. In [Had10], average clicks on bluff ads are taken as a way of discrimination. In another model [CdQC12], CAPTCHAs are used to ensure that the click is legitimate. In [FLDH+16b], Social Network Analysis is used to find three top ranked ad networks used to inject the click-fraud malware. In [TKC+19], a hybrid deep learning model consisting of an Auto Encoder, a Neural Network and a Semi-supervised Generative

Adversarial Network (GAN) was devised to predict click fraud. In [LLCX15], if the ad is shown to be attested, they first find the eigenvalues of the displayed ad images. Then, if the eigenvalues of the image match those stored in their server, the click is certified as legitimate. Coalition made by fraudsters can be discovered by similarity-seeker algorithm [MAEA07]. A reverse ad algorithm has been devised [KNB08] that checks whether a system is a robot or not. Another method has been devised to find the probability of fraud by finding gold standard users [DGZ12]. The idea of appropriate feature selection [KJ97] for optimum results in the earlier stages of working with click fraud is another approach. Due to the immense data associated with advertisements, to reduce feature dimensions, embedding techniques [LH15, COL17] and neural networks have been used. Hence, essential information was extracted. Another method used to extract meaningful information was devised by Google researchers [WFFW17]. Structures of deep learning models differ depending on the problem as discussed in [ASKR12, SMKR13]. Their structures depend on the size of the dataset and these models usually require numerous parameters to be tuned. A model called RTILKE [XJWX19] obtains the embedding of data by learning a robust similarity function.

3.2.2 Cascaded Forest

Cascaded Forests is a part of the gcForest model proposed in [ZF17]. GcForest is one of the ensemble based models as they use multiple learners to obtain a combined result. Ensemble models facilitated with deep neural network features yield superior results as compared to using only deep neural networks [KFCRB15].

As stated in [ZF17], the deep forest model tries to mimic the functionality of deep learning models without the intense hyperparameter tuning through certain

features which are listed below:

1. Cascade-by-Cascade Processing
2. In-model feature transformation
3. Dataset Flexibility

The cascade forest in the gcForest works based on Boosting [FS97]. It decides the number of cascades based on the dataset. Cascade structures have had outstanding performances in object detection tasks [VJ+01]. Each grade of the cascade is an ensemble of an ensemble. In [Web00], Bagging is used as a base learner for boosting. To achieve feature transformation, GcForest uses ensemble approach in the same grade. After processing the data in one grade of learners, the processed output is used as the input for the next grade [Bre96, TW99]. Cross-validation is used between grades based on other studies [Zho12, TW99]. For a good ensemble to overcome overfitting, the constituent learners in each grade must exhibit a high level of accuracy and diversity. In our case, we have enhanced diversity by adding another type of forest. Other models that use gcForest are Ensemble Trees and Cascaded Model (ETCF) [QZL18] which is a model for Click-through Rate prediction. It uses Gradient Boosting for feature transformation and then gcForest for classification.

3.2.3 XGBoost

XGBoost or Extreme Gradient Boosting is a classifier developed by Chen et al. [CG16]. This model is used for supervised learning. Due to its high algorithmic efficiency, the execution speed and model performance are very high.

Boosting is a technique where models are added sequentially, improving the previous models by correction of errors. Gradient boosting is an approach where

models are created that predict the residuals of preexisting models and then a final prediction is made.

XGBoost is vital to our approach as it is used in the Cascaded Forest for feature transformation and acts as our primary classifier.

Some of the features of XGBoost are listed below :

1. Parallelization
2. Out of core computation
3. Cache optimization
4. Distributed computing

3.3 Proposed Approach

3.3.1 Pipeline

Figure 3.1 depicts the pipeline of our approach. It consists of three stages i.e., Pre-processing, Feature transformation based on Cascaded Forest and XGBoost classification with minimal hyperparameter tuning. Let $R^{n \times d}$ be the set of all real-valued matrices with n rows and d columns. We consider the dataset A ϵ $R^{n \times d}$ that represents any of the datasets considered for experimentation. Let A_{ij} represent the i'th row and j'th column, A_{i*} represent the i'th row and A_{*j} represent the j'th column of A. Let the dataset A be read using pandas and be considered a data frame.

Pre-processing

For A, if any Null or Nan values exist in any row A_{i*}, that row is dropped. The model accepts only int and float datatypes. Dataset A might have a column that

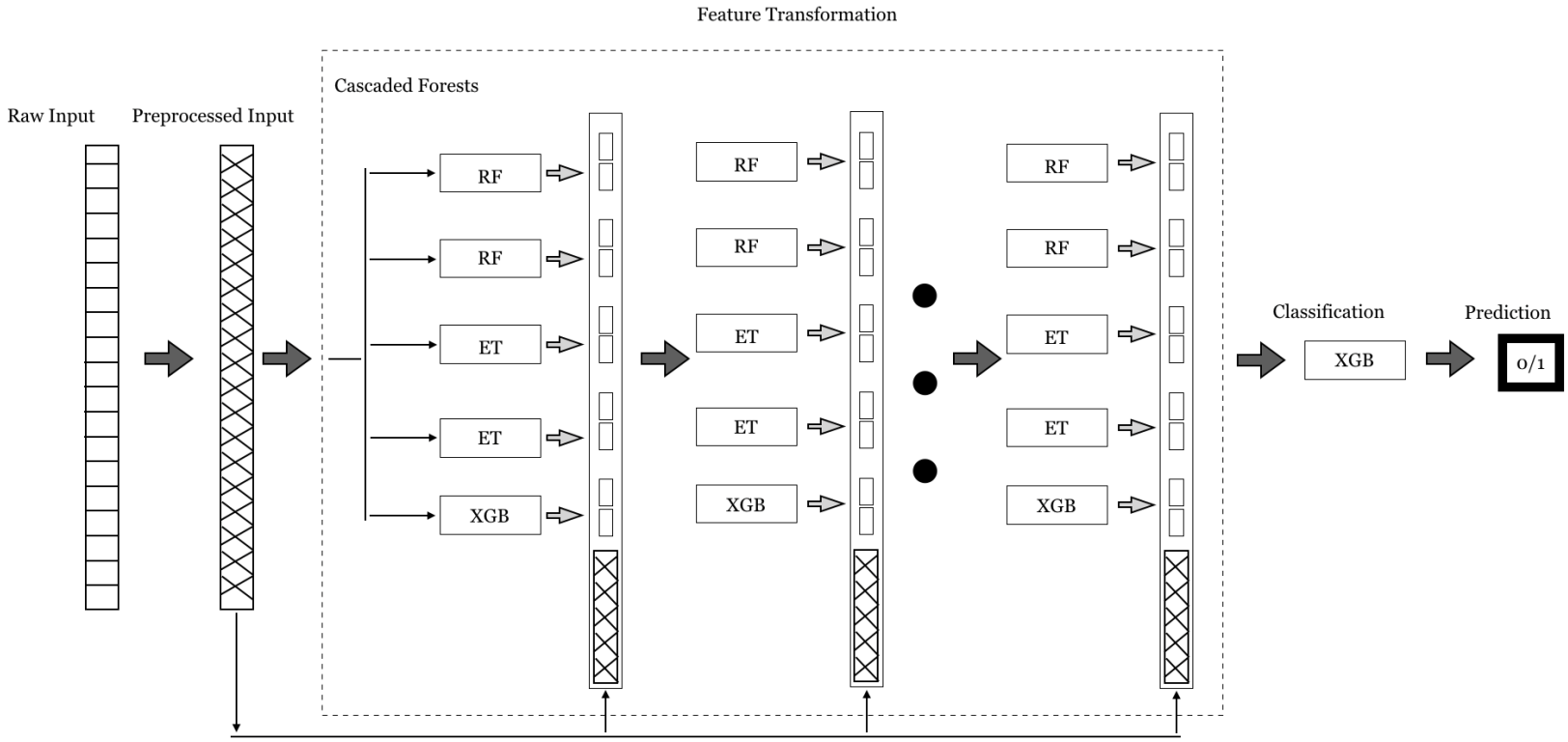


Figure 3.1: Proposed CFXGB Pipeline

represents the timestamp of click (for click fraud datasets). Let this column be A_{*j}. If A_{*j} datatype is not in the datetime64 format, then it will have to be converted to it. The converted column is then split into separate columns A_{*j+1}, A_{*j+2}, ..., A_{*j+n} based on their time distribution, for example, day, hour, minute and second. Upon the completion of column splitting, the original column A_{*j} is dropped from A. Now the Dataset A is split into four datasets W, X, Y, Z for training and testing the data. Testing data has 20% of the data.

$$W = A^{a \times d-1} \tag{3.1}$$

$$X = A^{a \times -1} \tag{3.2}$$

$$Y = A^{n-a \times d-1} \tag{3.3}$$

$$Z = A^{n-a \times -1} \tag{3.4}$$

where ***a*** *represents 80% of n i.e., Number of rows*

W and X are the training datasets (train_x(3.1) and train_y(3.2)), and Y and Z are the testing datasets (test_x (3.3)and test_y(3.4)). If there is no application of any sampling method, then W, X, Y and Z need to be converted from pandas data frame format to array format. Finally, we feed W and X into the Cascaded Forest for feature transformation.

Cascaded Forest Feature Transformation

The Cascaded Forest is made up of three ensemble models which are Random Forests [Bre01], Extremely Randomized trees (Extra Trees) [GEW06] and XGBoost[CG16]. Each of these ensemble models helps boost the model's performance. The flowchart of this transformer is given in fig 4.1. The cascade forest structure proposed by Zhou

et al.[ZF17] is used as the feature transformer. Cascaded Forests work in the form of layers. The output data of a cascade is given to the next cascade for processing. In addition to the Extremely Random Forest and the Random Forest in Zhou's [ZF17] model, the XGBoost Classifier was added to the Cascaded Forests to enhance diversity (Based on equation (3.5) obtained from error ambiguity decomposition [ZF17, KV95])

$$E_{ensemble} = \overline{E(Classifier_i)} - \overline{A(Classifier_i)} \tag{3.5}$$

where error of the ensemble is indicated by $E_{ensemble}$*, the average error of individual classifiers in the ensemble is indicated by* $\overline{E(Classifier_i)}$ *and the diversity among the individual classifiers is denoted by* $\overline{A(Classifier_i)}$ *where* $i = 1, 2, 3, ..., n^{th} classifier$

The training sets W and X are used to obtain transformed features. Let us consider a binary classification. Each forest outputs the likelihood of data belonging to a class for all the classes. Hence, there will be two classes outputted by each forest for each data observation. All the class vectors from each forest are concatenated together and then re-concatenated to the original data, as shown in figure 1.
K-fold validation is applied to prevent overfitting for each class vector from a forest. The number of cascades formed depends on the number of early stopping rounds given. Early stopping rounds is a parameter used in the Cascaded Forest which is used to limit the number of layers to be added. We assume three early stopping rounds. Once it detects there is no increase in accuracy in 3 layers, it will stop cascading additional layers. Finally, the last layer would output a new encoded array after re-concatenation. Let us consider datasets W and X again which have data sizes $a \times (d-1)$ and $a \times 1$ respectively. Upon feeding them to the Cascaded Forest, considering five forests in total, we get 10 class vectors as output (Two class vectors from each forest). These class vectors are compared with the training label

to obtain accuracy for that cascade. We concatenate all these class vectors with W and get a new data array with size $a \times (d+9)$ for W. This array is then fed to the next cascade with the same forests. We again obtain 10 class vectors from this encoded array and calculate accuracy in the same fashion. Once again, we concatenate them with the original array W. This cycle will continue until the number of cascaded layers exceeds a specified threshold parameter called max layers or until there is no improvement in accuracy over the number of early stopping layers. Max layers is a parameter that limits the number of layers. It stops the cycle of the addition of layers even if there is a possibility of improving accuracy. But, if this parameter is set to zero, only the early stopping rounds parameter will stop the cycle. Finally, the last layer will again output 10 class vectors. These vectors can be used to calculate the accuracy of the Cascaded Forest (as a classifier). However, the approach we have used is to re-concatenate the class vectors obtained from the final layer to the original data array W and perform classification by XGBoost. Let this concatenated array be E.

Similarly, we transform the testing array in the same fashion. Let this array be E'. The XGBoost classifier uses array E' for testing.

XGBoost Classification with Mini Hyperparameter Tuner and Prediction

The XGBoost classifier or Extreme Gradient Boosting [CG16] then trains itself on the encoded array E. The parameters are mildly tuned based on just two parameters i.e., maximum depth and learning rate. The list of parameters given for tuning is indicated in section 3.3.2. Once the training is complete, XGBoost classifier predicts whether a click is fraudulent or not on all observations in array E'. Finally, we compare these results with the actual values and output the corresponding results.

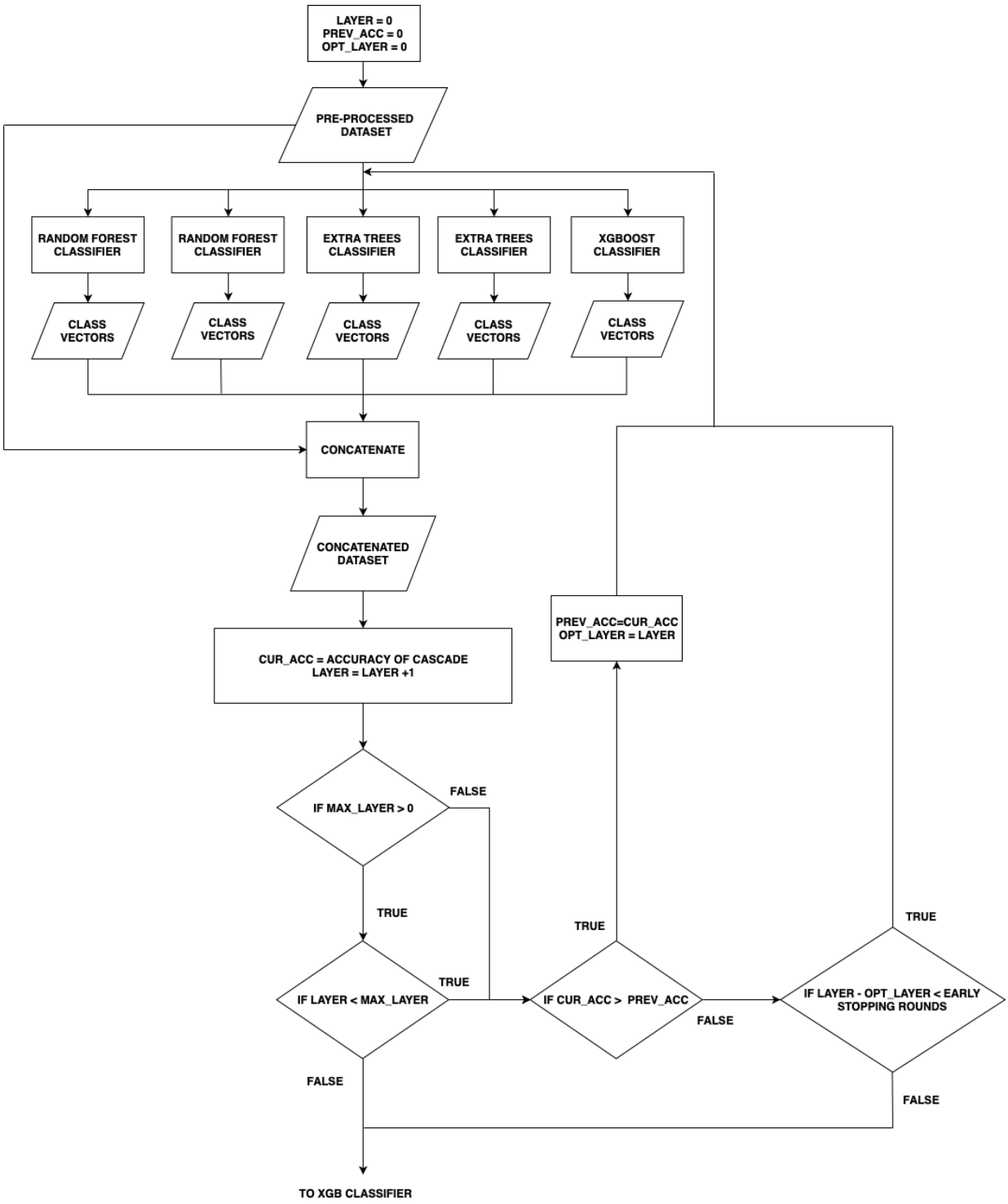


Figure 3.2: Flowchart Depicting Feature Transformation by Cascaded Forest

This model proposes the use of Cascaded Forests as a feature transformer by concatenating new features that can help the Xgboost classifier understand the dataset better. It also excludes the Multigrain Scanning feature of deep forests and still works well with fewer parameters that are to be tuned.

3.3.2 Parameters Considered

Cascaded Forest

- Random Forest Classifier: 2
- Extra Trees Classifier: 2
- XGBoost Classifier: 1
- Early Stopping Rounds: 3
- For all Classifiers :
    1. Number of Folds: 5
    2. Number of Estimators: 100
- Other parameters for XGBoost
    1. Learning Rate: 0.3
    2. Max Depth: 4

XGBoost Classifier

- Number of Estimators : 100
- Mini Hyperparameter Tuner
    1. Learning rate : [0.05,0.1,0.2,0.3]
    2. Max Depth : [2,3,4]

3.4 Experiment

3.4.1 Datasets

We conducted experiments on five datasets in which three datasets were click fraud prediction datasets and two were intrusion detection datasets. We also consider intrusion detection datasets to validate our claims of our model performing better than deep learning models in section 3.4.2.

Click Fraud Prediction Datasets

- TalkingData dataset [Kag18]
- Avazu dataset [Kag15]
- Kad dataset[Kag17]

Intrusion detection Datasets

- UNSW-NB15 dataset [uns15]
- CICIDS2017 dataset [unb17]

3.4.2 Works Compared

We compare our proposed approach to the following models:

- Click Fraud Prediction:
  ETCF [QZL18], Hybrid Deep Learning [TKC$^+$19], RTILKE [XJWX19] models
- Intrusion Detection:
  DNN [FD19] and other comparisons [MS15b, GH16, MS16, PT17, Fri02, HHLD, RD18].

The Kad dataset is used to prove the flexibility of our proposed model in comparison with deep learning models that cannot adapt to some datasets and are sometimes overly complicated.

3.4.3 Experimental setup

All the experiments were conducted on an Intel I7 4 core CPU with 16GB RAM and the Flounder server provided by FIU. (AMD Opteron Processor 6380 with 64 cores and 504GB RAM). Implementation is done in Python.

3.4.4 Data Pre-processing

In this section, we discuss the data processing performed for each dataset before feeding the data into the Cascaded Forests. We have attempted to do the same pre-processing for a fair comparison with previous works.

TalkingData Dataset

TalkingData dataset is an AdTracking Fraud Dataset which has records of 200 million clicks with eight features over four days. In the data pre-processing stage, attributed time was dropped. Click time was separated into separate columns i.e., day, hour, minute and second. Two additional columns were added based on repetition of unique IPs in one hour and ten hours.

Case 1: Pruning of certain features was done through the removal of categorical data that have occurrences less than 5. 40,000 rows of data are considered in which there is 1:1 ratio of both classes (click and no click). No sampling here as the dataset is balanced.

Case 2: 1 million rows of data in which the ratio of classes match the ratio at 200 million rows are considered. Under-sampling was used to balance the imbalanced dataset.

Case 3: Same data preprocessing was performed as in [TKC+19] by considering about 900,000 rows of data with a 1:1 ratio of classes.

Avazu Dataset

This dataset is a Click fraud dataset consisting of clicks recorded over ten days. There are about 40 million rows of data with 24 features. We carried out the same pre-processing as with TalkingData dataset i.e., Separation of the 'hour of click' column into separate columns i.e., month, day and hour. The columns from index 4 to 13 were converted from 'object' datatype to 'int' datatype. We also add one column based on click frequency from the device_ip in 10 hours.

Case 1: Pruning of certain features was done through the removal of categorical data that have occurrences less than 5. 20,000 rows of data in which there is 1:1 ratio of both classes (click and no click) is considered. No sampling here as the dataset is balanced.

Case 2: A random sample of 1 million rows of data was considered and under-sampling was applied to balance the imbalanced dataset.

Kad Dataset

This is an Advertising dataset which has 1000 rows and ten features. We divide the 'timestamp' feature as stated for the above datasets into month, day, hour and second columns. We converted the columns 'Country' and 'City' into unique integer values. We also dropped the 'Ad Topic Line' feature.

UNSW-NB15 Dataset

This dataset is an intrusion detection dataset created by a research group in Australia. There are two million records with 44 features in this dataset. Four datasets form the UNSW-NB15 dataset and they were concatenated row-wise. The columns were then renamed based on the information given in [uns15]. We have attempted to do the same pre-processing as said in [FD19] to compare the models. Some other necessary cleaning like removal of spaces in the data was also performed.

CICIDS2017 Dataset

This dataset is another intrusion detection dataset released in 2017. They have 2.8 million records of cyberattacks with 79 columns. There are a total of eight datasets that form the CICIDS2017 [unb17]. They are concatenated row-wise and then the target label was converted from a multi-class into a binary class i.e. either an attack or benign. We leave the multi-class classification for future work. The features Flow Bytes and Flow Packets had data that exceeded the range of float32 resulting in the dropping of these features. Finally, we sampled 1 million rows for training and testing. Due to the imbalance in the data, we applied under-sampling to balance the dataset.

As mentioned in all these cases, all of the datasets with a column having click-time have been separated into their respective time divisions of an hour, minute and second. This separation is done since the model does not take datetime64 datatype.

3.5 Results and Discussion

In this section, we list and explain the several metrics used to measure the performance of the model. Then we present experimental results on the different datasets

listed in 3.4.1 and compare them with other existing models. We then justify the parameters taken for the model and portray the importance of XGBoost as a classifier.

3.5.1 Evaluation Parameters

As given by the equations 3.6 - 3.10, we use AUC, Accuracy, Precision, Recall and F1-score as the evaluation metrics to compare the performance between models. The formulae for each are given below.

$$AUC = \frac{1}{2} \times (\frac{t_{posi}}{t_{posi} + f_{negi}} + \frac{t_{negi}}{t_{negi} + f_{posi}}) \tag{3.6}$$

$$Accuracy = \frac{t_{posi} + t_{negi}}{t_{posi} + t_{negi} + f_{posi} + f_{negi}} \tag{3.7}$$

$$Precision = \frac{t_{posi}}{t_{posi} + f_{posi}} \tag{3.8}$$

$$Recall = \frac{t_{posi}}{t_{posi} + f_{negi}} \tag{3.9}$$

$$F1score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.10}$$

3.5.2 Experimental Results and Comparison

In comparison with the ETCF model [QZL18], we have compared results with TalkingData dataset and Avazu Dataset. The results are in Table 3.1.

In Table 3.2, we compare the Hybrid deep learning model [TKC+19] with our proposed approach on the TalkingData dataset. (Case 3)

In Table 3.3, results compare CFXGB with RTILKE and RILKE model from [XJWX19]. The metric used is AUC.

Table 3.4 highlights the results based on 1 million rows of data for TalkingData and Avazu datasets. (Case 2)

Table 3.1: Comparison with ETCF and other Models [QZL18]

DATASET	MODEL	AUC	P	R	F1
TALKINGDATA (CASE 1)	ETCF[QZL18]	96.0	91.0	90.4	90.4
	SVM[QZL18]	93.2	89.6	87.7	87.6
	Naive Bayes[QZL18]	94.6	90.8	89.7	89.6
	GBDT[QZL18]	94.2	90.6	89.1	89.0
	Random Forest[QZL18]	90.4	87.7	84.6	84.2
	CFXGB	98.97	99.0	99.0	99.0
AVAZU (CASE 1)	ETCF[QZL18]	75.5	69.8	69.7	69.6
	SVM[QZL18]	70.3	65.3	65.2	65.2
	Naive Bayes[QZL18]	72.9	67.3	67.1	67.0
	GBDT[QZL18]	74.1	68.0	67.9	67.9
	Random Forest[QZL18]	68.7	64.7	63.3	62.5
	CFXGB	98.96	99.0	99.0	99.0

Note: P: Precision, R: Recall

Table 3.2: Comparison of TalkingData Dataset (Case 3)

MODEL	AUC	PRECISION	RECALL	F1
HYBRID DL[TKC$^+$19]	97.90	95.0	94.0	94.0
CFXGB	99.97	99.0	99.0	99.97

In Table 3.5 and 3.6-3.7, different models are compared based on the UNSW-NB15 and CICIDS2017 datasets respectively.

Based on the results shown in these tables, CFXGB has surpassed all the latest models in terms of performance.

3.5.3 Discussions

Parameter Sensitivity

Based on several experiments conducted, we have fixed several parameters in the proposed model. These parameters were initialised and used for all datasets considered. For comparative analysis, different values for these parameters were tested for best performance. AUC was used to evaluate the performance of the parameter.

Table 3.3: Comparison of Datasets with [XJWX19]

DATASET	**RILKE[XJWX19]**	**RTILKE[XJWX19]**	**CFXGB**
TALKINGDATA (CASE 2)	82.0	83.0	97.78
AVAZU (CASE 2)	85.0	87.0	92.62
KAD	88.0	89.0	96.81

Table 3.4: Results Based on 1 Million Rows of Data

DATASET	**AUC**	**PRECISION**	**RECALL**	**F1**	**ACCURACY**
TALKINGDATA (CASE 2)	97.04	98.0	98.0	98.0	97.77
AVAZU (CASE 2)	87.24	93.0	93.0	93.0	92.62

All the AUC values were scaled for better visualization in Figure 3.3. The values considered for each parameter is given below.

1. **Early Stopping Rounds**: In the case of early stopping rounds, we have considered the Avazu dataset (Case 2). The values for early stopping rounds tested on are [1,2,3,4,5]. The plot is shown in Figure 3.3(a). The peak is found at value 3.
2. **Number of Estimators**: In the case of Number of Estimators, we have considered the CICIDS2017 dataset. The values of the number of estimators tested on are [50,100,200,400]. The plot is shown in Figure 3.3(b). The peak is found at value 100.
3. **Number of Folds**: In the case of the number of folds, we have considered the Kad dataset. The values of the number of folds tested on are [2,3,5,7,10]. The plot is shown in Figure 3.3(c). The peak is found at value 5.
4. **Maximum Depth in XGBClassifier**: In the case of Maximum Depth in XGBClassifier, we have considered the Kad dataset. The values of depth

Table 3.5: Comparison of Models with UNSW-NB15 Datasets [FD19]

AUTHOR	CLASSIFIER	ACCURACY(%)
Primartha and Tama [PT17]	Random Forest	95.5
	Multilayer Perception	83.5
N. Moustafa, et al. [MS17]	Naive Bayes	79.5
	Expectation-Maximization	77.20
	Linear Regression	83.0
Belouch, et al. [BEI17]	RepTree	87.8
	Naive Bayes	80.04
	Random Tree	86.59
	Decision Tree	86.13
	Artificial Neural Network	86.31
Zewairi, et al. [AZAA17]	Deep Learning	98.99
Faker, et al.[FD19]	Deep Neural Network	99.19
Our Work	CFXGB	99.65

Table 3.6: Comparison of models with CICIDS2017 datasets [FD19]

AUTHOR	CLASSIFIER	ACCURACY(%)
Resende and Drummond [RD18]	Genetic + Profiling	92.85
J. Han, et al. [HHLD]	SVM + Genetic	99.85
Faker, et al.[FD19]	Deep neural network	97.73
Our Work	CFXGB	99.91

tested on are [2,3,4,5]. The plot is shown in Figure 3.3(d). The peak is found at value 4.

5. **Learning Rate in XGBClassifier**: In the case of Learning Rate in XGB-Classifier, we have considered the CICIDS2017 dataset. The values of Learning Rate tested on are [0.01,0.1,0.3,0.5]. The plot is shown in Figure 3.3(e). The peak is found at value 0.3.

Table 3.7: Comparison of models with CICIDS2017 datasets [FD19]

AUTHOR	**CLASSIFIER**	**F1**
Sharafaldin [SLG18]	K-Nearest Neighbour	96.0
	Random Forest	98.0
	ID	98.0
	Adaboost	77.0
	Multilayer Perception	77.0
	Naive Bayes	88.0
	Quadratic Discriminant Analysis	97.0
Our Work	CFXGB	98.99

Table 3.8: Deep Learning Models vs CFXGB in Terms of Unknown Parameters

Deep Learning Models [ZF17]	CFXGB
Activation Functions :	**Cascaded Forest :**
Sigmoid, ReLU, linear etc.	No. of Forests : 5
Construction of Neural Network :	Early Stopping Rounds: 3
No. Hidden layers : ?	No. of folds : 5
No. Nodes in Hidden layers : ?	No. of Trees in forest : 100
Kernel Size : ?	Tree Growth: Till pure leaf
For Optimization :	**XGBoost :**
Learning Rate : ?	No. of trees : 100
Momentum : ?	Learning Rate : {0.05, 0.1, 0.2, 0.3}
L1/L2 weight regularization penalty : ?	Maximum Depth: {2, 3, 4}

Deep Learning Vs. CFXGB

As seen in table 3.8, the number of parameters to be tuned is very less compared to CFXGB. Most of the parameters in this model are fixed and do not require tuning to obtain excellent performance. The greater the number of parameters to be tuned, the higher the computational power and time required to achieve excellent results. The XGBoost classifier has mild tuning for Learning rate and maximum depth to obtain best results for all datasets considered.

XGBoost Importance

The Cascaded Forest transformer can also be used as a classifier. The final layer could output the class associated with each data observation.

However, our performance results show that XGBoost performs better as a classifier with feature transformation by Cascaded Forest. Plots that compare both methods are given in Figure 3.4. We have done comparative analysis by checking performance of the model for different parameter values. The parameters considered for comparison are Early stopping rounds, Number of estimators, Max Depth in XGboost and Learning Rate in XGBoost. For the parameters we have chosen for the model, there is considerable difference in performance between CFXGB and Cascaded Forest classifier. AUC was used to evaluate the performance of the parameter. All the AUC values were scaled for better visualization.

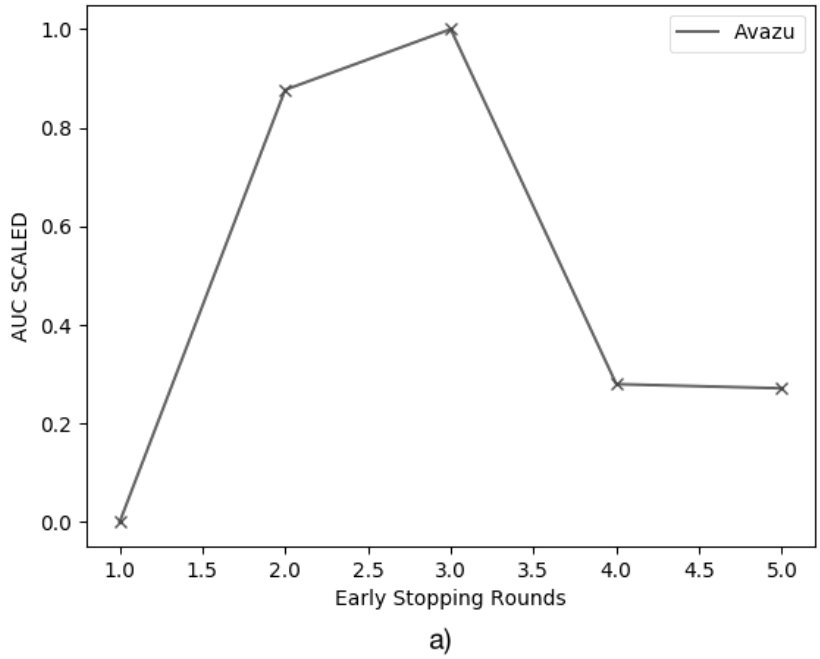
3.6 Future Work

If high computational resources are available, a larger number of trees and different kinds of forests can also be added to improve results. Feature selection could be applied to datasets as a pre-processing step to improve results. The final stage of XGBoost classifier's parameters can be tuned further using deep grid search hyper-parameter tuning to get even higher performance.

3.7 Conclusion

With the evergrowing market of online advertising, companies are now shifting their focus towards selling their products on websites and mobile applications. As a result of this, the issue of click fraud has grown exponentially in the recent past.

Click fraud is the illegal clicking of advertisements that leads to the wasted funds of the advertisers. To counter this issue, several methods to detect click fraud have been devised. Click fraud detection is used to protect the advertiser by classifying clicks into valid and fraudulent clicks. It is mainly implemented using deep learning models. However, deep learning models require many parameters to be tuned and hence require considerable amounts of time to give good results. To combat this, we propose a machine learning model, CFXGB, a hybrid of the Cascaded Forest and XGBoost which accurately identifies faulty clicks. It uses the Cascaded Forest to transform the features by concatenating the original dataset's predicted class vectors to it, and re-predicting the class vectors iteratively. In the last layer, the original dataset is concatenated to the final class vectors and fed into the XGBoost classifier for click fraud prediction. Apart from click fraud datasets, we have also considered intrusion detection datasets to validate the claims of CFXGB performing better than deep learning models. The use of the Cascaded Forest as a feature transformer and then XGBoost as a classifier, has shown a considerable advantage over merely using the Cascaded Forest as a classifier. Several experiments were conducted on different datasets to find the best parameter values. They were initialized based on the experimental results and were the same for all the datasets. On performing comparative analysis, using various click fraud detection models, we infer that CFXGB performs outstandingly well.



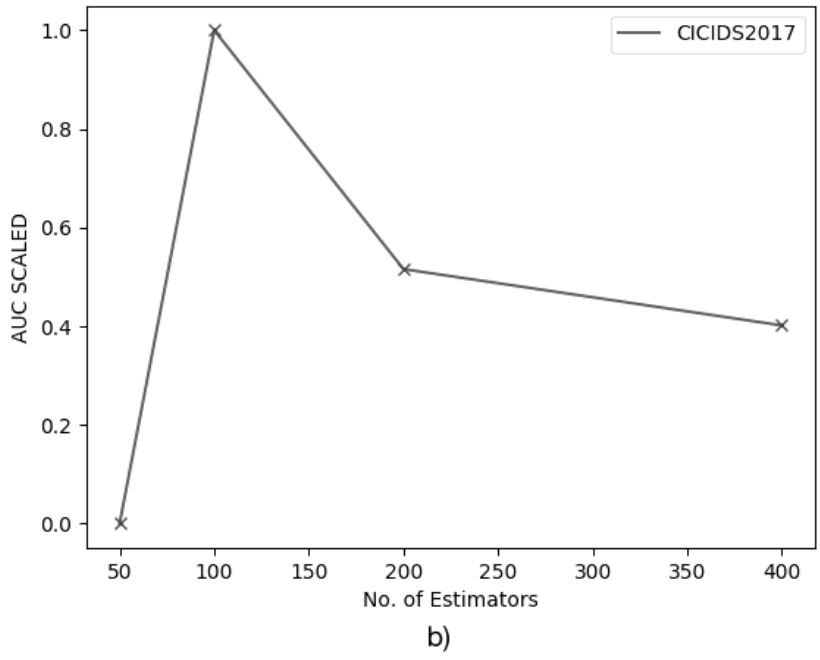
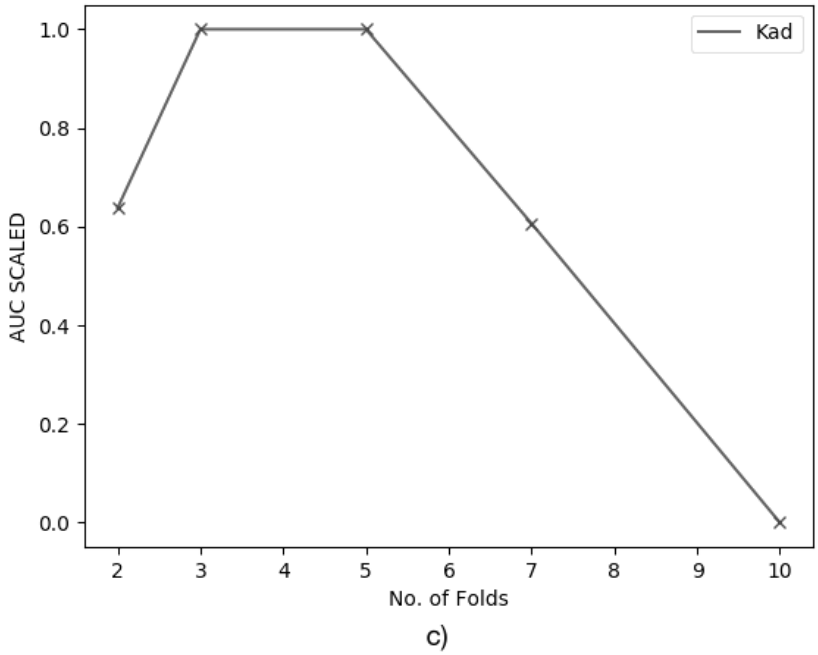


Figure 3.3: Peak Values of Parameters in the Model. **a)** Early Stopping Rounds has Peak Performance at Value 3. **b)** No. of Estimators has Peak Performance at Value 100.



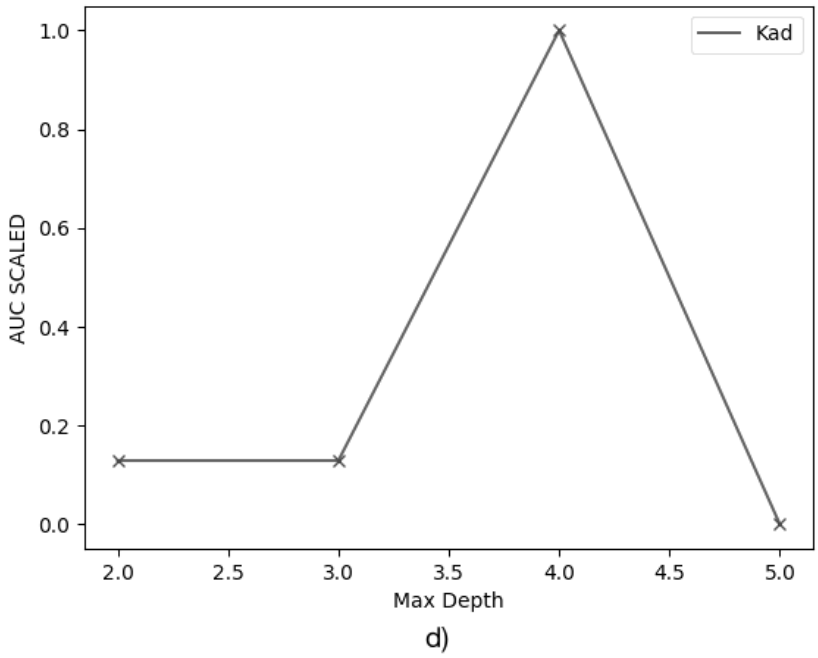


Figure 3.3: (Continued) Peak Values of Parameters in the Model. **c)** No. of Folds has Peak Performance at Value 5. **d)** Maximum Depth has Peak Performance at Value 4.

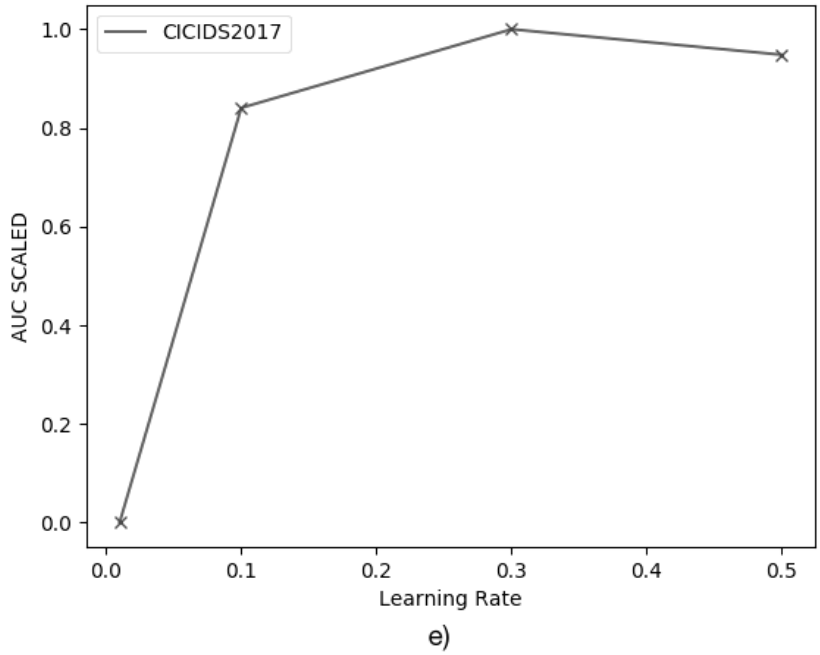
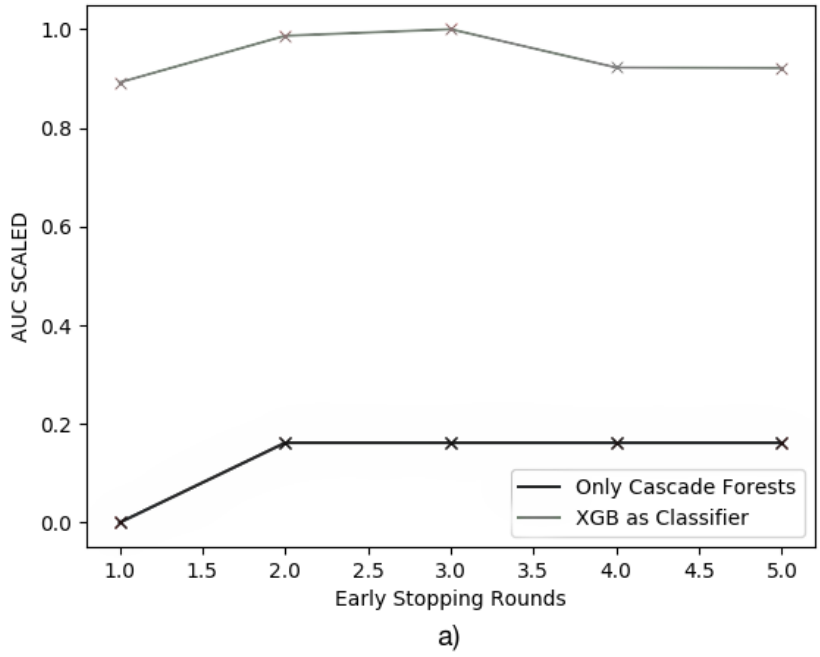


Figure 3.3: (Continued) Peak Values of Parameters in the Model. **e)** Learning Rate has Peak Performance at Value 0.3.



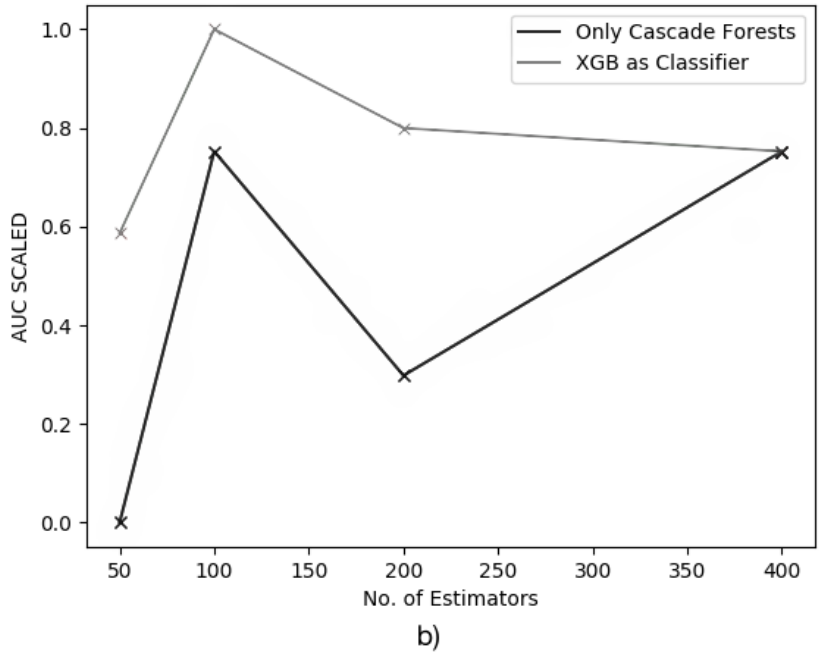
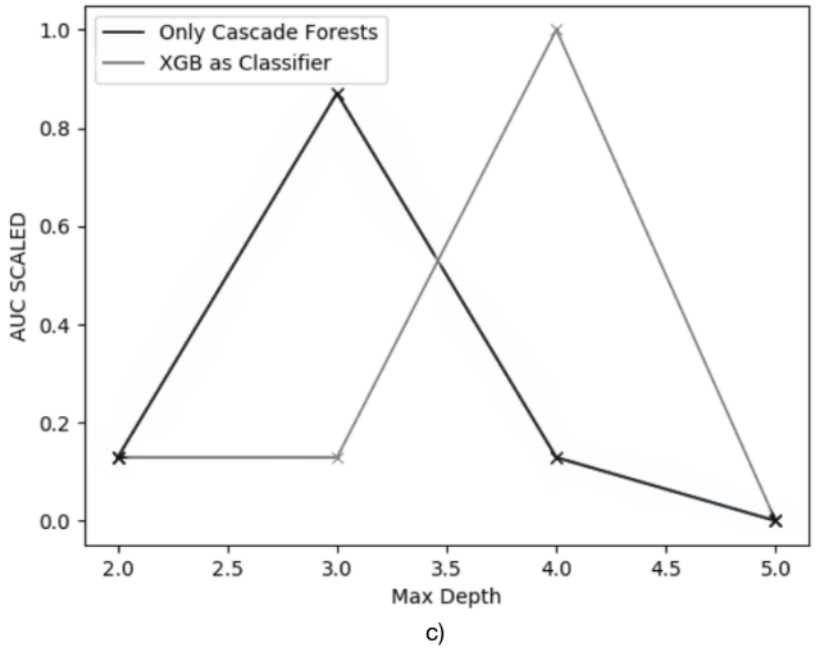
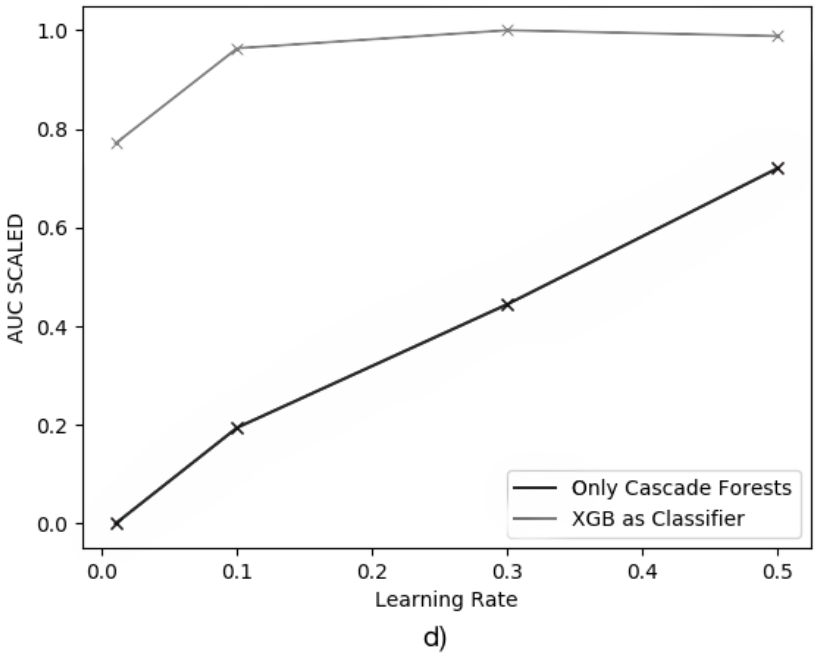


Figure 3.4: Comparison of CFXGB vs. Cascade Forest Classifier **a)** Early Stopping Rounds Against AUC **b)** No. of Estimators Against AUC.



c)



d)

Figure 3.4: (Continued) Comparison of CFXGB vs. Cascade Forest Classifier **c)** Maximum Depth in XGBoost Classifier Against AUC **d)** Learning Rate in XGBoost Classifier Against AUC.

CHAPTER 4

KSMOTE: AN EXTENSION OF SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE FOR IMBALANCED DATASETS

More often than not, data collected in real-time tends to be imbalanced i.e., the samples belonging to a particular class are significantly more than the others. This degrades the performance of the predictor. One of the most notable algorithms to handle such an imbalance in the dataset by fabricating synthetic data, is the "Synthetic Minority Oversampling Technique (SMOTE)". However, data imbalance is not solely responsible for the poor performance of the classifier. Certain research works have demonstrated that noisy samples can have a significant role in miss-classifying the dataset. Also, handling large data is computationally expensive. Hence, data reduction is imperative. In this work, we put forth a novel extension of SMOTE by integrating it with the Kalman filter. The proposed method, KSMOTE, filters out the noisy samples both, in the raw data and the synthetically generated samples, thereby reducing the size of the dataset. Our model is validated with a wide range of datasets. An experimental analysis of the results shows that our model outperforms the presently available techniques.

4.1 Introduction

Data is categorized into different classes, where a few of them might have an excessive number of samples, leading to an imbalance. Since the data has a minority class, the result of the predictor tends to be biased. This becomes a detriment to the performance of the learning model. Most of the real-time datasets associated with medicine [TDK10], text classification [LLS09], intrusion detection [KTPA12], and click fraud detection [FP97] are not balanced. A binary dataset having 95%

positive samples may obtain an extremely high classification accuracy. However, this accuracy may be incorrect due to over-fitting.

Extensive research has been performed in the recent past to formulate a solution for the issue of handling imbalanced data, and several solutions have also been suggested[HG08]. Several re-sampling techniques [BSL11, CBHK02, KM$^+$97, SW08] are available to balance the dataset, among which SMOTE[CBHK02] is a widely recognized technique. In the previous years, 85 versions of SMOTE have been proposed. We provide a thorough review of all the variants in Section 8.2.

Past research shows that an imbalance in the class samples is not the only concern, as other factors such as noise and borderline samples may hinder the performance of the learning algorithm [GSM07, Jap03, NSW10]. Applying SMOTE on imbalanced datasets gives better results, but it generates synthetic samples, resulting in a notable increase in the size of the data. Presently, the data collected in real-time is extremely large (BIG DATA) [ZE$^+$11]. SMOTE can be considerably improved by performing certain modifications (Borderline-SMOTE 1,2 [HWM05], Safe-level SMOTE [BSL09]) or by adding some extensions (SMOTE-IPF [SLSH15], ENN or TL[BPM04]).

Filters have not been commonly used in combination with SMOTE until now. Hence, we propose an extension called KSMOTE which employs the Kalman Filter [BW$^+$01] to remove noisy data samples. The use of the Kalman filter as a data-reduction method improves the efficacy of the classifier and reduces the processing overhead.

We use three Click-Fraud datasets, an Intrusion Detection dataset and a few UCI[BM98] datasets that are considered by previous researchers, to evaluate our work. Furthermore, we make comparisons with several, existing variants of SMOTE. Metrics such as Recall (Rcl), Accuracy (Acry), F1-Score (F1scr) and Precision (Pres)

have been used to provide comparisons. Area Under the Curve (AUC) is another very good metric to verify overfitting caused by SMOTE.[Bra97].

4.1.1 Organization of the Chapter

In Section 8.2, we discuss the related works in terms of the research we have carried out on SMOTE. In Section 4.3, we demonstrate the preliminary concepts behind this work, and our novel, proposed approach followed by an in-depth explanation of our concept. In the 4.4th Section, we discuss the experimental setup and the details of the datasets we have used. We also present the evaluation metrics used in our work, analysis methodology and discuss our results that improve the state-of-art methods concerning various metrics. In the 4.5th Section, our work is concluded.

4.2 Related Work

To overcome the imbalance, various resampling techniques have been presented[BSL11, CBHK02, KM$^+$97, SW08]. SMOTE was proposed by Chawla in the year 2002[CBHK02], which makes use of KNN graphs to generate synthetic data. In[BPM04], the author proposes to extend SMOTE by integrating it with ENN and TL noise filters. In[HWM05], only the borderline samples are considered and are oversampled. In[CHS$^+$06], authors have used different Prototype-based resampling methods like KNN and Support Vector Machine (SVM) to balance the dataset. In[WXWZ06], the authors have applied the LLE algorithm to process the dataset and then oversampled the dataset by using SMOTE.

In[CCS06], the authors use RIPPER as an underlying rule classifier and also a clustering-based method for oversampling is proposed. In [DLCF07], the method proposed averages the neighbors to obtain the mean example to oversample the data,

also the author only considers the positive dataset to locate the nearest occurrences utilizing the weighted distance. In[HBGL08], the authors have proposed to alter the decision boundary in the direction close to the difficult samples by using some techniques. In[DLCFG08], the authors use a collection of classifiers to select the samples from the dataset, and weighted distance is used to balance the dataset. In[GA08], the authors use 4 different topologies to oversample the minority class using a polynomial fitting function. In[TC08], the authors have proposed to readjust the direction of the synthetic minority samples by generating data along the first component axis.

In[SW08], the authors have applied amplification methods and selective preprocessing techniques and compared SMOTE with NCR. Before generating the data samples, positive instance is assigned to the safe level and regenerates the data points (DP) around the line with various weights[BSL09]. In[HLMH09], the authors classify the outnumbered samples into 3 different groups as noise, border and security samples using the distance and then balance the data according to the groups. In[GCZ09], the authors have used Isomap to map the training data, later SMOTE is applied, and the data is reduced by applying NCR method. In[CCCG10], the authors have used differential evolution clustering algorithm along with SVM and SMOTE with SVM, and a hybrid approach is proposed. In[CGC10], the authors generate partitions using k-means and samples are clustered. Later, a threshold is defined and samples with cluster index lesser than the threshold are regenerated.

In[KW10], the authors decide the count of samples to be generated from every data point, and generate samples to balance the dataset. In[CW11], the authors obtain the distribution report and the density report of the DP and balance the data. In[CCV11], the authors have proposed a new under-sampling and oversampling technique to resample the data and balance the dataset so that there is no loss of

data and addition of too many samples. In[FTW11], the authors use a margin-based rule to sample the synthetic data; this process overcomes the over-generalization of data samples. In[RCBH12], Rough-Set-Theory(RST) and SMOTE are used together to handle the imbalance in the dataset. In[MS11], the method considers the more local neighborhood of the minority sample (Considers next k+1 neighbor) and gives better approximations.

In[BIM11], the authors have incorporated unsupervised clustering in the generation of synthetic data. This method ensures that the samples generated always lie inside the minority region and avoid wrong samples. In[DP11], the authors have combined SMOTE with Evolutionary Sampling Technique (EST) to over-sample or under-sample the data. In [DW11], the method randomly generates data points in the minority region unlike SMOTE. In[ZW11], the authors check whether the samples are crossed or not and are grouped accordingly, then new data points are generated based on the different groups. In[FNHMG11], the authors have proposed a 2 stage algorithm. In 1st stage the data is balanced and in 2nd phase different patterns are generated and the data is over sampled. In[FB12], the authors have proposed to pre-process the data by making use of SVM and the data is balanced with the SVM predictions.

In[PW12], the authors remove the data points from the minority region that are not relevant, this will give a precise minority region. In[BSL12], the data is generated along the shortest path and the newly generated samples lie near the centroid. In[SZWQ12], the proposed method dynamically generates different data points around the negative class data point. This will eliminate noise and make the boundary more distinct. Also, smoothing techniques are proposed by the author. According to[BIM13], weights for the negative class samples are found depending on the distance from the positive class which will generate accurately balanced dataset.

In[BS13], the authors propose a tool for selecting a variant of SMOTE, either safe level or borderline; synthetic samples are generated in the safe region determined by a mechanism. In[HL13], the authors propose a 3 step algorithm where at first, positive samples in the lower decision and the negative samples around the boundary is calculated. In the second step, SMOTE is applied on the dataset. Next, the data is balanced and processed.

In[NKOK13], codebooks are obtained from Learning Vector Quantization (LVQ) technique and the data is balanced based on the codebooks obtained. In[SMG13], the authors have proposed a method SYNTHETIC OVERSAMPLING OF INSTANCES (SOI) to resample data inside the clusters. These clusters are from the minority class instances, 2 methods are presented in the paper, SOI by Clustering and Jittering (SOI-CJ), and SOI by Clustering (SOI-C). In[ZYGH13], the authors have proposed a hybrid method combining quasi-linear SVM and assembled SMOTE. In[Kot14], author has showcased 3 different variants of SMOTE; SMOTE-OUT is a strategy to handle very close vectors by creating samples outside the area of the dashed line. SMOTE-COSINE- Euclidian formula and the cosine similarity are consolidated together to obtain the new nearest neighbor (NN). Selected SMOTE- certain attributes are synthesized based on feature selection emphasizing the dimension of significant attributes. In[LZWX13], the negative samples are over sampled using Improved SMOTE (ISMOTE) and the positive samples are under sampled using distance-based under-sampling (DUS) technique. Both the methods are combined to obtain a balanced dataset. In[BIYM14], the proposed method assigns different weights to the samples depending on the Euclidean distance from positive class sample. This weight is then used for balancing the dataset.

In[GHC+14], the authors use kernel density to over sample the dataset and balance it. According to[LTC+14], the method proposed additively generates new

data until an appropriate dataset is obtained. In[ZL14], the raw data will have a probability distribution which is unknown. The newly generated data should also have the same probability distribution, then the data will be accurate and precise. In[AK15], the authors propose a filter approach using the Game Theory (GT). In[LZLF14], to handle the imbalance in the dataset, the boundary samples are selected and resampled. The author says that this will improve the quality of the dataset. In[MMAM14], the authors have proposed DST method that improves the accuracy. Anomalous samples are removed from the negative class. The top three samples are then considered based on a criteria and synthetic data is generated based on these samples.

In[JQL15], the minority class data samples are resampled by finding the similarity between the samples using Minority Cloning Technique (MCT). In[XLLT14], the authors have proposed to combine triangular area sampling and NN with SMOTE and the dataset is balanced. In[RGN14], Gaussian distribution in Q-union is used to resample the data and balance it. In[HHY$^+$14], a supervised method is used to balance the dataset by generating new samples. Also, TargetSOS, a new predictor is proposed by the authors. In[BJD15], modeling efficiency of denoising autoencoders are used to propose a new approach. This will balance the dataset and is an alternative to SMOTE. In[GHE15], Principal Component Analysis (PCA) and multiclass SVM are combined together and a hybrid approach is proposed to sample the data.

In[SLSH15], the authors have presented a filtering method using IPF noise filter to manage the samples at the borderline and the samples that are noisy . In[TH15], kernal density is estimated and the difficulty level is found, based on which the samples are adaptively generated to balance the dataset. In[XJYL15], the authors proposed MOT2LD, that creates clusters by mapping the samples. Weights are assigned based on the importance and the dataset is balanced accordingly. In[YNWC15],

Voronoi diagram is generated and the data points that lie on the border of the 2 classes are found and based on these data points, the dataset is balanced. In[LKL15], the authors generate the samples and then decide whether to keep the sample or not based on the location of the sample. This method will take care of the noisy data and the issue of over fitting. In[DTHS15], the authors change the label of the data samples and then uses SPY method to balance the data.

In[LFZ15], two metaheuristics are combined together to obtain the best value for the parameter. The value of the accuracy depends on the value of the kappa specified by the user. In[RX16], the authors propose to use OUPS that performs oversampling based on the requirement. The probability of group membership is found and the data points are resampled based on propensity rate. In[TCOMT16], the method handles the imbalance by generating data corresponding to every data point. In[BS16], the authors have proposed to balance the dataset by using SMOTE and to handle the overlsampled data by using Rough Set Theory(RST). In[YHL16], the authors have proposed a method to restrict the neighborhood size. The method will determine the value for every minority instance and assures safety for generating synthetic data. In[JLX16], the authors have proposed genetic-algorithm-based-SMOTE (GAST). Optimal sampling rates are estimated and their optimal combination is found. The dataset is then balanced by generating new samples.

In[NLY16], clustering technique is used and each cluster is oversampled based on the Euclidean distance. In[RGL$^+$16], fuzzy rough set theory [DP90] is used as a pre-processing tool. A threshold is then defined and if a sample does not cross it, it will be deleted. In[CGLR$^+$17], the authors use support vector to generate new samples. PSO is used to handle the noise in the dataset.

In[MF17], the authors have proposed to cluster the samples using CURE and then get rid of noise and outliers from the dataset. The dataset is then balanced by

resampling. In[Riv17], the authors propose a method to eliminate the noise prior to the resampling of the dataset.

In[LK17], authors have showcased a method where Gaussian Probability Distribution in the feature space is combined and new data is sampled, diverged from the line. In[KW17], the method is proposed in 2 phases. Firstly, the neighborhoods are cleaned. Secondly, synthetic samples are generated selectively. In[SS17], Adaptive Neighbor Synthetic Minority Oversampling Technique (ANS) is proposed which dynamically adjusts the number of neighbors needed to oversample the minority regions. In[DBL18], k-means has been combined with SMOTE which generates samples in the deficient minority area and the class label is not considered while generating the synthetic data.

4.3 Proposed Approach

4.3.1 Preliminary Concept

SMOTE

If the classes are not proportionally distributed, then the data is said to be imbalanced. Most of the real-time datasets suffer from data imbalance, where normal samples have many more occurances when compared to abnormal samples. The classifiers running on these datasets are generally overfitted or underfitted. There are numerous resampling techniques that have been proposed to handle this. SMOTE is one such algorithm to handle imbalanced data effictively. SMOTE over-samples the data to achieve better results. Negative class samples are resampled to handle the imbalance. Depending on the degree of oversampling that needs to be performed, neighboring data points using the kNN algorithm are chosen. Typically, k

is assigned with 5 to oversample the data. Initially, the distance between a sample and its nearest neighbour is calculated. In the next step, the distance is multiplied with an arbitrary number ranging between 0 and 1 following which, it is added to the sample. An arbitrary point is selected along the line segment between the two specified samples.

Kalman Filter

The Kalman filter[Kal60] was proposed to solve the Wiener problem. Fundamentally, this filter is a union of numerical conditions that gives a proficient solution of the least-squares technique. It is an efficient algorithm that can support the past, future, and the present estimations. It is basically a two stage algorithm. In the first step, estimates of the current state variables are calculated. In the next stage, weighted averages are used to update the estimates. We use the covariance grids T and P to determine noisy data. Consider $g \in P^x$ as the measurements and $a \in P^i$ as the state of the discrete controlled linear system. At time n, the process and measurement equations are shown in Equations 7.1 and 7.2 respectively.

$$a_n = B_n a_{n-1} + c_n f_n + d_n \tag{4.1}$$

$$g_n = y_n a_n + e_n \tag{4.2}$$

Where $f_n \in P^j$ is the control-input model at time n, a_n is the state at n, g_n is the measurement at n. B_n is $i \times i$ matrix relating state at $n-1$ & state at n. c_n is $i \times j$ matrix relating control input at n & state at n. y_n is $x \times i$ matrix relating state and measurement at n. e_n and d_n are measurement noise and process noise respectively with covariance matrix P_n and T_n and with a mean of 0.

Our Approach

Algorithm 1 Kalman filter Application and Obtaining Mean and Covariance for each Row of the Data

Input: Data C
Output: Train data appended with mean and covariance column KalD

1: Split the data C into Train D and Test T randomly with 8:2 ratio.
2: D_res = SMOTE(D) // apply SMOTE algorithm on train data D
3: Apply Kalman filter using pykalman package
4: Obtain the number of columns (NC) (Label (or) output column need not be considered)
5: kf = KalmanFilter(ISM, NDO)
6: msr = D_res
7: kf = kf.em(msr, niter = 5)
8: mean, covar = kf.filter(msr)
9: KalD = Append mean and covar to D_res.
10: **return** $KalD$

Algorithm 2 Calculating Number of Samples(nos) to Remove in each Iteration from each Class

Input: D, D_res, Number of Iterations Q
Output: rem, the nos to be removed in each iteration.

1: B = nos in D
2: A = nos in D_res
3: N = percentage of data increased after SMOTE.
4: $M = \left\lfloor \frac{N}{Q} \right\rfloor$
5: $Y = \frac{M}{100} \times A$
6: $Y = \frac{Y}{number\ of\ classes}$
7: **return** rem

Our proposed model extends SMOTE by incorporating the Kalman filter. Noisy samples may cause the classifier to miss classify the data, hence such samples should be removed. Kalman filter is used to sift through the data and eliminate such noisy samples.

Algorithm 3 Removing Data Samples and Classifying the Result

Input: KalD, rem, Q
Output: Classifier result on the datasets for each iteration

if (most of the rows have the same covariance value.) **then**
  for $j = 1$ to Q **do**

    for $i = 1$ to $numberofclasses$ **do**
      $data_{i+1} = data_i[max\ value\ covar]\ AND\ data_i[label]$
      $data_{i+1} = data_{i+1}[remove\ rem\ random\ rows]$
      $data_j = data_j.append(data_{i+1})$
    end for
    $data_{number\ of\ classes+1} = data_{i+1}[covar! = maxcovar]$
    $data_j = data_j.append(data_{number\ of\ classes+1})$
    $data_{trail} = dropmeanandcovarcolumns$
    Separate X_train having data attributes and y_train having output label from $data_{trail}$
    Apply Random Forest Classifier
  end for
else
  for $j = 1$ to Q **do**

    for $i = 1$ to $numberofclasses$ **do**
      $data_{i+1} = data_i[label]$
      $data_{i+1} = data_{i+1}$ [remove rem samples starting from highest covariance value]
      $data_j = data_j.append(data_{i+1})$
    end for
    $data_{trail} = dropmeanandcovarcolumns$
    Separate X_train having data attributes and y_train having output label from $data_{trail}$
    Apply Random Forest Classifier
  end for
end if

Our proposed approach follows a three stage procedure. Stages 1, 2 and 3 are depicted by Algorithms 1, 2 and 3 respectively. We take dataset 'C' as an input and Q iterations are to be performed to remove the data samples. The noisy samples can be removed in a varying percentage based on the iterations specified by the user. SMOTE is applied on the training dataset, after which, the Kalman filter object is created with $initial\_state\_mean(ISM)$ as 0 and $n\_dim\_obs(NDO)$ as NC and applied on the oversampled data. A part of 'pykalman' package[Duc19] is modified and used for the same purpose. In the Kalman Filter object, we call the Expectation-Maximization (EM) algorithm[Moo96]. The training data is considered as measurements (msr) and given as an input to the filter. In our experiments, we have considered the *niter* to be 5 for the EM algorithm because it avoids overfitting as explained in [Duc19]. This algorithm calculates the maximum likelihood of parameters iteratively. On applying the filter, the mean and covariance values are obtained corresponding to each row of the dataset.

We employ some notations to describe the second stage of the method. B indicates the number of rows before applying SMOTE and A denotes the number of rows after applying SMOTE. N portrays the percentage of increase in the data (synthetic samples generated). The percentage of data to be removed is calculated as shown in Equation 7.3.

$$M = \left\lfloor \frac{N}{Q} \right\rfloor \tag{4.3}$$

$$Y = \frac{M}{100} \times A \tag{4.4}$$

Equation 7.4 gives the count of data samples to be removed in each iteration.

$$Y = \frac{Y}{nos} \tag{4.5}$$

To maintain data balance, Y samples are discarded proportionally from all classes as shown in Equation 7.5. In Stage 3, we obtain a count of all unique values of covariances that were calculated. If a considerably large number of rows have the same covariance, then, depending on the *nos* calculated, the data samples with the highest covariance are dropped from each class equally. The Random Forest [Bre99] classifier is run on the new dataframe. If the covariance is different for most of the rows, then, the data is sorted according to the covariance values. The data samples with the highest covariances are dropped iteratively from every class proportionally. The values in the covariance matrix is the amount of noise present, hence we filter the noise according to the covariance values. After this process, the Random Forest is applied. Results are computed for each iteration and the finest result is considered. The dataset corresponding to that result gives the best performance when used with the classifier. Figure 4.1 depicts the working of KSMOTE.

4.4 Experimental Results and Discussions

4.4.1 Experimental Setup

Here, we describe the datasets, the analysis methodology and the evaluation metrics used in our experiments. Furthermore, an in-depth comparison of the classifier's results is provided. We have validated the experimental outcomes with other contemporary techniques viz., SMOTE, ADASYN, BorderlineSMOTE, SMOTEENN and SMOTETomek. The NN parameter's value is initialised as 5. In each iteration, our model removes data proportionally from all the classes thereby maintaining the class balance. The random forest classifier is employed after which, multiple metrics

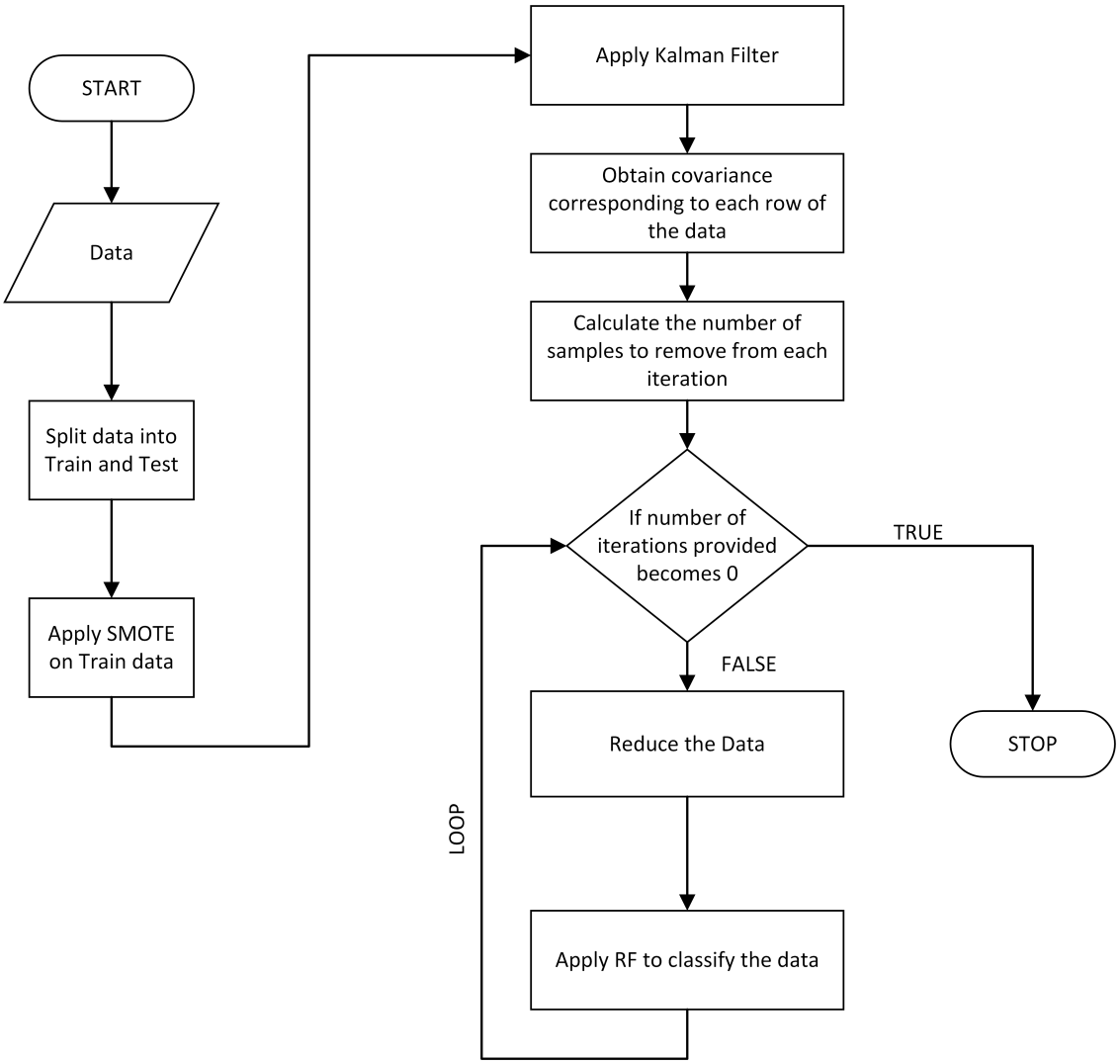


Figure 4.1: The Flowchart Representing our Approach KSMOTE

are computed. The experiments were conducted on a Windows platform with the Intel i7 8 core Processor and a 16GB RAM.

4.4.2 Datasets

Several real-time datasets regarding click-fraud, six UCI benchmark datasets and the UNSW-NB15 Intrusion Detection dataset [uns15] are used. All of these datasets have unproportional class distributions. Under the domain of click fraud, the Talk-

ing Data[Kag18] [TBC+19], the Avazu Click-Through Rate (CTR) [Kag15] and the Criteo datasets [Kag14] have been considered. The Talking Data dataset contains 9 attributes. Data preprocessing was performed by separating the attribute 'click_time' into different four attributes, 'day', 'hour', 'min' and 'sec'. A million random samples were considered from the entire dataset which originally had 184,903,890 entries. The Criteo dataset was randomly sampled prior to usage and all the rows with NaN values were dropped. The attribute 'hour of click' in the Avazu dataset was split into different columns. The data was collected over a period of ten days and is chronologically ordered. We made use of a million random samples for the experiment. Although the Glass dataset is defined by 7 classes, one of the classes has no samples assigned to it. This particular class was dropped before the experiment was taken forward. The New Thyroid dataset has 3 class labels. The dataset was clean and did not require any preprocessing. There are 8 classes in the Ecoli dataset. SMOTE requires a minimum number of samples to execute, however, three out of the eight classes did not cross the minimum threshold. They were dropped in our experiment. The UNSW-NB15 dataset had 2540047 rows before the NaN values were dropped. Four columns denoting the ip adresses and port numbers were dropped to increase the efficiency of the classifier. The string values were converted to numbers with the help of label encoding. Six datasets from the UCI Archive are used. These six datasets are commonly used in multiple papers and are now treated as benchmark datasets. Table 6.2 provides a brief outline of the datasets used in our experiments. It specifies the number of samples and the type of class distribution.

Table 4.1: Dataset Description

Dataset	No. of Samples	Class
Pima	768	Binary
Ecoli	336	Multi-class
Haberman	306	Binary
New Thyroid	215	Multi-class
Hepatitis	115	Binary
Glass	214	Multi-class
Talking Data	1,000,000	Binary
Avazu	1,000,000	Binary
Display Ad. Challenge- Criteo Labs	756,554	Binary
UNSW NB15	2,540,047	Binary

4.4.3 Analysis Methodology

To test the proposed model, we considered the contemporary techniques, namely, SMOTE, SMOTEENN, SMOTETomek, ADASYN and BorderlineSMOTE. To compare the different models, parameters such as Acry, AUC, Pres, Rcl and F1scr were made use of. A mathematical definition of these metrics is presented in the following subsection.

4.4.4 Evaluation Metrics

One of the most popular and commonly used metrics, accuracy, is computed for all the datasets. The positive samples are denoted by pspl whereas the negitive samples are abbreviated to nspl. Also, we make use of the following notations: True Positive as Trupstv, True Negative as Trnt, False Positive as Fspst and False Negative as Fsnt. The accuracy of the predictor is defined in Equation 7.6.

$$Acry = \frac{Trupstv + Trnt}{pspl + nspl} \tag{4.6}$$

The biggest drawback of using accuracy as a metric is that it might be incorrect due to overfitting of the learning model. Overfitting is an undesirable trait that occurs when the algorithm predicts values based on erroneous data. Hence, we have also made use of other metrics to evaluate our work. The AUC is one such parameter which is not greatly affected by overfitting.

We have also used Pres, Rcl and F1scr which are calculated as given in Equations 7.9,7.10 and 7.11.

$$Pres = \frac{Trupstv}{Trupstv + Fspst} \tag{4.7}$$

$$Rcl = \frac{Trupstv}{Trupstv + Fsnt} \tag{4.8}$$

$$F1scr = 2 \times \frac{Pres \times Rcl}{Pres + Rcl} \tag{4.9}$$

4.4.5 Results

Table 4.2 portrays the outcome of running the classifier on the raw binary datasets. #0's indicates the negative samples and # 1's denotes the number of positive samples. Table 4.3 depicts the results of classification on the raw multiclass datasets. We can observe that there is a significant imbalance in the data as the AUC values indicate that the model is overfitted. For the same reason, we balance the datasets before using them.

We conducted the experiments by applying the current SMOTE algorithms on all the datasets, and the results are tabulated in Table 4.4.

Table 4.4 demonstrates a comparison of KSMOTE with the existent methods using Binary datasets. Table 4.5 demonstrates a comparison of KSMOTE with

Table 4.2: Results on Raw Data- Binary classification

Datasets	**#1's**	**#0's**	**Acry**	**AUC**	**Rcl**	**F1scr**	**Pres**
Talking	197411	602589	0.85609	0.72235	0.45765	0.61141	0.92075
Pima	221	399	0.76624	0.7293	0.6	0.64706	0.70213
Haberman	118	56	0.69355	0.55430	0.88637	0.80413	0.73585
Hepatitis	26	94	0.83871	0.6875	0.375	0.54546	1
Avazu	136122	663878	0.83265	0.51087	0.0249	0.04785	0.61522
Criteo	190488	414755	0.71167	0.56449	0.1663	0.26664	0.67224
UNSW	256982	1775055	0.97554	0.934	0.87843	0.90068	0.92409

Table 4.3: Results on Raw Data- Multi class classification

Datasets	**Acry**	**Rcl**	**F1scr**	**Pres**
New Thyroid	0.99	0.98	0.97	0.98
Ecoli	0.90164	0.9	0.89	0.89
Glass	0.74419	0.74	0.75	0.79

existent methods using multiclass datasets. The following observations are made based on the outcomes.

- In 4 datasets, our model has considerably outperformed the other models.
- We make use of AUC as the comparative metric as it is a standard measure and is not affected by overitting.
- The AUC of our model is lesser than ADASYN for the Avazu Dataset, but the difference between accuracy and AUC scores is less significant in our model. Hence, we infer that our model is less overfitted.
- The performance of our model is better for the click fraud datasets and most of the benchmark datasets.
- To summarize, the proposed model shows good results when tested with six datasets. Although the accuracy might decrease, the AUC scores obtained are higher, denoting a better model.

Table 4.4: Results of Re-sampled Data and Comparison of our Model- Binary Class

Datasets	**Method**	**Acry**	**AUC**	**Rcl**	**F1scr**	**Pres**
TalkingData	SMOTE	0.9094	0.88079	0.82404	0.81852	0.81307
	ADASYN	0.9049	0.88118	0.83414	0.81306	0.79303
	blSMOTE	0.90387	0.87781	0.82612	0.80995	0.7944
	SMOTEENN	0.88013	0.86797	0.84384	0.77732	0.72053
	SMOTETOMEK	0.9103	0.88152	0.82442	0.82006	0.81575
	KSMOTE	0.91049	**0.88359**	0.83023	0.82141	0.81277
Pima	**SMOTE**	0.79221	**0.79082**	0.78724	0.69812	0.62712
	ADASYN	0.77273	0.78873	0.82979	0.69027	0.59091
	blSMOTE	0.78572	0.78018	0.76596	0.68572	0.62069
	SMOTEENN	0.74676	0.76407	0.80852	0.66087	0.55883
	SMOTETOMEK	0.77273	0.77083	0.76596	0.6729	0.6
	KSMOTE	0.78572	0.78615	0.78724	0.69159	0.61667
Haberman	SMOTE	0.59678	0.55838	0.75676	0.69136	0.63637
	ADASYN	0.61291	0.58487	0.72973	0.69231	0.65854
	blSMOTE	0.99	0.61244	0.86487	0.75295	0.66667
	SMOTEENN	0.64517	0.62487	0.72973	0.71053	0.69231
	SMOTETOMEK	0.62904	0.59838	0.75676	0.70887	0.66667
	KSMOTE	0.6613	**0.65136**	0.70271	0.71233	0.72223
Hepatitis	SMOTE	0.90323	0.83797	0.75	0.66667	0.6
	ADASYN	0.90323	0.83797	0.75	0.66667	0.6
	blSMOTE	0.83871	0.69445	0.5	0.44445	0.4
	SMOTEENN	0.87097	0.81945	0.75	0.6	0.5
	SMOTETOMEK	0.83871	0.80093	0.75	0.44445	0.42858
	KSMOTE	0.87097	**0.87097**	0.75	0.6	0.5
Avazu	SMOTE	0.7853	0.56224	0.22478	0.26195	0.31384
	ADASYN	0.78313	0.56551	0.23629	0.26972	0.31418
	blSMOTE	0.78739	0.56005	0.21611	0.25627	0.31476
	SMOTEENN	0.78987	0.55746	0.20585	0.24929	0.31599
	SMOTETOMEK	0.7859	0.56288	0.22546	0.26308	0.31575
	KSMOTE	**0.77993**	**0.56483**	0.23983	0.26926	0.30693
Criteo	SMOTE	0.64626	0.62729	0.57605	0.50636	0.45171
	ADASYN	0.63509	0.62353	0.59229	0.50554	0.44096
	blSMOTE	0.63968	0.62313	0.57842	0.50278	0.44464
	SMOTEENN	0.54036	0.60692	0.78675	0.51881	0.38701
	SMOTETOMEK	0.64509	0.62737	0.57949	0.50703	0.45067
	KSMOTE	0.64634	**0.62797**	0.57834	0.50741	0.45197
UNSW NB15	SMOTE	0.98768	0.99293	0.99994	0.95359	0.91135
	ADASYN	0.98786	**0.99301**	0.9999	0.95423	0.91256
	blSMOTE	0.98781	0.993	0.99994	0.95403	0.91215
	SMOTEENN	0.98036	0.99042	0.99675	0.95195	0.91101
	SMOTETOMEK	0.98144	0.99342	0.99975	0.95336	0.91110
	KSMOTE	0.98764	0.99293	0.99994	0.95335	0.91086

Table 4.5: Results on Re-sampled Data and Comparison of our Model- Multi Class

Datasets	Method	Acry	Rcl	F1scr	Pres
New Thyroid	SMOTE	0.97675	0.98	0.98	0.98
	ADASYN	0.97675	0.98	0.98	0.98
	blSMOTE	0.97675	0.98	0.98	0.98
	SMOTEENN	**0.99**	0.99	0.99	0.99
	SMOTETOMEK	0.97675	0.98	0.98	0.98
	KSMOTE	0.97675	0.98	0.98	0.98
Ecoli	SMOTE	0.90164	0.9	0.98	0.98
	ADASYN	0.90164	0.93	0.98	0.98
	blSMOTE	0.90164	0.9	0.98	0.98
	SMOTEENN	0.90210	0.94	0.94	0.94
	SMOTETOMEK	0.91375	0.95	0.96	0.98
	KSMOTE	**0.95082**	0.95	0.95	0.95
Glass	SMOTE	0.60466	0.6	0.62	0.68
	ADASYN	0.62675	0.62	0.65	0.69
	blSMOTE	0.62791	0.63	0.64	0.68
	SMOTEENN	0.62791	0.63	0.63	0.66
	SMOTETOMEK	**0.62791**	0.63	0.64	0.7
	KSMOTE	0.60466	0.6	0.62	0.68

A plot of the various DPs is presented in Figure 4.2a. The DPs are scattered and a plot of the decision boundary is also portrayed. In Figure 4.2b, we show the plot for SMOTETomek. The DPs are filtered and few of the DP are removed. The plot of the decision boundary seems to have changed significantly. In Figure 4.2c, we have plotted the points after applying KSMOTE and we can infer that the DPs are filtered. The model performs better on removal of these points, thereby proving these points to be noisy. In this plot, the decision boundary has changed, but only by a small margin.

We can observe the plot of the DPs as shown in Figure 4.3a. The DP are plotted for the SMOTE algorithm. We can also observe the plot of the decision boundary. Figure 4.3b portrays the plot for SMOTEENN, where the DPs are filtered. A significant reduction in the number of datapoints is also observed along with the

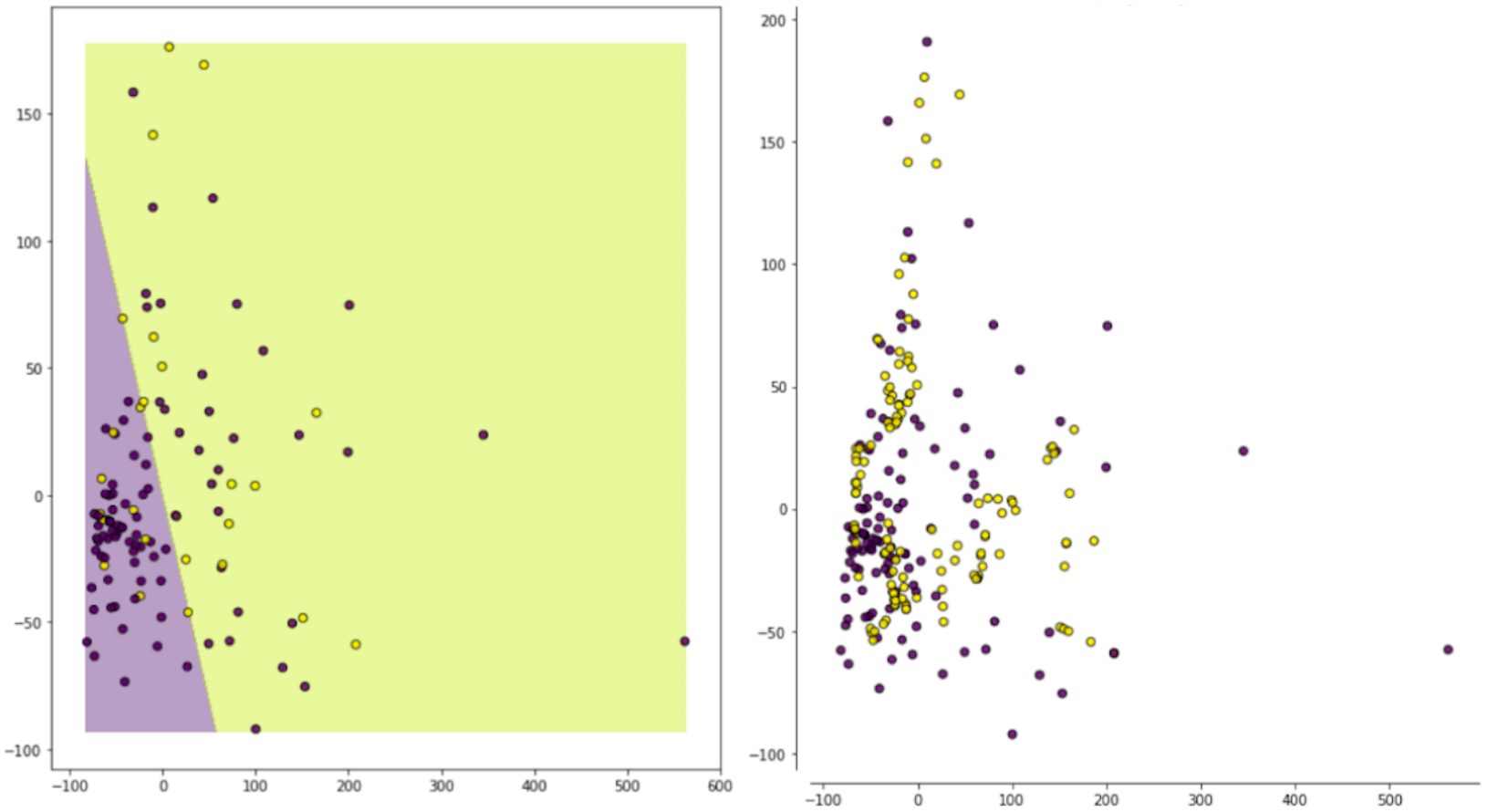
drastic change in the decision boundary. Figure 4.3c represents our results where the data is filtered but not reduced as much when compared to SMOTEENN. The datapoints that were discarded were noisy. In this plot, a change in the decision boundary is observed, which is quite similar to that of SMOTE and SMOTEENN.

4.5 Conclusion

The removal of noisy samples is the main focus of our research. This reduces the computational overhead by reducing the size of the data and increases the efficacy of the classifier. We have proposed an extension of SMOTE by integrating it with the Kalman filter. By incorporating the Kalman filter, KSMOTE can filter the noisy data effectively by dropping the erroneous samples in the original and fabricated data. The Kalman Filter made use of EM algorithm, to which we set *niter* value to 5 as it prevents overfitting. Based on the number of iterations specified by the user, we calculated the number of samples to be removed from each of the class dynamically, which will maintain the balance in the dataset. The data samples are removed depending on the covariance values. If the number of iterations are very small, then a large number of samples will be deleted at the same time, which may result in the removal of the required data. We may not be able to get an optimized dataset. Conversely, if the number of iterations are very large, the classifier is run many times and it is an unwanted time overhead. We considered 5 iterations as optimal as we can obtain better results and there is no time overhead.

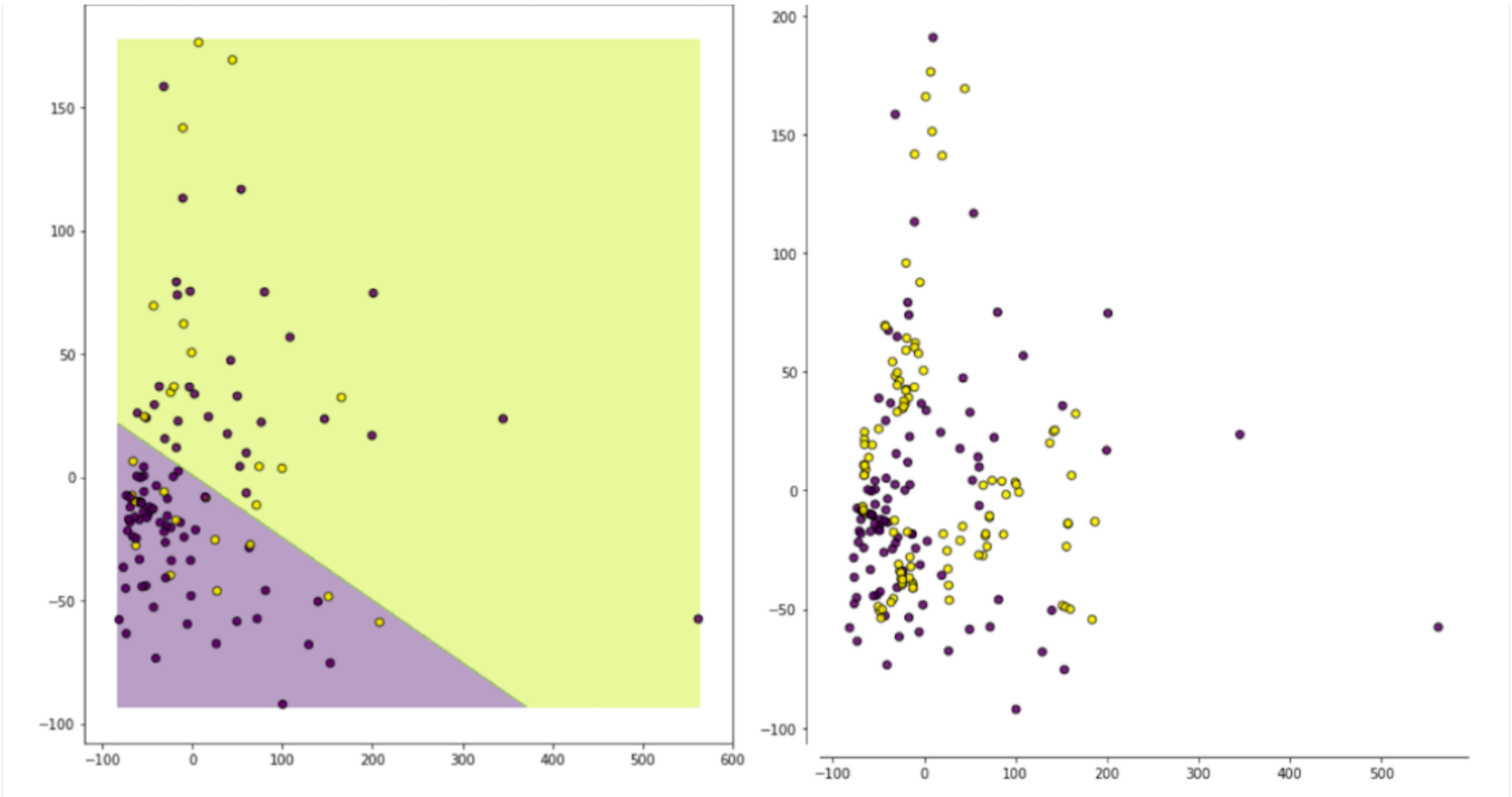
A wide range of real-time binary and multiclass datasets associated with different fields are considered. Multiple evaluation techniques have been employed to compare our models with various oversampling techniques like SMOTE, ADASYN, BordelineSMOTE, SMOTETOMEK and SMOTEENN. We have achieved notable results

compared to the other models. The AUC score is primarily given importance to, for the comparison of the models. For the future work, Kalman Filter from pykalman can be researched to improvise the running time as the time complexity is in terms of cubes.



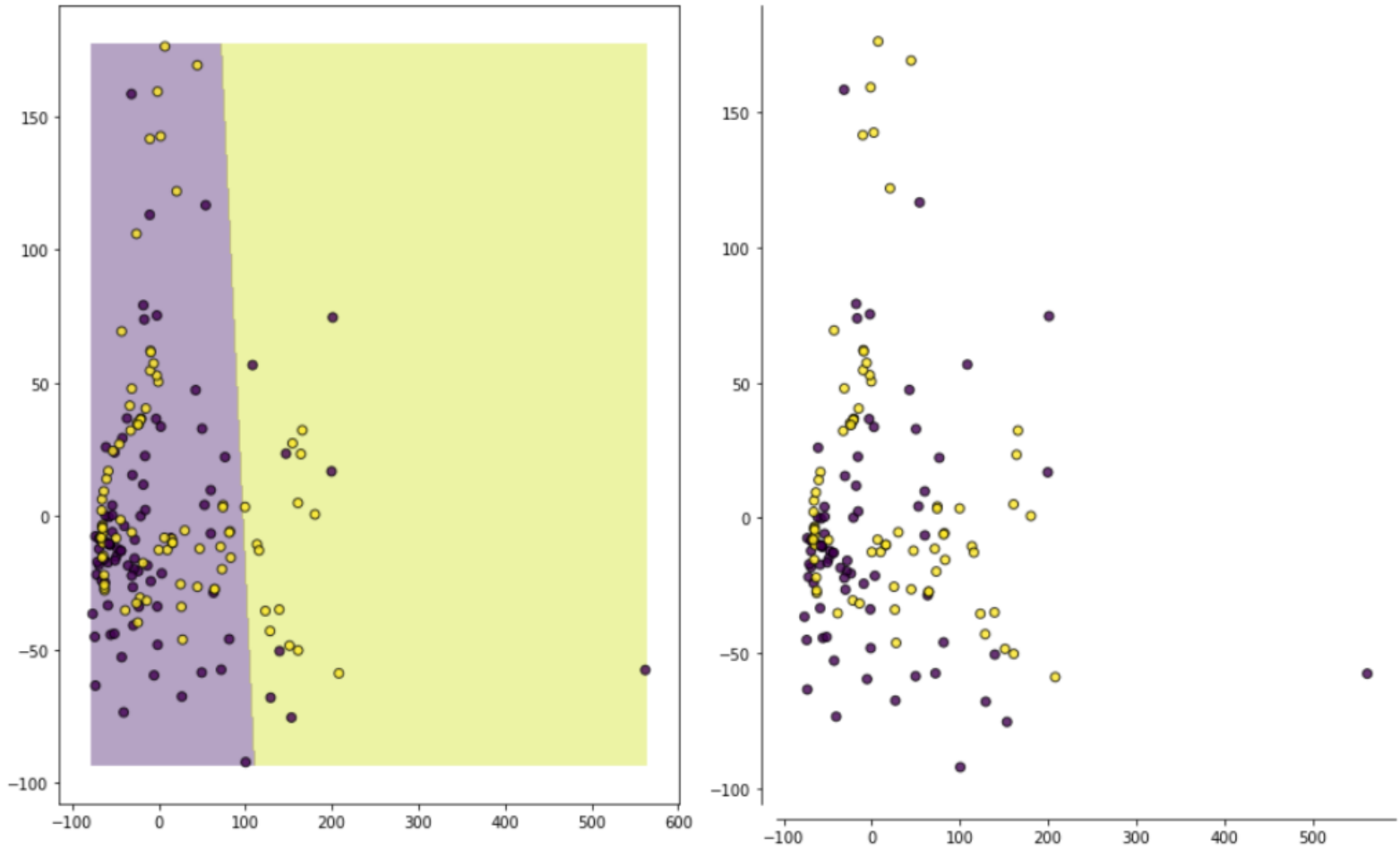
(a) SMOTE Plot on Hepatitis Dataset

Figure 4.2: Decision Boundary and DP Plot for Hepatitis data, (a) Represents the Decision Boundary and DP Scatter for SMOTE.



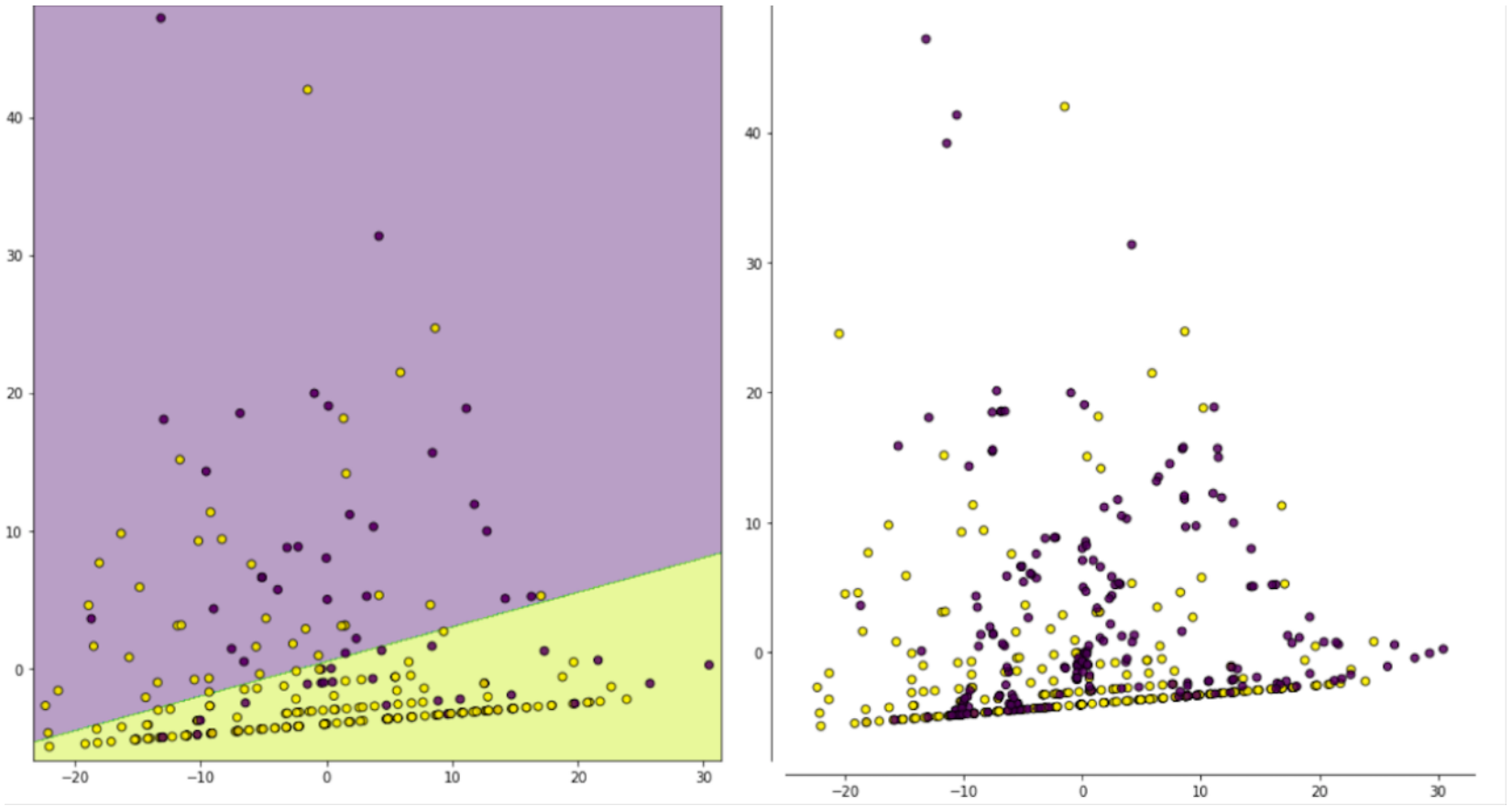
(b) SMOTETOMEK Plot on Hepatitis Dataset

Figure 4.2: (Continued) Decision Boundary and DP Plot for Hepatitis Data, (b) Represents the Decision Boundary and DP Scatter for SMOTETOMEK.



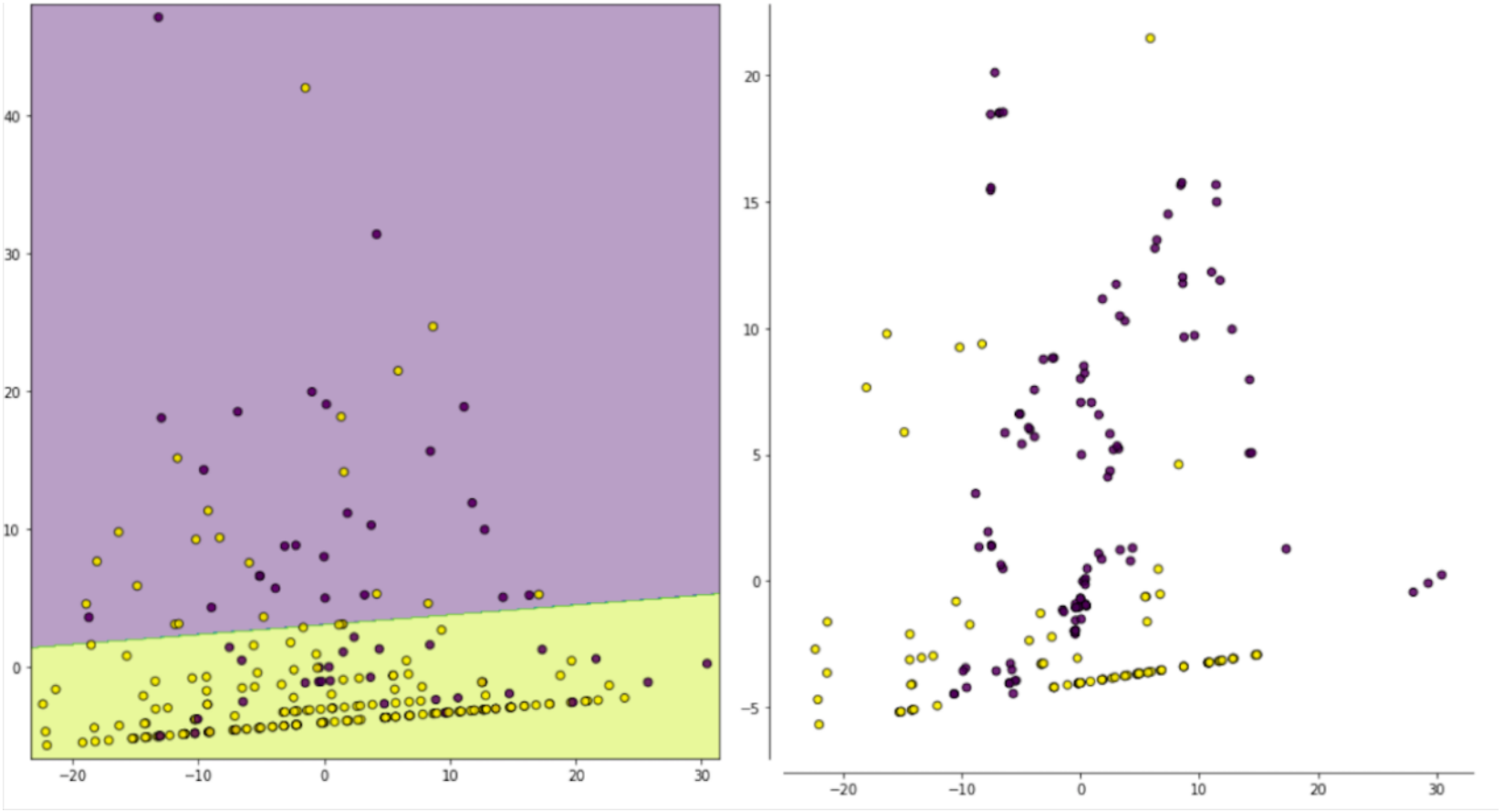
(c) KSMOTE Plot on Hepatitis Dataset

Figure 4.2: (Continued) Decision Boundary and DP Plot for Hepatitis Data, (c) Represents the Decision Boundary and DP Scatter for KSMOTE.



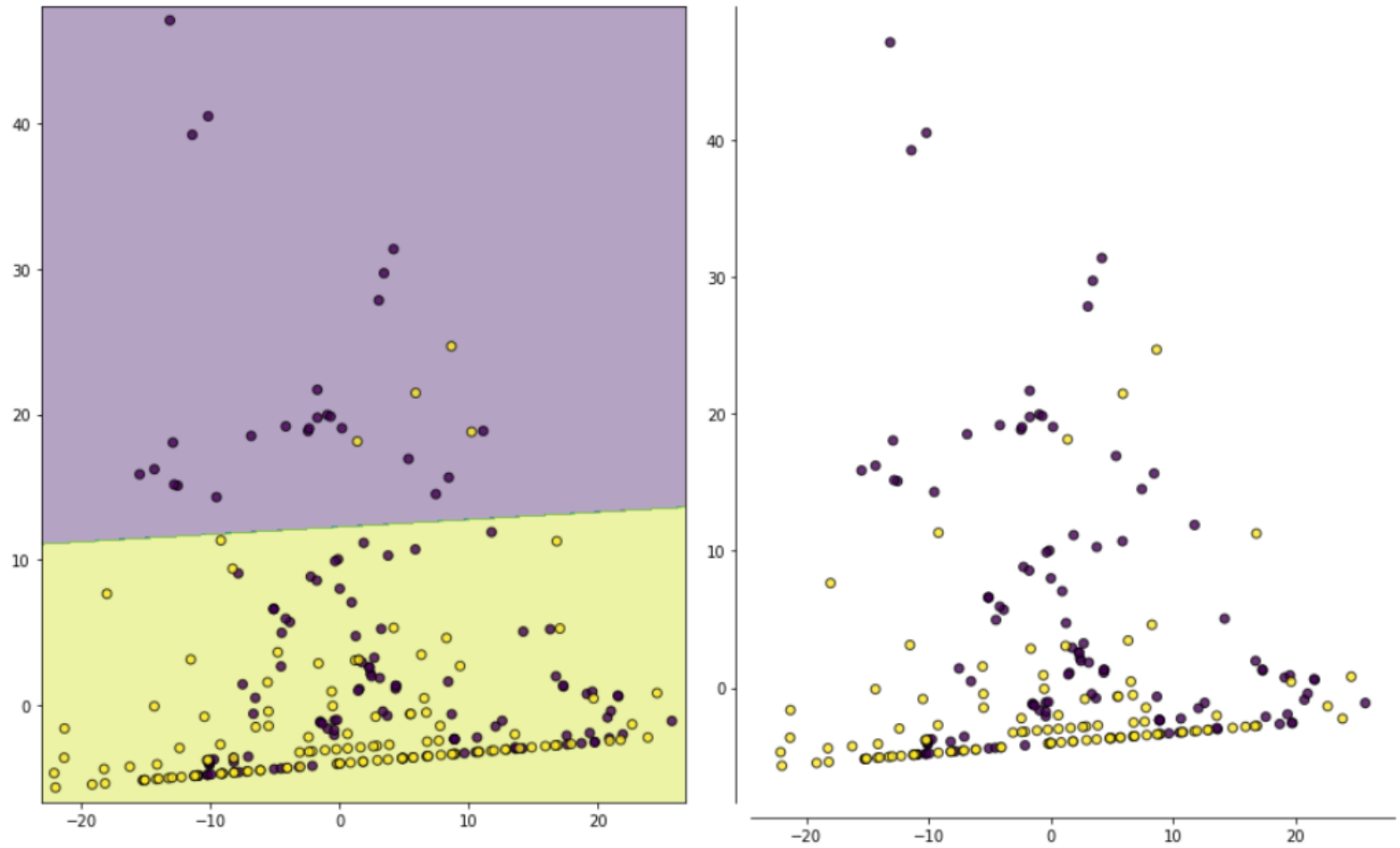
(a) SMOTE Plot on Haberman Dataset

Figure 4.3: Decision Boundary and DP Plot for Haberman Data, (a) Represents the Decision Boundary and DP Scatter for SMOTE.



(b) SMOTEENN Plot on Haberman Dataset

Figure 4.3: (Continued) Decision Boundary and DP Plot for Haberman Data, (b) Represents the Decision Boundary and DP Scatter for SMOTEENN.



(c) KSMOTE Plot on Haberman Dataset

Figure 4.3: (Continued) Decision Boundary and DP Plot for Haberman Data, (c) Represents the Decision Boundary and DP Scatter for KSMOTE.

CHAPTER 5

A MULTI-TIME-SCALE TIME SERIES ANALYSIS FOR CLICK FRAUD FORECASTING USING BINARY LABELED IMBALANCED DATASET

Click fraud refers to the practice of generating random clicks on a link in order to extract illegitimate revenue from the advertisers. We present a novel generalized model for modeling temporal click fraud data in the form of probability or learning based anomaly detection and time series modeling with time scales like minutes and hours. The proposed approach consists of seven stages: Pre-processing, data smoothing, fraudulent pattern identification, homogenizing variance, normalizing auto-correlation, developing the Auto-regression (AR) and Moving Average (MA) models and fine tuning along with evaluation of the models. The objective of the proposed work is to first, model multi-time-scale time series data on AR/MA by relying only on time and the label without the need of too many attributes and secondly, to model different time scales separately on AR and MA models. Then, we evaluate the models by tuning forecasting errors and also by minimizing Akaike Information Criteria (AIC) and Bayesian Information Criteria (BIC) to obtain a best fit model for all time scale data. Through our experiments we also demonstrated that the Probability based model approach is better as compared to the Learning based probabilistic estimator model.

5.1 Introduction

Fraudulent behavior forecasting becomes significant in the modern era, just because of the tremendous amount of data available in order to find out the odd observation. In our case, the goal is to find out the probability of a click being fraudulent. This can be done by checking the pattern in which these ads are clicked, so as to validate the legitimacy of the taps/clicks and an informative block of data can be provided to the organizations. These organizations can then manage complex scenarios by having better insights through which they can tweak their existing algorithms to prepare their system better to tackle new and modern attacks in this domain. This forecasting of fraudulent behavior can help the ad network and the advertisers in optimizing their businesses and efficiently managing their resources. For an ad network, knowing the number of valid clicks on the ads in advance would help them manage their resources efficiently. On the other hand, crucial knowledge on forecasted demand would help the advertisers to understand the market sentiment on the basis of their product demand and to alter production to meet hiked demand or to avoid extra cost when it dips.

The challenges in fraud behavior forecasting are based on the basic premise of detecting fraudulent clicks effectively. The two significant challenges are detailed as follows : (1) Due to the presence of a broad base of the user group, and approx. 1.8 billion records in the dataset [Kag18, TKC$^+$19], it was a computationally intensive task in terms of scalability to go through a massive chunk of data, and processing it to achieve experimental results. (2) Since the heuristics available to distinguish a fraud click from a valid click are limited, and a botnet could easily impersonate an actual user and click an ad, effectively classifying clicks as genuine or fake is logically complex.

5.1.1 Contribution Summary

We present a generalized model for modeling temporal click fraud data. The proposed model consists of four stages: Pre-analysis and pre-processing, Probabilistic/learning-based data smoothing, fraudulent pattern identification, and time-series model fitting. The objective of proposed work are: firstly, model multi-time-scale time series data on AR/MA with only relying on time and the label without the need of too many attributes. Secondly, to model different time scales separately on AR and MA models. Then, we evaluate the models by tuning forecasting errors and also with minimizing AIC and BIC to obtain a best fit model for all time scale data. Choosing AIC or BIC as a criterion mainly depends on our requirement where AIC is chosen to select more efficient model in terms of accuracy and small forecasting errors and on other hand is BIC, if we want to select a model that fits for different training data without becoming progressively worse in terms of forecasting performance. The summary of our contributions in this work are as follows:

1. Extension of Box-Jenkins [NB13, BJRL15] and Boroojeni et al. methodology [BAB+17] for modelling of ad clicking activity forecasting to show the future possible fraudulent behavior.
2. In our proposed approach, we model multi-time-scale seasonality to forecast the fraudulent behavior in terms of minutes and hours interval.
3. Our proposed approach can be considered as an extension of Seasonal Autoregressive Integrated Moving (SARIMA) model [Tay10].
4. AIC, BIC, and residual errors are used to fine tune the model.

5.1.2 Organization of the Chapter

Section 5.2 discusses the related works and reviews the literature of cybersecurity countermeasures for fraud clicks. In section 5.3, we discuss the preliminary concepts behind this work. In section 5.4 we propose our approach and explain each and every component of the approach in detail along with a discussion on experimental results, and finally in Section 5.5 we conclude our work.

5.2 Related Work

It has been observed that the advent of competition in the costs set by ad-network is directly proportional to the increase in revenues. This also suggests that the stability of prices at the equilibrium level depends on the quality of the stability of ad networks [DM14]. There has been a constant increase in the frauds that are targeted against mobile ads over the past decade, hitting the Mobile ad industry by huge costs, which are of the order of billions of dollars, as per a 2013 report [LNGL14]. In [MP11], various statistical models were evaluated to identify Internet Protocol (IP) addresses involved in fraudulent clicks. In [KKL$^+$14], statistical models were used to identify and flag an activity on a Domain Network Server based on recurring patterns in an interval. In [CSC14], Feature extraction is done for an app on Android platform to perform Machine Learning to identify suspicious ads. In [DGZ12], the probability of click fraud is determined based on Bayesian calculations which are used to set a baseline to identify genuine users. In [KNB08], reverse ad technique is applied to separate bot clicks from human clicks. In [PDG$^+$14], the research addresses the problem of zero-access malware by training a learning model which distinguishes suspicious and non-suspicious IPs. In [CMP10], non-dynamic wavelets of data were analyzed using Time domain Analysis to gauge the pattern of click

frauds. In [KJ97], wrapper approach has been suggested to identify best features for a significant improvement in accuracy of the click fraud detection model. In [KZRM13], a simulation of 8 botnets was done to distinguish between bot and human clicks using machine-induced decision tree.

In [TZX+15], focus was given on revealing crowd frauds in internet advertising using crowd fraud features. [LZL+14, VVERA+16] incorporated the concept of bi-partite graph propagation to automate the process of identifying search engine based click frauds, while [HLD17] incorporated the same technique in mobile ad fraud. A graph-based automated mechanism that reflects fraudulent telephone numbers was proposed by the author with the aid of HITS principle in [TYH+15]. In [AB79], Gaussian distributions was selected as a parameter to detect click frauds. In [Bye98], Poisson mixture model is used to detect the abnormalities.

By evaluating a plethora of related works we arrive at the conclusion that our work is the very first attempt of a multi-time-scale time series analysis for click fraud forecasting using binary labeled imbalanced dataset.

5.3 Preliminaries

5.3.1 Time Series

Time series are the temporal measurements of a series of activities where a sequence of values are collected on the same variable over time. Here the values in the series are successive data points where each value is paired with a time stamp. Time series data analysis is more popular in fields like signal processing, finance data, stock market data, weather forecasting and power grids where temporal measurements are involved. Time series analysis is one of the statistical techniques for analyzing

time series data to extract statistical observations and other characteristics of data. It is also famous for forecasting future values based on specific previous time-based observations. Forecasting is based on time series-based data modeling with the help of prediction models. Equation 5.6 and 5.7 represents a time series X_t where X_t is a time series were t is time interval at an indexed time T, *data_point* is a value paired with time stamp t in X_t over a time T.

$$X = \{X_t | t \in T\} \tag{5.1}$$

where t is time interval at an indexed time T

$$X_t = \{t, data\_point\} \tag{5.2}$$

where *data_point* is a value paired with time stamp t in X_t over a time T.

5.3.2 ARMA model

AR model is one of the statistical techniques where it represents a type of random process describing stationary behavior like Wide Sense Stationary (WSS). AR model is used to predict the future value where it considers the output variable as a linear function of previously observed values and an error component which is a stochastic non-deterministic term. To predict the output variable, the model makes use of regression analysis where the output variable is represented as a function of previously observed actions. Hence the model equation is shown in the form of Stochastic Difference Equation (equation 5.8) below :

$$X_t = (\sum_{i=0}^{p} \phi_i L^i) X_t \varepsilon_t \tag{5.3}$$

where X_t is a random process in a given time series. A series X_t is said to be WSS when it has a non-varying mean over the time, i.e. $E(X_t) = \mu x$, ϕ_i is is the

coefficient of i_{th} AR term, ε_t is the error term, and L^i specifies lag value for X_t such that $X_t -> X_t - 1$.

MA is another component with AR in ARIMA or ARMA model of time series. It has a more complicated stochastic structure, since it consists of more than one interlocking stochastic difference equation in a random process. The MA model is also used to predict the future values, in which it considers the output variable as a linear function of the current and previously observed values and various stochastic non-deterministic terms. Equation 5.9 represents the MA model where θ_i is the co-efficient of i_{th} MA term, θ_i is the co-efficient of i_{th} MA term

$$X_t = \left(1 + \sum_{i=0}^{q} \theta_i L^i\right) \varepsilon_t \tag{5.4}$$

AR and MA are special cases of the generic model $ARMA(p, q)$ or $ARIMA(p, d, q)$ where p specifies the order of AR, q specifies the order of MA, and d is the differencing or integrated part used to smoothen the non-stationary time series data as $X_t = (1 - L)^d X_t$. In a nutshell, $ARMA(p, q)$ is as shown in equation 5.10. This equation will also be used by ARIMA after smoothening the non-stationary time series data.

$$ARMA(p, q) = X_t = \left(\sum_{i=0}^{p} \phi_i L^i\right) X_t + \left(1 + \sum_{i=0}^{q} \theta_i L^i\right) \varepsilon_t \tag{5.5}$$

ARIMA model is more flexible than a prediction model since $ARIMA(0, 0, 0)$ is interpreted as a zero parameter model which has no dependency between the terms. $ARIMA(1, 0, 0)$ is interpreted as $AR(1)$ process, $ARIMA(0, 1, 0)$ is an integrated one and $ARIMA(0, 0, 1)$ is a $MA(1)$ model.

5.3.3 Imbalanced Dataset with Binary Classification

One of the challenging tasks is to model both the time series model and the supervised learning model with an imbalanced dataset. This incorporates the case when the dataset is highly slanted towards one value. For example, if the output value in the dataset has a binary classification, and if it is highly skewed towards either positive value '1' or a negative value '0', then it is called as an imbalanced dataset. In this case, there is a possibility that the time series or learning models do not perform well. In order to tackle this situation, we propose an approach in pre-processing and data-smoothing section of the actual chapter.

5.4 Proposed Approach

In an ad network, the click dataset is a collection of temporal data on a mobile platform. Here temporal data is a continuous log of click actions performed on certain ads over a duration. In order to plot this kind of time series data as a function of its past values, we assume that there exists a pattern which is recurrent in nature. We assume a pattern, which we model in order to generate a function, to identify the fraudulent behavior of clicks using multi-time-scaled based prediction. In this section, we propose a methodology to identify a best time series-based model that statistically understands the click pattern behavior promptly. The quality of the ad click time series data is judged based on the model's accuracy used to estimate and forecast future fraudulent click behavior in Ad networks. This kind of modeling will help the ad networks to take future security measures against possible click fraud activities in their network. In the proposed approach, we utilize the AIC/BIC as one of the metric to figure out a model that represents the close estimation time series of the observed Ad click data, to evaluate the process of identifying the best model.

Including AIC/BIC, the proposed approach also takes into account certain residual time series errors, called forecasting errors. These errors are used to measure the quality of the best model whose residual time series is non-deterministic and checks whether the model accurately fits the observed data.

In the proposed approach as shown in Figure 5.1, Where X_t is a Time series were t is time interval at an indexed time T, $D^{(1)}_{t(fl)}$ is a Time series dataset with features f like ip, app, os, device, channel and a label $l = \{fraudulent\ or\ not\}$, $D^{(2)}_{t(fl)}$ is a Time series dataset with a feature $f = click\_time$ and a label $l = LR\_P\_val$, $X^{(1)}_{tl}$ is a Time series X_t obtained from Learning based probabilistic estimator, where $t = \{minute\, or\, hour\}$ and $l = LR\_P\_val$, $D^{(2)}_{P(fl)}$ is a Time series dataset with a feature $f = click\_time$ and a label $l = \{P(ip), P(app), P(device), P(channel), P(all)\}$, where P stands for probability of being fraudulent and $P(all)$ is the combined impact of all features to calculate probability of being fraudulent, ACF stands for Auto Co-relation Function, $PACF$ stands for Partial Auto Co-relation Function, $X^{(1)}_{tl}$ is a Time series X_t obtained from Learning based probabilistic estimator, where $t = \{minute\ or\ hour\}$ and $l = LR\_P\_val$, at_i where at is any of the attributes probability of being fraudulent of i_{th} occurrence, $X^{(1)}_{tp}$ is a Time series X_t obtained from Probabilistic-based modeling, where $t = \{minute\ or\ hour\}$ and $p = min\_P$ i.e. min_P is minimum among the ip_P, app_P, os_P, $device_P$, $channel_P$, and all_P, and $X^{(i)}_{tk}$ is a Time series X_t obtained after applying some form of transformation in homogeneity property check and/or in sationarity check, where $k = \{l\ or\ p\}$ and number of transformation applied $i = 1, 2, 3, 4$:

1. We perform pre-analysis on the data and then based on pre-analysis, we categorize the data into 6 datasets, each indexed with click time.
2. Then, we prepare the time series data representing ads clicks into two variants: time series data based on (i) learning approach and (ii) probabilistic modeling.

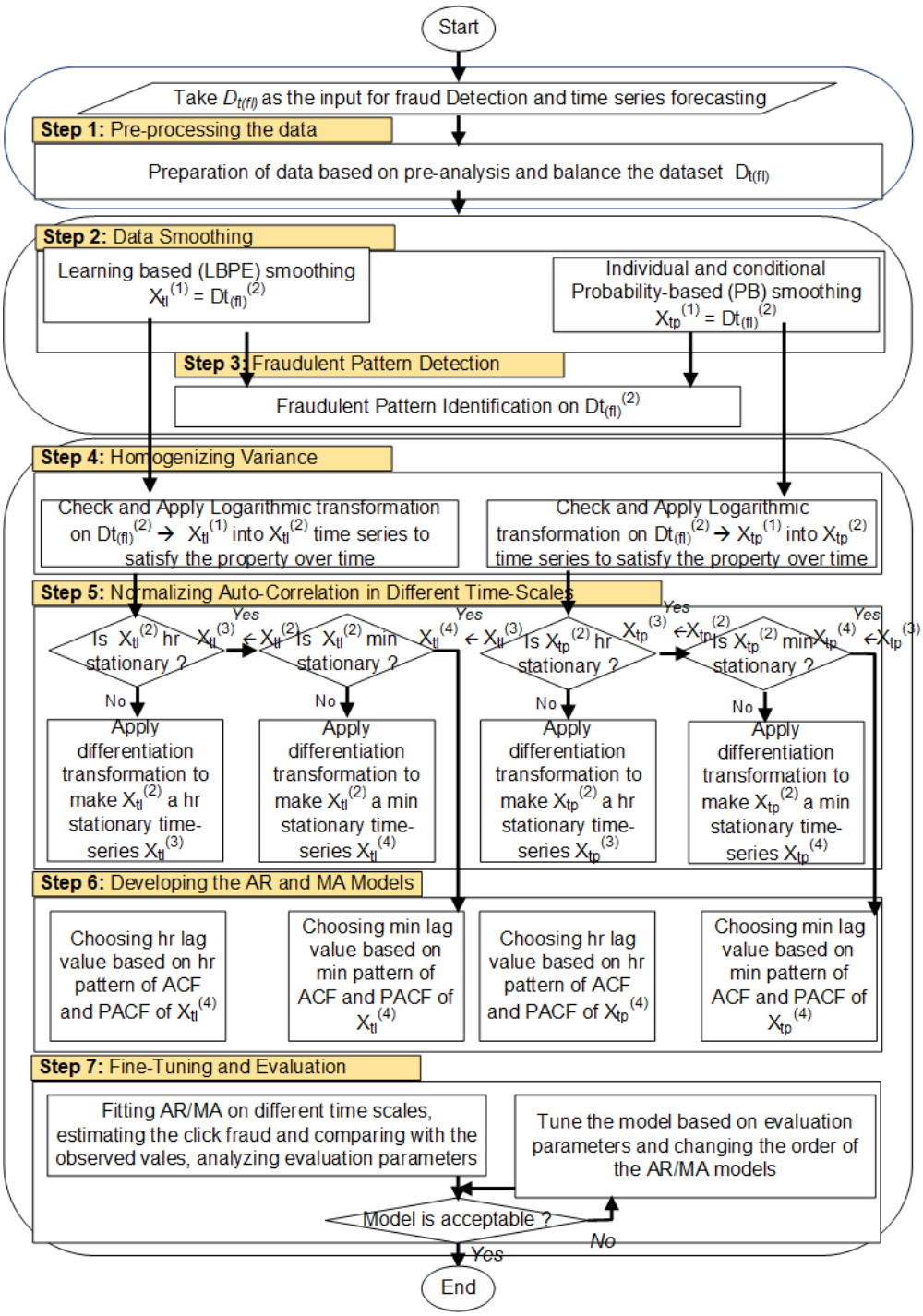


Figure 5.1: The Flowchart Representing the Proposed Approach for Creating a Forecast Model for Forecasting Fraudulent Behavior of Ads Clicks on Multi-time-scale Time Series Data.

In the proposed approach, further steps are performed on these two variants of time series ads click data.

3. Assuming a high probability of positive click behavior (too many download clicks observed with respect to each attribute) we perform fraudulent pattern detection to set a prediction threshold value.

4. To model the time series, we check the homogeneity property of the data to verify the normality and homogeneity of variance. If it does not hold true, then we transform the time series to make its variance homogeneous and to balance the normality.

5. Data Stationary Criteria: (i) We plot the ACF and PACF plots on different time scales to check the stationarity property. Here minutes and hours are the time scales and this step is followed to identify a non-stationary pattern of the time series data. Generally, time scales are just like seasonal cycles, calculated weekly, daily, and annually. Here, we considered minutes and hours as our seasonal cycles, because the time span considered for data collection was set for four days, which is of a shorter order than days or weeks. (ii) In order to cross verify the observations drawn from the ACF and PACF plots, we apply Augmented Dickey-Fuller (ADF) [Ful09] and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) [KPSS92] tests or Rolling Mean (RM) plots and Rolling Standard Deviation (RSTD) plots. These statistical tests are performed to cross validate the stationarity property of time series data. (iii) In order to remove non-stationary data in time series, we apply dereferencing and logarithmic transformation on the data. This step is performed repeatedly until the time series satisfies the tests mentioned above.

6. Next, we model the AR and MA model using the transformed time series data based on ACF, PACF, ADF, KPSS, RM, and RSTD behavior.

7. Finally, the model is selected and evaluated for the forecasting of fraudulent and non-fraudulent click behavior by trying different values of AIC/BIC and minimizing the residual errors.

5.4.1 Pre-processing

Before using the dataset for time series analysis, we need to pre-process the data. The main challenges in the dataset were to tackle imbalanced data and to handle the existence of binary labels. In our experiment, the time series data should consist of a time-indexed label where the label should have certain frequency in the values indexed by time. The accuracy of time series model forecasting also gets affected by the above considerations. For our experiment, we have used real-time dataset provided by Kaggle [Kag18]. The dataset contains 184,903,890 real-time ad click observations collected on a mobile platform. Though, having an extensive and a large dataset acts as an excellent source for information analysis, it's skewed nature towards a particular label would hamper the accuracy. Hence, based on the IP address of click and app id, the dataset is divided into six classes as described in [TKC+19]. Among these 6 classes, we chose one of them for our experiment which consists of 25,974 rows of click observations. For our experiment, we used an Intel i7- 8750H CPU at 2.2 GHz, 6 cores with 32 GB RAM, Jupyter NoteBook Python 3.7. Model train-test-validation ratio was 80:20. (Detailed information about the dataset and pre-analysis results are reported in the previous chapter 2)

5.4.2 Data Smoothing

In this section, we describe further steps taken for the pre-processing of data using smoothing based on the learning and probabilistic models [IR83] to satisfy homogeneity and stationarity properties. Due to the binary nature of output column, we merge various attributes to give us a probability value, which is considered along with the timestamp in order to form a time-probability pair using the 2 methods defined in the subsection. Attributes are eliminated by taking into account their impact value on the label. In remaining sections, we examine time series with learning-based output values and probabilistic-based output values.

Learning Based Probabilistic Estimator (LBPE) Modeling

In this approach, in order to have a time series data in the form of time-indexed label, we apply logistic regression on the actual ads click dataset. We do this to calculate the predicted probability i.e. LR_P_val of binary events occurring based on certain independent variables i.e. $D^{(1)}_{t(f=ip,app,os,device,channel,l=binary)}$ using equation 5.6. To train and test the logistic regression model we select the training and testing data randomly using K-fold cross validation method. The predicted probability values of each row is considered as our label to model time series indexed by *click_time*. The model accuracy was found to be 96%. In order to model the AR/MA model in two time scales, the *click_time* was considered in minutes and seconds, and the two time series data are represented by equation 5.7 and 5.8 below:

$$D^{(2)}_{t(f=click\_time,l=LR\_P\_val)} = LR\left(D^{(1)}_{t(f=\{ip,app,os,device,channel\},l=\{fraudulent|not\})}\right) \quad (5.6)$$

$$X^{(1)}_{t=min,l=l} = D^{(2)}_{t(f=click\_time,l=LR\_P\_val)} \quad (5.7)$$

$$X^{(1)}_{t=hr,l=l} = D^{(2)}_{t(f=click\_time,l=LR\_P\_val)} \quad (5.8)$$

Probabilistic-based (PB) Modeling

Using equation 5.9 and 5.10, we calculate the individual probabilities of each attribute with respect to the probability of being fraudulent.

$$\begin{aligned} &D^{(2)}_{P(f=click\_time,l=\{P(ip),P(app),P(os),P(device),P(channel),P(all)\})} \\ &= P\left(D^{(1)}_{t(f=\{ip,app,os,device,channel\},l=\{fraudulent|not\})}\right) \end{aligned} \tag{5.9}$$

$$P(unique\_ip(i)\ is\ fraudulent) = \frac{total\ fraudulent\ count\ of\ unique\_ip(i)}{total\ (fraudulent\ and\ not)\ count\ of\ unique\_ip(i)} \tag{5.10}$$

The probability of being fraudulent, which includes the combined impact of all attributes against the output label is calculated, i.e. $P(all)$ as shown in equation 5.11.

$$\begin{aligned} P(all) &= P(is\_attributed|ip, app, device, os, channel) \\ &= \frac{P(ip, app, device, os, channel, is\_attributed)}{P(ip, app, device, os, channel)} \end{aligned} \tag{5.11}$$

We chose the final label to be indexed by time using equation 5.12 and 5.13, where we calculate the shortest distance with LR_P_val and at_i where at is any of the attributes probability of being fraudulent of i_{th} occurrence and $at = P(ip), P(app), P(os), P(device), P(channel), P(all)$ and multi-time-scaled series data is obtained from equation 5.14 and 5.15.

$$\begin{aligned} &ip\_P, app\_P, os\_P, device\_P, channel\_P, all\_P \\ &= \left| \frac{\sum_{i=0}^{totalobservations} at_i}{totalobservations} - \frac{\sum_{i=0}^{totalobservations} LR\_P\_val_i}{totalobservations} \right| \end{aligned} \tag{5.12}$$

$$D^{(2)}_{p(f=click\_time,l=min\_P)} = min\left\{ ip\_P, app\_P, os\_P, device\_P, channel\_P, all\_P \right\} \tag{5.13}$$

$$X^{(1)}_{t=min,p=l} = D^{(2)}_{t(f=click\_time,l=min\_P)} \tag{5.14}$$

$$X^{(1)}_{t=hr,p=l} = D^{(2)}_{t(f=click\_time,l=min\_P)} \tag{5.15}$$

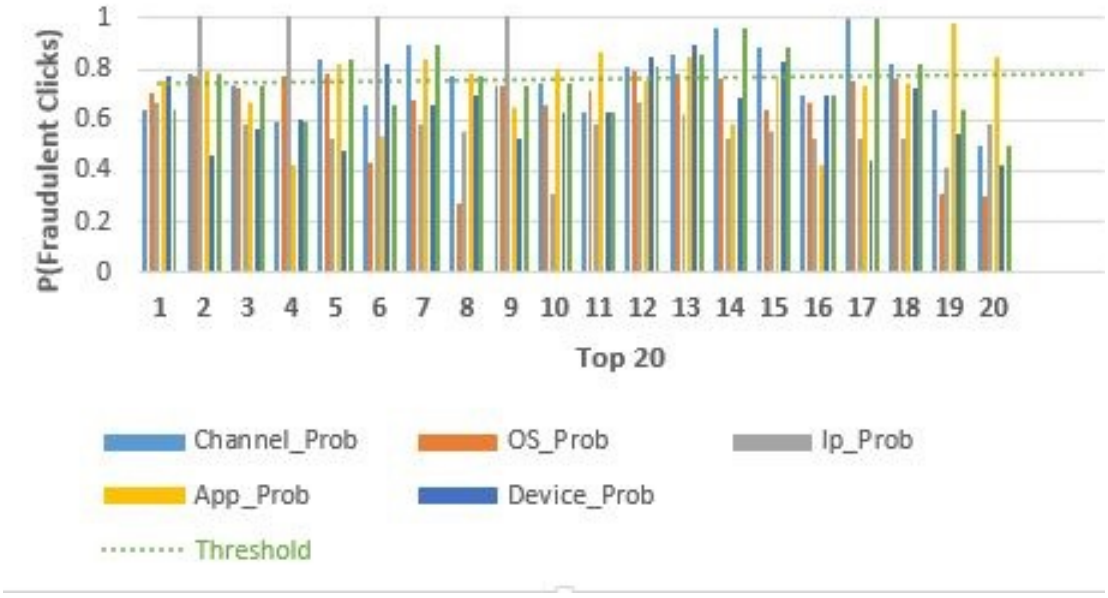


Figure 5.2: Probability of being Fraudulent of Top 20 Ranked all Attribute and Threshold.

5.4.3 Fraudulent Pattern

Figure 5.2 shows the probability of a particular observation being fraudulent on the basis of attributes like channel, os, ip, app, and device. For plotting the graph, top 20 observations were taken into account, ranked on the basis of their click counts. In Figure 5.2, for the top first observation, each color-coded vertical line represents a unique attribute's probability of being fraudulent. Likewise, the remaining 19 observations are represented in the same fashion. Based on the graph, we notice that for most of the top 20 observations, the average probability of being fraudulent is 0.8. Therefore, in our work, we take 0.8 as the standard threshold value above which any given observation is considered as fraud. There is no industry standard to set the threshold, but we are hypothesizing the value based on our dataset and results.

Table 5.1: Homogeneity Property Validation

Box-Cox transformation	**Before**		**After**	
Time series	**NT**	**VT**	**NT**	**VT**
$D^{(2)}_{t(f-click\_time,l=LR\_P\_val)}$	$5.0416*10^{-8}$	$0.192*10^{-5}$	0.0872	0.0631
$D^{(2)}_{p(f-click\_time,l=min\_P)}$	$2.7*10^{-10}$	$0.0132*10^{-7}$	0.0567	0.0598

Note: NT: Normality Test, VT: Variance Test

5.4.4 Homogeneity Property

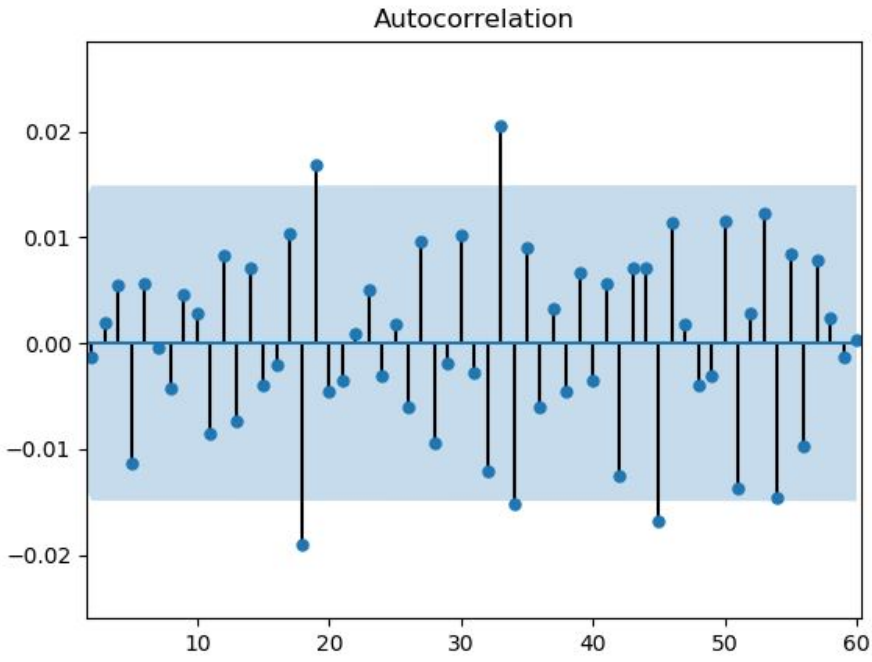
Before initiating work on the AR/MA model, it is important to check whether our data is ready i.e. whether the homogeneity property holds true. If the property is not satisfied, then we need to apply logarithmic transformations on data using Box-Cox transformation method [BC64]. In order to check the homogeneity, we use statistical hypothesis test i.e. T-Test [Man00, Box87, Stu08, Dod08]. The T-Test adds to the table, criteria like normality and variance. To satisfy the T-Test, time series data should have a normal distribution and a non-varying mean. If not, we need to apply the transformation until T-Test criteria are satisfied. To check the normality we use the Shapiro-walk [SW65] and to check the variance we use Levene variance test [BF74], where the results hold the normality and non-varying variance property if the resulting values after the test are greater than the preset probability threshold of 0.05. By looking at table 5.1, we can say homogeneity property holds along with the time series after applying box-cox transformations.

5.4.5 Time Scales and Stationarity Check

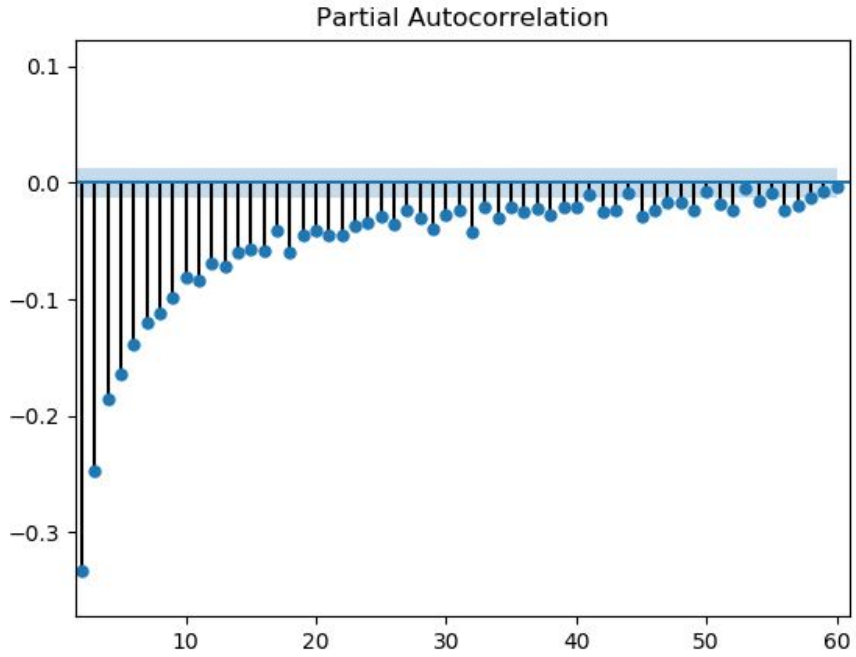
We have considered two-time scales: minutes and hours, for LBPE and PB model time series. In order to check the stationarity property, we use ACF and PACF plots on seasonal data, as it is mandatory for the time series to satisfy the Wide Sense Stationary (WSS) property. We do this to verify stationarity of data. In

ACF, if in the first few lags, the spikes fall off or decline suddenly exactly at or near non-seasonal or seasonal cycles, then it means that the data is stationary. If not, transformation is mandatory to satisfy the stationarity of the time series. Figure 5.3a-5.3h and Table 5.5 shows the ACF and PACF plots and lag values.

An obvious way to cross-validate our drawn observation is to plot RM and RSTD on time series data. If the RM and RSTD remains constant over time, it indicates stationary data. Another approach (which we followed) is to perform the ADF [Ful09] and KPSS [KPSS92] tests together on the time series. These tests are the statistical methods used to check whether the time series data is stationary or not. ADF test makes a null hypothesis on the data as it is non-stationary prior to performing the test where as KPSS test is opposite of ADF. In ADF test results into ADF statistics value, P-value, and critical values. If ADS statistics value is more negative than the significant level values given in table 5.4 and the P-value is less than the $\alpha = 0.05$, then it indicates that the time series is stationary. Otherwise, transformation is mandatory. In KPSS test results into KPSS statistics value, P-value, and critical values. If KPSS statistics value is less than the critical values and the P-value is greater than $\alpha = 0.05$, then the series is stationary. Otherwise, transformation is mandatory. There are 4 possible conclusions that can be derived depending upon different permutations of ADF and KPSS test results being stationary or non-stationary. Table 5.2 shows four possible conclusions. Table 5.3 and 5.4 shows the results of stationary property check before and after transformation.

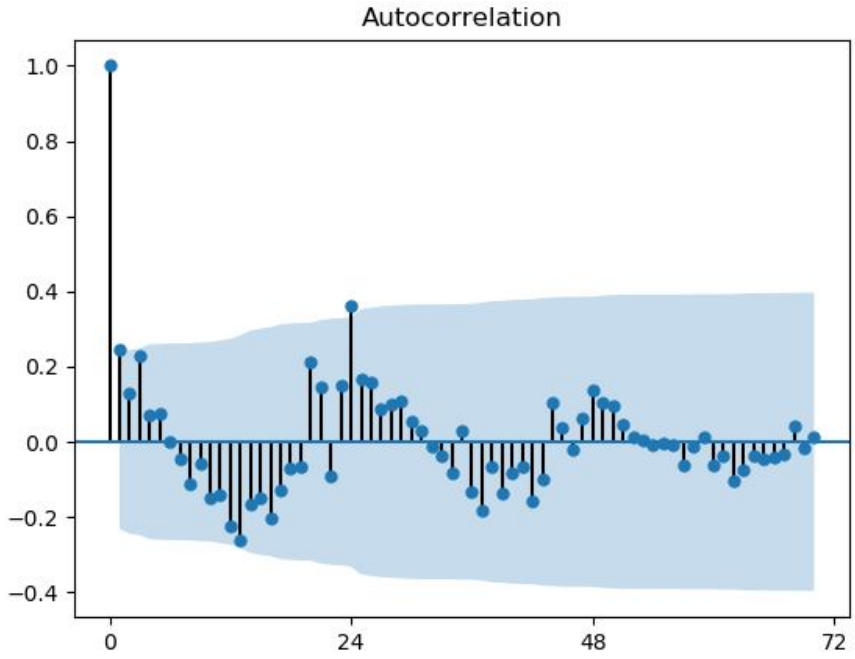


(a)

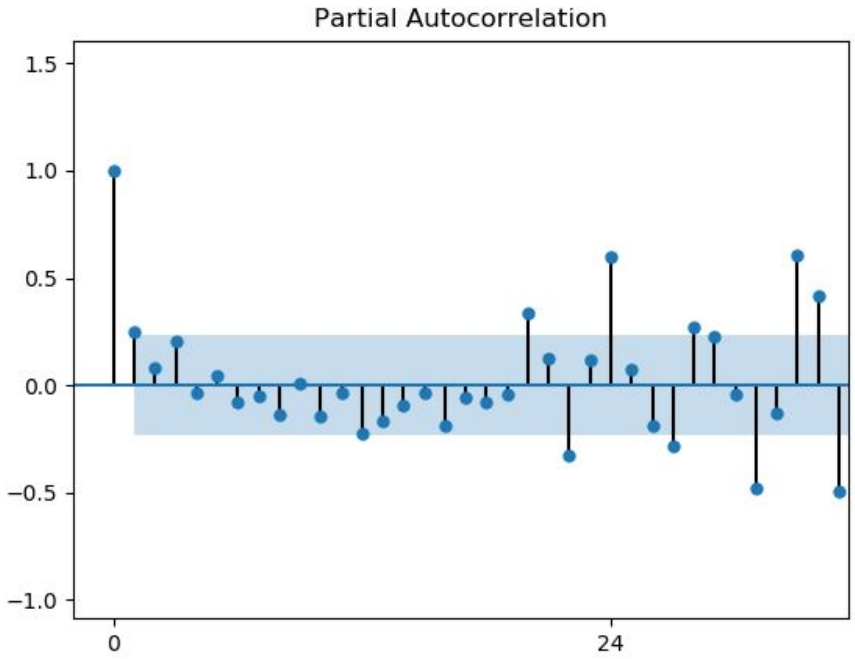


(b)

Figure 5.3: Time Series with Minute-scale and LBPE Modeling. **(a)** ACF ,**(b)** PACF .

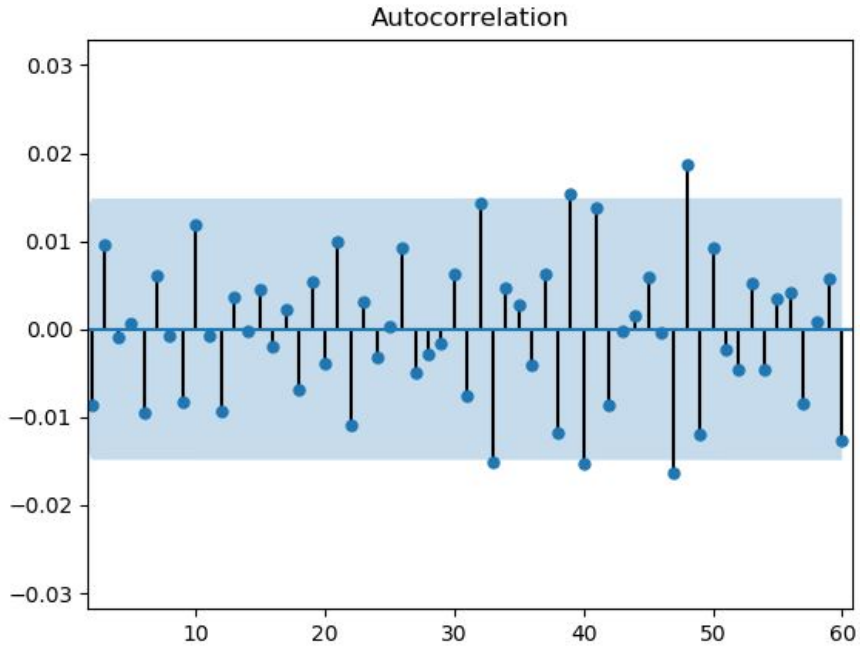


(c)

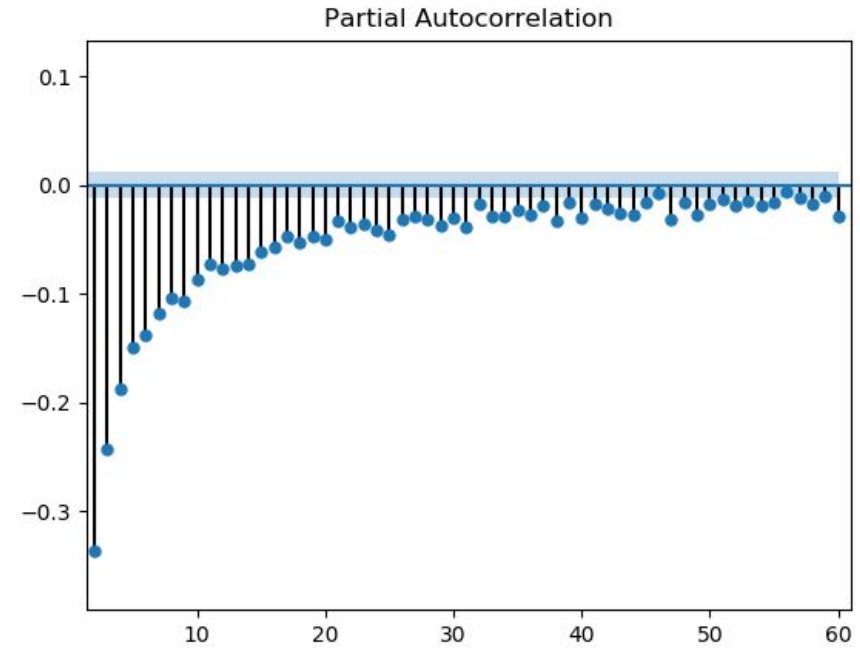


(d)

Figure 5.3: (Continued) Time Series with Hour-scale and LBPE Modeling. **(c)**ACF, **(d)**PACF.

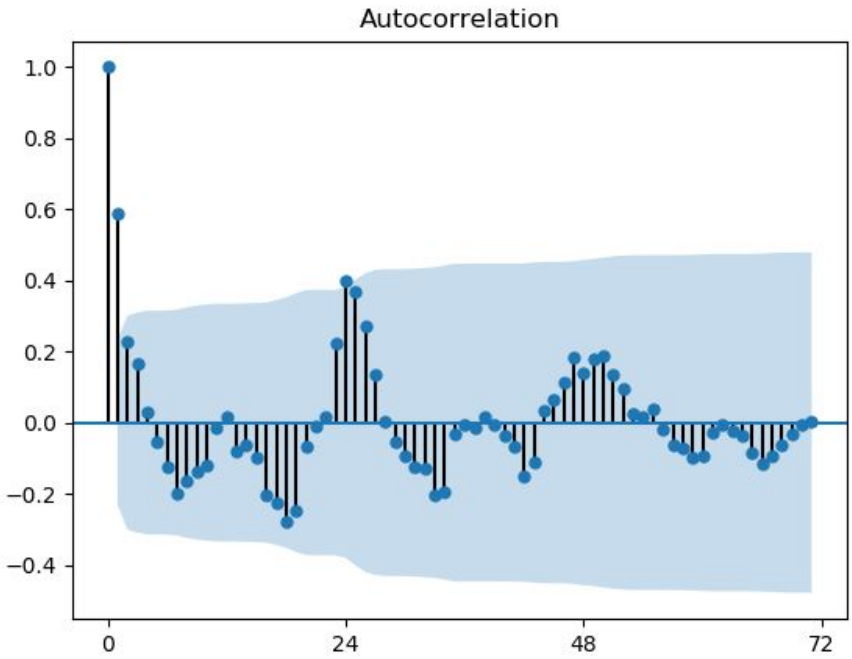


(e)

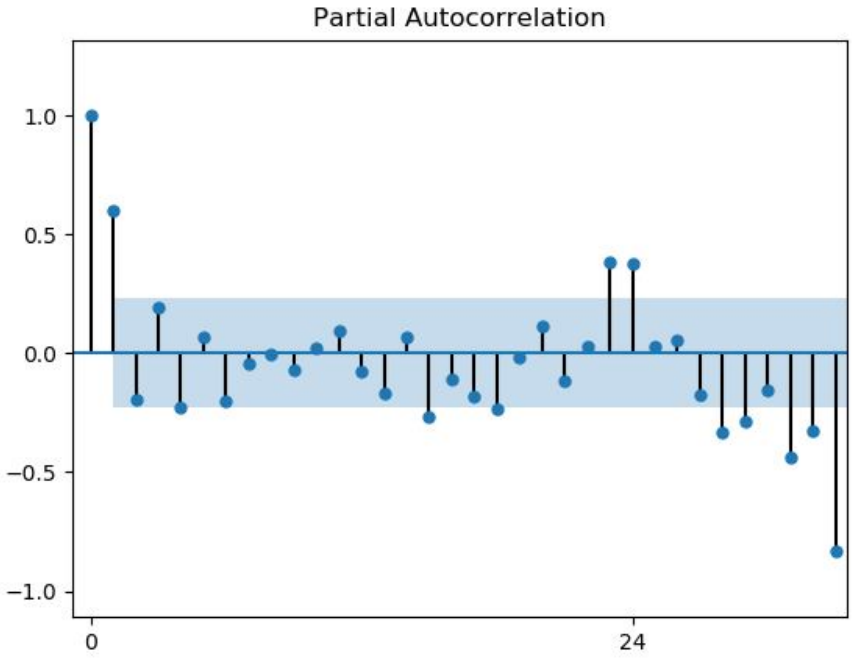


(f)

Figure 5.3: (Continued) Time Series with Minute-scale and PB Modeling. **(e)** ACF, **(f)** PACF.



(g)



(h)

Figure 5.3: (Continued) Time Series with Hour-scale and PB Modeling. **(g)** ACF, and **(h)** PACF.

Table 5.2: ADF and KPSS Conclusion about Stationarity of Time Series

if ADF says	**if KPSS says**	**Conclusion**
Stationary	Not stationary	Series is difference stationary and we have to perform differencing or differencing with a shifts or differencing with a seasonal shift or log transformation along with differencing with a shift repeatedly to make it stationary
Not stationary	Stationary	Series is trend stationary and we have remove the trend by performing any of the transformation to make it stationary
Stationary	Stationary	Series is stationary
Not stationary	Not stationary	Series is not stationary and we have to perform any of the transformation repeatedly to make it stationary

5.4.6 AR/MA Model's Construction, Fine-Tuning and Evaluation

Finally, using the transformed time series from the previous step and by observing the ACF and PACF values, we can estimate the order parameters for AR/MA model. Table 5.5 of ACF and PACF illustrates that ACF exhibits a sine wave and PACF exhibits an exponentially declining wave at lags. In order to select the best fit models from the set of evaluation parameters, we use AIC, BIC and forecasting errors. AIC measures the relative quality of model for a given set of data, BIC helps to select a model from a finite set of models, forecasting errors are the differences between the observed and expected value which is the value of unpredictability. Like forecasting errors, we have used scale-dependent errors like Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and others like Mean Error (ME). Table 5.6 shows the set of models that are chosen based on the measures mentioned above, after fine-tuning the model. We presented 2 approaches

Table 5.3: Stationarity Check using ADF and KPSS Tests before Transformation

Test	**Time series**	**Statistics**	**P-value**	**Criteria Values**			**Conclusion**
				1%	**5%**	**10%**	
ADF	$X^{(2)}_{t=min,l=l}$	-27.298	0.0	-3.430	-2.861	-2.566	Stationary
KPSS	$X^{(2)}_{t=min,l=l}$	1.284	0.010	0.790	0.463	0.347	Not Stationary
ADF	$X^{(2)}_{t=hr,l=l}$	-6.140	$7.98\text{x}10^{-08}$	-3.526	-2.903	-2.588	Stationary
KPSS	$X^{(2)}_{t=hr,l=l}$	0.206	0.100	0.739	0.463	0.347	Stationary
ADF	$X^{(2)}_{t=min,p=l}$	-16.525	$2.044\text{x } 10^{-29}$	-3.430	-2.861	-2.566	Stationary
KPSS	$X^{(2)}_{t=min,p=l}$	1.206	0.010	0.739	0.463	0.347	Not Stationary
ADF	$X^{(2)}_{t=hr,p=l}$	-3.754	0.003	-3.530	-2.905	-2.590	Stationary
KPSS	$X^{(2)}_{t=hr,p=l}$	0.183	0.100	0.739	0.463	0.347	Stationary

in the data smoothing stage: LBPE, and PB modelling. The results obtained in table 5.6 clearly suggest that on comparing both approaches in terms of hours and minutes separately, the PB modelling approach gives better results than the LBPE approach. This is evident from the fact that the error values along with the AIC and BIC obtained by applying the PB model are much smaller than those obtained by applying the LBPE, which concludes that the PB model has an edge over the LBPE model.

5.5 Conclusion

In this chapter, we presented a generalized multi-time-scale time series model to forecast click fraud behavior in terms of minutes and hours. Our proposed approach also allows us to forecast the behavior in terms of seconds, and even a smaller

Table 5.4: Stationarity Check using ADF and KPSS Tests after Transformation

Test	**Time series**	**Statistics**	**P-value**	**Criteria Values**			**Conclusion**
				1%	**5%**	**10%**	
ADF	$X^{(4)}_{t=min,l=l}$	-38.323	0.0	-3.430	-2.861	-2.566	Stationary
KPSS	$X^{(4)}_{t=min,l=l}$	0.0010	0.100	0.739	0.463	0.347	Stationary: log and one diferrencing with a shift
ADF	$X^{(4)}_{t=hr,l=l}$	-6.140	$7.98\text{x}10^{-08}$	-3.526	-2.903	-2.588	Stationary
KPSS	$X^{(4)}_{t=hr,l=l}$	0.206	0.100	0.739	0.463	0.347	Stationary
ADF	$X^{(4)}_{t=min,p=l}$	-50.285	0.0	-3.430	-2.861	-2.566	Stationary
KPSS	$X^{(4)}_{t=min,p=l}$	0.0009	0.100	0.739	0.463	0.347	Stationary: one differencing with a shift
ADF	$X^{(4)}_{t=hr,p=l}$	-3.754	0.003	-3.530	-2.905	-2.590	Stationary
KPSS	$X^{(4)}_{t=hr,p=l}$	0.183	0.100	0.739	0.463	0.347	Stationary

timescale could be examined. This is the very first attempt that has been made in this regard, which deals with forecasting click fraud behavior using AR and MA time series modelling. In the proposed approach, a raw dataset with multiple attributes is taken, through which the probability of being fraud was calculated by applying LBPE and PB modelling. Using this information, the fraudulent pattern is identified, and a threshold value is set in order to classify the data as fraudulent or not. Next, homogeneity property of the data is evaluated. If this property is not satisfied, the variance is homogenized, and the normality is balanced by applying the box-cox transformation to the data. The data is tailored to account for 2 different timescales – minutes and hours, and then stationarity of the data is verified using ACF, PACF and statistical models. If the stationarity property is not satisfied, some transformation techniques are applied. Then by using ACF and PACF plots, for each time series, we identify the AR and MA terms, and then we validate our

Table 5.5: ACF and PACF Exponentially Declining Behaviors on Time Series Data at Different Scales.

$X^{(4)}_{t=min,l=l}$			$X^{(4)}_{t=hr,l=l}$			$X^{(4)}_{t=min,p=l}$			$X^{(4)}_{t=hr,p=l}$		
Lags	ACF	PACF	Lags	ACF	PACF	Lags	ACF	PACF	Lags	ACF	PACF
1	-0.002	-0.35	60	+1.0	+1.0	1	-0.009	-0.34	60	+1.0	+1.0
2	+0.002	-0.25	120	+0.28	+0.25	2	+0.01	-0.25	120	+0.6	+0.6
3	+0.005	-0.18	180	+0.013	+0.1	3	-0.001	-0.19	180	+0.23	-0.22
4	-0.012	-0.16	240	+0.023	+0.2	4	+0.001	-0.16	240	+0.17	+0.22
5	+0.005	-0.14	300	+0.07	-0.1	5	-0.01	-0.15	300	+0.03	-0.28
6	-0.001	-0.13	360	+0.08	+0.1	6	+0.006	-0.13	360	-0.05	+0.03
7	-0.005	-0.12	420	+0.0	-0.2	7	-0.001	-0.11	420	-0.16	-0.21
8	+0.004	-0.1	480	-0.03	-0.1	8	-0.008	-0.12	480	-0.2	-0.02
9	+0.003	-0.08	540	-0.11	-0.22	9	+0.012	-0.08	540	-0.17	+0.0
10	-0.008	-0.09	600	-0.06	+0.0	10	-0.001	-0.07	600	-0.16	-0.03

identified models by checking the least forecasting errors, AIC and BIC. The error data turns out to be very minimal, suggesting some white noise. In the end, our approach produced various models, by minimizing forecasting errors, AIC and BIC, which we have considered as a metric to choose the best models, and also showed that the PB model approach is better as compared to the LBPE model.

Table 5.6: Fine Tuning: Model Order Selection Based on Different Criteria for Multi-time-scale

Selection Criteria	**Time-Series: AR(term) |MA(term) Model**	**ME (%)**	**MSE (%)**	**MAE (%)**	**RMSE (%)**	**AIC (thousands)**	**BIC (thousands)**
Minimizing errors	$X^{(4)}_{t=min,l=l}$: AR(0)|MA(2)	**-1.52**	10.31	53.0	32.10	187.4	187.4
	$X^{(4)}_{t=hr,l=l}$: AR(2)|MA(1)	1.65	**0.41**	4.79	**6.44**	-0.180	-0.168
	$X^{(4)}_{t=hr,l=l}$: AR(0)|MA(3)	1.56	0.45	**4.77**	6.77	-0.183	-0.171
	$X^{(4)}_{t=hr,l=l}$: AR(0)|MA(6)	**0.81**	0.51	5.31	7.19	-0.179	-0.161
	$X^{(4)}_{t=min,p=l}$: AR(0)|MA(1)	-0.02	**0.41**	5.39	**6.40**	-66.5	-66.4
	$X^{(4)}_{t=min,p=l}$: AR(5)|MA(0)	0.001	0.46	**5.31**	6.80	-63.0	-62.9
	$X^{(4)}_{t=min,p=l}$: AR(0)|MA(0)	**0.0001**	0.78	5.82	8.84	-49.5	-49.4
	$X^{(4)}_{t=hr,p=l}$: AR(0)|MA(1)	0.08	**0.0077**	0.68	**0.87**	-0.470	-0.463
	$X^{(4)}_{t=hr,p=l}$: AR(2)|MA(0)	0.064	0.0091	**0.669**	0.957	-0.465	-0.456
	$X^{(4)}_{t=hr,p=l}$: AR(6)|MA(0)	**0.04**	0.0092	0.72	0.95	-0.465	-0.447
Minimizing AIC	$X^{(4)}_{t=min,l=l}$: AR(0)|MA(2)	-1.52	10.31	53.0	32.10	**187.4**	187.4
	$X^{(4)}_{t=hr,l=l}$: AR(0)|MA(3)	1.56	0.45	4.77	6.77	**-0.183**	-0.171
	$X^{(4)}_{t=min,p=l}$: AR(0)|MA(1)	-0.02	0.41	5.39	6.40	**-66.5**	-66.4
	$X^{(4)}_{t=hr,p=l}$: AR(0)|MA(1)	0.08	0.0077	0.68	0.87	**-0.470**	-0.463
Minimizing BIC	$X^{(4)}_{t=min,l=l}$: AR(0)|MA(2)	-1.52	10.31	53.0	32.10	187.4	**187.4**
	$X^{(4)}_{t=hr,l=l}$: AR(1)|MA(0)	1.84	0.41	4.78	6.46	-0.182	**-0.175**
	$X^{(4)}_{t=min,p=l}$: AR(0)|MA(1)	-0.02	0.41	5.39	6.40	-66.5	**-66.4**
	$X^{(4)}_{t=hr,p=l}$: AR(0)|MA(1)	0.08	0.0077	0.68	0.87	-0.470	**-0.463**

CHAPTER 6

LEARNING-BASED MODEL TO FIGHT AGAINST FAKE LIKE CLICKS ON INSTAGRAM POSTS

Online social networks (OSN) are one of the favorite places where people share posts like their photos, videos, and text to gain popularity. On the other hand, the marketing industry tries to gain the popularity of their advertisement using such OSN's. Popularity of a particular post depends on the number of likes received by that post. To increase one's social worth, people try to use this market by artificially increasing the likes on their posts. There is a lack of research in the current literature on Instagram which is one of the growing OSNs. Our work focuses on detecting valid and fake like of posts with the application of learning model taking into consideration several popular factors. We developed an automated learning model to detect fake liking behavior on the Instagram post. The learned model can accurately differentiate between the legitimate and fake liker with an accuracy of 97% with ensemble-based learning model and also autoencoder is used to detect bots activity.

6.1 Introduction

Click fraud could happen anywhere, like in OSN, for example Facebook, Twitter, Instagram, and LinkedIn, etc. or in an online/in-app advertisement network, for example Google, Bling, AdMob, Facebook, and Instagram, etc. Advertisement network is a place to drive revenue and OSN is a place to drive popularity. Here, click plays a vital role in both the networks, for example; firstly, like clicks on an image

post or video post constitutes popularity level of the poster and the impact based on the content of the post. Secondly, clicks on ads constitute to impact on earning of a publisher and advertiser. However, it is challenging to detect whether the clicks are legitimate or not. Figure 6.1 depicts clicks impact in OSN as popularity index and in ad networks as revenue index. As the popularity of the post increases, the OSN recommender system starts displaying ads to the relevant audience based on location, demographics, interests, behaviors, custom audiences, lookalike audience, and automated targeting [JL08, ins18]. However, the recommender system fails to suggest relevant ads to the audience if there are large number of fake likes. On the other hand, these fake like clicks also contribute to the popularity of the fake news on OSN. Likewise, it is difficult to avoid spreading fake news among the people in the OSN. As the popularity of the fake news increases then it remains as an active post in the post feeds that in turn keeps on appearing in all the accounts [dLSB17, FVD$^+$16]. Following are the possible ways that get fake likes clicks: one of the straightforward ways is to influence friends to provide fake likes, artificially increase the popularity with the help of botnets, buying fake likes from the paid services, getting fake likes for free from click spammers, to become a member of collusion network.

6.1.1 Summary of contribution

Collection of data based on different parameters and attacks, identification of important features, performed pr-analysis to show the relationship between the follower and following participation in valid and invalid like clicks, we developed an automated learning model to detect fake liking behavior on the Instagram post, and we also examined autoencoder loss function to differentiate bots and human clicks.

Figure 6.1: Clicks impact in Online Social Network as Popularity Index and in Ad Networks as Revenue Index.

6.1.2 Organization of the Chapter

In Section 6.2 we review the related works, in Section 6.3 we define and explain the problem formulation, in Section 6.4 we provide details about data collection, description of dataset, pre-analysis and experimental setup, in Section 6.5 we provide information about the algorithms used in the experiment, in Section **??** we discuss the presented results, and in Section 8.6 & 6.8 we conclude our discuss and future work.

6.2 Related Work

When it comes to the influence of OSN, we are mostly interested in how the different elements of a particular social network can impact the significant aspects of the same

platform. Some studies details these impacts. For instance, [LES07] explains the variety of influence of profile-centric parameters on the friends count on Facebook. Their results shows that the friends was impacted by common interests, such as college, place of stay, common department, and the same university, even with the updated time. Similarly, In [GBCT13] author researched on pinterest, where they found that female with less followers can use highly specific words (for instance: want, look, need, and use) and this can leads to having higher re-pins on pinterest. Finally studies from [SHPC10], suggested that tags and URLs have higher impact on retweeting business.

Most of the OSN are being effected by various malicious activities. Studies regarding these malicious activities like fake collaboration have been done widely on Twitter [BMRA10, VB17, GCS+15], Facebook [GHW+10, LCW10, BSLL+16, BXG+13, DCFJ+14], YouTube [LMC+16], but on a very lesser extent, on Instagram [CYYP14]. Beutel et al. uses the 'lock-step' behaviour for detecting fake likers on Facebook by considering temporal aspects [BXG+13]. In [GCS+15], the author focused on malicious behavior of retweeting the tweets on Twitter by examining some trends in networks and temporal patterns. Network and temporal features are difficult to obtain even though they are effective in study. To overcome this, we can use content-based analysis which yields better results. In [BSLL+16], the author try to detect the fake page liking behavior on Facebook using significant features such as 'profile' and 'post' features of likers.

Our work adds more features to the content based method in two different ways: First, by taking into consideration the relationship between a liker and a poster. Second, by also collecting the relationships of the follower and following of the both legitimate and fake post's liker upto three level in breadth first search manner.

6.3 Problem Formulation

Since there is a lack of research in the current literature on Instagram which is one of the growing OSNs. Our work focuses on detecting valid and fake like of posts with the application of learning model. Figure 6.2 depicts the architecture of valid and fake like clicks participation on Instagram posts.

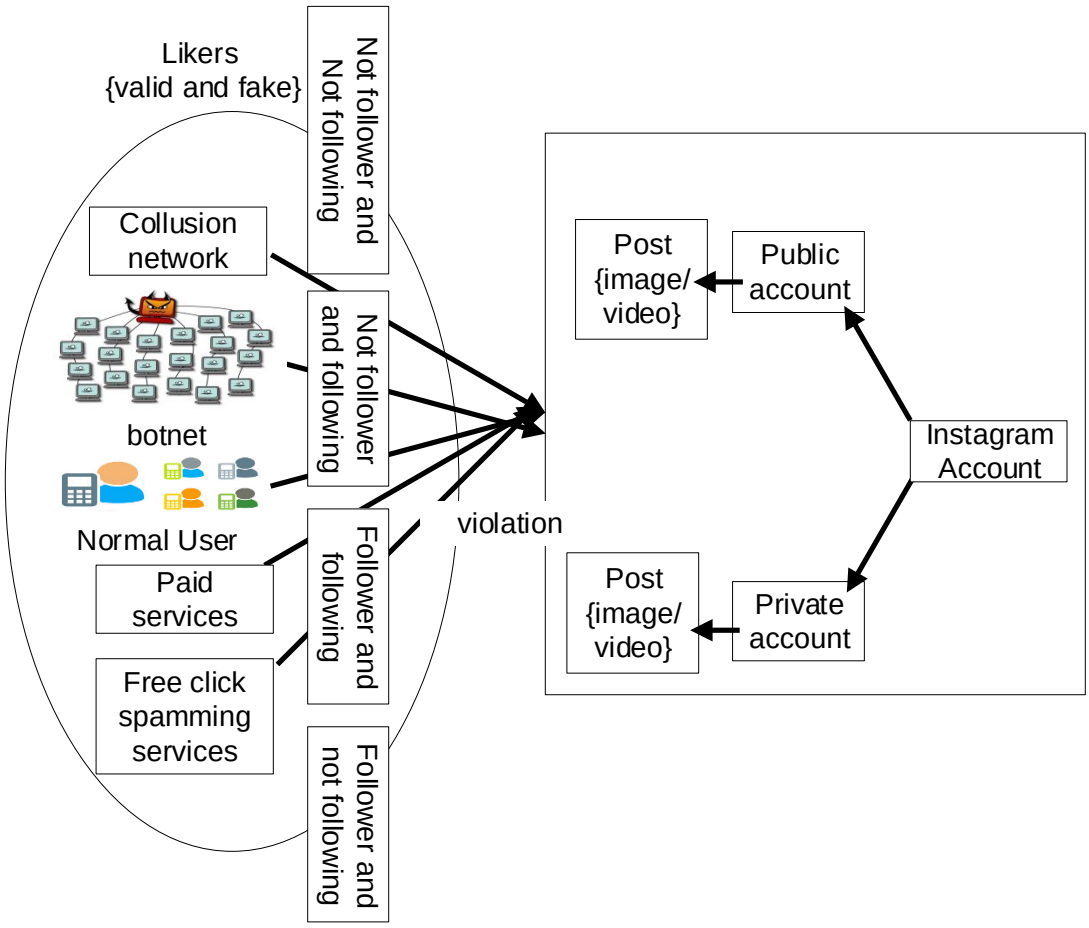


Figure 6.2: Architecture of Fake Clicks through Different Participants and Valid Clicks through Genuine User

Post: Content that is sharable in the Instagram like image, video etc.

Poster: One who posts an image or video on his/her account.

$$poster_i\{post_j\} = \{image, video\} \tag{6.1}$$

In equation 6.1 a poster i could have j number of posts where $i, j = 0, 1, 2....n$

Follower: An account becomes a follower when he/she legally gets approved by the account to which it was requested. Once approved, from there onwards followers get updates on all the content shared by the approved account and also have rights to participate in expressing emotions in the form of like or comment.

Following: It is the reverse of the follower, where an approver account is legally accepted by another account to follow his/her posts.

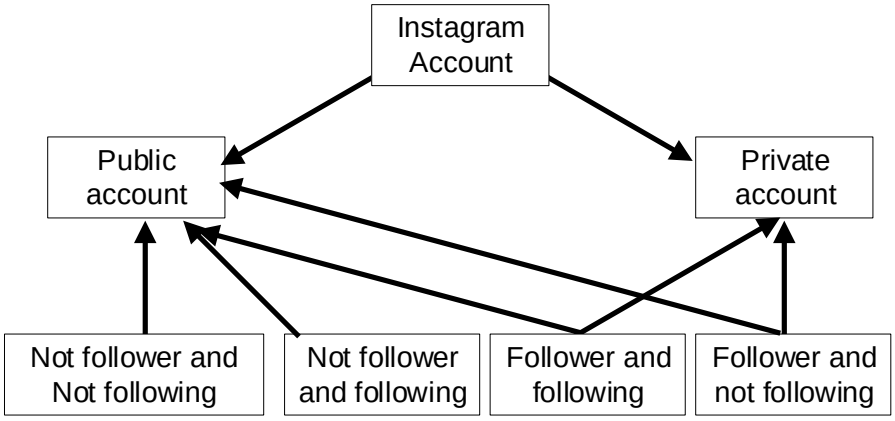


Figure 6.3: Accounts to Followers and Followings Accessibility Relationship (Upward Arrows means Accessible)

Account_type: poster or liker account type can be public or private mode in their account privacy settings. Here, if an account type is private, then his/her posts can be accessible by only his/her follower's account, but followings accounts can have access only if they exist again in followers list. In case of public account type, his/her posts can be accessible by his/her followers, followings, followers-followers, followers-following, followings-followers, followings-following i.e. a public account

content can be accessible by anyone having a profile on Instagram. In figure 6.3 followers and followings belongs to an account that has approved followers request to follow and is following either followers or other accounts.

FW: Follower, FG: Following, FWS: Followers, FGS: Followings

$$account\_type = \{public, private\} \tag{6.2}$$

$$\begin{aligned} poster_i\{account\_type\} = \{post_j, FW_p, FG_q, \\ FWS_p\_FW_r, FWS_p\_FG_s, \\ FGS_q\_FW_t, FGS_q\_FG_u, \\ rest\_of\_all_v\} \end{aligned} \tag{6.3}$$

In equation 6.3 and 6.4 where $i, j, p, q, r, s, t, u, v = 0, 1, 2...n$. If the account type is public, then the account posts are accessible by anyone as follows:

Follower could be same as following i.e.

$accessibility = \{liker | liker \in (follower == following)\}$

Following could be same as follower i.e.

$accessibility = \{liker | liker \in (following == follower)\}$

Follower and following could be distinct i.e.

$accessibility = \{liker | liker \in \{following, follower\}\}$

Any one i.e.

$accessibility = \{liker | liker \notin \{follower, following\}\}$.

$$\begin{aligned} poster_i\{public\} = \{post_j, FW_p, FG_q, \\ FWS_p\_FW_r, FWS_p\_FG_s, \\ FGS_q\_FW_t, FGS_q\_FG_u, \\ rest\_of\_all_v\} \end{aligned} \tag{6.4}$$

$$poster_i\{private\} = \{post_j, FW_p, FG_q\} \tag{6.5}$$

In equation 6.5 $i, j, p, q = 0, 1, 2...n$. If the account type is private, then the account posts are accessible as follows:

Only follower i.e.

$accessibility = \{liker | liker \in follower\}$

only following who is also a follower of the account i.e.

$accessibility = \{liker | liker \in (following == follower)\}$

As shown in figure 6.2 violation means participating in fake like clicks on the posts. This could happen because of botnet attacks, paid service or click spammer service. There is no barrier defined as a trust based access to authenticate these clicks [TPIS18]. A special case is when a genuine user participates in fraudulent clicks activity without having a legitimate interest and also could be a part of the collusion network. With these, we also assume that a like click on a partially viewed video post as fake like click as in [SAM+18].

Hence to solve aforementioned problem we propose a learning based model which is intelligent and automated in nature to detect fake and valid like clicks in Instagram.

6.4 Dataset Description, and Pre-analysis

Dataset Description: We collected real-time data from different account and posts for a period of one month using selenium web driver tool and manual data collection was done for legitimate private accounts. We also deployed honeypots by creating fake profiles with private and public mode. We uploaded the posts by mentioning "For testing purpose only. Please don't like it". From all genuine and fake account posts, we collected the like click observations generated by genuine

users, programmed botnets and fake spammers. Here only genuine users like clicks on public and private accounts posts are considered as valid emotions and others as fake. In case of honeypots, i.e. fake profile with fake posts the like clicks are considered as fake. The dataset contains 10,346 observations among which 5,988 observation are valid, and 4,368 observations are fake like clicks observations. The dataset consists of 38 attributes among them 37 are features, and one is a label. Table 6.1 describes the attributes of the dataset.

Feature Extraction: It serves to construct a feature vector for our machine learning model. It takes two types of features: numeric and text.

1. For numeric based features, we performed normal standardization by scaling each feature. This allow us to mitigate feature scaling issues that arise during classification methods which heavily rely on some distance metric.
2. For text based features, we parsed the test data and allocated it a unique integer number which is acceptable by the machine learning model.

Feature Engineering: In Instagram, we can set-up the accounts either as private or public. Each account type has their own pros. Inorder to build an effective machine learning model we divided dataset into 4 different groups with varied {Public, Private} & {Legitimate, Fake} accounts. Based on principal component analysis technique four features which had the lowest variance were dropped.

Pre-analysis: We have done analysis based on followings and followers of the likers who likes the particular post for both legitimate and fake Instagram account.

From figure 6.4, we observe that the frequency of the fake liker's #post is mostly near zero as compared to the frequency of the legitimate liker's #post.

For further analysis, we divided the total number of the followers and following into four range of groups viz. (0-500), (500-10k), (10k-1m), and (>1m).

Table 6.1: Characteristics of Dataset

Attributes	**Description**
Acc_Type	Account type of posters. It can be public or private
Acc_id	The id of the poster account
acc_no_of_post	Poster number of posts count
acc_no_fws	Poster number of followers count
acc_no_fgs	Poster number of followings count
Postid	Post id
post_time	Posted time in UTC
Liker_id	Account id of the post liker
is_liker_fw	Is post liker is in follower list of the poster account
is_liker_fg	Is post liker is in the following list of the poster account
liker_no_of_post	Post liker number of posts count
liker_no_fws	Post liker number of followers count
liker_no_fgs	Post liker number of followings count
$liker\_fgs\_id\_d_1, d_2, ..., d_n$	In post liker account following list first following id_d_1 to nth following id_d_n
$liker\_fgs\_no\_of\_post\_d_1, d_2, ..., d_n$	Post liker each followings number of post count
$liker\_fgs\_no\_fws\_d_1, d_2, ..., d_n$	Post liker each followings number of followers count
$liker\_fgs\_no\_fg\_d_1, d_2, ..., d_n$	Post liker each followings number of followings count
$liker\_fws\_id\_d_1, d_2, ..., d_n$	In post liker account follower list first follower id_d_1 to nth follower id_d_n
$liker\_fws\_no\_of\_post\_d_1, d_2, ..., d_n$	Post liker each follower number of post count
$liker\_fws\_no\_fws\_d_1, d_2, .., d_n$	Post liker each follower number of followers count
$liker\_fws\_no\_fg\_d_1, d_2, ..., d_n$	Post liker each follower number of followings count
is_like	Is the like is valid or fake

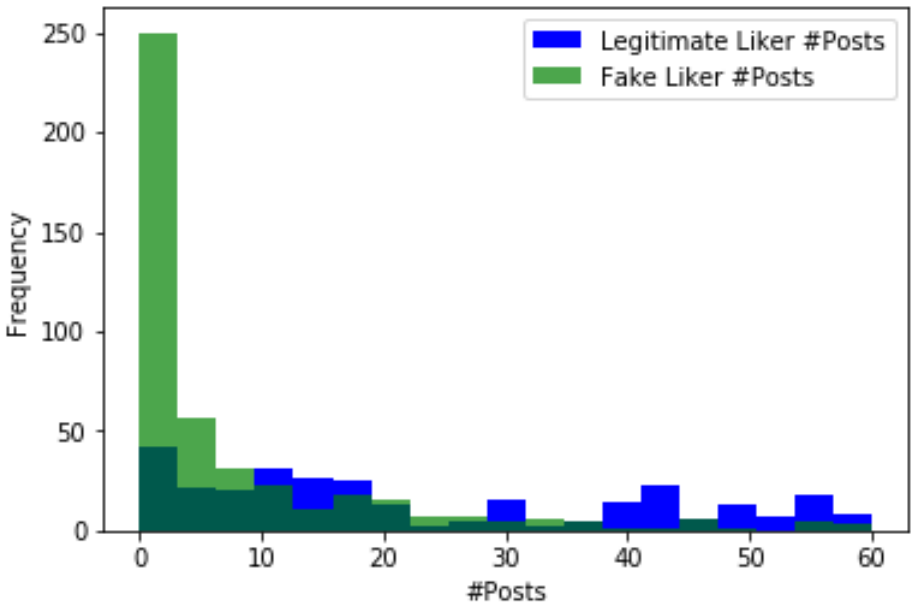


Figure 6.4: Legitimate Vs Fake Liker on Posts.

Since our main focus is on the number of followers and followings for both legitimate and fake accounts, we analyzed the histogram plots of their multiple combinations and found some interesting trends. From the plots we can notice that legitimate followers and followings have higher rank as compared to fake followers and followings in the range of (0-500) whereas it's vice-versa in all the remaining ranges.

Case 1. Following_Followers: Figure 6.5-6.7 shows #(Following_Follower) in range (0-500), (500-10k), and (>1m) for case 1. For each likers post, Following_Followers means the frequency of #followers of the followings for both legitimate and fake account post upto breadth level of 3. D1,D2 and D3 represents the breadth level of following and follower. L: Legitimate, F: Fake

Case 2. Following_Followings: Figure 6.8 and 6.9 shows #(Following_Followings) in range (0-500), and (500-10k) for case 2. For each likers post, Following_Followings means the frequency of #followings of the followings for both legitimate and fake

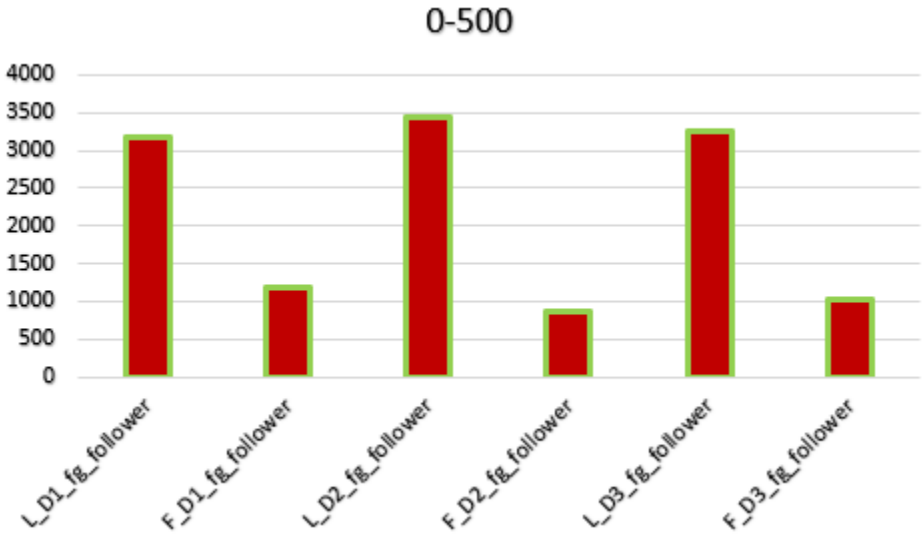


Figure 6.5: Frequency of Number of Followers of the Followings for both Legitimate and Fake Account Post upto Breadth Level of 3 in range (0-500).

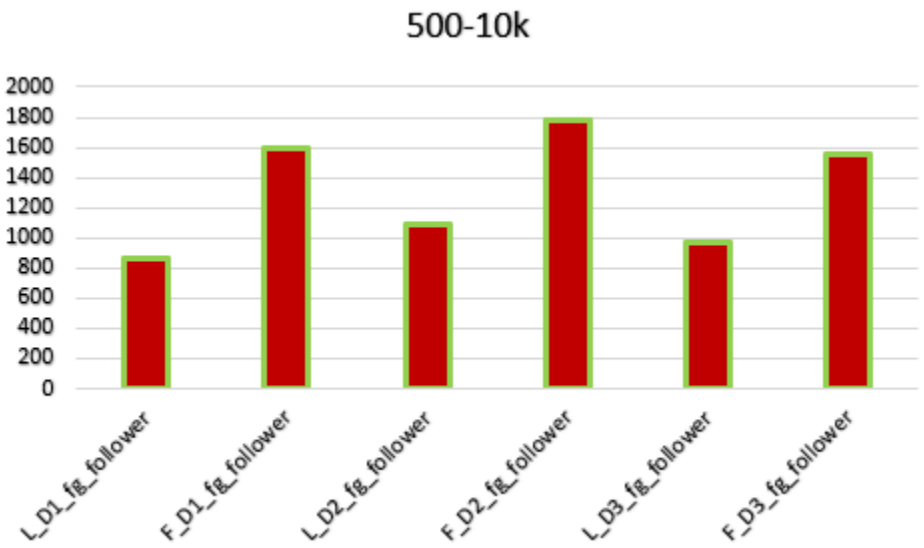


Figure 6.6: Frequency of Number of Followers of the Followings for both Legitimate and Fake Account Post upto Breadth Level of 3 in range (500-10k).

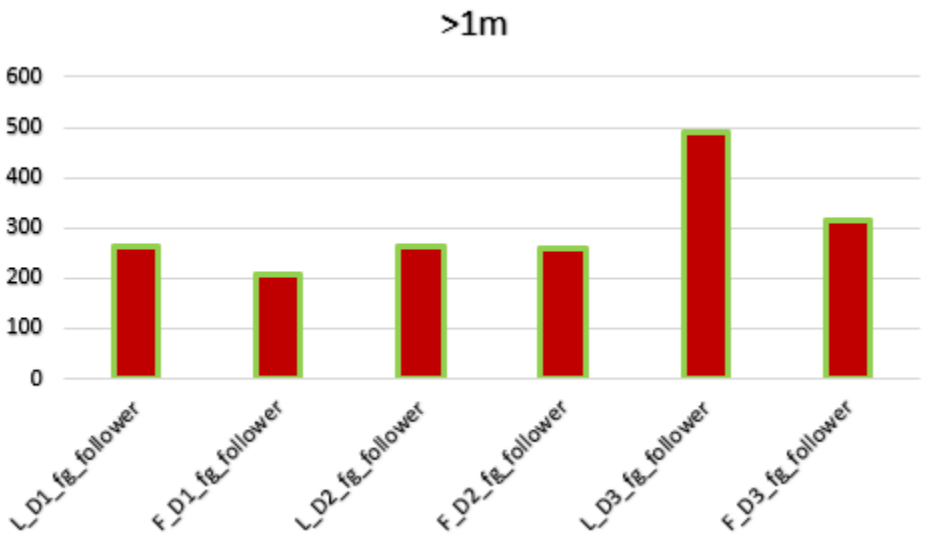


Figure 6.7: Frequency of Number of Followers of the Followings for both Legitimate and Fake Account Post upto Breadth Level of 3 in range (>1m).

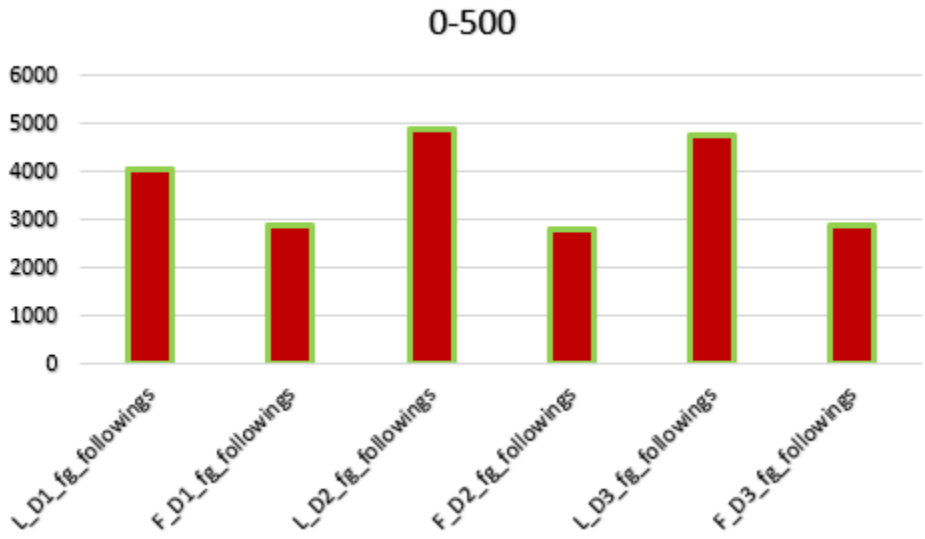


Figure 6.8: Frequency of Number of Followings of the Followings for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (0-500).

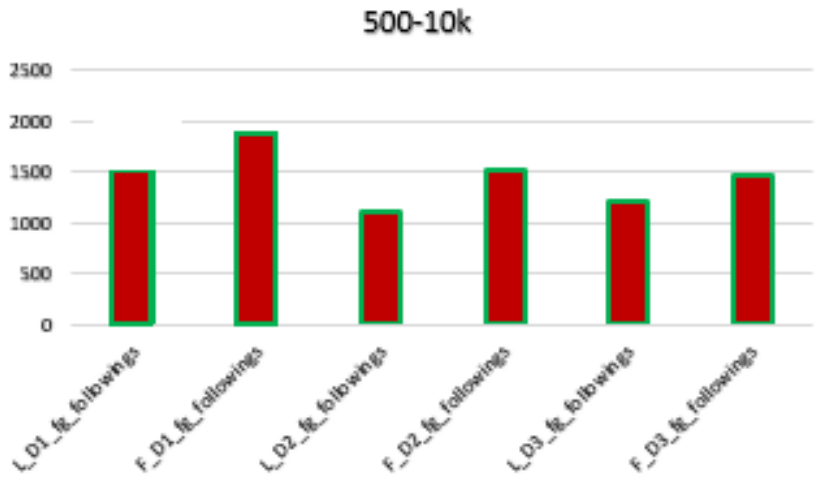


Figure 6.9: Frequency of Number of Followings of the Followings for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (500-10k).

account post upto breadth level of 3.

Case 3. Follower_Followings: Figure 6.10 and 6.11 shows #(Follower_Followings) in range (0-500), and (500-10k) for case 3. For each likers post, Follower_Followings means the frequency of #followings of the followers for both legitimate and fake account post upto breadth level of 3.

Case 4. Follower_Follower: Figure 6.12-6.14 shows #(Follower_Follower) in range (0-500), (500-10k), and (10k-1m) for case 4. For each likers post, Follower_Follower means the frequency of #follower of the followers for both legitimate and fake account post upto breadth level of 3.

6.5 Classification Algorithms

We have considered various classification methods in terms of single and ensemble based classifiers to detect fake and genuine like clicks as follows: Logistic Regression (LR), Support Vector Machine (SVM) in two versions like radial basis function and sigmoid kernels, K Nearest Neighbors (KNN) with two versions like Uniform

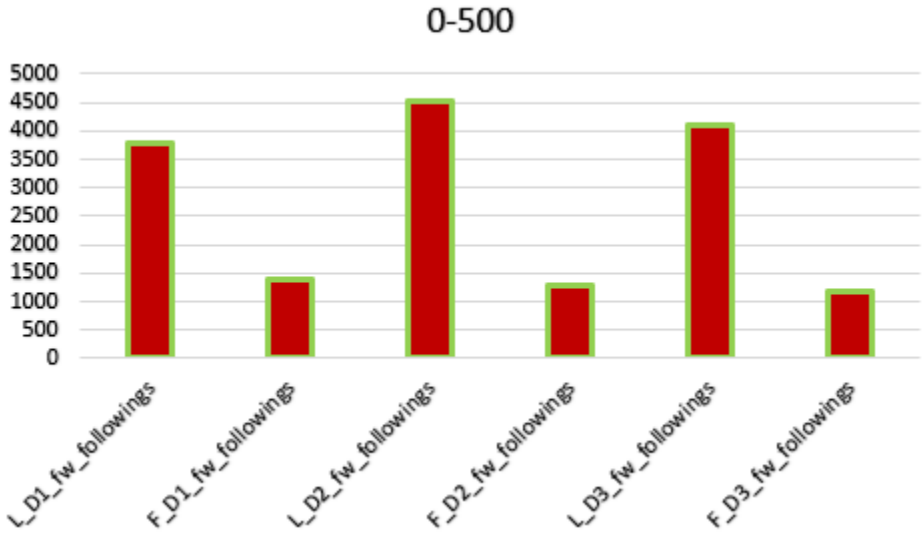


Figure 6.10: Frequency of Number of Followings of the Followers for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (0-500).

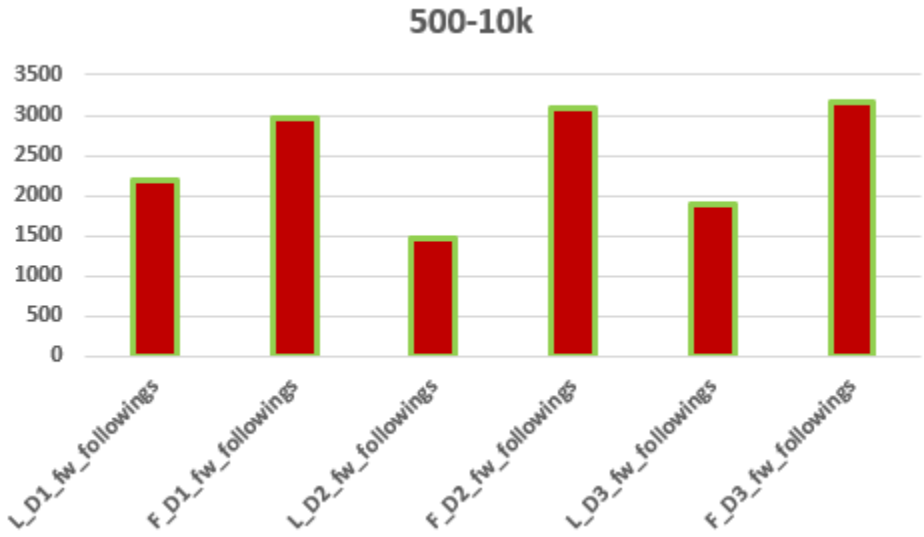


Figure 6.11: Frequency of Number of Followings of the Followers for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (500-10k).

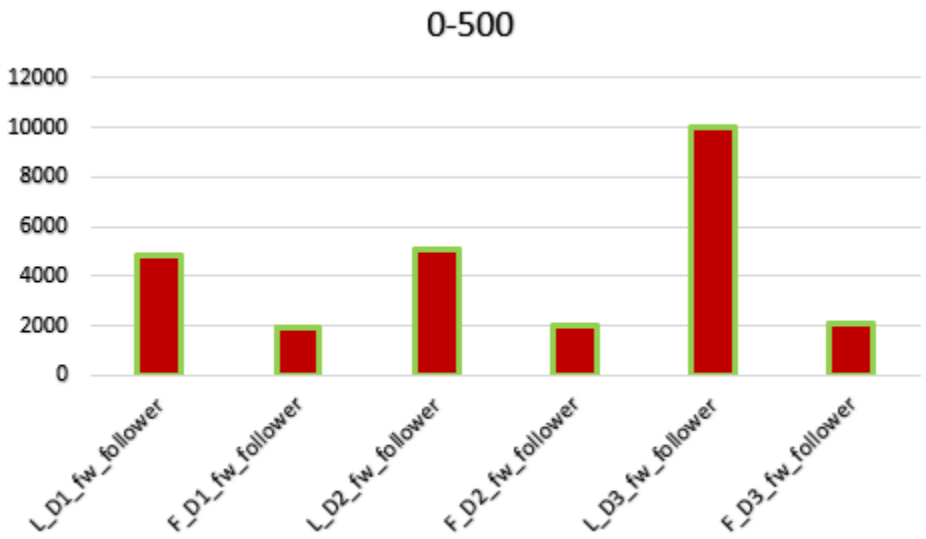


Figure 6.12: Frequency of Number of Follower of the Followers for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (0-500).

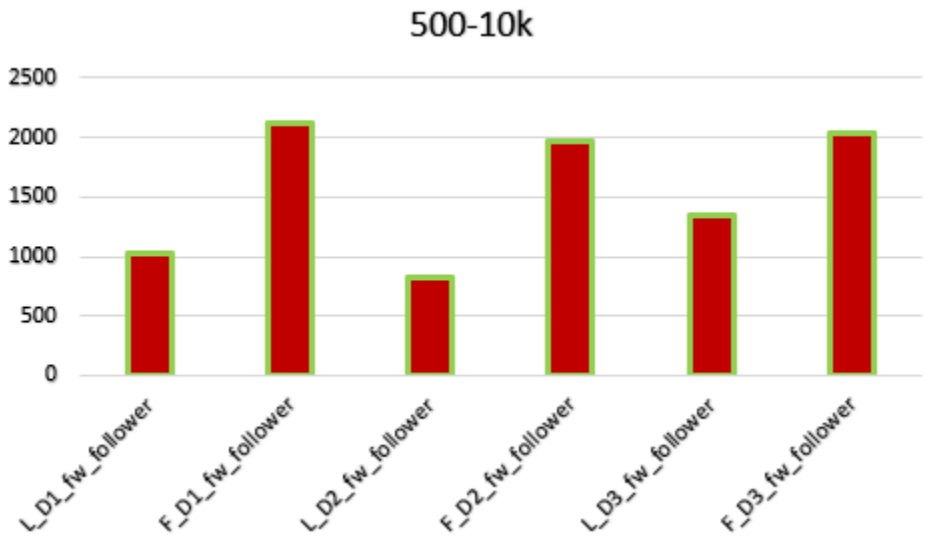


Figure 6.13: Frequency of Number of Follower of the Followers for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (500-10k).

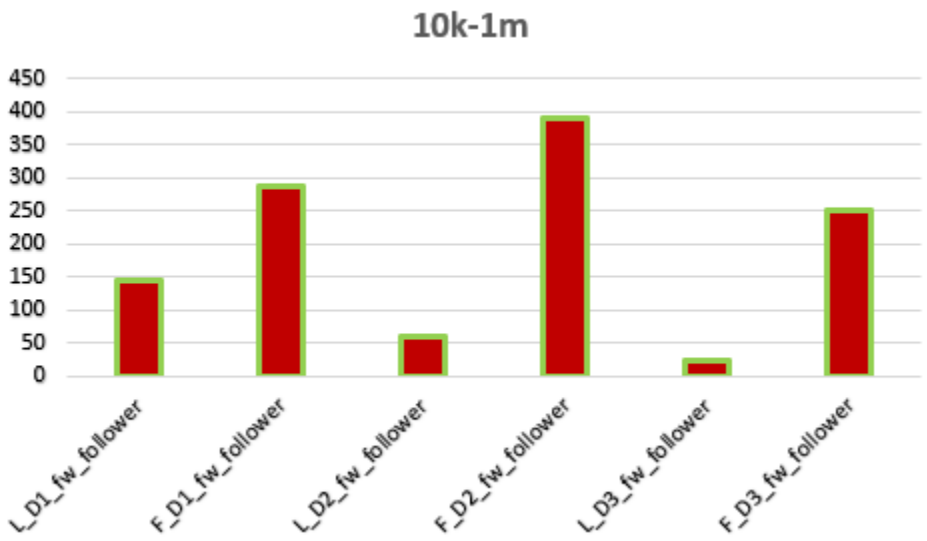


Figure 6.14: Frequency of Number of Follower of the Followers for both Legitimate and Fake Account Post upto Breadth Level of 3 in Range (10k-1m).

and Distance based kernels, Naive Bayes (NB) with two versions like Gaussian and Multinomial based kernels, Artificial Neural Network (ANN), and ensemble based like Random Forest (RF) with Gini and Entropy kernels, and Extra Tree (ERT) classifiers with Gini. With these, we also experimented bot detection approach proposed and suggested by [VAK+16, TKC+19] using autoencoder method.

6.6 Results and discussions

In this section, we discuss the evaluation metrics used to measure the performance of each model, model parameters and model validation used. Including these, we discuss the presented results in three parts namely: Single vs. ensemble based model, Autoencoder bot detection method, and ANN.

6.6.1 Evaluation Metrics

Metrics like precision, recall, accuracy are used to measure the performance of the classifiers. We have reported individual as well as all feature datasets result using the following equations.

$$d_i \in \{Gprivate\_Fprivate, Gprivate\_Fpublic, Gpublic\_Fprivate, Gpublic\_Fpublic\} \tag{6.6}$$

where i = 1, 2, 3, 4.

$$Precision(d_i) = \frac{TP(d_i)}{TP(d_i) + FP(d_i)} \tag{6.7}$$

$$Recall(d_i) = \frac{TP(d_i)}{TP(d_i) + FN(d_i)} \tag{6.8}$$

$$Accuracy(d_i) = \frac{(TP(d_i) + TN(d_i))}{(TP(d_i) + FP(d_i) + TN(d_i) + FN(d_i))} \tag{6.9}$$

where TP: True Positive, TN: True Negative, FP: False Positive, and FN: False Negative. We have also reported macro-averaged results using the following equations.

$$Precision = \frac{1}{4}\sum_{i=0}^{4} Precision(d_i) \tag{6.10}$$

$$Recall = \frac{1}{4}\sum_{i=0}^{4} Recall(d_i) \tag{6.11}$$

$$Accuracy = \frac{1}{4}\sum_{i=0}^{4} Accuracy(d_i) \tag{6.12}$$

6.6.2 Model Parameters

We configure our classifier algorithms with a different range of parameters as follows: For the Support Vector Machine (SVM), we use kernel as rbf, and sigmoid. For Random Forest and extra tree classifier, we use criterion values as entropy and gini and estimators between 1 to 10. For Logistic Regression, we set the range of the

cost function from 0.1 to 10. For the KNN, we set the weight options as uniform or distance and the nearest neighbor to be searched for in between 1 to 10. We set alpha to 1.0 for all the variants of the naïve bayes like Gaussian, Bernoulli, and Multinomial.

6.6.3 Model Evaluation

We use k-Fold cross-validation to evaluate our machine learning model. By using such evaluation method we divide the training dataset into k number of sub-datasets. The primary purpose of using k-fold cross validation is to use all of the training dataset for training the model and also to get the unbiased estimate of the model.

Table 7.5 shows the results of single and ensemble-based methods trained and tested using K-fold cross-validation on five different class of datasets including dataset with all features. Based on the result among single based learning model LR performs better with an accuracy of 92%. However, as overall, random forest (in particular "gini" as a criterion) and extra tree classifier performs best among single and ensemble based learning model with an accuracy of 97%.

6.6.4 Auto Encoder

In an auto encoder, there are two parts [VLBM08].

Encoder

Here the noise is added to the real data and is passed through a similar neural network as used in training, only now it gives an eight float32 output instead of one.

$$x\_data = OriginalData \tag{6.13}$$

Table 6.2: Results of Single and Ensemble-based Methods

M	**Datasets**															**Macro Average**		
	Gpri_Fpri			***Gpri_ Fpu***			***Gpu_Fpri***			***Gpu_Fpu***			***All features***					
	P	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**
LR	0.89	0.92	0.90	0.95	0.95	0.95	0.90	0.92	0.91	0.92	0.92	0.92	0.93	0.93	**0.93**	0.91	0.93	**0.92**
SVM (rbf)	0.97	0.97	0.85	0.97	0.97	0.85	0.76	0.58	0.79	0.76	0.62	0.78	0.97	0.97	0.86	0.97	0.97	0.85
SVM (sig)	0.49	0.97	0.49	0.50	0.97	0.50	0.50	0.97	0.51	0.52	0.92	0.50	0.55	0.97	0.57	0.49	0.97	0.49
KNN (Uni)	0.72	0.87	0.76	0.71	0.86	0.74	0.66	0.75	0.71	0.67	0.78	0.72	0.72	0.84	0.76	0.69	0.82	0.73
KNN (Dis)	0.75	0.91	0.80	0.72	0.91	0.77	0.70	0.83	0.76	0.68	0.84	0.74	0.76	0.89	0.80	0.71	0.87	0.77
NB (Gau)	0.67	0.90	0.72	0.72	0.88	0.76	0.54	0.83	0.62	0.58	0.84	0.65	0.61	0.88	0.67	0.63	0.86	0.69
NB (Mul)	0.55	0.74	0.53	0.61	0.76	0.62	0.57	0.26	0.57	0.48	0.16	0.52	0.67	0.48	0.62	0.55	0.48	0.56
RF (Gin)	0.96	0.96	0.97	0.96	0.96	0.95	0.95	0.93	0.96	0.96	0.94	0.97	0.94	0.95	**0.96**	0.96	0.95	**0.97**
RF (Ent)	0.96	0.96	0.95	0.96	0.96	0.97	0.96	0.97	0.95	0.96	0.93	0.95	0.95	0.94	0.94	0.96	0.95	0.96
ERT (Gin)	0.97	0.97	0.96	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95	0.96	0.95	0.95	**0.96**	0.96	0.96	**0.97**

Note: M: Method, P: Precision, R: Recall, A: Accuracy, Gprivate_Fprivate: Gpri_Fpri, Gprivate_Fpublic: Gpri_Fpu, Gpublic_Fprivate: Gpu_Fpri, Gpublic_Fpublic: Gpu_Fpu

Now we use the encoder's neural network to compress the noisy data, we just created

$$x\_compressed = encoder(x\_data) \tag{6.14}$$

Decoder

Here the output of encoder travels through another Neural network, and it finally gives an eight float32 output.

$$y\_pred = decoder(x\_compressed) \tag{6.15}$$

The Decoder output is compared with real input using root mean squared and based on the error, the auto encoder is then trained using Adam backprop as shown before.

$$E = Error = \sqrt{((x_{data} - y_{pred})^2)} \tag{6.16}$$

Then we take the Log of the Error as our Loss for deciding if the given like is a bot or human.

$$Loss = log(E) \tag{6.17}$$

Now since we trained our data on Users, this helps us detect bots when the Encoder is not able to regenerate the data.

We use the above autoencoder to discard the bots.

Discussion on Autoencoder Results

Due to the scarcity of data, we divided the dataset representing real likes in the ratio of 4:1 for training and testing.

We trained the autoencoder only on the real likes, as it will help us in setting the loss threshold. We got a Loss of 28.597 for real likes and a Loss of 29.391 for bot likes. The significant difference between Losses of the bot and real likes shows that auto encoder can distinguish between them.

6.6.5 Neural Network

We also ran our dataset on an ANN [GB10b, RHW86] and divided the train and test data in the 4:1 ratio. On neural network using batch normalization [IS15] and dropout [SHK+14] of 0.5 at each layer, except the output layer, we got precision of 0.91and accuracy of about 0.905. We trained the ANN using Adam Gradient Descent [KB14].

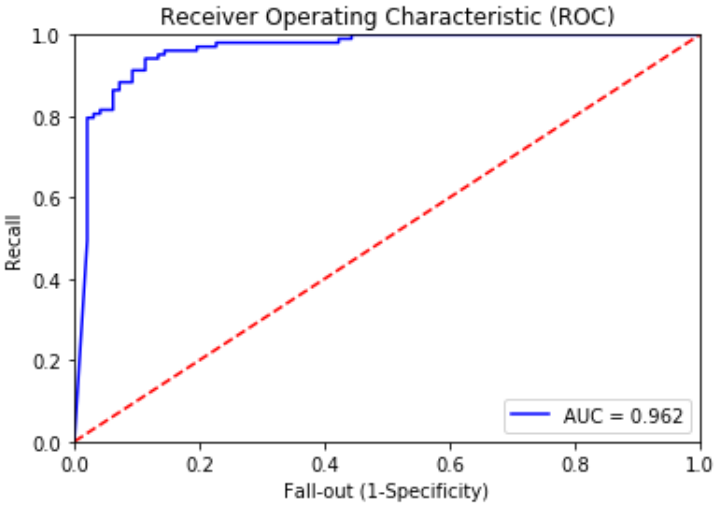


Figure 6.15: ROC curve of ANN

Discussion on ANN Results

Due to lack of sample, ANN results are down as compared to other models. However, with significant amount of data ANN may perform better than others.

6.7 Conclusion

With this work, we have shown that popularity index of posts on Instagram could be increased by fake like clicks through several attacks. We deployed honeypots and programmed botnets to create attacks and captured the data from all type of

accounts with valid and fake like clicks on several posts. We have explored all the unique parameters of the post such as to #(likers, followers, followings) by scrapping the data with several breadth/levels. We also analyzed the data where we found some pattern on valid and fake like clicks based on a number of posts, follower-following relationship, following-follower relationship, follower-follower relationship, and following-following relationship. Based on the experimental results, LR is an accurate predictor among the pool of single based methods, and Random forest is the best among single and ensemble-based methods. We also achieved better results with autoencoder to detect bots in our dataset with the help of loss function.

6.8 Future Work

We restricted ourselves to detect fake and valid like click but no views in case of the video posts. Firstly, with the number of views, we can also consider even more features to validate whether a view is legitimate or fake based on chosen criteria. One of the possible directions is to collect data based on several criteria's like depth and breadth levels exploration of the post , like click path and so on . Secondly, we can investigate further to propose a hybrid learning model in combination with ANN.

CHAPTER 7

MRFI AND ARFI: HYBRIDS OF FILTER AND WRAPPER FEATURE SELECTION APPROACHES

Feature selection has emerged as a craft, using which, we boost the performance of our learning model. Feature or Attribute Selection is a data preprocessing technique, where only the most informative features are considered and given to the predictor. This reduces the computational overhead and improves the efficiency and correctness of the classifier. Attribute Selection is commonly carried out by applying some filter or by using the performance of the learning model to gauge the quality of the attribute subset. Metric Ranked Feature Inclusion and Accuracy Ranked Feature Inclusion are the two novel hybrid feature selection methods we introduce in this chapter. These algorithms follow a two-stage procedure, the first of which is feature ranking, followed by feature subset selection. Our methods differ in the way they rank the features but follow the same subset selection technique. Multiple experiments have been conducted to assess our models. We compare our results with numerous works of the past and validate our models using 12 datasets. From the results, we infer that our algorithms perform better than many existent models.

7.1 Introduction

In Machine Learning, computer algorithms study statistical models to distill certain information from the data. This information is used to perform various tasks without constant human intervention by relying solely on inferences and patterns. Unfortunately, the nature and quality of the data fed to the learning algorithm determine it's performance. Many times, the data might be inadequate, noisy, or

erroneous, which leads to a loss in the regularity and accuracy of the predictions made by the machine. To avoid this, we have to rectify and remodel the data that the algorithm operates on. Either row correction or column correction are the possible solutions. The rows signify the input data, while the columns signify the features. A measurable property of the process under consideration is known as a feature. Each row is characterized by a vector of features and the target. The target or the class signifies the category to which that sample belongs.

Dimensionality Reduction is considered to be the best approach for column correction. Feature Selection (FS) and Feature Extraction are the two primary techniques of reducing the number of dimensions. Variable Extraction or Feature Extraction is the act of converting the given feature subset into a subset of lower dimensionality, where new features are fabricated by the combination of the available features. The number of dimensions can be minimized on applying FS as it picks a set of features from the initial set. FS can be carried out by mainly three methods: Wrapper, Embedded and Filter.

FS algorithms using the filter technique, pick features based on some score or statistical measure that is allocated to each feature. The predictor is not considered while choosing the best subset of variables in the filter approach These algorithms are computationally less expensive and fast, but may not always give the best feature subset. FS algorithms that are classified as wrapper methods can be considered as search algorithms, where many combinations of features are created, evaluated, and then compared with each other. The evaluation of each subset is performed with the help of the predictive model. The model runs on each subset, following which the subsets are assigned scores based on their performances. These scores are then used to pick the optimal feature subset. Many wrapper methods give better results, but cause a large overhead and may take extremely long periods of time if

the feature set is extensive. Nowadays, many methods which are a combination of filters and wrappers, called Hybrids, are being devised. These Hybrid algorithms exploit the advantages of both methods while overcoming many of the disadvantages. FS algorithms employing the embedded method, choose attributes which contribute heavily to the correctness of the learning model during it's creation.

7.1.1 Summary of Contribution

The existent FS algorithms are useful but do not always prove to be extremely helpful when we use certain machine learning algorithms like Random Forests. In this chapter, we propose two new FS techniques, Metric Ranked Feature Inclusion (MRFI) and Accuracy Ranked Feature Inclusion (ARFI), which can be used effectively across a variety of learning models. Our proposed algorithms are hybrids of the wrapper and wrapper methods and follow a two phase process. The first phase takes inspiration from the filter technique, and we assign scores for the features to rank them. For the first proposed algorithm, the score is assigned to each feature after clustering the data with the help of that feature alone. We use K-Means to cluster the data and then apply a clustering metric by the name of V-Measure. ARFI involves scoring each feature based on the accuracy of a classifier (Random Forest), which is evaluated with only that particular feature. Ranking the features using these techniques truly brings out their importance to the label. The next stage of the algorithm, i.e., the feature subset selection phase, avoids redundancy. Here, the variables are iteratively added to the optimal subset one by one, and each time, the learning model is evaluated. The recently added feature is retained or dropped depending on the calculated accuracy. The second stage behaves as the wrapper part. Both MRFI and ARFI share the same feature subset selection technique.

We validated our models with the various datasets and compared our results with another standard FS technique, Recursive Feature Elimination (RFE). Our models outperformed RFE with every dataset and gave us positive results.

7.1.2 Organization of the Chapter

The chapter provides a thorough review of the extensive research conducted in the past, regarding attribute selection, in Section 7.2. A detailed explanation about our algorithms and their preliminaries is given in Section 7.3. In Section 7.4, the hardware requirements and the various datasets used have been described. Section 7.5 contains discussions about our experimental outcomes. Lastly, Section 7.6 provides an outline of the work we have carried out.

7.2 Related Work

The efficacy of any predictor can be considerably improved by applying FS. It lessens the number of columns and thereby reduces noise. Lots of research has been done in this field and many survey and review papers describe various FS algorithms [MG03, BCSMAB+14, CS14]. Several kinds of FS algorithms can be implemented, but we focus on the wrapper, filter and some hybrid methods of variable selection.

7.2.1 Filter Approach

In [PK17], an FS technique based on correlation is proposed, in which the features are ranked based on the extent of redundancy between the attributes and their predictive capability. In [DCSL02], the entropy measure of a cluster is used to determine whether the data has distinct clusters or not. Kira and Rendell created

the FS technique called Relief [KR$^+$92]. In this algorithm, weights are allocated to every variable, and KNN is used to modify the weights. Almuallim and Dietterich developed another extremely famous algorithm by the name of FOCUS [AD92]. This algorithm conducts a comprehensive check of all feature subsets and then finds the minimal subset that can provide accurate labeling of the training data. Symmetrical Uncertainty is adopted as the goodness measure in [YL03].

Koller and Sahami [KS96] proposed a method which involves the elimination of a predecided number of features using backward elimination coupled with cross entropy. In [LS$^+$96], Liu and Setiono have implemented a method which uses random sampling to search for all feature subsets. Minimum Description Length of a feature subset, as the evaluation metric, was proposed by Pfahringer [Pfa95]. He makes use of Simple Decision Tables to add or remove features. In [QSZT15], a new method of FS based on Synonym Merge, Part Of Speech and Contribution Value is used to classify Chinese text. The FS model in [LMJY17] works on the principle of multi-objective mutual information. It considers both redundancy and relevance to the class. It makes use of NSGA, which is a multi-objective search algorithm. In [Bat94], the author proposes an algorithm based on the union of mutual information and pruning.

Fleuret [Fle04] had proposed another method to choose attributes based on mutual information. This method uses the conditional mutual information maximization criterion. In [CS03], the author presents a unique method of reusing the discarded features after applying FS. The multitask method of learning is used to provide extra information to the classifier through the model's output. In [PLD05], the author proposes a two-stage method based on maximum dependency, maximum relevance but minimum redundancy. Franklin and Vasudevan [VV16] propose a method by the name of Highly Correlated FS (HCFS). HCFS initially sets the

pertinence threshold, then finds associations among feature pairs and also among features and classes. The algorithm excludes uncorrelated features by building a tree. The feature tree is partitioned based on the relevance threshold. From this partitioning, the best feature cluster is then selected.

An FS approach, namely GClust, using interquartile range and clustering [KNK+19], has been proposed. Initially, the genes that correctly predict the classes for the inputs are chosen. The remaining genes are then clustered based on their similarity, and genes with the highest ranks are picked with the help of the Lasso method. On combining this with the initial subset, the final optimal feature subset is obtained. Kononenko [Kon94] proposed an extension of the RELIEF model and called it ReliefF. The extension is handy as it can deal with noisy, incomplete data. Moreover, it can handle multi-class datasets effectively. A unique model for FS for multiple label classification has been proposed in [SCML13]. In their model, the goodness of the attributes is evaluated on the basis of Information gain and ReliefF. The standard multi-label FS approach is then applied to convert the multiple labels into a single one. For this purpose, Binary Relevance and Label Powerset methods are used. In [WWK+13], the author proposes a method based on maximum weight but minimum redundancy. The weight of a feature denotes its importance, and by using this method, we can find the subset which is most beneficial and also least correlated. Hall and Smith propose a new FS method hinged on another correlation based heuristic that can be used for the selection of a proper subset [HS98].

A method named INTERACT is proposed in [ZL09] where the feature interaction is taken into consideration. Certain features may not be very relevant to the target when considered separately but might be extremely important when considered with other features. This dependency on other features is the concept of feature interactions. Irreducibility is an intrinsic character of feature interactions, which

is not considered by most FS algorithms. A FS algorithm to improve the efficacy of microarray classification is proposed in [LV11]. The proposed approach uses the Kruskal Wallis test on auxiliary data and then goes on to rank the features on the basis of their aggregated p-values. In [NHCD10], an FS technique based on norm minimization using $l_{2,1}$ is proposed. The joint norm minimization is applied to both loss function and regularization. Multiclass microarray data is a domain where FS is being used very widely. For the same purpose, in [SF12] the author proposes a model which makes use of Partial Least Squares and a decomposition technique. This model is applied to sets of two class subproblems, one versus one and one versus rest.

Song *et al.* presented a new model [SSG+12], using the Hilbert Schmidt Independence criterion, which is a nonparametric dependence measure, just like mutual information. In [NM09], the author proposes a new method for subset selection in microarray data with the help of entropic filtering algorithm. This method can be used to find attribute subsets that increase the normalized multivariate conditional entropy when considered together for problems related to classification. Meyer *et al.* [MSB08] proposed a model for FS associated with microarray data specifically for a huge number of attributes and only a few samples. The method makes use of Double Input Symmetrical Relevance, an information-theoretic selection which is based on variable complementarity. A greedy FS technique is proposed in [HBK14], where mutual information is used to evaluate feature-feature and feature-label information. NSGA is used to choose the optimal feature subset.

A comprehensive study of various statistical methods like Pearson's Coefficient and Correlation Criteria that are used to filter data, and their mathematical implementations are described in [GE03, TAL14, Nov16].

7.2.2 Wrapper Based Approach

We discuss several wrapper techniques that have been presented. In [MG71], the author proposes seven techniques to pick an optimal set of features. The first method uses expected probability of error. The second method chooses more features with minimum correlation using the initially picked features. The third approach is to check which feature can accurately distinguish two classes, pick the feature and repeat. The fourth is to perform Principal Component Analysis. The fifth is a small modification of the fourth, omitting those with smaller contributions. The sixth method chooses the features that make the most significant contributions to the eigenvectors. The seventh method is a mixture of the first two. In [Cha73], a dynamic approach to feature subset selection is proposed. A branch and bound solution is proposed for the same FS problem in [NF77]. The Particle Swarm Optimisation (PSO) technique is proposed for the same problem by Kennedy *et al.* in [SE99]. The method of Sequential Forward Selection, was proposed by Whitney [Whi71]. The algorithm begins with an unfilled subset and adds one feature at a time after gauging them. In [PNK94], the author proposes a method to perform Sequential Floating Forward Selection in which backtracking is used to exclude variables. An improvement to this method is proposed in [NC09]. An extra step is added to replace the weakest feature in the currently selected subset and then check whether removing that feature and adding another proves to be beneficial.

In [APST05], the author proposes a two-stage method, the first of which involves reducing the prediction error over a monitoring dataset. The second stage is comprised of a simulated annealing technique to lessen the number of explanatory attributes. The use of a Genetic Algorithm (GA) to conduct feature subset selection was initially proposed by many, but notably by [YH98, PIGP$^+$93]. FS, by extending the GA, has been done by [Esh91] and [CDS06]. In [ISBL02], the author proposes a

method to conduct Sequential Forward Selection (SFS) for microarray cancer class prediction. The author proposes a wrapper method [RRAR06] to rank the features and then select them. Incremental Ranked Usefulness is used for the ranking process. Sharma *et al.* propose a new method to select features that may perform weakly when considered individually but work well with other genes [SIM11]. The genes are divided into a small subset of size h, and then further divided into smaller informative subsets of size r. These smaller subsets are iteratively merged into a bigger, more informative subset of features. The author presents a method based on Kernel Density Estimation [WGNdPB13], a nonparametric estimator, used to select the feature subset. Non-parametric estimators work well for sparse and scarce data, especially in the field of Bioinformatics.

In [CS96], the author proposes a method called SET-Gen, to create multiple feature subsets, with the help of a GA, along with a wrapper evaluation function. They are then evaluated using 10 fold cross validation. Caruana and Freitag [CF94] and John, Kohavi and Pfleger [JKP94] evaluate several wrapper methods, which make use of hill climbing, like SLASH, Backward Stepwise Elimination, Forward Selection, Backward Elimination and Forward Stepwise Selection. The author proposes two methods based on the the random mutation hill climbing algorithm and the Monte Carlo sampling algorithm [Ska94]. In [RPN+14], a novel method of FS is proposed, which makes use of the Bat Algorithm and the Optimum-Path forest. In [EZH16a], the author proposes two novel algorithms. The first is based on the ant lion optimization operators, and the second is based on using the continuous steps of the same, as thresholds, after squashing them. In [SP13], an algorithm employing the Artificial Bee Colony and a perturbation parameter is presented. Mafarja *et al.* propose two methods based on Whale Optimisation Algorithm (WOA) [MM18]. The Tournament and Roulette Wheel selection mechanisms are used in the first

approach, and Crossover and Mutation algorithms are used to improve the Whale Algorithm in the second technique.

In [MAH+18], a Binary Dragonfly optimization is proposed for FS with the help of time-varying transfer functions. In [EZH16b], the author proposes a new binary version of the Grey Wolf Optimisation technique, which is implemented for FS. Yang *et al.* propose an ensemble based wrapper method for FS [YLZ+13], specifically for imbalanced class distribution. Chaouki and Saoussen [KK17] propose a method of FS for intrusion detection systems, using the wrapper method, enforced with the GA method. Gang and Jin Chen [CC15] propose a method of using wrapper methods with Support Vector Machines (SVM), namely Cosine Similarity Measure SVM to remove the unnecessary features. In [LLWS14], another SVM based technique is proposed, in which a statistics based wrapper is used in unison with the SVM for Financial Distress Identification. Lei *et al.* propose a method of FS for object based image classification [MLG+17]. Their model uses a novel wrapper technique with the help of polygon based cross validation.

7.2.3 Hybrid Approaches

A hybrid method of classification, which uses Modified Information FS and Modified Binary Cuckoo Search is proposed in [JLYX17]. In [JBS10], a new method, namely class dependent density based feature elimination is proposed. It uses a measure called diff-criterion to rank the features and then perform a feature subset selection on the ranked features. In [LT13], the author proposes a new method of FS using a combination of SVM, Recursive feature elimination and normalized mutual information. In [SMQ+14], the author proposes a method to perform FS with a combination of SVMs and Minimum redundancy maximum relevance (mrMR). Zhu,

Ong and Dash propose another method, which integrates a feature ranking system in the traditional GA [ZOD07]. In [YMZ17], the author proposes a method of FS, on the basis of ranking them initially, and then selecting a subset of attributes. The feature ranking is done on the basis of the AUC of their decision tree model. The features are then selected based on a new logical algorithm.

In [MM17], the authors give us another method for FS using WOA; this time, a hybrid model. The model is based on WOA combined with simulated annealing. In [HBXC15], Hu *et al.* propose a method to select features for short term load forecasting. They implement a filter method, Partial Mutual Information followed by the firefly algorithm, which is the wrapper portion. Basant and Namita propose a method of FS [AM13], on the basis of Rough Set Theory and Information Gain, which is then applied for Sentimental Analysis. A hybrid PSO is proposed in [MG16] by developing a new local search technique and has been named HPSO-LS. The authors of [ZYL14] propose a new technique to extract features by building a hybrid of SVM and K-Means algorithms. A new FS algorithm called TRSFFQR is developed and proposed in [J$^+$16]. TRSFFQR, which stands for Tolerance Rough Set FireFly based Quick Reduct, is used for FS in MRI Brain Images classification. The techniques that have been applied are evident from the name.

Two new algorithms, PSO Relative Reduct and PSO Quick Reduct, have been proposed as FS algorithms for medical datasets in [IAJ14]. A thorough method of FS is proposed in [TA15], where Weighted Least Squares Twin SVM is used as a classification technique. SFS is used as the search strategy, and finally, correlation FS is used to gauge the weight of every attribute. In [FD19], Faker and Dodgu propose a hybrid method of ranking the features by applying a clustering algorithm followed by validating the clusters using homogeneity and then selecting a subset from the ranked features. A new approach, which makes use of the ReliefF algorithm

followed by optimal feature subset selection with the help of SVM, is presented in [ZZC+18]. Wang and Feng [WF18] proposed a method in which two feature subsets are created using two optimal filters. A union operation based on feature weights is developed to consolidate the two subsets. High quality clusters can be produced with a hierarchical agglomerative clustering algorithm without requiring the cluster number.

7.3 Proposed Approach

Here, we explain the necessary background to understand our proposed algorithms and then explain them in detail.

7.3.1 Preliminaries

Here, we describe the various algorithms and metrics that we make use of in the proposed algorithms. In the feature ranking step of MRFI, we make use of K-Means and V-Measure. We make use of Random Forests for the feature ranking stage of ARFI and the entire feature subset selection stage.

K-Means

K-Means falls under the category of unsupervised machine learning and is a clustering algorithm. It is used to segregate samples into the best suited group on the basis of the information already available to the algorithm. K unique clusters or groups are created such that they are sufficiently far apart from each other spatially. The distance is measured in Euclidean Distance so that clear and valid results are rendered when information is mined from them. Centroids are the centers of clusters,

and data is iteratively categorized into clusters based on a data point's distance from the centroids. The most optimal solution for all the points is found iteratively as:

1. K data points are randomly chosen as centroids.
2. The distance between every point in the data set and the K randomly chosen centroids are calculated.
3. Each point is allocated to the closest cluster, on the basis of the distances calculated.
4. Centroids are reassigned by finding the average of all data points in a cluster.
5. If the centroid changes, then the process is redone from the step where the centroids are calculated until all the centroids remain the same. The clustering is complete when the centroids do not change their positions.

Mathematically, K Means seeks to reduce the squared error (objective) function. It is described below:

$$J = \sum_{a=1}^{m} \sum_{b=1}^{n} (||x_a - v_b||)^2 \tag{7.1}$$

Where, $||x_a - v_b||$ is the Euclidean Distance between a centroid, v_b, and a point, x_a, iterated over m points in the a^{th} cluster, for all the n clusters [Mac67].

V-Measure

It is used to evaluate external clusters based on conditional entropy. It measures the goodness of the completeness and homogeneity of a cluster. Their harmonic mean is

the V-Measure score of a cluster. Homogeneity of a cluster is satisfied when all the samples of a cluster are in the same, unique class. Completeness is satisfied when all the data points belonging to a single class are a part of the same cluster.

For a mathematical definition, let us consider a dataset comprising of N data points. Let these data points be partitioned into some classes, $P = \{p_x | x = 1, \ldots, m\}$ and some clusters, $Q = \{q_y | i = 1, \ldots, m\}$. The contingency table is denoted as T. This table represents the clustering solution, such that $T = \{t_{xy}\}$. Here, t_{xy} symbolizes the number of samples that are elements of the cluster q_y and members of class p_x. Let homogeneity and completeness be represented as H and C respectively [RH07]. Then V-Measure is given by:

$$V_\beta = \frac{(1+\beta) \times H \times C}{(\beta \times H) + C} \tag{7.2}$$

Homogeneity, H, can be defined as:

$$\begin{cases} 1 & if F(P,Q) = 0 \\ 1 - \frac{F(P,Q)}{F(P)} & else \end{cases} \tag{7.3}$$

where,

$$F(P,Q) = -\sum_{q=1}^{|Q|} \sum_{p=1}^{|P|} \frac{t_{pq}}{N} log \frac{t_{pq}}{\sum_{p=1}^{|P|} t_{pq}} \tag{7.4}$$

$$F(P) = -\sum_{p=1}^{|P|} \frac{\sum_{q=1}^{|Q|} t_{pq}}{m} log \frac{\sum_{q=1}^{|Q|} t_{pq}}{m} \tag{7.5}$$

Completeness, C, can be defined as:

$$\begin{cases} 1 & if F(Q,P) = 0 \\ 1 - \frac{F(Q,P)}{F(Q)} & else \end{cases} \tag{7.6}$$

where,

$$F(Q,P) = -\sum_{p=1}^{|P|}\sum_{q=1}^{|Q|}\frac{t_{pq}}{N} log \frac{t_{pq}}{\sum_{q=1}^{|Q|} t_{pq}} \tag{7.7}$$

$$F(Q) = -\sum_{q=1}^{|Q|}\frac{\sum_{p=1}^{|P|} t_{pq}}{m} log \frac{\sum_{p=1}^{|P|} t_{pq}}{m} \tag{7.8}$$

Random Forests (RF)

Random Forests can be classified under supervised learning. They are ML algorithms which use ensemble learning. In ensemble learning, we combine the same or different types of algorithms several times, to create a more robust prediction model. Random forests use multiple decision trees and are called forests for the same reason. They can be used for Classification and Regression.

The Random Forest Classifier randomly picks a certain number of features from the entire database. A decision tree is then built using these features. A large

number of trees are constructed in the same way, each selecting a random attribute subset of equal size. Once the forest has been created, each tree predicts the category to which the record belongs. The record is allocated to the class with most number of votes.

7.3.2 Metric Ranked Feature Inclusion (MRFI)

MRFI is a two stage process, the first of which is ranking the features, and the next stage is choosing the best attribute subset from the ranking. The ranking stage is carried out by employing K-Means and V-Measure. We split the entire dataset into testing and training datasets in the ratio of 4:1 with the standard scikit learn libraries [PVG+11]. Another dataframe to store the features in their ranked order is declared, with two columns, Name and Importance. The model selects a feature from the entire feature set, and K-Means clustering is performed, using only that feature and the target. The number of classes determine the value of K. After clustering the training data, we find the V-Measure Score of the clustering. V-Measure, being the harmonic mean of completeness and homogeneity, gives us a good understanding of the quality of the clustering. The obtained score is assigned as the importance of each feature. The entire process is carried out for each feature individually. The features, along with their importance, are then stored in the dataframe. That dataframe is then sorted to obtain a feature ranking, from most importance to least importance.

The feature subset selection stage is performed using the ranking of the features. We devise a novel algorithm for this process. The first feature in the ranking is taken, and then the accuracy of the random forest classifier is calculated. The next feature is combined with the other features in the optimal feature subset from

the feature ranking, and the accuracy of the same classifier is recomputed. If the accuracy increases, we retain the feature in the optimal subset. On the other hand, if the accuracy decreases, we drop the attribute from the optimal feature subset. This process is carried out iteratively for all the attributes, to obtain the final, optimal feature subset.

7.3.3 Accuracy Ranked Feature Inclusion (ARFI)

Just like MRFI, ARFI also involves ranking the features and then choosing the best attribute subset from those ranked features. To rank the features, a feature is taken, and the accuracy of the random forest classifier is computed. The importance of the feature is assigned with the obtained accuracy. We carry out this process for each feature, one at a time and then add each feature with its importance to a new dataframe called Features. This dataframe has two columns, Name and Importance. The dataframe is then sorted as per the importance of the attributes, in descending order.

The next phase is the same as the one used in MRFI.

This two stage process is followed to obtain the most optimal attribute subset. The relevance of every feature is computed by ranking them and this helps in picking the most important features. Our feature subset selection method is also a novel algorithm to choose attributes from the feature ranking. This subset selection method helps us to pick features with lesser redundancy, as it evaluates the subset with the classifier to check the performance. Often, some features may not be highly relevant when considered individually but may perform really well when considered in unison with other features. Our approaches take these factors into account and

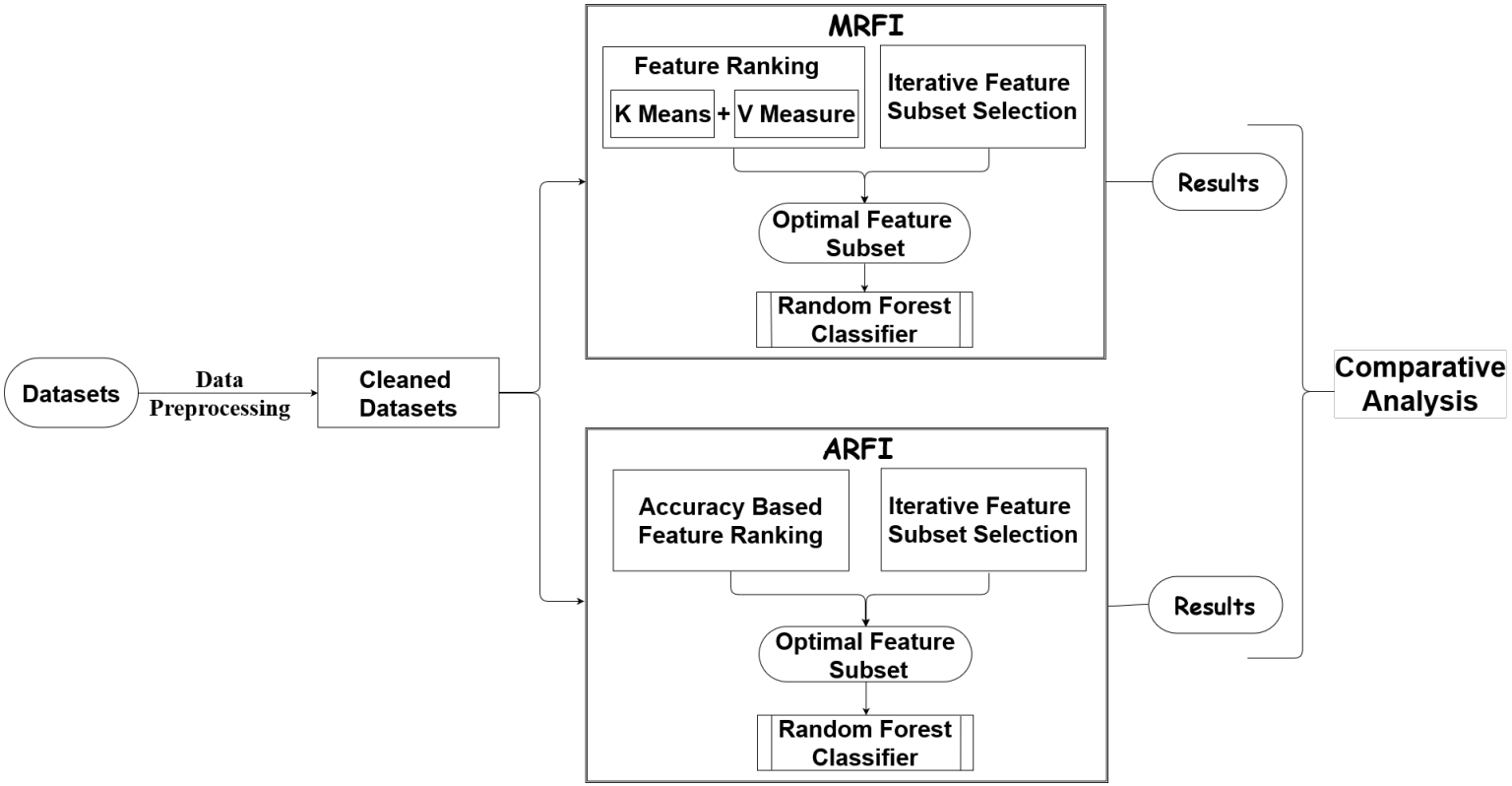


Figure 7.1: Flowchart Depicting Proposed Approach

give us the subset where the features perform extremely well together and are not very redundant. We present the diagrammatic flow of our proposed approach in Fig. 7.1.

7.4 Experiment

In this section, we give the hardware description for our experiment. We also give details about the datasets used and how we cleaned them.

7.4.1 Experimental Setup

All the experiments have been implemented in Python. We made use of an Intel i7 8 Core CPU which has a 16GB RAM and the Flounder Server (AMD Opteron Processor 6380 with 64 cores and 504GB RAM).

Dataset	**Feature Count**	**Class**
UNSW - NB15	47	Binary, Multiclass (9)
Abalone	8	Multiclass (28)
Avazu	16	Binary
Breast Cancer	10	Binary
Criteo	39	Binary
Heart Disease	13	Multiclass (5)
Ionosphere	34	Binary
Iris	4	Multiclass (3)
Lung Cancer	56	Multiclass (3)
Lymphography	18	Multiclass (4)
Talking Data	9	Binary

Table 7.1: An Outline of the Various Datasets used in our Experiments

7.4.2 Datasets

For our experiments, we make use of 11 datasets in total. Firstly, we use the UNSW-NB15 Dataset [uns15], a standard dataset for Intrusion Detection Systems. Three click fraud datasets, Avazu [Kag15], Criteo [Kag14] and Talking Data [Kag18] have also been experimented on. The remaining datasets are standard, benchmark datasets which are available in the UCI Machine Learning Repository [DG17]. This repository is a storehouse of databases, created by David Aha and other graduate students from UCIrvine. We make use of the Abalone, Breast Cancer, Heart Disease, Ionosphere, Iris, Lung Cancer and Lymphography datasets to evaluate and validate our models. The UNSW dataset can be used as a binary dataset and a multiclass dataset. The Abalone, Heart Disease, Iris, Lung Cancer and Lymphography datasets fall under the category of multiclass datasets, whereas the Breast Cancer, Ionosphere, Avazu, Talking Data and Criteo datasets fall under the category of binary datasets. A few details about the number of features, and the various types of datasets considered, are shown in Table 7.1.

7.4.3 Data Preprocessing

UNSW - NB15

The UNSW dataset initially has 2540047 rows. We use 43 out of the 47 features for classification. There are two label columns, 'attack_cat' for multiple classification and 'Label' for binary classification. There are 9 types of attacks which are considered for multiclass classification. We use the entire UNSW dataset, for which we append the four datasets given in [85]. Then, we manually assign the column names mentioned in the information file of UNSW dataset. We drop the first four columns, 'scrip', 'sport', 'dstip' and 'dsport', as they are just identification numbers

and are not significant. The column 'attack_cat' is used to perform multiclass classification. All the NaN values in the 'attack_cat' column are dropped to retain only the attack types. The remaining NaN values in the dataset are filled with zeroes as they are all numerical values which represent some count. The same classes occur multiple times in the 'attack_cat' column, with different names and white spaces. These white spaces are stripped, and the names are standardized. The 'ct_ftp_cmd' column has string representations of numbers. We convert them back to numbers and encode the 'service', 'proto' and 'service' columns. We normalize the dataset using the Standard Scaler. For the binary classification using the same dataset, we do not consider the 'attack_cat' column and all the remaining NaN values are dropped. The remaining preprocessing of this dataset is carried out in the same way as that of the multiclass classification version.

Abalone

The Abalone dataset has a total of 4177 entries, categorized into 28 classes. All 8 features are used. We perform encoding to convert the column with genders into numbers.

Avazu

The Avazu dataset is a click fraud dataset recorded over 10 days. We split the column called 'hour of click' into three separate columns. The class ratio of the 200 million rows of the original dataset is maintained when the rows are reduced to 1 million rows. The original dataset consisted of 16 features and one label column.

Breast Cancer Wisconsin (Original)

This consisted of 699 rows before the empty values were dropped. There are 10

features that we make use of, along with one target column. We replace the string representation of numbers with the actual numbers in some columns.

Criteo

The Criteo dataset is another click fraud dataset which we use to validate our models. It has 756554 rows and 39 features. We dropped all the rows which had NaN values.

Cleveland Heart Disease

The dataset providing information about Heart Diseases in Cleveland has 303 rows. It consists of 13 features and one target column with 5 classes. We dropped all the rows which contained undetermined values and replaced the string representations of all the numbers with the actual numbers.

Ionosphere

The Ionosphere dataset has 351 rows and 34 features that we use. We perform label encoding on the target column and then conduct our experiment.

Iris

The famous Iris dataset consists of only 4 features and only 150 rows. The rows are classified into three labels. We perform label encoding on this dataset. We also shuffle the entire dataset to get a good mix of all the classes.

Lung Cancer

Another famous dataset called the Lung Cancer dataset has been used. It is composed of only 32 rows of data but has 56 features. The number of rows is further

reduced after dropping the missing values. It has 3 classes into which the rows can be categorized. We run the classification algorithms after the data is shuffled to get a good mix of all three classes.

Lymphography

The Lymphography dataset is composed of 148 rows and has 18 features. The label column has 4 classes. We perform random shuffling to get a uniform distribution of classes to help the machine learn effectively.

Talking Data

Talking Data has a million rows, and they are denoted with 9 features. The column called 'attributed time' is dropped because it consisted of a large number of NaN values. The attribute 'click time' composed of the time-stamps, is split into four new attributes: 'day', 'hour', 'minute', and 'second'. We randomly select 1 million observations from around 200 million, but the class ratio is kept constant [TKC+19].

7.5 Results and Discussions

We present our results and make comparisons with previous work after giving details about the classifier, the various metrics used, and our method of analysis.

7.5.1 Base Classifier

We use Random Forests as our base classifier to carry out multiclass and binary classification. A thorough working of the random forest classifier has been described in Section 7.3.

7.5.2 Evaluation Metrics

To gauge the efficacy of our classifier, we employ specific metrics, namely, Recall (Rec), Accuracy (Acc), Precision (Prec) and F1 Score. Furthermore, we evaluate the AUC score for binary classification. These evaluation metrics make use of: T_{Posi}, which represents the correctly predicted positives values, T_{Negi}, which describes the correctly obtained negative values, F_{Posi}, representing the wrongly obtained positive values and F_{Negi} describing the wrongly predicted negative values [exs16]. The metrics, as mentioned above, are computed with the formulae given below:

$$Acc = \frac{T_{Posi} + T_{Negi}}{T_{Posi} + F_{Posi} + F_{Negi} + T_{Negi}} \tag{7.9}$$

$$Prec = \frac{T_{Posi}}{T_{Posi} + F_{Posi}} \tag{7.10}$$

$$Rec = \frac{T_{Posi}}{T_{Posi} + F_{Negi}} \tag{7.11}$$

$$F1 \quad Score = 2 \times \frac{Rec \times Prec}{Rec + Prec} \tag{7.12}$$

The Receiving Operator Characteristic (ROC) curve is a plot of the T_{Posi} rate against the F_{Posi} rate. These values are plotted for all possible cut-off values. A popular metric used to cross check the above metrics and avoid overfitting and underfitting, is the Area Under the Receiving Operator Characteristic Curve (AUC). It can also be interpreted as the average T_{Posi} rate for all F_{Posi} rate.

7.5.3 Method of Analysis

To compare the above metrics and validate our models, we follow a standard order. After cleaning the dataset, we run the base classifier, i.e., the random forest classifier without any FS and record the results. Then, we run MRFI to obtain an optimal feature subset. We rerun the random forest classifier with this new feature subset and record the results. We conduct the same experiment with ARFI. Once our algorithms have been evaluated, we perform RFE on the original dataset and compute the above metrics. The RFE FS model ranks the features based on feature importances, and then recursively eliminates the worst feature according to the ranking. The feature elimination takes place after it evaluates the entire subset with the classifier. RFE is a wrapper method and has proven to be extremely efficient in the past. RFE requires an external parameter which tells it how many features are to be considered. The parameter is given based on the number of features considered by our models. Next, we examine the results of the original dataset, our algorithms and RFE. Our models have performed exceedingly well, as can be seen from the results. We have presented them in the form of tables and plots below.

7.5.4 Discussions

Unlike RFE, our models do not need to know the number of features beforehand. Our FS algorithms iteratively add features and do not need a fixed, minimum or maximum number of features. Figure 7.3 depicts the same. Only feature subsets with a minimum accuracy of 83 percent have been shown in the figure. Each point denotes a feature subset that is being evaluated. As can be seen, subsets are evaluated immediately after their creation. Only if the performance of the learning model does not decrease, the most recently included feature is considered in the final sub-

Dataset	FSA	#features	A	P	R	F1
UNSW - NB15	None	43	89.260	89.620	89.260	88.890
	RFE	10	89.473	89.670	89.470	88.010
	ARFI	10	90.108	90.890	90.110	88.860
	MRFI	14	90.068	91.030	90.070	88.820
Abalone	None	8	24.521	23.050	24.520	22.890
	RFE	1	17.584	17.830	17.580	17.420
	ARFI	1	25.120	22.520	25.120	21.970
	MRFI	1	25.120	22.520	25.120	21.970
Heart Disease	None	13	46.667	34.350	46.670	39.310
	RFE	6	40.000	29.540	40.000	33.510
	ARFI	5	51.667	48.720	51.670	48.760
	MRFI	6	48.333	41.370	48.330	43.890
Iris	None	4	93.333	93.330	93.330	93.330
	RFE	2	93.333	93.330	93.330	93.330
	ARFI	2	96.667	96.920	96.670	96.580
	MRFI	2	96.667	96.920	96.670	96.580
Lung Cancer	None	56	50.000	37.500	50.000	40.000
	RFE	29	50.000	38.890	50.000	43.330
	ARFI	10	83.333	88.890	83.330	82.220
	MRFI	29	83.333	88.890	83.330	82.220
Lymphography	None	18	80.000	80.190	80.000	79.720
	RFE	11	86.667	77.330	80.000	78.460
	ARFI	6	93.333	94.290	93.330	93.390
	MRFI	11	90.000	90.050	90.000	89.920

Table 7.2: Experimental Results of Classification on Multiclass Datasets

Note: P: Precision, R: Recall, A: Accuracy (%), F1: F1 Score, FSA: FS Algorithm

set. The occurrence of multiple points of the same FS algorithm, along one vertical line, represents subsets that do not perform as well as their immediate predecessors. This happens due to redundancy. Even though the attributes are ranked by their relevance, the redundancy between them may cause the subset to underperform. ARFI and MRFI overcome this issue in the second stage of their algorithms. For the Avazu dataset, ARFI considers 8 features, whereas MRFI considers 9, as the addition of any more features reduces the accuracy of the learning model. From Table 7.2, it is clear that both MRFI and ARFI give us outstanding results for all

six multiclass datasets. Both the proposed models outperform RFE and even tend to improve the performance of the predictor.

When compared to each other, ARFI performs better than MRFI in three of the datasets, namely UNSW, Heart Disease and Lymphography. In the other three datasets, they both give similar levels of performance. From our results, it appears that ARFI considers lesser redundant features, as it always selects lesser or equal number of features compared to MRFI, and performs better with those features.

Dataset	**FSA**	**#features**	**A**	**P**	**R**	**F1**	**AUC**
UNSW - NB15	None	43	99.939	99.940	99.940	99.940	99.513
	RFE	19	99.894	99.894	99.894	99.894	98.923
	ARFI	19	99.924	99.924	99.924	99.924	99.261
	MRFI	29	99.918	99.20	99.20	99.20	99.920
Breast Cancer	None	10	99.270	99.280	99.270	99.270	99.057
	RFE	9	98.540	98.570	98.540	98.530	98.113
	ARFI	9	99.999	99.999	99.999	99.999	99.999
	MRFI	7	99.999	99.999	99.999	99.999	99.999
Ionosphere	None	34	94.366	94.370	94.370	94.370	93.238
	RFE	21	92.958	92.900	92.960	92.910	90.857
	ARFI	21	95.775	95.760	95.770	95.740	94.238
	MRFI	16	95.774	94.238	95.760	95.770	95.740
Talking Data	None	9	95.173	95.170	95.170	95.080	91.673
	RFE	6	94.788	94.780	94.790	94.680	91.068
	ARFI	3	95.121	95.150	95.120	95.010	91.360
	MRFI	6	94.223	94.180	94.220	94.110	90.401
Criteo	None	39	73.567	72.330	73.570	70.320	62.420
	RFE	3	67.043	63.480	67.040	63.850	55.903
	ARFI	3	70.270	67.670	70.270	67.210	59.381
	MRFI	3	70.140	67.290	71.140	65.750	57.628
Avazu	None	16	82.896	78.210	82.900	78.190	54.388
	RFE	8	82.993	78.080	82.990	77.510	53.137
	ARFI	8	83.328	79.260	83.330	78.130	54.040
	MRFI	10	83.295	79.170	83.290	78.280	54.322

Table 7.3: Experimental Results of Classification on Binary Datasets

Note: P: Precision, R: Recall, A: Accuracy (%), F1: F1 Score, FSA: FS Algorithm

Research Work	Classifier	Accuracy(%)
Tama & Primartha[PT17]	RF	95.50
	Multilayer Perceptron	83.50
Moustafa & Slay[MS15a]	Naive Bayes	79.50
	Expectation Maximisation	77.20
	Linear Regression	83.00
Belouch *et al.*[BEI17]	Naive Bayes	80.04
	RepTree	87.80
	Decision Tree	86.13
	Random Tree	86.59
	Artificial Neural Network	86.31
Salaf *et al.* [BHI18]	Naive Bayes	74.19
	RF	97.49
	Decision Tree	95.82
	SVM	92.28
Al et. al [AZAA17]	Deep Learning	98.99
Faker & Dogdu[FD19]	Gradient Boosted Tree	97.92
	RF	98.86
	Deep Neural Network	99.19
Our Work	**RF**	**99.94**
	RF + MRFI	99.92
	RF + ARFI	99.92

Table 7.4: Comparison of Binary Classification with Previous Works using UNSW-NB15 Dataset

For the binary datasets (Table 7.3), ARFI gives good results when used with the Breast Cancer, Ionosphere and Avazu datasets. The accuracy and the AUC of the classifier after applying ARFI fall for the UNSW, Talking Data and Criteo datasets. The reason for this can be explained after understanding the results proposed in [CHC+12]. When there is no additional informational being extracted with the help of FS, the evaluation metrics might not increase and may even get negatively affected. Furthermore, when the sample size is big enough, the classifier can get trained well enough to predict values more accurately on its own. The effect of FS also depends on the features and the degree of correlation between them. The classifier used can also affect the improvement in performance after applying FS,

as some datasets tend to perform better with particular classifiers. MRFI gives us similar results, as it performs well on the same datasets as ARFI.
ARFI tends to give us superior results when compared with MRFI for four datasets, considering the accuracy. In the Breast Cancer and Ionosphere datasets, they both render similar levels of accuracy. MRFI gives a better AUC value for the Avazu dataset but falls behind ARFI in all the other datasets.

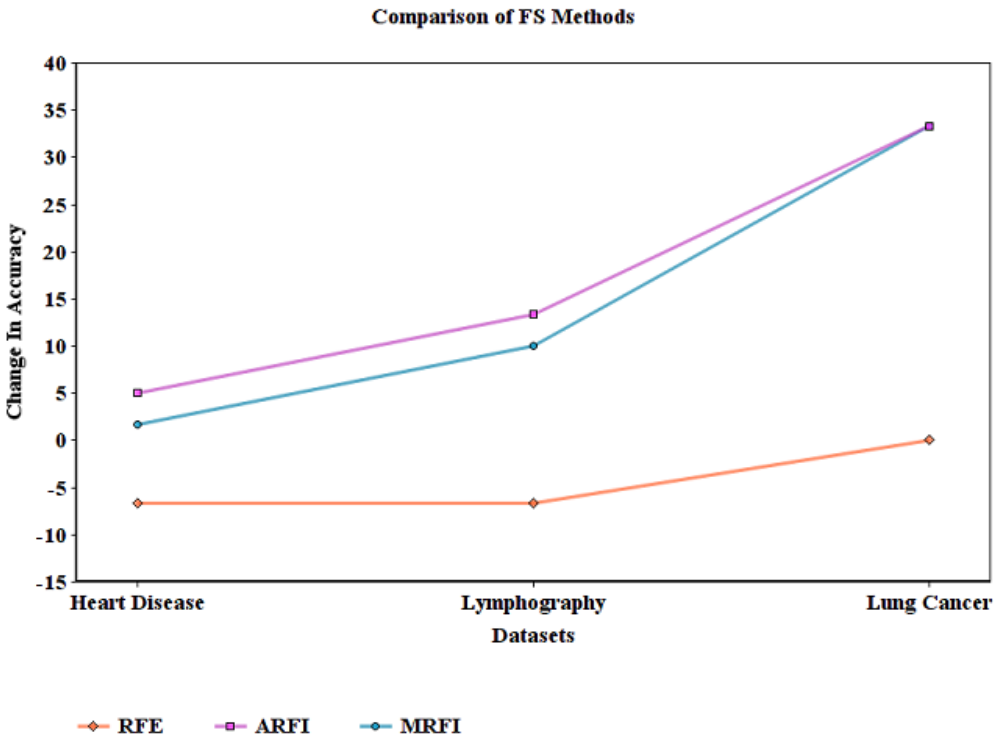
Figure 7.2(a) portrays the change in accuracy after performing FS on the Heart Disease, Lung Cancer and Lymphography datasets. The larger variations in accuracy are seen in this figure. On the other hand, Fig. 7.2(b) represents the smaller changes in accuracy observed on applying FS on the Iris, UNSW and Abalone datasets. From the plots, we infer that ARFI and MRFI perform considerably better than RFE, as the changes in accuracy are preferable.

Table 7.5: Comparison of Multiclass Classification with Previous Works using UNSW-NB15 Dataset

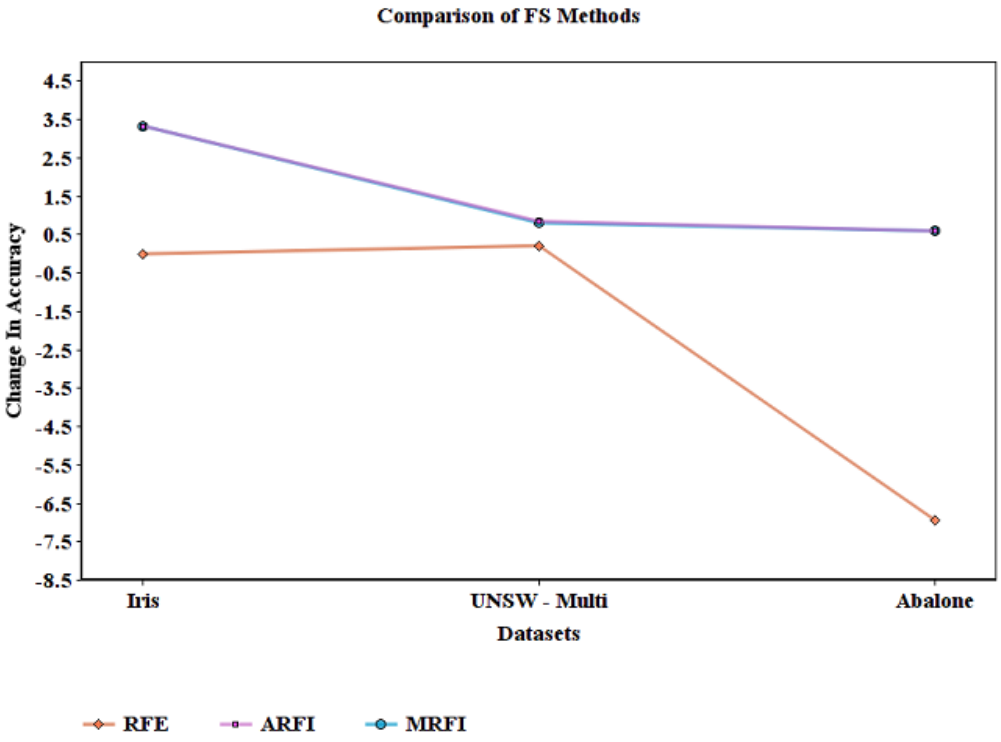
Research Work	**Classifier**	**Accuracy(%)**
Belouch *et al.* [BEI17]	Naive Bayes	73.86
	RepTree	79.20
	Artificial Neural Network	78.14
	Random Tree	76.21
Our Work	RF	89.26
	RF + MRFI	90.07
	RF + ARFI	**90.10**

Figures 7.2(c) and 7.2(d) depict changes in accuracy on performing FS on various datasets. It is evident that our models perform satisfactorily when compared to RFE for most datasets.

Both our models' results considerably outdo the results obtained after applying RFE. Now, we compare our models and their performance with other models previ-

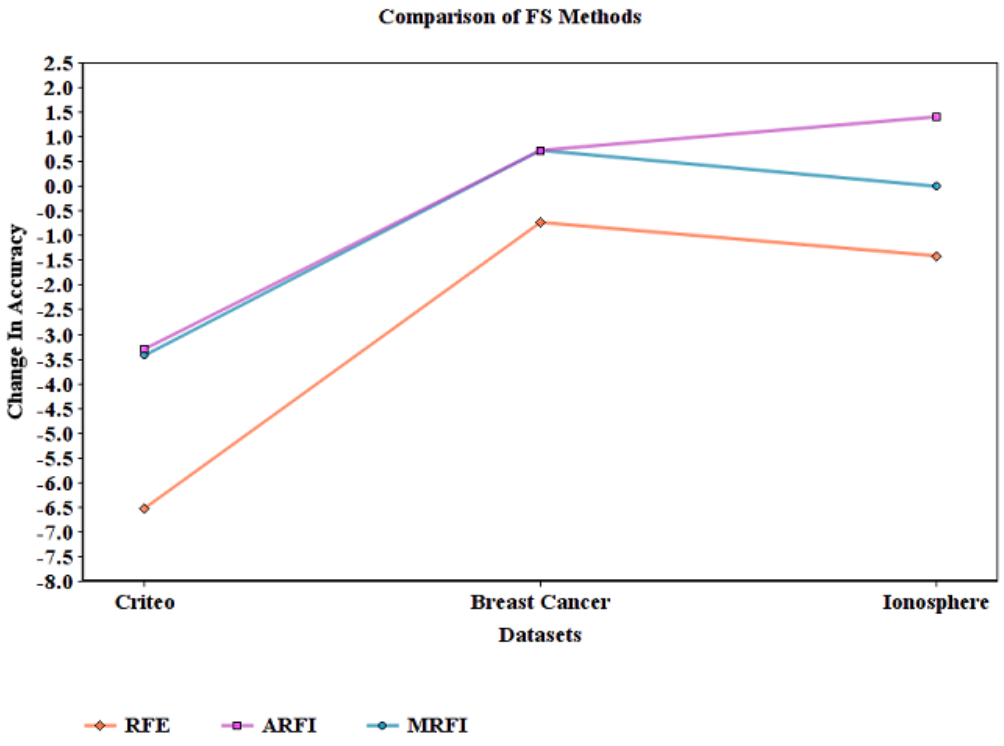


(a)

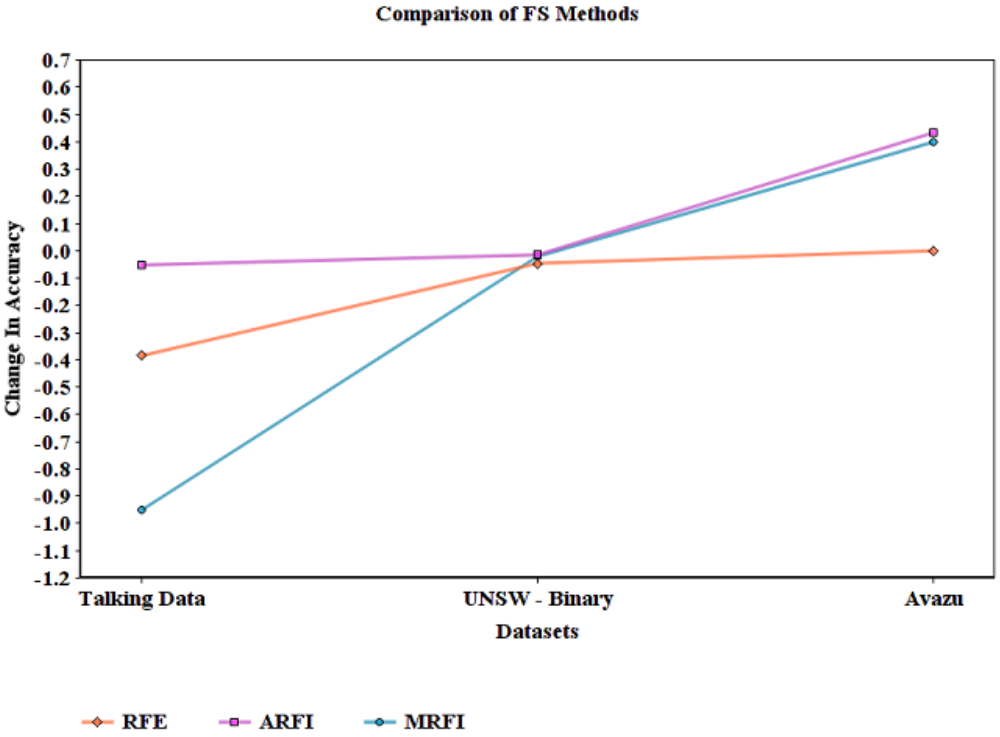


(b)

Figure 7.2: Changes in Accuracy for Different Datasets using Three Feature Selection Models. (a) Depicts Larger Changes in Accuracy after Applying FS on the Multiclass Datasets. (b) Depicts Smaller Changes in Accuracy after Performing FS on the Multiclass Datasets.



(c)



(d)

Figure 7.2: (Continued) Changes in Accuracy for Different Datasets using Three Feature Selection Models. (c) Depicts Greater Changes in Accuracy after Applying FS on the Binary Datasets. (d) Depicts Minute Changes in Accuracy after Performing FS on the Binary Datasets.

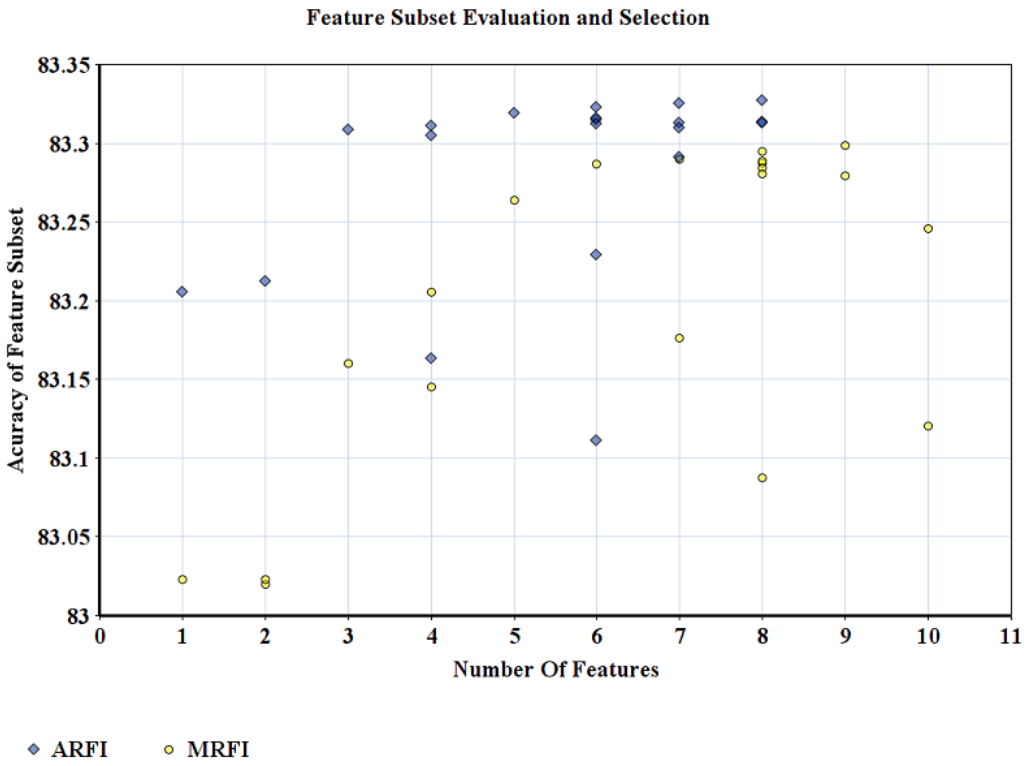


Figure 7.3: Scatter Plot of the Accuracy of Multiple Feature Subsets, that were Created by ARFI and MRFI, Depicting the Feature Subset Selection Procedure for the Avazu Dataset.

ously applied on the UNSW-NB15 dataset. We also compare them with the Talking Data and Ionosphere Datasets.

Table 7.4 compares various results for the UNSW dataset for Binary Classification. It compares the results on the basis of accuracy. Our model obtains the highest accuracy and gives the best performance. Another noticeable fact, is that our RF performs much better than other RF models that have been used before. This is traced to the method of preprocessing the UNSW dataset, including normalization, and the parameter tuning that we have performed on the RF.

Comparisons of our work with previous work concerning the UNSW dataset for multiclass classification have been shown in Table 7.5. Our model clearly outperforms the model proposed in [BEI17].

Table 7.6: Comparison of Binary Classification on Talking Data Dataset with Previous Works

Research Work	**Classifier**	**Prec**	**Rec**	**F1 Score**
Qiu *et al.* [QZL18]	SVM	0.896	0.877	0.876
	Naive Bayes	0.908	0.897	0.896
	ETCF	0.910	0.904	0.904
	GBDT	0.906	0.891	0.890
	RF	0.877	0.846	0.842
Our Work	RF	0.951	0.951	0.950
	RF + MRFI	0.941	0.942	0.941
	RF + ARFI	**0.951**	**0.951**	**0.950**

When compared to the previous work of Qiu *et al.*, our models have higher precision, recall and F1 scores. The dataset under consideration is the Talking Dataset. These results can be seen in Table 7.6.

Research Work	**Classifier**	**Accuracy(%)**
Liu & Zhang [LZ16]	KNN + LS	88.32
	KNN + FS	89.18
	KNN + CS	89.59
	KNN + Lasso	87.46
	KNN + CGS	91.32
Ghaemi *et al.* [GFD16]	J48	93.16
	3NN	92.30
	5NN	89.43
	RBF-SVM	94.58
	1NN	89.52
	J48	95.12
Our Work	RF	94.36
	RF + MRFI	**95.77**
	RF + ARFI	**95.77**

Table 7.7: Comparison of Binary Classification on Ionosphere Dataset with Previous Works

Table 7.7 portrays a comparison between the work done on the Ionosphere dataset. It is evident that both our models outperform most of the other FS models.

RBF-SVM and J48 give better results than our base classifier and MRFI, but ARFI outperforms both of them too.

7.6 Conclusion

Feature Selection is an essential tool that is used to select a feature subset using which the performance of the classifier can be improved. FS is vital as it reduces the training time of the model under consideration, reduces overfitting, and more importantly, avoids the curse of dimensionality. In this chapter, we present two new FS methods, MRFI and ARFI. Both the models are hybrids of filter and wrapper methods of FS. MRFI employs K-Means and V-Measure scores to rank the features, whereas ARFI ranks the features based on the accuracy of the predictor (Random Forests). Both our methods follow the same feature subset selection technique. We compare our models with Recursive Feature Elimination, a state-of-the-art FS technique, using 12 datasets. Furthermore, we gauge their performance with the help of previous work done on the same datasets. We observe that our models have performed well and have given superior results. The evaluation metrics improve drastically, and the accuracy of the random forest classifier also increases considerably, thereby overcoming the drawbacks of the current FS algorithms.

CHAPTER 8

MINI-BATCH NORMALIZED MUTUAL INFORMATION: A HYBRID FEATURE SELECTION METHOD

Feature Selection has been a significant preprocessing procedure for classification in the area of Supervised Machine Learning. It is mostly applied when the attribute set is very large. The large set of attributes often tend to misguide the classifier. Extensive research has been performed to increase the efficacy of the predictor by finding the optimal set of features. The feature subset should be such that it enhances the classification accuracy by the removal of redundant features. We propose a new feature selection mechanism, an amalgamation of the filter and the wrapper techniques by taking into consideration the benefits of both the methods. Our hybrid model is based on a two phase process where we rank the features and then choose the best subset of features based on the ranking. We validated our model with various datasets, using multiple evaluation metrics. Furthermore, we have also compared and analyzed our results with previous works. The proposed model outperformed many existent algorithms and has given us good results.

8.1 Introduction

One of the essential phases in classification is to determine the useful set of features for the classifier. In supervised as well as in unsupervised learning, the large volume of data has become a significant problem and is becoming more prominent with the increase in data samples and the number of features in each sample. The main

intention of reducing the dimension by keeping a minimum number of features is to decrease the computation time, obtain greater accuracy, and reduce overfitting.

Dimensionality reduction is divided into 2 categories: Feature Extraction (FE) and Feature Selection (FS). In FE, we transform the existing features into new features with lesser dimensionality employing a linear or a nonlinear combination of features. In this method, the actual data is manipulated and hence not immune from distortion under transformation. In the FS process, we select the feature's subset based on some criteria. Many of the attributes in the dataset may be utterly irrelevant to the class or redundant when considered along with other features. The accuracy of the induced classifier is decreased by the presence of irrelevant or redundant features [JKP94]. Identifying such features and removing them reduces the dimensionality which inturn reduces the computation time while improving the accuracy. In [Kus99], they state that the overabundance of features rendered the nearest neighbor approach on Internet Advertisement dataset.

FS has many applications in various fields like image processing, natural language processing, bioinformatics, data mining, and machine learning(ML) [HBK14]. The selection method is divided into two standard categories based on their working modules, classifier independent 'filter' technique, and classifier dependent 'wrapper' and 'embedded' technique.

The filter technique, a classifier independent process, performs the selection of the features based on statistical metrics like distance, correlation, consistency measure, and mutual information (MI). It either ranks the features or provides a relevant subset of features associated with the class label. It improves the computational efficiency and scales down the data dimensionality by entirely being independent of the classifier [SIL07]. The drawback of this process is the lack of knowledge regarding the relationship between feature attributes and target class.

The classifier dependent systems rely upon the classifier for the selection process. The wrapper method uses the outcome of the classifier to obtain the subset of features, making it biased to a classifier. Also, it is vulnerable to overfitting, mostly when the quantity of data is very less[BPZL12]. The embedded method makes use of the classifier in the training phase and selects the optimal features like a learning procedure. When compared to the wrapper method, they are less vulnerable to overfitting and computation is much faster[LPd+12].

8.1.1 Summary of Contribution

We propose a combination of filter and wrapper method, which has the advantage of both the techniques. It is fast and general like the filter method. At the same time, it accounts to learning algorithm obtaining the best set of features without the need for the user to input the feature number unlike most of other established algorithms like RFE.

In this work, we cluster the data using mini-batch Kmeans clustering and rank them using normalized mutual information(NMI), a measure to calculate the relevance and the redundancy between the candidate attribute and the class. We apply a greedy search method by using Random Forest to get the optimal set of features. However, our method is flexible in terms of the learning algorithm that can be used in our process.

8.1.2 Organization of the Chapter

Section 8.2 discusses the related works regarding various standard techniques as well as different hybrid approaches. In section 8.3, we discuss the preliminary concepts

behind this work, propose our techniques and we elaborate each component of our work in detail. In section 8.4 and 8.5, we show experimental results and compare them with the previous works, and in Section 8.6, we conclude our work and give light to the future works.

8.1.3 Abbreviations and Acronyms

All Features(AF), Feature Selection (FS), Feature Extraction (FE), Mini Batch K-Means Normalized Mutual Information Feature Inclusion(KNFI), Mini batch K-Means Normalized Mutual Information Feature Elimination (KNFE), Normalized Mutual Information (NMI), Random Forest(RF),

8.2 Related Work

Filter Method

In [GE03], Guyon *et al.* give information about all the developments to improve the performance of the model using statistical analysis. They have come up with a simple approach where less computation is required. It does not consider the dependency between the features but considers each feature as an independent one. In [SIL07], Saeys *et al.* show that the various FS methods have shown impressive results in the field of bioinformatics. Using Weka tool, Pushpalatha *et al.* [PK17] perform CFS based filter approach to rank with five search techniques. In [DCSL02], Dash *et al.* choose the best feature subset for clustering by evaluating the various subsets of features. It considers the effect of the underlying clusters with no unanimous agreement in evaluating the clusters.

Wrapper Method

In [GCJ18], they have a variant of particle swarm optimization (PSO) to determine

the least number of features which results in finer classification. It is a wrapper based technique named as competitive swarm optimizer (CSO). Also, they have proposed an archive technique to reduce computational cost. In [JKL19], they optimized the multi objective function of a pre-existing wrapper method to a single objective function to reduce the computation cost by adding a new evaluation function. In [MM18], they introduce a new wrapper method which is mainly based upon Whale Optimization Algorithm (WOA), with slight changes to make the model work even for binary datasets. In [XYW18], they performed feature selection with genetic algorithm and extreme machine learning, which is computationally efficient in comparison with other wrapper methods.

Hybrid Method

In [VA19], Venkatesh *et al.* came up with a hybrid approach of filter and wrapper method by considering MI and RFE. In [SSA+19], Sharmin uses MI as a metric for creating a framework for selecting features and discretization at the same time based on x^2 test. In [HBK14], they consider the information between the attributes and the classes. They have considered the MI of the candidate attribute with all the attributes in the selected set of feature attributes. Genetic algorithm is used to select attributes that increase the MI with the label class and decrease the MI with other feature attributes. In [Bat94], they introduce Mutual Information Feature Selection (MIFS), an incremental greedy search method to select the most likely 'k' features among n features. Instead of calculating MI between the attributes and the classes, they calculated MI between the attributes i.e., the previously selected attributes and the set of candidate attributes. The performance tends to degrade if there are significant errors in estimating MI. In [KC02], they proposed an improvised method of MIFS to improve the estimations of MI between the class labels and the input attributes called MIFS-U. In [PLD05], they proposed a method mRMR which

avoids expansion of subset where the redundancy divides over the cardinality $\|C\|$ of the selected subset. In [BPZL12], they have justified that this alteration allows mRMR to outperform the established MIFS & MIFS-U techniques. In [ETPZ09], they normalized the value of MI to curb down the value between zero and unity, which removed the bias towards multivalued features. The proposed approach of normalizing the value of mutual information in the FS process, namely NIMFS, which is as an upgraded model of mRMR, MIFS-U, and MIFS to find the irrelevant and redundant features. They also proposed genetic algorithm based feature selection process. In [HGV11], Haury *et al.* give the comparative analysis of FS method based upon stability and interpretability of the classes. It suggests that a simple filter method outperforms more complex embedded and wrapper method. In [FD19], they have considered homogeneity metric as a measure to rank and remove the least ranked features. In [ZXM19], they proposed a hybrid filter and wrapper method where they created a subset of features with bootstrapping strategy. For each subset, classification accuracy is calculated to find the optimized subset.

8.3 Proposed Approach

8.3.1 Preliminaries

Mini Batch K-means Method

K-means is one of the popular clustering algorithms. With the increase in dataset size, the computation time increases as the entire data needs to be present in the main memory[AV06]. Because of this, we prefer Mini Batch K-means for large datasets. We intend to apply a fixed size of small random batches of data for easy

storage in the memory. In every iteration, the cluster is updated taking new random samples from the dataset. For a given dataset $D = x_1, x_2, x_3, ..., x_p, x_i \in R^{m*n}, x_i$ represents the records in an n-dimensional real vector. The number of records in dataset D is 'm'. We obtain a set S of cluster center $s \in R^{m*n}$ to decrease over the dataset D of records $s \in R^{m*n}$ as shown in the following function.

$$Min \sum_{x \in T} \|f(S, x) - x\|^2 \tag{8.1}$$

Where, f(S,x) yields the nearest cluster center $s \in S$ to record x. If K is the number of clusters, it is given by k =$|S|$. We randomly select K records by using Kmeans++ to initialize the centers and we set the cluster centers S to be equal to the values of these. In our case, we have considered the number of clusters equal to the number of class. When the data is huge, the convergence rate of the original Kmeans significantly drops. In this case, an improved K-means called Mini Batch Kmeans is introduced [Scu10].

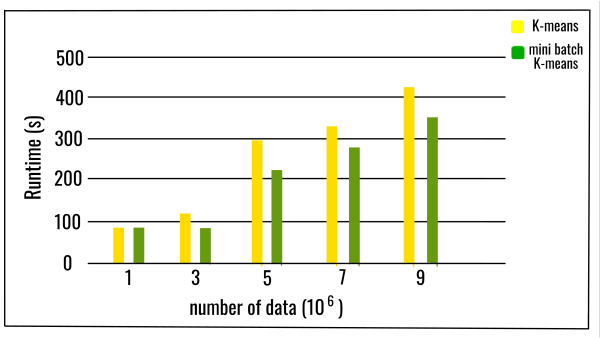


Figure 8.1: Runtime Analysis of K-Means and MiniBatch K-Means.[gee19]

Normalized Mutual Information (NMI)

NMI is one of the ways for measuring the criteria of cluster quality, which is information-theoretic interpretation. This measure calculates the cluster quality

with cluster number. Mathematically :

$$NMI(\Omega, S) = \frac{MI(\Omega; S)}{[G(\Omega) + G(S))]/2} \tag{8.2}$$

where Ω is the set of clusters and S is the set of classes. Here MI is given by the formula:

$$MI(\Omega; S) = \sum_{k} \sum_{j} P(d_k \cap s_j) log \frac{P(d_k \cap s_j)}{P(d_k)P(s_j)} \tag{8.3}$$

where $P(d_k)$ =probability of document in cluster d_k ,

$P(s_j)$= probability of document in cluster s_j,

$P(d_k \cap s_j)$=probability of document being in the convergence of d_k and s_j.

NMI increases the knowledge of the class by evaluating the amount of information obtained from the clusters. The value is 0 when the clustering is random concerning the class and gives no knowledge about the class. MI reaches maximum value if it perfectly recreates the classes. G is the entropy. Mathematically :

$$G(\Omega) = -\sum_{k} P(d_k) log P(d_k) \tag{8.4}$$

This gives the entropy of cluster levels. The normalization in Eqn. 8.2 by the denominator solves the problem of purity. It also formalizes that fewer clusters are better since the entropy usually increases with the increase in cluster number.[CDM08] The value of NMI is always between 0 and 1.

8.3.2 Our Approach

Here, we present our proposed hybrid filter-wrapper approach for the FS. There are two objective functions in our FS. First, the feature ranking function based on the filter approach and second, the selection of optimal features based upon the rankings. This optimal selection is a wrapper based method that depends upon the

outcome of the learning algorithm. Our approach is independent of any number of a class labels and is suitable to use with any classifier. In our experiments, we have considered Random Forest as the classifier. However, we can use any classifier. Our approach has 2 phases;

Feature Ranking

In the first phase, the main idea is to separately cluster the features one by one based upon the total classes in the dataset. Our objective is to have a selection algorithm which takes less computation time in comparison to the existing algorithms. Since the data are large these days, we have considered mini-batch K-means, which takes into account a batch of data and performs clustering. The computation time, in this case, is much lesser than the normal K-means clustering. The cluster's quality is the metric to find the relation of that feature with the class. As the cluster quality increases, the feature tends to be more relevant and is considered to be more important. The use of NMI gives a cluster score between 0 to 1. The high ranking score indicates better classification using the candidate feature. The cluster score for all the features is evaluated separately. Comparing the score of each feature, we obtain the ranking list. This ranking obtained is based upon the individual relationship between the candidate attribute and the class label.

Feature Selection

In the FS problem, a feature variable may have a dependency on other variables. The dependent features tend to produce imbalanced results when acted upon together and hence, is considered a redundant feature. The redundant feature tends to deteriorate the classification process, and we remove those in our process. We considered the ranking obtained from the first phase as the base for the selection

of features. We consider this to have a linear approach of selecting the features to get the optimal features in minimum time. When the feature size in the dataset increases, comparison with all the possible subsets is an impractical approach and seems to be computationally very expensive. We present two approaches for the selection of features. They are:

1. Feature Inclusion: This is almost a linear selection approach where the ranked features from phase one are added one by one into the subset. If the addition of the features enhances the classification accuracy, we consider the feature or else we discard the feature. Here, the highest ranked feature is initially included in the list as shown in step one of Algorithm 4. We add the next ranked feature and obtain its performance. If the performance increases, we add the feature into the list or else discard the feature. The feature is removed if it does not perform well with the selected subset, considering that it is redundant as it degrades the classification model. This process loops for all the features, as shown in Algorithm 4. This process is named MiniBatch K-Means Normalized Mutual Information Feature Inclusion (**KNFI**)

2. Least Ranked Feature Exclusion: This is a linear elimination approach where the least ranked features are eliminated one by one from the entire set of features. Initially, the list consists of all the features and the classification accuracy is calculated for the entire list. Then, in every loop, one least ranked feature is removed from the list. This process is carried out until the list becomes empty. The highest performance among all the iterations is considered as the outcome of our approach, as shown in algorithm 5. This process is named Mini-Batch K-Means Normalized Mutual Information least ranked Feature Exclusion (**KNFE**)

Algorithm 4 Ranking Based Feature Inclusion for Optimal Feature Subset (KNFI)

Input: Set of ranked features $S = \{f_0, f_1, f_2, ...f_m\}$, where m = total number of features, obtained from the feature ranking phase, f_0 is the highest ranked feature and f_m is the least ranked feature.

Output: prints the selected set of features

Initialisation :

1: Lst = S[0] prev=0

LOOP Process

2: **for** k = 0 to m-1 **do**
3: x_tst = x_tst [Lst]
4: x_tr =x_tr [Lst]
5: train the model based on any classifier and store the accuracy on acc
6: **if** acc > prev **then**
7: **if** $(k \neq m-1)$ **then**
8: Add S[k + 1] into the Lst
9: prev=acc
10: **else**
11: Print Lst
12: **end if**
13: **else**
14: Remove S [k] object from the Lst
15: **if** $(k \neq m-1)$ **then**
16: Add S[k + 1] to the Lst
17: **else**
18: Print Lst
19: **end if**
20: **end if**
21: **end for**
22: **return** *Lst*

Algorithm 5 Ranking based Feature elimination(KNFE)

Input: Set of ranked features $S = \{f_0, f_1, f_2, ...f_m\}$, where m = total number of features,.f_0 is the least ranked feature and f_m is the highest ranked feature.

Output: prints the result for every eliminated feature from the feature list

Initialization :

```
Lst = S prev=0
LOOP Process
for k = 0 to m-1 do
    x_tst = x_tst [ Lst ]
    x_tr =x_tr [ Lst ]
    //train the model based on any classifier and store the accuracy on acc
    //print the result along with the evaluation metrics
    if acc > prev then
        prev=acc // to store the greatest accuracy
        fet=i // to store the no. of feature eliminated
    end if
    delete Lst[0] //deleting the least ranked feature
end for
return
```

8.4 Experiment

8.4.1 Experimental Setup

The conduction of all the experiments is performed in Python Language using the python libraries. Florida International University provided us the required hardware. We used an Intel i7 4 core CPU with 16GB RAM. Also for large datasets, we used the Flounder Server (AMD Opteron Processor 6380 with 64 cores and 504GB RAM.

8.4.2 Datasets

Here we have considered nine datasets from the ML Repository of UCI [Fra10], three-click fraud datasets, one Intrusion Detection dataset, and Sonar dataset. We have tested upon two versons of TalkingData datset. The information of these datasets

is given below. We selected fifteen datasets haivng different number of features, instances, and classes. Also, we have considered both binary as well as multiclass datasets, which are shown in Table 8.1 and Table 8.2 respectively.

Table 8.1: Binary Datasets used in Experiment

Dataset	**Features**	**Instances**
UNSW_NB15[uns15]	47	2,540,047
TalkingData(version 1)[TKC$^+$19]	9	1,000,000
TalkingData(version 2)[TKC$^+$19]	9	913,692
Criteo[Kag14]	39	756,554
Avazu[Kag15]	16	1,000,000
Ionosphere	34	351
Breast_Cancer[Fra10]	10	699
Spambase[Fra10]	57	4,601
Sonar[D.18]	60	208

Table 8.2: MultiClass Datasets used in Experiment

Dataset	**Features**	**# Classes**	**Instances**
UNSW_NB15[uns15]	47	9	2,540,047
Lung_Cancer[Fra10]	56	3	32
Lymphographic[Fra10]	18	4	148
Iris[Fra10]	4	3	150
Heart Disease [Fra10]	13	5	303
Abalone [Fra10]	8	28	4,177

8.4.3 PreProcessing

UNSW_NB15 Dataset

It is an intrusion detection dataset that takes into consideration the instances of both the normal activities and the attack activities. To avoid overfitting due to a large number of normal activities, we have removed the normal activity instances. Initially, the data was in 4 different CSV files. We merged all the CSV files into a single dataset and performed the experiments. We removed the socket information(i.e.,

source ip address, source port number, destination ip address and destination port number) such that model becomes independent of them. We removed the white spaces present in some of the multiclass labels. All the categorical values were converted to the numerical values as the classifier can only learn numerical values. The different ranges of numerical data in the features become a challenge for the classifier to train the model[FD19]. To compensate this, we performed normalization on the entire data.

TalkingData Dataset

It is an AdTracking Fraud Dataset[Kag18] which has records of 200 million clicks over four days. It has features like app ID, os, IP address, click time, device type, channel, attributed time, and target label as is_attributed. In the preprocessing stage, we dropped the attributed time. We separated Click time into separate columns, i.e., day, hour, minute, and second. Two variants of the above mentioned dataset were used. In the first version, we considered one million rows of data in which the ratio of classes match the ratio at 200 million rows (Talkingdata Version 1) is taken. 913692 data samples were used for the second variant, where the rows were equally categorized into two classes (Talkingdata Version 2)[TKC$^+$19].

Avazu

This dataset is a Click fraud dataset consisting of clicks recorded over ten days and has features like id, click (Target Label), device_id, device_ip, an hour of click, and so on. We do the preprocessing i.e., separation of the 'hour of click' column into separate columns. We consider 1 million rows of data in which the ratio of classes match the ratio at 200 million rows to reduce the data size

Criteo

It is a Click fraud dataset that consists of 40 features. To clean the data, we have removed instances with 'NaN' values.

Ionosphere Dataset

In the Ionosphere dataset provided UCI repository, we converted the class labels (‘*good*,”*bad*′) into numerical values.

Breast Cancer, Lung Cance, Heart Disease datasets

In this dataset, there are some missing values represented by a question mark(‘?′). We removed the instances containing ? as a cleaning process.

Lymphography Dataset, Iris Dataset

These datasets were clean, and no preprocessing step had to be applied. However, we performed resampling as the instances with the same classes were together in the actual dataset.

Abalone Dataset

In this dataset, the first feature consists of categorical string values that we converted into numerical values.

Spambase Dataset, Sonar Dataset

These datasets are considered to compare our model with other research approaches. The spambase dataset is taken from UCI repository, and Sonar dataset is taken from Kaggle dataset. The datasets were clean with no NaN values, and no preprocessing was needed.

We normalized the entire data by using MinMaxScalar function for all the datasets.

8.5 Results and Discussion

8.5.1 Base Classifier : Random Forest

RF is a prevalent supervised ML technique that is flexible and very easy to use[Bre01]. As the name implies, RF has a large number of individual decision trees. Each de-

cision tree acts as an individual classifier. We get a class prediction from each tree in the RF, and the class that gets the most votes becomes the model prediction of RF. With the increase in the number of trees, the classifier has a greater ability to resist noise and obtain greater accuracy. The RF, being a simple classifier built on decision trees, can easily adapt to large changes in the data size, having the benefit of scalability[Liu14].

8.5.2 Evaluation Metrics

The accuracy of the algorithm needs to be evaluated by certain standard metrics. For binary classification, we have considered the standard metric, Area Under Curve (AUC) and also the F1 Score, which is computed based upon the Precision and Recall score. For the multiclass dataset, we have considered the F1 Score as the evaluation criteria. The F1 Score can also be obtained from the confusion matrix. This metric can only be used for the test data whose true values are already known such that we get a confusion matrix.

We can obtain the following information from the confusion matrix:

- True Positive (TrPos): model correctly predicting Positive cases as Positive.
- False Positive (FlPos): model incorrectly predicting the Negative cases as Positive.
- False Negative (FlNeg): model incorrectly predicting positive cases as Negative.
- True Negative (TrNeg): model correctly predicting negative cases as positive.

Precision score(Pr): It measures accuracy based upon correctly predicted cases.

$$Pr = \frac{TrPos}{TrPos + FlPos} \tag{8.5}$$

Recall score(RC) : It is the TrPos rate to predict the ofteness of predicting positive.

$$RC = \frac{TrPos}{TrPos + FlNeg} \tag{8.6}$$

F1 Score(F1) : F1 is the weighted average of recall and precision of each class.

$$F1 = 2\left(\frac{Pr * RC}{Pr + RC}\right) \tag{8.7}$$

ROC-AUC curve is a standard metric to measure the performance of the classification model. The probability curve between the true positive rates against false positive rates is referred to as ROC. AUC represents the degree of separability. The higher the AUC, the more the efficiency of the model.

8.5.3 Analysis Method

To empirically test the advantages and disadvantages of our method, we performed several experiments on real-world datasets with four different approaches. They are:

1. Considering all the features present in the dataset for classification and calculation of its accuracy, AUC (for binary datasets), precision, recall, F-1 Score. We represent this as **AF**.
2. Our approach (**KNFI**), where we perform classification based on the ranked features and determine its evaluation metrics. Without the need of the user to specify the number of optimal features, our approach automatically calculates it. This number has been considered as the base number for performing RFE, where we explicitly have to provide the required number of optimal features.
3. Using RFE (Recursive Feature Elimination), a standard process, provided by Scikit_learn [PVG+11], selects features by recursively considering the small set of features. The user explicitly has to give the desired subset number (k),

and then it returns the best accuracy from the best subset with k features. In our experiment, we have considered the value of K, referring to our KNFI approach.

4. Our second approach **KNFE**, where we remove the least ranked features one after another, performing the classification and calculating its evaluation metrics. The best accuracy obtained after removing 'k' features is considered as the comparing value with other methods.

A comparative analysis is performed for the results obtained from the four methods in terms of various evaluation metrics, as mentioned above. We can observe that our approach takes less computation time compared to the existing methods, and in many datasets, it produced better results.

8.5.4 Discussions

For Binary Datasets

In the UNSW_NB15 dataset, both our KNFI and KNFE methods improvised the learning algorithm to obtain greater accuracy, AUC, and F1-Score, as shown in Table 8.3. KNFI selected 17 features and stood superior in terms of all the evaluation metrics. Also, the evaluation metrics greatly increased in the Ionosphere dataset as in Table 8.4 for our 6 selected features among the 34 features. Most of the redundant features were removed, giving us better results.

Table 8.3: Experimental Results of UNSW_NB15 Binary Datasets

Method	**Ftr**	**Acc**	**AUC**	**F1**
AF	43	99.93	99.46	99.93
KNFI	17	**99.963**	99.614	99.96
RFE	17	99.960	99.612	99.96
KNFE	-6	**99.944**	99.96	99.94

Table 8.4: Experimental Results of Ionosphere Datasets

Method	**Ftr**	**Acc**	**AUC**	**F1**
AF	34	92.96	90.91	92.84
KNFI	6	**97.18**	95.23	97.14
RFE	6	91.54	7.92	91.55
KNFE	-7	**95.77**	94.238	95.74

We have a slight increase in accuracy for the Avazu dataset as in Table 8.5 for both of our approaches. However, the AUC is slightly decreased in both the methods. The decrease in AUC could be due to the presence of imbalanced data. The F1-Score is a much better metric of measurement[unk70]. The F1-Score remained constant with an increase in accuracy, giving us a better-trained model with the selected features. This is showed in Table 8.5. Also, in the TalkingData dataset (version 2), the accuracy increased slightly for KNFI. However, for KNFE, it showed zero elimination of features for the best classification accuracy meaning all the features are independent and contributing for the classification model.

Table 8.5: Experimental Results of Avazu Dataset

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	25	83.029	54.235	77.89
KNFI	7	**83.4375**	53.283	77.89
RFE	7	83.075	53.013	77.63
KNFE	-17	**83.381**	52.456	77.36

Table 8.6: Experimental Results of Talking Dataset Version 2

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	9	99.9179	99.9179	99.92
KNFI	4	**99.919**	99.919	99.92
RFE	4	99.919	99.919	99.92
KNFE	0	**99.9179**	99.917	99.92

In the Spambase dataset, our KNFE approach enhanced the classification accuracy along with all the evaluation metrics by removing three redundant features.

From KNFI approach, the accuracy slightly reduced, taking least prediction time and performed well in comparision to RFE. This is shown in Table 8.7. Also, in the Sonar dataset, KNFE method outperformed all other approaches by removing nine redundant features. Our KNFI approach also gave better results compared to the AF and the RFE methods, as shown in Table 8.8. The relevance of the features in Sonar Dataset is shown in fig. 8.2. Some features tend to have very high importance in accordance to the class label and some features tend to have no importance or very low importance in accordance to the class label. We obtain the ranking of the features and then preform KNFI and KNFE. In fig. 8.3, we show the change in the accuracy as we eliminate the least ranked features one at a time. There is a drastic decrease in accuracy as we eliminate large number of features. For a particular number of features eliminated, we observed the highest accuracy.

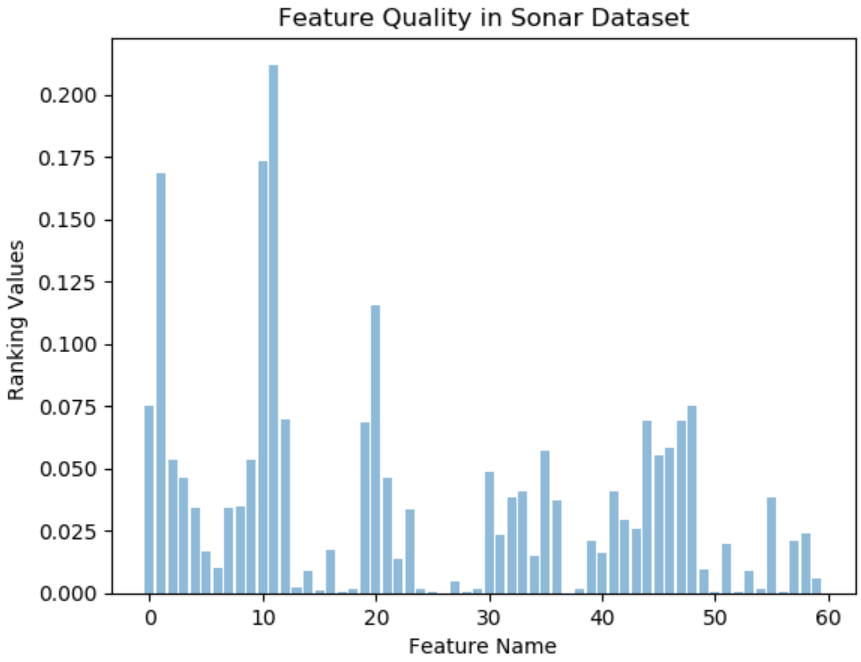


Figure 8.2: Feature Ranking for the Sonar Dataset

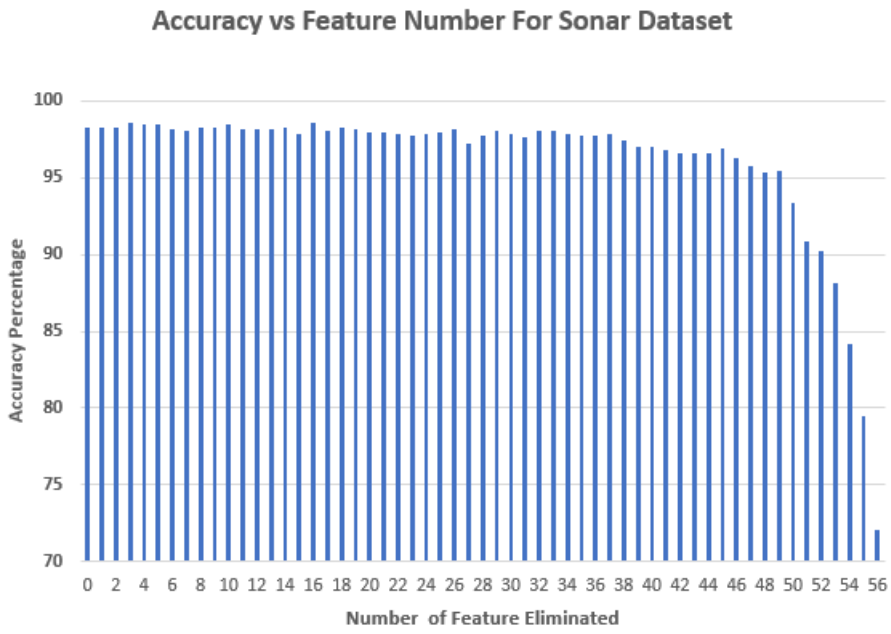


Figure 8.3: Change in Accuracy in the KNFE Method

Table 8.7: Experimental Results of Spambase Dataset

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	57	98.04	97.69	98.04
KNFI	15	**97.82**	97.52	97.82
RFE	15	97.285	96.69	97.27
KNFE	-3	**98.58**	98.301	98.93

Table 8.8: Experimental Results of Sonar Dataset

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	60	92.86	93.05	92.88
KNFI	3	**95.24**	95.138	95.24
RFE	3	88.09	88.88	88.16
KNFE	-9	**97.62**	97.91	97.63

However, in the TalkingData (Version 1), Criteo and Breast Cancer datasets shown in Table 8.9, 8.10 and 8.11 respectively, the performance seems to drop when performing KNFI process. However, KNFE gave either better results or the same results. This case appears when all the features tend to contribute to fitting the

model. In such a scenario, either few features are removed or zero features are removed as in case the of TalkingData dataset (Table 8.9). The difference in prediction for AF contribution and zero feature elimination in KNFE is due to the change in the pattern of features provided during the training of data. The performance decreased in KNFI model. Whenever proper information is not extracted from the FS process, the classification accuracy may be negatively affected. The corealtion of the features also affect the FS process. Furthermore, when the sample size is big, the classifier predicts values well with the entire attributes. Also some datasets tend to perform well with other classifers [JBB15].

Table 8.9: Experimental Results of Talking dataset Version 1

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	8	95.127	91.672	95.08
KNFI	6	**94.252**	90.434	94.14
RFE	6	94.784	91.059	94.67
KNFE	0	**95.20**	91.72	95.11

Table 8.10: Experimental Results of Criteo Dataset

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	39	73.545	62.386	70.29
KNFI	3	**70.205**	57.725	65.85
RFE	3	70.268	55.902	63.85
KNFE	-5	**73.545**	62.45	70.33

Table 8.11: Experimental Results of Breast Cancer Dataset

Method	**Ftr.**	**Acc**	**AUC**	**F1**
AF	10	98.540	98.113	98.53
KNFI	4	**97.810**	97.517	97.81
RFE	4	94.890	93.744	94.84
KNFE	-3	**98.540**	98.113	98.53

Multiclass Datasets

In most of the MultiClass datasets, we can observe the positive impact of our KNFI as well as KNFE techniques. In UNSW_NB15 dataset (Table 8.12), the accuracy increased by 0.781 percent along with the increase in F1 Score. Our model selected 16 out of 43 features to get the most efficient results. Our KNFI method enhanced the accuracy and outperformed all other methods giving us good results.

Table 8.12: Experimental Results of UNSW_NB15 Dataset

Method	**Ftr.**	**Acc**	**F1**
AF	43	89.326	88.87
KNFI	16	**90.107**	88.88
RFE	16	89.356	89.02
KNFE	-18	**89.591**	89.02

For the Lung_cancer dataset (Table 8.13), both our methods doubled the accuracy as well as the F1 Score and took the least prediction time. Similarly, for the Lymphographic dataset (Table 8.14), our KNFE method gave better results when compared to all the methods.

Table 8.13: Experimental Results of Lung_Cancer Dataset

Method	**Ftr.**	**Acc**	**F1**
AF	56	33.333	37.78
KNFI	3	**66.666**	68.25
RFE	3	50.00	52.78
KNFE	-14	**66.666**	68.25

Table 8.14: Experimental Results of Lymphography Dataset

Method	**Ftr.**	**Acc**	**F1**
AF	18	86.66	85.19
KNFI	2	**90.00**	89.78
RFE	2	76.66	80.00
KNFE	-2	**86.66**	75.17

The Iris Dataset (Table 8.15) performed well when selecting two of the best features from all the four features. The heart disease dataset (Table 8.17) had a massive fifteen percent increase in accuracy along with a considerable increase in F1 Score using KNFI. Even KNFE increased the accuracy.

Table 8.15: Experimental Results of Iris Dataset

Method	**Ftr.**	**Acc**	**F1**
AF	4	96.666	96.67
KNFI	2	**99.9999**	99.99
RFE	2	99.999	99.999
KNFE	-4	**99.999**	99.999

Table 8.16: Experimental Results of Heart Disease Dataset

Method	**Ftr.**	**Acc**	**F1**
AF	13	41.667	34.60
KNFI	4	**56.667**	51.53
RFE	4	43.333	36.71
KNFE	-11	**51.667**	40.90

For the Abalone dataset, our KNFI did not produce improved the performance. However, our KNFE increased the preformance. The dataset contains less number of features and many classes. This makes the prediction of classification much tricky. Also, if additional knowlegde is not obtained form the FS method, it may not increase the performance. [JBB15].

Table 8.17: Experimental Results of Abalone Dataset

Method	**Ftr.**	**Acc**	**F1**
AF	8	24.521	22.86
KNFI	1	**21.650**	20.27
RFE	1	17.344	17.14
KNFE	0	**25.239**	23.61

Other Compared Works

Other than RFE, we also compared our work with other previous works. In comparison with the previous studies of the UNSW_NB15 dataset, our approach of KNFI produced improved results for binary as well as multiclass datasets. As a preprocessing step, we remove all the instances that have 'NaN' values, which decreases the number of instances. This has enhanced the performance of the classifier. When our model is run on this dataset, the efficacy of the predictor increased significantly. These results can be seen in Tables (8.18 & 8.19).

Table 8.18: Comparision of Accuracy for Binary UNSW_NB15 with Previous Studies

Study	**Method**	**Accuracy**
Zewairi, *et al.*[AZAA17]	Deep Learning	98.99
Primartha and Tama [PT17]	Random Forest	95.5
	Multilayer Perceptron	83.50
Nour, *et al.*[MS17]	Naive Bayes	79.50
	Linear Regression	83.00
	Expectation-Maximization	77.20
Belouch, *et al.*[BEI17]	Random Tree	86.59
	Naive Bayes	80.40
	RepTree	87.80
	Artificial Neural Network	86.31
	Decision Tree	86.13
Faker, *et al.*	Gradient Boosted Tree	97.92
	Random Forest	98.86
	Deep Neural Network	99.19
Our Work	Random Forest(AF)	99.93
	KNFI	**99.963**
	KNFE	**99.944**

Table 8.19: Comparision of Accuracy for UNSW_NB15 MultiClass with Previous Studies

Study	**Method**	**Accuracy**
Belouch, *et al.*[BEI17]	Random Tree	76.21
	Naive Bayes	73.86
	RepTree	79.20
	Artificial Neural Network	78.14
Our Work	Random Forest(AF)	89.326
	KNFI	**90.107**
	KNFE	**89.591**

We compared the Ionosphere dataset with the existing hybrid feature selection methods. We can observe in Table 8.20 that both KNFI and KNFE methods produced much better results with greater classification accuracy.

Table 8.20: Comparision of Ionosphere Data with Previous Studies

Method	**# Ftr.**	**F1**	**RC**	**Pr**	**Acc**
Venkatesh *et al.*[VA19]	15	95.09	94.65	95.70	95.28
HGEFS [XYW18]	n.a.	n.a.	n.a.	n.a	91.33
FSFOA [GFD16]	n.a.	n.a.	n.a.	n.a	95.12
KNFI	6	97.14	97.18	97.29	**97.18**
KNFE	-7	95.74	95.77	95.76	**95.77**

We compare the Spambase dataset and Sonar dataset with the previous works performed in [ETPZ09], [Bat94], [KC02], [PLD05] in terms of classification accuracy since other evaluation metrics have not been provided. They have calculated the rate of classification for the different number of selected features. As a comparison metric, we have taken the instances with the highest accuracy as presented in their papers.

For comparative analysis, we have also calculated the accuracy using KNFE for the same number of features as provided in the previous papers. Also, we have evaluated using KNFI and KNFE. They are shown in Table 8.21 and Table 8.22.

Our method outperformed other methods giving us good results. The KNFE(MAX) represents our method without any constraint of number of required features.

Table 8.21: Comparision of Accuracy for Spambase Dataset with Previous Studies

Ftr. selection method	**# features**	**accuracy**
GAMIFS[ETPZ09]	3	83.50
NMIFS[ETPZ09]	3	75.8
MIFS[Bat94]	3	78.4
MIFS-U[KC02]	3	81.2
OFS-MI [ETPZ09, CH05]	3	78.4
KNFE	3	**84.15**
KNFI	15	**97.82**
KNFE(MAX)	54	**98.59**

Table 8.22: Comparision of Accuracy for Sonar Dataset with Previous Studies

Ftr. selection method	**# features**	**accuracy**
NMIFS[ETPZ09]	15	86.73
MIFS(β=0.5)[Bat94]	15	85.96
MIFS-U($\beta = 0.5$)[KC02]	15	84.04
HGEFS[XYW18]	N.A.	83.00
FSFOA[GFD16]	N.A	86.98
KNFE	15	**92.85**
KNFI	3	**95.24**
KNFE(MAX)	51	**97.62**

8.6 Conclusion

This work presented a new hybrid method taking into consideration the advantages of both filter and wrapper method with no constraint for the user to input the number of features required. In our approach, we used the NMI as a metric to rank the features after clustering by Mini-Batch K Means. Once we obtained the ranked features, we came up with two methods to select the features; the feature

inclusion method (KNFI) and feature exclusion method (KNFE). We came up with an algorithm for the feature inclusion method, and in the feature removal method, we removed the least important features to get the best performance accuracy. In most of the datasets, KNFI performed well taking least number of features whereas, in datasets with least relationship among the features, KNFE method performed well. For future work, optimizing the time taken to get the selected features would help to reduce time complexity. Also, we can come up with better metrics to get the actual relationships among the features such that the redundant features are eliminated.

[AB79] Bovas Abraham and George EP Box. Bayesian analysis of some outlier problems in time series. *Biometrika*, 66(2):229–236, 1979.

[AD92] Hussein Almuallim and Thomas G Dietterich. Efficient algorithms for identifying relevant features. In *In Proceedings of the Ninth Canadian Conference on Artificial Intelligence*. Citeseer, 1992.

[AK15] B. A. Almogahed and I. A. Kakadiaris. Neater: filtering of over-sampled data using non-cooperative game theory. *Soft Computing*, 19(11):3301–3322, Nov 2015.

[AM13] Basant Agarwal and Namita Mittal. Sentiment classification using rough set based hybrid feature selection. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 115–119, 2013.

[APS+11] D. Antoniou, M. Paschou, E. Sakkopoulos, E. Sourla, G. Tzimas, A. Tsakalidis, and E. Viennas. Exposing click-fraud using a burst detection algorithm. In *2011 IEEE Symposium on Computers and Communications (ISCC)*, pages pp. 1111–1116, Kerkyra, Greece, June 2011.

[APST05] Alex Alexandridis, Panagiotis Patrinos, Haralambos Sarimveis, and George Tsekouras. A two-stage evolutionary algorithm for variable selection in the development of rbf neural network models. *Chemometrics and Intelligent Laboratory Systems*, 75(2):149–162, 2005.

[ASKR12] Ebru Arisoy, Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20–28. Association for Computational Linguistics, 2012.

[AV06] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Symposium on computational geometry*, volume 6, pages 1–10, 2006.

[AZAA17] Malek Al-Zewairi, Sufyan Almajali, and Arafat Awajan. Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system. *2017 Interna-*

tional Conference on New Trends in Computing Sciences (ICTCS), 2017.

[BAB+17] Kianoosh G Boroojeni, M Hadi Amini, Shahab Bahrami, SS Iyengar, Arif I Sarwat, and Orkun Karabasoglu. A novel multi-time-scale modeling for electric power demand forecasting: From short-term to medium-term horizon. *Electric Power Systems Research*, 142:58–73, 2017.

[Bat94] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[BC64] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964.

[BCN+14] B. Biggio, I. Corona, B. Nelson, B. B. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli. *Security Evaluation of Support Vector Machines in Adversarial Environments*, pages pp. 105–153. Springer International Publishing, Cham, 2014.

[BCSMAB+14] Verónica Bolón-Canedo, Noelia Sánchez-Marono, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014.

[BEI17] Mustapha Belouch, Salah El, and Mohamed Idhammad. A two-stage classifier approach using reptree algorithm for network intrusion detection. *International Journal of Advanced Computer Science and Applications*, 8(6), 2017.

[BF74] Morton B Brown and Alan B Forsythe. Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346):364–367, 1974.

[BFR14] B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):pp. 984–996, April 2014.

[BHI18] Mustapha Belouch, Salah El Hadaj, and Mohamed Idhammad. Performance evaluation of intrusion detection based on machine learning using apache spark. *Procedia Computer Science*, 127:1–6, 2018.

[BIM11] Sukarna Barua, Md. Monirul Islam, and Kazuyuki Murase. A novel synthetic minority oversampling technique for imbalanced data set learning. In Bao-Liang Lu, Liqing Zhang, and James Kwok, editors, *Neural Information Processing*, pages 735–744, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[BIM13] Sukarna Barua, Md Monirul Islam, and Kazuyuki Murase. Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 317–328. Springer, 2013.

[BIYM14] S. Barua, M. M. Islam, X. Yao, and K. Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, Feb 2014.

[BJD15] C. Bellinger, N. Japkowicz, and C. Drummond. Synthetic oversampling for advanced radioactive threat detection. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 948–953, Dec 2015.

[BJRL15] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[BM98] Catherine L Blake and Christopher J Merz. Uci repository of machine learning databases, 1998, 1998.

[BMRA10] Fabrício Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgílio Almeida. Detecting spammers on twitter. In *In Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, 2010.

[BNJT10] M. Barreno, B. Nelson, A. Joseph, and J. Tygar. The security of machine learning. *Machine Learning*, 81(2):pp. 121–148, Nov 2010.

[BNL11] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In Chun-Nan Hsu and Wee Sun Lee,

editors, *Proceedings of the Asian Conference on Machine Learning*, volume 20 of *Proceedings of Machine Learning Research*, pages pp. 97–112, South Garden Hotels and Resorts, Taoyuan, Taiwain, 14–15 Nov 2011. PMLR.

[Box87] Joan Fisher Box. Guinness, gosset, fisher, and small samples. *Statist. Sci.*, 2(1):45–52, 02 1987.

[BPM04] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29, 2004.

[BPZL12] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of machine learning research*, 13(Jan):27–66, 2012.

[Bra97] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[Bre96] Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

[Bre99] Leo Breiman. Random forests. *UC Berkeley TR567*, 1999.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BS13] Chumphol Bunkhumpornpat and Sitthichoke Subpaiboonkit. Safe level graph for synthetic minority over-sampling techniques. In *2013 13th International Symposium on Communications and Information Technologies (ISCIT)*, pages 570–575. IEEE, 2013.

[BS16] Katarzyna Borowska and Jarosław Stepaniuk. Imbalanced data classification: A novel re-sampling approach combining versatile improved smote and rough sets. In Khalid Saeed and Władysław Homenda, editors, *Computer Information Systems and Industrial Management*, pages 31–42, Cham, 2016. Springer International Publishing.

[BSL09] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 475–482. Springer, 2009.

[BSL11] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Mute: Majority under-sampling technique. In *2011 8th International Conference on Information, Communications & Signal Processing*, pages 1–4. IEEE, 2011.

[BSL12] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Dbsmote: Density-based synthetic minority oversampling technique. *Applied Intelligence*, 36(3):664–684, Apr 2012.

[BSLL+16] Prudhvi Ratna Badri Satya, Kyumin Lee, Dongwon Lee, Thanh Tran, and Jason (Jiasheng) Zhang. Uncovering fake likers in online social networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2365–2370, New York, NY, USA, 2016. ACM.

[BW+01] Gary Bishop, Greg Welch, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-23175):41, 2001.

[BXG+13] Alex Beutel, Wanhong Xu, Venkatesan Guruswami, Christopher Palow, and Christos Faloutsos. Copycatch: Stopping group attacks by spotting lockstep behavior in social networks. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 119–130, New York, NY, USA, 2013. ACM.

[Bye98] S Byers. Nearest-neighbor clutter removal for estimating features in spatial point process. *J. American Statistical Association*, 93(442):596–604, 1998.

[CBHK02] 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.

[CC15] Gang Chen and Jin Chen. A novel wrapper method for feature selection and its applications. *Neurocomputing*, 159:219–226, 2015.

[CCCG10] Leichen Chen, Zhihua Cai, Lu Chen, and Qiong Gu. A novel differential evolution-clustering hybrid resampling algorithm on imbalanced datasets. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 81–85. IEEE, 2010.

[CCS06] David A Cieslak, Nitesh V Chawla, and Aaron Striegel. Combating imbalance in network intrusion datasets. In *GrC*, pages 732–737, 2006.

[CCV11] Silvia Cateni, Valentina Colla, and Marco Vannucci. Novel resampling method for the classification of imbalanced datasets for industrial and other real-world problems. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 402–407. IEEE, 2011.

[CDM08] Prabhakar Raghavan Christopher D. Manning. Introduction to information retrieval, 2008.

[CdQC12] R. A. Costa, R. J. G. B. de Queiroz, and E. R. Cavalcanti. A proposal to prevent click-fraud using clickable captchas. In *2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, pages pp. 62–67, MD, USA, June 2012.

[CDS06] Oscar Cordón, Sergio Damas, and Jose Santamaría. Feature-based image registration by means of the chc evolutionary algorithm. *Image and Vision Computing*, 24(5):525–533, 2006.

[CF94] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *Machine Learning Proceedings 1994*, pages 28–36. Elsevier, 1994.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[CGC10] S. Chen, G. Guo, and L. Chen. A new over-sampling method based on cluster ensembles. In *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pages 599–604, April 2010.

[CGLR+17] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodriguez, Asdrúbal López, José Ruiz Castilla, and Adrian Trueba. Pso-based method

for svm classification on skewed data sets. *Neurocomputing*, 228:187 – 197, 2017. Advanced Intelligent Computing: Theory and Applications.

[CH05] T. W.S. Chow and D. Huang. Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information. *Trans. Neur. Netw.*, 16(1):213–224, January 2005.

[Cha73] Chieng-Yi Chang. Dynamic programming as applied to feature subset selection in a pattern recognition system. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):166–171, 1973.

[CHC+12] Carlton Chu, Ai-Ling Hsu, Kun-Hsien Chou, Peter Bandettini, ChingPo Lin, Alzheimer's Disease Neuroimaging Initiative, et al. Does feature selection improve classification accuracy? impact of sample size and feature selection on classification using anatomical magnetic resonance images. *Neuroimage*, 60(1):59–70, 2012.

[Cho15] F. Chollet. Keras: Deep learning library for theano and tensorflow, 2015.

[CHS+06] Gilles Cohen, Mélanie Hilario, Hugo Sax, Stéphane Hugonnet, and Antoine Geissbuhler. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial intelligence in medicine*, 37(1):7–18, 2006.

[cli19] Click fraud statistics: The click fraud blog, Mar 2019.

[CMP10] O. Chertov, V. Malchykov, and D. Pavlov. Non-dyadic wavelets for detection of some click-fraud attacks. In *International Conference on Signals and Electronic Circuits*, Gliwice, Poland, Sep. 2010.

[COL17] Tim Chenoweth, Zoran ObradoviĆ, and Sauchi Stephen Lee. Embedding technical analysis into neural network based trading systems. In *Artificial Intelligence Applications on Wall Street*, pages 523–541. Routledge, 2017.

[CS96] Kevin J Cherkauer and Jude W Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *KDD*, volume 96, pages 315–318, 1996.

[CS03] Rich Caruana and Virginia R de Sa. Benefitting from the variables that variable selection discards. *Journal of machine learning research*, 3(Mar):1245–1264, 2003.

[CS14] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[CSC14] J. Crussell, R. Stevens, and H. Chen. Madfraud: Investigating ad fraud in android applications. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages pp. 123–134, NY, USA, 2014. ACM.

[CW11] Qinghua Cao and Senzhang Wang. Applying over-sampling technique based on data density and cost-sensitive svm to imbalanced learning. In *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*, volume 2, pages 543–548. IEEE, 2011.

[CYYP14] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 477–488, New York, NY, USA, 2014. ACM.

[D.18] Anh D. sonar data set, Feb 2018.

[DBL18] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1 – 20, 2018.

[DCFJ+14] Emiliano De Cristofaro, Arik Friedman, Guillaume Jourjon, Mohamed Ali Kaafar, and M. Zubair Shafiq. Paying for likes?: Understanding facebook like fraud using honeypots. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 129–136, New York, NY, USA, 2014. ACM.

[DCSL02] Manoranjan Dash, Kiseok Choi, Peter Scheuermann, and Huan Liu. Feature selection for clustering-a filter solution. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 115–122. IEEE, 2002.

[DG17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[DGZ12] V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages pp. 175–186, NY, USA, 2012. ACM.

[DLCF07] Jorge De La Calleja and Olac Fuentes. A distance-based over-sampling method for learning from imbalanced data sets. In *FLAIRS Conference*, pages 634–635, 2007.

[DLCFG08] Jorge De La Calleja, Olac Fuentes, and Jesús González. Selecting minority examples from misclassified data for over-sampling. In *FLAIRS Conference*, pages 276–281, 2008.

[dLSB17] Carolina Alves de Lima Salge and Nicholas Berente. Is that social bot behaving unethically? *Commun. ACM*, 60(9):29–31, August 2017.

[DM14] L. Dritsoula and J. Musacchio. A game of clicks: Economic incentives to fight click fraud in ad networks. *SIGMETRICS Perform. Eval. Rev.*, 41(4):pp. 12–15, April 2014.

[Dod08] Yadolah Dodge. *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008.

[DP90] Didier Dubois and Henri Prade. Rough fuzzy sets and fuzzy rough sets. *International Journal of General Systems*, 17(2-3):191–209, 1990.

[DP11] T. Deepa and M. Punithavalli. An e-smote technique for feature selection in high-dimensional imbalanced dataset. In *2011 3rd International Conference on Electronics Computer Technology*, volume 2, pages 322–324, April 2011.

[DTHS15] X. T. Dang, D. H. Tran, O. Hirose, and K. Satou. Spy: A novel resampling method for improving classification performance in imbalanced data. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, pages 280–285, Oct 2015.

[Duc19] Daniel Duckworth. pykalman, 2019.

[DW11] Yanjie Dong and Xuehua Wang. A new over-sampling approach: Random-smote for learning from imbalanced data sets. In Hui Xiong and W. B. Lee, editors, *Knowledge Science, Engineering and Management*, pages 343–352, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Esh91] Larry J Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of genetic algorithms*, volume 1, pages 265–283. Elsevier, 1991.

[ETPZ09] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, 20(2):189–201, 2009.

[exs16] Accuracy, precision, recall & f1 score: Interpretation of performance measures, Nov 2016.

[EZH16a] Eid Emary, Hossam M Zawbaa, and Aboul Ella Hassanien. Binary ant lion approaches for feature selection. *Neurocomputing*, 213:54–65, 2016.

[EZH16b] Eid Emary, Hossam M Zawbaa, and Aboul Ella Hassanien. Binary grey wolf optimization approaches for feature selection. *Neurocomputing*, 172:371–381, 2016.

[FB12] MAH Farquad and Indranil Bose. Preprocessing unbalanced data using support vector machine. *Decision Support Systems*, 53(1):226–233, 2012.

[FD19] Osama Faker and Erdogan Dogdu. Intrusion detection using big data and deep learning techniques. In *Proceedings of the 2019 ACM Southeast Conference*, pages 86–93. ACM, 2019.

[FLDH+16a] M. Faou, A. Lemay, D. Décary-Hétu, J. Calvet, F. Labréche, M. Jean, B. Dupont, and J. M. Fernande. Follow the traffic: Stopping click fraud by disrupting the value chain. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages pp. 464–476, Auckland, New Zealand, Dec 2016.

[FLDH+16b] Matthieu Faou, Antoine Lemay, David Décary-Hétu, Joan Calvet, François Labrèche, Militza Jean, Benoit Dupont, and José M Fernande. Follow the traffic: Stopping click fraud by disrupting the value chain. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, pages 464–476. IEEE, 2016.

[Fle04] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine learning research*, 5(Nov):1531–1555, 2004.

[FNHMG11] Francisco Fernández-Navarro, César Hervás-Martínez, and Pedro Antonio Gutiérrez. A dynamic over-sampling procedure based on sensitivity for multi-class problems. *Pattern Recognition*, 44(8):1821 – 1833, 2011.

[FP97] Tom Fawcett and Foster Provost. Adaptive fraud detection. *Data mining and knowledge discovery*, 1(3):291–316, 1997.

[Fra10] Andrew Frank. Uci machine learning repository. *http://archive. ics. uci. edu/ml*, 2010.

[Fri02] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[fru19] Click fraud - the 5 most common forms of click fraud., 2019.

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[FTW11] Xiannian Fan, Ke Tang, and Thomas Weise. Margin-based oversampling method for learning from imbalanced datasets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 309–320. Springer, 2011.

[Ful09] Wayne A Fuller. *Introduction to statistical time series*, volume 428. John Wiley & Sons, 2009.

[FVD+16] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Commun. ACM*, 59(7):96–104, June 2016.

[GA08] Sami Gazzah and Najoua Essoukri Ben Amara. New oversampling approaches based on polynomial fitting for imbalanced data sets. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 677–684. IEEE, 2008.

[GB10a] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages pp. 249–256, 2010.

[GB10b] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[GBCT13] Eric Gilbert, Saeideh Bakhshi, Shuo Chang, and Loren Terveen. "i need to try this"?: A statistical overview of pinterest. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2427–2436, New York, NY, USA, 2013. ACM.

[GCJ18] Shenkai Gu, Ran Cheng, and Yaochu Jin. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing*, 22(3):811–822, 2018.

[GCS+15] Maria Giatsoglou, Despoina Chatzakou, Neil Shah, Christos Faloutsos, and Athena Vakali. Retweeting activity on twitter: Signs of deception. In Tru Cao, Ee-Peng Lim, Zhi-Hua Zhou, Tu-Bao Ho, David Cheung, and Hiroshi Motoda, editors, *Advances in Knowledge Discovery and Data Mining*, pages 122–134, Cham, 2015. Springer International Publishing.

[GCZ09] Qiong Gu, Zhihua Cai, and Li Zhu. Classification of imbalanced data sets by using the hybrid re-sampling algorithm based on isomap. In Zhihua Cai, Zhenhua Li, Zhuo Kang, and Yong Liu, editors, *Advances in Computation and Intelligence*, pages 287–296, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[gee19] Ml: Mini batch k-means clustering algorithm, May 2019.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[GFD16] Manizheh Ghaemi and Mohammad-Reza Feizi-Derakhshi. Feature selection using forest optimization algorithm. *Pattern Recognition*, 60:121–129, 2016.

[GH16] Hossein Gharaee and Hamid Hosseinvand. A new feature selection ids based on genetic algorithm and svm. In *2016 8th International Symposium on Telecommunications (IST)*, pages 139–144. IEEE, 2016.

[GHC+14] Ming Gao, Xia Hong, Sheng Chen, Chris J. Harris, and Emad Khalaf. Pdfos: Pdf estimation based over-sampling for imbalanced two-class problems. *Neurocomputing*, 138:248 – 259, 2014.

[GHE15] S. Gazzah, A. Hechkel, and N. Essoukri Ben Amara. A hybrid sampling method for imbalanced data. In *2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15)*, pages 1–6, March 2015.

[GHW+10] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y. Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 681–683, New York, NY, USA, 2010. ACM.

[GPAM+14] I. Goodfellow, J. Pouget-Abadie, M Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, pages pp. 2672–2680, 2014.

[GSM07] Vicente García, Jose Sánchez, and Ramon Mollineda. An empirical study of the behavior of classifiers on imbalanced and overlapped data sets. In *Iberoamerican Congress on Pattern Recognition*, pages 397–406. Springer, 2007.

[Had10] H. Haddadi. Fighting online click-fraud using bluff ads. *SIGCOMM Comput. Commun. Rev.*, 40(2):pp. 21–25, April 2010.

[HAH+18] Sajad Homayoun, Marzieh Ahmadzadeh, Sattar Hashemi, Ali Dehghantanha, and Raouf Khayami. Botshark: A deep learning approach for botnet traffic detection. In *Cyber Threat Intelligence*, pages 137–153. Springer, 2018.

[HBGL08] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008.

[HBK14] Nazrul Hoque, Dhruba K Bhattacharyya, and Jugal K Kalita. Mifs-nd: A mutual information-based feature selection method. *Expert Systems with Applications*, 41(14):6371–6385, 2014.

[HBXC15] Zhongyi Hu, Yukun Bao, Tao Xiong, and Raymond Chiong. Hybrid filter–wrapper feature selection for short-term load forecasting. *Engineering Applications of Artificial Intelligence*, 40:17–27, 2015.

[HG08] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

[HGV11] Anne-Claire Haury, Pierre Gestraud, and Jean-Philippe Vert. The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. *PloS one*, 6(12):e28210, 2011.

[HHLD] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database in pervasive computing and applications (icpca). In *2011 6th international conference on*, pages 363–366.

[HHY+14] Jun Hu, Xue He, Dong-Jun Yu, Xi-Bei Yang, Jing-Yu Yang, and Hong-Bin Shen. A new supervised over-sampling algorithm with application to protein-nucleotide binding residue prediction. *PLOS ONE*, 9(9):1–10, 09 2014.

[HIRR18] Ch. Md. Rakin Haider, A. Iqbal, A. Hasan Rahman, and M. Sohel Rahman. An ensemble learning based approach for impression fraud

detection in mobile advertising. *Journal of Network and Computer Applications*, 112:pp. 126 – 141, 2018.

[HL13] Feng Hu and Hang Li. A novel boundary oversampling algorithm based on neighborhood rough set model: Nrsboundary-smote. *Mathematical Problems in Engineering*, 2013, 2013.

[HLD17] Jinlong Hu, Junjie Liang, and Shoubin Dong. ibgp: A bipartite graph propagation approach for mobile advertising fraud detection. *Mobile Information Systems*, 2017, 2017.

[HLMH09] Shengguo Hu, Yanfeng Liang, Lintao Ma, and Ying He. Msmote: improving classification performance when training data is imbalanced. In *2009 second international workshop on computer science and engineering*, volume 2, pages 13–17. IEEE, 2009.

[HS98] Mark A Hall and Lloyd A Smith. Practical feature subset selection for machine learning. 1998.

[HWM05] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[IAJ14] H Hannah Inbarani, Ahmad Taher Azar, and G Jothi. Supervised hybrid feature selection based on pso and rough sets for medical diagnosis. *Computer methods and programs in biomedicine*, 113(1):175–185, 2014.

[ins18] instagram.com. Build your business on instagram, 2018.

[IR83] S. S. Iyengar and M. S. Rao. Statistical techniques in modeling of complex systems: Single and multiresponse models. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(2):175–189, March 1983.

[IS15] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, February 2015.

[ISBL02] Iñaki Inza, Basilio Sierra, Rosa Blanco, and Pedro Larrañaga. Gene selection by sequential search wrapper approaches in microarray

cancer class prediction. *Journal of Intelligent & Fuzzy Systems*, 12(1):25–33, 2002.

[J+16] G Jothi et al. Hybrid tolerance rough set–firefly based supervised feature selection for mri brain tumor image classification. *Applied Soft Computing*, 46:639–651, 2016.

[Jap03] Nathalie Japkowicz. Class imbalances: are we focusing on the right issue. In *Workshop on Learning from Imbalanced Data Sets II*, volume 1723, page 63, 2003.

[JBB15] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205. IEEE, 2015.

[JBS10] Kashif Javed, Haroon A Babri, and Mehreen Saeed. Feature selection based on class-dependent densities for high-dimensional binary data. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):465–477, 2010.

[JKL19] Liangxiao Jiang, Ganggang Kong, and Chaoqun Li. Wrapper framework for test-cost-sensitive feature selection. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[JKP94] George H John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 1994.

[JL08] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 219–230, New York, NY, USA, 2008. ACM.

[JLX16] Kun Jiang, Jing Lu, and Kuiliang Xia. A novel algorithm for imbalance data classification based on genetic algorithm improved smote. *Arabian Journal for Science and Engineering*, 41(8):3255–3266, Aug 2016.

[JLYX17] Yun Jiang, Xi Liu, Guolei Yan, and Jize Xiao. Modified binary cuckoo search for feature selection: a hybrid filter-wrapper ap-

proach. In *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pages 488–491. IEEE, 2017.

[JQL15] Liangxiao Jiang, Chen Qiu, and Chaoqun Li. A novel minority cloning technique for cost-sensitive learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(04):1551004, 2015.

[Kag14] Kaggle.com. Display advertising challenge, 2014.

[Kag15] Kaggle.com. Click-through rate prediction, 2015.

[Kag17] Kaggle.com. Advertising dataset, 2017.

[Kag18] Kaggle.com. Talkingdata adtracking fraud detection challenge, 2018.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[KB14] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, December 2014.

[KC02] Nojun Kwak and Chong-Ho Choi. Input feature selection for classification problems. *IEEE transactions on neural networks*, 13(1):143–159, 2002.

[KFCRB15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[KJ97] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[KK17] Chaouki Khammassi and Saoussen Krichen. A GA-LR wrapper approach for feature selection in network intrusion detection. *computers & security*, 70:255–277, 2017.

[KKL+14] J. Kwon, J. Kim, J. Lee, H. Lee, and A. Perrig. Psybog: Power spectral density analysis for detecting botnet groups. In *2014 9th In-*

ternational Conference on Malicious and Unwanted Software: The Americas (MALWARE), pages pp. 85–92, PR, USA, Oct 2014.

[KM+97] Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Nashville, USA, 1997.

[KMRW14] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. *Advances in Neural Information Processing Systems*, pages pp. 3581–9, 2014.

[KNB08] Brendan J Kitts, Tarek Najm, and Brian Burdick. Identifying automated click fraud programs, November 13 2008. US Patent App. 11/745,264.

[KNK+19] Zardad Khan, Muhammad Naeem, Umair Khalil, Dost Muhammad Khan, Saeed Aldahmani, and Muhammad Hamraz. Feature selection for binary classification within functional genomics experiments via interquartile range and clustering. *IEEE Access*, 7:78159–78169, 2019.

[Kon94] Igor Kononenko. Estimating attributes: analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer, 1994.

[Kot14] Fajri Koto. Smote-out, smote-cosine, and selected-smote: An enhancement strategy to handle imbalance in data level. In *2014 International Conference on Advanced Computer Science and Information System*, pages 280–284. IEEE, 2014.

[KPSS92] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159 – 178, 1992.

[KR+92] Kenji Kira, Larry A Rendell, et al. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134, 1992.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996.

[KTPA12] Kok-Chin Khor, Choo-Yee Ting, and Somnuk Phon-Amnuaisuk. A cascaded classifier approach for improving detection rates on rare attack categories in network intrusion detection. *Applied Intelligence*, 36(2):320–329, 2012.

[Kus99] Nicholas Kushmerick. Learning to remove internet advertisements. In *Agents*, pages 175–181. Citeseer, 1999.

[KV95] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238, 1995.

[KW10] Y. Kang and S. Won. Weight decision algorithm for oversampling technique on class-imbalanced learning. In *ICCAS 2010*, pages 182–186, Oct 2010.

[KW17] Michał Koziarski and Michał Woźniak. Ccr: A combined cleaning and resampling algorithm for imbalanced data classification. *International Journal of Applied Mathematics and Computer Science*, 27(4):727–736, 2017.

[KZRM13] B. Kitts, J. Y. Zhang, A. Roux, and R. Mills. Click fraud detection with bot signatures. In *2013 IEEE International Conference on Intelligence and Security Informatics*, pages pp. 146–150, WA, USA, June 2013.

[LCW10] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: Social honeypots + machine learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 435–442, New York, NY, USA, 2010. ACM.

[LES07] Cliff A.C. Lampe, Nicole Ellison, and Charles Steinfield. A familiar face(book): Profile elements as signals in an online social network. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 435–444, New York, NY, USA, 2007. ACM.

[LFZ15] J. Li, S. Fong, and Y. Zhuang. Optimizing smote by metaheuristics with neural network and decision tree. In *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pages 26–32, Dec 2015.

[LH15] Sungjin Lee and Yifan Hu. Joint embedding of query and ad by leveraging implicit feedback. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 482–491, 2015.

[Liu14] Yingchun Liu. Random forest algorithm in big data environment. *Computer Modelling & New Technologies*, 18(12A):147–151, 2014.

[LK17] Jonggeun Kim Lee, Hansoo and Sungshin Kim. Gaussian-based smote algorithm for solving skewed class distributions. *INTERNATIONAL JOURNAL of FUZZY LOGIC and INTELLIGENT SYSTEMS*, 17(4):229–234, 2017.

[LKL15] Jaedong Lee, Noori Kim, and Jee-Hyong Lee. An over-sampling technique with rejection for imbalanced class learning. In *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*, IMCOM '15, pages 102:1–102:6, New York, NY, USA, 2015. ACM.

[LLCX15] Wenhao Li, Haibo Li, Haibo Chen, and Yubin Xia. Adattester: Secure online mobile advertisement attestation using trustzone. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 75–88. ACM, 2015.

[LLS09] Ying Liu, Han Tong Loh, and Aixin Sun. Imbalanced text classification: A term weighting approach. *Expert systems with Applications*, 36(1):690–701, 2009.

[LLWS14] Hui Li, Chang-Jiang Li, Xian-Jun Wu, and Jie Sun. Statistics-based wrapper for feature selection: An implementation on financial distress identification with support vector machine. *Applied Soft Computing*, 19:57–67, 2014.

[LMC+16] Yixuan Li, Oscar Martinez, Xing Chen, Yi Li, and John E. Hopcroft. In a world that counts: Clustering and detecting fake social engagement at scale. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 111–120, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[LMJY17] Mahdieh Labani, Parham Moradi, Mahdi Jalili, and Xinghuo Yu. An evolutionary based multi-objective filter approach for feature se-

lection. In *2017 World Congress on Computing and Communication Technologies (WCCCT)*, pages 151–154. IEEE, 2017.

[LNGL14] B. Liu, S. Nath, R. Govindan, and J. Liu. Decaf: Detecting and characterizing ad fraud in mobile apps. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages pp. 57–70, Berkeley, CA, USA, 2014. USENIX Association.

[LPd+12] Sungyoung Lee, Young-Tack Park, Brian J d'Auriol, et al. A novel feature selection method based on normalized mutual information. *Applied Intelligence*, 37(1):100–120, 2012.

[LS+96] Huan Liu, Rudy Setiono, et al. A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer, 1996.

[LT13] Xiaoming Liu and Jinshan Tang. Mass classification in mammograms using selected geometry and texture features, and a new svm-based feature selection method. *IEEE Systems Journal*, 8(3):910–920, 2013.

[LTC+14] Victoria López, Isaac Triguero, Cristóbal J. Carmona, Salvador García, and Francisco Herrera. Addressing imbalanced classification with instance generation techniques: Ipade-id. *Neurocomputing*, 126:15 – 28, 2014. Recent trends in Intelligent Data Analysis Online Data Processing.

[LV11] Liang Lan and Slobodan Vucetic. Improving accuracy of microarray classification by a simple multi-task feature selection filter. *International journal of data mining and bioinformatics*, 5(2):189–208, 2011.

[LZ16] Mingxia Liu and Daoqiang Zhang. Pairwise constraint-guided sparse learning for feature selection. *IEEE Transactions on Cybernetics*, 46(1):298–310, 2016.

[LZL+14] Xin Li, Min Zhang, Yiqun Liu, Shaoping Ma, Yijiang Jin, and Liyun Ru. Search engine click spam detection based on bipartite graph propagation. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 93–102, New York, NY, USA, 2014. ACM.

[LZLF14] K. Li, W. Zhang, Q. Lu, and X. Fang. An improved smote imbalanced data classification method based on support degree. In *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*, pages 34–38, Oct 2014.

[LZWX13] Hu Li, Peng Zou, Xiang Wang, and Rongze Xia. A new combination sampling method for imbalanced data. In Zengqi Sun and Zhidong Deng, editors, *Proceedings of 2013 Chinese Intelligent Automation Conference*, pages 547–554, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[MAA07] A. Metwally, D. Agrawal, and A. EI Abbadi. Detectives: Detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages pp. 241–250, NY, USA, 2007. ACM.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[MAEA07] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Detectives: detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th international conference on World Wide Web*, pages 241–250. ACM, 2007.

[MAH+18] Majdi Mafarja, Ibrahim Aljarah, Ali Asghar Heidari, Hossam Faris, Philippe Fournier-Viger, Xiaodong Li, and Seyedali Mirjalili. Binary dragonfly optimization for feature selection using time-varying transfer functions. *Knowledge-Based Systems*, 161:185–204, 2018.

[Man00] Richard Mankiewicz. *The story of mathematics*. Cassell, 2000.

[MF17] Li Ma and Suohai Fan. Cure-smote algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests. *BMC Bioinformatics*, 18(1):169, Mar 2017.

[MG71] Anthony N Mucciardi and Earl E Gose. A comparison of seven techniques for choosing subsets of pattern recognition properties. *IEEE Transactions on Computers*, 100(9):1023–1031, 1971.

[MG03] Dunja Mladenić and Marko Grobelnik. Feature selection on hierarchy of web documents. *Decision Support Systems*, 35(1):45–87, 2003.

[MG16] Parham Moradi and Mozhgan Gholampour. A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy. *Applied Soft Computing*, 43:117–130, 2016.

[MLG+17] Lei Ma, Manchun Li, Yu Gao, Tan Chen, Xiaoxue Ma, and Lean Qu. A novel wrapper approach for feature selection in object-based image classification using polygon-based cross-validation. *IEEE Geoscience and Remote Sensing Letters*, 14(3):409–413, 2017.

[MM17] Majdi M Mafarja and Seyedali Mirjalili. Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing*, 260:302–312, 2017.

[MM18] Majdi Mafarja and Seyedali Mirjalili. Whale optimization approaches for wrapper feature selection. *Applied Soft Computing*, 62:441–453, 2018.

[MMAM14] S. Mahmoudi, P. Moradi, F. Akhlaghian, and R. Moradi. Diversity and separable metrics in over-sampling technique for imbalanced data classification. In *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 152–158, Oct 2014.

[Moo96] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, Nov 1996.

[MP11] A. Metwally and M. Paduano. Estimating the number of users behind ip addresses for combating abusive traffic. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages pp. 249–257, NY, USA, 2011. ACM.

[MS11] T. Maciejewski and J. Stefanowski. Local neighbourhood extension of smote for mining imbalanced data. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 104–111, April 2011.

[MS15a] Nour Moustafa and Jill Slay. The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems. *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 2015.

[MS15b] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[MS16] Nour Moustafa and Jill Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31, 2016.

[MS17] Nour Moustafa and Jill Slay. A hybrid feature selection for network intrusion detection systems: Central points. *arXiv preprint arXiv:1707.05505*, 2017.

[MSB08] Patrick Emmanuel Meyer, Colas Schretter, and Gianluca Bontempi. Information-theoretic feature selection in microarray data using variable complementarity. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):261–274, 2008.

[NB13] Theresa Hoang Diem Ngo and Warner Bros. The box-jenkins methodology for time series models. In *Proceedings of the SAS Global Forum 2013 conference*, volume 6, pages 1–11, 2013.

[NC09] Songyot Nakariyakul and David P Casasent. An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 42(9):1932–1940, 2009.

[NF77] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, (9):917–922, 1977.

[NHCD10] Feiping Nie, Heng Huang, Xiao Cai, and Chris H Ding. Efficient and robust feature selection via joint 2, 1-norms minimization. In *Advances in neural information processing systems*, pages 1813–1821, 2010.

[NKOK13] Munehiro Nakamura, Yusuke Kajiwara, Atsushi Otsuka, and Haruhiko Kimura. Lvq-smote–learning vector quantization based synthetic minority over–sampling technique for biomedical data. *BioData mining*, 6(1):16, 2013.

[NLY16] Iman Nekooeimehr and Susana K. Lai-Yuen. Adaptive semi-unsupervised weighted oversampling (a-suwo) for imbalanced datasets. *Expert Systems with Applications*, 46:405 – 416, 2016.

[NM09] Félix F González Navarro and Lluís A Belanche Muñoz. Gene subset selection in microarray data using entropic filtering for cancer classification. *Expert Systems*, 26(1):113–124, 2009.

[Nov16] Jasmina Novaković. Toward optimal feature selection using ranking methods and classification algorithms. *Yugoslav Journal of Operations Research*, 21(1), 2016.

[NRH+10] B. Nelson, B. Rubinstein, L. Huang, A. Joseph, S. Lee, S. Rao, and J. Tygar. Query strategies for evading convex-inducing classifiers. *CoRR*, abs/1007.0484, 2010.

[NSW10] Krystyna Napierała, Jerzy Stefanowski, and Szymon Wilk. Learning from imbalanced data in presence of noisy and borderline examples. In *International Conference on Rough Sets and Current Trends in Computing*, pages 158–167. Springer, 2010.

[NW16] A. Nisser and N. B. Weidman. Measuring ethnic preferences in bosnia and herzegovina with mobile advertising. *PLOS ONE*, 11(12):pp. 1–11, 12 2016.

[PDG+14] P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, V. Paxson, S. Savage, and G. M. Voelker. Characterizing large-scale click fraud in zeroaccess. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages pp. 141–152, NY, USA, 2014. ACM.

[Pfa95] Bernhard Pfahringer. *Compression-Based Feature Subset Selection.* 1995.

[PIGP+93] William F Punch III, Erik D Goodman, Min Pei, Lai Chia-Shun, Paul D Hovland, and Richard J Enbody. Further research on feature

selection and classification using genetic algorithms. In *ICGA*, pages 557–564, 1993.

[PK17] KR Pushpalatha and Asha Gowda Karegowda. Cfs based feature subset selection for enhancing classification of similar looking food grains-a filter approach. In *2017 2nd International Conference On Emerging Computation and Information Technologies (ICECIT)*, pages 1–6. IEEE, 2017.

[PLD05] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):1226–1238, 2005.

[PNK94] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125, 1994.

[ppc19] Ultimate list of ad click fraud statistics, 2019.

[PT17] Rifkie Primartha and Bayu Adhi Tama. Anomaly detection using random forest: A performance revisited. *2017 International Conference on Data and Software Engineering (ICoDSE)*, 2017.

[PVG+11] 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. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[PW12] Kamthorn Puntumapon and Kitsana Waiyamai. A pruning-based approach for searching precise and generalized region for synthetic minority over-sampling. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 371–382. Springer, 2012.

[QSZT15] Sijun Qin, Jia Song, Pengzhou Zhang, and Yue Tan. Feature selection for text classification based on part of speech filter and synonym merge. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 681–685. IEEE, 2015.

[QZL18] Xiaokang Qiu, Yuan Zuo, and Guannan Liu. Etcf: An ensemble model for ctr prediction. In *2018 15th International Conference*

on Service Systems and Service Management (ICSSSM), pages 1–5. IEEE, 2018.

[RCBH12] Enislay Ramentol, Yailé Caballero, Rafael Bello, and Francisco Herrera. Smote-rsb*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and information systems*, 33(2):245–265, 2012.

[RD18] Paulo Angelo Alves Resende and André Costa Drummond. Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. *Security and Privacy*, 1(4):e36, 2018.

[RGL+16] E. Ramentol, I. Gondres, S. Lajes, R. Bello, Y. Caballero, C. Cornelis, and F. Herrera. Fuzzy-rough imbalanced learning for the diagnosis of high voltage circuit breaker maintenance: The smote-frst-2t algorithm. *Engineering Applications of Artificial Intelligence*, 48:134 – 139, 2016.

[RGN14] Tongwen Rong, Huachang Gong, and Wing W. Y. Ng. Stochastic sensitivity oversampling technique for imbalanced data. In Xizhao Wang, Witold Pedrycz, Patrick Chan, and Qiang He, editors, *Machine Learning and Cybernetics*, pages 161–171, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[RH07] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature International Journal of Science*, pages pp. 323–536, 1986.

[Riv17] William A. Rivera. Noise reduction a priori synthetic over-sampling for class imbalanced data sets. *Information Sciences*, 408:146 – 161, 2017.

[RPN+14] Douglas Rodrigues, Luís AM Pereira, Rodrigo YM Nakamura, Kelton AP Costa, Xin-She Yang, André N Souza, and João Paulo Papa.

A wrapper approach for feature selection based on bat algorithm and optimum-path forest. *Expert Systems with Applications*, 41(5):2250–2258, 2014.

[RRAR06] Roberto Ruiz, José C Riquelme, and Jesús S Aguilar-Ruiz. Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recognition*, 39(12):2383–2392, 2006.

[RSF17] I. Riadi, Sunardi, and A. Firdonsyah. Forensic investigation technique on android's blackberry messenger using nist framework. *International Journal of Cyber-Security and Digital Forensics*, 6(4):pp. 198+, October 2017.

[RX16] William A. Rivera and Petros Xanthopoulos. A priori synthetic over-sampling methods for increasing classification sensitivity in imbalanced data sets. *Expert Systems with Applications*, 66:124 – 135, 2016.

[S.19] Thejas G. S. *Dr. S.S. Iyengar: Four decades of Contribution towards Research and Educational Enhancement Between USA & India*. Miami, FL, USA, July 2019.

[SAM+18] Indira Sen, Anupama Aggarwal, Shiven Mian, Siddharth Singh, Ponnurangam Kumaraguru, and Anwitaman Datta. Worth its weight in likes: Towards detecting fake likes on instagram. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '18, pages 205–209, New York, NY, USA, 2018. ACM.

[SCML13] Newton SpolaôR, Everton Alvares Cherman, Maria Carolina Monard, and Huei Diana Lee. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151, 2013.

[Scu10] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[SE99] Yuhui Shi and Russell C Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1945–1950. IEEE, 1999.

[SF12] Sebastian Student and Krzysztof Fujarewicz. Stable feature selection and classification algorithms for multiclass microarray data. *Biology direct*, 7(1):33, 2012.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–58, 2014.

[SHPC10] B. Suh, L. Hong, P. Pirolli, and E. H. Chi. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *2010 IEEE Second International Conference on Social Computing*, pages 177–184, Aug 2010.

[SIL07] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[SIM11] Alok Sharma, Seiya Imoto, and Satoru Miyano. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):754–764, 2011.

[Ska94] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning Proceedings 1994*, pages 293–301. Elsevier, 1994.

[SLG18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116, 2018.

[SLSH15] José A. Sáez, Julián Luengo, Jerzy Stefanowski, and Francisco Herrera. Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184 – 203, 2015.

[SMG13] Atlantida I Sanchez, Eduardo F Morales, and Jesus A Gonzalez. Synthetic oversampling of instances using clustering. *International Journal on Artificial Intelligence Tools*, 22(02):1350008, 2013.

[SMKR13] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for

lvcsr. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8614–8618. IEEE, 2013.

[SMQ+14] Yongkui Sun, Lei Ma, Na Qin, Meilan Zhang, and Qianyong Lv. Analog filter circuits feature selection using mrmr and svm. In *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, pages 1543–1547. IEEE, 2014.

[SP13] Mauricio Schiezaro and Helio Pedrini. Data feature selection based on artificial bee colony algorithm. *EURASIP Journal on Image and Video Processing*, 2013(1):47, 2013.

[SRUed] Thejas G. S., Rishabh Ritweek, and Utkarsh. Information feedster, India. Patent 3989/CHE/2014, filed Aug. 2014, Awarded.

[SS17] Wacharasak Siriseriwan and Krung Sinapiromsaran. Adaptive neighbor synthetic minority over-sampling technique under 1nn outcast handling. *Songklanakarin J. Sci. Technol*, 39:565–576, 2017.

[SSA+19] Sadia Sharmin, Mohammad Shoyaib, Amin Ahsan Ali, Muhammad Asif Hossain Khan, and Oksam Chae. Simultaneous feature selection and discretization based on mutual information. *Pattern Recognition*, 91:162–174, 2019.

[SSG+12] Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13(May):1393–1434, 2012.

[Stu08] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[SW65] SS Shaphiro and MB Wilk. An analysis of variance test for normality. *Biometrika*, 52(3):591–611, 1965.

[SW08] Jerzy Stefanowski and Szymon Wilk. Selective pre-processing of imbalanced data for improving classification performance. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 283–292. Springer, 2008.

[SZWQ12] Senzhang Wang, Zhoujun Li, Wenhan Chao, and Qinghua Cao. Applying adaptive over-sampling technique based on data density and

cost-sensitive svm to imbalanced learning. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, June 2012.

[TA15] Divya Tomar and Sonali Agarwal. Hybrid feature selection based weighted least squares twin support vector machine approach for diagnosing breast cancer, hepatitis, and diabetes. *Advances in Artificial Neural Systems*, 2015:1, 2015.

[TAL14] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data classification: Algorithms and applications*, page 37, 2014.

[Tay10] James W. Taylor. Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, 204(1):139 – 152, 2010.

[TBC+19] G. S. Thejas, Kianoosh G. Boroojeni, Kshitij Chandna, Isha Bhatia, S. S. Iyengar, and N. R. Sunitha. Deep learning-based model to fight against ad click fraud. In *Proceedings of the 2019 ACM Southeast Conference*, ACM SE '19, pages 176–181, New York, NY, USA, 2019. ACM.

[TC08] Sheng Tang and Si-ping Chen. The generation mechanism of synthetic minority class examples. In *2008 International Conference on Information Technology and Applications in Biomedicine*, pages 444–447. IEEE, 2008.

[TCOMT16] Fredy Rodríguez Torres, Jesús A. Carrasco-Ochoa, and José Fco. Martínez-Trinidad. Smote-d a deterministic version of smote. In José Francisco Martínez-Trinidad, Jesús Ariel Carrasco-Ochoa, Victor Ayala Ramirez, José Arturo Olvera-López, and Xiaoyi Jiang, editors, *Pattern Recognition*, pages 177–188, Cham, 2016. Springer International Publishing.

[TDI+19] G. S. Thejas, Surya Dheeshjith, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. CFXGB: An optimized and effective learning approach for click fraud detection. *ACM Transactions on Intelligent Systems and Technology*, 2019. Under Review.

[TDK10] F Boray Tek, Andrew G Dempster, and Izzet Kale. Parasite detection and identification for automated thin blood film malaria di-

agnosis. *Computer vision and image understanding*, 114(1):21–32, 2010.

[TGI+19] G. S. Thejas, Rameshwar Grag, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. MRFI and ARFI: Hybrids of filter and wrapper feature selection approaches. *IEEE Access*, 2019. In submission preparation.

[TH15] B. Tang and H. He. Kerneladasyn: Kernel based adaptive synthetic data generation for imbalanced learning. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 664–671, May 2015.

[THI+19] G. S. Thejas, Yashas Hariprasad, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. K-SMOTE: An extension of synthetic minority over-sampling technique for imbalanced datasets. *IEEE Transactions on Knowledge and Data Engineering*, 2019. Under Review.

[TJI+19] G. S. Thejas, Sajal Raj Joshi, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. Mini-batch normalized mutual information: A hybrid feature selection method. *IEEE Access*, 2019.

[TJI+20] G. S. Thejas, Daniel I. Jimenez, S. S. Iyengar, Jerry Miller, N. R. Sunitha, and Prajwal Badrinath. COMB: A hybrid variant of feature selection technique. In *In 2020 ACM Southeast Conference (ACMSE 2020)*, New York, USA, 2020. ACM. Under Review.

[TKC+19] G. S. Thejas, G.Boroojeni Kianoosh, Kshitij Chandna, Isha Bhatia, S. S. Iyengar, and N. R. Sunitha. Deep learning-based model to fight against ad click fraud. In *2019 ACM Southeast Conference (ACMSE 2019)*, ACM '19, New York, NY, USA, 2019. ACM.

[TKI+19] G. S. Thejas, Kundan Kumar, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. AI-NLP analytics: A thorough comparative investigation on india-usa universities branding on the trending social media platform instagram. In *Proceedings of IEEE 4th International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS-19)*, Karnataka, India, 2019. IEEE.

[TPIS18] G. S. Thejas, T. C. Pramod, S. S. Iyengar, and N. R. Sunitha. Intelligent access control: A self-adaptable trust-based access

control (SATBAC) framework using game theory strategy. In Nageswara S.V. Rao, Richard R. Brooks, and Chase Q. Wu, editors, *Proceedings of International Symposium on Sensor Networks, Systems and Security*, pages pp. 97–111, Cham, 2018. Springer International Publishing.

[TSB+19] G. S. Thejas, Jayesh Soni, Kianoosh G. Boroojeni, S. S. Iyengar, Kanishk S, Prajwal Badrinath, N. R. Sunitha, Nagarajan Prabhakar, and Himanshu Upadyaya. A multi-time-scale time series analysis for click fraud forecasting using binary labeled imbalanced dataset. In *Proceedings of IEEE 4th International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS-19)*, Karnataka, India, 2019. IEEE.

[TSC+19] G. S. Thejas, J. Soni, K. Chandna, S. S. Iyengar, N. R. Sunitha, and N. Prabakar. Learning-based model to fight against fake like clicks on instagram posts. In *IEEE SoutheastCon 2019*, pages pp. 1–8, Huntsville, Alabama, USA, April 2019. In press.

[TW99] Kai Ming Ting and Ian H Witten. Issues in stacked generalization. *Journal of artificial intelligence research*, 10:271–289, 1999.

[TYH+15] Vincent S. Tseng, Jia-Ching Ying, Che-Wei Huang, Yimin Kao, and Kuan-Ta Chen. Fraudetector: A graph-mining-based framework for fraudulent phone call detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 2157–2166, New York, NY, USA, 2015. ACM.

[TZX+15] Tian Tian, Jun Zhu, Fen Xia, Xin Zhuang, and Tong Zhang. Crowd fraud detection in internet advertising. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1100–1110, Republic and Canton of Geneva, Switzerland, 2015. International World Wide Web Conferences Steering Committee.

[unb17] unb.ca. Intrusion detection evaluation dataset (cicids2017), 2017.

[unk70] On the dangers of auc, Jan 1970.

[uns15] unsw.adfa.edu.au. Unsw-nb15 dataset, 2015.

[VA19] B Venkatesh and J Anuradha. A hybrid feature selection approach for handling a high-dimensional data. In *Innovations in Computer Science and Engineering*, pages 365–373. Springer, 2019.

[VAK+16] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. Ai 2: Training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud, HPSC, and IDS*, pages pp. 49–54, NY, USA, April 2016.

[VB17] Svitlana Volkova and Eric Bell. Identifying effective signals to predict deleted and suspended accounts on twitter across languages. In *ICWSM*, 2017.

[VJ+01] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3, 2001.

[VLBM08] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th Int. Conf. on Machine Learning*, ICML '08, pages pp. 1096–1103, NY, USA, 2008. ACM.

[VV16] D Franklin Vinod and V Vasudevan. A filter based feature set selection approach for big data classification of patient records. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 3684–3687. IEEE, 2016.

[VVERA+16] Véronique Van Vlasselaer, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Gotcha! network-based fraud detection for social security fraud. *Management Science*, 63(9):3090–3110, 2016.

[Web00] Geoffrey I Webb. Multiboosting: A technique for combining boosting and wagging. *Machine learning*, 40(2):159–196, 2000.

[WF18] Youwei Wang and Lizhou Feng. Hybrid feature selection using component co-occurrence based feature relevance measurement. *Expert Systems with Applications*, 102:83–99, 2018.

[WFFW17] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, page 12. ACM, 2017.

[WGNdPB13] Maria Fernanda B Wanderley, Vincent Gardeux, René Natowicz, and Antônio de Pádua Braga. Ga-kde-bayes: an evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In *ESANN*, 2013.

[Whi71] A Wayne Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 100(9):1100–1103, 1971.

[WWK+13] Jianzhong Wang, Lishan Wu, Jun Kong, Yuxin Li, and Baoxue Zhang. Maximum weight and minimum redundancy: a novel framework for feature subset selection. *Pattern Recognition*, 46(6):1616–1627, 2013.

[WXWZ06] Juanjuan Wang, Mantao Xu, Hui Wang, and Jiwu Zhang. Classification of imbalanced data by using the smote algorithm and locally linear embedding. In *2006 8th international Conference on Signal Processing*, volume 3. IEEE, 2006.

[XJWX19] Ying Xie, Dan Jiang, Xinmei Wang, and Rongbin Xu. Robust transfer integrated locally kernel embedding for click-through rate prediction. *Information Sciences*, 491:190–203, 2019.

[XJYL15] Zhipeng Xie, Liyang Jiang, Tengju Ye, and Xiaoli Li. A synthetic minority oversampling method based on local densities in low-dimensional space for imbalanced learning. In Matthias Renz, Cyrus Shahabi, Xiaofang Zhou, and Muhammad Aamir Cheema, editors, *Database Systems for Advanced Applications*, pages 3–18, Cham, 2015. Springer International Publishing.

[XLLT14] Yu Hui Xu, Hui Li, Lu Ping Le, and Xiao Yun Tian. Neighborhood triangular synthetic minority over-sampling technique for imbalanced prediction on small samples of chinese tourism and hospitality firms. In *2014 Seventh International Joint Conference on Computational Sciences and Optimization*, pages 534–538, July 2014.

[XYW18] Xiaowei Xue, Min Yao, and Zhaohui Wu. A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm. *Knowledge and Information Systems*, 57(2):389–412, 2018.

[YH98] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.

[YHL16] Jaesub Yun, Jihyun Ha, and Jong-Seok Lee. Automatic determination of neighborhood size in smote. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, IMCOM '16, pages 100:1–100:8, New York, NY, USA, 2016. ACM.

[YL03] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[YLZ+13] Pengyi Yang, Wei Liu, Bing B Zhou, Sanjay Chawla, and Albert Y Zomaya. Ensemble-based wrapper methods for feature selection and class imbalance learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 544–555. Springer, 2013.

[YMZ17] Akhiat Yassine, CHAHHOU Mohamed, and Ahmed Zinedine. Feature selection based on pairwise evalution. In *2017 Intelligent Systems and Computer Vision (ISCV)*, pages 1–6. IEEE, 2017.

[YNWC15] William A. Young, Scott L. Nykl, Gary R. Weckman, and David M. Chelberg. Using voronoi diagrams to improve classification performances when modeling imbalanced datasets. *Neural Computing and Applications*, 26(5):1041–1054, Jul 2015.

[ZE+11] Paul Zikopoulos, Chris Eaton, et al. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[ZF17] Zhi-Hua Zhou and Ji Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.

[Zho12] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[ZL09] Zheng Zhao and Huan Liu. Searching for interacting features in subset selection. *Intelligent Data Analysis*, 13(2):207–228, 2009.

[ZL14] Huaxiang Zhang and Mingfang Li. Rwo-sampling: A random walk over-sampling approach to imbalanced data classification. *Information Fusion*, 20:99 – 116, 2014.

[ZOD07] Zexuan Zhu, Yew-Soon Ong, and Manoranjan Dash. Wrapper–filter feature selection algorithm using a memetic framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):70–76, 2007.

[ZW11] L. Zhang and W. Wang. A re-sampling method for class imbalance learning with credit data. In *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, volume 1, pages 393–397, Sep. 2011.

[ZXM19] Jixiong Zhang, Yanmei Xiong, and Shungeng Min. A new hybrid filter/wrapper algorithm for feature selection in classification. *Analytica Chimica Acta*, 2019.

[ZYGH13] B. Zhou, C. Yang, H. Guo, and J. Hu. A quasi-linear svm combined with assembled smote for imbalanced data classification. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Aug 2013.

[ZYL14] Bichen Zheng, Sang Won Yoon, and Sarah S Lam. Breast cancer diagnosis based on feature extraction using a hybrid of k-means and support vector machine algorithms. *Expert Systems with Applications*, 41(4):1476–1482, 2014.

[ZZC+18] Xiaolong Zhang, Qing Zhang, Miao Chen, Yuantao Sun, Xianrong Qin, and Heng Li. A two-stage feature selection and intelligent fault diagnosis method for rotating machinery using hybrid filter and wrapper method. *Neurocomputing*, 275:2426–2439, 2018.

VITA

THEJAS GUBBI SADASHIVA

2016–Present	Ph.D., Computer Science Discovery Lab School of Computing and Information Sciences Florida International University, Miami, Florida
2011–2016	Assistant Professor Dept. of Computer Science and Engineering Siddaganga Institute of Technology Tumakuru, Karnataka, India
2010–2011	Trainee "E" RADAR division Electronics and Radar Development Establishment (LRDE) Defence Research & Development Organization (DRDO)
2009–2011	Master of Technology, Computer Science Dept. of Computer Science and Engineering Sri Siddhartha Institute of Technology (SSIT) Tumakuru, Karnataka, India
2005–2009	Bachelor of Engineering, Computer Science Dept. of Computer Science and Engineering Sri Siddhartha Institute of Technology (SSIT) Visvesvaraya Technological University (VTU) Tumakuru, Karnataka, India

PATENT, BOOK, PUBLICATIONS AND PRESENTATIONS

Patent Thejas G. S., Rishabh Ritweek, and Utkarsh. Information feedster, India. Patent 3989/CHE/2014, filed Aug. 2014, Awarded

Book Thejas G. S. *Dr. S.S. Iyengar: Four decades of Contribution towards Research and Educational Enhancement Between USA & India.* Miami, FL, USA, July 2019

- G. S. Thejas, Sajal Raj Joshi, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. Mini-batch normalized mutual information: A hybrid feature selection method. *IEEE Access*, 2019
- G. S. Thejas, G.Boroojeni Kianoosh, Kshitij Chandna, Isha Bhatia, S. S. Iyengar, and N. R. Sunitha. Deep learning-based model to fight against ad click fraud. In *2019 ACM Southeast Conference (ACMSE 2019)*, ACM '19, New York, NY, USA, 2019. ACM

- G. S. Thejas, J. Soni, K. Chandna, S. S. Iyengar, N. R. Sunitha, and N. Prabakar. Learning-based model to fight against fake like clicks on instagram posts. In *IEEE SoutheastCon 2019*, pages pp. 1–8, Huntsville, Alabama, USA, April 2019. In press
- G. S. Thejas, T. C. Pramod, S. S. Iyengar, and N. R. Sunitha. Intelligent access control: A self-adaptable trust-based access control (SATBAC) framework using game theory strategy. In Nageswara S.V. Rao, Richard R. Brooks, and Chase Q. Wu, editors, *Proceedings of International Symposium on Sensor Networks, Systems and Security*, pages pp. 97–111, Cham, 2018. Springer International Publishing
- G. S. Thejas, Jayesh Soni, Kianoosh G. Boroojeni, S. S. Iyengar, Kanishk S, Prajwal Badrinath, N. R. Sunitha, Nagarajan Prabhakar, and Himanshu Upadyaya. A multi-time-scale time series analysis for click fraud forecasting using binary labeled imbalanced dataset. In *Proceedings of IEEE 4th International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS-19)*, Karnataka, India, 2019. IEEE
- G. S. Thejas, Kundan Kumar, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. AI-NLP analytics: A thorough comparative investigation on india-usa universities branding on the trending social media platform instagram. In *Proceedings of IEEE 4th International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS-19)*, Karnataka, India, 2019. IEEE
- G. S. Thejas, Yashas Hariprasad, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. K-SMOTE: An extension of synthetic minority over-sampling technique for imbalanced datasets. *IEEE Transactions on Knowledge and Data Engineering*, 2019. Under Review
- G. S. Thejas, Surya Dheeshjith, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. CFXGB: An optimized and effective learning approach for click fraud detection. *ACM Transactions on Intelligent Systems and Technology*, 2019. Under Review
- G. S. Thejas, Rameshwar Grag, S. S. Iyengar, N. R. Sunitha, and Prajwal Badrinath. MRFI and ARFI: Hybrids of filter and wrapper feature selection approaches. *IEEE Access*, 2019. In submission preparation
- G. S. Thejas, Daniel I. Jimenez, S. S. Iyengar, Jerry Miller, N. R. Sunitha, and Prajwal Badrinath. COMB: A hybrid variant of feature selection technique. In *In 2020 ACM Southeast Conference (ACMSE 2020)*, New York, USA, 2020. ACM. Under Review