Fall 2-8-2024

Customer Churn Prediction Based on Sentiment Score

Shadha Al-Safi
shadahamed@aucegypt.edu

Graduate Studies

Customer Churn Prediction Based on Sentiment Score

A Thesis Submitted by

Shadha Hamed Al-Safi

to the

Department of Computer science and engineering

2/7/2024

in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science

Declaration of Authorship

I, Shadha Al-Safi, declare that this thesis titled, "[Thesis title]" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Shadha Hamed Al-Safi

Date:

2/8/2024

Abstract

In recent years, the telecommunications industry has witnessed intensified competition, wherein the expense associated with acquiring new consumers exceeds that of sustaining existing ones. Consequently, predicting customer churn prior to its occurrence has become essential. This study proposes a sentiment-based customer churn prediction model in which the sentiment of customers is predicted using Random Forest. Subsequently, the derived sentiment predictions are combined with additional features to predict customer churn. The ensemble technique is applied to predict churn, consisting of K-nearest neighbors, Support Vector Machines, Random Forest as base learners, and Multiple Layer Perceptron as a meta learner. Moreover, mutual information is applied to select the pertinent features impacting customer churn, and the class imbalance is handled through the utilization of the class weighted technique. The results of the experiments reveal that the proposed model surpassed the state-of-the-art customer churn models as it achieved an accuracy of 98.86%, an AUC of 99.47 %, and an F1-score of 97.77%.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Ahmed Rafea and Dr. Mona Farouk, for their invaluable guidance throughout the project duration and for providing constructive feedback that greatly contributed to the quality of the content in this thesis.

I am filled with immeasurable appreciation and deepest gratitude for my husband and my children, whose love and support made it possible for me to complete this thesis successfully. Their unwavering encouragement has been my driving force.

A special thanks to my family, whose prayers and encouragement have served as a constant source of motivation, inspiring me to give my best in this endeavors.

Lastly, I extend my heartfelt thanks to all individuals who played a role in supporting and encouraging me during my graduate studies.

Shadha Hamed Al-Safi

Contents

List of Figures

List of Tables

List of Abbreviations

CCP	**Customer Churn Prediction**
KNN	**K-nearest Neighbor**
MI	**Mutual Information**
RF	**Random Forest**
SVM	**Support Vector Machine**

Chapter 1

Introduction

1.1 Problem definition

Customer churn is the act of customers ending their association with a company or service provider [1]. This phenomenon not only diminishes enterprises' revenue but also impacts their competitive standing [2]. Furthermore, it is well established that acquiring new clients demands a substantial investment of resources, surpassing the cost of retaining existing clients [3]. To mitigate customer attrition and enhance revenue, telecom operators must formulate efficient marketing plans grounded in comprehensive consumer data.

In this context, churn management becomes imperative in the telecommunications industry. Various solutions have been proposed to address the issue of churn, with predicting customer churn being the most widely adopted. Many researchers and enterprise analysts aim to identify churn before it happens to mitigate its adverse effects. The primary objective of predicting customer churn is to assist companies in developing effective strategies for customer retention and industry revenue growth. Today, telecommunications corporations possess a wealth of customer information, including demographic details (age, gender, marital status, etc.), profiles, call logs, voicemail messages, short messages, financial information, and other interaction details with the service [4]. This information is critical for predicting customer churn accurately.

The utilization of demographic information, specifically age and gender, is common in predicting churn [5]. Studies have indicated that customers below the age of 45 are more likely to churn. Additionally, they have demonstrated the influence of familial and social connections within the same telecom industry, revealing that customers are more likely to churn if someone with whom they share a social relationship has already churned. The studies also encompassed other service-related variables. For instance, factors such as call prices, call frequency, and call duration were considered in predicting customer churn.

Two methods are commonly employed for dealing with client churn: (1) reactively and (2) proactively. In the reactive method, the company waits for customers to signal their intention to leave before offering tempting retention incentives. On the other hand, the proactive technique anticipates turnover and provides suitable incentives beforehand. In essence, the proactive approach is framed as a binary classification problem distinguishing churners and non-churners [4]. Numerous modules have been developed to differentiate

churners from non-churners, with a predominant focus on constructing machine learning and deep learning models [1, 3]. Single models, hybrid techniques, and optimized classification methods are widely implemented [6].

Furthermore, additional approaches have been devised to delve deeper into customer churn prediction, such as identifying the reasons behind leaving the company and understanding the factors influencing customer churn. Issues such as poor customer service, high service costs, and low service quality are prevalent factors contributing to churn [7]. [8] categorized the reasons for churn into uncontrollable, controllable, and unknown categories. For instance, customer behavior in cases of uncontrollable reasons like death or location changes is not apparent, as customers may be satisfied with services but compelled to churn. This category may not accurately indicate customer churn, as it was not previously known. On the other hand, controllable causes enhance predictive performance in identifying churners when incorporated into the prediction model.

The sentiment of customers and their satisfaction score are highly correlated with customer churn [5]. Several studies were conducted to explore the relationship between sentiment and churn. In a study by [9], the correlation was examined using Pearson correlation, where the authors monitored the value of the correlation between the sentiment growth rate of a company over a specific period and the churn rate for this company in the same period. They found a strong relationship between the sentiment rate and the churn rate. Another investigation by [10] demonstrated the association between negative sentiment and customer churn by analyzing the tweets of five banks to identify weaknesses and issues. They asserted that tweets that involved terms related to churn were included within the tweets that were classified as negative sentiment. [5] conducted a further study to establish a link between sentiment and customer churn. The study worked on Twitter mining to predict customer loss in Saudi telecommunication companies. The researchers claimed that customer feedback and opinions shared on social media influence the decision of customers to stay or leave, so they incorporated the customer satisfaction rate of customers in the company as a variable in churn prediction models.

1.2 Motivation

Anticipating customer churn represents a well-established concept within the field of business analytics. Numerous models have been developed to refine predictive capabilities and identify potential churners promptly, thereby mitigating the risk of customer churn. Nevertheless, the integration of sentiment analysis for the purpose of predicting customer churn stands out as a compelling and indispensable endeavor for businesses aiming to thrive in today's competitive landscape.

The sentiment of customers measures how customers feel about products or services. This valuable information equips companies with the means to systematically monitor customer experiences and pinpoint areas of concern. Identification of these critical pain points empowers relevant organizational units to make informed decisions aimed at optimizing the customer experience. One of the main advantages of using customer sentiment is that it allows companies to detect issues early. When a substantial number of customers express negative sentiments about a specific product or service, it serves as an early warning sign, enabling proactive

decision-making to prevent potential churn.

In the context of telecommunications enterprises navigating intense global competition, the pursuit of customer satisfaction emerges as a key motivator. An abundance of research has demonstrated a beneficial relationship between customer satisfaction and both customer loyalty and customer attrition. Through a nuanced understanding of customer sentiments, companies can transition from reactive to proactive customer retention strategies. Ultimately, this tactical change builds a strong customer base, giving these businesses a competitive advantage in the dynamic and constantly changing market environment.

The underlined motivation for this thesis lies in the exploration of the consequential impact of sentiment analysis on predicting customer churn. Specifically, this thesis seeks to elaborate on how incorporating the sentiment of customers as a variable input into the model of churn facilitates the early detection of customers at risk of churn.

1.3 Research Objective

This thesis has two objectives:

1. Investigate the impact of using customer sentiments represented in different forms (positive/negative, discrete positive/negative/neutral, or as discrete value on a scale of 1 to 5) on the quality of churn prediction.

2. Assess the performance improvement of the churn prediction after doing the following:
   a. Using a class_weighted approach to handle the enhancement of minority class classification,
   b. Using an ensemble method

To achieve these objectives, multiple research questions are proposed:

1. Does the ensemble method enhance the churn prediction?
2. Does including sentiment in different forms affect the customer churn prediction results?
3. What is the best number of sentiment level to predict churn?
4. Does handling imbalanced data improve the performance of the classification of the minority class?

1.4 Thesis layout

The rest of the thesis is organized as follows:

Chapter 2: Literature Review: In this chapter, we will address the related work to investigate what has already been done in the field and explore the state-of-the-art technologies utilized to solve the problem.
Chapter 3: Research methodology: In this chapter, we will define the database used and the techniques of preprocessing. We will also go through the steps of the model setup and the evaluation metrics used.
Chapter 4: Experiments and Results: In this chapter, we will provide our findings, discuss them, and construct comparisons.

Chapter 2

Literature Review

This chapter discusses various approaches employed for customer churn prediction, such as machine learning, deep learning, and ensemble approaches, as well as the research that used sentiment as a feature to predict customer churn. Then, we will conclude with the final remark.

2.1. Machine learning approach

In the field of customer churn prediction (CCP), researchers commonly employ a machine learning (ML) approach to identify customers at risk of churning [1]. Their efforts are directed towards refining prediction models and conducting comparative analyses. For instance, [2] investigated the efficacy of various ML techniques in addressing the challenge of customer churn. The study compared advanced ML methods, namely Random Forest (RF), Support Vector Machines (SVM), and K-nearest neighbors (KNN), along with their optimized algorithms. The findings suggested that the performance of ML methods varies across different problem domains. Notably, optimizing hyperparameters and handling imbalanced data were identified as key strategies leading to improved results in multiple experiments. In the next subsection, we will delve into a discussion of the most popular ML techniques utilized for customer churn.

2.1.1. Tuned Kernel Support Vector Machines (SVM):

SVM has been employed in numerous studies, including [3], which aim to identify the best SVM kernel for churn-related issues. The authors conducted multiple kernel SVM models to determine the most suitable one through various experiments involving hyperparameter tuning and handling imbalanced data. They trained the models on the Orange Company dataset, which included customer activity data regarding calls and SMSs. Their findings demonstrated that RBF kernels performed the best after tuning, dimensionality reduction, using Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS), and handling imbalanced data using the Synthetic Minority Oversampling Technique (SMOTE), achieving 98% on the Orange Company dataset.

2.1.2. CHAID Decision Tree

In the study [7], the authors combined clustering and classification to use a hybrid approach for churn prediction. Their approach comprises three stages, which are data preparation, clustering, and churn prediction. In the data preparation stage, they applied an IBM dataset that included 7043 samples and 21 features. The authors chose the IBM dataset because it has a lot of different variables, such as gender, age, number of dependents, partners, and senior citizens; how the services are used (Internet service, device protection, online backup, online security, streaming movies, streaming TV, and technical support; multiple lines); how the customers pay (contract, paperless billing, and payment method); how much money the

customers are worth; and the length of time they have been customers (tenure variable); and churn behavior, indicating churn as 0 or 1. All these features are nominal or binomial except for the customer monetary value and tenure group, which are numeric variables.

The second stage involves clustering. In this stage, the K-mean clustering algorithm was applied to group customers based on data variables, excluding customer ID and churn variables. Each customer was assigned to a particular cluster based on Euclidean distance. The difference between clusters was determined through ANOVA analysis for numeric variables and Chi-square analysis for nominal variables. Each cluster exhibited a unique distribution of feature values. For every cluster, the authors calculated the number of churns and non-churns, providing the percentage of churners in each cluster relative to the total number of churners.

In the third stage, customer churn prediction was performed using a decision tree at both the overall data level and each cluster level. Feature selection for the decision tree involved utilizing the chi-square and p-value. The chi-square test finds if two categorical variables are related by adding up the squared differences between the observed and expected frequencies, with the expected frequency being used as a standard. The p-value represents the probability of obtaining a specific set of observations if the null hypothesis were true. The two techniques revealed that gender, phone service, and customer ID were not deemed important predictors for churn and were subsequently removed. The overall churn prediction accuracy for the full dataset was 79%, while for the six clusters, it varied with accuracies of 89%, 71.3%, 68.7%, 88.6%, 92.6%, and 70.8%, respectively.

2.1.3. Adaptive Customer Churn Prediction (ACCP)

A study conducted by [8] proposes an adaptive learning approach for churn prediction using the Naïve Bayes base learner classifier with a genetic algorithm-based feature weighting approach. Both the genetic algorithm and feature weights contribute to optimizing the Naive Bayes baseline performance.

The genetic algorithm seeks to find the optimal solution by creating an initial population and initializing individuals randomly based on the available data. Subsequently, the fitness of each individual, representing a set of parameters, is evaluated. The selection process involved choosing parameters with high fitness values as partners. The genetic information for parents was combined through crossover, and random changes were introduced via mutation.

Another optimization for the model involves assigning weights to features instead of selecting a subset of them to prevent data loss. To determine optimal weights for accurate class label prediction, the researchers also employed the genetic algorithm. However, their study did not utilize any technique to handle imbalanced data.

The model was applied to three public datasets, which were the BigML Telecom dataset comprising 3333 customer records, the IBM Telco dataset with 7043 customer records, and the Cell2Cell dataset, which has 51047 observations. The proposed approach surpassed multiple base learners, such as Naïve Bayes with the default setting, Deep-BP-ANN, CNN, Neural Network, Linear Regression, XGBoost, KNN, Logit Boost, SVM, and PCALB methods, achieving accuracies of 95%, 97%, and 98.5% when applied to the three datasets, respectively.

2.2. Deep Learning

Beyond traditional machine learning (ML) methods, the utilization of a deep learning approach has gained prominence to enhance prediction accuracy and construct more effective models, particularly in dealing with big data. A typical DL model involves multiple layers of neural networks built on top of each other to implement a forward pass and a backward pass. The main advantage of the DL model is its adaptability to learn from diverse and large datasets. It also allows for end-to-end learning, where the model learns directly from raw data to produce the desired output, which reduces the need for extensive preprocessing. [11] emphasized that by applying a DL approach, businesses can gain clear insights into customer behavior, optimize customer distribution, and construct more accurate models for predicting churners.

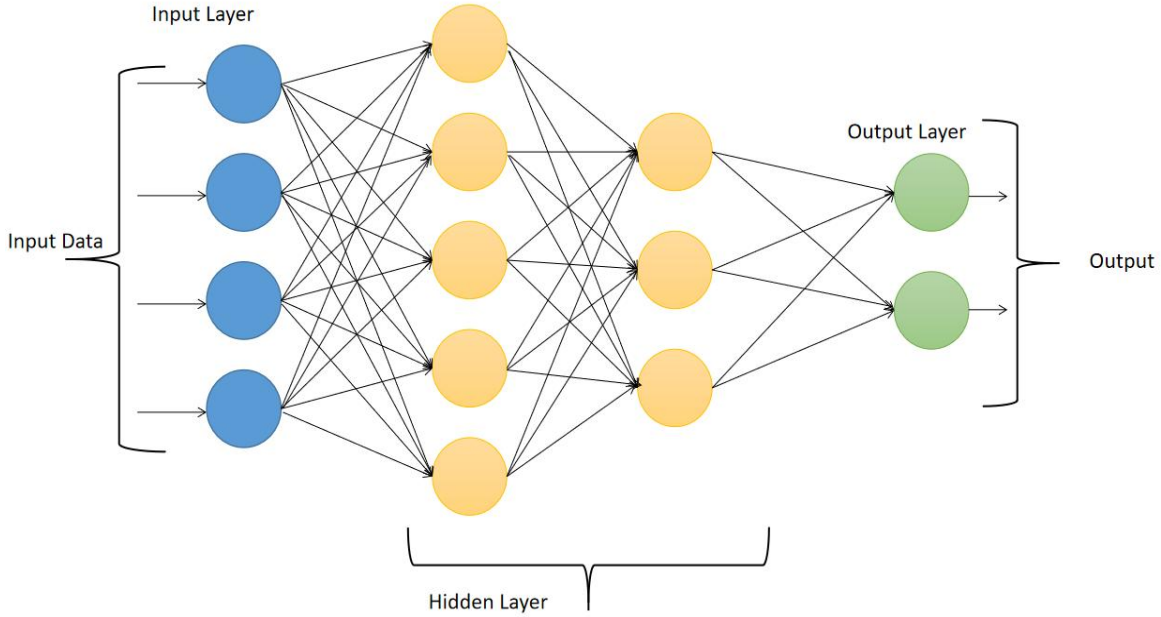


Figure 1: Basic DL architectural

2.2.1. Deep-BP-ANN

The model proposed by [11] incorporates an early stopping technique to prevent overfitting, along with two feature selection methods: Variance Thresholding [12] and Lasso Regression [13].

According to variance thresholding, the authors generated a correlation matrix to assess the correlation between variables. Features with a correlation coefficient above 0.5 or equal were excluded from the dataset. On the other hand, the Lasso Regression works by introducing a penalty term to the standard linear regression objective function based on the absolute values of the coefficients multiplied by a regularization parameter (alpha). Features corresponding to non-zero coefficients in the model are considered important, while those with zero coefficients are effectively excluded.

To handle imbalanced data, the authors applied a Random Over Sampling technique, involving the random reproduction of minority instances from the current dataset. The model's performance was evaluated on two datasets: the Cell2Cell dataset, consisting of 71,047 instances and 58 attributes, and the IBM dataset, with 7043 instances and 21 attributes. The authors

claimed that their model outperforms several ML methods, including XGBoost, Logistic Regression, Naïve Bayes, and KNN, as well as deep learning techniques employing holdout or 10-fold cross-validation. The model achieved an accuracy of 88.12% and 79.38% for both datasets, respectively, while the highest accuracy for the other compared methods, with 10-fold cross-validation, was 86.57% and 73.90% for the two datasets, respectively.

2.2.2. Deep Convolutional Neural Network (DCNN)

A study by [14] showed how deep learning can help with telecom data challenges. The experiment utilized a dataset of call details records (CDR) from a telecommunications company, capturing customer activities in calls, SMS events, and recharge events over a specific period of two months. Customer status in this dataset was distributed into five categories: idle (initial state), active, suspend (active without outgoing calls), disable (do not receive or make outgoing calls), and pool. In the preprocessing stage, the authors transformed the five categories into two, categorizing customers with idle, active, or suspended status as non-churners and those with disabled and pool status as churners.

Three models were constructed in this study, including CONV-D1, CONV-SVM, and CONV-RF. CONV-D1 consists of two sequential convolutional layers, followed by a RELU activation function layer. The latter two models employed supervised machine learning on features obtained from the deep learning model (ConvNets). In other words, the authors trained binary classifiers based on CNN features, including the number of epochs, learning rate, momentum, weight decay, and mini-batch size. The deep convolutional neural network (DCNN) was then compared with traditional ML methods, specifically support vector machines, random forests, and gradient boosting classifiers (GBC). There was no technique used for handling imbalanced data.

The result indicates that the different architecture of DCNN was more effective than traditional ML algorithms. The highest F1 score was achieved by CONV-RF at 91%, followed by CONV-SNM and CONV-D1 at 88% and 86%, respectively. In contrast, SVM, RF, and GBC achieved F1 scores of 82%, 78%, and 83%, respectively.

2.2.3. Boosting Internet Card Churn Prediction(Boosting ICCP)

The ICCP model put forth by [15] deals with the classification of customer churn in the Internet card industry. The authors constructed a boosting model consisting of a feature extraction component and a learning architecture.

In the feature extraction phase, new features were derived from existing ones, for example, extracting the number of abnormal days from the daily networking duration. Principle Component Analysis (PCA) was then employed to select the most dominant features.

Subsequently, the identified features were fed into an embedding layer, generating low-dimensional space vectors. These concatenated vectors were introduced into a transformer layer for normalization. Then, they integrated into a lightweight multiple-layer perceptron (lightweight MLP) layer to capture correlation information. The result information was then fed into the MLP classifier component for churn prediction. It's noteworthy that no technique was applied for handling imbalanced data.

The utilized data consists of two types of user records: 4 million internet card (IC) users and 22 million traditional card (TC) users. However, the primary focus of the study was on IC users. The main attributes for this dataset included call activities, internet activity demographics,

such as age and gender, and temporal networking behaviors.

2.3 Ensemble approach

The process of combining predictions from multiple models is known as ensemble learning. This technique is widely applied across different fields. Ensemble techniques employ different models as base learners, each trained independently on the dataset. The final prediction is achieved by combining the prediction of the base learners using specific rules, as illustrated in Figure 2.

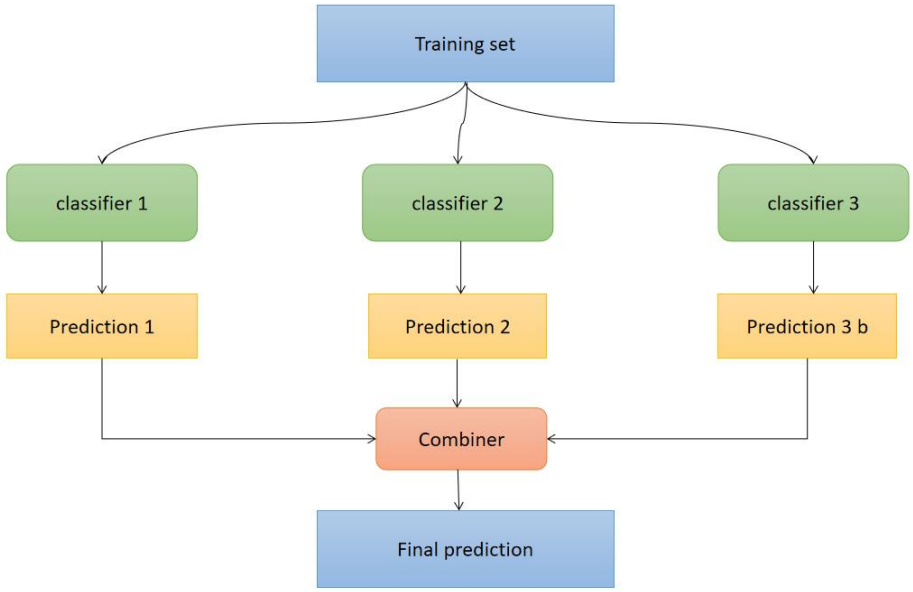


Figure 2: A type of Ensemble technique

2.3.1. Hybrid ensemble learning approach.

One of the popular ensemble methods is bagging, as applied by [16], who aimed to enhance it by combining it with another ensemble or neural network technique. They applied three ensemble techniques to outperform any base classifier.

Boosted Bagging (BoBag): In this technique, a Robust Boost and Decision Tree were utilized to combine the ideas of boosting and bagging. During the bagging phase, multiple subsets of the training data were generated using bootstrap sampling. Then, Robust Boost was applied to each subset independently, creating a set of diverse weak learners. The final prediction was produced by majority voting for all boosted models.

Bagged Bagging (BaBag): This technique combined bagging with itself, meaning using bagging for each bagged sample, where the combination of decision trees was applied for the two phases of bagging. Additionally, majority voting produced the final prediction.

Bagging of neural networks with learning based on a genetic algorithm (BNNGA): In this technique, the combination of three strong algorithms—the neural Neural Network, Bagging, and Genetic algorithm—was utilized to construct a hybrid algorithm.

The models were examined in three datasets: the UCI ML repository dataset with 5000number of samples, Cell2Cell involving 3333 observations, and IBM including 7043 records.

They found that their proposed hybrid approach performs better than simple bagging and boosting algorithms and outperforms the base classifiers and other machine learning algorithms, such as Robust Boost, AdaBoostM1, Logit Boost, Naive Bayes, Decision tree, KNN, SVM, and NN.

Among the three proposed techniques, bagged bagging performs the best in terms of accuracy at 97%, followed by BoBag, then BNNGA at 92.6% and 77.5%, respectively. In contrast, Robust Boost, AdaBoostM1, LogitBoost, Naive Bayes, Decision Tree, KNN, SVM, and NN achieved accuracies of 91%, 88.9%, 87.9%, 91%, 79%, 54%, and 91%, respectively.

2.3.2. XGboost

Boosting is an ensemble method that employs a sequential process to train base learners. It involves multiple weak-base learners, each correcting the errors of the previous model. In a study by [17], XGboost showed a high score when employed on the SeriaTel dataset, outperforming the Gradient Boosted Machine (GBM) Tree, Random Forest, and Decision Tree. The primary focus of this study was to generate a new type of feature from existing features in the dataset.

The SeriaTel dataset is a private dataset consisting of customer data, network and call logs related to internet, calls, SMS transactions, mobile information, tower locations, and problems. The new type of feature introduced is termed “Social Network Analysis (SNA) features," summarizing the relationship between each pair of customers. A graph was utilized with nodes and edges to connect customers based on call details and record data for the last four months. The extracted SNA features included 12 metrics, such as the number of friends connecting with the customer, the number of friends the customer is connecting with, the maximum cosine similarity with SyriaTel customers, and how much the customer friends know each other. These SNA features were combined with real features and used in the four models to predict customer churn.

The study conducted three experiments with their model using different data balancing techniques, including without balancing, with oversampling, and with undersampling. The best prediction performance was achieved by XGBoost without balancing, achieving an AUC of 93.3%, followed by GBM at 90.89%. Random Forest and Decision Tree achieved 78.74% and 72.2%, respectively.

2.3.3. Stacking model with soft voting

[18] proposed the model as an ensemble-learning technique that makes use of soft voting and stacking models. The stack was built in two levels, starting with Xgboost for level 1 and the Logistic Regression, Decision Tree, and Naïve Bayes machine-learning algorithms for level two. They utilized the Cell2Cell dataset to examine their model. To enhance system performance, they derive new features using equidistant grouping of customer behavior features. This feature construction approach helped discover latent information from implicit features and improved accuracy compared to the original dataset. The stacked model also contributed to enhancing the performance of base algorithms and achieved 98.09%.

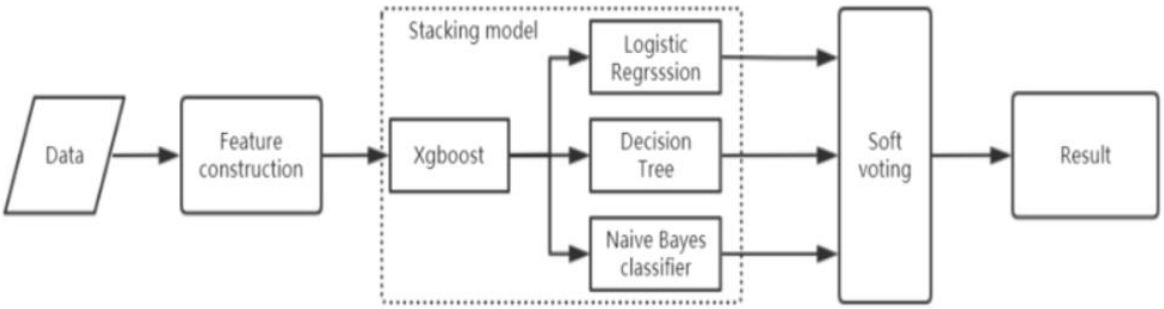


Figure 3: Stacking model with soft voting

2.3.4. Bagging-based selective ensemble model (BASE)

A novel bagging-based selected ensemble model for customer churn prediction was suggested by [19] in order to improve performance on imbalanced data. The bagging aggregation approach and base classifier training were altered by the authors. The authors used a cost-weighted negative binomial distribution during the training phase to produce random class ratios in the subsets and guarantee variety among base classifiers in order to handle the problem of unbalanced data. They initially divided the full dataset into positive and negative sets, further partitioning each set into subsets. The size of the positive subsets was the same, and the size of the negative subsets was found using a negative binomial distribution. To modify the training subsets, the cost of misclassification was multiplied by the size of each subset.

The authors used cost-sensitive logistic regression with a lasso penalty for classier aggregation, combining base classifiers to produce the final results.

The method's effectiveness was assessed through extensive experiments on ten real-world telecommunication data sets, including Chile (5300 records, 41 attributes), Duke (100462 instances, 173 attributes), KDDcup (50000 samples, 231 features), UCI ML repository dataset (3333 records, 19 attributes), Tele dataset (4350 instances, 87 attributes), and five Korean datasets with records between 2000 and 26224 and attributes between 9 and 36. Most of these data sets include social-demographic, behavioral, and payment attributes.

The Fisher score technique was applied for feature selection, ranking the features based on their ability to differentiate classes. It is calculated as the ratio of between-class and within-class variance.

The BASE model demonstrated superior performance with a 97.3% AUC, outperforming various bagging and boosting techniques such as SMOTEBagging (95.8%), UnderBagging (97.1%), Balanced Random Forest (96.8%), and SMOTEBoost (96.4%).

2.3.5. SVMs Stacking

A study conducted by [20] built several stacking methods based on support vector machine algorithms. The authors implemented five experiments to combine SVM with various techniques, including neural networks, three types of decision trees (C5, CHAID, and C&R), and KNN.

The result showed that the SVM+CHAID stack outperformed other types of stacks, achieving an accuracy of 85.8%, while SVM+C5, SVM+C&RTree, SVM+Neural NET, and SVM + K-NN achieved 0.854%, 0.849%, 0.828%, and 0.812%, respectively.

Despite the absence of feature selection methods or techniques for handling imbalanced data, the study analyzed the decision tree structure to identify the most important features causing churn based on the IBM dataset. The findings revealed that contract type, internet service, and the duration of customer engagement significantly affect customer churn.

2.3.6. SBLSTM-RNN Model

A study conducted by [21] developed a stacked bidirectional long-short-term memory (SBLSTM) and recurrent neural network (RNN) model for customer churn prediction using the arithmetic optimization algorithm (AOA).

Initially, the AOA was employed in the preprocessing phase to select the most suitable features for churn prediction. AOA is a new metaheuristic model that relies on the basic arithmetic operators (division, multiplication, addition, and subtraction). It starts by generating a random set of candidate solutions (population). Each object in the search space is characterized by its density, volume, and acceleration, which are updated until the steady state is reached. For each iteration, several equations and normalization were applied to choose the best solution.

Subsequently, the SBLSTM and RNN models were applied after hyperparameter tuning to predict churners, achieving an accuracy of 97.89% without addressing the imbalanced data. The model was evaluated on two health insurance datasets. The first dataset pertains to vehicle insurance, encompassing 42,213 records and 12 attributes related to vehicle information and customer demographic data. The second dataset concerns another insurance dataset with 45,211 records and 16 anonymized features (feature_0 to feature_15).

2.3.7. Nested Ensemble

In a study conducted by [22], stacking was nested within bagging and boosting to create a new hybrid ensemble technique (Bagged-Stacked, Boosted-Stacked) for customer churn prediction. Instead of a single base learner, a stacked learner was utilized at level 0, followed by a meta learner at level 1.

1) Boosted-Stacked: In this model, a combination of different models, namely Naive Bayes, Decision Tree, and KNN, was initialized as 3-stacked learners. Then, multiple iterations were performed, as in normal boosting, to create a number of weak models. Their output was then combined to form a Boosted-Stacked model at level 0. These are then combined with the decision tree meta-leaner at level 1 to produce a final prediction with an accuracy of 97.27% in the UCI dataset and 86.33% in the SATO dataset.

2) Bagging-Stacked: the same 3 stacked classifiers with base learners DT, KNN, and ANN at level 0 and DT at level 1 were utilized. However, in the Bagging-Stacked model, eight bootstrapped samples were generated from the original training dataset to be trained using the 3-stacked learner on each bootstrap sample. Then, the outcomes were fed to the level-1 learner to produce predictions for each sample set. These predictions were combined using majority voting to produce a final prediction with an accuracy of 98.4% in the UCI dataset and 90.3% in the SATO dataset.

The datasets utilized in this study, without addressing imbalanced data, were the UCI dataset (5000 samples, 20 attributes) and the Balanced Private SATO dataset from SouthAsia Wireless Telcom Company (2000 samples, 13 attributes). Most of the features in the UCI dataset were related to call details records of total calls during the day, evening, or night. The SATO

dataset contains revenue information, customer's account details, and customer usage details, in addition to call details records. For feature selection, a greedy feature elimination method was employed to select a subset of features based on the accuracy increase or decrease.

2.3.8 Enhanced Ensemble Stacking Model

The model proposed by [23] incorporated ensemble stacking along with uplifting-base strategies. It includes Random Forest, Decision Tree, Naïve Bayes, and Gradient Boosted Trees to provide first-level predictions, which were then processed into second-level predictions using a heuristic-based combiner to provide the final predictions.

The data utilized in this study was the UCI churn dataset, which included 3333 samples and 20 attributes. To address the issue of imbalanced data, the authors selected independent algorithms to predict churners and independent algorithms to predict non-churners. The authors identified algorithms with a true positive rate higher than a particular true threshold as the best algorithms for churners. On the other hand, the authors identified algorithms with a true negative rate greater than a particular false threshold as the best algorithms for non-churners. Both types of algorithms were then applied to test data to generate true data and false data, which were intersected to identify common data. The best prediction algorithms based on true positives were then applied to the common data to generate the final prediction.

Uplifting-base strategies are techniques that aim to improve certain outcomes. In the context of telecom churn prediction, uplifting strategies involve identifying customers who are likely to churn and implementing strategies to retain them. Customer uplifting was performed for the proposed model on the final prediction, suggesting taking actions to positively influence or retain customers. The model achieved an accuracy of 94.3%.

2.3.9. Optimized Weighted Ensemble

In a study by [24], a series of weights was applied to each base learner based on the importance of the respective base learner to optimize an ensemble learning model. The ensemble model consisted of k-nearest neighbors, CatBoost, and random forest algorithms.

The dataset utilized in this research was the Cell2Cell telecom company dataset, which included 71,047 instances and 58 attributes categorized into personal data, customer care service data, credit scores, usage patterns, and value-adding services. No specific feature selection technique was employed. However, to address imbalanced data, the SMOTE oversampling technique was utilized.

Powell's optimization approach is used to establish the ideal weights for the base learners based on their prediction performance. Weaker basis learners received smaller weights, whereas stronger base learners received bigger weights. Subsequently, soft voting was used to combine the predictions of the base learners. The model achieved an accuracy of 84.11%.

2.4. Using Sentiment as a Feature for Churn Prediction

In this subsection, we will present the studies that showed a relationship between customer churn and their sentiment, in addition to using the sentiment as a feature for churn prediction.

2.4.1. Correlation between sentiment and churn in Indian telecom companies

In a study by [9], tweets for five Indian companies were analyzed over a six-month period. After extracting tweets using Tweepy, a Python library for Twitter mining, the authors employed preprocessing and text cleaning. Polarity, ranging from -2 (highly negative) to 2 (highly positive), was calculated and assigned for each tweet using TextBlob and a Naive Bayes classifier based on a user-defined dictionary.

For each company, the total number of tweets for each polarity was calculated and multiplied with the polarity value to find the sentiment score for each company per month. Subsequently, the growth rate of sentiment was calculated for each month, with August as a base month. Finally, validation using IBM SPSS was implemented to examine the Pearson correlation between the growth rate of sentiment (%) and the subscriber's addition growth rate (%) for each company, as identified from the real data.

The correlation values for months from September to January indicated a strong relationship between sentiment and churn prediction, achieving 0.879, 0.884, 0.925, 0.906, and 0.911, respectively.

2.4.2. Investigate factors affect customer churn in bank sector

In a study conducted by [25], the authors investigated tweets about five banks to identify the most significant factors affecting individuals' decisions to switch to another bank. They began by identifying posts that mentioned one of the five banks using the API. Subsequently, they tested the polarity of each tweet using identified positive and negative terms, such as 'hate', 'frustrated' for negative sentiments, and 'great' for positive sentiments, while neutral posts were disregarded.

Additionally, terms related to churn within negative posts were identified, such as 'I am leaving' and 'never again'. Human raters then applied in-depth analysis to validate the results and build a codebook of terms related to churn. The study discovered a correlation between churn and negative sentiment, finding that 7.3% of the postings with negative sentiment had phrases associated with it.

Finally, aspect-level sentiment analysis was conducted based on the NLP lexicon provided by the human raters for posts containing churn terms. The goal was to extract the most significant factors affecting customer retention. Customer service, ethics or reputation, price, staff or human resources, bank facilities, banking products, and customer acquisition or retention were the definitions of these criteria.

2.4.3. Evaluate customer loyalty from sentiment.

In a study by [26], the authors gathered seven different English datasets that included user comments about mobile applications. After collecting comments, the authors gathered information such as ratings, user names of related players, dates, votes, and comments for each mobile application. The user ratings ranged from 1 to 5, and to analyze sentiment, authors categorized rates 1, 2, and 3 as negative sentiment and rates 4 and 5 as positive sentiment. The authors used the quantity of distinct comments a single user made three times or more to analyze consumer loyalty. Loyal customers were identified as those willing to provide three or four pieces of feedback on how to enhance a company's services or products.

The following stage involved the use of deep learning algorithms, word embedding strategies, and deep contextualized word representations. FastText, GloVe, and Word2Vec were used for word embedding. We used BERT, MBERT, DBERT, and RoBERTa for deep contextualized word representations. Convolutional neural networks, recurrent neural networks, and long short-term memory networks were utilized for deep learning. Word embedding approaches guarantee that phrases in a specific context are correlated and have similar syntactic and semantic properties. With a mean accuracy rating of 86.09%, the DBERT model demonstrates the highest sentiment categorization success.

To enhance classification performance, word embedding substitution (WES) and random undersampling (RUS) techniques were employed for each dataset to handle imbalanced data. In the WES approach, documents from the minority class were chosen to be replicated multiple times. When words in duplicate documents were replaced, all available synonyms were used to create new documents. Resampling the data improved classification by nearly 1%.

According to the authors, a majority of users' attitudes can be used to predict a user's loyalty in the early stages, particularly if the program has a negative aspect. The authors concluded the connection between the sentiment analyzed from customer feedback and their loyalty by examining the results of positive comments. They showed that users who commented on the mobile application three and four times about requests to improve the application were the same users who reported their positive comments about the application. A limitation of this study was the identification of customer loyalty, as the authors did not provide evidence for identifying customers with three or four comments requesting enhancements as loyal customers. Moreover, the classification occurred only for sentiment without predicting churners.

2.4.4. Factors affect customer churn

In a study conducted by [27], the factors that affect customer churn were identified. The study was implemented on the Iranian mobile operator call-center dataset, which included 3150 customers. To identify the most influential factors that affect customer churn, a binomial logistic regression model was employed using SPSS software to investigate the direct factors. These direct factors included the number of failed calls, subscription length, customer complaints, amount of charge, length of calls, number of calls, frequency of SMS, number of distinct calls, type of service, and group age. The authors identify 'customer complaints' as a dissatisfaction factor.

Binomial logistic regression is a type of regression analysis that models the relationship between a binary dependent variable and one or more independent variables. Like other types of regression, logistic regression generates slops and a constant, which are used to calculate 'logits'. Logits represent the natural logarithm of the odds for a category. The final regression calculations provide information about the important factors.

In this study, customer churn was used as the dependent variable, while other factors were used as independent variables. The general coefficient examined the effectiveness of the independent variables on the dependent variable. The result indicated that the most significant factors affecting customer churn were customer complaints, which represent a categorical value of 0 or 1 and indicate customer dissatisfaction; the number of services used by customers, such as calls and short messages; demographic information, such as age and gender; and the number of service failures.

2.4.5 SentiChurn model

The utilization of sentiment as a variable in churn prediction was addressed by [5]. The research employed Twitter mining to predict customer loss, focusing on Arabic tweets from three telecom companies in Saudi Arabia. The proposed model has been constructed based on two models: AraBERT and Bidirectional Gated Recurrent Units (Bi-GRU). The study consisted of three main stages: data collection and variable definition, date preparation and satisfaction rate computation, and churn prediction with evaluation.

In the first step, the data consisted of historical data provided by the companies and customer satisfaction rates measured from Twitter mining. The variables that were to be utilized for the prediction model were identified from three resources: a literature review, surveys, and interviews with experts from each company.

In the second stage, the historical data was preprocessed. Additionally, the authors collected tweets related to the three companies by monitoring the top hash tags referring to them from January 2017 until June 2017, resulting in 795,500 Saudi tweets. The authors classified each tweet into positive and negative, calculated the total unique tweets for each company, and finally computed the percentage of satisfaction sentiment for each company based on the total positive tweets. They developed a model to predict customer satisfaction on the AraCust corpus based on the predefined companies STC, Mobily, and Zain and achieved 31.06%, 34.25%, and 32.06%, respectively. These percentages were then evaluated by comparing them with the actual sentiment percentage calculated by surveys. The satisfaction percentage was then combined with variables from historical data to produce the final dataset. These additional variables included age, gender, family member, overdue bill, long period, new customer, inactivity, low data, low talk, no internet or talk and SMS, no value-added service, and churn status.

The final step was predicting customer churn using this final dataset. To evaluate the churn prediction, the authors calculate the churn rate by dividing the total number of churners by the number of customers, multiplying by 100, and comparing it with the actual churn rate provided by the companies. The proposed SentiChurn model proved its efficiency by achieving an accuracy of 95.8%. A limitation of this research is that sentiment is used only as a continuous variable. Furthermore, the impact of adding sentiment to the churn predictive model is not clear, as the study did not demonstrate how the model performs without incorporating sentiment.

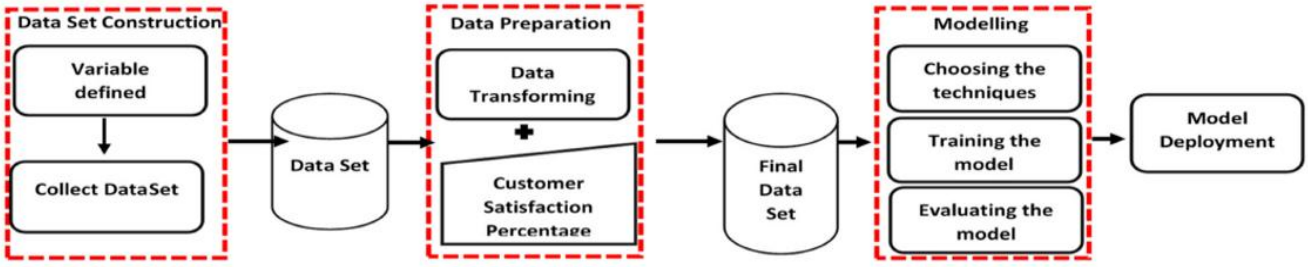


Figure 4: SentiChurn model architecture

2.5 Concluding Remarks

Derived from our extensive literature review, the standard variables employed for customer churn prediction in the telecommunications industry encompass demographic information (such as age and gender), call details, SMS events, recharge and billing information, contract length, internet services, technical support, and social network metrics (including the number of friends or family members). However, the utilization of sentiment as a predictor of customer churn remains limited.

Numerous research endeavors have been initiated to elucidate the correlation between sentiment analysis and customer churn. A noteworthy illustration of the integration of sentiment as a feature in the churn prediction model is evident in the findings of a specific study [5], which reported a remarkable accuracy of 95.8%. It is crucial to note, however, that this particular study incorporated sentiment as a general measure representing the overall percentage of customer satisfaction with the company. The exploration of treating sentiment as a distinct and individualized score for each customer within the churn prediction model is a facet yet to undergo scrutiny. Nevertheless, it is imperative to recognize the necessity of conducting a meticulous search for datasets that encompass sentiment as one of their features. The IBM dataset [28] emerges as an exemplary candidate meeting these criteria, encompassing a sentiment score for each customer alongside other pertinent customer-related data. The best result for predicting customer churn using the IBM dataset and without sentiment was 97% accuracy.

Addressing the issue of imbalanced data requires careful consideration, as emphasized in prior studies. The data pertaining to customer churn is predominantly imbalanced, with churners representing the minority class. Previous research endeavors have commonly employed oversampling and undersampling techniques to mitigate this imbalance. However, the investigation conducted by [19] introduced a novel methodology, utilizing a cost-weighted negative binomial distribution technique. This approach maintains the same size for positive samples while dynamically resizing the negative sample through a negative binomial distribution. The adjustment of negative samples is governed by the cost of misclassification. Significantly, this innovative approach yielded better results in comparison to traditional oversampling and undersampling methods, achieving accuracies of 97.3%, 96.4%, and 97.1%, respectively.

Our literature review also highlights the prevalent utilization of ensemble models within the domain of customer churn. These models have demonstrated significant success, with the majority of prior studies employing stacking, bagging, and boosting ensemble techniques. Notably, stacking and bagging emerged as the most frequently employed methods. These two ensemble techniques exhibited notable efficacy, achieving accuracy rates of approximately 97% and 98%. Moreover, our investigation reveals a gap in the literature concerning the examination of the super learner ensemble method within the context of customer churn, particularly with respect to the IBM dataset. Despite being recognized as a potent ensemble method [29], the super learner ensemble has not yet been explored in this specific domain. Addressing this gap may contribute valuable insights to the existing body of knowledge on customer churn prediction.

Chapter 3

Research Methodology

To achieve our objective, we started by searching for a suitable dataset to construct our experiments, landing on the IBM Telco Customer Churn dataset [28]. This open-source database contains 7043 observations with 33 attributes. The selection of this dataset was based on its alignment with our research objective, which focuses on predicting customer churn based on sentiment. Alongside sentiment and churn status, the dataset includes information on demographics, location, population, and services.

Our approach involves examining the impact of incorporating sentiment at different levels as a feature to predict customer churn. Unlike previous studies where sentiment was used as a customer satisfaction percentage for a company at a specific time, we aim to utilize the sentiment score for each customer individually to predict their churn status. Three levels of sentiment will be examined to determine which level is more beneficial for enhancing churn prediction. Predictions will be made based on 5-score sentiment levels, 3-score sentiment levels, and 2-score sentiment levels.

To address the issue of imbalanced data, we employed the class-weighted technique. This method assigns specific weights to each class based on their importance during the training phase. We opted for this approach to give more significance to the unsatisfied customer class (the negative class), as they are more likely to churn. By using class weighting, we aim to reduce the misclassification of unsatisfied customers, thereby improving the accuracy of churn prediction.

The results of the sentiment model predictions were integrated with other features selected by the mutual information technique to predict customer churn. For churn prediction, we intend to apply the ensemble method. This approach was chosen due to its promising results. We specifically focus on the variation when selecting candidate algorithms for the ensemble model. We will apply the stack ensemble approach, where the KNN, SVM, and RF will be utilized as base learners and the MLP will be utilized as a meta learner model to produce the final customer churn prediction.

Figure 5 visually illustrates the overarching architecture of the proposed churn prediction model, providing a graphical representation of the model's structure and workflow.

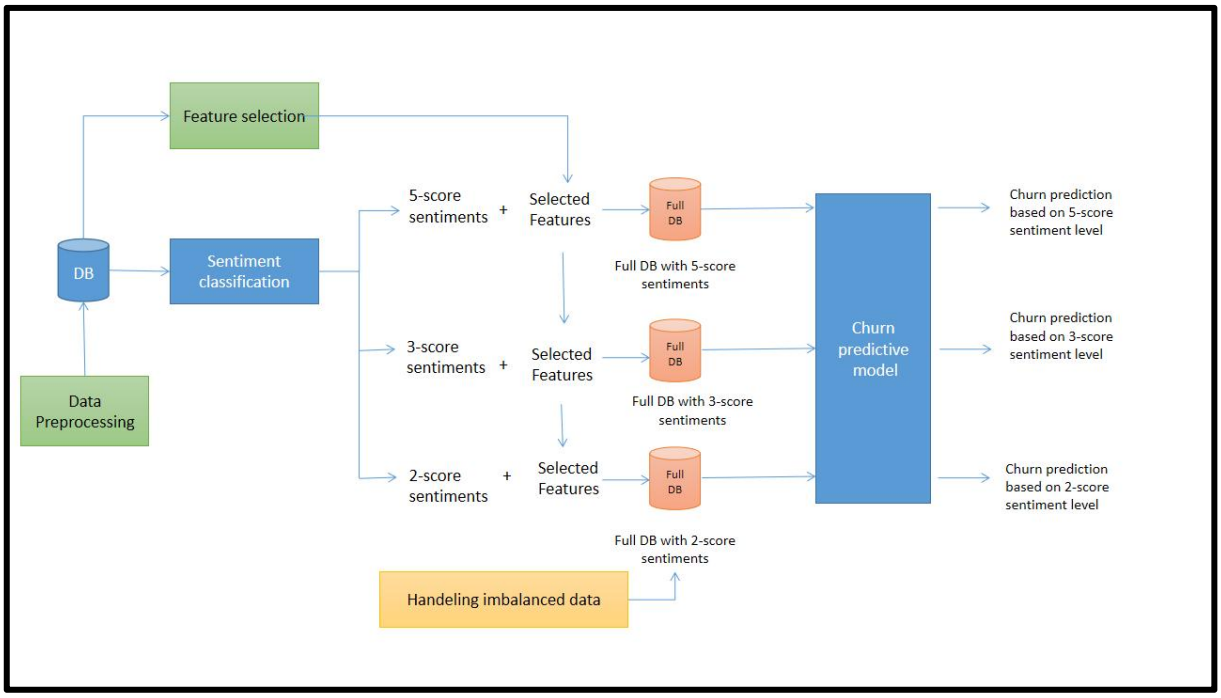


Figure 5: Proposed Model Architecture

As depicted in Figure 5, our methodology will begin with the preparation of the dataset. Multiple techniques for data preprocessing will be applied to facilitate its utilization in the sentiment classification model and churn prediction. The sentiment classification process will produce three distinct types of sentiment categorization: 5-score sentiment classification, 3-score sentiment classification, and 2-score sentiment classification. Following that, measures will be implemented to address imbalanced data in the 2-score sentiment classification. This will allow for more accurate predictions to be used in the churn prediction. The three types of sentiment classification that the sentiment classifier creates will be combined with the preprocessed data after the best features for predicting churn have been selected. The three types of datasets created will be subsequently introduced into the churn prediction model to predict customers who are likely to churn. Ultimately, the comparison will be carried out to choose the best sentiment level for churn prediction.

In the following sections, we will provide detailed explanations for each step of our proposed methodology. We will start by discussing how we prepare the data and the techniques utilized for data preprocessing and feature selection. Subsequently, we will explain how we build a baseline system and incorporate sentiment as a feature. We will also address the strategy that will be used to mitigate the issue of imbalanced data. Furthermore, we will describe the implementation of an ensemble method for churn prediction. Finally, we will outline the evaluation metrics used throughout the research.

3.1 Data Preparation

The IBM dataset that will be utilized in this research encompasses numerous features illustrated in Table 1, with the aim of predicting customer churn. However, for the

specific task of churn prediction, data preprocessing will be conducted, wherein features most pertinent to customer churn will be systematically selected.

Table 1: Features of IBM dataset

Category	Features
Demographics	Customer ID - Count - Age - Senior Citizen - Married - Dependents- Number of Dependent
Location	State - City - Zip Code-Lat Long - Latitude- Country -
Population	ID - Population
Services	Quarter - Referred a Friend - Number of Referrals - Tenure in Months - Offer - Phone Services- Avg Monthly Long Distance Charges- Multiple Lines - Internet Service - Avg Monthly GB Download - Online Security - Online Backup - Device Protection Plan - Premium Tech Support - Streaming TV
Status	Satisfaction Score - Customer Status - Churn Label - Churn Value - Churn Score - Churn Category - CLTV - Churn Reason

The process of data preparation plays a crucial role in enhancing the performance of machine learning models. In this research, a series of preprocessing steps were employed to ensure the quality and suitability of the data for model training. This process transforms the data into a format that is suitable for training machine learning models, using the following steps:

3.1.1 Data Preprocessing

- **Remove unnecessary columns**: Redundant variables and unique identifiers ('customer ID','ID') will be removed. Features with constant values ('Count', 'Country', 'State', 'Quarter') will be ignored, as they do not contribute to the prediction. Columns such as 'Zip Code', 'Lat Long', 'Latitude', 'Longitude', 'City', 'population', and variables highly related to each other will be removed. For instance, 'Age', 'Referred a Friend', and 'Dependents' will be eliminated, retaining their corresponding features: 'Under 30', 'Number of Referrals', 'Number of Dependents'. The variable 'Married' will be removed due to its compatibility with the variable 'Partner'. Lastly, columns highly related to 'Churn Value', such as 'Churn Reason', 'Churn Category', 'Churn Label', and 'Churn Score', will be removed.

- **Handle null values**: Null values for 'Offer' and 'Internet Type' will be filled with 'Unknown' to prevent data loss, given that the null values for these two variables are

3896 and 1545, respectively. After filling in the null values for 'Offer' and 'Internet Type', the remaining null values in other features (19 null values) will be removed.

- **Value replacement**: Values of 'No phone service' and 'No internet service' for both 'Multiple Lines' and 'Online Security' will be replaced with 'No' to deal with it easily during the classification.

- **Discretization**: It is applied to convert continuous and numerical data into discrete categories or group similar data points into discrete categories [30]. This technique simplified the data, making it more manageable. Thresholds of 5 and 10 were identified for features such as 'Number of Referrals' and 'Number of Dependents' respectively. The values for these features will be categorized as '>5' and '<=5' for 'Number of Referrals', and '>=10' and '<10' for 'Number of Dependents.

This technique was also used to prepare the sentiment levels examined in our research. The data includes a 5-score sentiment level that is categorized from 1 to 5, corresponding to strong negative, negative, neutral, positive, strong positive, respectively. In data preparation, we will extract the two other levels of sentiment: a 3-score sentiment level and a 2-score (binary) sentiment level. To categorize sentiment into three levels, we will specify three intervals for sentiment score ('>3 score', '3 score', and '<3 score'), later called positive, neutral, and negative classes, respectively. Then, to categorize sentiment into two categories, we will specify two intervals for sentiment score ('>=3 score' and '<3 score'), later called positive and negative classes, respectively.

- **Data transformation:** Data transformation strategies can considerably improve model performance. This research will apply two data transformation methods: One-Hot Encoder and Normalization.

  - **One-Hot Encoder (OHE) (dummy)**: This popular method, based on binary encoding [31], represents the existence of a variable with 1 and its absence with 0. Each attribute with N categories is transformed into an N-dimensional vector with 0s and 1s. In this research, categorical data that will be transformed into a one-hot encoder includes: 'Gender', 'Senior Citizen', 'Partner', 'Phone Service', 'Multiple Lines', 'Internet Service', 'Online Security', 'Online Backup', 'Device Protection', 'Tech Support', 'Streaming TV', 'Streaming Movies', 'Contract', 'Paperless Billing', 'Payment Method', 'Under 30', 'Number of Dependents', 'Number of Referrals', 'Offer','Internet Type', 'Streaming Music', 'Unlimited Data'.

- **Normalization**: The normalization method improves the performance of the model by reducing the effects of outliers [31]. Thus, z-score [32] will be implemented in this study to standardize the values of numerical data. The standardized z-score is computed via the following Equation 1:

$$Y_{scaled} = \frac{Y - Mean}{Stan^{dev}} \quad (1)$$

Where:

Y : Feature set

Stand dev: Standard deviation.

According to this study, the following features will be normalized: 'Tenure Months', 'Monthly Charges', 'Total Charges', 'Avg Monthly Long Distance Charges', 'Avg Monthly GB Download', 'Total Refunds', 'Total Extra Data Charges', 'Total Long Distance Charges', 'Total Revenue', 'CLTV'.

3.1.2 Feature Selection

Following data preprocessing, the identification of the optimal subset of features for churn prediction will be undertaken. In this research, the mutual information technique is employed to select the most relevant features related to customer churn.

Mutual Information (MI) serves as a feature selection technique, quantifying the mutual information between two variables and assessing the extent to which a variable can extract information from another, thereby mitigating uncertainty. The MI method has demonstrated efficacy in dimensionality reduction and performance enhancement in classification tasks [33]. To elucidate the relationship between variables in our dataset and the 'Churn Value,' we applied Equation 2 to calculate the Mutual Information score for each variable, as follows:

$$MI(X,Y) = \sum_{x\in X}\sum_{y\in Y} P(X,Y) \log\left(\frac{P(x,y)}{P(x)\,p(y)}\right) \quad (2)$$

Where:

MI(X;Y) is the mutual information between random variables X and Y.

P(x,y) is the joint probability mass function of X and Y.

P(x) and P(y) are the marginal probability mass functions of X and Y, respectively.

This satisfied the properties:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \quad (3)$$

Where:

H(X) and H(Y) are the entropy of X and Y respectively.

H(X, Y) is the joint entropy between X and Y

After computing the mutual information for all variables, the threshold value for feature selection was established by calculating the average MI score of the features. Features with values greater than the calculated average were selected for inclusion.

3.2 Experiment-1 Building the Baseline System

In this experiment, we will establish a baseline system as a test for our experiment. We will start by constructing four classifiers, namely K-Nearest Neighbor (KNN) [34], Support Vector Machine (SVM) [35], Random Forest (RF) [36], and a Multiple Layer Perceptron (MLP) classifier [37]. These classifiers will be trained on the dataset after data preprocessing and feature selection. Then, we will build an ensemble classifier consisting of KNN, SVM, and RF as base learners and the MLP as a meta learner. The data will be retrained using this ensemble classifier. The results of the ensemble will be compared against the individual performances of the four classifiers. The classifier yielding the highest F1-score will be selected as the baseline system for further evaluation.

3.3 Experiment-2 Building a churn prediction system using sentiment as a feature

In this experiment, we aim to enhance the baseline system by incorporating sentiment analysis into the data utilized for churn prediction, thereby addressing the primary research question. Initially, the integration of sentiment at various levels into the database will be executed. This involves introducing sentiment data into the system and constructing a sentiment classification model, facilitating three distinct classifications. Subsequently, the churn prediction model will be developed based on these three tiers of sentiment levels.

To incorporate sentiment information into the database, three distinct sentiment levels will be introduced. The first is the 5-score sentiment level, categorized from 1 to 5, corresponding to strong negative, negative, neutral, positive, and strong positive sentiments. The second is the 3-score sentiment level, consisting of positive, neutral, and negative classes. The third is the 2-score sentiment level, representing positive and negative classes. Subsequently, the KNN, SVM, RF, and MLP will classify sentiment into five categories. The

classifier with an F1-score will be chosen as our sentiment classifier, which will then be applied to classify customers based on the three sentiment levels.

Following the implementation of sentiment classification, we will integrate the predictions from each sentiment classification with other features selected from the IBM dataset using the mutual information technique. This process will yield three distinct datasets: one with a 5-score sentiment, another with a 3-score sentiment, and finally, a dataset with a 2-score sentiment. After creating the dataset, the baseline system will be trained again using these three separate datasets to predict customer churn.
Ultimately, a comparative analysis of the outcomes will be conducted to address the secondary research question, aiming to ascertain the most effective sentiment level for predicting churners.

3.4 Experiment -3 Handling imbalanced data

The imbalance within the dataset holds the potential to introduce bias into the binary sentiment classification model, as it tends to prioritize the majority class, which is representative of satisfied customers. According to our methodology, it is essential to emphasize that sentiment predictions serve as inputs for churn prediction, as depicted in Figure 6. Consequently, we aim to enhance the sensitivity of the sentiment classification model towards the class of dissatisfied customers, given their higher likelihood of churning. This can be achieved using the class-weighted technique.

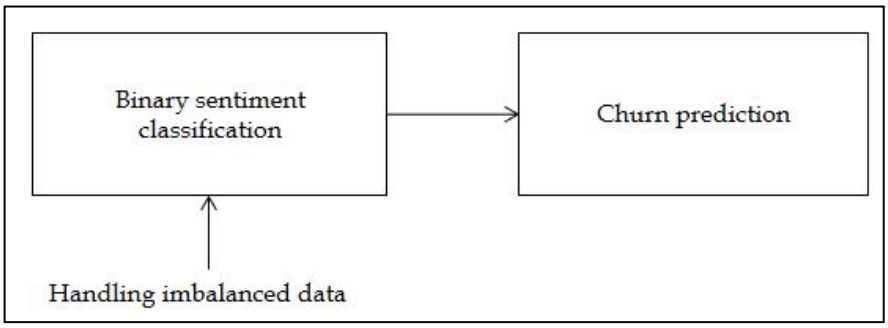


Figure 6: handle imbalanced data

Resampling techniques, such as oversampling and undersampling, are deemed unsuitable for our research objectives. This is due to their impact on altering the sample size, which needs to remain consistent for subsequent utilization in churn prediction. Furthermore, oversampling may induce overfitting, especially in cases where the generated examples lack diversity, thereby extending the training time[38]. Conversely, undersampling may result in the loss of valuable information[39].

In contrast, the application of class-weighted techniques is favored, as it does not alter the sample size. Instead, it assigns greater importance to the minority class, representing dissatisfied customers. The application of the class-weighted technique involves assigning distinct levels of importance to each class during the training process. The adjustment involves altering the loss function to assign various classes various weights. The minority class is given more weight than the majority class because it is seen as more important. This makes mistakes in the minority class have a bigger effect and come with harsher punishments than mistakes in the majority class.

Following the adjustment of the loss function, we will retrain the binary sentiment classification model and compare the last result with those obtained prior to addressing imbalanced data. The primary goal of this comparison is to investigate the impact of handling imbalanced data on sentiment classification and churn prediction results.

The weights for each class will be computed through a grid search optimization algorithm. To ascertain the optimal weights for each class, the grid search selects values within a range of 0 and 1 for both w_0 and w_1, such that for each assigned value x to w_0 , (1-x) will be assigned to w_1 .

After the calculation of weights for each class, they are incorporated into the loss function, the binary cross-entropy loss function in our case. The weighted binary cross-entropy loss function is expressed as Equation 4:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [w_0 y_i \, log(\hat{y_i}) + w_1(1 - y_i) \, log\,(1 - \hat{y_i})] \quad (4)$$

Where:
N is the number of samples in the dataset.
y_i is the true label of the i-th sample, either 0 or 1
$\hat{y_i}$ is the predicted probability of the i-th sample belonging to class 1.

3.5 Evaluation metrics

In order to measure the performance of both sentiment classification and churn prediction models, we utilized state-of-the-art evaluation metrics such as accuracy, AUC, precision, recall, and f1-score. [40].

Chapter 4

Experiments and Results

In this chapter, we will discuss the experiments conducted during this research and present our findings. In our experiments, we utilized some packages from Python, such as Scikit-Learn [41], Matplotlib [42], and Seaborn [43], for data preprocessing, model construction, and visualization. We will also describe the features used in the churn prediction model and explain how the baseline system for churn prediction was built. Next, we will discuss sentiment classification, which includes how sentiment at different levels is used as a feature in the churn prediction model. Furthermore, we will address the issue of imbalanced data in binary sentiment classification and see its impact on churn prediction.

4.1 Experiment-1 Building the Baseline System

This experiment was conducted in several phases. Initially, we identified the features that will be used for churn prediction and determined the hyper-parameters that will be utilized for each classifier, along with the data format to be supplied to the classifiers. Subsequently, five experiments were conducted to build different classifiers, namely KNN, SVM, RF, MLP, and ensemble. The selection of these classifiers was motivated by similar works in churn prediction [2], [44].The last subsection compares the results of these classifiers to choose the baseline system for our research.

4.1.1 Identifying the features selected using mutual information

Following data preprocessing, feature selection was conducted utilizing mutual information (MI) to identify the most pertinent features for churn prediction [33]. The average of MI_scores for all features was computed, resulting in a value of 0.015. Subsequently, features with MI_scores greater than the computed average were selected. Table 2 illustrates the features chosen through mutual information, along with their corresponding data types and the modality of their values.

Table 2: Selected features before preprocessing

Feature Name	Type type	Modality / Min-Max
Partner	Binomial	Yes, No
Internet Service	Binomial	Yes, No
Online Security	Binomial	Yes, No
Tech Support	Binomial	Yes, No
Contract	Categorical	Month-to-Month, One Year, Two Year.
Paperless Billing	Binomial	Yes, No
Payment Method	Categorical	Bank Withdrawal, Credit Card, Mailed Check
Number of Referrals	Numeric	[0,11]
Offer	Categorical	None, Offer A, Offer B, Offer C, Offer D, and Offer E
Internet Type	Categorical	No, DSL, Fiber Optic, Cable.
Unlimited Data	Binomial	Yes, No
Tenure Months	Numeric .	[1; 72]
Monthly Charges	Numeric	[18,25; 8684,80]
Total Charges	Numeric	[18,80; 118,75]
Avg Monthly GB Download	Numeric .	[0, 85]
Total Long Distance Charges	Numeric	[0, 3564.72]
Total Revenue	Numeric	[21.36, 11979.34]

4.1.2 Determining the Hyper-parameters for Each Classifier

To choose the optimal values for the hyper-parameters of each classifier, we applied a systematic hyper-parameter tuning strategy to enhance the performance of the machine learning models. This approach involved the use of a grid search methodology along with 5-fold cross-validation, according to many systems that utilized the same approach for hyper-

parameter tuning [45], [46], and [47]. The main goal of this approach is to identify the most effective combination of hyper-parameters that maximizes the model's performance.

In the model training process, the grid search systematically generated multiple models, exploring diverse hyper-parameter combinations with the primary objective of optimizing the F1-score. The performance evaluation of each model occurred through 5-fold cross-validation, ensuring robustness and facilitating the selection of the best-performing configuration. This technique was implemented using the Scikit-Learn package in Python [41].

The ranges of hyper-parameter values explored by the grid search algorithm were determined based on similar works in churn prediction. These values, along with the optimal configurations identified through the grid search, are presented in Table 3. The subsequent subsection elucidates the implementation details of each classifier.

Table 3: Hyper-parameter setting for all classifiers [1]

Classifier	Hyperparameter	Search space	Best value for Hyper-parameter
KNN	n_ neighbors [2]	[1, …, 20] [2]	3
SVM	C[3]	[0.1, …, 50] [2]	50
RF	n_estimators[4]	[10,…,100] [2]	83
	max_depth[5]	[5,…, 50] [2]	10
	min_samples_split[6]	[2,…,11] [2]	2
	min_samples_leaf [7]	[1, …,11] [2]	1
MLP	Hidden layer_size[8]	[(10,) , (25,), (30,), (50,)] [49]	(10,)
	Solver [9]	[‘adam’, ‘Sigmoid’] [49]	‘adam’

[1] The numbers after the range of search space is for the works that utilized the same range
[2] Determines how many neighbors will be considered when making a prediction.
[3] A regularization parameter that controls the trade-off between achieving a low training error and a low testing error.
[4] The number of decision trees in the forest.
[5] Controls the maximum depth of each decision tree in the forest.
[6] Specifies the minimum number of samples required to split an internal node.
[7] Sets the minimum number of samples required to be in a leaf node.
[8] Determines the architecture of the neural network as a tuple that involves the number of neurons in the corresponding hidden layer.
[9] An optimization algorithm used that affects the speed and performance of the neural network during training.

4.1.3 Preparing data to be fed into classifiers

Various techniques for data preprocessing were applied to prepare the data for training. The primary concern of data preprocessing was the data transformation, where we handled the categorical data and applied normalization. The entire process of data preprocessing was explained in detail in subsection 3.1.1.

The standard format for data fed into machine learning classifiers comprises a two-dimensional dataframe for the feature matrix and a one-dimensional series for the target variable. Each row in the feature matrix represents a sample, and each column represents a feature. These features align with those selected by Mutual Information (MI), as depicted in Table 2. Additionally, the one-dimensional series contains labels for churn values corresponding to each sample in the feature matrix (X). Figure 7 shows a part of the data frame of the churn prediction model.

Contract_Two year	Paperless Billing_Yes	Payment Method_Electronic check	...	Internet Type_Fiber Optic	Internet Type_unKnown	Unlimited Data_Yes	Tenure Months	Monthly Charges	Total Charges	Avg Monthly GB Download	Total Long Distance Charges	Total Revenue	ChurnValue
False	True	False	...	False	False	True	2.0	53.85	108.15	21.0	20.94	129.09	1
False	False	False	...	False	True	False	1.0	18.80	18.80	0.0	43.57	62.37	0
False	True	True	...	True	False	True	3.0	80.00	241.30	22.0	57.54	298.84	0
False	True	True	...	True	False	True	59.0	94.75	5597.65	14.0	967.01	6564.66	0
False	True	False	...	True	False	True	5.0	80.10	398.55	13.0	61.75	460.30	0
False	True	True	...	True	False	True	2.0	70.70	151.65	51.0	18.24	169.89	1
False	True	True	...	True	False	True	50.0	95.70	4816.70	11.0	946.00	5762.70	1
False	True	False	...	True	False	True	1.0	69.90	69.90	28.0	4.99	74.89	1
False	False	True	...	True	False	False	53.0	90.80	4921.20	19.0	133.03	5114.23	0
True	False	False	...	False	True	False	15.0	25.20	387.90	0.0	700.20	1088.10	0

Feature matrix

Target Value

Figure 7: A part of data frame that fed to churn predictive models

It is noteworthy that after the determination of optimal hyper-parameters, all classifiers were trained on a dataset randomly partitioned into a 70% training set and a 30% testing set.

4.1.4 Experiment 1-1: K-nearest neighbor (KNN)

In this experiment, we utilized the sklearn. neighbors' package in Python to implement a KNN classifier. Each row in the predefined data format represents a point with multi-dimensional space based on the number of features. To calculate the distance between two

points, we utilized the default distance measure, Euclidean Distance, which was calculated using Equation 5:

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (5)$$

n = number of dimensions
X_i, y_i = data points

We applied the model with three numbers of neighbors, and then we called these parameters as shown in Figure m. The (p =2) parameter refers to the distance metric. = Euclidean Distance)

```
knn_clf = KNeighborsClassifier(n_neighbors=3, weights='distance', p=2)
knn_clf.fit(X_train, y_train)
knn_Y_pred = knn_clf.predict(X_test)
```

Figure 8: KNN function call

The implementation of KNN resulted in 74.07% accuracy, 49.38% precision, 43.91% recall, and a 46.48% F1-score. Figure 9 illustrates the confusion matrix for KNN.

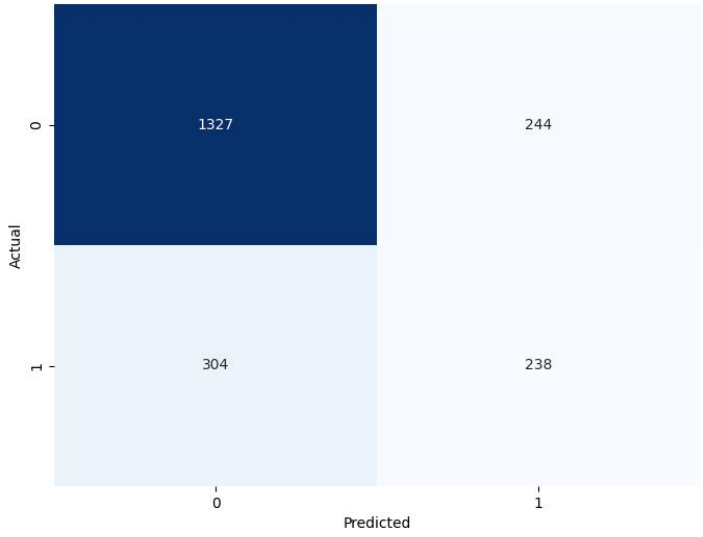


Figure 9: KNN confusion matrix

4.1.5 Experiment 1-2: Support Vector Machine (SVM)

In this experiment, we utilized the sklearn.svm package to implement an SVM model with a regularization parameter equal to 50. The data was already normalized, as explained in subsection 3.1.1. We applied the model using the function of SVC, as shown in Figure 10.

```
svm_clf = SVC(C=50)
svm_clf.fit(X_train, y_train)
svm_Y_pred = svm_clf.predict(X_test)
```

Figure 10: SVM function call

After implementing the model, it achieved 79.27% accuracy, 70.97% precision, 32.47% recall, and a 44.56% F1-score. Figure 11 presents the confusion matrix for SVM.

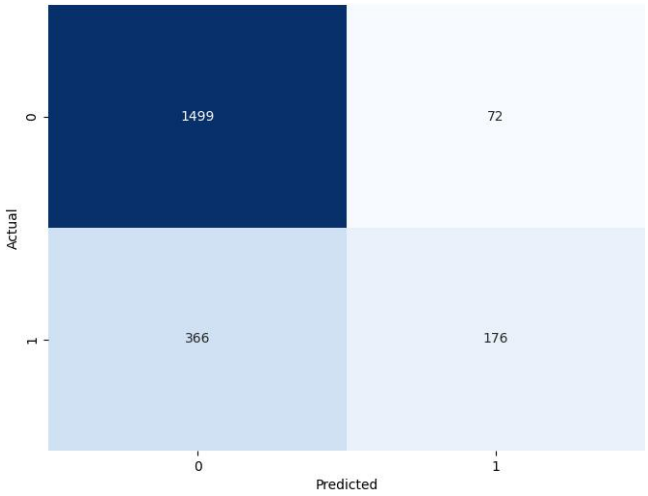


Figure 11: SVM confusion matrix.

4.1.6 Experiment 1-3: Random Forest (RF)

The experiment involved the use of an RF classifier. The RF classifier involves several decision trees, each acting as a simple estimator. Each decision is exposed to a different number of features and a different sample of the original dataset, and as such, every tree makes its own predictions. Subsequently, the predictions of all these estimators are concatenated together via majority voting to produce the final view of the problem. Figure 12 shows the general architecture of RF. The red circles indicate a hypothetical path the tree took to reach its decision.

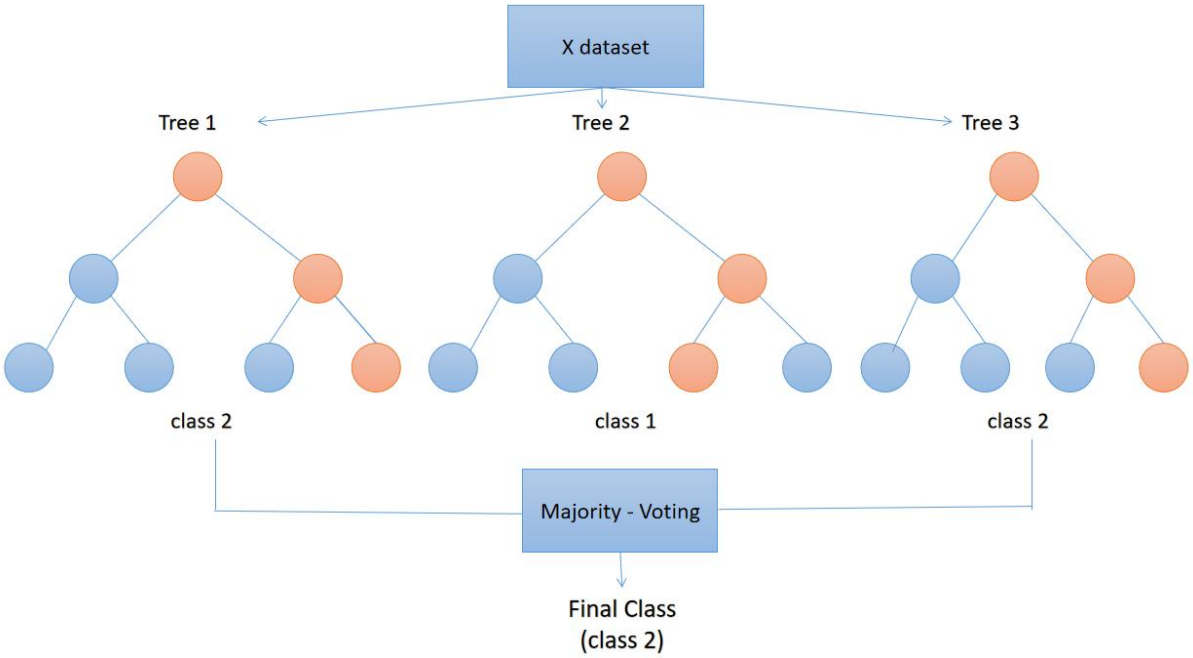


Figure 12: RF Architecture

The experiment was implemented using the sklearn.ensemble package. We call the function of the Random Forest classifier as shown in Figure 13.

```
rf_clf = RandomForestClassifier(n_estimators=83, max_depth=10, min_samples_split=2,min_samples_leaf=1)
rf_clf.fit(X_train, y_train)
rf_pred = rf_clf.predict(X_test)
```

Figure 13: RF function call

While training the model, four hyper-parameters were utilized. The first one is n_estimators, which is assigned to 83. This parameter refers to the number of trees in the forest. The second one is max_depth, which indicates the maximum depth of each tree in the forest, and it is set to 10. The third parameter, min_samples_split, refers to the minimum number of samples required to split an internal node, and it is set to 2. The last one is min_sample_leaf, which is assigned to 1 to ensure that each leaf node has at least one instance. The data was fed to the model in the same format as explained in subsection 4.1.3. Then the predict method of RandomForestClassifier directly provides the majority vote predictions based on the ensemble of decision trees it has been trained on. The function was provided with a testing set to return an array of predictions for each instance. The implementation of RF achieved an accuracy of 80.79%, precision of 65.89%, recall of 52.30%, and F1-score of 58.14%. Figure 14 displays the confusion matrix for RF.

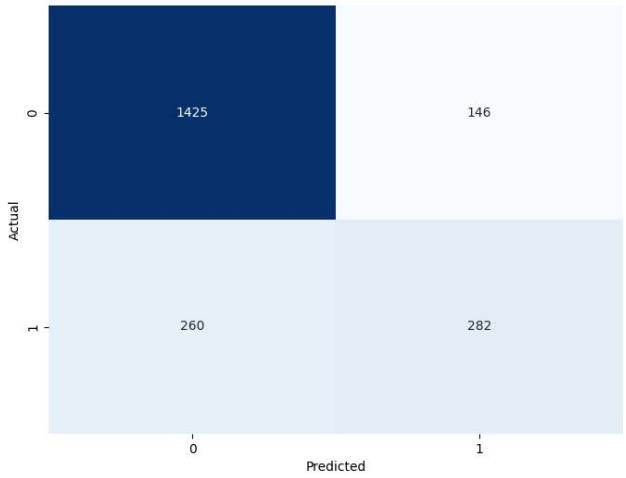


Figure 14: RF Confusion Matrix

4.1.7 Experiment 1-4: Multi-layers perceptron (MLP)

In this experiment, we implemented the MLP using the sklearn.neural_network package. The model was applied using one hidden layer that consists of 10 neurons. The solver parameter is the optimization algorithm used to update weights during training. In this classifier, the 'adam' optimization algorithm was used. Figure 15 shows the function calling for MLP.

```
mlp_clf = MLPClassifier(hidden_layer_sizes=(10,), solver = 'adam')
mlp_clf.fit(X_train, y_train)
mlp_pred = mlp_clf.predict(X_test)
```

Figure 15: MLP function call

To train the model, we fed the data with the predefined format, which resulted in 79.65% accuracy, 64.51% precision, 45.94% recall, and a 53.66% F1-score. Figure 16 shows the confusion matrix for MLP.

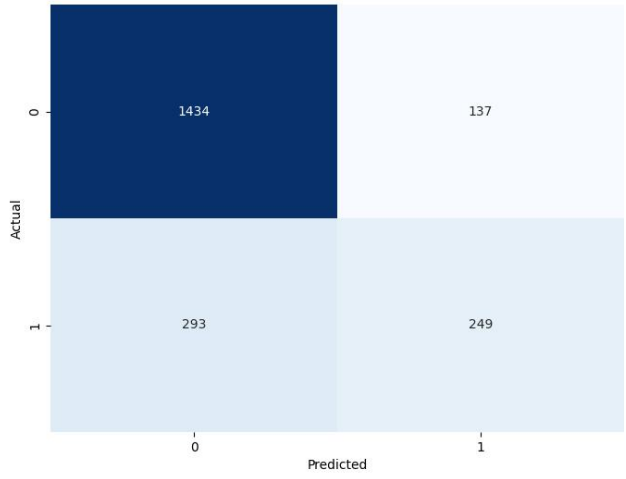


Figure 16: MLP Confusion matrix

4.1.8 Experiment 1-5: Ensemble

In this experiment, we constructed an ensemble classifier by combining KNN, SVM, and RF as base learners and MLP as a meta-learner, all using the same hyperparameters used for experiments (1-1, 1-2, 1-3, and 1-4). Figure 17 shows the architecture of our ensemble model.

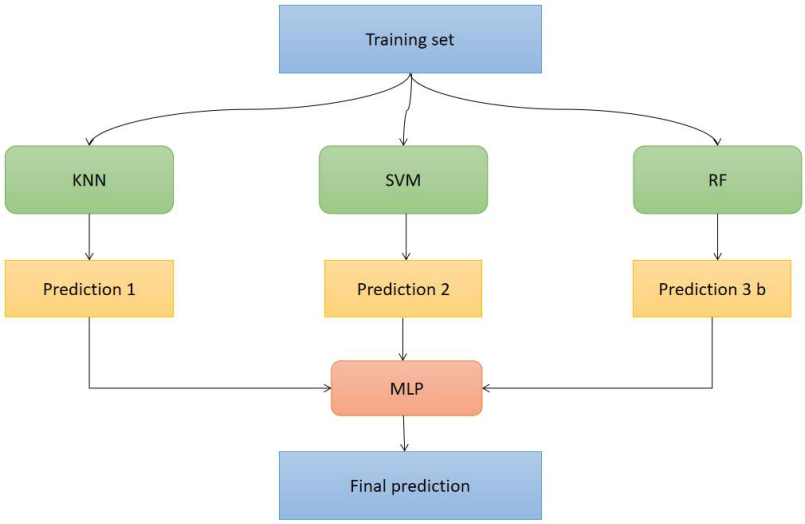


Figure 17: Ensemble Architecture

The base learners were trained, and their predictions were then combined using the meta-learner to predict customer churn. The meta-learner is a higher-level model that learns from the performance of multiple base learners across different tasks or datasets. The method of meta-learning makes use of the meta-knowledge gathered from numerous prior learning experiences. It is different from base learners in the area of adaptation. While the base learners adapt to learn from the training dataset, the meta learner is designed to adapt to new tasks quickly based on the knowledge gained from the base learners. It learns the learning process itself, which enables it to generalize and adapt to new tasks with limited data.

The learning process follows a stacking approach. The base learners independently generate predictions on the training data, and these predictions are combined into a new dataset. The meta-learner (MLP) is then trained on this dataset to capture higher-level patterns and relationships among the diverse predictions made by the base learners. The meta-learner's training focuses on understanding how the base learners collectively contribute to the overall predictive performance. Once trained, the meta-learner is used to make final predictions on new data, leveraging the complementary strengths of the individual base learners. Algorithm 1 shows the pseudo code for our ensemble model

Algorithm1: Ensemble Learning with Meta Learner

Input:
- Training data (training_data)
- Testing data (testing_data)

Output:
- Ensemble predictions on testing data (final_predictions)
- Performance evaluation of the ensemble on testing data

Procedure:
1. Initialize base models:
   - KNN = create_KNN(n_neighbor =3, weights='distance', p=1)
   - SVM = create_SVM(C=50)
   - RF = create_RF(n_estimators=83, max_depth=10, min_samples_leaf=1,
     min_samples_split=2)

2. Train base models on the training data:
   - predictions_KNN = train(KNN, training_data)
   - predictions_SVM = train(SVM, training_data)
   - predictions_RF = train(RF, training_data)

3. Combine base model predictions into a meta dataset:
   - meta_dataset = concatenate(predictions_KNN, predictions_SVM, predictions_RF)

4. Initialize meta learner model:
   - meta_learner_MLP = create_meta_learner_MLP()

5. Train meta learner on the meta dataset with true labels:
   - MLP_predictions = train(meta_learner_MLP, meta_dataset, true_labels)

6. Combine base model predictions on testing data:
   - predictions_KNN_test = predict(KNN, testing_data)
   - predictions_SVM_test = predict(SVM, testing_data)
   - predictions_RF_test = predict(RF, testing_data)

7. Combine testing data predictions into a meta dataset:
   - meta_dataset_test = concatenate(predictions_KNN, predictions_SVM, predictions_RF)

8. Use meta learner to make final predictions on testing data:
   - final_predictions = predict(meta_learner_MLP, meta_dataset_test)

9. Evaluate the performance of the ensemble on the testing data:
   - evaluate(final_predictions, true_labels_test)

End Algorithm

The ensemble classifier was implemented using the sklearn.ensemble package in Python. The code for calling base learners, meta learner, and the stack ensemble function is shown in figure 18.

```
from sklearn.ensemble import StackingClassifier

base_models = [
    ('knn', KNeighborsClassifier(n_neighbors=3, weights='distance', p=2)),
    ('svm', SVC(C=50)),
    ('rf', RandomForestClassifier(max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=83))
]

# Step 2: Define your meta-model
meta_model =MLPClassifier(hidden_layer_sizes=(10,), solver = 'adam')

# Step 3: Create the stacked ensemble model
stacked_model = StackingClassifier(estimators=base_models, final_estimator=meta_model)

stacked_model.fit(X_train.values, y_train.values)
stack_Y_pred = stacked_model.predict(X_test.values)
```

Figure 18: Ensemble function call

The ensemble classifier achieved 80.88% accuracy, 65.27% precision, 54.42% recall, and a 59.37% F1-score. Figure 19 displays the confusion matrix for the ensemble. Table 4 presents the results of the ensemble classifier compared with the four previous classifiers.

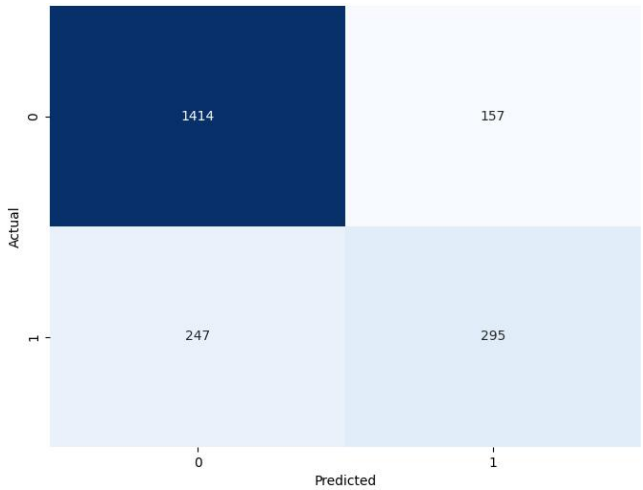


Figure 19: Ensemble confusion matrix

4.1.9 Building the baseline system

To determine our baseline system, we selected the one exhibiting the highest F1-score. The choice of F1-score as our comparative metric is supported by its substantial importance in the context of churn-related problems. The misclassification of a customer who is likely to churn can have consequential ramifications for a business. Precision, denoting proficiency in accurately identifying actual churners, and recall, signifying the ability to capture all factual churners, assume critical significance in this domain. The F1 score, representing the harmonic mean of precision and recall, offers a balanced evaluation of a model's efficacy.

According to Table 4, the ensemble method yielded the highest F1-score at 59.37%, surpassing the Random Forest at 58.14%. Subsequently, the MLP, SVM, and KNN achieved F1-scores of 53.66%, 44.56%, and 46.48%, respectively. Consequently, the ensemble model was selected as the baseline system for our research.

Table 4: Results of ensemble compared with other classifiers

Classifier	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
KNN	74.07	49.38	43.91	46.48
SVM	79.27	70.97	32.47	44.56
RF	80.79	65.89	52.03	58.14
MLP	79.65	64.51	45.94	53.66
Ensemble	80.88	65.27	54.42	**59.37**

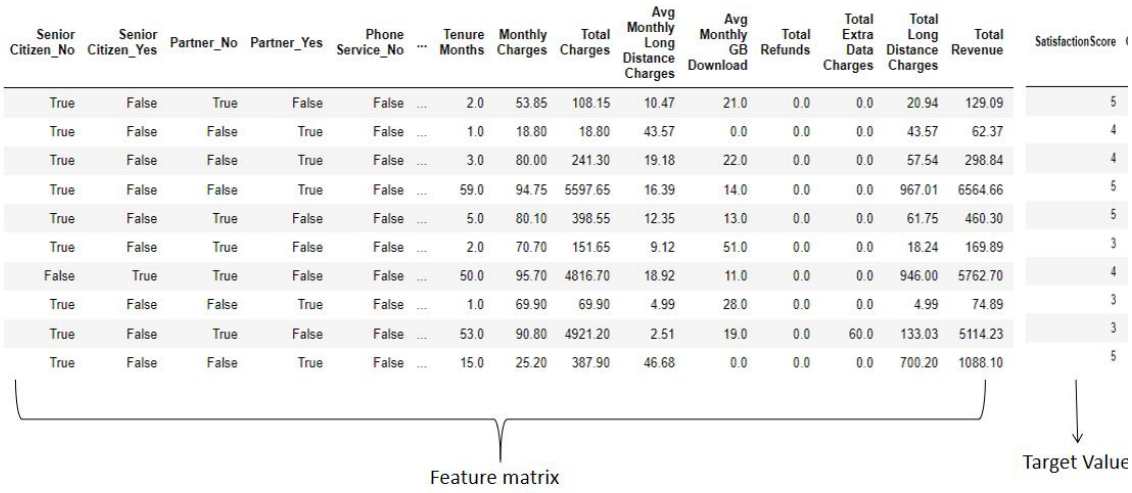
4.2 Experiment-2 Building a churn prediction system using sentiment as a feature

In this experiment, we incorporated sentiment as a feature in the churn prediction model. The initial phase encompassed sentiment classification, wherein we started by selecting the most effective classifiers for sentiment from a variety of options. Subsequently, this chosen classifier was implemented multiple times, each iteration corresponding to different sentiment levels. Finally, sentiment was added to the churn prediction model to assess its impact on identifying churners.

4.2.1 Building the sentiment classifier

To construct the sentiment classification model, the four previously mentioned classifiers were implemented with default settings to identify the classifier with the highest F1-score in the 5-class sentiment classification. The data fed to sentiment classifiers differed from the data used

for churn prediction in terms of the number of features used. In sentiment classification, we utilized all features within the database, and the target variable was the sentiment score. Figure 20 shows a part of sentiment classification data frame .



Senior Citizen_No	Senior Citizen_Yes	Partner_No	Partner_Yes	Phone Service_No	...	Tenure Months	Monthly Charges	Total Charges	Avg Monthly Long Distance Charges	Avg Monthly GB Download	Total Refunds	Total Extra Data Charges	Total Long Distance Charges	Total Revenue	SatisfactionScore
True	False	True	False	False	...	2.0	53.85	108.15	10.47	21.0	0.0	0.0	20.94	129.09	5
True	False	False	True	False	...	1.0	18.80	18.80	43.57	0.0	0.0	0.0	43.57	62.37	4
True	False	False	True	False	...	3.0	80.00	241.30	19.18	22.0	0.0	0.0	57.54	298.84	4
True	False	False	True	False	...	59.0	94.75	5597.65	16.39	14.0	0.0	0.0	967.01	6564.66	5
True	False	True	False	False	...	5.0	80.10	398.55	12.35	13.0	0.0	0.0	61.75	460.30	5
True	False	True	False	False	...	2.0	70.70	151.65	9.12	51.0	0.0	0.0	18.24	169.89	3
False	True	True	False	False	...	50.0	95.70	4816.70	18.92	11.0	0.0	0.0	946.00	5762.70	4
True	False	False	True	False	...	1.0	69.90	69.90	4.99	28.0	0.0	0.0	4.99	74.89	3
True	False	True	False	False	...	53.0	90.80	4921.20	2.51	19.0	0.0	60.0	133.03	5114.23	3
True	False	False	True	False	...	15.0	25.20	387.90	46.68	0.0	0.0	0.0	700.20	1088.10	5

Figure 20: A part of Data frame for sentiment classification

KNN, SVM, and Random Forests can handle multi-class classification tasks naturally using the Scikit_Learn. The key distinction arises when the target variable (y) is fed to the model with a different number of classes, an automated scenario in which the functions can adeptly adjust. Therefore, we applied each model with default parameters as shown in Figure 21.

```
rf_clf = RandomForestClassifier()
knn_clf = KNeighborsClassifier()
sv_clf = SVC(random_state=42)
```

Figure 21: Default call for RF, KNN, and SVM

To explore the parameters assigned by default to each classifier, we run *.get_params()* function after the name of each classifier and get the result that demonstrated in Figure 22.

```
bootstrap: True
ccp_alpha: 0.0
class_weight: None
criterion: gini
max_depth: None
max_features: sqrt
max_leaf_nodes: None
max_samples: None
min_impurity_decrease: 0.0
min_samples_leaf: 1
min_samples_split: 2
min_weight_fraction_leaf: 0.0
n_estimators: 100
n_jobs: None
oob_score: False
random_state: 42
verbose: 0
warm_start: False
```

(a) Default hyperparameter for RF

```
algorithm: auto
leaf_size: 30
metric: minkowski
metric_params: None
n_jobs: None
n_neighbors: 5
p: 2
weights: distance
```

(b) Default hyperparameter for KNN

```
C: 1.0
break_ties: False
cache_size: 200
class_weight: None
coef0: 0.0
decision_function_shape: ovr
degree: 3
gamma: scale
kernel: rbf
max_iter: -1
probability: False
random_state: 42
shrinking: True
tol: 0.001
verbose: False
```

(c) Default hyperparameter for SVM

Figure 22: Default hyper- parameters for RF, SVM, KNN

Regarding MLP, the number of neurons in the output layer of the MLP classifier was adjusted to 5 neurons instead of 2, and the softmax activation function was applied to the output layer. Figure 23 shows the architecture of MLP.

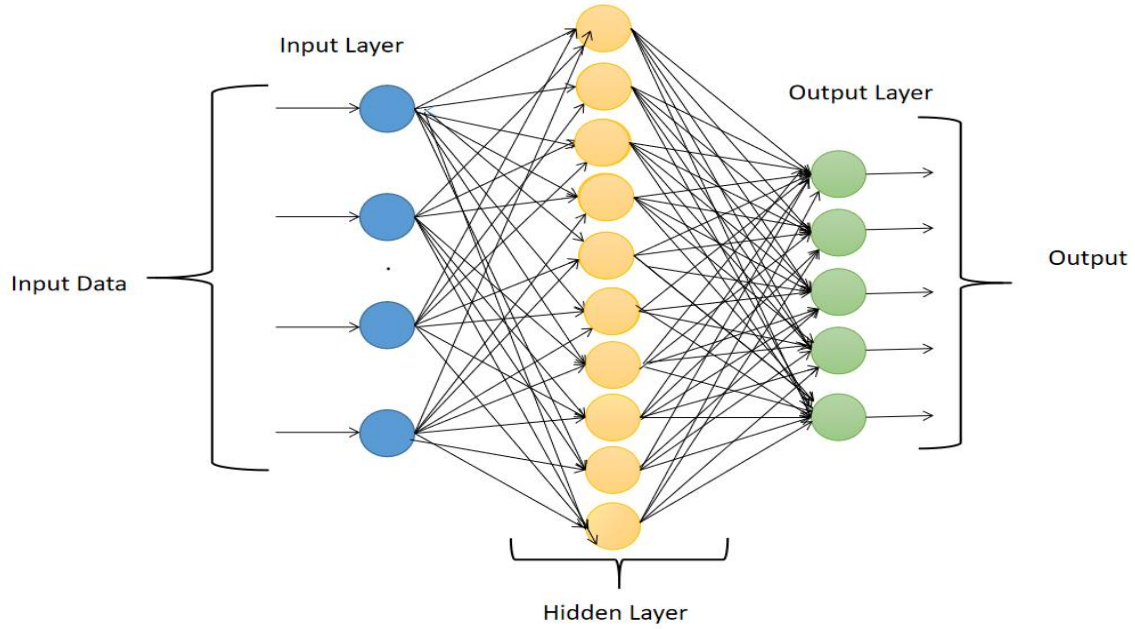


Figure 23: MLP architecture for 5-score sentiment classification

The evaluation of the models also underwent changes. Recall, precision, and F1-score were calculated separately for each class, and then the final score for each metric was the

average of the metrics across all classes, as described in [45]. The results for the four classifiers on the 5-score sentiment are presented in Table 5.

Table 5: Results of four sentiment classification for KNN, SVM, RF, and MLP

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
KNN	31.20	29.94	31.20	30.43
SVM	38.16	14.56	38.16	21.08
RF	43.95	41.76	43.95	41.87
MLP	32.60	37.31	32.60	29.05

After reviewing Table 5, it is evident that the random forest classifier yielded the highest F1-score. Consequently, it will be utilized for sentiment classification.

4.2.2 Running the Sentiment Classifier

The random forest algorithm was applied to categorize customers based on three sentiment levels: a 5-score sentiment level, a 3-score sentiment level, and a 2-score sentiment level. The preparation of sentiment score levels was detailed in Subsection 3.1.1. The random forest was executed three times with an identical structure. In the process of both binary and multi-classification prediction, each tree 'votes' for a class, and the class with the majority of votes across all trees is designated as the final predicted class. Consequently, we invoked the same default random forest function, as illustrated in Figure 14, during each iteration. The identical hyperparameters depicted in Figure 15 (a) were also explored with similar values for the three types of classification. However, the 'n_classes_' attribute revealed a varying number of classes, aligning with the number of classes in the target variable (y). The three iteration will be explained in the following subsections:

4.2.2.1 Sentiment classification with 5 categories

In this phase, the random forest was utilized to classify sentiment into five distinct categories: strongly positive, positive, neutral, negative, and strongly negative. The primary alteration in the Random Forest architecture was setting the 'n_classes' attribute to 5, corresponding to the number of classes in the target variable. This attribute was set by default when we fed the RF model with target y, which involves 5 classes. Notably, customers with neutral sentiment constituted the largest subset, representing 73.8%, followed by those with positive sentiment at 25.4%, while very positive, very negative, and negative sentiments

accounted for 16.3%, 13.1%, and 7.4%, respectively.

4.2.2.2 Sentiment classification with 3 categories

In this phase, the random forest was applied to classify sentiment into positive, neutral, and negative categories. For this iteration, the 'n_classes' attribute was set to 3. In this classification, customers with sentiment scores of 4 and 5 represented the positive class, those with a sentiment score of 3 represented the neutral class, and those with sentiment scores of 1 and 2 constituted the negative class. The distribution of sentiments in this iteration was 37.8%, 41.7%, and 20.4% for positive, neutral, and negative sentiments, respectively.

4.2.2.3 Sentiment classification with 2 categories

In this phase, the random forest was applied to classify sentiment into positive and negative classes. Here, the 'n_classes' attribute retained the default setting of 2. The positive class encompassed customers with a sentiment score of 3 or higher, constituting 79.6%, while the negative class included customers with a sentiment score less than 3, representing 20.4%. The distribution of sentiments for each level is depicted in Figure 24, and the results of sentiment classification are demonstrated in Table 6.

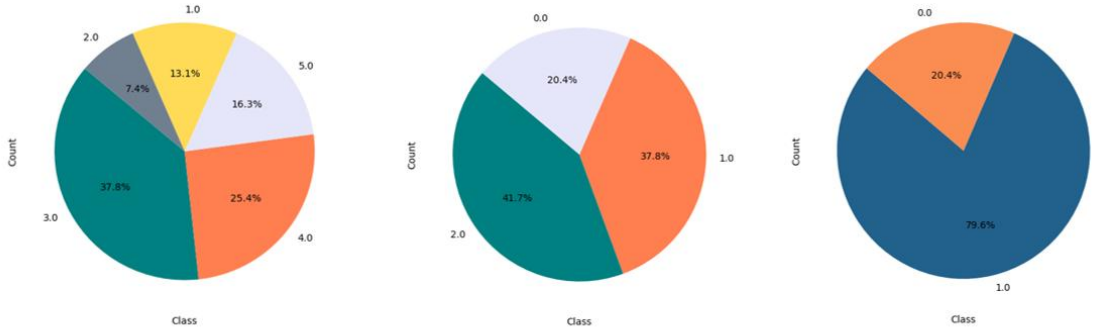


Figure 24: Sentiment distribution

The pseudocode for applying these steps is illustrated in Algorithm 2.

Algorithm 2: Sentiment Classification with Random Forest

Input:
- 5_score Sentiment Data: Features (X) and target variable (y) with 5 classes
- 3_score Sentiment Data:Features (X) and target variable (y) with 3 classes
- 2_score Sentiment Data:Features (X) and target variable (y) with 2 classes

1. First Iteration (5-score Sentiment Data):
  - Create a Random Forest classifier (rf_5score) for 5 distinct sentiment categories.
  - Train the classifier on X_train and y_train_5score.
  - Make predictions on X_test to obtain predictions_5score.
  - Evaluate the distribution of sentiments and performance metrics for each sentiment class, then average them.

2. Second Iteration (3-score Sentiment Data):
  - Create a Random Forest classifier (rf_3score) for positive, neutral, and negative categories.
  - Train the classifier on X_train and y_train_3score.
  - Make predictions on X_test to obtain predictions_3score.
  - Evaluate the distribution of sentiments and performance metrics for each sentiment class, then average them.

3. Final Iteration (2-score Sentiment Data):
  - Create a Random Forest classifier (rf_2score) for positive and negative classes.
  - Train the classifier on X_train and y_train_2score.
  - Make predictions on X_test to obtain predictions_2score.
  - Evaluate the distribution of sentiments and performance metrics for the 2-score sentiment classes.

Output:
- Results of sentiment classification for each iteration.

Table 6: Sentiment classification results using RF

Sentiment level	Accuracy (%)	AUC (%)	Precision (%)	Recall (%)	F1-score (%)
5-score sentiment	43.95	79.51	41.76	43.95	41.87
3-score sentiment	64.85	79.50	63.32	64.85	63.35
2-score sentiment	93.61	96.59	98.51	93.45	95.91

After conducting sentiment classification, we integrated the sentiment predictions with features selected through mutual information, as delineated in Table 2. We appended the sentiment score for each customer to the features associated with the same customer using the corresponding index. This combination results in a new dataset that serves as input into the baseline model for churn prediction. The code for these steps was shown in Figure 25.

```
# Store data for the 1st iteration
fed_data_5score = feature_mi_df.copy()
fed_data_5score['sentiment_5score_pred']=rf_prediction_5score
fed_data_5score['ChurnValue'] = df['ChurnValue']
fed_data_5score.to_csv('fed_data_5.csv', index=False)

# Store data for the 2nd iteration
fed_data_3score = feature_mi_df.copy()
fed_data_3score['sentiment_3score_pred']=rf_prediction_3score
fed_data_3score['ChurnValue'] = df['ChurnValue']
fed_data_3score.to_csv('fed_data_3.csv', index=False)

# Store data for the 3th iteration
fed_data_5score = feature_mi_df.copy()
fed_data_3score['sentiment_5score_pred']=rf_prediction_2score
fed_data_3score['ChurnValue'] = df['ChurnValue']
fed_data_3score.to_csv('fed_data_2.csv', index=False)
```

Figure 25: Integrate sentiment with other features

4.2.3 Adding Sentiment Predictions as feature in Churn Prediction Model

In this experiment, we retrain the ensemble classifier again on the three augmented databases to investigate the implications of incorporating sentiment. In the first iteration, the dataset that contained a 5-score sentiment was input into the ensemble model for the classification of customers into churners and non-churners. In the second iteration, the ensemble was trained on the dataset containing 3-score sentiment levels for churn prediction. Finally, the dataset comprising two-score sentiment levels was applied for churn prediction within the ensemble model. The performance of the ensemble demonstrated a notable

enhancement, as delineated in Table 7. This table presents the outcomes of the ensemble, offering insights into the discernible impact of incorporating sentiment.

Table 7: Ensemble results

Classifier	Input Data	Accuracy (%)	AUC (%)	F1-score (%)
Ensemble	No-sentiment	80.88	85.76	59.36
	5-score sentiment	96.88	97.74	93.66
	3-score sentiment	**98.86**	99.47	97.77
	2-score sentiment	98.20	97.56	96.42

Table 7 distinctly illustrates that optimal performance was achieved with datasets characterized by the 3-score sentiment level, followed by those with positive and negative sentiment levels, the 5-score sentiment level, and ultimately, the dataset devoid of sentiment. The AUC curves for the ensemble predictions across all statuses are illustrated in Figures 26.

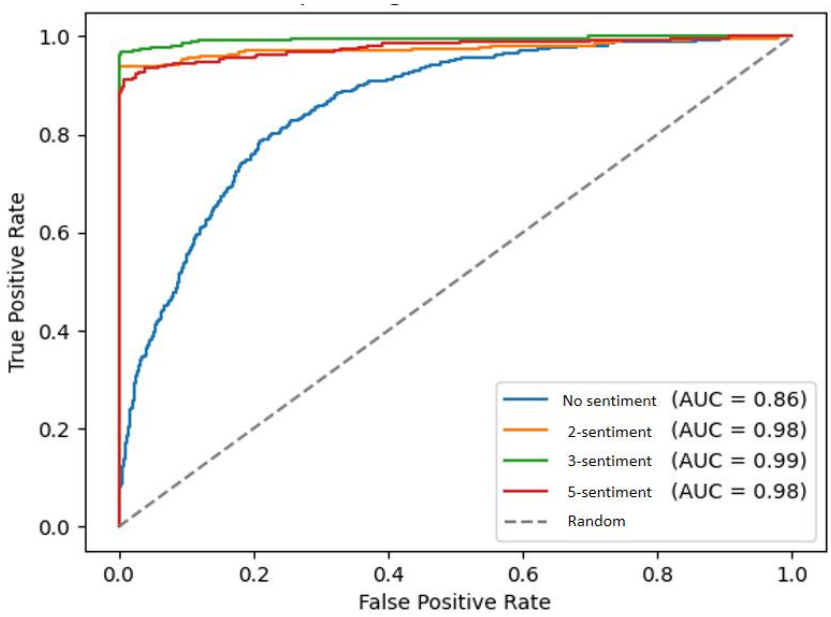


Figure 26: AUC curve for ensemble predictions

Figure 26 displays the Receiver Operating Characteristic (ROC) curve for ensemble classifiers when applied both without sentiment and with various sentiment levels. This graphical representation elucidates the classifiers' efficacy in discerning customer churn within a binary classification system.

The ROC curve in Figure 26 plots the False Positive Ratio (FPR) against the True Positive Ratio (TPR) at various probability thresholds ranging from 0 to 1 . These thresholds represent

the decision boundaries at which the predicted probabilities are used to classify instances into positive or negative classes. To generate multiple thresholds, we utilized *roc_curve* function from sklearn.metrics package. The predicted probabilities for the positive class was calculated. Then, starting from the maximum predicted probability, the function iterates through the sorted probabilities, treating each as a decision threshold. For each threshold, it classifies instances with predicted probabilities above the threshold as positive and those below as negative. At each threshold, the True Positive Rate (TPR) and False Positive Rate (FPR) were calculated based on the true labels (y_test) and the predicted binary labels obtained using the current threshold. The number of thresholds returned in the thresholds array is determined by the number of unique predicted probabilities. Essentially, each unique predicted probability becomes a candidate threshold. The call for *roc_curve* function is illustrated in Figure 27.

```
from sklearn.metrics import roc_curve, auc

y_scores = Ensemble.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)
```

Figure 27: prepare values for ROC curve

Finally, The TPR and FPR values at each threshold are used to build the ROC curve.The FPR represents the proportion of unchurned instances incorrectly classified as churned, while the True Positive Ratio (TPR) signifies the proportion of churned instances correctly identified. These metrics are formally defined in Equations 6 and 7:

$$\text{FPR} = \frac{FP}{FP+TN} \quad (6)$$

$$\text{TPR} = \frac{TP}{TP+FN} \quad (7)$$

Here, TP + FN corresponds to the total instances of churn, encompassing both correctly identified churners (TP) and instances where unchurned data was incorrectly classified as churned (FP). As the threshold changes, the TPR and FPR values change, resulting in different points on the ROC curve. Each threshold corresponds to a point on the ROC curve.

The Area Under Curve (AUC) criterion quantifies the model's ability to distinguish between different classes [46]. A higher AUC value signifies superior predictive performance, particularly in distinguishing class 0 as non-churning and class 1 as churning.

The optimal position for the ROC curve is situated at the coordinates (0, 1) in the upper-left corner [44]. As anticipated, the ROC curve for ensemble predictions without sentiment exhibits the lowest performance. Conversely, the performance of ensembles utilizing sentiment demonstrates improvement. The ROC curve for the ensemble with a 3-score sentiment leans more toward the upper-left corner, reaching a point of 99%, indicative of high TPR and low FPR. This implies the model's effectiveness in accurately identifying churners (TP) while minimizing misclassifications of unchurned instances as churned (FP). The ensemble predictions based on the 2-score and 5-score of sentiment also resulted in a high AUC score of approximately 98%. In

contrast, the model without sentiment reveals a gradual increase in false positives along with an increase in true positives.

4.3 Experiment 3 - Handle imbalanced data in 2-score sentiment classification:

The introduction of discrete positive and negative sentiments led to an imbalanced dataset. Consequently, we applied class weighting to handle this imbalance. A grid search was applied to determine optimal values for both classes within a range from 0 to 0.99, resulting in weights of 0.5067736032524353 for the minority class and 0.49322639674756474 for the majority class. The results of the 2-score sentiment classification before and after incorporating class weights are illustrated in Table 8. Noticeably, the impact of class-weighted was very low. The accuracy, AUC, precision, and F1-score increased only by 0.03%, 0.11%, 0.22%, and 0.01%, respectively, while the recall decreased by 0.18%.

Table 8: Sentiment classification results

Sentiment level	**Accuracy** (%)	**AUC** (%)	**Precision** (%)	**Recall** (%)	**F1-score** (%)
2-score sentiment	93.61	96.59	98.51	93.45	95.91
2-score sentiment + class-weighted	93.64	96.70	98.73	93.27	95.92

To assess the impact of addressing imbalanced data in churn prediction, we conducted an additional iteration for churn prediction based on the 2-score sentiment. In this iteration, we utilized predictions generated from the 2-score sentiment classification after addressing the imbalanced data. Table 9 illustrates the ensemble results before and after handling imbalanced data in sentiment classification.

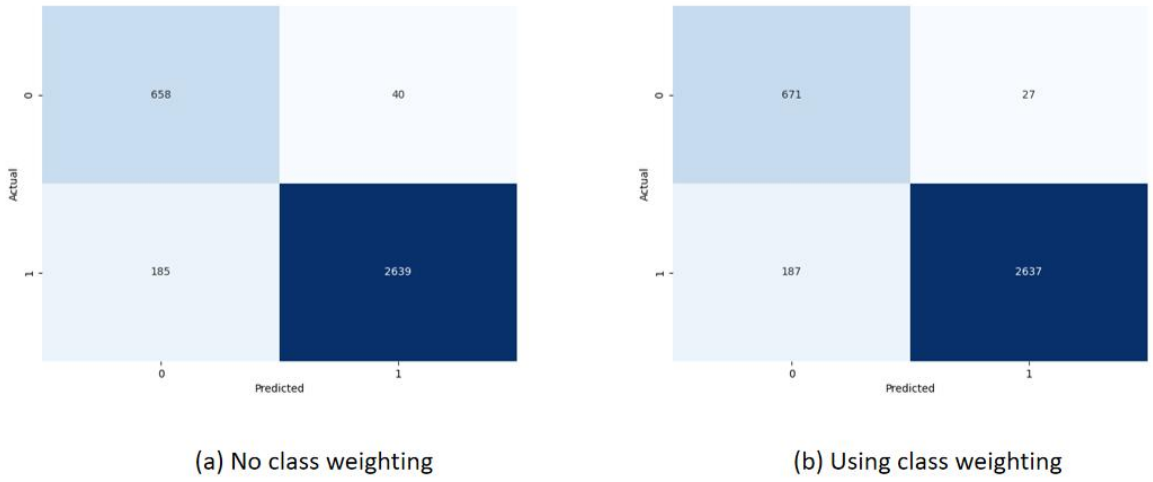
Table 9: Ensemble results based on 2-score sentiment

Classifier	**Input Data**	**Accuracy** (%)	**AUC** (%)	**F1-score** (%)
Ensemble	2-score sentiment + without class-weighted	98.20	97.56	96.42
	2-score sentiment + using class-weighted	98.39	98.00	96.81

Table 9 demonstrates a slight improvement in the F1-score for ensemble prediction based on the 2-score sentiment level. We expected this small improvement in churn prediction because the data that was input into the churn model after class weighting wasn't very improved from the one without it.

For further clarification, we took a close look at confusion matrices for sentiment classification before and after class weighted. These confusion matrices are illustrated in Figure 11. It is evident that the model, after the class-weighed technique, enhances the correct

classification of unsatisfied customers (the negative class) from 658 to 671 out of the entire number of unsatisfied customers, which is 698. This is good for our problem, as accurately predicting unsatisfied customers is of greater importance as they are more likely to churn. However, we can see that the model without class-weighted performed well in identifying unsatisfied customers, as Figure 28 (a) indicates that it correctly identified 658 out of the total 698. This implies that it only misclassified 40 unsatisfied customers. The justification for these results is that the random forest itself is effective and recommended for learning from imbalanced data [47].



(a) No class weighting (b) Using class weighting

Figure 28: 2-score sentiment classification confusion matrices

4.5 Comparison of the best Result with the previous studies

We conducted a comparative analysis of our optimal findings with state-of-the-art models discussed in the related work, which do not incorporate sentiment as a variable in predicting customer churn. The first study utilized a naive Bayes classifier to predict customer churn along with a features-weighted technique, enhancing their performance to a 96.1% F1-score. Another study employed the ensemble method, combining random forest classifiers with logistic regression and applying hyperparameter tuning using a grid search algorithm to achieve 95.2% accuracy. Moreover, there is another study that utilized a hybrid approach to predict customer churn by integrating the decision tree with clustering. This involved applying K-means clustering to generate different clusters and then applying the decision tree for each cluster and the overall dataset. The last approach produced low performance, achieving 79.5% accuracy for the full dataset. Our proposed model demonstrated its performance, surpassing the best among the existing models by approximately 1% for the F1-score. This result is shown in Table 10, illustrating how well the ensemble method works with datasets that have three levels of sentiment, using KNN, SVM, and RF as base learners, with MLP acting as the meta-learner.

Table 10: Comparison of Our Finding with Previous Related Works

Related work	**Accuracy** (%)	**AUC** (%)	**f1-score** (%)
Adaptive model [8]	97.5	-	96.1
RF-GS-LR [2]	95.2	94.6	-
Hybrid model (CHAID-DT)[7]	79.5	-	-
Ensemble with 3-score of sentiment (*Proposed model*)	**98.86**	**99.47**	**97.77**

Chapter 5

Conclusion and Future Work

Customer churn poses a substantial challenge for businesses, particularly within the telecommunications industry. Proactively predicting churn assists companies in mitigating its significant impact. Consequently, numerous prior studies have endeavored to refine models for forecasting customer churn. Despite various factors utilized in customer churn prediction, such as demographic information, service details, and call activities, the incorporation of sentiment as a feature for churn predictive models remains limited. This research examines the impact of adding sentiment as a feature for churn prediction. Throughout the research, we aim to answer our research question through several experiments.

In response to research question one, “Does the ensemble method enhance the churn prediction?" Our findings demonstrate notable enhancements in model performance through the implementation of the ensemble classifier, leading to an increase in the F1-score when compared to alternative classifiers by 1.23%. Addressing research question two, “Does including sentiment in different forms affect the customer churn prediction results?" The incorporation of sentiment at various levels slightly augmented the model's predictive capabilities for customer churn by improving the F1-score by 34.3%. Distinct levels of sentiment yielded improved results, showcasing varied outcomes based on the sentiment input levels, where 5-score sentiment achieved 93.66%, 3-score achieved 97.77%, and 2-score sentiment achieved 96.42% F1-score. In response to question three, “What is the best number of sentiment levels to predict churn?" Our observations indicate that the utilization of a 3-score sentiment classification contributed to optimal results. The ensemble model exhibited remarkable performance with 98.86% accuracy, a 99.47 AUC, and a 97.77 F1-score. Regarding question four, “Does handling imbalanced data improve the performance of the classification of the minority class?" A small performance boost was evident when employing the class-weighted method to address data imbalances in a 2-score sentiment classification model, resulting in a small improvement in churn prediction from 96.42% to 96.81% F1-score.

Overall, our contribution validates the significance of incorporating sentiment as a feature for predicting churners and delves into the distinctions arising from the inclusion of various levels of sentiment in churn prediction. Additionally, we illustrate the efficacy of the

ensemble model in enhancing the F1-score when compared to alternative machine learning classifiers, including KNN, SVM, RF, and MLP. Moreover, we highlight how handling imbalanced data in sentiment classification affects customer churn prediction, depending on how much sentiment classification improves.

In our research, we investigate the relationship between sentiment analysis and churn prediction by incorporating sentiment at different levels as a feature within a churn predictive model to assess its impact on enhancing the churn prediction model.
Nevertheless, there is still potential for additional investigation. We can examine the correlation between sentiment and churn using methodologies applied in similar works but with private datasets. These methodologies may involve applying statistical tools such as Pearson correlation or Binomial Logistic Regression to elucidate the connection between negative sentiment rates and churn rates.

Another possibility for future research involves conducting similar experiments on datasets comprising customer reviews and corresponding churn statuses. Subsequently, sentiment analysis tools can be implemented to extract sentiment polarity from the reviews, which can then be utilized as a feature in the churn prediction model. Additionally, applying topic modeling techniques to such datasets can facilitate the identification of the most influential factors contributing to customer churn.

Reference

[1] Lalwani, P., Mishra, M. K., Chadha, J. S., & Sethi, P. (2022). Customer churn prediction system: A machine learning approach. Computing, 104(2), 271–294. https://doi.org/10.1007/s00607-021-00908-y

[2] Edwine, N., Wang, W., Song, W., & Ssebuggwawo, D. (2022). Detecting the Risk of Customer Churn in Telecom Sector: A Comparative Study. Mathematical Problems in Engineering, 2022, 1–16. https://doi.org/10.1155/2022/8534739

[3] Y., N. N., Ly, T. V., & Son, D. V. T. (2022). Churn prediction in telecommunication industry using kernel Support Vector Machines. PLOS ONE, 17(5), e0267935. https://doi.org/10.1371/journal.pone.0267935

[4] Geiler, L., Affeldt, S., & Nadif, M. (2022). A survey on machine learning methods for churn prediction. International Journal of Data Science and Analytics, 14(3), 217–242. https://doi.org/10.1007/s41060-022-00312-5

[5] Almuqren, L., Alrayes, F. S., & Cristea, A. I. (2021). An Empirical Study on Customer Churn Behaviours Prediction Using Arabic Twitter Mining Approach. Future Internet, 13(7) , 175. https://doi.org/10.3390/fi13070175

[6] Ki, Li, Mci., Z. H. (2022). The Effectiveness of Homogeneous Classifier Ensembles on Customer Churn Prediction in Banking, Insurance, and Telecommunication Sectors. International Journal of Computational and Experimental Science and Engineering, 8(3), 77–84. https://doi.org/10.22399/ijcesen.1163929

[7] Pejić Bach, M., Pivar, J., & Jaković, B. (2021). Churn Management in Telecommunications: Hybrid Approach Using Cluster Analysis and Decision Trees. Journal of Risk and Financial Management, 14(11), 544. https://doi.org/10.3390/jrfm14110544

[8] Amin, A., Adnan, A., & Anwar, S. (2023). An adaptive learning approach for customer churn prediction in the telecommunication industry using evolutionary computation and Naïve Bayes. Applied Soft Computing, 137, 110103. https://doi.org/10.1016/j.asoc.2023.110103

[9] Khoh, W. H., Pang, Y. H., Ooi, S. Y., Wang, L.-Y.-K., & Poh, Q. W. (2023). Predictive Churn Modeling for Sustainable Business in the Telecommunication Industry: Optimized Weighted Ensemble Machine Learning. Sustainability, 15(11), 8631. https://doi.org/10.3390/su15118631

[10] Ranjan, S., & Sood, S. (2018). Twitter Sentiment Analysis of Indian Telecom Companies for subscriber churn prediction.

[11] Customer Churn Prediction in Telecommunication Industry Using Deep Learning. (2022). Information Sciences Letters, 11(1), 185–198.

[12] Effrosynidis, D., & Arampatzis, A. (2021). An evaluation of feature selection methods for environmental data. Ecological Informatics, 61, 101224. https://doi.org/10.1016/j.ecoinf.2021.101224

[13] Zhang S, Zhu F, Yu Q, Zhu X (2021) Identifying DNA-binding proteins based on multi-features and LASSO feature selection. Biopolymers 112:e23419. https://doi.org/10.1002/bip.23419

[14] Chouiekh, A., & El Haj, E. H. I. (2020). Deep Convolutional Neural Networks for Customer Churn Prediction Analysis: International Journal of Cognitive Informatics and Natural Intelligence, 14(1), 1–16 https://doi.org/10.4018/IJCINI.2020010101

[15] Wu, F., Ren, J., Lyu, F., Yang, P., Zhang, Y., Zhang, D., & Zhang, Y. (2022). Boosting Internet Card Cellular Business via User Portraits: A Case of Churn Prediction. IEEE INFOCOM 2022 -

IEEE Conference on Computer Communications, 640–649. https://doi.org/10.1109/INFOCOM48880.2022.9796869
[16] Tavassoli, S., & Koosha, H. (2022). Hybrid ensemble learning approaches to customer churn prediction. Kybernetes, 51(3), 1062–1088. https://doi.org/10.1108/K-04-2020-0214
[17] Prabadevi, B., Shalini, R., & Kavitha, B. R. (2023). Customer churning analysis using machine learning algorithms. International Journal of Intelligent Networks, 4, 145–154. https://doi.org/10.1016/j.ijin.2023.05.005
[18] Xu, T., Ma, Y., & Kim, K. (2021). Telecom Churn Prediction System Based on Ensemble Learning Using Feature Grouping. Applied Sciences, 11(11), 4742. https://doi.org/10.3390/app11114742
[19] Zhu, B., Qian, C., Vanden Broucke, S., Xiao, J., & Li, Y. (2023). A bagging-based selective ensemble model for churn prediction on imbalanced data. Expert Systems with Applications, 227, 120223. https://doi.org/10.1016/j.eswa.2023.120223
[20] Shabankareh, M. J., Shabankareh, M. A., Nazarian, A., Ranjbaran, A., & Seyyedamiri, N. (2022). A Stacking-Based Data Mining Solution to Customer Churn Prediction. Journal of Relationship Marketing, 21(2), 124–147. https://doi.org/10.1080/15332667.2021.1889743
[21] Jajam, N., Challa, N. P., Prasanna, K. S. L., & Deepthi, C. H. V. S. (2023). Arithmetic Optimization With Ensemble Deep Learning SBLSTM-RNN-IGSA Model for Customer Churn Prediction. IEEE Access, 11, 93111–93128. https://doi.org/10.1109/ACCESS.2023.3304669
[22] Ahmed, M., Afzal, H., Siddiqi, I., Amjad, M. F., & Khurshid, K. (2020). Exploring nested ensemble learners using overproduction and choose approach for churn prediction in telecom industry. Neural Computing and Applications, 32(8), 3237–3251. https://doi.org/10.1007/s00521-018-3678-8
[23] Ahmed, A. A. Q., & Maheswari, D. (2019). An enhanced ensemble classifier for telecom churn prediction using cost based uplift modelling. International Journal of Information Technology, 11(2), 381–391. https://doi.org/10.1007/s41870-018-0248-3
[24] Khoh, W. H., Pang, Y. H., Ooi, S. Y., Wang, L.-Y.-K., & Poh, Q. W. (2023). Predictive Churn Modeling for Sustainable Business in the Telecommunication Industry: Optimized Weighted Ensemble Machine Learning. Sustainability, 15(11), 8631. https://doi.org/10.3390/su15118631
[25] Lappeman, J., Franco, M., Warner, V., & Sierra-Rubia, L. (2022). What social media sentiment tells us about why customers churn. Journal of Consumer Marketing, 39(5).
[26] Zeynep Hilal Kilimci (2022) Prediction of user loyalty in mobile applications using deep contextualized word representations, Journal of Information and Telecommunication, 6:1, 43-62, DOI: 10.1080/24751839.2021.1981684
[27] Galitsky, B., De La Rosa, J.-L., & Kovalerchuk, B. (2011). Discovering common outcomes of agents' communicative actions in various domains. Knowledge-Based Systems, 24(2), 210–229. https://doi.org/10.1016/j.knosys.2010.06.004
[28] IBM. (2022) Telco customer churn https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=samples-telco-customer-churn
[29] Phillips, R. V., Van Der Laan, M. J., Lee, H., & Gruber, S. (2023). Practical considerations for specifying a super learner. International Journal of Epidemiology, 52(4), 1276–1285. https://doi.org/10.1093/ije/dyad023
[30] Gulati, V., & Raheja, N. (2021). Efficiency Enhancement of Machine Learning Approaches through the Impact of Preprocessing Techniques. 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), 191–196. https://doi.org/10.1109/ISPCC53510.2021.9609474
[31] Zhao, M., Zeng, Q., Chang, M., Tong, Q., & Su, J. (2021). A Prediction Model of Customer

Churn considering Customer Value: An Empirical Research of Telecom Industry in China. Discrete Dynamics in Nature and Society, 2021, 1–12. https://doi.org/10.1155/2021/7160527
[32] Kumar Mohapatra, S., Mishra, S., Tripathy, H. K., & Alkhayyat, A. (2022). A sustainable data-driven energy consumption assessment model for building infrastructures in resource constraint environment. Sustainable Energy Technologies and Assessments, 53, 102697. https://doi.org/10.1016/j.seta.2022.102697
[33] Macedo, F., Valadas, R., Carrasquinha, E., Oliveira, M. R., & Pacheco, A. (2022). Feature selection using Decomposed Mutual Information Maximization.
[34] Kramer, O. (2013). K-Nearest Neighbors. In O. Kramer, Dimensionality Reduction with Unsupervised Nearest Neighbors (Vol. 51, pp. 13–23). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38652-7_2
[35] Suthaharan, S. (2016). Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning (Vol. 36). Springer US. https://doi.org/10.1007/978-1-4899-7641-3
[36] Rigatti, S. J. (2017, January 1). Random Forest. Journal of Insurance Medicine. https://doi.org/10.17849/insm-47-01-31-39.1
[37] Bharadwaj, S., Anil, B. S., Pahargarh, A., Pahargarh, A., Gowra, P. S., & Kumar, S. (2018). Customer Churn Prediction in Mobile Networks using Logistic Regression and Multilayer Perceptron(MLP). 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT), 436–438. https://doi.org/10.1109/ICGCIoT.2018.8752982
[38] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, Nguyen, N. N., & Duong, A. T. (2021). Comparison of Two Main Approaches for Handling Imbalanced Data in Churn Prediction Problem. Journal of Advances in Information Technology, 12(1), 29–35. https://doi.org/10.12720/jait.12.1.29-35
[39] Pustokhina, I. V., Pustokhin, D. A., Nguyen, P. T., Elhoseny, M., & Shankar, K. (2023). Multi-objective rain optimization algorithm with WELM model for customer churn prediction in telecommunication sector. Complex & Intelligent Systems, 9(4), 3473–3485. https://doi.org/10.1007/s40747-021-00353-6
[40] D.J. Hand, P. Christen, N. Kirielle, F*: an interpretable transformation of the F-measure, Mach. Learn. 110 (3) (2021) 451–456
[41] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (n.d.). Scikit-learn: Machine Learning in Python. MACHINE LEARNING IN PYTHON.
[42] Matplotlib — Visualization with Python. (n.d.). https://matplotlib.org/
[43] Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021.
[44] Wu, S., Yau, W.-C., Ong, T.-S., & Chong, S.-C. (2021). Integrated Churn Prediction and Customer Segmentation Framework for Telco Business. IEEE Access, 9, 62118–62136. https://doi.org/10.1109/ACCESS.2021.3073776
[45] Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-Label Confusion Matrix. IEEE Access, 10, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048
[46] Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-Label Confusion Matrix. IEEE Access, 10, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048
[47] Khoshgoftaar, T. M., Golawala, M., & Hulse, J. V. (2007). An Empirical Study of Learning from Imbalanced Data Using Random Forest. 19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007), 310–317. https://doi.org/10.1109/ICTAI.2007.46