Customer Churn Prediction In Telecommunication Industry Using Machine Learning

Faculty of Mathematical Sciences and Informatics in the University of Khartoum

Alsura Mohamed Ahmed Elmubarak 015-219
Azza Omer Mohamed Omer 015-118
Ranas Ali Mohamed Ahmed Daffa Allah 015-526

Supervisor: Dr. Doha Elsharief

January 2023

Abstract

Customer churn prediction is the process of identifying customers who have a high tendency to cancel the service they get for a variety of reasons. Due to its direct impact on the company's revenues, telecommunications companies are working to create methods to predict probable customer churn and further figure out the necessary operational procedures to avoid this cancellation.

This study investigates the most suitable machine learning model to predict customer churn. Particularly, several machine learning models including Logistic Regression, Decision Trees, Support Vector Machines, Gradient Boosting Classifier, and CatBoost were implemented and optimized based on a relevant dataset obtained from Kaggle. The analysis process was performed through applying cross-validation and feature engineering on the gathered data, and in order to measure the performance of the developed models, the Area Under Curve (AUC) standard evaluation metric was adopted.

The results of our research shows that feature engineering has huge impact on enhancing the model's performance. It is also demonstrated that the most successful model in handling the customer losing problem is the CatBoost Model.

ملخص

أحب القراءة كثيرا

Acknowledgment

First and foremost praise to almighty God for providing us with faith and strength to thrive through our way.

Words cannot express our gratitude to our supervisor, Dr Doha Elsharief, for her invaluable help, support and feedback. We could not have undertaken this research without the valuable consultancy from Muhammed Eltayeb, who generously provided his extended knowledge, expertise and moral support.

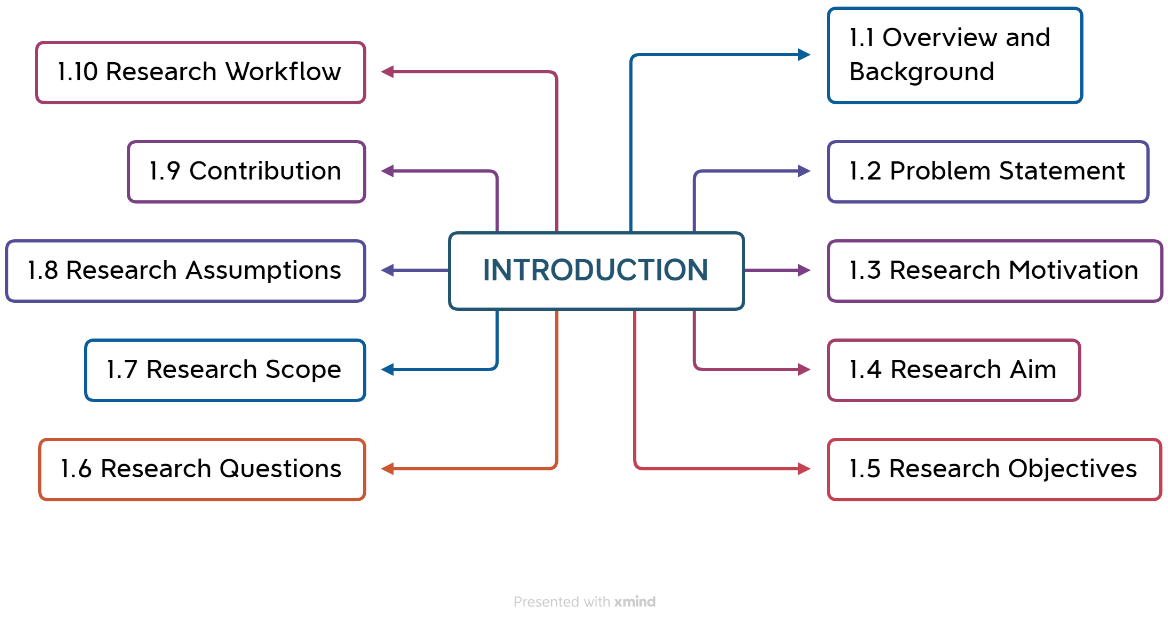
We are also grateful to our classmates and friends who impacted and inspired us. Lastly, we would be remiss in not mentioning our family, especially our parents. Their belief in us has kept our spirits and motivation high throughout this research.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION



1.1 Overview and background

In this chapter, we introduce the reader to the problem under study, including brief definitions and a quick summary of the problem, research aim and objectives, the study's assumptions, and limits.

1.1.1 Customer Churn

Customer Churn (also called customer attrition or churn rate) is the percentage of the total churners from the total subscribers. In other words, customer churn is the percentage of customers that stopped using the company's product or service during a specific time frame. Or simply given, it's when customers change their services from one service provider to another.

1.1.2 Customer Churn prediction and Customer Retention

Customers that have gone dormant for more than three months are said to be churner subscribers in the mobile telecom industry. Because it directly affects a company's revenue, customer retention is one of the most crucial operational goals in the telecommunications sector. As a result, businesses engage in a range of operations to build lasting relationships with their customers and to reduce customer churn.

Understanding customer switching behaviour and determining the causes of churning are crucial for achieving customer retention operations. Understanding customer switching behaviour and determining the causes of churning are crucial for achieving customer retention operations. Customer retention is made simple and successful for the company through customer churn prediction, which helps companies keep consumers. Businesses must proactively identify consumers who are likely to churn by examining their behaviour, and then aim to focus their efforts and resources on keeping those customers.

In general High competition, the cost of acquiring new customers and retaining the current ones are some of the main elements affecting the churn rate in the telecommunication industry in Sudan. if not adequately addressed, A high churn rate has an immediate impact on the company, slowing or preventing its growth.

Retaining customers, which has been shown to be more profitable than gaining new ones, will be aided by accurately anticipating consumers who are going to churn on time. It is always difficult to predict consumers who are about to churn in the telecom industry and to create efficient churn prediction models to identify probable churners due to the variety of features and dynamic nature of data.

1.2 Problem Statement

Investigating the most suitable machine learning model to predict the customer churn problem in the telecommunication industry based on several evaluation metrics.

1.3 Research Motivation

In Sudan, the conflict between telecommunication companies is growing stronger, now more than ever due to the inadequate use of machine learning models in identifying early churn signals to retain customers. the approach to solving the churn problem was by conducting a meeting every three months to traditionally check whether the customer stopped using the company's service or not. later came the approach of using machine learning models but still, it is not equipped in a way that can produce the most efficient results possible.

Machine learning techniques have been used for the statistical analysis and validation of datasets, These techniques have also been used for the classification of data due to their ability to handle complex problems efficiently.

The proposed work is motivated by the fact that the developed churn prediction models will be able to predict potential churners in telecommunications customers' data accurately thereby helping companies to identify the customers at the point of leaving them.

1.4 Research Aim

Identify early churn signals in the telecommunication industry to help retain customers.

1.5 Research objectives

The objectives satisfying the aim:

- Investigate the most accurate customer churn prediction model.
- Determine which features have the greatest impact on churn likelihood

1.6 Research Questions

If we want to achieve these objectives, we need to answer the following questions:

1. Which prediction model is more accurate based on the available dataset?
2. What are the determinant factors of Customer Churn?

1.7 Research Scope

This Research focuses mainly on solving the problem of customer churning for the telecommunications sector.

1.8 Research Assumptions

The research assumes that the online data acquired from Kaggle[1] can satisfy the research needs in deriving a solution to local Customer Churn problems.

1.9 Contribution

The main contribution of our work is to develop a churn prediction model which assists Sudanese telecom operators to predict customers who are most likely subject to churn by performing feature engineering on the data before it is fed to the model, which improved CatBoost's performance.

1.10 Research Workflow

The workflow of this study is explained in the following flowchart

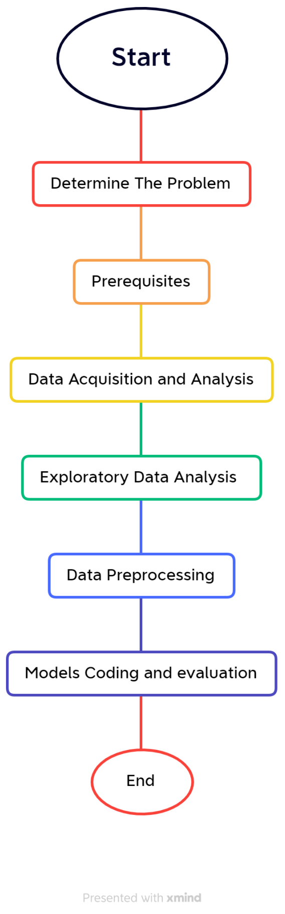
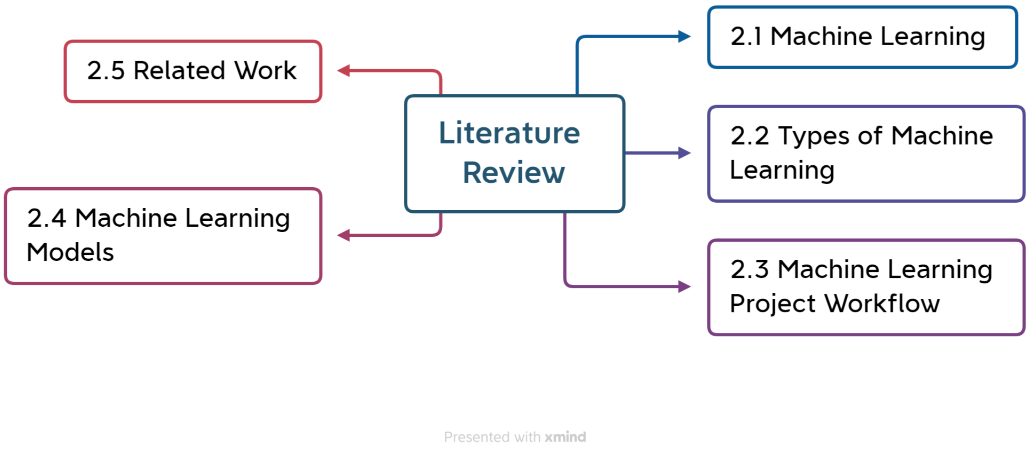

Start
Determine The Problem
Prerequisites
Data Acquisition and Analysis
Exploratory Data Analysis
Data Preprocessing
Models Coding and evaluation
End
Presented with xmind

Figure 1.1: Research Workflow

Chapter 2

LITERATURE REVIEW


Literature Review
2.1 Machine Learning
2.2 Types of Machine Learning
2.3 Machine Learning Project Workflow
2.4 Machine Learning Models
2.5 Related Work
Presented with xmind

This chapter highlights our project starting with a machine learning background and outlines the most popular techniques commonly used in predicting customer churn, and summarizes some related papers.

2.1 Machine Learning

according to A. Géron [10], Machine Learning is the science (and art) of programming computers so they can learn from data. a more engineering-oriented definition is:

"A computer program is said to learn from experience with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

Tom Mitchell 1997

In this case, task T is to predict churning customers, experience E is the training data, and the performance measure P needs to be defined; for example, you can use the ratio of correctly classified customers. This particular performance measure is called accuracy, and it is often used in classification tasks.

2.2 Types of Learning

1- Supervised learning
In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels. supervised learning tasks are classification and Regression. Regression is to predict a target numeric value, such as the price of a car, given a set of features (mile, age, brand, etc.).

2- Unsupervised learning
In unsupervised learning, as you might guess, the training data is unlabeled. it uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention

3- Semi Supervised learning
Semi-supervised learning is a learning problem that involves a small number of labelled examples and a large number of unlabeled examples. For this type specialized semis-supervised learning algorithms are required.

4- Reinforcement Learning
The learning system, of Reinforcement learning (called an agent in this context), can observe the environment, in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

2.3 Machine Learning Project Workflow

The basic Workflow of machine learning projects starts with taking a look at the big picture and framing the problem by answering questions such as:

- What is the objective of the business?
- What does the current solution look like?
- Is the system supervised during training?
- What kind of task is it? (classification or regression)
- How is the flow of data? is it continuous?

The above questions aid in gaining reference on performance and insight on how to solve the problem. Then, the next step then would be to gather the data, explore and visualize it to gain insights and prepare it for the model.
After selecting the most suitable model for the proposed problem and training it, comes the step of fine-tuning the model and lastly presenting the solution.

2.4 Machine Learning Models

In the following section, we briefly present five well-established and popular techniques used for churn prediction, taking into consideration reliability and efficiency.

1- Artificial Neural Network
Artificial Neural Networks (ANNs) are a popular approach to addressing complex problems, such as the churn prediction problem. Neural networks can be hardware-based (neurons are represented by physical components) or software-based (computer models), and can use a variety of topologies and learning algorithms. One popular supervised model is the Multi-Layer Perceptron trained with variations of the Back-Propagation algorithm (BPN). BPN is a feed-forward model with supervised learning. In the case of the customer churn problem. Neural networks achieve better performance compared to Decision Trees.

2-Support Vector Machines
Support Vector Machines (SVM), also known as Support Vector, are supervised learning models used for classification and regression analysis, with associated learning algorithms that analyze data and recognize patterns. SVM is a machine learning technique based on structural risk minimization where Kernel functions have been employed for improving performance. In the churn prediction problem, SVM outperforms DT and sometimes ANN, depending mainly on the type of data and data transformation.

2- Decision Trees Learning
Decision Trees (DTs) are tree-shaped structures representing sets of decisions capable to generate classification rules for a specific dataset or as Berry and Linoff noted [8] a structure that can be used to divide up a large collection of records into successively smaller sets of records by applying a sequence of simple decision rules More descriptive names for such tree models are Classification Trees or Regression Trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. DT has no great performance in capturing complex and nonlinear relationships between the attributes. Yet, in the customer churn problem, the accuracy of a DT can be high, depending on the form of the data.

3- Naïve Bayes
A Naïve Bayes (NB) classifier assumes that the presence (or absence) of a particular feature of a class (i.e., customer churn) is unrelated to the presence (or absence) of any other feature. In statistics, naive Bayes classifiers are a family of simple ”probabilistic classifiers” based on applying Bayes' theorem with strong (naive) independence assumptions between the features [2]. The NB classifier achieves good results on the churn prediction problem for the wireless telecommunications industry and it can also achieve improved prediction rates compared to other widely used algorithms.

4- Logistic Regression Analysis

Regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modelling and analyzing several variables when the focus is on the relationship between a dependent variable and one or more independent variables. In terms of customer churning, regression analysis is not widely used, and that is because linear regression models are useful for the prediction of continuous values. On the other hand, Logistic Regression or Logit Regression analysis (LR) is a type of probabilistic statistical classification model. It is also used to produce a binary prediction of a categorical variable (e.g., customer churn) which depends on one or more predictor variables (e.g., customers' features) In the churn prediction problem, Logistic Regression is usually used after proper data transformation is applied on initial data, with quite good performance and sometimes performs as well as Decision Trees.

5- Gradient Boosting Classifier

Regular gradient boosting algorithms seek to improve the prediction power by training a sequence of weak models, each overcoming the shortcomings of its predecessors. in gradient boosting Instead of adjusting the weights of data points, Gradient boosting focuses on the difference between the prediction and the ground truth. some may argue that it is the most accurate way of predicting customer attrition. Gradient boosting is also known as gradient tree boosting, stochastic gradient boosting (an extension), and gradient boosting machines, or GBM for short. Gradient boosting is an effective machine learning algorithm and is often the main, or one of the main, algorithms used to win machine learning competitions (like Kaggle) on tabular and similar structured datasets.

6- Gradient Boosting with CatBoost

CatBoost is a third-party library that provides an efficient implementation of the gradient-boosting algorithm. It is developed by Yandex researchers and engineers. CatBoost provides a gradient boosting framework which among other features attempts to solve for Categorical features using a permutation-driven alternative compared to the classical algorithm. Kaggle listed CatBoost as one of the most frequently used Machine Learning (ML) frameworks in the world. It was listed as the top-8 most frequently used ML framework in the 2020 survey [3].

2.5 Related Work

In this part, we will introduce a brief overview of results from similar studies. Throughout the years, many researchers compared different Machine learning and Deep learning models, some methodologies used direct machine-learning-based models while others used hybrid models with improved data analysis and Pre-processing.

In the year 2006 Guo-en1 and Wei-dong [17] applied the prediction abilities of support vector machine to customer churn prediction and compared this method with the artificial neural network, decision trees, logistic regression and naïve Bayesian classifier. They found that boosted SVM (SVM-POLY with AdaBoost) had the best accuracy rate and provided an effective measurement for customer churn with an accuracy rate of 0.9088 in the first dataset and an accuracy rate of 0.5963 in the second dataset. based on the accuracy rate achieved SVM model was superior to the other investigated methods.

In 2014 A. Keramati et al. [14] conducted a comparison employing data mining classification techniques including Decision Tree, Artificial Neural Networks, K-Nearest Neighbors, and Support Vector Machine to improve churn prediction. Their proposed hybrid methodology showed that an accuracy of above 95% for recall and precision can easily be achieved.

T. Vafeiadis et al. [16] published a paper in the year 2015 evaluating the suitability of state-of-the-art machine learning methods on the problem of churning. They compared the performance of five of the most popular state-of-the-art classification methods; multi-layer artificial neural networks, decision trees, support vector machines, naïve Bayes classifiers and logistic regression classifiers. Compared to their boosting version as well to enhance their performance. The two top performing methods in terms of corresponding testing error were a two-layer back-propagation network and decision tree classifier with an accuracy of 94% and an f-measure of 77% approximately. While in terms of investigating the impact of the application of boosting, boosted SVM (SVM-POLY) with AdaBoost achieved an accuracy of almost 97% and an f-measure of over 84%.

The authors A.ahmed and D.linen[5] in 2017 studied some of the most important machine learning techniques (Random and advanced undersampling, Gradient boosting machine and WRF, Logistic regression with Gentle AdaBoost, Improved SMOTE and AdaBoost). It was concluded that the most accurate churn prediction is obtained when utilizing hybrid methods instead of single algorithms.

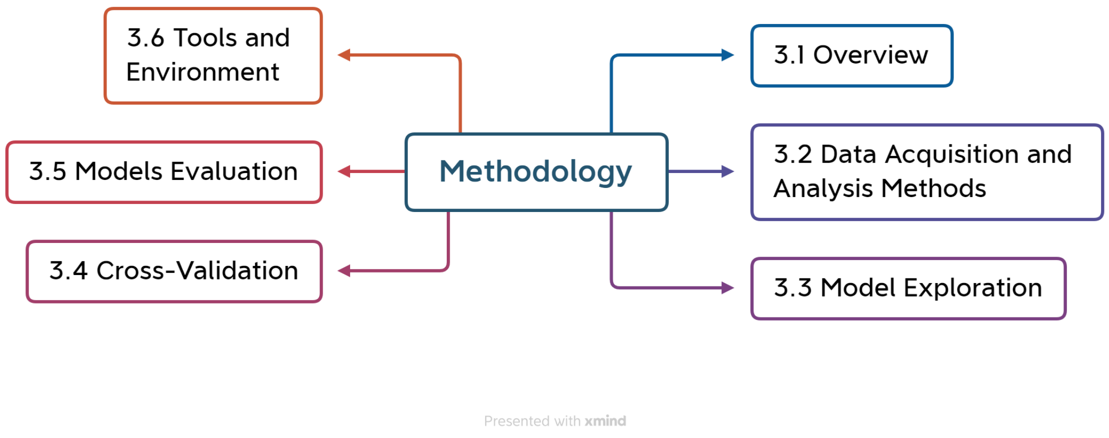
In 2018 two papers were published, [4] by Sanket Agrawal et al. who demonstrated churn prediction using a deep learning approach and achieved an 80.03% accuracy and the study [11] by Sebastiaan Höppner et al. that presented a new classifier called ProfTree that uses an evolutionary algorithm for learning profit-driven decision trees and came with the conclusion that ProfTree was the overall most profitable model in comparison with 6 other tree-based models.

The year 2020 presented us with many studies in the fields, some of which, the study by Xin Hu et al. [12] studied the prediction results and confidence of decision tree and neural network model and designed a combined prediction model of customer churn that obtained an accuracy as high as 98.87%. Also, the researchers Hemlata Jain et al.[13] studied two machine learning techniques; logistic regression and logit boost for predicting customer churn, the result showed that Logistic regression had an accuracy of 85.2385% while Logit Boost also had an accuracy 85.1785%. Another study the same year by Al-Mashraie et al. [6] investigated the determinant factors of customers' churn and built an accurate prediction model by comparing the performance measure of Four models; logistic regression, SVM(support vector machine), Decision tree and Random forest The accuracy results of all four predictive models was high. (greater than 86%). In July of the same year Alboukaey et al. [7] investigated daily churn prediction (instead of monthly) based on the daily dynamic behaviour of customers using the RFM-based model, Statistic-based model, LSTM-based model and CNN-based model, proposed models predicted churners earlier and more accurately than the monthly models.

And during the year 2022, Sulim Kim et al. [15] published their study that investigates customer churn prediction using Decision trees and their analysis showed that the maximum prediction accuracy obtained was 90% based on F-measure. While in the same year, Wael Fujo et al. [9] published their model Deep-BP-ANN which accurately outperformed the existing deep learning techniques.

Chapter 3

RESEARCH METHODOLOGY



In this chapter, we present a quick review of the proposed methodology to solve the customer churn problem in the telecommunication industry that was previously introduced in chapter one, this chapter also provides information on the data source and how this data will be analyzed, the methodologies used to gather requirements for the solution as well as the tested models.

3.1 Overview

The main purpose of this research is to help companies redirect their money and effort toward retaining customers by accurately predicting the churn rate which directly affects the company's revenue. It was proven in a study [1] that a 5% improvement in the churn rate enabled an increase in profits by 85%.
Forecasting customer churn with the help of machine learning models and data analysis is a powerful way to identify strategies and methods to lower churn and increase customer retention and recognize which features make customers churn or retain.

3.2 Data acquisition and analysis methods

3.2.1 Data source

Data gathering is one of the most crucial steps in machine learning workflows. The project's potential accuracy and usefulness are determined by the quality of the applied data. Therefore, our data was collected/gathered from a popular data science platform, namely ”Kaggle” that provides access to a wide variety of top-quality datasets and also supports dataset handling.

3.2.2 Data validation

As shown in the research assumption, we assume that the data collected from the organization's data source is valid, verified and collected properly.

3.2.3 Data Analysis and Visualization

The data will be analyzed and transformed into graphs so they show clearly how close/far the organization is from its objectives and aid the decision-making process.
To have a basic idea about the data, data will be summarized in numbers that describe the characteristics of data. Visualization is a very important step in the process, it will give a good understanding of the data, and by putting the target variable on the x-axis and different variables on the y-axis, we can easily understand the data.
in our project we used the following tools and libraries to visualize.

Plotly Express
Plotly Express is the easy-to-use, high-level interface to Plotly, which operates on a variety of types of data and produces easy-to-style figures.

1- Px.Pie:
In px.pie, data visualized by the sectors of the pie is set in values. The sector labels are set in names.A pie chart is a circular graph that is divided into segments or slices of pie (Like a Pizza). It is used to represent the percentage or proportional data where each slice of the pie represents a category. This is very similar to the Bar chart but in a circular fashion. We can use pie chart when there is no order defined in the labels, if there were too many labels then we will use Bar charts. We will use Pie Charts for attribute balance checking, and categorical data proportion.

2- Px.Bar
With px.bar, each row of the Data Frame is represented as a rectangular mark. To aggregate multiple data points into the same rectangular mark, the bar plot can be customized using keyword arguments.

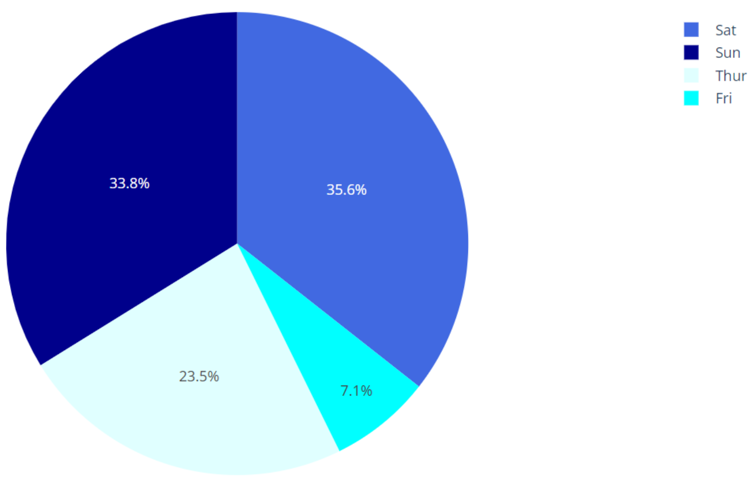


Figure 3.1: Plotly Express Pie Chart

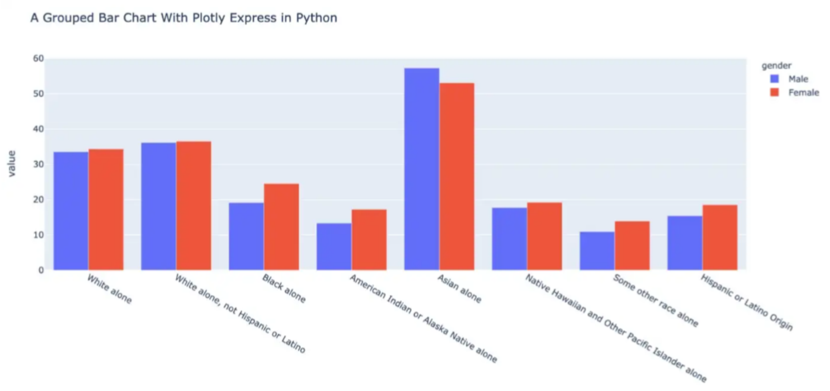


Figure 3.2: Plotly Express bar Chart

3- Px.histogram

A histogram is a representation of the distribution of numerical data, where the data are binned and the count for each bin is represented. More generally, in Plotly a histogram is an aggregated bar chart, with several possible aggregation functions (e.g. sum, average, count...).

3.3 Model Exploration

3.3.1 Data Cleaning and Preparation

We will start investigating the data by exploring and removing any missing values, checking if any column adjustments are needed and whether each column has the correct corresponding data type.

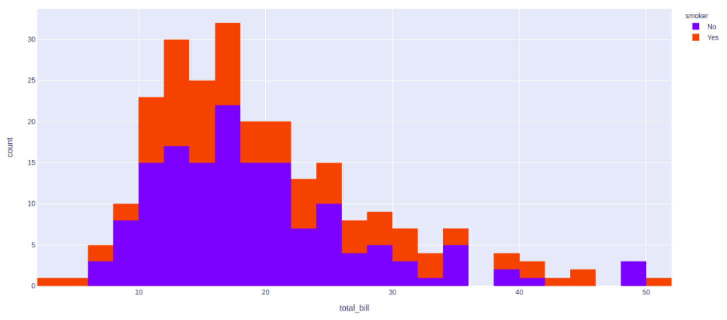


Figure 3.3: Plotly Express histogram Chart

3.3.2 Exploratory Data Analysis (EDA)

To better understand the patterns in the data and potentially form some hypothesis of the features or the input variable behaviour. First, we will look at the distribution of individual variables and then slice and dice our data for any interesting trends which help us on building a model or model selections e.g. whether it is regression or classification.

3.3.3 Data Preprocessing

Here we will convert all the categorical variables into numeric variables (Encoding), and explore the relationship between the features. Also, we will apply some feature engineering based on which collection of features increases the score validation most, we will also use the generalized linear model (GLM) to gain more insight into which features affect our target variable (churn) the most and in which way. lastly, we will apply feature scaling before feeding them to the models.
We will use this data later for building the prediction models but for exploring the data (EDA) we will use the original data frame.

3.3.4 Model Building

After going through the above EDA and preprocessing we will develop some predictive models and compare them, we will then build the models based on the recommended models from previous works discussed earlier, and if we did not achieve a satisfying score based on the proposed evaluation metric then we will return to the feature engineering phase to collect more or fewer features.
The recommended models based on the related work defined in the previous chapter and experts consulting are:

1. Decision tree
2. Support Vector Machine
3. Logistic Regression

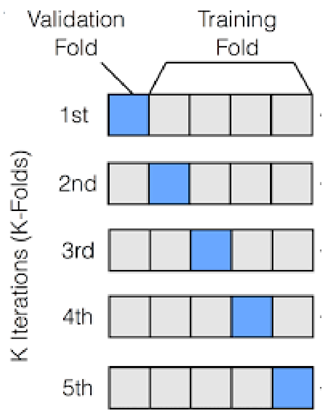


Figure 3.4: k-fold by k = 5

4. Gradient Boosting Classifier
5. CatBoost

3.4 Cross-Validation

Cross-Validation is a statistical technique or re-sampling technique used to assess the effectiveness of machine learning models on a small data sample. For the purpose of improving machine learning models, we decided to apply one of the most commonly used variations of cross-validation.

3.4.1 Stratified K-Fold cross-validation

The Stratified K-Fold cross-validation technique is a potent tool. It is a technique for separating data into train and test sets while taking the data's class imbalance into consideration. The data is divided into k-folds, which can be implemented with k greater than 1 folds.

K-Fold Cross-Validation is also known as k-cross, k-fold cross-validation, k-fold CV, and k-folds, and the model is trained on (k - 1) folds while being tested on the last fold. Each fold acts as the test set once throughout this process, which is repeated k times. The Stratified K-Fold approach is very helpful for datasets that are imbalanced, in which the proportions of the classes in the train and test sets differ noticeably. We can prevent over-fitting by stratifying the folds to make sure that the class proportions are comparable between the train and test sets. In particular, when the target variable is imbalanced, this method makes sure that one class of data is not over-represented.

3.5 Models Evaluation

After training the model we have to evaluate it to understand if it needs to be improved and how well it is going to perform when applied to the real world. The evaluation metrics we plan to use on our customer churn prediction problem are:

3.5.1 Area Under the Curve k-fold cross validation

The Receiver Operating Characteristic (ROC) curve is summarized by the Area Under the Curve (AUC), which measures a binary classifier's ability to discriminate between classes. The higher the AUC the better the model performs at differentiating between the positive and negative classes.

The classifier can accurately differentiate between all of the positive and negative class points when AUC = 1. However, if the AUC had been 0, the classifier would have predicted that every negative would be a positive and every positive would be a negative.

There is a high likelihood that the classifier will be able to separate the positive class values from the negative ones when the AUC is smaller than 1 but greater than 0.5. This is the case because more True positives and True negatives can be detected by the classifier than False positives and False negatives.

The classifier cannot distinguish between Positive and Negative class points when AUC = 0.5, i.e., the classifier predicts either a random class or a fixed class for each data point. Therefore, a classifier's ability to distinguish between positive and negative classes is improved by a higher AUC value.

3.5.2 Precision and Recall

These are classification metrics that are mainly defined on:

- True Positives (TP): Number of outcomes that are positive and are predicted positive
- True Negatives (TF): Number of outcomes that are negative and are predicted negative
- False Positives (FP): Number of outcomes that are negative but predicted positive, which is a Type 1 error
- False Negatives (FN): Number of outcomes that are positive but predicted negative, which is a Type 2 Error

Precision is the number of correctly labelled positive instances out of all positive labelled instances.

$$precision = \frac{TP}{TP + FP}$$

While Recall is the number of correctly labelled positive instances out of all positive instances.

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

3.5.3 F1-score or F-measure

It is a combination of precision and recall and it outputs only one number, the higher the F1 score the better the performance of our model. However, the classifier will only get a high F-score if both precision and recall are high. The f1-Score ranges between [0,1].

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * Precision * Recall}{Precision + Recall}$$

This Score relays true performance on both balanced and imbalanced datasets because it takes into account both the precision and recall ability of the model.

3.6 Tools and Environment

3.6.1 Google Colab

Colaboratory, or ”Colab” for short, is a Google Research product. Colab allows anyone to write and run arbitrary Python code in the browser, making it ideal for machine learning, data analysis, and education. Colab is a hosted Jupyter notebook service that requires no setup to use.

3.6.2 Python

Python is a highly interpreted, object-oriented, high-level programming language. It is also famous for its dynamic semantics. especially because of dynamic binding and typing. A great choice of libraries is one of the main reasons Python is the most popular programming language used for Machine Learning.

A library is a module or a group of modules published by different sources like PyPi, which include a Pre-written piece of code that allows users to reach some functionality or perform different actions. Python libraries provide base-level items so developers do not have to code them from the very beginning every time. Machine Learning requires continuous data processing, and Python’s libraries let you access, handle and transform data. The following libraries were used in the models development in our research:

- Scikit-Learn
Also known as sklearn, is a free Python machine learning package. we will use it in handling Machine learning algorithms for classification, logistic regression and others.

- Pandas
It is a high-performance open-source package for the Python programming language. we plan on using it in importing data and in our data analysis.

- NumPy
It is a versatile array package. It comes with a high-performance multidimensional array object as well as tools for manipulating it, we will use it to perform a wide variety of mathematical operations on arrays.

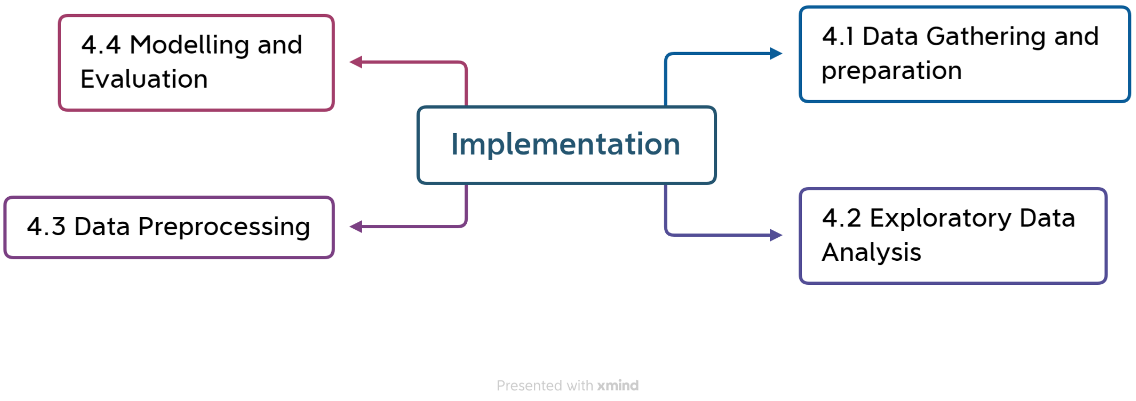
- Matplotlib
Matplotlib is a Python package that allows you to create static, animated, and interactive visualization. Data visualization in our research is commonly done with plots, graphs, and charts, and Matplotlib is a library for doing this.

- Statsmodels
We will use statsmodel, which is a Python module that offers classes and functions, in estimating a wide range of statistical models, running statistical tests, and exploring statistical data.

- Seaborn
Seaborn is a Python data visualization library based on matplotlib. We prefer to visualize the comparison between features and models using seaborn for that it provides more flexible high-level interface.

Chapter 4

IMPLEMENTATION

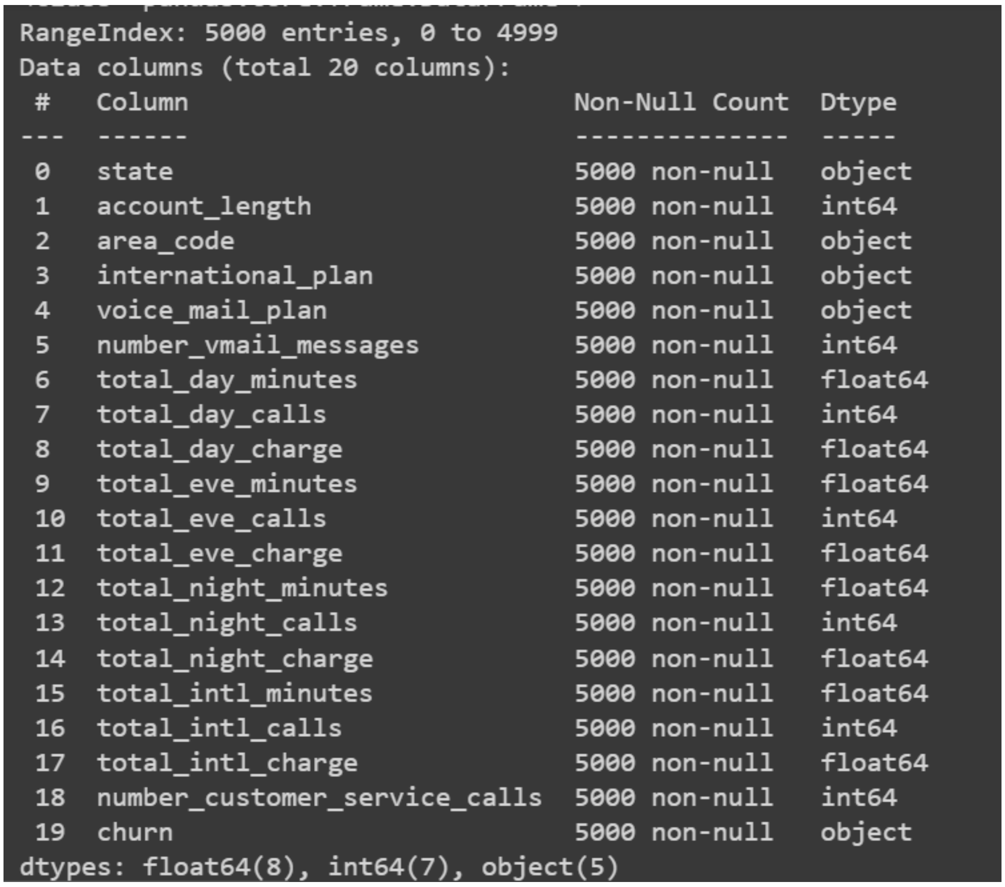

4.4 Modelling and Evaluation
4.1 Data Gathering and preparation
Implementation
4.3 Data Preprocessing
4.2 Exploratory Data Analysis
Presented with xmind

In this Chapter we will provide a detailed explanation of the technical part of our research that includes an overview of the data used and it's analysis, along with the implementation of the models.

4.1 Data gathering and Preparation

In this research we are working on Telecom Dataset downloaded from kaggle [1], The dataset was uploaded as a Pandas dataframe. The dataset contains 5000 observations and 20 columns, 19 independent features and 1 target Boolean variable "churn" which indicates the class of the sample. After importing the required libraries which are numpy, pandas, scikit-learn, statsmodels, seaborn ,matplotlib and catboost and the dataset we now arrive at the step of cleaning and preparing the data.

After inspecting the data it was seen that the column names are in place and columns had the correct corresponding data type so no adjustments were needed here, most of the categorical variables had binary classes ('Yes' or 'No'), while only a few had multinomial classes. It was also shown that there are no missing values (NaN) since the count of all columns is 5000 which is the total number of entries, also we can see that we are dealing with two other data types float and int.



```
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 20 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   state                          5000 non-null   object
 1   account_length                 5000 non-null   int64
 2   area_code                      5000 non-null   object
 3   international_plan             5000 non-null   object
 4   voice_mail_plan                5000 non-null   object
 5   number_vmail_messages          5000 non-null   int64
 6   total_day_minutes              5000 non-null   float64
 7   total_day_calls                5000 non-null   int64
 8   total_day_charge               5000 non-null   float64
 9   total_eve_minutes              5000 non-null   float64
 10  total_eve_calls                5000 non-null   int64
 11  total_eve_charge               5000 non-null   float64
 12  total_night_minutes            5000 non-null   float64
 13  total_night_calls              5000 non-null   int64
 14  total_night_charge             5000 non-null   float64
 15  total_intl_minutes             5000 non-null   float64
 16  total_intl_calls               5000 non-null   int64
 17  total_intl_charge              5000 non-null   float64
 18  number_customer_service_calls  5000 non-null   int64
 19  churn                          5000 non-null   object
dtypes: float64(8), int64(7), object(5)
```

Figure 4.1: high-level summary of the dataset

After creating a function that displays the unique values in the dataset, an output in the form of the table above was obtained. and from the table we concluded that the dataset contains 4 categorical features, 15 numerical features And the prediction feature 'churn'.
These features can be divided into:

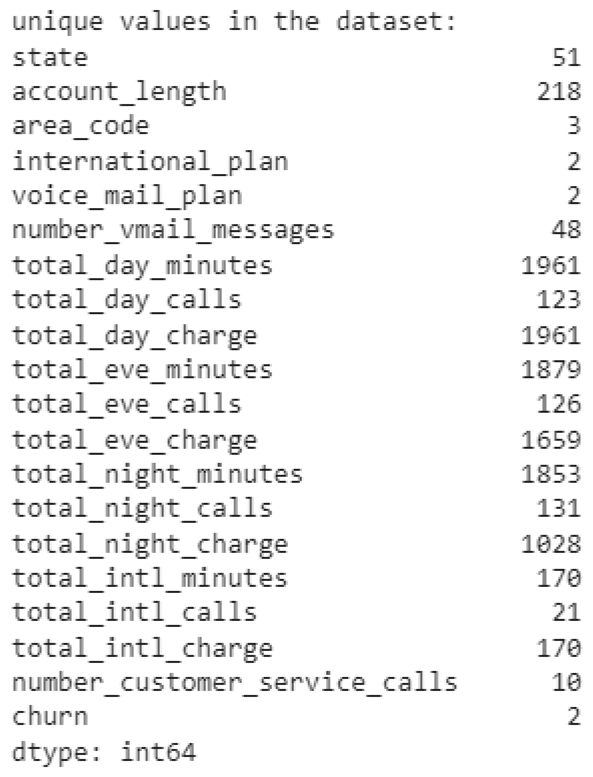
1. Demographic customer information:
   State and Area Code

2. Services that each customer has signed up for:
   International Plan and Voice Mail Plan

3. Customer account information:
   account length, number of voicemail messages, total minutes, charges and calls (per day, night, evening and internationally) along with the total number of customer service calls.

4.2 Exploratory Data Analysis

Here we performed initial investigations on the data to discover patterns and spot anomalies and try to gather as many insights from it with the help of summary statistics and graphical representations.



```
unique values in the dataset:
state                              51
account_length                    218
area_code                           3
international_plan                  2
voice_mail_plan                     2
number_vmail_messages              48
total_day_minutes                1961
total_day_calls                   123
total_day_charge                 1961
total_eve_minutes                1879
total_eve_calls                   126
total_eve_charge                 1659
total_night_minutes              1853
total_night_calls                 131
total_night_charge               1028
total_intl_minutes                170
total_intl_calls                   21
total_intl_charge                 170
number_customer_service_calls      10
churn                               2
dtype: int64
```

Figure 4.2: Unique Values in the dataset

In order to make sense of the data we started by having a look at the distribution of our target variable to see how it spreads.

The distribution graph above, shows the percentage of the churners and non-churners in our dataset, it also shows that this is a binary classification problem with an imbalanced target:
Churn – No = 82.3%
Churn – Yes = 17.7%

After that we investigated the relationship of each individual categorical feature with the target variable to get a sense of which features might be important.

In Figure 4.4 , we can see that there is a variation in the distribution of churn in states, the state with the highest churn rate (count = 31) is NJ state and there is a higher proportion of churners among customers with area code 415 in Figure 4.5 while churners of the remaining two areas have approximately close percentages, which implies that these two features may not have a huge impact on the target variable since the variation of the distribution among their values was not high.

Also, in Figure 4.6 and Figure 4.7 customers with no international plan and no voicemail plan have higher proportion of churners than users with subscriptions in those plans, The huge variation between customers with subscriptions and those without is an indicator that these two features may be among the most optimal features to train our models with.

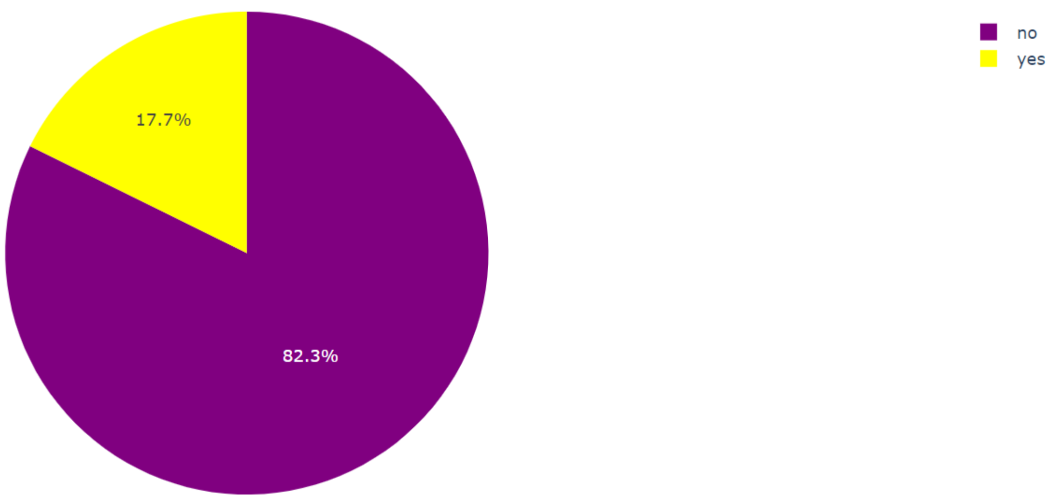


Figure 4.3: Distribution of Churn

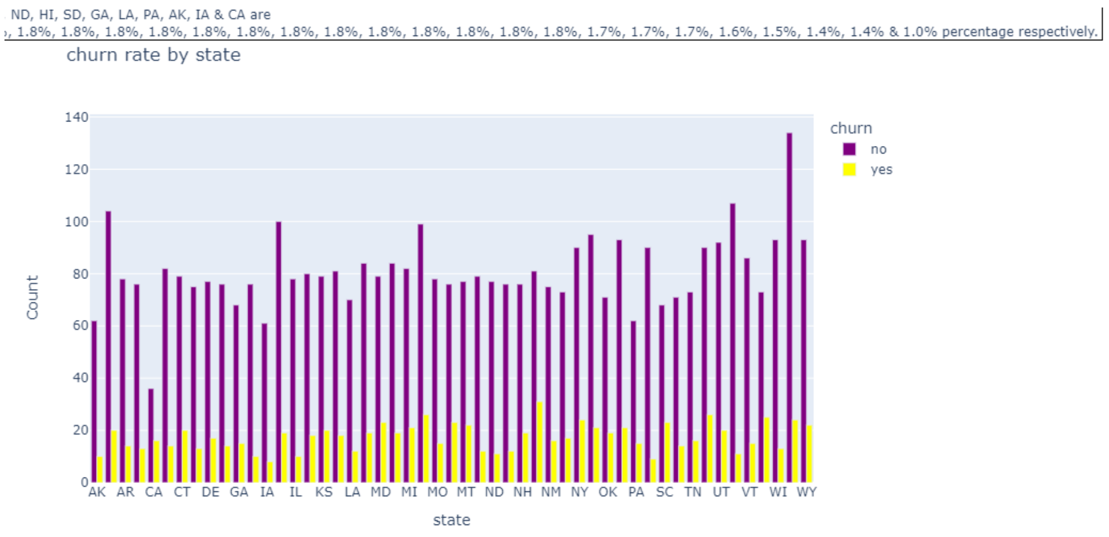


Figure 4.4: Churn rate by state

After investigating and analysing the categorical features we approached the numerical features in the same manner and it resulted in the following graphs in Figure 4.8. We noticed that most of the features were normally distributed but there were few features had data skewed to the right.

Real-world datasets often contain features that are varying in degrees of magnitude, range and units. Based on our assumption that the dataset in this study satisfies our need to derive a solution to local real-data and in order for machine learning models to interpret these real-features on the same scale, we need to perform feature scaling.

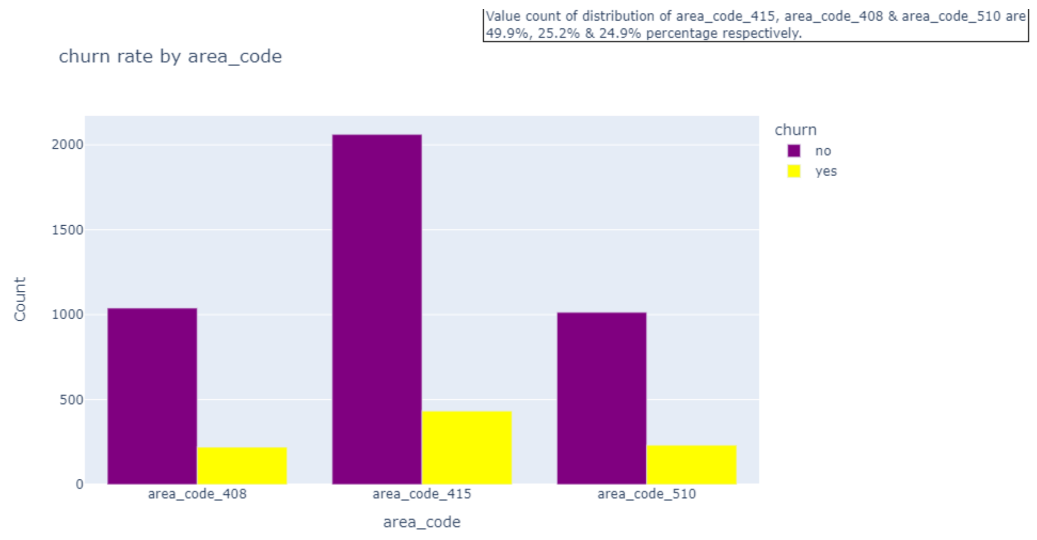


Figure 4.5: Churn rate by Area code

The machine learning algorithms that do not require feature scaling is mostly non-linear ML algorithms such as Decision Trees, Gradient Boosting Classifier and AdaBoost. while logistic Regression are Support Vector Machine are linear Machine Learning models.

Algorithms like Decision Trees, gradient Boosting, etc, are not significantly affected by feature scaling since the trees in these algorithms are constructed based on conditions and are not dependent on the range of values. Therefore, we decided to apply feature scaling on numeric data as a preprocessing step.

In regression based models, such as Logistic Regression, the presence of skewed features degrades the model's ability to describe typical cases as it has to deal with rare cases on extreme values. Therefore, in order to ensure that the model's capabilities are not affected, skewed data had to be transformed to approximate a normal distribution. The method we applied was Min-max normalization which will be explained in more details later in this chapter.

4.3 Data Preprocessing

4.3.1 Transforming the data

Since machine learning models can only work with numerical values, we used label encoding on the independent columns of type Object. Label encoding is the process of transforming labels into a numeric form so that they can be read by machines.

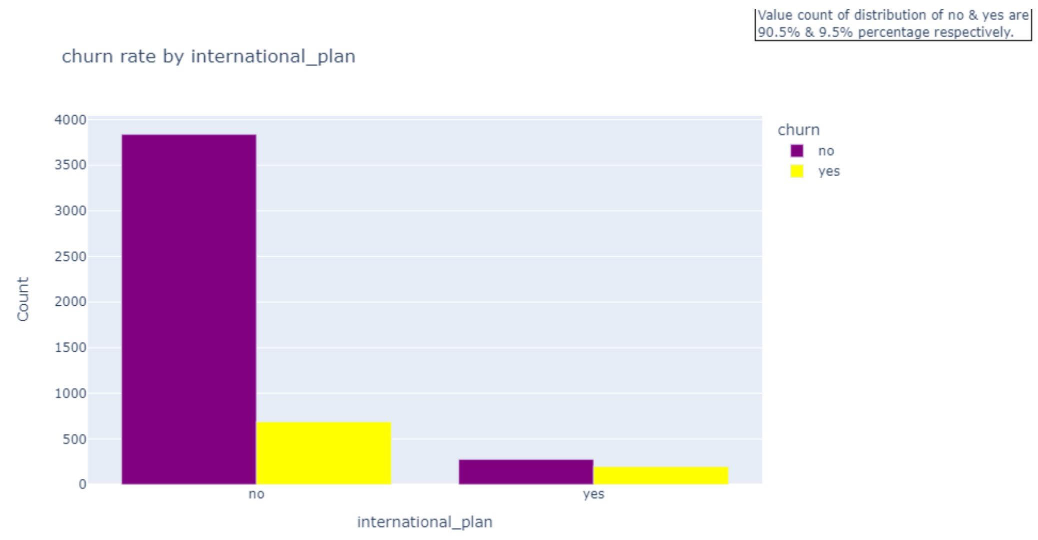


Figure 4.6: Churn rate by international plan service

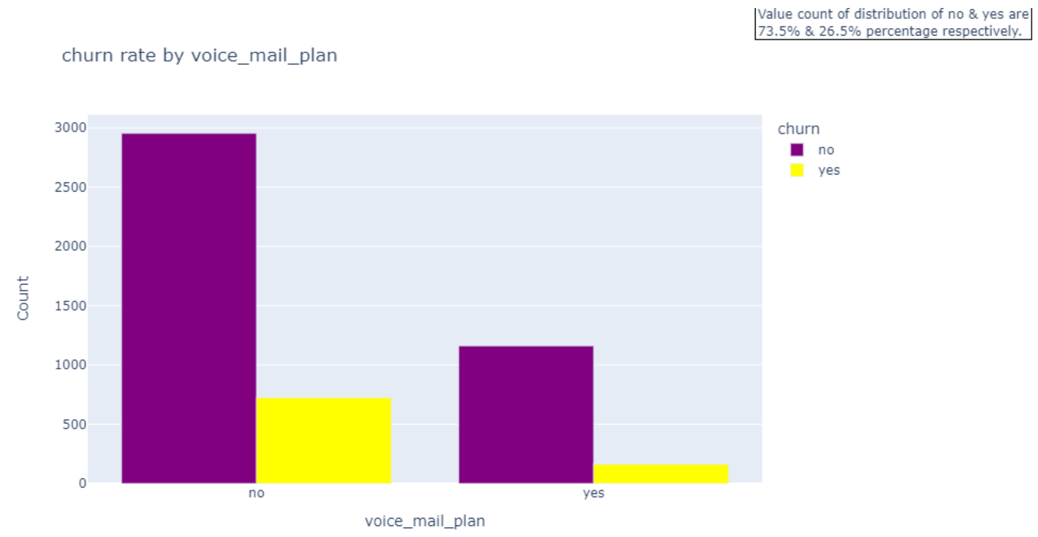


Figure 4.7: Churn rate by voicemail plan service

The operation of those labels can then be better determined by machine learning algorithms. It is a significant supervised learning preprocessing step for the structured dataset. For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones. As we mentioned in the previous section we applied feature scaling to the regression based model which is a method used to normalize the range of independent variables or features of data. We used Min-max scaler to transform the data so that each feature is in the same range between 0 and 1.

4.3.2 Correlation between features

Before feeding the Data to the model, it is desirable to reduce the number of features to reduce the computational cost and improve the performance of the model because Some models perform worse when they use variables that are highly correlated with others because the model cannot

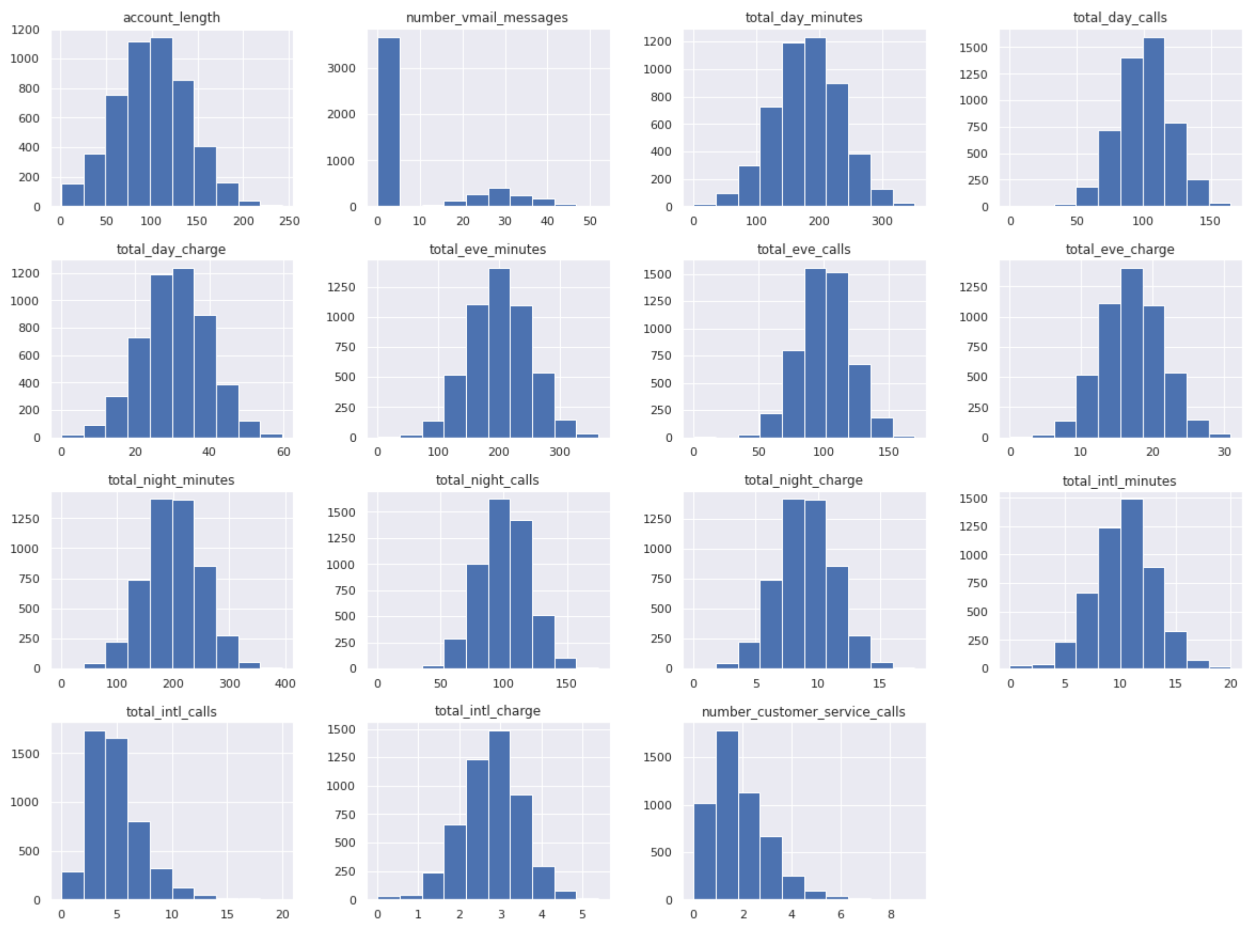


Figure 4.8: Numerical Features Visualization

distinguish the differences between these two variables when measuring variations in the output variable.

Correlation measures the linear relationship between two variables. Features with high correlation are more linearly dependent and have almost the same effect on the dependent variable. Therefore, we can drop one of them.

If the value is +1 or close to it, then we say the variables are positively correlated. and they vary in the same direction simultaneously.
If the value is -1 or close to it, then we say the variables are negatively correlated. They also vary in the opposite direction at the same time.
If the value is 0 or close to it, then we say the variables are not correlated.

Based on the above we evaluated the linear relationship between each feature and the target variable by plotting a correlation matrix Figure 4.9 and selected those features that had the strongest relationship with the target variable.

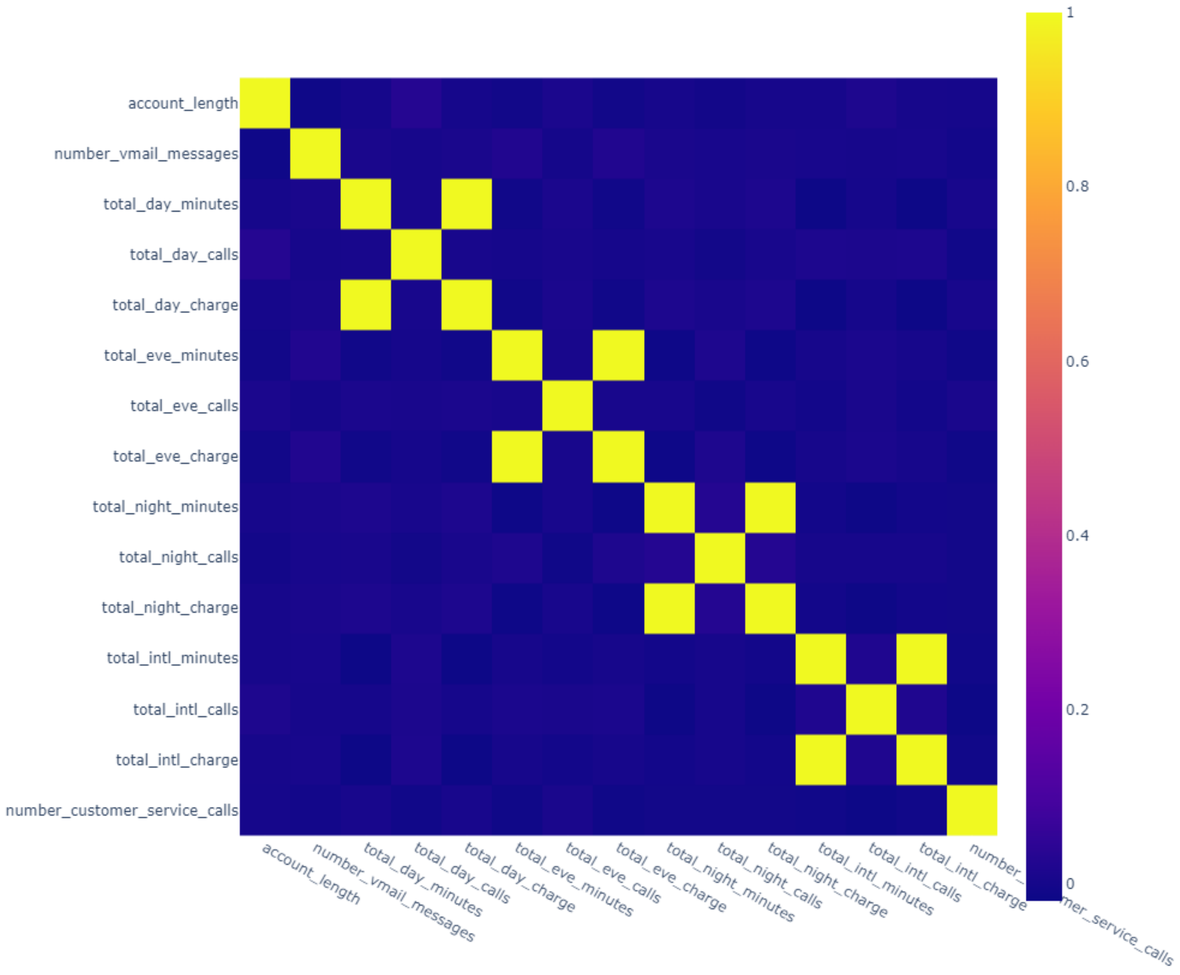


Figure 4.9: Correlation Matrix

From the heatmap in Figure 4.9 for example, We can see that 'total day minutes' and 'total day charges' have a strong positive correlation of 1, and the same with 'total night minutes' and 'total night charges', 'total international minutes' and 'total international charges', while 'Voicemail plan' and 'number of voicemail messages' have a strong positive correlation of 0.95.

hence we drop the following features:

- Total day minutes
- Total evening minutes
- Total night minutes
- Total international minutes
- Voicemail Plan

Also, We can see that the churn rate has the highest correlation with international plan, followed by number of customer service calls.

4.3.3 Feature Engineering

Feature Engineering is the process of transforming data to increase the predictive performance of machine learning models. Feature engineering basically means that you deduce some hidden insights from the crude data, and make some meaningful features out of it.

Statistics aggregation is one of the most powerful ways to create new features from existing ones based on which group of features increases the score validation the most when applied to the K-fold cross-validation. We created new features by grouping them into two categories 'charges' and 'evening activity' and Calculated the descriptive statistics Minimum, Maximum, Standard deviation along with the Ratio, Range and Sum for each group of features.

4.3.4 Generalized Linear Model

As we mentioned before Churn Prediction is a classification problem in either customers churn or retain after a given period of time, so to guide us in building our models We used the generalized linear model (GLM) to gain some insights into the features by answering the following questions:

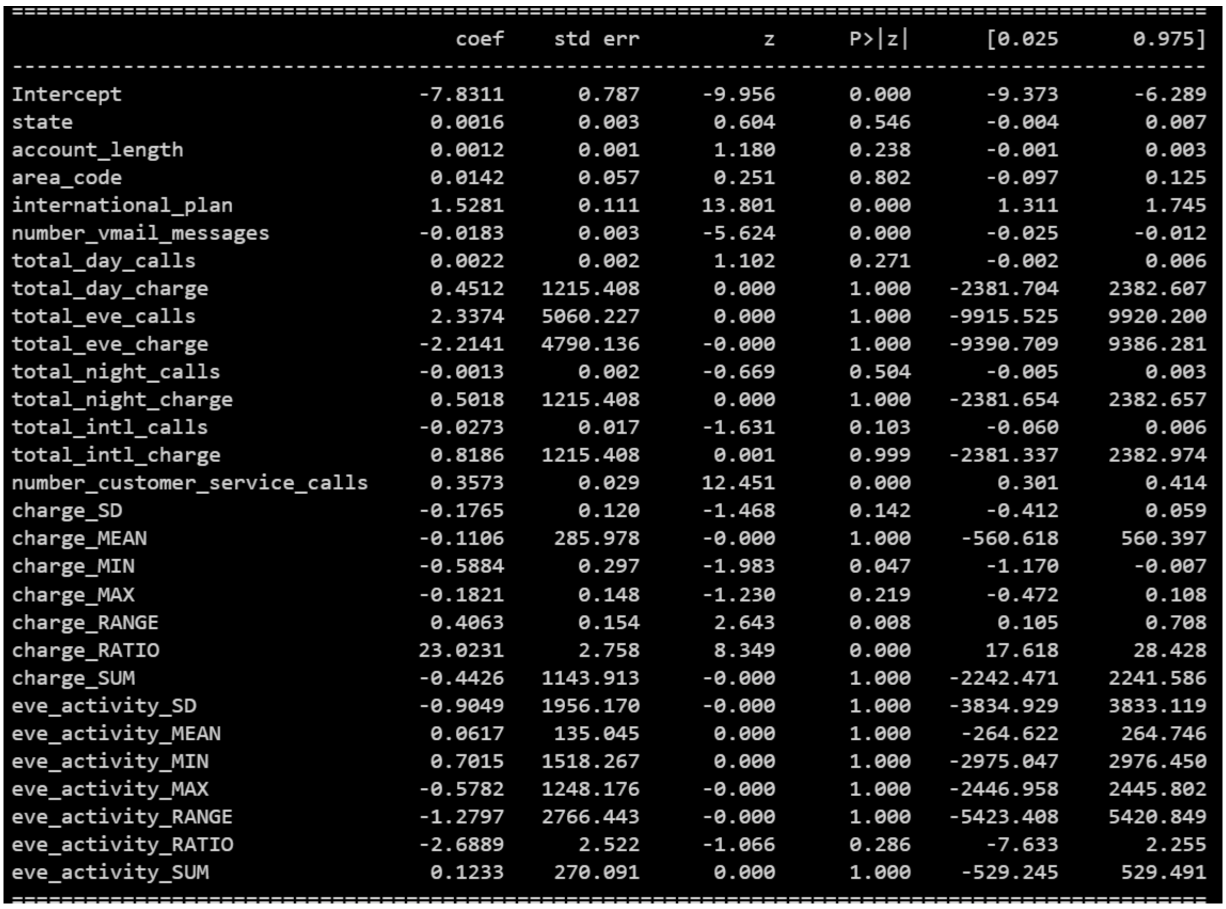
1. Which features make customers churn or retain?
2. What are the most features to train a model with high performance?

To answer the first question we look at the $(P > |z|)$ column in Figure 4.10, if the absolute p-value is smaller than 0.05, it means that the feature affects Churn in a statistically significant way and those features are:
international plan, number of voicemail messages, Charge Minimum, Charge Range, Charge Ratio and number of customer service calls.

The answer to the second question is found by looking at the exponential coefficient values in Figure 4.11 which estimate the expected change in churn through a given feature by a change of one unit. Values more than 1 indicate increased churn while Values less than 1 indicate that churn is happening less, we notice that all the float values of all features greater than 1.

4.4 Modelling and Evaluation

On the previous chapter we defined the evaluation metrics that were chosen to be employed on the models, in this chapter we will provide a thorough explanation for those metrics along with the proposed models implemented in this research.



```
==============================================================================================
                                  coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Intercept                      -7.8311      0.787     -9.956      0.000      -9.373      -6.289
state                           0.0016      0.003      0.604      0.546      -0.004       0.007
account_length                  0.0012      0.001      1.180      0.238      -0.001       0.003
area_code                       0.0142      0.057      0.251      0.802      -0.097       0.125
international_plan              1.5281      0.111     13.801      0.000       1.311       1.745
number_vmail_messages          -0.0183      0.003     -5.624      0.000      -0.025      -0.012
total_day_calls                 0.0022      0.002      1.102      0.271      -0.002       0.006
total_day_charge                0.4512   1215.408      0.000      1.000   -2381.704    2382.607
total_eve_calls                 2.3374   5060.227      0.000      1.000   -9915.525    9920.200
total_eve_charge               -2.2141   4790.136     -0.000      1.000   -9390.709    9386.281
total_night_calls              -0.0013      0.002     -0.669      0.504      -0.005       0.003
total_night_charge              0.5018   1215.408      0.000      1.000   -2381.654    2382.657
total_intl_calls               -0.0273      0.017     -1.631      0.103      -0.060       0.006
total_intl_charge               0.8186   1215.408      0.001      0.999   -2381.337    2382.974
number_customer_service_calls   0.3573      0.029     12.451      0.000       0.301       0.414
charge_SD                      -0.1765      0.120     -1.468      0.142      -0.412       0.059
charge_MEAN                    -0.1106    285.978     -0.000      1.000    -560.618     560.397
charge_MIN                     -0.5884      0.297     -1.983      0.047      -1.170      -0.007
charge_MAX                     -0.1821      0.148     -1.230      0.219      -0.472       0.108
charge_RANGE                    0.4063      0.154      2.643      0.008       0.105       0.708
charge_RATIO                   23.0231      2.758      8.349      0.000      17.618      28.428
charge_SUM                     -0.4426   1143.913     -0.000      1.000   -2242.471    2241.586
eve_activity_SD                -0.9049   1956.170     -0.000      1.000   -3834.929    3833.119
eve_activity_MEAN               0.0617    135.045      0.000      1.000    -264.622     264.746
eve_activity_MIN                0.7015   1518.267      0.000      1.000   -2975.047    2976.450
eve_activity_MAX               -0.5782   1248.176     -0.000      1.000   -2446.958    2445.802
eve_activity_RANGE             -1.2797   2766.443     -0.000      1.000   -5423.408    5420.849
eve_activity_RATIO             -2.6889      2.522     -1.066      0.286      -7.633       2.255
eve_activity_SUM                0.1233    270.091      0.000      1.000    -529.245     529.491
==============================================================================================
```

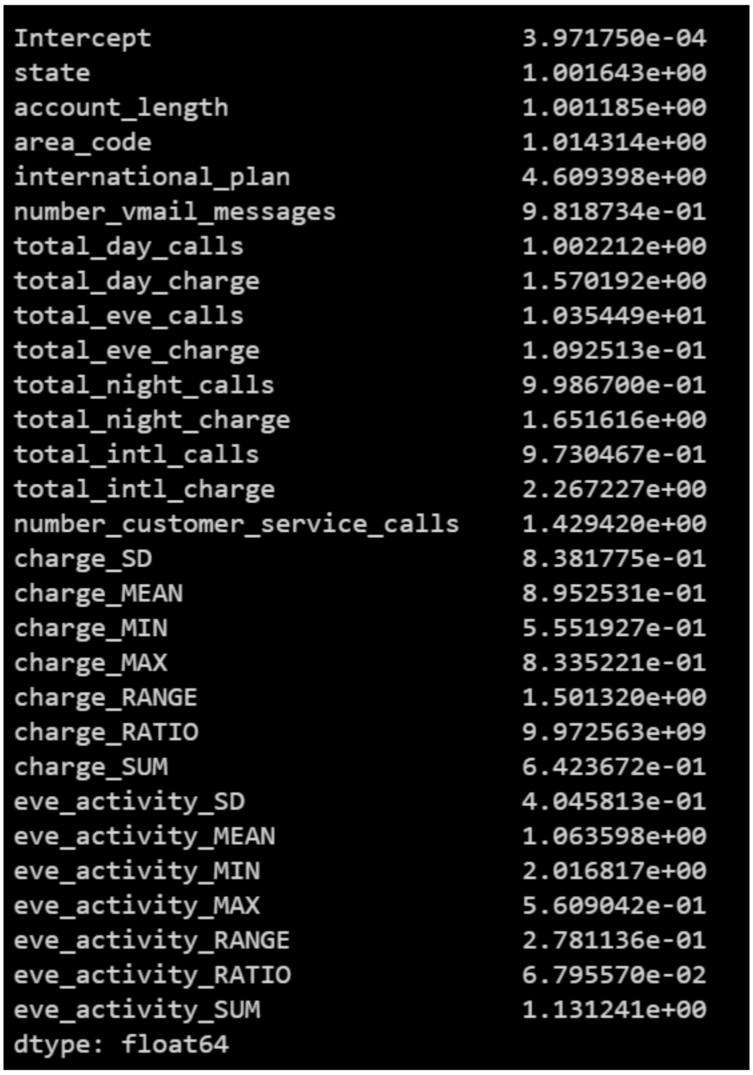
Figure 4.10: Generalized Linear Model

Accuracy of the model is the most common evaluation metric when comparing machine learning models, However, it is not recommended when dealing with imbalanced data (when one class occurs significantly more abundant than the other). Simply put, If the churn rate is just 17.7%, then a naive classifier labeling everything as ”not churn” has 84% accuracy and can increase by definition. This sounds good on paper but is bad in practice. This problem can be avoided by choosing a more suitable metric such as AUC, precision at fixed recall, recall at fixed precision, and the F1-score for all models in that they are AUC metric is less sensitive to the data structure.

We Chose to apply the mean of the AUC score before and after applying feature engineering in order to compare the impact of feature engineering on the AUC when dealing with imbalanced class problem. The AUC helps to visualize how well the machine learning classifier is performing in comparison to other classifiers. A perfect classifier will have an AUC equal to 1, whereas a purely random classifier will have an AUC equal to 0.5, Having selected the performance metrics, we were truly ready to start modelling our machine learning algorithms using different libraries.

4.4.1 Models Evaluation before and after Feature Engineering

Model evaluation involves using the available dataset to fit a model and estimate its performance when making predictions on unseen examples.



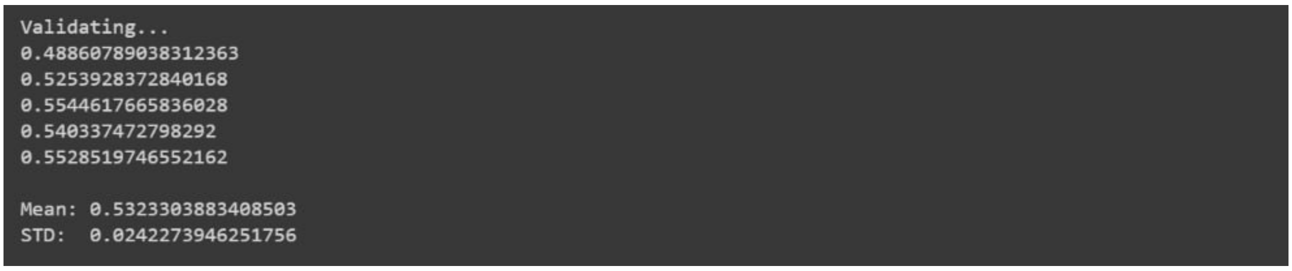
```
Intercept                         3.971750e-04
state                             1.001643e+00
account_length                    1.001185e+00
area_code                         1.014314e+00
international_plan                4.609398e+00
number_vmail_messages             9.818734e-01
total_day_calls                   1.002212e+00
total_day_charge                  1.570192e+00
total_eve_calls                   1.035449e+01
total_eve_charge                  1.092513e-01
total_night_calls                 9.986700e-01
total_night_charge                1.651616e+00
total_intl_calls                  9.730467e-01
total_intl_charge                 2.267227e+00
number_customer_service_calls     1.429420e+00
charge_SD                         8.381775e-01
charge_MEAN                       8.952531e-01
charge_MIN                        5.551927e-01
charge_MAX                        8.335221e-01
charge_RANGE                      1.501320e+00
charge_RATIO                      9.972563e+09
charge_SUM                        6.423672e-01
eve_activity_SD                   4.045813e-01
eve_activity_MEAN                 1.063598e+00
eve_activity_MIN                  2.016817e+00
eve_activity_MAX                  5.609042e-01
eve_activity_RANGE                2.781136e-01
eve_activity_RATIO                6.795570e-02
eve_activity_SUM                  1.131241e+00
dtype: float64
```

Figure 4.11: Exponential coefficient values

It is a challenging problem as both the training dataset used to fit the model and the test set used to evaluate it must be sufficiently large and representative of the underlying problem so that the resulting estimate of model performance is not too optimistic or pessimistic.

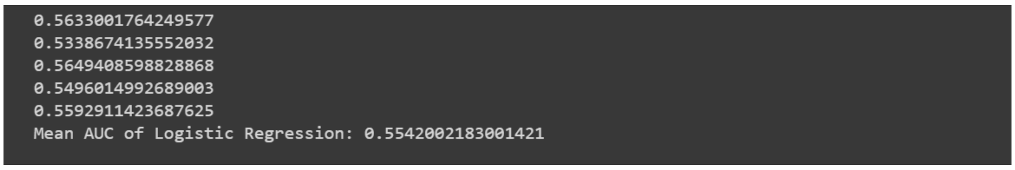
The two most common approaches used for model evaluation are the train/test split and the k-fold cross-validation procedure. Both approaches can be very effective in general, although they can result in misleading results and potentially fail when used on classification problems with a severe class imbalance. Instead, the techniques must be modified to stratify the sampling by the class label, called stratified k-fold cross-validation.

After using the stratified k-fold cross-validation to divide our main data into a representative 5 folds we will proceed with evaluating the model’s performance on our imbalanced Dataset.



```
Validating...
0.48860789038312363
0.5253928372840168
0.5544617665836028
0.540337472798292
0.5528519746552162

Mean: 0.5323303883408503
STD:  0.0242273946251756
```

Figure 4.12: Mean AUC of Logistic Regression before Feature Engineering



```
0.5633001764249577
0.5338674135552032
0.5649408598828868
0.5496014992689003
0.5592911423687625
Mean AUC of Logistic Regression: 0.5542002183001421
```

Figure 4.13: Mean AUC of Logistic Regression after feature Engineering

1- logistic regression

From Figure 4.12 the output of evaluating the model shows that the mean AUC of Logistic regression was 53% before applying feature engineering to the dataset and achieved a slightly higher mean AUC of 55% after feature engineering as shown in figure 4.13.

2- Decision Tree

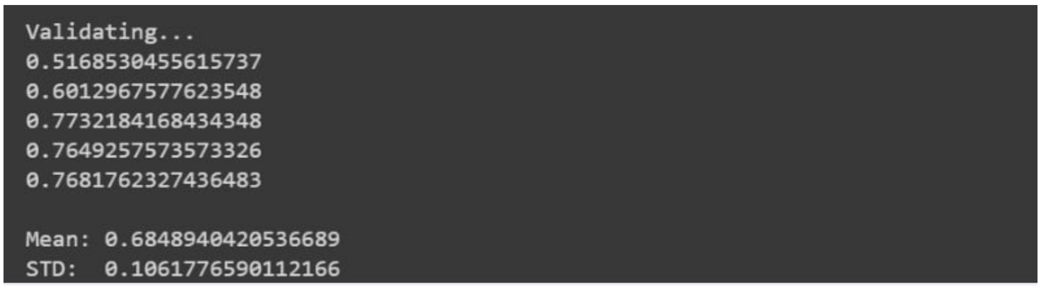
The Decision tree model scored a mean AUC of 68% as shown in Figure 4.14 before feature engineering the data and scored a slightly higher mean of 75% after the feature engineering step as seen in Figure 4.15.

3- Support Vector Machine

The output of Figure 4.16 shows that the SVM model achieved a mean AUC of 55% after using Stratified k-fold cross Validation and a mean AUC of of 50% after applying feature engineering as shown in Figure 4.17.

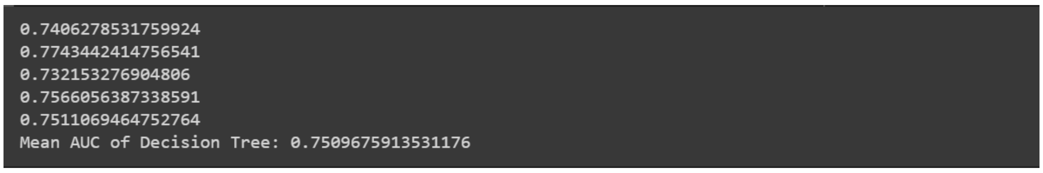
4- Gradient Boost

As Shown in the output of Figure 4.18 and Figure 4.19 the gradient boost model achieved a mean AUC of 71% and then it increased to 79% after features engineering.



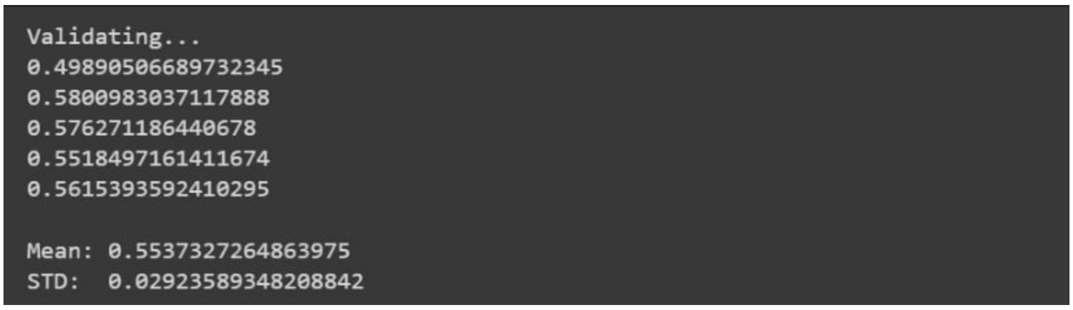
```
Validating...
0.5168530455615737
0.6012967577623548
0.7732184168434348
0.7649257573573326
0.7681762327436483

Mean: 0.6848940420536689
STD:  0.1061776590112166
```

Figure 4.14: Mean AUC of Decision Tree before Feature Engineering



```
0.7406278531759924
0.7743442414756541
0.732153276904806
0.7566056387338591
0.7511069464752764
Mean AUC of Decision Tree: 0.7509675913531176
```

Figure 4.15: Mean AUC of Decision Tree after Feature Engineering

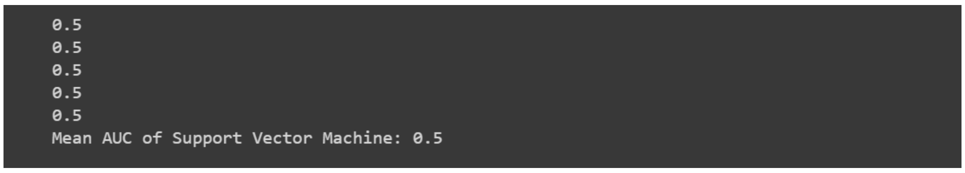


```
Validating...
0.49890506689732345
0.5800983037117888
0.576271186440678
0.5518497161411674
0.5615393592410295

Mean: 0.5537327264863975
STD:  0.02923589348208842
```

Figure 4.16: Mean AUC of SVM before Feature Engineering

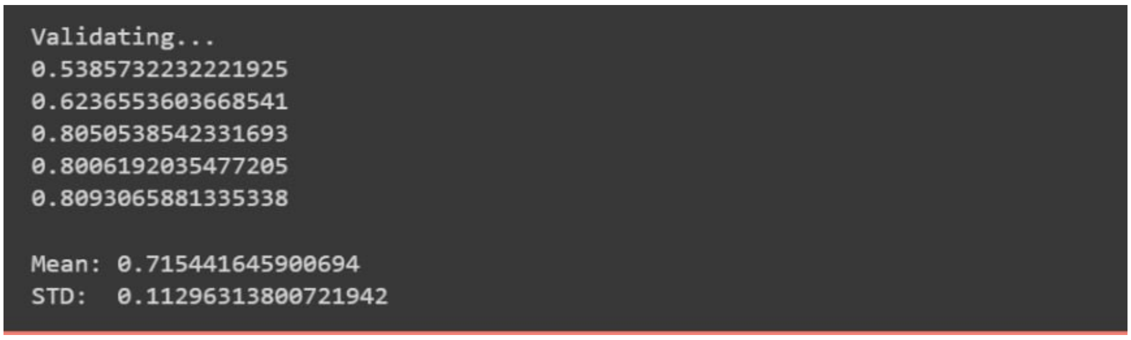
5- CatBoost

The CatBoost model with generic hyperparameters configuration achieved a mean AUC of 73% that highly increased to 79% after feature engineering.



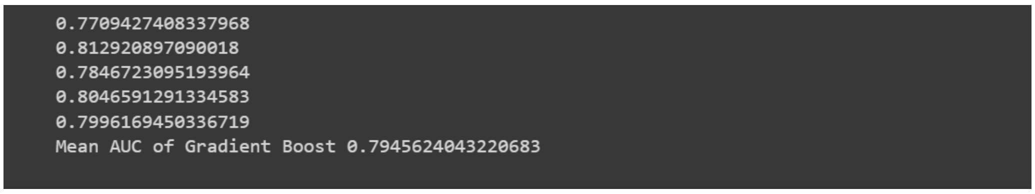
```
0.5
0.5
0.5
0.5
0.5
Mean AUC of Support Vector Machine: 0.5
```

Figure 4.17: Mean AUC of SVM after Feature Engineering



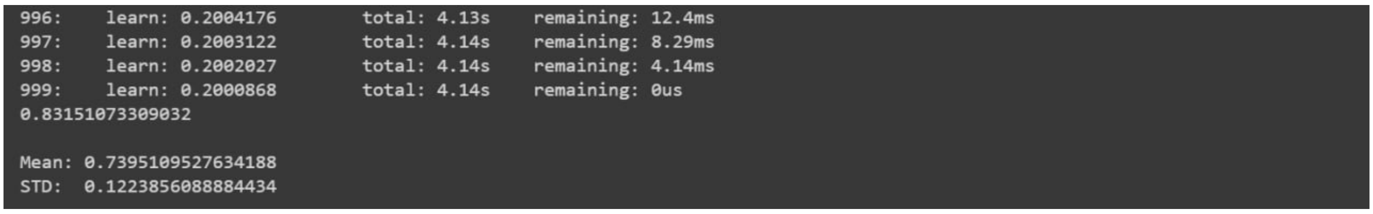
```
Validating...
0.5385732232221925
0.6236553603668541
0.8050538542331693
0.8006192035477205
0.8093065881335338

Mean: 0.715441645900694
STD:  0.11296313800721942
```

Figure 4.18: Mean AUC of Gradient Boosting Classifier before Feature Engineering



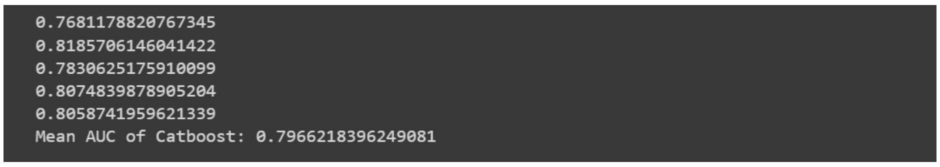
```
0.7709427408337968
0.812920897090018
0.7846723095193964
0.8046591291334583
0.7996169450336719
Mean AUC of Gradient Boost 0.7945624043220683
```

Figure 4.19: Mean AUC of Gradient Boosting Classifier after Feature Engineering



```
996:    learn: 0.2004176        total: 4.13s    remaining: 12.4ms
997:    learn: 0.2003122        total: 4.14s    remaining: 8.29ms
998:    learn: 0.2002027        total: 4.14s    remaining: 4.14ms
999:    learn: 0.2000868        total: 4.14s    remaining: 0us
0.83151073309032

Mean: 0.7395109527634188
STD:  0.1223856088884434
```

Figure 4.20: Mean AUC of CatBoost Classifier before Feature Engineering

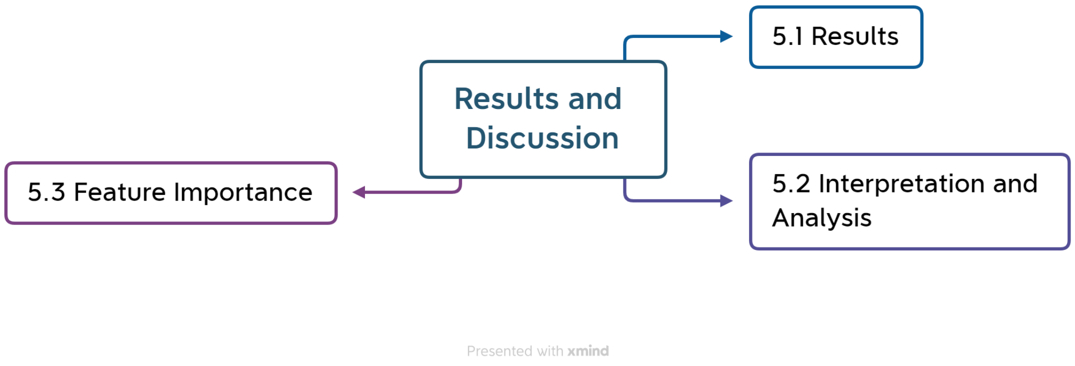


```
0.7681178820767345
0.8185706146041422
0.7830625175910099
0.8074839878905204
0.8058741959621339
Mean AUC of Catboost: 0.7966218396249081
```

Figure 4.21: Mean AUC of CatBoost Classifier after Feature Engineering

Chapter 5

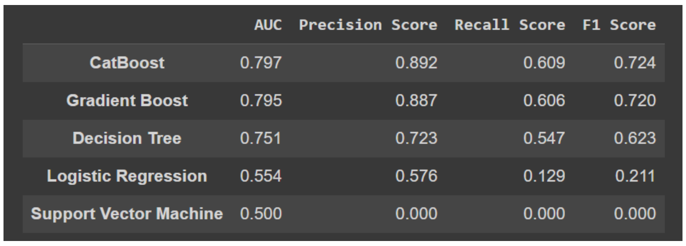
RESULT AND DISCUSSION



As the reader might remember, we thoroughly explained the evaluation metrics used in this research, and the reason we chose them was because of the class imbalance problem we encountered in our dataset. This Chapter extensively explains the results of the evaluation methods that were selected in the methodology phase to be used on the models.

5.1 Results

After applying stratified K-fold cross validation on our models and the results of testing the AUC mean before and after applying Feature engineering to the dataset we found that the score was higher after feature engineering except for the support vector machine, so we investigated in depth the score of the other evaluation metrics as shown in figure 5.1.



	AUC	Precision Score	Recall Score	F1 Score
CatBoost	0.797	0.892	0.609	0.724
Gradient Boost	0.795	0.887	0.606	0.720
Decision Tree	0.751	0.723	0.547	0.623
Logistic Regression	0.554	0.576	0.129	0.211
Support Vector Machine	0.500	0.000	0.000	0.000

Figure 5.1: Evaluation Metrics Scores

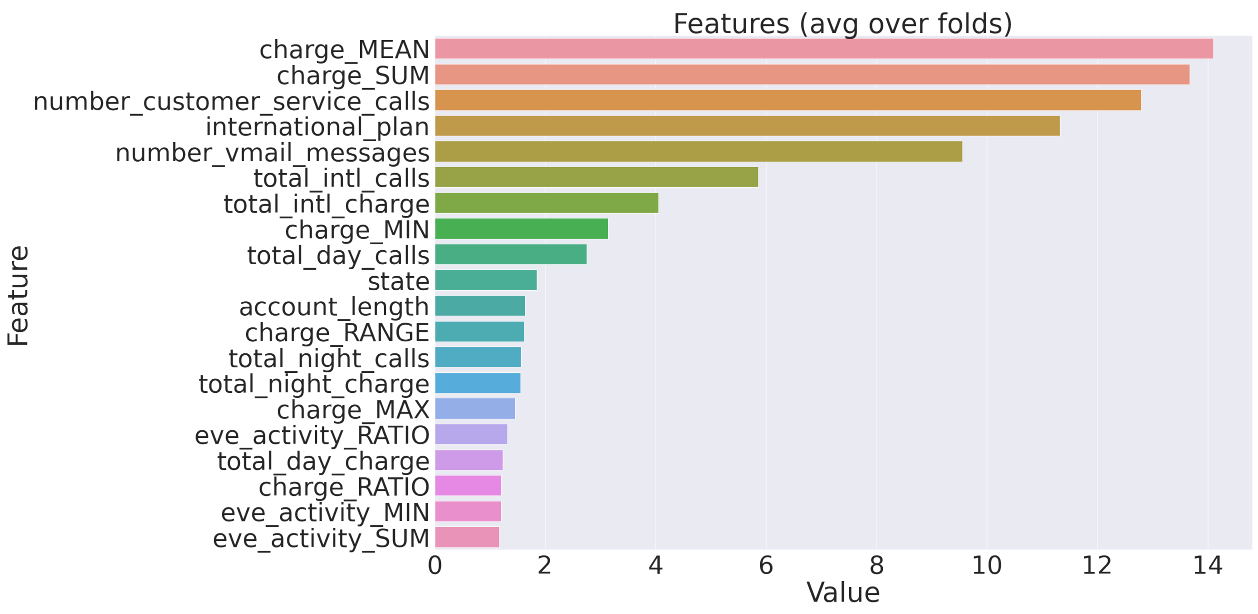


Figure 5.2: Feature Importance

5.2 Interpretation and Analysis

from the above results we can tell that support vector machine and logistic regression model failed to discriminate between classes and showed that they are purely random classifiers.
While, the rest of the models performance was acceptable and might show even more higher and accurate results if more hyperparameter tuning was applied.

CatBoost model significant performance proved to be the most suitable model to apply on the dataset, the transition from 73% to 79% with just few hyperparameter configurations shows that it is the most effective classifier and will produce better results than the other models.

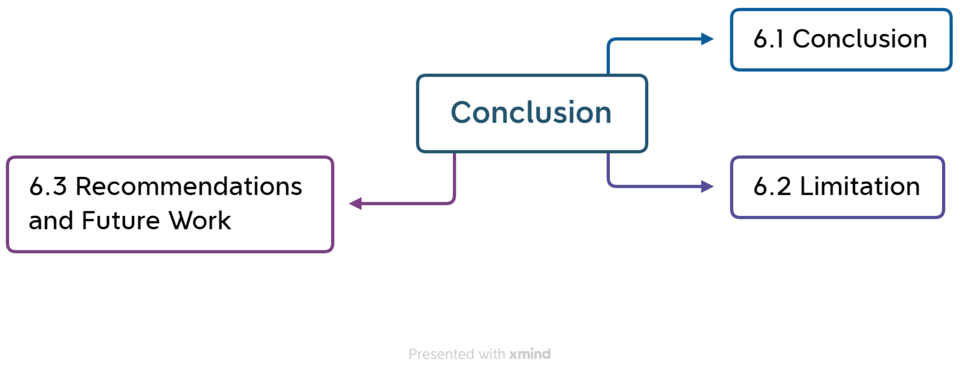
5.3 Feature Importance

Based on the results and analysis we concluded that the top five features that affect churn the most are ’charge mean’, ’charge sum’, ’number of customer service calls’, ’international plan’ and ’number of voicemail messages’.

The highest two are a result of our feature engineering phase which shows that it is an important step to apply when dealing with imbalanced data as shown in Figure 5.2.
while the remaining three were a result of applying the generalized linear model (GLM) and had a P-value less than 0.05 which indicated that they have a huge impact on churn as proven.

Chapter 6

CONCLUSION



The entire research is wrapped up in this chapter, which also summarizes the conclusions drawn from the study. including the limitations and recommendations for improvements in the future.

6.1 Conclusion

In this thesis, it was shown that retaining customers is necessary because increased company competition may cause prices to decline and increased customer churn. We started by deeply investigating the problem and reviewing similar works regarding customer churn problem using machine learning models therefore arriving at identifying the problem and domain, we gathered our data from a secondary source and prepared it in order to gain the highest validation score possible before feeding them to the model, such as, feature engineering and scaling. and that led us to conclude that CatBoost Classifier model is the most accurate customer churn prediction model when compared to the remaining machine learning models.

And, according to our research objective regarding determining the most important factor that influences customer churn, it was found that the top three features are ’charge Mean’, ’charge Sum’ and ’number of customer service calls’ .

6.2 Limitations

The unavailability of local data due to data protection and privacy led us to use open-source data, in the hope it may help us achieve high accuracy and solve local churn prediction problems.

6.3 Recommendations and Future work

During our research, we came to believe that feature engineering and feature selection can have a huge impact on the performance of the model, we attempted using the Recursive Feature elimination as feature selection method and it reduced our dataset to half the total number of features, nominating only 14 optimal features out of 28 (after the feature engineering phase), while still maintaining the same level of performance achieved when applying the dataset containing the whole 28 features, We believe that this can be investigated in greater depth in the future and We feel it will yield respectable findings.
Figure 6.1 shows the result of using RFE on our dataest.

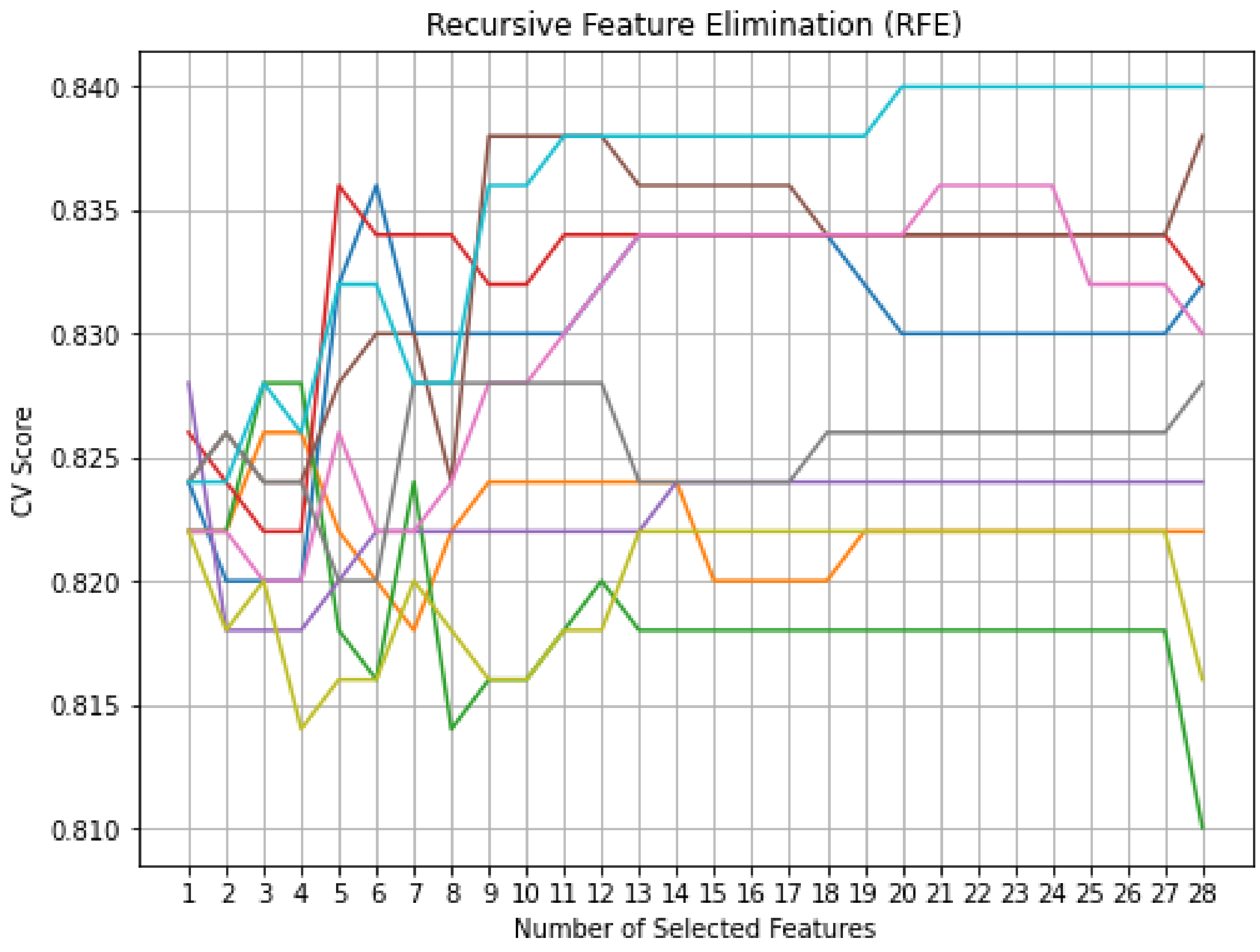


Figure 6.1: Recursive Feature Elimination (RFE)

Bibliography

[1] Customer Churn Prediction 2020 — Kaggle.

[2] Naive Bayes classifier. (2023, January 24). In Wikipedia.

[3] Naive Bayes classifier. (2023, January 24). In Wikipedia.

[4] Sanket Agrawal, Aditya Das, Amit Gaikwad, and Sudhir Dhage. Customer churn prediction modelling based on behavioural patterns analysis using deep learning. In *2018 International conference on smart computing and electronic enterprise (ICSCEE)*, pages 1–6. IEEE, 2018.

[5] Ammara Ahmed and D Maheswari Linen. A review and analysis of churn prediction methods for customer retention in telecom industries. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–7. IEEE, 2017.

[6] Mohammed Al-Mashraie, Sung Hoon Chung, and Hyun Woo Jeon. Customer switching behavior analysis in the telecommunication industry via push-pull-mooring framework: A machine learning approach. *Computers & Industrial Engineering*, 144:106476, 2020.

[7] Nadia Alboukaey, Ammar Joukhadar, and Nada Ghneim. Dynamic behavior based churn prediction in mobile telecom. *Expert Systems with Applications*, 162:113779, 2020.

[8] Michael JA Berry and Gordon S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons, 2004.

[9] S Wael Fujo, Suresh Subramanian, and Moaiad Ahmad Khder. Customer churn prediction in telecommunication industry using deep learning. *Information Sciences Letters*, 11(1):24, 2022.

[10] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[11] Sebastiaan Höppner, Eugen Stripling, Bart Baesens, Seppe vanden Broucke, and Tim Verdonck. Profit driven decision trees for churn prediction. *European journal of operational research*, 284(3):920–933, 2020.

[12] Xin Hu, Yanfei Yang, Lanhua Chen, and Siru Zhu. Research on a customer churn combination prediction model based on decision tree and neural network. In *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 129–132. IEEE, 2020.

[13] Hemlata Jain, Ajay Khunteta, and Sumit Srivastava. Churn prediction in telecommunication using logistic regression and logit boost. *Procedia Computer Science*, 167:101–112, 2020.

[14] Abbas Keramati, Ruholla Jafari-Marandi, Mohammad Aliannejadi, Iman Ahmadian, Mahdieh Mozaffari, and Uldoz Abbasi. Improved churn prediction in telecommunication industry using data mining techniques. *Applied Soft Computing*, 24:994–1012, 2014.

[15] Sulim Kim and Heeseok Lee. Customer churn prediction in influencer commerce: an application of decision trees. *Procedia Computer Science*, 199:1332–1339, 2022.

[16] Thanasis Vafeiadis, Konstantinos I Diamantaras, George Sarigiannidis, and K Ch Chatzisavvas. A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55:1–9, 2015.

[17] Guo-en Xia and Wei-dong Jin. Model of customer churn prediction on support vector machine. *Systems Engineering-Theory & Practice*, 28(1):71–77, 2008.

Appendix

This section includes important parts of our python code used in building and evaluating the models.

Here is a link to the code:
`https://colab.research.google.com/drive/1vUDsFpbh-yb3pakP6-tvJhMwodtR3nwe`
`scrollTo=qgXftJTVsB5n`

Importing Libraries

```
import numpy as np #linear algebra
import pandas as pd #data processing
pd.set_option('display.max_columns', None) #display all columns
import plotly.express as px #visualization library
import matplotlib.pyplot as plt #visualization library
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_auc_score
from sklearn.svm import SVC
!pip install catboost
from catboost import CatBoostClassifier
from sklearn import preprocessing #models library
from sklearn.metrics import roc_auc_score, precision_score,recall_score,
f1_score
```

Data Acquiring Cleaning and Preparation

```
cust_churn_data = pd.read_csv('CCP_data.csv')

cust_churn_data.head()

cust_churn_data.info()

def uniqueval(df, message):
  print(f'{message}:')
  print(df.nunique())

uniqueval(cust_churn_data, 'unique values in the dataset')
```

Exploratory Data Analysis

```
target_instance = cust_churn_data["churn"].value_counts().to_frame()
target_instance = target_instance.reset_index()
target_instance = target_instance.rename(columns={'index':'category'})
fig = px.pie(target_instance, values='churn', names='category',
color_discrete_sequence=["purple","yellow"],
title="Distribution of Churn")
fig.show()

#Defining bar chart function
def bar(feature, df= cust_churn_data ):
    #Groupby the categorical feature
    temp_df = cust_churn_data.groupby([feature, 'churn']).size()
    .reset_index()
    temp_df = temp_df.rename(columns={0:'Count'})
    #Calculate the value counts of each distribution
    and it's corresponding Percentages

    value_counts_df = df[feature].value_counts().to_frame().
    reset_index()
    categories = [cat[1][0] for cat in
    value_counts_df.iterrows()]

    #Calculate the value counts of each distribution
    and it's corresponding Percentages

    num_list = [num[1][1] for num in
    value_counts_df.iterrows()]
```

```
div_list = [element / sum(num_list) for element
in num_list]
percentage = [round(element * 100,1) for element
in div_list]

#Defining string formatting for graph annotation
#Numeric section
def num_format(list_instance):
    formatted_str = ''
    for index,num in enumerate(list_instance):
        if index < len(list_instance)-2:
            formatted_str=formatted_str+f'{num}%,
            ' #append to empty string(formatted_str)
        elif index == len(list_instance)-2:
            formatted_str=formatted_str+f'{num}% & '
        else:
            formatted_str=formatted_str+f'{num}%'
    return formatted_str
#Categorical section
def str_format(list_instance):
    formatted_str = ''
    for index, cat in enumerate(list_instance):
        if index < len(list_instance)-2:
            formatted_str=formatted_str+f'{cat}, '
        elif index == len(list_instance)-2:
            formatted_str=formatted_str+f'{cat} & '
        else:
            formatted_str=formatted_str+f'{cat}'
    return formatted_str


#Running the formatting functions
num_str = num_format(percentage)
cat_str = str_format(categories)

#Setting graph framework
fig = px.bar(temp_df, x=feature, y='Count',
color='churn', title=f'churn rate by {feature}',
barmode="group",
color_discrete_sequence=["purple", "yellow"])
```

```
    fig.add_annotation(
                text=f'Value count of distribution of
                {cat_str} are<br>{num_str} percentage respectively.',
                align='left',
                showarrow=False,
                xref='paper',
                yref='paper',
                x=1.4,
                y=1.3,
                bordercolor='black',
                borderwidth=1)
    fig.update_layout(
        # margin space for the annotations on the right
        margin=dict(r=400),
    )

    return fig.show()

#state feature plot
bar('state')

#area code feature plot
bar('area_code')

#international plan feature plot
cust_churn_data.loc[cust_churn_data.
international_plan==0,'international_plan'] = "No"
#convert 0 to No in all data instances
cust_churn_data.loc[cust_churn_data.
international_plan==1,'international_plan'] = "Yes"
#convert 1 to Yes in all data instances
bar('international_plan')

#voice mail plan feature plot
cust_churn_data.loc[cust_churn_data.
voice_mail_plan==0,'voice_mail_plan'] = "No"
#convert 0 to No in all data instances
cust_churn_data.loc[cust_churn_data.
voice_mail_plan==1,'voice_mail_plan'] = "Yes"
#convert 1 to Yes in all data instances
```

```
bar('voice_mail_plan')

#Numerical Features Histograms
num_feats = list(cust_churn_data.select_dtypes
(include=['int64', 'float64', 'int32']).columns)

cust_churn_data[num_feats].hist(figsize=(20,15));
```

Transforming the data

```
feats = list(cust_churn_data.select_dtypes
(include=['object','category']).columns)
le = LabelEncoder()
df = cust_churn_data
for f in feats:
    le.fit(df[f])
    cust_churn_data[f] = le.transform(cust_churn_data[f])

# Plot the correlations as a heatmap

corr = cust_churn_data.corr()
fig = px.imshow(corr, text_auto=False, width=1000, height= 900)

fig.show()

#dropping features
cust_churn1 = cust_churn_data.drop
(['total_day_minutes','total_eve_minutes',
'total_night_minutes','total_intl_minutes',
'voice_mail_plan'], axis=1)
print(); print(cust_churn1.head())
```

Feature Engineering

```
def Statistics(features,name):
    for dataset in [cust_churn1]:
        dataset[f"{name}_SD"] = dataset[features].std(axis=1)
        dataset[f"{name}_MEAN"] = dataset[features].mean(axis=1)
        dataset[f"{name}_MIN"] = dataset[features].min(axis=1)
        dataset[f"{name}_MAX"] = dataset[features].max(axis=1)
        dataset[f"{name}_RANGE"] = dataset[f"{name}_MAX"] -
        dataset[f"{name}_MIN"]
        dataset[f"{name}_RATIO"] = (dataset[f"{name}_MIN"] /
        dataset[f"{name}_MAX"]).fillna(0)
        dataset[f"{name}_SUM"] =
        dataset[features].sum(axis=1)

feats = ['total_day_charge','total_eve_charge',
'total_night_charge','total_intl_charge']
Statistics(feats,'charge')

feats = ['total_eve_calls','total_eve_charge']
Statistics(feats,'eve_activity')
```

Generalized Linear Model

```
import statsmodels.api as sm #import statsmodels
import statsmodels.formula.api as smf #support for formulas
from statsmodels.formula.api import glm #to use glm() directly

#Change variable name separators to '_'
all_columns = [column.replace(" ", "_").
replace("(", "_")
.replace(")", "_").replace("-", "_") for column in
cust_churn1.columns]

#Effect the change to the dataframe column names
cust_churn1.columns = all_columns

#Prepare it for the GLM formula
glm_columns = [e for e in all_columns if e not in
['churn']]
glm_columns = ' + '.join(map(str, glm_columns))

#Fiting it to the Generalized Linear Model
glm_model = smf.glm(formula=f'churn ~ {glm_columns}',

data=cust_churn1, family=sm.families.Binomial())

res = glm_model.fit()
print(res.summary())
```

```
np.exp(res.params)
```

Scaling numeric Variables

```
from sklearn import preprocessing
import pandas as pd
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
cust_churn1[['account_length','number_vmail_messages'
,'total_day_calls','total_day_charge','total_eve_call
s','total_eve_charge','total_night_calls','total_nigh
t_charge','total_intl_calls','total_intl_charge','num
ber_customer_service_calls','eve_activity_SUM','eve_a
ctivity_SD','eve_activity_RANGE','eve_activity_RATIO'
,'eve_activity_MAX','eve_activity_MIN','eve_activity_
MEAN','charge_SD','charge_RANGE','charge_RATIO','char
ge_MAX','charge_MIN','charge_MEAN','charge_SUM']] =
scaler.fit_transform(cust_churn1[['account_length','n
umber_vmail_messages','total_day_calls','total_day_ch
arge','total_eve_calls','total_eve_charge','total_nig
ht_calls','total_night_charge','total_intl_calls','to
tal_intl_charge','number_customer_service_calls','eve
_activity_SUM','eve_activity_SD','eve_activity_RANGE'
,'eve_activity_RATIO','eve_activity_MAX','eve_activit
y_MIN','eve_activity_MEAN','charge_SD','charge_RANGE'
,'charge_RATIO','charge_MAX','charge_MIN','charge_MEA
N','charge_SUM']])
```

Modeling

```
lr_params = {'C':10, 'max_iter': 10000}
sv_params = {'C':500.0, 'kernel':'rbf'}
dt_params = {'max_depth' : 15}
gb_params = {'max_depth' : 3}
cb_params = {'depth': 5, 'iterations': 1000,
'learning_rate': 0.01, 'verbose':0}

lr = LogisticRegression(**lr_params, random_state=42)
sv = SVC(**sv_params, random_state=42)
dt = DecisionTreeClassifier(**dt_params, random_state=42)
gb = GradientBoostingClassifier(**gb_params, random_state=42)
cb = CatBoostClassifier(**cb_params, random_state=42)
```

```
#Save the predictions to calculate the metrics later
preds = pd.DataFrame(index=cust_churn1.index)

X = cust_churn1.drop('churn',axis=1)
y = cust_churn1['churn']


scores = []
for train_index, test_index in StratifiedKFold(n_splits=5, shuffle=True,
random_state=1).split(X,y):
    X_train, X_test = X.values[train_index], X.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]

    lr.fit(X_train,y_train)
    y_preds = lr.predict(X_test)
    preds.loc[test_index,'LR'] = y_preds.copy()

    scores.append(roc_auc_score(y_test,y_preds))
    print(scores[-1])

print(f'Mean AUC of Logistic Regression: {np.mean(scores)}')

X = cust_churn1.drop('churn',axis=1)
y = cust_churn1['churn']


scores = []
for train_index, test_index in StratifiedKFold(n_splits=5, shuffle=True,
random_state=1).split(X,y):
    X_train, X_test = X.values[train_index], X.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]

    dt.fit(X_train,y_train)
    y_preds = dt.predict(X_test)
    preds.loc[test_index,'DT'] = y_preds.copy()

    scores.append(roc_auc_score(y_test,y_preds))
    print(scores[-1])
print(f'Mean AUC of Decision Tree: {np.mean(scores)}')

X = cust_churn1.drop('churn',axis=1)
y = cust_churn1['churn']


scores = []
for train_index, test_index in StratifiedKFold(n_splits=5, shuffle=True,
random_state=1).split(X,y):
    X_train, X_test = X.values[train_index], X.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]

    sv.fit(X_train,y_train)
    y_preds = sv.predict(X_test)
    preds.loc[test_index,'SVM'] = y_preds.copy()

    scores.append(roc_auc_score(y_test,y_preds))
    print(scores[-1])

print(f'Mean AUC of Support Vector Machine: {np.mean(scores)}')

X = cust_churn1.drop('churn',axis=1)
y = cust_churn1['churn']


scores = []
for train_index, test_index in StratifiedKFold(n_splits=5, shuffle=True,
random_state=1).split(X,y):
    X_train, X_test = X.values[train_index], X.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]

    gb.fit(X_train,y_train)
    y_preds = gb.predict(X_test)
    preds.loc[test_index,'GB'] = y_preds.copy()

    scores.append(roc_auc_score(y_test,y_preds))
    print(scores[-1])

print(f'Mean AUC of Gradient Boost {np.mean(scores)}')
```

```
X = cust_churn1.drop('churn',axis=1)
y = cust_churn1['churn']


scores = []
for train_index, test_index in StratifiedKFold(n_splits=5, shuffle=True,
random_state=1).split(X,y):
    X_train, X_test = X.values[train_index], X.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]

    cb.fit(X_train,y_train)
    y_preds = cb.predict(X_test)
    preds.loc[test_index,'CB'] = y_preds.copy()

    scores.append(roc_auc_score(y_test,y_preds))
    print(scores[-1])

print(f'Mean AUC of Catboost: {np.mean(scores)}')
```

Metrics measurment

```
from sklearn.metrics import roc_auc_score, precision_score, recall_score
f1_score
#Metrics for Logistic Regression
print('Logistic Regression')
print("AUC: ",roc_auc_score(y, preds['LR']))
print("Precision: ",precision_score(y, preds['LR']))
print("Recall: ",recall_score(y, preds['LR']))
print("F1 Score: ",f1_score(y, preds['LR']))

from sklearn.metrics import roc_auc_score, precision_score, recall_score
f1_score
#Metrics for SVM
print('SVM')
print("AUC: ",roc_auc_score(y, preds['SVM']))
print("Precision: ",precision_score(y, preds['SVM']))
print("Recall: ",recall_score(y, preds['SVM']))
print("F1 Score: ",f1_score(y, preds['SVM']))

from sklearn.metrics import roc_auc_score, precision_score, recall_score
f1_score
```

```
#Metrics for Catboost
print('Catboost')
print("AUC: ",roc_auc_score(y, preds['CB']))
print("Precision: ",precision_score(y, preds['CB']))
print("Recall: ",recall_score(y, preds['CB']))
print("F1 Score: ",f1_score(y, preds['CB']))

from sklearn.metrics import roc_auc_score, precision_score, recall_score
f1_score
#Metrics for Decision Tree
print('Decision Tree')
print("AUC: ",roc_auc_score(y, preds['DT']))
print("Precision: ",precision_score(y, preds['DT']))
print("Recall: ",recall_score(y, preds['DT']))
print("F1 Score: ",f1_score(y, preds['DT']))

from sklearn.metrics import roc_auc_score, precision_score, recall_score
f1_score
#Metrics for Gradient Boost
print('Gradient Boost')
print("AUC: ",roc_auc_score(y, preds['GB']))
print("Precision: ",precision_score(y, preds['GB']))
print("Recall: ",recall_score(y, preds['GB']))
print("F1 Score: ",f1_score(y, preds['GB']))

#Plot the Features Importances
def plotImp(model, X , num = 20, fig_size = (40,
20)):
    feature_imp = pd.DataFrame({'Value':model.feature_importances_,
    'Feature':X.columns})
    plt.figure(figsize=fig_size)
    sns.set(font_scale = 5)
    sns.barplot(x="Value", y="Feature",
    data=feature_imp.sort_values(by="Value",
                            ascending=False)[0:num])
    plt.title('Features (avg over folds)')
    plt.tight_layout()
    plt.savefig('importances-01.png')
    plt.show()
    sns.set()
```

```
plotImp(cb,cust_churn1.drop('churn',axis=1))

from sklearn.metrics import roc_auc_score, precision_score, recall_score
f1_score

index_=['Logistic Regression',
        'Decision Tree',
        'Support Vector Machine',
        'CatBoost',
        'Gradient Boost']

all_models = {'AUC':np.round([roc_auc_score(y, preds['LR']),
                                            roc_auc_score(y,

                               preds['DT']),
                                            roc_auc_score(y,
                                            preds['SVM']),
                                            roc_auc_score(y,
                                            preds['CB']),
                                            roc_auc_score(y,
                                            preds['GB']),], 3),
             'Precision Score':np.round([precision_score(y, preds['LR'])
                                            precision_score(y,
                                            preds['DT']),
                                            precision_score(y,
                                            preds['SVM']),
                                            precision_score(y,
                                            preds['CB']),
                                            precision_score(y,
                                            preds['GB']),], 3),
             'Recall Score':np.round([recall_score(y, preds['LR']),
                                            recall_score(y,
                                            preds['DT']),

                            recall_score(y,
                            preds['SVM']),
                                           recall_score(y,
                                           preds['CB']),
                                            recall_score(y,
                                        preds['GB']),], 3),
           'F1 Score':np.round([f1_score(y, preds['LR']),
                                        f1_score(y, preds['DT']),
                                        f1_score(y, preds['SVM']),
                                        f1_score(y, preds['CB']),
                f1_score(y, preds['GB']),], 3)
}

final_results_df = pd.DataFrame(data=all_models, index=index_)
final_results_df.sort_values(by='AUC', ascending=False)
```