Customer Churn Prediction in Telecommunication Companies (Supervised Machine Learning)

Mitra Bitaraf Fazel

School of Computing and Digital Technology

Faculty of Computing, Engineering and the Built Environment

Birmingham City University

May 2023

Abstract

This dissertation investigates a solution for predicting customer churn in telecommunication companies using supervised machine learning techniques. The proposed solution involves four main steps: data preparation, feature selection, model training, and model evaluation. The data preparation step involves cleaning and transforming the data to make it suitable for analysis. The feature selection step uses a decision tree classifier to identify the most important features for predicting customer churn. The model training step involves training a machine learning model using the selected features and a labelled dataset. Finally, the model evaluation step uses a cross-validation method to evaluate the model's performance.

The investigated supervised machine learning solutions were tested on a dataset containing information about customer gender, seniority, partner status, dependent status, and tenure. The results show that the solutions can effectively predict customer churn and can be used by telecommunication companies to improve customer retention. However, overfitting was observed and hyperparameter tuning is performed to overcome overfitting. The best decision tree classifier model identified tenure as the most important feature for predicting customer churn, followed by seniority and partner status. Furthermore, it achieved an accuracy of 80% and an F1 score of 0.6, indicating that it can be a useful tool for telecommunication companies to reduce customer churn and improve customer satisfaction.

Acknowledgements

I would like to express my sincere gratitude to all those who have contributed in my BSc journey. Firstly, I would like to thank my supervisor, Dr. Atif Azad for his invaluable guidance, support, and encouragement throughout the project. His expertise, insights, and suggestions were crucial in shaping my work and achieving the best possible outcomes.

I would like to express my appreciation to my friends, and family for their understanding, encouragement, and motivation throughout my project. Their unwavering support has kept me focused and driven to accomplish my goals.

Table of Contents

Glossary

AI	Artificial intelligence
ML	Machine learning
DT	Decision tree
SML	Supervised Machine Learning
LR	Logistic Regression
XGBoost	Extreme Gradient Boosting
SVM	Support Vector Machine

List of Figures

List of Tables

1 Introduction

In recent years, customer churn has become a significant issue for telecommunication companies as the competition in the market has become more intense (Bhattacharyya and Dash, 2022). Customer churn refers to the phenomenon where customers terminate their subscriptions or contracts with a company and switch to another provider. This phenomenon leads to a loss of revenue and market share for telecommunication companies (Huang et al., 2012). Therefore, predicting customer churn has become a crucial task for these companies. In this research, we aim to investigate and analyse a predictive model to identify customers who are likely to churn.

Telecommunication companies generate large volumes of data, including customer demographic information, service usage patterns, contract details, and payment history, among others. This data can be used to understand customer behaviour and predict churn. Predictive models can help telecommunication companies in several ways, such as identifying customers who are likely to churn and taking proactive measures to retain them, identifying the reasons for churn and addressing them, and optimizing marketing strategies to attract new customers.

To achieve this goal, we will use a dataset containing customer information from a telecommunication company. The dataset includes various features that can be used to predict customer churn. We will use machine learning techniques such as Cox proportional hazard regression, random survival forest, and gradient boosting survival analysis to develop predictive models. These models will be evaluated based on their confusion matrix, accuracy, precision, recall, and area under the curve (AUC).

The findings of this dissertation can be beneficial for telecommunication companies to retain their customers, optimize marketing strategies, and improve customer satisfaction. Moreover, this research can contribute to the academic literature by providing insights into the factors that contribute to customer churn in telecommunication companies and the effectiveness of different predictive models for customer churn.

1.1 Problem Definition

Customer churn prediction is an important problem for telecommunication companies (Bhattacharyya and Dash, 2022). Churn refers to the customers who cancel their subscription or stop using the company's services. In today’s competitive market losing customers can lead to significant revenue losses and affect the company's reputation. Therefore, predicting which customers are likely to churn could help companies take proactive measures to retain them (Zhang, Moro, and Ramos, 2022). The goal here is to build a predictive model that can accurately identify the customers who are likely to churn based on their demographic, behavioural, usage pattern and some other important features that will be discussed further in the later sections.

1.2 Scope

This project investigates application of different machine learning techniques and algorithms i.e. decision trees, logistic regression, and XGBoost, in the marketing field for predict customer churn. The focus here is predicting customer churn, while the strategies of customer acquisition and retaining the customers who are at risk are out of the scope of this research.

1.3 Rationale

The telecommunications industry has some of the highest customer churn rates among all industries, which means that a large number of customers are leaving or cancelling their services (Zhang, Moro, and Ramos, 2022). This has a significant impact on the profitability of telecommunication companies, making it crucial to understand and predict churn. By predicting customer churn, telecommunication companies can take proactive measures to retain customers and reduce the cost of customer acquisition. Predicting customer churn also can help companies stay ahead and at the top of the competition by offering personalized services and offers to retain customers.

This project benefits both companies and customers in the telecommunication industry by predicting customer churn through data mining techniques. This project could help businesses perform in-depth analysis of customer behaviour and support decision-making processes, resulting in reduced churn rates, improved customer services, and reduced costs with increased profits. By comparing different machine learning algorithms to predict the most accurate customer churn rate, this project will classify customers based on their characteristics, allowing companies to provide personalized services and offers. Ultimately, customers can receive improved services, bonuses, and special offers relevant to their individual history with the company. Furthermore, telecommunication companies generate large amounts of data on customer usage and behaviour. By leveraging this data and using machine learning algorithms to predict churn, companies can gain insights into customer behaviour and make data-driven decisions.

1.4 Project Aim and Objectives

The aim of this study is to develop multiple supervised machine learning models, compare them, and choose the best model for predicting customer churn in the telecommunications industry, utilizing historical customer data to identify patterns and trends that may indicate an increased likelihood of churn. The goal is to accurately predict which customers are at risk of leaving the company, and to use this information to develop targeted interventions to reduce churn rates and improve customer retention.

Study the most important factors contribute to customer churn in telecommunication companies, and how can machine learning models be used to predict and prevent customer churn.

The objectives of this project are as follows:

- Identify patterns and trends in customer behaviour that predict churn.
- Develop a predictive model that utilizes these important factors and variables, using machine learning techniques, with the aim of accurately identifying customers who are at risk of churning.
- Use the prediction model to target at-risk customers with retention campaigns and incentives.
- Use insights gained from the prediction model to inform decisions about product development and marketing strategies.
- Evaluate the performance of the predictive model using appropriate evaluation metrics in order to assess its effectiveness in predicting customer churn.
- Identify areas where further research is needed, and make recommendations for future work in the field of customer churn prediction in telecommunications companies.

1.5 Background Information

More recently, telecommunication companies have started to leverage machine learning algorithms, such as decision trees, random forests, and neural networks, to build more accurate predictive models (Rahman and Kumar, 2020). These models can analyse larger and more complex datasets, including unstructured data such as call transcripts and social media interactions. The use of machine learning can enable more precise targeting of at-risk customers and more effective retention campaigns. Overall, companies are continuously refining their churn prediction models and incorporating new data sources and machine learning techniques to improve the accuracy and effectiveness of their customer retention efforts.

2 The Literature Review

2.1 Themes

The current project focuses on the theme of customer churn prediction, which is the main topic of investigation. To structure the review, several themes were identified and explored, including: (1) definition and importance of customer churn prediction and its methods; (2) supervised machine learning (SML) and its applicability for customer churn prediction; (3) SML algorithms, specifically Decision trees, Logistic regression, Super Vector machine, Extreme Gradient Boosting; (4) data cleaning and pre-processing; (5) extraction, including the analysis of correlations between different features; and (6) evaluation metrics, which are used to quantify the performance of SML. Key words were used to search the literature for relevant studies and information.

2.2 Review of Literature

2.2.1 customer churn prediction

Keyword: A description of customer churn prediction

Customer churn prediction is the process of using data mining and machine learning techniques to predict which customers are most likely to stop doing business with a company. This can be used to identify high-risk customers, so that the company can take proactive steps to retain them, such as offering special promotions or incentives. Churn prediction models typically use historical data on customer behaviour, such as purchase history and customer demographics, to make predictions about future behaviour. These models are commonly used in industries such as telecommunications, finance, and e-commerce (Churn, 2003; Zaki and O'malley, 2007)

Keyword: The importance of customer churn prediction (why, for whom, and how?)

The prediction of customer churn is critical for businesses and organizations that rely on recurring revenue from customers. It is particularly relevant in industries such as telecommunications, finance, e-commerce, and SaaS where customer acquisition costs are high and retaining customers is vital (Reichheld and Sasser, 1996). By predicting customer churn, businesses can take proactive measures to retain customers and reduce the costs associated with acquiring new ones, thereby improving overall revenue. Additionally, customer churn prediction can inform better customer segmentation and targeting strategies, inform product development and marketing efforts, and improve the customer experience. Ultimately, predicting customer churn can lead to increased customer lifetime value, reduced operational costs, and improved profitability (Srivastava and S. Rangaswamy, 2002; Zaki and O'malley, 2007).

Keyword: The methods have been used for customer churn prediction

The prediction of customer churn, or the likelihood that a customer will discontinue their relationship with a company, can be accomplished through various methods in the field of machine learning. Some of the commonly used techniques include logistic regression, decision trees, random forests, gradient boosting, neural networks, support vector machines, K-Means clustering, association rule mining, and hybrid models (Zaki and O'malley, 2007; Raza and Besar, 2017). Each method presents its own advantages and disadvantages and the choice of which one to use will depend on the characteristics of the data and the specific business problem at hand. The use of labelled data in supervised machine learning methods, such as decision trees and logistic regression, enables the training of models that can predict customer churn based on past customer behaviour (Raza and Besar, 2017). On the other hand, unsupervised learning methods like K-Means clustering can identify customer groups with similar characteristics and behaviours, aiding in the prediction of customer churn. The combination of multiple methods in hybrid models can also improve the accuracy of churn prediction. (Wilcox, 2002; Proença et al., 2010).

Table 1. Comparison of customer churn prediction methods (Wilcox, 2002; Zaki and O'malley, 2007; Raza and Besar, 2017; Proença et al., 2010)

Method		**Advantage**	**Disadvantage**
Supervised Machine Learning	Logistic Regression	Simple to implement, easy to interpret, handles categorical and numerical data, can handle multiple input variables	Assumes linearity between the input variables and output variable, assumes independence of input variables
	Decision Trees	Simple to interpret, handles categorical and numerical data, can handle multiple input variables, can be visualized	Prone to overfitting, assumption of independence of input variables
	Random Forest	Handle high-dimensional data, non-linear relationships and handle missing data, can be used for feature selection	Prone to overfitting, not easy to interpret
	Gradient Boosting	Handle high-dimensional data, non-linear relationships and handle missing data, can be used for feature selection, powerful method for feature selection	Prone to overfitting, not easy to interpret
	Neural Networks	Handle high-dimensional data, non-linear relationships and handle missing data, can handle complex data, can handle interactions among the input variables	Can be difficult to interpret, require a large amount of data
	Support Vector Machines	Handle high-dimensional data, non-linear relationships and handle missing data, can handle complex	Can be difficult to interpret, require a large amount of data

	data, can handle interactions among the input variables	
K-Means Clustering	Unsupervised learning, can identify groups of customers who have similar characteristics and behaviour	Assumes spherical shape of clusters, sensitive to initial conditions, not easy to interpret
Association rule mining	Can find association between different variables and customer churn	Not easy to interpret, require a large amount of data
Hybrid models	Combining multiple methods can improve the accuracy of the prediction	Complexity, not easy to interpret

Among the methods above, supervised machine learning methods provide a wide range of advantages to explore large datasets with nonlinear interdependencies. The following summarises these advantages, as well as the disadvantages.

Advantages of supervised machine learning method

Supervised machine learning is a widely used method for customer churn prediction that can achieve high accuracy when trained on a large amount of labelled data. It is capable of handling large amounts of data and extracting useful information, as well as automating the prediction process and handling multiple input variables (Raza and Besar, 2017; Proença et al., 2010). However, it requires labelled data that may be difficult or expensive to obtain, and can overfit the data if not properly validated, leading to poor performance on new data. Additionally, supervised machine learning algorithms can be complex and difficult to interpret, and assume independence of input variables, which may not always be true (Rahman, M. and Kumar, V., 2020). They can also be inflexible when it comes to handling new, unseen data that deviates from the training data. The choice of method for a specific problem will depend on the characteristics of the data and the business problem, as well as the available resources (Zaki and O'malley, 2007; Raza and Besar, 2017).

2.2.2 Supervised machine learning (SML)

Applications of SNL include classification and regression which are discussed in the following. Figure 1 shows schematically how SML estimates a trend in a set of data, i.e., regression, versus compartmenting a data set by labelling them, i.e., classification. Each of these applications address different problems in engineering, medicine, data science, etc. The following explores each of these SNL applications (Zaki and O'malley, 2007; Proença et al., 2010).

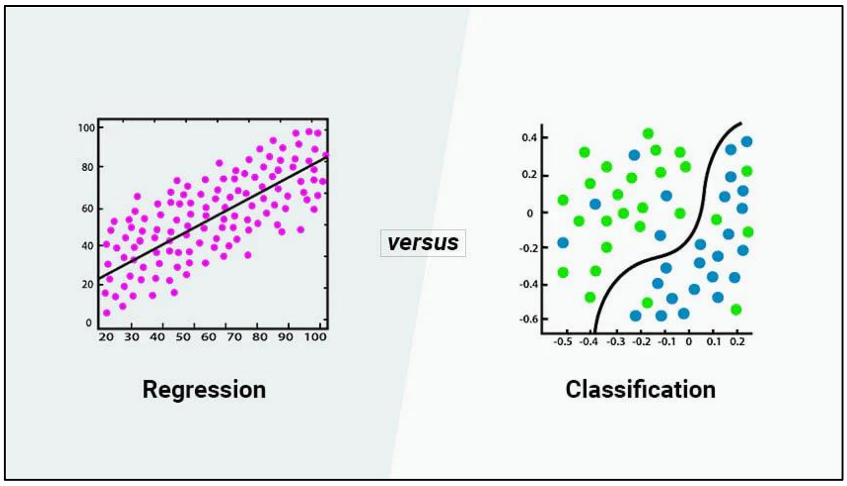


Figure 1. Schematic of the application of supervised machine learning for regression versus classification problems (https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article)

Keyword: Classification task

Supervised machine learning methods carry out classification tasks by learning the relationship between the input variables and the output variable (churn or non-churn) from a labelled training dataset. An example is to train a SML for separating spam emails from ham email (see figure 2). SML can be trained to recognise specific common features in spam emails and correlate them with the spam label.

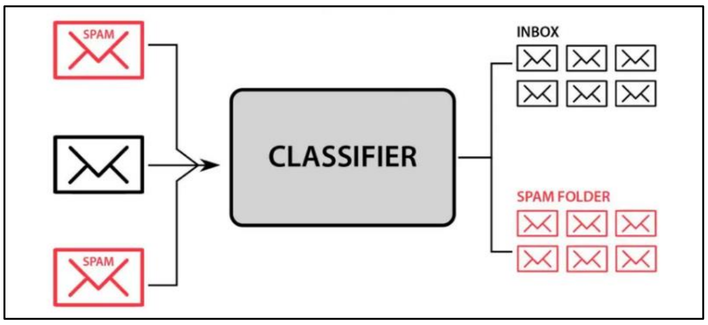


Figure 2. SML for classifying email into spam and ham emails (https://www.analytixlabs.co.in/blog/classification-in-machine-learning).

The process of classification in supervised machine learning involves several steps: data preparation, feature selection, model training, evaluation, tuning, and deployment (Rahman, M. and Kumar, V., 2020). The first step is to clean and pre-process the data. Then, the input features are selected, followed by training a model on labelled training data. The trained model is evaluated on a separate dataset, and its parameters may be adjusted for improved performance. Finally, the satisfactory model is deployed for making predictions on new data. The steps and techniques can vary for different methods, but the overall process remains similar.

Keyword: Regression task

Supervised machine learning methods can also be used to carry out regression tasks, which involve predicting a continuous output variable based on a set of input variables. The underlying theory and equations for different regression methods can vary, but a common approach is linear regression (Saravanan and Sujatha, 2018).

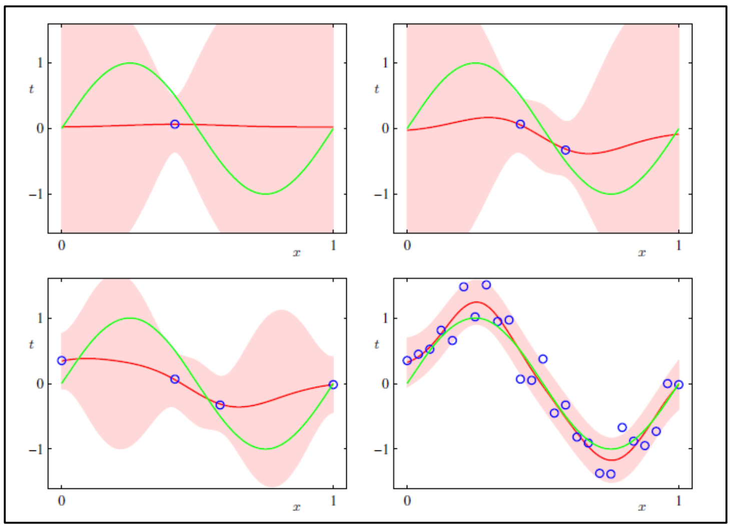


Figure 3. Application of SML for regression with different number of data, where green line in the plots represent the target pattern, blue circles are the sample points used to train the SML, and the red lines are the SML estimations (https://stats.stackexchange.com/questions/311266/gaussian-basis-function-in-bayesian-linear-regression).

Linear regression is a method that models the relationship between the input variables and the output variable as a linear equation. The equation can be represented as:

$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n * x_n$

Where y is the output variable, x_1, x_2, ..., x_n are the input variables, and b_0, b_1, b_2, ..., bn are the coefficients of the equation (McGraw-Hill, 1997). The goal of linear regression is to find the values of the coefficients that best fit the data.

To find the values of the coefficients, the method of least squares is used. The method of least squares is a technique for finding the values of the coefficients that minimize the sum of the squared residuals. The residual is the difference between the predicted value and the actual value. The sum of the squared residuals is represented by the following equation:

$\mathrm{SSE} = \Sigma(y - y')^2$

Where y is the actual value, y' is the predicted value, and Σ represents the sum over all data points (McGraw-Hill, 1997).

Once the coefficients have been estimated, the linear regression model can be used to make predictions on new, unseen data. The model can be used to predict the output variable for a given set of input variables by plugging the values of the input variables into the equation and solving for the output variable (Mitchell and McGraw-Hill, 1997; Hastie, 2009; James et al., 2013).

2.2.3 Supervised machine learning algorithms

As discussed previously, decision tree and random forest are two of the SML algorithm that are widely used due to the advantages which they offer.

Keyword: Decision trees classification

The Decision Tree Algorithm is a supervised learning method that can be used for both classification and regression tasks. It builds a tree-like model of decisions and their possible outcomes based on the concepts of entropy and information gain. The goal is to reduce the entropy by making a series of decisions that split the data into subsets based on the values of the input variables. The algorithm starts with the root node that represents the entire dataset and calculates the entropy of the dataset. It then selects the input variable with the highest information gain and splits the data into subsets. This process is repeated recursively until a stopping criterion is met. The final result is a tree structure that represents the decisions and their consequences with leaf nodes representing class labels. The tree can be used for predictions by traversing the tree and arriving at a leaf node that represents the prediction (Gupta et al., 2017).

An example of using the Decision Tree Algorithm is in a guessing game, where a person eliminates options by asking questions and categorizing objects to arrive at the correct one. For example, a person imagines an object such as a car or a bus, and the other person eliminates options by asking questions such as "does it have wheels?" until the correct object is identified. However, the decision tree algorithm can be prone to overfitting, which can be addressed through techniques like pruning, cross-validation, and bagging (James et al., 2013).

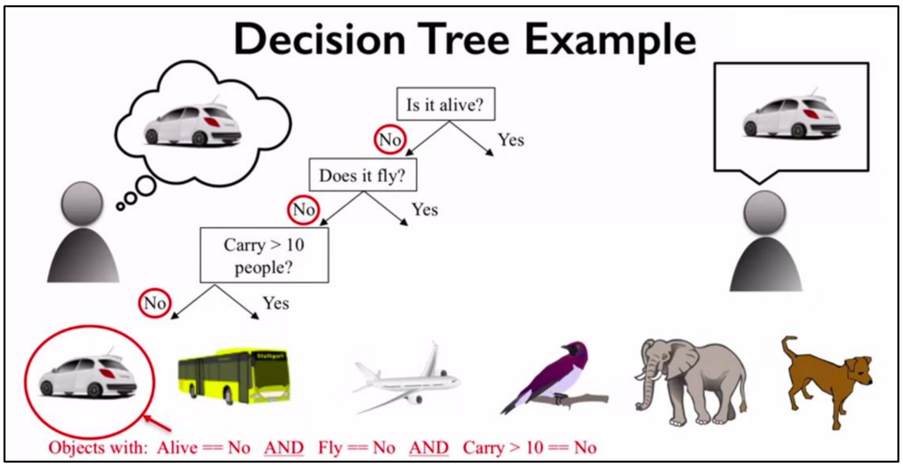


Figure 4. Example of decision tree in a guessing game (https://machine-learning-and-data-science-with-python.readthedocs.io/en/latest/assignment5_sup_ml.html)

The decision tree algorithm uses the following main equations to calculate information gain and entropy.

Information Gain: This equation calculates the decrease in entropy after a dataset is split on an attribute. It is calculated as the difference between the current entropy and the weighted average entropy of the subsets. The equation is as follows:

$\mathrm{IG}(S, A) = \mathrm{Entropy}(S) - \Sigma\ (|S_v| / |S|) * \mathrm{Entropy}(S_v)$

Where S is the current dataset, A is the attribute used to split the dataset, S_v is the subset of S for a given value of A, $|S_v|$ is the number of instances in S_v, and |S| is the number of instances in S (Podgorelec et al., 2002).

Entropy: This equation calculates the impurity of a dataset. It is calculated as the sum of the probability of each class i multiplied by the log base 2 of the probability of the class i. The equation is as follows:

$\mathrm{Entropy}(S) = -\Sigma\ p(i|S) * \log 2(p(i|S))$

Where S is the dataset, i is the class, and $p(i|S)$ is the probability of class i in dataset S (Podgorelec et al., 2002).

These equations are used to calculate the information gain and entropy at each decision point in the decision tree algorithm. The input variable with the highest information gain is selected to split the data, reducing the entropy and making the decision tree more accurate (Breiman; 2001; Podgorelec et al., 2002).

Keyword: **Logistic regression**

Logistic regression is a type of binary classification algorithm. It is a type of statistical analysis used to predict the outcome of a categorical dependent variable (e.g., yes/no) based on one or more independent variables (Saravanan, and Sujatha, 2018). It works by modelling the probability of the dependent variable based on the values of the independent variables

In logistic regression, the dependent variable is modelled using a logistic function (also known as a sigmoid function), which maps any real-valued number into a value between 0 and 1. This value represents the probability of the dependent variable being in a particular category. The logistic function is defined as follows:

p = 1 / (1 + e^-(b0 + b1*x1* + *b2*x2 + ... + bk*xk))

where p is the probability of the dependent variable being in category 1, e is the base of the natural logarithm (approximately 2.71828), b0, b1, b2, ..., bk are the coefficients of the logistic regression equation x1, x2, ..., xk are the values of the independent variables (Saravanan, and Sujatha, 2018).

The logistic regression equation is estimated using maximum likelihood estimation. The coefficients of the equation are chosen to maximize the likelihood of observing the actual dependent variable values given the values of the independent variables (Kleinbaum et al, 2002).

Once the logistic regression equation has been estimated, it can be used to predict the probability of the dependent variable being in category 1 for any set of values of the independent variables. A threshold can then be set to convert the probability into a binary outcome (e.g., if the probability is greater than 0.5, predict category 1, otherwise predict category 0).

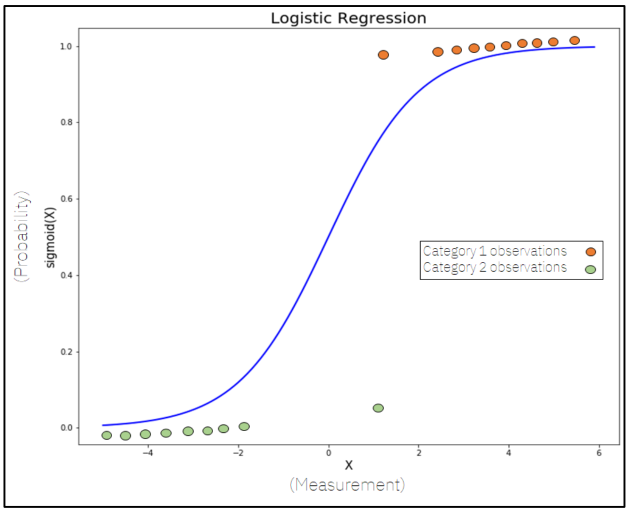


Figure 5. An example of visualisation of logistic regression.
(https://towardsdatascience.com/logistic-regression-explained-9ee73cede081)

Keyword: **Support Vector machine**

Support vector machines (SVM) is one of the powerful machine learning algorithms used for both classification and regression analysis, it also widely used in many applications such as image and text classification. It works by finding the hyperplane that best separates the data into different classes. The SVM algorithm works by first mapping the input data into a higher-dimensional feature space, where it becomes easier to separate the classes using a hyperplane. The hyperplane is chosen to maximize the margin between the classes, where the margin is defined as the distance between the hyperplane and the nearest data points from each class. The points closest to the hyperplane are called support vectors (Westreich, Lessler, and Funk, 2010).

The SVM algorithm has a parameter called C, which controls the trade-off between maximizing the margin and minimizing the classification error. A higher value of C will result in a smaller margin but fewer misclassifications, while a lower value of C will result in a larger margin but more misclassifications (Westreich, Lessler, and Funk, 2010). There is also a technique called the kernel trick to handle the non-linearly separable data. It maps the input data into a higher-dimensional space, where the data becomes linearly separable. This allows SVM to classify non-linearly separable data by finding a hyperplane in the higher-dimensional feature space (DatabaseCamp, n.d.)

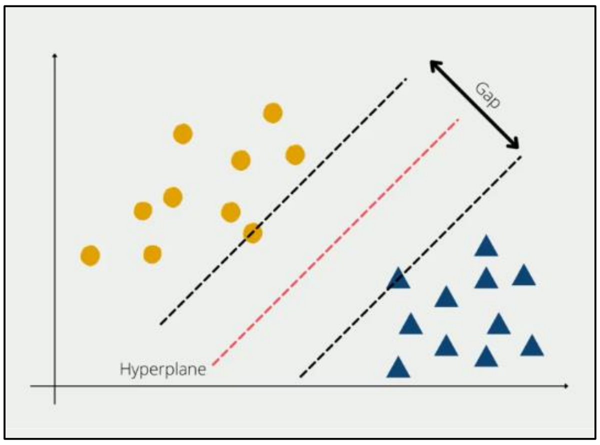


Figure 6. Support Vector Machine Example. (https://databasecamp.de/en/ml/svm-explained)

Keyword: **Extreme Gradient Boosting**

Extreme Gradient Boosting (*XGBoost*) is a popular machine learning algorithm that belongs to the family of gradient boosting algorithms. It is used in machine learning problems, both regression and classification tasks (Wang and Ross, 2018).

It is designed to improve the performance of traditional gradient boosting models by adding some additional features and regularization techniques. XGBoost works by building a series of decision trees and combining their outputs to make a final prediction. During training, the algorithm iteratively fits decision trees to the residuals of the previous trees, with the goal of reducing the overall prediction error (Gupta et al., 2017).

In XGBoost, each tree is trained to approximate the negative gradient of the loss function with respect to the predicted values. This means that each tree is designed to correct the errors made by the previous trees, with the ultimate goal of reducing the overall error (Wang and Ross, 2018). XGBoost has several additional features that make it more powerful than traditional gradient boosting algorithms. For example, it includes a regularization term in the objective function to prevent overfitting, and it uses a technique called column subsampling to reduce the correlation between the trees and improve their diversity. In Addition, it supports parallel processing and has built-in handling for missing values (Gupta et al., 2017).

It also has been used successfully in many real-world applications, such as predicting customer churn, forecasting stock prices, and detecting fraudulent transactions. (Hanif, I., 2020)

2.2.4 Data cleaning and pre-processing

Keyword: Data splitting e.g., balanced and imbalanced, Random state

Data splitting is an important step in the machine learning process that involves dividing a dataset into multiple subsets for training and evaluating a model. The subsets are typically a training set, a validation set, and a test set. The main goal of data splitting is to provide an accurate estimate of the model's performance on unseen data. This is achieved by evaluating the model on subsets that are independent of the training set, which helps prevent overfitting (Gudivada, Apon and Ding, 2017).

There are several methods for data splitting, including balanced data splitting, imbalanced data splitting, random state data splitting, k-fold cross-validation, leave-one-out cross-validation, time series data splitting, and stratified sampling. Each method has its own advantages and limitations, and the choice of method depends on the characteristics of the data and the problem being solved. For example, the balanced data splitting method ensures that the class distribution is roughly the same in all subsets, while the imbalanced data splitting method can be useful when the data is naturally imbalanced (Gudivada, Apon and Ding, 2017). The random state data splitting method is easy to implement but may lead to overfitting or under-fitting. It is important to experiment with different methods and evaluate the performance of the model to select the best one for a particular problem (Kuhn and Johnson, 2013; Bhargava and Agrawal, 2018).

Keyword: Data visualisation (EDA)

Data visualization is an important part of pre-processing in machine learning as it helps to understand and prepare the data for modelling. It presents the data in a way that makes patterns, trends, and relationships easier to identify, allowing for insights into the data that might not be obvious from raw data. These insights inform pre-processing steps like normalizing or scaling the data to improve model performance. Data visualization plays a key role in the pre-processing step of the machine learning pipeline (Alpaydin, 2010).

Keyword: Missing values

Dealing with missing values in machine learning is an important pre-processing step as missing data can significantly impact the performance of machine learning algorithms. There are several methods for addressing missing values, including mean/median/mode imputation, K-Nearest Neighbours imputation, linear interpolation imputation, multiple imputation, regression imputation, model-based imputation, and data deletion (Gudivada, Apon and Ding, 2017). The choice of method will depend on the specific characteristics of the data, the amount of missing data, and the requirements of the problem. The appropriate method should be chosen based on a thorough understanding of the data and the problem, and a combination of methods may be used in some cases to achieve optimal results (Gnat, S., 2021).

Keyword: Outliers

Outliers can have a detrimental effect on the accuracy of machine learning models, as they can significantly skew the distribution of the data and cause a shift in the mean (Alpaydin, 2010). To address this issue, several methods are employed in machine learning, including data cleaning, winsorisation, data transformation, outlier detection, and robust modelling (Gnat, S., 2021). The appropriate method for dealing with outliers will depend on the characteristics of the data, the amount of outlier data, and the requirements of the problem. In some instances, a combination of methods may be used to achieve optimal results (Gudivada, Apon and Ding, 2017).

Keyword: Encoding

The encoding of categorical data involves converting categorical variables into numerical representations that can be used by machine learning algorithms. This is a crucial pre-processing step and there are several methods for encoding including *one-hot* encoding, *label* encoding, *ordinal* encoding, *binary* encoding, *count* encoding, *target* encoding, *and frequency* encoding (Seger, 2018). The method selected depends on the problem specifications and characteristics of the data. Encoding impacts the success of the modelling process and it is important to choose the appropriate encoding method for the data (Alpaydin, 2010; Seger, 2018).

Keyword: Scaling

Scaling refers to the process of transforming variables to a common scale to ensure that all features have equal influence in a machine learning model (Ahsan et al., 2021). It is an important step in pre-processing for many algorithms as the features' range and distribution can impact the performance of the model.

There are several techniques for scaling, including:

1. Standardization: Subtracting the mean and dividing by the standard deviation, which results in a distribution with mean 0 and standard deviation 1. Mathematically, this can be expressed as:

$x' = (x - \mu) / \sigma$

where x is the original feature, x' is the standardized feature, μ is the mean and σ is the standard deviation.

2. Min-Max Scaling: Rescaling the feature values between two specific values by subtracting the minimum value and dividing by the range. The equation can be expressed as:

$x' = (x - xmin) / (xmax - xmin)$

where x is the original feature, x' is the rescaled feature, *xmin* is the minimum value and *xmax* is the maximum value. When the min-max scaling is performed between 0 and 1, it is referred to as normalisation (Ahsan et al., 2021).

2.2.5 Model selection and feature engineering techniques

Keyword: Hyper Parameter Tuning

Hyperparameter tuning is a crucial process in machine learning that involves finding the best combination of hyperparameters for a given dataset (Hertel et al., 2020). Hyperparameters, which are determined before training the model, can significantly affect the model's performance. The process involves selecting a range of values for each hyperparameter, training the model on a subset of data with different combinations of hyperparameters, and evaluating the model's performance on a validation set to find the optimal combination (Kaur, Aggarwal and Rani, 2020). There are various techniques for hyperparameter tuning, such as grid search, random search, Bayesian optimization, and evolutionary algorithms. Hyperparameter tuning can improve model performance and prevent overfitting, but it can also be time-consuming and computationally expensive, especially for complex models. It is an essential part of machine learning model development and should be used in conjunction with other techniques to ensure optimal performance (Kaur, Aggarwal and Rani).

Keyword: Grid search

Grid search is a technique for hyperparameter tuning in machine learning that involves evaluating all possible combinations of hyperparameters in a specified range to find the optimal combination that results in the best model performance (Liashchynskyi and Liashchynskyi, 2019). Hyperparameters are pre-defined settings for a machine learning algorithm that must be specified before training the model. Grid search creates a grid of all possible hyperparameter combinations and trains and evaluates a model for each combination. The performance of each model is evaluated on a validation set, and the best combination of hyperparameters is selected based on the highest performance score. While grid search can be an exhaustive search that guarantees finding the optimal combination, it can be computationally expensive, especially when the number of hyperparameters or their range is large. Grid search is a popular technique for hyperparameter tuning and can be used in conjunction with other techniques to achieve optimal performance (Liashchynskyi and Liashchynskyi, 2019; Bergstra and Bengio, 2012).

Keyword: Random Search

Random search is a hyperparameter tuning technique in machine learning that involves randomly selecting hyperparameters within a specified range and training and evaluating a model for each combination (Bergstra and Bengio, 2012). Hyperparameters are pre-defined settings for a machine learning algorithm that must be specified before training the model. In random search, the hyperparameters are randomly sampled within the specified range, and the model is trained and evaluated on a validation set. The process is repeated for a predefined number of iterations or until a satisfactory model is achieved. Random search is computationally less expensive than grid search because it does not exhaustively search all possible hyperparameter combinations. Random search can explore a larger search space of hyperparameters in a given time frame, and it can often find better performing models than grid search. Random search can be used in conjunction with other hyperparameter tuning techniques to achieve optimal performance (Liashchynskyi and Liashchynskyi, 2019; Bergstra and Bengio, 2012).

Keyword: Cross Validation

Cross-validation is a technique for assessing the performance of a machine learning model by dividing the available data into multiple subsets or folds. In this technique, a model is trained on a subset of the data, and the remaining data is used for validation. This process is repeated for each fold, and the performance is evaluated by computing the average score across all folds (King, Orhobor, and Taylor, 2021). Cross-validation helps to prevent overfitting by providing a more accurate estimate of the model's performance on unseen data. The most common type of cross-validation is k-fold cross-validation, where the data is divided into k equal-sized subsets. The model is trained on k-1 folds and validated on the remaining fold, and this process is repeated k times, with each fold used once for validation. Cross-validation can be used to compare different models or different hyperparameter settings for the same model. It is a widely used technique in machine learning model development and evaluation (Schaffer, 1993; King, Orhobor, and Taylor, 2021).

2.2.6 Evaluation Techniques and Metrics

Keyword: Confusion Matrix

The Confusion Matrix is a tool used to evaluate the performance of a classification model by comparing predicted values to actual values. It comprises four quadrants: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), representing the number of correctly and incorrectly predicted observations as positive or negative (Goetz et al., 2015). Evaluation metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC can be calculated from the quadrants. The Confusion Matrix provides a comprehensive and concise summary of a model's performance, highlighting areas for improvement and strong performance. It can be used for binary classification problems and extended to multi-class classification problems (Bhatore, Mohan and Reddy, 2020).

Keyword: Accuracy Score

Accuracy is a commonly used evaluation metric in machine learning that measures the proportion of correct predictions made by a model. It is calculated by dividing the number of correct predictions by the total number of predictions. The accuracy score ranges from 0 to 1, where 1 represents a perfect model and 0 represents a model that makes only incorrect predictions.

The formula for accuracy is:

Accuracy = (*True Positive* + *True Negative*) / (*True Positive* + *True Negative* + *False Positive* + *False Negative*)

True Positive (*TP*) represents the number of observations that are correctly predicted as positive. *True Negative* (*TN*) represents the number of observations that are correctly predicted as negative. *False Positive* (*FP*) represents the number of observations that are incorrectly predicted as positive. *False Negative* (*FN*) represents the number of observations that are incorrectly predicted as negative (Brett, 2015; Alpaydin, 2010).

Keyword: Precision Score

Precision is a commonly used evaluation metric in machine learning that measures the proportion of true positive predictions made by a model among all positive predictions. It is calculated by dividing the number of true positive predictions by the total number of positive predictions made by the model (Alpaydin, 2010). The precision score ranges from 0 to 1, where 1 represents a perfect precision (all positive predictions are true) and 0 represents a model that only makes incorrect positive predictions.

The formula for precision is:

Precision = *True Positive* / (*True Positive* + *False Positive*)

True Positive (*TP*) represents the number of observations that are correctly predicted as positive. *False Positive* (*FP*) represents the number of observations that are incorrectly predicted as positive.

Precision is particularly useful when the cost of false positives is high or when the goal is to minimize the number of false alarms. For example, in a medical diagnosis system, a high precision score is desired because a false positive can lead to further unnecessary tests or treatments. In contrast to accuracy, precision is not affected by class imbalance, and it's less sensitive to the threshold of the classifier. However, precision alone is not enough to evaluate the model's performance, particularly when the goal is to maximize recall, in such cases, the F1-Score can be used as it's a balance between precision and recall (Bhatore, Mohan and Reddy, 2020).

Keyword: Recall Score

Recall is another commonly used evaluation metric in machine learning, it measures the proportion of true positive predictions made by a model among all actual positive observations. It is calculated by dividing the number of true positive predictions by the total number of actual positive observations (Kuhn and K. Johnson, 2013). The recall score ranges from 0 to 1, where 1 represents a perfect recall (all actual positive observations are predicted as positive) and 0 represents a model that fails to predict any actual positive observations as positive.

The formula for recall is:

Recall = *True Positive* / (*True Positive* + *False Negative*)

True Positive (*TP*) represents the number of observations that are correctly predicted as positive. *False Negative* (*FN*) represents the number of observations that are incorrectly predicted as negative (Guyon and Elisseeff, 2003; Bhatore, Mohan and Reddy, 2020).

Keyword: F1 Score

F1 score is a commonly used evaluation metric in machine learning that is a combination of precision and recall. It is the harmonic mean of precision and recall, and it ranges from 0 to 1, where 1 represents a perfect score and 0 represents a model that makes only incorrect predictions. The F1 score is useful when you want to balance precision and recall (Goetz et al., 2015).

The formula for F1 score is: *F1 score* = 2 * (*Precision* * *Recall*) / (*Precision* + *Recall*)

F1 score gives equal weight to precision and recall, which means that it will be high if both precision and recall are high, and it will be low if either precision or recall is low. It's particularly useful when the data set is imbalanced and in cases when the goal is to maximize both precision and recall.

The F1 score is a good metric to use when the goal is to balance precision and recall, however, it can be affected by the threshold of the classifier, which means that the F1 score can be increased or decreased by changing the threshold. To overcome this limitation, a combination of precision, recall, and F1 score with other evaluation metrics such as ROC-AUC can be used to evaluate the performance of the model (Guyon and Elisseeff, 2003; Kuhn and K. Johnson, 2013).

Keyword: AUC-ROC

AUC-ROC (Area Under the Receiver Operating Characteristic curve) is a commonly used evaluation metric in machine learning, particularly in binary classification problems. It is a measure of a classifier's performance, which represents the ability of a model to distinguish between two classes, i.e., positive and negative. It ranges from 0 to 1, where 1 represents a perfect score and 0 represents a model that makes random predictions.

The AUC-ROC score is calculated by finding the area under the ROC curve, which is a plot of the TPR against the FPR at different classification thresholds. A model with a higher AUC-ROC score has a better ability to distinguish between the positive and negative classes, and therefore it's considered a better model. AUC-ROC is a robust evaluation metric, and it's not affected by the threshold of the classifier, it's useful when the goal is to maximize the true positives while minimizing the false positives. It's worth noting that AUC-ROC is a good evaluation metric when the data is imbalanced, and it's not affected by the threshold of the classifier, it's also useful when the goal is to maximize the true positives while minimizing the false positives (Guyon and Elisseeff, 2003; Bhatore, Mohan and Reddy, 2020).

Keyword: Log loss

Log loss is a logarithmic measure that calculates the difference between the predicted probability of a class and the true label of that class. The advantage of using log loss is that it can capture the uncertainty of the model's predictions, which is especially useful in cases where the model's output is a probability score. Additionally, log loss is a continuous metric that provides a clear indication of the model's accuracy and allows for easy comparison between different models. However, the use of log loss also has some disadvantages. For example, it can be sensitive to outliers, and it does not provide a clear threshold for model performance, making it difficult to interpret in practical terms. Moreover, log loss penalizes misclassifications more severely than correct classifications, which may not be desirable in some applications (Vovk, 2015).

In the table below, the advantages and disadvantages of the evaluation metrics are summarised.

Table 2. Comparison of the evaluation metrics[1]

Evaluation metric	**Advantage**	**Disadvantage**
Confusion Matrix	Provides a clear and concise summary of a model's performance	Only useful for binary classification problems
Accuracy	Easy to understand and widely used as a baseline	Can be affected by class imbalance
Precision	Useful when the cost of false positives is high	Can be affected by class imbalance
Recall	Useful when the cost of false negatives is high	Can be affected by the threshold of the classifier

[1] *Guyon and Elisseeff, 2003; Alpaydin, 2010; Kuhn and K. Johnson, 2013; Goetz et al., 2015; Vovk, 2015; Brett, 2015; Jain and Mahajan, 2016; Bhatore, Mohan and Reddy, 2020*

F1-Score	A balance between precision and recall	Can be affected by the threshold of the classifier
AUC-ROC	A robust evaluation metric not affected by the threshold of the classifier	Not useful for multi-class classification problems
Log loss	It is a widely used evaluation metric for binary classification problems, particularly in situations where the model's output represents the probability of a positive class.	It can be sensitive to outliers and imbalanced datasets. Outliers can significantly affect the log loss score because the logarithm of a very small probability approaches negative infinity, resulting in a high log loss score

3 The Design, The Plan, and The Implementation

3.1 The Design

3.1.1 Identification of chosen algorithms

- ❖ Decision tree: Decision trees have become a widely-used algorithm for predicting customer churn due to several advantages they offer. The interpretability, ability to handle mixed data types, capability to capture complex feature interactions, and robustness to missing data and outliers. The interpretability of decision trees is especially valuable in business contexts, where their predictions can directly influence decision-making. In addition, decision trees can uncover some underlying patterns within the data that may not be apparent with other algorithms, a critical factor in predicting customer churn where early detection is an important factor for retaining valuable customers. The adaptability and versatility of decision trees make them an essential predictive tool for customer churn prediction, with potential implications in different domains beyond business.

- Random Forest: Random forest is a popular algorithm for customer churn prediction due to its ability to provide highly accurate predictions while mitigating the issue of **overfitting**, which is a common problem with **decision trees**. Random forest works by constructing multiple decision trees and aggregating their results to make a final prediction. This approach reduces the likelihood of individual decision trees overfitting the data, as each tree is trained on a randomly selected subset of the dataset. Moreover, random forest can handle both categorical and continuous data, capture interactions between features, and identify important variables for prediction, making it a valuable tool for customer churn prediction. Additionally, the **interpretability** of random forest is similar to that of decision trees, allowing **non-technical stakeholders** to understand the model's output, which is essential in a business setting. The ability of random forest to provide **accurate** predictions, being **interpretable**, and being **robust** makes it a valuable algorithm for customer churn prediction.
- Extreme Gradient Boosting: Extreme Gradient Boosting (XGBoost) has gained popularity as an algorithm for customer churn prediction due to its ability to provide highly accurate predictions with **minimal overfitting**. XGBoost is an ensemble learning method that combines several weak models to create a **strong predictive model**. It uses a gradient boosting framework, which iteratively improves the model's accuracy by **minimizing a cost function**. Furthermore, XGBoost can handle both numerical and categorical data and capture complex feature interactions, making it well-suited for customer churn prediction. Plus, XGBoost has advanced regularization techniques that prevent overfitting, such as L1 and L2 regularization and early stopping. The ability of XGBoost to provide accurate predictions, reducing the risk of overfitting, along with its versatility and robustness, make it a popular choice for customer churn prediction in various domains.
- Logistic regression: Logistic regression is a popular algorithm for customer churn prediction due to its **simplicity**, **interpretability**, and **ability to handle binary classification problems**. Logistic regression works by modelling the probability of a binary outcome using a linear combination of predictor variables. It is easy to interpret the model, which can provide insight into the factors that influence customer churn. Moreover, logistic regression is **computationally efficient** and can handle both numerical and categorical data, making it a versatile algorithm for customer churn prediction. Additionally, logistic regression can be extended to handle multiclass

classification problems, which is useful in situations where customers may be classified into more than two categories.

3.1.2 Identification of chosen evaluation techniques and metrics

The following evaluation metrics are popular for customer churn prediction because they provide a comprehensive assessment of the model's performance in different aspects, including overall accuracy, precision, recall, and the trade-off between them. Providing a comprehensive assessment of the model's performance in different aspects, allowing for a thorough evaluation of the model's effectiveness in predicting customer churn.

- ❖ **Confusion Matrix:** The confusion matrix provides a summary of the model's predictions in terms of true positives, false positives, true negatives, and false negatives, allowing for a detailed evaluation of the model's performance.
- ❖ **Accuracy score:** The accuracy score measures the percentage of correct predictions, making it a straightforward metric for evaluating the model's overall performance.
- ❖ **Precision score:** The precision score measures the proportion of correct positive predictions out of all positive predictions, making it useful for assessing the model's ability to minimize false positives.
- ❖ **Recall score:** The recall score measures the proportion of actual positive cases that the model correctly identified, making it useful for assessing the model's ability to minimize false negatives.
- ❖ **F1 score:** The F1 score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance in terms of precision and recall.
- ❖ **Roc-auc score:** The ROC-AUC score measures the model's ability to distinguish between positive and negative classes, providing insight into the model's sensitivity and specificity.
- ❖ **Log loss score:** The log loss score measures the difference between the predicted probability and the actual outcome, providing a quantitative assessment of the model's calibration.

3.2 Concept solution and plan

The concept solution in this dissertation consists of 4 main steps, outlined below.

Step 1: Literature Review and Algorithm Selection

In the first step, the existing literature on customer churn prediction models are reviewed and the most appropriate algorithms for this research is selected, e.g. decision tree, random forest, and XGBoost. The algorithms are selected based on their performance, popularity, and ease of implementation.

Step 2: Data Collection and Pre-processing

In this step, the required data from the relevant sources are collected and pre-processed to ensure that it is clean and ready for modelling. I will perform various data cleaning and transformation techniques, such as missing data imputation, outlier detection, and feature engineering.

Step 3: Model Implementation and Testing

In the third step, the selected algorithms are implemented and their hyperparameters are tuned using cross-validation techniques. I will evaluate the performance of each algorithm using appropriate metrics such as accuracy, precision, recall, and F1 score. Furthermore, the performance of different algorithms is compared to select the best performing algorithm.

Step 4: Results Analysis and Reporting

In the final step, the results obtained from the previous steps are analysed and the findings are reported. I will identify the most important features that contribute to customer churn and visualize the results using visualization techniques. Finally, I will draw conclusions and provide recommendations based on the findings.

To ensure timely completion of this research, the Gantt chart below which outlines the development steps and timelines for each step of the research is created. This Gantt chart is to be updated regularly to ensure that the project is progressing as planned (find in appendix).

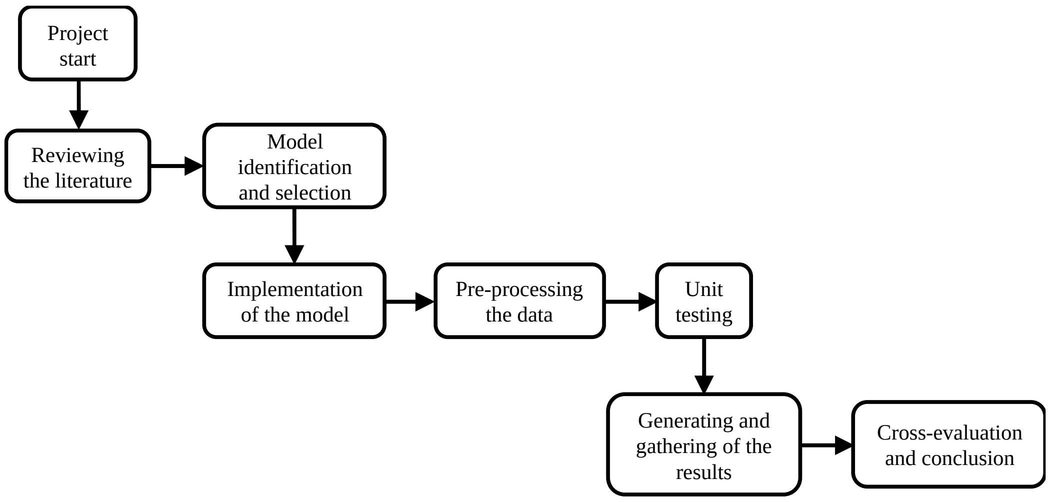

Project start
Reviewing the literature
Model identification and selection
Implementation of the model
Pre-processing the data
Unit testing
Generating and gathering of the results
Cross-evaluation and conclusion

Figure 7. Block diagram showing the planned steps to identify and implement the algorithms

3.3 The Implementation

3.3.1 The Dataset Identification

The dataset available on Kaggle provides information on telecom customers and their churn status, i.e., whether they have cancelled their service or not. This dataset (Blastchar, (n.d.)) includes demographic data, such as the customer's gender and seniority, as well as service usage information, such as the internet service provider used and monthly charges incurred. With a total of 7032 rows and 20 columns, this dataset offers a valuable resource for researchers to study customer behaviour and identify the factors that contribute to customer churn. By analysing this dataset, researchers can gain insights into the characteristics and behaviours of customers who are more likely to churn and develop effective strategies to mitigate customer churn in the telecom industry. The following table is representing the dataset features and target description.

Table 3 The column names and the description of the columns

No.	Name of columns	Description
1	*'customerID'*	Unique identifier for each customer.
2	*'gender'*	Customer gender (male or female).
3	*'SeniorCitizen'*	Whether the customer is a senior citizen or not (1 or 0).
4	*'Partner'*	Whether the customer has a partner or not (Yes or No)
5	*'Dependents'*	Whether the customer has dependents or not (Yes or No).
6	*'tenure'*	Number of months the customer has been with the company.
7	*'PhoneService'*	Whether the customer has a phone service or not (Yes or No)
8	*'MultipleLines'*	Whether the customer has multiple lines or not (Yes, No, or No phone service).
9	*'InternetService'*	Customer’s internet service provider (DSL, Fibre optic, or No).
10	*'OnlineSecurity'*	Whether the customer has online security or not (Yes, No, or No internet service).
11	*'DeviceProtection'*	Whether the customer has device protection or not (Yes, No, or No internet service).

12	*'TechSupport'*	Whether the customer has tech support or not (Yes, No, or No internet service).
13	*'StreamingTV'*	Whether the customer has streaming TV or not (Yes, No, or No internet service).
14	*'StreamingMovies'*	Whether the customer has streaming movies or not (Yes, No, or No internet service).
15	*'Contract'*	The contract term of the customer (Month-to-month, One year, or Two year).
16	*'PaperlessBilling'*	Whether the customer has paperless billing or not (Yes or No).
17	*'PaymentMethod'*	The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), or Credit card (automatic)).
18	*'MonthlyCharges'*	The amount charged to the customer monthly.
19	*'TotalCharges'*	The total amount charged to the customer.
20	*'Churn'*	Whether the customer churned (cancelled the service) or not (Yes or No).

3.3.2 Data splitting

In machine learning, it is crucial to split the data into training, validation, and testing sets to evaluate the performance of a model. The purpose of splitting data is to train the model on one part of the data and evaluate its performance on another independent part (Birba, 2020).

In the current project, the *train_valid_test_split* function is used to split the *df_customer* dataset into training, validation, and testing sets based on the specified proportions. The *train_size*, the *valid_size*, and the *test_size* parameter specifies the percentage of data to use for training, validation, and testing respectively. The '*target'* parameter specifies the target variable or the column that needs to be predicted. The random_state = 42 parameter is used to ensure that the split is reproducible, i.e., it means that the results of the function are consistent and can be replicated across different systems and environments. The training set contains ~80% of the original data, to ensure that the model has enough data to learn, the training set is larger compared to other two sets. In this case, the model has more examples to learn from, to be able to identifying patterns and relationships in the data more accurately. Also, by having a larger training set, the model is less likely to overfit and more likely to learn general patterns rather than memorising data. This can lead to better model performance on new, unseen data (test data).

The validation set contains ∼10%, and the testing set contains ∼10% of the original data. This split ensures that the model is trained on a sufficient amount of data and is evaluated on unseen data. The validation set is used to tune hyper-parameters and make decisions about the model's architecture, while the testing set is used to evaluate the final performance of the model.
Checking for the missing values in the training, validation, and testing data:
Three matrices generate the missing values in the three different subsets. The figures 8,9, and 10 are representing the plots for the x_train, x_val, and the x_test. The msno.matrix() function from the missingno library is used to generate the plot. This plot visualizes the locations and density of missing values in the dataset, where the white lines represent missing values. As there is not any white lines in the following figures, it means that there are no missing values in any of the following subsets in the dataset. This helps to identify patterns in the missing data, and provides an overall view of the missingness in the dataset. The plt.title() function is used to add a title to the plot, and plt.show() displays the plot in the output.

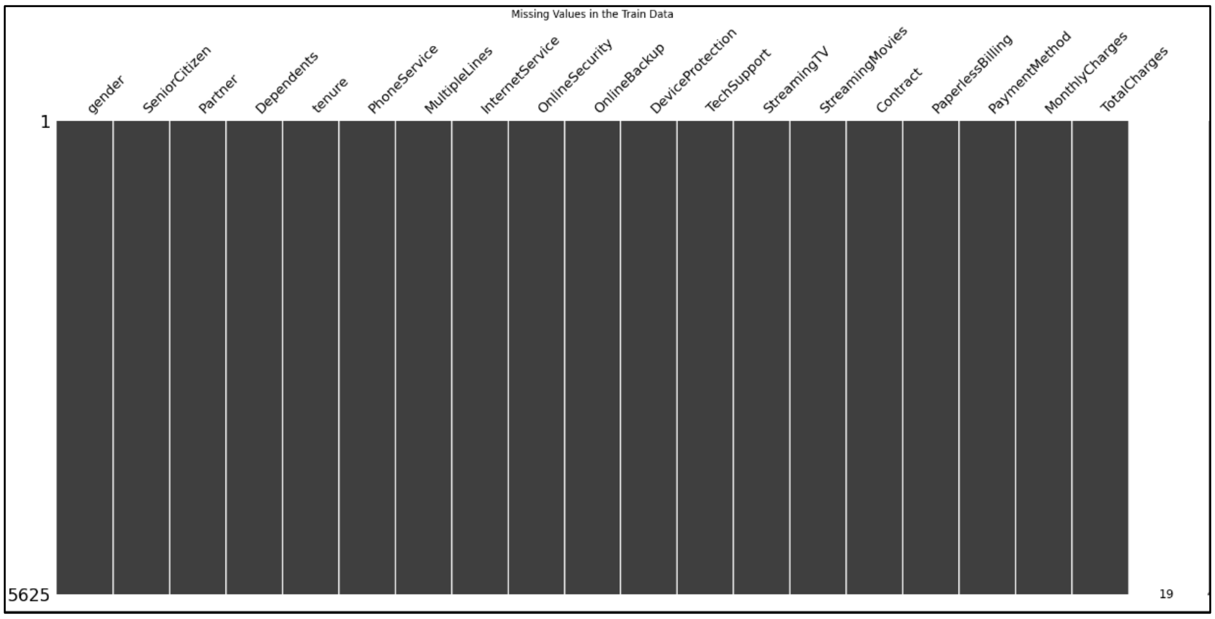


Figure 8. Checking for missing values in the training set

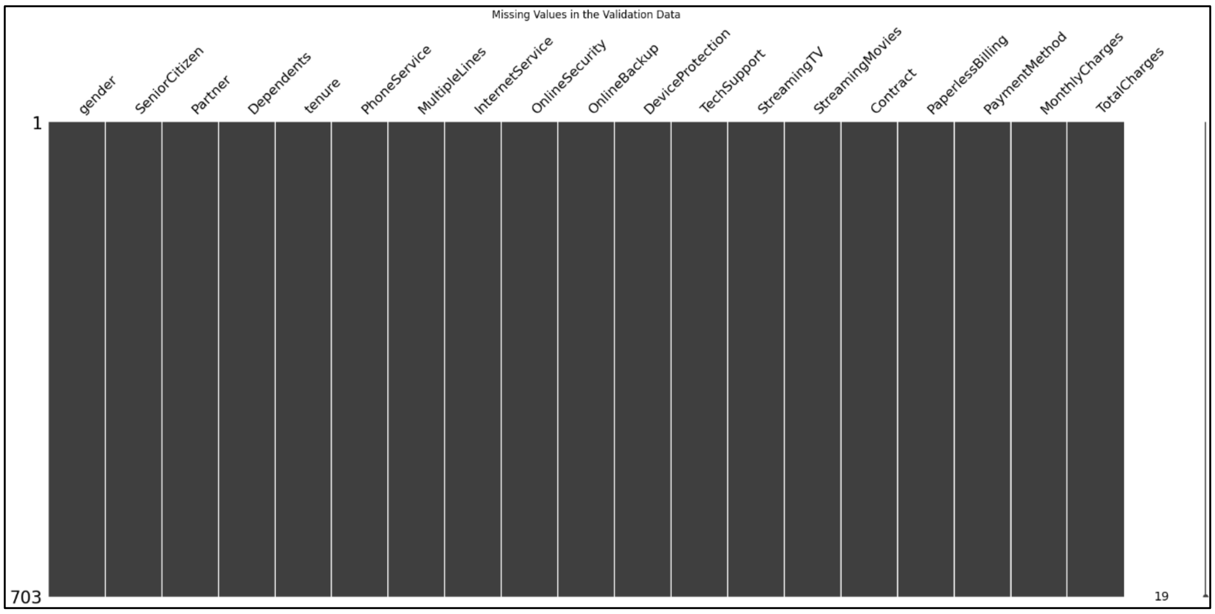


Figure 9. Checking for the missing values in the validation set

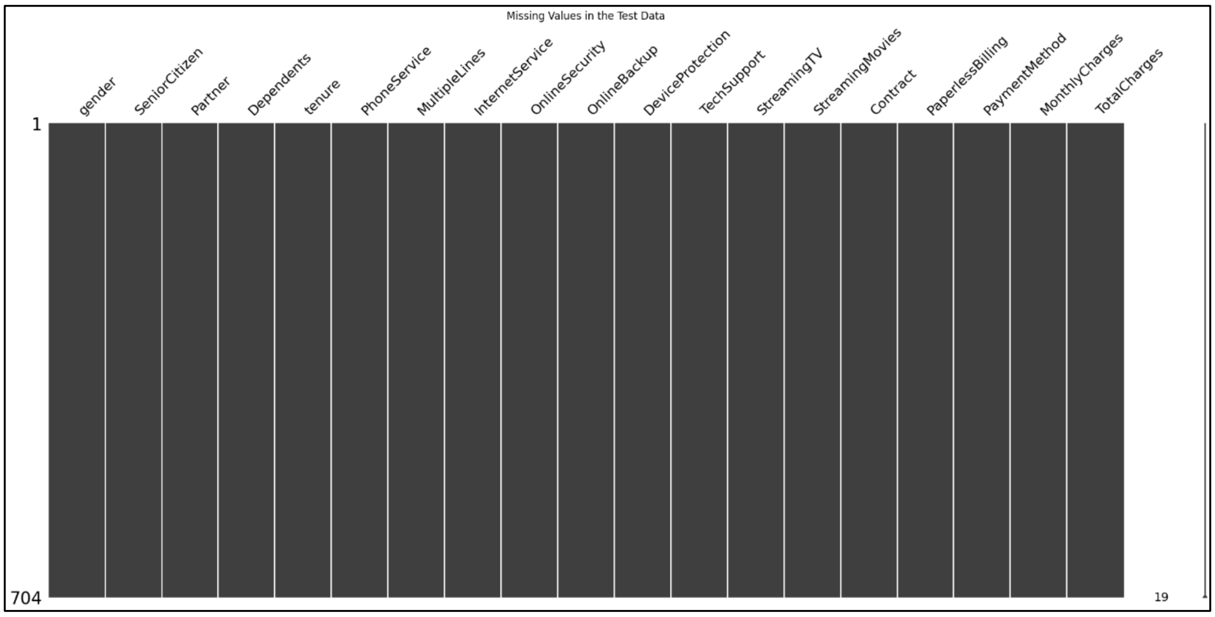


Figure 10. Checking for the missing values in the testing set

3.3.3 Exploratory Data Analysis

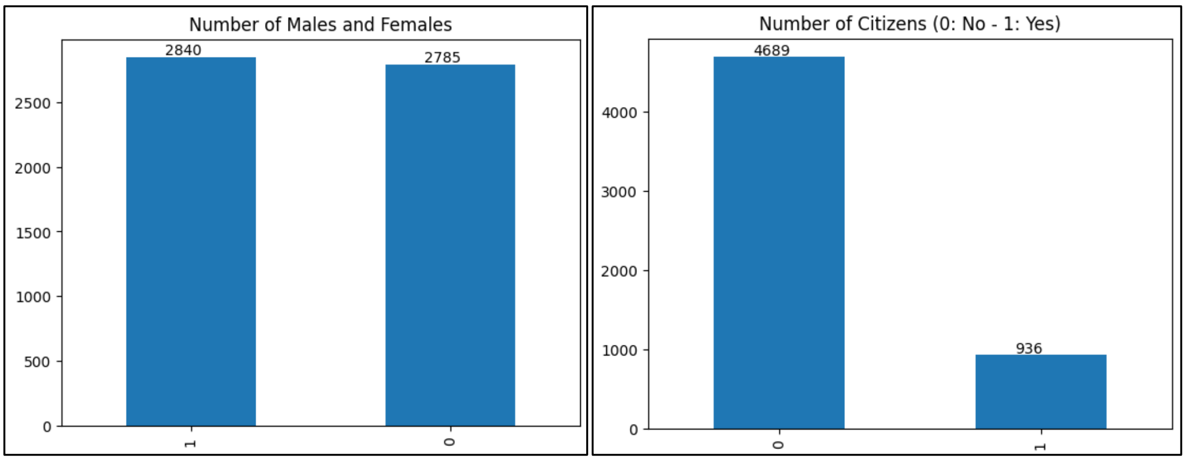


Figure 11. (a) The number of males and females in the dataset. (b) The number of citizens and non-citizens in the dataset.

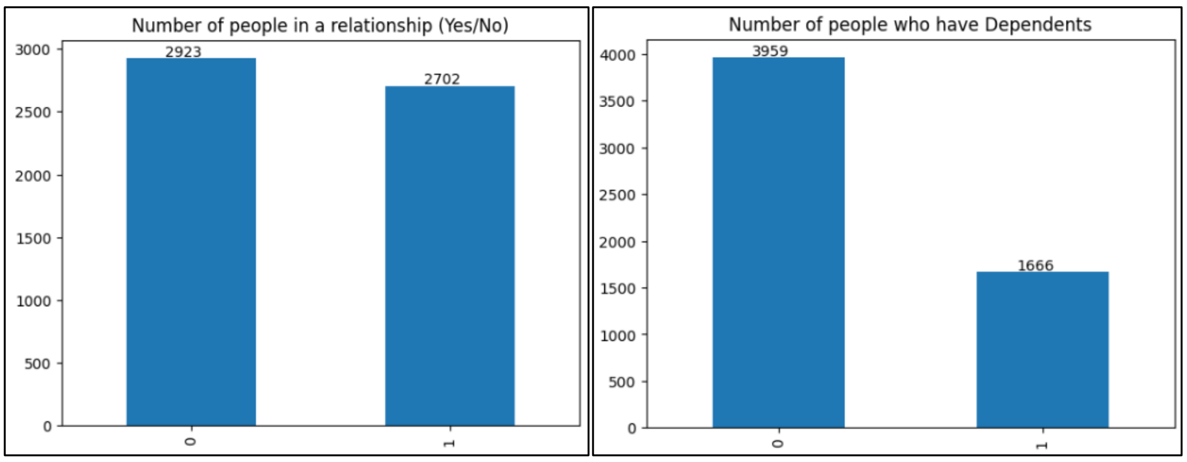


Figure 12. (a) The number of people in a relationship (b) The number of people who have dependents

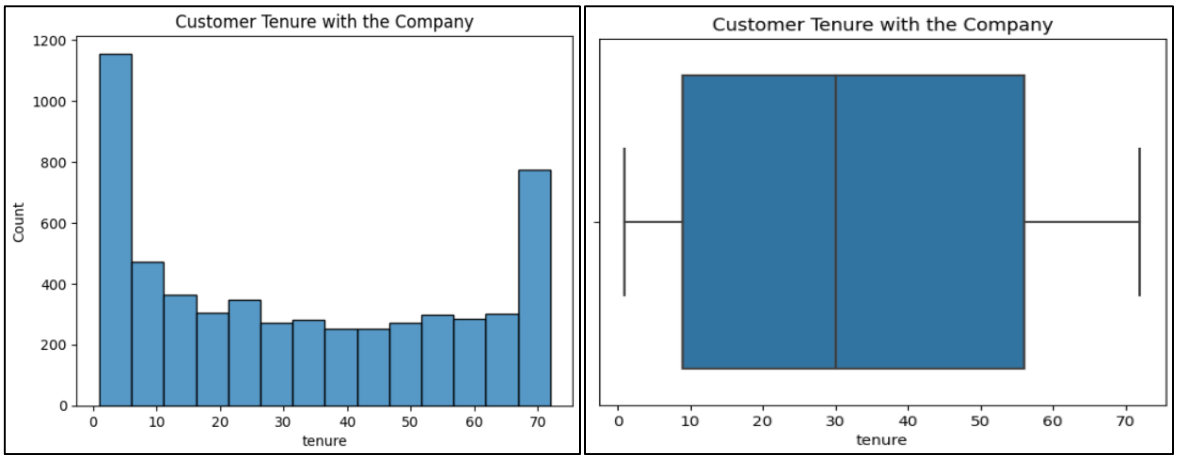


Figure 13. The customer tenure (It shows that how long a customer has been with the company) (a) The histogram of the customer tenure (b) The box and whisker plot of a customer tenure

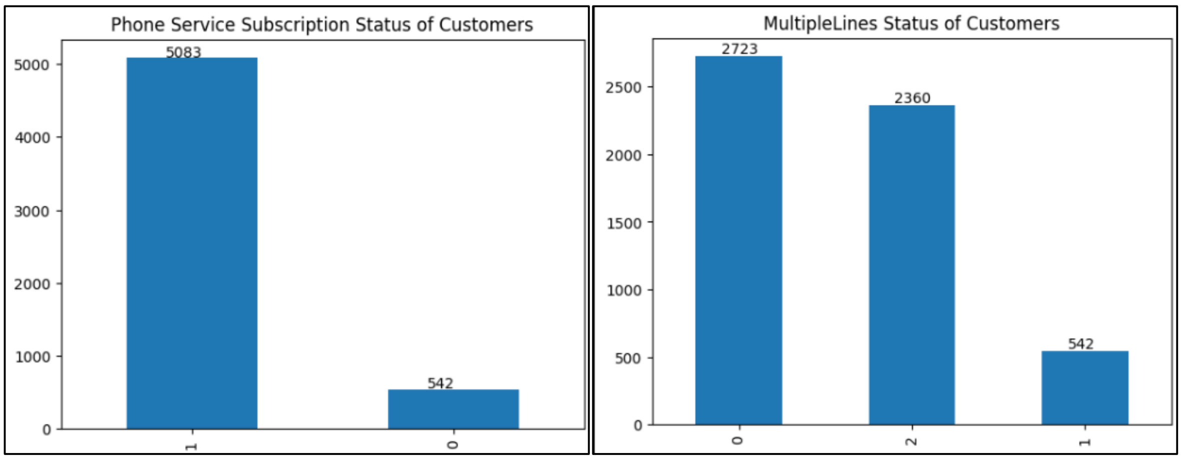


Figure 14. (a) The bar chart of the number of customers subscribed or not subscribed to phone services. (b) The bar plot of the count of each category in the "MultipleLines" column of the x_train DataFrame (MultipleLines : ['Yes' 'No' 'No phone service'])

The following figures represents two bar plots, the distribution of InternetService and OnlineSecurity status among customers in the training set. The value_counts() method is used to count the frequency of each unique value in the columns, and then plot(kind='bar') is used to create the bar plot. The title() method is used to set the title of the plot, and the text() method is used in the for loop to add the count of each category to the top of its corresponding bar, and show() is called to display the plot.

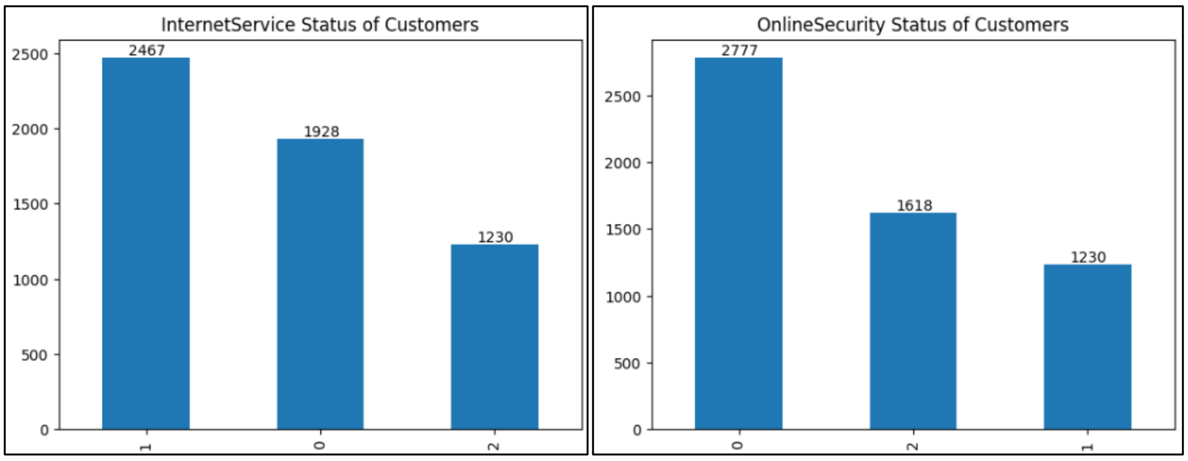


Figure 15. (a) The bar plot of the Internet Service of the customers. (b) The bar plot of the online security service of the customers

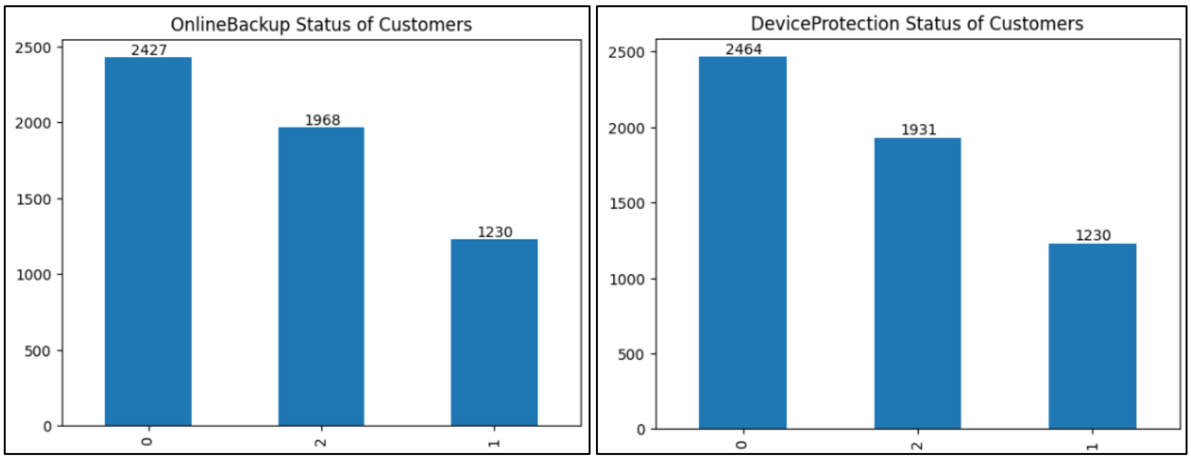


Figure 16. (a) The bar plot of the Online backup service of the customers. (b) The bar plot of the Device Protection Service

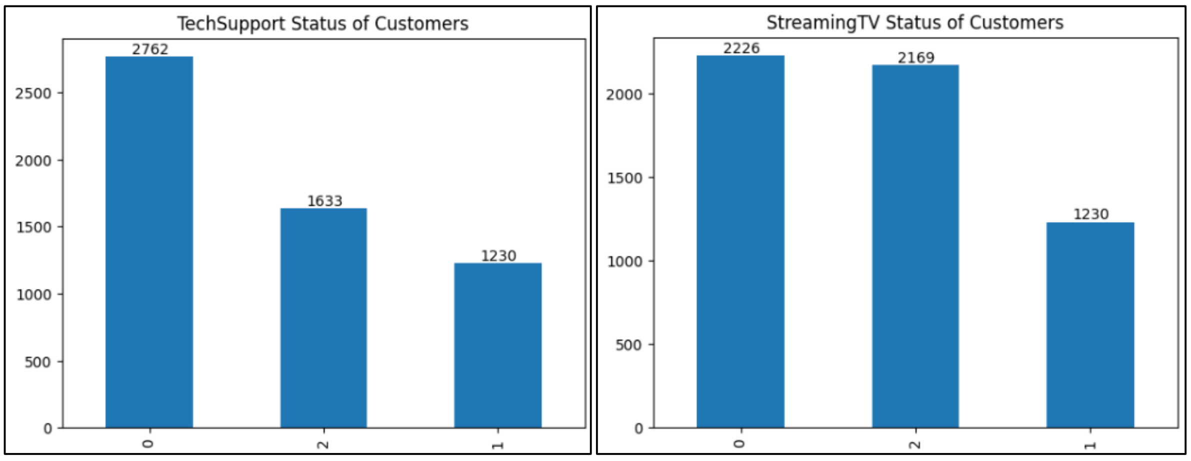


Figure 17. (a) The bar plot of the Tech support service of the customers. (b) The Streaming TV service of the customers

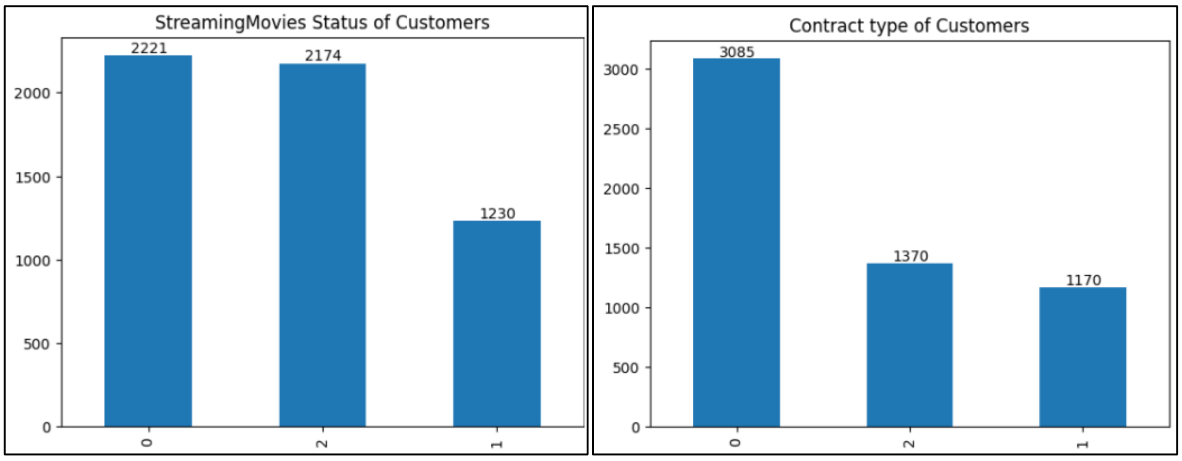


Figure 18. (a) The bar plot of the number of people who has streaming movies service. (b) The bar plot of the contract type of the customer (Contract: ['Month-to-month' 'Two year' 'One year'])

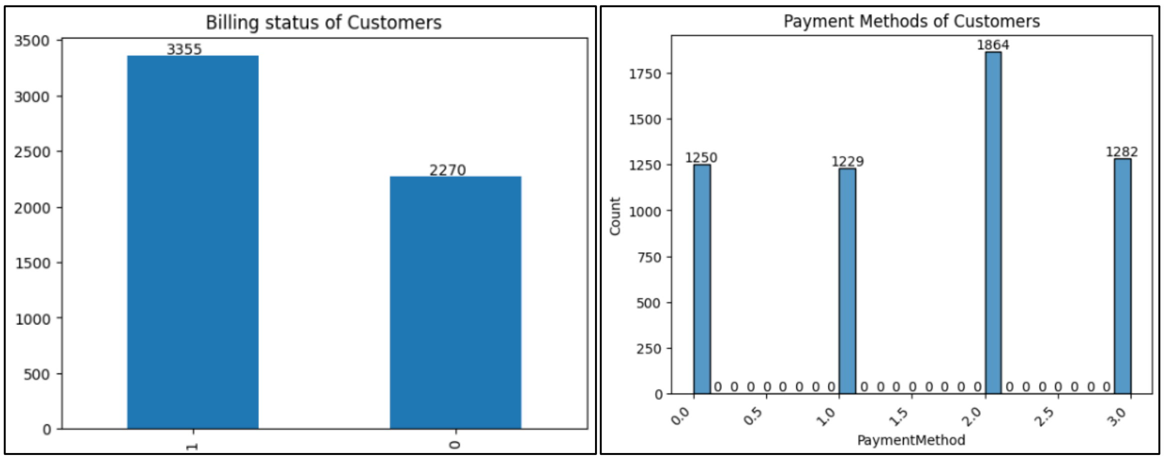


Figure 19. (a) The bar plot of the billing status of the customers (PaperlessBilling: ['Yes' 'No']). (b) The distribution of the payment methods of the customers (['Credit card (automatic)', 'Bank transfer (automatic)', 'Electronic check', 'Mailed check'])

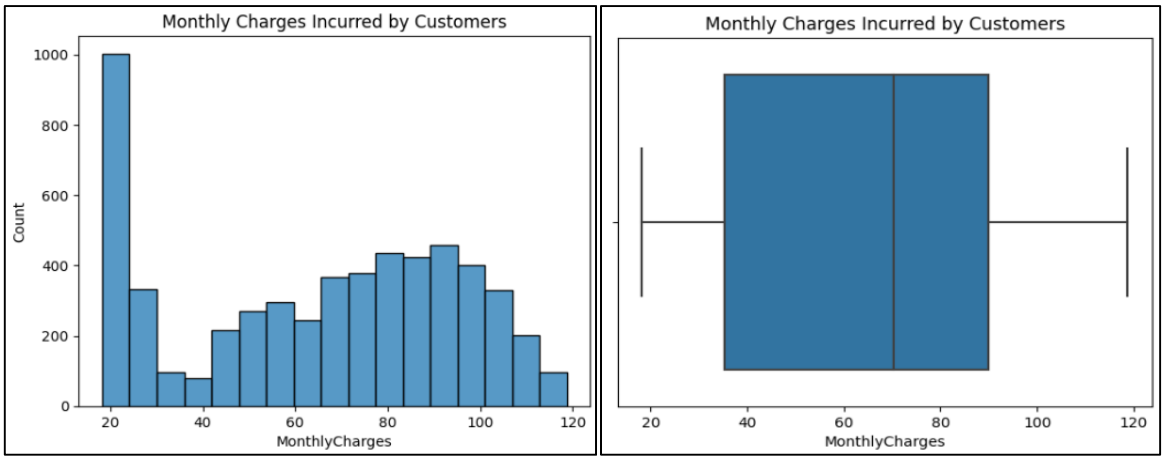


Figure 20. The distribution of the Monthly charges of the customers as a histogram and the box and whisker plot.

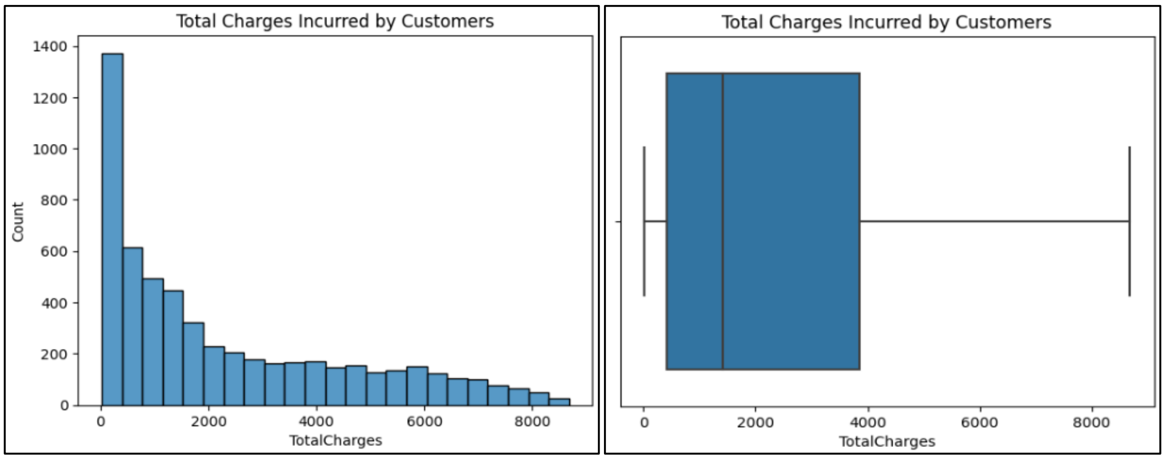


Figure 21. The distribution of the Total charges of the customers as a histogram and the box and whisker plot.

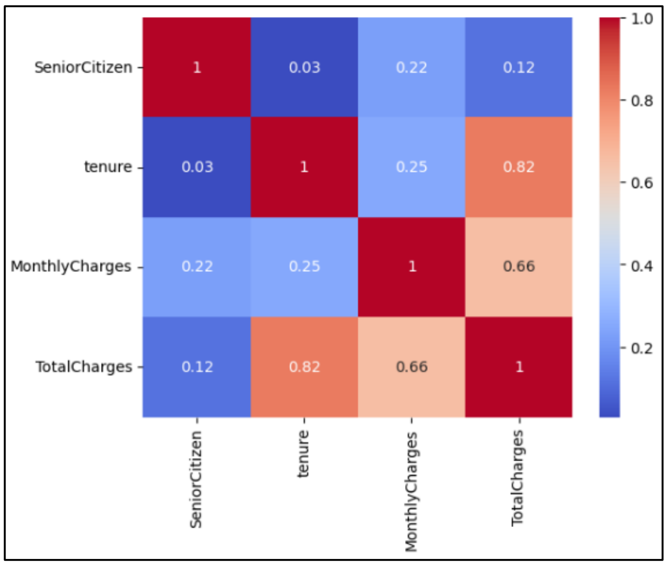


Figure 22. the initial correlation matrix of the numerical values in the dataset

Analysing the features towards the target variable:

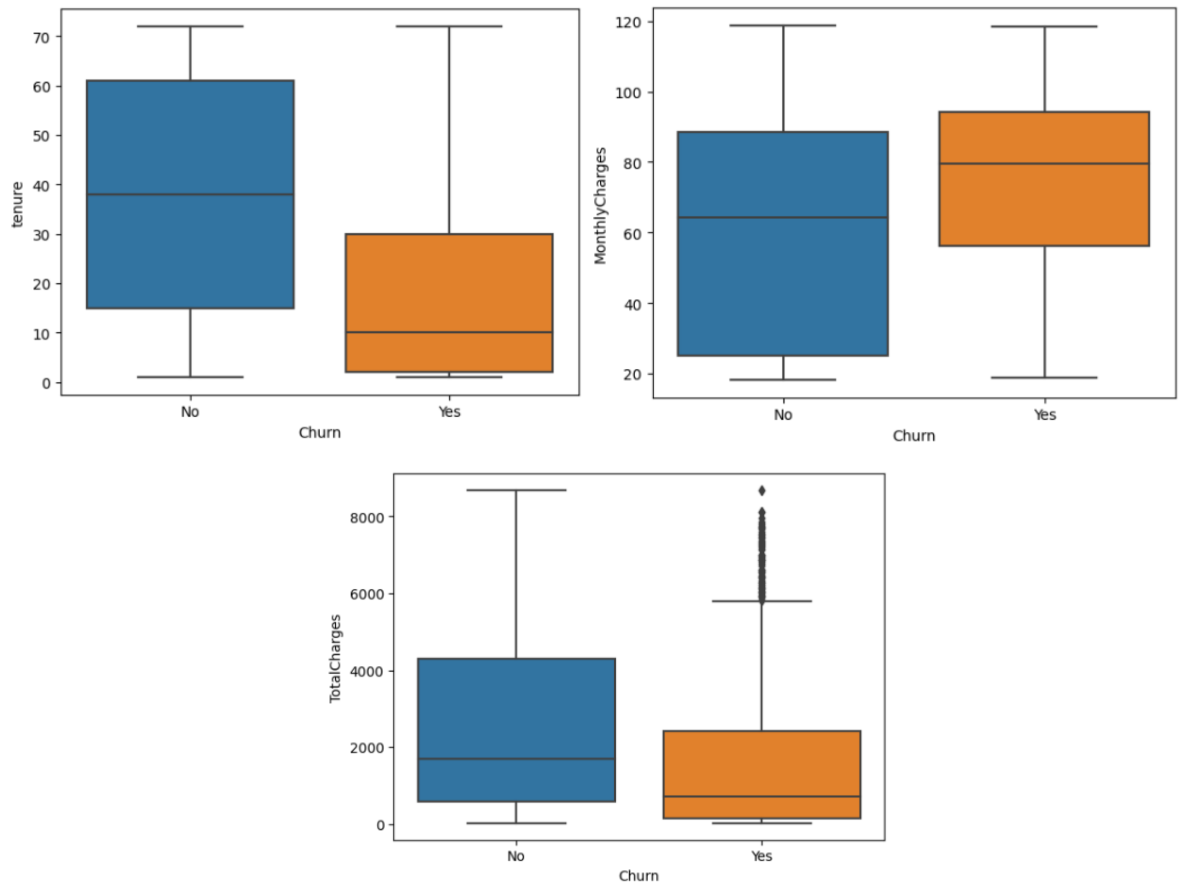
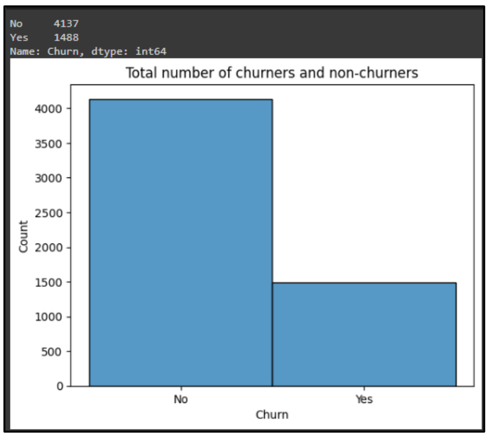


Figure 23. (a)The box plot of the churn vs. tenure feature (b) The box plot of the churn vs. tenure feature (c)

The following figure is a histogram using the seaborn library for the 'Churn' variable (The target variable) in the 'temp_train' dataset, which has two possible values: 'Yes' or 'No'. The count of

'Yes' and 'No' values in the 'Churn' variable, with 'No' having 4137 observations and 'Yes' having 1488 observations.



```
No     4137
Yes    1488
Name: Churn, dtype: int64
```

Figure 24. Analysing the target variable (Churn)

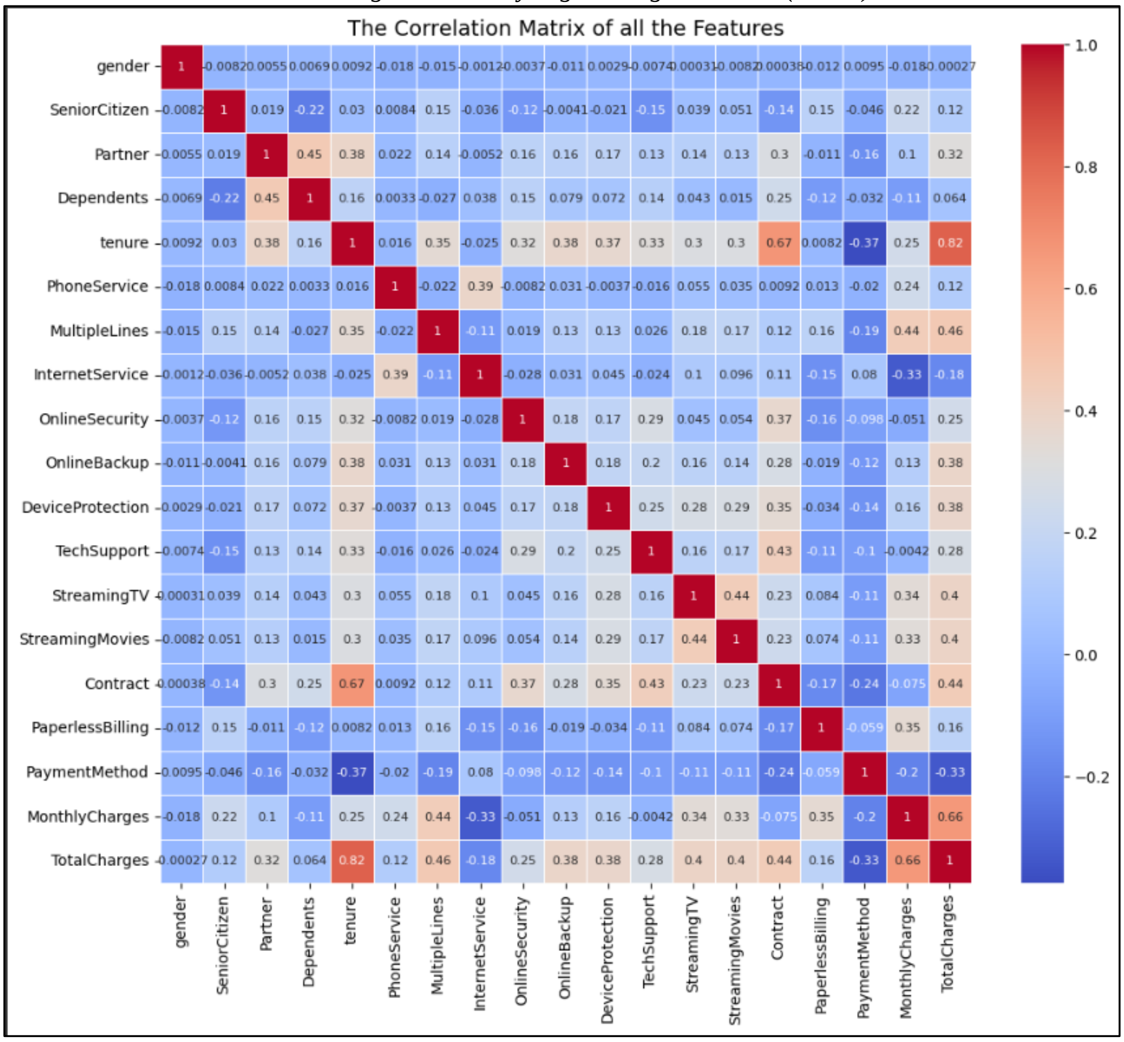


Figure 25. The correlation matrix of all features in the dataset

3.3.4 Data Pre-processing and cleaning

Data pre-processing refers to the set of techniques and processes that are applied to raw data to transform it into a clean, structured format that is suitable for analysis. The goal of data pre-processing is to prepare the data in a way that it can be easily and effectively analysed by machine learning algorithms, statistical models, or other analytical tools.

Data cleaning is removing or correcting errors, duplicates, and missing values in the data. Dropping the 'customerID' column from the x_train, x_val, and x_test DataFrames and modify the DataFrame without this column. The 'axis=1' argument specifies that the column should be dropped, as opposed to a row (which would be axis=0). The customerID column should be dropped from all training, validation, and testing subsets because it is an identifier and does not provide any predictive value for the churn prediction model. Including it in the model could lead to overfitting and reduced performance on unseen data. Since the customerID is unique for each customer, it could also leak information about the training data into the model, which could lead to biased predictions on new data. Therefore, it is good to remove this column before training any predictive models.

Data transformation is converting the data into a standardized format or scale, such as normalization or standardization. Scaling the numerical columns in the training, validation, and testing data using the StandardScaler object from the scikit-learn library. The columns_to_scale list contains the names of the columns with numerical values. Then, the fit method is called on the scaler object to compute the mean and standard deviation of each column to be used for scaling. Finally, the transform method is called on the same scaler object to scale the numerical columns in the training data.

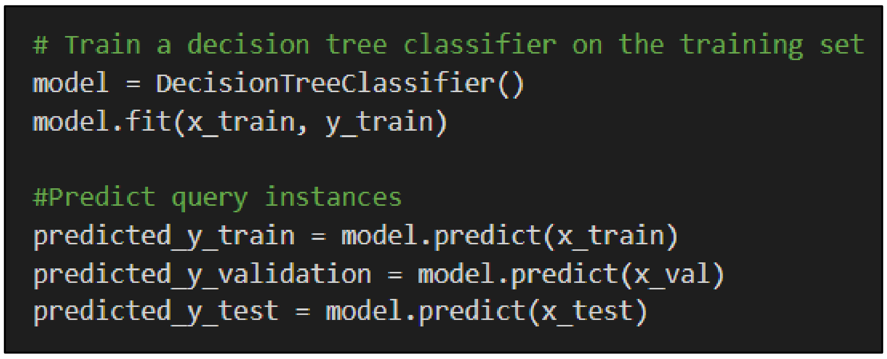
Data reduction is reducing the amount of data by selecting relevant features or applying dimensionality reduction techniques.

Data encoding is converting categorical variables into numerical values. The LabelEncoder is used from sklearn.preprocessing to encode each categorical column in the x_train, x_val, x_test dataframes as numerical values. This is done to convert categorical variables to numerical variables so that they can be used as inputs to machine learning models. The label encoder assigns a unique integer to each category in the column.

Overall, data pre-processing is an essential step in the data analysis pipeline, as it helps to ensure that the data is accurate and complete for further processes. The exact same data pre-processing steps required to be done to the validation and testing data as well as applied on the training data to ensure that the models are tested on data that is similar to the training data. If the same pre-processing does not perform on all the subsets, the models may perform well on the training data but not generalize well to new, unseen data. Therefore, to avoid any bias or inconsistencies, we must perform the exact data pre-processing on all three subsets.

3.3.5 The model selection and Implementation

Training a machine learning model involves fitting the model on the training data and then evaluating its performance on the validation set. In the following code snippet, a decision tree classifier is trained on the training set using the DecisionTreeClassifier() function from the scikit-learn library. The .fit() method is used to fit the model on the training data. After the model is trained, it is used to predict the target variable on the training, validation, and testing sets using the .predict() method. The predicted values are stored in predicted_y_train, predicted_y_validation, and predicted_y_test variables, respectively. These predicted values can be used to evaluate the performance of the model on the different datasets. However, it is important to note that the performance on the training set may not be a good indicator of the performance on the validation and testing sets. Therefore, it is important to evaluate the model's performance on all three datasets to ensure that it generalizes well to new, unseen data.



```python
# Train a decision tree classifier on the training set
model = DecisionTreeClassifier()
model.fit(x_train, y_train)

#Predict query instances
predicted_y_train = model.predict(x_train)
predicted_y_validation = model.predict(x_val)
predicted_y_test = model.predict(x_test)
```

Figure 26. the code snippet of the initial decision tree model including all the features

The shap library is used to calculate and visualize feature importance in a machine learning model. In the following code snippet, a TreeExplainer object is created using the model that was trained on the training set. This object is then used to calculate the SHAP values for each feature in the validation set (x_val). The SHAP values represent the contribution of each feature to the model output. A positive SHAP value means that the feature increased the predicted value for that instance, while a negative SHAP value means that the feature decreased the predicted value. The shap.summary_plot() function is used to visualize the feature importance in a summary plot. This plot shows the top features that contributed the most to the model output for each class. The features are ranked in descending order of their importance. The colour of each feature bar represents the value of that feature for each instance, with red indicating high feature values and blue indicating low feature values. The blue represents class 0 and red represents class 1. Therefore, this plot is showing the impact of each feature on predicting whether a customer will churn (class 1) or not (class 0). The features with the highest absolute SHAP values have the greatest impact on the model's prediction.

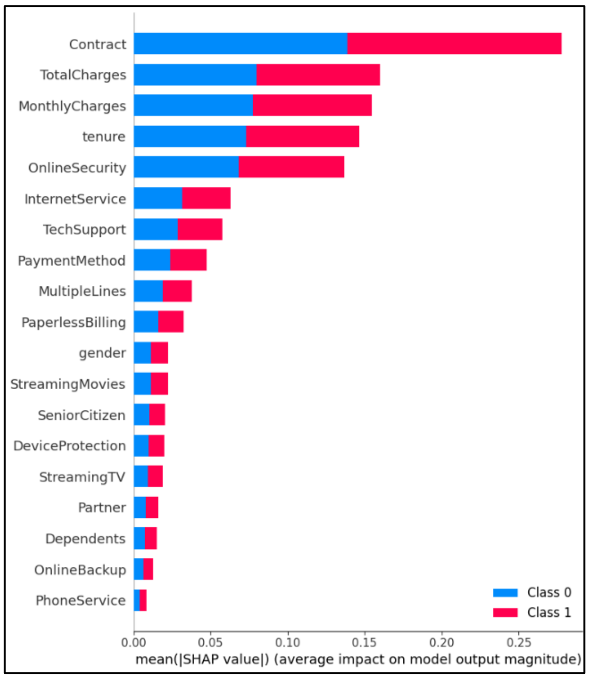


Figure 27. A summary plot of Shapley analysis on the initial model of decision tree

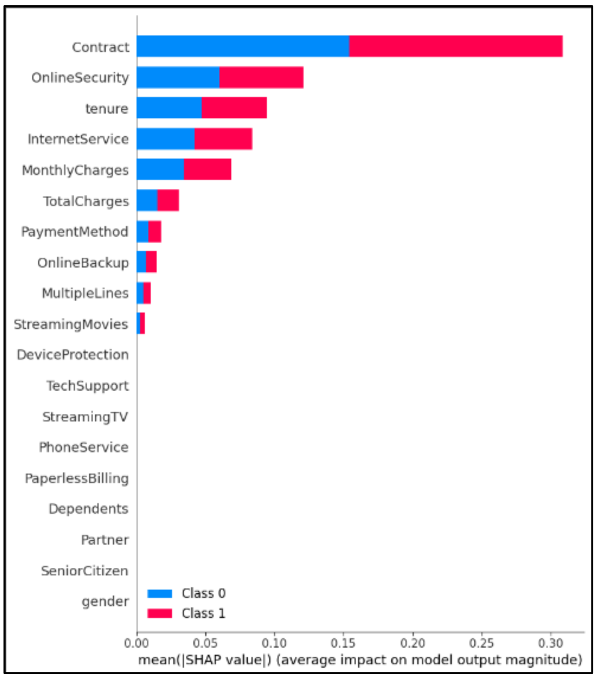


Figure 28. A summary plot of Shapley analysis on the hyperparameter tuned model of decision tree

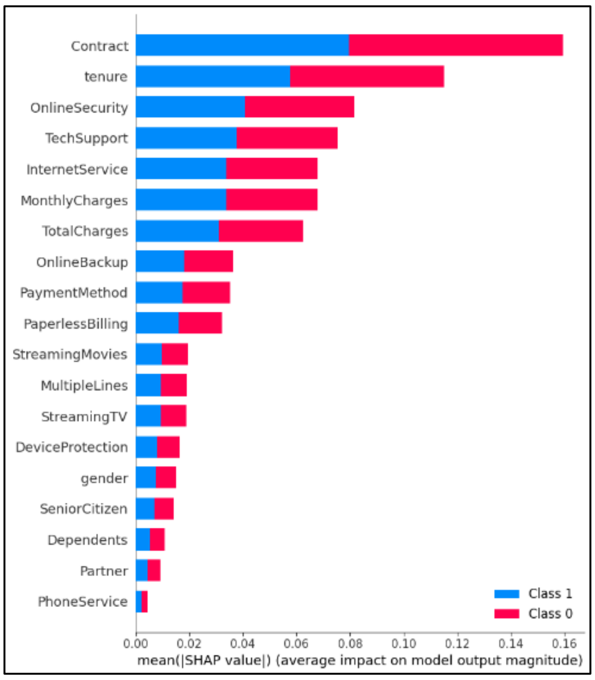


Figure 29. The Shapley Analysis of the initial Random Forest model including all the features

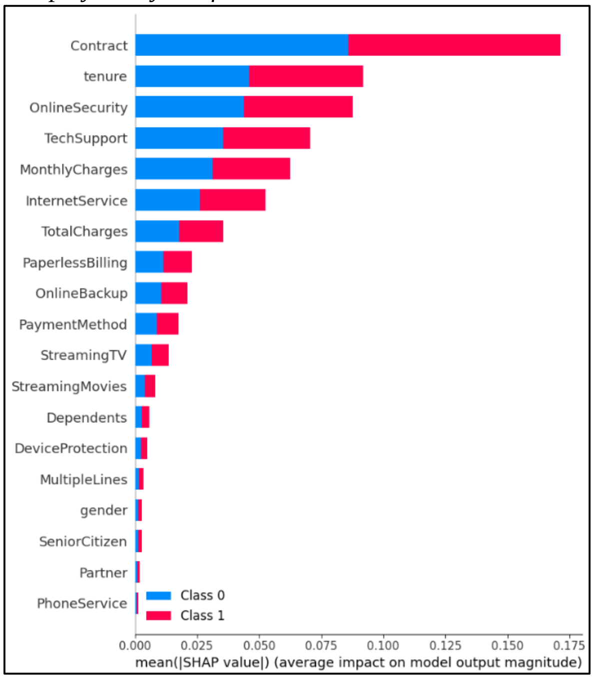


Figure 30. The Summary plot of Shapley Analysis of the Random Forest model including all the features – hyperparameter Tuned Model

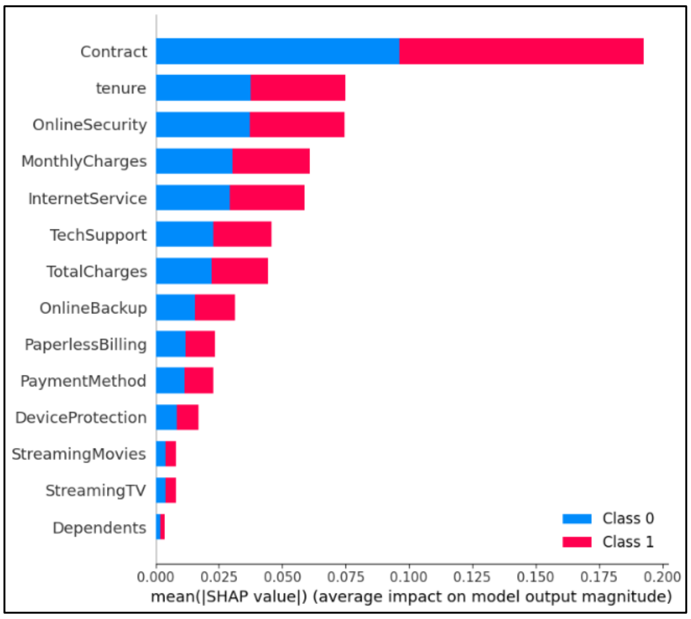


Figure 31. The Summary plot of the Shapley Analysis on the third model –Some of the Features have been dropped due to their contribution

Table 4. The pre-processing and steps have been taken in different models

Model	Pre-processing	Justification
Decision tree – model 1	• 19 features - Dropping one column (*customerID*) • Encoding the categorical values • Shapley Analysis on all the features	*customerID*: The customer id column is used for identifying individual customers and does not provide any meaningful information about their behaviour or characteristics that could be useful in predicting churn. *Label encoding:* Encoded the categorical values (features and the target)
Decision tree – model 2	• 19 features - Dropping one column (*customerID*) • Encoding the categorical values • Grid Search • Hyper parameter Tuning - Criterion - *Max_depth* - *Min_smaples_split* - *Min_samples_leaf*	Grid search: A dictionary called *param_grid* is defined with the hyperparameters to be tuned. The criterion, the *Max_depth*, the *Min_smaples_split*, the *Min_samples_leaf*, and *Max_leaf_nodes* hyperparameters is being tuned, with the values *'gini'* and *'entropy'* for the criterion and a range of numbers for other hyperparameters. A *GridSearchCV* object is created with the machine learning model to be tuned, the *param_grid*

- *Max_leaf_nodes*	dictionary, and the number of cross-validation folds to use (in this case, 5). The *GridSearchCV* object is fit to the training data using the fit method. This will train and evaluate the model for all combinations of hyperparameters in the *param_grid* dictionary. Finally, the best hyperparameters found by the grid search are printed to the console using the *best_params_* attribute of the *GridSearchCV* object. Criterion: entropy is a measure of the average amount of information needed to identify a random sample. *Max_depth*: giving a range of numbers as a dictionary between 1 and 21. *hyperparameters = {'max_depth': range(1, 21)}* *Min_smaples_split:* *param_grid = {* *'min_samples_split':* *[2, 20, 50, 100, 200, 300,* *310, 350, 400, 500, 1000]}* *{'min_samples_split': 300}* *Min_samples_leaf:* *param_grid = {'min_samples_leaf':* *[1, 5, 10, 20, 50, 60, 65, 70]}* *Best min_samples_leaf: {'min_samples_leaf': 60}* *Max_leaf_nodes:* *param_grid = {'max_leaf_nodes':* *[2, 4, 8, 16, 32, 33, 34, 64, None]}* *Best hyperparameters: {'max_leaf_nodes': 32}*

Random Forest – Model 1	• 19 features - Dropping one column (*customerID*) • Encoding the categorical values • Grid Search • Shapley Analysis on all the features	*Label encoding:* Encoded the categorical values (features and the target) A summary plot of Shapley Analysis on all the features A grid search is also done to find the best hyperparameters out of a specific range for n_estimators and max_depth of the random forest model
Random Forest – Model 2	• 19 features - Dropping one column (*customerID*) • Encoding the categorical values • Grid Search • Hyper parameter Tuning - n_estimators - max_depth	The param_grid dictionary specifies the different values of n_estimators and max_depth that the grid search will try. The GridSearchCV object then performs a cross-validation with 5 folds for each combination of hyperparameters and evaluates the performance of the model with the scoring metric that was specified during the model initialization. According to the GridSearchCV results, the best hyperparameters for the random forest on the current model are {'max_depth': 10, 'n_estimators': 20}. This means that the optimal number of decision trees (n_estimators) for the model is 20 and the maximum depth of each decision tree (max_depth) is 10. These hyperparameters were selected based on the cross-validation performance of the model on the training data.
XGBoost - Model 1	• 19 features - Dropping one column (*customerID*) • Encoding the categorical values • Grid Search • Shapley Analysis on all the features	

XGBoost - Model 2		Using GridSearchCV to search over a grid of hyperparameters (n_estimators, max_depth, and learning_rate) for the XGBoost model. The cv parameter specifies the number of cross-validation folds to use. After the grid search is complete, the best hyperparameters found by GridSearchCV are printed, and the best_estimator_ attribute of the GridSearchCV object is used to obtain the best model. Finally, the best_xgb_model is used to make predictions on the training, validation, and test data.
XGBoost - Model 3	• Encoding the categorical values • Grid Search for best hyper parameters • Hyper parameter Tuning • Shapley Analysis (keep the columns with higher contribution)	objective='binary:logistic' specifies the objective function for binary classification problems. random_state=42 sets the random seed for reproducibility purposes. learning_rate=best_params['learning_rate'] sets the learning rate for the algorithm. The learning rate determines the step size at each iteration while moving toward a minimum of a loss function. max_depth=best_params['max_depth'] sets the maximum depth of a tree. This parameter controls the complexity of the model. n_estimators=best_params['n_estimators'] sets the number of trees in the model. Increasing the number of trees generally improves model performance but also increases computation time. The best number of trees is usually found through hyperparameter tuning.

3.4 Limitations and Options

3.4.1 Decision Tree

Some of the limitations of decision trees are explained below.

Overfitting: Decision trees are prone to overfitting, which occurs when the model is too complex and fits the training data too closely. This can lead to poor performance on new, unseen data (Gupta et al., 2017).

Instability: Decision trees can be unstable, meaning that small variations in the data can result in a completely different tree structure. This can make it difficult to interpret the model and make it less reliable (Podgorelec et al., 2002; Gupta et al., 2017).

Sensitivity to small changes: Decision trees are sensitive to small changes in the data, which can lead to different splitting decisions and different tree structures (Gupta et al., 2017).

Some of the options for decision trees are:

Pruning: Pruning is a technique used to reduce the size of the tree and prevent overfitting. It involves removing branches that do not improve the performance of the model on the validation set (Breiman; 2001).

Choosing the splitting criterion: The splitting criterion determines how the tree splits the data at each node. Popular splitting criteria include Gini impurity and entropy (Breiman; 2001; Gupta et al., 2017).

Handling missing values: Decision trees can handle missing values in the data. One approach is to use surrogate splits, which use other features to make the split when the primary feature has missing values.

Ensembles: Decision trees can be combined into ensembles, such as random forests or gradient boosting, to improve their performance and reduce their sensitivity to small changes in the data.

3.4.2 Random Forest

Random Forest is a popular and powerful machine learning algorithm that has a number of limitations and options, outlined below.

Limitations of Random Forest algorithm

- ❖ Random Forest can sometimes overfit to the training data, especially when the number of trees is very large.
- ❖ Random Forest may not be able to accurately predict values outside the range of the training data.
- ❖ Random Forest can be slow to train, especially when the number of trees is large.

- ❖ It can struggle with imbalanced datasets, where one class is much more common than the other (Navada et al., 2011; Charbuty and Abdulazeez, 2021).

Options of Random Forest algorithm

- ❖ The number of decision trees in the forest can be tuned as a hype parameter. More trees generally lead to better performance, but can also make the model slower to train and more prone to overfitting.
- ❖ The maximum depth of each tree is a hyper parameter that limits the depth of each decision tree. Shallower trees are less likely to overfit, but may also be less expressive.
- ❖ The number of features to consider at each split is another hyper parameter that determines the number of features that are randomly sampled at each split in the decision tree. This helps prevent overfitting and can lead to better performance.
- ❖ The criterion for splitting nodes can be used in random forest models. Different criteria for splitting nodes in the decision trees, such as Gini impurity or entropy. The choice of criterion can affect the performance of the model.
- ❖ Random Forest can use different sampling methods for creating the training data for each tree, such as bootstrap sampling or subsampling. These methods can affect the diversity and performance of the forest (Navada et al., 2011; Charbuty and Abdulazeez, 2021).

3.4.3 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a powerful machine learning algorithm for both regression and classification tasks. Like any machine learning algorithm, it has its limitations and options. Some of the limitations of XGBoost are:

- ❖ XGBoost can easily overfit the training data, especially when the dataset is small. This can be mitigated by using regularization techniques such as L1 and L2 regularization.
- ❖ XGBoost can be computationally expensive, especially when the dataset is large or when the number of features is high. This can be mitigated by using parallel processing and sampling techniques.

And some of the options of XGBoost are:

- ❖ XGBoost has several hyperparameters that can be tuned to improve the performance of the model. Some of the important hyperparameters include the **learning rate**, **the maximum depth of the tree**, **the number of trees**, and **the regularization parameters**.

- XGBoost can be accelerated using **GPU**s to further speed up training. This is especially useful when the dataset is large or when the number of features is high.

3.4.4 Logistic Regression

Logistic regression is a widely used algorithm for classification tasks. Some of its limitations are:

- Logistic regression assumes that the decision boundary between the classes is linear. This means that it may not perform well if the classes are not linearly separable.
- Logistic regression is sensitive to outliers, as they can strongly influence the coefficients and cause overfitting.
- If the number of features is very large, logistic regression may not perform well due to the curse of dimensionality.
- Logistic regression assumes that the features are independent of each other. If this assumption is violated, the model may not perform well.

And some of the options are:

- Logistic regression can be regularized to prevent overfitting. L1 and L2 regularization are commonly used.
- Feature engineering can help to improve the performance of logistic regression.
- Logistic regression can be modified to handle imbalanced data by using techniques such as oversampling, undersampling, and cost-sensitive learning.

4 Evaluation

Evaluation is a crucial step in machine learning that involves assessing the model's performance by measuring various metrics such as accuracy, precision, recall, F1 score, ROC AUC score, etc. The ultimate goal is to determine how well the model can make predictions on new data. Evaluation is vital to ensure that the model is reliable and accurate enough to be deployed in real-world scenarios. Techniques such as cross-validation, holdout validation, and bootstrapping can be used for evaluation. The choice of evaluation metrics and techniques should be carefully made to ensure the model's suitability for deployment in real-world scenarios.

4.1 Evaluation Methodology

4.1.1 Evaluation techniques and Metrics

- ❖ Learning Curves

The following plots the learning curves of the Decision tree models and the code snippet, which shows how the accuracy of the model changes as the size of the training set increases.

The train_sizes, train_scores, valid_scores = learning_curve(estimator=model, X=x_train, y=y_train, train_sizes=np.linspace(0.1, 1.0, 10), cv=5, scoring='accuracy', n_jobs=-1) uses the learning_curve() function from scikit-learn to compute the training and cross-validation scores for the model using increasing amounts of training data. The train_sizes parameter specifies the fraction of the dataset used for training, while the cv parameter specifies the number of cross-validation folds. The scoring parameter specifies the scoring metric to use for evaluation, which is set to accuracy in this case. The train_scores_mean = np.mean(train_scores, axis=1) code computes the mean training scores across all cross-validation folds for each training set size. The train_scores_std = np.std(train_scores, axis=1) code computes the standard deviation of the training scores across all cross-validation folds for each training set size.

The valid_scores_mean = np.mean(valid_scores, axis=1) code computes the mean cross-validation scores across all cross-validation folds for each training set size. The valid_scores_std = np.std(valid_scores, axis=1) code computes the standard deviation of the cross-validation scores across all cross-validation folds for each training set size. The plt.plot(train_sizes, train_scores_mean, label='Training score') code plots the mean training scores against the training set size.

The plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean + train_scores_std, alpha=0.1) code fills the area between the upper and lower bounds of the training scores with a light color to indicate the variance in the scores.

The plt.plot(train_sizes, valid_scores_mean, label='Cross-validation score'): This code plots the mean cross-validation scores against the training set size. The plt.fill_between(train_sizes, valid_scores_mean - valid_scores_std, valid_scores_mean + valid_scores_std, alpha=0.1) code fills the area between the upper and lower bounds of the cross-validation scores with a light colour to indicate the variance in the scores. The plt.xlabel('Training set size') code sets the label of the x-axis.

The plt.ylabel('Accuracy') code sets the label of the y-axis. The plt.legend() code adds a legend to the plot. The plt.show() code displays the plot.

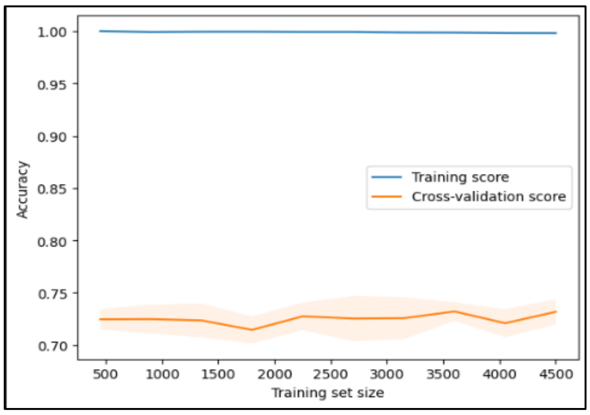


Figure 32. The Learning curve of DT1

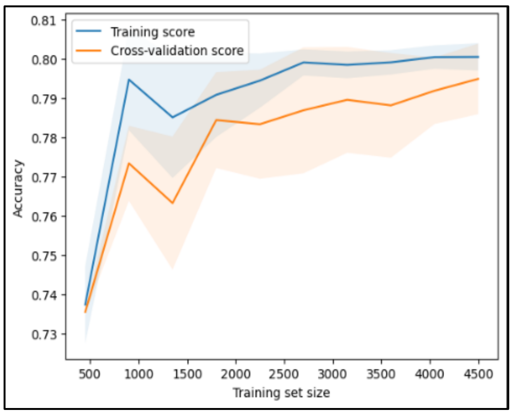


Figure 33. The Learning Curve of DT2

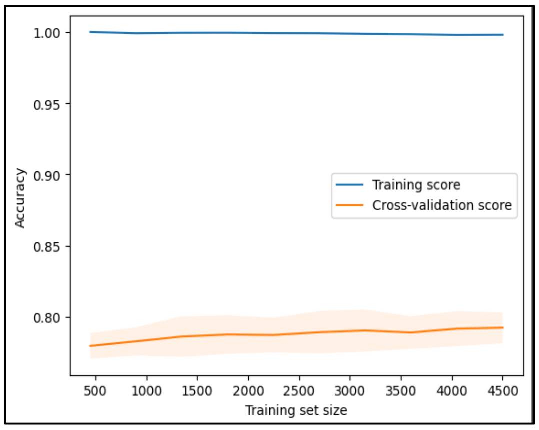


Figure 34. The L Curve of the initial RF Model(HP)

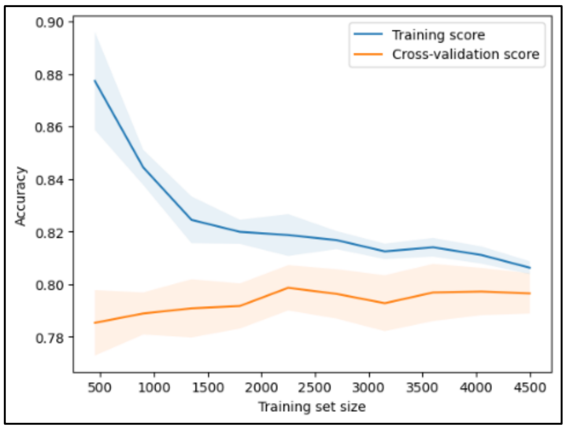


Figure 35. The L Curve of the Second RF

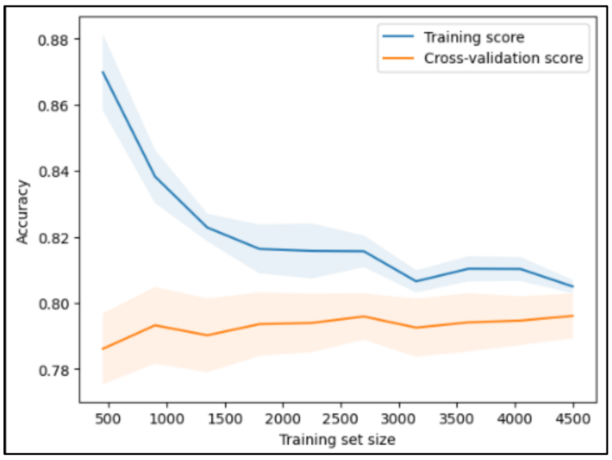


Figure 36. The Learning Curve of the Random Forest – HP and Shapley Analysis

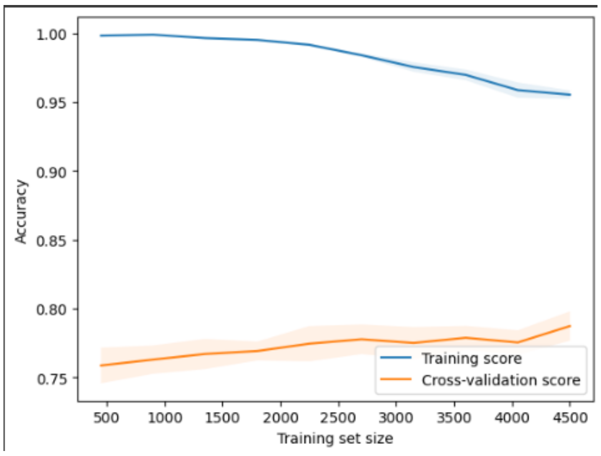


Figure 37. The Learning Curve of the XGboost– Initial Model

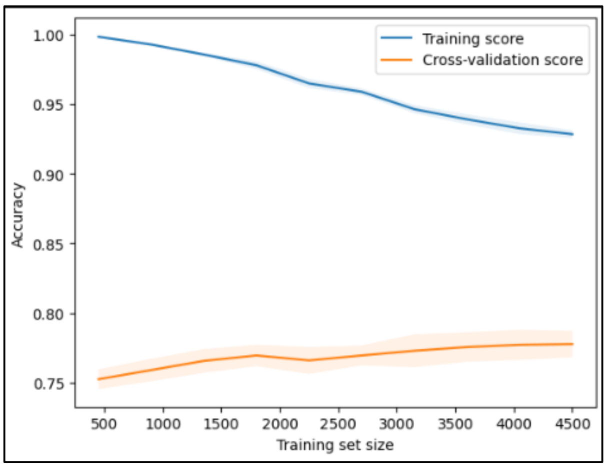


Figure 38. The Learning Curve of the XGboost– 3rd Model

❖ **Cross Validation:**

The cross-validation technique used for the decision tree model, k-fold cross-validation with k=5 (cv=5). The cross_val_score function from sklearn.model_selection module is used to perform cross-validation on the training dataset. The cross-validation scores are the evaluation scores of the model for each fold of the cross-validation. The cross-validation scores are [0.72622222, 0.744, 0.71911111, 0.73155556, 0.74044444]. These scores indicate the performance of the model on five folds of the data.
The mean cross-validation score is the average of the cross-validation scores and the score is 0.73. The standard deviation of the cross-validation scores is also calculated, and it is 0.01. This indicates the degree of variability in the performance of the model across the different folds. The low standard deviation indicates that the model's performance is more consistent across the different partitions of the data. As a result, the model’s (DT1) mean cross-validation score and the standard deviation indicate that the model's performance is consistent and it is expected to generalize well to new, unseen data.

4.2 Results

4.2.1 Decision Tree Results

4.2.1.1 Model 1

The following figures represent that there are 4136 true negative predictions, 1 false positive prediction, 12 false negative predictions, and 1476 true positive predictions in the training set, 412 true negative predictions, 96 false positive predictions, 82 false negative predictions, and 113 true positive predictions in the validation set, 410 true negative predictions, 108 false positive predictions, 90 false negative predictions, and 96 true positive predictions in the testing set.

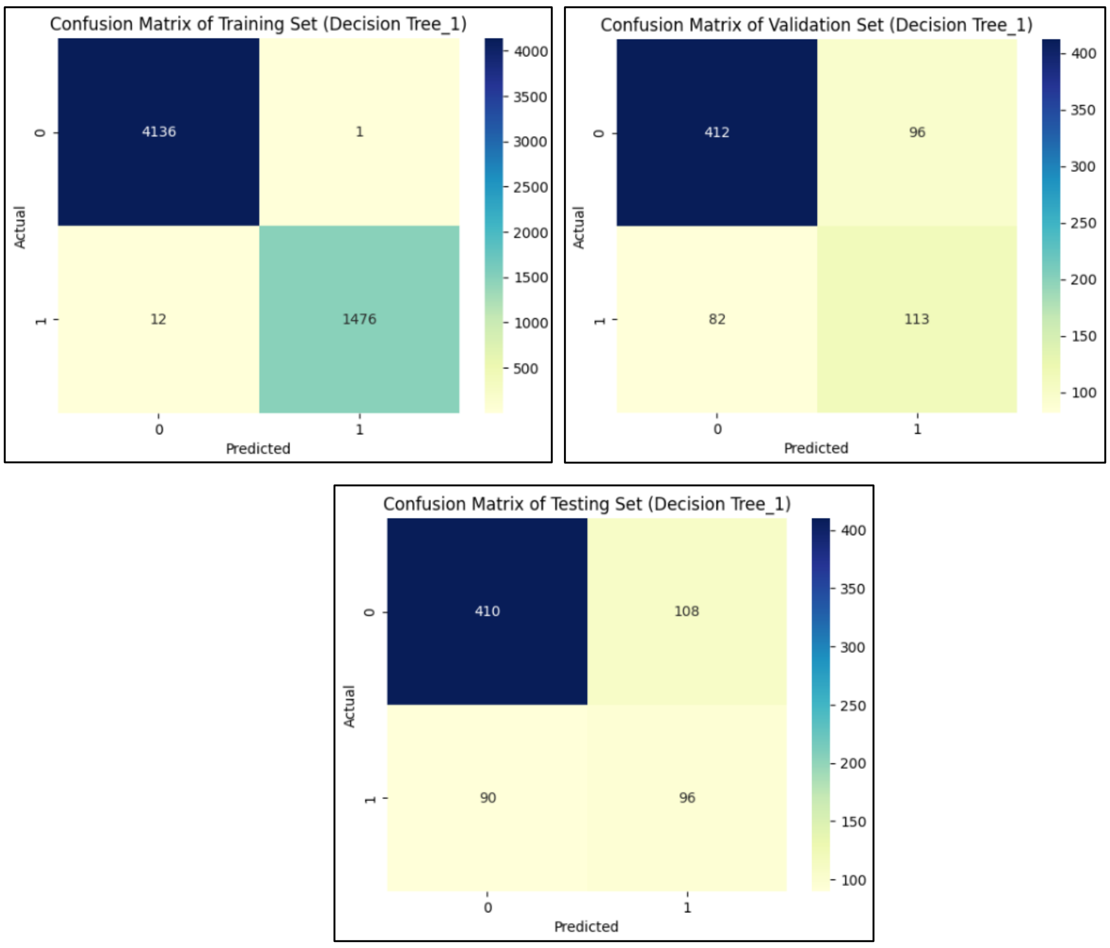


Figure 39. The confusion Matrix of the Training, Validation, and Testing (Decision Tree Results – No scaling)

The following figures represent the performance metrics of the initial decision tree model with all the features of customer churn prediction. Here's what each metric means:

Accuracy measures the proportion of correct predictions out of total predictions. The accuracy scores for training, validation, and testing are 0.997, 0.740, and 0.716, respectively. Precision measures the proportion of true positive predictions out of total positive predictions. The precision scores for training, validation, and testing are 0.999, 0.528, and 0.466, respectively. Recall measures the proportion of true positive predictions out of total actual positive cases. The recall scores for training, validation, and testing are 0.992, 0.579, and 0.511, respectively. F1-score is a harmonic mean of precision and recall. It combines both metrics to provide a better understanding of the overall performance. The F1-scores for training, validation, and testing are 0.996, 0.553, and 0.487, respectively. AUC score measures the area under the receiver operating characteristic (ROC) curve. The AUC scores for training, validation, and testing are 0.996, 0.690, and 0.650, respectively. The closer the AUC score is to 1, the better the model's ability to distinguish between positive and negative classes. Log loss measures the performance of a classification model by penalizing false classifications. The lower the log loss, the better the model's performance. The log loss scores for training, validation, and testing are 0.083, 9.383, and 10.240, respectively.

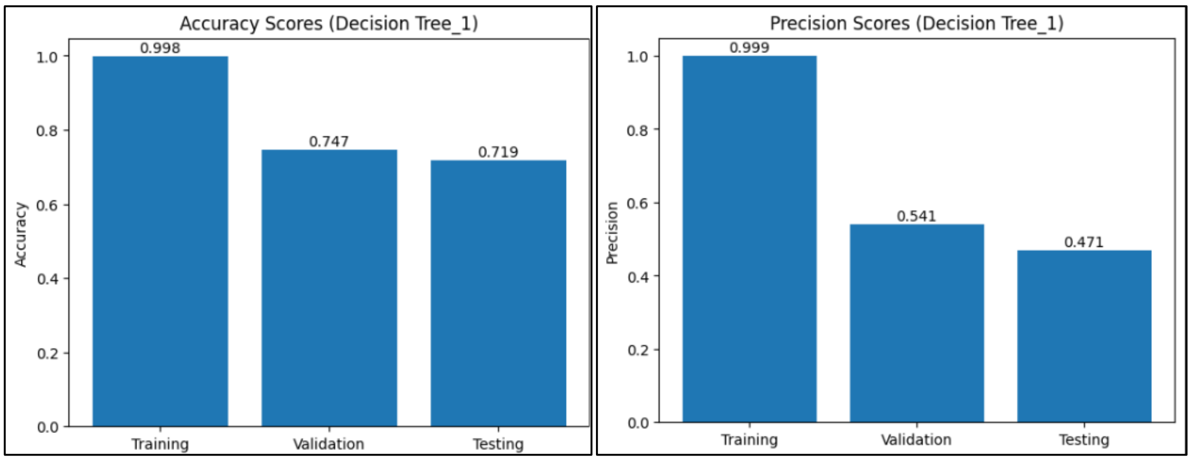


Figure 40. (a) The Accuracy scores of three subsets (DT - With Scaling) – (b) The Precision Scores of three subsets (DT - No scaling)

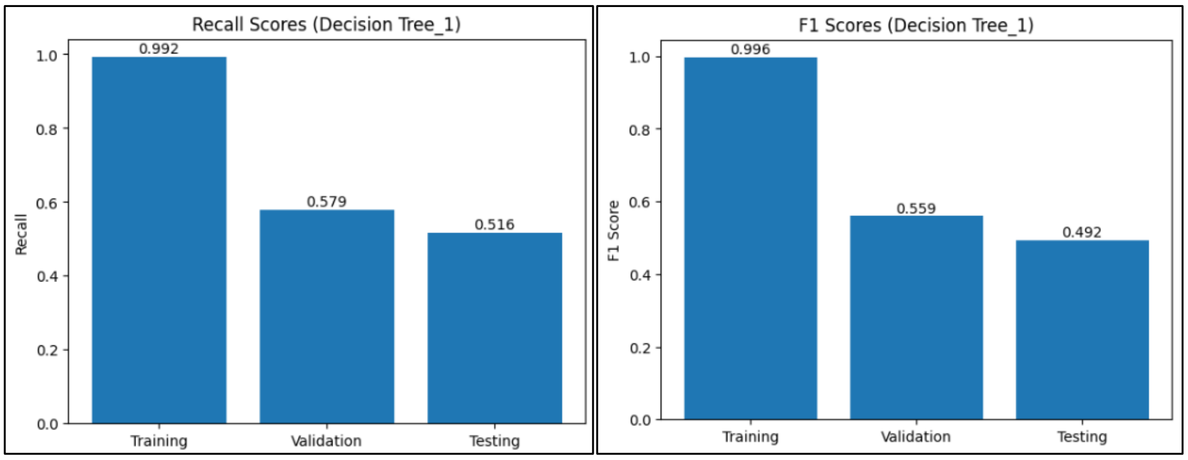


Figure 41. (a) The recall scores of three subsets (DT - With Scaling) – (b) The F1 Scores of three subsets (DT - No scaling)

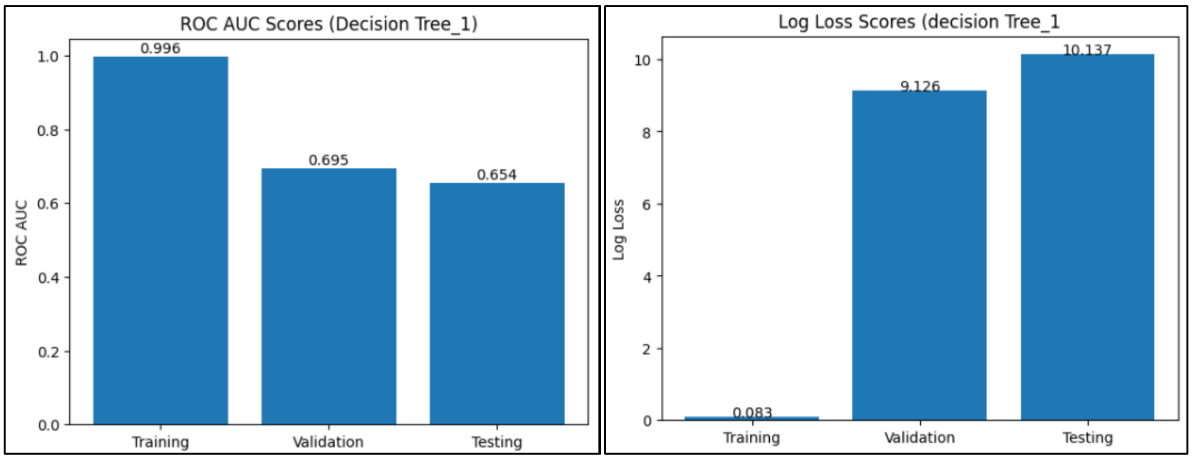


Figure 42. (a) The ROC AUC scores of three subsets (DT - With Scaling) – (b) The Log Loss Scores of three subsets (DT - No scaling)

Based on the evaluation metrics present in the figures, the initial decision tree model has good performance on the training set with high accuracy, precision, recall, F1-score, and AUC-score. However, the model is overfitting on the training data, as seen by the significant difference between the accuracy score of the training set and the validation and testing sets. Also, the precision, recall, and F1-score are much lower on the validation and testing sets compared to the training set, indicating that the model is not generalizing well to new data. The AUC-score is also relatively low on the validation and testing sets, suggesting that the model's ability to distinguish between positive and negative classes is not strong. Furthermore, the log loss scores for the validation and testing sets are high, which is not desirable, as lower log loss scores indicate better predictions. Therefore, the current model needs improvement to better generalize to new data and make more accurate predictions.

4.2.1.2 Model 2 (Hyper Parameters Tuned):

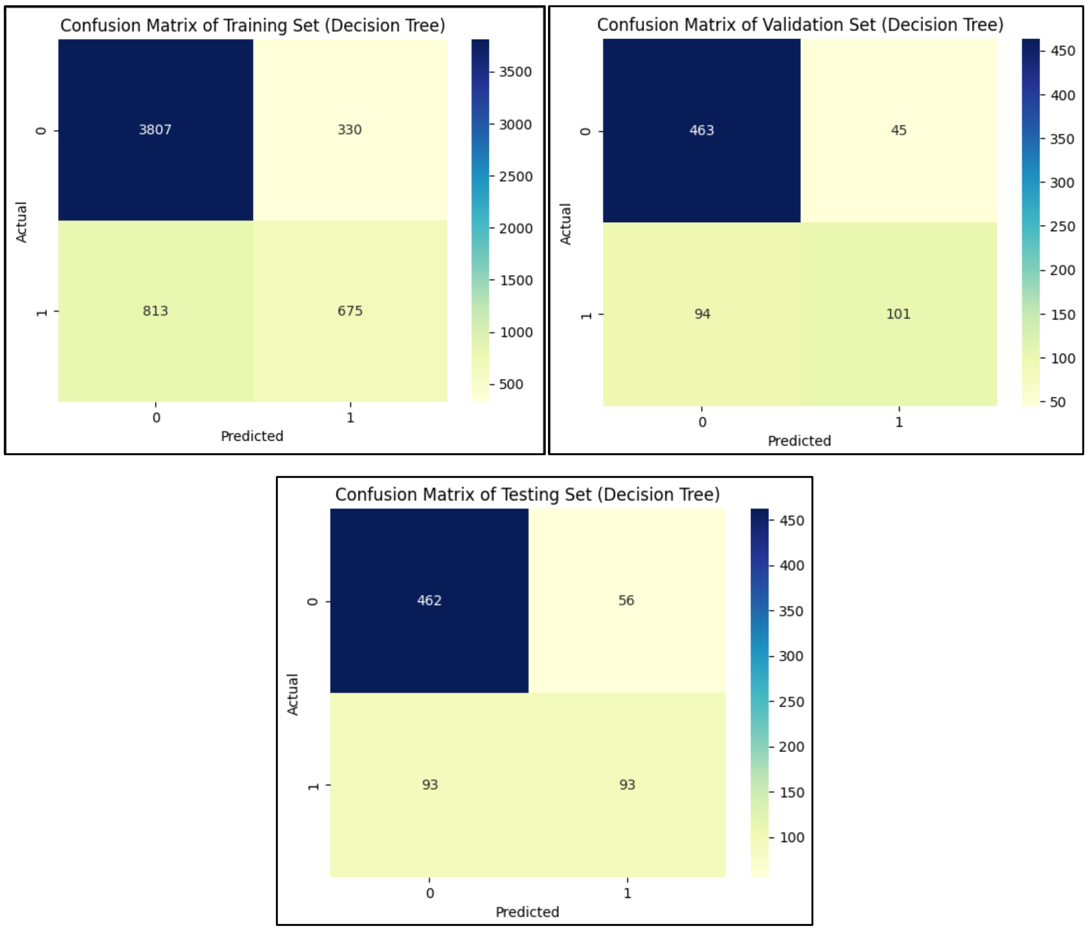


Figure 43. The Confusion Matrix of the Training, Validation, and Testing (Decision Tree – Hyperparamter Tuned)

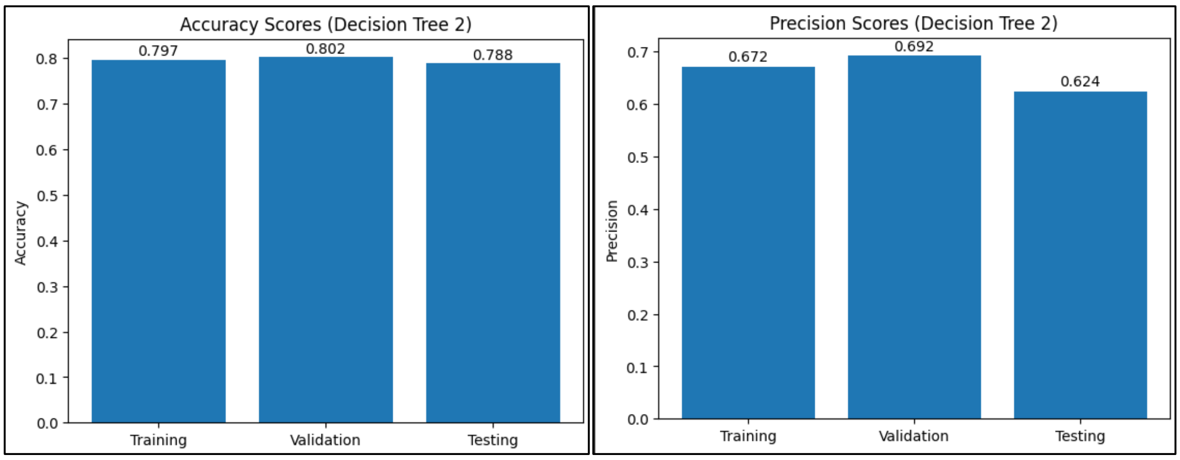


Figure 44. (a) The Accuracy scores of three subsets (DT - With Scaling) – (b) The Precision Scores of three subsets (DT - Hyperparamter Tuned)

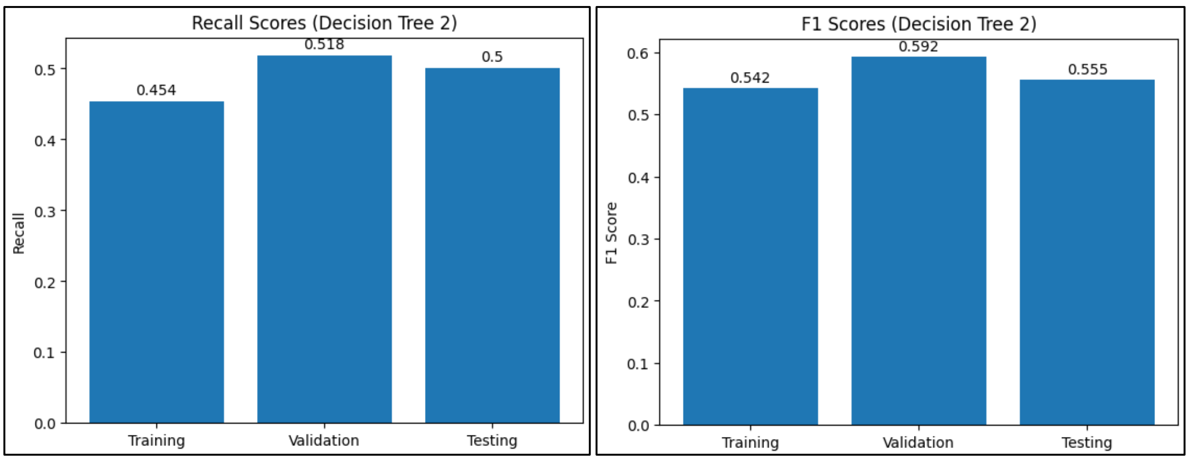


Figure 45. (a) The recall scores of three subsets (DT - With Scaling) – (b) The F1 Scores of three subsets (DT - Hyperparamter Tuned)

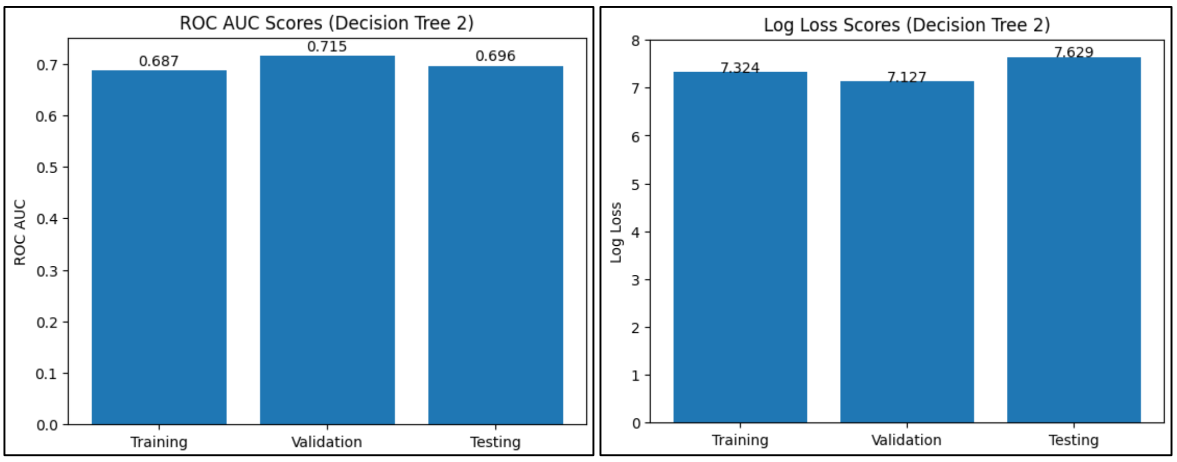


Figure 46. (a) The ROC AUC scores of three subsets (DT - With Scaling) – (b) The Log Loss Scores of three subsets (DT - Hyperparamter Tuned)

4.2.2 Random Forest Results

4.2.2.1 Model 1

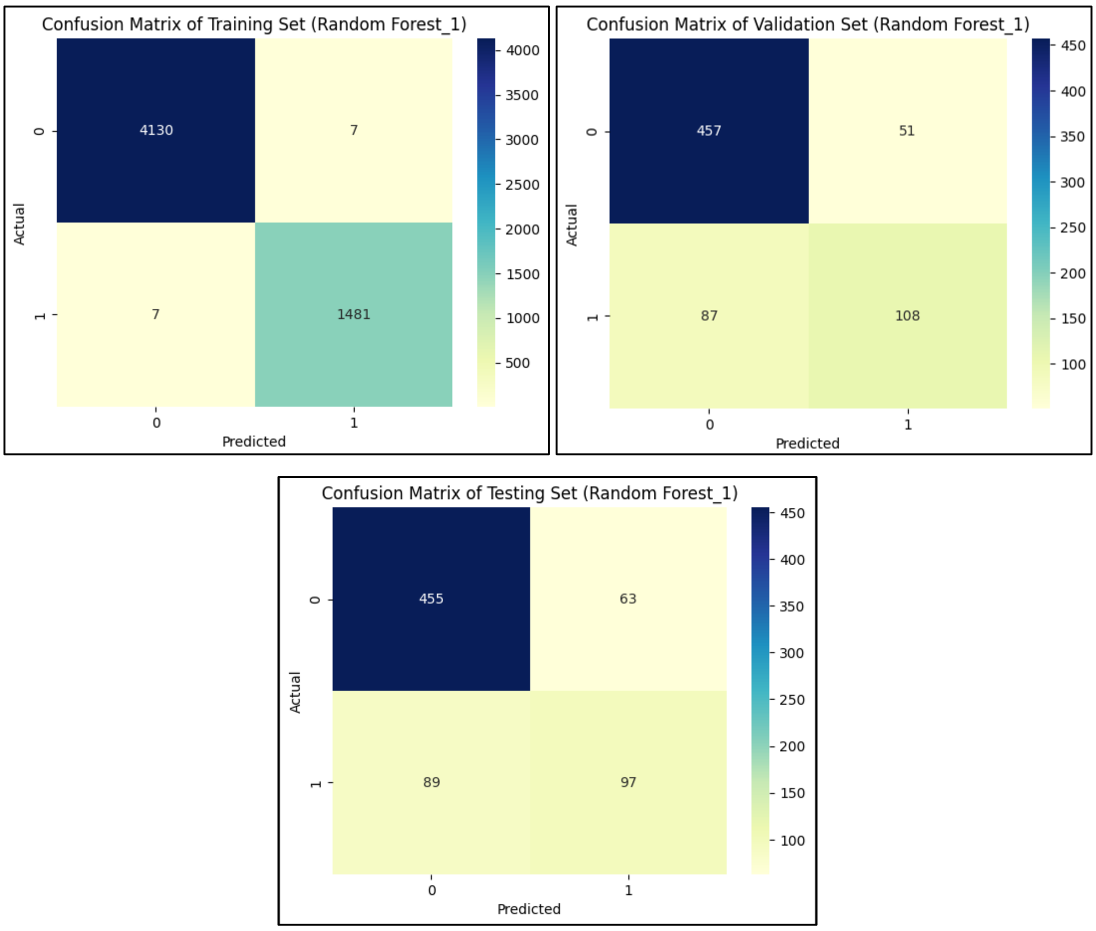


Figure 47. The Confusion Matrix of the Training, Validation, and Testing (Random Forest – Initial Model)

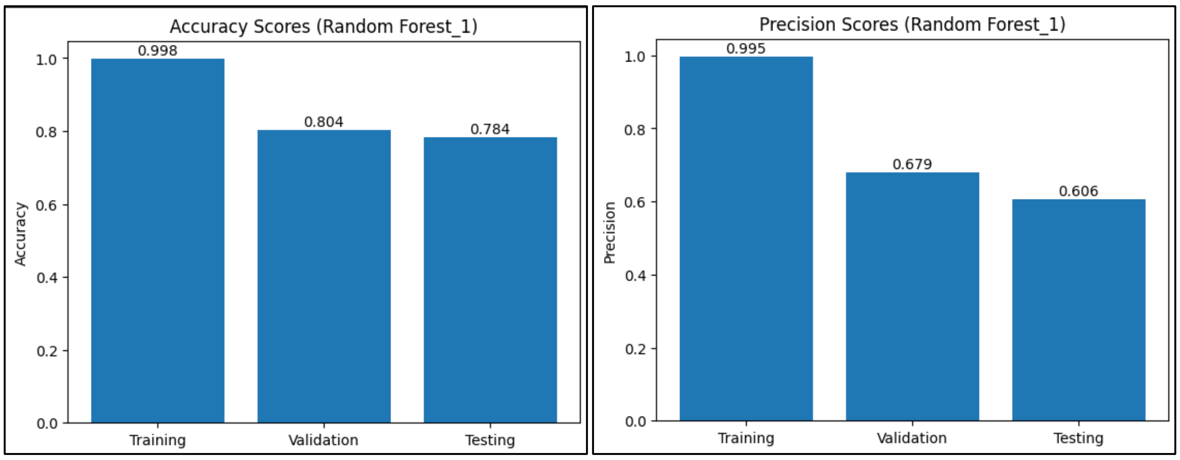


Figure 48. (a) The Accuracy scores of three subsets (Random Forest – Initial Model) – (b) The Precision Scores of three subsets (Random Forest – Initial Model)

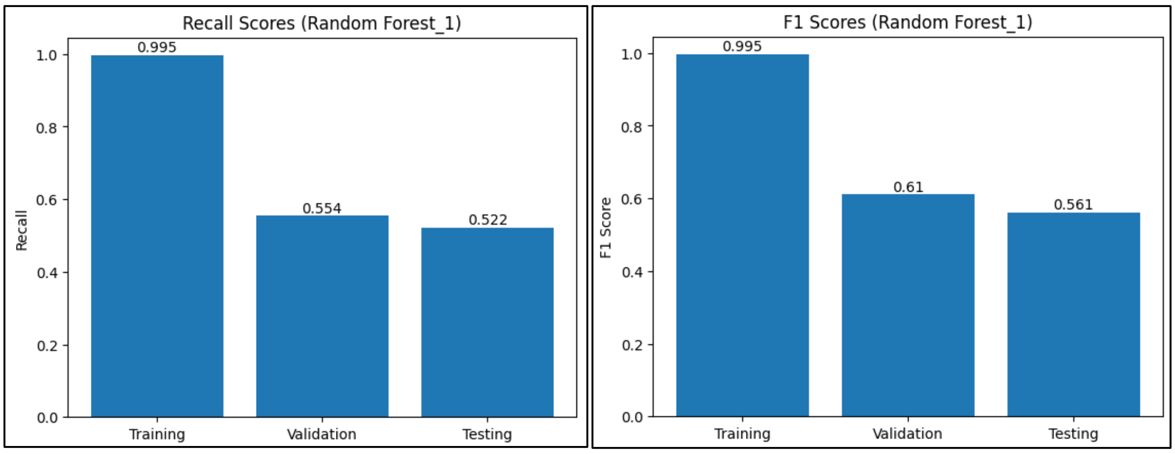


Figure 49. (a) The recall scores of three subsets (Random Forest – Initial Model) – (b) The F1 Scores of three subsets (Random Forest – Initial Model)

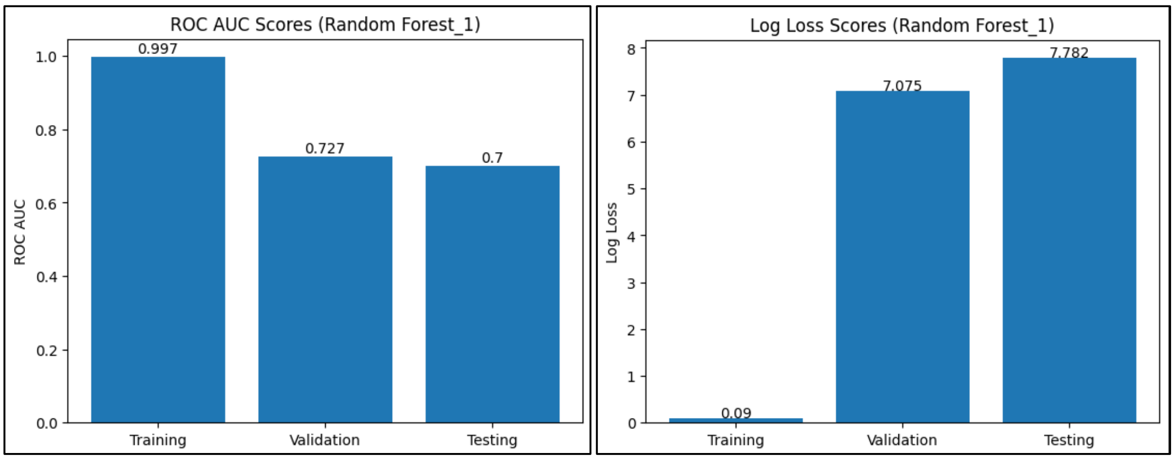
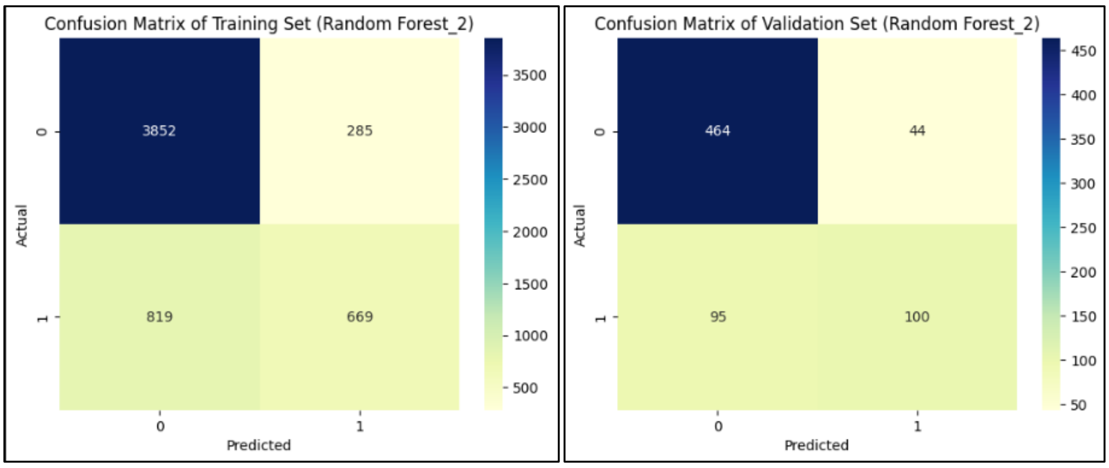


Figure 50. (a) The ROC AUC scores of three subsets (Random Forest – Initial Model) – (b) The Log Loss Scores of three subsets (Random Forest – Initial Model)

4.2.2.2 Model 2



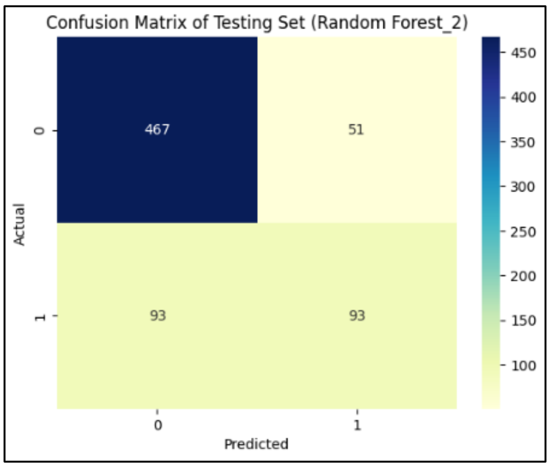


Figure 51. The Confusion Matrix of the Training, Validation, and Testing (Random Forest – Hyperparameter Tuned)

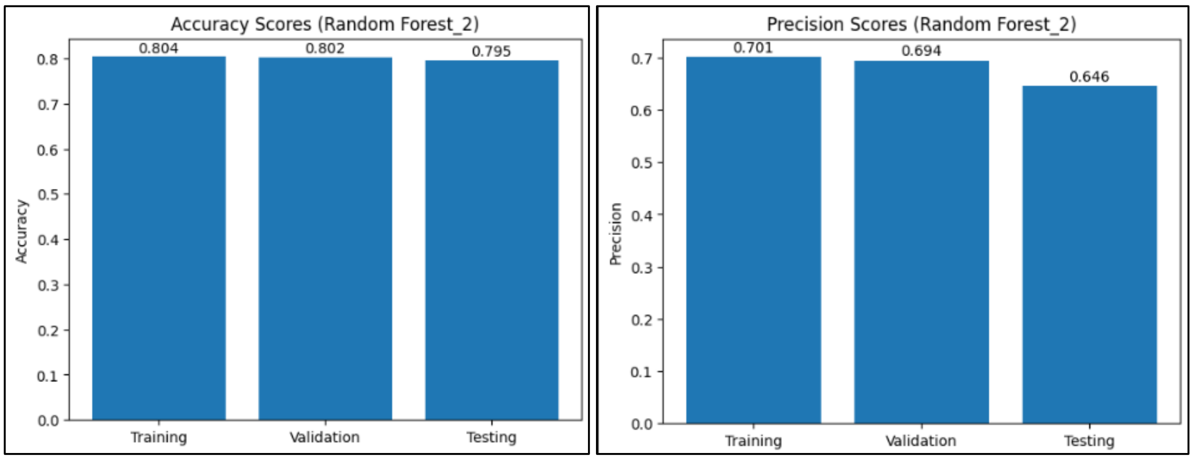


Figure 52. (a) The Accuracy scores of three subsets (Random Forest – model 2) – (b) The Precision Scores of three subsets (Random Forest- model 2)

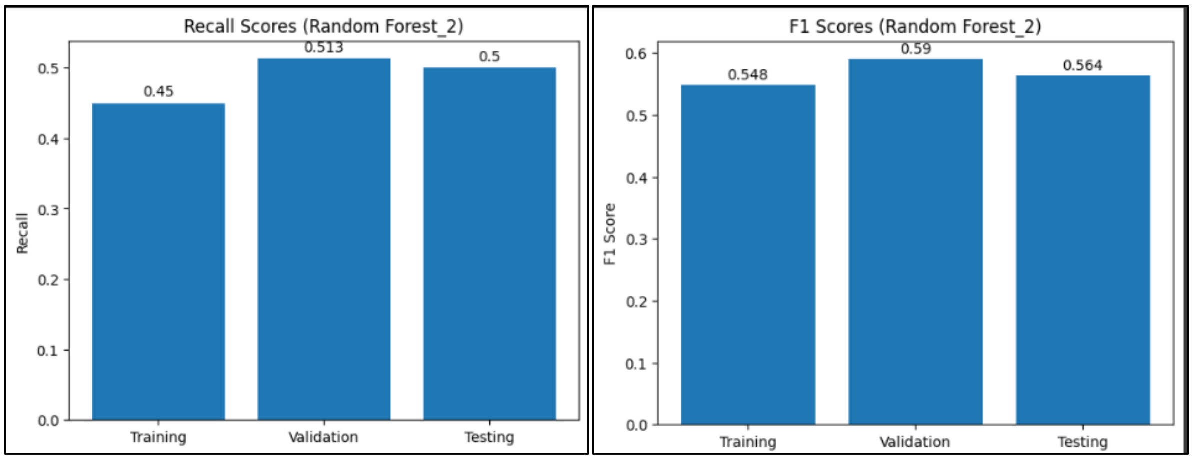


Figure 53. (a) The recall scores of three subsets (Random Forest – model 2) – (b) The F1 Scores of three subsets (Random Forest – model 2)

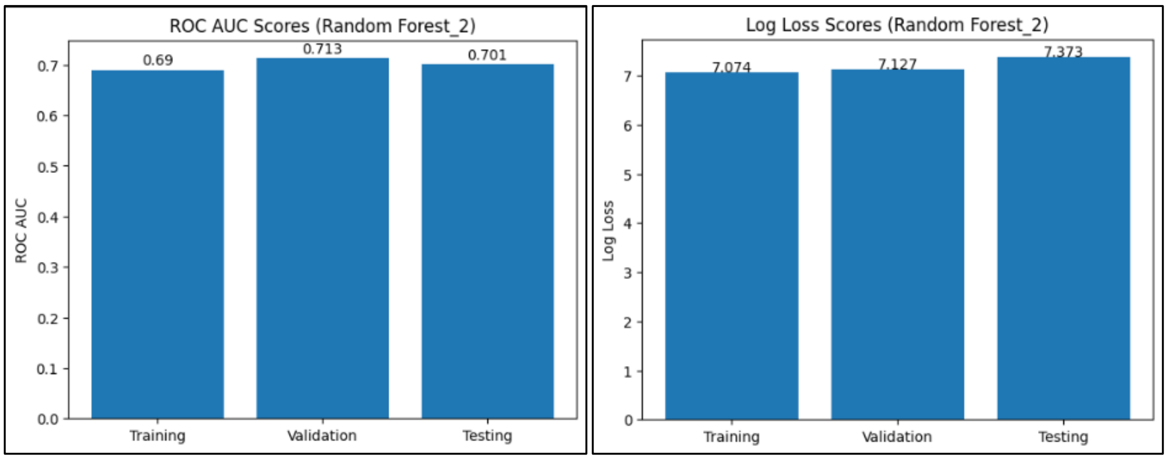
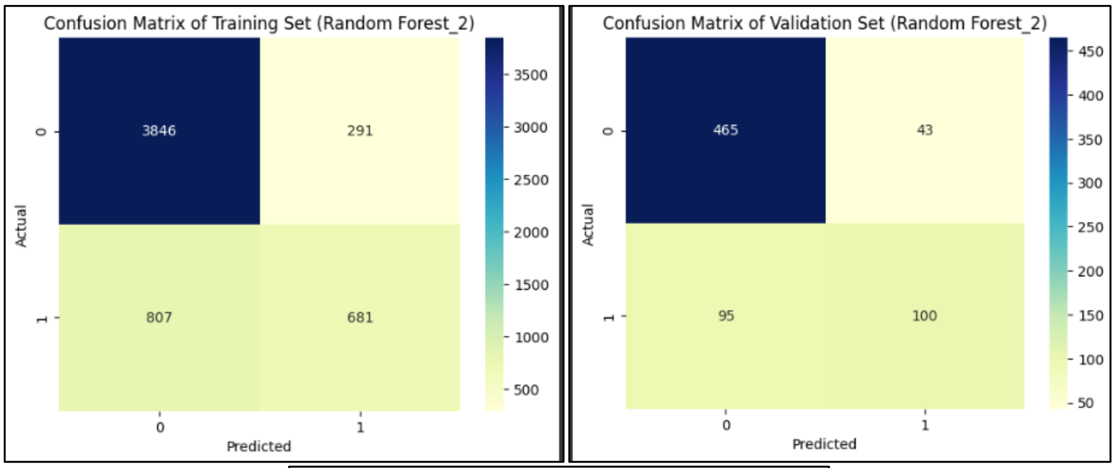


Figure 54. (a) The ROC AUC scores of three subsets (Random Forest – Model2) – (b) The Log Loss Scores of three subsets (Random Forest –Model2)

4.2.2.3 Model 3



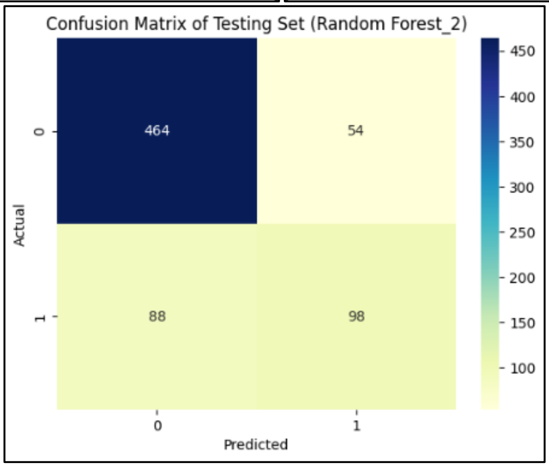


Figure 55. The Confusion Matrix of the Training, Validation, and Testing (Random Forest – model 3)

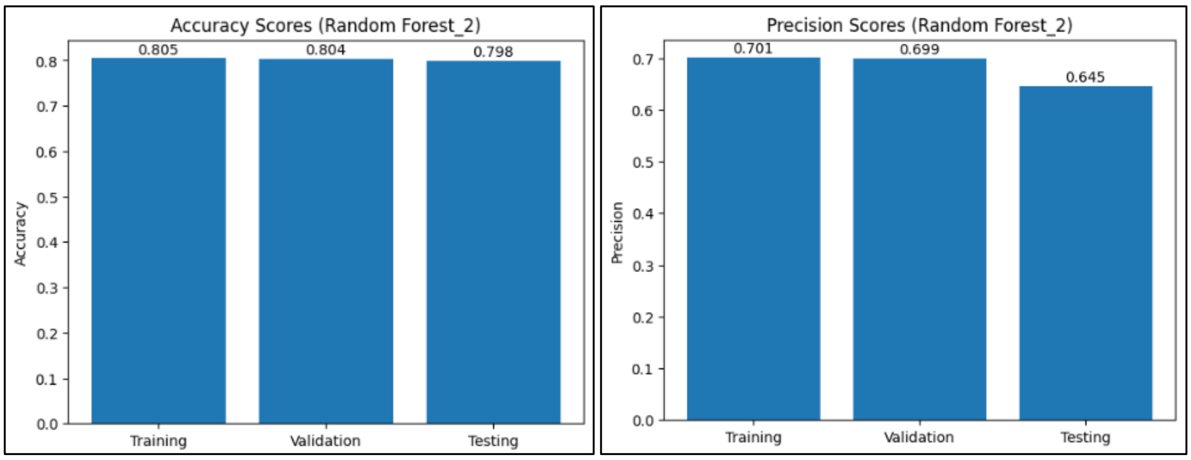


Figure 56. (a) The Accuracy scores of three subsets (Random Forest – model 3) – (b) The Precision Scores of three subsets (Random Forest- model 3)

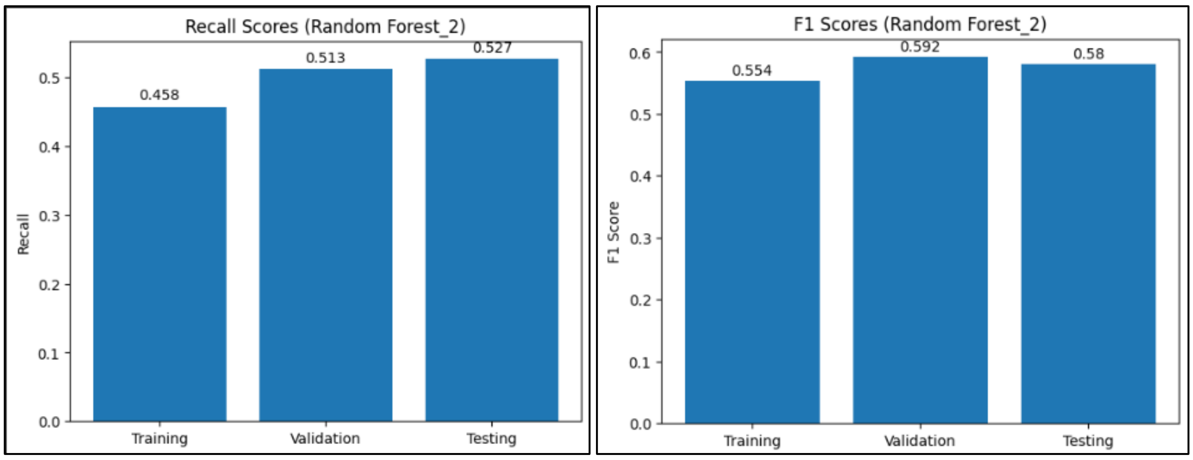


Figure 57. (a) The recall scores of three subsets (Random Forest – model 3) – (b) The F1 Scores of three subsets (Random Forest – model 3)

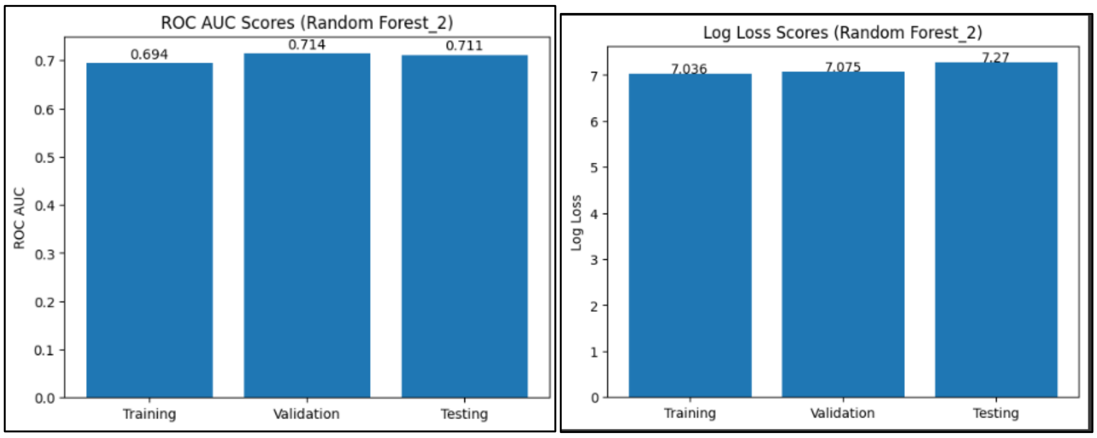


Figure 58. (a) The ROC AUC scores of three subsets (Random Forest – Model3) – (b) The Log Loss Scores of three subsets (Random Forest –Model3)

4.2.3 Extreme Gradient Boosting Results

4.2.3.1 Model 1

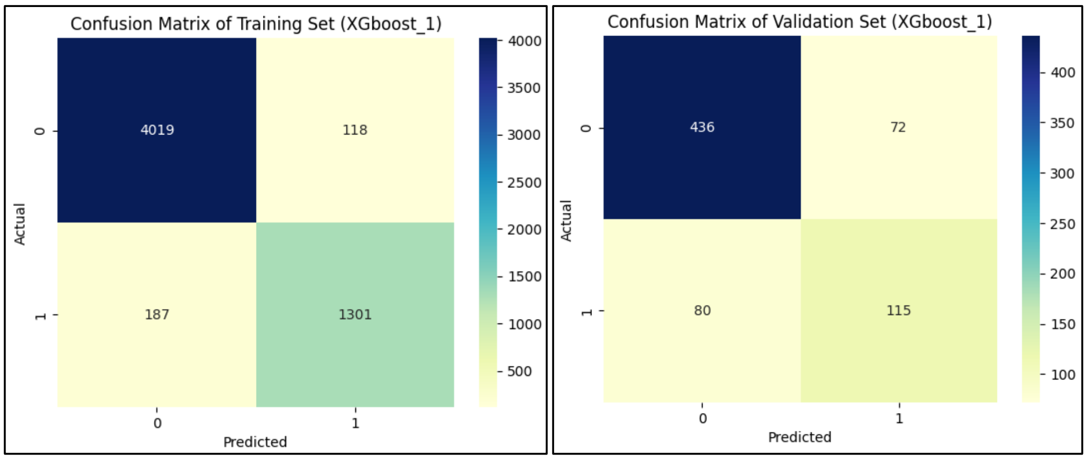
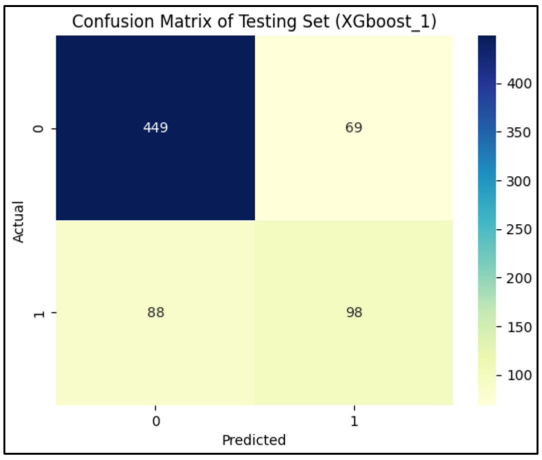



Figure 59. The Confusion Matrix of the Training, Validation, and Testing (Xgboost– model 1)

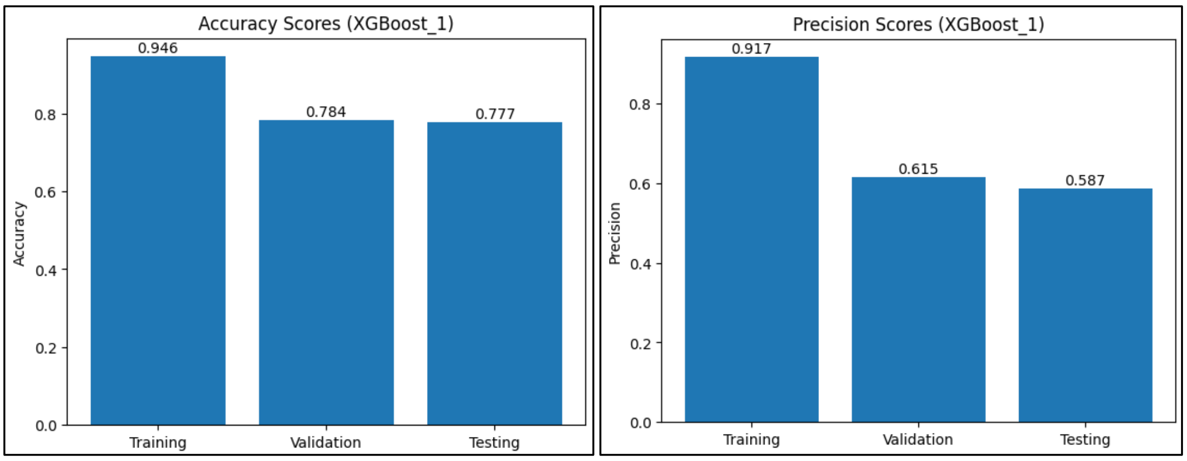


Figure 60. (a) The Accuracy scores of three subsets (Xgboost– model 1) – (b) The Precision Scores of three subsets (Xgboost– model 1)

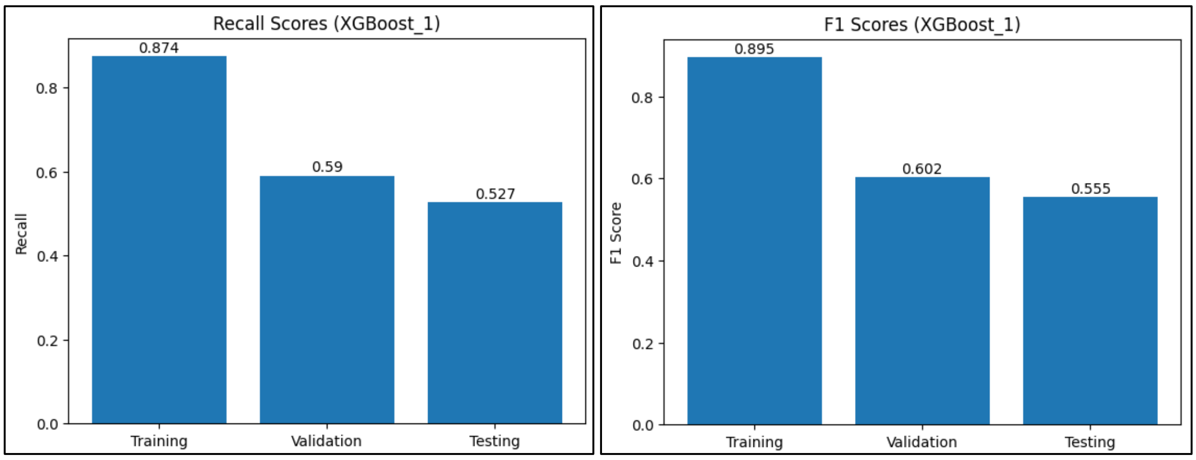


Figure 61. (a) The recall scores of three subsets (Xgboost– model 1) – (b) The F1 Scores of three subsets (Xgboost– model 1)

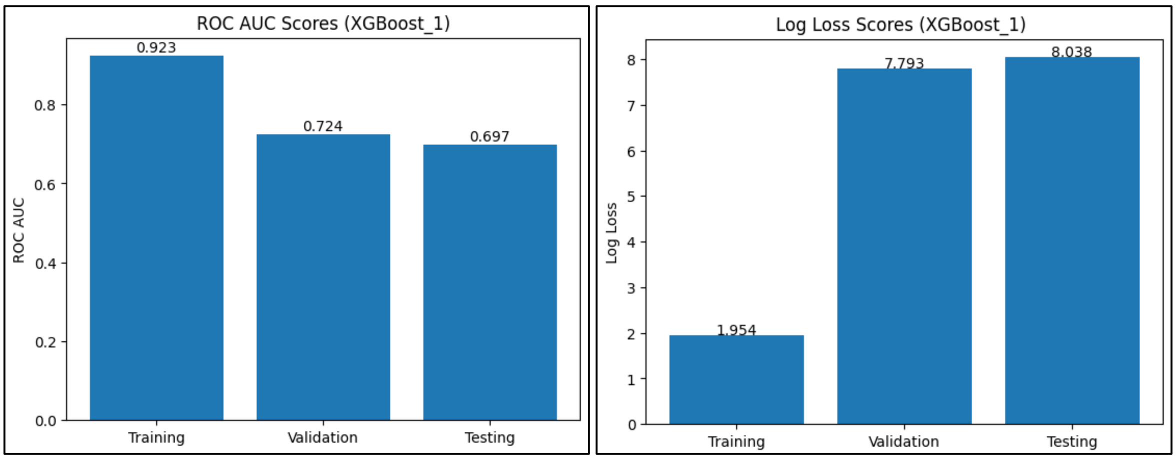
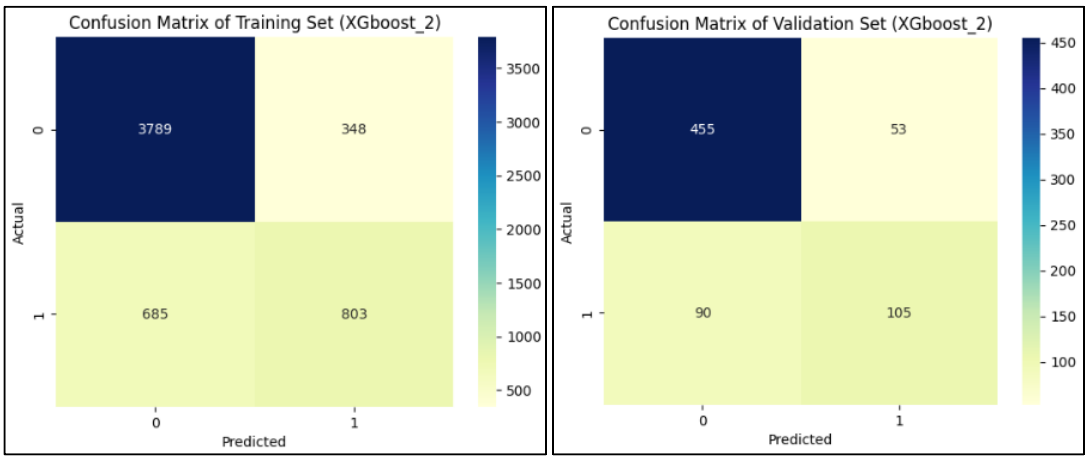


Figure 62. (a) The ROC AUC scores of three subsets (Xgboost– model 1) – (b) The Log Loss Scores of three subsets (Xgboost– model 1)

4.2.3.2 Model 2



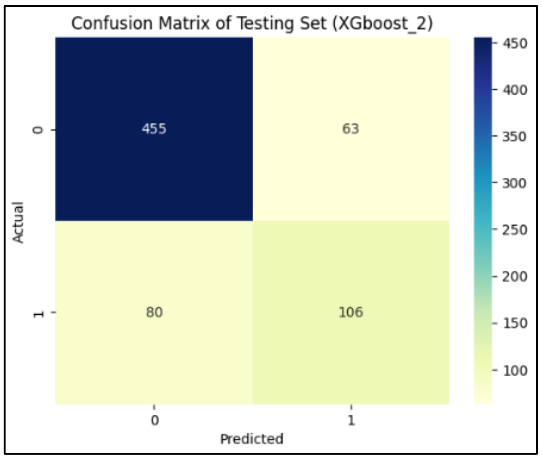
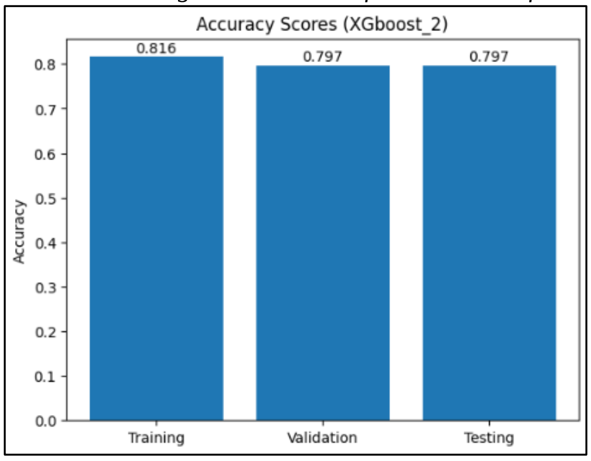


Figure 63. The Confusion Matrix of the Training, Validation, and Testing Xgboost– model 2)



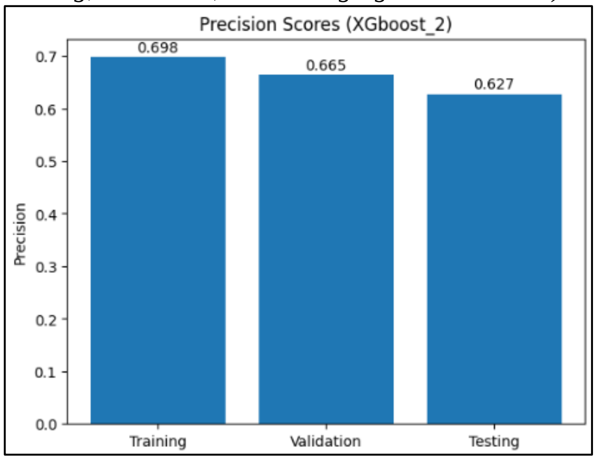
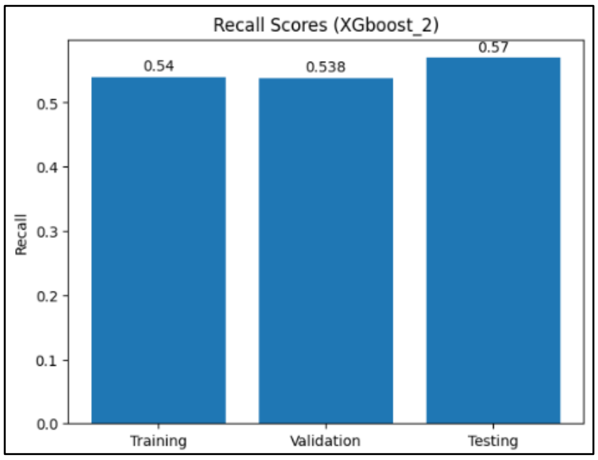


Figure 64. (a) The Accuracy scores of three subsets (Xgboost– model 2) – (b) The Precision Scores of three subsets (Xgboost– model 2)



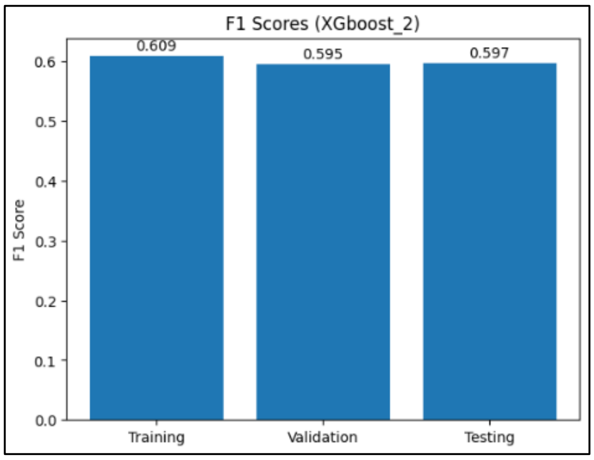


Figure 65. (a) The recall scores of three subsets (Xgboost– model 2) – (b) The F1 Scores of three subsets (Xgboost– model 2)

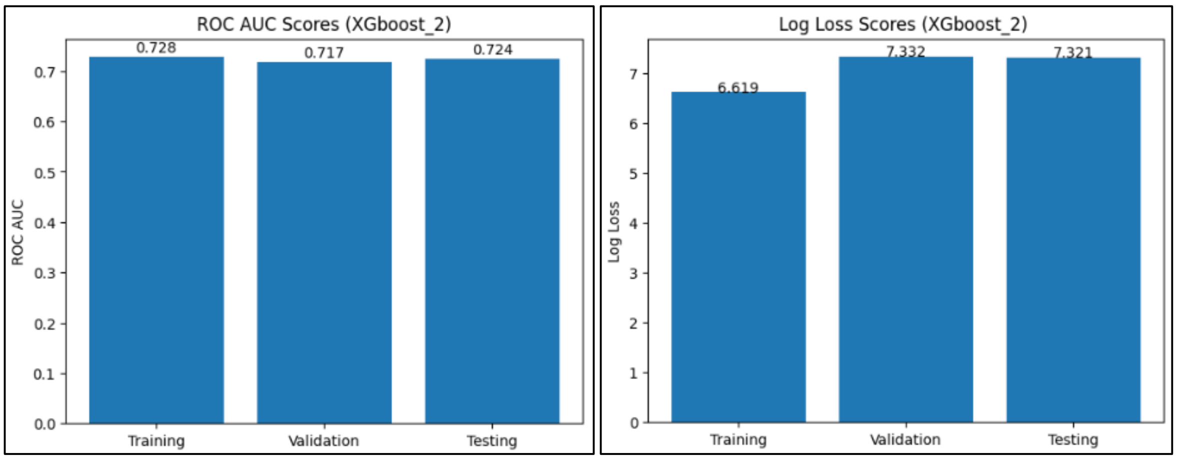


Figure 66. (a) The ROC AUC scores of three subsets (Xgboost– model 2) – (b) The Log Loss Scores of three subsets (Xgboost– model 2)

4.3 Communication and Discussion

DT1 Scaling vs. DT1 No Scaling:

Decision tree models are non-parametric and use threshold values of individual features to split the data into different nodes. Consequently, scaling the features does not have a significant impact on the performance of the model, as the decision is based on the order of the feature values rather than their absolute magnitudes. While feature scaling is a common technique to normalize the range of feature values and can be useful for some machine learning models, such as logistic regression or support vector machines, decision tree models are not affected by the scale of the features. Therefore, it is generally not necessary to perform feature scaling for decision tree models.

In the DT1 (Scaling) model, the feature scaling is applied and the performance of the decision tree model dropped. So, it may be more appropriate to train the model without feature scaling or experiment with other hyper parameters that can improve the model's performance.

The decision tree (model 1) model's performance on the training set is quite good with high accuracy, precision, recall, F1-score, and AUC-score. However, the performance on the validation and testing sets is relatively lower, with lower accuracy, precision, recall, F1-score, and AUC-score. This indicates that the model may have overfit to the training data, as it is performing significantly better on the training set compared to the validation and testing sets. The model may have learned the noise or random fluctuations in the training data, which may not generalize well to new and unseen data.

The log loss score on the validation and testing sets is also significantly higher than the training set, indicating that the model is less confident about its predictions on the validation and testing sets. This is likely due to the overfitting of the model on the training data. Therefore, it may be necessary to further evaluate the model's performance and consider tuning its hyperparameters or using a different algorithm to improve its generalization performance.

Comparing RF1, RF2, and RF3:

Based on the evaluation metrics, RF1 appears to be overfitting the training data, as it has very high accuracy, precision, recall, and F1-score on the training set, but lower scores on the validation and testing sets. This is likely due to the model being too complex and fitting to the noise in the training data.

On the other hand, RF2 and RF3 appear to be performing similarly on the validation and testing sets, with relatively consistent accuracy, precision, recall, and F1-score across the two sets. RF2 has slightly higher scores than RF3, but the differences are not significant.

It appears that RF3 may be slightly underfitting the data, as its scores on both the training and testing sets are slightly lower than those of RF2. However, the differences are relatively small, so it is possible that RF3 is simply performing at a similar level to RF2, but with slightly different parameter settings.

Based on the evaluation metrics results, RF2 appears to be the best-performing model on both the validation and testing sets, with relatively consistent scores and no significant signs of overfitting or underfitting.

Extreme Gradient Boosting – model1: Based on the evaluation metrics, it seems like the first model is performing well on the training set with a high accuracy score, precision score, recall score, F1 score, and AUC score. However, the model seems to be overfitting since the accuracy, precision, recall, F1, and AUC scores are lower on the validation and testing sets. The log loss score is also quite high, which suggests that the model's predicted probabilities are not calibrated well. In general, a lower log loss score indicates better calibrated predicted probabilities. To improve the model's performance, considering implementing techniques such as regularization or adjusting the hyperparameters of the model through techniques such as grid search or randomized search might improve the performance of the model on the validation and testing sets. Also, considering collecting more data or feature engineering will improve the quality of the data. Finally, considering more advanced techniques such as deep learning will also suggested to improve the model's performance.

Based on the evaluation metrics results on the validation and testing sets, compare the two XGboost model 1 and 3:

- ❖ The accuracy score of model1 on the training set is higher than that of model3, but the accuracy score of model1 on the validation and testing sets is lower than that of model3. Therefore, in terms of accuracy, model3 performs better on the validation and testing sets.
- ❖ Model1 has a higher precision score on the training set compared to model3, but on the validation and testing sets, model3 has a higher precision score. Hence, model3 performs better in terms of precision on the validation and testing sets.
- ❖ The recall score of model1 on the training set is higher than that of model3, but on the validation and testing sets, model3 has a higher recall score. Therefore, in terms of recall, model3 performs better on the validation and testing sets.
- ❖ The F1-score of model1 on the training set is higher than that of model3, but on the validation and testing sets, model3 has a higher F1-score. Therefore, in terms of F1-score, model3 performs better on the validation and testing sets.
- ❖ The AUC-score of model1 on the training set is higher than that of model3, but on the validation and testing sets, model3 has a higher AUC-score. Therefore, in terms of AUC-score, model3 performs better on the validation and testing sets.
- ❖ Model1 has a lower log loss score on the training set compared to model3, but on the validation and testing sets, model3 has a lower log loss score. Therefore, in terms of log loss score, model3 performs better on the validation and testing sets. As a result of comparing XG1 and XG3, model3 performs better on the validation and testing sets compared to model1 based on most of the evaluation metrics.

5 Conclusion

This project focuses on comparing the performance of three tree-based models: Decision Tree, Random Forest, and Extreme Gradient Boosting. Initially, each model was trained and evaluated on a dataset to determine their baseline performance. Grid search was then employed to find the best hyperparameters for each model based on cross-validation and five folds to enhance their performance. To assess the performance of the models on unseen data, learning curves were generated, and various evaluation metrics such as accuracy, precision, recall, F1-

score, AUC-score, and log loss were computed and compared for each model on the validation and testing sets. The evaluation results provided valuable insights into the performance of each model and their effectiveness in predicting the outcome variable of interest. Overall, this study provides a comprehensive comparison of the three tree-based models and their potential application in solving classification problems.

6 Recommendations for future work

Survival analysis is a statistical technique used to analyse the time until an event of interest occurs. In customer churn prediction, the event of interest is when a customer stops using the service. Survival analysis can be used to provide more accurate insights into the time to customer churn. It can help identify which factors or variables increase or decrease the risk of customer churn at a given time. This information can be used to take proactive measures to prevent customer churn, such as targeted marketing or personalized retention offers. In addition, survival analysis can help identify patterns in customer behaviour leading up to churn. This information can be used to create customer segmentation based on their likelihood to churn and retention strategies. In general, survival analysis in customer churn prediction models can provide a more comprehensive understanding of the time to churn and help telecommunication companies take proactive steps to reduce customer churn rates.

Feature engineering involves creating new features from the existing features to improve the performance of the model. Here are some feature engineering ideas for the existing columns in the customer churn prediction dataset:

- ❖ Tenure: creating a new feature that represents the remaining tenure of each customer's contract.
- ❖ InternetService, OnlineSecurity, OnlineBackup, and StreamingMovies columns all represent whether or not a customer has a particular service. Combining these features into a single feature that represents the number of services a customer has. As an example, create a new feature called "Number of Services" that takes on values from 0 to 4 depending on how many of these services a customer has.

- The contract column represents the type of contract a customer has (e.g. monthly, yearly, two-year). A new feature could have created to represent the length of the customer's contract in months.
- The TotalCharges column represents the total amount a customer has paid over the course of their contract. A new feature could have created that represents the average monthly payment for each customer by dividing their total charges by their tenure. This could help identify customers who have consistently paid more or less than average.

Collecting more data can help improve the performance of the model by providing it with more examples to learn from. Trying collecting more data or using existing data from different sources to see if it improves the model's performance.

7 References

Ahsan, M.M., Mahmud, M.P., Saha, P.K., Gupta, K.D. and Siddique, Z., 2021. Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, *9*(3), p.52. [Accessed 17 April 2023].

Bergstra, J. and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2). [Accessed 17 April 2023].

Bhatore, S., Mohan, L. and Reddy, Y.R., 2020. Machine learning techniques for credit risk evaluation: a systematic literature review. *Journal of Banking and Financial Technology*, *4*, pp.111-138. [Accessed 17 April 2023].

Bhattacharyya, J. and Dash, M.K., 2022. What do we know about customer churn behaviour in the telecommunication industry? A bibliometric analysis of research trends, 1985–2019. *FIIB Business Review*, *11*(3), pp.280-302. [Accessed 17 April 2023].

Birba, D.E., 2020. A Comparative study of data splitting algorithms for machine learning model selection. [Accessed 17 April 2023].

Charbuty, B. and Abdulazeez, A., 2021. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, *2*(01), pp.20-28. [Accessed 17 April 2023].

DatabaseCamp. (n.d.). SVM Explained. [online] Available at: https://databasecamp.de/en/ml/svm-explained [Accessed 17 April 2023].

Goetz, J.N., Brenning, A., Petschko, H. and Leopold, P., 2015. Evaluating machine learning and statistical prediction techniques for landslide susceptibility modeling. *Computers & geosciences*, *81*, pp.1-11. [Accessed 17 April 2023].

Gudivada, V., Apon, A. and Ding, J., 2017. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, *10*(1), pp.1-20. [Accessed 17 April 2023].

Gupta, B., Rawat, A., Jain, A., Arora, A. and Dhami, N., 2017. Analysis of various decision tree algorithms for classification in data mining. *International Journal of Computer Applications*, *163*(8), pp.15-19. [Accessed 17 April 2023].

Hanif, I., 2020. Implementing extreme gradient boosting (xgboost) classifier to improve customer churn prediction. [Accessed 17 April 2023].

Hertel, L., Collado, J., Sadowski, P., Ott, J. and Baldi, P., 2020. Sherpa: Robust hyperparameter optimization for machine learning. *SoftwareX*, *12*, p.100591. [Accessed 17 April 2023].

Huang, B., Kechadi, M.T. and Buckley, B., 2012. Customer churn prediction in telecommunications. *Expert Systems with Applications*, *39*(1), pp.1414-1425. [Accessed 17 April 2023].

Kaur, S., Aggarwal, H. and Rani, R., 2020. Hyper-parameter optimization of deep learning model for prediction of Parkinson’s disease. *Machine Vision and Applications*, *31*, pp.1-15. [Accessed 17 April 2023].

King, R.D., Orhobor, O.I. and Taylor, C.C., 2021. Cross-validation is safe to use. *Nature Machine Intelligence*, *3*(4), pp.276-276. [Accessed 17 April 2023].

Kleinbaum, D.G., Dietz, K., Gail, M., Klein, M. and Klein, M., 2002. *Logistic regression* (p. 536). New York: Springer-Verlag. [Accessed 17 April 2023].

Liashchynskyi, P. and Liashchynskyi, P., 2019. Grid search, random search, genetic algorithm: a big comparison for NAS. *arXiv preprint arXiv:1912.06059*. [Accessed 17 April 2023].

Navada, A., Ansari, A.N., Patil, S. and Sonkamble, B.A., 2011, June. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE control and system graduate research colloquium* (pp. 37-42). IEEE. [Accessed 17 April 2023].

Podgorelec, V., Kokol, P., Stiglic, B. and Rozman, I., 2002. Decision trees: an overview and their use in medicine. *Journal of medical systems*, *26*, pp.445-463. [Accessed 17 April 2023].

Rahman, M. and Kumar, V., 2020, November. Machine learning based customer churn prediction in banking. In *2020 4th international conference on electronics, communication and aerospace technology (ICECA)* (pp. 1196-1201). IEEE. [Accessed 17 April 2023].

Saravanan, R. and Sujatha, P., 2018, June. A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification. In *2018 Second international conference on intelligent computing and control systems (ICICCS)* (pp. 945-949). IEEE. [Accessed 17 April 2023].

Schaffer, C., 1993. Selecting a classification method by cross-validation. *Machine learning*, *13*, pp.135-143. [Accessed 17 April 2023].

Seger, C., 2018. An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. [Accessed 17 April 2023].

Vovk, V., 2015. The fundamental nature of the log loss function. *Fields of Logic and Computation II: Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, pp.307-318. [Accessed 17 April 2023].

Wang, F. and Ross, C.L., 2018. Machine learning travel mode choices: Comparing the performance of an extreme gradient boosting model with a multinomial logit model. *Transportation Research Record*, *2672*(47), pp.35-45. [Accessed 17 April 2023].

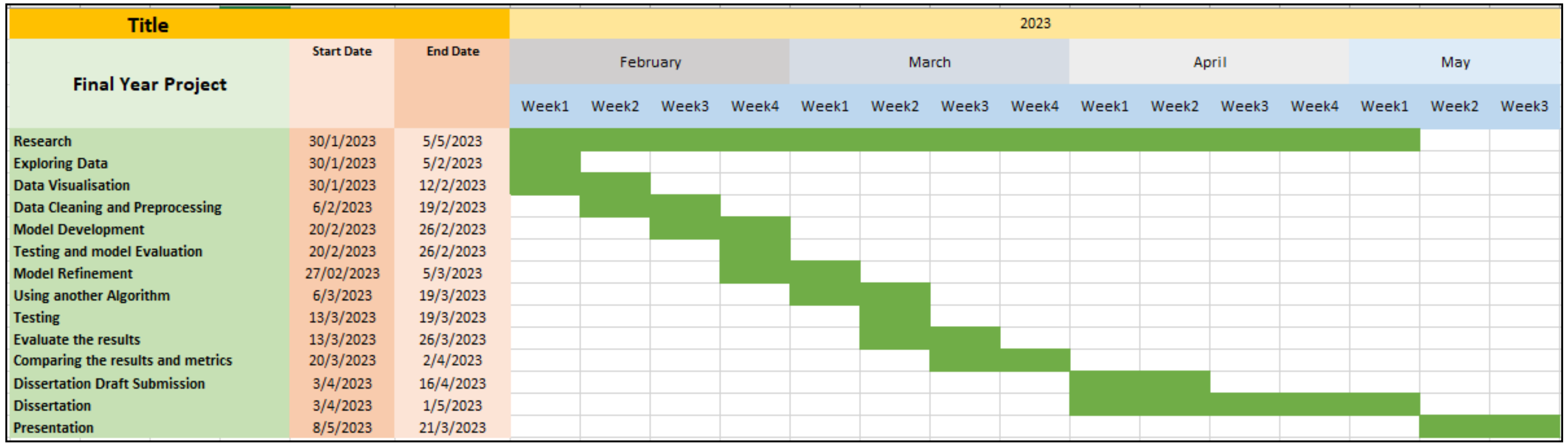
Westreich, D., Lessler, J. and Funk, M.J., 2010. Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression. *Journal of clinical epidemiology*, *63*(8), pp.826-833. [Accessed 17 April 2023].

Zhang, T., Moro, S. and Ramos, R.F., 2022. A data-driven approach to improve customer churn prediction based on telecom customer segmentation. Future Internet, 14(3), p.94. [Accessed 17 April 2023].

Blastchar. (n.d.). Telco Customer Churn. Kaggle. Retrieved from https://www.kaggle.com/blastchar/telco-customer-churn. [Accessed 17 April 2023]

8 Appendices

1. The Gantt Chart of the Project



Title			2023														
Final Year Project	Start Date	End Date	February				March				April				May		
			Week1	Week2	Week3	Week4	Week1	Week2	Week3	Week4	Week1	Week2	Week3	Week4	Week1	Week2	Week3
Research	30/1/2023	5/5/2023	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
Exploring Data	30/1/2023	5/2/2023	🟩														
Data Visualisation	30/1/2023	12/2/2023	🟩	🟩													
Data Cleaning and Preprocessing	6/2/2023	19/2/2023		🟩	🟩												
Model Development	20/2/2023	26/2/2023			🟩	🟩											
Testing and model Evaluation	20/2/2023	26/2/2023				🟩											
Model Refinement	27/02/2023	5/3/2023				🟩	🟩										
Using another Algorithm	6/3/2023	19/3/2023					🟩	🟩									
Testing	13/3/2023	19/3/2023						🟩									
Evaluate the results	13/3/2023	26/3/2023						🟩	🟩								
Comparing the results and metrics	20/3/2023	2/4/2023							🟩	🟩							
Dissertation Draft Submission	3/4/2023	16/4/2023									🟩	🟩					
Dissertation	3/4/2023	1/5/2023									🟩	🟩	🟩	🟩	🟩		
Presentation	8/5/2023	21/3/2023														🟩	🟩

Figure 67. The Gantt Chart of the Project

2. The plot of the Decision Tree (Hyperparameter tuning)

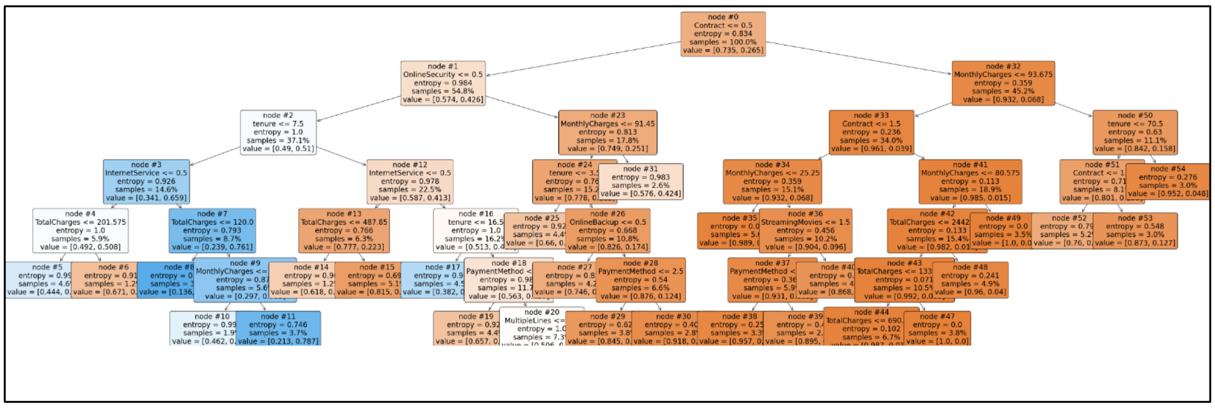


Figure 68. The pot of the Decision Tree – Hyperparameter Tuning

3. The results of the initial Decision tree with scaling (Standard Scaler)

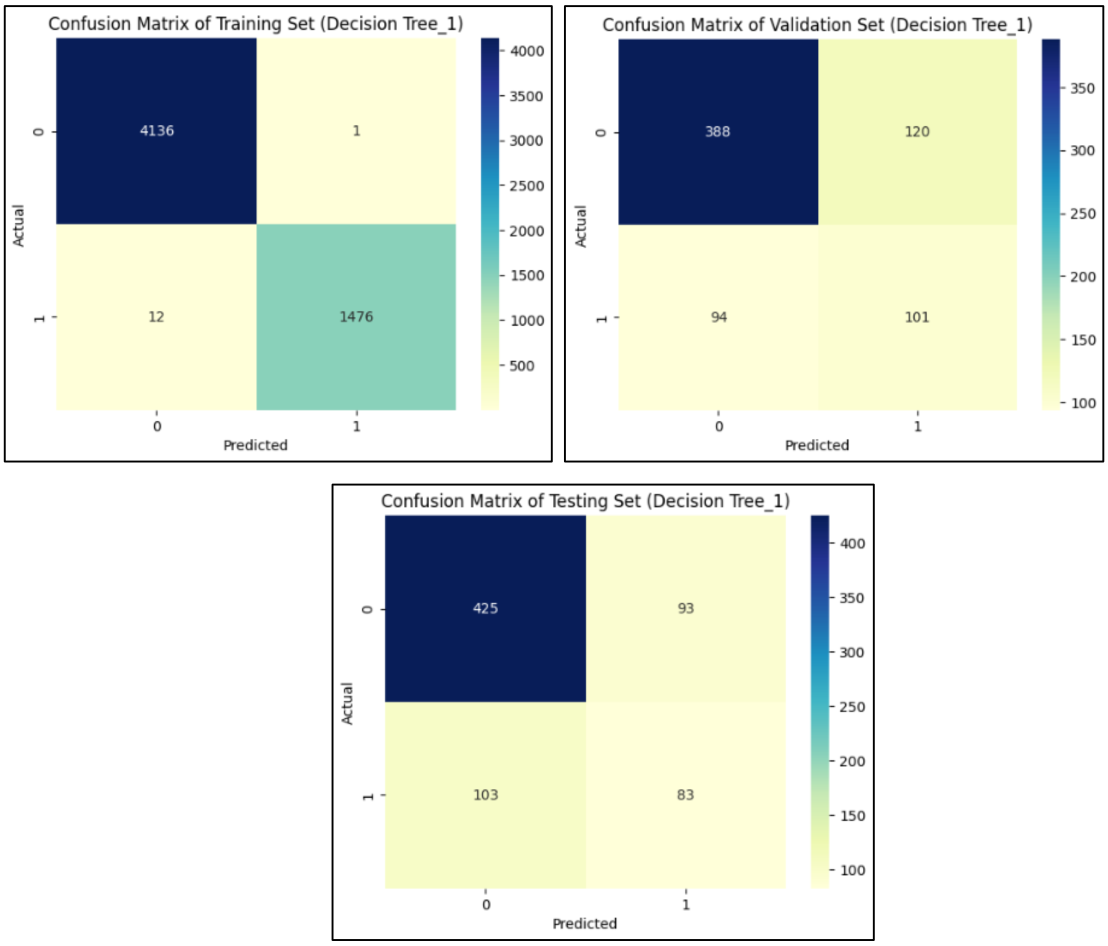


Figure 69. The Confusion Matrix of the Training, Validation, and Testing (Decision Tree – Scaling)

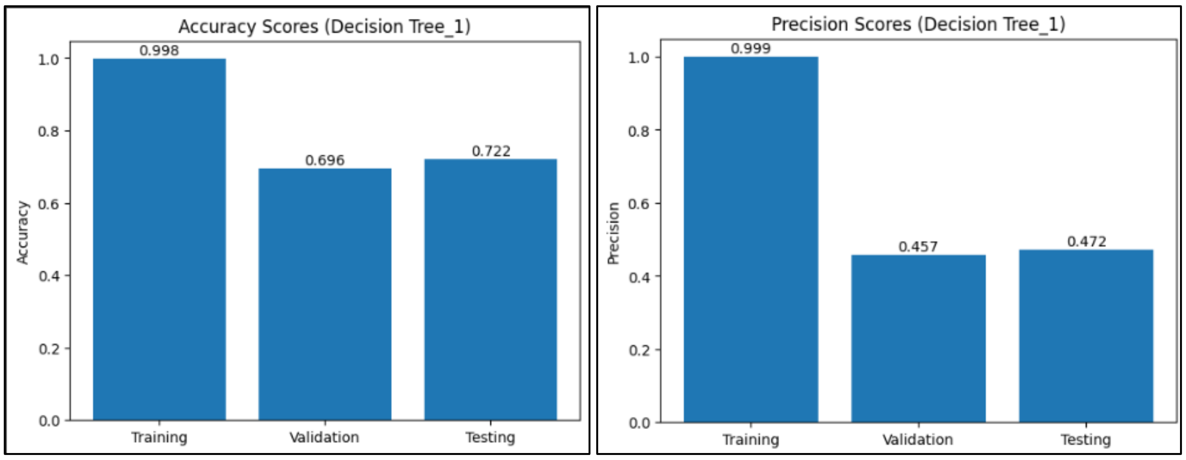


Figure 70. (a) The Accuracy scores of three subsets (DT - With Scaling) – (b) The Precision Scores of three subsets (DT - With Scaling)

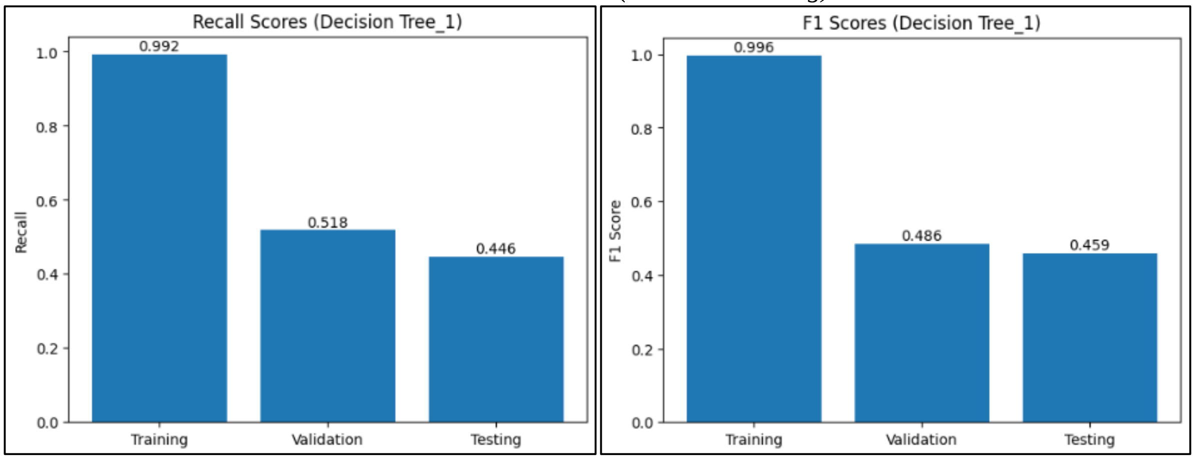


Figure 71. (a) The Recall scores of three subsets (DT - With Scaling) – (b) The F1 Scores of three subsets (DT - With Scaling)

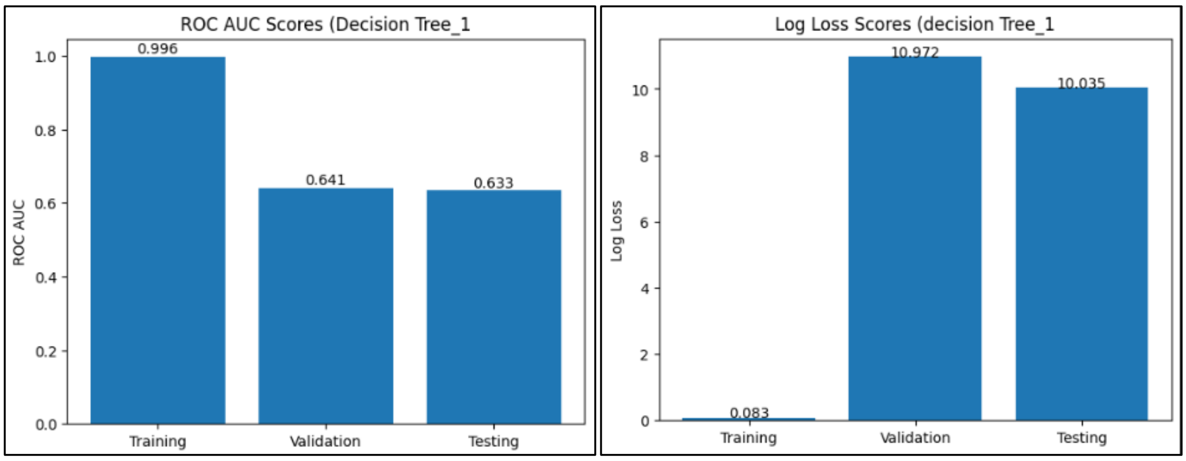
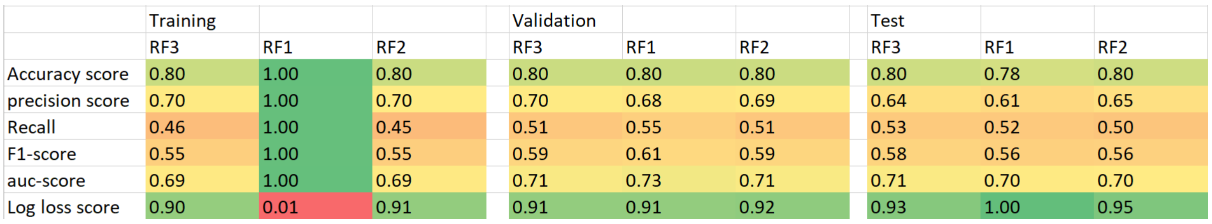


Figure 72. (a) The ROC AUC scores of three subsets (DT - With Scaling) – (b) The Log Loss Scores of three subsets (DT - With Scaling)

4. The RF Models Results



	Training			Validation			Test		
	RF3	RF1	RF2	RF3	RF1	RF2	RF3	RF1	RF2
Accuracy score	0.80	1.00	0.80	0.80	0.80	0.80	0.80	0.78	0.80
precision score	0.70	1.00	0.70	0.70	0.68	0.69	0.64	0.61	0.65
Recall	0.46	1.00	0.45	0.51	0.55	0.51	0.53	0.52	0.50
F1-score	0.55	1.00	0.55	0.59	0.61	0.59	0.58	0.56	0.56
auc-score	0.69	1.00	0.69	0.71	0.73	0.71	0.71	0.70	0.70
Log loss score	0.90	0.01	0.91	0.91	0.91	0.92	0.93	1.00	0.95