CITY UNIVERSITY OF HONG KONG

DEPARTMENT OF PHYSICS

BACHELOR OF SCIENCE IN APPLIED PHYSICS / PHYSICS

2022-2023 DISSERTATION

Machine Learning Methods in Finance

Credit Scoring using Machine Learning

by

SHEIKH Hamza SID: 55344555

March 2023

Credit Scoring using Machine Learning

By

SHEIKH Hamza SID: 55344555

Submitted in partial fulfilment of the

requirements for the degree of

BACHELOR OF SCIENCE

IN

APPLIED PHYSICS / PHYSICS

from

City University of Hong Kong

March 2023

Project Supervisor :	Prof ZHANG R Q

Table of Contents

List of Figures

Abstract

Creditworthiness is a term used to describe a borrower's ability to repay their debts on time and in full. It is a measure of how likely a borrower is to default on their financial obligations, such as loans, credit cards, or other lines of credit, and is an assessment that lenders make to determine whether to extend credit to the borrower, and at what terms. Credit scoring is the process of analyzing an individual's credit history to determine their ability to pay back borrowed money or to make payments on time.

The Give Me Some Credit Kaggle competition from 2011 is centered around developing a model to predict the probability that a borrower will experience financial distress within the next two years. Participants are challenged to build a model using historical data provided, which will assist borrowers to make better financial decisions. The competition aims to improve the state of credit scoring, with a focus on building a probability of default model that outperforms current algorithms, particularly with respect to identifying people who may suffer financial hardship in the near future.

The idea is to use suitable data preprocessing followed by appropriate machine learning models. In this dissertation, I will be going through my attempt at this competition.

1. Introduction

The concept of credit

Credit generally refers to the ability to borrow money, goods or services with the understanding that you will pay it back later along with any applicable interest or fees. It is a financial tool that allows individuals and businesses to obtain resources they need to achieve their goals or meet their needs without having to pay the full amount upfront. This can include things like loans, credit cards, or lines of credit. Credit can also refer to an individual's financial reputation, which is based on factors such as their credit history, credit score, and overall creditworthiness. [1]

The concept of credit has been around for thousands of years, with evidence of credit transactions dating back to ancient civilizations such as Mesopotamia and Ancient Egypt. In these early societies, credit was typically extended in the form of loans from local merchants or moneylenders, often with high interest rates. [2]

In the centuries that followed, credit continued to be an important aspect of economic activity, with early banks and moneylenders playing a key role in lending money to fund business ventures and other initiatives. The first credit bureaus also emerged during this time, with businesses like R.G. Dun & Company and Dun & Bradstreet collecting data on the creditworthiness of individuals and companies. [3]

In the mid-20th century, the concept of credit was transformed with the introduction of credit cards, which enabled consumers to make purchases using a line of credit provided by a financial institution. The Diners Club card, introduced in 1950, was one of the first widely recognized credit cards, and was soon followed by a wave of competitors such as American Express and Visa. Today, credit plays a central role in the global economy, with businesses and individuals using various forms of credit to fund everything from day-to-day expenses to major business projects and investments. Credit is also closely tied to the financial health of individuals, as credit scores and credit histories are used by lenders to determine whether to extend credit and at what interest rates. [4]

Credit Scoring

Credit scoring is a statistical analysis performed by lenders and financial institutions to determine the creditworthiness of a person or a small business. It involves collecting data on an individual's past credit history and using that information to create a numerical score that represents their creditworthiness. This score, known as a credit score, is then used by lenders as a tool to assess the risk associated with lending to that individual. Higher credit scores are

generally seen as indicators of lower credit risk, and can help individuals qualify for loans, credit cards, and other financial products. [5]

The history of credit scoring can be traced back to the early 19th century, when commercial lenders in the United States began using credit reporting to evaluate the risk of extending credit to borrowers. However, the first modern credit scoring model was not developed until 1958, when Bill Fair and Earl Isaac created Credit Bureau Inc., which later became known as Fair Isaac Corporation or FICO. In 1989, FICO worked with national credit bureaus to create a credit scoring model that could be used to evaluate all consumers, paving the way for the widespread use of credit scores as we know them today. The first credit bureau, the Mercantile Agency, was pioneered by Bradstreet in 1857, and would assume a more lasting form in 1864 when it was renamed R.G. Dun & Co. Today, credit reports and credit scores are used extensively by lenders, employers, and landlords to evaluate people's creditworthiness. [3]

Artificial Intelligence

Artificial intelligence (AI) is a field of computer science that involves creating intelligent machines that can perform tasks that typically require human intelligence, such as recognizing patterns, understanding natural language, decision making, and learning from experience. AI encompasses various techniques such as machine learning, deep learning, natural language processing, robotics, and other specialized fields. The goal of AI is to create machines that can think, learn, and act like humans, and potentially even surpass human intelligence in some domains. [6]

The history of artificial intelligence (AI) goes back several decades to the early days of computer science. The field of AI was founded on the idea that machines could be made to perform tasks that would normally require human intelligence, such as reasoning, perception, learning, and problem-solving. The earliest work in AI can be traced back to the 1940s, when computer scientists began creating symbolic systems that could perform basic calculations and logical operations. In the 1950s and 1960s, researchers began to develop more sophisticated approaches to AI, such as rule-based systems, neural networks, and decision trees. However, despite the significant progress made in AI research, the field remained largely experimental and speculative until the 1980s, when advances in computer hardware and software helped to make AI more practical and accessible to a wider audience. Today, AI is a rapidly growing field that's being used to develop intelligent systems in a wide range of applications, from natural language processing and image recognition to robotics and self-driving cars. [6]

Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that involves creating algorithms and models that can automatically learn from data and improve their performance over time without being explicitly programmed. Essentially, ML involves feeding large amounts of data into a computer system, which then tries to discover patterns and relationships in the data that can be used to make predictions or take actions. There are several different types of machine learning algorithms, including supervised learning, unsupervised learning, and reinforcement learning. These algorithms are used in a wide range of applications, from image and speech recognition to fraud detection and predictive analytics. Overall, machine learning is a powerful technology that is revolutionizing many fields and enabling new forms of automation and intelligent decision-making. [7]

The history of machine learning dates back to the early days of computing and artificial intelligence (AI). The earliest work on machine learning can be traced back to the 1940s and 1950s, when computer scientists began developing symbolic systems that could perform basic calculations and logical operations. In the 1960s and 1970s, researchers began developing more sophisticated approaches to machine learning, such as decision trees and rule-based systems. However, despite these advances, machine learning remained a largely experimental and speculative field until the 1990s, when it was reorganized as a separate field with a new goal of achieving artificial intelligence using statistical models and other techniques. Today, machine learning has become a pillar of modern AI and is used in a wide range of applications, from image and speech recognition to predictive analytics and fraud detection. The history of machine learning is marked by a constant interplay between theoretical advances in mathematics and computer science and practical applications in various domains, and this interplay continues to drive innovation and progress in the field. [8]

Machine Learning and Credit Scoring

Machine learning is increasingly being used in credit scoring as it can provide significant benefits over traditional methods. ML models can take into account a wider range of factors and more complex relationships between them, leading to more accurate predictions of creditworthiness.[9] Here are some ways in which machine learning is used in credit scoring:

1. Prediction of default risk: Machine learning models can predict the likelihood of a borrower defaulting on their loan based on data from credit reports, such as payment history, credit utilization, length of credit history, etc.

2. Credit decisioning: Machine learning models can be used to make decisions on whether to approve or deny a credit application based on a borrower's creditworthiness.

3. Fraud detection: Machine learning models can help detect fraud by identifying patterns in fraudulent transactions and alerting lenders when suspicious activity is detected.

4. Risk assessment: Machine learning models can be used to assess the risk associated with lending to a particular borrower or group of borrowers. This can help lenders make more informed decisions about interest rates, loan amounts, and other terms.

Overall, machine learning can help automate routine processes, increase the speed of service, reduce the costs of solving standard tasks, enhance accuracy and improve decision-making in credit scoring.

Machine Learning Models

Decisions Trees

In machine learning, decision trees are a type of supervised learning algorithm used for both classification and regression tasks.
A decision tree is a tree-like flowchart structure in which internal nodes represent test on an attribute, branches represent the outcome of the test, and leaf nodes represent the class or the target variable.
A decision tree is created by recursively finding the best attribute to split the data into subsets that maximizes the separation of the classes, until the leaf nodes are pure or the maximum tree depth is reached. The resulting tree can then be used for prediction of the target variable for new instances by traversing the tree from the root to the appropriate leaf based on the attribute values of the instance.
Decision trees can be applied to a wide range of machine learning tasks, including classification, regression, and even feature selection. They are easy to interpret and visualize, making them useful for exploring and understanding the underlying structure of the data. However, decision trees can also be prone to overfitting if the tree is too deep, and may not perform well on complex tasks with large amounts of data. To address these issues, various techniques such as pruning and ensemble methods can be used. [10]

Gradient Boosting

Gradient boosting is a machine learning technique used in regression and classification tasks. It is a type of boosting algorithm that combines multiple weak learners to form a strong learner. In gradient boosting, each subsequent model focuses on the mistakes of the previous model and tries to correct them using a gradient descent algorithm. This process is repeated iteratively until a satisfactory level of accuracy is achieved.

Gradient boosting is a popular approach because it is capable of producing highly accurate predictions and can handle complex non-linear relationships between input features and the target variable. It is commonly implemented using decision trees as the weak learners, but other models such as linear models and neural networks can also be used.

However, gradient boosting can also be prone to overfitting if the number of trees is too high or the learning rate is too low. Therefore, it requires careful tuning and regularization to achieve the best possible performance. [10]

Random Forests

Random forests in machine learning is a popular supervised learning algorithm that combines the outputs of multiple decision trees to improve the accuracy and robustness of the predictions. Each decision tree in the forest is built on a subset of the training data and uses a random subset of the features. This randomness helps to reduce overfitting and increases the diversity of the models in the ensemble, which in turn helps to improve the accuracy of the predictions.
To make a prediction for a new input, the random forest algorithm takes an average or weighted average of the predictions of all the decision trees in the forest. Random forests are commonly used for both regression and classification tasks. [10]

Logistic Regression

Logistic regression in machine learning is a popular supervised learning algorithm used for classification problems, where the goal is to predict the probability of an observation belonging to a certain class or category. It is a form of regression analysis that models the relationship between a dependent binary variable and one or more independent variables.
In logistic regression, the probabilities are modeled using a logistic function or sigmoid curve that maps any real-valued input to a value between 0 and 1. The curve has an S-shape and provides a smooth transition between the two possible outcomes. The logistic regression algorithm uses the training data to estimate the parameters of the curve that best fit the data, and

then uses the resulting model to predict the probability of the target variable for new observations.
Logistic regression is a widely used machine learning algorithm due to its simplicity, interpretability, and effectiveness for many classification problems. It is often used in applications such as spam filtering, fraud detection, and medical diagnosis. [10]

Machine Learning and Physics

Machine learning and physics are related in several ways. One of the main applications of machine learning in physics is in data analysis, as large experimental and computational physics datasets can be analyzed using machine learning techniques. Machine learning can also be used to make predictions and identify patterns in physical systems, as well as to optimize the design of experiments or equipment.

There are several examples in the search results that demonstrate how machine learning classification models can be applied in physics. One example is in, where modern deep-learning techniques were used to improve the performance and speed of classification models for an experimental high-energy physics use case. [11]

Another example is in, where the performance of machine learning models was discussed during a recent physics run, specifically for cavity identification and fault detection.
Machine learning was applied in planetary space physics using random forest classification models. The k-Nearest Neighbors algorithm was trained using scattering models and target spectra in the mesopelagic using physics-informed machine learning. [12]

And rock physics was used to evaluate coal misclassification, and various machine learning models were used to classify the facies.
Overall, machine learning classification models are a powerful tool for analyzing large and complex datasets in physics, and have potential applications in many areas such as high energy physics, planetary physics, and more.

Python, Libraries and Modules [13]

Python is a popular and widely used programming language for machine learning for a variety of reasons, including:

1. Large set of libraries and frameworks: Python has a large number of libraries and frameworks that are specifically designed for machine learning tasks. These libraries provide a range of useful tools, algorithms and functions that can be used to process, analyze and visualize data.
2. Versatility and flexibility: Python is a general-purpose programming language that can support a broad range of tasks, making it very versatile. It is also very flexible, allowing developers to choose from a range of different tools and techniques to build and train machine learning models.

3. Ease of use: Python is a simple and easy-to-learn language, making it an ideal choice for beginners in machine learning. The language is readable and has a syntax that is easy to understand.
4. Fast and efficient: Python has several libraries and modules that are optimized for performance and speed, making it possible to process large amounts of data quickly.
5. Large community support: Python has a large and active community of developers, researchers, and practitioners who contribute to the development of libraries, frameworks, and tools for machine learning. This community provides a rich set of resources, tutorials, and documentation that make it easy to get started with machine learning in Python.

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. Some of the specific features of Pandas include:

- Data frame object for data manipulation with integrated indexing
- Tools for reading and writing data between in-memory data structures and different file formats
- Data alignment and integrated handling of missing data
- Reshaping and pivoting of data sets
- Label-based slicing, indexing, and subsetting of large data sets
- Time series-data functionality
- Filtering and selection of data based on categorical variables.

Seaborn is a Python data visualization library that is built on top of the Matplotlib library. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn is particularly useful for:

- Creating visually appealing statistical graphics with built-in color palettes and themes
- Visualizing linear models, time series, and categorical data
- Creating complex plots such as heatmaps, clustermaps, and facet grids
- Visualizing data distributions and relationships between variables Overall, Seaborn provides a powerful toolkit for exploratory data analysis and visualization in Python.

NumPy is a Python library used for working with arrays. It adds powerful data structures to Python that guarantee efficient operations with arrays, especially matrices, and sophisticated mathematical functions. NumPy is widely used in mathematical, scientific, and engineering applications, and provides an array object that is faster and more efficient than traditional Python lists. Some of the key features of NumPy include:

- Multi-dimensional array objects (ndarrays)
- Broadcasting functions
- Linear algebra, Fourier transforms, and random number generation

- Memory-efficient container programming
- C and Fortran integration capabilities

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a range of functions for creating 2D and 3D plots, graphs, histograms, and many other types of visualization. Matplotlib works well with NumPy and other numerical calculation libraries in Python. Some of the key features of Matplotlib include:

- Easy creation of line, scatter, and bar plots.
- Support for complex figures, including subplots, insets, and grids.
- Customization of plot labels, colors, and markers.
- Support for interactive plotting with the Qt or wxPython GUI toolkits.
- Extensive documentation and a large user community.

The **random** library in Python is a built-in module that provides functions for generating pseudo-random numbers. These functions are often used in simulations, games, and scientific applications to introduce randomness into a system.

The **scipy.stats** module is a sub-package of the SciPy library in Python which provides a wide range of functions for statistical analysis and probability distributions. Some of the operations that can be performed using scipy.stats include:

- Computation of basic statistical operations such as mean, median, mode, variance, standard deviation, etc.
- Fitting data to different probability distributions and estimating the parameters of those distributions.
- Performing a wide range of statistical tests such as t-tests, ANOVA, Chi-Square tests, and many others.
- Creating confidence intervals for different statistics such as mean, variance, correlation coefficient, etc.
- Generating random numbers from different probability distributions such as Normal, Poisson, Binomial, and many others.

spearmanr is a function available in the scipy.stats module in Python that computes the Spearman rank-order correlation coefficient, also known as the Spearman correlation coefficient. The Spearman correlation coefficient is a nonparametric measure of the monotonicity of the relationship between two variables, which means it measures the extent to which the relationship between two variables can be described using a monotonic function. scipy.stats.shapiro() is a function available in the scipy.stats module in Python that performs the Shapiro-Wilk test for normality. This test is used to determine if a random sample came from a normal distribution. The null hypothesis of the test is that the sample came from a normally distributed population.

The **warnings** module provides a way to issue warning messages when certain conditions arise in a program, without necessarily terminating the execution of the program. Warnings can indicate potential problems or issues that the developer should be aware of, but that do not necessarily require immediate attention or intervention.

variance_inflation_factor is a function in the statsmodels package that can be used to calculate the variance inflation factor (VIF) for a set of predictor variables in a regression model

add_constant is a function that can be used to add a constant column to a set of predictor variables in a regression model.
In many regression models, a constant value, also known as an intercept, is included in the linear equation. This is represented by a column of all 1's in the predictor matrix. By adding this constant column using the add_constant function, you can calculate the intercept in a regression model along with the coefficients of the predictor variables.

train_test_split() is a function in the Scikit-learn library (commonly abbreviated as sklearn) that is used to split a given dataset into two parts: a training set and a testing set. This is a very common technique used in machine learning, as it allows you to evaluate your model on data that it has not seen before.

StratifiedKFold is a function in Scikit-learn library that performs K-fold cross-validation. Unlike the regular K-fold cross-validation, StratifiedKFold returns stratified folds, which preserve the percentage of samples for each class.

GridSearchCV is a function in the Scikit-learn library (commonly abbreviated as sklearn) that performs a hyperparameter search. It searches through a specified grid of hyperparameters to find the best combination for a given model.

StandardScaler is a pre-processing step in Python's scikit-learn library that standardizes the features of a dataset. The purpose of StandardScaler is to scale the features of a dataset such that each feature has a mean of zero and a variance of one. It does this by computing the mean and standard deviation of each feature in the training set, and then transforming the features by subtracting the mean and dividing by the standard deviation. This transformation makes it easier to compare different features, and is particularly useful when using certain types of machine learning algorithms, such as linear regression and support vector machines. The transformed dataset is often referred to as a "standardized dataset".

In Python, **Counter** is a subclass of dict from the collections module that is used for counting hashable objects. It provides a way to count the occurrences of an item in a list or any other iterable object. The Counter object takes an iterable as input and returns a dictionary-like object with elements as keys and their count as values.

SMOTE stands for Synthetic Minority Over-Sampling Technique and it is a commonly used oversampling method to address the problem of class imbalance in machine learning. Specifically, SMOTE works by generating synthetic samples of the minority class in the dataset to balance out the classes.

In Python, **SMOTE** is implemented in scikit-learn as a class called SMOTE. After importing the SMOTE class from the imblearn.over_sampling module, you can create an instance of the SMOTE class and fit it to your training data. Once the SMOTE object is fitted, you can apply it to your training data to obtain a new, balanced dataset with oversampled minority classes.

accuracy_score, **f1_score**, **precision_score**, and **recall_score** are all metrics used to evaluate the performance of binary or multiclass classification models.
accuracy_score computes the accuracy of the classifier, which is the proportion of correct predictions among all predictions.
f1_score computes the harmonic mean of precision and recall, which is useful when you want to find a balance between precision and recall.
precision_score computes the proportion of true positive predictions among predictions that were classified as positive.
recall_score (also known as sensitivity or true positive rate) computes the proportion of true positive predictions among actual positive instances.
confusion_matrix is a table that summarizes the performance of a classification model, showing the number of true positives, false positives, true negatives, and false negatives.
classification_report is a text report that includes precision, recall, f1-score, and other metrics, as well as a summary of the number of examples in each class.
roc_curve computes the Receiver Operating Characteristic (ROC) curve, which is a plot of the true positive rate against the false positive rate at various classification thresholds.
roc_auc_score computes the Area Under the Curve (AUC) of the ROC curve, which is an aggregate measure of the performance of the classifier across various thresholds.
auc function computes the Area Under the Curve (AUC) using the trapezoidal rule.

Overall, these metrics are used to provide a comprehensive evaluation of the performance of classification models.

XGBClassifier in Python is a class in the XGBoost library that implements the gradient boosting algorithm for classification problems. It is a machine learning model that can classify input data into different categories based on the given features. The XGBClassifier allows you to specify parameters such as the maximum depth of the decision trees, the learning rate, and the number of trees in the ensemble. It is known for its speed and accuracy in solving large-scale classification problems, and provides flexibility and customizability in handling various tasks.

The **DecisionTreeClassifier** is a class in the sklearn.tree module of the Scikit-learn library in Python. It is a machine learning algorithm used for classification problems. The algorithm works by constructing a decision tree model that can predict the class label of a data point based on the feature values of that data point. The decision tree model is constructed by recursively partitioning the feature space into smaller and smaller regions, based on the values of the input features, until each region contains predominantly data points of a single class.
The DecisionTreeClassifier can handle both binary and multi-class classification problems.

The **RandomForestClassifier** is a class in the sklearn.ensemble module of the Scikit-learn library in Python. It is a machine learning algorithm used for classification problems, and is an extension of the decision tree algorithm.
Rather than using a single decision tree, the random forest algorithm constructs multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. Each decision tree in the random forest is constructed independently, using a randomly selected subset of the training data and a randomly selected subset of the input features.
During prediction, the class predicted by each tree is counted and the class with the most votes is returned as the final output.

SVC stands for Support Vector Classifier, which is a class in the sklearn.svm module of the Scikit-learn library in Python. SVC is a machine learning algorithm used for classification problems, particularly when the dataset is not linearly separable. The algorithm works by mapping the data points to a higher-dimensional space and finding the hyperplane that best separates them.

ELI5 is a Python package used for debugging machine learning classifiers and explaining their predictions. It provides support for various Machine Learning models such as decision trees, neural networks, and support vector machines. The package provides a unified API for various models, making it easy to visualize and debug the models' performance.

In Python, the time module provides various functions for working with time-related information. It can be used to represent different types of data such as seconds, dates, and calendars, and to perform various operations on them.

LogisticRegression is a classification algorithm in Python that is used for predicting the probability of a binary (two-class) or multi-class outcome. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression.
However, unlike linear regression which models a continuous dependent variable, logistic regression predicts the probability that an observation belongs to a certain category.
In scikit-learn, LogisticRegression is a class that provides an implementation of logistic regression. It is used to fit and predict binary and multiclass problems. It can also be used for regularization, which helps to prevent overfitting of data.

Grid search is a method for tuning the hyperparameters of a machine learning model in Python. It involves defining a grid of possible hyperparameter values for a given model and then exhaustively searching over that grid to find the best combination of hyperparameters that result in the highest performance of the model.

In other words, grid search simplifies the process of selecting the best hyperparameters for a model, as it allows you to simultaneously test multiple combinations and select the one that gives the most satisfactory results. The process of selecting hyperparameters using grid search is automated and more efficient for low-dimensional hyperparameter spaces.
The algorithm tests all the possible combinations of provided hyperparameters and returns a combination that results in the best performance of the model, as determined by an evaluation metric like accuracy or F1-score. Grid search is a widely used and effective method for hyperparameter optimization in machine learning.

Dataset

The Kaggle Competition "Give Me Some Credit" is a data science competition that was hosted on Kaggle in 2011. The goal of the competition was to predict the probability of a borrower going into financial distress within two years of being granted a loan. [14]

The dataset for this competition contains various features related to borrowers, such as their age, income, debt, credit history, and other financial and demographic characteristics. The target variable is a binary variable indicating whether or not the borrower experienced financial distress within two years of being granted a loan.

The dataset consists of two files: train and test. The train set contains approximately 150,000 observations, and the test set contains approximately 100,000 observations. Competitors were tasked with building a machine learning model based on the train set to predict the probability of default for the borrowers in the test set.

This competition aimed to improve on the state of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years. The various features included in the dataset are as follows:

- SeriousDlqin2yrs: This is the target variable which indicates whether the borrower experienced financial distress within two years of being granted a loan. It's a binary variable, with a value of 1 indicating that the borrower experienced financial distress, and 0 indicating that they did not.

- RevolvingUtilizationOfUnsecuredLines: This feature represents the amount of money the borrower owes on credit cards and other revolving credit accounts, as a proportion of their total credit limit.
- age: This feature represents the age of the borrower.
- NumberOfTime30-59DaysPastDueNotW: This feature represents the number of times the borrower has been delinquent (30-59 days past due) with their payments, but has not fully defaulted.
- DebtRatio: This feature represents the borrower's monthly debt payments, divided by their monthly gross income.
- MonthlyIncome: This feature represents the borrower's monthly income.
- NumberOfOpenCreditLinesAndLoans: This feature represents the number of open credit lines and loans the borrower has.
- NumberOfTimes90DaysLate: This feature represents the number of times the borrower has been delinquent (90 or more days past due) with their payments.
- NumberRealEstateLoansOrLines: This feature represents the number of real estate loans or lines of credit the borrower has.
- NumberOfTime60-89DaysPastDueNotW: This feature represents the number of times the borrower has been delinquent (60-89 days past due) with their payments, but has not fully defaulted.
- NumberOfDependents: This feature represents the number of dependents the borrower has.

Procedure:

I highly recommend that the code is followed as the procedure is read.
We start coding our program by importing all the necessary libraries and modules.

The dataset provided includes two .csv files: a training dataset and a test dataset.

It can be observed that the target variable has NaN values. NaN stands for "Not a Number" and is a special floating-point value that represents an undefined or unrepresentable value. In Python, NaN values are commonly used to represent missing or invalid data points in arrays or dataframes. NaN is a placeholder that is often used in numerical computations to indicate missing, undefined, or otherwise erroneous values. When performing computations on arrays or dataframes that contain NaN values, these values will often propagate through the computation, and it is important to recognize and handle them appropriately. These NaN values must be predicted.

The .info() method is used to print a concise summary of a Pandas dataframe. It displays information such as the total number of entries, the data type of each column, and the number of non-null values in each column. It can be a useful tool for understanding the structure and characteristics of a dataframe.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150000 entries, 1 to 150000
Data columns (total 11 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   SeriousDlqin2yrs                      150000 non-null  int64
 1   RevolvingUtilizationOfUnsecuredLines  150000 non-null  float64
 2   age                                   150000 non-null  int64
 3   NumberOfTime30-59DaysPastDueNotWorse  150000 non-null  int64
 4   DebtRatio                             150000 non-null  float64
 5   MonthlyIncome                         120269 non-null  float64
 6   NumberOfOpenCreditLinesAndLoans       150000 non-null  int64
 7   NumberOfTimes90DaysLate               150000 non-null  int64
 8   NumberRealEstateLoansOrLines          150000 non-null  int64
 9   NumberOfTime60-89DaysPastDueNotWorse  150000 non-null  int64
 10  NumberOfDependents                    146076 non-null  float64
dtypes: float64(4), int64(7)
memory usage: 13.7 MB
```

Figure 1 - Data Summary

It is observed that there are null values in MonthlyIncome and NumberOfDependents. These null values must be dealt with in order to use the data set for modelling.

The .describe() method in Python's pandas library computes descriptive statistics of a DataFrame or Series. It is used to get a summary of the distribution of continuous variables, including the count, mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th

percentile, and maximum. As observed below, the statistical distribution can be seen here for all the variables.

	SeriousDlqin2yrs	RevolvingUtilizationOfUnsecuredLines	age	NumberOfTime30-59DaysPastDueNotWorse	DebtRatio	MonthlyIncome	NumberOfOpenCreditLinesAndLoans	NumberOfTime
count	150000.000000	150000.000000	150000.000000	150000.000000	150000.000000	1.202690e+05	150000.000000	15
mean	0.066840	6.048438	52.295207	0.421033	353.005076	6.670221e+03	8.452760	
std	0.249746	249.755371	14.771866	4.192781	2037.818523	1.438467e+04	5.145951	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	
25%	0.000000	0.029867	41.000000	0.000000	0.175074	3.400000e+03	5.000000	
50%	0.000000	0.154181	52.000000	0.000000	0.366508	5.400000e+03	8.000000	
75%	0.000000	0.559046	63.000000	0.000000	0.868254	8.249000e+03	11.000000	
max	1.000000	50708.000000	109.000000	98.000000	329664.000000	3.008750e+06	58.000000	

Figure 2 - Data Statistics

The code print("Percentage of null values column wise:")
(df.isnull().sum()/df.shape[0])*100 computes the percentage of missing values (i.e., NaN or null values) for each column in a pandas DataFrame named df.
The .isnull() method returns a Boolean DataFrame of the same shape as df, where each element is True if the corresponding element in df is null, and False otherwise. Calling the .sum() method on this Boolean DataFrame returns the number of null values in each column of df. Dividing this count by the total number of rows in df (i.e., df.shape[0]) gives the proportion of null values in each column. Finally, multiplying this proportion by 100 gives the percentage of null values for each column.

```
Percentage of null values column wise:
SeriousDlqin2yrs                       0.000000
RevolvingUtilizationOfUnsecuredLines   0.000000
age                                    0.000000
NumberOfTime30-59DaysPastDueNotWorse   0.000000
DebtRatio                              0.000000
MonthlyIncome                         19.820667
NumberOfOpenCreditLinesAndLoans        0.000000
NumberOfTimes90DaysLate                0.000000
NumberRealEstateLoansOrLines           0.000000
NumberOfTime60-89DaysPastDueNotWorse   0.000000
NumberOfDependents                     2.616000
dtype: float64
```

Figure 3 - Percentage of null values in each feature

As observed, almost 20% null values are there in MonthlyIncome and 2.61% in NumberOfDependents. We will be inputting them on the basis of what type of distribution they have.

We then proceed by writing code implementing user defined functions to observe the distribution of each of the features. These user defined functions will be used for plotting, for finding the distributions, for checking normality, performing k-fold cross validation, plotting ROC-AUC curves and other necessary tasks that will be explained as we move forward.

The plots for feature vs frequency for each of the feature are displayed below:

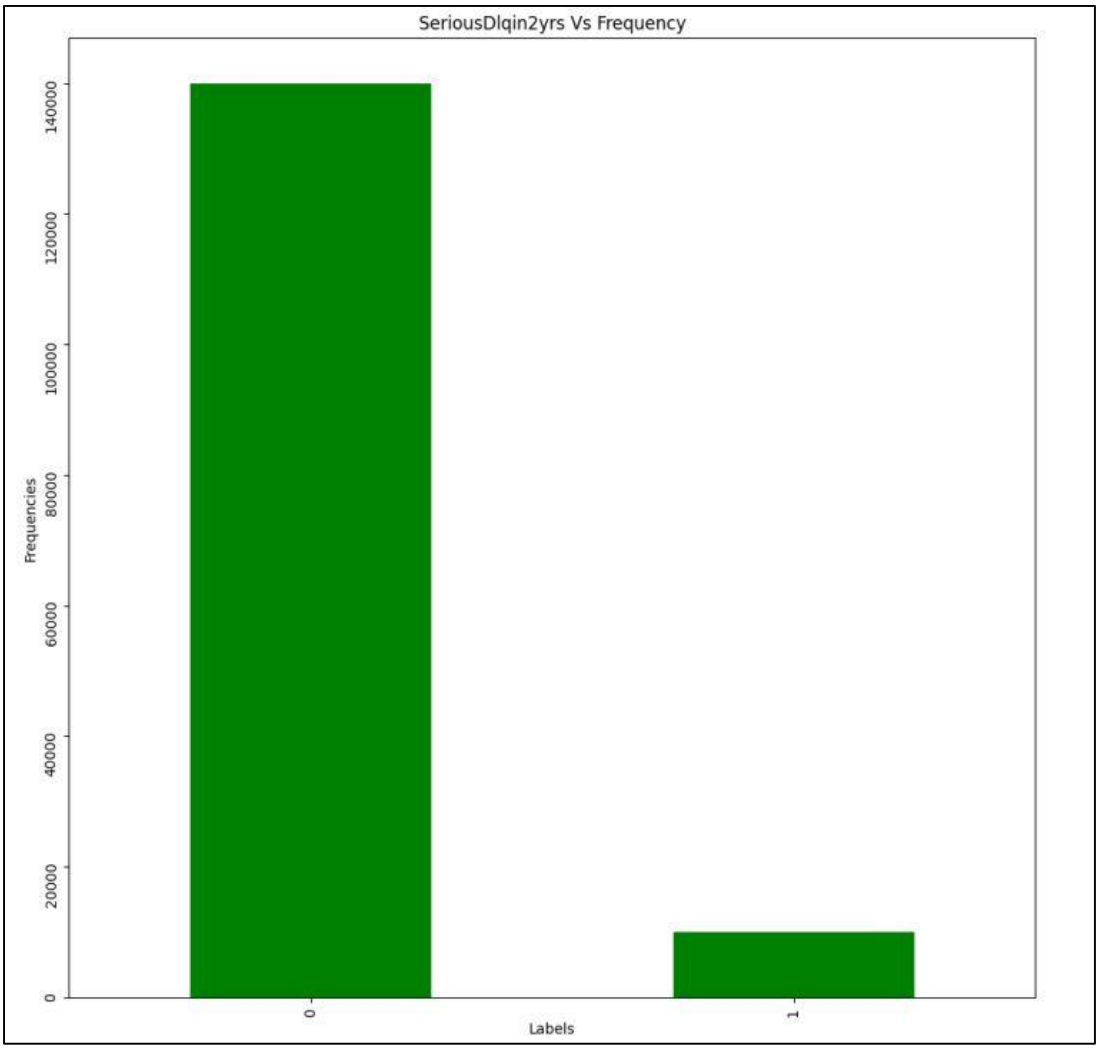

Figure 4 - SeriousDlqin2yrs vs Frequency

Feature "SeriousDlqin2yrs" has highest frequency for the class: 0 Feature "SeriousDlqin2yrs" has lowest frequency for the class: 1

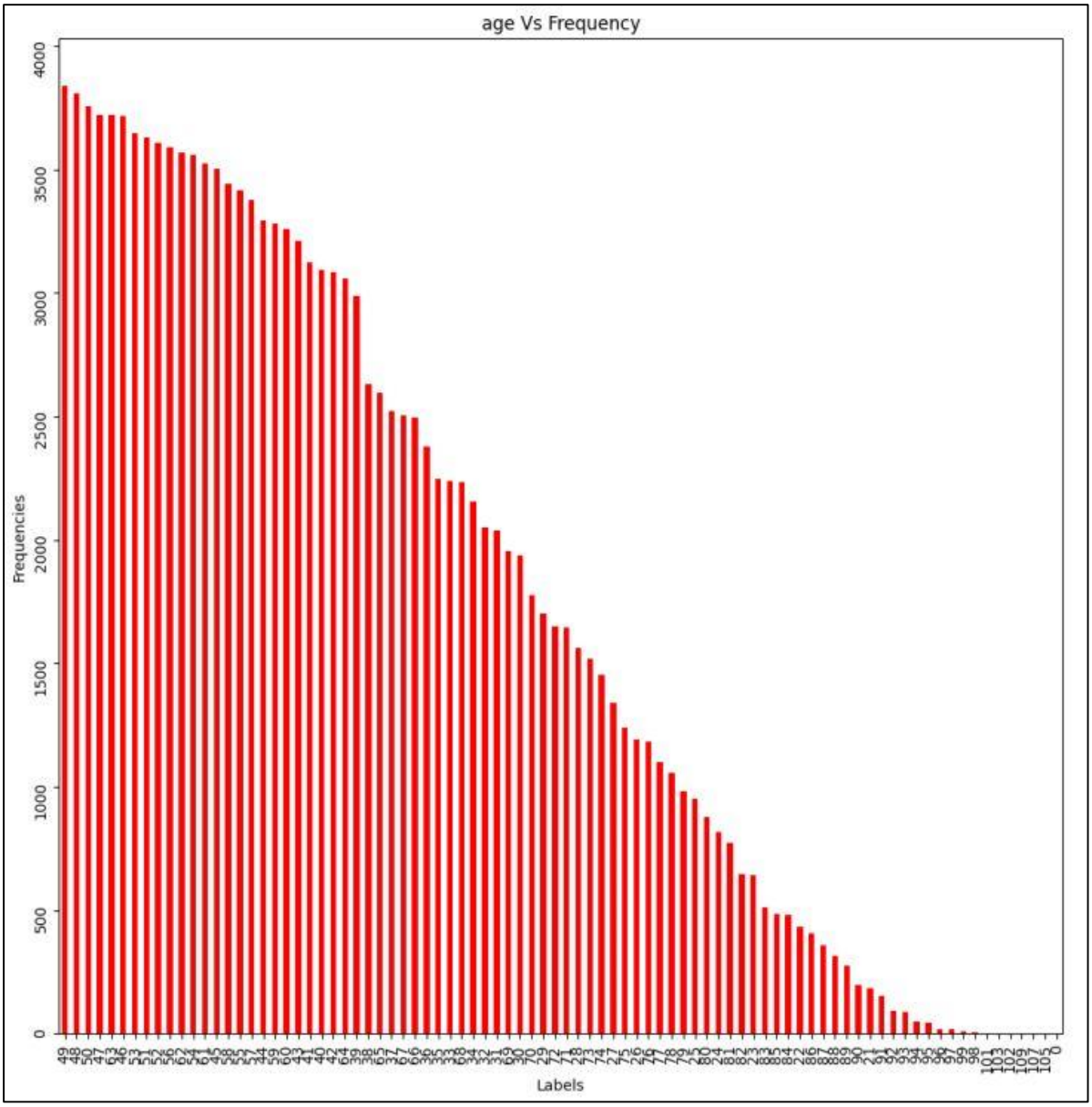

Figure 5 - Age vs Frequency

Feature "age" has highest frequency for the class: 49
Feature "age" has lowest frequency for the class: 0

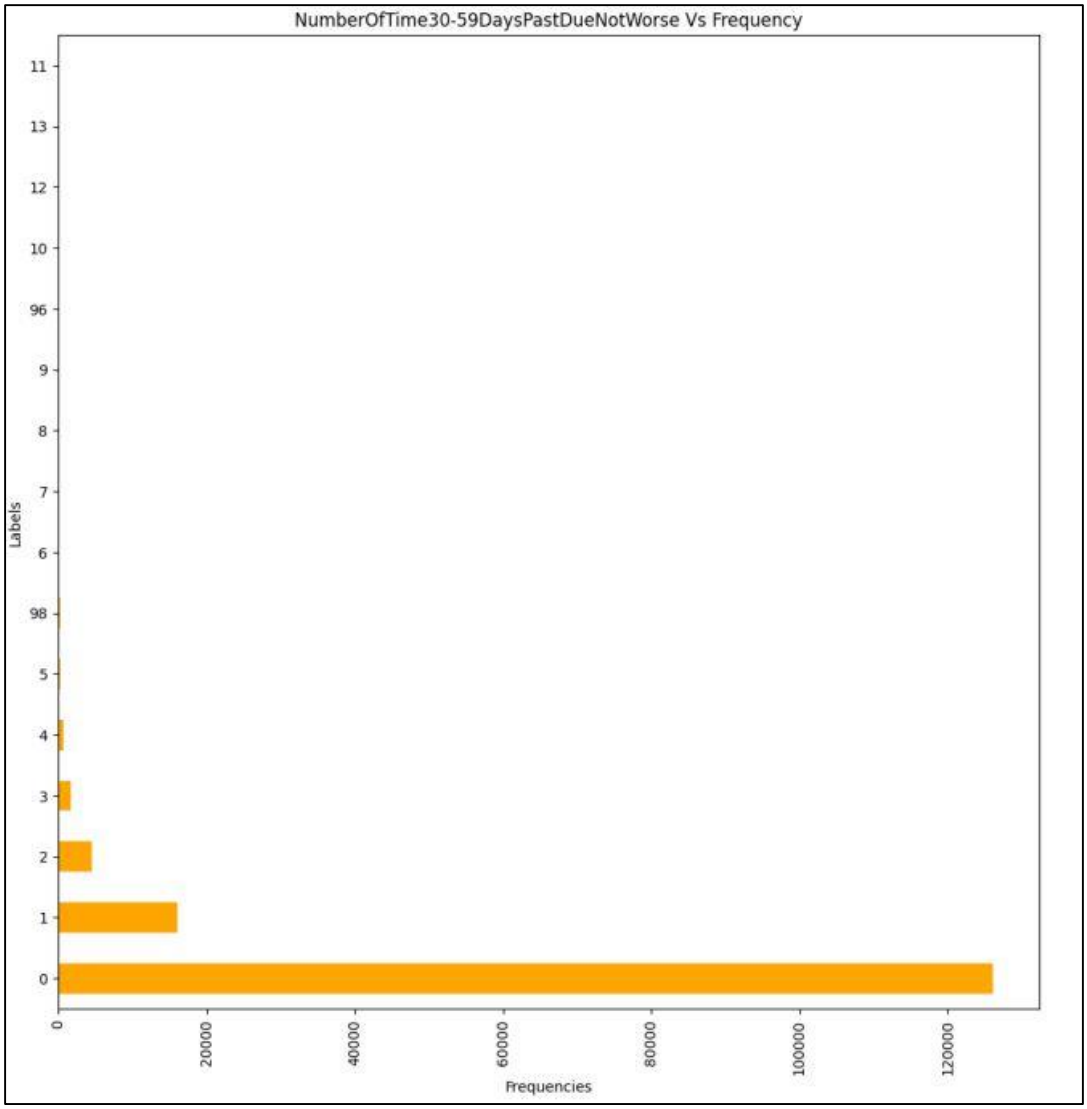

Figure 6 - NumberOfTime30-59DaysPastDueNotWorse vs Frequency

Feature "NumberOfTime30-59DaysPastDueNotWorse" has highest frequency for the class: 0
Feature "NumberOfTime30-59DaysPastDueNotWorse" has lowest frequency for the class: 11

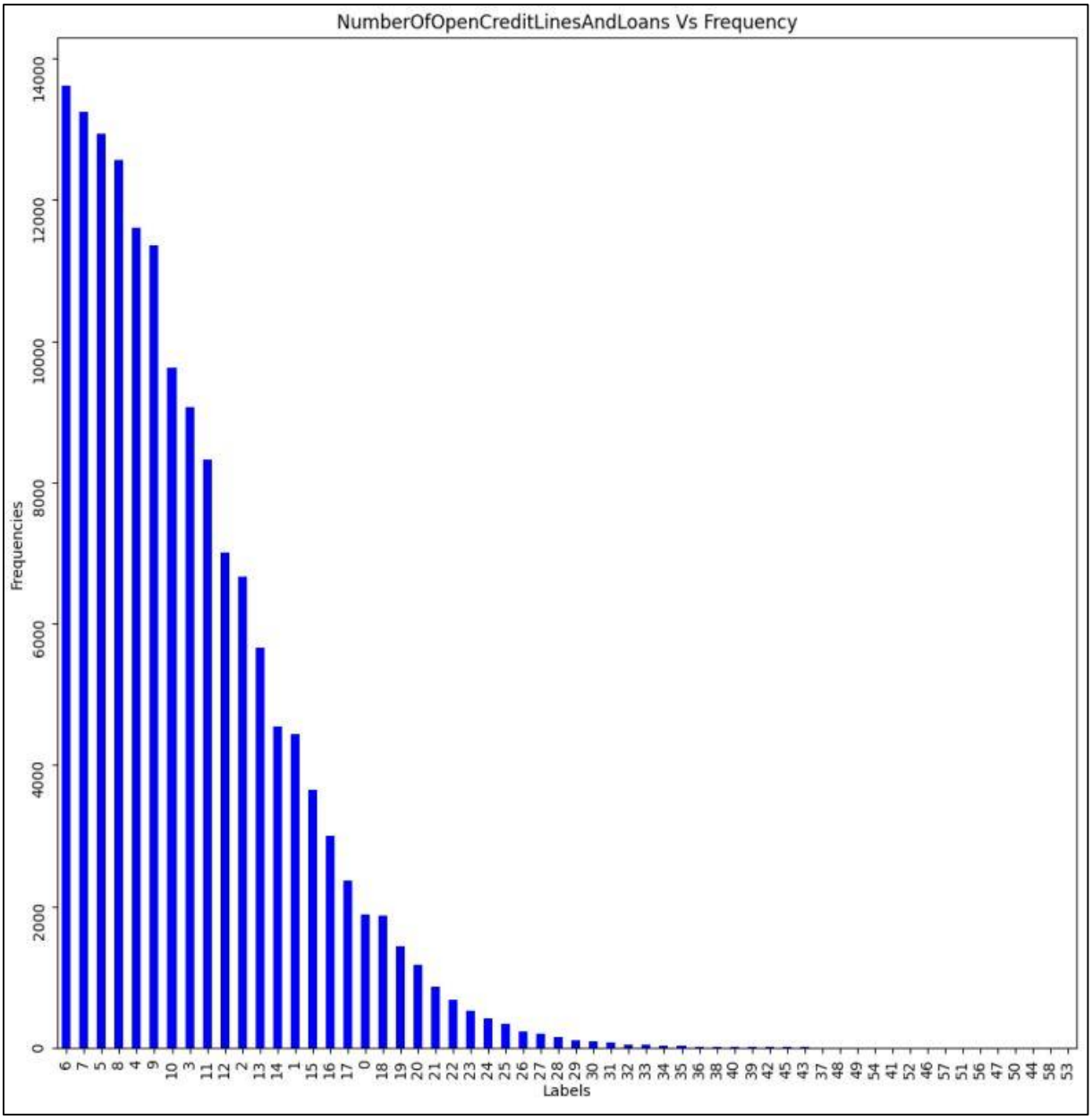

Figure 7 - NumberOfOpenCreditLinesAndLoans vs Frequency

Feature "NumberOfOpenCreditLinesAndLoans" has highest frequency for the class: 6
Feature "NumberOfOpenCreditLinesAndLoans" has lowest frequency for the class: 53

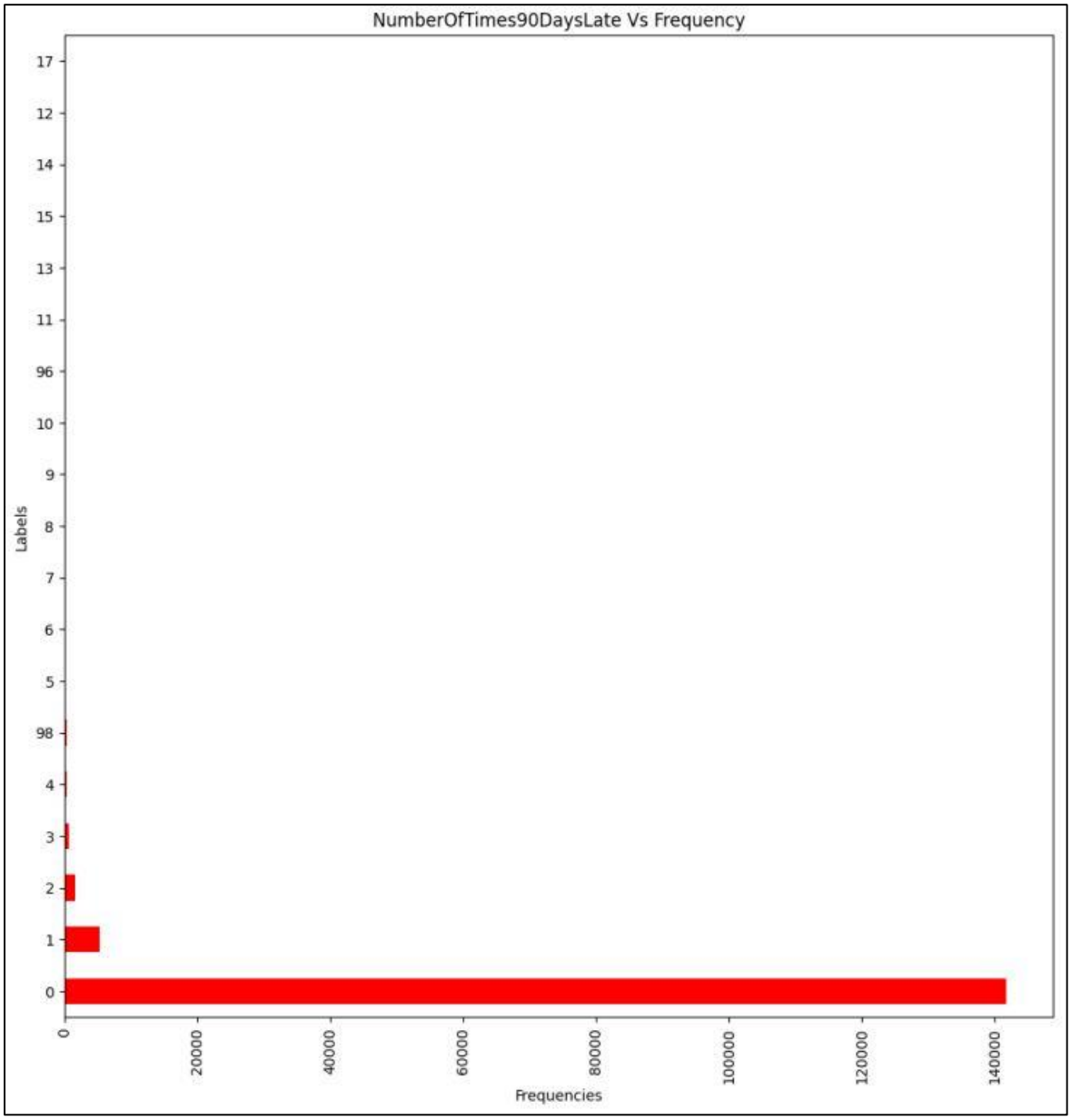

Figure 8 - NumberOfTimes90DaysLate vs Frequency

Feature "NumberOfTimes90DaysLate" has highest frequency for the class: 0
Feature "NumberOfTimes90DaysLate" has lowest frequency for the class: 17

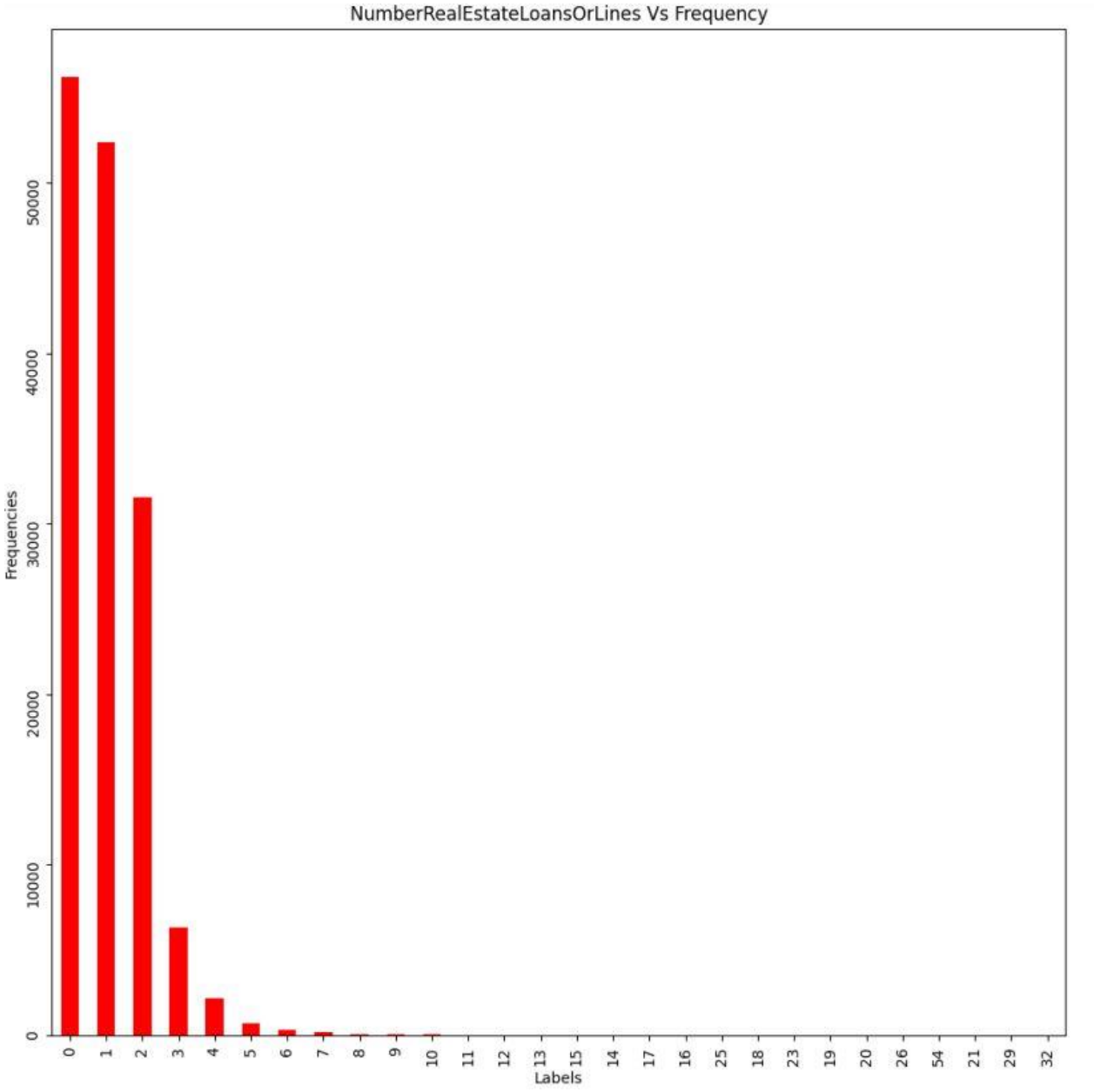

Figure 9 - NumberRealEstateLoansOrLines vs Frequency

Feature "NumberRealEstateLoansOrLines" has highest frequency for the class: 0
Feature "NumberRealEstateLoansOrLines" has lowest frequency for the class: 32

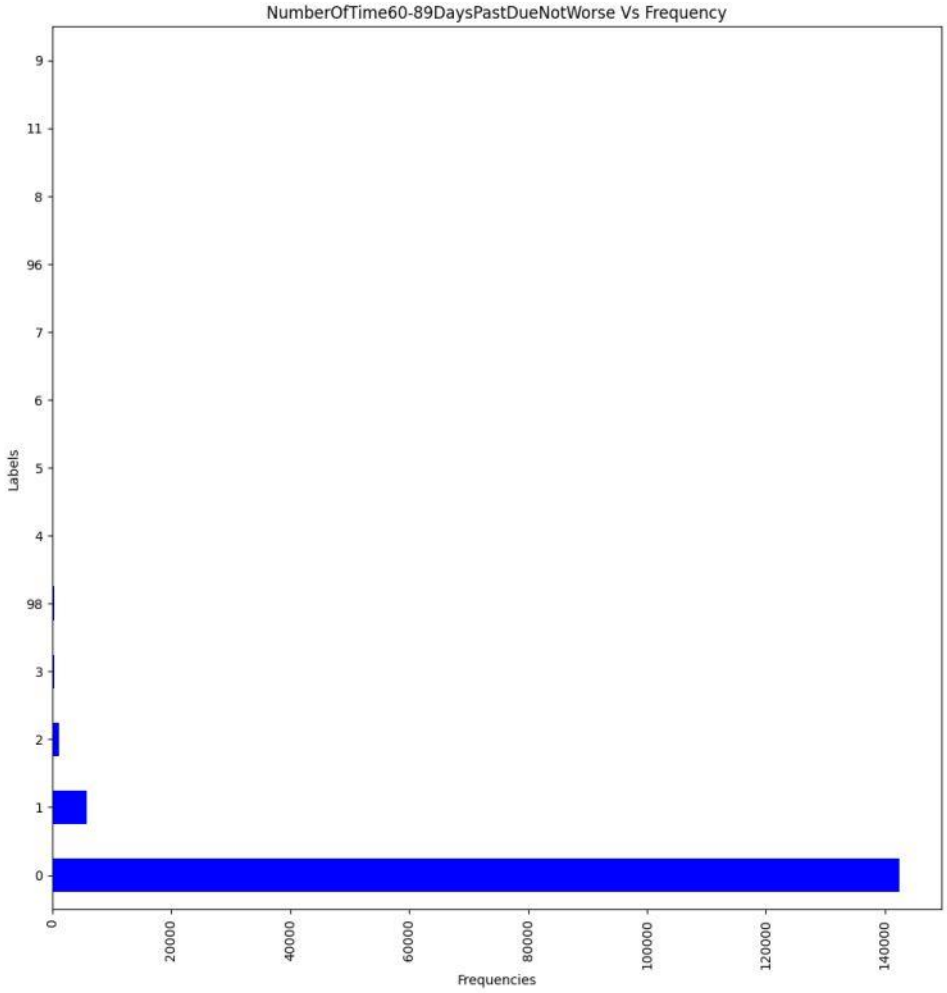

Figure 10 - NumberOfTime60-89DaysPastDueNotWorse vs Frequency

Feature "NumberOfTime60-89DaysPastDueNotWorse" has highest frequency for the class: 0
Feature "NumberOfTime60-89DaysPastDueNotWorse" has lowest frequency for the class: 9

Now, we perform a normality test on our features to check if the feature if normally distributed or not.

'RevolvingUtilizationOfUnsecuredLines' is not normally distributed. Hence, (reject H0), alternate hypothesis is true.

'DebtRatio' is not normally distributed. Hence, (reject H0), alternate hypothesis is true.

'MonthlyIncome' is normally distributed. Since here is greater than a 5% chance of a result as extreme as the sample result when the null hypothesis is true. (fail to reject H0)

Then, we observe the distribution of the NumberOfDependents. And we find that the skewness for the column NumberOfDependents is: 1.5882423788858833.

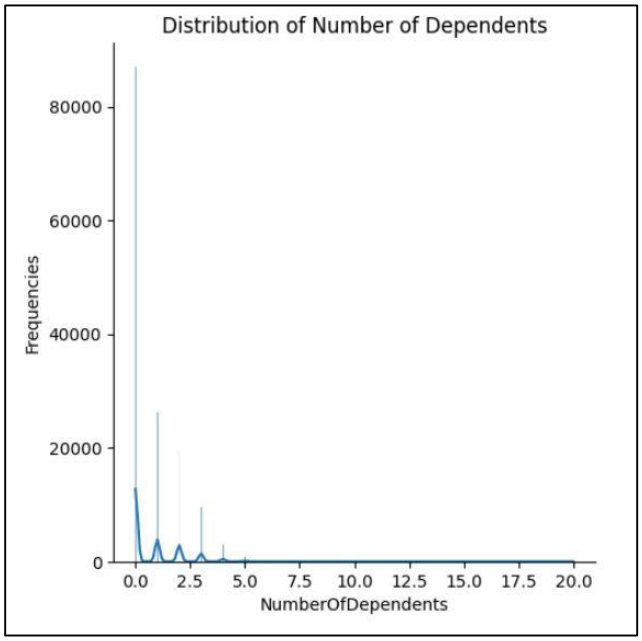

Figure 11 - Distribution of Number of Dependents

This is a right skewed distribution.

Then, we observe the distribution of MonthlyIncome and we find that the skewness for the column MonthlyIncome is: 114.0403179452332.

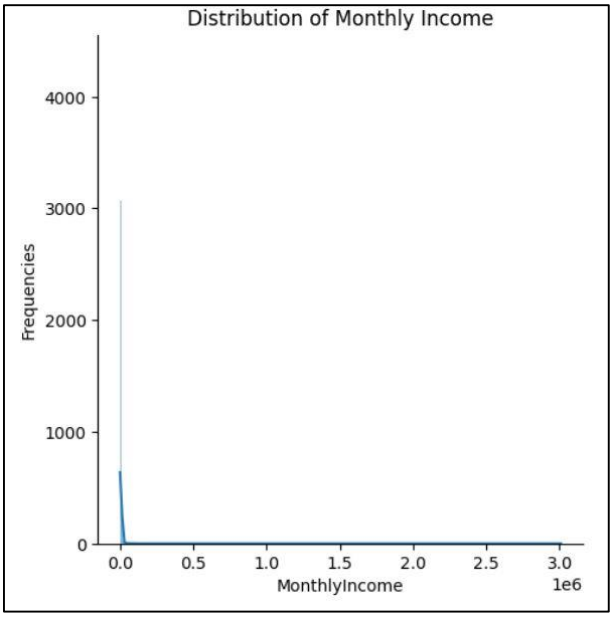

Figure 12 - Distribution of Monthly Income

This is a right skewed distribution.

Since both columns are skewed, lets proceed by imputing their null values with their median value.

We would be modelling on tree based models which are not affected by outliers thus we can skip the removal of outliers. Then, we continue by producing a heat-map of the correlation matrix for the variables in df, with cells colored according to their correlation values and annotated with the correlation coefficient values.

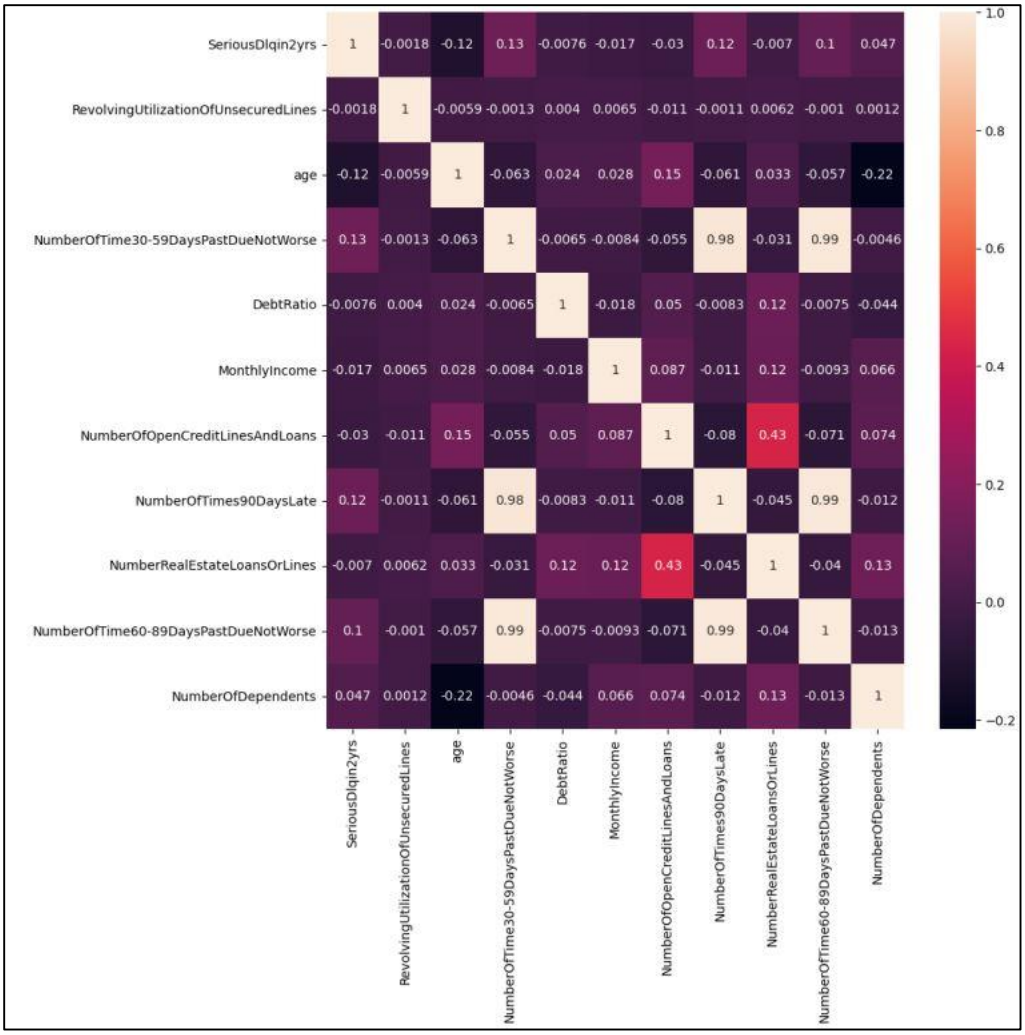

Figure 13 - Heat Map of Correlation Matrix

We observed that NumberOfTime30-59DaysPastDueNotWorse, NumberOfTimes90DaysLate, NumberOfTime60-89DaysPastDueNotWorse have a very strong correlation. That is greater than 0.9.

It is generally considered to be good practice to avoid including correlated features in the dataset as these features will likely not bring any additional information of value and instead increase the complexity of the algorithm and hence, increase the likelihood of errors. But instead of blatantly dropping these features, we must check the Variance Inflation Factor (VIF) first.

The idea is to keep on removing the feature giving the greater VIF until the interaction decreases:

	Variable	VIF
0	NumberOfTime30-59DaysPastDueNotWorse	40.220761
1	NumberOfTimes90DaysLate	72.341097
2	NumberOfTime60-89DaysPastDueNotWorse	91.128082

Figure 14 - Initial VIF

VIF has dropped, we need to get it under 5 as when VIF < 5, this indicates low correlation among variables under ideal conditions. This will guarantee that multicollinearity is not an issue in the model.

	Variable	VIF
0	NumberOfTime30-59DaysPastDueNotWorse	30.744857
1	NumberOfTimes90DaysLate	30.744857

Figure 15 - VIF after removing NumberofTime60-89DaysPastDueNotWorse

Now, VIF is under 5, thus we would have to drop those two features.

	Variable	VIF
0	NumberOfTime30-59DaysPastDueNotWorse	1.0

Figure 16 - VIF < 5

Although, multicollinearity is not an issue with tree based models, we can continue without dropping these variables. The above procedure has been performed for purposes of analysis.

Since the ranges for the features are not uniform, we would need to perform certain scaling techniques.

K-fold cross-validation is a technique for assessing the performance of a machine learning model. In K-fold cross-validation, the dataset is split into k subsets or folds of approximately equal size. The model is trained on k-1 of these folds and tested on the remaining fold. This process is repeated k times, with each of the k subsets used exactly once as the validation data. The results of the k-folds are then averaged to produce a single estimate of model performance. We are going to perform cross-validation to get an average result on 5 folds rather than just 1 fold.
The code X, y = df.drop("SeriousDlqin2yrs", axis = 1), df["SeriousDlqin2yrs"] is used to split a given dataframe 'df' into two parts - one containing all the features (stored in 'X') and one containing the target variable value (stored in 'y').

Now, we are ready to proceed with modelling.

Model Results and Discussion

Decision Tree Baseline Model

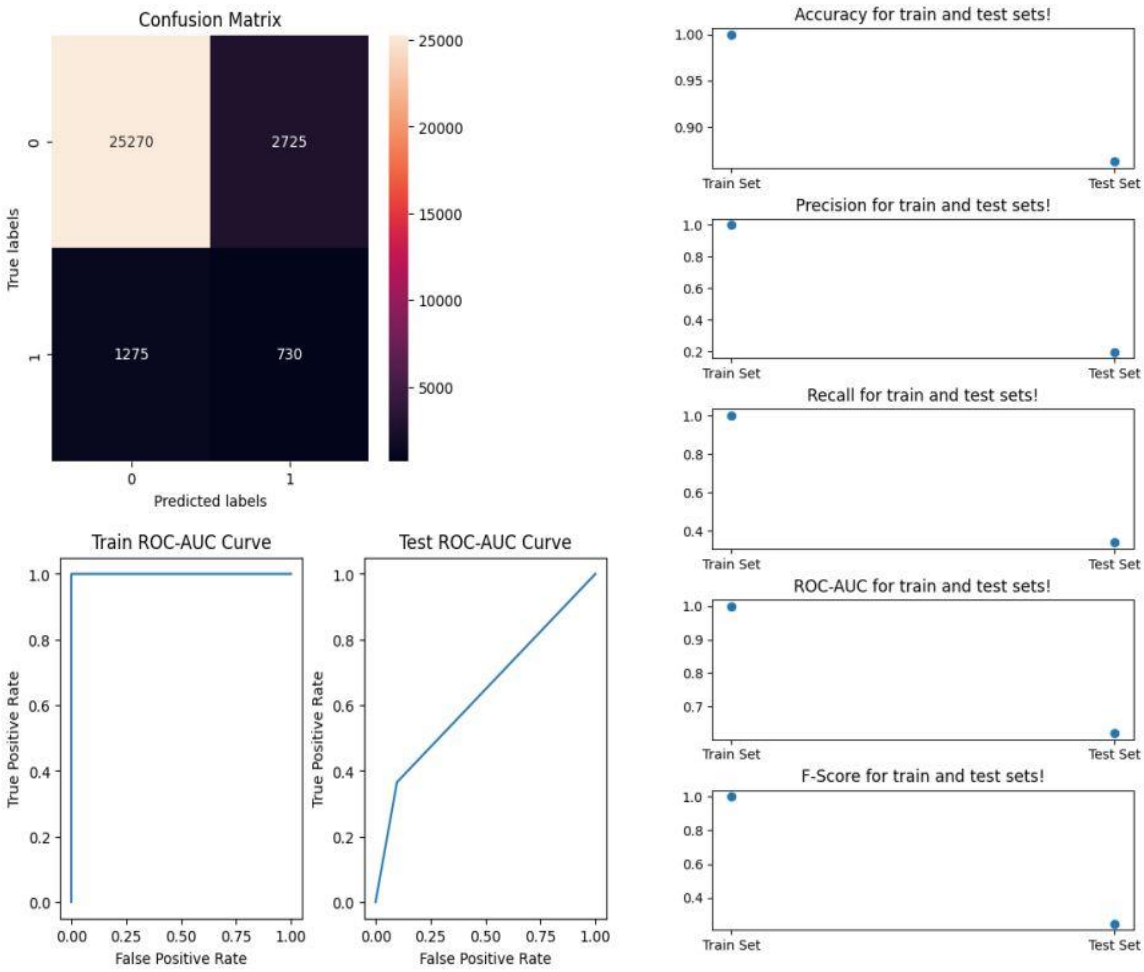

Figure 17 - Decision Tree Baseline Model Results

The model appears to be performing very well on the training set with close-to-perfect accuracy and other metrics. However, its performance drops significantly on the test set, indicating poor generalization to new data. The accuracy on the test set is much lower than the training set, which highlights its inability to generalize. Additionally, precision, recall, F-score, and ROC AUC score on the test set are also much lower, indicating that the model is making many more false positive and false negative predictions on unseen data. The results suggest that the model is overfitting and too specific to the training set.

Optimized Decision Tree

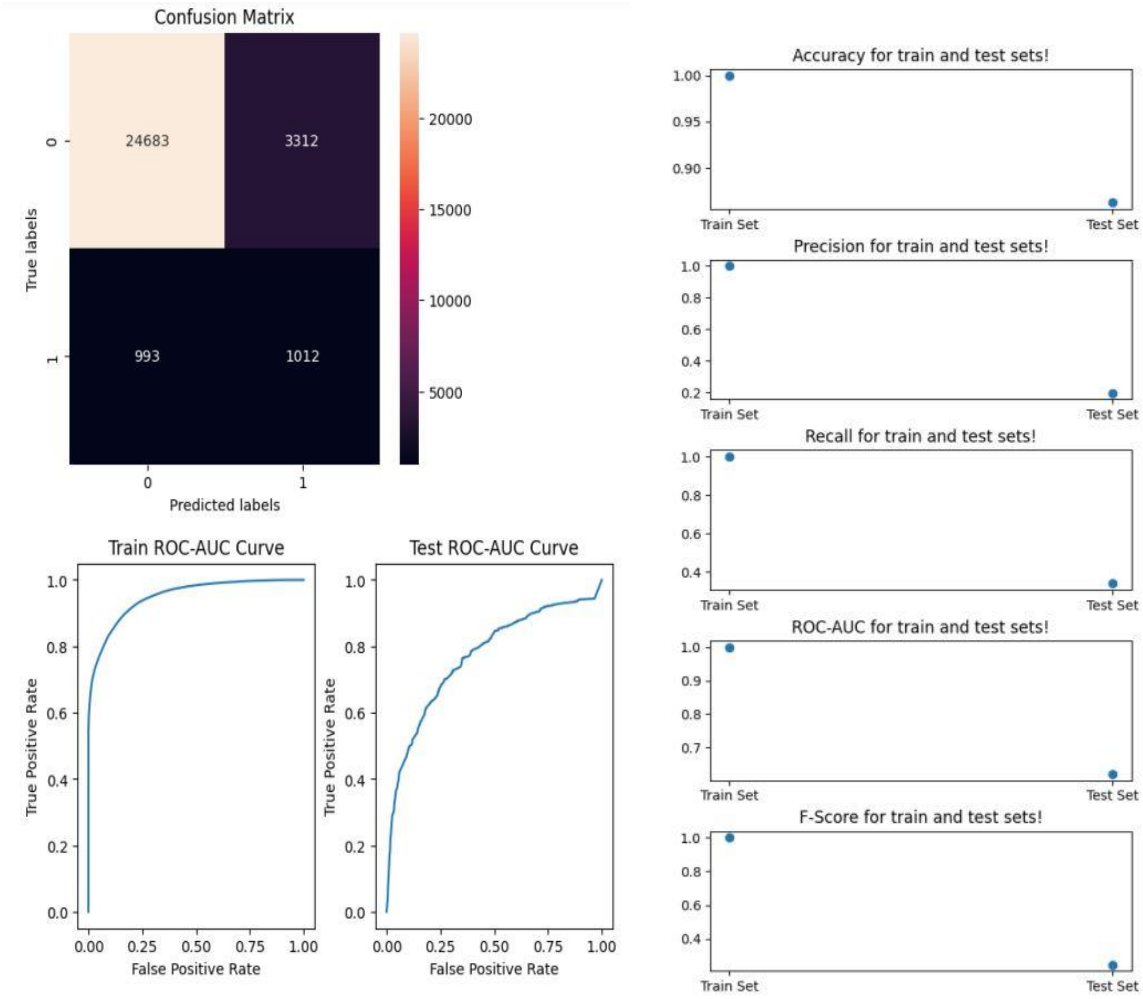

Figure 18 - Optimized Decision Tree Results

We can see that the model achieved higher accuracy and precision on the train set than on the test set. This suggests that the model may have overfit to the train data and is not generalizing well to new, unseen data.
Furthermore, the recall on the test set is notably higher than on the train set. This may suggest that the model is identifying more true positives on the test set than on the train set, which is a positive indication of generalization.
The F-score on the test set is lower than on the train set, which may also suggest overfitting to the train data.
Finally, the ROC AUC score on the test set is notably lower than on the train set, which is another indication of overfitting.

XGBoost Baseline Model

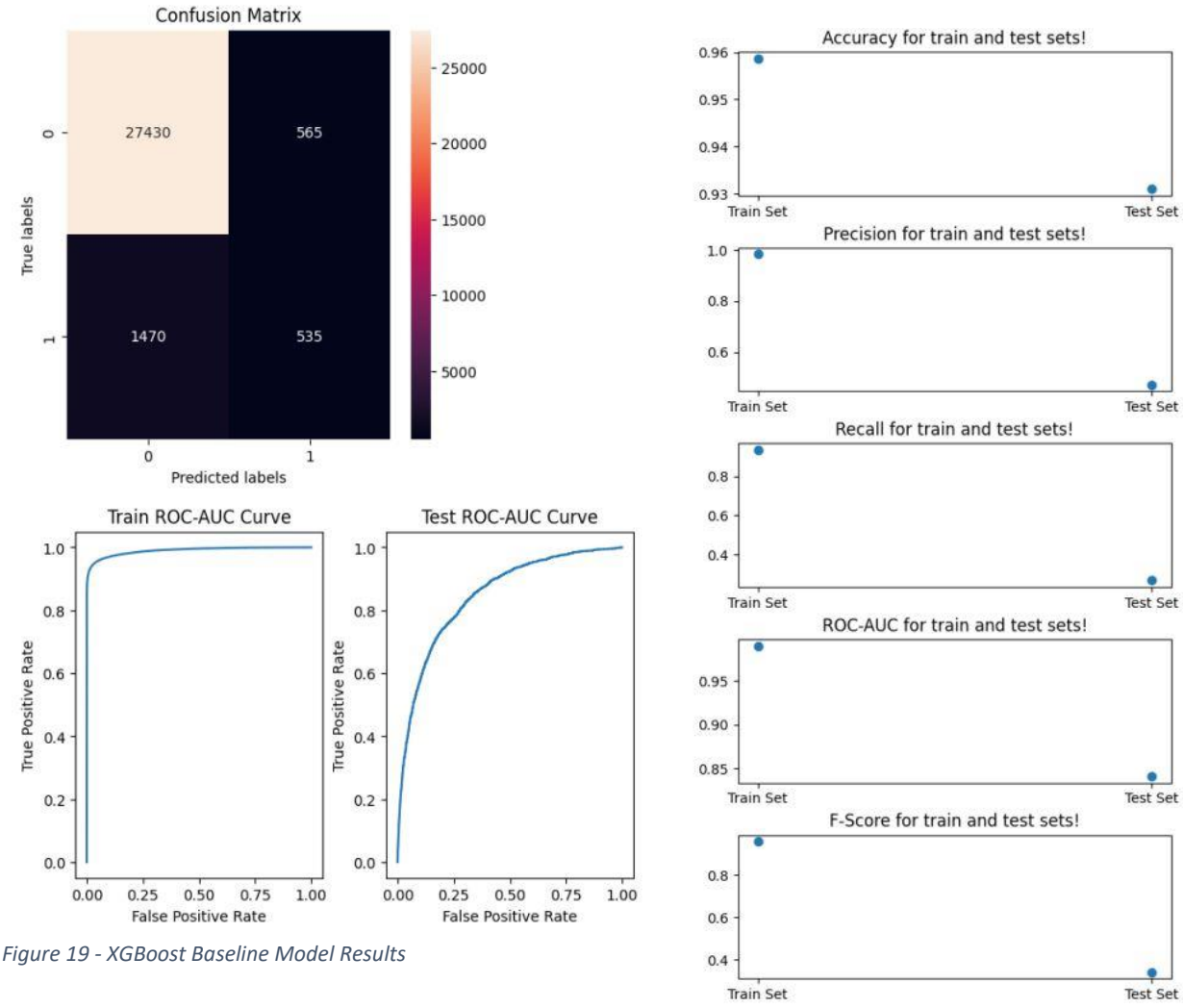

Figure 19 - XGBoost Baseline Model Results

Based on the sets of results provided, it seems that the model performs well on the train set, with high accuracy, precision, recall, and F score, as well as a high ROC score. However, it seems that the model does not perform as well on the test set, with notably lower accuracy, precision, recall, and F score, as well as a lower ROC score.

This suggests that the model may be overfitting to the training data, meaning that it is able to fit the training data very well but does not generalize as well to new, unseen data. The discrepancy between the performance on the train set and test set is an indication that the model is too complex compared to the amount of available data, and is capturing noise from the data which leads to poor performance on validation data.

Optimized XGBoost

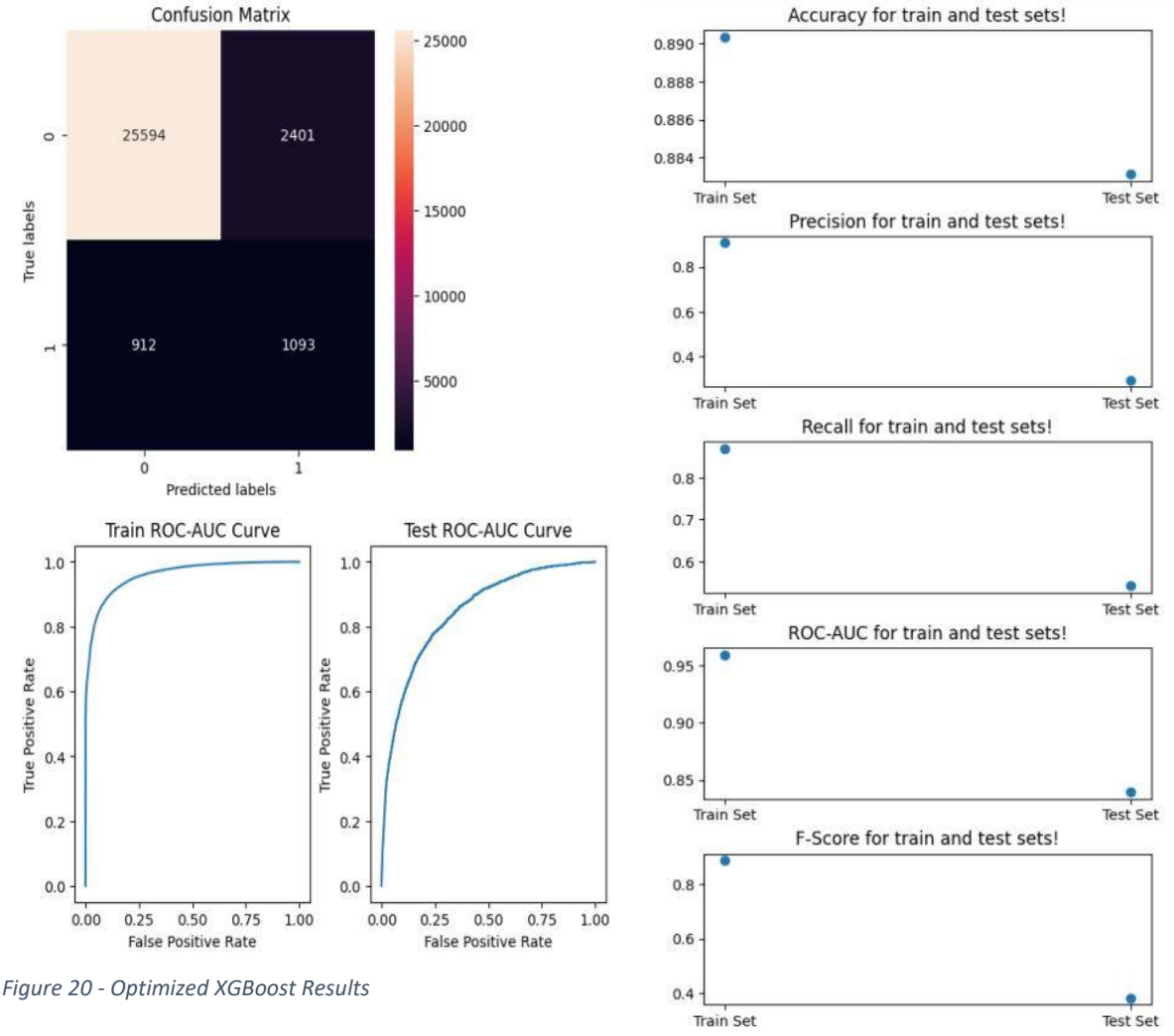

Figure 20 - Optimized XGBoost Results

It appears that the model is performing better on the training set than on the test set. This could indicate that the model is overfitting to the training data, which means it is not generalizing well to new data.

The accuracy, precision, recall, and F-score metrics all indicate how well the model is performing in terms of identifying positive and negative cases. It's good to see that the accuracy and F-score are high on both sets, but the precision and recall are noticeably lower on the test set. This again suggests that the model may not be generalizing well to new data. The results show that the model is performing well on the training set with high ROC values, but less so on the test set.

Random Forest Baseline Model

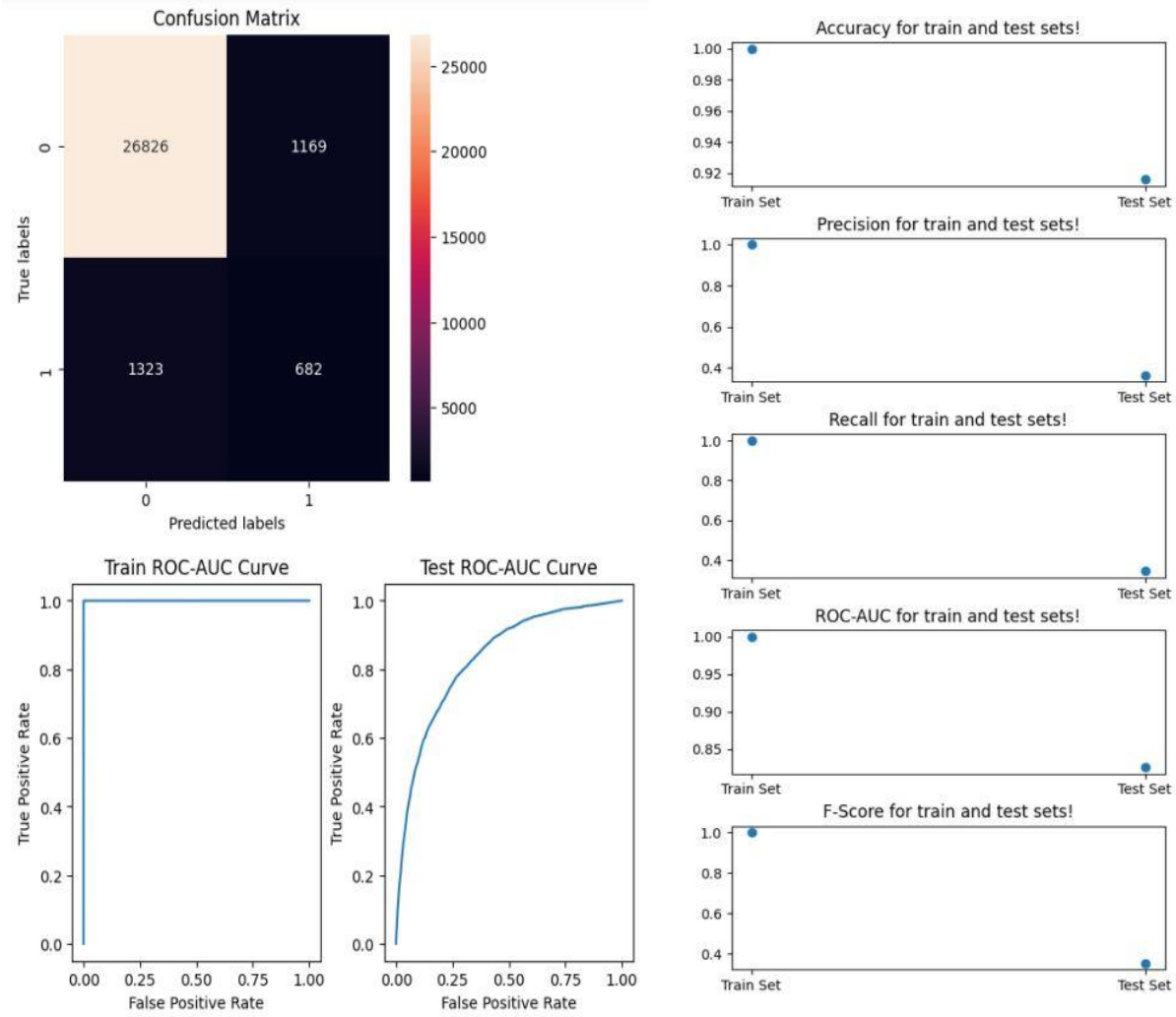

Figure 21 - Random Forest Baseline Model Results

The model seems to be performing very well on the training set with near-perfect scores and a high ROC value. However, on the test set, the model's performance drops noticeably lower, especially in terms of precision, recall and F-score. This may indicate that the model is overfitting to the training data and needs further tuning to generalize better to new data.
The ROC curve confirms this as well, with a high value on the training set but a lower value on the test set. This may indicate that the model is not able to accurately balance between true positives and false positives for new data, which could affect its use in real-world applications.

Optimized Random Forest

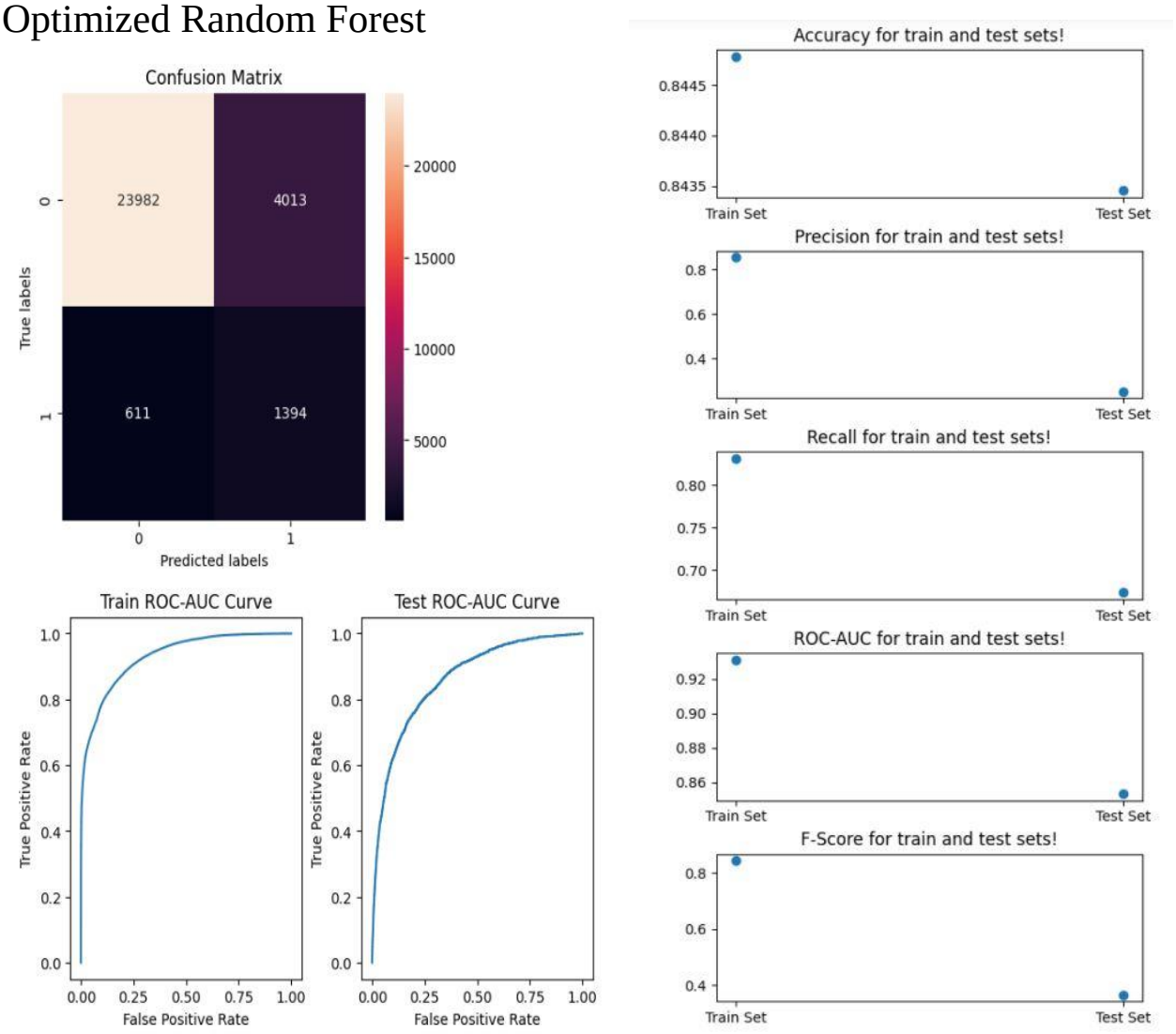

Figure 22 - Optimized Random Forest Results

The model appears to perform quite well on the training set, with consistently high accuracy, precision, recall, and F score values, as well as a high ROC value. However, the performance on the test set is not as strong, with lower values for all metrics except for recall.
This suggests that the model may be overfitting to the training data, resulting in poor generalization to new data. To improve the performance on the test set, some potential strategies could include using more data for training, using regularization techniques to prevent overfitting, or adjusting hyperparameters such as learning rate or the number of layers in the model architecture.
It's also worth noting that the precision value on the test set is unusually low, which may indicate a class imbalance issue or other problem with the data.

Logistic Regression

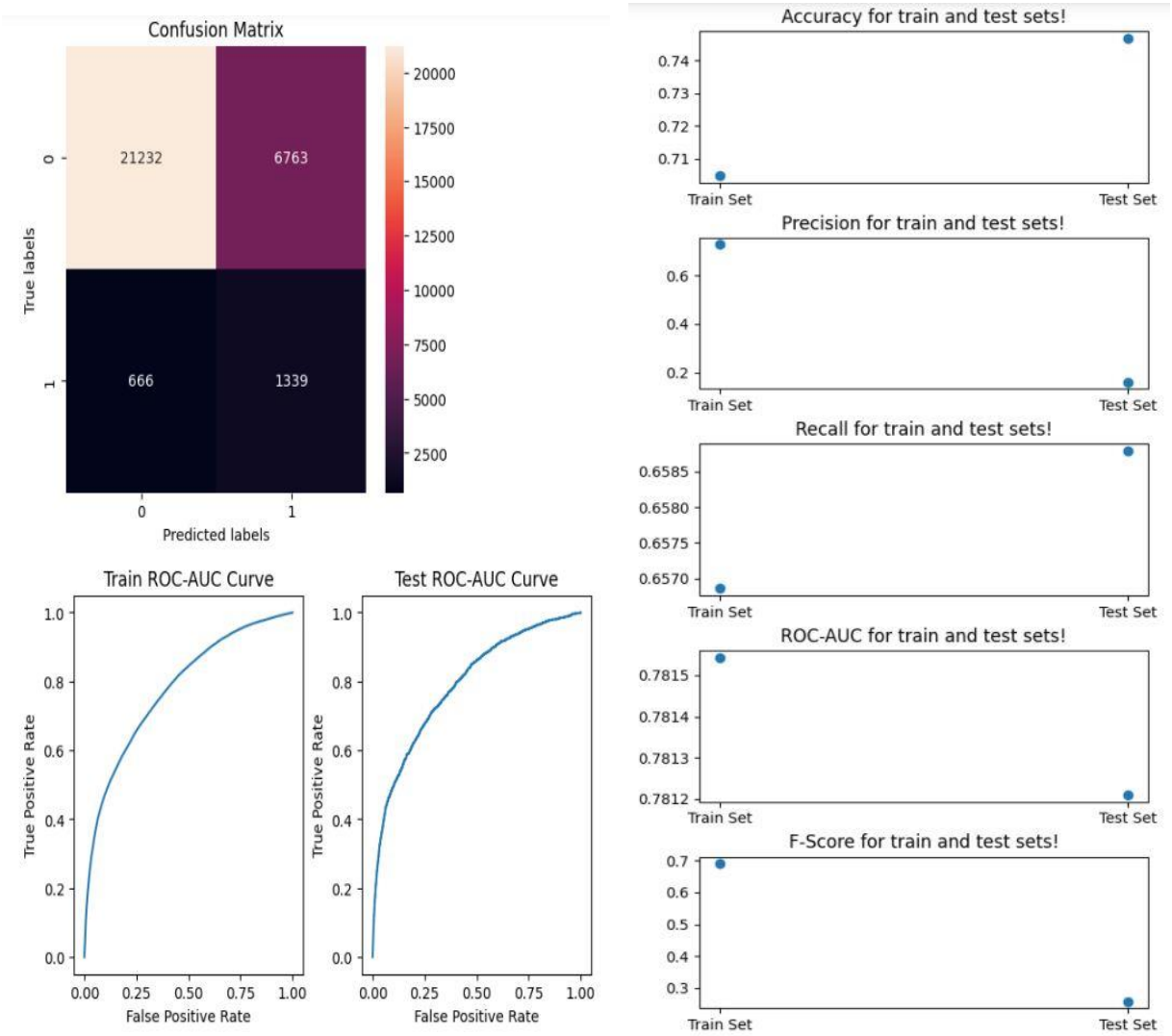

Figure 23 - Logistic Regression Results

We can see that the mean accuracy of the model on the test set is slightly higher than on the training set, which is a good sign as it indicates the model has not overfit on the training set. However, the mean precision score on the test set is considerably lower than on the training set, suggesting the model may not be generalizing well to new data.

The mean recall scores are similar for both training and test sets, which indicates that the model is correctly identifying both positive and negative instances with similar accuracy. The F1-scores are similar to the accuracy scores, suggesting that the model is performing well in correctly identifying both classes.

The mean ROC scores on both sets are similar, indicating that, overall, the model has good predictive ability in distinguishing between the positive and negative classes.

Model Results Summary Table

model_name	accuracy_train	accuracy_test	precision_train	precision_test	recall_train	recall_test	f1_train	f1_test	roc_train	roc_test
LogisticRegression	0.704831	0.746727	0.726606	0.160470	0.656859	0.658788	0.689959	0.258060	0.781543	0.781210
DecisionTreeClassifier	0.999795	0.862440	0.999848	0.195375	0.999743	0.339120	0.999795	0.247911	1.000000	0.619921
DecisionTreeClassifier_optimized	0.871390	0.853187	0.886513	0.232580	0.852046	0.517152	0.868848	0.320406	0.950428	0.761700
XGBClassifier	0.958714	0.931013	0.982611	0.472112	0.933956	0.267306	0.957666	0.341245	0.989744	0.841241
XGBClassifier_optimized	0.890350	0.883140	0.908029	0.296217	0.868710	0.542290	0.887925	0.383039	0.959582	0.839658
RandomForestClassifier	0.999784	0.915793	0.999755	0.363020	0.999812	0.343806	0.999784	0.353113	0.999985	0.825573
RandomForestClassifier_optimized	0.844784	0.843453	0.854489	0.250542	0.831099	0.673947	0.842630	0.365273	0.931241	0.853227

Figure 24 - Model Results Summary Table

Conclusion

ROC AUC is considered to be a good measure for classification problems such as credit scoring because it provides an overall measure of model performance that takes both true positive rate and false positive rate into account. In credit scoring, it is important to correctly identify customers who are likely to default and label them as such, while also avoiding labelling customers who are not likely to default as defaulters. ROC AUC is sensitive to these scenarios, and provides a measure of how well a model can differentiate between positive and negative instances at various probability thresholds. Additionally, ROC AUC is insensitive to imbalanced classes, which is often the case with credit scoring datasets. [15]

According to the results, the logistic regression model has the highest ROC AUC score of 0.781543 on the training data and 0.781210 on the test data. The ROC AUC score measures the ability of the classifier to distinguish between positive and negative classes, where a score of 1 indicates perfect classification and a score of 0.5 indicates random classification.

Here, the logistic regression model's ROC AUC score indicates that the model is able to effectively distinguish between the two classes. Additionally, the precision, recall, and F1 scores of the logistic regression model are also relatively high compared to the other models, indicating a good balance between the true positive and false positive rates.

Overall, based on the provided search results, it seems that the logistic regression model is the best choice in this scenario as it achieves the highest ROC AUC score and a good balance of precision, recall, and F1 scores.

References

[1] Hand, D. & Henley, W. Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society: Series A (Statistics in Society). 1997.

[2] Hand, D. Modelling consumer credit risk. IMA Journal of Management Mathematics. 2001.

[3] Thomas, L.C., Edelman, D.B. & Crook, J.N. Credit scoring and its applications. Society for Industrial and Applied Mathematics. (2002).

[4] L.C. Thomas, A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers, Int. J. Forecast. 2000.

[5] Thomas LC. Credit scoring and its applications. (2002)

[6] Joshi AV. Machine Learning and artificial intelligence (2019).

[7] Mira, J. Symbols versus connections: 50 years of artificial intelligence. 2008.

[8] Cook D. & Holder L. A client-server interactive tool for integrated artificial intelligence. 2001.

[9] Ni H. An introduction to machine learning in quantitative finance (2021)

[10] Hand, D. & Henley, W. Statistical classification methods in consumer credit scoring: A review. Journal of the Royal Statistical Society: Series A (Statistics in Society). 1997.

[11] Strong, G. C. *On the impact of selected modern deep-learning techniques to the performance and celerity of classification models in an experimental high-energy physics use case*. (2020, May 8).

[12] A; C. E. B. C. L. (n.d.). *Classification of Broadband Target Spectra in the mesopelagic using physics-informed machine learning*. The Journal of the Acoustical Society of America.

[13] Bowles, M. (2015, April 27). *Machine learning in Python: Essential techniques for predictive analysis*.

[14] *Give me some credit*. Kaggle. (n.d.). https://www.kaggle.com/c/GiveMeSomeCredit

[15] Hackeling, G. *Mastering machine learning with scikit-learn*. (2017, July 24)