BRAC UNIVERSITY

Inspiring Excellence

CSE422: Artificial Intelligence

Project Report

Project Title: Credit Score Prediction

Group No: 20, CSE422 Lab Section: 10, Spring 2024	
ID	**Name**
21201080	MD. ABRAR RAHMAN SHAFIN
24141246	ARABINDA PAUL TURJA

Table of Contents

Introduction

This project focuses on the development of a predictive model for bank credit scores. The aim is straightforward: to create a reliable tool that accurately forecasts credit scores for individuals applying for loans or other financial products.

The problem we're addressing is the need for efficient and precise credit assessment methods within banking institutions. Current systems often rely on outdated techniques and limited data, leading to inaccuracies and potential biases in credit score determinations.

Our motivation for undertaking this project is rooted in the desire to streamline the lending process for banks and improve financial inclusivity for customers. By creating a robust credit score prediction model, we aim to assist banks in making more informed decisions regarding loan approvals, thereby reducing the risk of default and optimizing their lending portfolios.

Dataset Description

Link:

Credit score classification (kaggle.com)

Number of features: 28

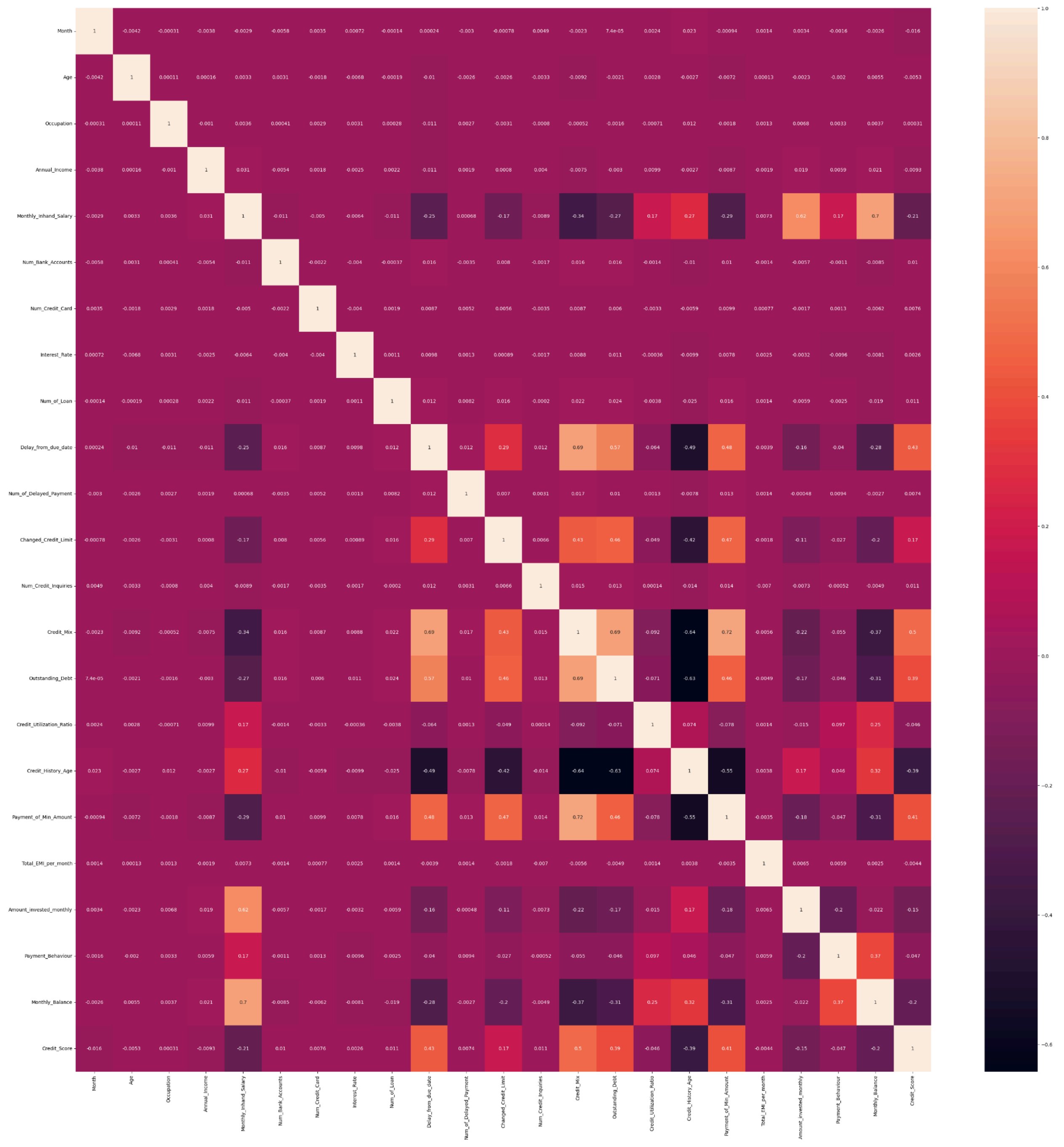
Classification or Regression: Classification, as the task involves classifying individuals into one of three categories: poor, standard, or good credit score. This is evident from the need to categorize individuals based on their creditworthiness into distinct groups, making it a classification problem.

Datapoints: 100003 rows

Types of Features:

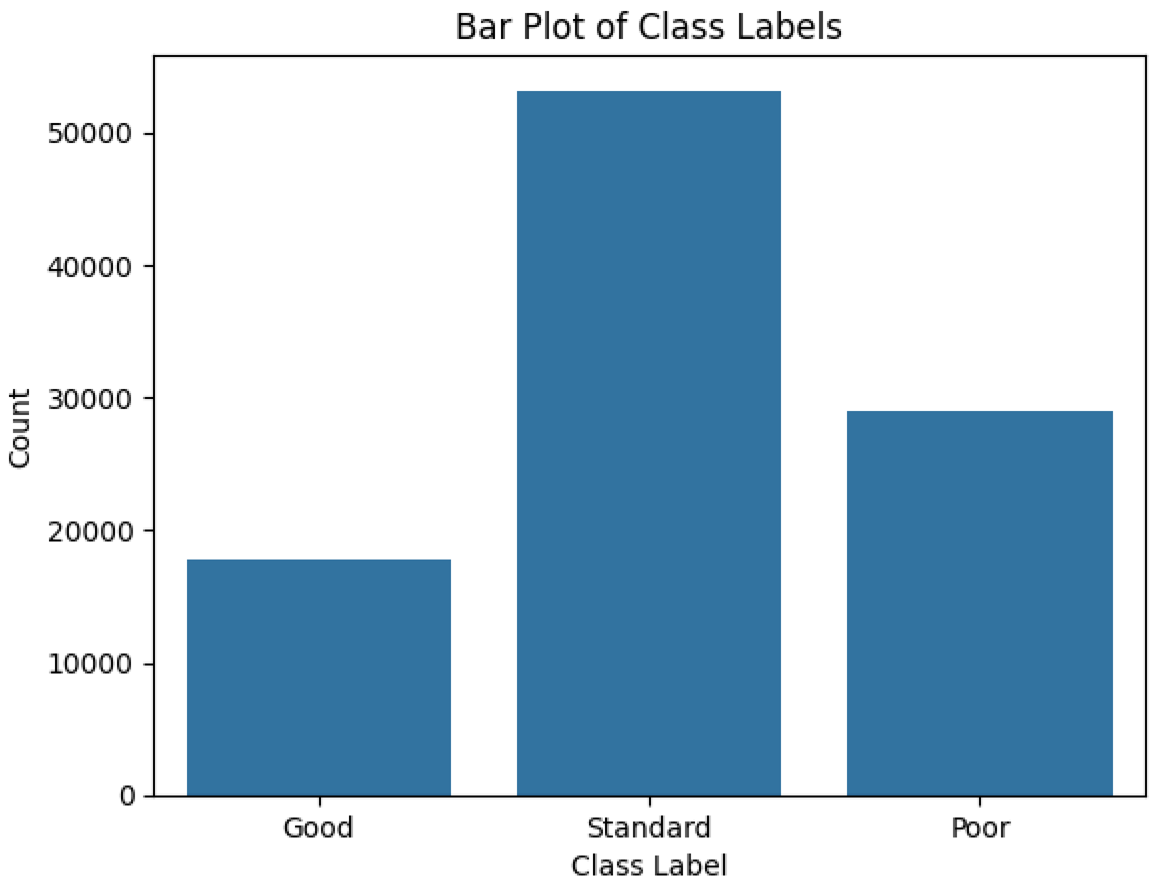
- Quantitative Features: These include numerical data such as age, annual income, monthly salary, number of bank accounts, number of credit cards, interest rate, number of loans, delay from due date, etc.
- Categorical Features: These include non-numerical data such as month, occupation, credit mix, payment of minimum amount, payment behavior, credit score, etc.

Correlation of all the features:



Output Feature Distribution: All unique classes do not have an equal number of instances. This indicates an imbalanced dataset where the “Standard” class is overrepresented compared to the “Poor” and “Good” class.

Class Distribution Bar Plot:

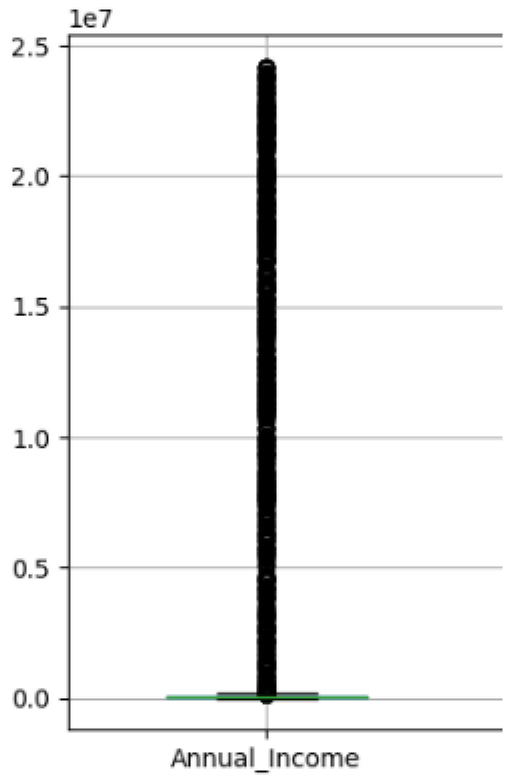


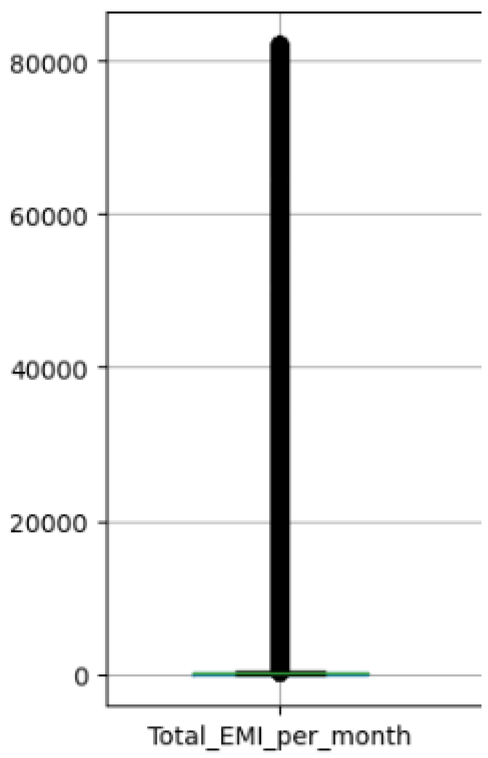
Data Preprocessing

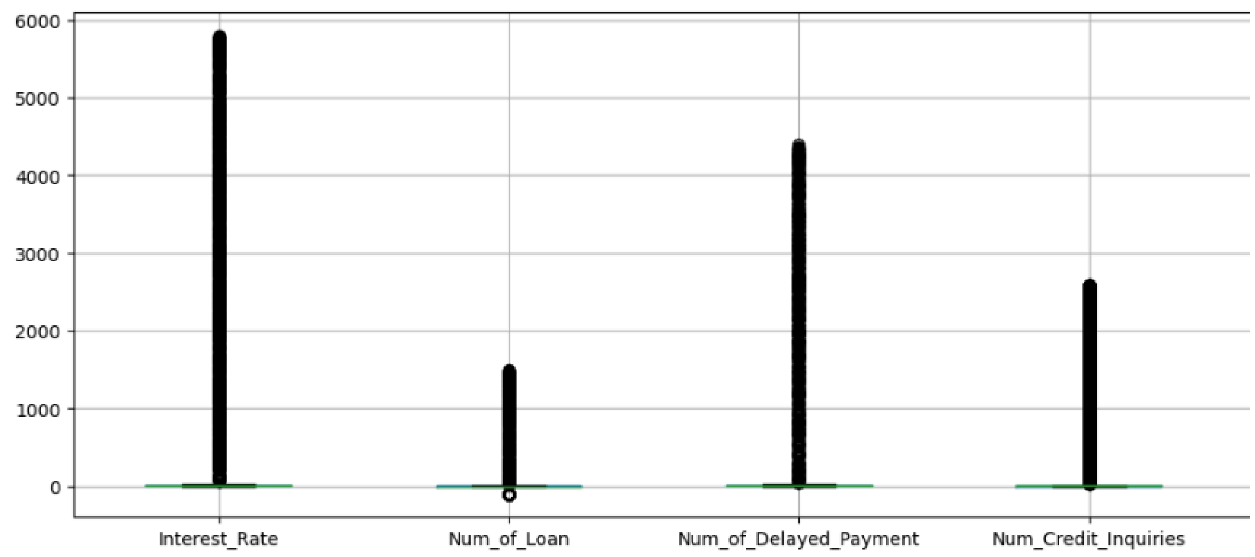
Faults:

- Null Values: Name, Monthly_Inhand_Salary, Type_of_Loan, Num_of_Delayed_Payment, Num_Credit_Inquiries, Credit_History_Age, Amount_invested_monthly, Monthly_Balance column have null values.

- Categorical Values: ID, Customer_ID, Month, Name, Occupation, Type_of_Loan, Credit_Mix, Credit_History_Age, Payment_of_Min_Amount, Payment_Behaviour, Credit_Score column have categorical values.
- Outside Domain Value: Some column carries irrelevant values, such as "Occupation", "Payment_Behaviour", and "Credit_Mix" contains these values respectively "_______", "!@9#%8", "_". Additionally, the "Delay_from_due_date" column has negative values.
- Outliers: "Annual_Income", "Interest_Rate", "Num_of_Loan", "Num_of_Delayed_Payment", "Num_Credit_Inquiries", "Total_EMI_per_month" have a massive amount of outliers.







Solutions:

- Imputing Null Values: Replacing null values with mode for columns (“Credit_Mix”, “Payment_Behavior”) with discrete values. Replacing null values with mean for columns (“Monthy_Inhand_Salary”, “Changed_Credit_Limit”, “Outstanding_Debt”, “Credit_History_Age”, “Amount_invested_monthy”, “Montly_Balance”) with continuous values. Removing these null rows will cause a huge data loss.
- Some categorical columns do not have any impact on the prediction, such as “ID”, “Customer_ID”, “Name”, “SSN”, and “Type_of_Loan”. So, we dropped these columns. For other categorical columns that may have an impact on the prediction need to be encoded first. So, we mapped specific code to those classes. As we have many irrelevant values in these columns, manually assigning values came in handy when dealing with those irrelevant values. One special column that needed to be handled differently is the “Credit_History_Age” column which contains the year and month together as a text. To handle this, we split year and month. Then converted it to months only. Now, we have only numeric columns, we can easily set irrelevant values to null and then remove those

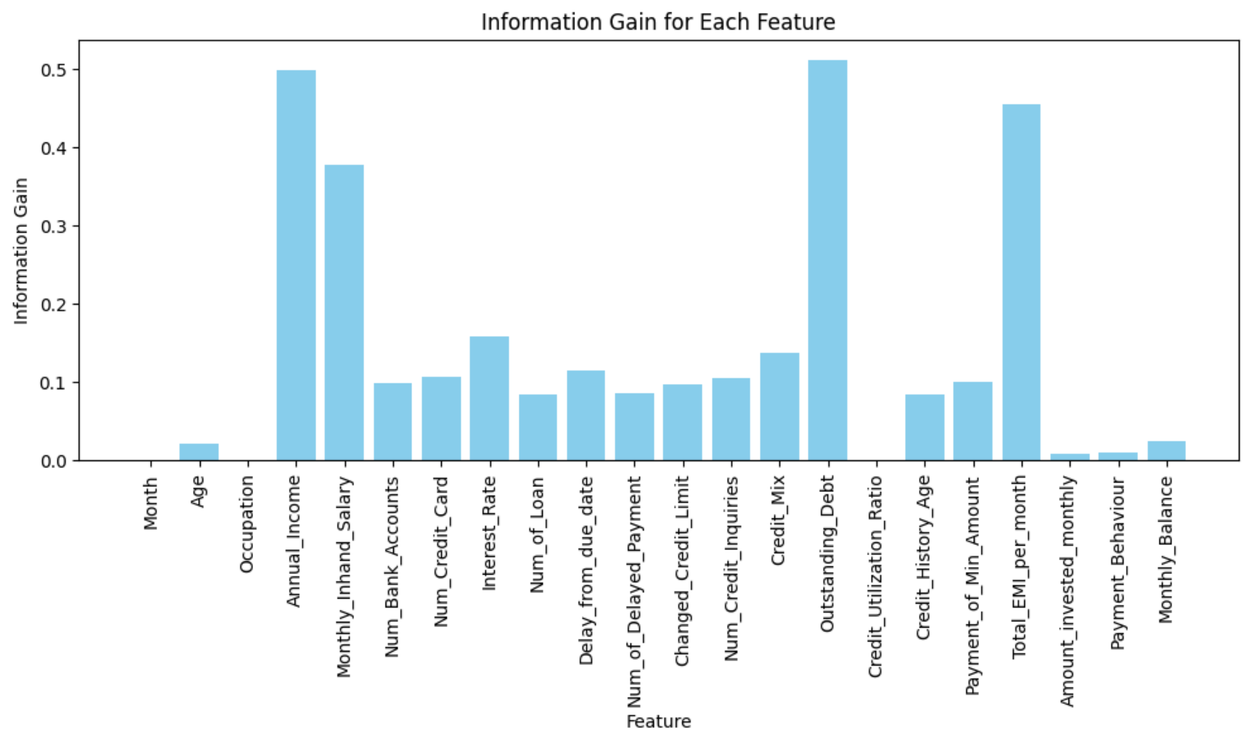
null rows. Special case for the “Delay_from_due_date” column as it has negative values which is invalid for this column. So, we set any negative value for this column to 0.

- For outliers, we set a valid range for each column. Any value outside this range will be replaced with null. Then, we just dropped the null rows.

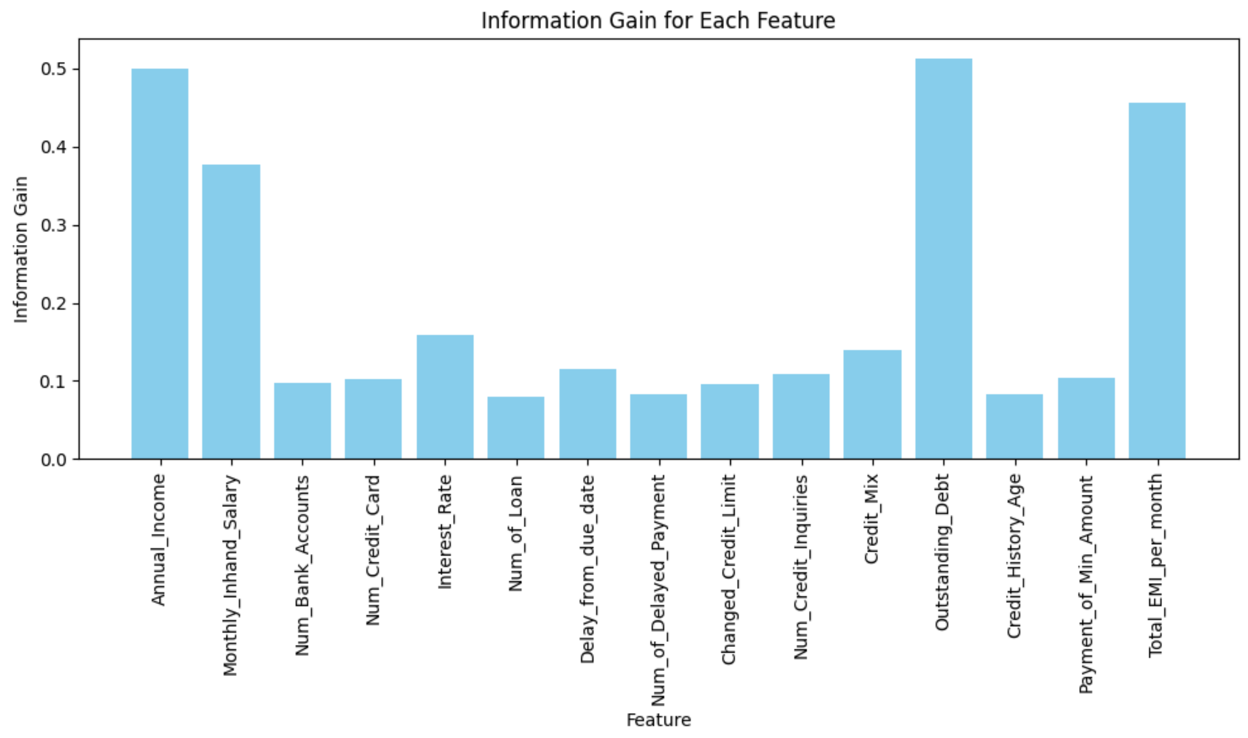
Feature Selection

To keep relevant and useful features for our model, we plotted a graph that shows how much information we can gain from each feature, meaning how pure a feature is. We dropped features that have information gain less than 0.05.

Before dropping:



After dropping:



After dropping, we are left with only features that are useful for our prediction.

Feature Scaling

As our data is prone to outliers, even after manually handling outliers, it is better to use scaling methods that are robust to outliers to increase prediction accuracy. In this case, we used a robust scaler to scale our training and testing data. The robust scaler divides the difference between the feature value and the median of the feature by its inter-quartile range which is the difference between the third quartile and the first quartile.

Dataset Splitting

As it is already mentioned that our label was not equally distributed, we stratified the label while splitting the dataset. Stratify ensures equal distribution of labels which helps the machine predict the result more accurately. We used 70% of the data for training and 30% of the data for testing.

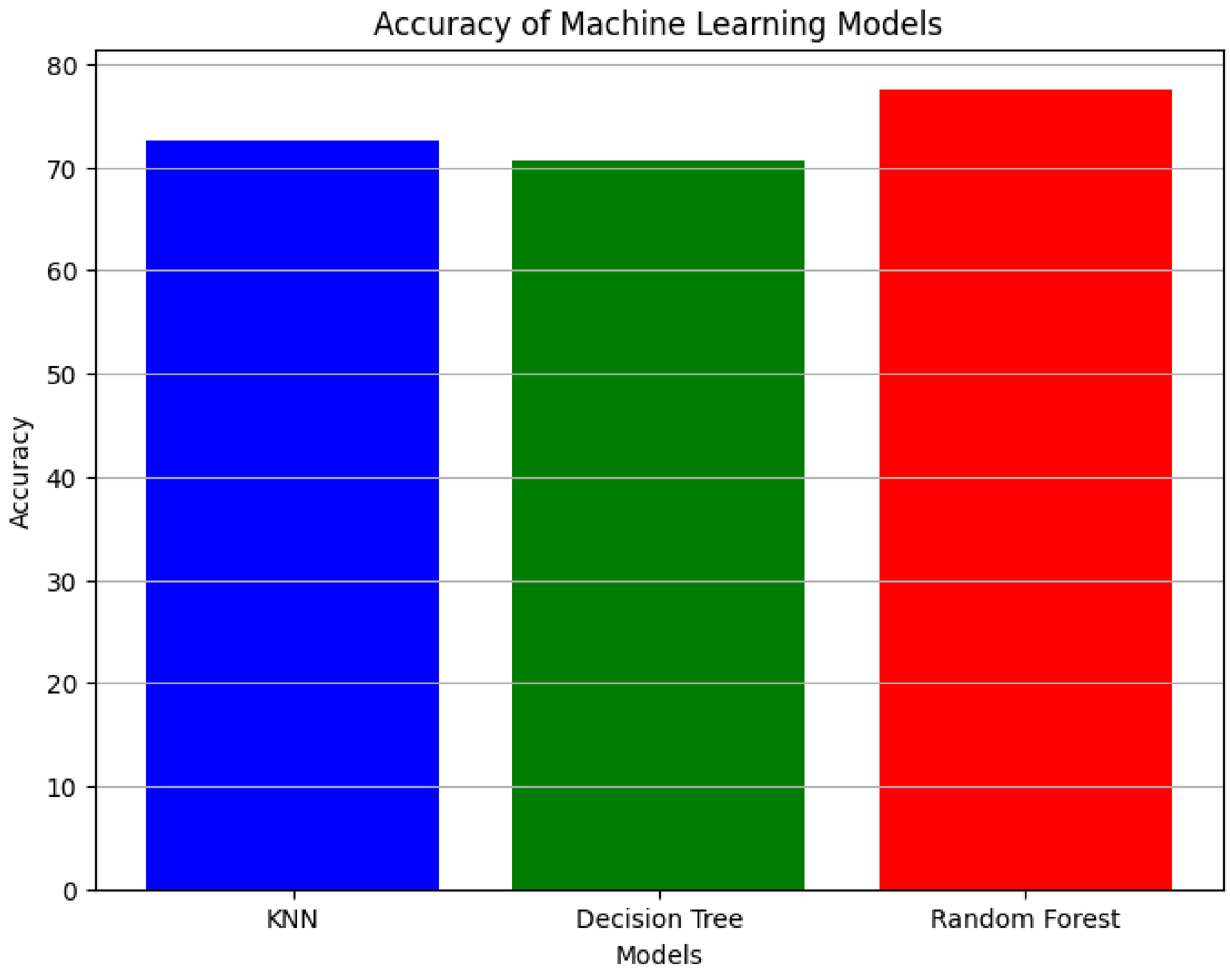
Model Training and Testing

At the beginning of our project, we mentioned that the problem we are dealing with is a classification problem. So, it is obvious classifiers will work better on our dataset. We chose the following three classifier models:

- KNN: KNN is a simple and intuitive algorithm used for classification and regression tasks, It works based on the idea that similar data points tend to belong to the same class or have similar target values. It's easy to understand and works well with small to moderately-sized datasets. It can handle non-linear relationships in the data.
- Decision Tree: Decision trees are versatile algorithms used for classification and regression tasks, They partition the feature space into regions and make decisions based on the values of features. Decision trees are highly interpretable and easy to visualize. They can handle both numerical and categorical data and can capture nonlinear relationships in the data.
- Random Forest: Random forest is an ensemble learning method based on decision trees. It builds multiple decision trees and combines their predictions to improve accuracy and robustness. Random forest is robust to overfitting, as it combines the prediction of multiple trees. It performs well with high-dimensional data and can handle missing values and outliers.

Comparison Analysis

Bar chart showcasing prediction accuracy of all models:



Accuracy, precision, recall, and F1 score for each model:

K-Nearest Neighbors:

Accuracy: 72.57%

Precision: 72.82%

Recall: 72.57%

F1 Score: 72.62%

Decision Tree Classifier:

Accuracy: 70.63%

Precision: 70.63%

Recall: 70.63%

F1 Score: 70.63%

Random Forest Classifier:

Accuracy: 77.55%

Precision:77.5%

Recall: 77.55%

F1 Score: 77.52%

Precision, recall comparison:

Precision measures the proportion of true positive predictions among all positive predictions made by the model. A higher precision indicates fewer false positives.

- Random Forest Classifier has the highest precision (77.5%), followed by K-Nearest Neighbors (72.82%), and Decision Tree Classifier (70.63%). This suggests that the Random Forest Classifier is better at minimizing false positive predictions compared to the other models.

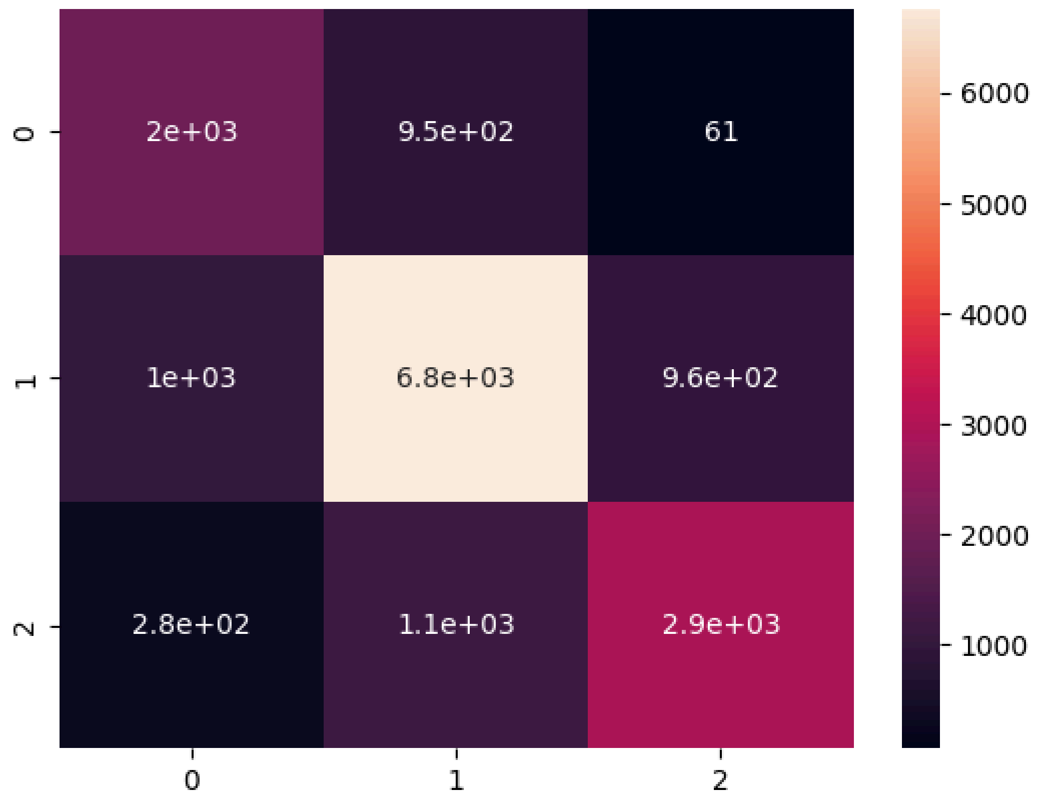
Recall measures the proportion of true positive predictions among all actual positive instances in the dataset. A higher recall indicates fewer false negatives.

- Random Forest Classifier has the highest recall (77.55%), followed by K-Nearest Neighbors (72.57%), and Decision Tree Classifier (70.63%). This suggests that the

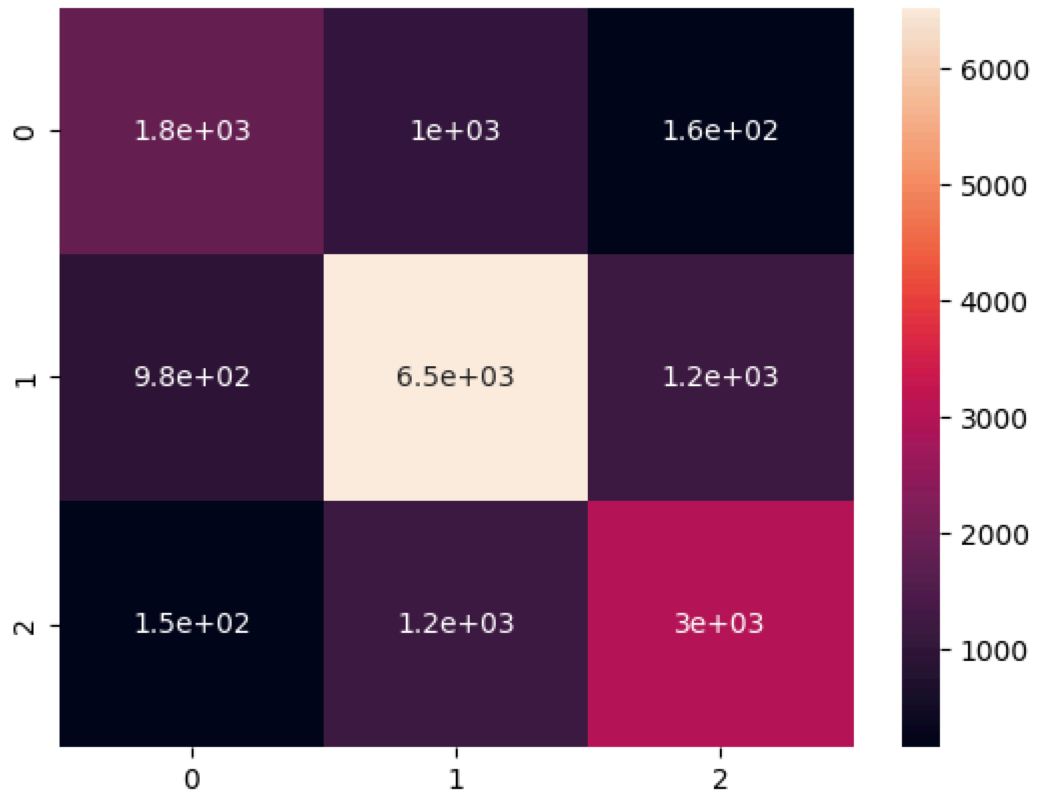
Random Forest Classifier is better at capturing positive instances in the dataset compared to the other models.

Confusion Matrix:

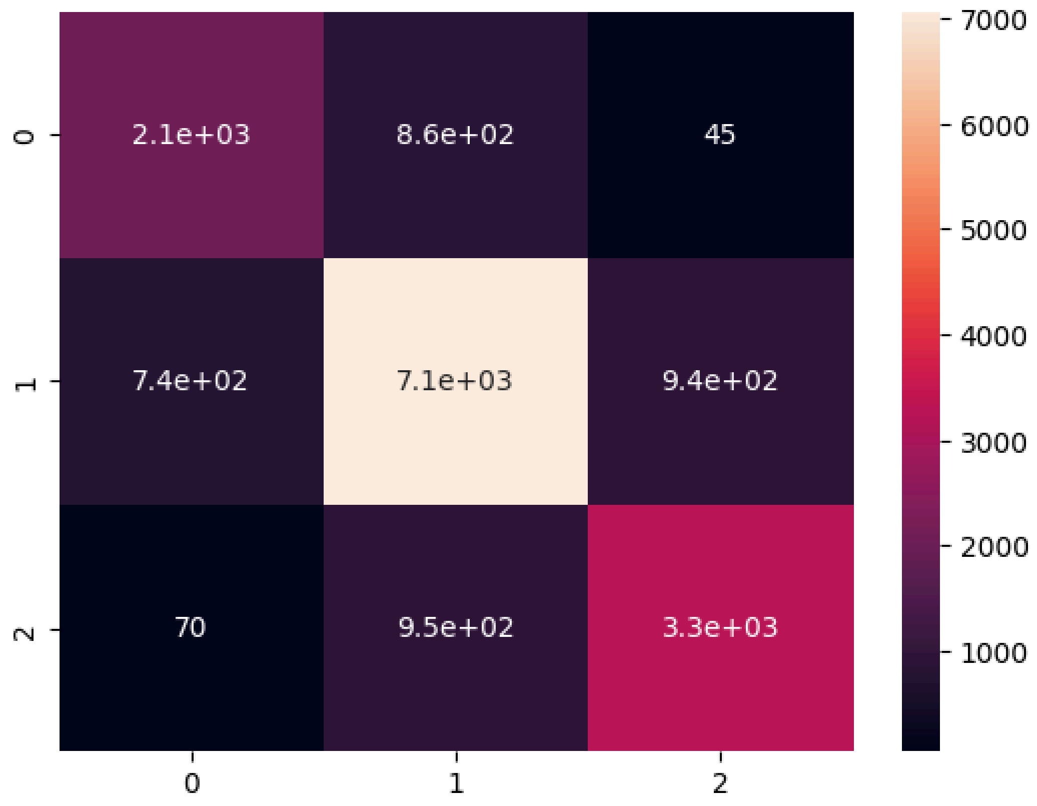
K-Nearest Neighbors:



Decision Tree Classifier:



Random Forest Classifier:



From the confusion matrix, we can infer that the Random Forest model has the lowest number of false negatives for 0 and 2.

Conclusion

Random Forest Classifier outperforms other models with the highest accuracy, precision, recall, and F1 score. K-Nearest Neighbors shows moderate performance and the decision tree classifier performs slightly lower than the others. Random Forest Classifier is recommended for its balanced precision-recall trade-off.