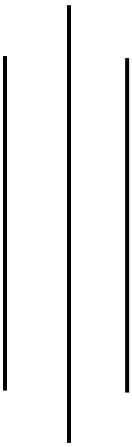

**TRIBHUVAN UNIVERSITY**

Faculty of Computer Science and Information Technology

## National College of Computer Studies

Paknajol, Kathmandu



# Credit Score Classification

Python Project Report

**Submitted by**

Smriti Shakya

B.Sc. CSIT 7th Sem

Section: B

Roll no. 19

**Submitted to**

Mr. Mausam Rajbanshi

(Python Lecturer)

September 15, 2024

# Abstract

Credit Score is a crucial factor in financial sector to determine a person's creditworthiness. Financial institutions use credit scores to assess the risk of lending money to individuals. The score provides a predictive indicator of the likelihood that a person will repay their debts.

The goal of this classification task is to predict the credit score category of individuals using various financial and behavioral features. The credit scores were classified manually which made it a tedious task. But this model segregates the people into credit score brackets which will reduce the manual efforts. I have applied three Machine Learning Models: Random Forest, Logistic Regression and Decision Tree Classification to identify patterns in the data and make accurate predictions. These models offer a scientific approach to credit score classification, improving the ability to classify credit scores based on demographic, financial and credit behavior variables and helping financial institutions make more informed decisions.

# Acknowledgement

I would like to express my sincere gratitude to Mr. Mausam Rajbanshi for his guidance throughout this project. Without his teachings and support, this project may not have come to fruition.

Additionally, I would also like to show my deepest appreciation to the NCCS family for providing the conducive environment and all the necessary support needed for successful project completion.

I acknowledge the contribution of articles and notes on various platforms for providing valuable knowledge on the related topics. This project would not have been possible without the collective efforts of all those mentioned above, and for that, I am sincerely grateful.

# Table of Contents

# List of Figures

# Chapter 1: Introduction

## 1.1 Introduction

Credit score classification is the process of categorizing individuals based on their creditworthiness into different credit score bands. Financial institutions use credit scores to assess the risk of lending money to individuals, as the score provides a predictive indicator of the likelihood that a person will repay their debts.

There are three credit scores that banks and credit card companies use to label their customers:

- Good
- Standard
- Poor

A person with a good credit score will get loans from any bank and financial institution. [1]

## 1.2 Problem Statement

The aim of this project is to develop a predictive model that can classify individuals' credit scores into categories based on features like income, credit history, debt, and financial behavior. The task is a supervised classification problem where the target variable is the Credit Score, and the features used are demographic, financial, and credit-related information.

By leveraging such features, the model helps financial institutions make informed decisions about loan approvals, credit limits, and interest rates. Key challenges include dealing with imbalanced data, selecting relevant features, and balancing model accuracy, precision, and recall, while also ensuring interpretability and regulatory compliance.

## 1.3 Objectives

- To preprocess the data to implement machine learning algorithms.
- To build and evaluate machine learning models to classify credit scores.
- To compare the performance of different classification algorithms.

# Chapter 2: Dataset

## 2.1 Dataset Overview

The dataset used for credit score classification contains a mix of demographic, financial, and behavioral information. There are 28 columns in Train dataset and 27 columns in Test dataset. Below are the key columns in the dataset:

- **Demographic Variables**: Age, Occupation
- **Financial Variables**: Annual Income, Monthly Inhand Salary, Total EMI per Month, Outstanding Debt, Amount Invested Monthly
- **Credit Behavior Variables**: Number of Bank Accounts, Number of Credit Cards, Number of Loans, Interest Rate, Number of Delayed Payments, Credit History Age, Credit Utilization Ratio, Payment Behavior

## 2.2 Key Insights

- **Credit Score**: The target variable representing the credit score category (Good, Standard, Poor).
- **Data Format**: The data type and format of some columns are unmatched. They need to be formatted properly.
- **Missing Values**: Some columns have missing values, which will need to be handled appropriately.
- **Outliers**: Certain financial features may have extreme values, which can skew the model's performance.

# Chapter 3: Feature Selection

## 3.1 Features

Based on the dataset, the following key features have been selected for building the model [2]:

1. **Annual Income**: Higher income usually indicates the ability to repay debts.
2. **Credit Utilization Ratio**: A higher ratio suggests the individual is using more of their available credit, which could indicate higher risk.
3. **Number of Delayed Payments**: Frequent delays in payments are highly indicative of a lower credit score.
4. **Number of Loans**: A high number of loans might increase financial strain.
5. **Interest Rate**: Higher interest rates might reflect riskier loans.
6. **Total EMI per Month**: High monthly debt burden can indicate financial stress.
7. **Number of Credit Inquiries**: Multiple recent inquiries may suggest a higher credit risk.
8. **Credit History Age**: Longer credit history typically correlates with a higher credit score.

Other important features like Occupation, Age, and Payment Behavior are included based on domain knowledge.

## 3.2 New Features

Based on the given information, new features are created from the relationship between existing ones.

## Adding New Features

```
# Total number of accounts (Bank Accounts + Credit Cards)
train['Total_Num_Accounts'] = train['Num_Bank_Accounts'] + train['Num_Credit_Card']

# Total outstanding debt per account
train['Debt_Per_Account'] = train['Outstanding_Debt'] / train['Total_Num_Accounts']

# Ratio of outstanding debt to annual income
train['Debt_to_Income_Ratio'] = train['Outstanding_Debt'] / train['Annual_Income']

# Total number of delayed payments per account
train['Delayed_Payments_Per_Account'] = train['Num_of_Delayed_Payment'] / train['Total_Num_Accounts']

# Total monthly expenses (EMI + Monthly Investments)
train['Total_Monthly_Expenses'] = train['Total_EMI_per_month'] + train['Amount_invested_monthly']
```

✓ 0.0s

# Chapter 4: Data Preprocessing

## 4.1 Importing Necessary Modules and Loading Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

✓ 8.0s

```
#load data in dataframe
data_train =pd.read_csv("./csv_files/credit score classification/train.csv")
data_test =pd.read_csv("./csv_files/credit score classification/test.csv")
```

✓ 8.0s

```
train.info()
```

✓ 0.6s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   ID                        100000 non-null  object
 1   Customer_ID               100000 non-null  object
 2   Month                     100000 non-null  object
 3   Name                      90015 non-null   object
 4   Age                       100000 non-null  object
 5   SSN                       100000 non-null  object
 6   Occupation                100000 non-null  object
 7   Annual_Income             100000 non-null  object
 8   Monthly_Inhand_Salary     84998 non-null   float64
 9   Num_Bank_Accounts         100000 non-null  int64
 10  Num_Credit_Card           100000 non-null  int64
 11  Interest_Rate             100000 non-null  int64
 12  Num_of_Loan               100000 non-null  object
 13  Type_of_Loan              88592 non-null   object
 14  Delay_from_due_date       100000 non-null  int64
 15  Num_of_Delayed_Payment    92998 non-null   object
 16  Changed_Credit_Limit      100000 non-null  object
 17  Num_Credit_Inquiries      98035 non-null   float64
 18  Credit_Mix                100000 non-null  object
 19  Outstanding_Debt          100000 non-null  object
...
 26  Monthly_Balance           98800 non-null   object
 27  Credit_Score              100000 non-null  object
dtypes: float64(4), int64(4), object(20)
memory usage: 21.4+ MB
```

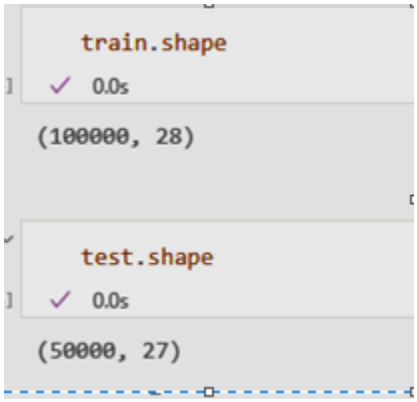
*Output is truncated. View as a scrollable element or open in a text editor. Adju*

**Result**:

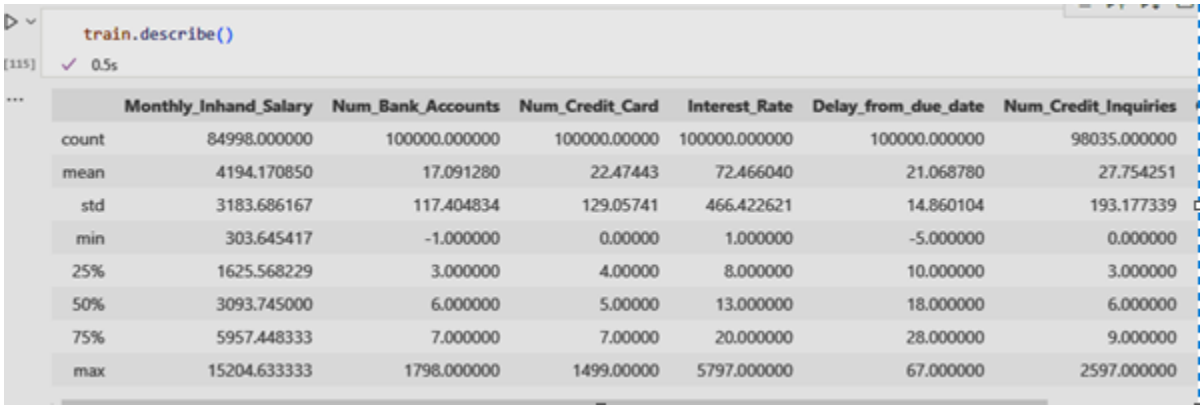
The df.info() provides a summary of the dataset. There are 100000 number of entries, 27 column names, non-null values,4 float64 values, 4 int64 values and 20 object values for input variables and 1 object value for target variable i.e. Credit_Score.

## 4.2 Exploratory Data Analysis

### i) Determining Rows and Column




### ii) Data Describe




| | Monthly_Inhand_Salary | Num_Bank_Accounts | Num_Credit_Card | Interest_Rate | Delay_from_due_date | Num_Credit_Inquiries |
|---|---|---|---|---|---|---|
| count | 84998.000000 | 100000.000000 | 100000.00000 | 100000.000000 | 100000.000000 | 98035.000000 |
| mean | 4194.170850 | 17.091280 | 22.47443 | 72.466040 | 21.068780 | 27.754251 |
| std | 3183.686167 | 117.404834 | 129.05741 | 466.422621 | 14.860104 | 193.177339 |
| min | 303.645417 | -1.000000 | 0.00000 | 1.000000 | -5.000000 | 0.000000 |
| 25% | 1625.568229 | 3.000000 | 4.00000 | 8.000000 | 10.000000 | 3.000000 |
| 50% | 3093.745000 | 6.000000 | 5.00000 | 13.000000 | 18.000000 | 6.000000 |
| 75% | 5957.448333 | 7.000000 | 7.00000 | 20.000000 | 28.000000 | 9.000000 |
| max | 15204.633333 | 1798.000000 | 1499.00000 | 5797.000000 | 67.000000 | 2597.000000 |

**Result:** The df.describe() gives a statistical summary of numerical columns, including metrics like mean, standard deviation, and quartiles.

### iii) Checking for null values

```
train.isnull().sum()
[116] ✓ 0.0s
```

| | |
|---|---|
| ID | 0 |
| Customer_ID | 0 |
| Month | 0 |
| Name | 9985 |
| Age | 0 |
| SSN | 0 |
| Occupation | 0 |
| Annual_Income | 0 |
| Monthly_Inhand_Salary | 15002 |
| Num_Bank_Accounts | 0 |
| Num_Credit_Card | 0 |
| Interest_Rate | 0 |
| Num_of_Loan | 0 |
| Type_of_Loan | 11408 |
| Delay_from_due_date | 0 |
| Num_of_Delayed_Payment | 7002 |
| Changed_Credit_Limit | 0 |
| Num_Credit_Inquiries | 1965 |
| Credit_Mix | 0 |
| Outstanding_Debt | 0 |
| Credit_Utilization_Ratio | 0 |
| Credit_History_Age | 9030 |
| Payment_of_Min_Amount | 0 |
| Total_EMI_per_month | 0 |
| Amount_invested_monthly | 4479 |
| Payment_Behaviour | 0 |
| Monthly_Balance | 1200 |
| Credit_Score | 0 |

dtype: int64

```
train.isnull().sum()
[27] ✓ 0.0s
```

| | |
|---|---|
| Age | 0 |
| Occupation | 0 |
| Annual_Income | 0 |
| Num_Bank_Accounts | 0 |
| Num_Credit_Card | 0 |
| Interest_Rate | 0 |
| Num_of_Loan | 0 |
| Delay_from_due_date | 0 |
| Num_of_Delayed_Payment | 0 |
| Changed_Credit_Limit | 0 |
| Num_Credit_Inquiries | 0 |
| Credit_Mix | 0 |
| Outstanding_Debt | 0 |
| Credit_Utilization_Ratio | 0 |
| Payment_of_Min_Amount | 0 |
| Total_EMI_per_month | 0 |
| Amount_invested_monthly | 0 |
| Payment_Behaviour | 0 |
| Monthly_Balance | 0 |
| Credit_Score | 0 |
| Credit_History_Age_Months | 0 |

dtype: int64

**Result:** The method df.isnull().sum() tells us whether there are any missing values present or not. It looks like there are lots of missing values to handle. The output on the right side is after all the missing values are handled.

**iv) Checking for duplicates**


**v) Data Formatting**


**Result**:

After the data formatting, there are 100000 number of entries, 29 column names, non-null values,9 float64 values, 5 int64 values and 15 object values for input variables.

## 4.3 Data Scaling

**i) Removing Top Outliers**

```
# removing top outliers
selected_columns_train = train[['Num_Bank_Accounts', 'Interest_Rate', 'Annual_Income', 'Num_of_Delayed_Payment',
                                'Num_Credit_Inquiries', 'Total_EMI_per_month', 'Num_of_Loan', 'Num_Credit_Card']]

percentile_threshold = 0.98
percentiles = selected_columns_train.quantile(percentile_threshold)

for column in selected_columns_train.columns:
    train = train[train[column] <= percentiles[column]]
```
[18] ✓ 2.0s

### ii) Scaling the data

```
# Scaling
train = train[train['Age'] < 60]
train = train[train['Num_Credit_Card'] <= 10]
train = train[train['Interest_Rate'] <= 50]
train = train[train['Num_of_Loan'] <= 12]
train = train[train['Num_Bank_Accounts'] <= 10]
train = train[train['Delay_from_due_date'] <= 60]
train = train[train['Changed_Credit_Limit'] <= 30]
train = train[train['Num_Credit_Inquiries'] <= 12]
train = train[train['Total_EMI_per_month'] <= 200]
train = train[train['Outstanding_Debt'] <= 1500]
```
[29] ✓ 0.5s

This Scaling is done to keep all the data within range. The outliers are also removed so the data distribution doesn't get skewed.

## 4.3 Filtering Data Values

### i) Dropping Columns

```
columns_to_drop = ['ID', 'Customer_ID', 'Month', 'Name', 'SSN',
                   'Credit_History_Age','Monthly_Inhand_Salary', 'Type_of_Loan']

train.drop(columns=columns_to_drop, inplace=True)
```
[24] ✓ 0.0s

```
total_missing_values = train.isnull().sum().sum()

if total_missing_values == 0:
    print("There are no missing values")
else:
    print("Total missing values:", total_missing_values)
```
[26] ✓ 0.0s

```
There are no missing values
```

### ii) Filtering Anolamous Data


There were anomalous data in columns Payment Behavior, Occupation and Credit Mix which were filtered out.

## 4.4 Encoding Data

### i) Label Encoding

### ii) Ordinal Encoding

Ordinal Encoding

```
categories = ['Bad', 'Standard', 'Good']

encoder = OrdinalEncoder(categories=[categories])

train['Credit_Mix_Encoded'] = encoder.fit_transform(train[['Credit_Mix']])
```

[33] ✓ 0.0s

```
categories_payment_behaviour = [
    'Low_spent_Small_value_payments',
    'Low_spent_Medium_value_payments',
    'Low_spent_Large_value_payments',
    'High_spent_Small_value_payments',
    'High_spent_Medium_value_payments',
    'High_spent_Large_value_payments'
]

encoder_payment_behaviour = OrdinalEncoder(categories=[categories_payment_behaviour])

train['Payment_Behaviour_Encoded'] = encoder_payment_behaviour.fit_transform(train[['Payment_Behaviour']])
```

[34] ✓ 0.0s

## 4.5 Data Visualization

### i) Box Plot

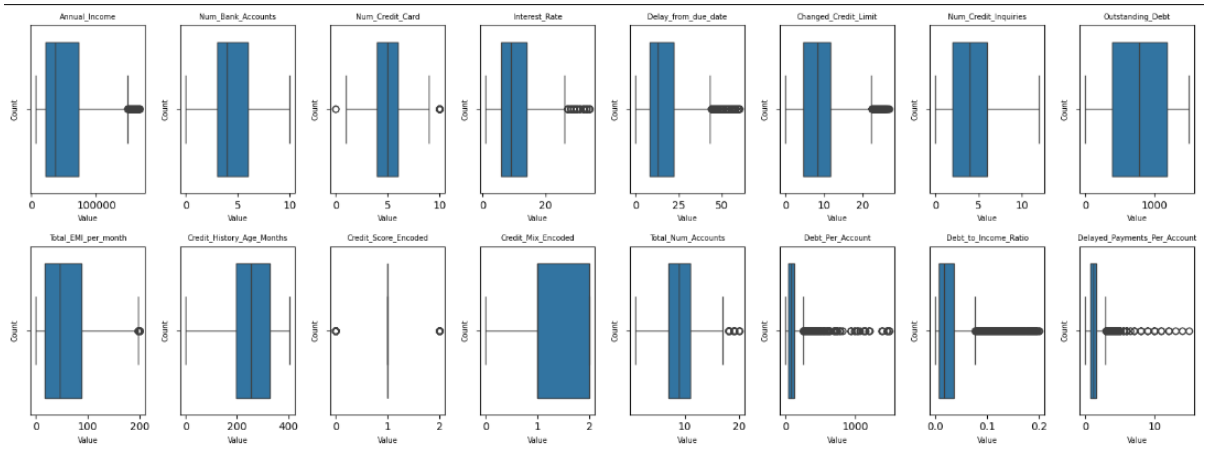



*Figure 1: Box Plot*

## ii) Heatmap

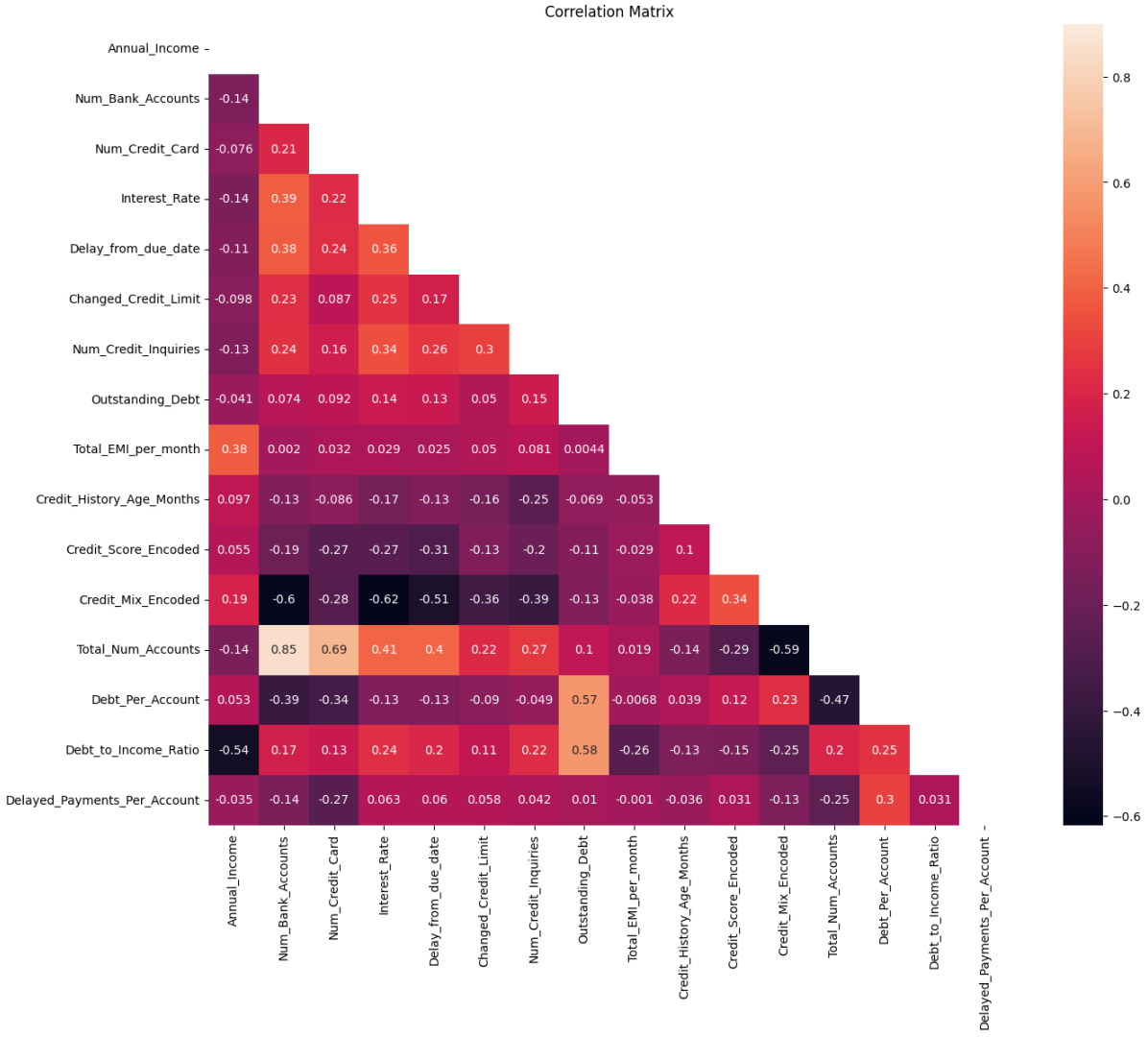



*Figure 2: Heatmap*

# Chapter 5: Model Building and Evaluation

## 5.1 Model

Train Test Split was done on training data to build a model to predict the credit score.

Model

```
y = train['Credit_Score_Encoded']

X = train[['Annual_Income', 'Num_Bank_Accounts', 'Num_Credit_Card',
       'Interest_Rate', 'Num_of_Loan', 'Delay_from_due_date',
       'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Outstanding_Debt', 'Total_EMI_per_month',
       'Credit_History_Age_Months', 'Credit_Mix_Encoded', 'Total_Num_Accounts',
       'Debt_Per_Account', 'Debt_to_Income_Ratio', 'Delayed_Payments_Per_Account']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=77)
```

✓ 0.0s

## 5.2 Logistic Regression

Logistic Regression was trained using the cleaned and scaled dataset. The model struggled with non-linear relationships between the features and the target, leading to moderate accuracy.

**Accuracy on training data**: 63%

## 5.3 Random Forest Classifier

Random Forest is an ensemble method that builds multiple decision trees and averages their predictions to improve accuracy and reduce overfitting. Since the dataset is large, it showed high accuracy and it is less prone to overfitting.

**Accuracy on training data**: 80%

## 5.3 Decision Tree Classifier

A Decision Tree Classifier was used with depth constraints to avoid overfitting. The model handled non-linear relationships well and performed better with the categorical features, but was prone to overfitting on deeper trees.

**Accuracy on training data**: 75%

## 5.4 Model Evaluation

Each model was evaluated on the test set using common classification metric accuracy. Here, Random Forest has the highest accuracy so, this model is the best.

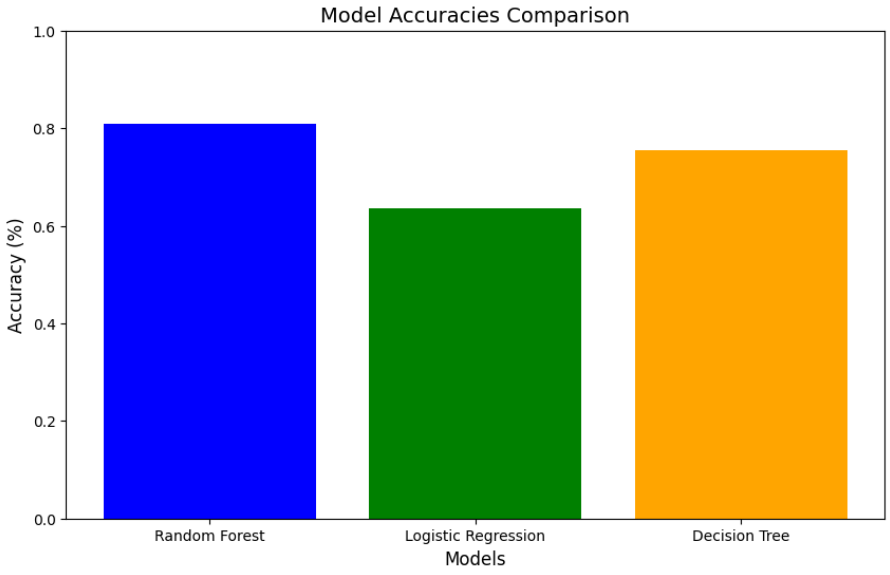



*Figure 3: Model Evaluation*

# Chapter 6: Conclusion

Based on the evaluation metrics, the Random Forest model proved to be the most effective algorithm for credit score classification, offering the best accuracy. Logistic Regression, while interpretable, was less effective at identifying risky borrowers, particularly in an imbalanced dataset. The Decision Tree provided some interpretability benefits but lacked the overall performance and generalizability of Random Forest.

Hence, Random Forest was used to predict the Test Data. The Test Data was cleaned in accordance to Train Data.

Performing Predictions on Test Data

```
# Make predictions on the test data using Random Forest model as it produced highest accuracy
y_pred_test = rf_classifier.predict(test)
```

[101] ✓ 12.1s

```
#Add predictions to the test dataframe
test['Predicted_Credit_Scores'] = y_pred_test

# Save only the predictions if 'ID' does not exist
test[['Predicted_Credit_Scores']].to_csv('Test_Predictions_Crdit_Score.csv', index=False)

# Display the first few rows of predictions
test[['Predicted_Credit_Scores']].head(8)
```

[103] ✓ 0.1s

| | Predicted_Credit_Scores |
|---|---|
| 1 | 2.0 |
| 3 | 2.0 |
| 5 | 2.0 |
| 12 | 1.0 |
| 14 | 1.0 |
| 17 | 1.0 |
| 19 | 1.0 |
| 20 | 2.0 |

# References

[1] A. Kharwal, "Credit Score Classification with Machine Learning," thecleverprogrammer, 5 December 2022. [Online]. Available: https://thecleverprogrammer.com/2022/12/05/credit-score-classification-with-machine-learning/. [Accessed August 2024].

[2] "How Credit Score is Calculated," shriramfinance, 21 February 2024. [Online]. Available: https://www.shriramfinance.in/article-how-credit-score-is-calculated. [Accessed September 2024].