A Data Mining Approach to Credit Risk Prediction

Likitha Santhapalli

1.INTRODUCTION

Credit scores are a key part of today's financial world. Banks, lenders, and other financial institutions depend on them to asses how likely someone is to repay a loan or manage credit responsibly. A strong credit score is likely to give better chances of loan approval, better or lower interest rates and wide access to financial services. On the other side, a low score can hold people back from those same opportunities. In the past, evaluating someone's creditworthiness was mostly a manual process. But with the rise of data and technology, automated credit scoring models have become the norm, offering faster, more consistent, and data-backed evaluations.

Importance of Credit Scoring:

Helps lenders decide faster: Credit scores give banks and lenders a quick way to determine if someone is worth the risk of giving a loan. It saves time and avoids guesswork.

Shows if someone's risky or not: If your score is high, it usually means you've been paying loans or credit card amounts on time. A low score kind of sets off alarms that you might not pay back.

Better deals if your score's good: People with good credit usually get better interest rates and more options. For example, you might qualify for a bigger loan or not have to pay as much interest.

Affects more than just loans: It's not just about borrowing money. Credit scores can affect renting a place, getting a credit card, and in some cases, even job applications.

Goal

This project focuses on classifying credit scores into three categories: ***Poor, Standard, and Good.*** We use a dataset containing various financial features of individuals, such as income, number of loans, payment behavior, credit card usage, and more. Our aim is to build a predictive model using data mining and classification techniques that can accurately determine a customer's credit score. The process includes data cleaning, feature engineering, model building, and evaluation using metrics like accuracy, precision, and recall.

2. OVERVIEW OF DATASET

Source: I have taken credit score dataset from Kaggle.
https://www.kaggle.com/code/nagavenkatesh/credit-score-classification-logistic-regression/input?select=train.csv

I have analyzed the dataset and the below are the findings of the respective dataset. The dataset has 100,000 records and a total of 28 columns. It contains both numerical and categorical columns, and the target variable is **Credit Score.**
The dataset includes the following features:

Sl.no	Feature	Description
1.	ID	A unique identifier for each record in the dataset
2.	Customer_ID	A unique ID assigned to each customer.
3.	Month	The month when the data was recorded.
4.	Name	The full name of the customer
5.	Age	The age of the customer.
6.	SSN	The Social Security Number for the customer.
7.	Occupation	The customer's job or profession.
8.	Annual_Income	The yearly income of the customer
9.	Monthly_Inhand_Salary	The amount the customer receives in hand monthly after deductions.
10.	Num_Bank_Accounts	Number of bank accounts the customer has.
11.	Num_Credit_Card	Number of credit cards the customer owns.
12.	Interest_Rate	Interest rate applicable on the customer's loans or credit.
13.	Num_of_Loan	Total number of loans the customer has
14.	Type_of_Loan	Types of loans the customer has taken
15.	Delay_from_due_date	Days the customer delayed their payment from the due date.
16.	Num_of_Delayed_Payment	Number of times the customer has delayed payments
17.	Num_Credit_Inquiries	Number of credit checks/inquiries made for the customer.
18.	Credit_Mix	The mix of credit types (like credit cards, loans, etc.) the customer.
19.	Outstanding_Debt	The total unpaid debt amount the customer currently owes.
20.	Credit_Utilization_Ratio	Percentage of available credit currently being used.
21.	Credit_History_Age	Duration (age) of the customer's credit history.
22.	Payment_of_Min_Amount	Whether the customer pays at least the minimum amount due.

23.	Monthly_Balance	Remaining balance after expenses, EMIs, and investments each month.
24.	Total_EMI_per_month	Total amount of all EMIs (Equated Monthly Installments) paid each month.
25.	Changed_Credit_Limit	Change in the customer's credit limit over time.
26.	Amount_invested_monthly	Monthly amount the customer invests (e.g., savings, mutual funds).
27.	Payment_Behaviour	Describes the customer's payment habits (e.g., prompt, delayed).
28.	Credit_Score	The credit rating/category assigned to the customer (e.g., Poor, Standard, Good).

The below is the additional list of details for the list of columns names and data types. I checked the dataset for missing values, and the summary below shows how many are missing in each column and I have not found any duplicate values.

```
<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 
 20  Credit_Utilization_Ratio  100000 non-null  float64
 21  Credit_History_Age        90970 non-null   object 
 22  Payment_of_Min_Amount     100000 non-null  object 
 23  Total_EMI_per_month       100000 non-null  float64
 24  Amount_invested_monthly   95521 non-null   object 
 25  Payment_Behaviour         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
```

Missing values (>0):

	Missing	Pct
Name	9985	9.98
Monthly_Inhand_Salary	15002	15.00
Type_of_Loan	11408	11.41
Num_of_Delayed_Payment	7002	7.00
Num_Credit_Inquiries	1965	1.96
Credit_History_Age	9030	9.03
Amount_invested_monthly	4479	4.48
Monthly_Balance	1200	1.20

```
duplicates = credit_train_data.duplicated().sum()
if duplicates > 0:
    print(f"Found {duplicates} duplicate rows!")
else:
    print("No duplicates found")
```

```
No duplicates found
```

I have also observed above that some of the numerical columns are represented as objects. Additionally, I have also examined the train.csv file and identified some data quality issues, including unexpected characters appended to values in columns like 'Age' and abnormally high negative values, particularly in the 'Monthly Income' column.

3.DATA PREPROCESSING

Before building the model, I have prepared or clean up the dataset to make sure everything was in good shape. This includes dealing with missing values, getting rid of any duplicate records and fixing other issues like weird characters in some columns and extreme and abnormal values as mentioned before. These steps helped to make the data cleaner and more reliable for training the model.

I have performed the following preprocessing steps: -

- ***Dropped Irrelevant Columns:***
  I removed columns like ID, Customer_ID, SSN, Name, and Month since they did not add any useful information for prediction. Most of them were just identifiers or has redundant data. –
- ***Cleaned Monthly_Balance:***
  This column has special characters like "(-,_)" and placeholder values (e.g., -333...). To make it clean I have stripped the special characters and the abnormal values are replaced. The column was then converted to numeric and missing values were filled using the median.
- ***Converted Credit_History_Age:***
  The data which is in "22 Years and 1 Months" format was converted to total months which helps us to treat the feature as a continuous numerical variable. –
- ***Age:***
  The Age column contained invalid entries, including negative values and ages above 100. Since loans and credit are only granted to adults, I filtered the data to include only ages between 18 and 100. Any values outside this range were removed and replaced with the median age. Additionally, missing values in the Age column were also filled using the median.
- ***Processed Numeric Columns Stored as Text:***
  A lot of numeric columns were actually stored as text and included unwanted characters. I cleaned them using regular expressions, converted them to proper numeric types, and replaced any missing values with the median. I also handled invalid ages by removing anything below 18 or over 100. –

- ***Cleaned Categorical Columns:***
  Some categorical fields like "Payment_behaviour", "Occupation"," Credit mix" etc. had inconsistent or strange values like blank strings, underscores ("__"), or random symbols (! \@9#%8). These are replaced with "Unknown" because their count is too high and I thought removing or deleting the rows is not a good approach.
- ***Handling outliers:***
  I handled outliers using adaptive IQR and domain-specific caps (e.g., 100% for utilization), and grouped rare categories into "Other" to reduce noise and overfitting. After all preprocessing, the data was clean, consistent, and ready for reliable model training.

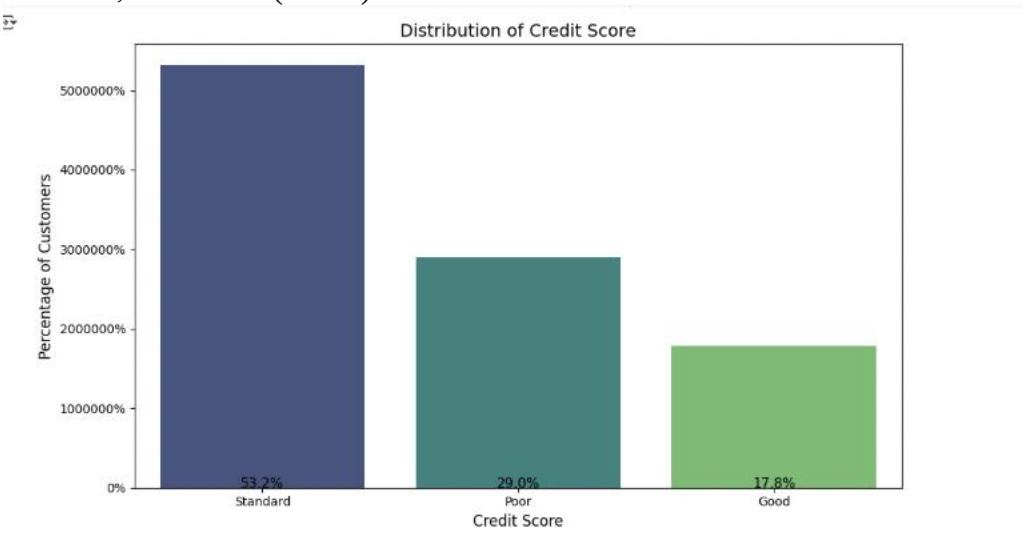
Finally, I have made sure after all this steps the field are converted to respective data types. After all transformations and imputations, I have checked the dataset for any missing values and they were none.

4.EXPLORATORY DATA ANALYSIS (AFTER CLEANING):

After completing the data cleaning process, I have done exploratory data analysis to understand the structure, patterns, and distributions within the dataset. This analysis will help us to identify relationships between features and the target variable (Credit_Score) and gives more information to further process.

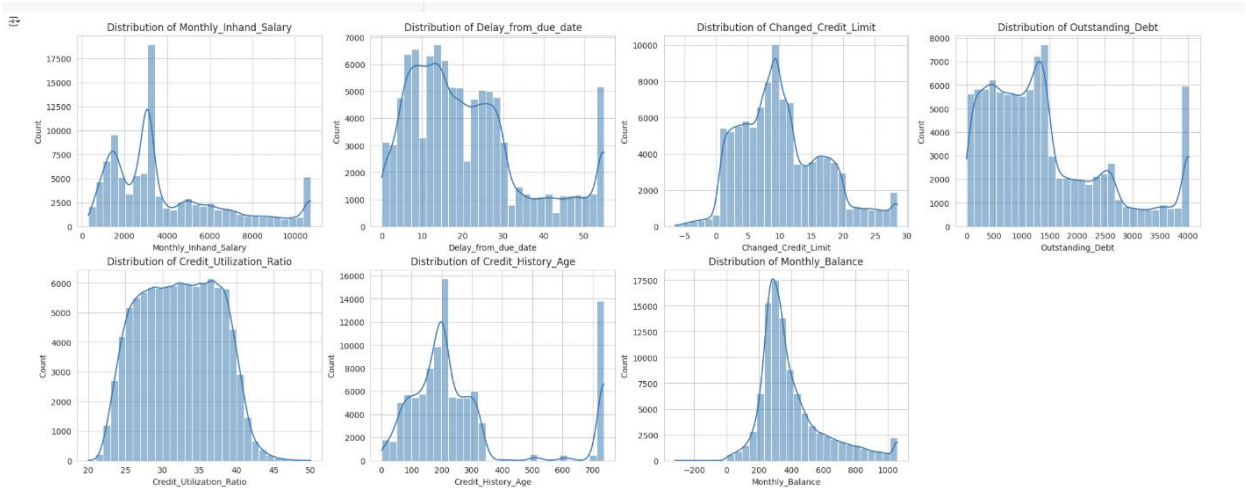
Distribution of Credit Scores

Standard: 53,174 records (53.2%)
Poor: 28,998 records (29.0%)
Good: 17,828 records (17.8%)

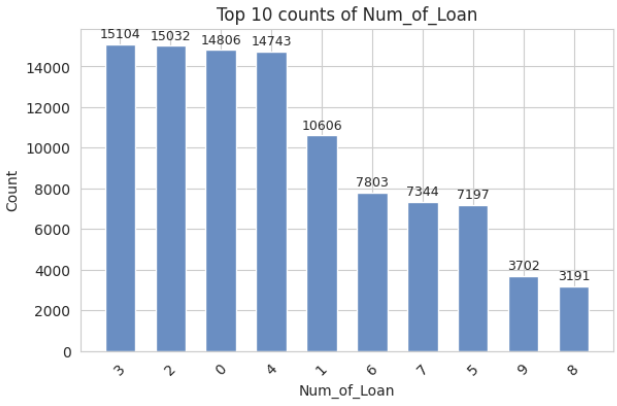


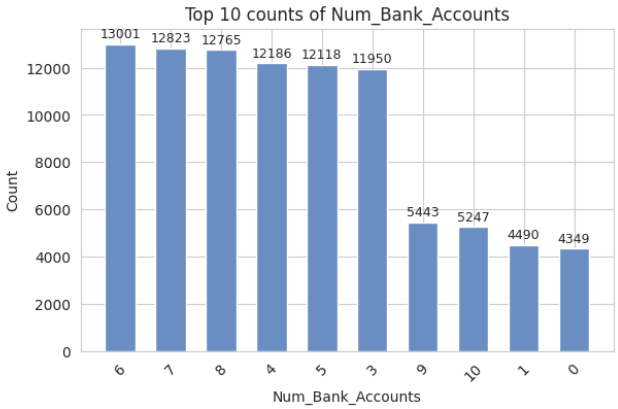
As we can see some class imbalance in the target variable, I decided not to artificially balance the classes because this data represents how credit ratings are distributed in the actual world and modifying it could make the model less applicable in real-world scenarios. I addressed the imbalance by employing modeling techniques that account for class frequencies and evaluation criteria rather than attempting to achieve balanced data. This method helps us to train the model on real world data. Univariate Visualization of Numerical data.

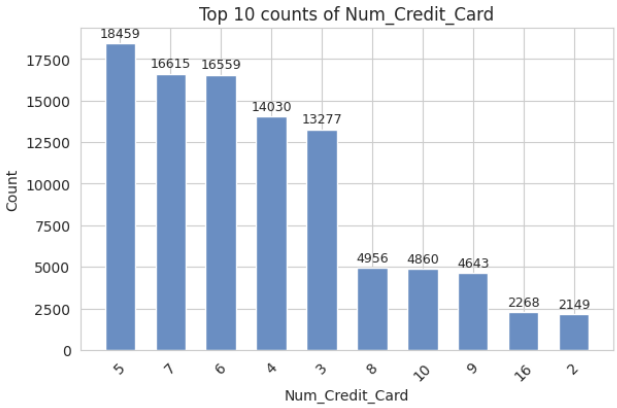
Univariate Analysis of Numerical Features:

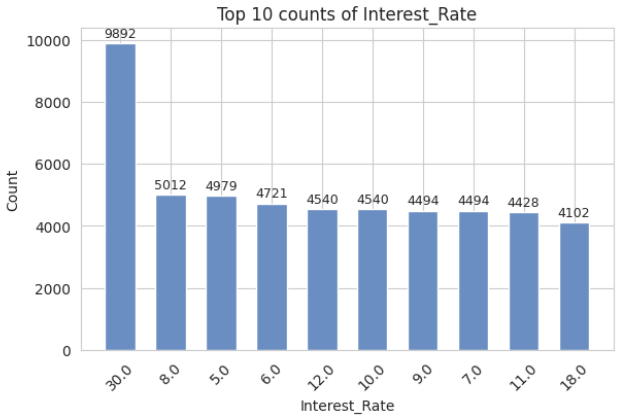


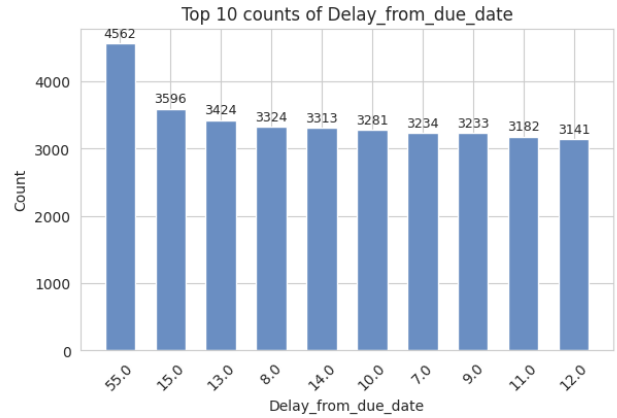
- ***Monthly Inhand Salary***
  When I looked at the distribution of **Monthly Inhand Salary**, I noticed it's right-skewed, with most people earning around ₹2000 to ₹3500. These seem like common salary slabs in the dataset. There are a few higher outliers, but they aren't very frequent.
- ***Delay from Due Date:***
  Many customers pay within 10–20 days past due, but there's a noticeable spike at the maximum delay (54 days), possibly indicating a system imposed cap or repeated defaulters.
- ***Changed Credit Limit:***
  Most customers experienced moderate changes (5–15 units), but smaller counts exist for very high or negative changes, indicating inconsistent credit behaviors or data edge cases.
- ***Outstanding Debt:***
  Debt levels cluster below ₹1500, yet a second spike around ₹4000 hints at a segment of customers with very high liabilities.
- ***Credit Utilization Ratio:***
  Values center tightly between 25%–40%, which is generally acceptable, indicating most customers are not over-leveraged. Very few go beyond 45%.
- ***Credit History Age (Months):***
  Most people have a credit history between 100–250 months (8–20 years), but there's a sharp rise near 700 months, which may reflect placeholder or capped values.
- ***Monthly Balance:***
  This feature shows a right-skewed bell shape, with most balances falling between ₹200 and ₹500. Negative balances are rare but exist, flagging some financial instability.

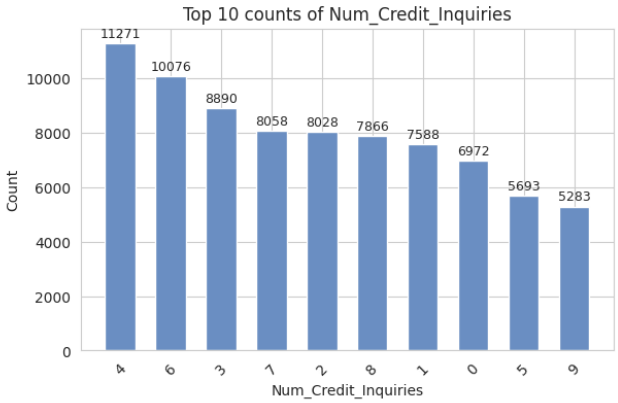

Top 10 counts of Num_of_Loan
Count
Num_of_Loan
3 2 0 4 1 6 7 5 9 8
15104 15032 14806 14743 10606 7803 7344 7197 3702 3191

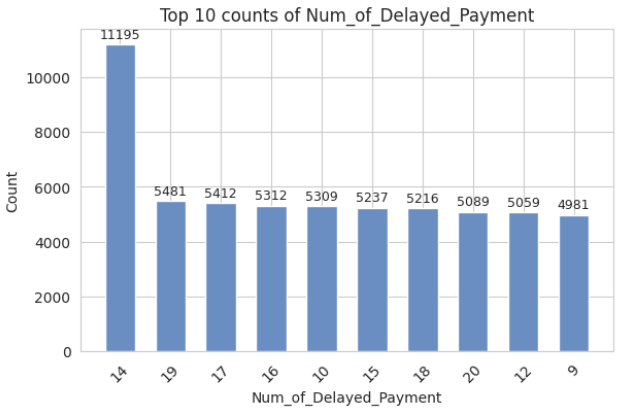

Top 10 counts of Num_Bank_Accounts
Count
Num_Bank_Accounts
6 7 8 4 5 3 9 10 1 0
13001 12823 12765 12186 12118 11950 5443 5247 4490 4349


Top 10 counts of Num_Credit_Card
Count
Num_Credit_Card
5 7 6 4 3 8 10 9 16 2
18459 16615 16559 14030 13277 4956 4860 4643 2268 2149


Top 10 counts of Interest_Rate
Count
Interest_Rate
30.0 8.0 5.0 6.0 12.0 10.0 9.0 7.0 11.0 18.0
9892 5012 4979 4721 4540 4540 4494 4494 4428 4102


Top 10 counts of Delay_from_due_date
Count
Delay_from_due_date
55.0 15.0 13.0 8.0 14.0 10.0 7.0 9.0 11.0 12.0
4562 3596 3424 3324 3313 3281 3234 3233 3182 3141

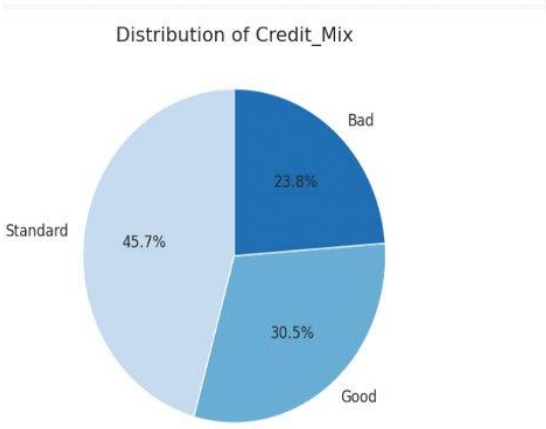

Top 10 counts of Num_Credit_Inquiries
Count
Num_Credit_Inquiries
4 6 3 7 2 8 1 0 5 9
11271 10076 8890 8058 8028 7866 7588 6972 5693 5283

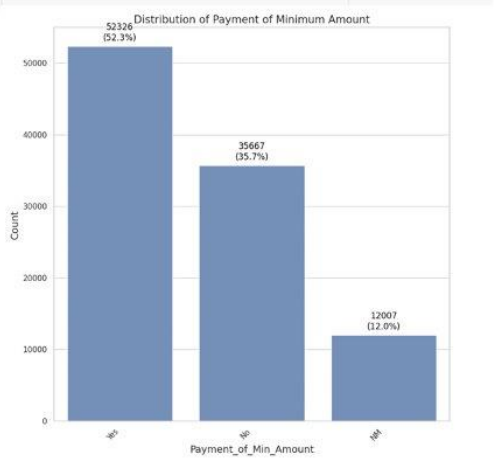

Top 10 counts of Num_of_Delayed_Payment
Count
Num_of_Delayed_Payment
14 19 17 16 10 15 18 20 12 9
11195 5481 5412 5312 5309 5237 5216 5089 5059 4981

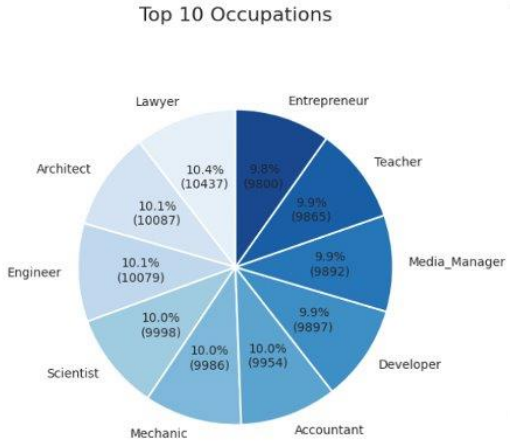
- ***Num_of_Loan:***
  Most individuals have between 2 to 4 loans, showing a moderate credit exposure.
  Interestingly, a good number have 0 active loans, which could either reflect financial stability or lack of credit history both important but very different scenarios in credit scoring.
- ***Num_Bank_Accounts:***
  People typically maintain 4 to 8 bank accounts, suggesting a preference for spreading finances across institutions. This can signal either financial organization or complexity, depending on transaction behaviors.
- ***Num_Credit_Card:***
  Having 3 to 5 credit cards is normal, but some individuals have more than 10+ cards, which may raise risk flags related to credit overextension or aggressive utilization which is an important predictor for default behavior.
- ***Interest_Rate:***
  The spike at 30% is unusual and likely reflects risk-based pricing or potential outliers. This level of interest is typically associated with subprime borrowers, and could be very informative when modeling poor credit scores.
- ***Delay_from_due_date:***
  The count spikes at 55 days, likely a maximum delay cap or habitual behavior. Most delays fall in the 10–20-day range, indicating recurring but not extreme payment issues.
- ***Num_Credit_Inquiries:***
  Most users have between 3 to 6 inquiries, suggesting moderate credit-seeking behavior. A very low number (0–1) may indicate financial stability or lack of credit involvement, while very high inquiries could suggest credit desperation.
- ***Num_of_Delayed_Payment:***
  The dominant spike at 14 delayed payments might indicate system-generated flags or repeated late behavior — this is likely a strong negative signal for creditworthiness.

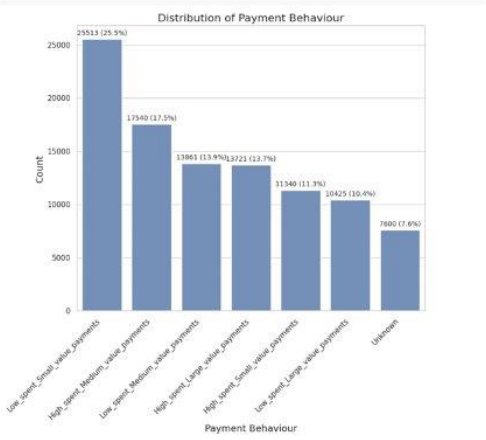
Univariate Analysis of Categorical Features:

- **Credit Mix**
  The majority of customers fall under *Standard* credit mix (45.7%), with *Good* (30.5%) and *Bad* (23.8%) making up the rest. This distribution may influence creditworthiness, as lenders often prefer clients with better credit mix.
- **Payment of Minimum Amount**
  Over 50% of individuals regularly pay the minimum amount due, while around 12% have ‘NM’ (Not Mentioned).This behavior is a strong indicator of repayment attitude and may directly affect risk profiles.









❖ **Occupation**

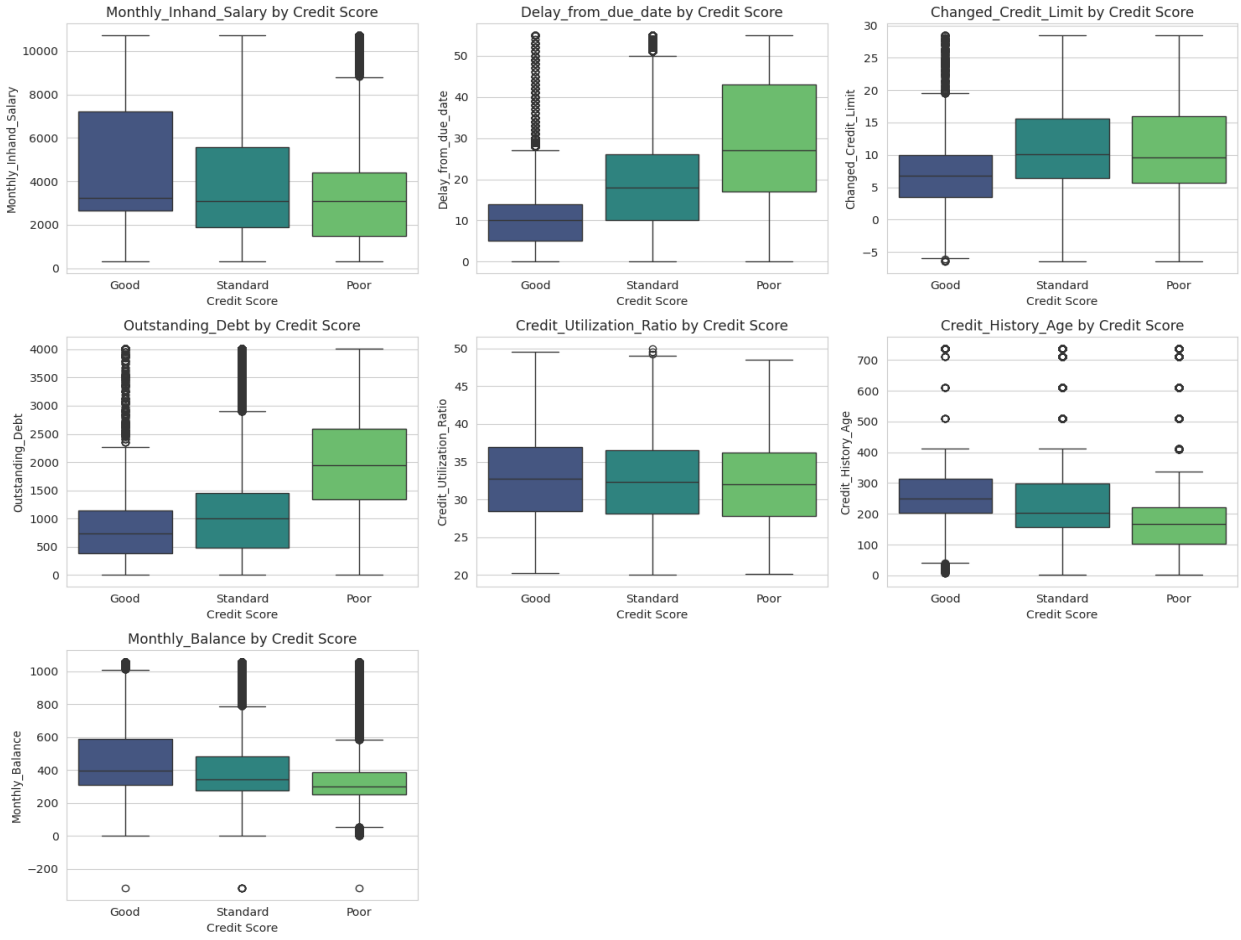
The top 10 occupations are quite evenly distributed, with *Lawyer*, *Architect*, and *Engineer* slightly leading. Diverse professional backgrounds reflect a balanced customer base, but some roles may correlate with better financial stability.

❖ **Payment Behaviour**

Low spent, Small value payments dominates with 25.5%, followed by medium and large payments across both low and high spenders. Spending and repayment patterns shown here can be engineered into behavioral features for better model performance.

BIVARIATIVE ANLYSIS:

NUMERICAL FEATURES VS CREDIT SCORE



These are some of the numerical features compared with credit score:

Insights from the above boxplot:

Monthly Inhand Salary

Customers with a Good credit score tend to have higher monthly in-hand income. Lower credit scores are associated with lower income, suggesting affordability plays a key role.

Delay from Due Date

Poor credit holders frequently delay payments longer. This delay sharply reduces as we move toward Good credit scores — timely payments matter.

Changed Credit Limit

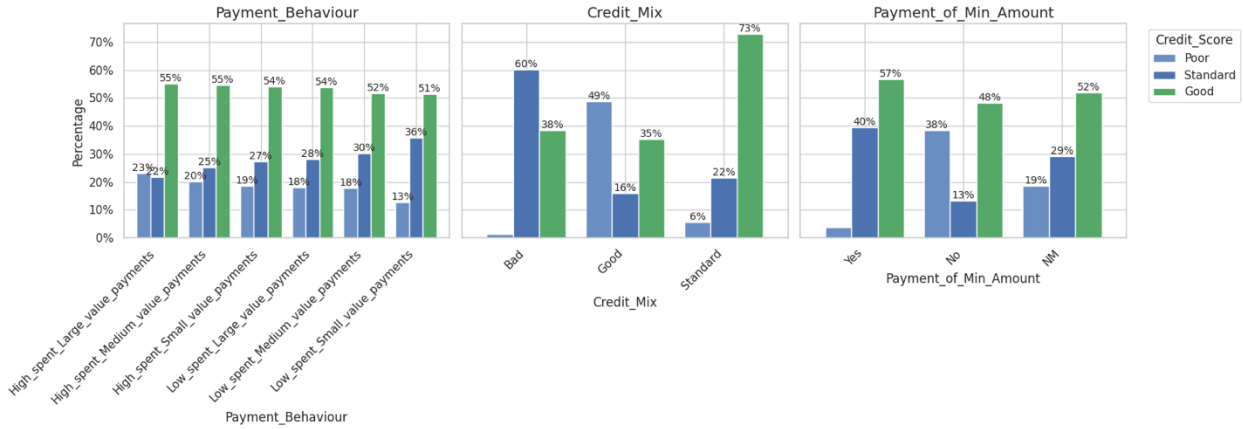
Those with lower scores show higher variation in credit limit changes, hinting at instability. *Good* credit individuals generally have moderate and stable limit changes.

Outstanding Debt

Higher outstanding debt is more common among *Poor* scorers. *Good* scorers typically maintain much lower debt levels.

Credit Utilization Ratio
A higher utilization ratio correlates with *Poor* credit scores. Balanced usage (below 30–35%) is more typical among *Good* scorers which shows responsible usage.
Credit History Age
People with longer credit histories generally score better. A short history is more often seen with *Poor* credit scores.
Monthly Balance
Good scorers maintain higher positive balances at month-end. Lower scores are linked to tight cash flow or even negative balances.

CATEGORICAL FEATURES VS CREDIT SCORE



Payment Behaviour: Users with low spending and small payments are more likely to have a Good credit score, while high spenders show a higher proportion of Poor scores.

Credit Mix: A Standard credit mix is strongly linked to Good scores. In contrast, a Bad credit mix often correlates with Poor credit performance.

Minimum Payment: Regularly paying even the minimum amount is associated with better scores. Those who skip it mostly fall into Poor or Standard categories.

5. FEATURE ENGINEERING

To make the data more informative for the model, I have created several new features based on financial logic and credit behavior patterns.
To make the dataset more informative for the models, I have also engineered a set of new features based on financial logic and credit behavior patterns.

First, I created a few financial ratio features:

- **Monthly Debt Ratio** = Total_EMI_per_month / Monthly_Inhand_Salary
  This feature tells us what portion of a person's salary goes toward EMI payments.
- **Annual Debt Ratio** = Outstanding_Debt / Annual_Income
  This captures how much debt they carry compared to their income.
- **Investment Rate** = Amount_Invested_Monthly / Monthly_Inhand_Salary
  It shows how much someone is able to invest from their salary.
- **Disposable Income Ratio** = (Monthly_Inhand_Salary - EMI) / Monthly_Inhand_Salary
  This helps assess how much money is left after fixed payments.

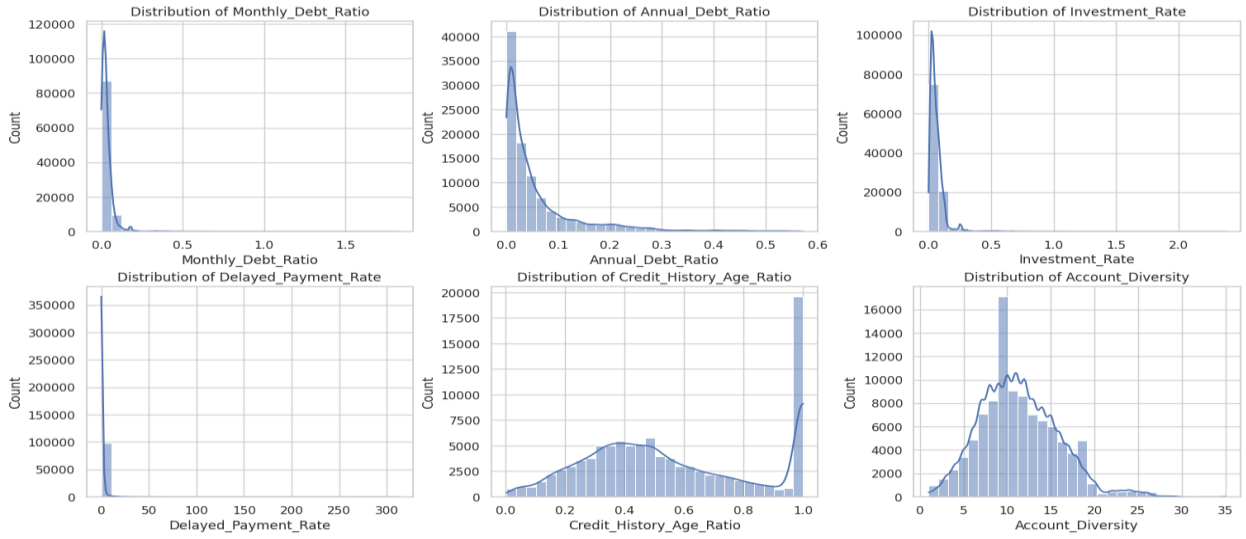
These ratios help the model more information about the financial stress or capacity than raw income or debt values alone.

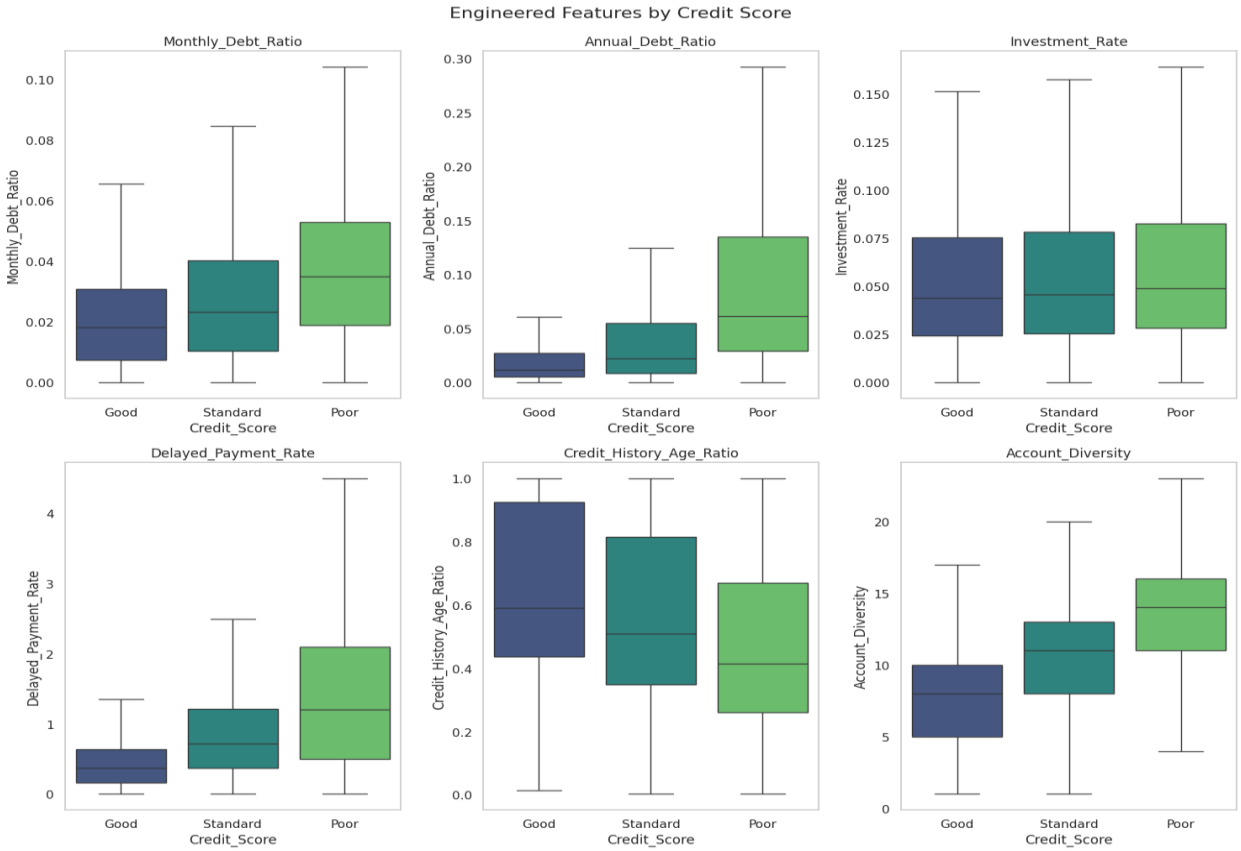
Next, I worked on **credit history interaction-based and derived features**

- I extracted binary indicators like whether a person is a high spender or makes large payments from their payment behavior.
- I converted the Credit_Mix category into numeric scores for model compatibility.
- I also calculated how frequently payments are delayed or credit inquiries are made, normalized by credit history length. This helps identify patterns of financial reliability.
- Another feature combined the number of bank accounts and credit cards to reflect account diversity, which can influence credit scores.
- Credit_History_Age_Ratio compares someone's credit history length with their actual age.
- Income_Debt_Payment_Interaction blends income, debt, and delay behavior to reflect financial discipline.
- Finally, I squared the credit utilization ratio to help capture any nonlinear effects it might have.

Together, these features transformed the raw dataset into something much richer — giving the model a more complete and nuanced view of each customer's financial situation and behavior.

6.EDA (AFTER FEATURE ENGINEERING)



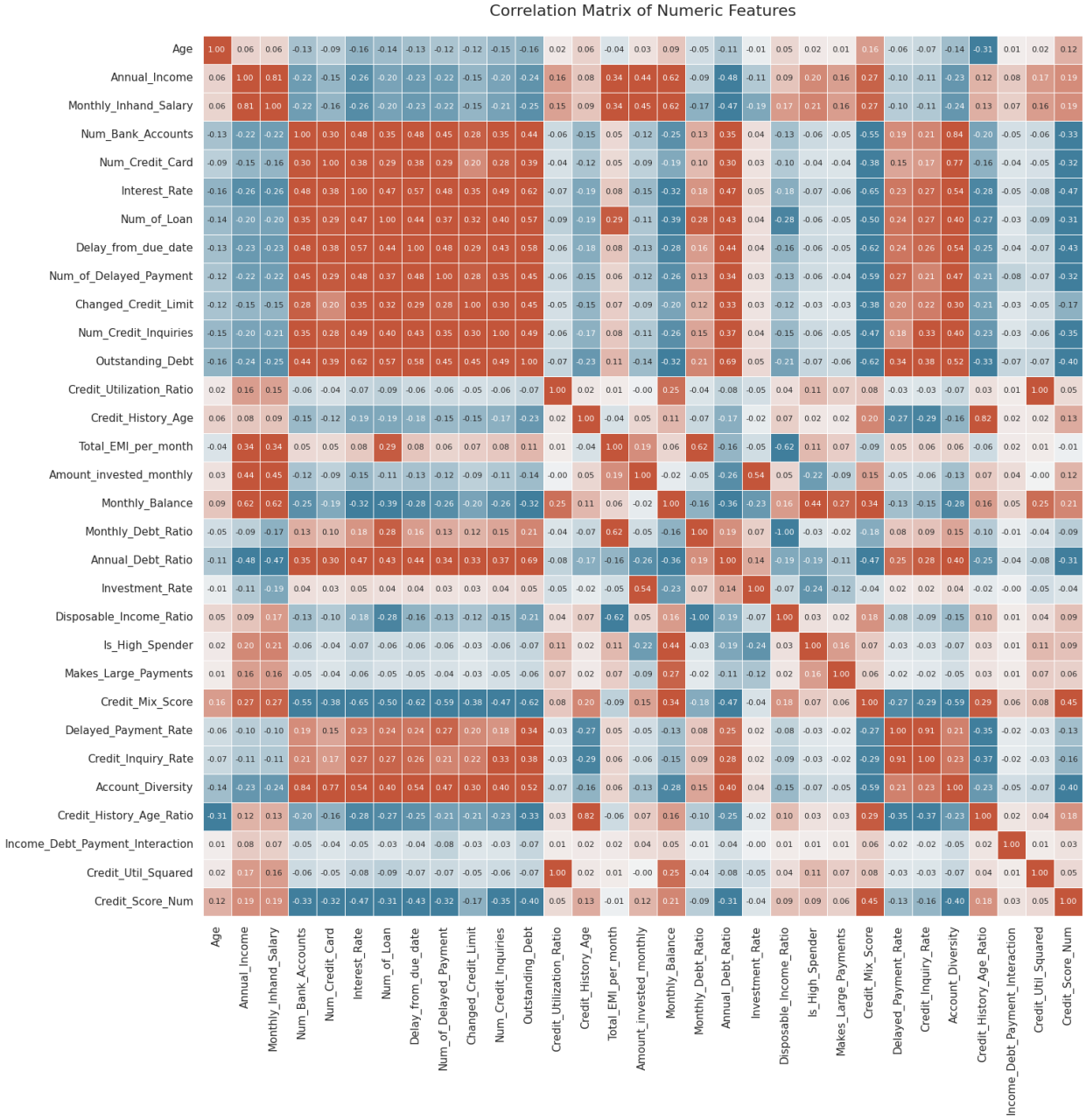


From the above box plots, I have observed this:

I see that people with poor credit scores tend to have higher monthly and annual debt ratios, meaning a bigger portion of their income goes toward paying off debts. Their delayed payment rates are also higher, which suggests they miss payments more often. On the other hand, people with good scores generally have lower debt and fewer delays.

Looking at investment rate and account diversity, those with poor credit scores slightly invest more and also seem to have more bank accounts and credit cards. Interestingly, people with good credit scores have a higher credit history age ratio, meaning they've been using credit longer compared to their age, which might contribute to their better scores.

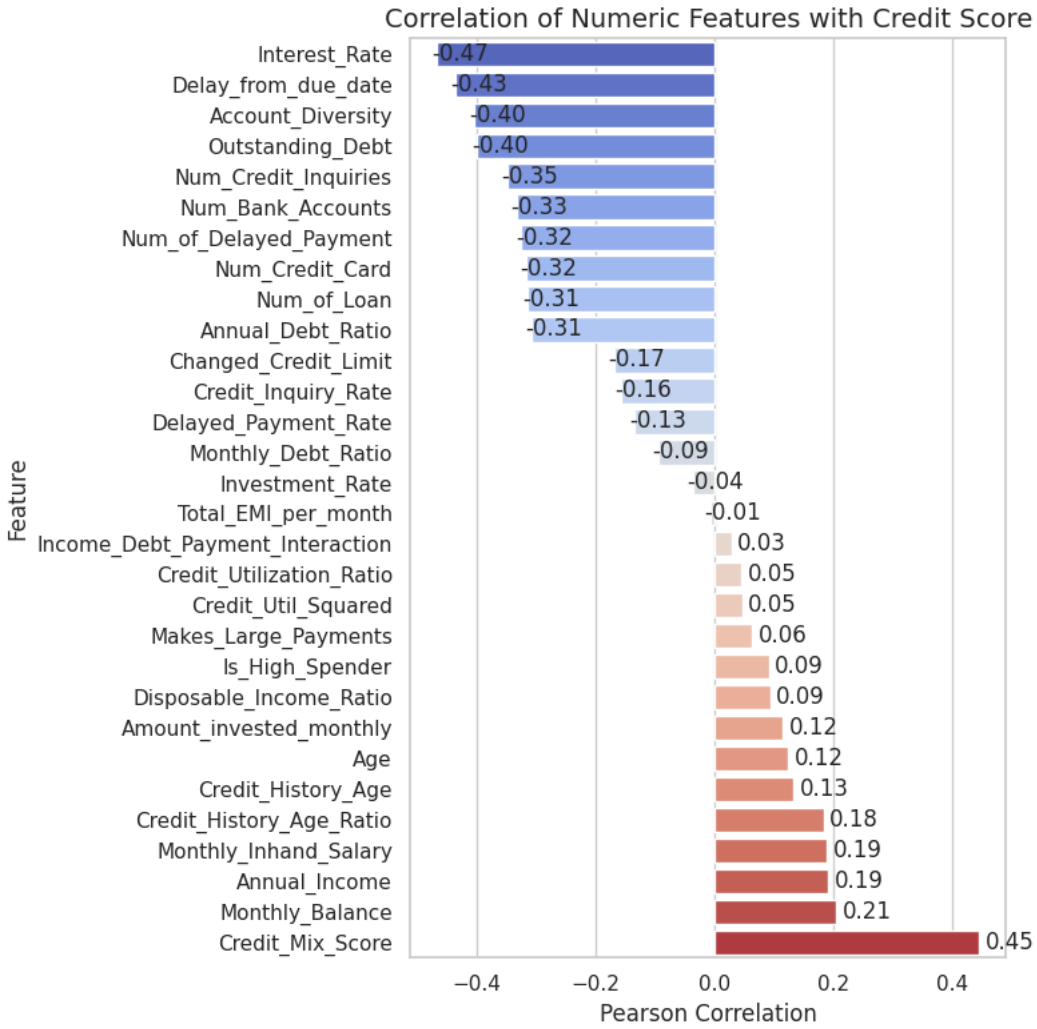
CORRELATION MATRIX

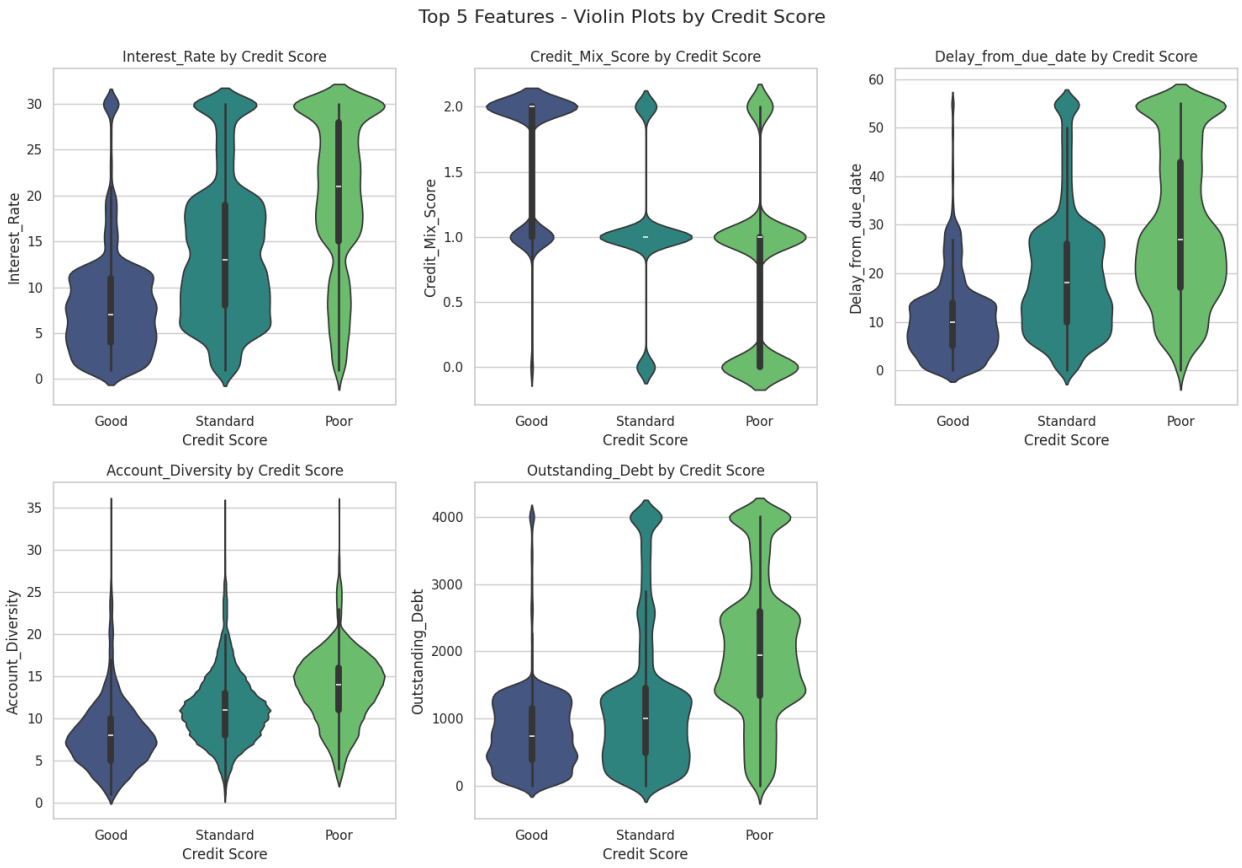


This correlation matrix shows how different numeric features relate to each other. Strong positive values (closer to 1) mean features move together, while strong negative values (closer to -1) move in opposite directions. It helps identify which financial factors are most connected to credit score and to each other.

We also have violin plots below which shows the top 5 features which gives more information or related to credit score.

This Pearson Correlation shows which numeric features impact credit score. Features on the right (positive correlation) like Credit_Mix_Score and Monthly_Balance are linked to higher scores, while those on the left (negative correlation) like Interest_Rate and Delay_from_due_date are linked to lower scores.





7.MODEL BUILDING

Before implementing the models, I went through several preprocessing steps to make sure the data was ready for training.

I started by encoding the target variable, Credit_Score, using *LabelEncoder*. Since this variable is ordinal with categories like *Poor*, *Standard*, and *Good*, I converted them into numerical values 0, 1, and 2. This format is easier for most machine learning algorithms to work with.

Next, I handled the categorical input features using one-hot encoding. This method allowed me to turn each category into a binary column. I also used the *drop_first=True* setting to avoid introducing multicollinearity among the features, which can affect model performance.

For splitting the data, I used an 80-20 train-test split, making sure to apply stratified sampling. This ensured that all three credit score categories were proportionally represented in both the training and testing sets, preserving the natural distribution of the data.

Finally, I standardized all numerical features using *StandardScaler*. This scaled each feature to have a mean of zero and a standard deviation of one, which helps the models treat all features equally—especially important when some values are on different scales or units.

8.MODELS:

To classify credit scores effectively, I experimented with four different machine learning models. Each model brings its own strengths and together they allowed me to explore various ways of capturing patterns in financial behavior.

Logistic Regression

I started with logistic regression as a baseline model. It's a linear classifier that works well when the relationship between input features and the target is roughly linear. Since credit scores are ordered categories (*Poor, Standard, Good*), logistic regression with an OvR (One-vs-Rest) strategy helped break the problem into separate binary classifications. It's simple, interpretable, and gives a good sense of how individual features influence the outcome.

Random Forest Classifier

Next, I used a Random Forest classifier, which is an ensemble of decision trees. It handles non-linear relationships and feature interactions much better than logistic regression. What I like about Random Forests is their robustness—they reduce overfitting by averaging multiple trees and are less sensitive to noise in the data. It also gives good feature importance scores, which helped me understand which variables influenced credit scores the most.

Gradient Boosting

Gradient Boosting builds trees sequentially, where each new tree tries to fix the errors of the previous one. This makes it a powerful model for structured/tabular data like this. Compared to Random Forests, Gradient Boosting models usually perform better but can be more sensitive to parameter tuning. I used it here to capture more complex relationships in the data and to see if boosting improved accuracy compared to bagging (Random Forest).

K-Nearest Neighbors (KNN)

I also included KNN because it's a simple yet intuitive model. It classifies a sample based on the majority class among its 'k' closest points. In the context of credit scoring, KNN helps by looking at similar customer profiles to make predictions. However, it's highly dependent on feature scaling (which I applied using StandardScaler), and it can struggle with high-dimensional data or class imbalance. Still, it served as a good contrast to tree-based methods.

Decision-Tree Classifier

The Decision Tree model works like a flowchart, splitting the data based on yes/no questions. It's very easy to understand and visualize, which makes it great for interpretation. However, if not controlled, it can become too complex and overfit the training data.

Support Vector Machine

SVM tries to find the best boundary that separates different categories in the data. It's powerful for complex datasets, especially when the data isn't clearly separated. That said, it can be slower and less efficient when dealing with very large datasets.

AdaBoost Classifier

AdaBoost builds a series of models, each one focusing more on the errors made by the previous one. It's good at improving accuracy and works well on classification problems. But it can be a bit sensitive to noisy or messy data.

EVALUATION METIRCS

Logistic Regression:

Logistic Regression served as a baseline classification model for predicting credit scores. Despite its simplicity, it achieved a reasonable overall accuracy of 64%. The model handled the 'Standard' class more confidently compared to 'Good' and 'Poor', as seen in the confusion matrix.

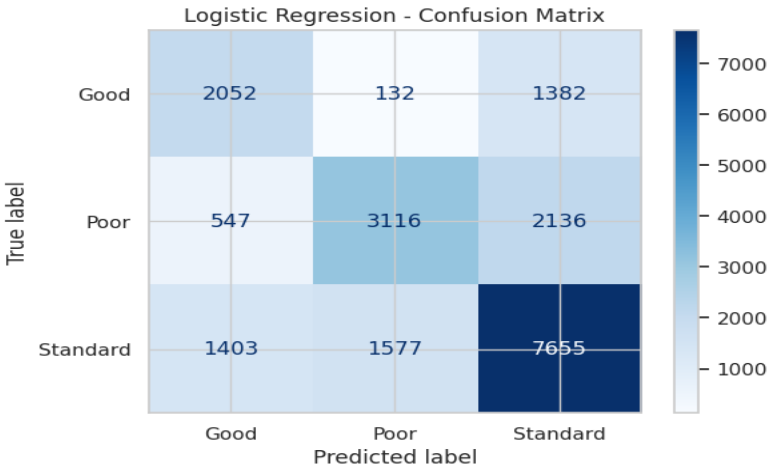
From the classification report:

- **Standard** class had the highest F1-score of **0.70**, with both precision and recall above 0.69.
- **Good** class performance was weaker, with a precision of **0.51** and recall of **0.58**, indicating that many 'Good' customers were misclassified, primarily as 'Standard'.
- **Poor** class had a decent precision (**0.65**) but struggled on recall (**0.54**), showing inconsistent identification of truly poor credit profiles.

Logistic Regression – Classification Report:

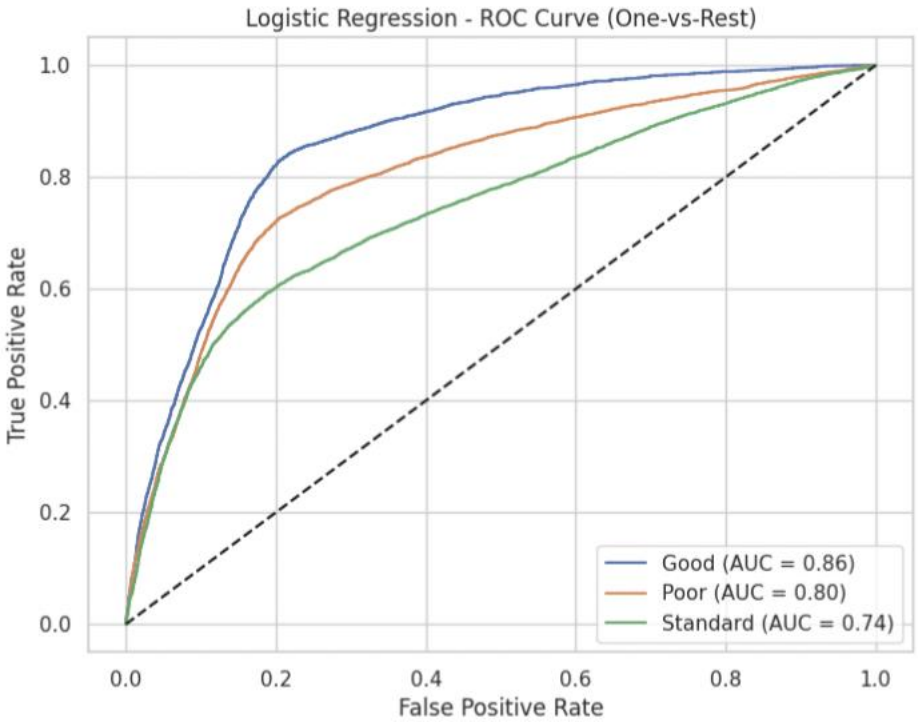
	precision	recall	f1-score	support
Good	0.51	0.58	0.54	3566
Poor	0.65	0.54	0.59	5799
Standard	0.69	0.72	0.70	10635
accuracy			0.64	20000
macro avg	0.61	0.61	0.61	20000
weighted avg	0.64	0.64	0.64	20000

The **confusion matrix** shows that 'Standard' was frequently predicted often by misclassifying the other two classes. This is likely due to the slightly imbalanced class distribution and the model's linear nature, which may not capture complex non-linear boundaries.



The **ROC-AUC scores** further supported these findings:

- **Good**: 0.86
- **Poor**: 0.80
- **Standard**: 0.74



These indicate that the model has a better ability to distinguish 'Good' customers from the rest, while 'Standard' overlaps more with others.

While simple and interpretable, Logistic Regression falls short in capturing the complex, non-linear relationships in credit score classification.

Random Forest Classifier:

Overall Accuracy: 78% — a significant jump from logistic regression, indicating better capture of non-linear relationships.

1.Classification Report

Class-wise F1-Scores:

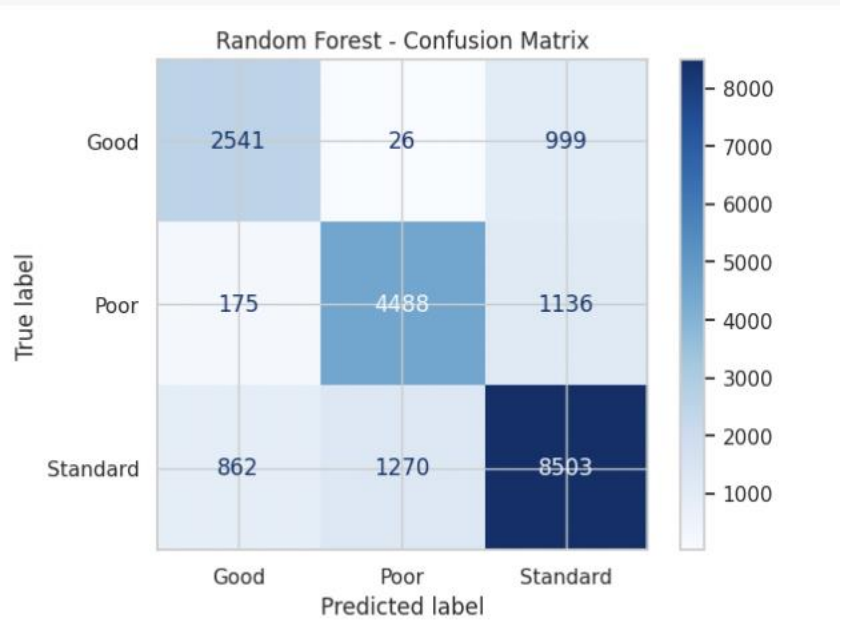
Good: 0.71,**Poor:** 0.77,**Standard:** 0.80
All three classes were handled quite evenly, showing balanced learning.

Random Forest – Classification Report:

	precision	recall	f1-score	support
Good	0.71	0.71	0.71	3566
Poor	0.78	0.77	0.77	5799
Standard	0.80	0.80	0.80	10635
accuracy			0.78	20000
macro avg	0.76	0.76	0.76	20000
weighted avg	0.78	0.78	0.78	20000

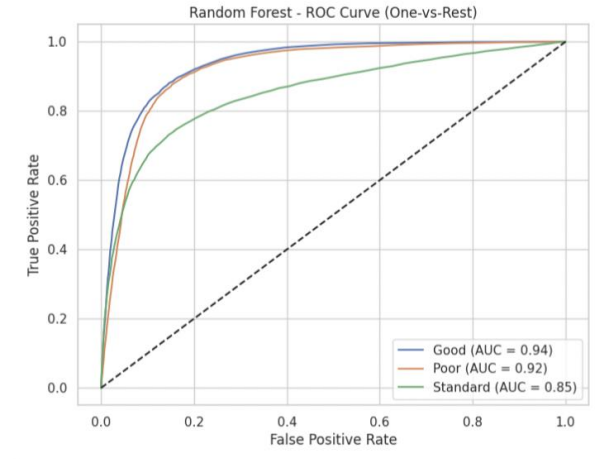
2.Confusion Matrix

Most misclassifications for 'Standard' class were between 'Poor' and 'Good' — a common challenge in ordinal classification. Very few 'Good' cases were wrongly predicted as 'Poor' (just 26), which is a good sign of the model’s reliability for high-credit individuals.



3.**AUC (One-vs-Rest):**

- Good = **0.94**
- Poor = **0.92**
- Standard = **0.85**
  These AUC scores suggest that the model distinguishes between all classes effectively, especially for the more extreme ends of the credit spectrum.

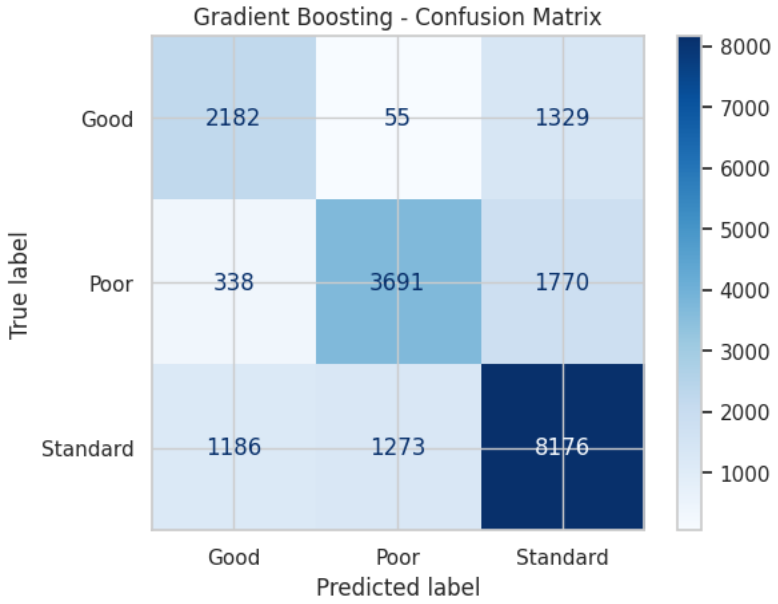


GRADIENT BOOSITING:

I used Gradient Boosting because it's great at handling structured data and can capture complex patterns by combining multiple weak learners.

1. Confusion Matrix Insights:

- It correctly identified **2182 Good**, **3691 Poor**, and **8176 Standard** credit scores.
- Misclassifications mostly happened between neighboring classes — for example, **Good** was often confused with **Standard**, which makes sense given their similarities.



2. Classification Report:

- **Precision:** Highest for *Poor* (0.74), meaning when it predicts Poor, it's mostly right.
- **Recall:** Highest for *Standard* (0.77), so it didn't miss many actual Standard cases.
- Overall accuracy: **70%**, better than Logistic Regression and a solid improvement.

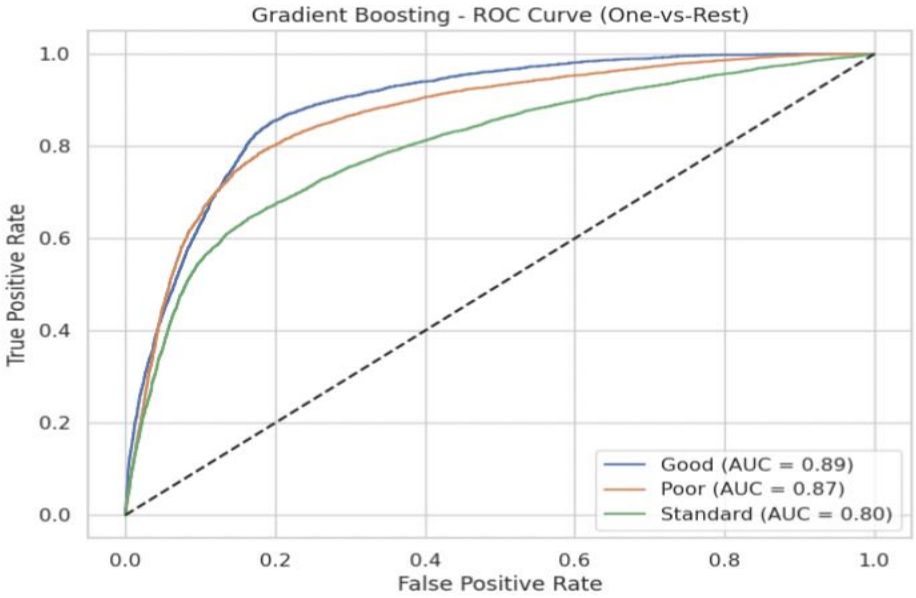
Gradient Boosting – Classification Report:

	precision	recall	f1-score	support
Good	0.59	0.61	0.60	3566
Poor	0.74	0.64	0.68	5799
Standard	0.73	0.77	0.75	10635
accuracy			0.70	20000
macro avg	0.68	0.67	0.68	20000
weighted avg	0.70	0.70	0.70	20000

3. ROC-AUC Score:

- AUC for:
  - **Good:** 0.89
  - **Poor:** 0.87
  - **Standard:** 0.80
    These are strong scores, showing the model distinguishes well between classes.

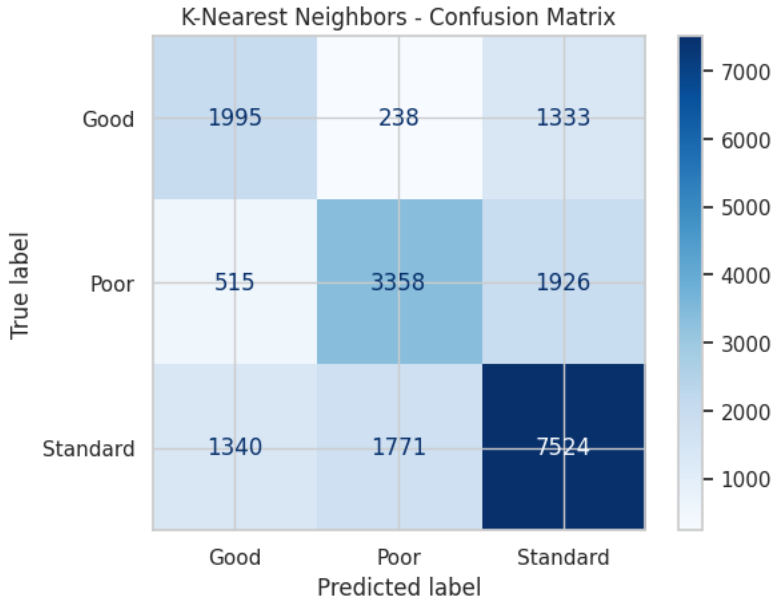
In short, Gradient Boosting performed very well in both identifying the correct credit categories and minimizing false predictions. It balances precision and recall effectively, especially for the *Standard* and *Poor* segments.



K-Nearest-Neighbor

KNN works by comparing each test record to the most similar data points from the training set. In this case, the model achieved an overall accuracy of **64%**.

1.Confusion Matrix:



2.**Classification Report**

From the classification report, it's clear that KNN performs better on the **Standard** credit score group, with an F1-score of **0.70**, but struggles more on **Good** and **Poor** categories, with F1-scores of **0.54** and **0.60** respectively. This suggests that KNN tends to perform better when the class is more balanced or has distinct feature patterns.

K-Nearest Neighbors - Classification Report:

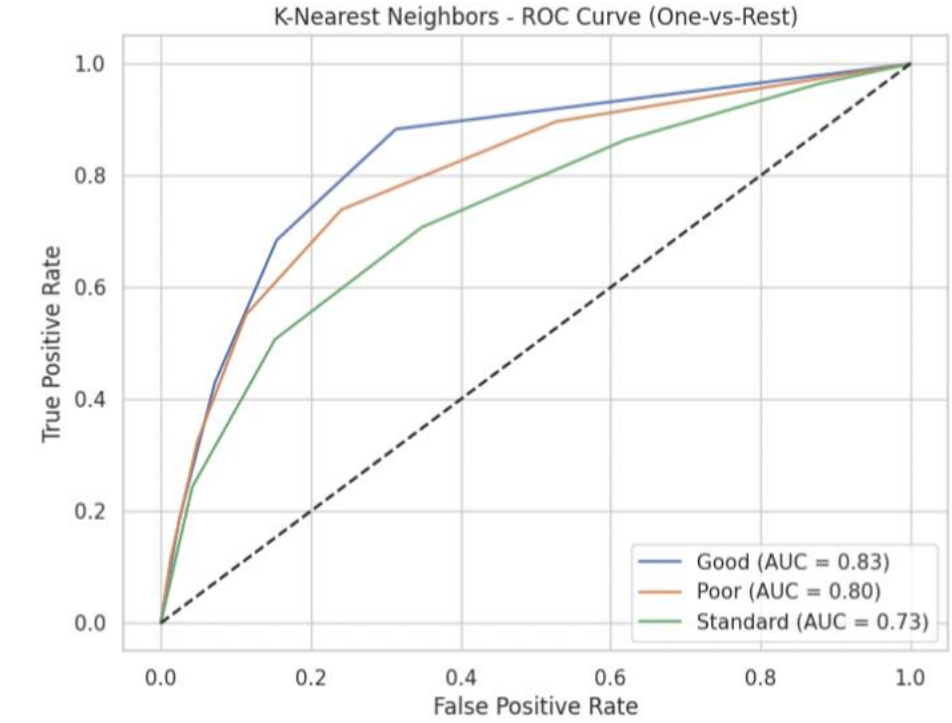
	precision	recall	f1-score	support
Good	0.52	0.56	0.54	3566
Poor	0.63	0.58	0.60	5799
Standard	0.70	0.71	0.70	10635
accuracy			0.64	20000
macro avg	0.61	0.62	0.61	20000
weighted avg	0.64	0.64	0.64	20000

3.ROC Curve

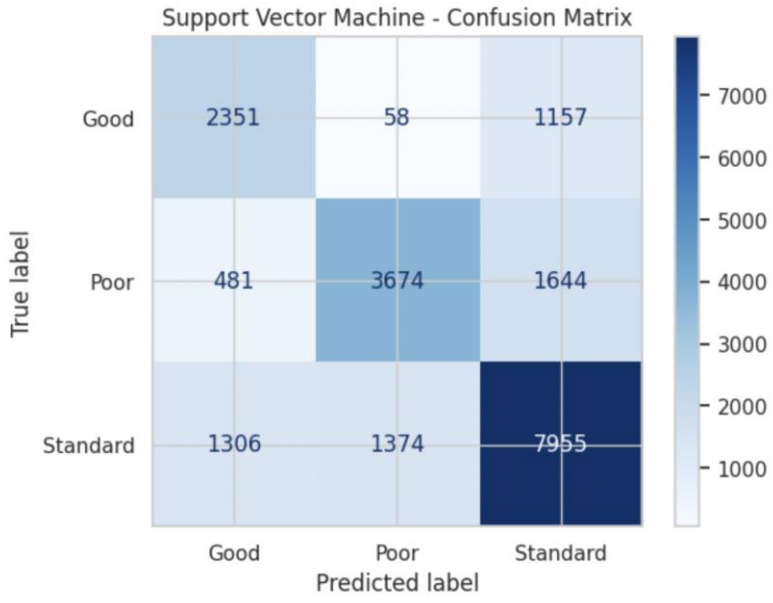
Looking at the ROC curve, the model does a decent job of separating the classes:

- **Good**: AUC = 0.83
- **Poor**: AUC = 0.80
- **Standard**: AUC = 0.73

These AUC values show that the model distinguishes “Good” credit scores best among the three. While not the strongest model overall, KNN still gives a fair baseline for performance and highlights how similar customer profiles affect prediction.

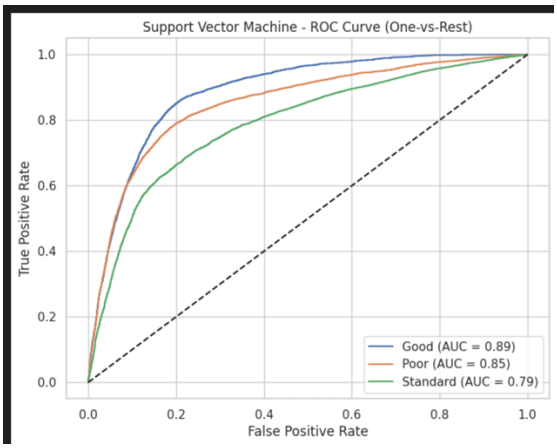


Support Vector Machine



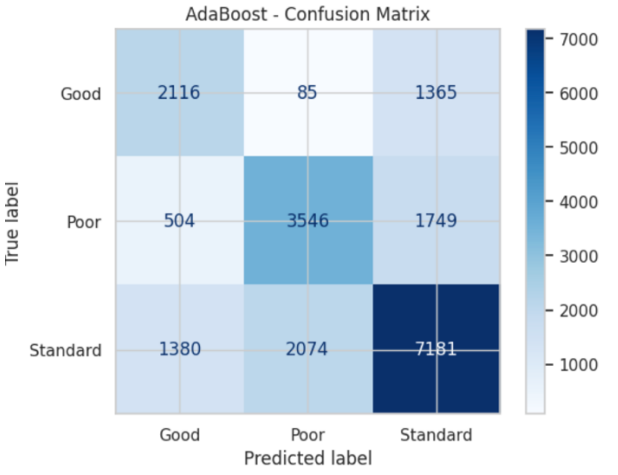
Support Vector Machine - Classification Report:

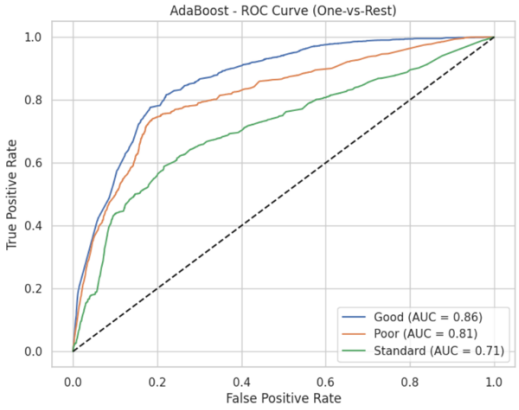
	precision	recall	f1-score	support
Good	0.57	0.66	0.61	3566
Poor	0.72	0.63	0.67	5799
Standard	0.74	0.75	0.74	10635
accuracy			0.70	20000
macro avg	0.68	0.68	0.68	20000
weighted avg	0.70	0.70	0.70	20000



1. The Support Vector Machine model achieved an overall **accuracy of 68%**.
2. It performed best in predicting the **Standard** category, with the highest F1-score of **0.78**.
3. The **Good** category had the lowest performance, with an F1-score of **0.61**.
4. Some misclassifications happened between **Good and Standard**, as seen in the confusion matrix.
5. The ROC curve shows the model has **decent AUC values**, with **Good class AUC = 0.88**, indicating fair classification ability.

Ada Boost



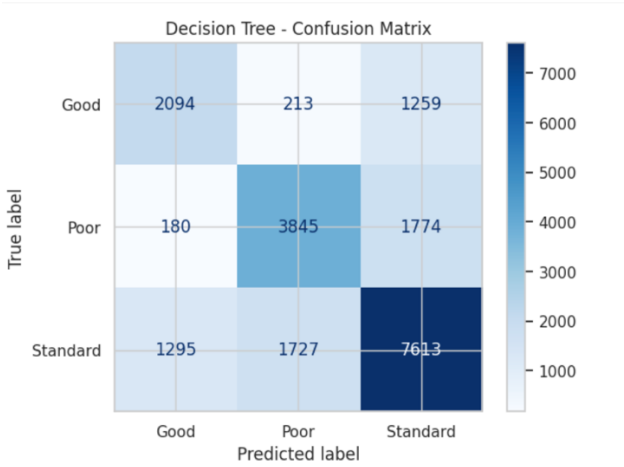


AdaBoost – Classification Report:

	precision	recall	f1-score	support
Good	0.53	0.59	0.56	3566
Poor	0.62	0.61	0.62	5799
Standard	0.70	0.68	0.69	10635
accuracy			0.64	20000
macro avg	0.62	0.63	0.62	20000
weighted avg	0.65	0.64	0.64	20000

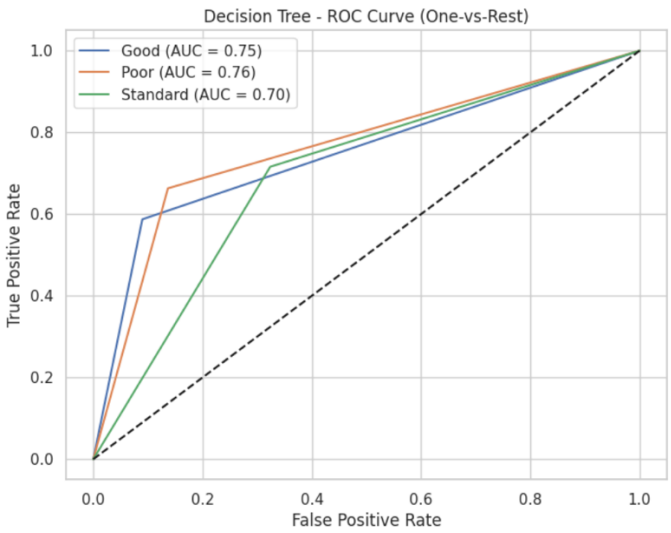
1. The AdaBoost model achieved an overall **accuracy of 64%**, slightly lower than SVM.
2. It performed best in predicting the **Poor** category, with an **F1-score of 0.69**.
3. The **Standard** category had the weakest performance, with an F1-score of **0.60**.
4. Misclassifications are noticeable, especially between **Standard and Poor** classes.
5. The **ROC curve** shows a fair performance overall, with the **Good class AUC at 0.86**.

Decision Tree



Decision Tree – Classification Report:

	precision	recall	f1-score	support
Good	0.59	0.59	0.59	3566
Poor	0.66	0.66	0.66	5799
Standard	0.72	0.72	0.72	10635
accuracy			0.68	20000
macro avg	0.66	0.66	0.66	20000
weighted avg	0.68	0.68	0.68	20000



- The Decision Tree model achieved an overall accuracy of 67 %.
- It performed best in predicting the **Standard** and **Poor** categories, both with an F1-score of **0.66**.
- The **Good** category had the lowest F1-score at **0.59**, indicating weaker performance.
- Misclassifications are mainly between **Good and Standard**, and also **Standard and Poor**.
- The ROC curve shows fair performance, with **AUC values** of **0.75 for Good**, **0.76 for Poor**, and **0.70 for Standard**.

COMPARISION

Model	Accuracy
Random Forest	77.66
Gradient Boosting	70.25
Logistic Regression	64.11
K-Nearest Neighbor	64.39
Support-Vector Machine	68.32
Ada Boost	64.52
Decision Tree	67.64

IMPROVEMENTS

we can extend this credit score prediction system more powerful and user-friendly by adding several future enhancements. For example, a real-time monitoring system could continuously track financial behavior and update scores instantly rather than relying on periodic checks.

Integrating the model into a mobile app would allow users to see how the information like missing a payment or increasing credit usage which might impact their score before making decisions. We could also provide personalized improvement plans that guide users step-by-step based on their financial habits. Additionally, tools like “what-if” simulators would help lenders visualize how different market conditions might affect customer risk. Lastly, a recommendation system based on behavior patterns of similar users could help in making smarter credit-related suggestions. These improvements would benefit both institutions and customers by making the system more accurate and inclusive.

CONCLUSION

In this project, I set out to build a credit risk prediction system that classifies customers into Poor, Standard, and Good credit score categories. I started by carefully cleaning and exploring the data, then engineered useful features to better capture financial behavior.

Out of all the models I tested, Random Forest performed the best with an accuracy of around 78%. It was followed by Gradient Boosting at 70%, and both K-Nearest Neighbors and Logistic Regression around 64%. Random Forest stood out because it does a great job capturing complex patterns in the data especially non-linear relationships.

From my analysis, I found that certain features like the credit utilization ratio, payment history, and debt-to-income levels had a strong impact on credit scores. These kinds of insights can be really helpful for banks and lenders when making credit decisions.

Overall, the model I built shows how machine learning can support better risk assessment. It has the potential to reduce loan defaults while helping trustworthy borrowers get access to credit more easily.

References:

I have taken some references from Kaggle to implement this credit score classification:

https://www.kaggle.com/datasets/parisrohan/credit-score-classification

Code File