CMP4011 Big Data and Cloud Computing

Project Report

Team 9

Name	Sec	B.N	Code
Ahmed Hosny	1	2	9202077
Ahmed Sabry	1	4	9202119
Basma Elhoseny	1	16	9202381
Zeinab Moawad Fayez	1	28	9202611

Contents

Problem Statement:

To assess the risk associated with approving a loan request from a customer, bankers rely on evaluating the customer's transactional history. We propose the development of an intelligent system that would analyze customer credit information, encompassing payment history, credit utilization, and length of credit. It would then assign a credit score to the customer, which the bank would use to determine loan approval based on a predefined threshold.

Data Set:

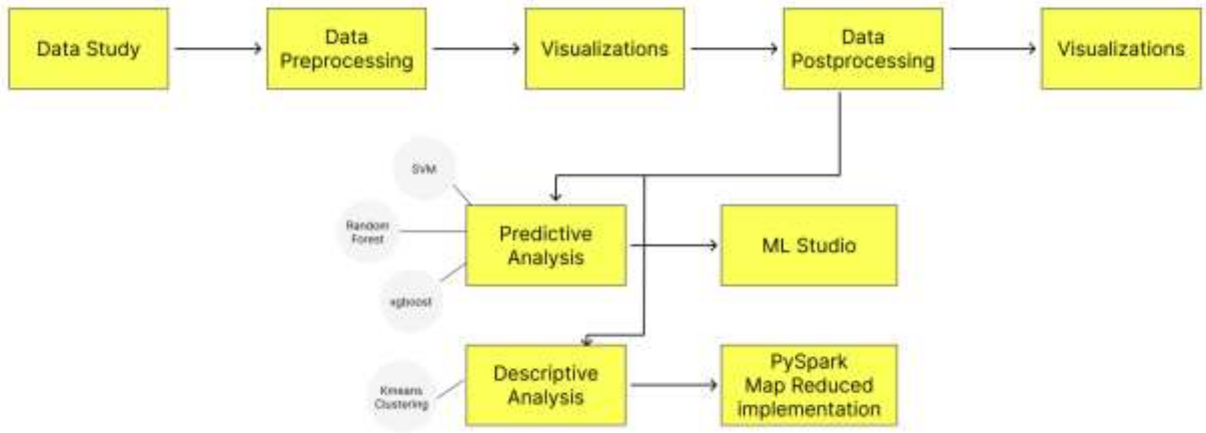
- **Main:** https://www.kaggle.com/datasets/parisrohan/credit-score-classification [31.14 MB 150K Example] [25 useful feature excluding name, id, ssn, and month

- **Features Description:**

	Feature	Description
1	ID	
2	Customer_ID	
3	Month	
4	Name	
5	Age	
6	SSN	
7	Occupation	The type of job or profession that the individual is engaged in
8	Annual_Income	Represents the annual income of the person.
9	Monthly_Inhand_Salary	Represents the monthly base salary of a person
10	Num_Bank_Accounts	Represents the number of bank accounts a person holds
11	Num_Credit_Card	Represents the number of other credit cards held by a person
12	Interest_Rate	Represents the interest rate on credit card
13	Num_of_Loan	Represents the number of loans taken from the bank
14	Type_of_Loan	Represents the types of loan taken by a person
15	Delay_from_due_date	Represents the average number of days delayed from the payment date
16	Num_of_Delayed_Payment	Represents the average number of payments delayed by a person
17	Changed_Credit_Limit	Represents the percentage change in credit card limit
18	Num_Credit_Inquiries	Represents the number of credit card inquiries The term "credit card inquiries" typically refers to requests made by individuals to obtain new credit cards
19	Credit_Mix	efers to the variety of credit types that an individual has in their credit profile. Lenders and credit scoring models consider credit mix as one of the factors when assessing creditworthiness.
20	Outstanding_Debt	Represents the remaining debt to be paid (in USD)
21	Credit_Utilization_Ratio	Represents the utilization ratio of credit card
22	Credit_History_Age	Represents the age of credit history of the person

23	Payment_of_Min_Amount	represents whether a customer has made only the minimum required payment on their financial obligation, such as a credit card bill or loan installment.
24	Total_EMI_per_month	Represents the Equated Monthly Installments payments (in USD)
25	Amount_invested_monthly	Represents the monthly amount invested by the customer (in USD)
26	Payment_Behaviour	typically, does not refer to an amount expressed in USD. Instead, it usually represents the pattern or history of how a customer makes payments on their financial obligations. This includes things like credit card bills, loan payments, utility bills, and other recurring expenses.
27	Monthly_Balance	Represents the monthly balance amount of the customer (in USD)
28	Credit_Score	Represents the bracket of credit score (Poor, Standard, Good)

Project Pipeline:



Analysis and Solutions:

Data Preprocessing:

DataPreprocessing Class includes all pre/post processing done on data: [data_preprocessing.py]

Preprocessing:

1. Drop unimportant features such as:
    a. 18ID
    b. Customer_ID
    c. Name
    d. SSN
    e. Month
    f. Credit_History_Age (for time series analysis)

2. Correct and Cleaning of some values like age containing 29_ & Conversion of numerical values read from csv as string back to numerical such as num_of_loans.
3. Handling Missing Values based on series of 8 (because 8 record represents same customer in 8 different months (Local [Per Customer])
    a. Categorial Missed Data are replaced by the most frequent within the 8 records of the customer.
    b. Continuous Missed Data are replaced by the mean of the 8 records of the customer.
    c. Continuous [non-float] Missed Data are replaced by the mod of the 8 records of the customer (such as age)
4. Convert Categorial Data to One hot encoder or Label encoded [According to the model used]
5. Normalization for Numerical Data [For Kmeans]
6. Standardization for Numerical Data [For other Predictive Models]

Postprocessing:

1. Further Cleaning for the data observed from visualizations 😉 (Global [Not per customer])
    a. Replace !@9#%8 in Payment_Behaviour by Low_spent_Small_value_payments.
    b. Replace _ in Credit_Mix by Standard which is the value for most of the customers.
2. Removing outliers which are out of bound of the normal distribution of the feature [mean ± 3*std]
3. Split the type_of_loan column into 9(unique) columns such that each is binary 0 or 1 [later we find out that these features are useless, they are the least features contributing to the credit_score decision (This idea is clear below 😊)]

Data Visualization:

After postprocessing Step we have used different visualizations to take insights from the data [preprocessing_visualizations.ipynb]

Categorial Features:

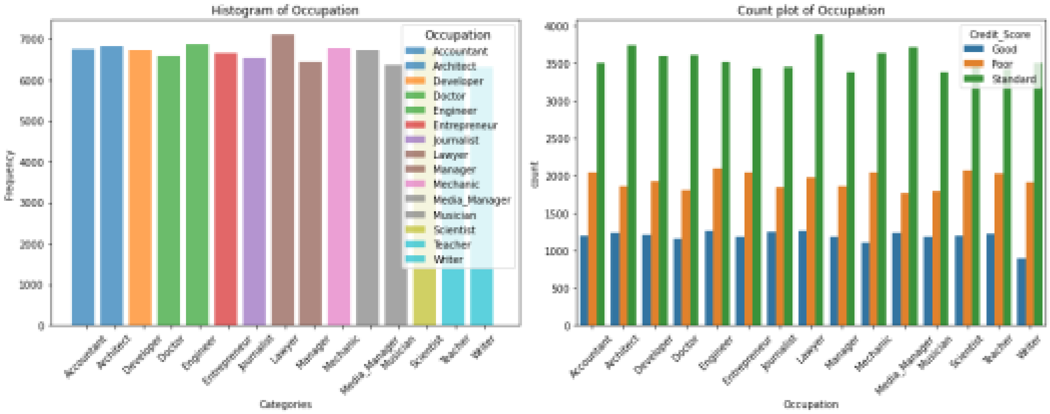


Figure 1 Histogram and Count plot for Occupation.

Same Distribution of credit Score across different occupation groups ➞ occupation isn't an effective feature No need to consider occupation in the model.

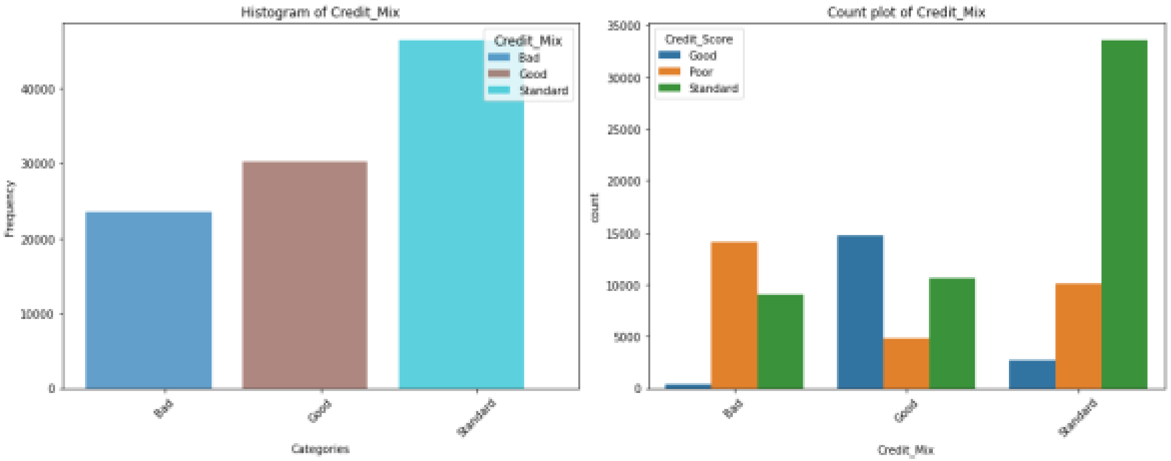


Figure 2 Histogram and Count plot for Credit Mix

Credit Mix has 3 Values Bad Good and Standard

- A customer having a bad credit mix is more likely to have a Bad credit score if not he is likely to have a Standard credit score, then a good credit score

- A customer having a good credit mix is more likely to have a good credit score if not he is likely to have a Standard credit score, then a Bad credit score

- A customer having a standard credit mix is more likely to have a Standard credit score if not he is likely to have a Poor credit score, then a good credit score

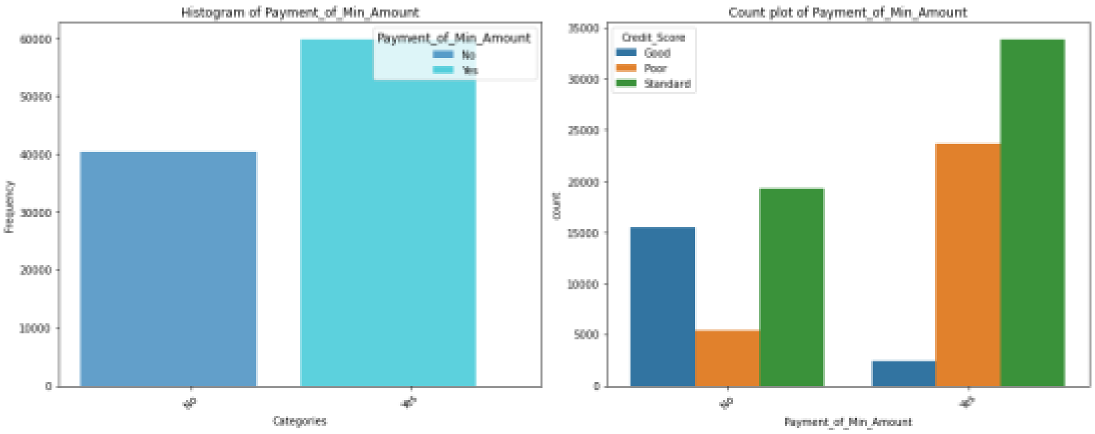


Figure 3 Histogram and Count plot for Payment of Min Amount

Most of the customers in the dataset have made the minimum payment on their financial obligation.

- A customer who has made only the minimum payment is more likely to have a Standard/Bad credit score if not he is likely to have a good credit score

- A customer who hasn't made only the minimum payment is more likely to have a Standard/Good credit score if not he is likely to have a Bad credit score

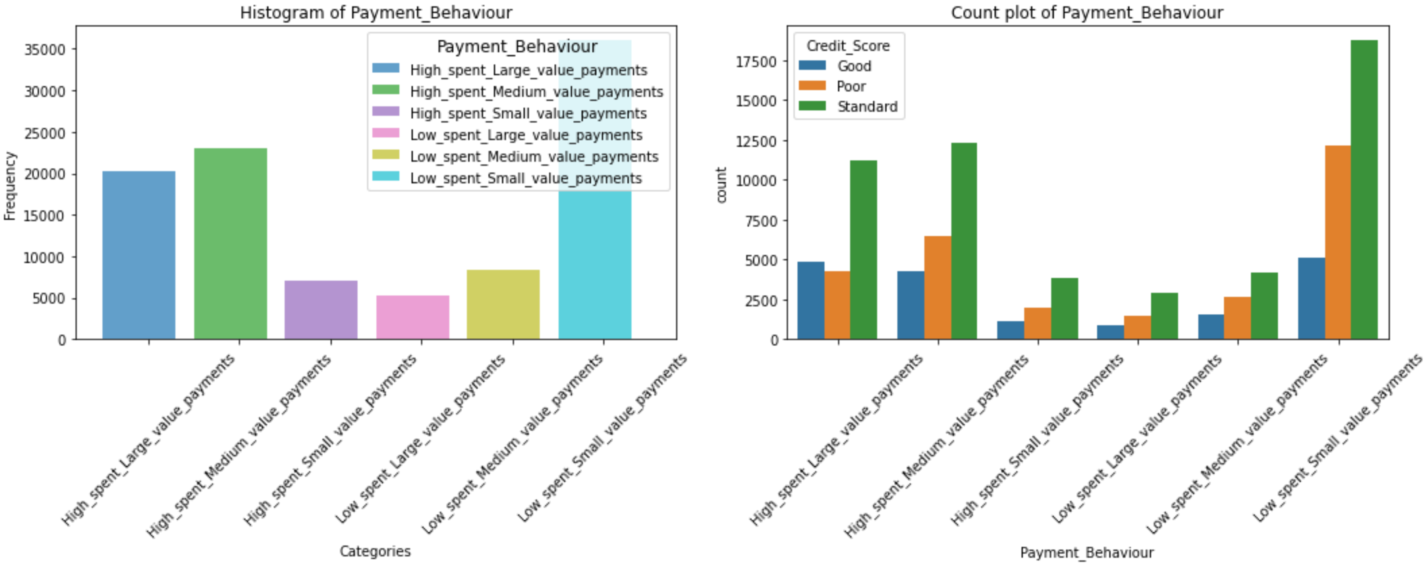


Figure 4 Histogram and Count plot for Payment Behavior

In the 6 Categories for the payment behavior Standard is the most dominant Class followed by the Poor except for High Spent Large Value Payments the good credit_score has a bit higher population than poor 😉

Numerical Features:

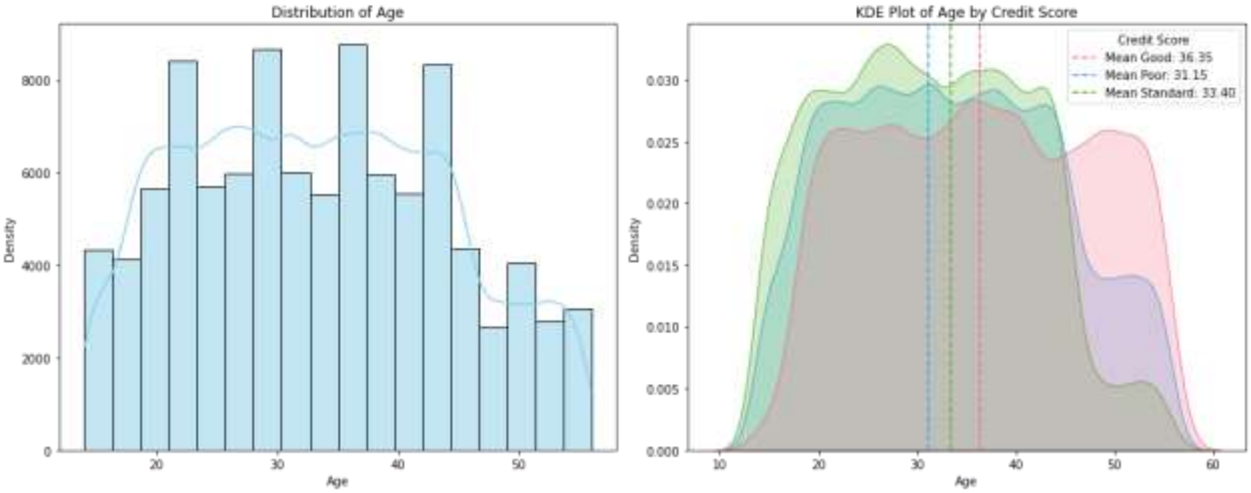


Figure 5 Distribution and KDE plot for Age.

Useless Feature to be dropped Later 😑

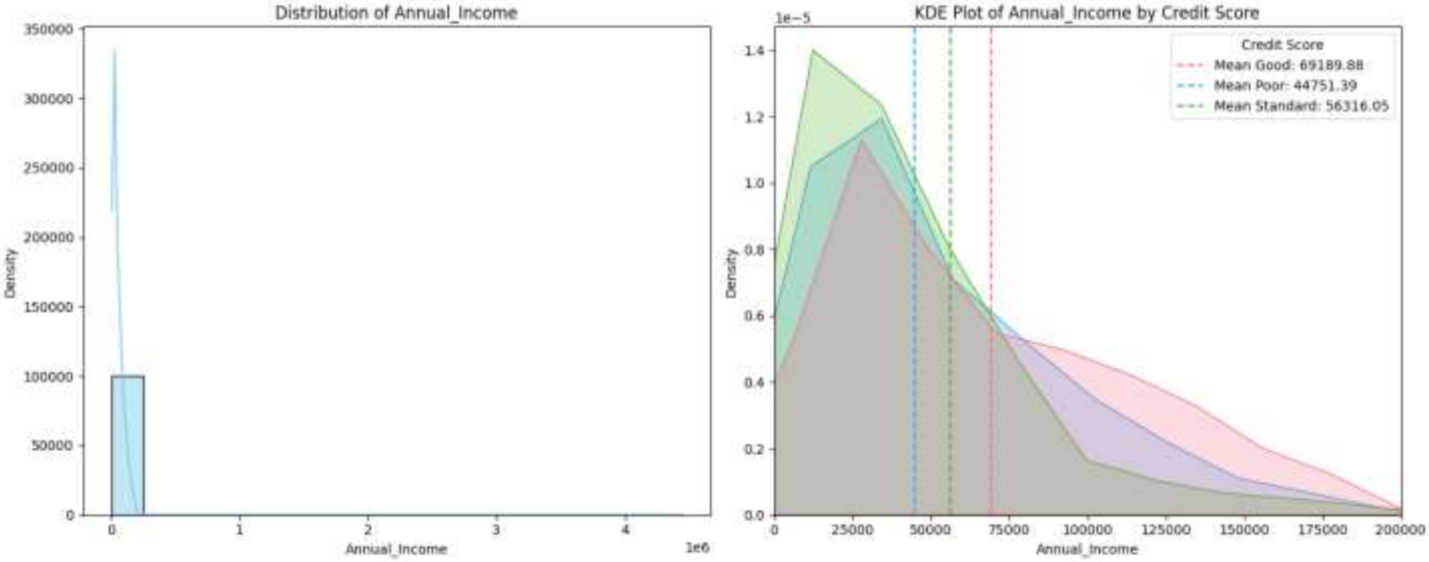


Figure 6 Distribution and KDE plot for Annual Income.

Overlapped Classes So this feature has no critical effect, starting from 75000 the customer is more probable to be have good credit score

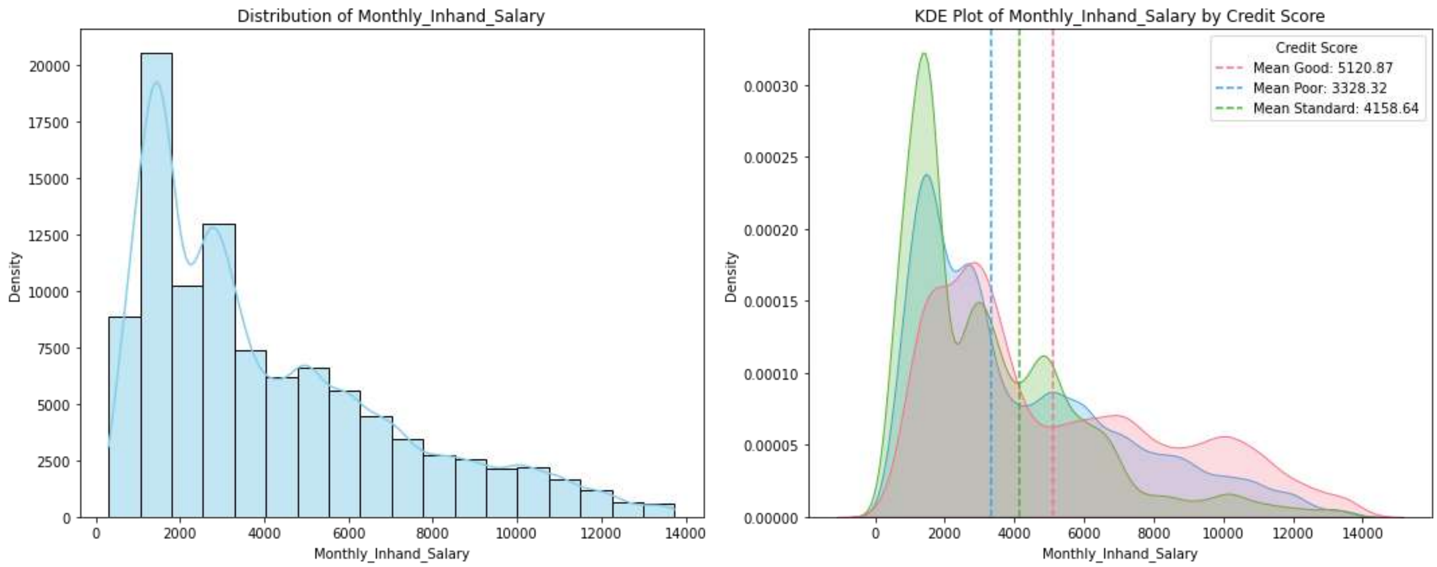


Figure 7 Distribution and KDE plot for Monthly In hand Salary

💭 Logical Hierarchy 😋

- High Monthly In hand Salary makes him most probably to have good credit score

- Moderate Monthly In hand Salary makes him most probably to have standard credit score

- Low Monthly In hand Salary makes him most probably to have poor credit score

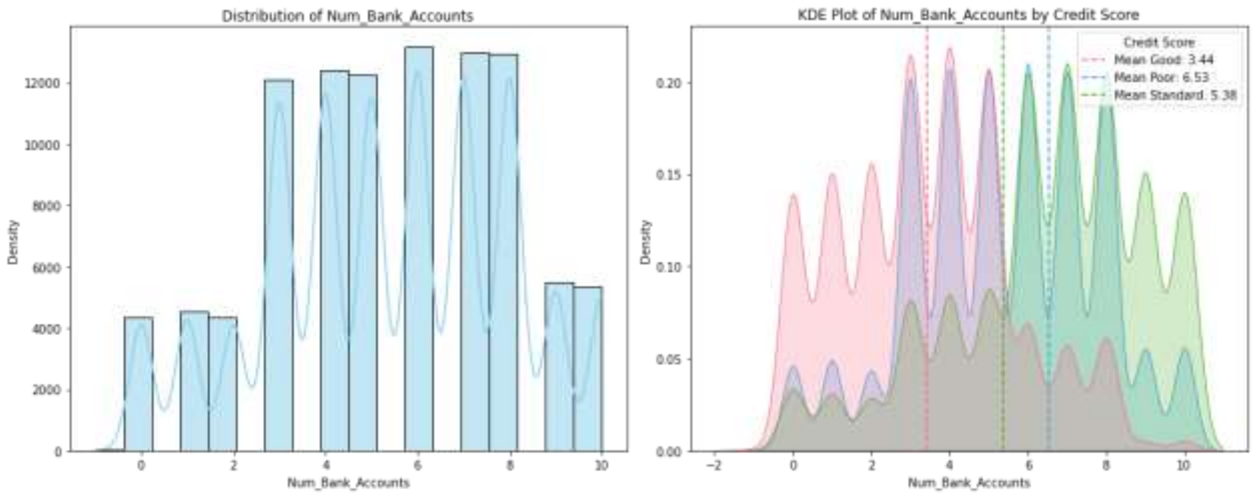


Figure 8 Distribution and KDE plot for No of Bank Accounts

💭 Logically 😋

- A person having a high number of bank accounts is more likely to have a good credit score

- A person having a moderate number of bank accounts is more likely to have a standard credit score

- A person having a low number of bank accounts is more likely to have a poor credit score

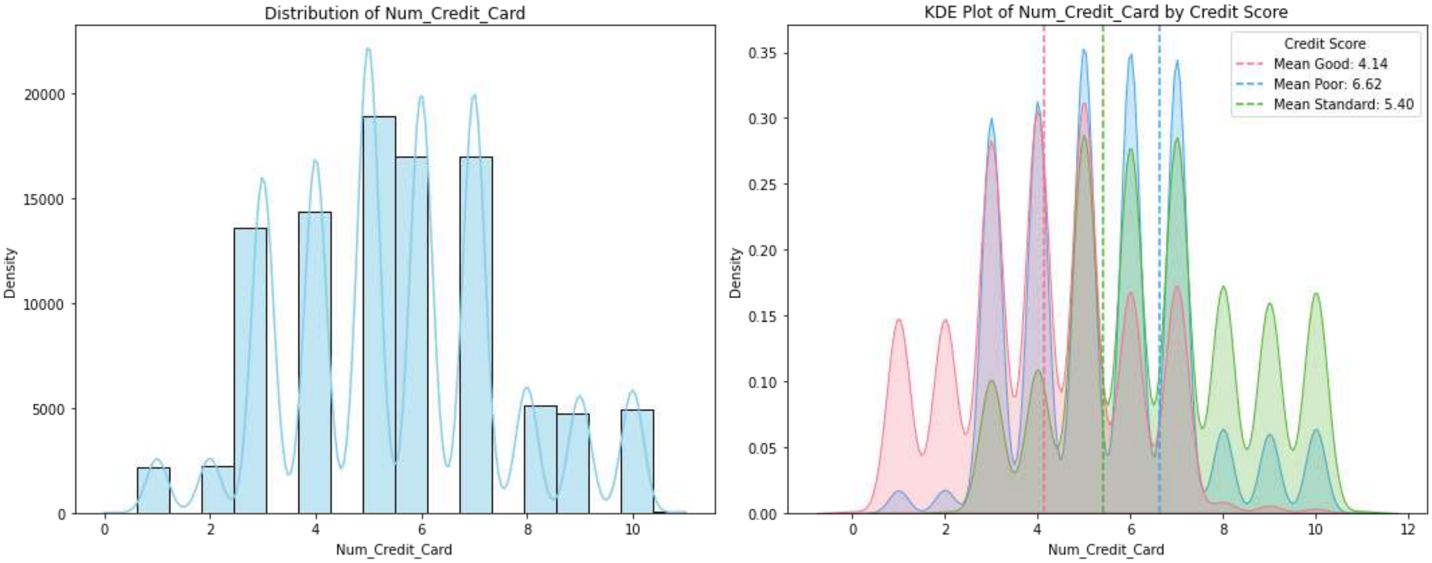


Figure 9 Distribution and KDE plot for No of Credit Cards

Same Insights as in no of bank accounts as they are correlated features 💡

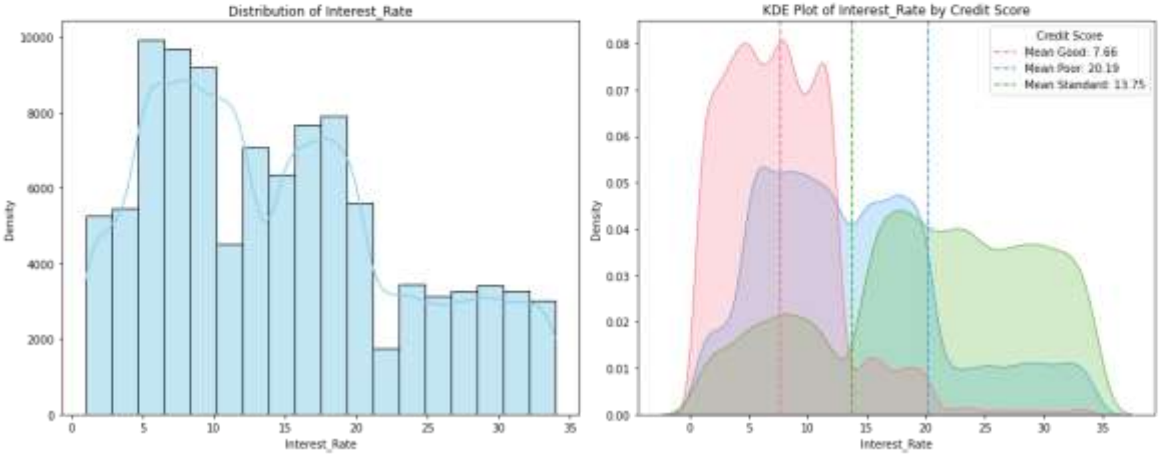


Figure 10 Distribution and KDE plot for Interest Rate

Same Insights as in num of bank accounts as they are correlated features 💡

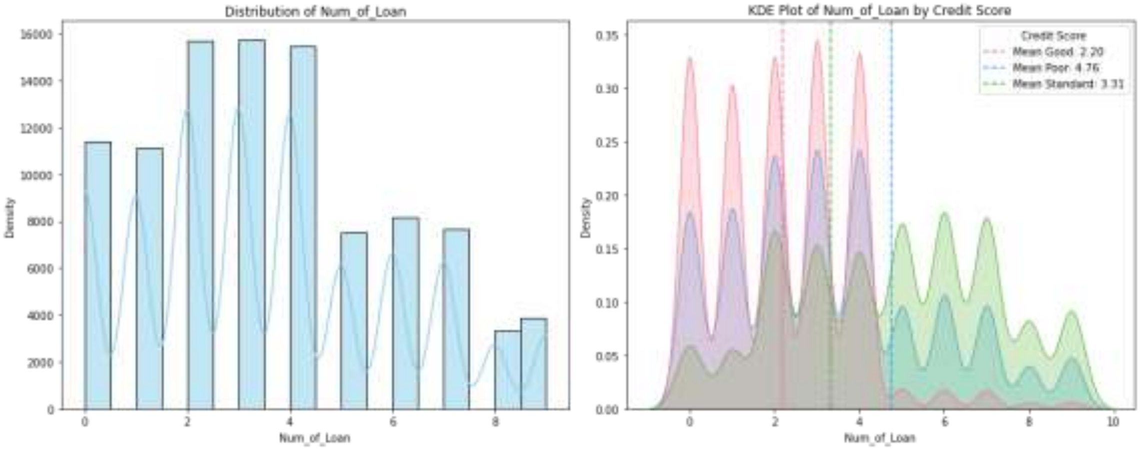


Figure 11 Distribution and KDE plot for No of Loans

Same Insights as in num of bank accounts as they are correlated features 💡

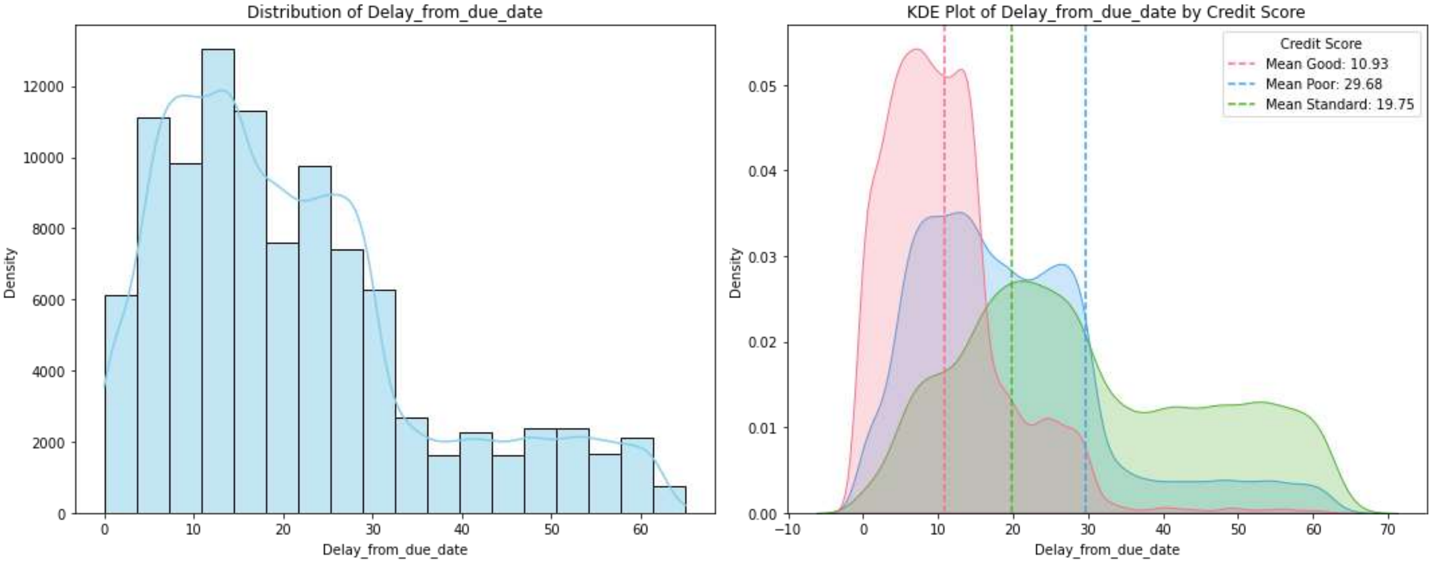


Figure 12 Distribution and KDE plot for Delay from Due Date

Same as num of bank accounts [Correlated Features] Good has higher delay from due data, this might be due to the type of his payments

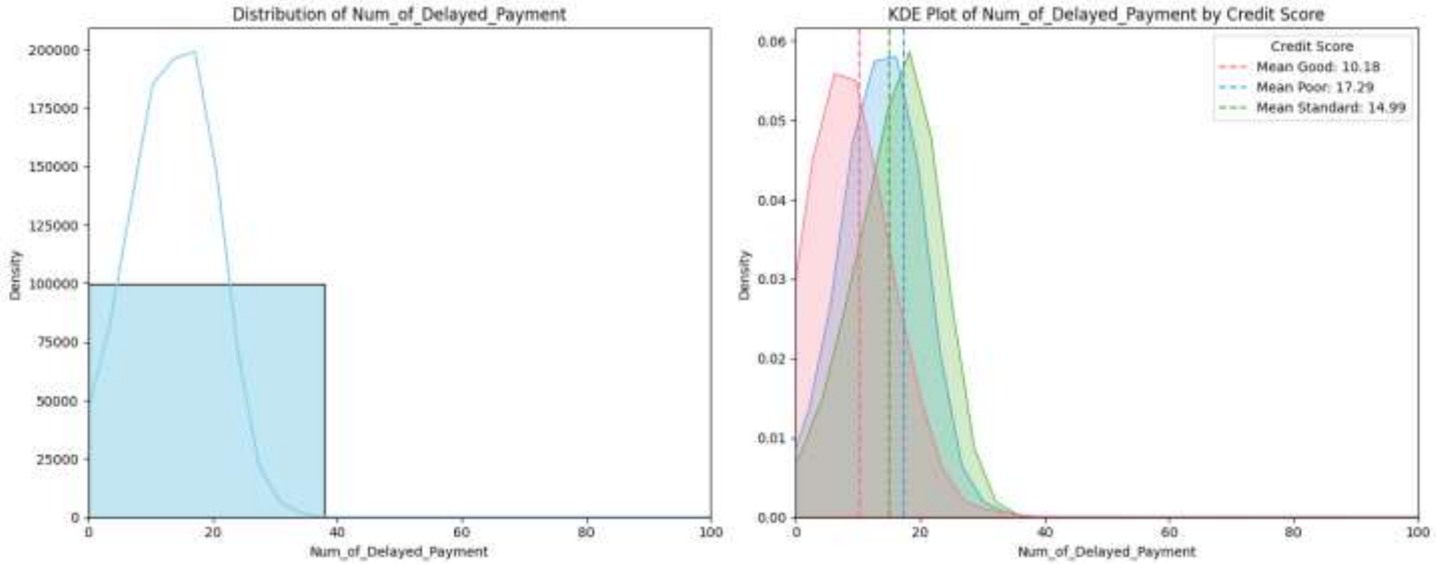


Figure 13 Distribution and KDE plot for Delayed Payments

🗨🗨 Although we may see the graph overlapped by zooming to the range 0-100 we see that low num of delayed payments for good credit customer while poor and standard are higher than good and they are near to each other

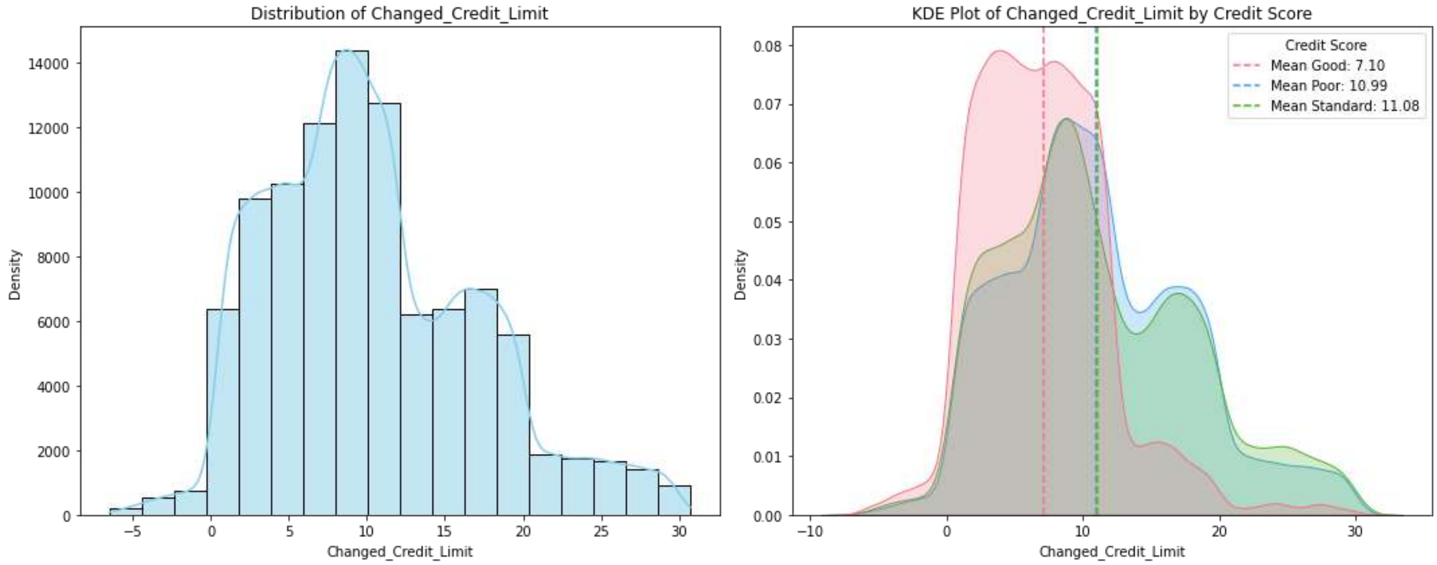


Figure 14 Distribution and KDE plot for changed credit limit

🗨🗨 Same Distribution for poor and standard which is higher than that for good this reflects that customer with good credit score doesn't change a lot his limit but good and poor need to change according to their precious usage :(

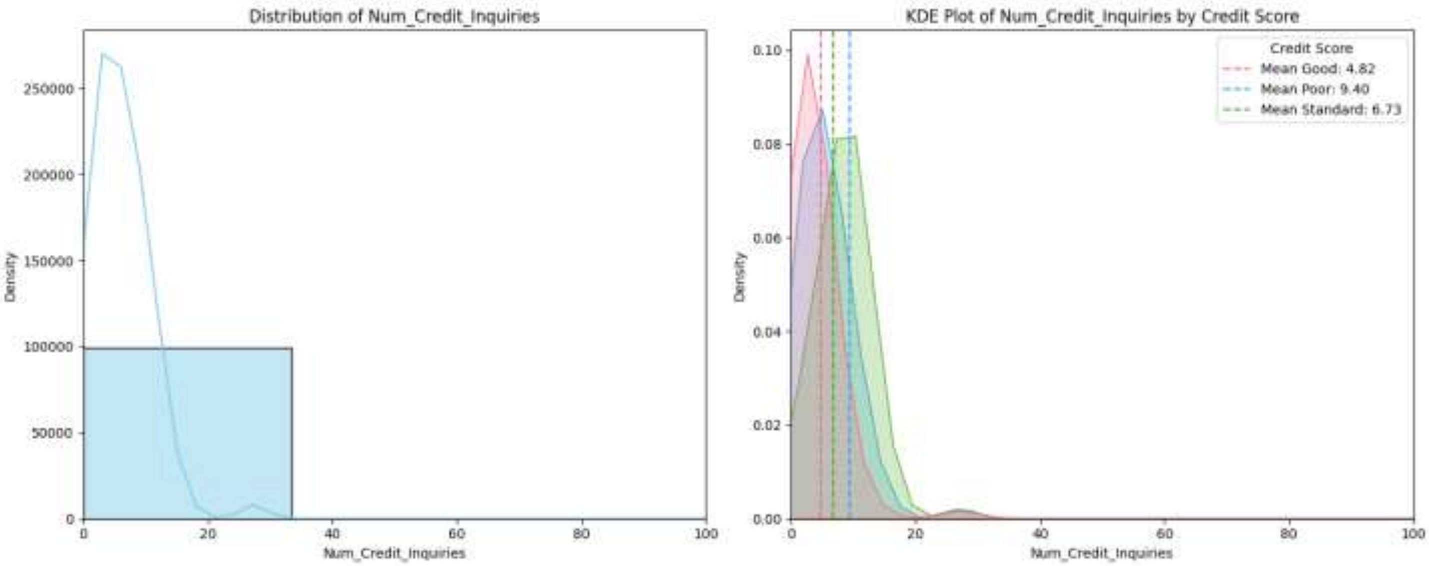


Figure 15 Distribution and KDE plot for Num of Credit Injuries

💭💭 higher mean value for poor credit_score because these type of customers don't have the cash money for they buying so they request for credit cards with nearest rate dos that they can buy their needs.

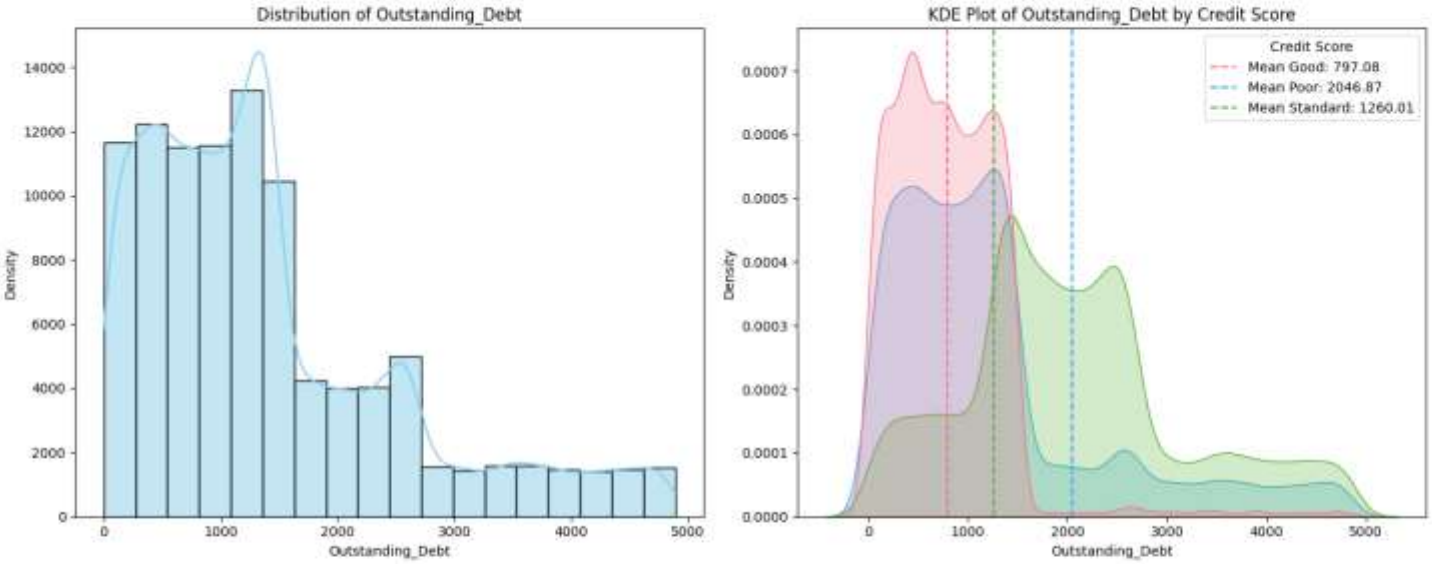


Figure 16 Distribution and KDE plot for Out Standing Debt

Good credit customer has outstanding debt more than 1800 :D

Features Correlation:

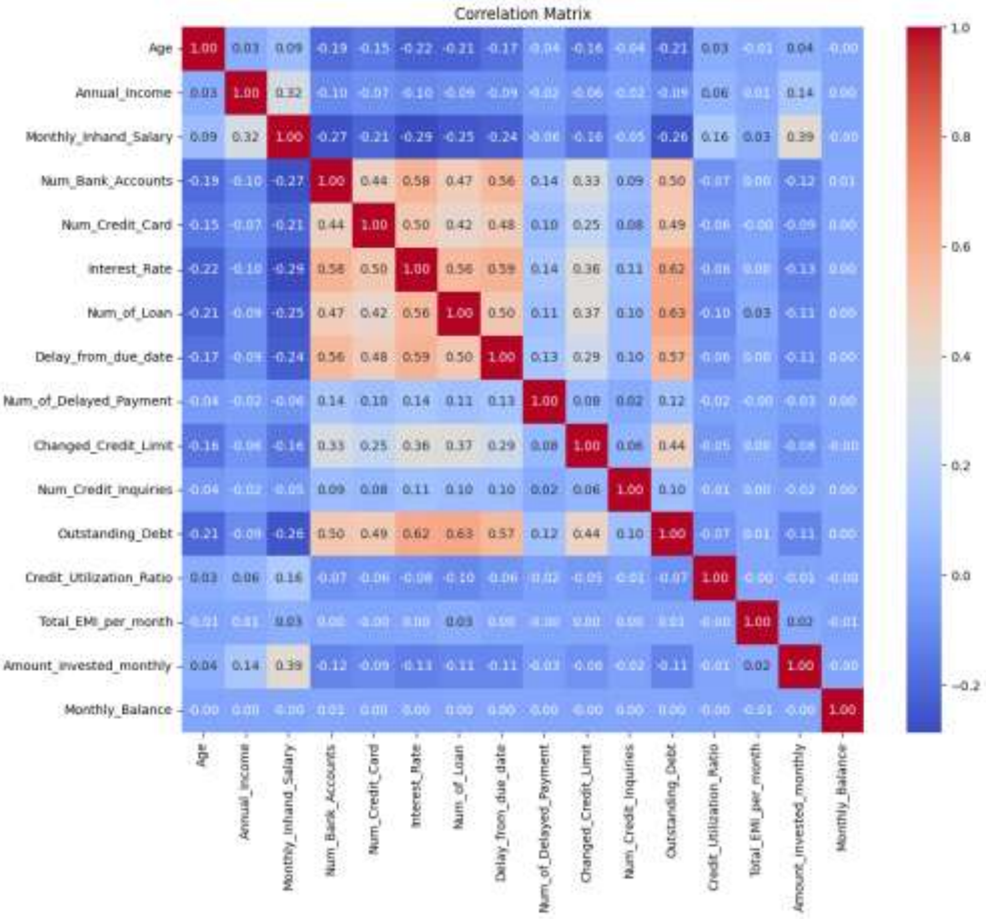


Figure 17 Correlation Between Features

From the graph it is clear that num_of_bank_accounts, num_of_credit_cards , interest_rate and delay_from_due_date are the most features correlated with each other than other features.

. Num of Banks accounts and num of credit cards are correlated with each other which is logic as no of accounts are more the customer will have more credit cards [0.44] 😉

. Num of Banks accounts and Interest Rate are correlated with each other [0.58]

. Num of Credit Cards and Interest Rate are correlated the more cards the person has the more outstanding balanced he will be outstanding, so the interest rate increases [0.50]

. Num of Banks accounts & Num of Credit Cards both are correlated with Delay from Due Date are as no of bank accounts/credit cards increases the outstanding balance increase so the customer is more likely to miss the due date [0.56] [0.48]

. Interest Rate and Delay from Due Date are correlated with each other as the delay from due date increases the bank needs to apply like a penalty so as a result the interest rate increases [0.59]

We have applied T-SNE on the 3 clusters of the data (good-standard-poor) to see clusters. [descrpitive_analysis.ipynb]

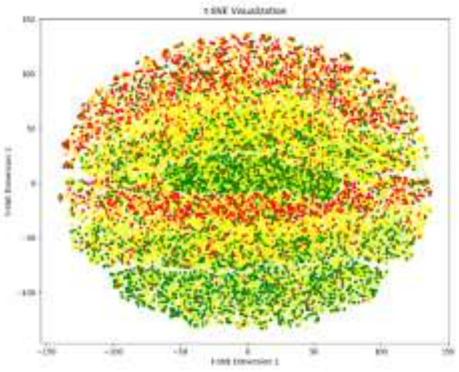


Figure 18 T-SNE for 3 Classes (With most of features)

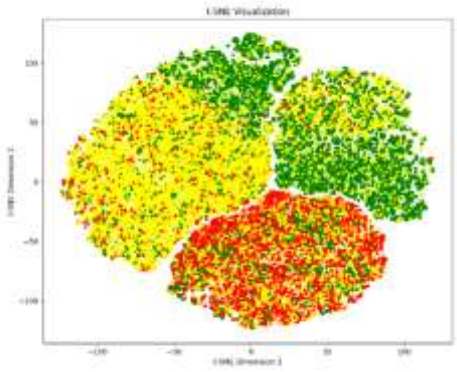


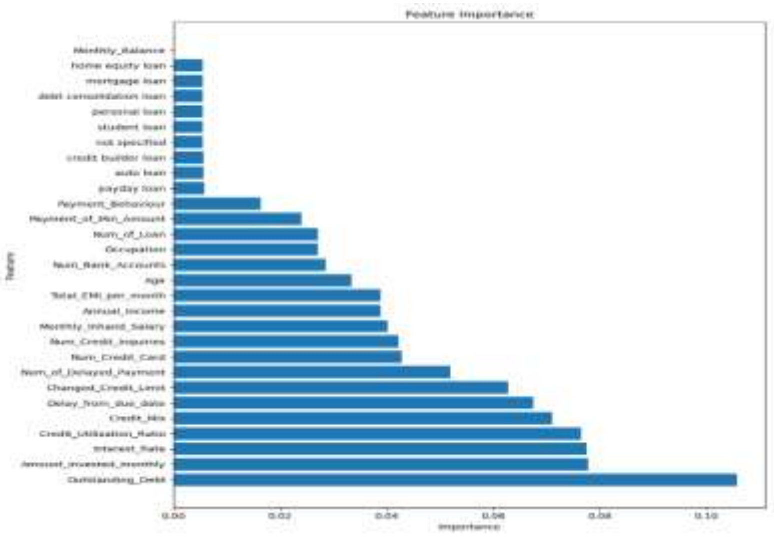
Figure 19 T-SNE for 3 Classes (After Dropping features of importance < 0.04)

Insights From Data:

- ✓ In Section Above the insights are written below each graph.
- ✓ No Correlation between Continuous Features [Insight from the correlation Matrix of between all features]

Business Insights

- ✓ Occupation & Age aren't and effective features for credit Score Classification [insight from EDA histogram]
- ✓ Num of banks accounts & credit cards are correlated together [Insight from EDA & Features correlation matrix]
- ✓ The customers of good credit score are mostly the ones with high delay from due date this might be due to the types of their payments which is thought to be high [Insight from EDA]
- ✓ The most important feature in the credit score decision is the outstanding debt, giving loan to a customer who has a many outstanding loans is risky 😃 [insight from important feature diagram of random forest]
- ✓ The Bank should know the following important features that affect the credit score classification.



Model/Classifier training:

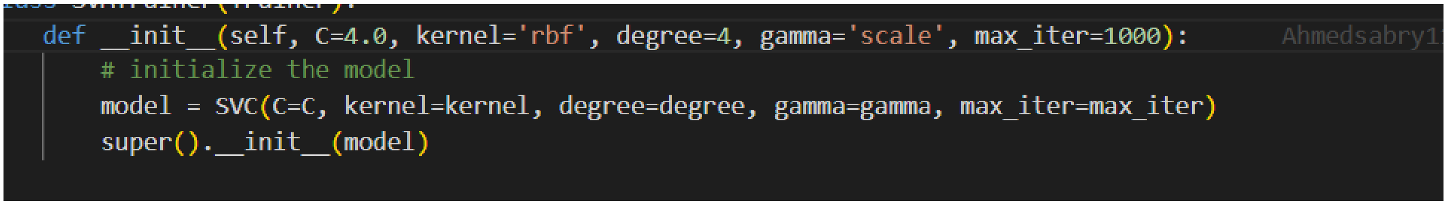
We have defined *Trainer* class which is common for training and evaluation for all predictive analysis models[trainer.py]

1. Splitting training data into train and test [NB: We haven't used the test subset provided with the data set because it is unlabeled]

Results and Evaluations:

SVM: svm.py

1. We used Postprocessed data.
2. Convert Categorial Data using Label Encoder (Label Encoder)
3. Standardization for Continuous Data (Standard Scaler)



```
class SVMTrainer(Trainer):
    def __init__(self, C=4.0, kernel='rbf', degree=4, gamma='scale', max_iter=1000):        Ahmedsabry1
        # initialize the model
        model = SVC(C=C, kernel=kernel, degree=degree, gamma=gamma, max_iter=max_iter)
        super().__init__(model)
```

Random Forest: random_forest.py

1. We used Postprocessed data.
2. Convert Categorial Data using Label Encoder (Label Encoder)
3. Standardization for Continuous Data (Standard Scaler)
4. From the trained Model we could plot the features Importance

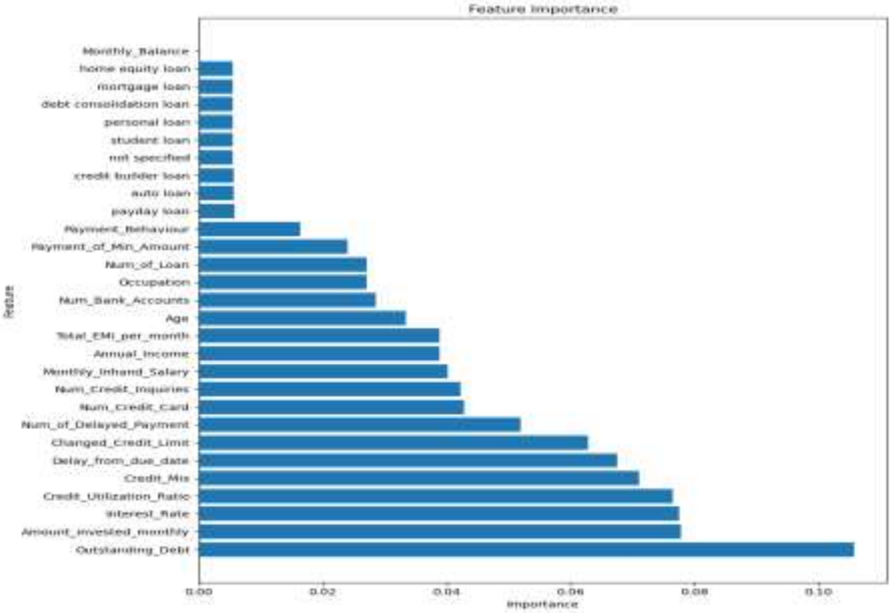
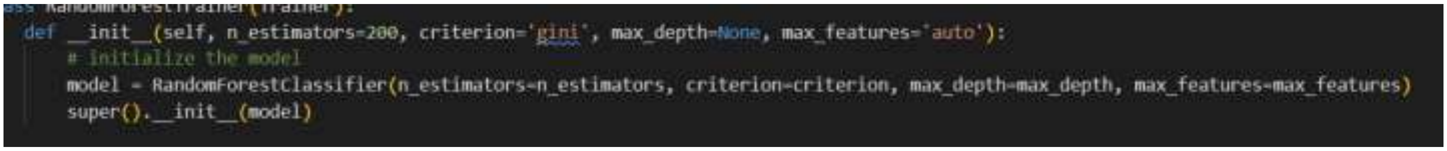


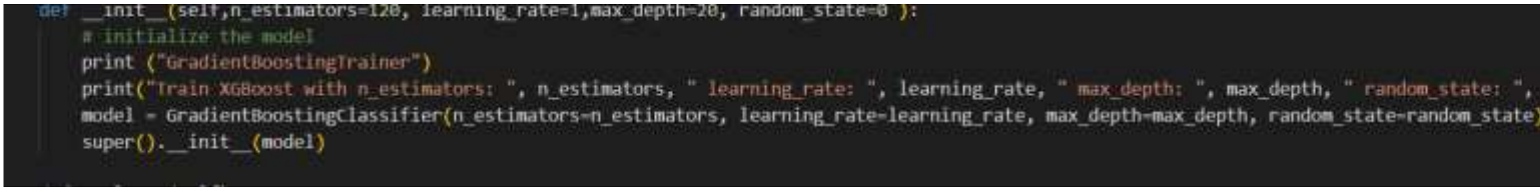
Figure 20 Feature Importance from Random Forest Classifier

The most important feature for credit_score prediction is Outstanding_debt (Represents the remaining debt to be paid in USD) which is logic 😎



```
ass RandomForestTrainer(Trainer):
    def __init__(self, n_estimators=200, criterion='gini', max_depth=None, max_features='auto'):
        # initialize the model
        model = RandomForestClassifier(n_estimators=n_estimators, criterion=criterion, max_depth=max_depth, max_features=max_features)
        super().__init__(model)
```

XgBoost: xgboost.py



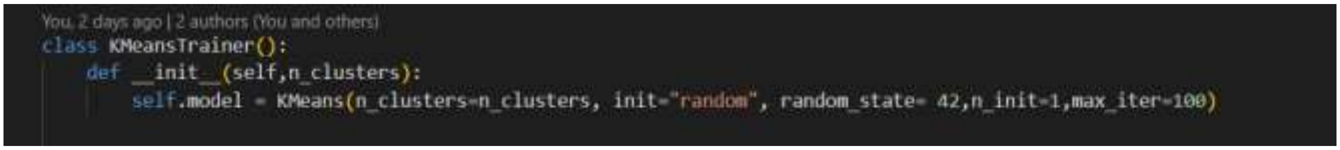
```
    def __init__(self,n_estimators=120, learning_rate=1,max_depth=20, random_state=0 ):
        # initialize the model
        print ("GradientBoostingTrainer")
        print("Train XGBoost with n_estimators: ", n_estimators, " learning_rate: ", learning_rate, " max_depth: ", max_depth, " random_state: ",
        model = GradientBoostingClassifier(n_estimators=n_estimators, learning_rate=learning_rate, max_depth=max_depth, random_state=random_state
        super().__init__(model)
```

Results:

Metric	SVM	Random Forest	XgBoost
Train			
Accuracy	0.787	1.0	1.0
Recall	0.758 0.777 0.802	1.0 1.0 1.0	1.0 1.0 1.0
Precision	0.679 0.797 0.823	1.0 1.0 1.0	1.0 1.0 1.0
F1-score	0.716 0.789 0.812	1.0 1.0 1.0	1.0 1.0 1.0
Test			
Accuracy	0.7546	0.804	0.791
Recall	0.710 0.743 0.777	0.758 0.835 0.802	0.738 0.803 0.803
Precision	0.641 0.772 0.790	0.775 0.786 0.825	0.767 0.783 0.803
F1-score	0.674 0.757 0.783	0.766 0.810 0.813	0.752 0.793 0.803

Kmeans: kmeans.py

We have used Sklearn built-in Kmeans Clustering for further segmentation of the customers.



```
You, 2 days ago | 2 authors (You and others)
class KMeansTrainer():
    def __init__(self,n_clusters):
        self.model = KMeans(n_clusters=n_clusters, init="random", random_state= 42,n_init=1,max_iter=100)
```

We have seen in Figures (1&2) that segmenting the customers into 3 classes according to the credit score isn't enough, there is a lot of overlapping between classes. So, we tried out different no of classes such as 5.

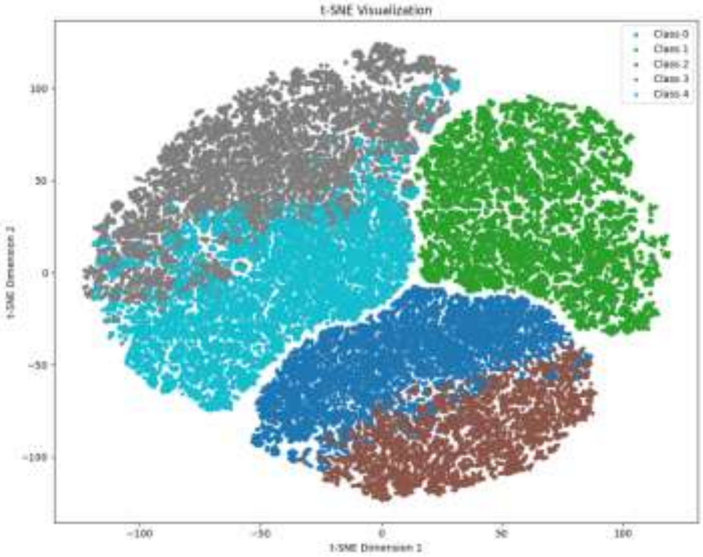


Figure 21 Segmenting into 5 Clusters

Kmeans (Map-Reduce): descriptive_analysis.ipynb

Mapper:

Computes the closest Centroid for each point.

Point(P) → (P, Ci), i is the index of the closest centroid.

Combiner:

Combine the points that belong to the same centroid (Partial Sum) [Per Machine]

(P1,C1),(P3,C1) → (C1, P1+P3), i is the index of the closest centroid.

Reducer:

Compute the new centroids by dividing the partial summation per centroid by the no of points in the newly computed centroid!

(C1, P1+P3) → (C1, (P1+P3)/2), i is the index of the closest centroid.

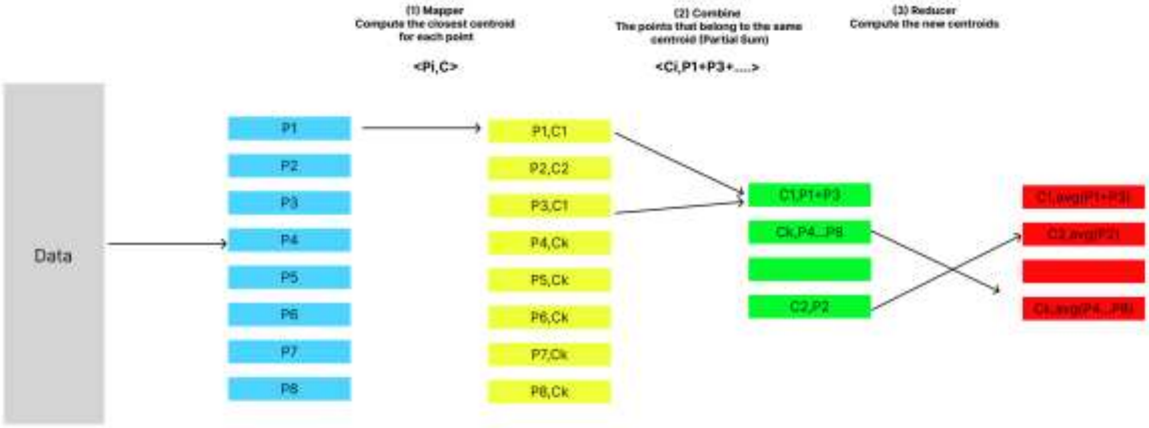


Figure 22 K means Clustering Map-Reduce

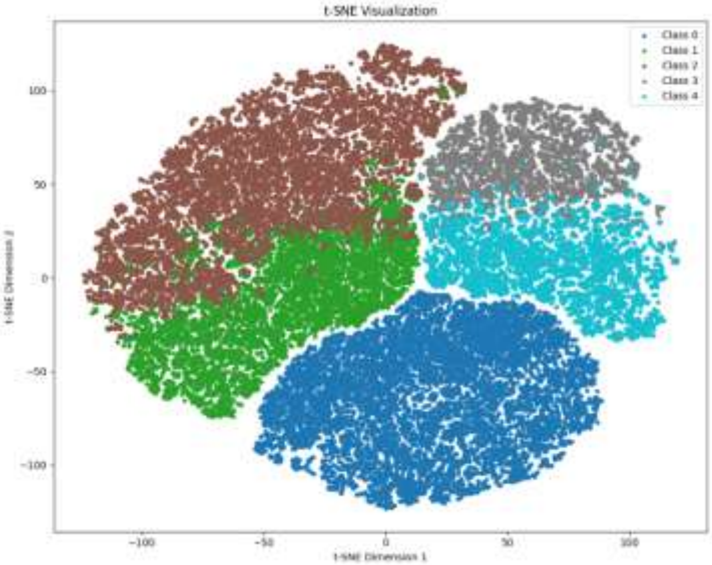


Figure 23 Kmeans Clustering Map-Reduced 5 Clusters

The same scatter plot for the TSNE Reduction is obtained from Sklearn 😎

- Converged after 25 iterations.

Kmeans Bench Marking: (5 Classes Clustering):

POC	Sklearn	Pyspark Map-Reduce From Scratch
Silhouette Score	0.25215559235815665 (Cases of all features used we got 0.09 ☹)	0.25276471972112385 (Same as Sklearn ☺)
Training Time	0.6300568580627441	124.21908068656921

The Sklearn version is very fast compared to the map-reduced one this is due to the parallelization implemented by the built-in version by Sklearn.

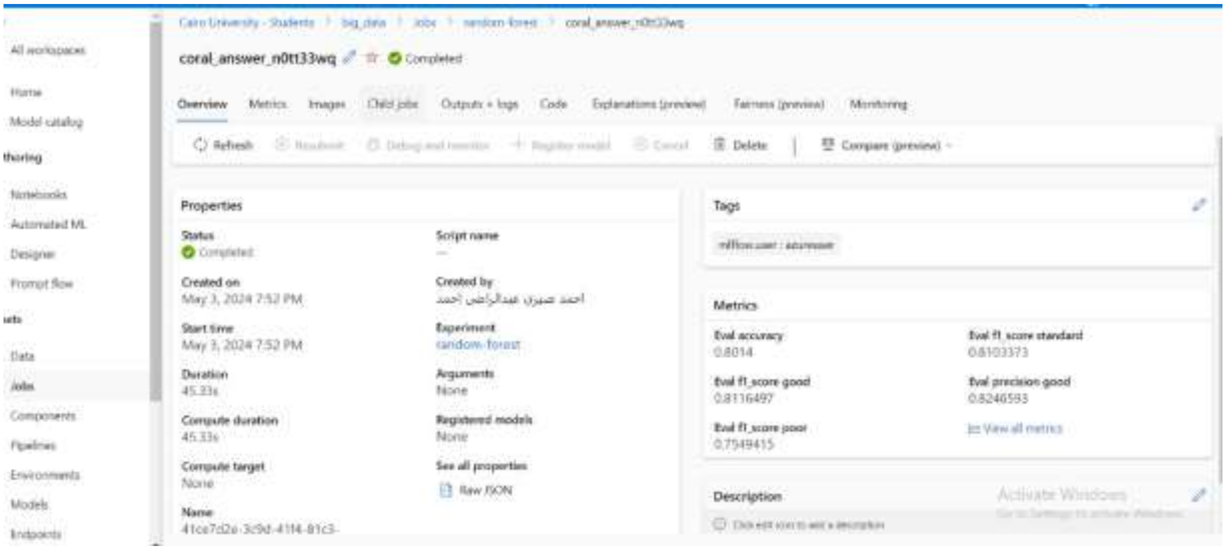
Bench Marking Problem (Others on Kaggle):

Accuracies: (Kaggle Notebooks no leader board)

- 77%
- 80.5%

Extra Work (Bonus):

- Training/Testing of Random Forest on Azure ML Studio



coral_answer_n0tt33wq Completed

Properties

Status: Completed

Created on: May 3, 2024 7:52 PM

Start time: May 3, 2024 7:52 PM

Duration: 45.33s

Compute duration: 45.33s

Compute target: None

Name: 41ce7d2e-3c9d-4114-81c3-

Script name: --

Created by: احمد صبري عبدالراضي احمد

Experiment: random-forest

Arguments: None

Registered models: None

See all properties: Raw JSON

Tags

mlflow.user : azureuser

Metrics

Eval accuracy: 0.8014

Eval f1_score standard: 0.8103373

Eval f1_score good: 0.8116497

Eval precision good: 0.8246593

Eval f1_score poor: 0.7549415

View all metrics

Description

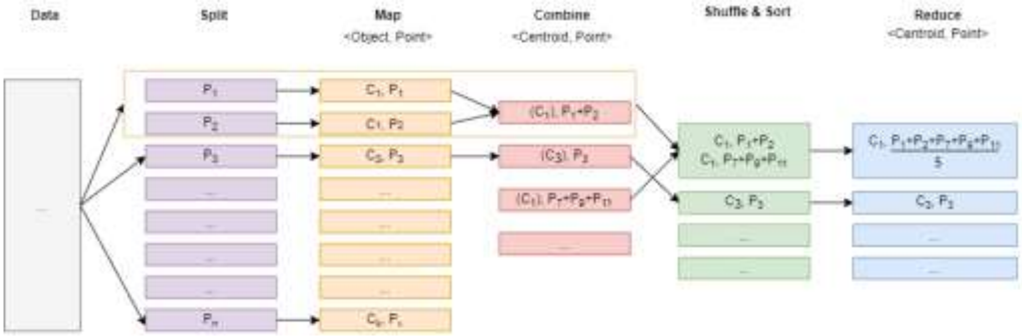
Click edit icon to add a description

Unsuccessful Trials:

- Using All features (excluding identifiers like SSN, Name ,..) in the models training → Too long time to train + we have faced curse of dimensionality (overfitting)

Future Works:

- Trying other Models (KNN, Decision Trees)
- Apply Map Reduce on Multi Machine


Data
Split
Map
<Object, Point>
Combine
<Centroid, Point>
Shuffle & Sort
Reduce
<Centroid, Point>
P_1
P_2
P_3
P_n
C_1, P_1
C_1, P_2
C_3, P_3
C_9, P_1
$(C_1), P_1+P_2$
$(C_3), P_3$
$(C_1), P_7+P_9+P_{11}$
C_1, P_1+P_2
$C_1, P_7+P_9+P_{11}$
C_3, P_3
$C_1, \frac{P_1+P_2+P_7+P_9+P_{11}}{5}$
C_3, P_3