AML PROJECT DELIVERABLE 2

CREDIT SCORE CLASSIFICATION

GOKUL NATESAN, MADHULIKA BALAKUMAR, SHUMAIL SAJJAD, YASH DANGE

CONTENT

1. INITIAL DATA EXPLORATION

At a glance:

By simply taking a look at some basic summary information about the data, we can identify various issues that need to be addressed about the data, before it is ready to be trained using a model.

NUMBER OF ROWS: 100,000

NUMBER OF COLUMNS: 28

TARGET COLUMN: CREDIT SCORE

Main Issues:

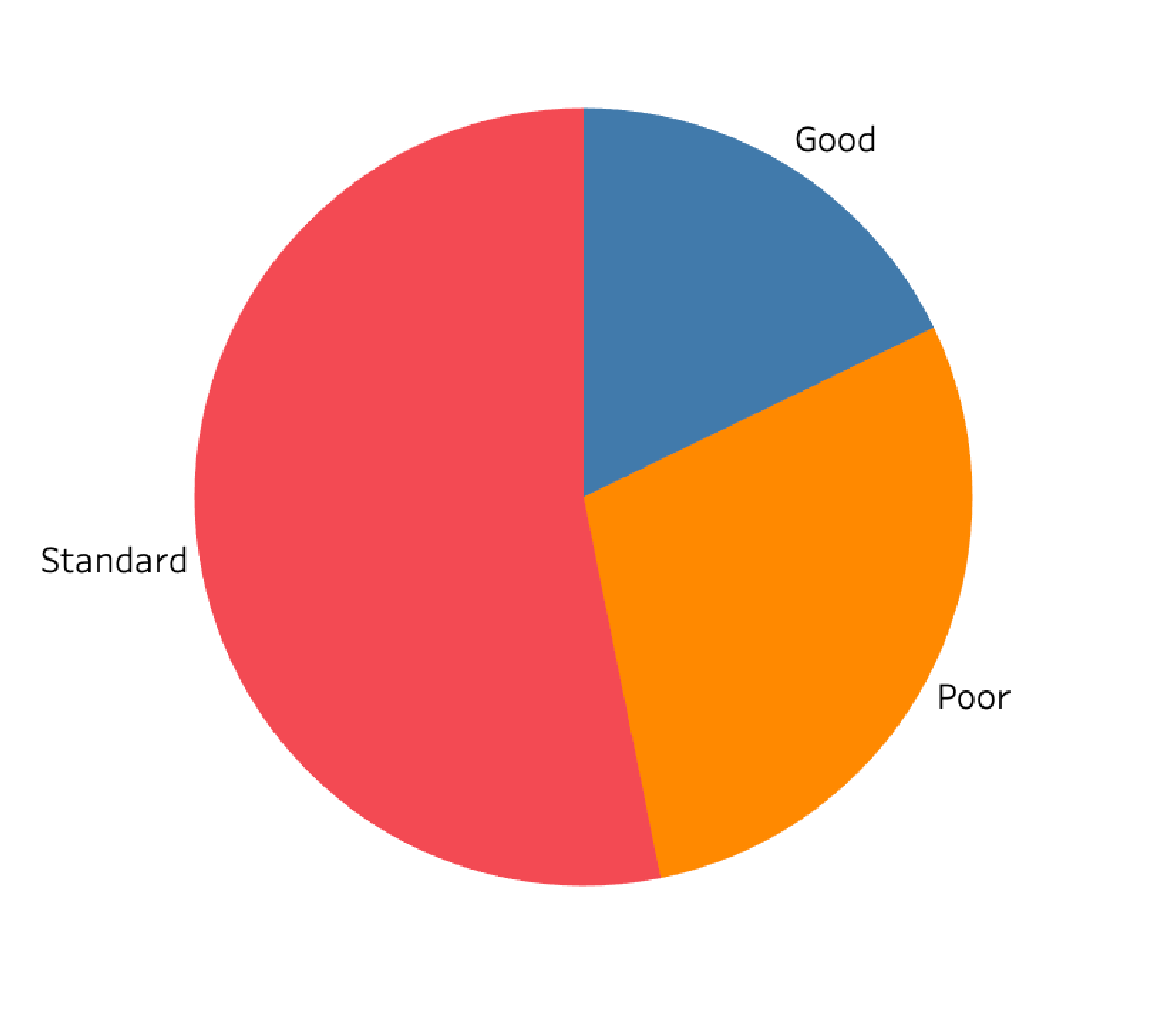
Missing Values,Incorrect Datatypes, Outliers, Frequent occurance of '_'

```
credit_df.info()

<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
```

CLASS BALANCE: CREDIT SCORE

As shown in the figure, there are three possible classes for the target Column, viz. Standard, Poor, and Good. It can be seen that a majority of the class values are Standard, and Good or Poor labels are comparatively less. Since there are relatively good number of representations of each class interval, we can say the dataset is balanced.

Note: Since we have three classes in the target column, we will be doing either of 'One vs One' or 'One vs Many' approach while model training.



2. CLEANING AND SAMPLING

Age Column

Upon a brief observation of the data, we notice that there are misplaced underscores '_' occasionally, the datatype is 'object' and some absurdly high values of age (5000) are present. Hence, we will fix this by

1. Eliminating the underscore
2. Converting the datatype to integer
3. Replacing 'abnormal' age values (here assumed to be under 10 and over 70) by corresponding age values of other columns with same customer ID and a 'normal' age value.

Before

Customer_ID	Month	Name	Age
CUS_0xd40	January	Aaron Maashoh	23
CUS_0xd40	February	Aaron Maashoh	23
CUS_0xd40	March	Aaron Maashoh	-500
CUS_0xd40	April	Aaron Maashoh	23
CUS_0xd40	May	Aaron Maashoh	23
CUS_0xd40	June	Aaron Maashoh	23
CUS_0xd40	July	Aaron Maashoh	23
CUS_0xd40	August		23

After

```
credit_df[credit_df['Customer_ID'] == 'CUS_0xd40']['Age']
```

```
0    23
1    23
2    23
3    23
4    23
5    23
6    23
7    23
Name: Age, dtype: int64
```

OCCUPATION COLUMN

Similar to previous techniques, we can simply replace occupation of '_______' with the correct occupation of the applicant by looking up the corresponding customer_id

```
: credit_df[credit_df['Customer_ID'] == 'CUS_0x4d43']['Occupation']

: 200    Entrepreneur
  201         _______
  202         _______
  203    Entrepreneur
  204    Entrepreneur
  205    Entrepreneur
  206    Entrepreneur
  207    Entrepreneur
  Name: Occupation, dtype: object
```

Before

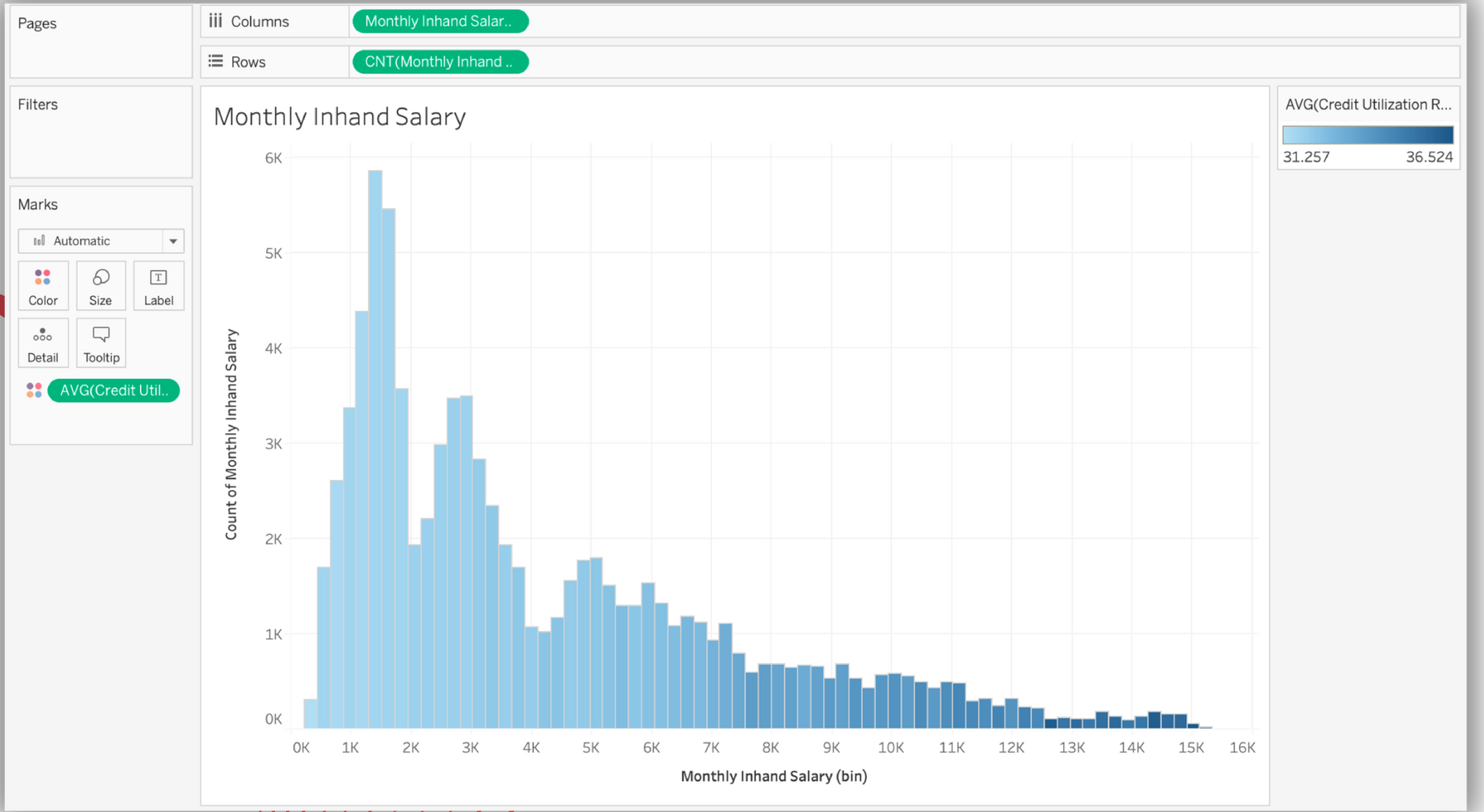
```
: credit_df[credit_df['Customer_ID'] == 'CUS_0x4d43']['Occupation']

: 200    Entrepreneur
  201    Entrepreneur
  202    Entrepreneur
  203    Entrepreneur
  204    Entrepreneur
  205    Entrepreneur
  206    Entrepreneur
  207    Entrepreneur
  Name: Occupation, dtype: object
```

After

Important thing to note here is that: even though all seemed fine about the occupation column in the initial data.info part (no missing values, datatype was categorical as expected), there were certain issues with the data. Unlike the Name column which couldn’t have any influence on our target column, the occupation might be an important feature and thereby needs to be cleaned properly. This goes to show that data cleaning should be done carefully and scrupulously.

3. INSIGHTS FROM DATA EXPLORATION

In this section, we will try to understand dependance or general trend of individual columns and the target column, and also check how these columns are correlated with each other.



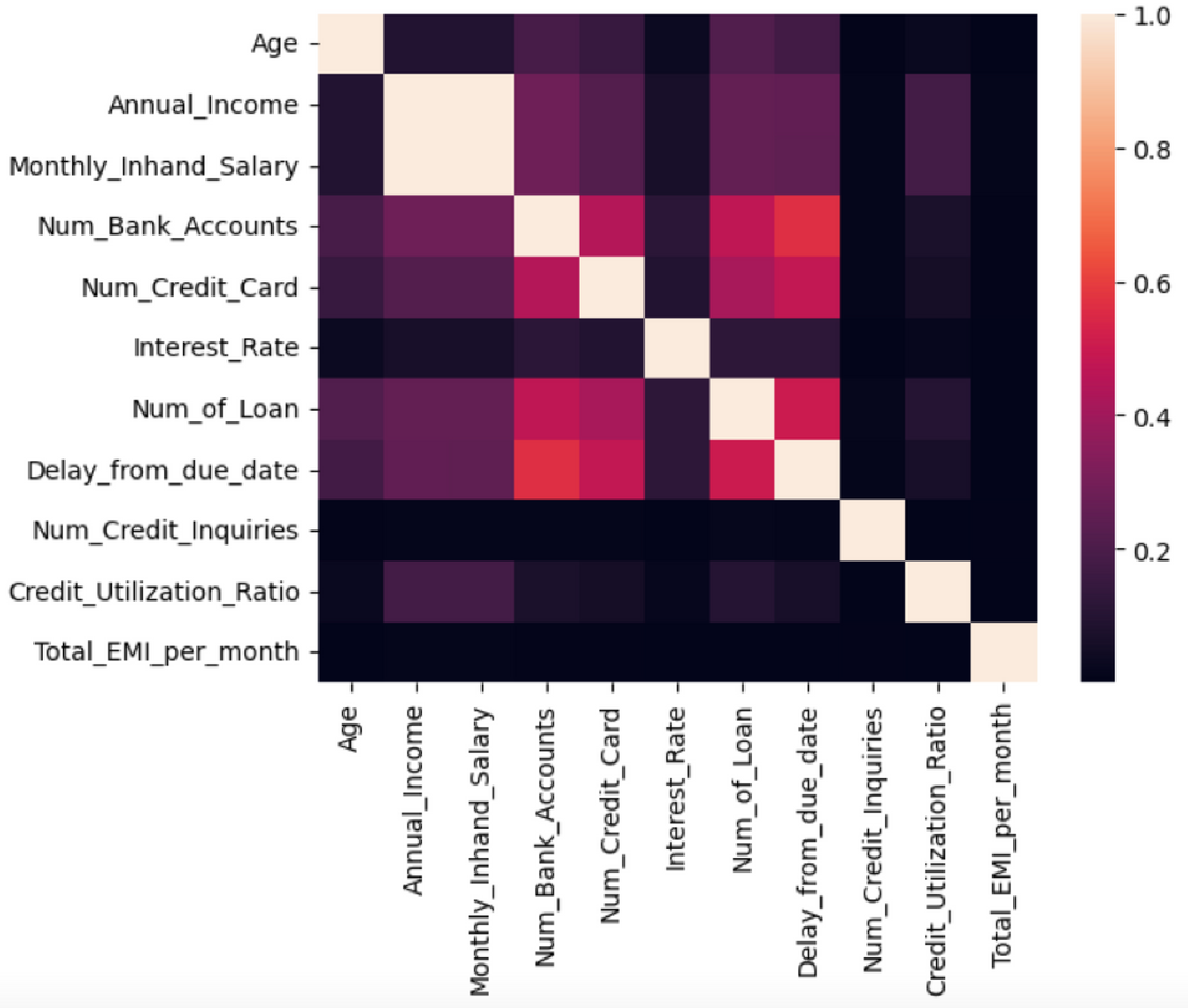
In the first graph, we plotted the frequency plot of 'monthly in hand' salaries of the people in the dataset. We have added the average credit utilisation ratio (in %) as a colour attribute. We can see that the data is skewed towards the left, suggesting that most people make about ~10K a month or less. People with higher income also seem to have a higher utilisation ratio of their credit line.

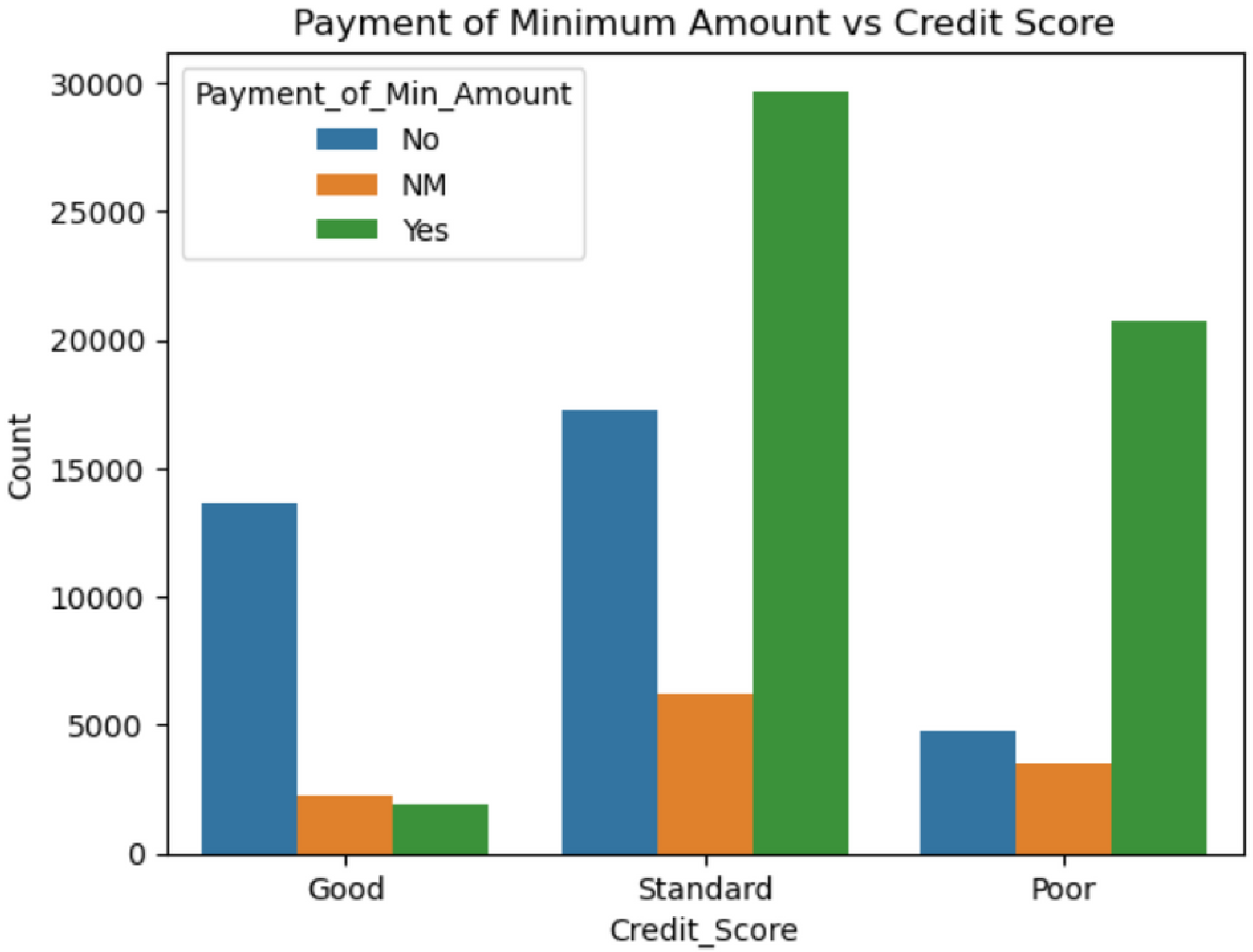
- In this correlation plot, we can see how the independent variables are associated with each other.

- As expected, Annual Income and Monthly inhand salary are almost perfectly positively correlated.

- Credit Utilization Ratio is also stronly correlated with the income value.

- Number of loans taken, Number of credit cards and number of bank accounts are also correlated, which is no surprise since having multiple loans at the same bank is unlikely.





Here, we have seen the distribution of people who only paid the minimum amount due in each class of the target column. As seen, in the ‘Good’ class, majority of people did not pay just the minimum amount due, but the full amount. In the poor class, this number is significantly higher and most people only paid the minimum amount. For standard, again, almost twice as many people only paid the minimum amount than the ones who paid potentially more than that. This goes to show that it is preferred to pay the entire amount for a good score rating, but it is not the only matric that matters.

4. MACHINE LEARNING TECHNIQUES

Since our dataset had multiple entries for each customer ID and almost the same demographic and income information across entries for customers, we used a custom splitter to group data by ‘Customer_ID’ and did a stratified split based on the target class.

We have applied a standard scaler to the dataset as models like Logistic Regression and SVM require this preprocessing step

Multiclass One-vs-One Logistic Regression model with default parameters:

```
Model - One-vs-One Logistic Regression
Accuracy: 0.6346
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.48      0.54      5906
           1       0.66      0.73      0.69     10482
           2       0.56      0.62      0.59      3612

    accuracy                           0.63     20000
   macro avg       0.62      0.61      0.61     20000
weighted avg       0.63      0.63      0.63     20000
```

Multiclass One-vs-One SVM Primal model:

Model - One-vs-One Linear SVM
Classification Report SVM Test:

	precision	recall	f1-score	support
0	0.65	0.45	0.53	5804
1	0.65	0.76	0.70	10603
2	0.52	0.51	0.51	3593
accuracy			0.63	20000
macro avg	0.61	0.57	0.58	20000
weighted avg	0.63	0.63	0.62	20000

Multiclass One-vs-One SVM Random Forest model:

Model - One-vs-One Random Forest
Accuracy: 0.70375
Classification Report:

	precision	recall	f1-score	support
0	0.73	0.66	0.69	5804
1	0.74	0.75	0.75	10603
2	0.57	0.63	0.60	3593
accuracy			0.70	20000
macro avg	0.68	0.68	0.68	20000
weighted avg	0.71	0.70	0.70	20000