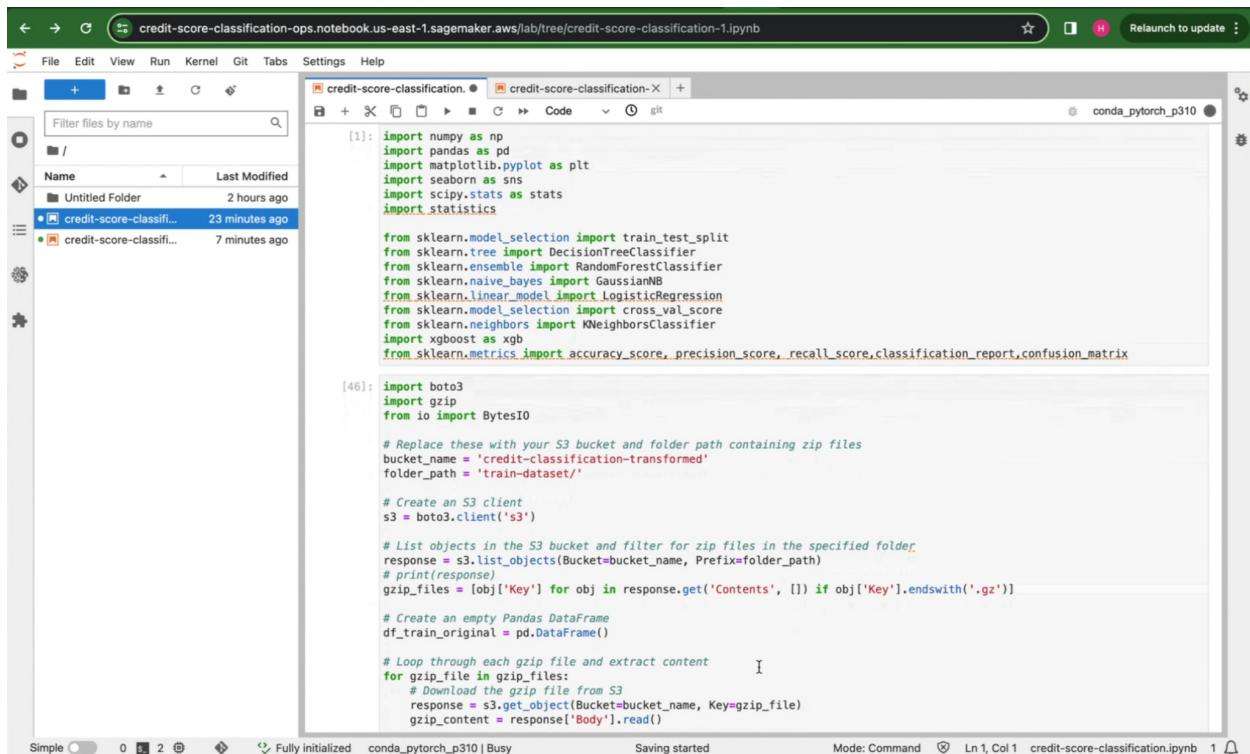


Fall 2023 ITCS 6190 Project Review Deliverable 3

Group 5

Machine Learning Framework

We used JupyterLab in SageMaker to work on our project. We used the scikit-learn Python library and tested different algorithms like DecisionTreeClassifier, RandomForestClassifier, KNeighborsClassifier, GaussianNB, and XGBClassifier for modeling, evaluating, and predicting customer credit scores. This helped us understand how well each algorithm performed and make informed decisions for our analysis.



The screenshot shows the JupyterLab interface running in a web browser. The title bar indicates the notebook is located at `credit-score-classification-ops.notebook.us-east-1.sagemaker.aws/lab/tree/credit-score-classification-1.ipynb`. The top navigation bar includes File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. A sidebar on the left shows a file tree with two files: `Untitled Folder` (modified 2 hours ago) and `credit-score-classification-1.ipynb` (modified 23 minutes ago). The main area is a code editor with tab titles `credit-score-classification-1.ipynb` and `credit-score-classification-1.ipynb`. The code itself imports various scikit-learn and other Python libraries, sets up an S3 client to download zip files, and loops through them to extract contents into a Pandas DataFrame. The status bar at the bottom shows the notebook is "Fully initialized" and "Busy".

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import statistics

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report, confusion_matrix

[46]: import boto3
import gzip
from io import BytesIO

# Replace these with your S3 bucket and folder path containing zip files
bucket_name = 'credit-classification-transformed'
folder_path = 'train-dataset/'

# Create an S3 client
s3 = boto3.client('s3')

# List objects in the S3 bucket and filter for zip files in the specified folder
response = s3.list_objects(Bucket=bucket_name, Prefix=folder_path)
# print(response)
gzip_files = [obj['Key'] for obj in response.get('Contents', []) if obj['Key'].endswith('.gz')]

# Create an empty Pandas DataFrame
df_train_original = pd.DataFrame()

# Loop through each gzip file and extract content
for gzip_file in gzip_files:
    # Download the gzip file from S3
    response = s3.get_object(Bucket=bucket_name, Key=gzip_file)
    gzip_content = response['Body'].read()
```

Load Data- Train dataset df_train.info()

The screenshot shows a Jupyter Notebook interface with a sidebar containing a file tree. The main area displays code and its output.

```
df_train_original = df_train_original.append(csv_data, ignore_index=True)
/tmp/ipykernel_24012/4264104709.py:33: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
df_train_original = df_train_original.append(csv_data, ignore_index=True)
```

3. Load Data

```
[47]: # df_train_original = pd.read_csv('/content/gdrive/My Drive/Cloud/CCDML/train.csv')
df_train = df_train_original.copy()
df_train.head()
```

	id	customer_id	month	Name	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	num_bank_accounts	num_credit_card
0	0x7f02	CUS_0x10aa	January	Carey Gillama	19	Musician	114432.03	9272.0	1	2
1	0x7f03	CUS_0x10aa	February	Carey Gillama	19	Musician	114432.03	9272.0	1	2
2	0x7f04	CUS_0x10aa	March	Carey Gillama	19	Musician	114432.03	9272.0	1	2
3	0x7f05	CUS_0x10aa	April	Carey Gillama	19	Musician	114432.03	9272.0	1	92
4	0x7f06	CUS_0x10aa	May	Carey Gillama	19	Musician	114432.03	9272.0	1	2

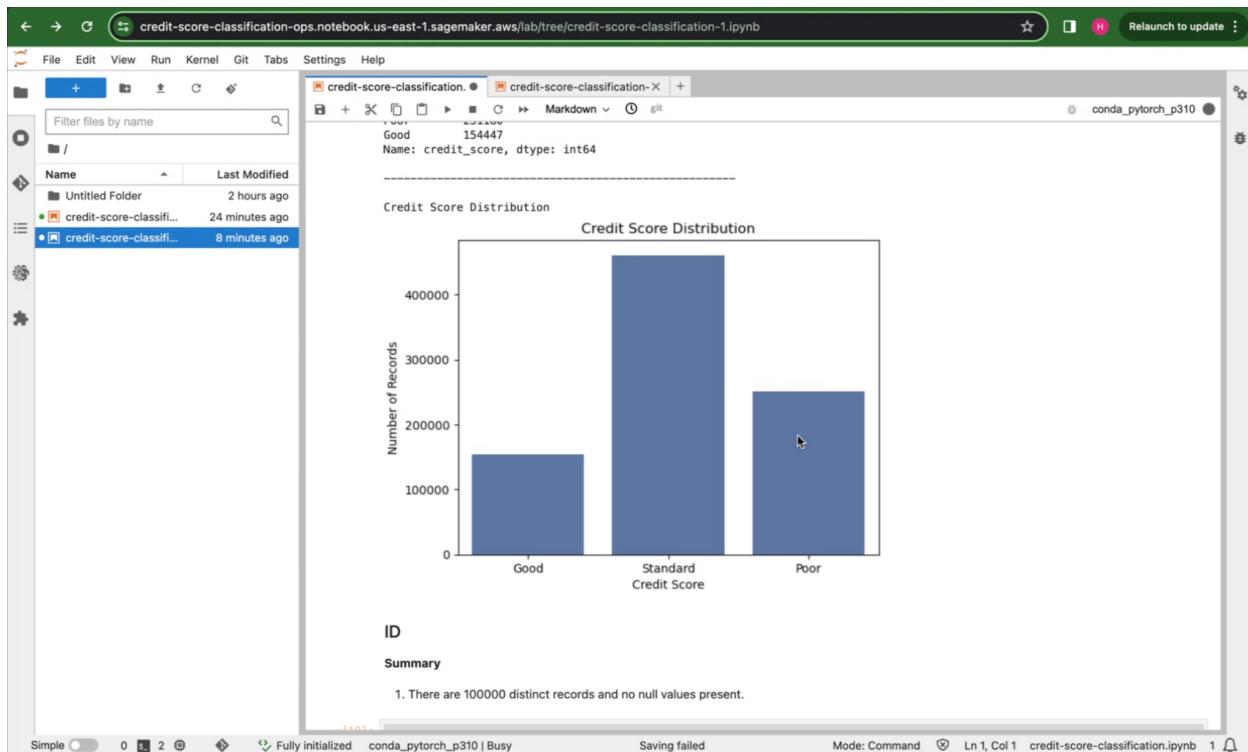
5 rows x 26 columns

4. Exploratory Data Analysis

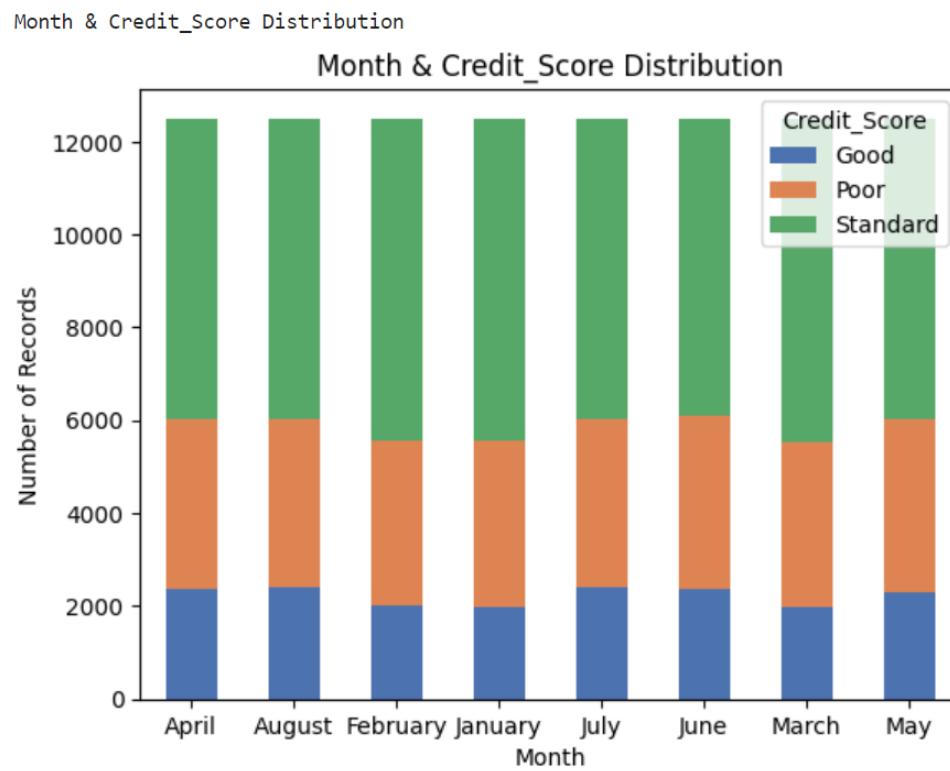
4.1 Preview Dataset

```
[4]: #Check_Data_Size
print("Train Data Size : ",df_train.shape)
Train Data Size : (866504, 26)
```

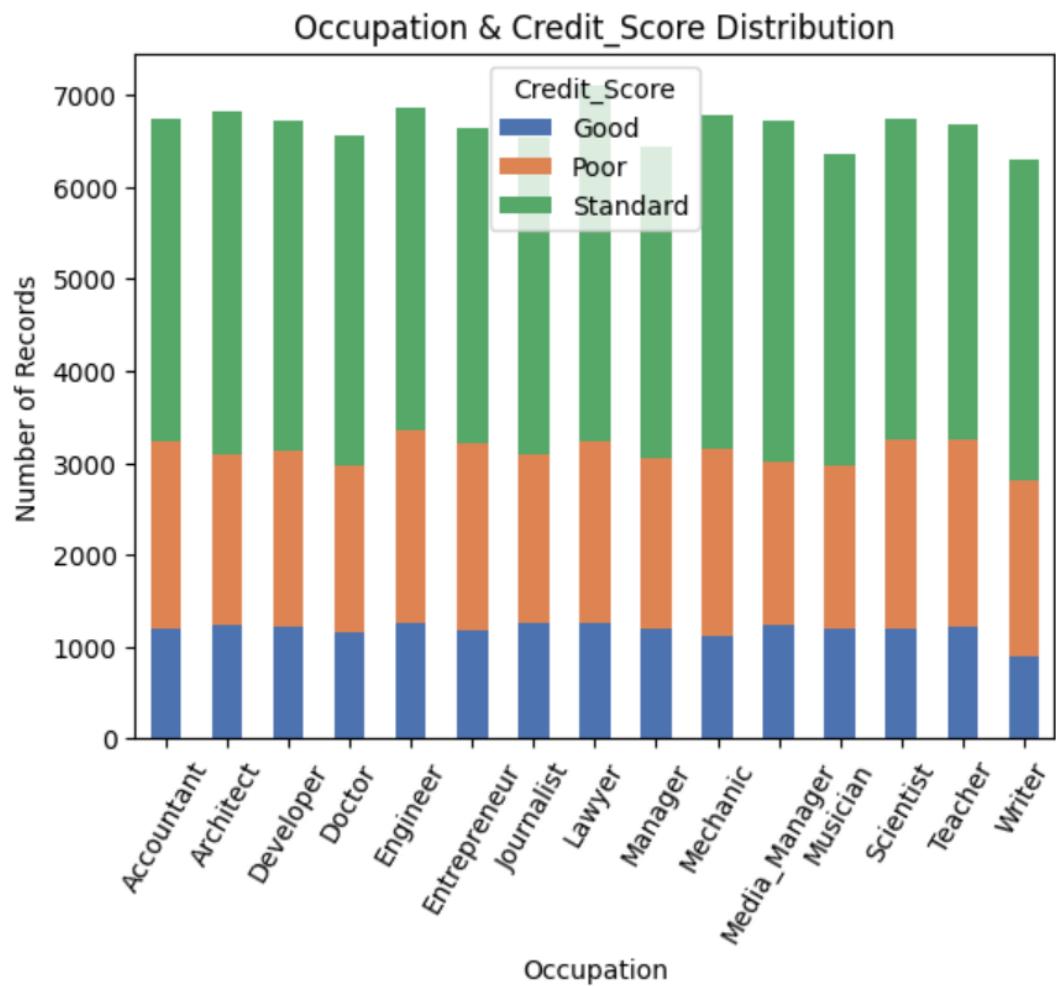
Exploratory Data Analysis Categorical Variables Credit Score Summary- Credit Score Distribution



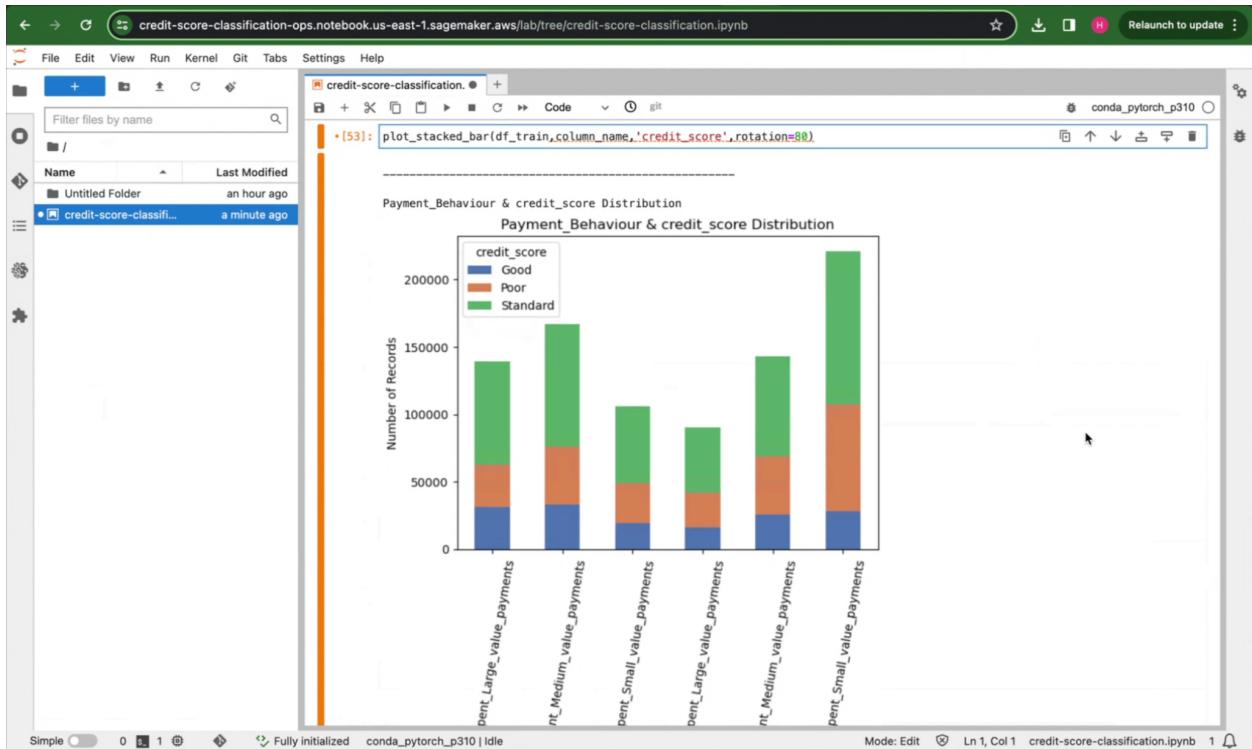
Month & Credit_Score Distribution



Occupation & Credit_Score Distribution

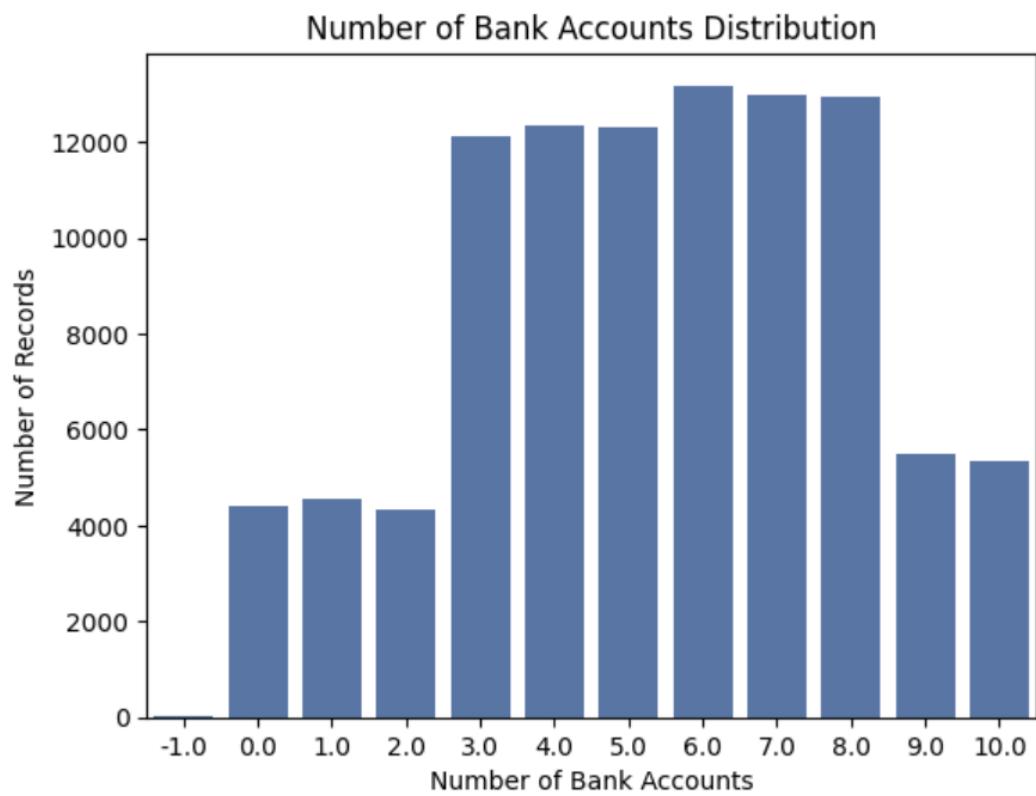


Payment_Behaviour & Credit_Score Distribution

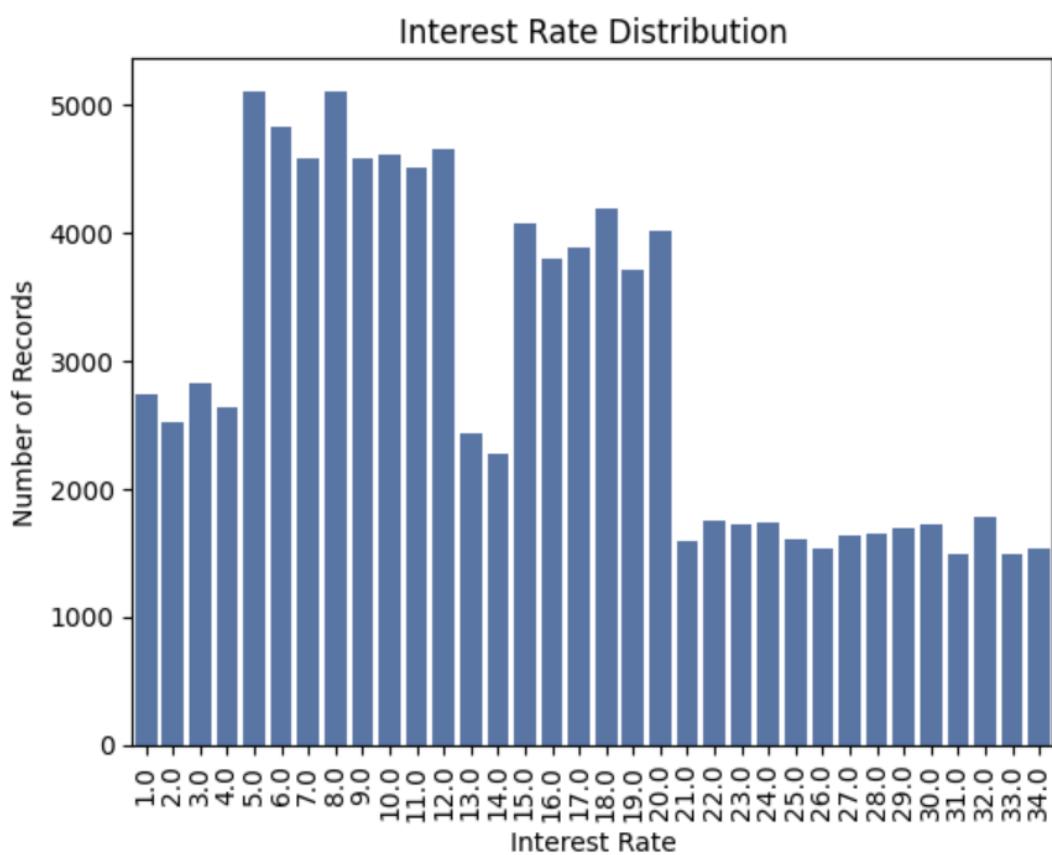


Numerical Variables

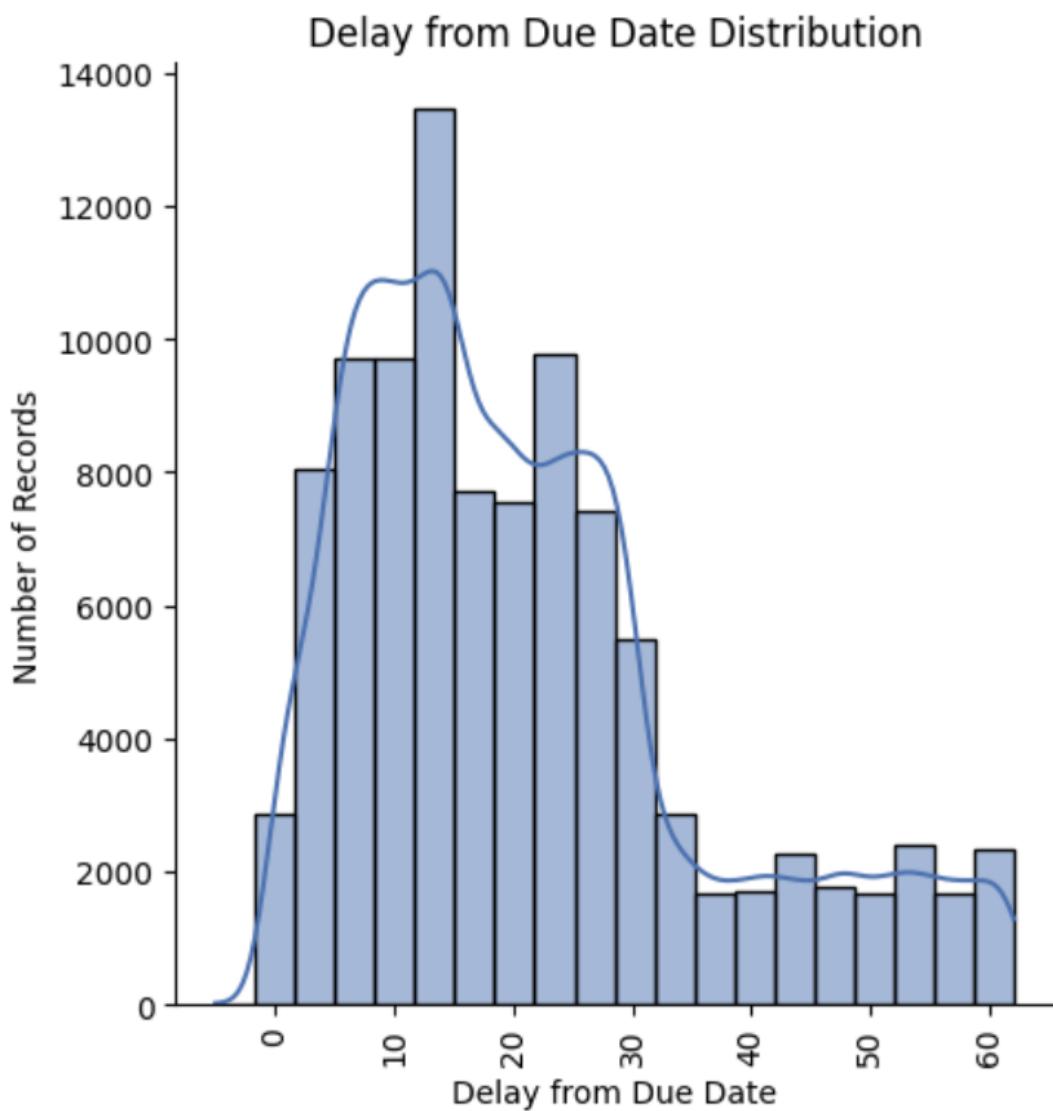
Number of Bank Accounts Distribution



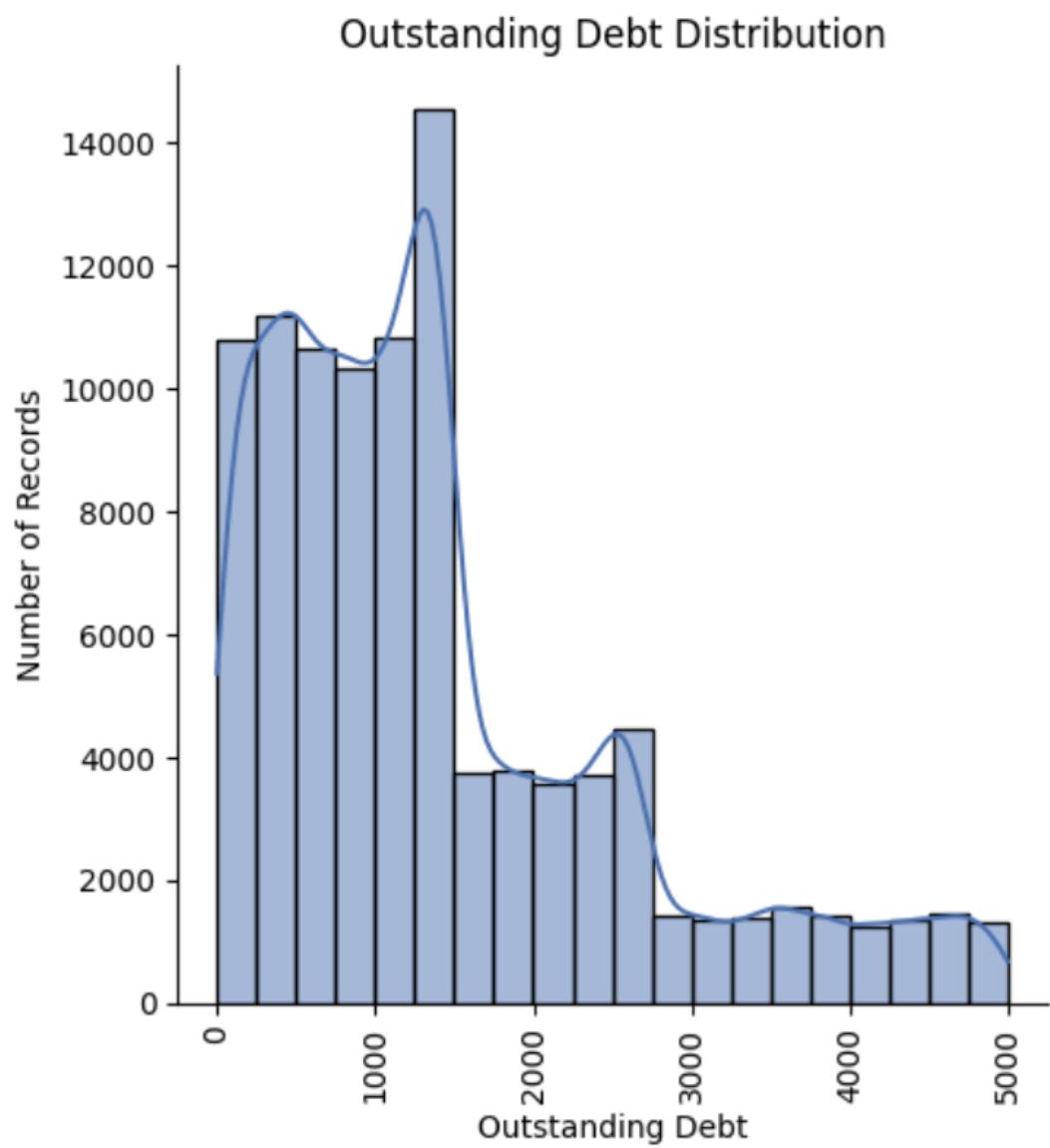
Interest Rate Distribution



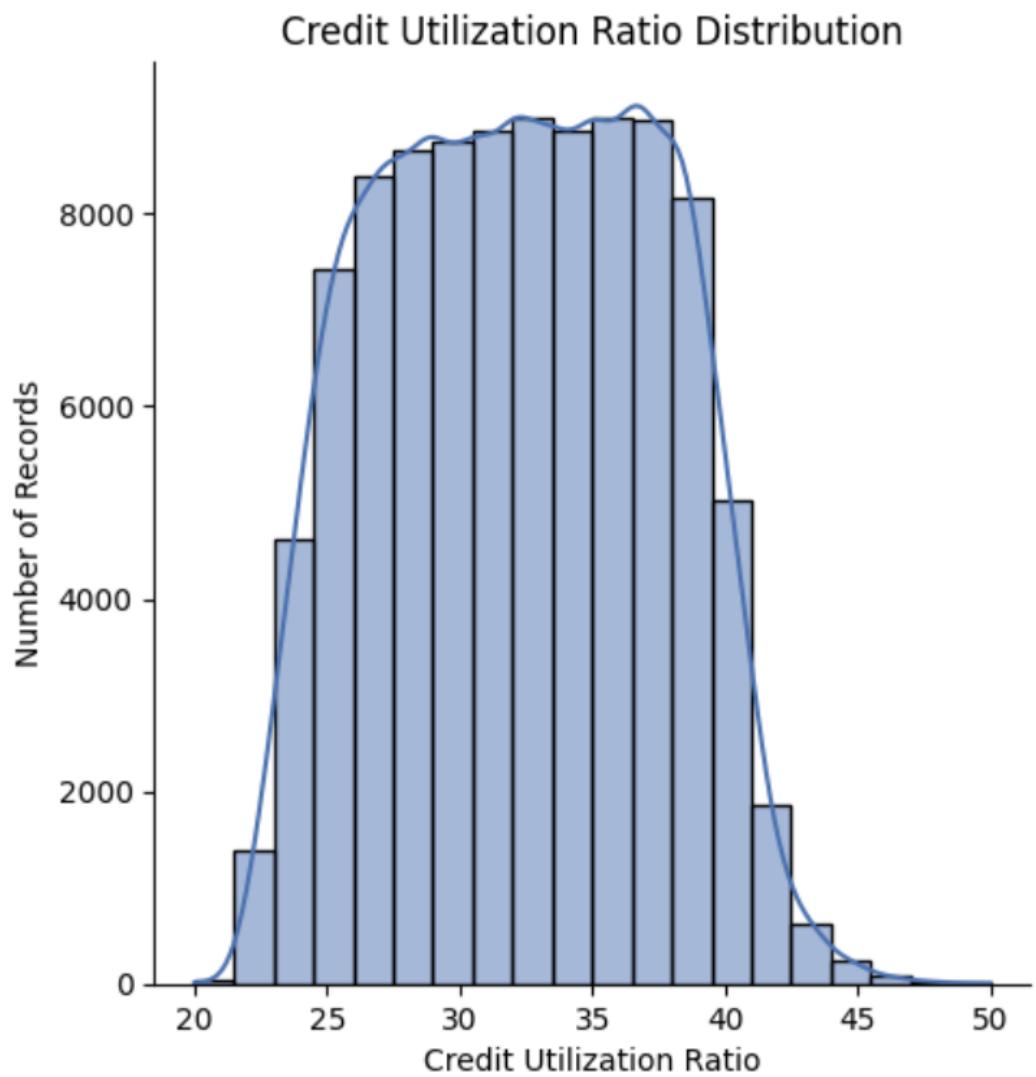
Delay from Due Date Distribution



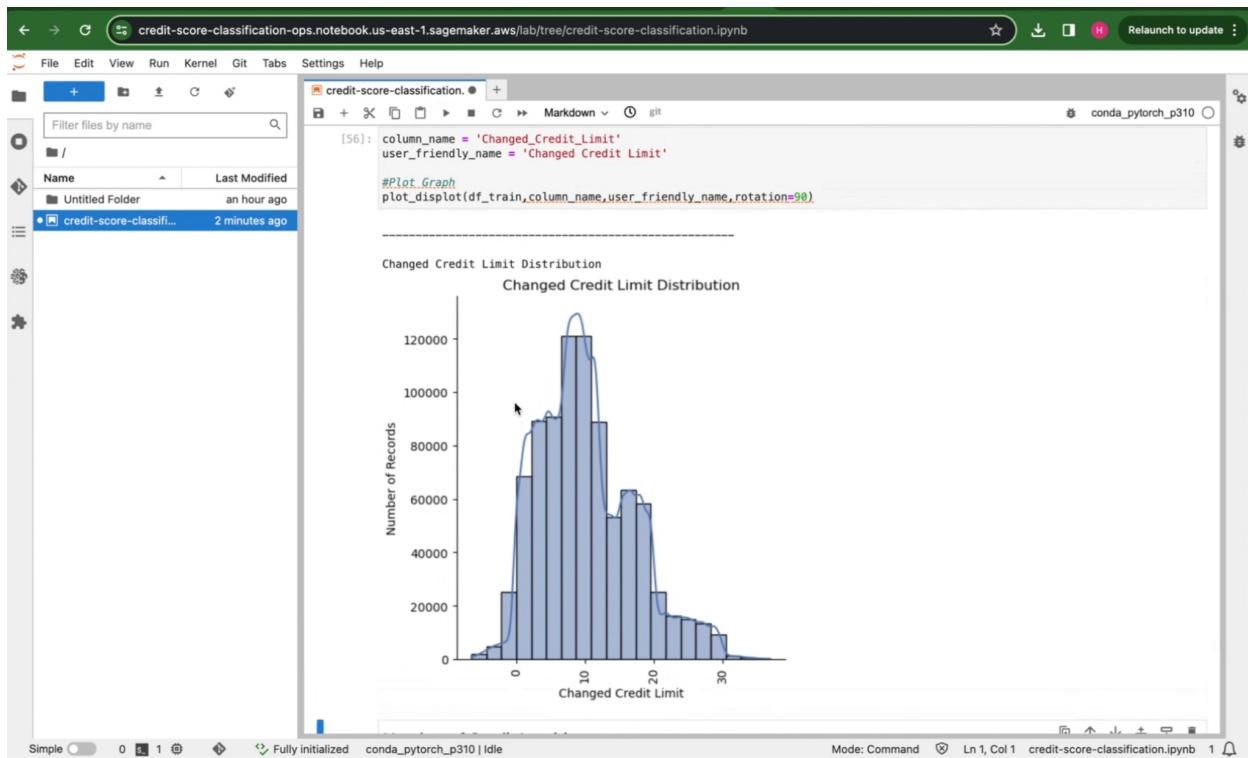
Outstanding Debt Distribution



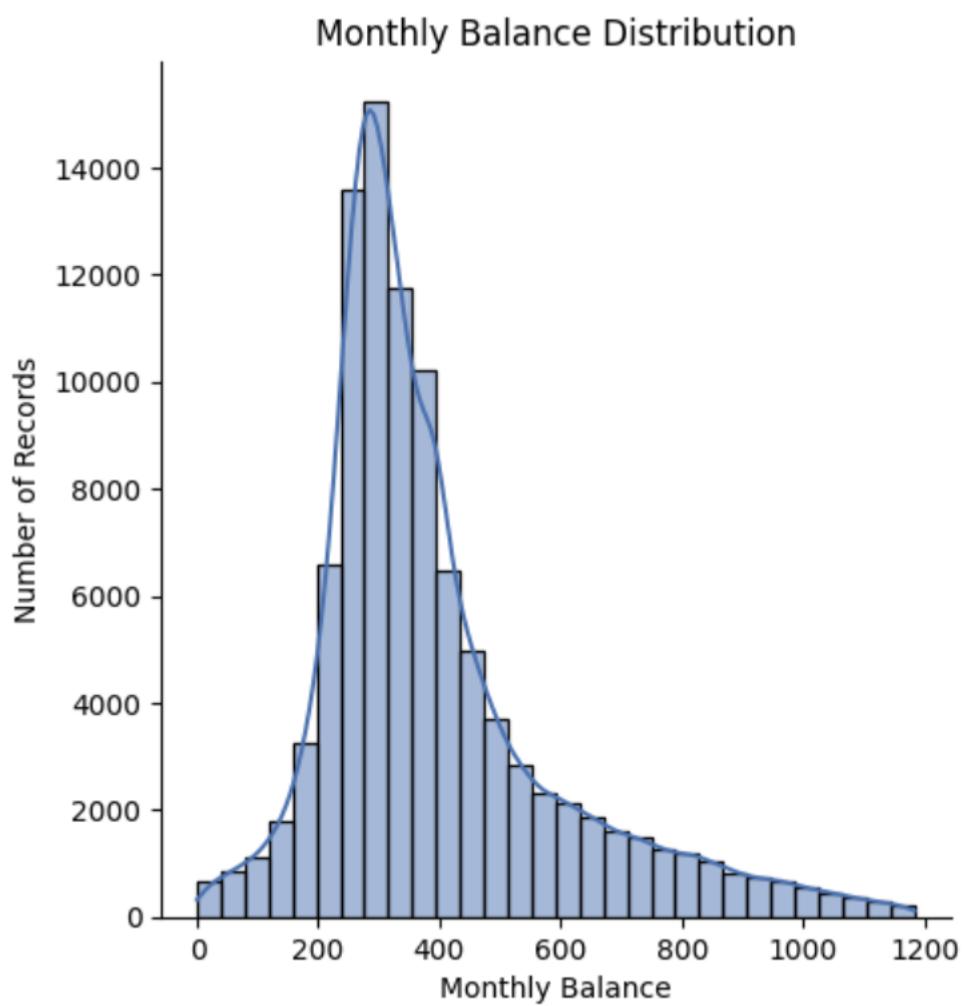
Credit Utilization Ratio Distribution



Changed Credit Limit Distribution



Monthly Balance Distribution



Data Transformation (df.head())

```
[53]: #Label Encoding
from sklearn.preprocessing import LabelEncoder

categorical_columns = ['Occupation','Type_of_Loan','Credit_Mix','Payment_of_Min_Amount','Payment_Behaviour','Credit_Score']
# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Loop through each column and apply label encoding
for column in categorical_columns:
    df_train[column] = label_encoder.fit_transform(df_train[column])
```

1

```
[54]: df_train.head()
```

	Month	Age	Occupation	Annual_Income	Monthly_Inhand_Salary	Num_Bank_Accounts	Num_Credit_Card	Interest_Rate	Num_of_Loan	Type_of_Loan	...	Credit_Mix
0	1	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	1
1	2	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	1
2	3	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	1
3	4	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	1
4	5	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	1

5 rows × 24 columns

Develop and Train Models, Evaluation and Validation

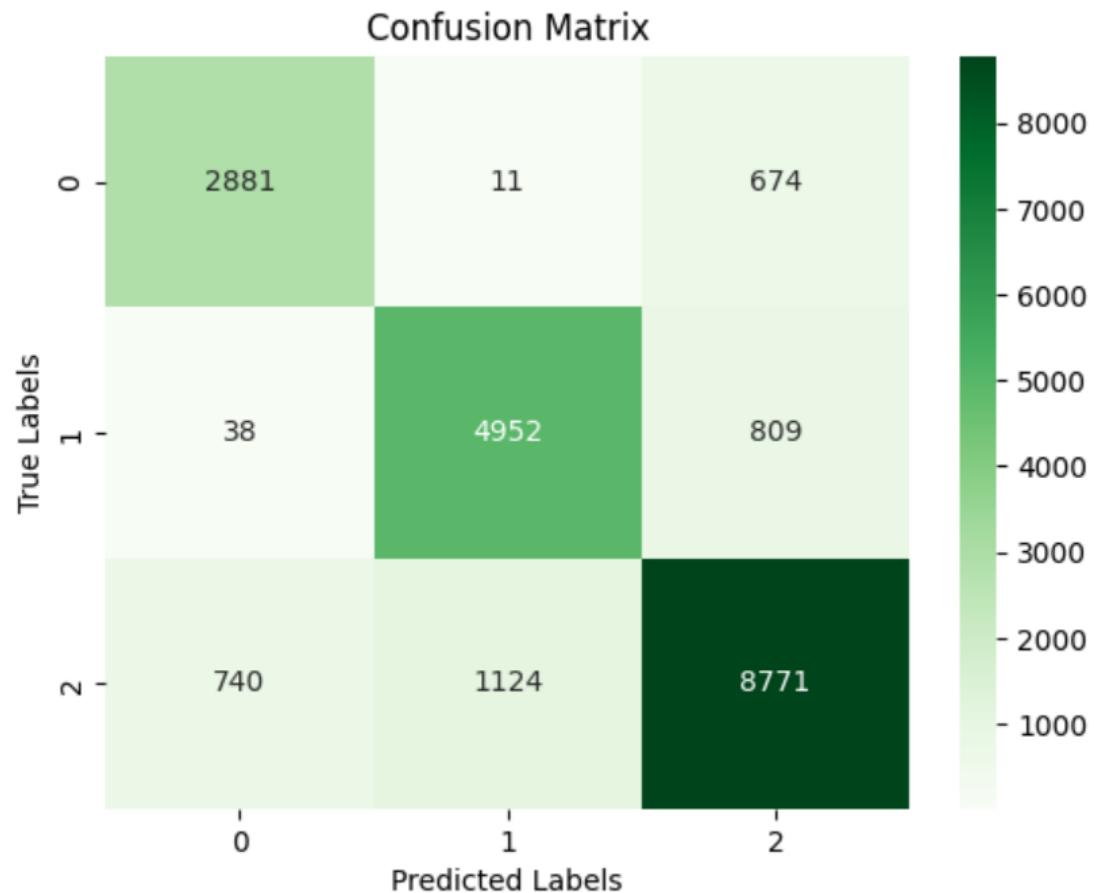
Approach 1

```
# Print the performance metrics list
```

Accuracy, Precision, Recall values for different classifiers

```
Classifier: Decision Tree
Average Accuracy: 0.7220
Average Precision: 0.7055
Average Recall: 0.7052
-----
Classifier: Random Forest
Average Accuracy: 0.8157
Average Precision: 0.8053
Average Recall: 0.8100
-----
Classifier: KNN
Average Accuracy: 0.7030
Average Precision: 0.6757
Average Recall: 0.6851
-----
Classifier: Gaussian NB
Average Accuracy: 0.6394
Average Precision: 0.6328
Average Recall: 0.6882
-----
Classifier: XGB
Average Accuracy: 0.7742
Average Precision: 0.7590
Average Recall: 0.7639
-----
```

Confusion matrix for Random Forest Classifier



Approach 2

We have addressed the issue of imbalance in data using Synthetic Minority Over-sampling Technique(SMOTE) and improved the accuracy of the algorithm

```
[64]: # Creating the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Training the classifier
rf_classifier.fit(X_train, y_train)

# Making predictions on the test set
y_pred = rf_classifier.predict(X_test)

# Evaluating the model
evaluate_model(y_test, y_pred)
```

Classification Report				
	precision	recall	f1-score	support
0	0.89	0.96	0.92	10635
1	0.87	0.91	0.89	10635
2	0.89	0.79	0.84	10635
accuracy			0.89	31905
macro avg	0.89	0.89	0.88	31905
weighted avg	0.89	0.89	0.88	31905

Confusion matrix

