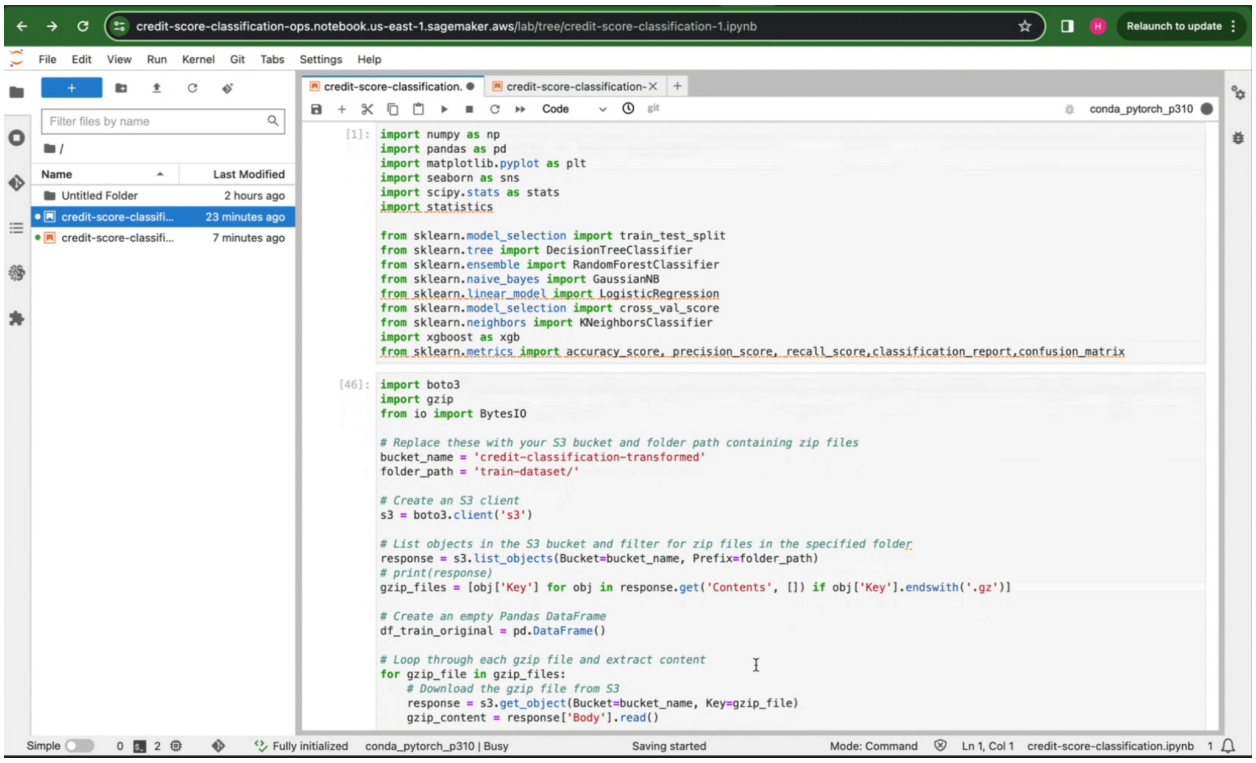
Fall 2023 ITCS 6190 Project Review Deliverable 3

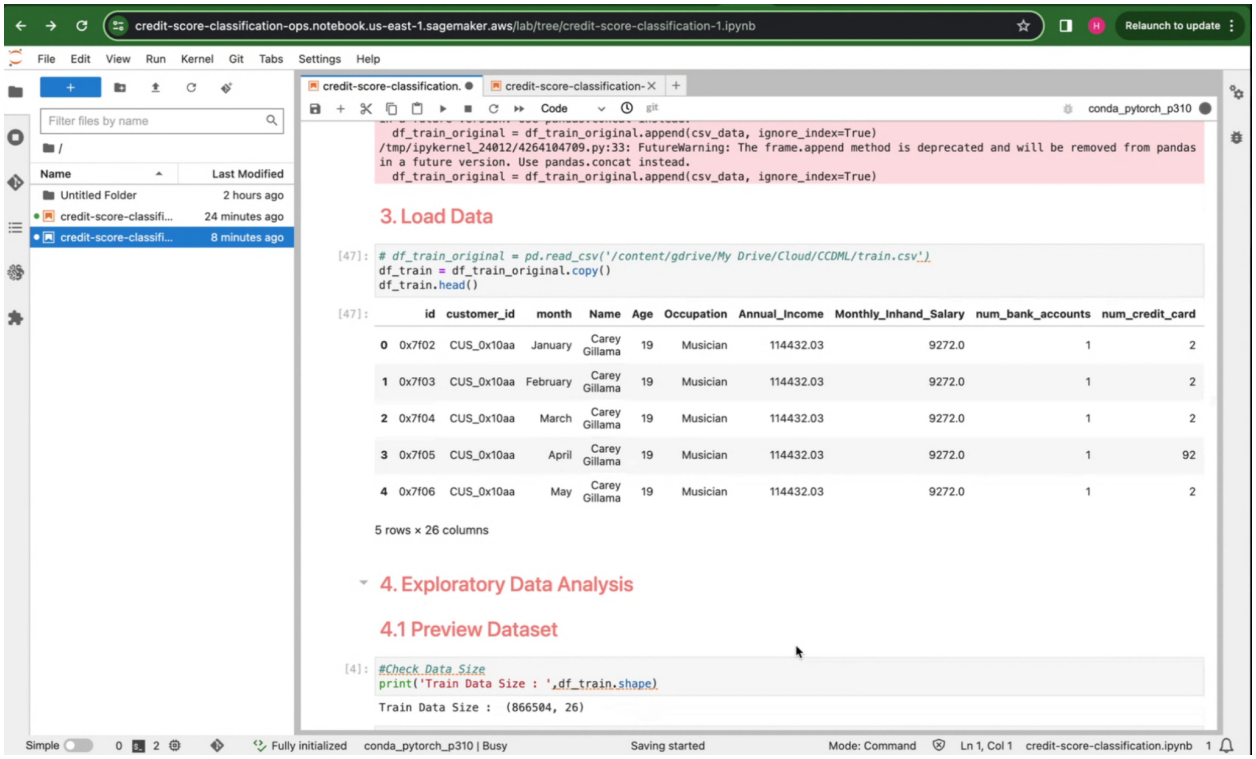
Group 5

Machine Learning Framework

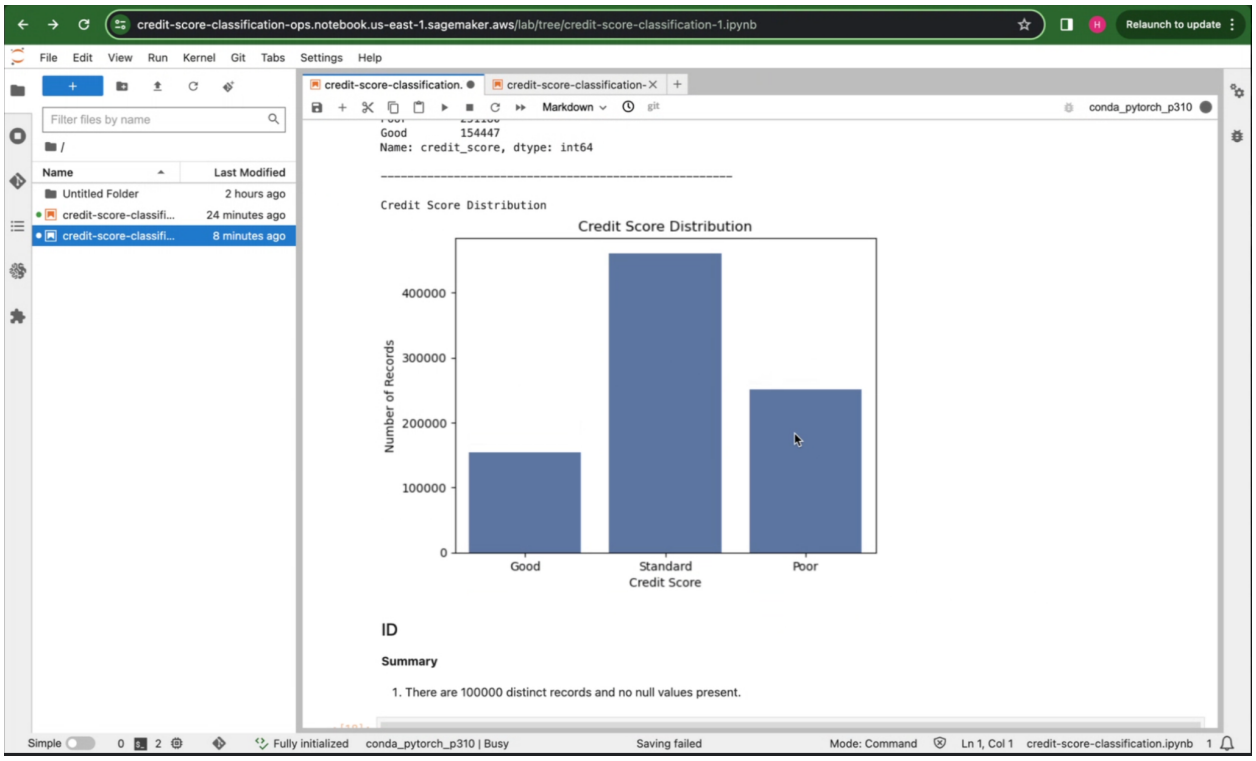
We used JupyterLab in SageMaker to work on our project. We used the scikit-learn Python library and tested different algorithms like DecisionTreeClassifier, RandomForestClassifier, KNeighborsClassifier, GaussianNB, and XGBClassifier for modeling, evaluating, and predicting customer credit scores. This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



Load Data- Train dataset df_train.info()

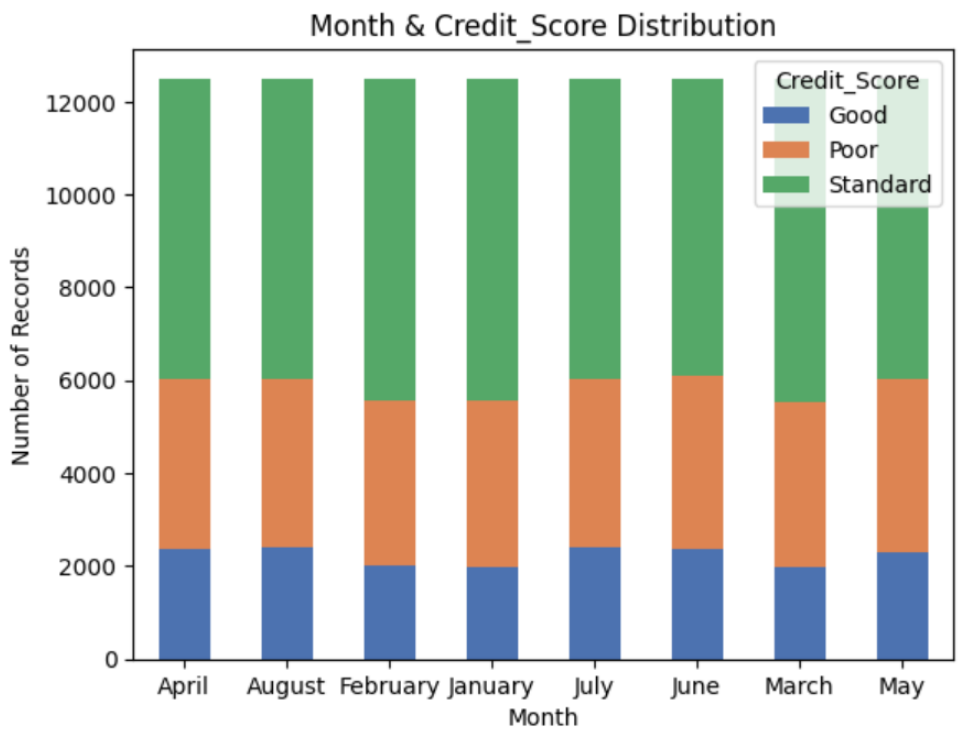


Exploratory Data Analysis Categorical Variables Credit Score Summary- Credit Score Distribution

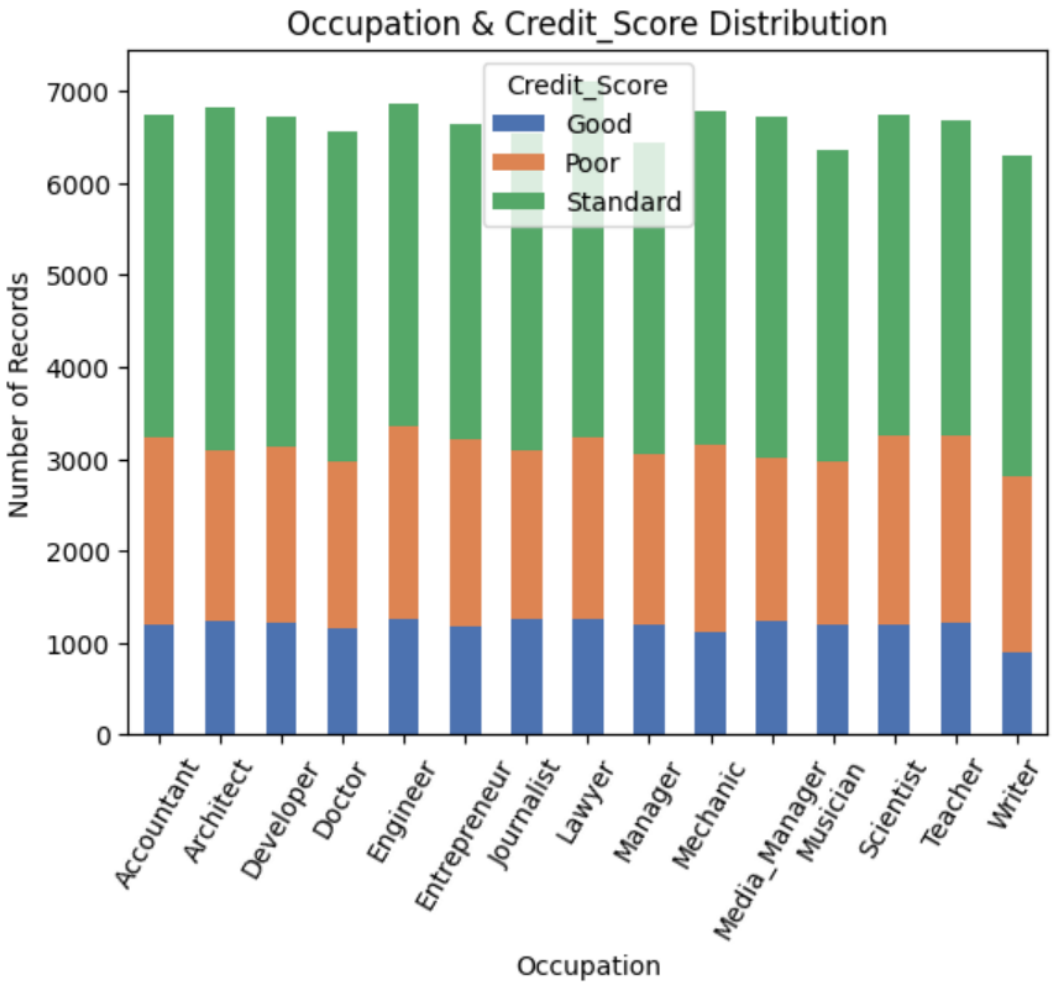


Month & Credit_Score Distribution

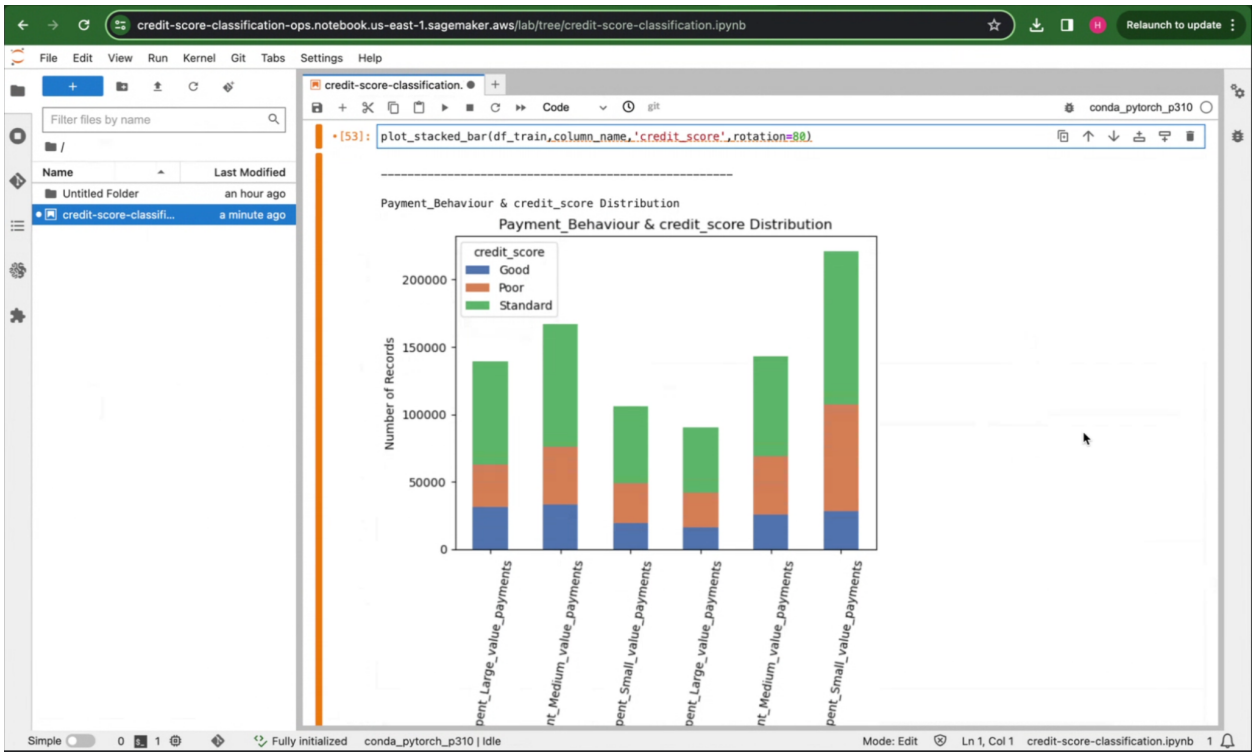
Month & Credit_Score Distribution



Occupation & Credit_Score Distribution

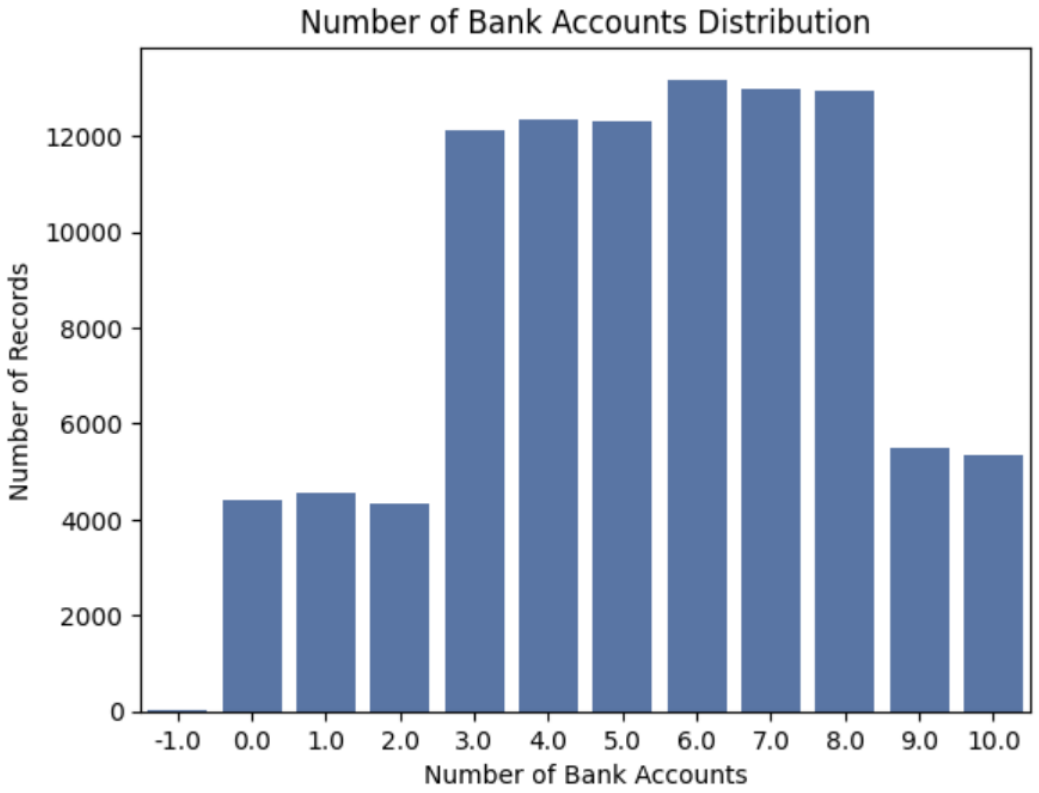


Payment_Behaviour & Credit_Score Distribution

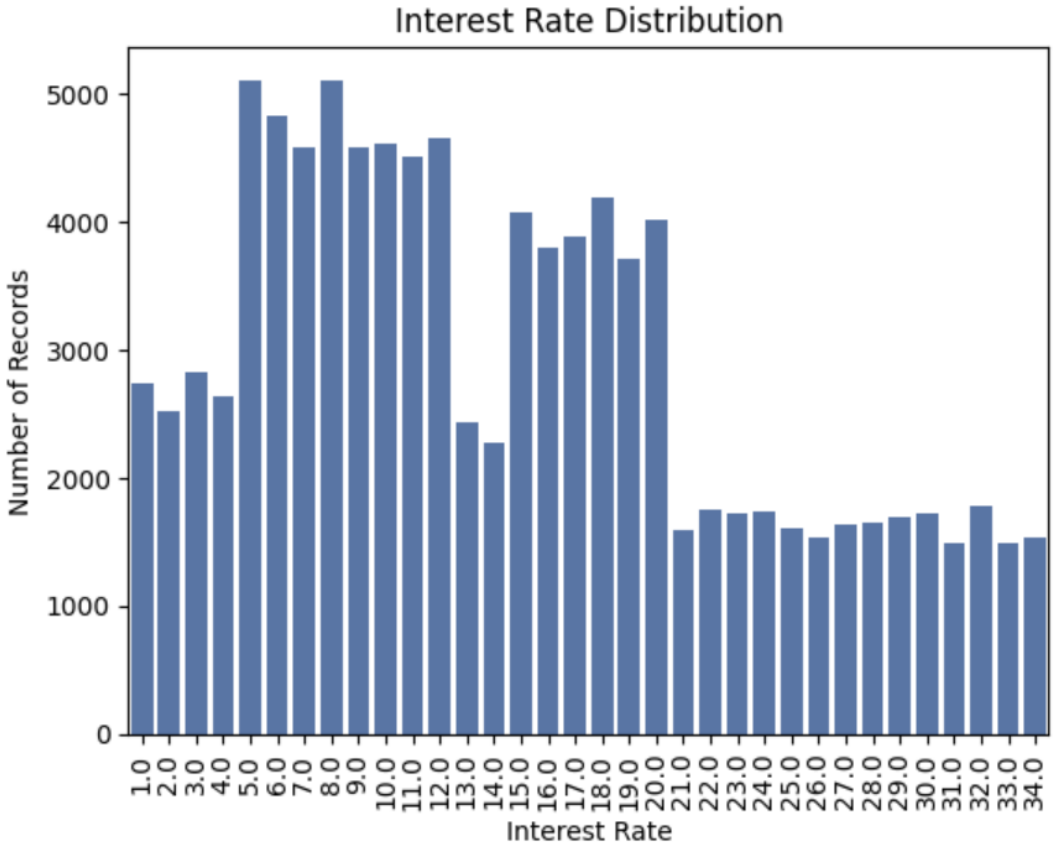


Numerical Variables

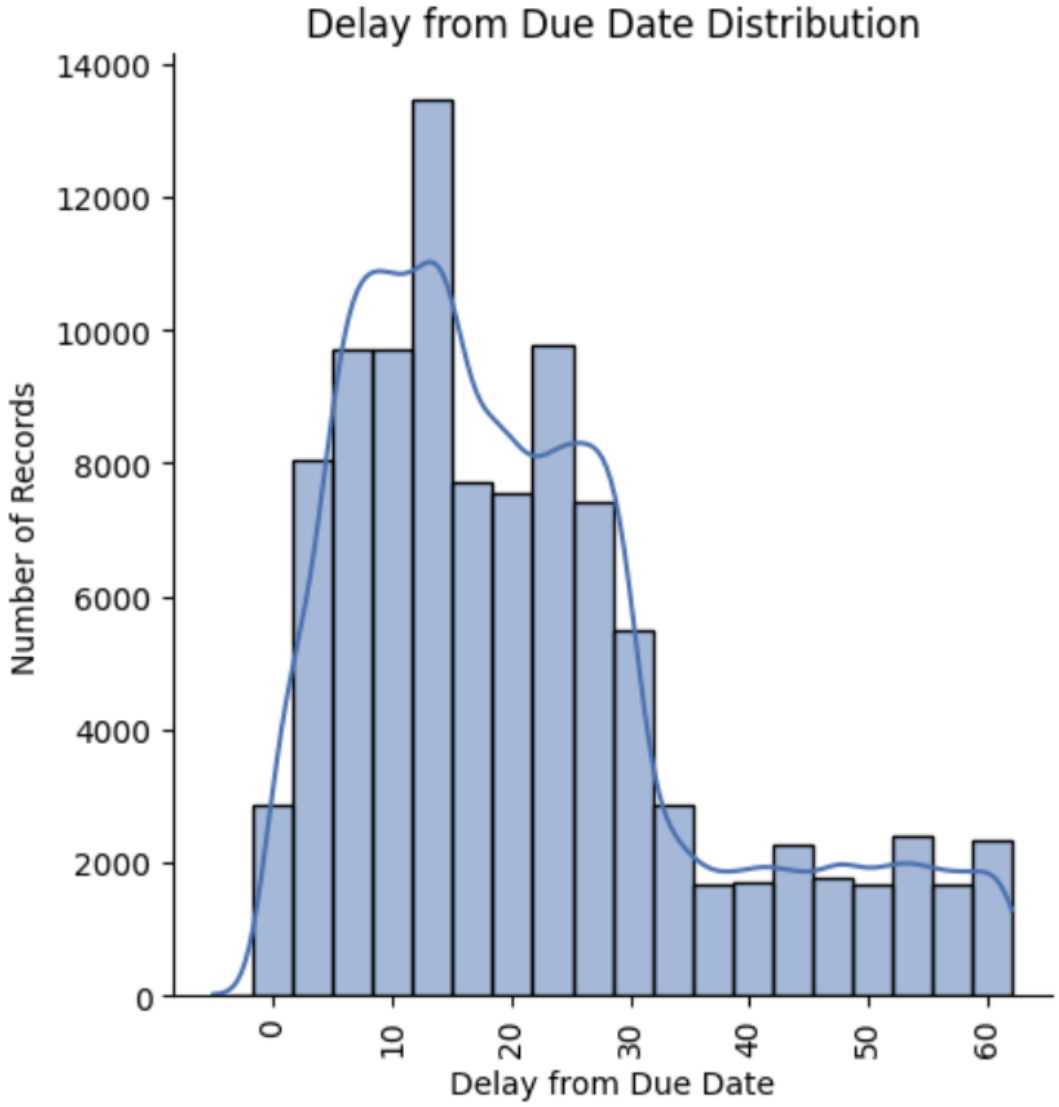
Number of Bank Accounts Distribution



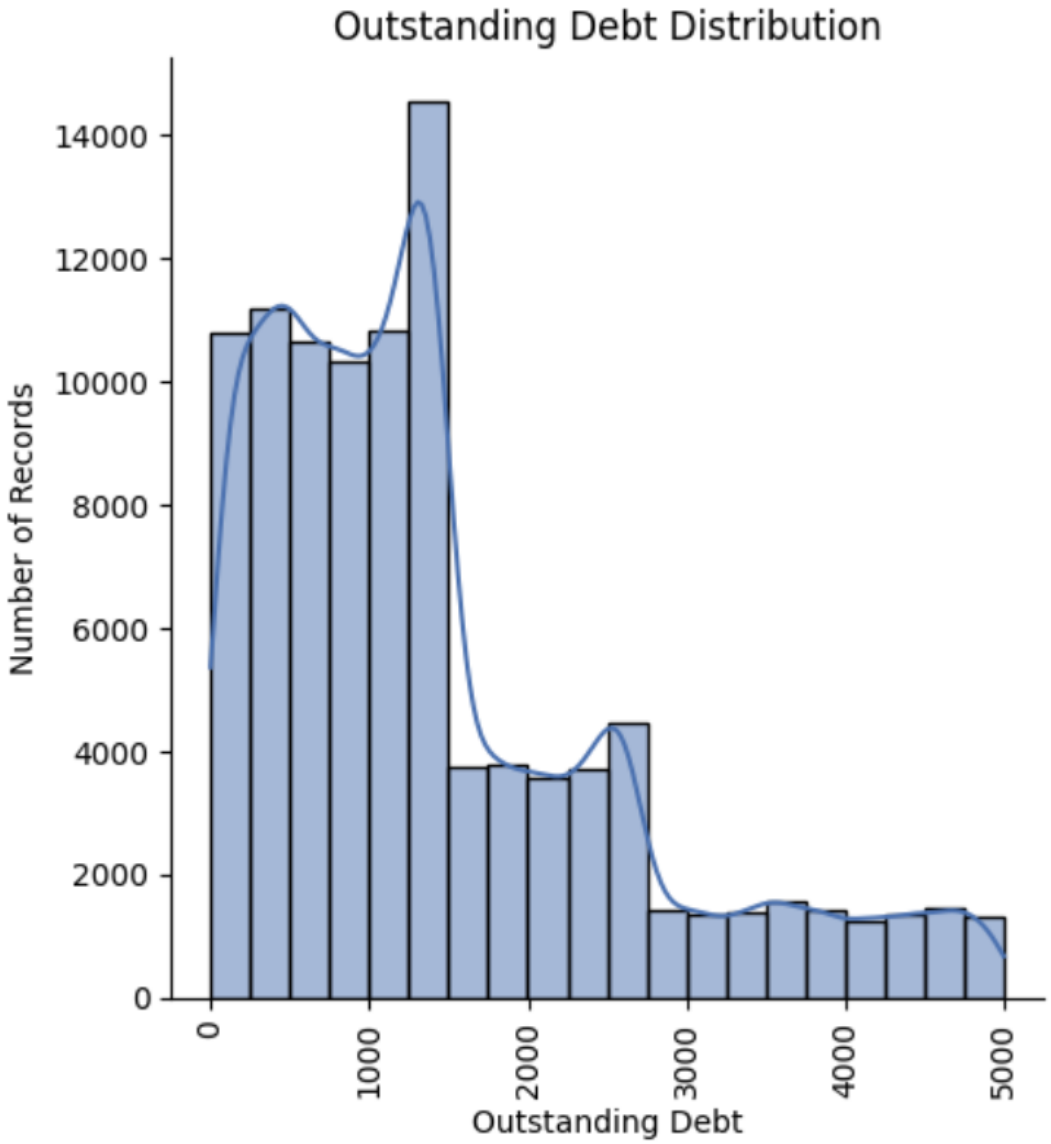
Interest Rate Distribution



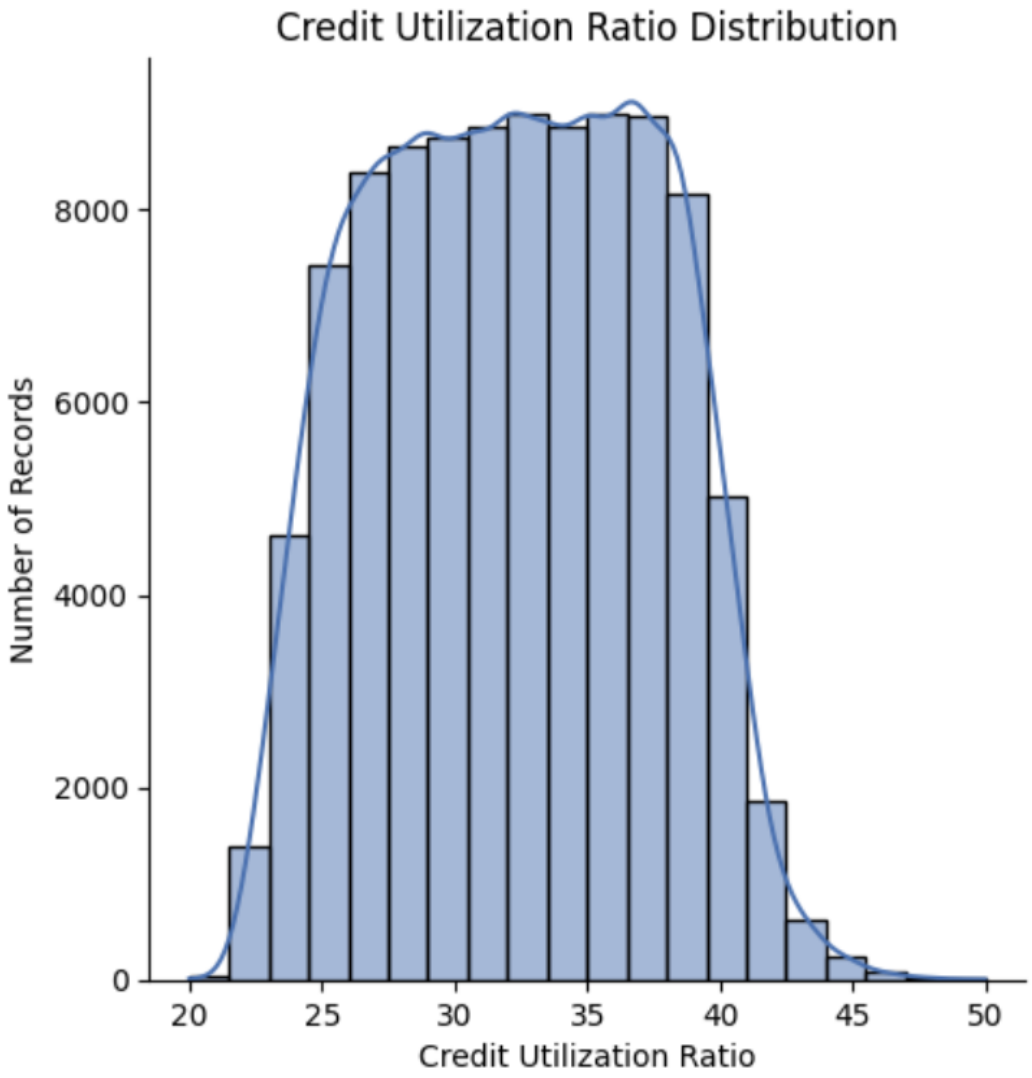
Delay from Due Date Distribution



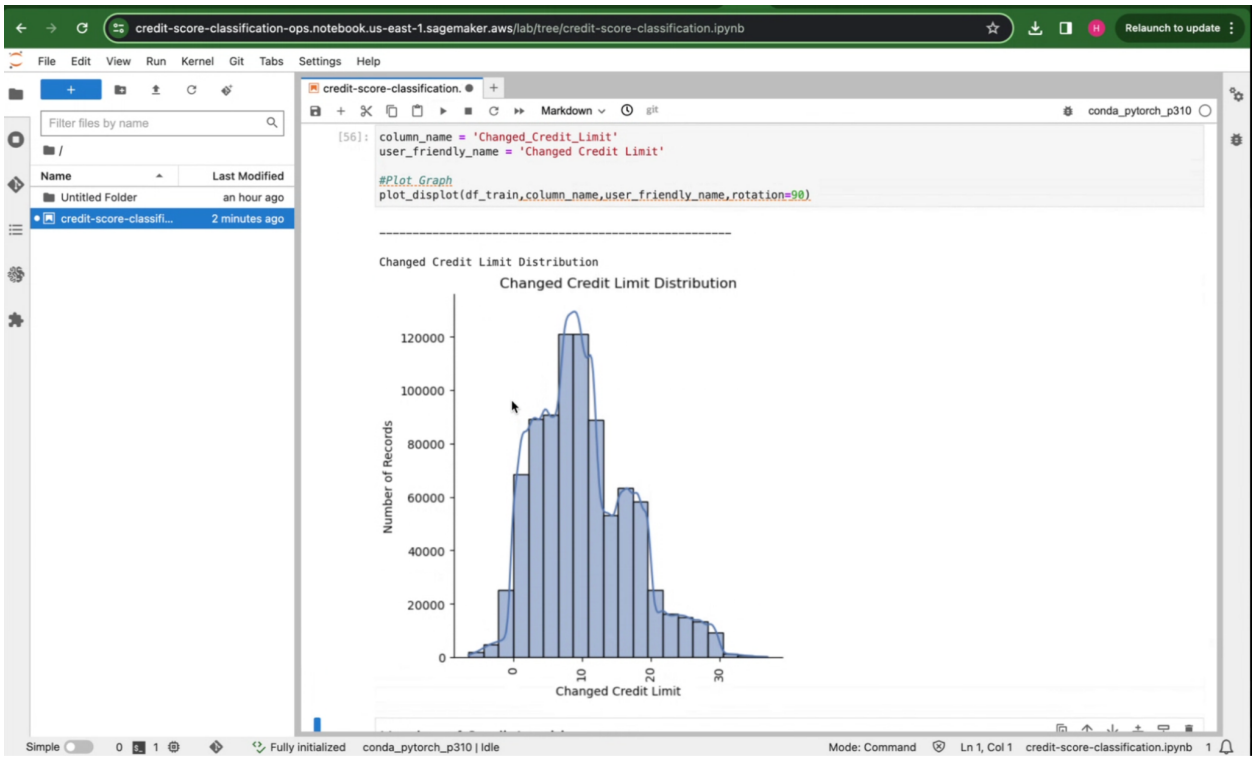
Outstanding Debt Distribution



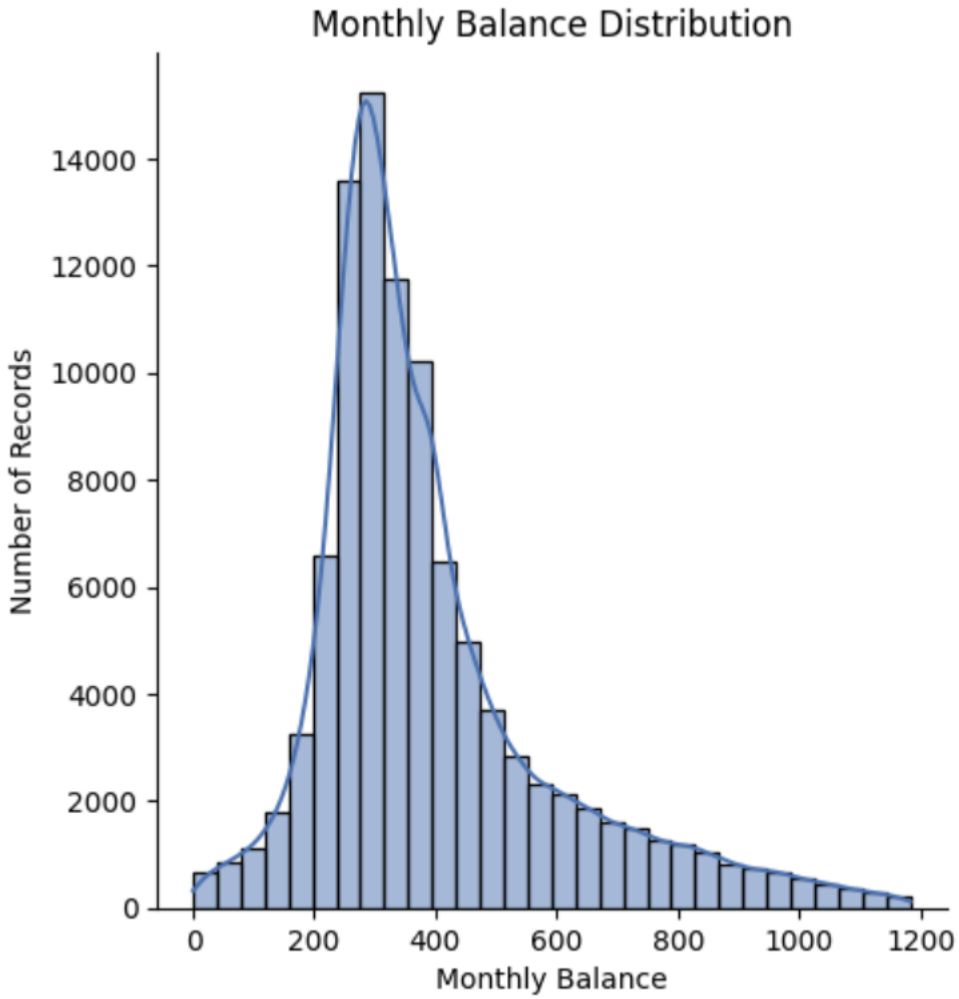
Credit Utilization Ratio Distribution



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])
```

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

[54]:

	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

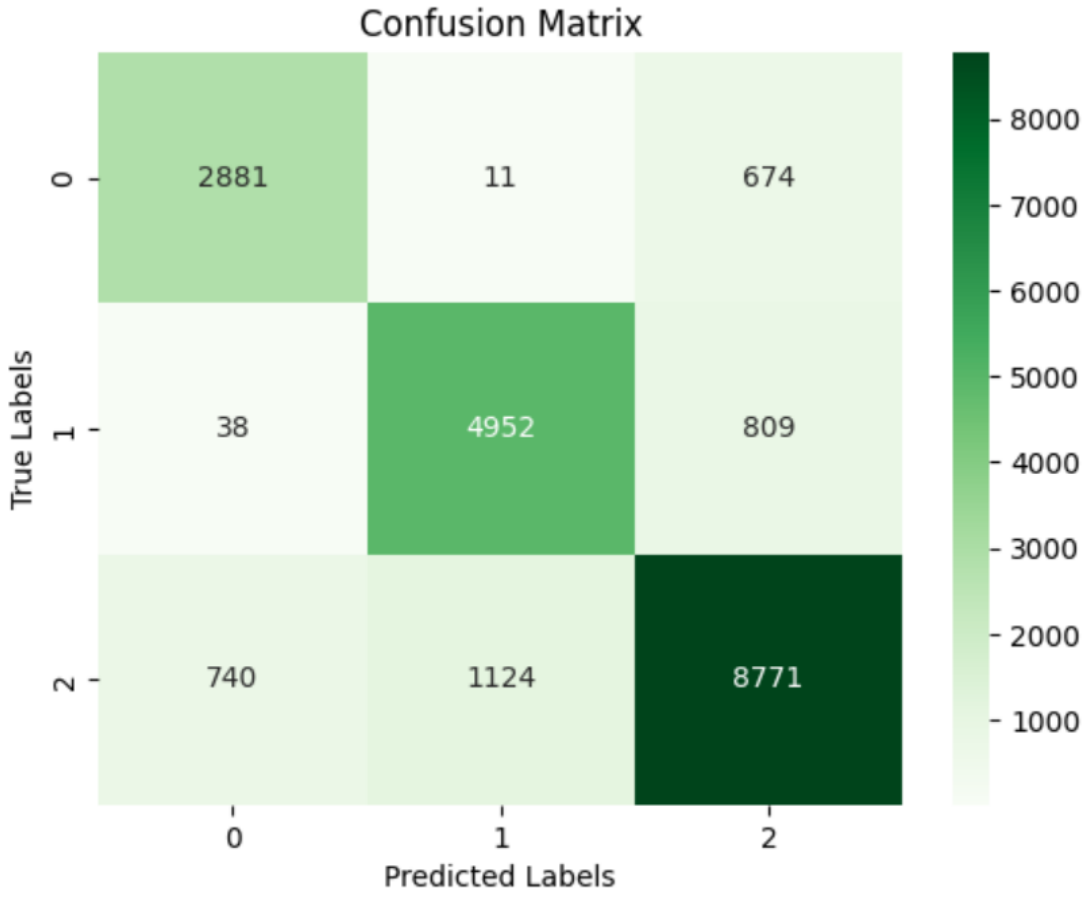
Approach 1

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: Gaussion 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



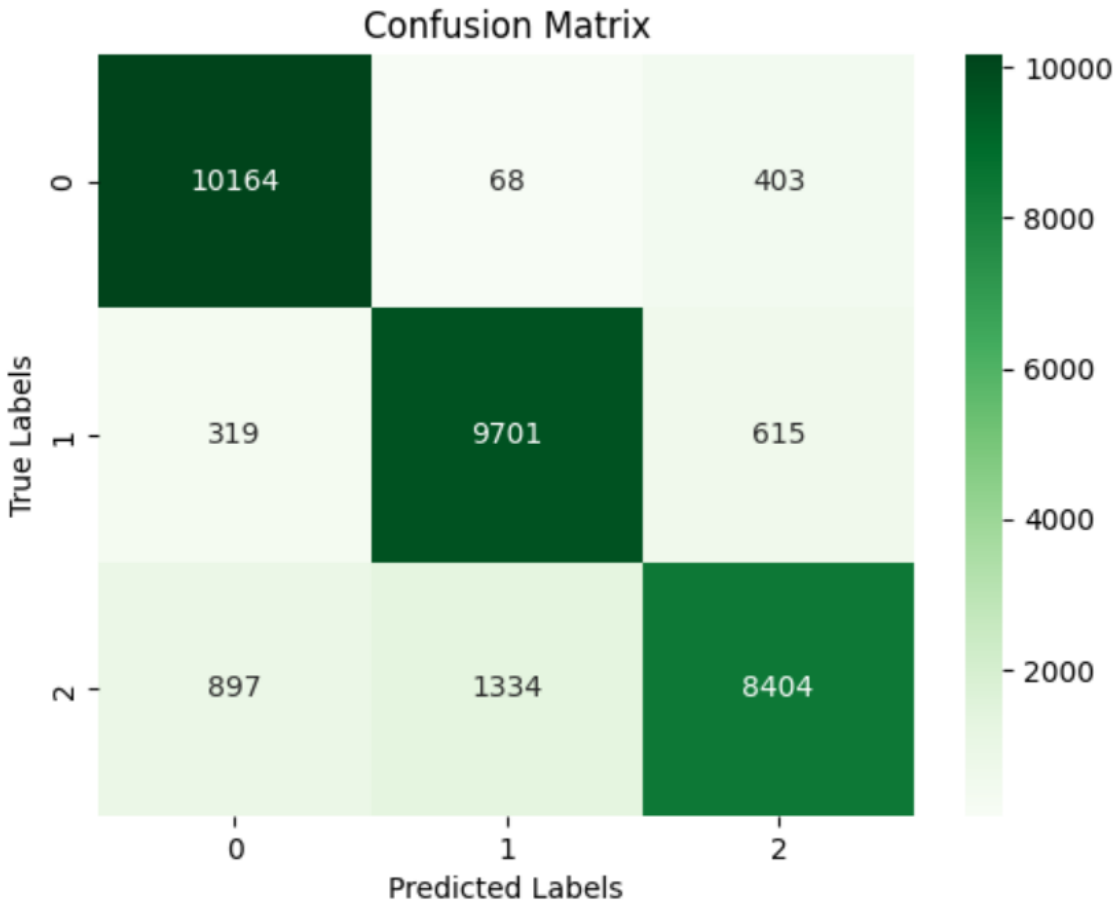
Approach 2

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



*****