



# Cloud Computing for Data Analysis Final Project

University of North Carolina at Charlotte

Credit Score Classification

Group- 5

Gaurav Avula  
Naga Nikhil Bijjala  
Harshini Karnati  
Rohan Katari  
Naga Srivatsav Machiraju

Date: 12/11/2023

# Business scenario overview

---

- The project addresses the problem of assessing individuals' creditworthiness accurately, helping financial institutions make informed lending decisions.
- Determining the credit worthiness of a individual using advanced machine learning algorithms.
- By developing a machine learning model for credit score classification, it aims to provide an opportunity to improve risk assessment, reduce defaults, and expand access to credit for qualified applicants.

# Solution overview

---

## A high-level description

- The scope of this project involves designing and implementing a machine learning model for credit score classification.
- It encompasses data collection, preprocessing, and feature engineering to create a comprehensive dataset for model training.

## Design Considerations

- The machine learning model will be selected, trained, and fine-tuned using various algorithms and techniques, with a focus on predictive accuracy and interpretability.
- Additionally, model evaluation and validation will be carried out rigorously to ensure its reliability.
- Integration with relevant AWS services for model deployment, data storage, real-time scoring, and monitoring.

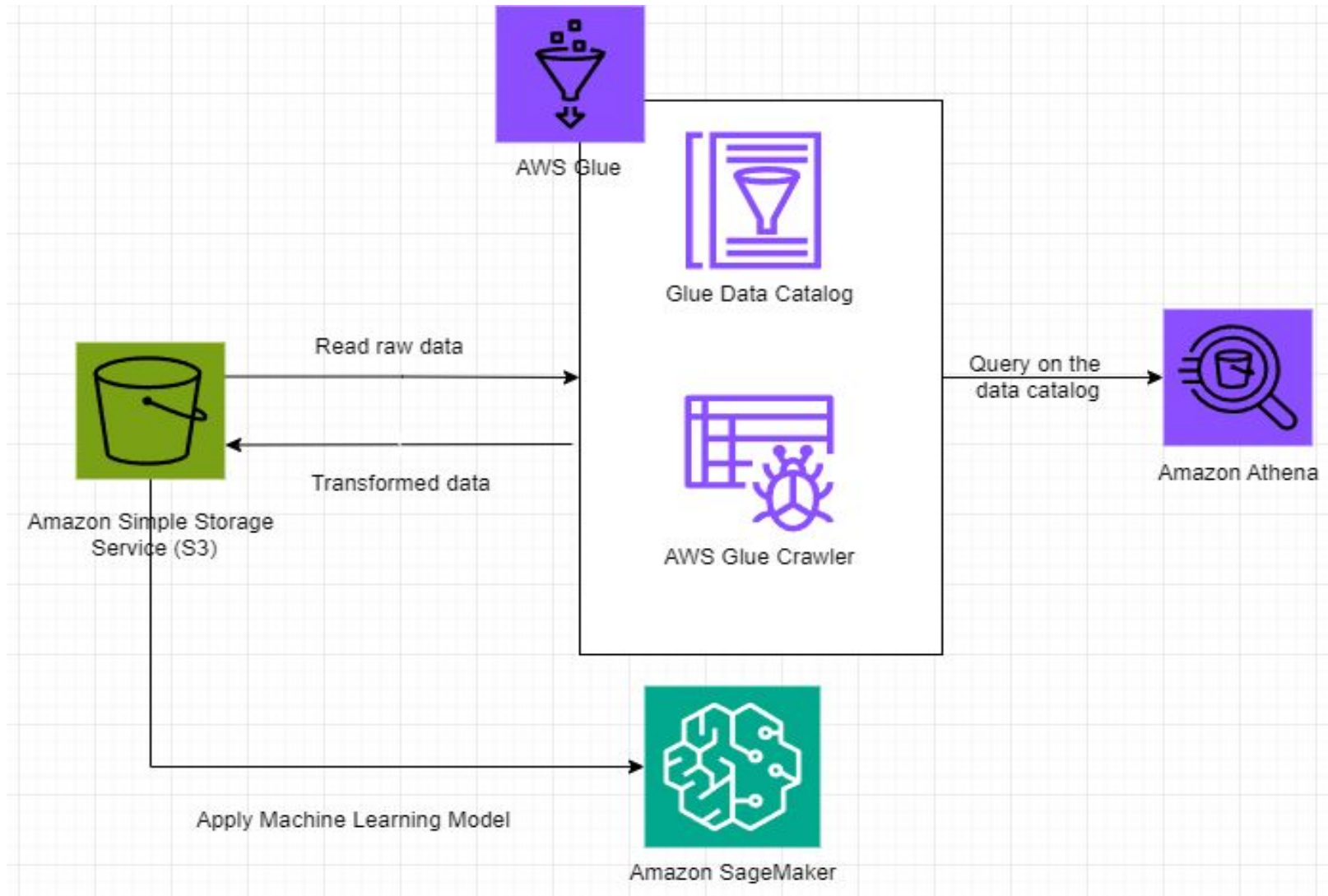
# Solution overview

---

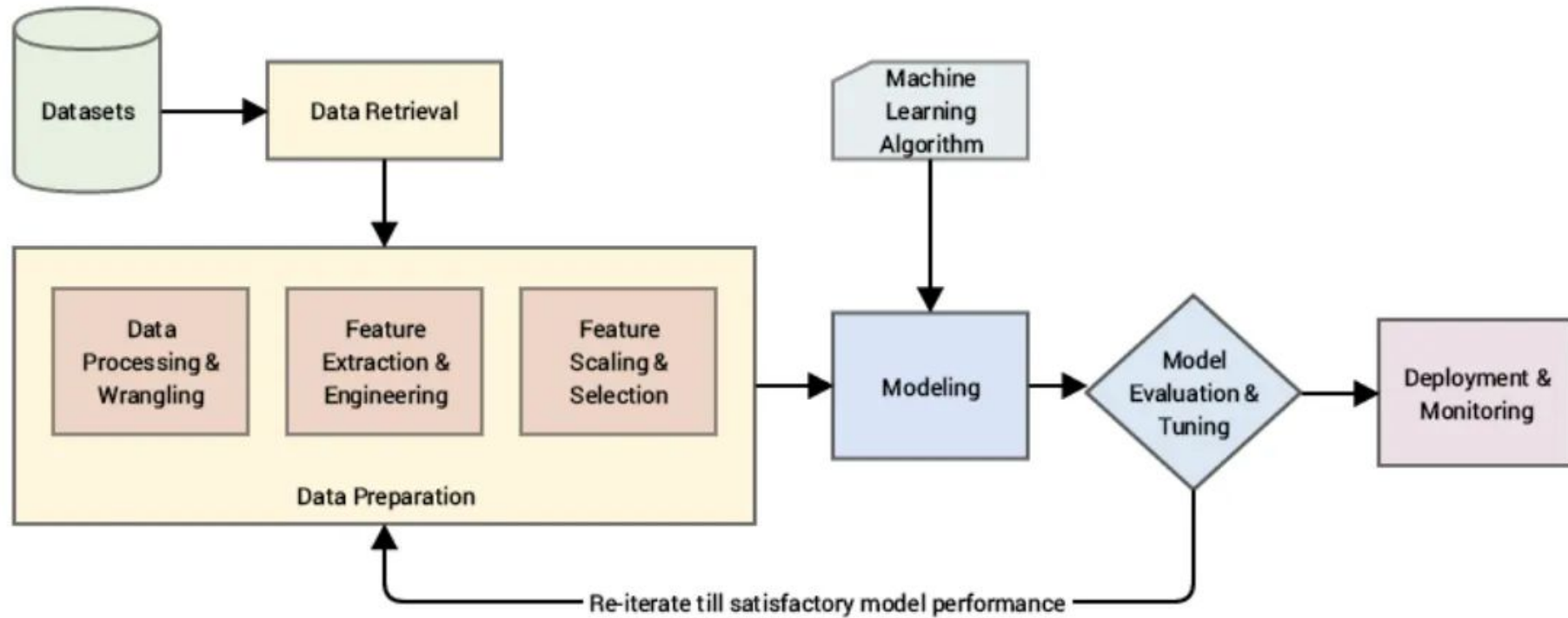
## Use Cases

1. **Automate credit approvals** with a machine learning model for swift and accurate assessments of creditworthiness.
2. **Mitigate risks** by using the model to assess and manage potential credit losses in the application process.
3. **Personalize financial offerings** by leveraging the model to tailor credit terms, interest rates, and limits based on predicted credit scores.

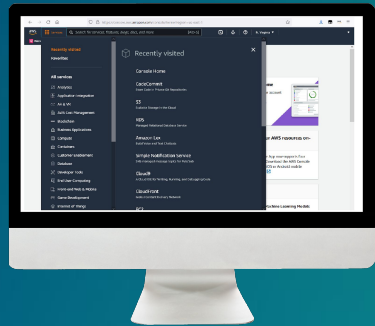
## Architecture diagram of the solution



# Machine Learning Lifecycle Applied



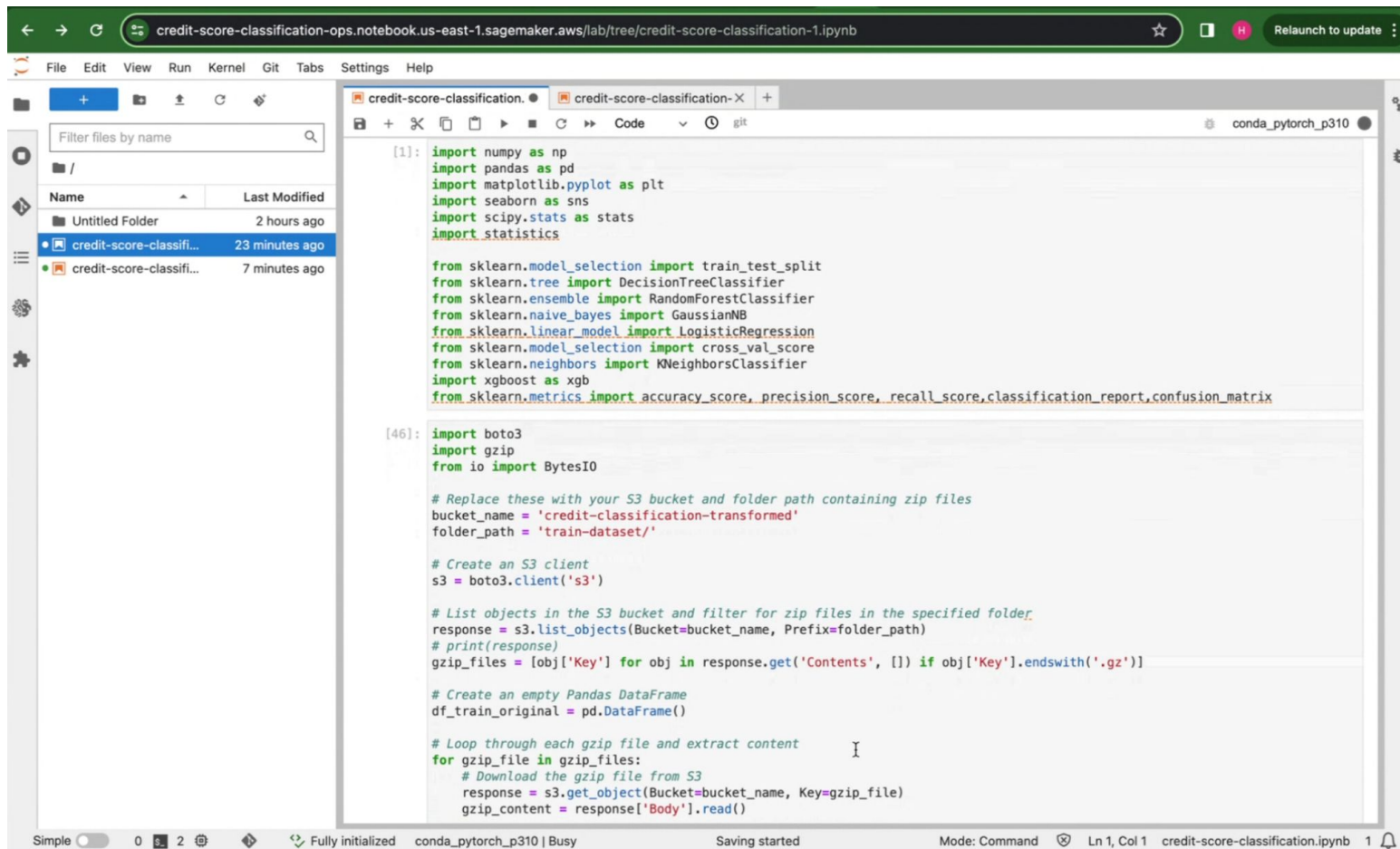
# Demo



## Project Demonstration following the Machine Learning Lifecycle

# Demo

## Data Retrieval



The screenshot displays the AWS SageMaker JupyterLab interface. The left sidebar shows a file explorer with a search bar and a list of files: 'Untitled Folder' (2 hours ago), 'credit-score-classifi...' (23 minutes ago), and 'credit-score-classifi...' (7 minutes ago). The main area shows a code editor with two code cells. The first cell contains imports for data science libraries. The second cell contains code to retrieve data from an S3 bucket.

```
[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()
```



# Demo

## Data Preprocessing

### 4.2 Helper Functions

Created following functions that will help in exploring, analysing & cleaning of the data

```
def get_column_details(df, column):  
    print("Details of", column, "column")  
  
    #DataType of column  
    print("\nDataType: ", df[column].dtype)  
  
    #Check if null values are present  
    count_null = df[column].isnull().sum()  
    if count_null==0:  
        print("\nThere are no null values")  
    elif count_null>0:  
        print("\nThere are ", count_null, " null values")  
  
    #Get Number of Unique Values  
    print("\nNumber of Unique Values: ", df[column].nunique())  
  
    #Get Distribution of Column  
    print("\nDistribution of column:\n")  
    print(df[column].value_counts())
```

```
def fill_missing_with_group_mode(df, groupby, column):  
    print("\nNo. of missing values before filling with group mode:", df[column].isnull().sum())  
  
    # Fill with local mode  
    mode_per_group = df.groupby(groupby)[column].transform(lambda x: x.mode().iat[0])  
    df[column] = df[column].fillna(mode_per_group)  
  
    print("\nNo. of missing values after filling with group mode:", df[column].isnull().sum())
```

```
#Method to clean categorical field  
  
def clean_categorical_field(df, groupby, column, replace_value=None):  
    print("\n-----")  
    print("\nCleaning steps ")  
  
    #Replace with np.nan  
    if replace_value!=None:  
        df[column] = df[column].replace(replace_value, np.nan)
```

# Demo

## Data Transformation

```
: #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])
```

```
: 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	...	Ci
0	1	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	
1	2	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	
2	3	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	
3	4	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	
4	5	23.0	12	19114.12	1824.843333	3.0	4.0	3.0	4.0	128	...	

5 rows × 24 columns

```
: #Split Input & Output Data
X = df_train.drop('Credit_Score',axis=1)
y = df_train['Credit_Score']
print(X.shape)
print(y.shape)
```

```
(100000, 23)
(100000,)
```

```
: #Normalize Data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

# Demo

## Modeling and Evaluation

```
# List of classifiers to test
classifiers = [
    ('Decision Tree', DecisionTreeClassifier()),
    ('Random Forest', RandomForestClassifier()),
    ('KNN', KNeighborsClassifier(n_neighbors=5)),
    ('Gaussian NB', GaussianNB()),
    ('XGB', xgb.XGBClassifier())
]

# Iterate over each classifier and evaluate performance
for clf_name, clf in classifiers:
    # Perform cross-validation
    scores = cross_val_score(clf, X_train, y_train, cv=5, scoring='accuracy')

    # Calculate average performance metrics
    avg_accuracy = scores.mean()
    avg_precision = cross_val_score(clf, X_train, y_train, cv=5, scoring='precision_macro').mean()
    avg_recall = cross_val_score(clf, X_train, y_train, cv=5, scoring='recall_macro').mean()

    # Print the performance metrics
    print(f'Classifier: {clf_name}')
    print(f'Average Accuracy: {avg_accuracy:.4f}')
    print(f'Average Precision: {avg_precision:.4f}')
    print(f'Average Recall: {avg_recall:.4f}')
    print('-----')
```

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

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

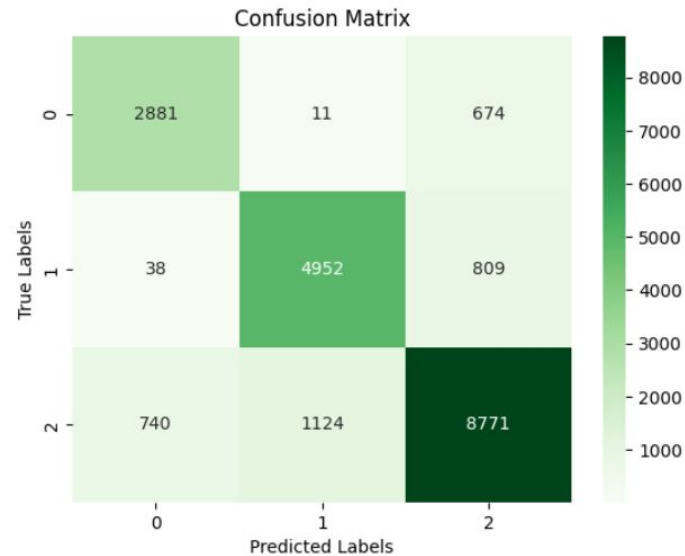
# Making predictions on the test set
y_pred = rf_classifier.predict(X_test)

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

# Demo- Results

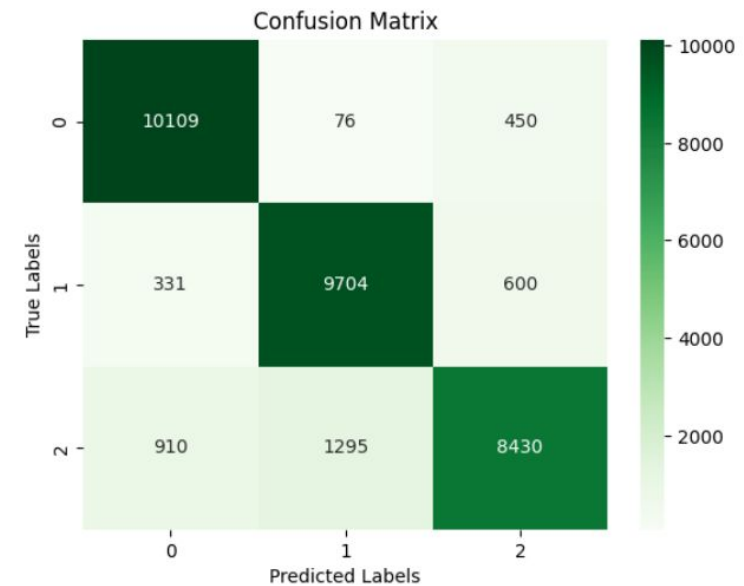
## Approach 1

Classification Report				
	precision	recall	f1-score	support
0	0.79	0.81	0.80	3566
1	0.81	0.85	0.83	5799
2	0.86	0.82	0.84	10635
accuracy			0.83	20000
macro avg	0.82	0.83	0.82	20000
weighted avg	0.83	0.83	0.83	20000



## Approach 2

Classification Report				
	precision	recall	f1-score	support
0	0.89	0.95	0.92	10635
1	0.88	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



# Lessons learned

---

- Initially, the model execution time on SageMaker was excessively long. We identified that the instance we used in SageMaker had a lower configuration. Once we updated it, the issue was resolved.
- We faced issues while cleaning the data, as there were numerous missing and erroneous values. To gain a better understanding of how to handle missing data, we referred to AWS SageMaker examples and tutorials.
- We have learned about various model evaluation metrics, such as precision, recall, F1-Score, and AUC. We also gained insights into AWS Cost Explorer and identified possible cost for each service that we have used.
- We have developed the model for predicting the credit score. As a next step, we will develop a GUI where businesses can enter the financial details of the user and determine the creditworthiness of the person.



# Thank you