**Title:** Fraud Detection Model for ABC Bank

**Name:** Prateek Sharma

**Date:** 03 November 2024

---

**Business Task:** The goal of this project is to build a machine learning model that helps ABC Bank identify and understand fraudulent credit transactions. The model will analyze transaction data to detect patterns that indicate fraud, enabling the bank to mitigate risks and protect customer accounts.

**Objective:** The objective is to develop a classification model capable of accurately distinguishing between legitimate and fraudulent transactions. This model will serve as a tool for the bank's fraud detection team to prioritize transactions that need further investigation.

**Dataset Source:** The dataset used for this project comes from, Kaggle's credit card fraud detection dataset. It contains transaction records, including various features that represent transaction details, user behavior, and other factors that can help identify fraudulent activity.

The dataset includes thousands of records with features like transaction amount, time of transaction, and anonymized transaction details (for privacy). The target variable, is_fraud, indicates whether a transaction is fraudulent (1) or not (0).

## Data Preparation and Cleaning:

In this project, we are analyzing credit card transaction data to build a model for detecting fraudulent transactions. The dataset was downloaded from Kaggle and organized in a structured folder system for ease of access. The analysis is being performed using Python due to its robust libraries for data manipulation and machine learning, which streamline the data preparation and model-building process.

**1) Import Necessary Libraries**

We begin by importing essential libraries that will assist us in data manipulation, visualization, and model building.

‘’’

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
```

**2) Load the Dataset**

The dataset is loaded from the specified directory into a Pandas DataFrame for further analysis.

```
df = pd.read_csv('Raw_Data/Raw_credit_card_transactions.csv')
```

**3) Initial Data Overview**

To understand the dataset's structure, we print the DataFrame's information and the first few rows.

```
print(df.info())

print(df.head())
```

**Output Overview:**

- The dataset contains 1,296,675 entries and 24 columns.
- Notable columns include trans_date_trans_time, amt, gender, and is_fraud.

**4) Drop Unnecessary Columns**

Certain columns do not contribute to our model's ability to predict fraud and can be dropped to reduce noise and complexity.

‘’’

df = df.drop(columns=['Unnamed: 0', 'trans_num', 'cc_num', 'first', 'last', 'street', 'city', 'state', 'zip', 'unix_time'])

‘’’

**5) Handle Missing Values**

The merch_zipcode column has missing values. We fill these with the most common value (mode) to maintain data integrity.

‘’’

df['merch_zipcode'].fillna(df['merch_zipcode'].mode()[0], inplace=True)

‘’’

**6) Convert Data Types**

We convert the trans_date_trans_time column to a datetime object to enable time-based feature extraction. The dob column is also converted to datetime, handling any errors gracefully.

‘’’

df['trans_date_trans_time'] = pd.to_datetime(df['trans_date_trans_time'])

df['dob'] = pd.to_datetime(df['dob'], errors='coerce')  # Convert dob to datetime and handle errors

‘’’

**7) Feature Engineering**

We create a new feature, age, derived from the dob column, which reflects the age of the cardholder at the time of the transaction.

‘’’

df['age'] = (df['trans_date_trans_time'].dt.year - df['dob'].dt.year).astype(int)

‘’’

**8) Drop Original Date Columns**

After extracting the age, we drop the original datetime and date of birth columns, as they are no longer necessary.

‘’’

df = df.drop(columns=['trans_date_trans_time', 'dob'])

‘’’

**9) Drop High-Cardinality Columns**

To streamline the model and reduce dimensionality, we drop high-cardinality categorical columns that may not contribute significantly to model performance.

‘’’

df = df.drop(columns=['merchant', 'job'])

‘’’

**10) Encode Essential Categorical Variables**

One-hot encoding is applied to the remaining categorical columns (category and gender) to convert them into a numerical format suitable for modeling.

‘’’

df = pd.get_dummies(df, columns=['category', 'gender'], drop_first=True)

‘’’

**11) Convert Remaining Columns to Numeric**

We ensure all columns in the DataFrame are numeric, converting any remaining object types.

‘’’

df = df.apply(pd.to_numeric, errors='coerce')

‘’’

**12) Check for Non-Numeric Columns**

We verify that no non-numeric columns remain after the previous steps.

‘’’

```
X = df.drop(columns=['is_fraud'])

y = df['is_fraud']

non_numeric_cols = X.select_dtypes(exclude=['number']).columns

print("Non-numeric columns:", non_numeric_cols)  # Should be empty
```

**13) Drop Any Remaining Non-Numeric Columns**

If there are any non-numeric columns after the checks, we drop them to ensure our features are numeric.

```
if len(non_numeric_cols) > 0:
   X = X.drop(columns=non_numeric_cols)
   print(f"Dropped columns: {non_numeric_cols}")
```

## Model Building

**Why Logistic Regression?** Logistic regression is a statistical method widely used for binary classification problems. It estimates the probability of a binary response based on one or more predictor variables. In this case, it's suitable because we are trying to predict the likelihood of fraudulent transactions (1) versus non-fraudulent transactions (0). Logistic regression provides a straightforward interpretation of the model coefficients, making it easier to understand the impact of various features on the outcome.

**1) Description of the Training and Test Split**

The data was initially split into subsets, with 80% allocated for training and the remaining 20% for testing to manage memory constraints. After that, the training set was further split to create training and testing datasets, where 30% of the original subset was used for testing. This approach ensures that we have a robust evaluation of the model's performance while accommodating memory limitations.

```
# Step 6: Train-test split

X_subset, _, y_subset, _ = train_test_split(X, y, test_size=0.8, random_state=42)
```

# Train-test split on subset due to memory constraints

X_train, X_test, y_train, y_test = train_test_split(X_subset, y_subset, test_size=0.3, random_state=42)

‘’’

**2) Training the Model**

A logistic regression model was instantiated with a maximum of 1000 iterations to ensure convergence. The model was then fitted to the training data, allowing it to learn the relationship between the features and the target variable (is_fraud).

‘’’

# Train Logistic Regression Model

model = LogisticRegression(max_iter=1000, random_state=42)

model.fit(X_train, y_train)

‘’’

## Evaluation Metrics

**1) Confusion Matrix**

‘’’

# Step 7: Evaluate Model

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

‘’’

This matrix provides a summary of the prediction results:

- **True Positives (TP)**: 0 (correctly predicted fraud cases)
- **True Negatives (TN)**: 77292 (correctly predicted non-fraud cases)
- **False Positives (FP)**: 50 (incorrectly predicted fraud cases)
- **False Negatives (FN)**: 459 (fraud cases that were predicted as non-fraud)

From this matrix, we can see that the model successfully identified the vast majority of non-fraudulent transactions but failed to detect fraudulent transactions (0 true positives). This suggests that the model may have a bias towards predicting the majority class.

**2) Other Key Metrics**

‘’’

print("\nClassification Report:\n", classification_report(y_test, y_pred))

print("\nROC AUC Score:", roc_auc_score(y_test, y_prob))

‘’’

The classification report provides the following metrics:

- **Accuracy**: 0.99
- **Precision** (for class 1): 0.00
- **Recall** (for class 1): 0.00
- **F1-Score** (for class 1): 0.00
- **Macro Avg**: Precision and recall averaged across classes (0.50).
- **Weighted Avg**: Precision and recall weighted by the number of instances in each class (0.99).

The ROC AUC Score is approximately **0.8189**, indicating that the model has a fair ability to distinguish between the positive and negative classes.

## Observations about the Model’s Performance

- The model exhibits high accuracy (99%), but this is misleading due to the class imbalance, where the number of non-fraudulent transactions far exceeds that of fraudulent transactions.
- The precision and recall for fraudulent transactions are both zero, indicating that while the model is good at identifying non-fraudulent transactions, it fails to predict any fraudulent cases effectively.
- The trade-off here is between sensitivity (the model's ability to correctly identify fraud) and specificity (the ability to correctly identify non-fraud). A potential next step could involve exploring different algorithms, adjusting the class weights, or using techniques such as oversampling the minority class to improve the model's performance on the fraud detection task.

## Future Steps

**1) Address Class Imbalance:** Since there are many more non-fraudulent transactions than fraudulent ones, try techniques like oversampling the fraud cases or undersampling the non-fraud cases. This will help the model learn better from the minority class (fraud cases).

**2) Experiment with Different Algorithms:** Logistic regression is just one option. You can try other models like Random Forest, Decision Trees, or Support Vector Machines. Different algorithms might capture patterns in the data better and improve your ability to detect fraud.

**3) Feature Selection and Engineering:** Review the features you are currently using and consider adding new ones or removing less important ones. Look for additional relevant variables that could improve model performance, such as transaction frequency or average transaction amount over time. This can help the model capture more significant patterns related to fraud detection.