Capstone Project
-
Credit Card Fraud Detection

Submitted by:

Lokesh Rajendran

AM. Prabhu Sankar

Date: 09/05/2023

Content

- Introduction
- Problem statement
- Objective
- Key Insights
- Appendix
- About the data
- Methodology
- Assumptions

Problem Statement: Part - I

Apprehending fraudsters is outside the scope of banking operations. So, the problem that we need to solve is to not stop the fraudsters but to identify and stop the fraudulent transactions that they are making. Several government entities are involved in trying to prevent or stop this crime.

Conduct the root cause analysis with the bank's PoC to understand the processes/structures that are already in place to deal with anomalous transactions. As you already know, the number of fraudulent transactions is rising day by day, and no proper action is being taken when an unauthorised transaction is made.

Problem Statement: Part - 2

In the banking industry, detecting credit card fraud using machine learning is not just a trend; it is a necessity for the banks, as they need to put proactive monitoring and fraud prevention mechanisms in place. Machine learning helps these institutions reduce time-consuming manual reviews, costly chargebacks and fees, and denial of legitimate transactions.

Suppose you are part of the analytics team working on a fraud detection model and its cost-benefit analysis. You need to develop a machine learning model to detect fraudulent transactions based on the historical transactional data of customers with a pool of merchants. You can learn more about transactional data and the creation of historical variables from the link attached here. You may find this helpful in the capstone project while building the fraud detection model. Based on your understanding of the model, you have to analyse the business impact of these fraudulent transactions and recommend the optimal ways that the bank can adopt to mitigate the fraud risks.

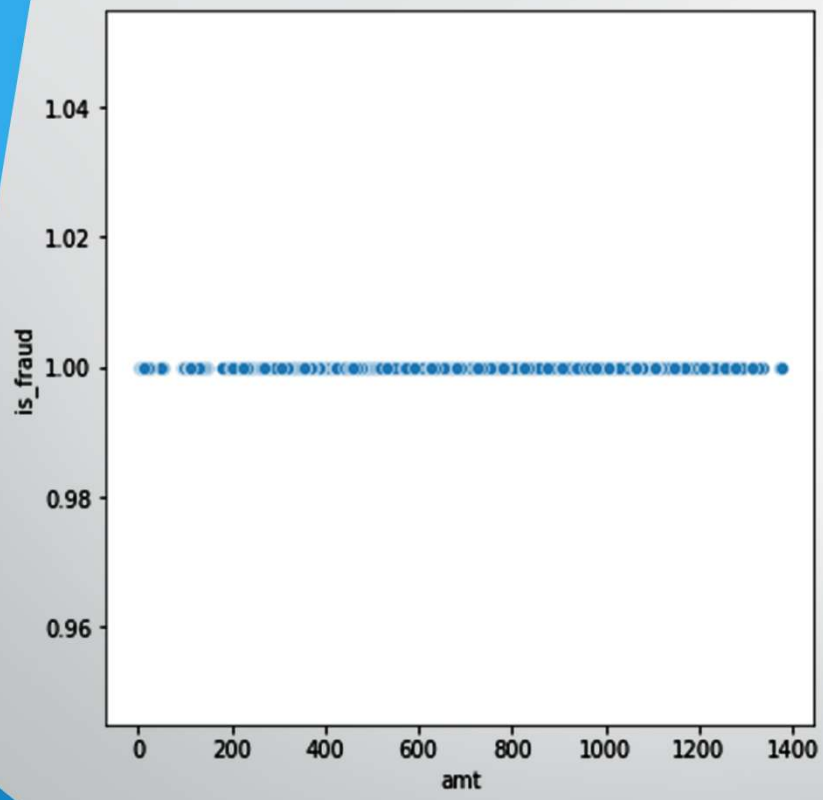
Solution Approach

- Data understanding and Manipulation
- Exploratory data analytics (EDA)
- Train/Test Data Splitting
- Model Building (Multiple)
- Model Evaluation
- Model Evaluation using fraudtest.csv
- Cost Benefit Analysis

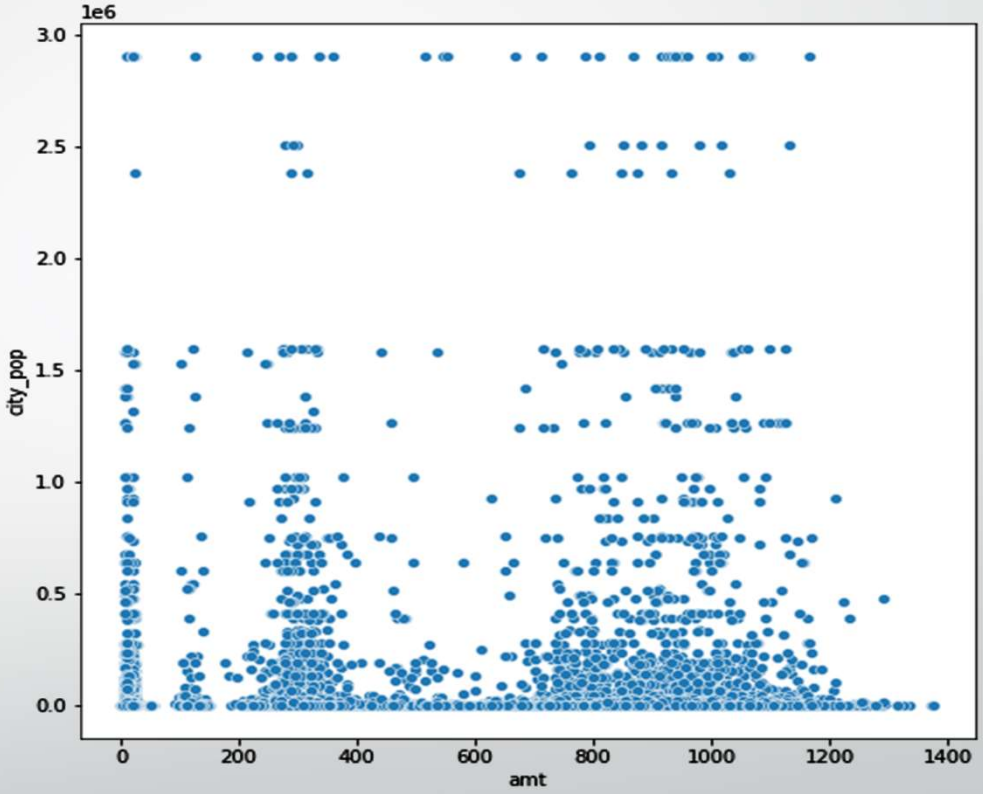
Data understanding and Manipulation

- Removed unwanted columns
- Modified data types as applicable
- Calculated distance based on latitude and longitude
- Created dummy variables

Exploratory data analytics (EDA)

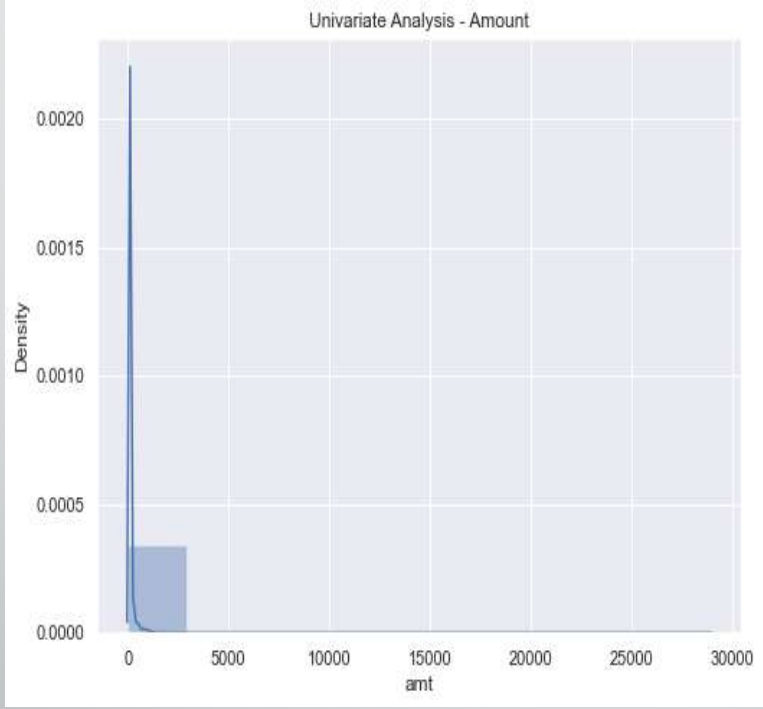


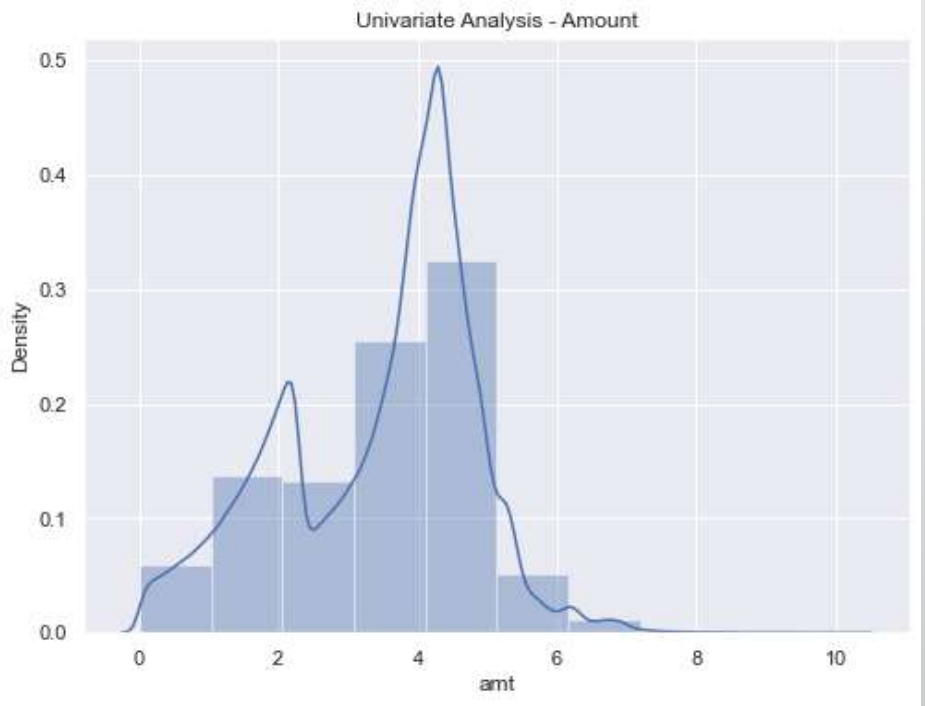
Fraud occurs at every turn.



Between the population of a city and the amount of spending, fraud does not clearly correlate..

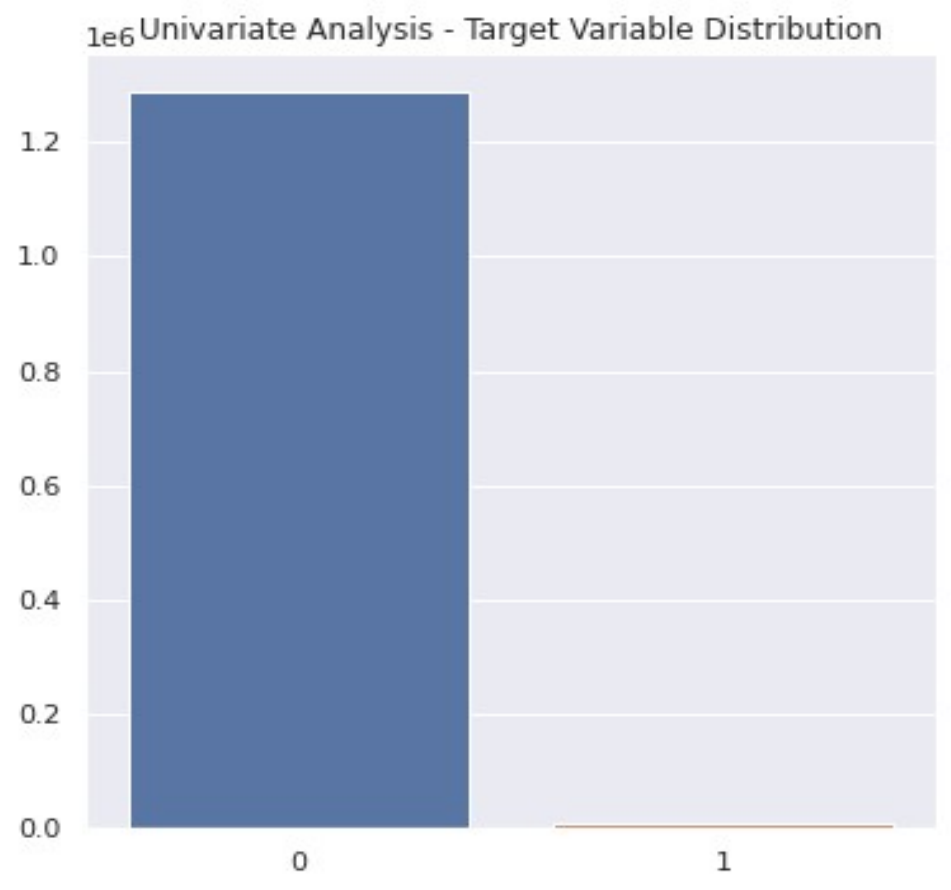
Exploratory data analytics (EDA)





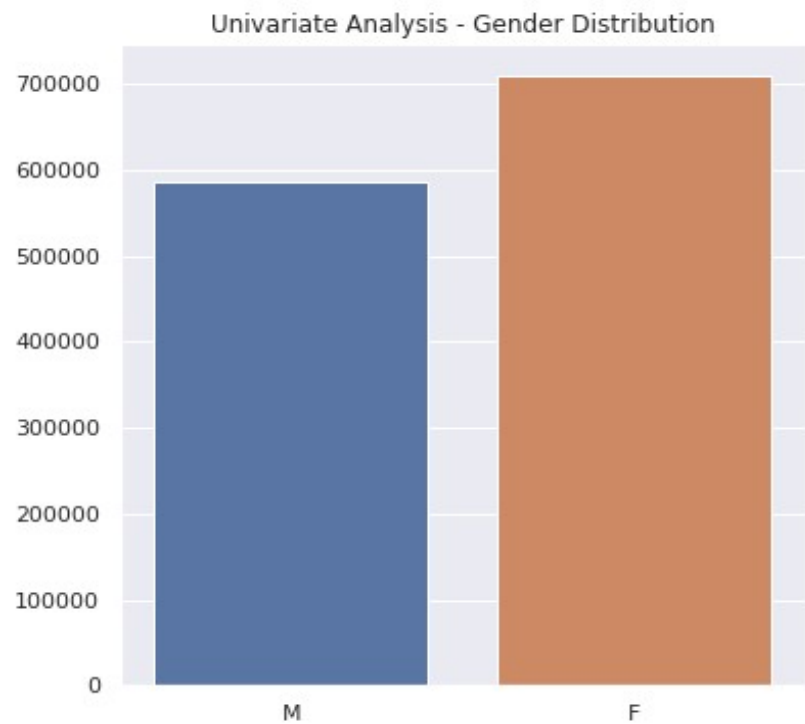
The amount column is more regularly distributed after addressing skewness.

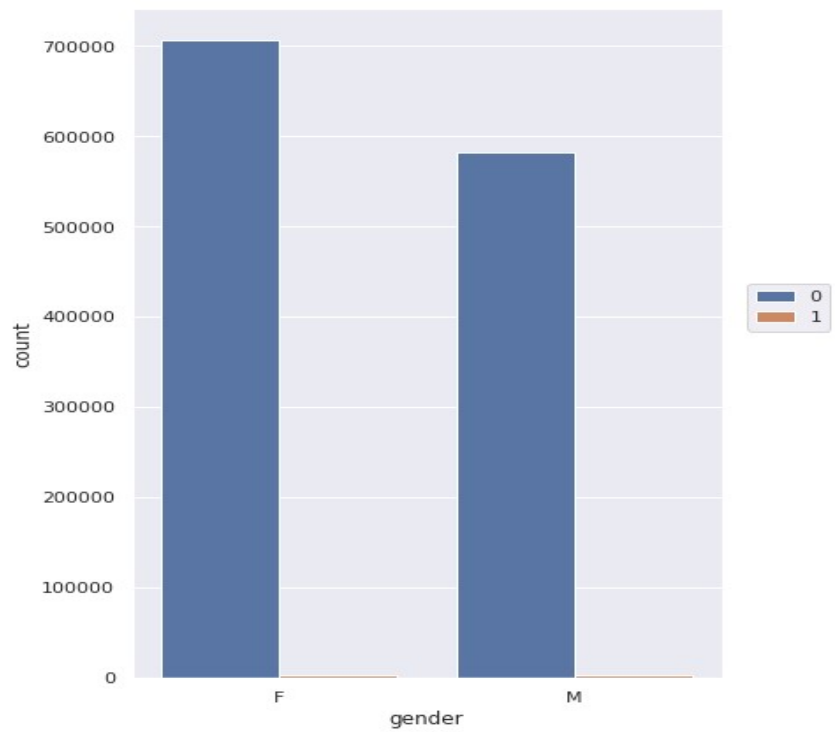
Exploratory data analytics (EDA)



The data show an obvious imbalance in the target variable which should be treated by sampling methods.

Exploratory data analytics (EDA)

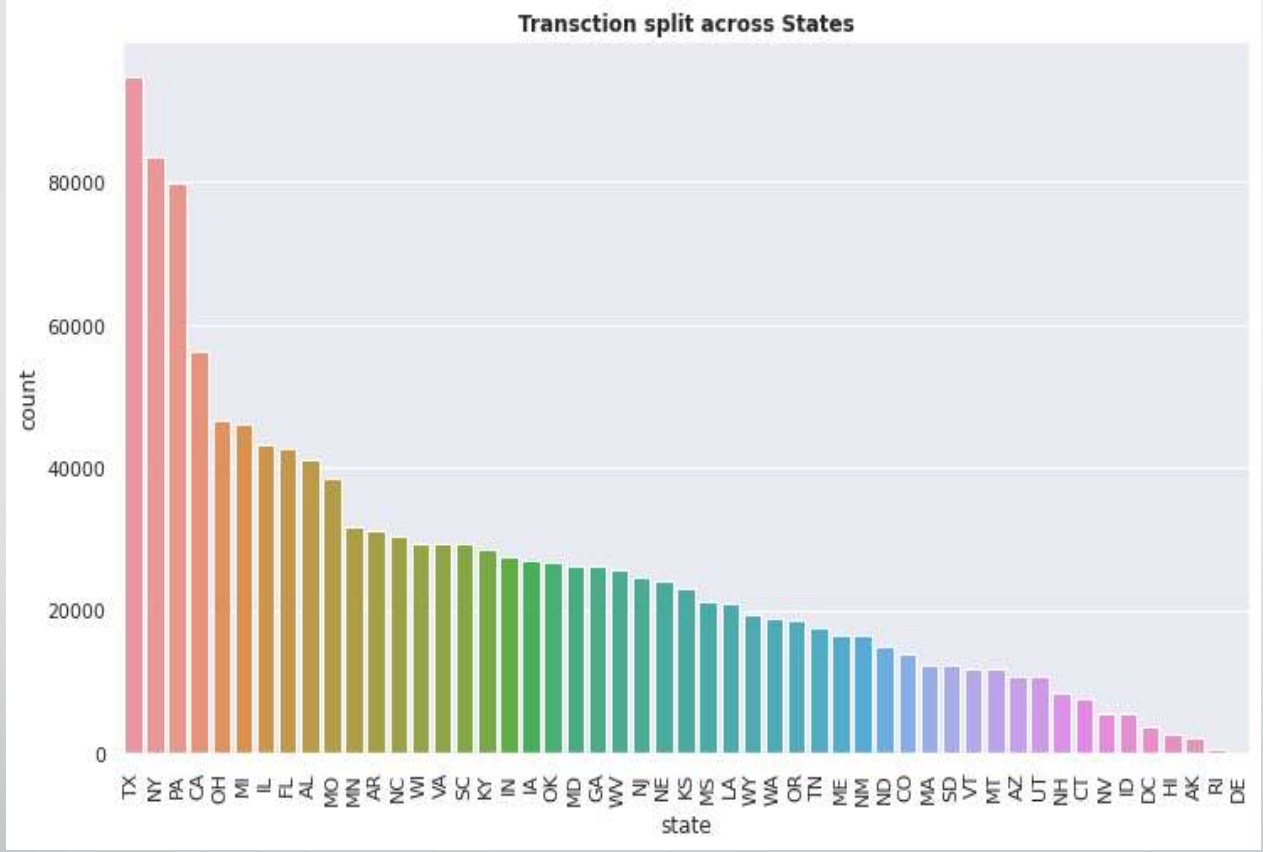




Women engage in more transactions than men.

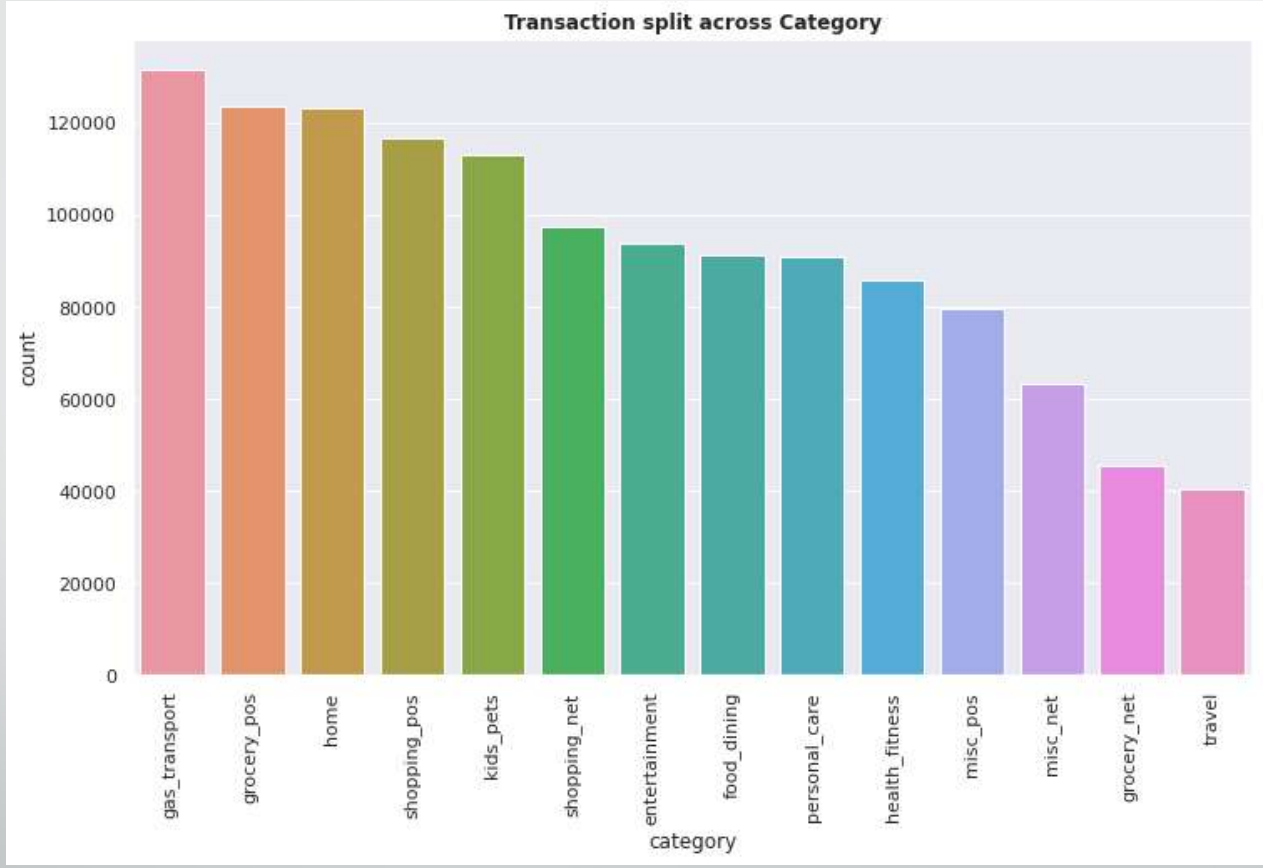
Imbalance between fraud and nonfraud for gender

Exploratory data analytics (EDA)



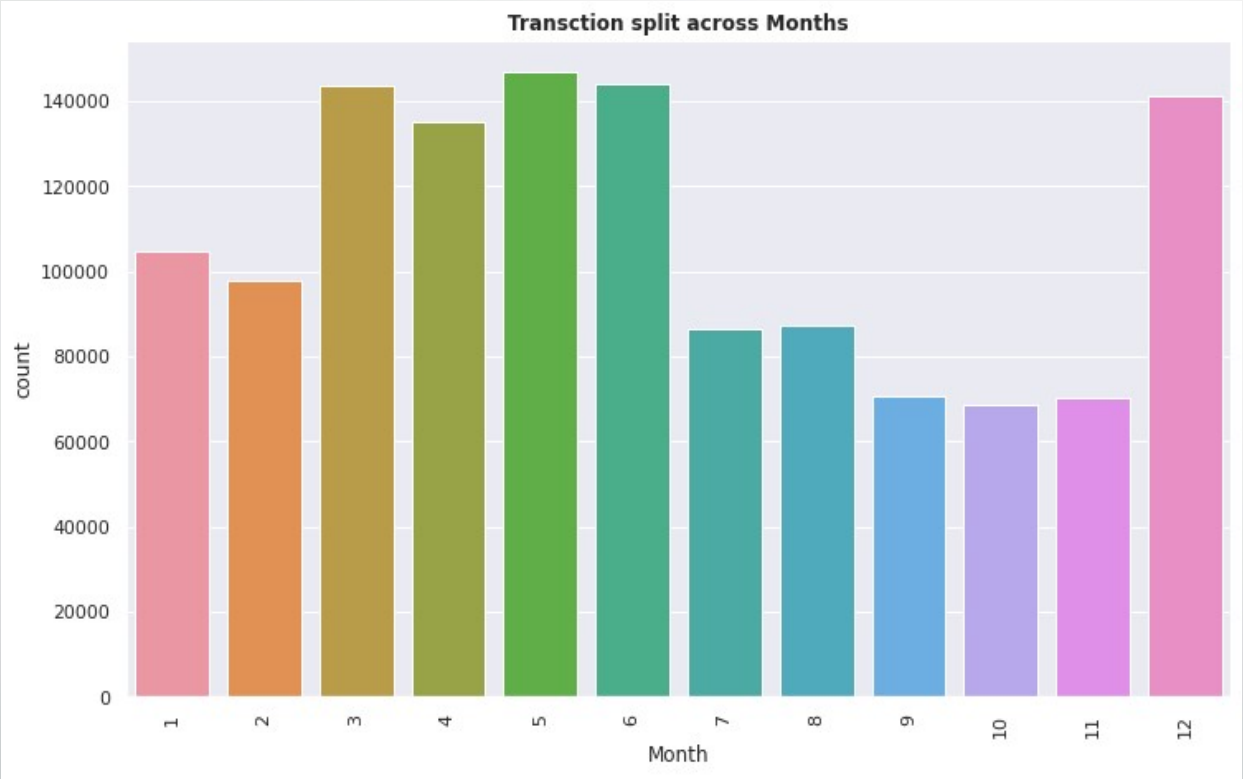
Most transactions take place in Texas.

Exploratory data analytics (EDA)



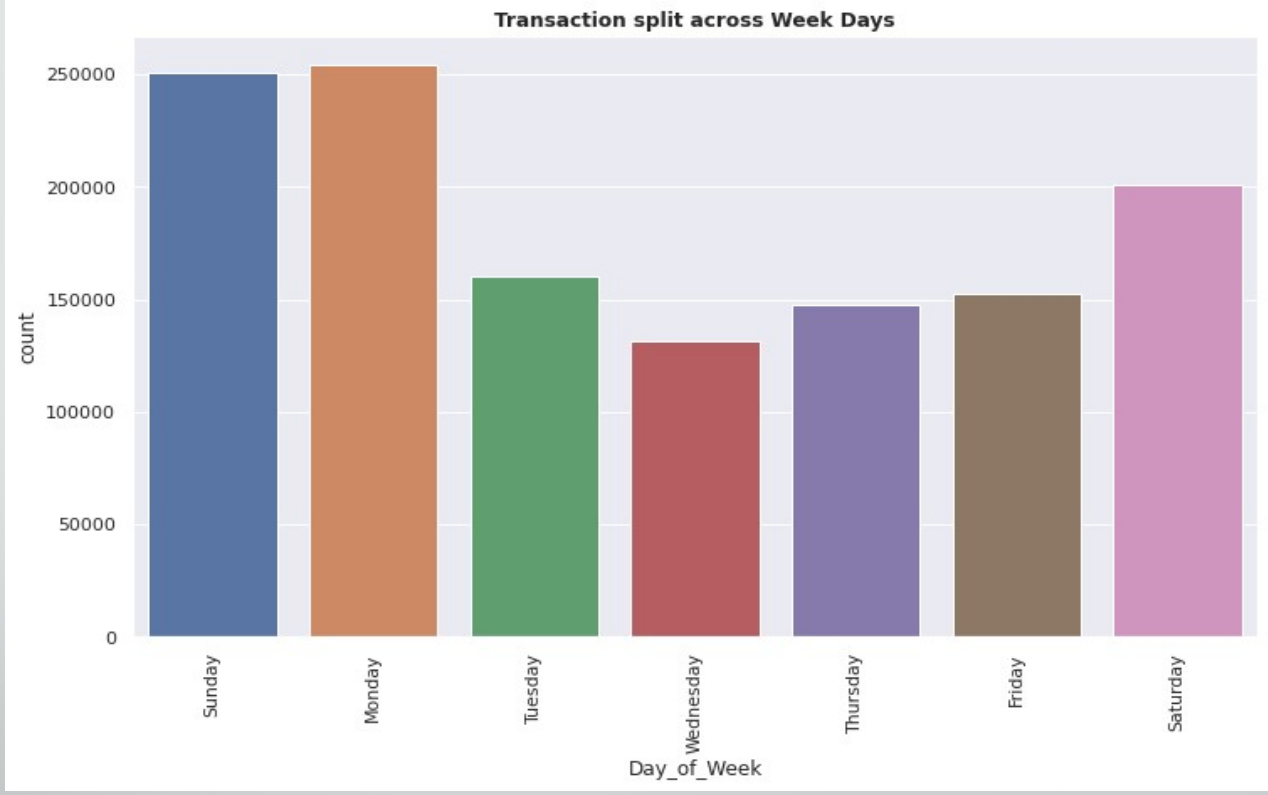
The largest number of transactions involves the transport of gas.

Exploratory data analytics (EDA)



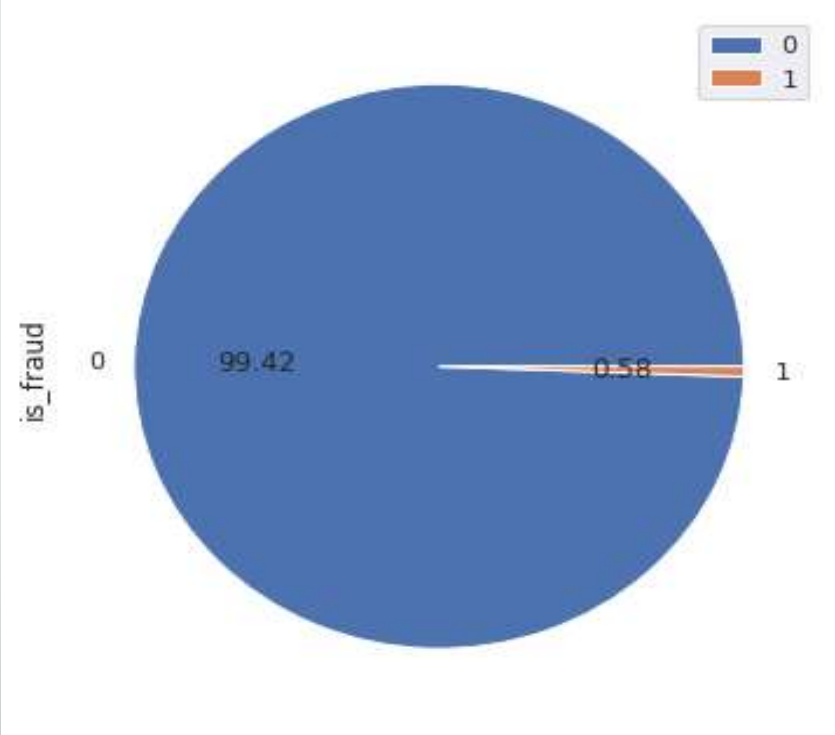
The highest number of deals occurred in May.

Exploratory data analytics (EDA)



The highest number of transactions are on Monday.

Data Imbalance



The data shows clear imbalance in the target variable (is_fraud) which needs to be handled by sampling methods

Train/Test Data Splitting

For model development, we must scale and divide the data into train and test data sets:

- **Train data will be split into 80% for training and 20% for testing**
- **MinMaxScaler() is used for amount city population and distance**

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.20, random_state=42)
```

```
[ ]  #Rescaling
     scaler = MinMaxScaler()
```

```
[ ]  # Apply scaler()
     num_vars = ["amt","city_pop","Dist"]

     X_train[num_vars] = scaler.fit_transform(X_train[num_vars])
```

Model Building and Evaluation

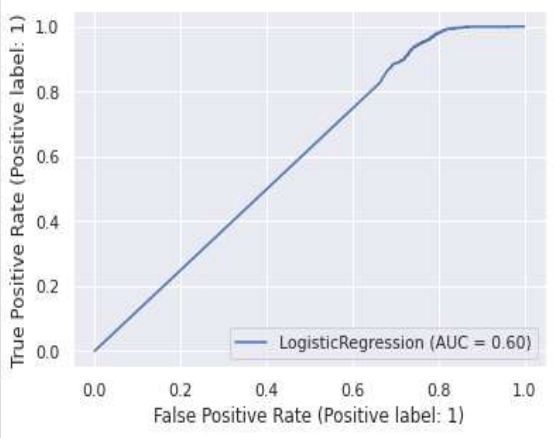
Model building done using sampling techniques and unsampled data.

Several models are constructed and selected according to AUC score and sensitivity.

For every model we have created three variants

- **Model with Unsampled Data**
- **Model using SMOTE**
- **Model using ADASYN**

Model Building – Logistic Regression



Unsampled

Train set performance:
0.0
[[1031149 205]
[5986 0]]

Test set performance:
1.0
[[9141 248674]
[0 1520]]

SMOTE

Train set performance:
0.735611632863207
[[763470 267884]
[272678 758676]]

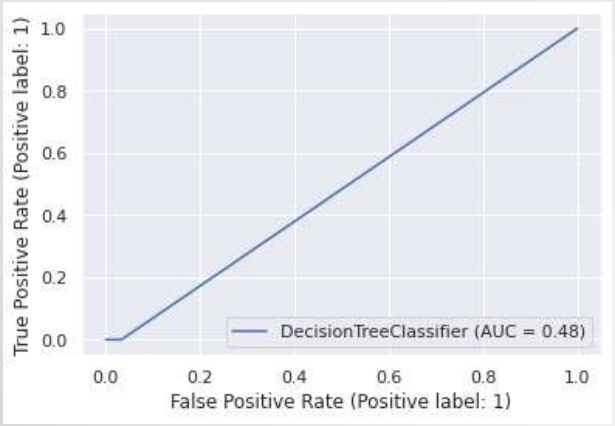
Test set performance:
1.0
[[9094 248721]
[0 1520]]

ADASYN

Train set performance:
0.676375464143407
[[742424 288930]
[333809 697661]]

Test set performance:
0.9993421052631579
[[15595 242220]
[1 1519]]

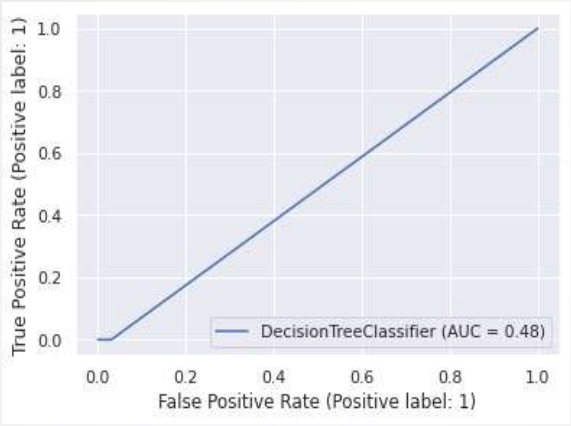
Model Building – Decision Tree Classifier



Unsampled

Train set performance:
0.6351486802539258
[[1030575 779]
[2184 3802]]

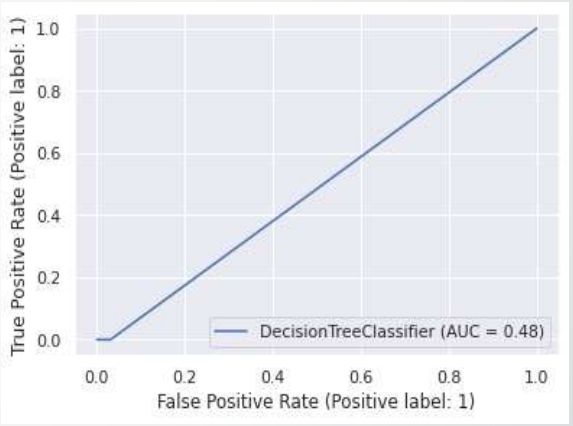
Test set performance:
0.0
[[257656 159]
[1520 0]]



SMOTE

Train set performance:
0.95934082768865
[[972962 58392]
[41934 989420]]

Test set performance:
0.0
[[255647 2168]
[1520 0]]

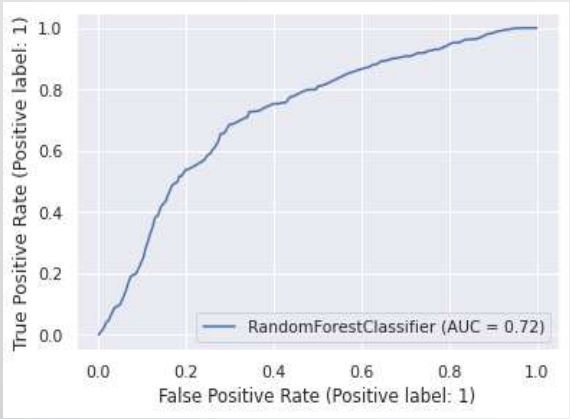


ADASYN

Train set performance:
0.9492384654909983
[[963488 67866]
[52359 979111]]

Test set performance:
0.0
[[255715 2100]
[1520 0]]

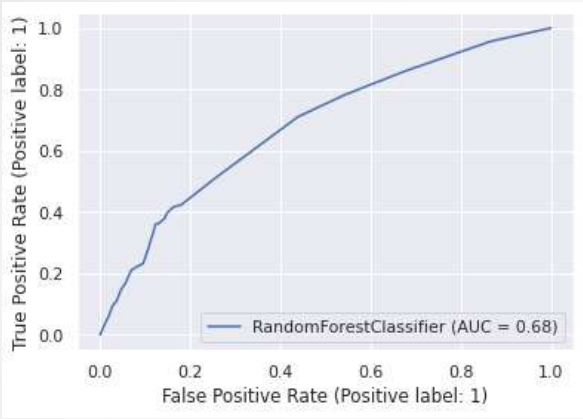
Model Building – Random Forest



Unsampled

Train set performance:
0.2891747410624791
[[1031144 210]
[4255 1731]]

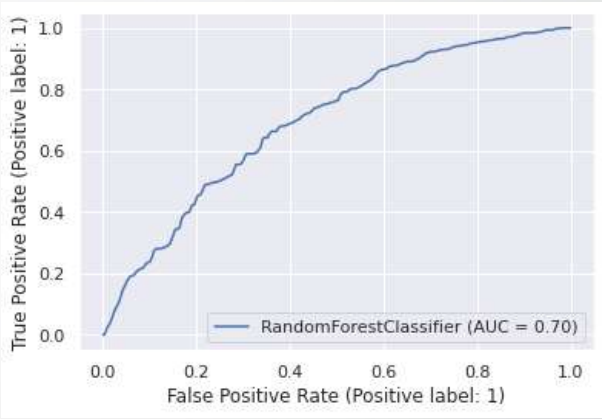
Test set performance:
0.0
[[257761 54]
[1520 0]]



SMOTE

Train set performance:
0.8854108288715611
[[1005539 25815]
[118182 913172]]

Test set performance:
0.8848684210526315
[[90332 167483]
[175 1345]]

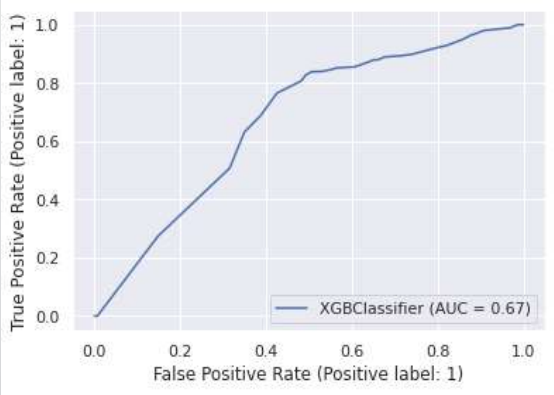


ADASYN

Train set performance:
0.8603129514188488
[[990398 40956]
[144083 887387]]

Test set performance:
0.8289473684210527
[[110763 147052]
[260 1260]]

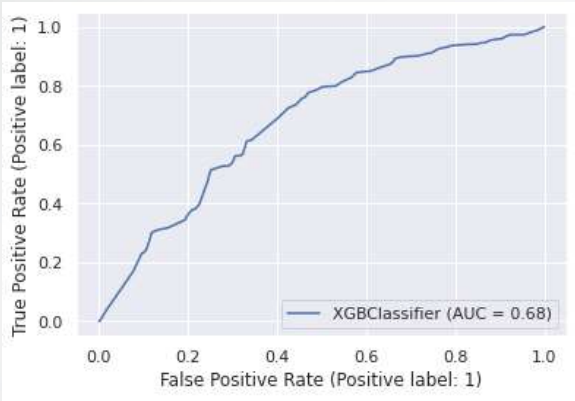
Model Building – XGBoost



Unsampled

Train set performance:
0.7053123955897094
[[1030694 660]
[1764 4222]]

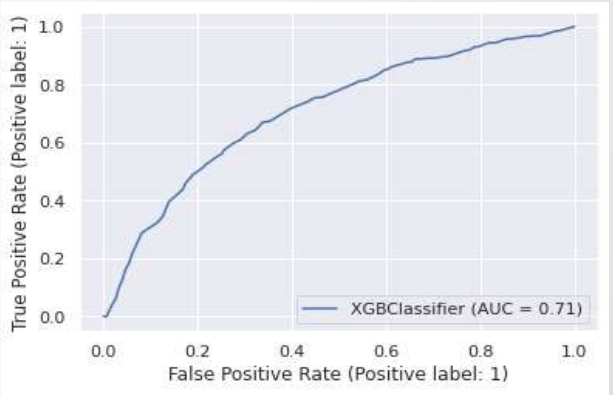
Test set performance:
0.0
[[257428 387]
[1520 0]]



SMOTE

Train set performance:
0.982996139056037
[[998819 32535]
[17537 1013817]]

Test set performance:
0.2315789473684210
5
[[232794 25021]
[1168 352]]



ADASYN

Train set performance:
0.9856660882042134
[[991152 40202]
[14785 1016685]]

Test set performance:
0.0
[[257066 749]
[1520 0]]

Model Building and Evaluation Summary

	Model	Recall on Train	Recall on Test	AUC Score
2	Logistic Regression - ADASYN	0.67	0.99	0.84
1	Logistic Regression - SMOTE	0.73	1.00	0.82
6	Random Forest - Unsampled	0.28	0.00	0.72
11	XGBoost - ADASYN	0.98	0.00	0.71
8	Random Forest - ADASYN	0.86	0.82	0.70
7	Random Forest - SMOTE	0.88	0.88	0.68
10	XGBoost - SMOTE	0.98	0.23	0.68
9	XGBoost - Unsampled	0.70	0.00	0.67
0	Logistic Regression - Unsampled	0.00	1.00	0.60
3	Decision Trees - Unsampled	0.63	0.00	0.48
4	Decision Trees - SMOTE	0.95	0.00	0.48
5	Decision Trees - ADASYN	0.94	0.00	0.48

Model Building using fraudtest.csv

Model	Recall
Logistic Regression – SMOTE	1
Logistic Regression – ADASYN	0.998
Decision Trees – SMOTE	0
Decision Trees – ADASYN	0
Random Forest – SMOTE	0.204
Random Forest – ADASYN	0.149
XGBoost - SMOTE	0.226
XGBoost - ADASYN	0

Model Building using fraudtest.csv

Based on the area under the 0.83 curve and sensitivity of 99.86% Sensitivity - Logistic regression ADASYN is best at predicting fraudulent transactions.

Cost Benefit Analysis

Part I: Analyse the dataset and find the following figures:

1. Average number of transactions per month

   77183.08

2. Average number of fraudulent transactions per month

   402.12

3. Average amount per fraudulent transaction

   530.66

Cost Benefit Analysis

Part II: Compare the cost incurred per month by the bank before and after the model deployment:

1. Cost incurred per month before the model was deployed = Average amount per fraudulent transaction * Average number of fraudulent transactions per month

   $ 213392.22

Cost Benefit Analysis

Part II: Compare the cost incurred per month by the bank before and after the model deployment:

Let TF be the average number of transactions per month detected as fraudulent by the model and let the cost of providing customer executive support per fraudulent transaction detected by the model = $1.5

2. Total cost of providing customer support per month for fraudulent transactions detected by the model = 1.5 * TF.

   $ 607.68

3. Average number of transactions per month that are fraudulent but not detected by the model (FN)

   16.25

Cost Benefit Analysis

Part II: Compare the cost incurred per month by the bank before and after the model deployment:

Let FN be the average number of transactions per month that are fraudulent but not detected by the model

4. Cost incurred due to these fraudulent transactions left undetected by the model

   = Average amount per fraudulent transaction * FN

   $ 8623.25

5. Therefore, the cost incurred per month after the model is built and deployed

   = 1.5*TF + Average amount per fraudulent transaction * FN

   $ 9230.94

6. Final savings = Cost incurred before - Cost incurred after.

   $ 204165.8


THANK YOU