Credit Card Fraud Detection

AUTHOR: SRIKUMAR PEYYALA

COURSE:- Executive Programme in Data Science September 2021

CONTENTS

OBJECTIVE

To develop a machine learning model to detect fraudulent transactions based on the historical transactional data of customers with a pool of merchants.

Background

- Credit card fraud is an inclusive term for fraud committed using a payment card, such as a credit card or debit card.
- The purpose may be to obtain goods or services, or to make payment to another account which is controlled by a criminal.
- In the banking industry, detecting credit card fraud using machine learning is not just a trend; it is a necessity for 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.

Understanding and Defining Fraud

Credit card fraud is any dishonest act or behaviour to obtain information without the proper authorisation of the account holder for financial gain. Among the different ways of committing fraud, skimming is the most common one. Skimming is a method used for duplicating information located on the magnetic stripe of the card. Apart from this, other ways of making fraudulent transactions are as follows:

- Manipulation or alteration of genuine cards
- Creation of counterfeit cards
- Stolen or lost credit cards
- Fraudulent telemarketing

Data

- The shape of the Train dataset is (1296675, 23)
- The shape of the Test dataset is (555719, 23)
- Merging both the data set .
- The train and test data obtained from kaggle are concatenated with credit_train on top of credit_test for further operations.
- The shape of the combined dataset is (**1852394, 22**)

Data

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	state	zip
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	Moravian Falls	NC	28654
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	Orient	WA	99160
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	Malad City	ID	83252
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	Boulder	MT	59632
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	Doe Hill	VA	24433

Data Cleaning

- Dropped the features which are not useful ((['**first**', '**last**','**dob**','**unix_time**','**job**','**state**','**street**'])
- Split feature " **trans_date_trans_time**" to **'Transaction_date'** & **'Transaction_Time'**.
- Data Check: No Missing Values but the present of **Outliers**.

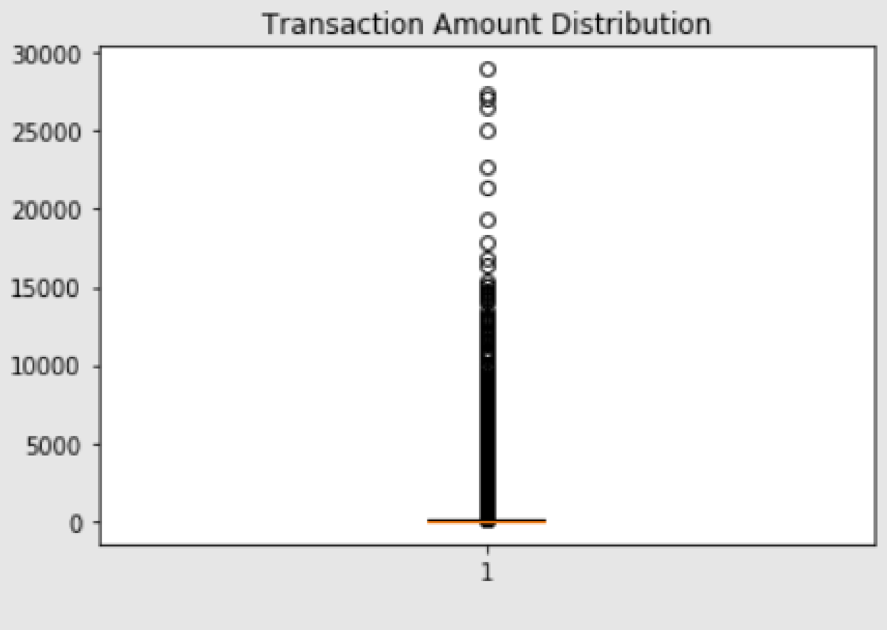
- 693 unique merchants classified as [high, medium, low] risk merchants based on number of fraudulent transactions.
- The unique values in categories are 14.
- The cities are binned as [high, medium, low] risk cities.
- Fraudulent transactions time ranges are:
  - 10 pm - 4 am   8169
  - 4 pm - 10 pm   606
  - 10 am - 4 pm   489
  - 4 am - 10 am   387

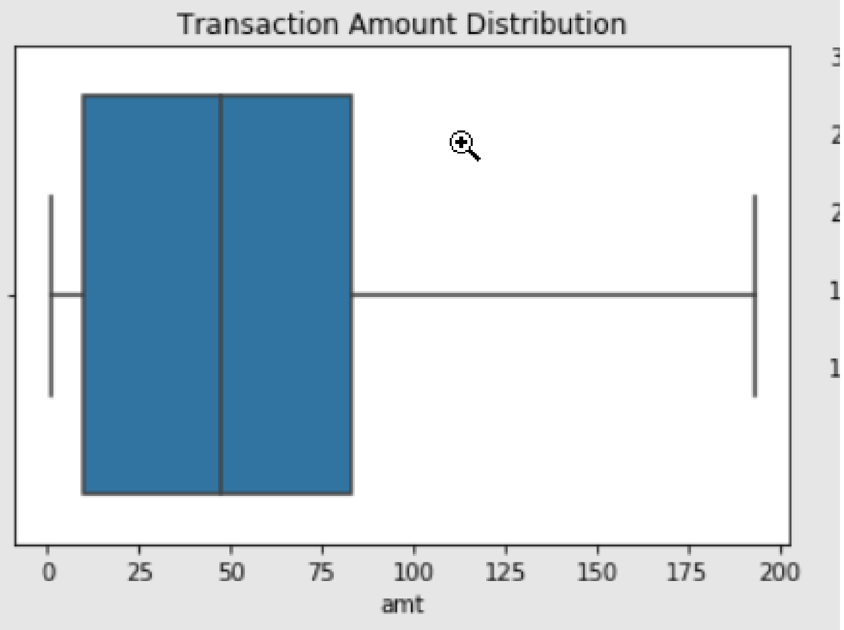
Exploratory Data Analysis

- The dataset is highly imbalanced with just 0.52% fraudulent transactions.

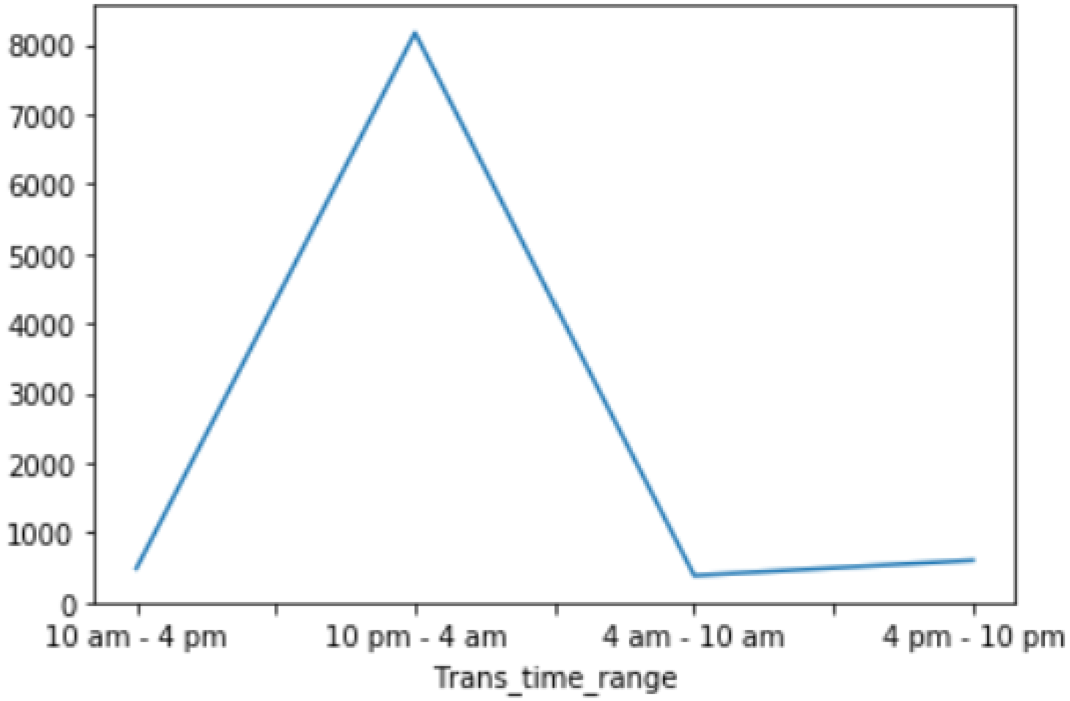
```
0    99.478999
1     0.521001
```

- The plots of transaction amount and City Population suggest large number of outliers in both the features.
- These will be treated using the appropriate methods below



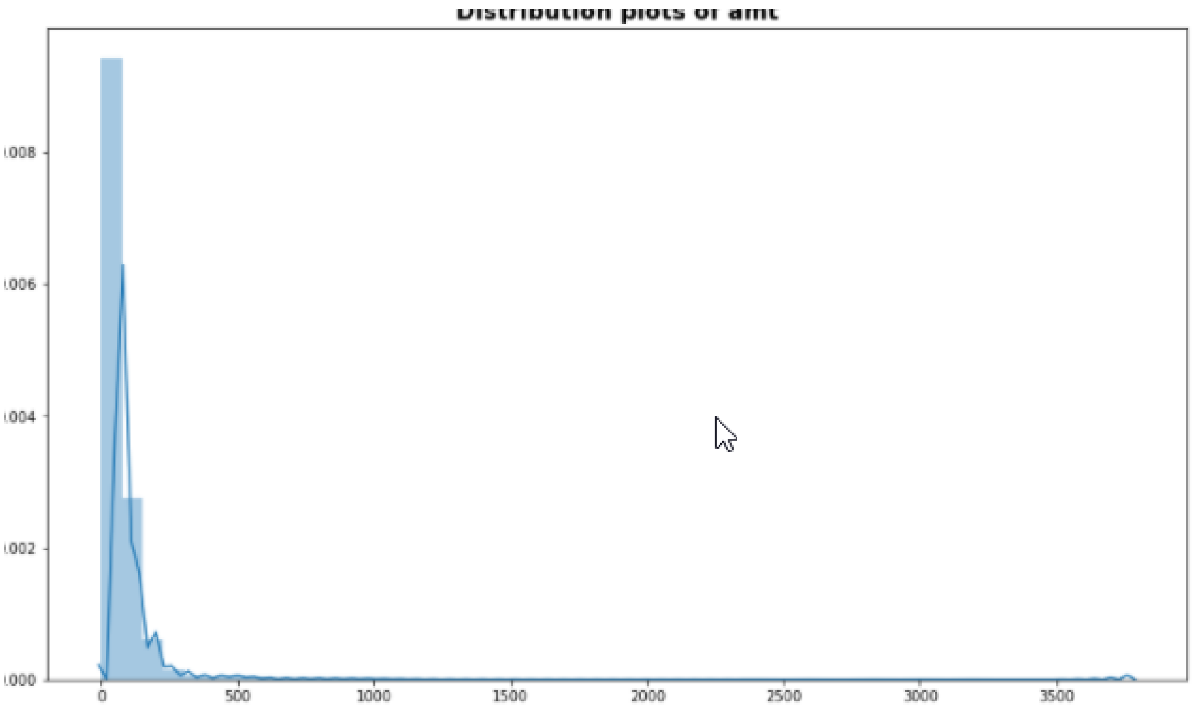


- Outlier Treatment of feature ‘amt’

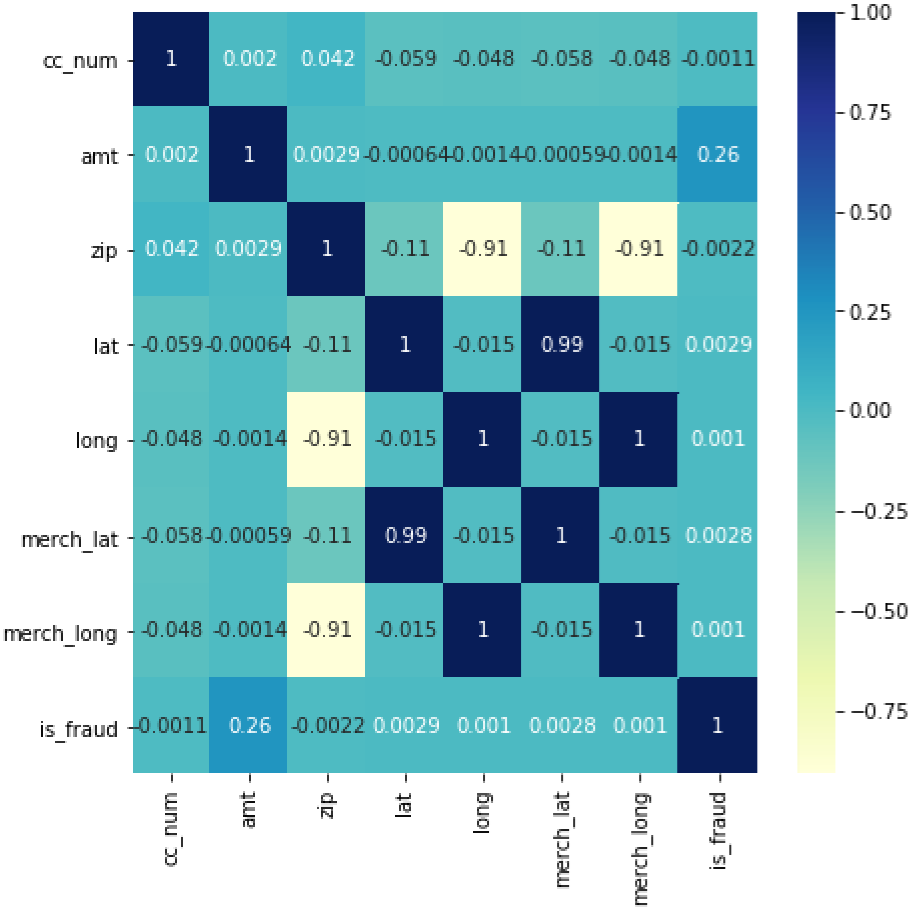


- Fraudulent Transaction Hours.

```
cc_num          2.851074
amt            11.635161
zip             0.078950
lat            -0.191999
long           -1.146919
merch_lat      -0.188097
merch_long     -1.143933
is_fraud       13.745675
dtype: float64
```

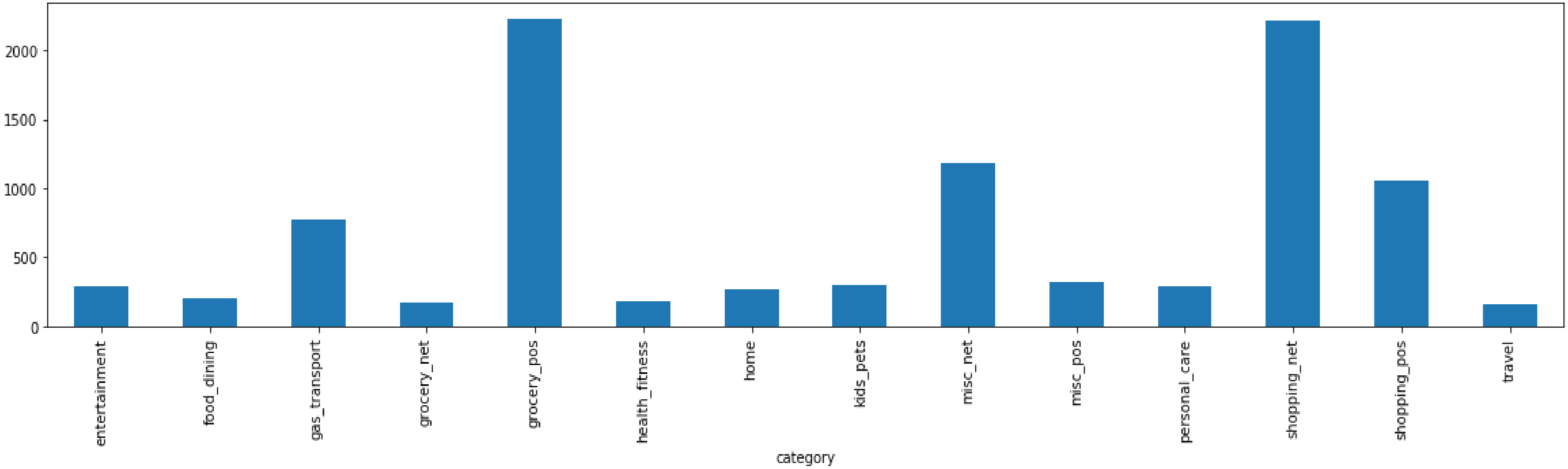


- All the Numerical Features were left skewed



Correlation Matrix

The customers Location Coordinates are highly correlated with the Merchants



CATEGORY OF PURCHASES

THE HIGHEST FRAUDULENT TRANSACTIONS HAPPENED **GROCERY POS** AND **SHOPPING_NETAMONG 14** CATEGORIES

THE BEST MODEL – Decision Tree with Hyperparameter Tuning

8 CLASSIFICATIONS MODELS DEVELOPED

1. Baseline Linear Model- Logistic Regression With-out balancing & With balancing data.

2. Decision Trees- With and Without Hyperparameter tuning.

3. Random Forests- With and Without Hyperparameter Tuning.

```
# Instantiating Stratified K-Fold Cross Validation
from sklearn.model_selection import StratifiedKFold
strat_k=StratifiedKFold(n_splits=3,random_state=100)

#Instantiating the Decision Tree Classfier

dth1 = DecisionTreeClassifier()

# Defining parameters for random search

params={
        'max_depth':[5,6,8,12],
        'min_samples_leaf':[10,12,15],

        'min_samples_split':[200,300,500],
        'criterion':['gini'],
        'class_weight':['balanced']
}
```

```
#Instantiating random search CV to with the parameters defined above
from sklearn.model_selection import RandomizedSearchCV
rand_search_dth=RandomizedSearchCV(estimator=dth1,param_distributions=params,cv=strat_k,random_state=100,verbose=True)

#Fitting RandomizedSearchCV on X_train and Y_Train

rand_search_dth.fit(X_train,y_train)
```

```python
#Instantiating random search CV to with the parameters defined above
from sklearn.model_selection import RandomizedSearchCV
rand_search_dth=RandomizedSearchCV(estimator=dth1,param_distributions=params,cv=strat_k,random_state=100,verbose=True)

#Fitting RandomizedSearchCV on X_train and Y_Train

rand_search_dth.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed: 46.0min finished
```

```
RandomizedSearchCV(cv=StratifiedKFold(n_splits=3, random_state=100, shuffle=False),
                   error_score=nan,
                   estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features=None,
                                                    max_leaf_nodes=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.0,
                                                    presort='deprecated',
                                                    random_state=None,
                                                    splitter='best'),
                   iid='deprecated', n_iter=10, n_jobs=None,
                   param_distributions={'class_weight': ['balanced'],
                                        'criterion': ['gini'],
                                        'max_depth': [5, 6, 8, 12],
                                        'min_samples_leaf': [10, 12, 15],
                                        'min_samples_split': [200, 300, 500]},
                   pre_dispatch='2*n_jobs', random_state=100, refit=True,
                   return_train_score=False, scoring=None, verbose=True)
```

```
Train_set perfomance:

Accuracy Score: 0.9683624080048574
AUC-ROC: 0.9807955703085228
Precision Score: 0.15940625881910256
Recall Score/Sensitivity: 0.993380556912555
F1 Score: 0.27472742748287555

Test Set Performance:

Confusion Matrix:
 [[2799395   91850]
 [    170   17376]]

Accuracy Score: 0.9683648636151583
AUC-ROC: 0.9792714303881325
Precision Score: 0.159083002215589‌7
Recall Score/Sensitivity: 0.9903111820357916
F1 Score: 0.2741299340548386
```

- **Decision Tree** with Historical Variable-Hyperparameter Tuning was found to be the most cost efficient with overall good Performance Metrics.
- Class Balancing: Class of weight method.
- Cost Function is minimized pretty good.
- Error Rate are pretty low.
- Hence the Model Poses High Bias & Low Variance.

Cost Benefit Analysis

Cost Benefit Analysis		
S. No	Questions	Answer(in $) (Model Comparison to choose the m
		Decision Tree with Hyperparameter Tuning (5.4.2.1)
1	Cost incurred per month before the model was deployed (b*c)	426815.475
2	Average number of transactions per month detected as fraudulent by the model (TF)	9102.166667
3	Cost of providing customer executive support per fraudulent transaction detected by the model	$1.5
4	Total cost of providing customer support per month for fraudulent transactions detected by the model (TF*$1.5)	13653.25
5	Average number of transactions per month that are fraudulent but not detected by the model (FN)	170
6	Cost incurred due to fraudulent transactions left undetected by the model (FN*c)	90219
7	Cost incurred per month after the model is built and deployed (4+6)	103872.25
8	Final savings = Cost incurred before - Cost incurred after(1-7)	**322943.225**

- Final Savings was found as $**322943.225.**

CONCLUSION

- THE LATE NIGHT AND EARLY MORNING HOURS ARE RISKY TIME PERIOD FOR FRAUDULENT TRANSACTION
- DEPLOYMENT OF MACHINE LEARNING MODEL USING DECISION TREES WITH HYPER PARAMETER TUNING SAVE AS LARGE AS OVER $**322943.225.**
- TRANSACTION TIME RANGE = 10:00PM – 4:00 AM

Recommendations

- Finex must be more vigilant during late night hours and must provide two factor authentication for every transactions.
- Finex set up daily limits on each user.
- The model suggests, kids's & pet's category purchases must be tracked more proactively.
- Any transactions on the card could be considered unsafe and immediate follow up with customer is must.

THANK YOU

SRIKUMAR PEYYALA
SPECIALIZATION - DATA ANALYST
COURSE:- Executive Programme in Data Science
September 2021