Data Methodology

Step 1: Storyboarding

- Went through the data to get familiarized with it and noted down important fields
- Made a mind map of the various slides of the presentation
- Made a rough template based on this mind map

Step 2: Data Wrangling

- Explored all the columns in the dataset by importing it to python notebook
- Checked for the Missing values. There are no missing values.
- Attached below is the Python Notebook along with a few snapshots. Create new columns from the existing column

```
#Modifying Datatypes for time and date columns and converting to datetime format
dataset_train['trans_date_trans_time'] = pd.to_datetime(dataset_train['trans_date_trans_time'], errors='coerce')
dataset_train['dob'] = pd.to_datetime(dataset_train['dob'], errors='coerce')
dataset_train['unix_time'] = pd.to_datetime(dataset_train['unix_time'], errors='coerce')

dataset_test['trans_date_trans_time'] = pd.to_datetime(dataset_test['trans_date_trans_time'], errors='coerce')
dataset_test['dob'] = pd.to_datetime(dataset_test['dob'], errors='coerce')
dataset_test['unix_time'] = pd.to_datetime(dataset_test['unix_time'], errors='coerce')
```

```
#Converting dob to age and removing dob column
dataset_train['Transaction_Date'] = (dataset_train['trans_date_trans_time']).dt.date.astype('datetime64[ns]')
dataset_train['age'] = dataset_train['Transaction_Date'].dt.year - dataset_train['dob'].dt.year
dataset_train.drop('dob',1,inplace=True)
```

```
#Creating new columns by splitting time and date columns
dataset_train['trans_year']=pd.DatetimeIndex(dataset_train['trans_date_trans_time']).year
dataset_train['trans_month']=pd.DatetimeIndex(dataset_train['trans_date_trans_time']).month
dataset_train['trans_time']=pd.DatetimeIndex(dataset_train['trans_date_trans_time']).hour

dataset_test['trans_year']=pd.DatetimeIndex(dataset_test['trans_date_trans_time']).year
dataset_test['trans_month']=pd.DatetimeIndex(dataset_test['trans_date_trans_time']).month
dataset_test['trans_time']=pd.DatetimeIndex(dataset_test['trans_date_trans_time']).hour
```

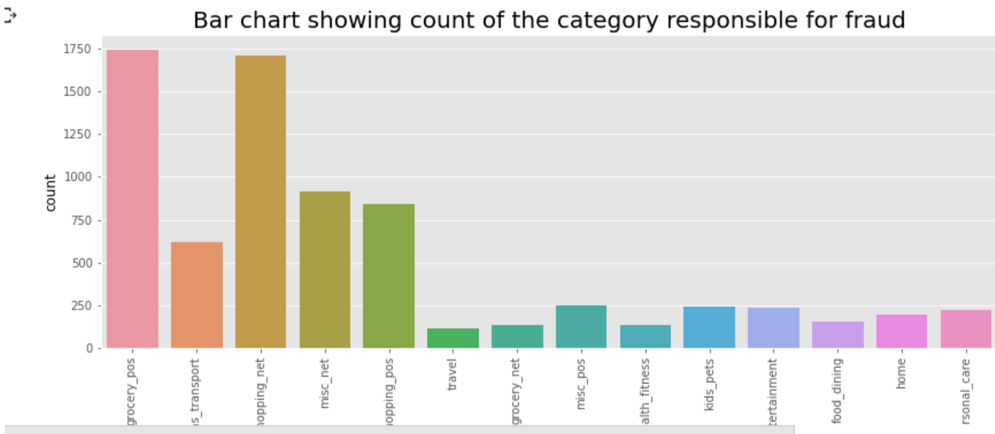
- Create a new column for the distance between the customer and merchant using lat long merch_long,merch_lat

```
#Distance between people co-ord and merchant co-ord
dataset_train['distance_people_to_merchant_co-ord(kms)'] = haversine_np(dataset_train['long'],dataset_train['lat'],dataset_train['merch_long'],dataset_train['mer
#longitude first, latitude second
dataset_test['distance_people_to_merchant_co-ord(kms)'] = haversine_np(dataset_test['long'],dataset_test['lat'],dataset_test['merch_long'],dataset_test['merch_la
```

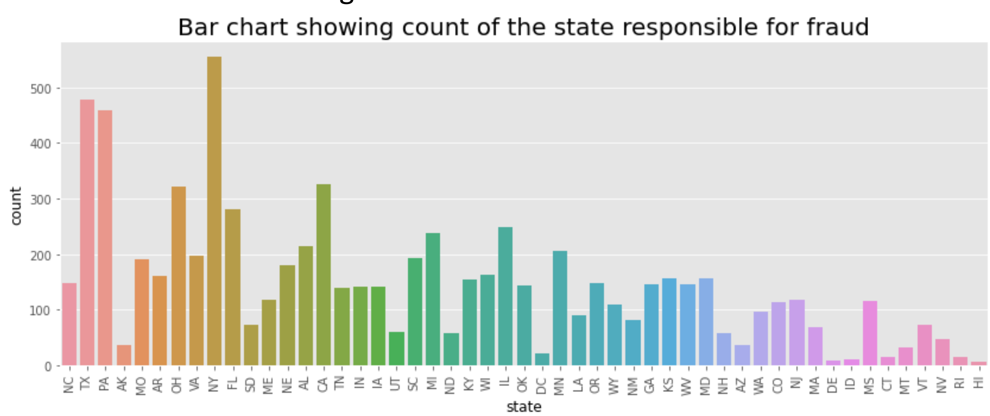
-
```
dataset_train.head()
```

Step 3: Data Analysis and Visualization

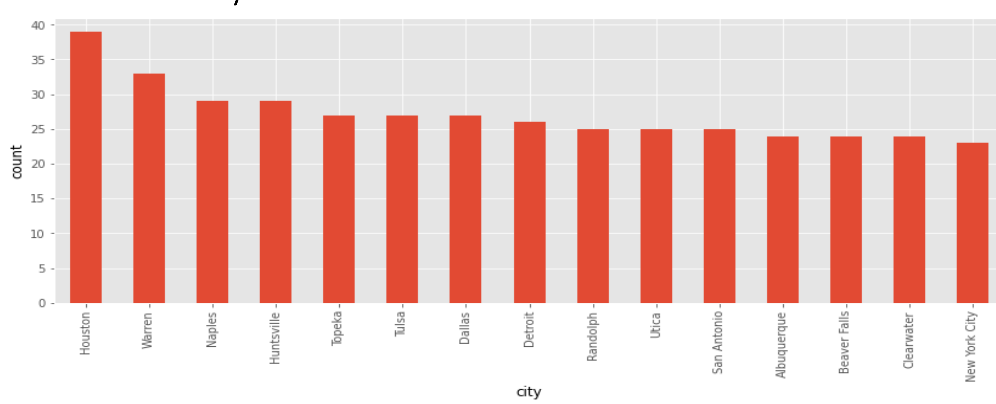
- Bar chart showing the categories that has more fraud values.



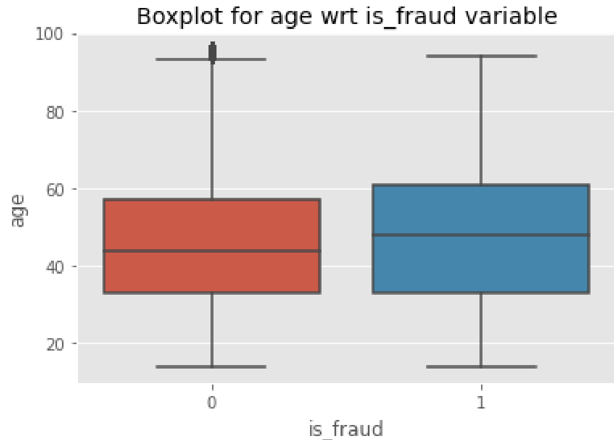
- Plot shows the states along with their fraud counts.

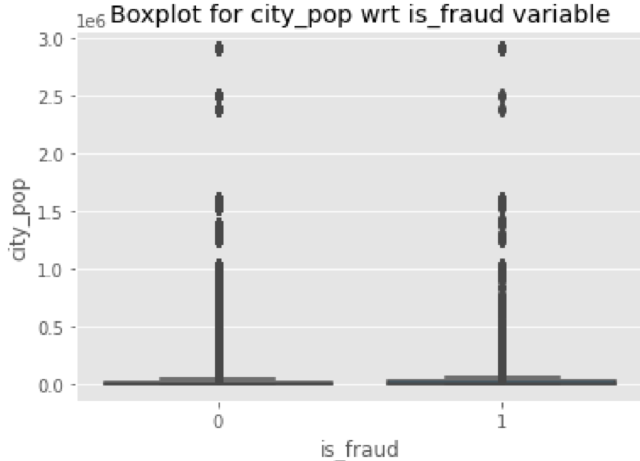


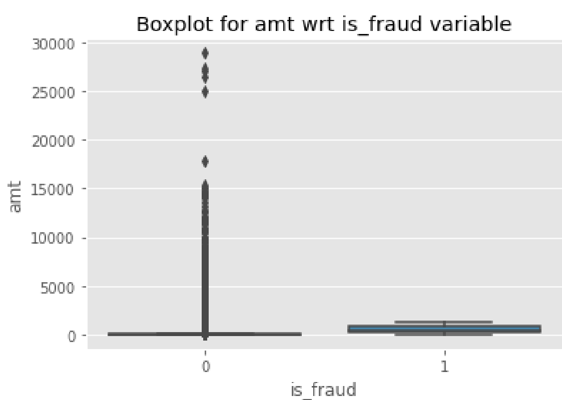
- Plot shows the city that have maximum fraud counts.

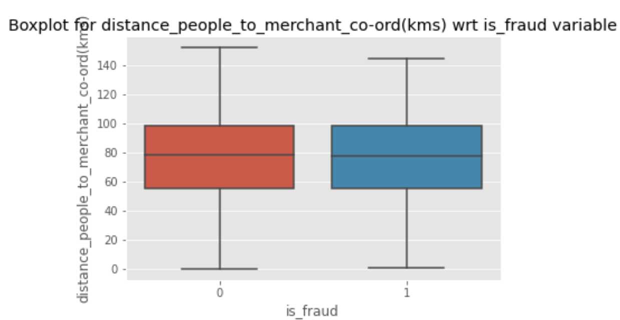


- The age ,amt ,distance_people_to_merchant column has no outliers for fraud cases while amt and city_pop statically shows outliers. However city population can vary drastically and none of them seems very high or very low. Hence, we will consider it as valid data.


Boxplot for age wrt is_fraud variable
age
100
80
60
40
20
0
1
is_fraud


Boxplot for city_pop wrt is_fraud variable
1e6
city_pop
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
is_fraud


Boxplot for amt wrt is_fraud variable
amt
30000
25000
20000
15000
10000
5000
0
0
1
is_fraud


Boxplot for distance_people_to_merchant_co-ord(kms) wrt is_fraud variable
distance_people_to_merchant_co-ord(kms)
140
120
100
80
60
40
20
0
0
1
is_fraud

Step 4: Data Modelling

- Prepare the training and validation set using stratified .

```
[ ] from sklearn.model_selection import train_test_split
```

```
[ ] X_train, X_valid, y_train, y_valid = train_test_split(X, y, train_size=0.7, random_state=42
```

```
[ ] # Inspecting the train and validation datasets
    X_train.shape, y_train.shape, X_valid.shape, y_valid.shape

    ((907672, 24), (907672,), (389003, 24), (389003,))
```

- And then use the power transformer for scaling.

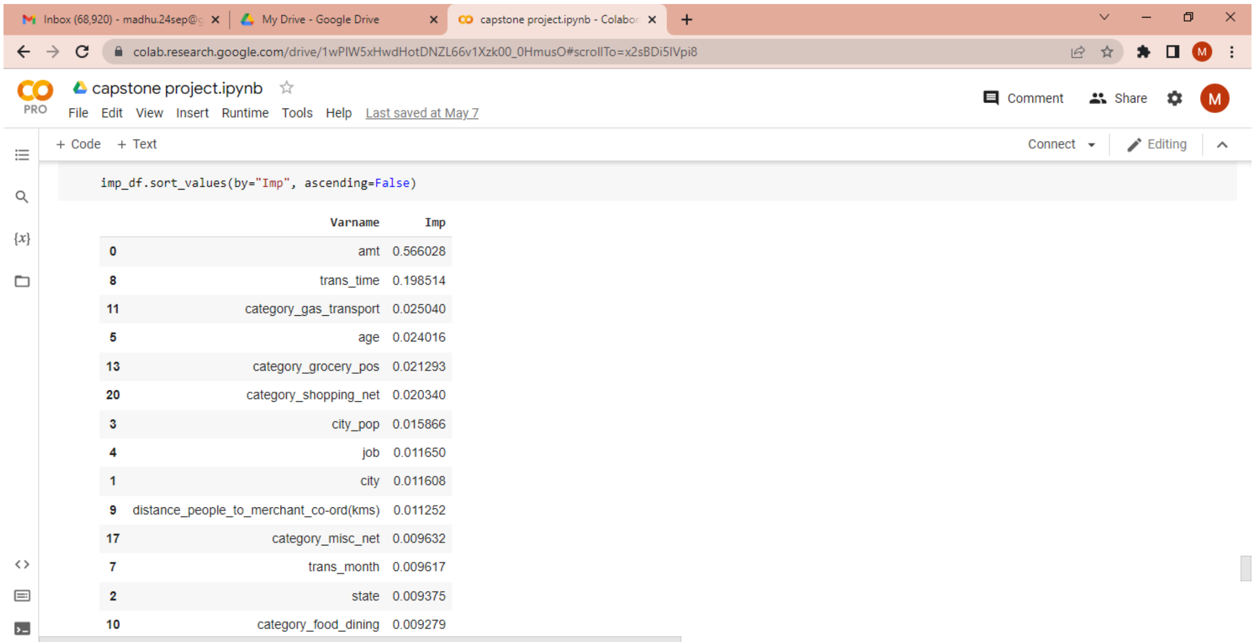
```
[ ] #Normalisation using power transformer
    scaler = PowerTransformer()

    X_train[['amt','city_pop','distance_people_to_merchant_co-ord(kms)','trans_year','trans_time','city', 'state','job','trans_month','age']] = scaler.fit_transform(
    X_valid[['amt','city_pop','distance_people_to_merchant_co-ord(kms)','trans_year','trans_time','city', 'state','job','trans_month','age']] = scaler.transform(X_va
    dataset_test[['amt','city_pop','distance_people_to_merchant_co-ord(kms)','trans_year','trans_time','city', 'state','job','trans_month','age']] = scaler.transform


    X_train.head()

              amt      city     state  city_pop      job      age  trans_year  trans_month  trans_time  distance_people_to_merchant_co-  ...  category_health_fit
```

- Next handle data imbalance . We have used the following techniques:-

  Random Under Sampling
  Random Over Sampling
  SMOTE
  ADASYN
- Built a logistic regression , decision tree, random forest and XgBoost model with the above-mentioned data imbalance techniques.
- The models are evaluated on performance metric AUC, F1-score,Senstivity and Specificity, precision and recall.
- Based on precision recall and f1-score we have finalised the Random Forest with over sampling technique and implement the model on the test set data.

```
AUC for the Random Forest Over sampling technique 0.9444477295884695
Accuracy    :  0.9957226583939005
Sensitivity :  0.8927738927738927
Specificity :  0.9961215664030464
Precision   :  0.4714426390940423
Recall      :  0.8927738927738927
F1_score: 0.6170452714676977
[[551427   2147]
 [   230   1915]]
```

- The important features that help in detecting fraud identified from the model are listed below.




```
imp_df.sort_values(by="Imp", ascending=False)
```

	Varname	Imp
0	amt	0.566028
8	trans_time	0.198514
11	category_gas_transport	0.025040
5	age	0.024016
13	category_grocery_pos	0.021293
20	category_shopping_net	0.020340
3	city_pop	0.015866
4	job	0.011650
1	city	0.011608
9	distance_people_to_merchant_co-ord(kms)	0.011252
17	category_misc_net	0.009632
7	trans_month	0.009617
2	state	0.009375
10	category_food_dining	0.009279

Note: High recall implies the more fraud cases are predicted correctly. Even though the precision is 47% it implies 47% non-fraud cases are wrongly quoted as fraud by the machine learning model. But we can arrange a representative for verifying by calling the customer if the transaction was genuine or not and proceed further.

- We have performed the cost-based analysis and derived the following conclusion. Before the model was deployed average cost for bank due to fraudulent transaction was 94443.72 per month. We make an average saving of 83809.14 after deploying the model for fraud analysis per month.

  Hence the bank can make a lot of saving by implementing the machine learning model to detect the frauds.

Step 5:Presentation

- Made the presentation adhering to best practices and pyramid principle.
- Here Business stake holders are our audience
- Added recommendations for the clients