DATS 6401: Visualization of Complex Data

Dr. Reza Jafari

Final Term Project

Atharva Haldankar

George Washington University

12-18-2022

ABSTRACT

This report focuses on developing a python code. The objective of this report is to apply the course learning objectives to a real dataset for Visualization of Complex Data A real world data is acquired and with the help of all the tools and knowledge from the learnt from this course is applied to Visualize the same. Finally a Dash application is developed to make it more appealing and user-friendly.

1: Introduction

Data visualization methods were used on the data which contains credit card transactions which are legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants. This data was taken from Kaggle. This data is consisted of two set i.e., the test and the train set. Since we had very few fraud transactions in the data so we combined all the "fraud" transactions from the train and test sets. Before working on the we pre-processed the data. We also perform PCA on the data for feature reduction and some statistical tests. Finally, we have visualized the data using different plots to understand the data better.

2 : Description of the dataset

Pre-processing of the data

The dataset is taken from Kaggle. First, the "trans_date_trans_time" was set as the index.

After setting the index we dropped all the unnecessary columns which are as follows:

1. "Unnamed:0"
2. "trans_date_trans_time"
3. "cc_num"
4. "first"
5. "last"
6. "street"
7. "lat"
8. "long"
9. "dob"
10. "unix_time"
11. "merch_lat"
12. "merch_long"

We extract all the "fraud" transactions form the test dataset and combined it with the main data set to reduce the imbalance between the two classes.
We now look for all the missing samples (if present) with the help of ".isna" and the ".sum" function. With the help of these functions, we find that there are no missing samples in this data.

```
print("The number of missing values in the dataset:",df_final.isna().sum().sum())
```

```
The number of missing values in the dataset: 0
```

To get a better idea of all the continuous columns we use the ".describe" function which gives us the following information.

```
print("The description of data\n",df_final.describe().to_string())
```

	amt	zip	city_pop	is_fraud
count	1.298820e+06	1.298820e+06	1.298820e+06	1.298820e+06
mean	7.110743e+01	4.879912e+04	8.878432e+04	7.430591e-03
std	1.620471e+02	2.689293e+04	3.018400e+05	8.588005e-02
min	1.000000e+00	1.257000e+03	2.300000e+01	0.000000e+00
25%	9.660000e+00	2.623700e+04	7.430000e+02	0.000000e+00
50%	4.758000e+01	4.817400e+04	2.456000e+03	0.000000e+00
75%	8.331000e+01	7.201100e+04	2.032800e+04	0.000000e+00
max	2.894890e+04	9.992100e+04	2.906700e+06	1.000000e+00

We make sure all the data we're working with are unique samples. I was able to achieve this by using the "set" function, which makes a list of all the unique values from the data and compare the length of the output of the "set" function with the length of the dataset.

```
print("All the entries in the dataset are unique:",len(df_final)==len(set(df_final.trans_num)))
All the entries in the dataset are unique: True
```

We now visualize the first 100 samples of the continuous columns of the data to get a picture of how the data moves with time before performing the z-transformation.

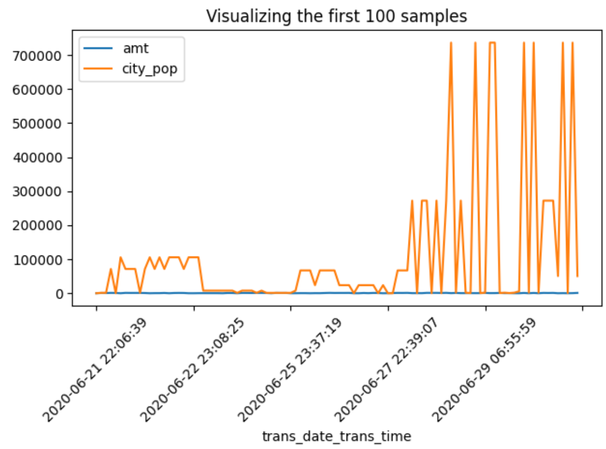

Figure 1: First 100 samples of the continuous variables.

Since it very hard to tell any relation between the two continuous columns. Now we perform z-transform on the above data and visualize the same.

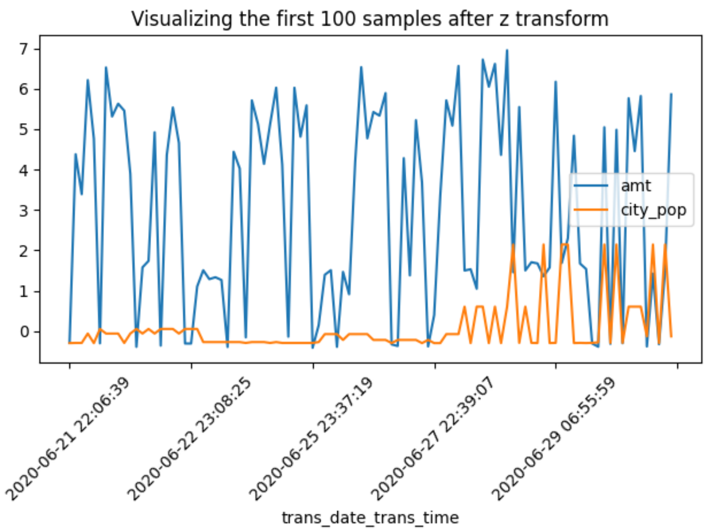

Figure 2: First 100 samples of the continuous variable after Z-transformation.

If we compare the Fig 1 with Fig 2 we can now see how the data of these two columns move with time together.

3 : Pre-processing

Outlier Detection using IQR method for transaction amount

We develop a function which calculates the 1st Quartile, the 3rd Quartile and then calculate the Inter quantile range.

With the output of the above function and formulas we calculate the upper limit and the lower limit and the upper limit the transaction amount column

```
Q1 and Q3 of the transaction amount is 9.66 $ & 83.31 $
IQR for the transaction amount is 73.65 $
Any amount < -100.82 $ and amount > 193.79 $ is an outlier
```

We calculate the number of outliers present in the transaction amount column.

```
The number of outliers present in the transaction amounts: 68547
```

After removing the outliers from the data we visualize the data with a boxplot.

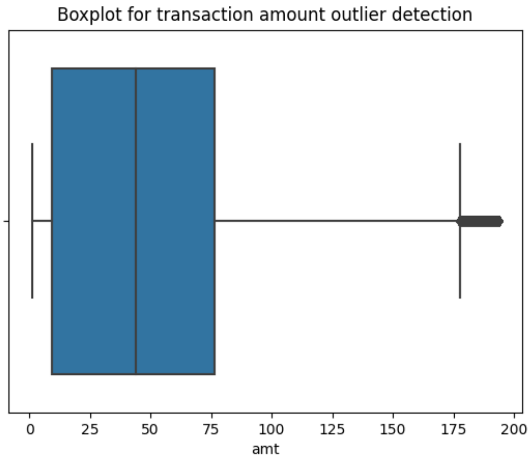

Figure 3: Boxplot after removing the outliers from the data using IQR method.

Even after removing the outliers, there're few still present in the data. So using Fig 3 we remove the rest of the outliers.

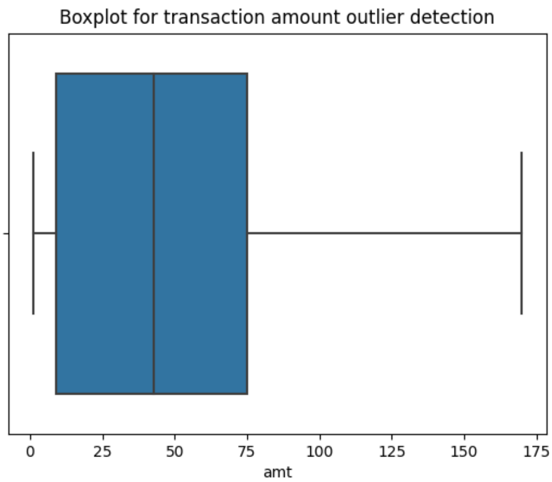

Figure 4: Boxplot after removing the outliers from the data using Boxplot.

Outlier Detection using IQR method for transaction amount

We develop a function which calculates the 1st Quartile, the 3rd Quartile and then calculate the Inter quantile range.

With the output of the above function and formulas we calculate the upper limit and the lower limit and the upper limit the transaction amount column

```
Q1 and Q3 of the city population is 725.0 & 19054.0
IQR for the city population is 18329.0
Any population < -26768.5 and population > 46547.5 is an outlier
```

We calculate the number of outliers present in the transaction amount column.

```
The number of outliers present in the city population: 229614
```

After removing the outliers from the data we visualize the data with a boxplot.

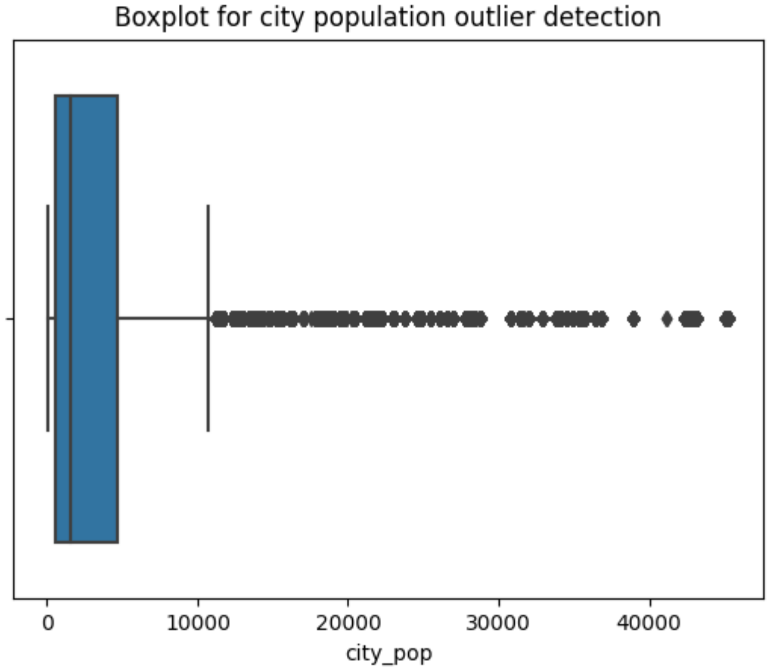

Figure 5: 1BOxplot for city_pop after removing the outliers

Even after removing the outliers, there're few still present in the data. So, using Fig 6 we remove the rest of the outliers.

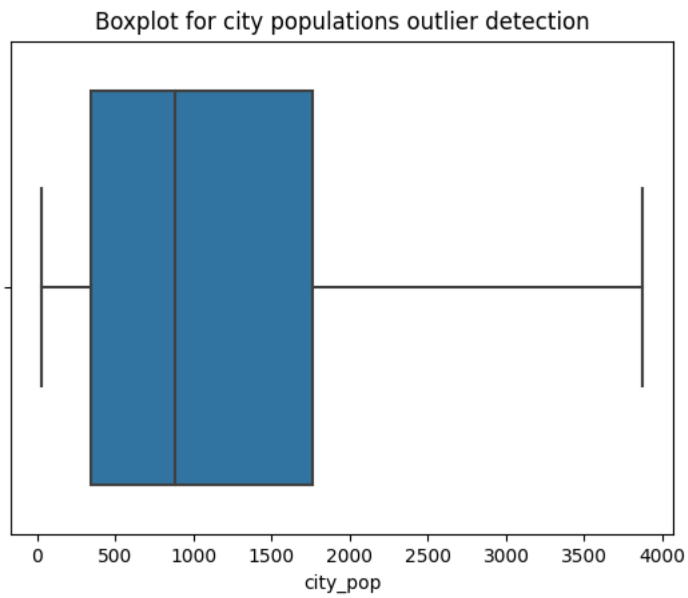

Figure 5: Boxplot for city_pop after removing the outliers

4 : Principal Component Analysis

We start Visualizing the data after standardizing it.
We use the following function to standardize the data.

```
scaler = StandardScaler()
scaled = scaler.fit_transform(df_final[["amt","city_pop"]])
```

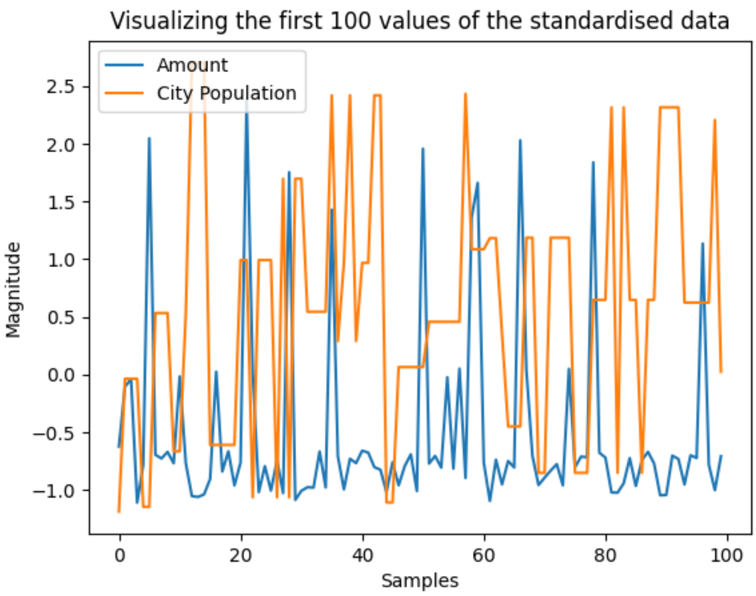

Figure 6: Visualisation for the first 100 values of the standardised data.

Calculating the singular values and the condition number to detect any collinearity.

```
SingularValues =  [1.59620451e+12 1.98602478e+09]
The condition number for the features =  28.349926349814407
```

Since the singular values are so high and the condition is low, we can conclude that there is very weak collinearity which is good for us.

We still perform PCA based on the "MLE" to check for a possible feature dimension reduction.

```
Explained Variance for Original Components [0.99814236]
    ...:
Explained Variance for Scaled Components [0.51826613]
```

```
SingularValues =  [724152.53564553 673107.46435447]
The condition number for the features =  1.037224638434283
```

We can see that the scaled data has a lower condition number and high singular values as well

4 : Normality test

Normality test for the Transaction Amount

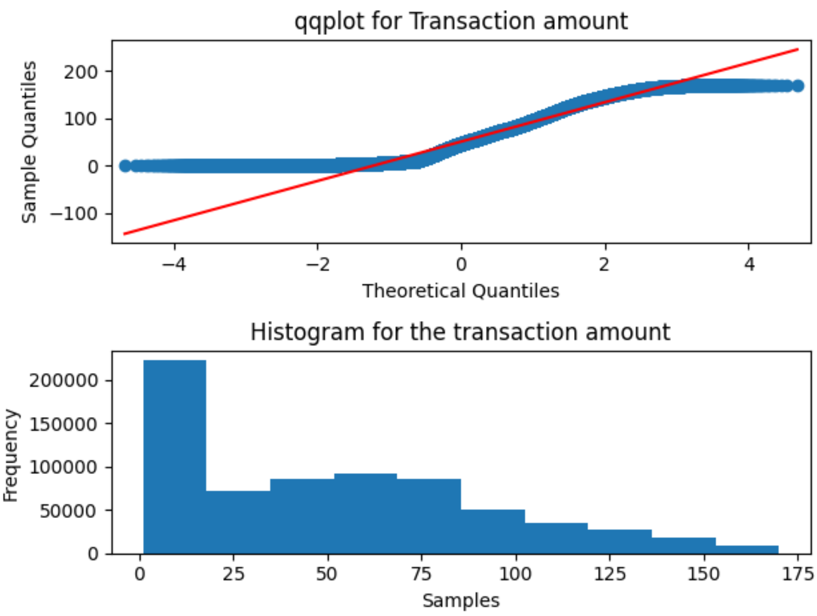

Figure 7: QQplot and histogram subplot to see distribution of the data

```
K-S test: Transaction Amount dataset: statistics= 0.12 p-value = 0.00
K-S test : Transaction Amount dataset is Not Normal
```

From the QQ plot, histogram and the KS test we can see that the data is not normal and is skewed i.e., not Gaussian distribution.

Normality test for the City Population

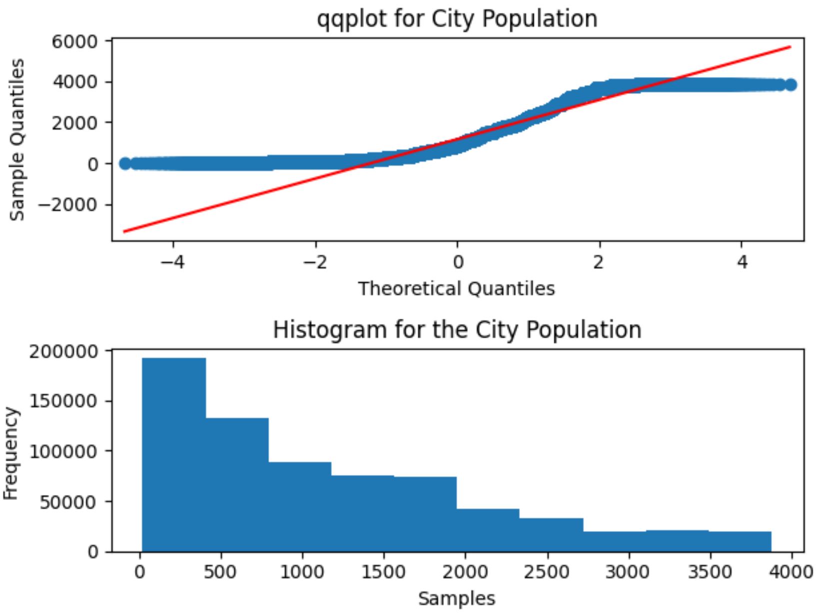

Figure 8: QQ Plot and histogram to visualize the distribution of the data

```
K-S test: City Population dataset: statistics= 0.13 p-value = 0.00
K-S test : City Population dataset is Not Normal
```

From the QQ plot, histogram and the KS test we can see that the data is not normal and is skewed i.e., not Gaussian distribution.

Normality transformation for the Transaction Amounts

Normality transformation for the City Population

5 : Heatmap & Pearson correlation coefficient matrix

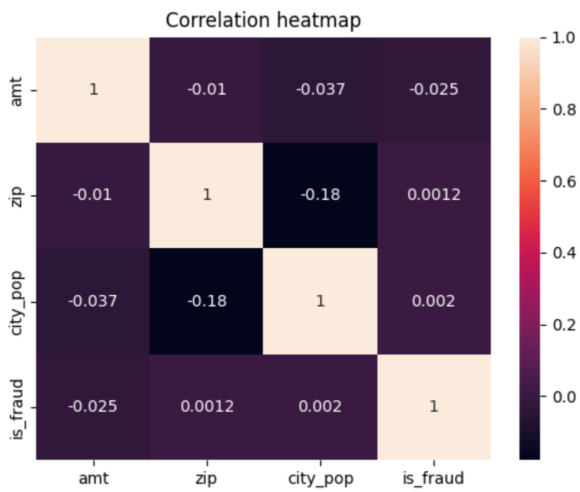

Figure 9: Correlation coefficient heatmap

From the above heatmap we can see that there isn't very strong correlation between any of features of this dataset.

6 : Visualizing the data

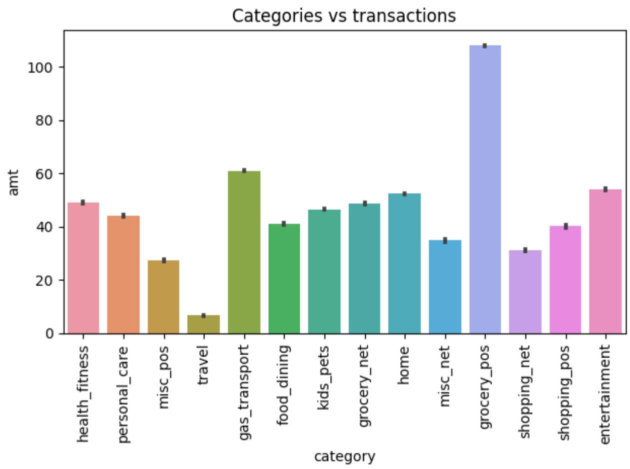

Figure 10: Categories vs Transactions bar plot

From the above plot we can see that "grocery_pos" has the maximum transactions in this data.

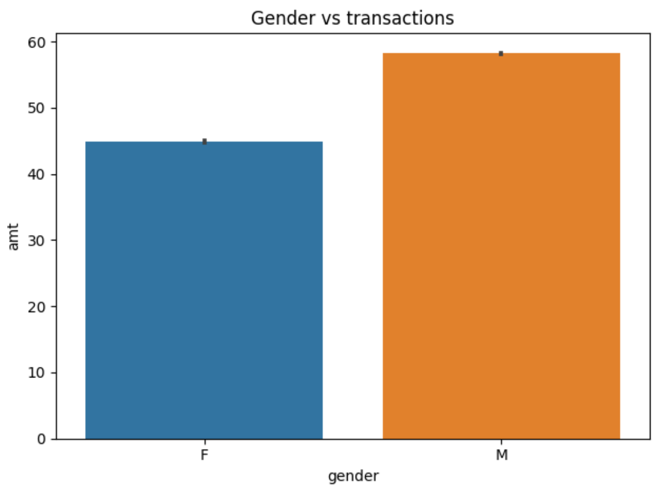

Figure 11: Gender VS transactions

From the above graph we can see that men have higher transactions compared to women.

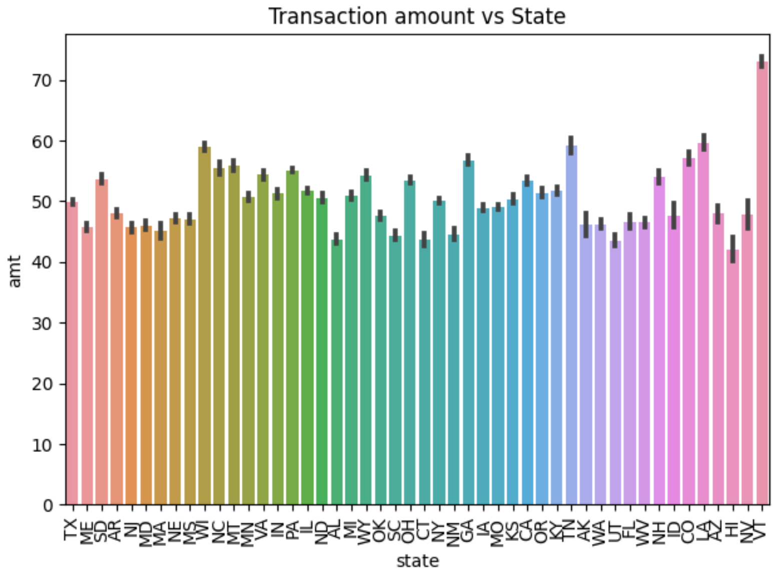

Figure 12: Transactions VS State

From the above plot we can see that there're maximum transactions in "VT" i.e., Vermont.

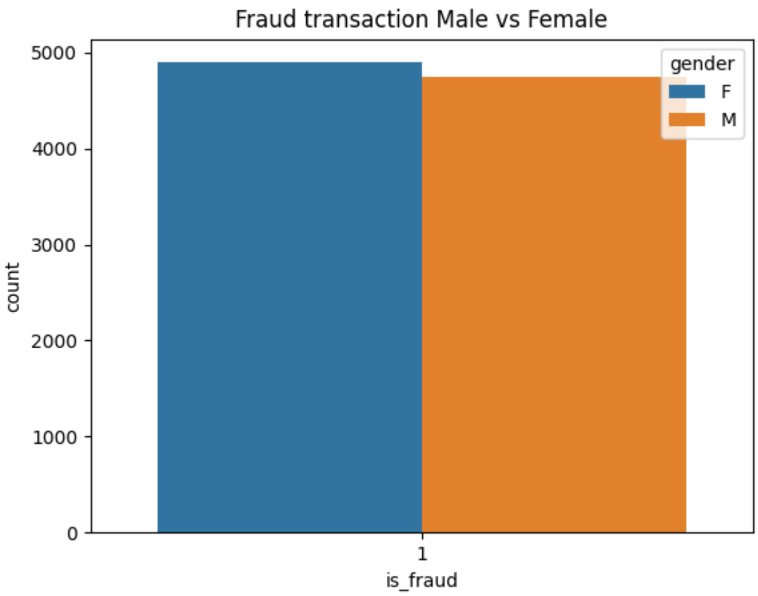

Figure 13: Fraud transaction Male VS Female.

From the above plot we can see that there're more female customers who were involved in a fraud transaction compared to a male customer.

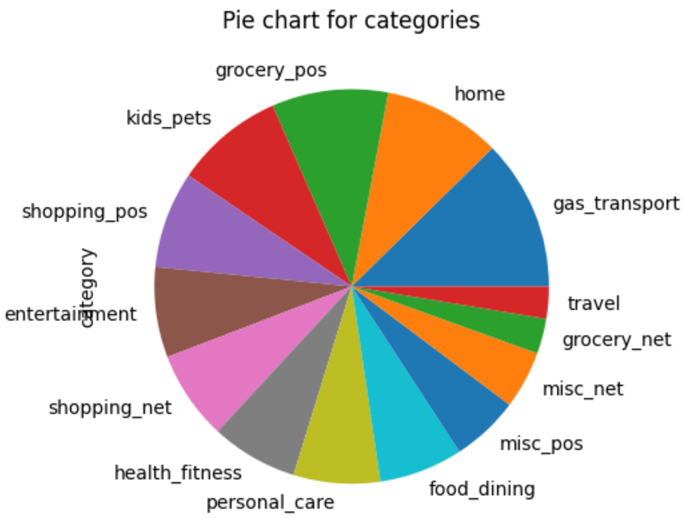

Figure 14: Pie chart for categories

From the above plot we can visualize the transactions in this data for different categories. It is evident that the 'gas_transport" has the maximum transactions in this dataset.

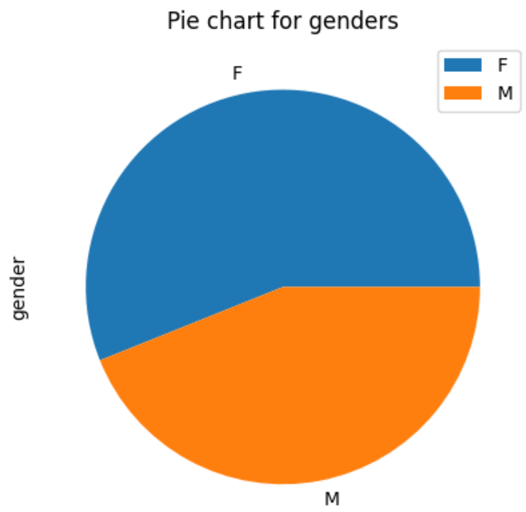

Figure 15: Male to Female ratio in this transaction.

From this pie chart we can tell that there is more female compared to the number of males in this data.

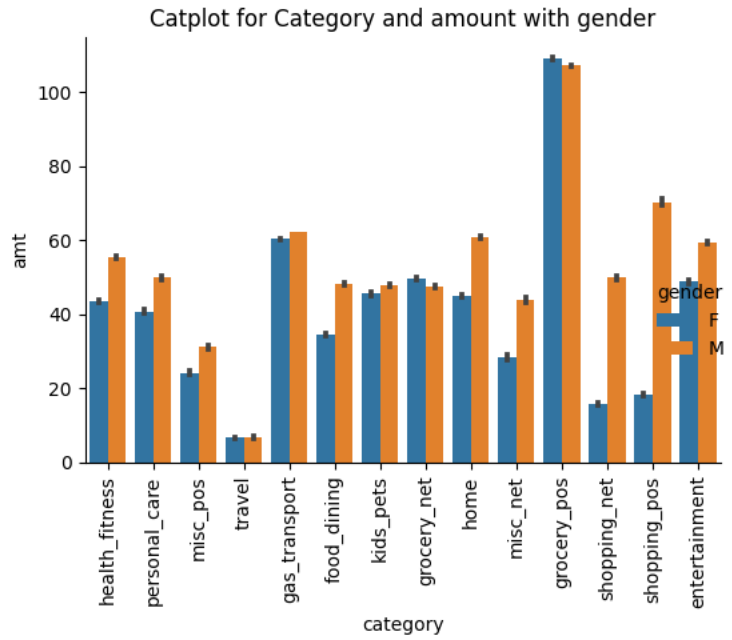

Figure 16: Catplot for categories and amount with gender

From the above plot we can get better information for the gender of customer spending for different categories.
From figure 21 we can see that amongst everyone in the data, the female customers have spent the most for “grocery_pos” in this data.

Figure 17: Violin plot for the state and its population with gender

From the above plot we can see the distribution of the city population for the different categories for male and female customers.

Figure 18: Violin plot for the state and it's transaction amounts with gender

From the above plot we can see the distribution of the transaction amount for the different categories for male and female customers.

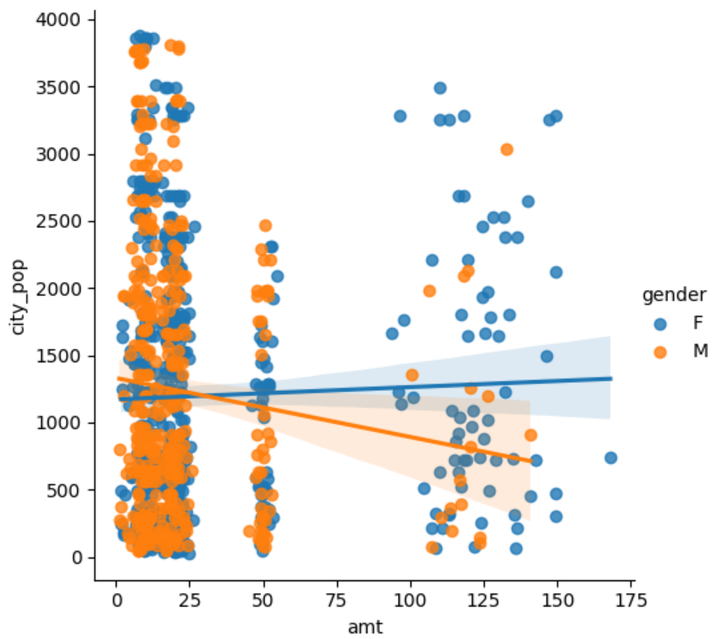

Figure 19: Scatter plot and regression line

The above plot visualizes with scatter plot for the city population and amount with the regression line.
Most of the data is between 0-25 and has an almost straight regression line for the females at 1100 and 1500 for men

7: Subplots

Figure 21: 2Subplot of above plots

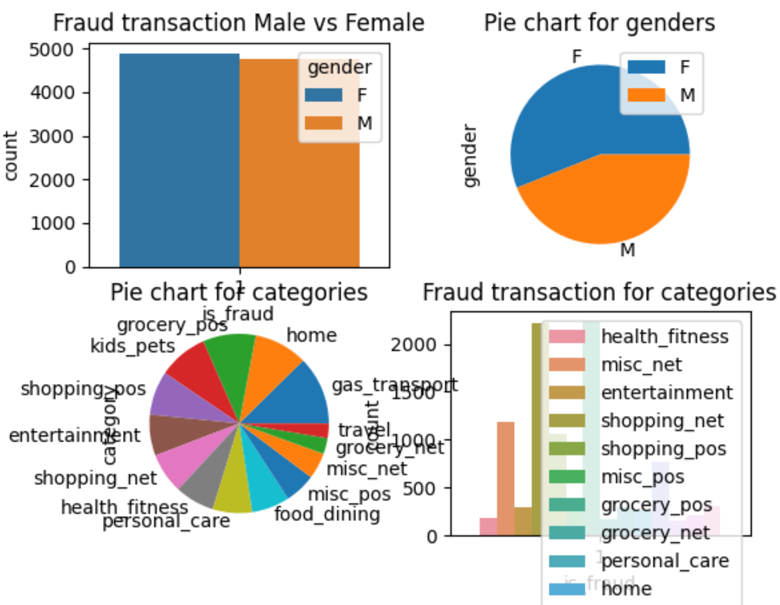

Figure 22: 2Subplots for above plots 2

Figure 22: 3

8 : Recommendation

```
Recommendation
 1: Making different plots
 2: Understand the data with the help of plots
 3:The dash app makes it easy in terms of the interface.
```

9 : Conclusion

The above plots we can get an idea of this entire data and design a model accordingly.
This also helps us understand dynamics of other features of the data and they're affected with the other features.

14: References

Datahttps://www.kaggle.com/datasets/kartik2112/fraud-detection

15 : Appendix

```
#%%
import dash
import dash_html_components as html
import matplotlib.pyplot as plt
import numpy as np
from dash import dcc
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd
import math
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler, SMOTE
from collections import Counter
from scipy import stats
from sklearn.decomposition import PCA
from statsmodels.graphics.gofplots import qqplot
from sklearn.preprocessing import StandardScaler
from numpy import linalg as LA
from normal_test import shapiro_test,ks_test,da_k_squared_test
#%%
def quartile(data):
    Q1 = np.percentile(data, 25, method='midpoint')
    Q3 = np.percentile(data, 75, method='midpoint')
    IQR = Q3 - Q1
    return Q1,Q3,IQR
#%%
df = pd.read_csv("/Users/atharvah/GWU/Sem 3 /Data Visualisation/Final
Project/archive/fraudTrain.csv") # reading the train data
df = df.set_index(df.trans_date_trans_time)
df = df.drop(columns=["Unnamed:
0","trans_date_trans_time","cc_num","first","last","street","lat","long","dob","uni
x_time","merch_lat","merch_long"])
df2 = pd.read_csv("/Users/atharvah/GWU/Sem 3 /Data Visualisation/Final
Project/archive/fraudTest.csv") # reading the test data
df2 = df2.set_index(df2.trans_date_trans_time)
df2 = df2.drop(columns=["Unnamed:
0","trans_date_trans_time","cc_num","first","last","street","lat","long","dob","uni
x_time","merch_lat","merch_long"])
fraud = df2[df2.is_fraud==1] # extracting all the 'fraud' transactions from the
test dataset
fraud = fraud.append(df[df.is_fraud==1]) # combining all the fraud transactions
from the test and the train dataset.
not_fraud = df[df.is_fraud==0] # Extracting all the "not fraud" data.
#%%
df_final = fraud
df_final= df_final.append(not_fraud) # to reduce the imbalance we only take 40500
"not fraud" transactions into consideration.
# Pre-processing the data
print("The number of missing values in the dataset:",df_final.isna().sum().sum()) #
Counting the missing values in the data
print("The description of data\n",df_final.describe().to_string())
print("All the entries in the dataset are
unique:",len(df_final)==len(set(df_final.trans_num)))
#%%
# Plotting first 100 samples for amount and populations
df_final[["amt","city_pop"]][:100].plot()
plt.title("Visualizing the first 100 samples")
plt.xticks(rotation = 45)
plt.tight_layout()
plt.show()
#%%
# z transformed data
def z_transform(df):
    mean = np.mean(df)
```

```
    std = np.std(df)
    trans = (df-mean)/std
    return trans

transformed_data = z_transform(df_final[["amt","city_pop"]])

transformed_data[:100].plot()
plt.title("Visualizing the first 100 samples after z transform")
plt.xticks(rotation = 45)
plt.tight_layout()
plt.show()
#%%
# Outlier detection for amount using interquartile method
Q1, Q3, IQR = quartile(df_final.amt)
print(f"Q1 and Q3 of the transaction amount is {round(Q1,2)} $ & {round(Q3,2)} $\n"
      f"IQR for the transaction amount is {round(IQR,2)} $\n"
      f"Any amount < {round(Q1-(1.5 * IQR),2)} $ and amount >
{round(Q3+(1.5*IQR),2)} $ is an outlier")

#%%
print("The number of outliers present in the transaction
amounts:",len(df_final.amt[(df_final.amt<-100.82)|(df_final.amt>193.79)]))
#%%
# Removing the outliers from the data
df_final = df_final[(df_final.amt<(Q3+(1.5*IQR)))&(df_final.amt>(Q1-(1.5 * IQR)))]
#%%
# Box plot for the transaction amounts after removing the outliers
sns.boxplot(df_final.amt)
plt.title("Boxplot for transaction amount outlier detection")
plt.show()
#%%
# From the above boxplot we can see that there are plenty of outlier which are yet
to be removed,so we use the boxplot to find out upper limit.
# Removing the outliers from the data using boxplot
df_final = df_final[(df_final.amt< 170)&(df_final.amt>-94.77)]
sns.boxplot(df_final.amt)
plt.title("Boxplot for transaction amount outlier detection")
plt.show()
#%%
# Outlier detection for city population using interquartile method
Q1, Q3, IQR = quartile(df_final.city_pop)
print(f"Q1 and Q3 of the city population is {round(Q1,2)} & {round(Q3,2)}\n"
      f"IQR for the city population is {round(IQR,2)}\n"
      f"Any population < {round(Q1-(1.5 * IQR),2)} and population >
{round(Q3+(1.5*IQR),2)} is an outlier")

#%%
print("The number of outliers present in the city
population:",len(df_final.city_pop[(df_final.city_pop<-
26768.5)|(df_final.city_pop>46547.5)]))
#%%
# Removing the outliers from the data
df_final = df_final[(df_final.city_pop<(Q3+(1.5*IQR)))&(df_final.city_pop>(Q1-(1.5
* IQR)))]
#%%
# Box plot for the city population after removing the outliers
sns.boxplot(df_final.city_pop)
plt.title("Boxplot for city population outlier detection")
plt.show()
#%%
# From the above boxplot we can see that there are plenty of outlier which are yet
to be removed,so we use the boxplot to find out upper limit.
# Removing the outliers from the data using boxplot
df_final = df_final[(df_final.city_pop< 3900)&(df_final.city_pop>-94.77)]
sns.boxplot(df_final.city_pop)
plt.title("Boxplot for city populations outlier detection")
plt.show()
#%%
```

```
scaler = StandardScaler()
scaled = scaler.fit_transform(df_final[["amt","city_pop"]])
print(scaled)
plt.plot(scaled[:,0][:100],label = "Amount")
plt.plot(scaled[:,1][:100],label = "City Population")
plt.title("Visualizing the first 100 values of the standardised data")
plt.legend()
plt.xlabel("Samples")
plt.ylabel("Magnitude")
plt.show()
#%%
H
=np.matmul(df_final[["amt","city_pop"]].values.T,df_final[["amt","city_pop"]].value
s)
s, d, v = np.linalg.svd(H)
print("SingularValues = ",d)
print("The condition number for the features =
",LA.cond(df_final[["amt","city_pop"]].values))
#%%
pca = PCA(n_components="mle")
ScaledComponents = pca.fit_transform(scaled)
print("Explained Variance for Scaled Components",pca.explained_variance_ratio_)
cum_sum_eigenvalues = np.cumsum(ScaledComponents)
plt.plot(np.arange(1,len(np.cumsum(pca.explained_variance_ratio_))+1,1),np.cumsum((
pca.explained_variance_ratio_)))
plt.xticks(np.arange(1,len(np.cumsum(pca.explained_variance_ratio_))+1,1))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.title("cumulative explained variance VS number of components")
plt.grid()
plt.show()
#%%
df_pca = pd.DataFrame(ScaledComponents,columns=['Principal col %i' % i for i in
range(ScaledComponents.shape[1])],index=df_final.index)
#%%
OriginalComponents = pca.fit_transform(df_final[["amt","city_pop"]])
print("Explained Variance for Original Components",pca.explained_variance_ratio_)
#%%
H =np.matmul(scaled.T,scaled)
s, d, v = np.linalg.svd(H)
print("SingularValues = ",d)
print("The condition number for the features = ",LA.cond(scaled))
#%%
#%%
# Normality test for transaction amount
qqplot(df_final.amt,line="s",ax=plt.subplot(2,1,1))
plt.title("qqplot for Transaction amount")

plt.subplot(2,1,2)
plt.hist(df_final.amt)
plt.title("Histogram for the transaction amount")
plt.xlabel("Samples")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
print(ks_test(df_final.amt,"Transaction Amount"))
#%%
# Normality test for transaction amount
qqplot(df_final.city_pop,line="s",ax=plt.subplot(2,1,1))
plt.title("qqplot for City Population")

plt.subplot(2,1,2)
plt.hist(df_final.city_pop)
plt.title("Histogram for the City Population")
plt.xlabel("Samples")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```

```
print(ks_test(df_final.city_pop,"City Population"))
#%%
transformed_pop = z_transform(df_final.amt)

qqplot(transformed_pop,line="s",ax=plt.subplot(2,1,1))
plt.title("qqplot for Transaction amount")

plt.subplot(2,1,2)
plt.hist(transformed_pop)
plt.tight_layout()
plt.show()
print(ks_test(transformed_pop,"Transaction Amount"))
# qqplot(transformed_pop,line="s")
# plt.title("qqplot for City population")
# plt.show()
print(ks_test(df_final.city_pop,"City population"))
#%%
corr = df_final.corr()
sns.heatmap(corr,annot = True)
plt.title("Correlation heatmap")
plt.show()
#%%
#Line plots
sns.lineplot(data = df_final,
             x = "amt")
plt.title("Line plot for the Transaction amounts using Seaborn")
plt.show()

#%%
df_final.city_pop.plot(kind = "line")
plt.title("Line plot for city population")
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
#%%
sns.barplot(data = df_final,
            y = "amt",
            x = "category")
plt.xticks(rotation = 90)
plt.title("Categories vs transactions")
plt.tight_layout()
plt.show()
#%%
sns.barplot(data = df_final,
            y = 'amt',
            x = "gender")
plt.title("Gender vs transactions")
plt.tight_layout()
plt.show()
#%%
sns.barplot(data = df_final,
            y = "amt",
            x = "state")
plt.title("Transaction amount vs State")
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
#%%
sns.countplot(data = fraud,
              x = "is_fraud",
              hue = "gender")
plt.title("Fraud transaction Male vs Female")
plt.show()
#%%
sns.countplot(data = fraud,
              x = "is_fraud",
              hue = "category")
plt.legend(loc = "upper right")
```

```
plt.title("Fraud transaction for categories")
plt.show()
#%%
df_final.category.value_counts().plot(kind = "pie")
plt.title("Pie chart for categories")
plt.show()
#%%
df_final.gender.value_counts().plot(kind = "pie")
plt.title("Pie chart for genders")
plt.legend()
plt.show()
#%%
sns.catplot(data = df_final,
            x = "category",
            y = "amt",
            hue = "gender",
            kind="bar")
plt.title("Catplot for Category and amount with gender")
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
#%%
sns.violinplot(data=df_final, x="category", y="city_pop", hue="gender",
               split=True)
plt.title("Violin plot for the state and it's population with gender")
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
#%%
sns.violinplot(data=df_final, x="category", y="amt", hue="gender",
               split=True)
plt.title("Violin plot for the state and it's transaction amounts with gender")

plt.xticks(rotation = 45)
plt.tight_layout()
plt.show()
#%%
sns.lmplot(
    data=df_final[:1000],
    x="amt", y="city_pop", hue="gender",
    height=5
)
plt.title("Scatter plot and regression line")
plt.show()
#%%
#%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plt.subplot(2,2,1)
df_final.city_pop.plot(kind = "line")
plt.title("Line plot for city population")
plt.xticks(rotation = 90)
plt.tight_layout()


sns.barplot(data = df_final,
            y = "amt",
            x = "category",
            ax = plt.subplot(2,2,2))
plt.xticks(rotation = 90)
plt.title("Categories vs transactions")
plt.tight_layout()


sns.barplot(data = df_final,
            y = 'amt',
            x = "gender",
            ax = plt.subplot(2,2,3))
plt.title("Gender vs transactions")
plt.tight_layout()
```

```
sns.barplot(data = df_final,
            y = "amt",
            x = "state",
            ax = plt.subplot(2,2,4))
plt.title("Transaction amount vs State")
plt.xticks(rotation = 90)
plt.tight_layout()
plt.show()
#%%

sns.countplot(data = fraud,
              x = "is_fraud",
              hue = "gender",
              ax = plt.subplot(2,2,1))
plt.title("Fraud transaction Male vs Female")


sns.countplot(data = fraud,
              x = "is_fraud",
              hue = "category",
              ax = plt.subplot(2,2,4))
plt.legend(loc = "upper right")
plt.title("Fraud transaction for categories")

plt.subplot(2,2,3)
df_final.category.value_counts().plot(kind = "pie")
plt.title("Pie chart for categories")


ax = plt.subplot(2,2,2)
df_final.gender.value_counts().plot(kind = "pie")
plt.title("Pie chart for genders")
plt.legend()
plt.show()
#%%
sns.catplot(data = df_final,
            x = "category",
            y = "amt",
            hue = "gender",
            kind="bar",
            ax = plt.subplot(2,2,5))
plt.title("Catplot for Category and amount with gender")
plt.xticks(rotation = 90)



sns.violinplot(data=df_final, x="category", y="city_pop", hue="gender",
               split=True,
               ax = plt.subplot(3,4,10))
plt.title("Violin plot for the state and it's population with gender")
plt.xticks(rotation = 90)



sns.violinplot(data=df_final, x="category", y="amt", hue="gender",
               split=True,
               ax = plt.subplot(3,4,11))
plt.title("Violin plot for the state and it's transaction amounts with gender")

plt.xticks(rotation = 45)

plt.tight_layout()
plt.show()
#%%
print("Recommendation\n 1: Making different plots \n 2: Understand the data with
the help of plots \n 3:The dash app makes it easy in terms of the interface.")
```