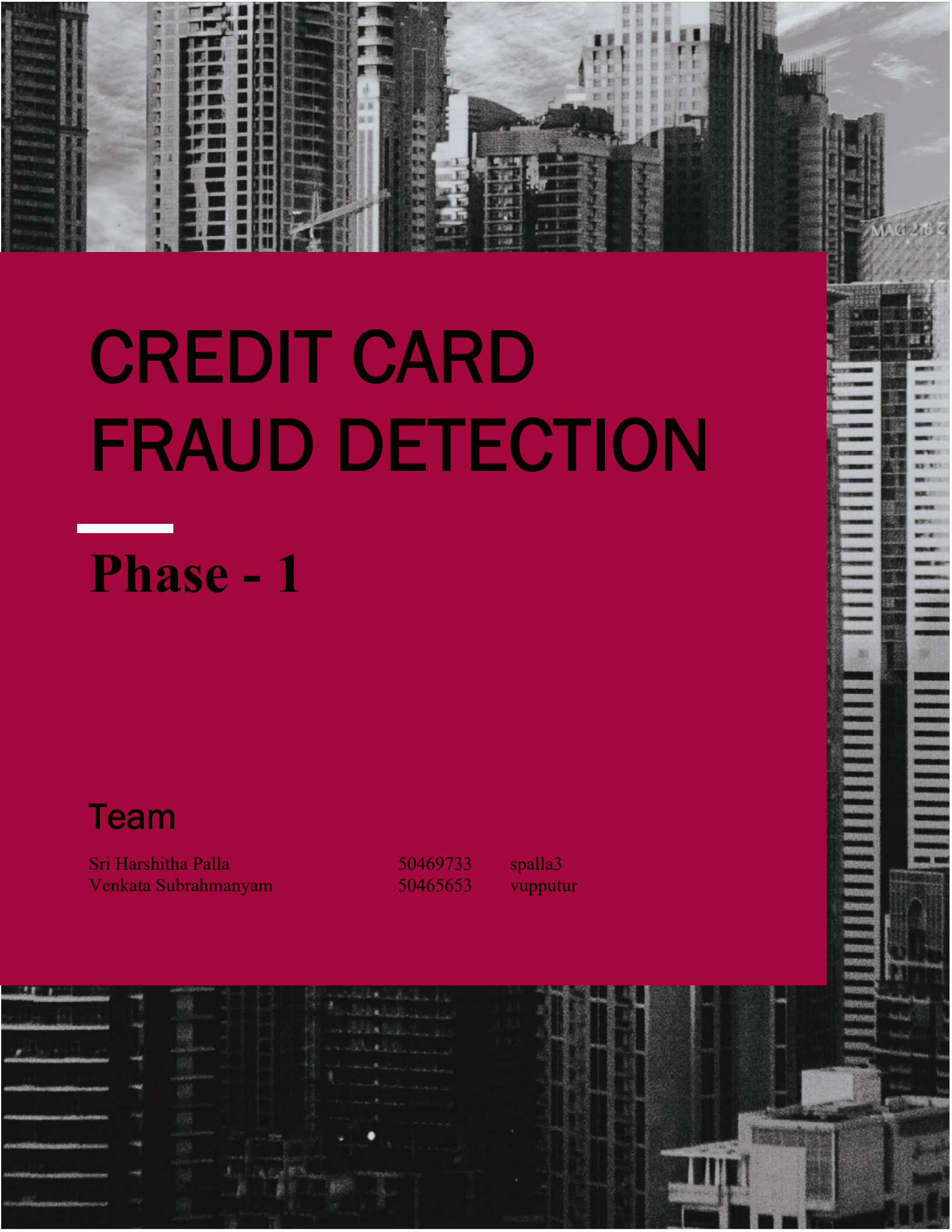


CREDIT CARD FRAUD DETECTION

Phase - 1

Team

Sri Harshitha Palla	50469733	spalla3
Venkata Subrahmanyam	50465653	vupputur

Problem Statement

With the increase in online transactions and e-commerce platforms, credit card fraud has become more common than before. The main objective of this project is to accurately detect fraudulent credit card transactions by using of various Machine Learning Algorithms. We will be using the Credit Card Transactions Fraud Detection Dataset from Kaggle to achieve this.

A. DISCUSS THE BACKGROUND OF THE PROBLEM LEADING TO YOUR OBJECTIVES. WHY IS IT A SIGNIFICANT PROBLEM?

Credit card fraud can result in significant financial losses for both cardholders and financial institutions. Fraudulent transactions can be very large, and often go unnoticed until the victim receives their credit card statement. By detecting and preventing fraud, financial institutions can save themselves and their customers from financial losses. Overall, credit card fraud detection is significant because it helps financial institutions to protect themselves and their customers from financial losses, maintain their reputation, comply with laws and regulations, and provide a better customer experience.

B. EXPLAIN THE POTENTIAL OF YOUR PROJECT TO CONTRIBUTE TO YOUR PROBLEM DOMAIN. DISCUSS WHY THIS CONTRIBUTION IS CRUCIAL?

Financial institutions process large volumes of credit card transactions every day, and data science techniques can help them to quickly and accurately analyze this data to detect and prevent fraud. This project uses machine learning algorithms that can identify patterns and anomalies in credit card transactions that may be indicative of fraudulent activity. These techniques can help financial institutions to identify fraudulent transactions more accurately and efficiently than traditional manual methods.

Data Sources

Dataset: Credit Card Transactions Fraud Detection Dataset

This dataset is downloaded from Kaggle[1]. This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers with a pool of 800 merchants

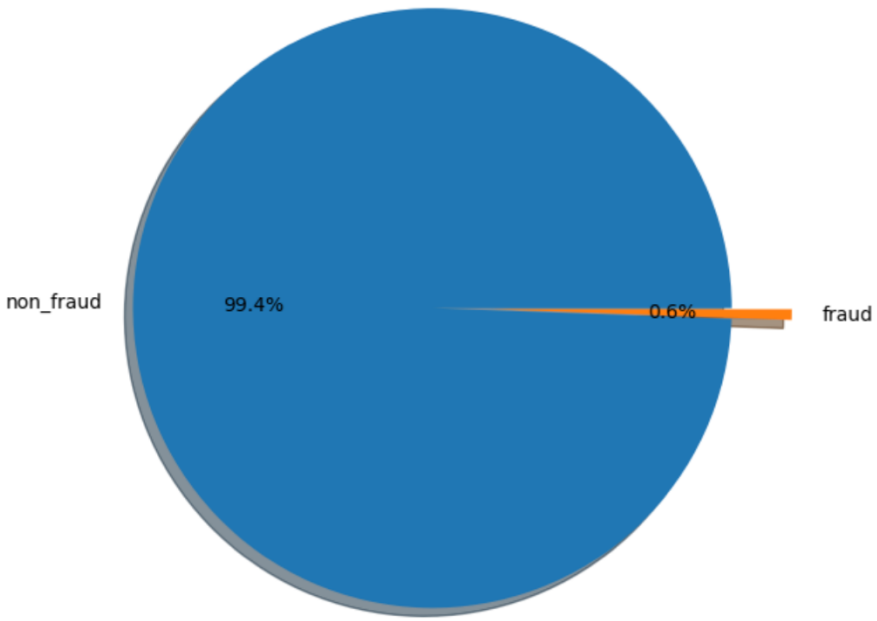
This Dataset contains two csv files – fraudTrain.csv and fraudTest.csv

1. fraudTrain.csv

   - 1048575 row and 23 columns

```
fraudTrain.shape
(1048575, 23)
```

   - Distribution of fraud and non-fraud transactions in the dataset



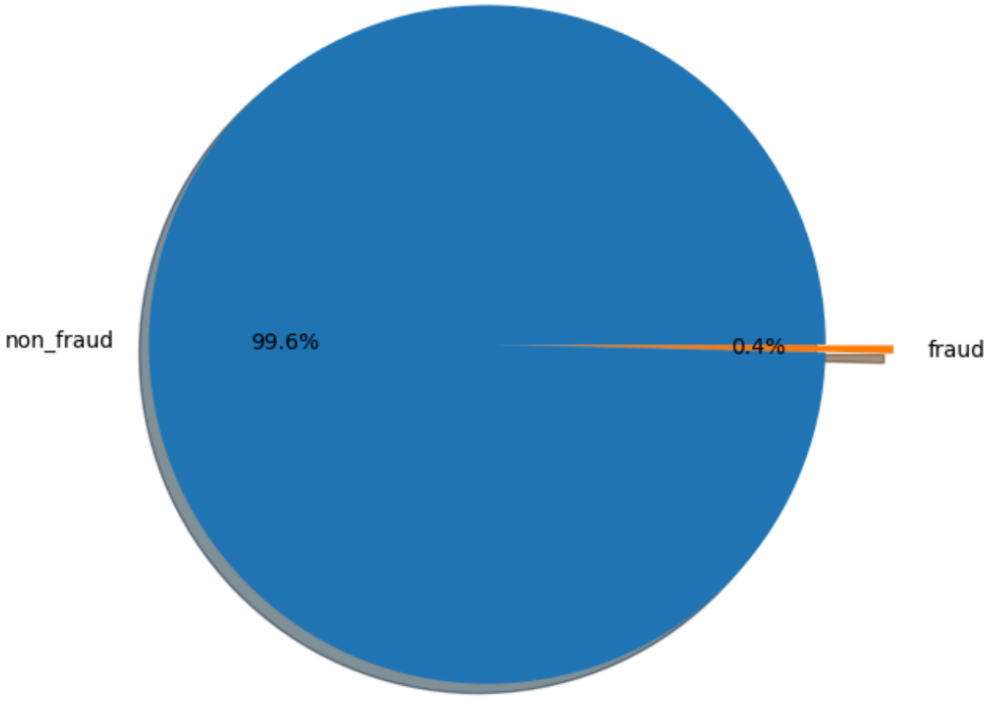
2. fraudTest.csv

- 555719 row and 23 columns

```
fraudTest.shape
```

```
(555719, 23)
```

- Distribution of fraud and non-fraud transactions in the dataset



Data Cleaning

1. Data Concatenation[2]

We will be combining the two csv files to facilitate data cleaning and preprocessing. We have added a new label to the files which will classify them. Next, we combined the two files.

```
# combining the two datasets

fraudTrain['label'] = 'fraudTrain'
fraudTest['label'] = 'fraudTest'

data = pd.concat([fraudTrain,fraudTest],axis=0,ignore_index=True)
print(data.shape)
data.head(5)
```

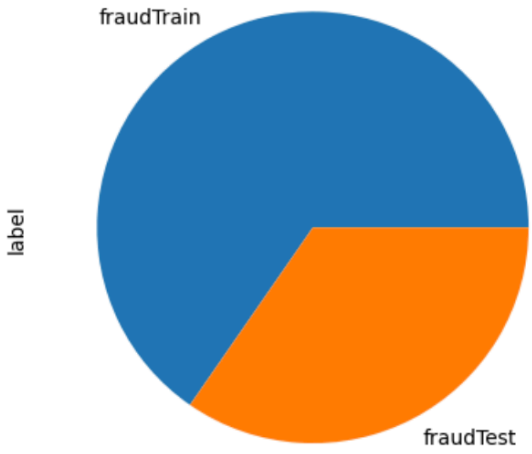
```
(1604294, 23)
```

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	...	long	city_pop	j
0	1/1/2019 0:00	2.703190e+15	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	Moravian Falls	...	-81.1781	3495.0	Psychologi counselli
1	1/1/2019 0:00	6.304230e+11	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	Orient	...	-118.2105	149.0	Spec educatior needs teach
2	1/1/2019 0:00	3.885950e+13	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	Malad City	...	-112.2620	4154.0	Natu conservati offic
3	1/1/2019 0:01	3.534090e+15	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	Boulder	...	-112.1138	1939.0	Patent attorn
4	1/1/2019 0:03	3.755340e+14	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	Doe Hill	...	-79.4629	99.0	Dan moveme psychotherap

5 rows × 23 columns

```
data['label'].value_counts().plot(kind='pie')
```

```
<AxesSubplot:ylabel='label'>
```



2. Removing null or missing values[3]

Count of null or missing values in each column:

```
data.isna().sum()
```

```
Unnamed: 0                0
trans_date_trans_time     1
cc_num                    2
merchant                  2
category                  0
amt                       5
first                     2
last                      5
gender                    7
street                    3
city                      6
state                     6
zip                       3
lat                       2
long                      4
city_pop                  5
job                       3
dob                       2
trans_num                 7
unix_time                 0
merch_lat                 1
merch_long                5
is_fraud                  0
label                     0
dtype: int64
```

After removing null or missing valued rows:

```
#removing null or missing values
data.dropna(inplace = True)
```

```
data.shape
```

```
(1604223, 24)
```

3. Removing duplicate rows[4]

```
#removing null or missing values
data.dropna(inplace = True)
```

```
data.shape
```

```
(1604223, 24)
```

4. Modifying datatypes[5]

Changing datatype of ‘trans_date_trans_time’ and ‘dob’ columns from object to datetime format

Initial datatypes:

```
data.dtypes
```

```
Unnamed: 0                 int64
trans_date_trans_time     object
cc_num                     int64
merchant                  object
category                  object
amt                      float64
first                     object
last                      object
gender                    object
street                    object
city                      object
state                     object
zip                        int64
lat                      float64
long                     float64
city_pop                   int64
job                       object
dob                       object
trans_num                 object
unix_time                  int64
merch_lat                float64
merch_long               float64
is_fraud                   int64
label                     object
dtype: object
```

Changing datatypes:

```
#changing object to datetime format

data['trans_date_trans_time'] = pd.to_datetime(data['trans_date_trans_time'])
data['dob'] = pd.to_datetime(data['dob'])
```

Datatypes after changing:

```
data.dtypes
```

```
Unnamed: 0                          int64
trans_date_trans_time    datetime64[ns]
cc_num                             int64
merchant                          object
category                          object
amt                              float64
first                             object
last                              object
gender                            object
street                            object
city                              object
state                             object
zip                                int64
lat                              float64
long                             float64
city_pop                           int64
job                               object
dob                      datetime64[ns]
trans_num                         object
unix_time                          int64
merch_lat                        float64
merch_long                       float64
is_fraud                           int64
label                             object
dtype: object
```

5. Splitting columns[6]

We observe that ‘trans_date_trans_time’ has a lot of unique values. So we will split the column into year-month, day, hour and add it to the dataset

```
data['trans_date_trans_time'].value_counts().sum()
```

```
1604223
```

```
#split trans_date_trans_time to year-month, day, hour

data['hour'] = data['trans_date_trans_time'].dt.hour
data['day'] = data['trans_date_trans_time'].dt.day_name().str[:3]
data['year-month'] = data['trans_date_trans_time'].dt.to_period('M')
```

New columns of the dataset:

```
data.columns
```

```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud', 'label', 'hour', 'day',
       'year-month'],
      dtype='object')
```

```
data.head(5)
```

t	first	last	gender	street	...	dob	trans_num	unix_time	merch_lat	merch_long	is_fraud	label	hour	day	year-month
'	Jennifer	Banks	F	561 Perry Cove	...	1988-03-09	0b242abb623afc578575680df30655b9	1325376018	36.011293	-82.048315	0	fraudTrain	0	Tue	2019-01
3	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	1978-06-21	1f76529f8574734946361c461b024d99	1325376044	49.159047	-118.186462	0	fraudTrain	0	Tue	2019-01
l	Edward	Sanchez	M	594 White Dale Suite 530	...	1962-01-19	a1a22d70485983eac12b5b88dad1cf95	1325376051	43.150704	-112.154481	0	fraudTrain	0	Tue	2019-01
)	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	1967-01-12	6b849c168bdad6f867558c3793159a81	1325376076	47.034331	-112.561071	0	fraudTrain	0	Tue	2019-01
5	Tyler	Garcia	M	408 Bradley Rest	...	1986-03-28	a41d7549acf90789359a9aa5346dcb46	1325376186	38.674999	-78.632459	0	fraudTrain	0	Tue	2019-01

6. Text to lower case[7]

Converting all columns having text to lower case

```
#conver all text to lower case

data['first'] = data['first'].str.lower()
data['merchant'] = data['merchant'].str.lower()
data['category'] = data['category'].str.lower()
data['last'] = data['last'].str.lower()
data['gender'] = data['gender'].str.lower()
data['street'] = data['street'].str.lower()
data['city'] = data['city'].str.lower()
data['state'] = data['state'].str.lower()
data['job'] = data['job'].str.lower()
```

Now the text is in lower case

```
data.head(10)
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	dob	
0	0	2019-01-01 00:00:00	2.703190e+15	fraud_rippin, kub and mann	misc_net	4.97	jennifer	banks	f	561 perry cove	...	1988-03-09	0b242abb623afc57857568
1	1	2019-01-01 00:00:00	6.304230e+11	fraud_heller, gutmann and zieme	grocery_pos	107.23	stephanie	gill	f	43039 riley greens suite 393	...	1978-06-21	1f76529f8574734946361c
2	2	2019-01-01 00:00:00	3.885950e+13	fraud_lind-buckridge	entertainment	220.11	edward	sanchez	m	594 white dale suite 530	...	1962-01-19	a1a22d70485983eac12b5b
3	3	2019-01-01 00:01:00	3.534090e+15	fraud_kutch, hermiston and farrell	gas_transport	45.00	jeremy	white	m	9443 cynthia court apt. 038	...	1967-01-12	6b849c168bdad6f867558c
4	4	2019-01-01 00:03:00	3.755340e+14	fraud_keeling-crist	misc_pos	41.96	tyler	garcia	m	408 bradley rest	...	1986-03-28	a41d7549acf90789359a9a

7. Removing unwanted text[8]

We will be removing ‘fraud_’ from merchant column

```
#remmoving unwated text

data['merchant'] = data['merchant'].str[6:]
```

New column values are as follows:

```
data['merchant']
```

```
0                    rippin, kub and mann
1              heller, gutmann and zieme
2                          lind-buckridge
3           kutch, hermiston and farrell
4                           keeling-crist
                          ...
555714                   reilly and sons
555715                    hoppe-parisian
555716                         rau-robel
555717                   breitenberg llc
555718                       dare-marvin
Name: merchant, Length: 1604223, dtype: object
```

8. Removing extra punctuations[9]

```
#removing extra punctuations

data=data.replace("!!","!")
data=data.replace("??","?")
data=data.replace("..",".")
data=data.replace("**","*")
data=data.replace(",,",",")
data=data.replace("//","/")
data=data.replace("__","_")
data=data.replace("--","-")
```

9. Creating a new column – age[10]

We will create a new column age with the help of ‘trans_date_trans_time’ and ‘dob’. This will help us to analyze data wrt age of the card holders.

```
#creating new column

data['age'] = np.round((data['trans_date_trans_time']-data['dob'])/np.timedelta64(1,'Y'))
```

```
data.columns
```

```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud', 'label', 'hour', 'day',
       'year-month', 'age'],
      dtype='object')
```

We observe the datatype of this column to be float

```
data.dtypes
```

```
Unnamed: 0                      int64
trans_date_trans_time    datetime64[ns]
cc_num                          float64
merchant                         object
category                         object
amt                             float64
first                            object
last                             object
gender                           object
street                           object
city                             object
state                            object
zip                             float64
lat                             float64
long                            float64
city_pop                        float64
job                              object
dob                      datetime64[ns]
trans_num                        object
unix_time                         int64
merch_lat                       float64
merch_long                      float64
is_fraud                          int64
label                            object
hour                              int64
day                              object
year-month                   period[M]
age                             float64
dtype: object
```

Therefore, we change it to int

```
data['age']=data['age'].astype(int)
```

```
data.dtypes
```

```
Unnamed: 0                      int64
trans_date_trans_time    datetime64[ns]
cc_num                          float64
merchant                         object
category                         object
amt                             float64
first                            object
last                             object
gender                           object
street                           object
city                             object
state                            object
zip                             float64
lat                             float64
long                            float64
city_pop                        float64
job                              object
dob                      datetime64[ns]
trans_num                        object
unix_time                         int64
merch_lat                       float64
merch_long                      float64
is_fraud                          int64
label                            object
hour                              int64
day                              object
year-month                   period[M]
age                               int64
dtype: object
```

```
data.head(5)
```

amt	first	last	gender	street	...	trans_num	unix_time	merch_lat	merch_long	is_fraud	label	hour	day	year-month	age
.97	jennifer	banks	f	561 perry cove	...	0b242abb623afc578575680df30655b9	1325376018	36.011293	-82.048315	0	fraudTrain	0	Tue	2019-01	31
.23	stephanie	gill	f	43039 riley greens suite 393	...	1f76529f8574734946361c461b024d99	1325376044	49.159047	-118.186462	0	fraudTrain	0	Tue	2019-01	41
.11	edward	sanchez	m	594 white dale suite 530	...	a1a22d70485983eac12b5b88dad1cf95	1325376051	43.150704	-112.154481	0	fraudTrain	0	Tue	2019-01	57
.00	jeremy	white	m	9443 cynthia court apt. 038	...	6b849c168bdad6f867558c3793159a81	1325376076	47.034331	-112.561071	0	fraudTrain	0	Tue	2019-01	52
.96	tyler	garcia	m	408 bradley rest	...	a41d7549acf90789359a9aa5346dcb46	1325376186	38.674999	-78.632459	0	fraudTrain	0	Tue	2019-01	33

10. Removing unwanted columns[11]

```
#removing unwanted columns

data.drop(['trans_date_trans_time','dob', 'first', 'last'] , axis=1, inplace=True)
```

```
data.columns
```

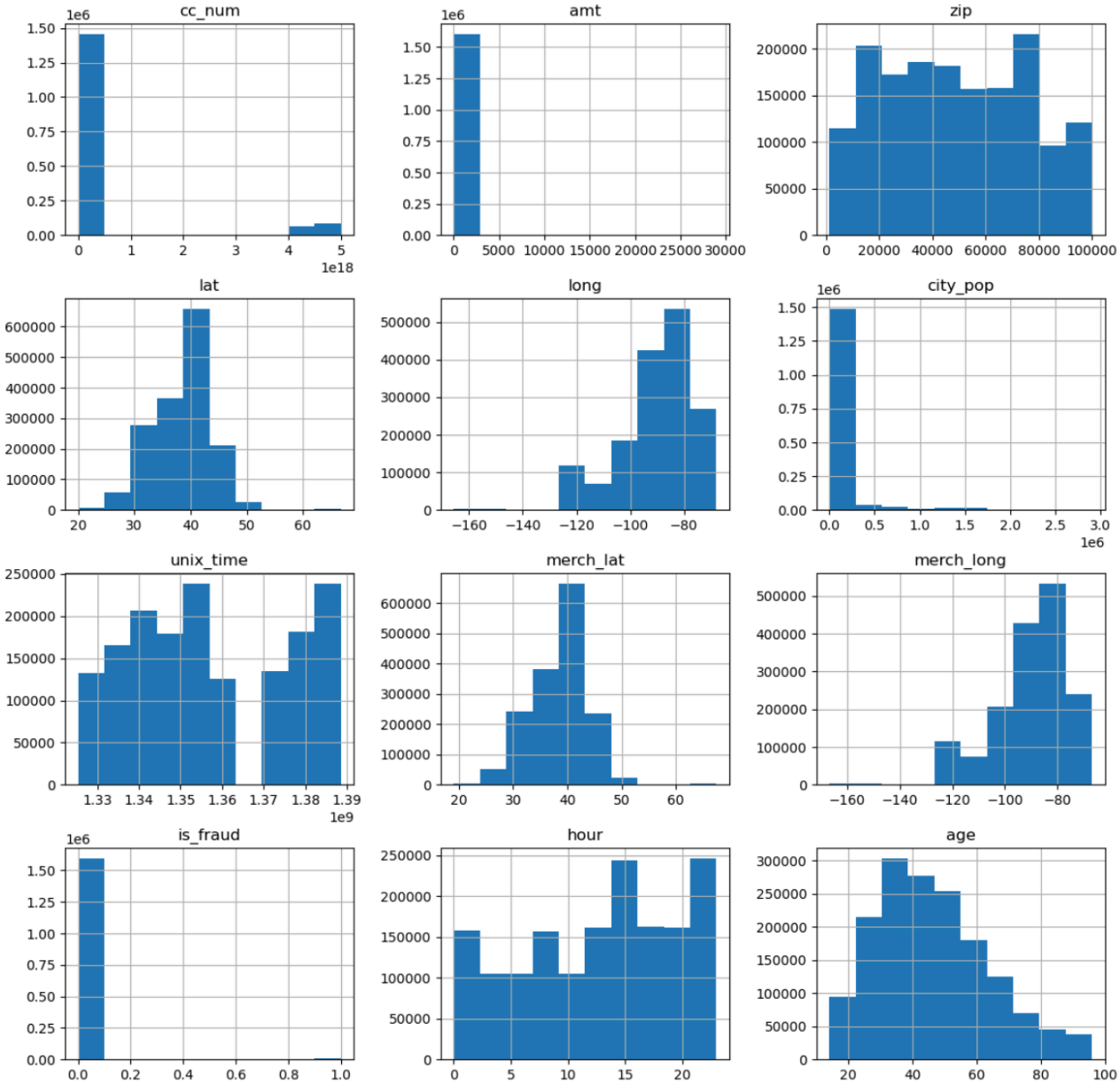
```
Index(['cc_num', 'merchant', 'category', 'amt', 'gender', 'street', 'city',
       'state', 'zip', 'lat', 'long', 'city_pop', 'job', 'trans_num',
       'unix_time', 'merch_lat', 'merch_long', 'is_fraud', 'label', 'hour',
       'day', 'year-month', 'age'],
      dtype='object')
```

Exploratory Data Analysis (EDA)

1. Histogram representation of the dataset[12]

```
data.hist(figsize=(14,14))
```

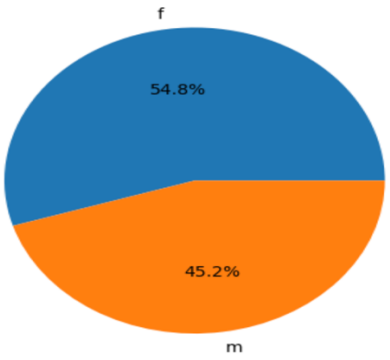
```
array([[<AxesSubplot:title={'center':'cc_num'}>,
        <AxesSubplot:title={'center':'amt'}>,
        <AxesSubplot:title={'center':'zip'}>],
       [<AxesSubplot:title={'center':'lat'}>,
        <AxesSubplot:title={'center':'long'}>,
        <AxesSubplot:title={'center':'city_pop'}>],
       [<AxesSubplot:title={'center':'unix_time'}>,
        <AxesSubplot:title={'center':'merch_lat'}>,
        <AxesSubplot:title={'center':'merch_long'}>],
       [<AxesSubplot:title={'center':'is_fraud'}>,
        <AxesSubplot:title={'center':'hour'}>,
        <AxesSubplot:title={'center':'age'}>]], dtype=object)
```



2. Trends in fraud and non-fraud transactions on the basis of gender[13]

We observe here that Females have more transactions than Males in out dataset

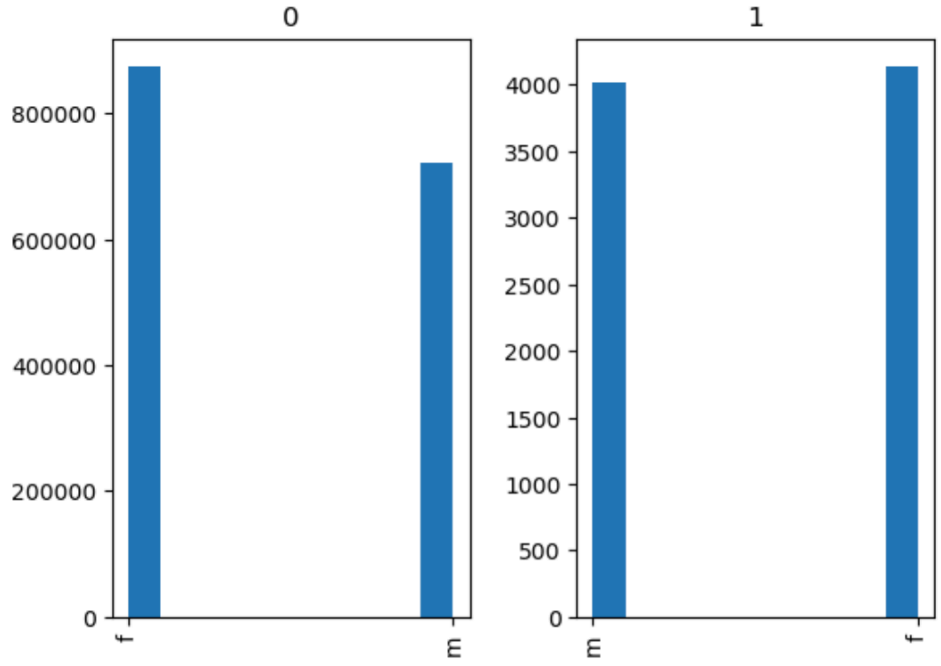
```
# gender pie chart
plt.pie(data['gender'].value_counts().values, labels=data['gender'].value_counts().index, autopct='%1.1f%%')
plt.show()
```



Here, we observe that males have comparatively less non fraud transactions than females. Also, count of fraud transaction is slightly more for females

```
#non-fraud and fraud transaction distribution on the basis of gender

data.hist(column="gender", by="is_fraud")
```

```
array([<AxesSubplot:title={'center':'0'}>,
       <AxesSubplot:title={'center':'1'}>], dtype=object)
```



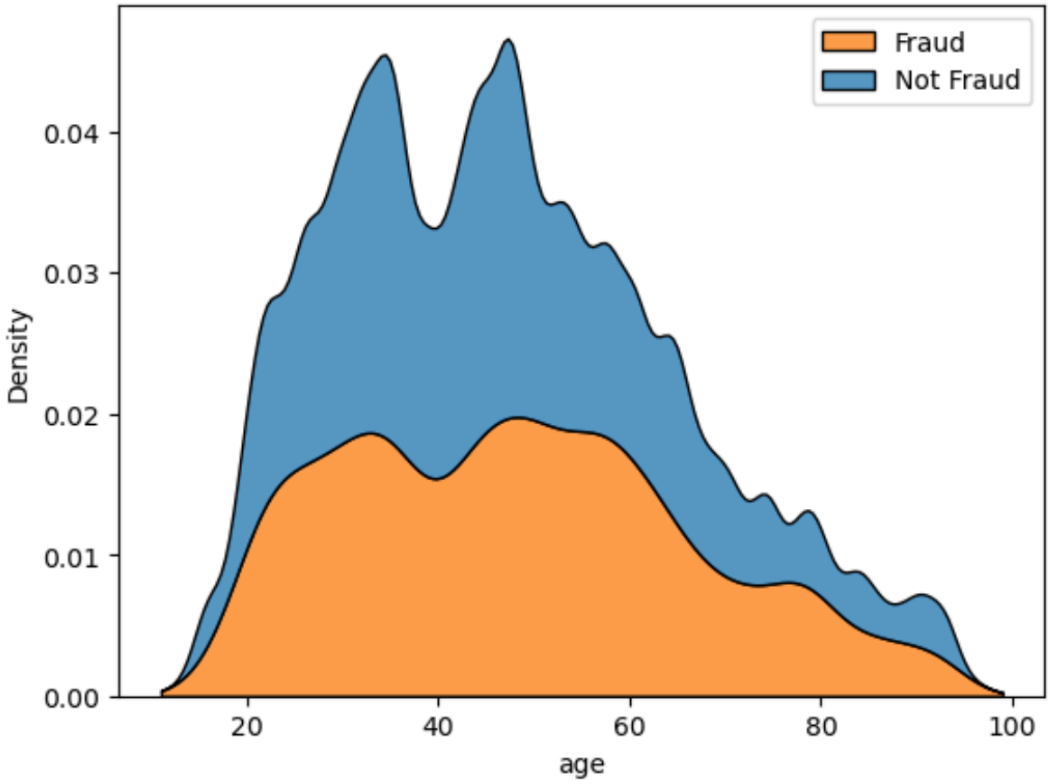
3. Fraudulent transactions distribution with respect to age of card holders[14]

Kernel density estimate (KDE) plot represents the data using a continuous probability density curve in one or more dimensions.

Here, we can observe that the ages 20-60 have more fraudulent transactions. Also, a dip is observed for age 40.

```
sns.kdeplot(data=data,x='age', hue='is_fraud',common_norm=False, multiple="stack")

plt.legend(labels=['Fraud', 'Not Fraud'])
```

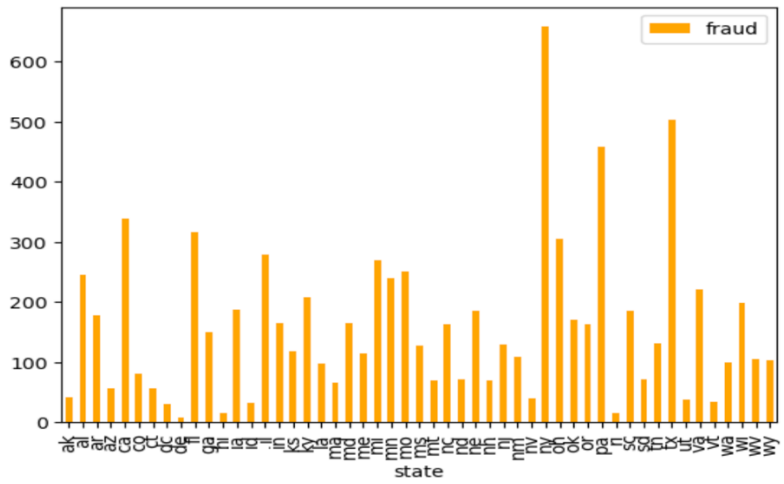
```
<matplotlib.legend.Legend at 0x7ff55c41aa90>
```



4. Bar plots for fraud transactions and non-fraud transactions in each state[15]

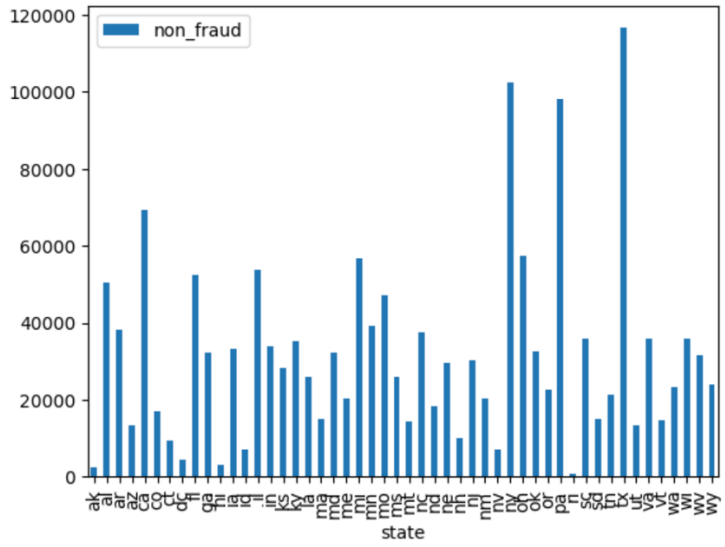
Fraud transactions:

```
# Fraud transactions

fraud_data = data[data['is_fraud'] == True]
fraud = fraud_data.groupby('state').size().reset_index(name='fraud')
fig, ax = plt.subplots()
fraud.plot.bar(x='state', y='fraud', ax=ax,color="orange")
plt.show()
```



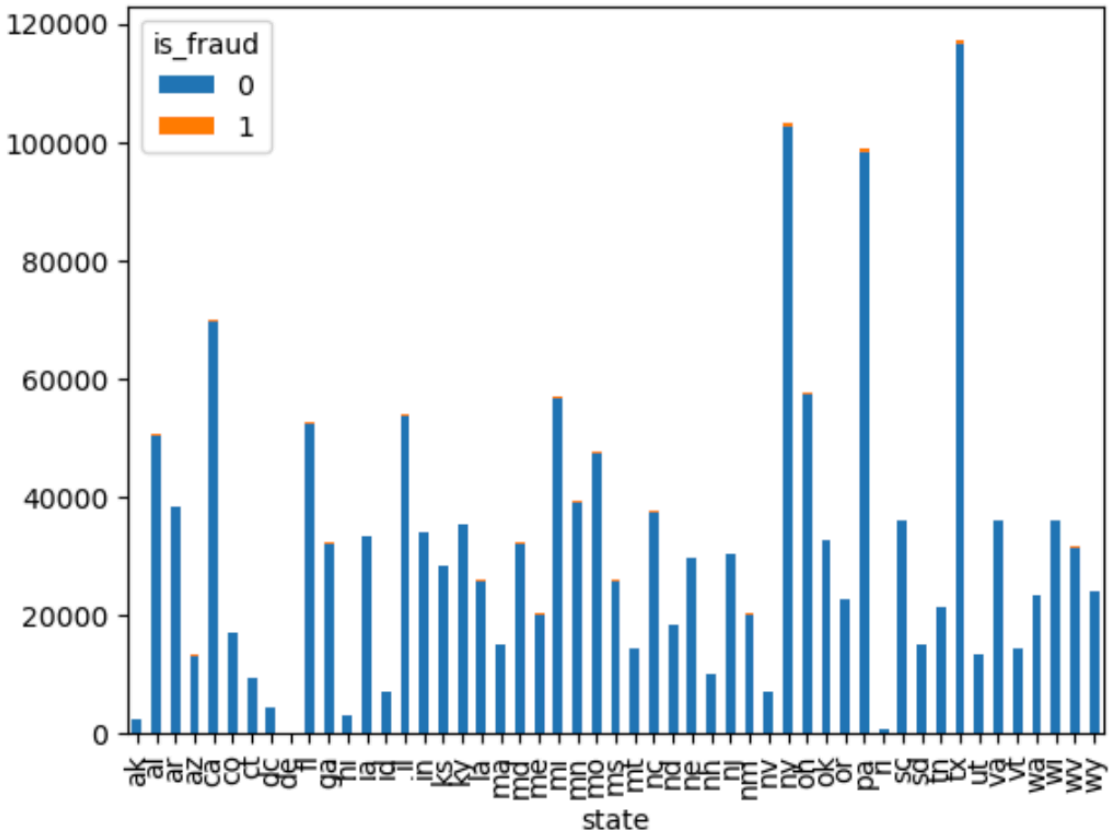
Non-Fraud transactions:

```
# Non-Fraud transactions

fraud_data = data[data['is_fraud'] == False]
non_fraud = fraud_data.groupby('state').size().reset_index(name='non_fraud')
fig, ax = plt.subplots()
non_fraud.plot.bar(x='state', y='non_fraud', ax=ax)
plt.show()
```



All transactions:

```
# Fraud and Non-Fraud transactions

counts = data.groupby(['state', 'is_fraud']).size().reset_index(name='counts')
pivot = counts.pivot(index='state', columns='is_fraud', values='counts')
fig, ax = plt.subplots()
pivot.plot.bar(ax=ax, stacked=True)
plt.show()
```



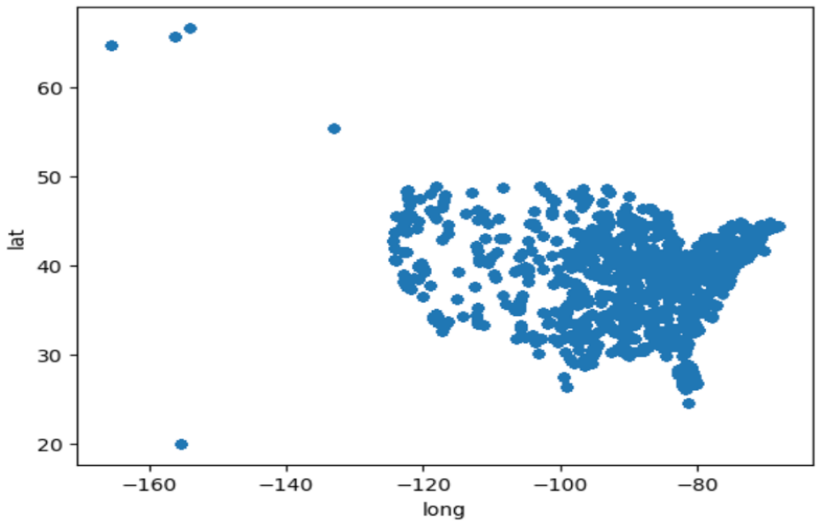
Here, we observe highest number of fraud transactions in NY, followed by Texas. While the highest number of non-fraud transactions are in Texas followed by NY

Overall, Texas records highest number of transactions in our dataset.

5. Geographic distribution of data points[16]

```
#geographic distribution of data points

data.plot(kind="scatter", x = "long", y= "lat")
```

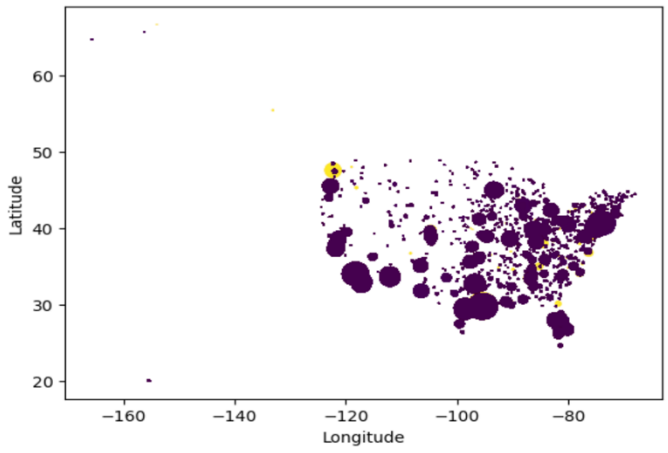
```
<AxesSubplot:xlabel='long', ylabel='lat'>
```



Here, we can observe that the transactions are mostly concentrated.

```
#population and fraud by location

plt.scatter(data['long'], data['lat'], s=data['city_pop']/10000, c  =data.is_fraud, alpha=0.2)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
```

```
Text(0, 0.5, 'Latitude')
```



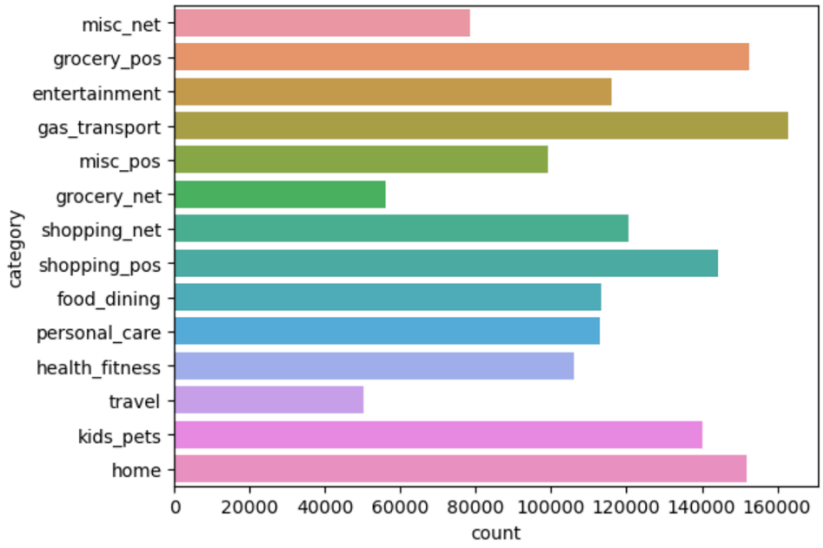
We also observer here that, fraud transactions(yellow) are found in both highly populated and low populated areas.

6. Categorical distribution of data[17]

All transactions:

```
#category wise distribution of data

sns.countplot(y='category', data=data)
```

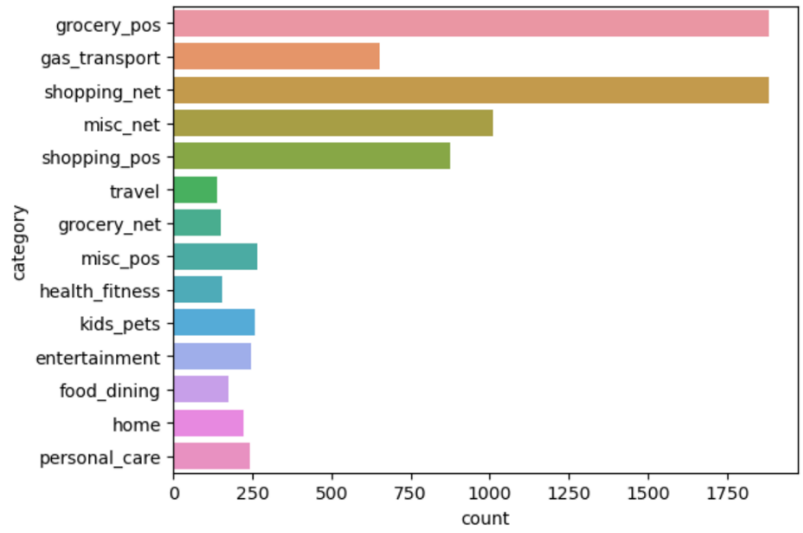
```
<AxesSubplot:xlabel='count', ylabel='category'>
```



Fraud Transactions:

```
#category wise distribution of fraud data

sns.countplot(y='category', data=data_fraud)
```

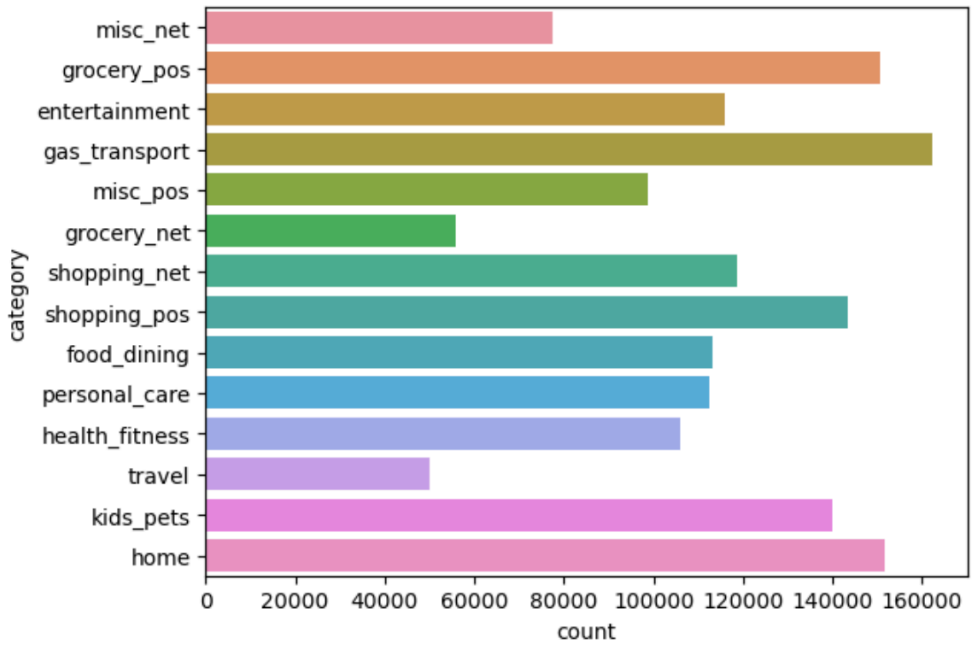
```
<AxesSubplot:xlabel='count', ylabel='category'>
```



Non-fraud transactions:

```
#category wise distribution of non-fraud data

sns.countplot(y='category', data=data_non_fraud)
```

```
<AxesSubplot:xlabel='count', ylabel='category'>
```



Here, we observe max number of transactions are in the categories gas_transport, grocery_pos, home, shopping_pos and kids_pets in the same order.

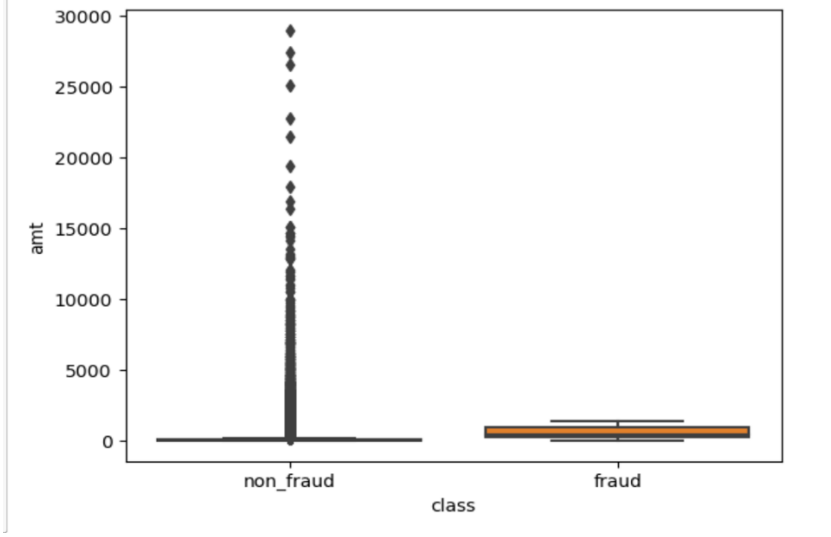
But, the order of max fraud transaction is observed to be different and is in the decreasing order - grocery_pos, shopping_net, misc_net, shopping_pos, gas_transport. Other categories have comparatively less fraud transactions.

Non fraud transactions have a similar distribution as the entire dataset.

7. Box plot for amount distribution[18]

```
sns.boxplot(data = data_copy, x = 'class', y = 'amt')
```

```
<AxesSubplot:xlabel='class', ylabel='amt'>
```



In the boxplot, it is clear that the fraud transactions do not have outlier amount. The concentration is observed with median of 384.910000 which is very high as compared to the non-fraud transactions with a median of 47.2.

Statistics of the data wrt amount:

```
data_copy['class']=data_copy['is_fraud'].map({1:'fraud',0:'non_fraud'})
stats = data_copy.groupby('class')['amt'].agg([np.min,np.max,np.mean,np.median])
stats.transpose()
```

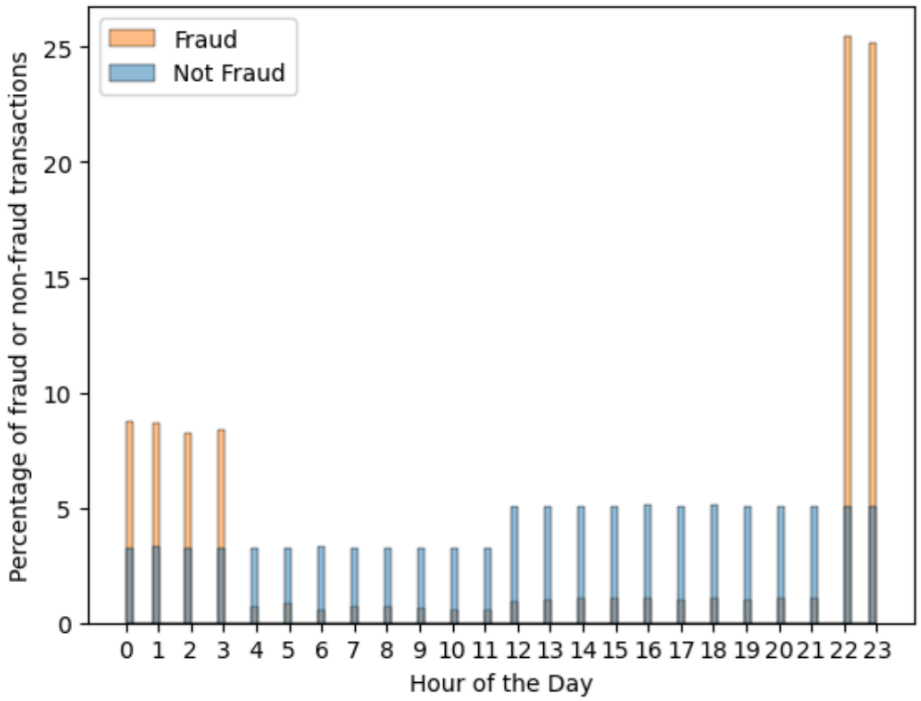
class	fraud	non_fraud
amin	1.180000	1.000000
amax	1371.810000	28948.900000
mean	529.990071	67.621361
median	384.910000	47.200000

We observe, mean value of the fraud transactions is higher (i.e., 529.990071) whereas non fraud transactions have a mean of 67.621361.

8. Percentage of fraud and non-fraud transactions in the hour of the day[19]

```
#percentage of fraud and non-fraud transactions in the hour of the day

hour_plot=sns.histplot(data=data, x="hour", hue="is_fraud", common_norm=False,stat='percent')
plt.xticks(np.arange(0,24,1))
hour_plot.set_ylabel('Percentage of fraud or non-fraud transactions')
hour_plot.set_xlabel('Hour of the Day')
plt.legend(labels=['Fraud', 'Not Fraud'])
```

```
<matplotlib.legend.Legend at 0x7fe29e3b1b80>
```



Observations from this graph make it very clear that the fraud transactions take place in high volume in the midnight.

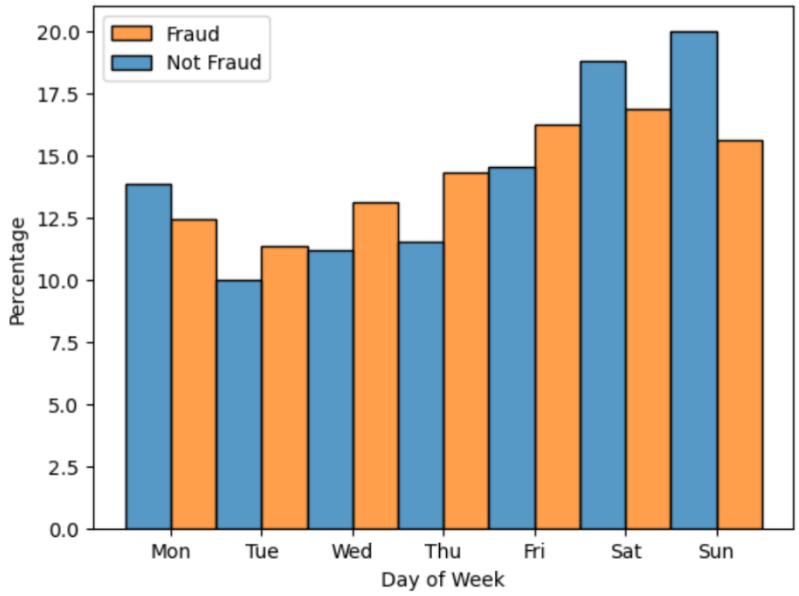
Non-fraud transactions are distributed evenly throughout the day.

9. Percentage of fraud and non-fraud transactions in the days of a week[19]

```
#percentage of fraud and non-fraud transactions in the days of the week

day_plot=sns.histplot(data=data, x="day", hue="is_fraud", common_norm=False,stat='percent',multiple='dodge')
day_plot.set_xticklabels(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
day_plot.set_ylabel('Percentage')
day_plot.set_xlabel('Day of Week')
plt.legend(labels=['Fraud', 'Not Fraud'])
```

```
/var/folders/39/pvtsdq8x67v91gdprr9x6rjm0000gn/T/ipykernel_37099/1648679265.py:10: UserWarning: FixedFormatter
only be used together with FixedLocator
  day_plot.set_xticklabels(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
```

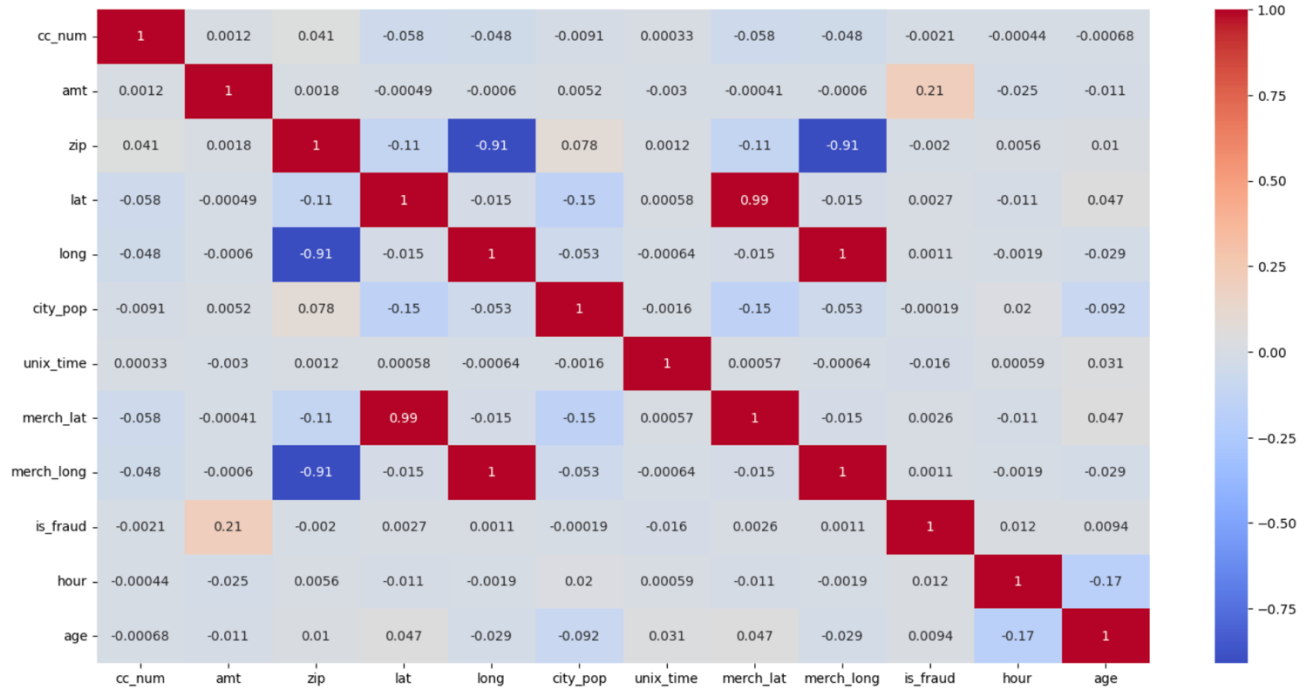
```
<matplotlib.legend.Legend at 0x7fe29eeef340>
```



Non-fraud transactions are observed more on Saturday and Sunday, while fraud transactions happen more on Friday and Saturday.

10. Correlation between columns of the data[20]

```
#correlation between columns of the data

fig = plt.figure(figsize=(18,9))
sns.heatmap(data.corr(),cmap='coolwarm', annot=True)
plt.show()
```



Little correlation observed between amount and fraud status of transactions. Otherwise, there is not much visible correlation.

References

[1]https://www.kaggle.com/datasets/kartik2112/fraud-detection

[2]https://www.tutorialspoint.com/python_pandas/python_pandas_concatenation.htm

[3]https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

[4]https://www.digitalocean.com/community/tutorials/pandas-drop-duplicate-rows-drop_duplicates-function

[5]https://www.statology.org/pandas-change-column-type/

[6]https://sparkbyexamples.com/pandas/pandas-split-column/

[7]https://pandas.pydata.org/docs/reference/api/pandas.Series.str.lower.html

[8]https://stackoverflow.com/questions/13682044/remove-unwanted-parts-from-strings-in-a-column

[9]https://towardsdatascience.com/remove-punctuation-pandas-3e461efe9584

[10]https://pandas.pydata.org/docs/getting_started/intro_tutorials/05_add_columns.html

[11]https://www.geeksforgeeks.org/python-delete-rows-columns-from-dataframe-using-pandas-drop/

[12]https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.hist.html

[13]https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.pie.html

[14]https://seaborn.pydata.org/generated/seaborn.kdeplot.html

[15]https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html

[16]https://pandas.pydata.org/pandas-docs/version/0.15.0/visualization.html

[17]https://seaborn.pydata.org/generated/seaborn.countplot.html

[18]https://seaborn.pydata.org/generated/seaborn.boxplot.html

[19]https://seaborn.pydata.org/generated/seaborn.histplot.html

[20]https://seaborn.pydata.org/generated/seaborn.heatmap.html