Credit Card Fraud Detection

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score,recall_score,accuracy_score,precision_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
#read train and test datasets

df = pd.read_csv('fraudTest.csv',index_col=[0])
```

In [3]:
```
df.shape
```

Out[3]:
```
(555719, 22)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['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'],
      dtype='object')
```

In [5]:
```
df['is_fraud'].value_counts()
```

Out[5]:
```
0    553574
1      2145
Name: is_fraud, dtype: int64
```

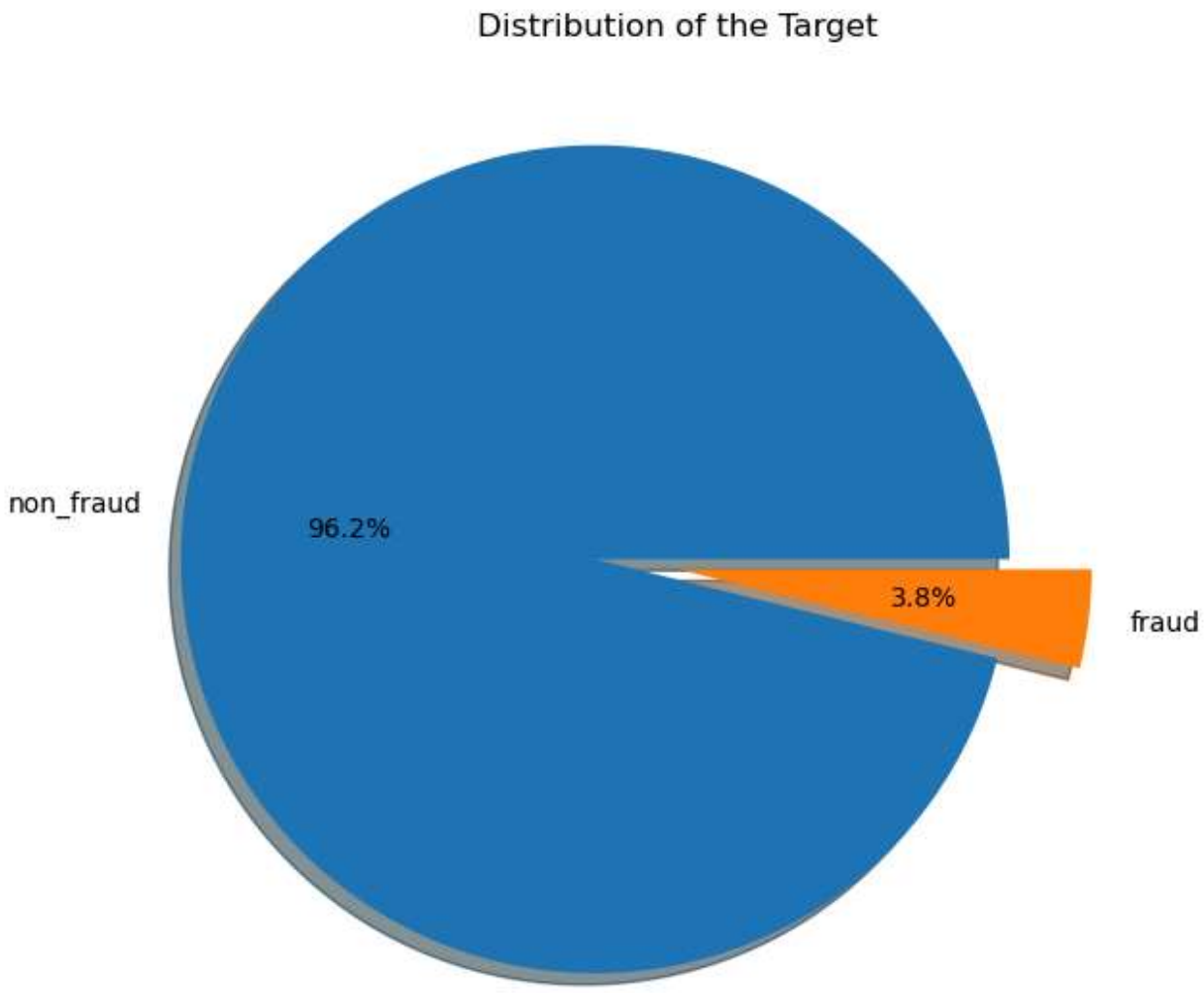
In [6]:
```
fraudulent_rows = df[df['is_fraud'] == 0]
random_selection = fraudulent_rows.sample(n=500000)
df = df.drop(random_selection.index)
df.reset_index(drop=True, inplace=True)
df.shape
```

Out[6]:
```
(55719, 22)
```

In [7]:
```
df['is_fraud'].value_counts()
```

Out[7]:
```
0    53574
1     2145
Name: is_fraud, dtype: int64
```

In [8]:
```
plt.figure(figsize = [7,7])
plot_var = df['is_fraud'].value_counts(normalize = True)
plt.pie(plot_var,
        autopct='%1.1f%%',
        labels = ['non_fraud','fraud'],
        explode = [0.2, 0],
        shadow = True)
plt.title('Distribution of the Target');
```



In [9]:
```
df.head(5)
```

Out[9]:

	trans_date_trans_time	cc_num	merchant	category	amt	first	last
0	2020-06-21 12:15:37	30407675418785	fraud_Daugherty LLC	kids_pets	19.55	Danielle	Evans
1	2020-06-21 12:16:47	571465035400	fraud_Gottlieb Group	kids_pets	42.40	Louis	Fisher
2	2020-06-21 12:18:41	4570636521433188	fraud_Welch, Rath and Koepp	entertainment	24.73	Christine	Leblanc
3	2020-06-21 12:25:09	379897244598068	fraud_Kautzer and Sons	personal_care	176.23	Frank	Key
4	2020-06-21 12:26:13	4302475216404898	fraud_Zemlak, Tillman and Cremin	personal_care	19.03	Daniel	Cain

5 rows × 22 columns

Data Cleaning

In [11]:
```
data = df
data.isna().sum()
```

Out[11]:
```
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
```

In [12]:
```
#Removing duplicate rows

data.drop_duplicates(inplace=True)
```

In [13]:
```
data.shape
```

Out[13]:
```
(55719, 22)
```

In [14]:
```
data.dtypes
```

Out[14]:
```
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
dtype: object
```

In [15]:
```
#Modifying 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'])
```

In [16]:
```
data.dtypes
```

Out[16]:
```
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
dtype: object
```

In [17]:
```
data.head(5)
```

Out[17]:

	trans_date_trans_time	cc_num	merchant	category	amt	first	last
0	2020-06-21 12:15:37	30407675418785	fraud_Daugherty LLC	kids_pets	19.55	Danielle	Evans
1	2020-06-21 12:16:47	571465035400	fraud_Gottlieb Group	kids_pets	42.40	Louis	Fisher
2	2020-06-21 12:18:41	4570636521433188	fraud_Welch, Rath and Koepp	entertainment	24.73	Christine	Leblanc
3	2020-06-21 12:25:09	379897244598068	fraud_Kautzer and Sons	personal_care	176.23	Frank	Key
4	2020-06-21 12:26:13	4302475216404898	fraud_Zemlak, Tillman and Cremin	personal_care	19.03	Daniel	Cain

5 rows × 22 columns

In [18]:
```
data['trans_date_trans_time'].value_counts().sum()
```

Out[18]:
```
55719
```

In [19]:
```
#Splitting columns
#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')
```

In [20]:
```
data.columns
```

Out[20]:
```
Index(['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', 'hour', 'day', 'year-month'],
      dtype='object')
```

In [21]:
```
data.head(5)
```

Out[21]:

	trans_date_trans_time	cc_num	merchant	category	amt	first	last
0	2020-06-21 12:15:37	30407675418785	fraud_Daugherty LLC	kids_pets	19.55	Danielle	Evans
1	2020-06-21 12:16:47	571465035400	fraud_Gottlieb Group	kids_pets	42.40	Louis	Fisher
2	2020-06-21 12:18:41	4570636521433188	fraud_Welch, Rath and Koepp	entertainment	24.73	Christine	Leblanc
3	2020-06-21 12:25:09	379897244598068	fraud_Kautzer and Sons	personal_care	176.23	Frank	Key
4	2020-06-21 12:26:13	4302475216404898	fraud_Zemlak, Tillman and Cremin	personal_care	19.03	Daniel	Cain

5 rows × 25 columns

In [22]:
```
data.columns
```

Out[22]:
```
Index(['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', 'hour', 'day', 'year-month'],
      dtype='object')
```

In [23]:
```
#Creating a new column - age

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

In [24]:
```
data.columns
```

Out[24]:
```
Index(['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', 'hour', 'day', 'year-month', 'age'],
      dtype='object')
```

In [25]:
```
data.dtypes
```

Out[25]:
```
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
hour                              int64
day                              object
year-month                   period[M]
age                             float64
dtype: object
```

In [26]:
```
data['age']=data['age'].astype(int)
```

In [27]:
```
data.dtypes
```

Out[27]:
```
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
hour                              int64
day                              object
year-month                     period[M]
age                               int32
dtype: object
```

In [28]:
```
#Removing unwanted columns

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

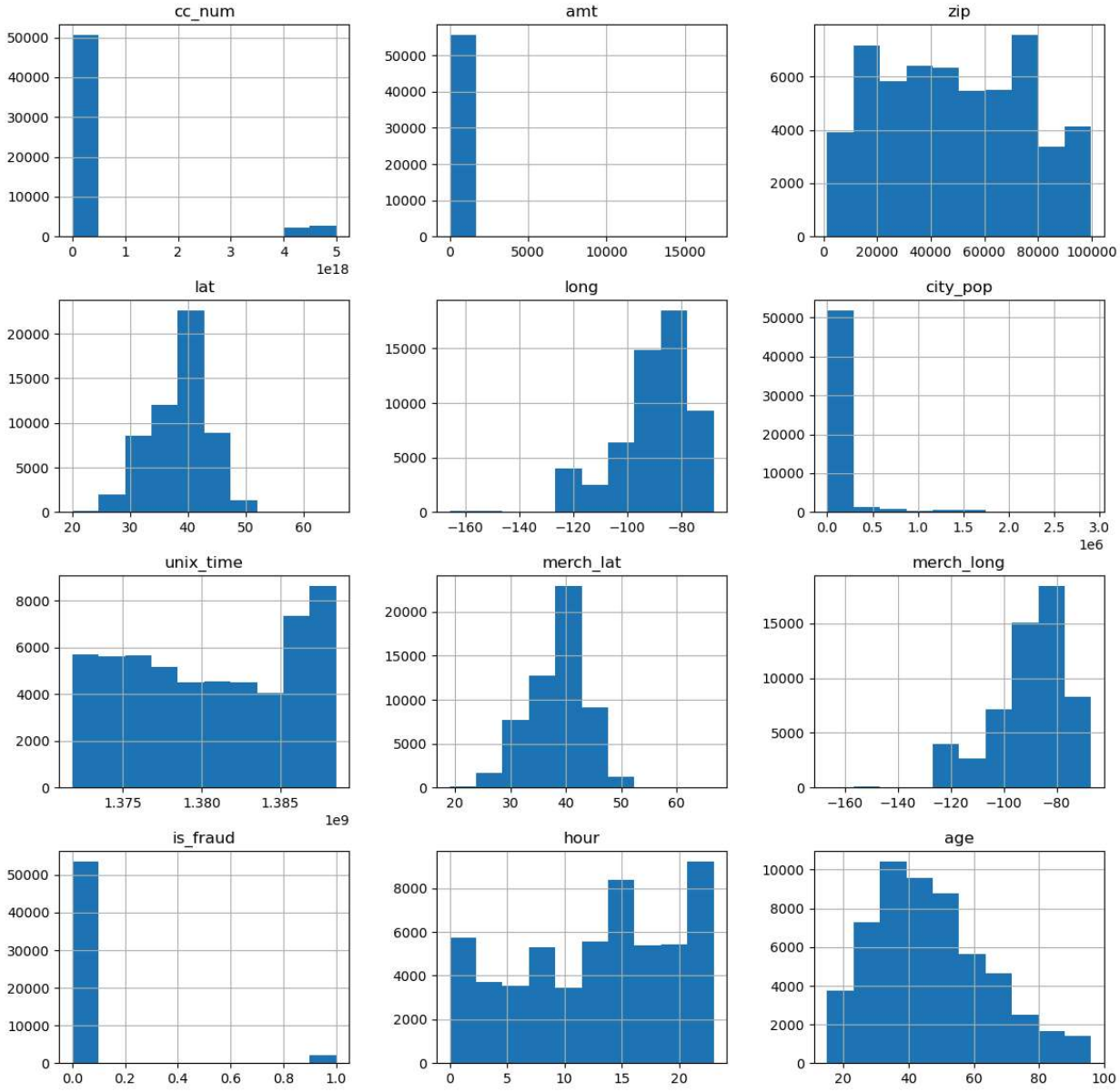
In [29]:
```
data.columns
```

Out[29]:
```
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', 'hour', 'day',
       'year-month', 'age'],
      dtype='object')
```

Exploratory Data Analysis (EDA)

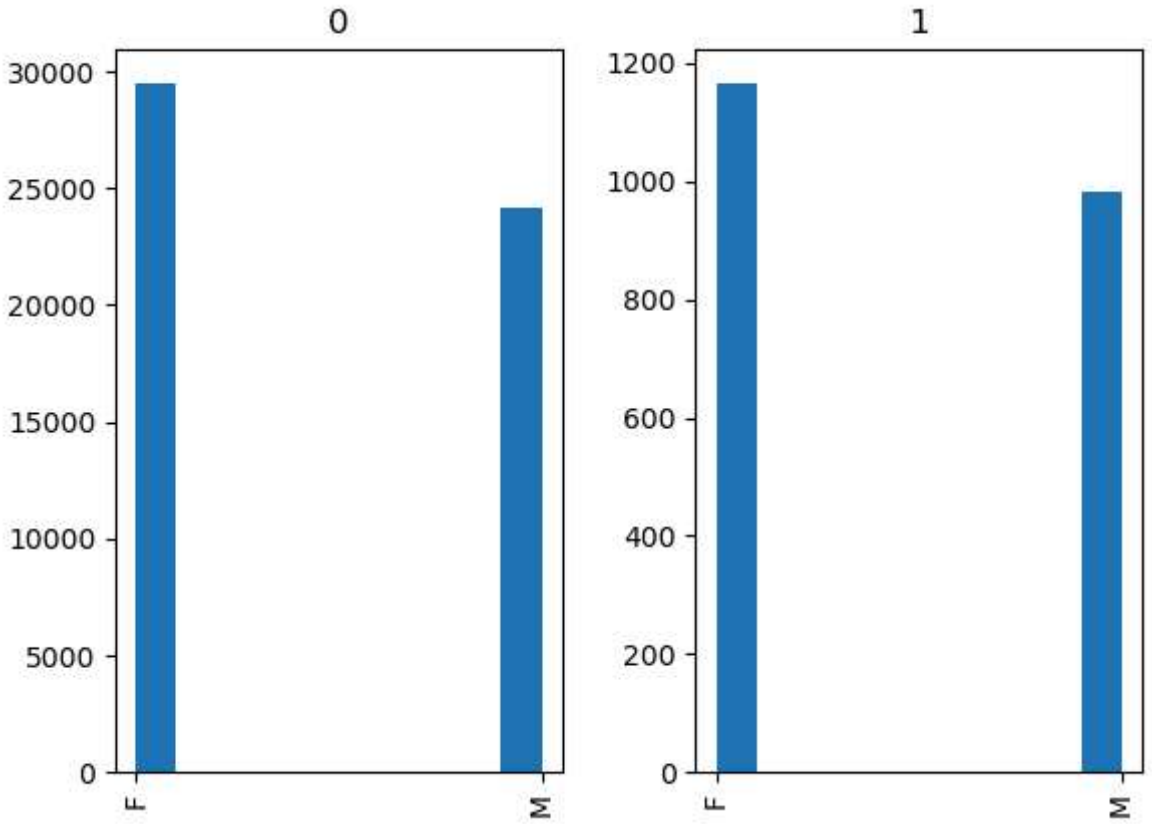
In [30]:
```
#Histogram representation of the dataset
data.hist(figsize=(14,14))
```

Out[30]:
```
array([[<Axes: title={'center': 'cc_num'}>,
        <Axes: title={'center': 'amt'}>, <Axes: title={'center': 'zip'}>],
       [<Axes: title={'center': 'lat'}>,
        <Axes: title={'center': 'long'}>,
        <Axes: title={'center': 'city_pop'}>],
       [<Axes: title={'center': 'unix_time'}>,
        <Axes: title={'center': 'merch_lat'}>,
        <Axes: title={'center': 'merch_long'}>],
       [<Axes: title={'center': 'is_fraud'}>,
        <Axes: title={'center': 'hour'}>,
        <Axes: title={'center': 'age'}>]], dtype=object)
```

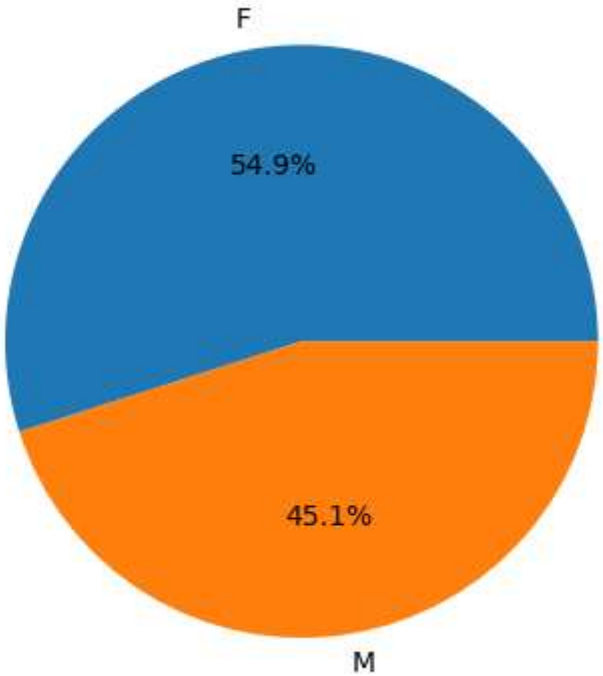


In [31]:
```
#non-fraud and fraud transaction distribution on the basis of gender

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

Out[31]:
```
array([<Axes: title={'center': '0'}>, <Axes: title={'center': '1'}>],
      dtype=object)
```



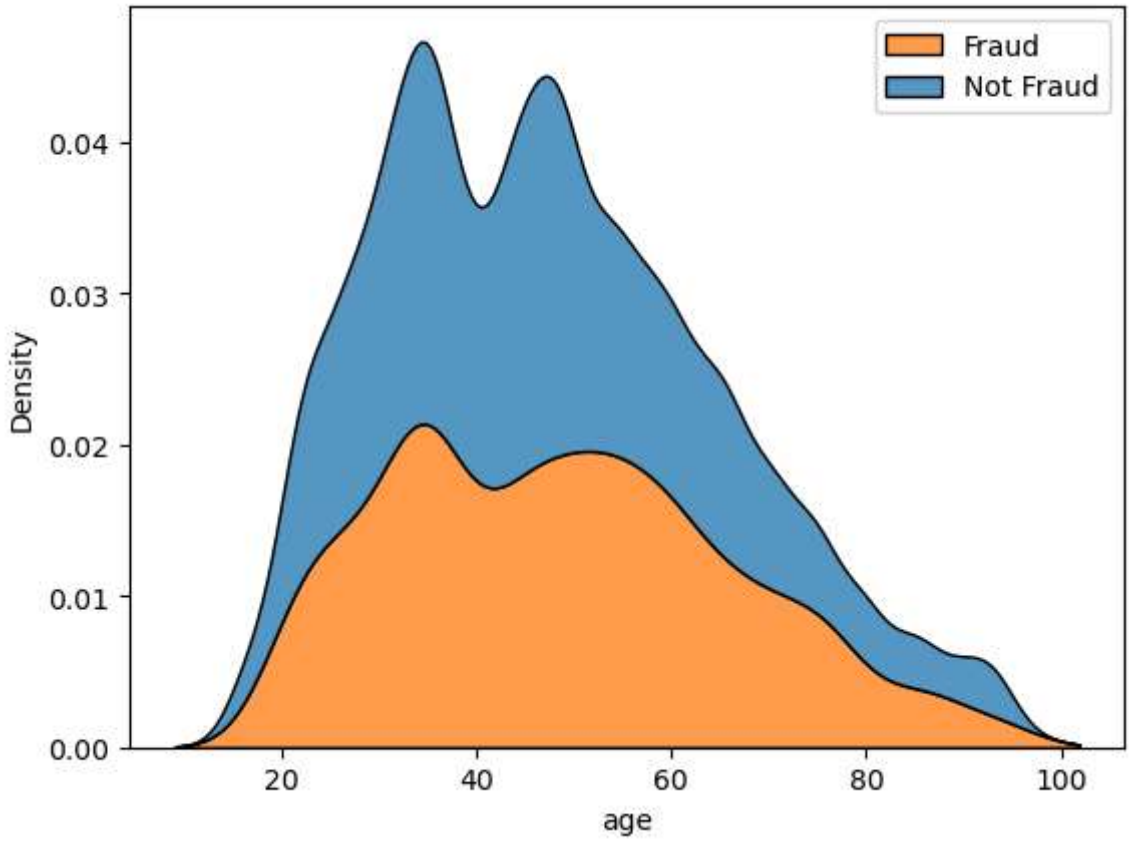
In [32]:
```
# gender pie chart

plt.pie(data['gender'].value_counts().values, labels=data['gender'].value_counts().inc
plt.show()
```



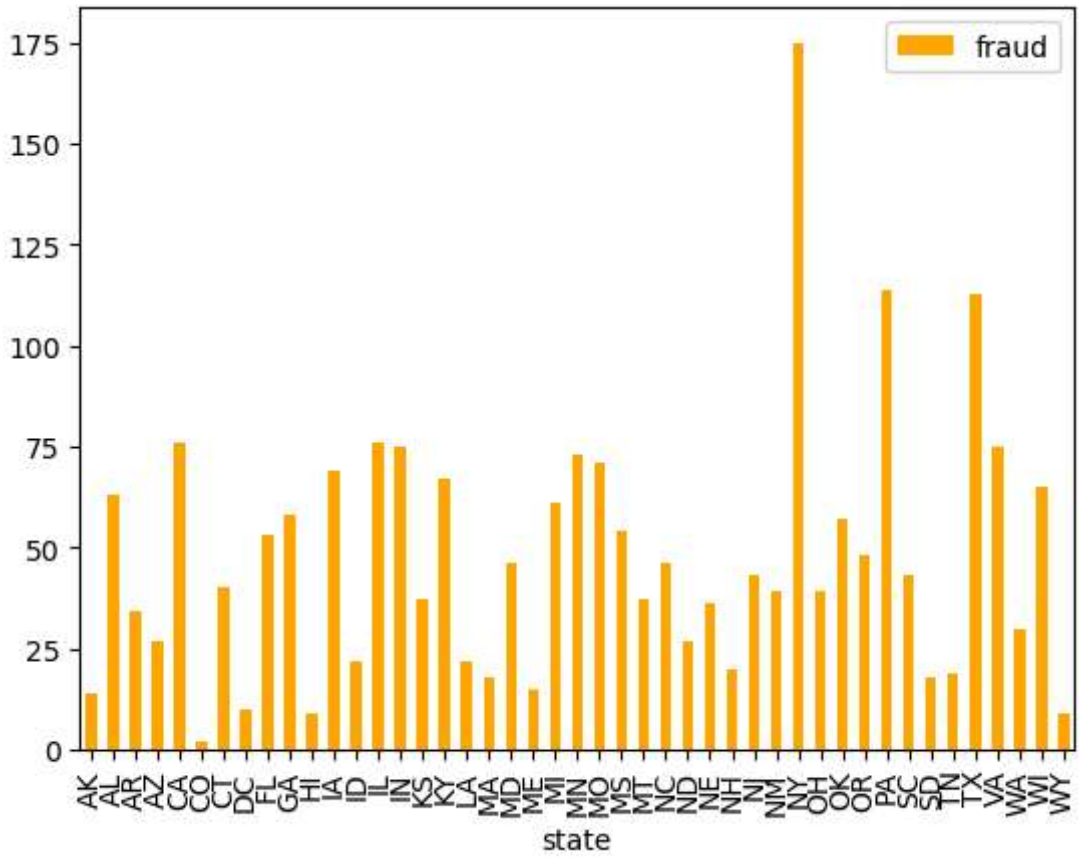
In [33]:
```
#Fraudulent transactions distribution with respect to age of card holders

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

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

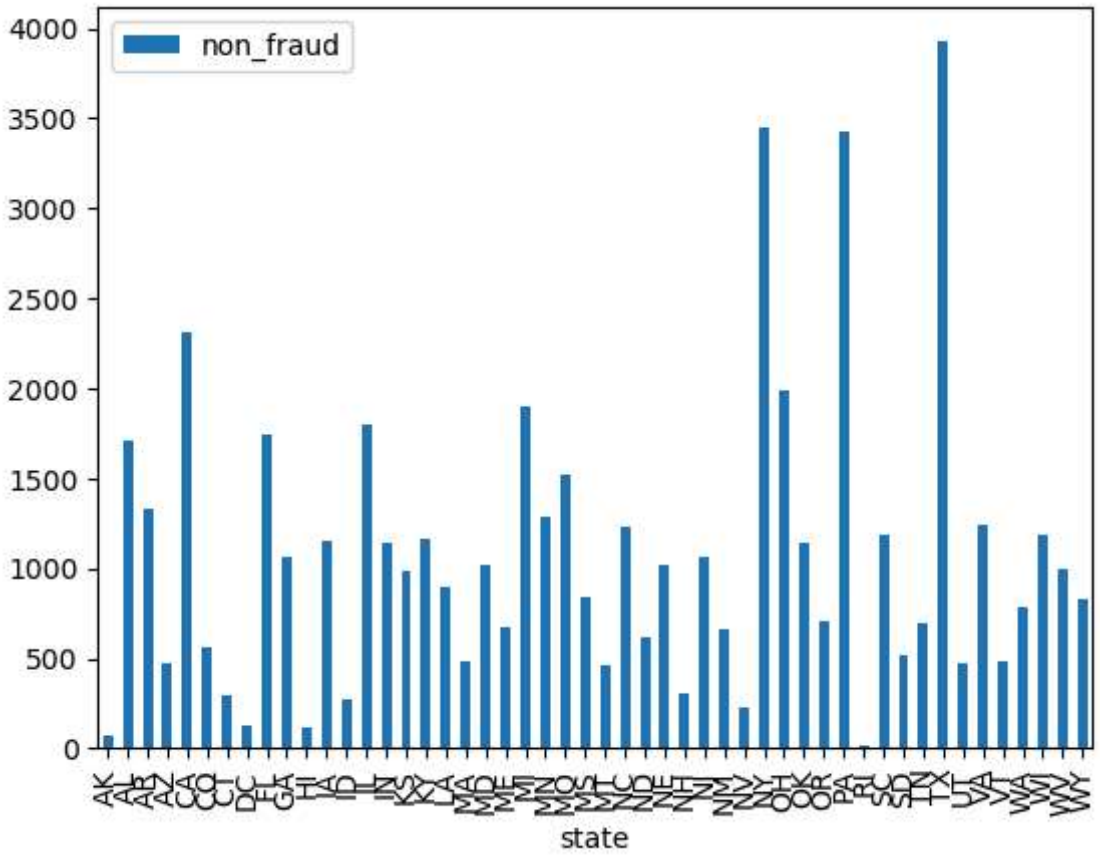
Out[33]:
```
<matplotlib.legend.Legend at 0x1b91f433520>
```



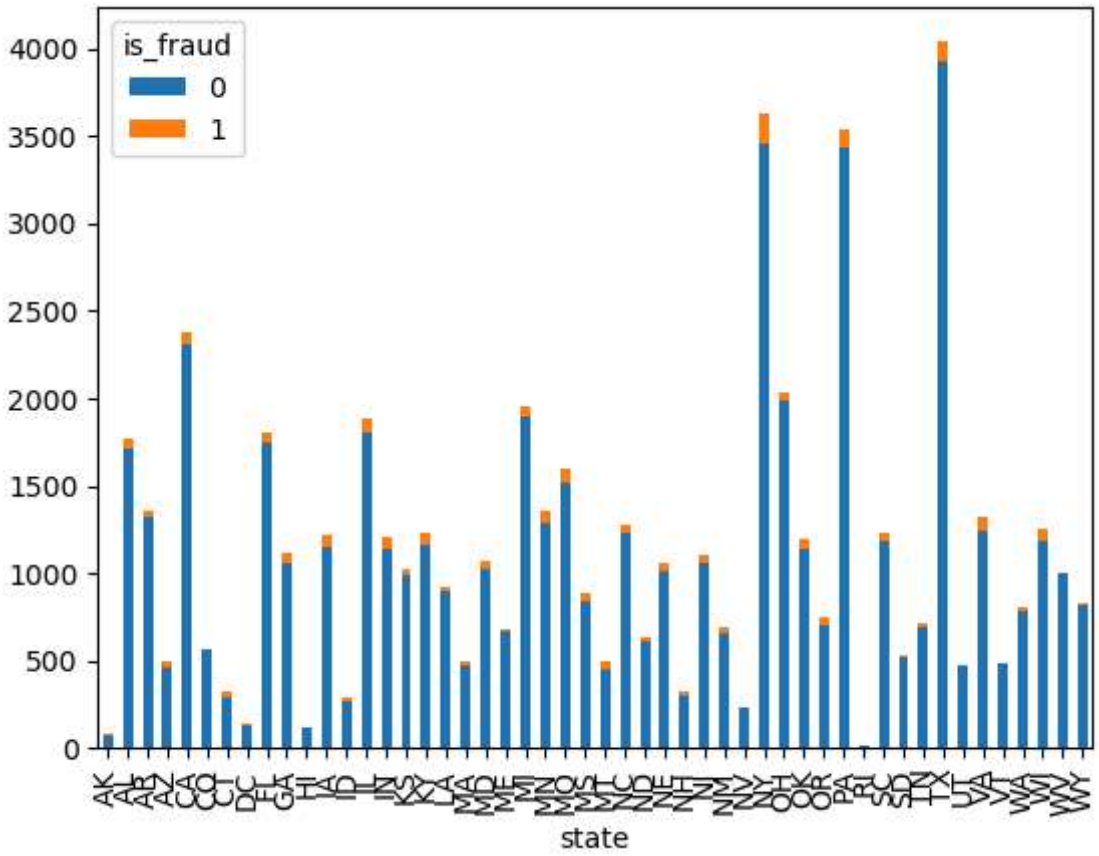
In [34]:
```
#Bar plots for fraud transactions and non-fraud transactions in each state
# Fraud transactions

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()
```



In [35]:
```
# 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()
```

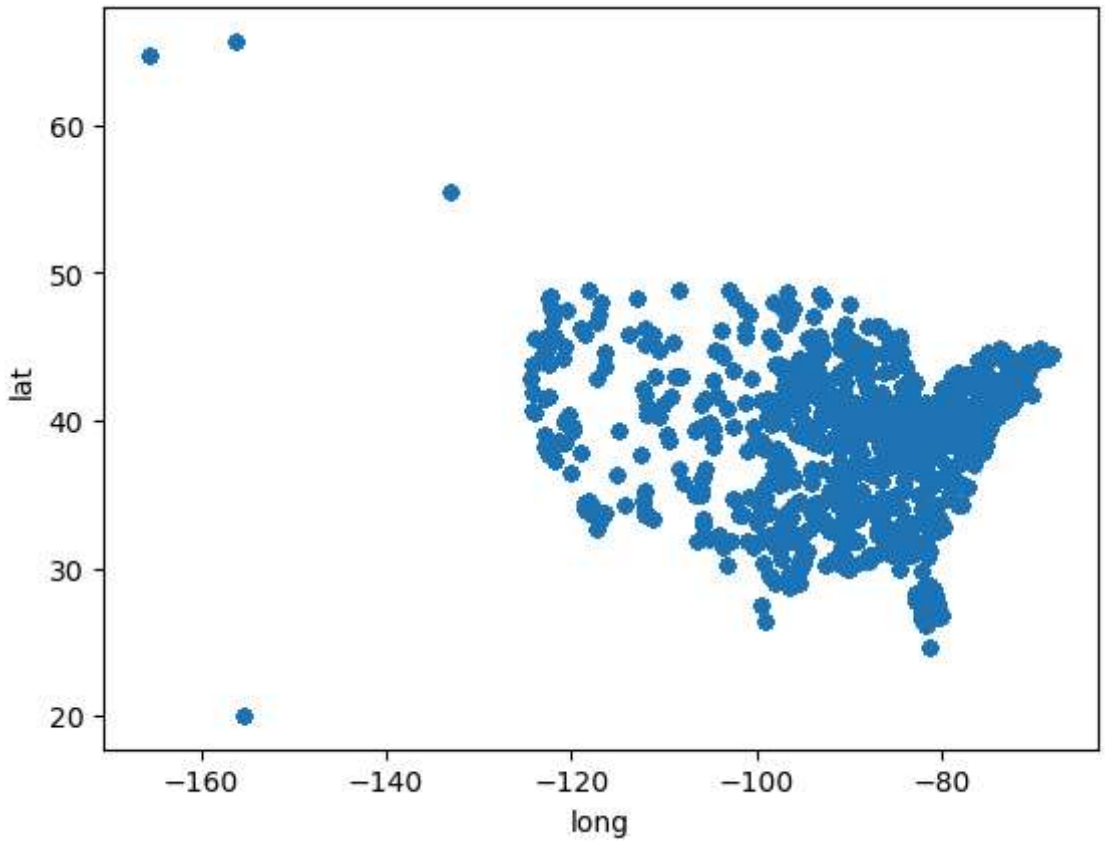


In [36]:
```
# Fraud and Non-Fraud transactions

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()
```



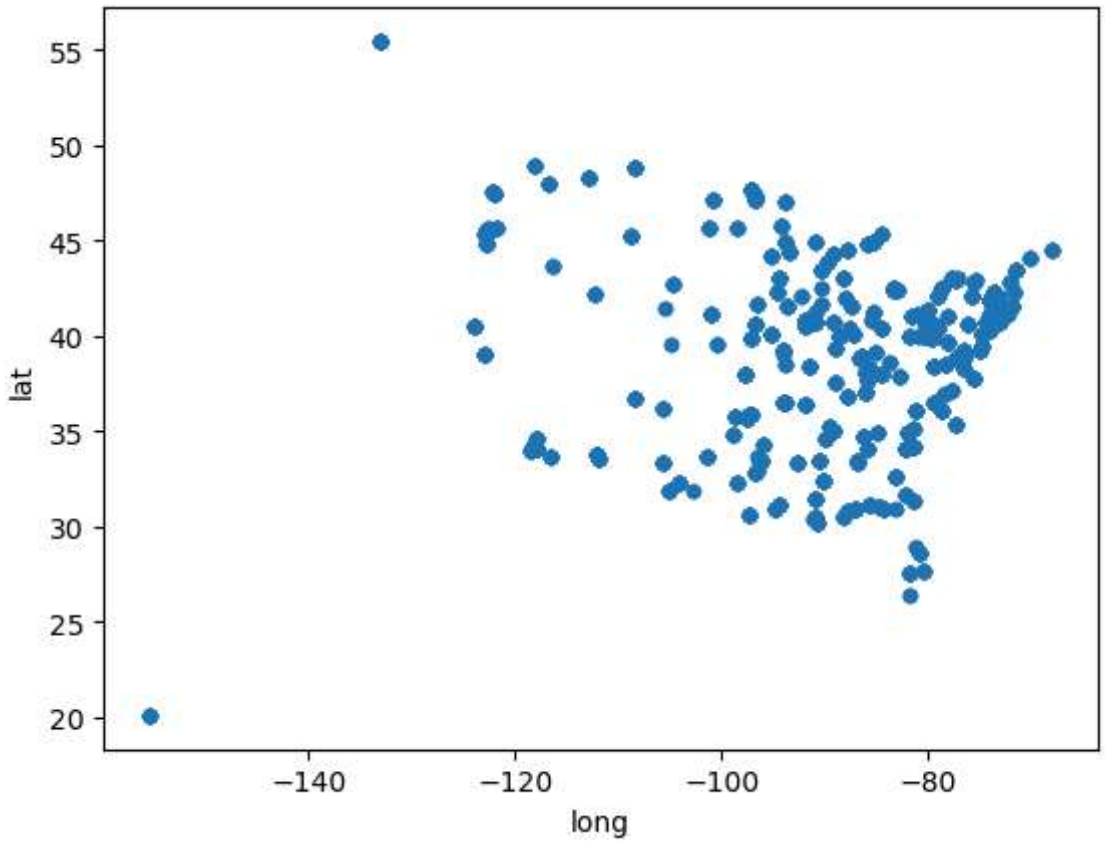
In [37]:
```
#Geographic distribution of data points

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

Out[37]:
```
<Axes: xlabel='long', ylabel='lat'>
```



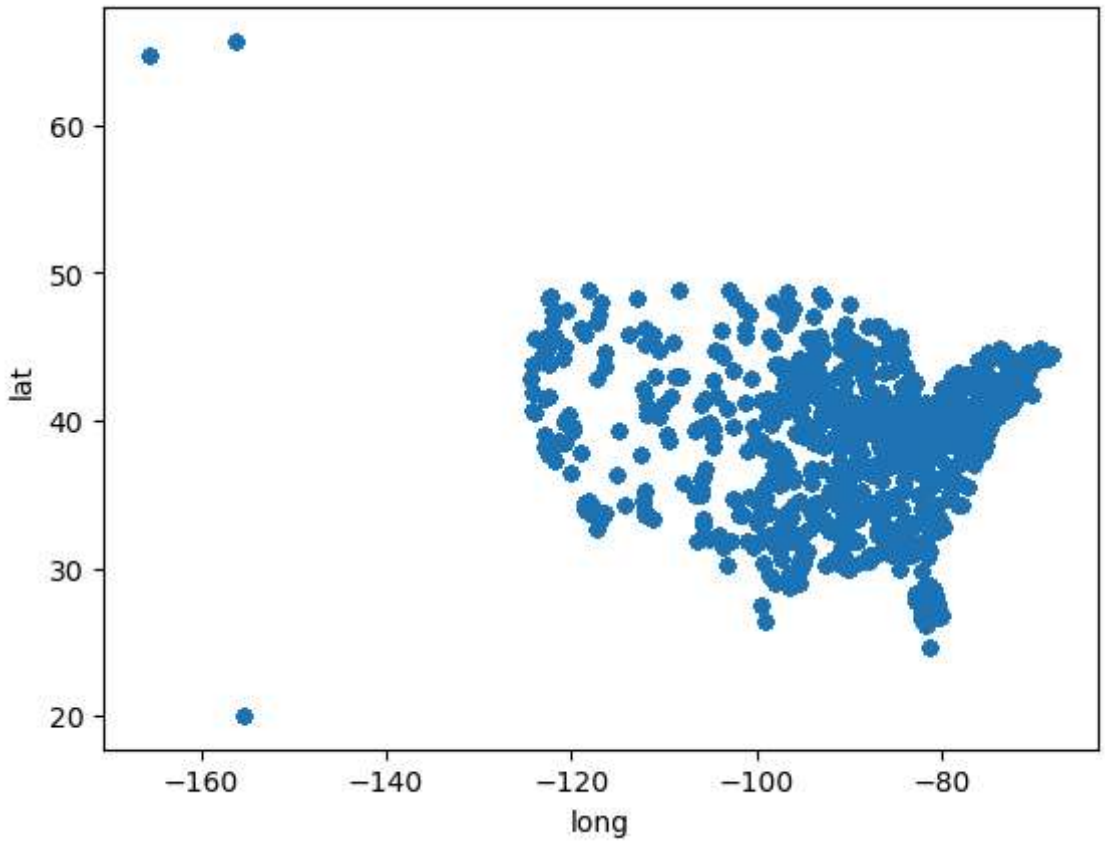
In [38]:
```
data_fraud = data[data['is_fraud'] == 1]
data_fraud.plot(kind="scatter", x = "long", y= "lat")
data_fraud.shape
```

Out[38]:
```
(2145, 22)
```



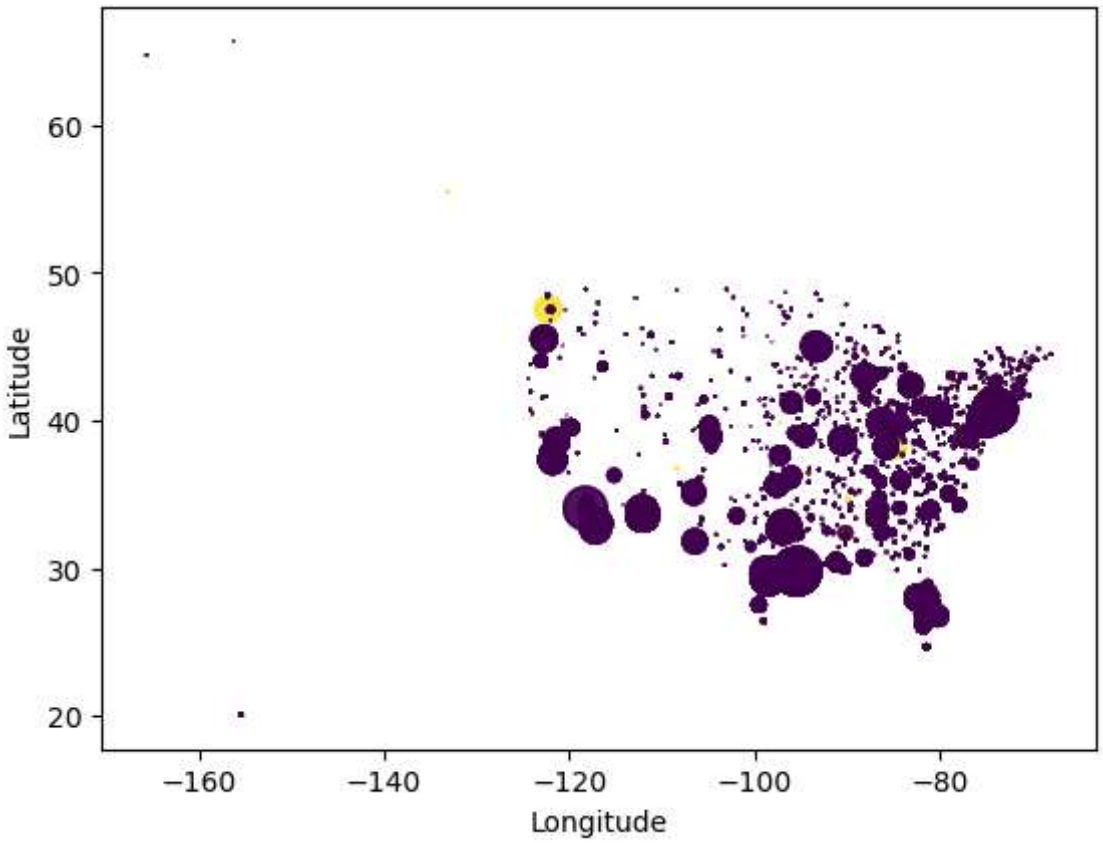
In [39]:
```
data_non_fraud = data[data['is_fraud'] == 0]
data_non_fraud.plot(kind="scatter", x = "long", y= "lat")
data_non_fraud.shape
```

Out[39]:
```
(53574, 22)
```



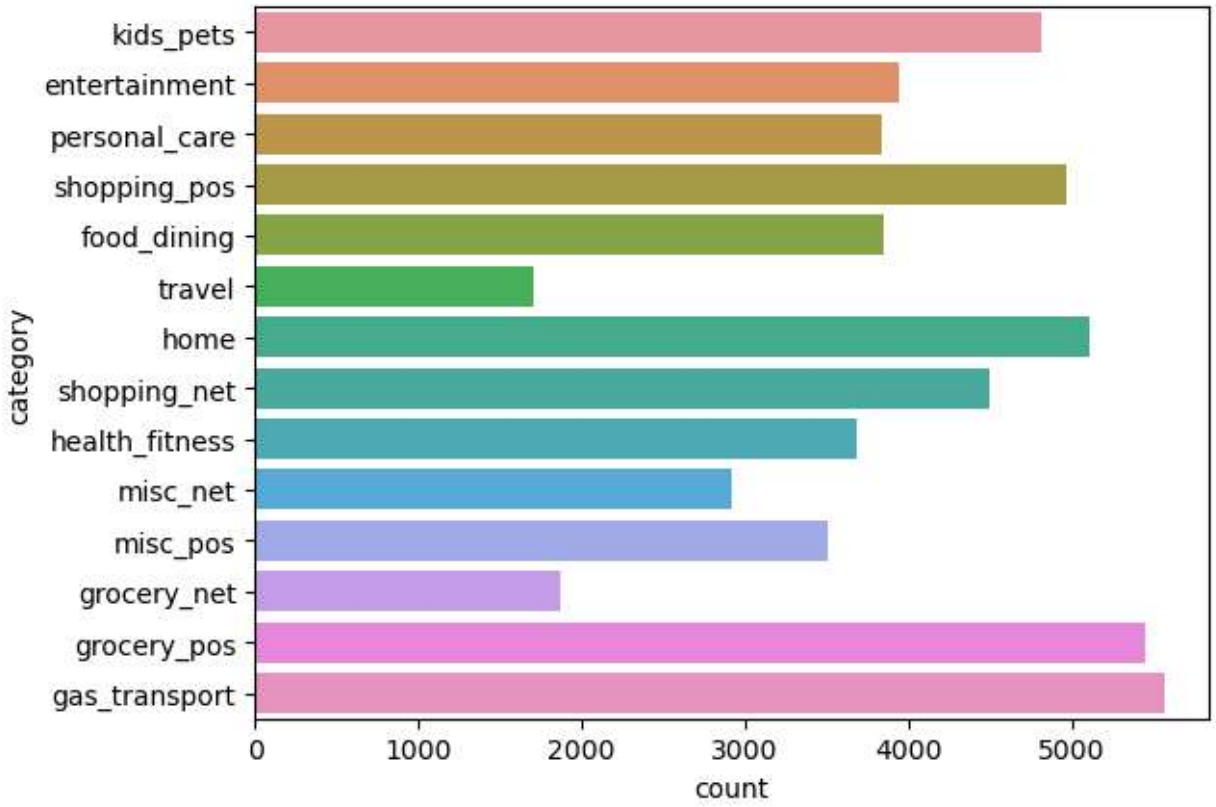
In [40]:
```
#population and fraud by location

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

Out[40]:
```
Text(0, 0.5, 'Latitude')
```



In [41]:
```
#category wise distribution of all data

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

Out[41]:
```
<Axes: xlabel='count', ylabel='category'>
```

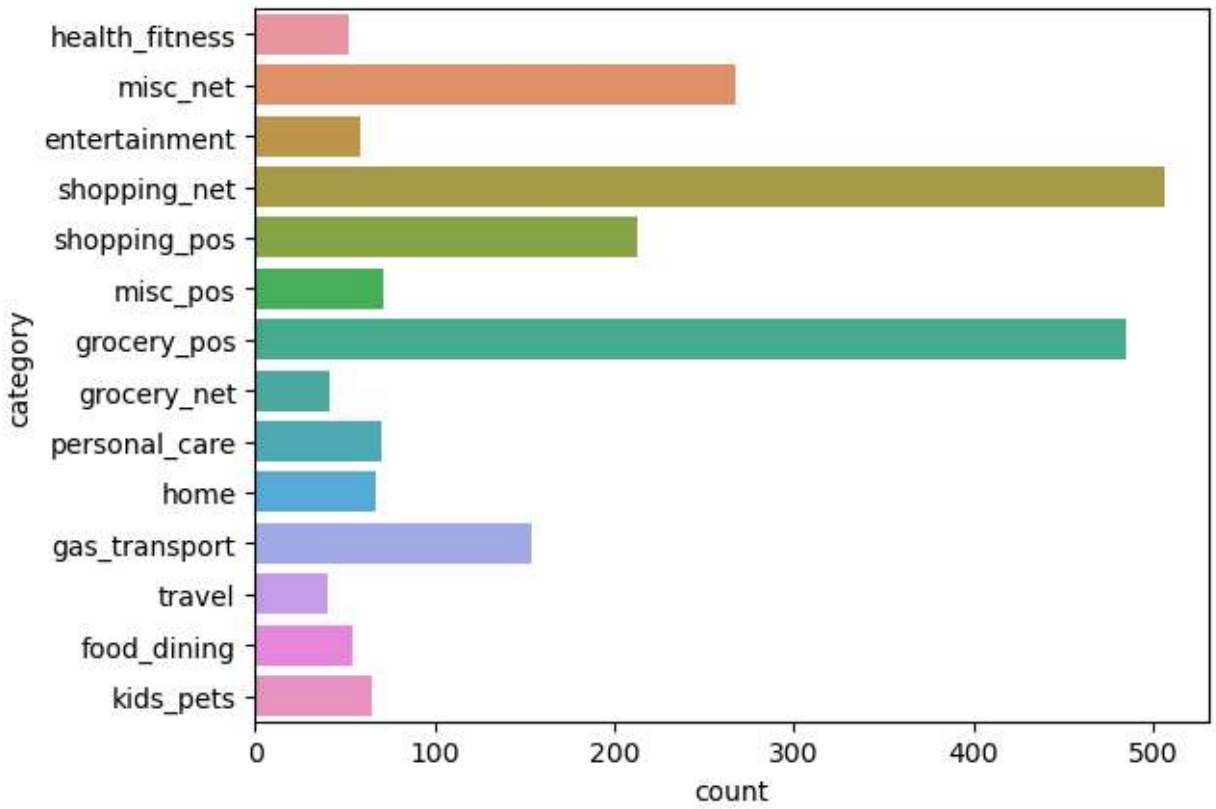


In [42]:
```
data['category'].value_counts()
```

Out[42]:
```
gas_transport     5566
grocery_pos       5451
home              5109
shopping_pos      4965
kids_pets         4819
shopping_net      4494
entertainment     3945
food_dining       3850
personal_care     3835
health_fitness    3689
misc_pos          3510
misc_net          2913
grocery_net       1869
travel            1704
Name: category, dtype: int64
```

In [43]:
```
#category wise distribution of fraud data


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

Out[43]:
```
<Axes: xlabel='count', ylabel='category'>
```

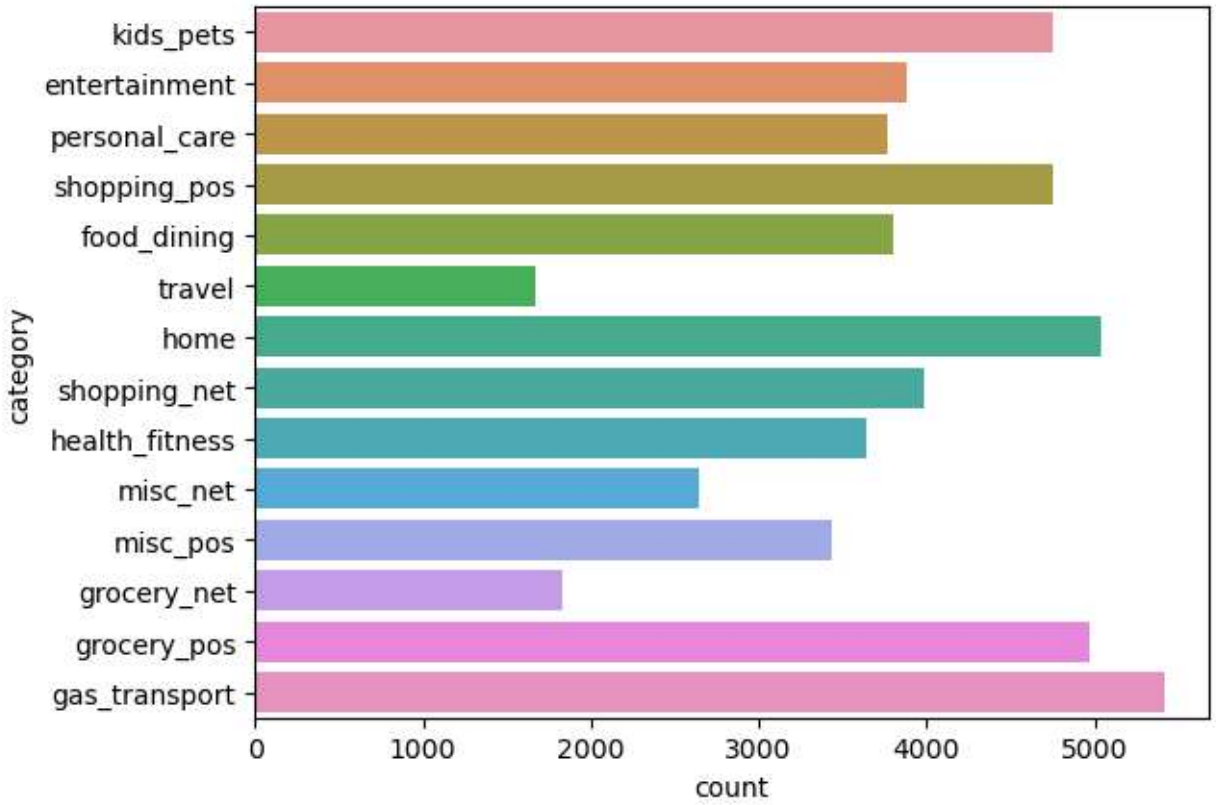


In [44]:
```
data_fraud['category'].value_counts()
```

Out[44]:
```
shopping_net      506
grocery_pos       485
misc_net          267
shopping_pos      213
gas_transport     154
misc_pos           72
personal_care      70
home               67
kids_pets          65
entertainment      59
food_dining        54
health_fitness     52
grocery_net        41
travel             40
Name: category, dtype: int64
```

In [45]:
```
#category wise distribution of non-fraud data

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

Out[45]:
```
<Axes: xlabel='count', ylabel='category'>
```



In [46]:
```
data_non_fraud['category'].value_counts()
```

Out[46]:
```
gas_transport     5412
home              5042
grocery_pos       4966
kids_pets         4754
shopping_pos      4752
shopping_net      3988
entertainment     3886
food_dining       3796
personal_care     3765
health_fitness    3637
misc_pos          3438
misc_net          2646
grocery_net       1828
travel            1664
Name: category, dtype: int64
```

In [47]:
```
data_copy=data.copy()
data_copy
```

Out[47]:

	cc_num	merchant	category	amt	gender	street	city	stat
0	30407675418785	fraud_Daugherty LLC	kids_pets	19.55	F	76752 David Lodge Apt. 064	Breesport	N
1	571465035400	fraud_Gottlieb Group	kids_pets	42.40	M	45654 Hess Rest	Fort Washakie	W
2	4570636521433188	fraud_Welch, Rath and Koepp	entertainment	24.73	F	5097 Jodi Vista Suite 811	Deltona	F
3	379897244598068	fraud_Kautzer and Sons	personal_care	176.23	M	5537 Jessica Plaza	Pewee Valley	K
4	4302475216404898	fraud_Zemlak, Tillman and Cremin	personal_care	19.03	M	384 Newman Forks Apt. 370	Belmond	I
...	...	...	...	...	...	...	...	
55714	30197398657930	fraud_Macejkovic-Lesch	shopping_pos	1.40	F	9727 Deleon Mountain	Creedmoor	N
55715	213112402583773	fraud_Baumbach, Hodkiewicz and Walsh	shopping_pos	25.49	F	4664 Sanchez Common Suite 930	Bradley	S
55716	3524574586339330	fraud_Heathcote, Yost and Kertzmann	shopping_net	29.56	F	94225 Smith Springs Apt. 617	Vero Beach	F
55717	3523843138706408	fraud_Prosacco, Kreiger and Kovacek	home	17.00	F	28812 Charles Mill Apt. 628	Plantersville	A
55718	4079773899158	fraud_Breitenberg LLC	travel	7.99	M	7020 Doyle Stream Apt. 951	Mesa	I

55719 rows × 22 columns

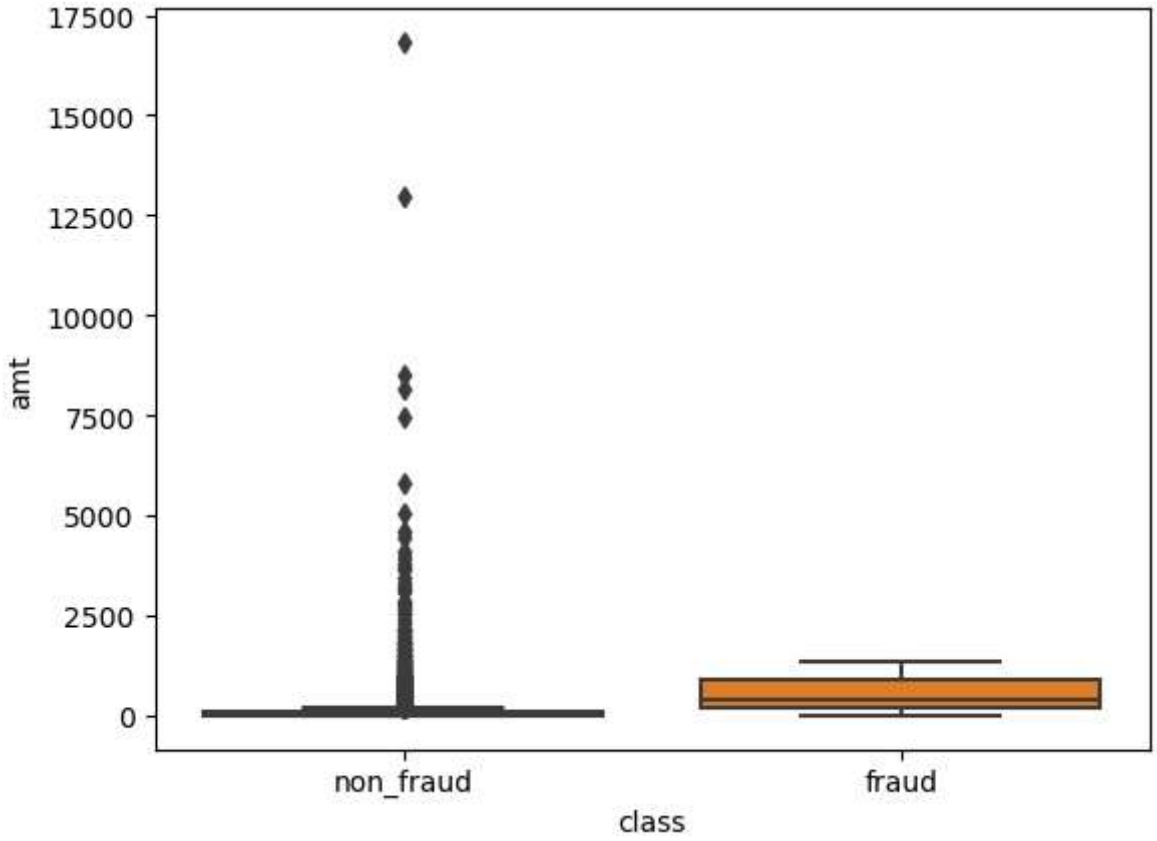
In [48]:
```
#statistics

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()
```

Out[48]:

class	fraud	non_fraud
amin	1.780000	1.000000
amax	1320.920000	16837.080000
mean	528.356494	67.603566
median	371.940000	47.110000

In [49]:
```
#Box plot for amount distribution
sns.boxplot(data = data_copy, x = 'class', y = 'amt')
```

Out[49]:
```
<Axes: xlabel='class', ylabel='amt'>
```

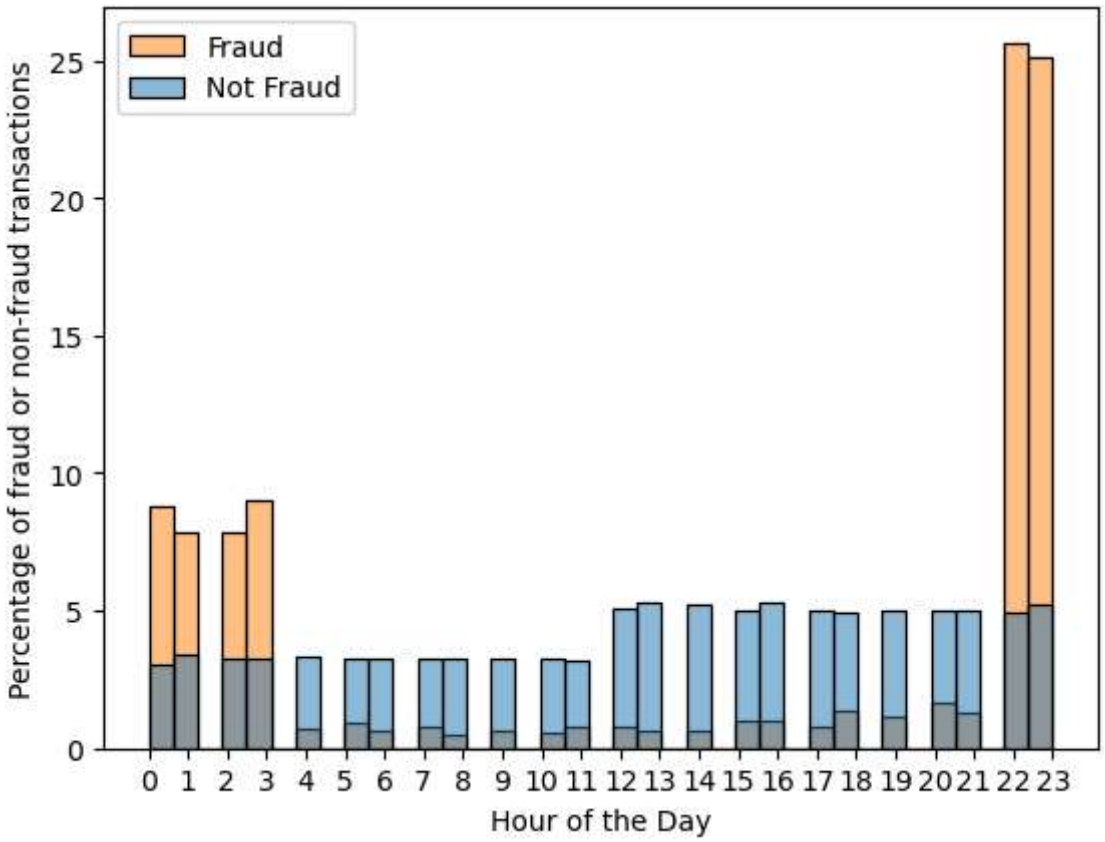


In [50]:
```
#Correlation between columns of the data

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

```
C:\Users\harshitha.palla\AppData\Local\Temp\ipykernel_19932\3117837662.py:4: FutureWa
rning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the value of
numeric_only to silence this warning.
  sns.heatmap(data.corr(),cmap='coolwarm', annot=True)
```

In [51]:
```
#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='pe
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'])
```

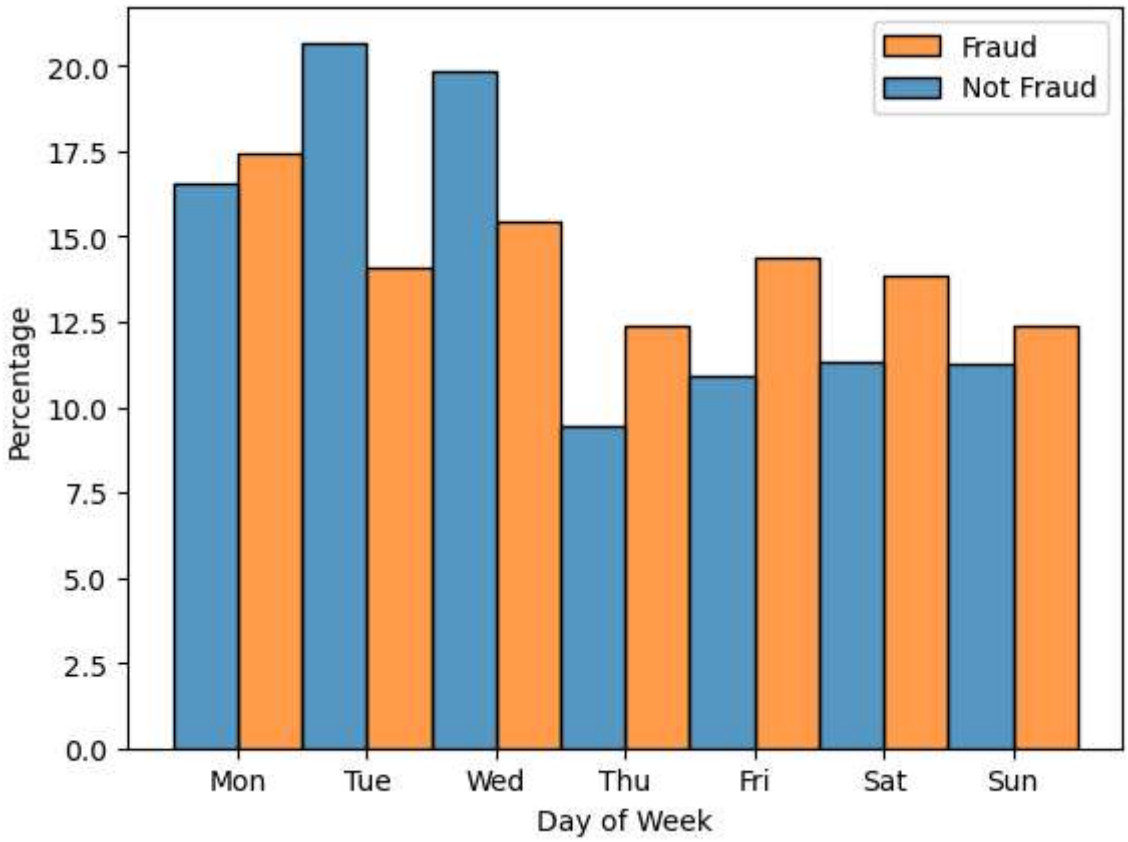
Out[51]:
```
<matplotlib.legend.Legend at 0x1b91e5ffb80>
```



In [52]:
```
#       Percentage of fraud and non-fraud transactions in the days of a week

day_plot=sns.histplot(data=data, x="day", hue="is_fraud", common_norm=False,stat='perc
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'])
```

```
C:\Users\harshitha.palla\AppData\Local\Temp\ipykernel_19932\3981283542.py:4: UserWarn
ing: FixedFormatter should only be used together with FixedLocator
  day_plot.set_xticklabels(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
```

Out[52]:
```
<matplotlib.legend.Legend at 0x1b91e3a2950>
```



One Hot Encoding

In [53]:
```
# one hot encoding for - category, gender, day, age

ohe_category = pd.get_dummies(data.category, prefix='category', drop_first=True)

ohe_gender = pd.get_dummies(data.gender, prefix='gender', drop_first=True)

ohe_day_of_week = pd.get_dummies(data.day, prefix='day', drop_first=True)
```

In [54]:
```
# concatenation of encoded data to our dataset

data_new = pd.concat([data, ohe_category, ohe_gender, ohe_day_of_week], axis=1)

data_new.head()
```

Out[54]:

	cc_num	merchant	category	amt	gender	street	city	state	zip
0	30407675418785	fraud_Daugherty LLC	kids_pets	19.55	F	76752 David Lodge Apt. 064	Breesport	NY	14816
1	571465035400	fraud_Gottlieb Group	kids_pets	42.40	M	45654 Hess Rest	Fort Washakie	WY	82514
2	4570636521433188	fraud_Welch, Rath and Koepp	entertainment	24.73	F	5097 Jodi Vista Suite 811	Deltona	FL	32725
3	379897244598068	fraud_Kautzer and Sons	personal_care	176.23	M	5537 Jessica Plaza	Pewee Valley	KY	40056
4	4302475216404898	fraud_Zemlak, Tillman and Cremin	personal_care	19.03	M	384 Newman Forks Apt. 370	Belmond	IA	50421

5 rows × 42 columns

In [55]:
```
data_new.columns
```

Out[55]:
```
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', 'hour', 'day',
       'year-month', 'age', 'category_food_dining', 'category_gas_transport',
       'category_grocery_net', 'category_grocery_pos',
       'category_health_fitness', 'category_home', 'category_kids_pets',
       'category_misc_net', 'category_misc_pos', 'category_personal_care',
       'category_shopping_net', 'category_shopping_pos', 'category_travel',
       'gender_M', 'day_Mon', 'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue',
       'day_Wed'],
      dtype='object')
```

In [56]:
```
# dropping unwanted columns

data_new.drop(['merchant','street','city','state','job', 'category','gender','day'],
              axis=1, inplace=True)
```

In [57]:
```
data_new.columns
```

Out[57]:
```
Index(['cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop', 'trans_num',
       'unix_time', 'merch_lat', 'merch_long', 'is_fraud', 'hour',
       'year-month', 'age', 'category_food_dining', 'category_gas_transport',
       'category_grocery_net', 'category_grocery_pos',
       'category_health_fitness', 'category_home', 'category_kids_pets',
       'category_misc_net', 'category_misc_pos', 'category_personal_care',
       'category_shopping_net', 'category_shopping_pos', 'category_travel',
       'gender_M', 'day_Mon', 'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue',
       'day_Wed'],
      dtype='object')
```

Logistic Regression

In [102…
```
#collecting numerical Data

data_lr = data_new.select_dtypes(include='number')
```

In [103…
```
data_lr.columns
```

Out[103]:
```
Index(['cc_num', 'amt', 'zip', 'lat', 'long', 'city_pop', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud', 'hour', 'age',
       'category_food_dining', 'category_gas_transport',
       'category_grocery_net', 'category_grocery_pos',
       'category_health_fitness', 'category_home', 'category_kids_pets',
       'category_misc_net', 'category_misc_pos', 'category_personal_care',
       'category_shopping_net', 'category_shopping_pos', 'category_travel',
       'gender_M', 'day_Mon', 'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue',
       'day_Wed'],
      dtype='object')
```

In [104…
```
#dropping less significant columns

data_lr.drop(['zip', 'lat', 'long', 'city_pop', 'unix_time', 'merch_lat','merch_long'],
```

In [105…
```
data_lr
```

Out[105]:

	cc_num	amt	is_fraud	hour	age	category_food_dining	category_gas_transport
0	30407675418785	19.55	0	12	29	0	0
1	571465035400	42.40	0	12	44	0	0
2	4570636521433188	24.73	0	12	32	0	0
3	379897244598068	176.23	0	12	90	0	0
4	4302475216404898	19.03	0	12	56	0	0
...	...	...	...	...	...	...	...
55714	30197398657930	1.40	0	23	33	0	0
55715	213112402583773	25.49	0	23	37	0	0
55716	3524574586339330	29.56	0	23	35	0	0
55717	3523843138706408	17.00	0	23	50	0	0
55718	4079773899158	7.99	0	23	55	0	0

55719 rows × 25 columns

In [106…
```
#feature selection

target = data_lr.is_fraud.values
features = data_lr.drop(['is_fraud'],axis=1).values
```

In [107…
```
#feature scaling

features = StandardScaler().fit_transform(features)
```

In [108…
```
#train, test, split

from sklearn.model_selection import train_test_split
features_train, features_test, target_train, target_test = train_test_split(features,
```

In [109…
```
#Prediction

lr = LogisticRegression(random_state=22)

model = lr.fit(features_train, target_train)

target_train_pred = model.predict(features_train)
target_test_pred = model.predict(features_test)
```

In [110…
```
print('target_train_pred: ',target_train_pred)
print('target_test_pred: ', target_test_pred)
```

```
target_train_pred:  [0 0 0 ... 0 0 0]
target_test_pred:  [0 0 0 ... 0 0 0]
```

Model evaluation

In [119…
```
#evaluating the model

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                            columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[119]:

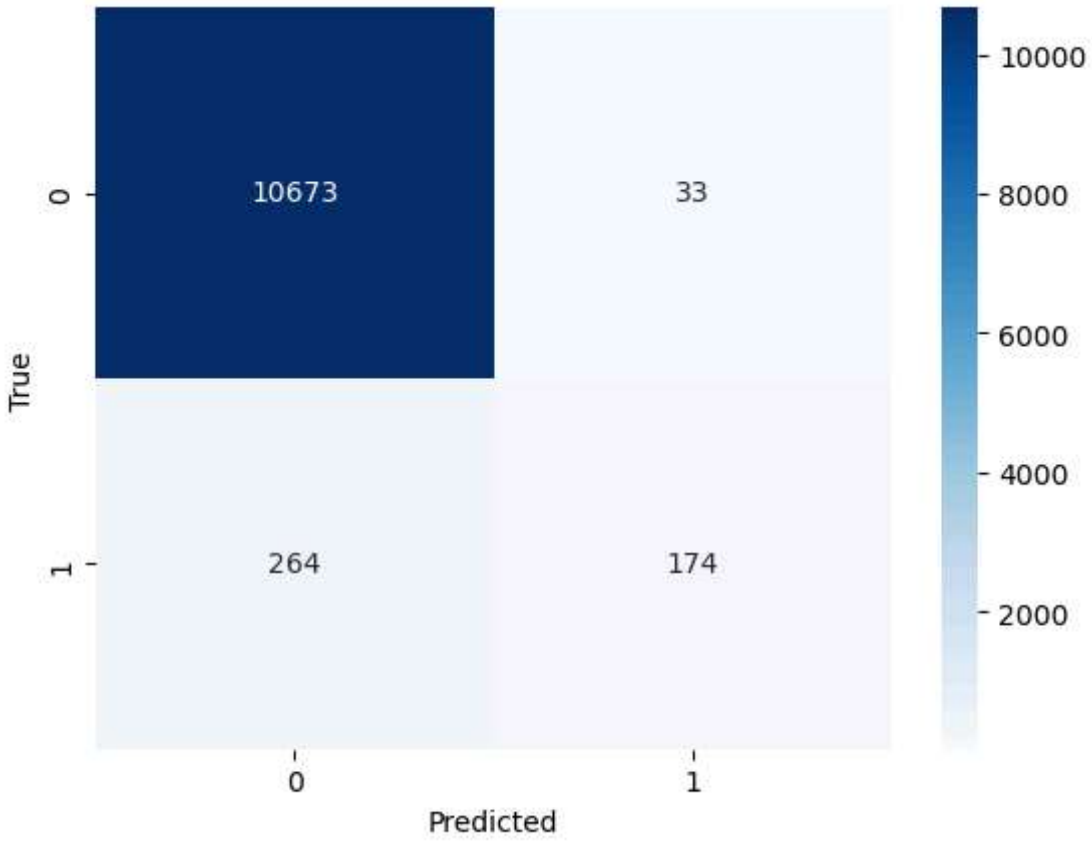
	Accuracy	Precision	Recall	F1 Score
0	0.990668	0.903382	0.853881	0.877934

In [112…
```
# evaluation training

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_train,target_train_pred)
precision = precision_score(target_train, target_train_pred)
recall = recall_score(target_train,target_train_pred)
f1 = f1_score(target_train, target_train_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                            columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[112]:

	Accuracy	Precision	Recall	F1 Score
0	0.973797	0.831488	0.396016	0.536508

In [113…
```
#Confusion matrix

c = confusion_matrix(target_test, target_test_pred)
sns.heatmap(c, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



Decision Tree

In [114…
```
#test train split

features_train, features_test, target_train, target_test = train_test_split(features,t
                                                                             test_size=
```

In [115…
```
#prediction

dt = DecisionTreeClassifier(max_depth=12)
model = dt.fit(features_train,target_train)
target_test_pred = model.predict(features_test)
print(target_test_pred)
```

```
[0 0 0 ... 0 0 0]
```

In [116…
```
target_train_pred = model.predict(features_train)
```

Model evaluation

In [117…
```
#evaluation

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)
```

```
df = pd.DataFrame([[accuracy, precision, recall, f1]],
                           columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[117]:

	Accuracy	Precision	Recall	F1 Score
0	0.990668	0.903382	0.853881	0.877934

In [118…

```
# evaluation training

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_train,target_train_pred)
precision = precision_score(target_train, target_train_pred)
recall = recall_score(target_train,target_train_pred)
f1 = f1_score(target_train, target_train_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                           columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[118]:

	Accuracy	Precision	Recall	F1 Score
0	0.997398	0.985357	0.946104	0.965332

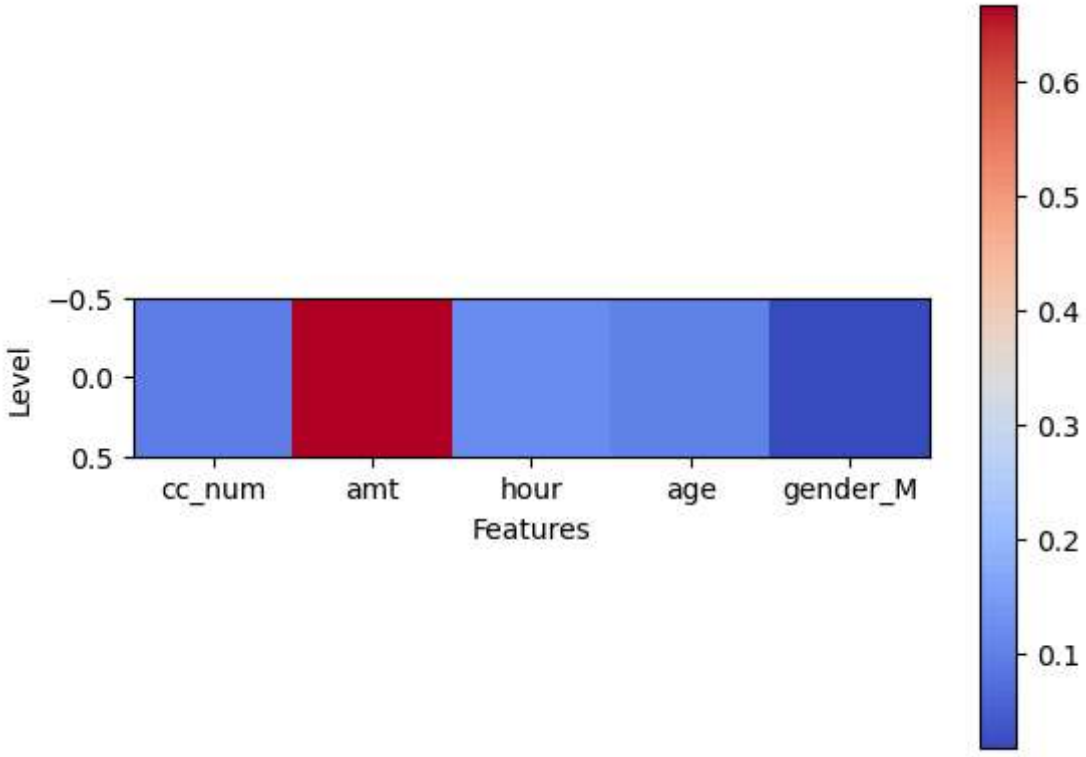
In [72]:

```
graph_features = data_lr.drop(['is_fraud','category_food_dining',
       'category_gas_transport', 'category_grocery_net',
       'category_grocery_pos', 'category_health_fitness', 'category_home',
       'category_kids_pets', 'category_misc_net', 'category_misc_pos',
       'category_personal_care', 'category_shopping_net',
       'category_shopping_pos', 'category_travel', 'day_Mon',
       'day_Sat', 'day_Sun', 'day_Thu', 'day_Tue', 'day_Wed'],axis=1)

fraud_or_not = data_lr.is_fraud

print(graph_features.columns)
```

```
Index(['cc_num', 'amt', 'hour', 'age', 'gender_M'], dtype='object')
```

In [75]:

```
#Decision Tree heatmap

dt2 = DecisionTreeClassifier()
dt2.fit(graph_features, fraud_or_not)

importance = dt2.feature_importances_

fig, ax = plt.subplots()
im = ax.imshow(importance.reshape(1, -1), cmap='coolwarm')
ax.set_xticks(np.arange(5))
ax.set_xticklabels(graph_features.columns)
cbar = ax.figure.colorbar(im, ax=ax)
ax.set_xlabel('Features')
ax.set_ylabel('Level')
plt.show()
```



Random Forest

In [76]:
```
# test train split

features_train, features_test, target_train, target_test = train_test_split(features,t
                                                                             test_size=
```

In [77]:
```
#model

rf = RandomForestClassifier(n_estimators=70,
                            random_state=34,
                            n_jobs=-1)
model = rf.fit(features_train,target_train)
```

In [78]:
```
#prediction

target_test_pred = model.predict(features_test)
print(target_test_pred)
```

```
[0 0 0 ... 0 0 0]
```

Model Evaluation

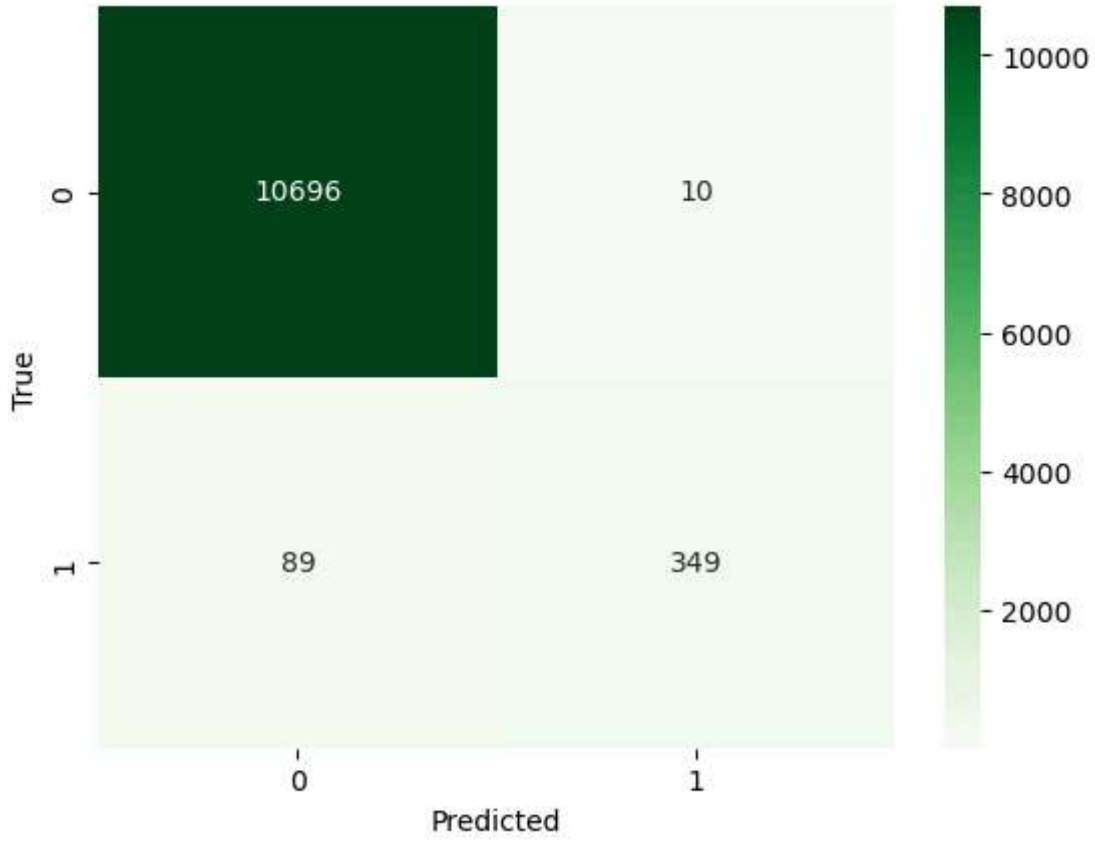
In [79]:
```
#evaluating the model

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)
```

```
df = pd.DataFrame([[accuracy, precision, recall, f1]],
                            columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[79]:

	Accuracy	Precision	Recall	F1 Score
0	0.991116	0.972145	0.796804	0.875784

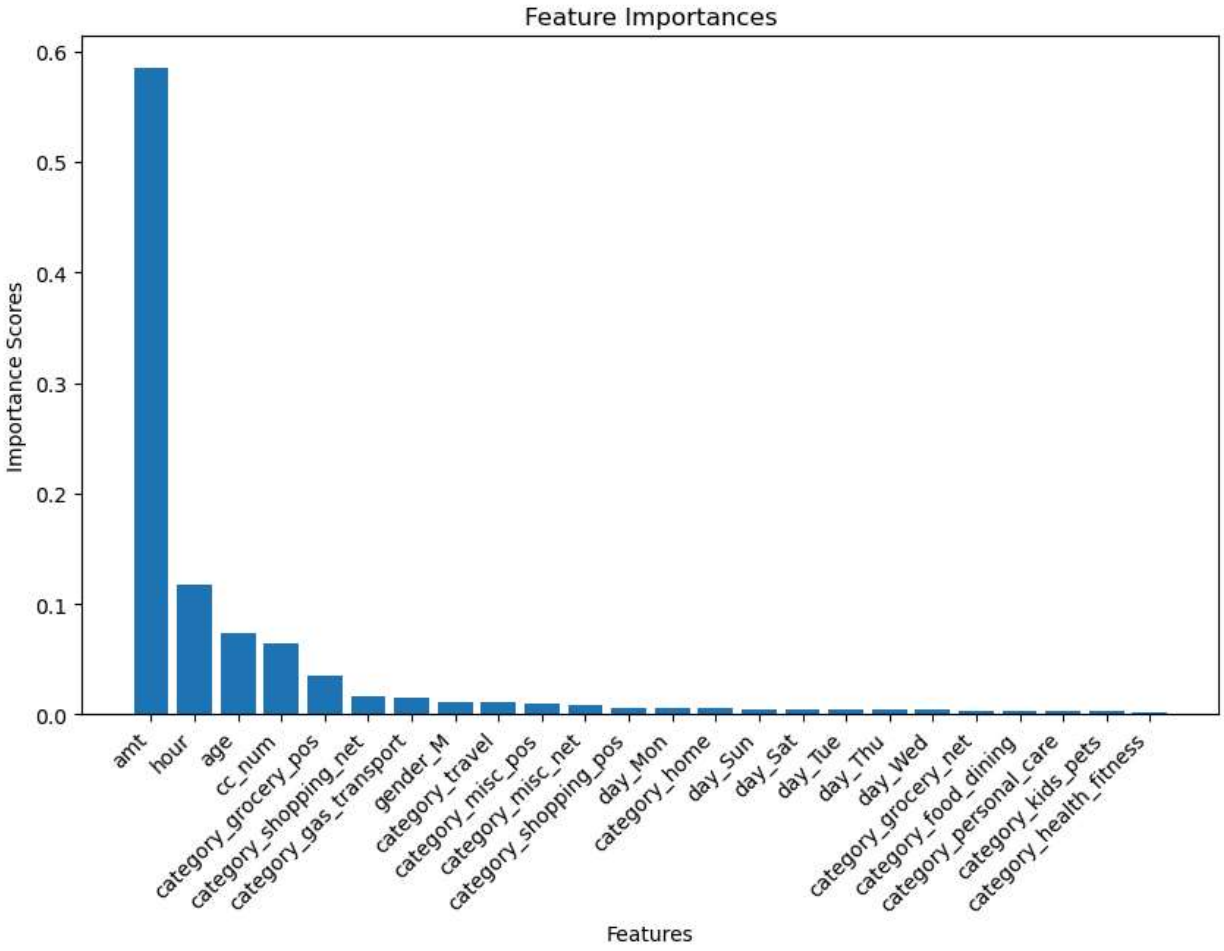
In [80]:
```
# Confusion Matrix

c = confusion_matrix(target_test, target_test_pred)
sns.heatmap(c, annot=True, cmap='Greens', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



In [81]:
```
#Feature Importance Plot

#Calculating importance of features

importance_scores = model.feature_importances_
feature_names = data_lr.drop(['is_fraud'],axis=1).columns
sorted_indices = importance_scores.argsort()[::-1]
sorted_importance_scores = importance_scores[sorted_indices]
sorted_feature_names = feature_names[sorted_indices]

#plot

fig, ax = plt.subplots(figsize=(10, 6))
```

```
ax.bar(sorted_feature_names, sorted_importance_scores)
ax.set_xticklabels(sorted_feature_names, rotation=45, ha='right')
ax.set_title("Feature Importances")
ax.set_xlabel("Features")
ax.set_ylabel("Importance Scores")
plt.show()
```

```
C:\Users\harshitha.palla\AppData\Local\Temp\ipykernel_19932\1690940902.py:15: UserWar
ning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(sorted_feature_names, rotation=45, ha='right')
```



Naive Bayes

In [82]:
```
# test train split

features_train, features_test, target_train, target_test = train_test_split(features,t
                                                                             test_size=
                                                                             )
```

In [83]:
```
#model

nb = GaussianNB()
nb.fit(features_train,target_train)
```

Out[83]:
```
▾ GaussianNB
GaussianNB()
```

In [84]:
```
#prediction

target_test_pred = nb.predict(features_test)
target_pred_prob = nb.predict_proba(features_test)
print('target_train_pred: ',target_train_pred)
print('target_test_pred: ', target_test_pred)
```

```
target_train_pred:  [0 0 0 ... 0 0 0]
target_test_pred:  [1 0 1 ... 1 0 0]
```

Model Evaluation

In [85]:
```
# evalluation metrics

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                          columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[85]:

	Accuracy	Precision	Recall	F1 Score
0	0.796752	0.122988	0.680365	0.208319

K Means

In [96]:
```
#train test split
from sklearn.preprocessing import normalize


features_train, features_test, target_train, target_test = train_test_split(features,t
```

In [97]:
```
#model

km = KMeans(n_clusters=2,random_state=0,algorithm="elkan",max_iter=10000)
km.fit(features_train)
```

```
C:\Users\harshitha.palla\AppData\Local\anaconda3\lib\site-packages\sklearn\cluster\_k
means.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[97]:

KMeans

```
KMeans(algorithm='elkan', max_iter=10000, n_clusters=2, random_state=0)
```

In [98]:

```
#prediction

traget_test_pred = km.predict(features_test)
```

Model Evaluation

In [99]:

```
# evaluation

#Accuracy, Preciision, Recall, F1 score,
accuracy = accuracy_score(target_test,target_test_pred)
precision = precision_score(target_test, target_test_pred)
recall = recall_score(target_test,target_test_pred)
f1 = f1_score(target_test, target_test_pred)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                          columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[99]:

	Accuracy	Precision	Recall	F1 Score
0	0.760589	0.040446	0.222222	0.068436

In [100…

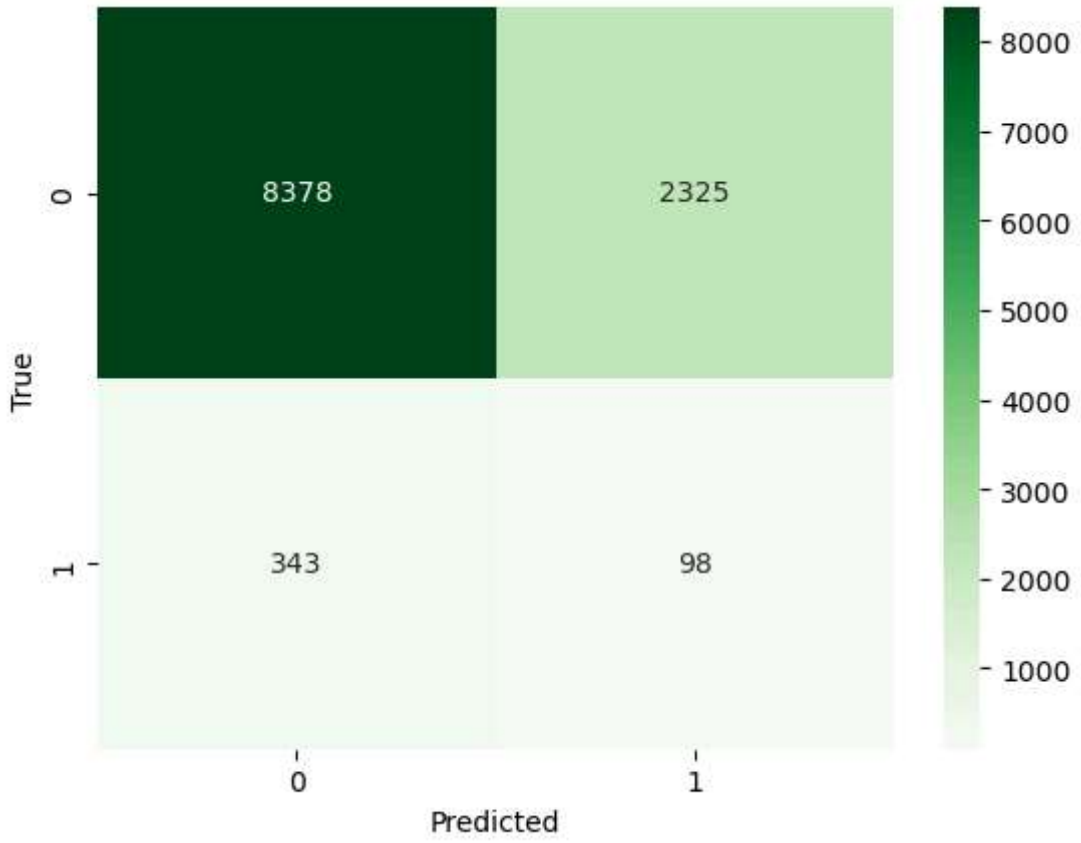
```
# evaluation

#Accuracy, Preciision, Recall, F1 score,
trn = km.predict(features_train)
accuracy = accuracy_score(target_train,trn)
precision = precision_score(target_train, trn)
recall = recall_score(target_train,trn)
f1 = f1_score(target_train, trn)


df = pd.DataFrame([[accuracy, precision, recall, f1]],
                          columns=['Accuracy', 'Precision', 'Recall', 'F1 Score'])
df
```

Out[100]:

	Accuracy	Precision	Recall	F1 Score
0	0.60507	0.040834	0.414906	0.074351

In [101…

```
c = confusion_matrix(target_test, target_test_pred)
sns.heatmap(c, annot=True, cmap='Greens', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```


8378
2325
343
98
0
1
True
0
1
Predicted
8000
7000
6000
5000
4000
3000
2000
1000

In [ ]: