Import Libraries

In [1]:
```
pip install yellowbrick
```

```
Requirement already satisfied: yellowbrick in c:\users\lenovo\anaconda3\lib\site-packages (1.5)
Requirement already satisfied: cycler>=0.10.0 in c:\users\lenovo\anaconda3\lib\site-packages (from
yellowbrick) (0.11.0)
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\lenovo\anaconda3\lib\site-packages
(from yellowbrick) (1.0.2)
Requirement already satisfied: numpy>=1.16.0 in c:\users\lenovo\anaconda3\lib\site-packages (from
yellowbrick) (1.21.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\lenovo\anaconda3\lib\site-pac
kages (from yellowbrick) (3.5.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\lenovo\anaconda3\lib\site-packages (from y
ellowbrick) (1.9.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\lenovo\anaconda3\lib\site-packages (fr
om matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lenovo\anaconda3\lib\site-packages (f
rom matplotlib!=3.0.0,>=2.0.2->yellowbrick) (4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lenovo\anaconda3\lib\site-packages
(from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\lenovo\anaconda3\lib\site-packages (f
rom matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:\users\lenovo\anaconda3\lib\site-packages (fro
m matplotlib!=3.0.0,>=2.0.2->yellowbrick) (21.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\lenovo\anaconda3\lib\site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (9.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\lenovo\anaconda3\lib\site-packages
(from scikit-learn>=1.0.0->yellowbrick) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\lenovo\anaconda3\lib\site-packages (from s
cikit-learn>=1.0.0->yellowbrick) (1.1.0)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\anaconda3\lib\site-packages (from pytho
n-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
```

In [2]:
```python
import numpy as np
import pandas as pd
from pandas.plotting import parallel_coordinates

import os
import sqlite3
import math
from collections import Counter
from pathlib import Path
from tqdm import tqdm

import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
import plotly
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.io as pio

from scipy.stats import skew
import yellowbrick
import sklearn
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

mpl.rcParams['font.family'] = 'monospace'
sns.set_theme(style="white", palette=None)
plotly.offline.init_notebook_mode()
plt.rcParams['figure.dpi'] = 300
plt.rcParams['savefig.dpi'] = 300
```

In [3]:
```
%matplotlib inline
```

In [4]:
```
# Reading csv files and drop the first column
df_train = pd.read_csv("fraudTrain.csv")
df_train.drop(df_train.columns[0], axis=1, inplace=True)

df_test = pd.read_csv("fraudTest.csv")
df_test.drop(df_test.columns[0], axis=1, inplace=True)

# First view 10 rows
df_train.head(10)
```

Out[4]:

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	stree
0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	56 Perr Cov
1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	4303 Riley Green Suit 39
2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	59 Whit Dal Suit 53
3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	944 Cynthi Cour Apt 03
4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	40 Bradle Res
5	2019-01-01 00:04:08	4767265376804500	fraud_Stroman, Hudson and Erdman	gas_transport	94.63	Jennifer	Conner	F	465 Davi Islan
6	2019-01-01 00:04:42	30074693890476	fraud_Rowe-Vandervort	grocery_net	44.54	Kelsey	Richards	F	88 Sara Statio Suit 62
7	2019-01-01 00:05:08	6011360759745864	fraud_Corwin-Collins	gas_transport	71.65	Steven	Williams	M	23 Flore Pas Suit 72
8	2019-01-01 00:05:18	4922710831011201	fraud_Herzog Ltd	misc_pos	4.27	Heather	Chase	F	688 Hick Strea Suit 95
9	2019-01-01 00:06:01	2720830304681674	fraud_Schoen, Kuphal and Nitzsche	grocery_pos	198.39	Melissa	Aguilar	F	2132 Taylo Square Suit 70

10 rows × 22 columns

In [5]:
```
df_train.columns
```

Out[5]:
```
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 [6]:
```
#Rename Columns

df_train.rename(columns={"trans_date_trans_time":"transaction_time", "cc_num":"credit_card_number",
```

In [7]:
```
#convert  date time column
df_train["transaction_time"] = pd.to_datetime(df_train["transaction_time"], infer_datetime_format=
df_train["dob"] = pd.to_datetime(df_train["dob"], infer_datetime_format=True)
```

In [8]:
```
from datetime import datetime

# Apply function utcfromtimestamp and drop column unix_time
df_train['time'] = df_train['unix_time'].apply(datetime.utcfromtimestamp)
df_train.drop('unix_time', axis=1)

# Add cloumn hour of day
df_train['hour_of_day'] = df_train.time.dt.hour
```

In [9]:
```
df_train[['time','hour_of_day']]
```

Out[9]:

	time	hour_of_day
0	2012-01-01 00:00:18	0
1	2012-01-01 00:00:44	0
2	2012-01-01 00:00:51	0
3	2012-01-01 00:01:16	0
4	2012-01-01 00:03:06	0
...	...	...
1296670	2013-06-21 12:12:08	12
1296671	2013-06-21 12:12:19	12
1296672	2013-06-21 12:12:32	12
1296673	2013-06-21 12:13:36	12
1296674	2013-06-21 12:13:37	12

1296675 rows × 2 columns

In [10]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 24 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   transaction_time    1296675 non-null  datetime64[ns]
 1   credit_card_number  1296675 non-null  int64
 2   merchant            1296675 non-null  object
 3   category            1296675 non-null  object
 4   amount(usd)         1296675 non-null  float64
 5   first               1296675 non-null  object
 6   last                1296675 non-null  object
 7   gender              1296675 non-null  object
 8   street              1296675 non-null  object
 9   city                1296675 non-null  object
 10  state               1296675 non-null  object
 11  zip                 1296675 non-null  int64
 12  lat                 1296675 non-null  float64
 13  long                1296675 non-null  float64
 14  city_pop            1296675 non-null  int64
 15  job                 1296675 non-null  object
 16  dob                 1296675 non-null  datetime64[ns]
 17  transaction_id      1296675 non-null  object
 18  unix_time           1296675 non-null  int64
 19  merch_lat           1296675 non-null  float64
 20  merch_long          1296675 non-null  float64
 21  is_fraud            1296675 non-null  int64
 22  time                1296675 non-null  datetime64[ns]
 23  hour_of_day         1296675 non-null  int64
dtypes: datetime64[ns](3), float64(5), int64(6), object(10)
memory usage: 237.4+ MB
```

In [11]:
```
#Change data type
# Credit card should be integer

df_train.credit_card_number = df_train.credit_card_number.astype('category')
df_train.is_fraud = df_train.is_fraud.astype('category')
df_train.hour_of_day = df_train.hour_of_day.astype('category')
```

In [12]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 24 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   transaction_time    1296675 non-null  datetime64[ns]
 1   credit_card_number  1296675 non-null  category
 2   merchant            1296675 non-null  object
 3   category            1296675 non-null  object
 4   amount(usd)         1296675 non-null  float64
 5   first               1296675 non-null  object
 6   last                1296675 non-null  object
 7   gender              1296675 non-null  object
 8   street              1296675 non-null  object
 9   city                1296675 non-null  object
 10  state               1296675 non-null  object
 11  zip                 1296675 non-null  int64
 12  lat                 1296675 non-null  float64
 13  long                1296675 non-null  float64
 14  city_pop            1296675 non-null  int64
 15  job                 1296675 non-null  object
 16  dob                 1296675 non-null  datetime64[ns]
 17  transaction_id      1296675 non-null  object
 18  unix_time           1296675 non-null  int64
 19  merch_lat           1296675 non-null  float64
 20  merch_long          1296675 non-null  float64
 21  is_fraud            1296675 non-null  category
 22  time                1296675 non-null  datetime64[ns]
 23  hour_of_day         1296675 non-null  category
dtypes: category(3), datetime64[ns](3), float64(5), int64(3), object(10)
memory usage: 212.7+ MB
```

EDA

In [13]:

```
np.round(df_train.describe(), 2)
```

Out[13]:

	amount(usd)	zip	lat	long	city_pop	unix_time	merch_lat	merch_long
count	1296675.00	1296675.00	1296675.00	1296675.00	1296675.00	1.296675e+06	1296675.00	1296675.00
mean	70.35	48800.67	38.54	-90.23	88824.44	1.349244e+09	38.54	-90.23
std	160.32	26893.22	5.08	13.76	301956.36	1.284128e+07	5.11	13.77
min	1.00	1257.00	20.03	-165.67	23.00	1.325376e+09	19.03	-166.67
25%	9.65	26237.00	34.62	-96.80	743.00	1.338751e+09	34.73	-96.90
50%	47.52	48174.00	39.35	-87.48	2456.00	1.349250e+09	39.37	-87.44
75%	83.14	72042.00	41.94	-80.16	20328.00	1.359385e+09	41.96	-80.24
max	28948.90	99783.00	66.69	-67.95	2906700.00	1.371817e+09	67.51	-66.95

In [14]:

```
groups = [pd.Grouper(key="transaction_time", freq="1W"), "is_fraud"]
df_ = df_train.groupby(by=groups).agg({"amount(usd)":'mean',"transaction_id":"count"}).reset_index(
```

In [15]:

```
df_
```

Out[15]:

	transaction_time	is_fraud	amount(usd)	transaction_id
0	2019-01-06	0	65.783042	9633
1	2019-01-06	1	548.677660	47
2	2019-01-13	0	68.687553	11896
3	2019-01-13	1	542.686240	125
4	2019-01-20	0	67.190627	11777
...	...	...	...	...
149	2020-06-07	1	670.901111	81
150	2020-06-14	0	69.000045	19835
151	2020-06-14	1	571.120225	89
152	2020-06-21	0	69.842152	17778
153	2020-06-21	1	457.665366	164

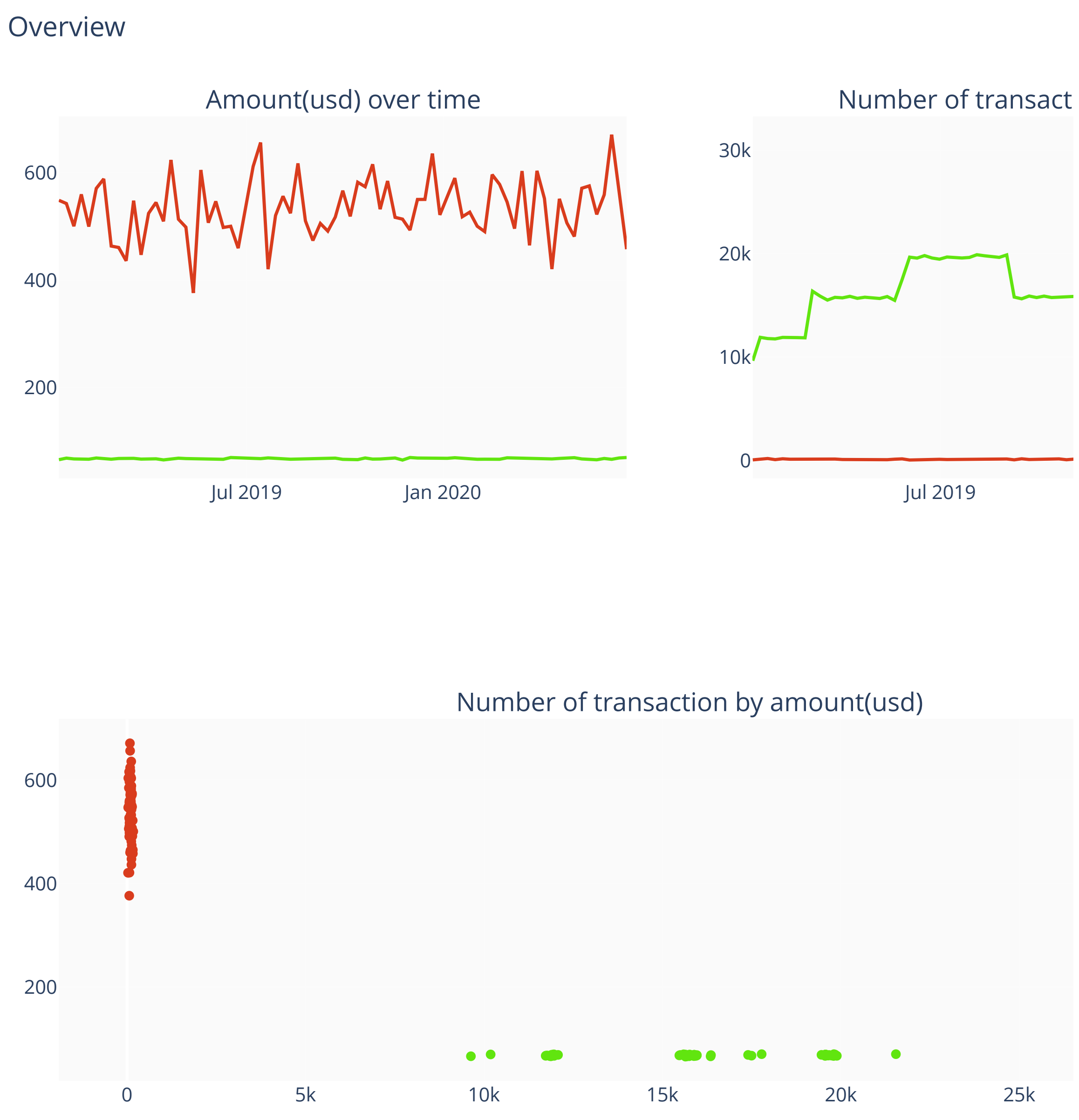
154 rows × 4 columns

In [16]:
```python
df_train.shape
```
Out[16]:
```
(1296675, 24)
```

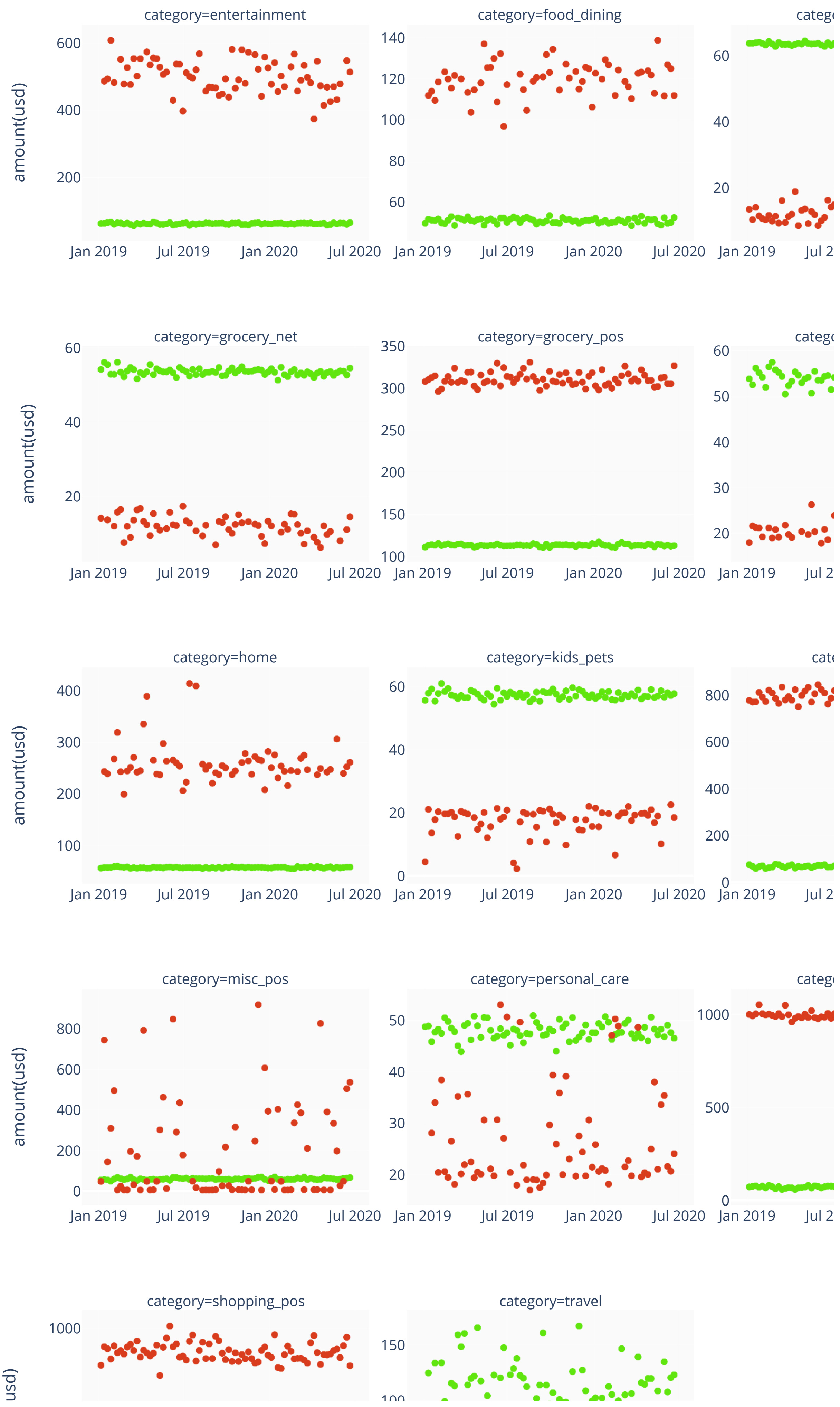
In [17]:
```python
def add_traces(df, x, y, hue, mode, cmap, showlegend=None):
    name_map = {1:"Yes", 0:"No"}
    traces = []
    for flag in df[hue].unique():
        traces.append(
            go.Scatter(
                x = df[df[hue]==flag][x],
                y = df[df[hue]==flag][y],
                mode=mode,
                marker=dict(color=cmap[flag]),
                showlegend=showlegend,
                name=name_map[flag]
            )
        )
    return traces
```

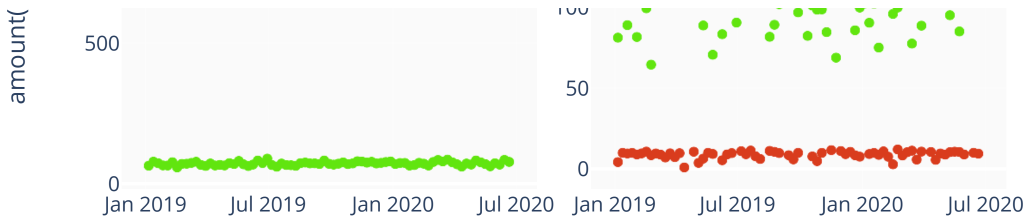
In [18]:
```python
fig = make_subplots(rows=2, cols=2,
                    specs=[
                        [{},{}],
                        [{"colspan":2},None]
                    ],
                   subplot_titles=("Amount(usd) over time", "Number of transactions overtime",
                                    "Number of transaction by amount(usd)")
                   )


ntraces = add_traces(df=df_,x='transaction_time',y='amount(usd)',hue='is_fraud',mode='lines',
                    showlegend=True, cmap=['#61E50F','#D93C1D'])

for trace in ntraces:
    fig.add_trace(
        trace,
        row=1,col=1
    )

ntraces = add_traces(df=df_,x='transaction_time',y='transaction_id',hue='is_fraud',mode='lines',
                    showlegend=False, cmap=['#61E50F','#D93C1D'])
for trace in ntraces:
    fig.add_trace(
        trace,
        row=1,col=2
    )

ntraces = add_traces(df=df_,x='transaction_id',y='amount(usd)',hue='is_fraud',mode='markers',
                    showlegend=True, cmap=['#61E50F','#D93C1D'])
```

```
for trace in ntraces:
    fig.add_trace(
        trace,
        row=2,col=1
    )

fig.update_layout(height=780,
                  width=960,
                  legend=dict(title='Is fraud?'),
                  plot_bgcolor='#fafafa',
                  title='Overview'
                 )

fig.show()
```

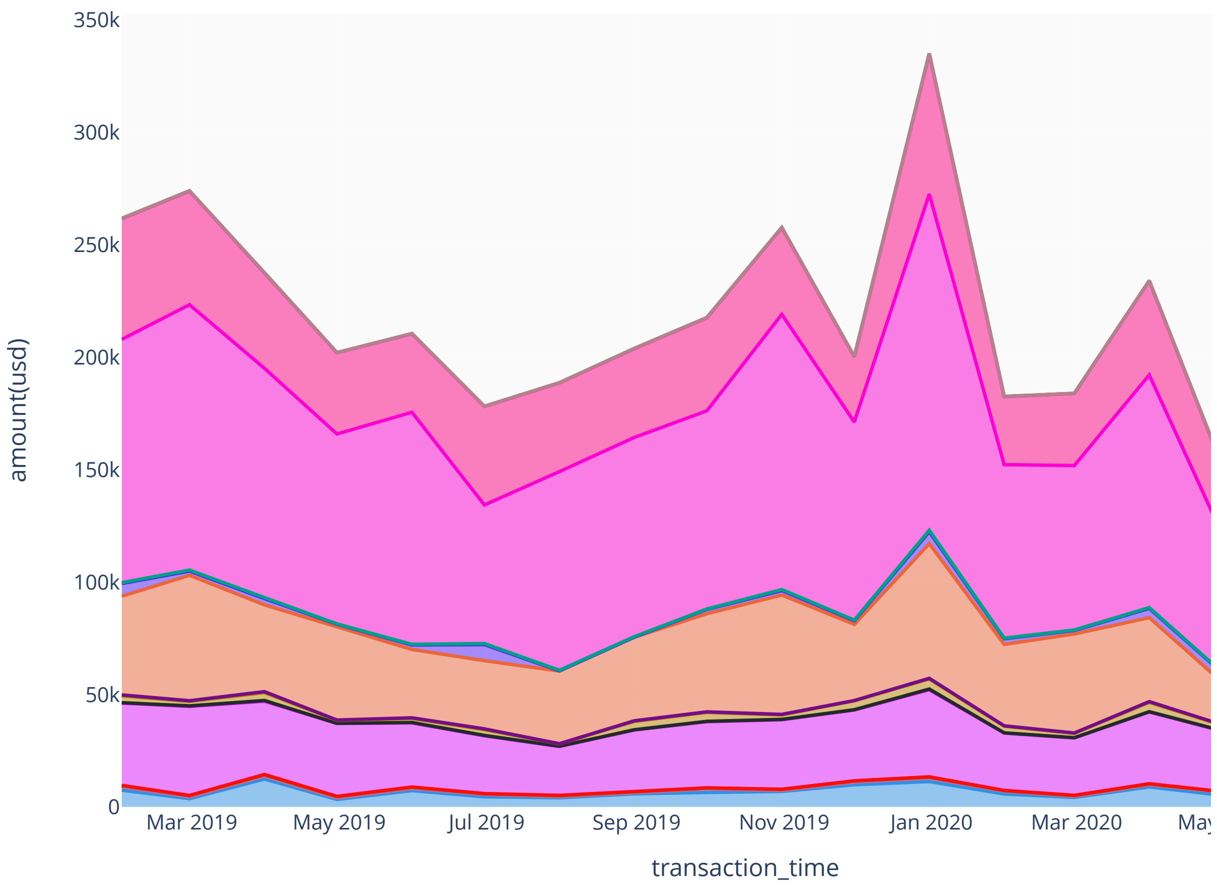


```
In [19]: df_ = df_train.groupby(by=[pd.Grouper(key="transaction_time", freq="1W"),
                                    'is_fraud','category']).agg({"amount(usd)":'mean',"transaction_id":"cour

         fig = px.scatter(df_,
                 x='transaction_time',
                 y='amount(usd)',
                 color='is_fraud',
                 facet_col ='category',
                 facet_col_wrap=3,
                 facet_col_spacing=.04,
                 color_discrete_map={0:'#61E50F', 1:'#D93C1D'}
         )

         fig.update_layout(height=1400,
                           width=960,
                           legend=dict(title='Is fraud?'),
                           plot_bgcolor='#fafafa'
                          )

         fig.update_yaxes(matches=None)
         fig.for_each_yaxis(lambda yaxis: yaxis.update(showticklabels=True))
         fig.for_each_xaxis(lambda xaxis: xaxis.update(showticklabels=True, title=''))

         fig.show();
```


category=entertainment
category=food_dining
catego
category=grocery_net
category=grocery_pos
catego
category=home
category=kids_pets
cate
category=misc_pos
category=personal_care
catego
category=shopping_pos
category=travel
amount(usd)
Jan 2019
Jul 2019
Jan 2020
Jul 2020



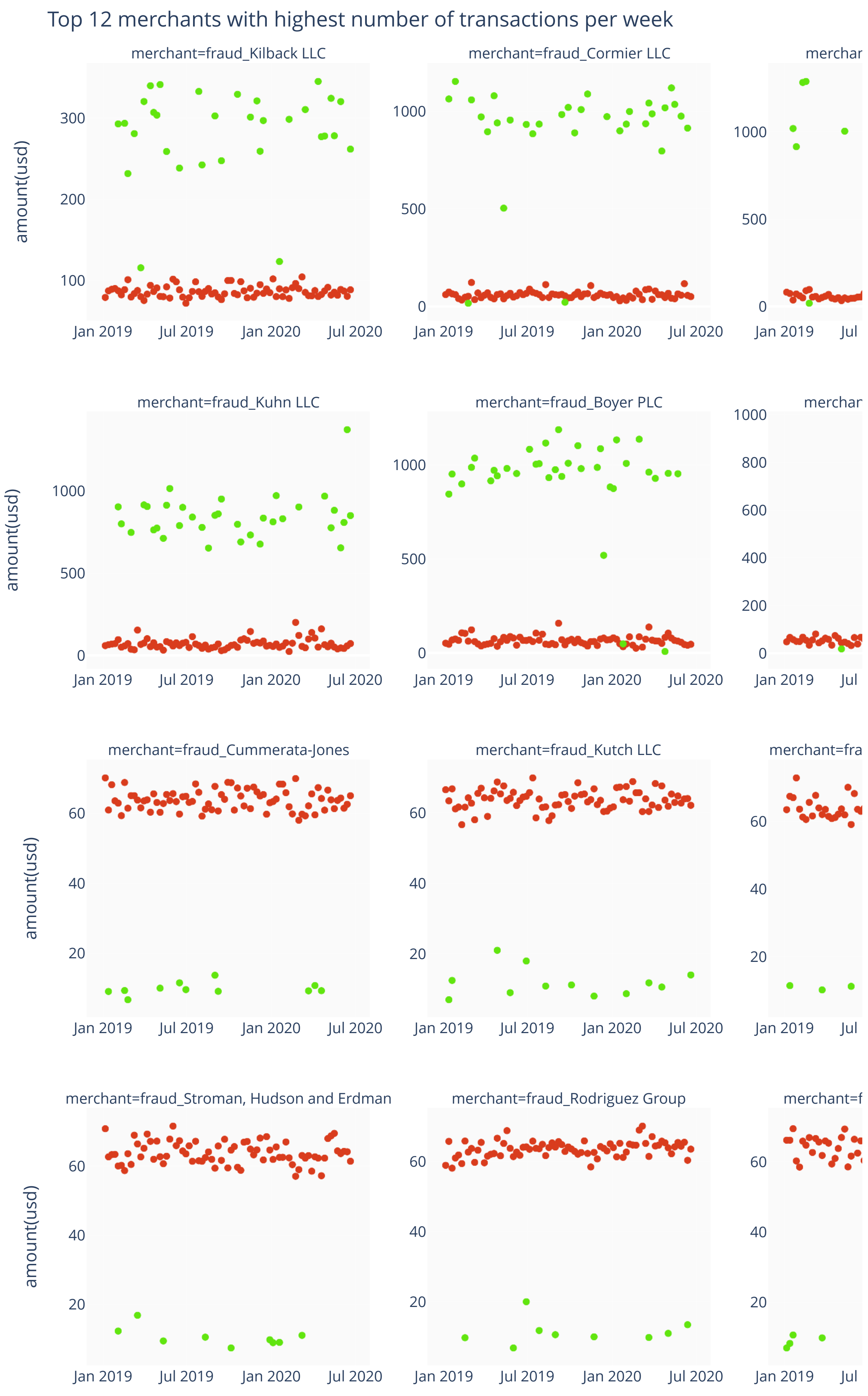
```
In [20]: df_ = df_train.groupby(by=[pd.Grouper(key="transaction_time", freq="1M"),
                                   'is_fraud','category']).agg({"amount(usd)":'sum',"transaction_id":"count

         fig = px.area(
             df_[df_.is_fraud==1],
             x='transaction_time',
             y='amount(usd)',
             color='category',
             color_discrete_sequence=px.colors.qualitative.Dark24
         )

         fig.update_layout(height=600,
                           width=960,
                           legend=dict(title='Categories'),
                           plot_bgcolor='#fafafa'
                          )

         fig.show()
```



```
In [21]: # Specified list of 12 merchants with the highest number of transactions.
         top12_merchants = df_train.merchant.value_counts()[:12]
```

```
df_ = df_train.groupby(by=[pd.Grouper(key="transaction_time", freq="1W"),'is_fraud',
                           'merchant']).agg({"amount(usd)":'mean',"transaction_id":"count"}).reset_

df_ = df_[df_.merchant.isin(top12_merchants.index)]
```

In [22]:
```
fig = px.scatter(df_,
        x='transaction_time',
        y='amount(usd)',
        color='is_fraud',
        facet_col ='merchant',
        facet_col_wrap=3,
        facet_col_spacing=.06,
        category_orders={'merchant': top12_merchants.index}, # order the subplots
        color_discrete_map={1:'#61E50F', 0:'#D93C1D'}
)

fig.update_layout(height=1200,
                  width=960,
                  title='Top 12 merchants with highest number of transactions per week',
                  legend=dict(title='Is fraud?'),
                  plot_bgcolor='#fafafa'
                 )

fig.update_yaxes(matches=None)
fig.for_each_yaxis(lambda yaxis: yaxis.update(showticklabels=True))
fig.for_each_xaxis(lambda xaxis: xaxis.update(showticklabels=True, title=''))

fig.show()
```

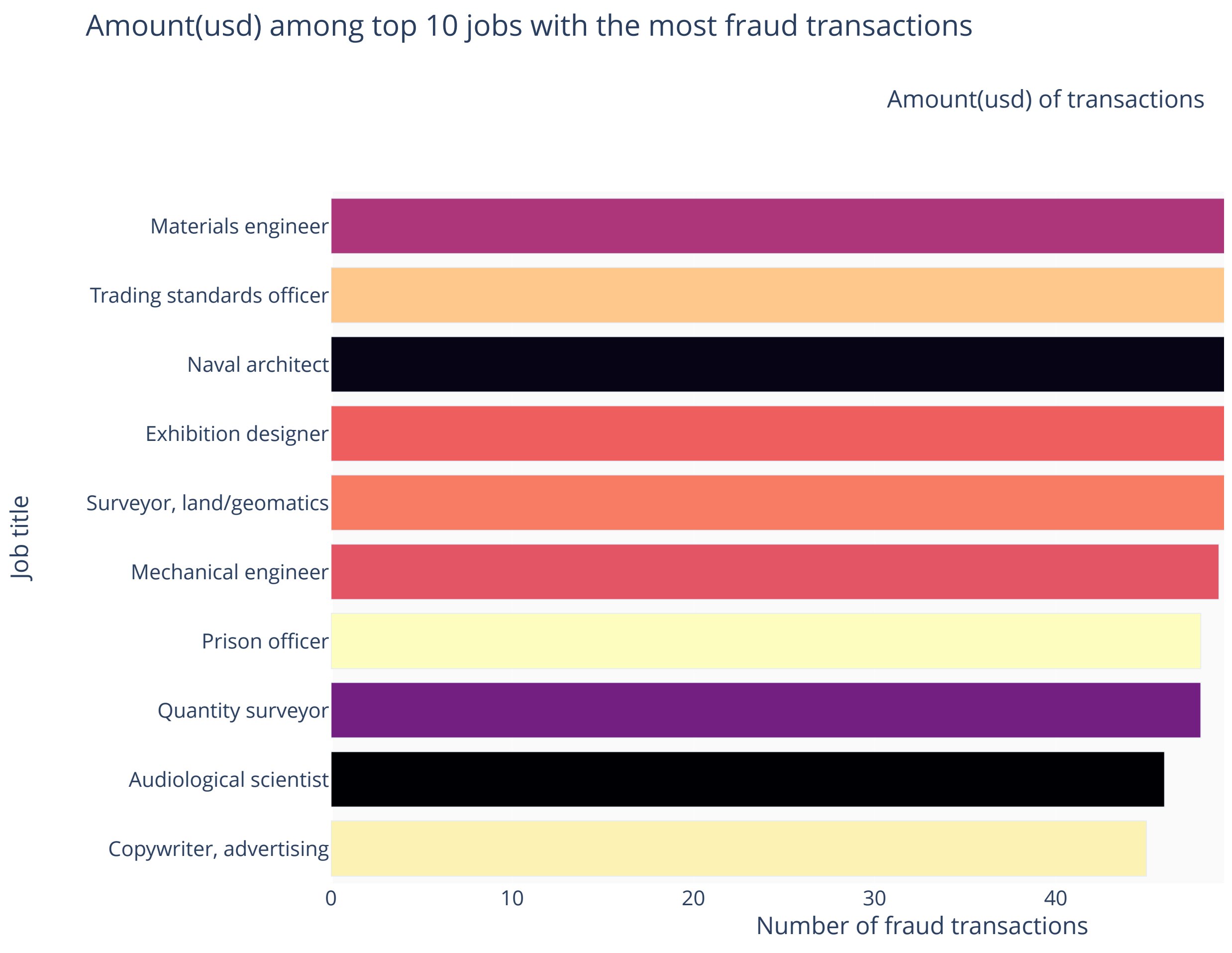

Top 12 merchants with highest number of transactions per week
merchant=fraud_Kilback LLC
merchant=fraud_Cormier LLC
merchar
merchant=fraud_Kuhn LLC
merchant=fraud_Boyer PLC
merchar
merchant=fraud_Cummerata-Jones
merchant=fraud_Kutch LLC
merchant=fra
merchant=fraud_Stroman, Hudson and Erdman
merchant=fraud_Rodriguez Group
merchant=f
amount(usd)
Jan 2019
Jul 2019
Jan 2020
Jul 2020
Jul

In [23]:
```
groups = ['is_fraud','job']
df_ = df_train.groupby(by=groups).agg({"amount(usd)":'mean',"transaction_id":"count"}).fillna(0).re

# Top 10 jobs had most fraud transactions.
df_ = df_[df_.is_fraud==1].sort_values(by='transaction_id',
                                       ascending=False).drop_duplicates('job', keep='first').iloc[:
df_
```

Out[23]:

	is_fraud	job	amount(usd)	transaction_id
780	1	Materials engineer	561.092097	62
971	1	Trading standards officer	478.137143	56
802	1	Naval architect	653.563962	53
681	1	Exhibition designer	524.067255	51
933	1	Surveyor, land/geomatics	510.914800	50
781	1	Mechanical engineer	531.585714	49
845	1	Prison officer	453.897500	48
875	1	Quantity surveyor	591.754167	48
536	1	Audiological scientist	660.311739	46
604	1	Copywriter, advertising	458.743556	45

In [24]:
```
fig = px.bar(df_,
             y='job', x='transaction_id',
             color='amount(usd)',
             color_continuous_scale=px.colors.sequential.Magma,
             labels={'job':'Job title',
                     'transaction_id': 'Number of fraud transactions'},
             category_orders = {"job": df_.job.values},
             width=960,
             height=600)

fig.update_layout(
    title=dict(
        text='Amount(usd) among top 10 jobs with the most fraud transactions'
    ),
    plot_bgcolor='#fafafa'
)

fig.update_coloraxes(
    colorbar=dict(
        title='Amount(usd) of transactions',
        orientation='h',
        x=1
    ),
    reversescale=True
)

fig.show()
```



```
In [25]: #Which credit card number/ credit card holder has most fraud transaction?

         groups = ['credit_card_number']
         df_ = df_train.groupby(by=groups).agg({"amount(usd)":'mean',"transaction_id":"count"}).fillna(0).re
         df_.sort_values('transaction_id', ascending=False, inplace=True)
         df_ = df_.head(10)
```

```
In [26]: df_ = df_train[df_train.is_fraud==1].groupby(by='hour_of_day').agg({'transaction_id':'count'}).rese

         fig = px.bar(data_frame=df_,
                x='hour_of_day',
                y='transaction_id',
                labels={'transaction_id':'Number of transaction'})

         fig.update_layout(
             title=dict(
                 text='Number of FRAUD transactions by hours of day'
             ),
             plot_bgcolor='#fafafa'
         )

         fig.update_xaxes(type='category')
```

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

Out[27]:
```
transaction_time      datetime64[ns]
credit_card_number          category
merchant                      object
category                      object
amount(usd)                  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]
transaction_id                object
unix_time                      int64
merch_lat                    float64
merch_long                   float64
is_fraud                    category
time                  datetime64[ns]
hour_of_day                 category
dtype: object
```

In [28]:
```
# Identify non-numeric columns
non_numeric_cols = df_train.select_dtypes(exclude=np.number).columns
```

In [29]:
```
non_numeric_cols
```

Out[29]:
```
Index(['transaction_time', 'credit_card_number', 'merchant', 'category',
       'first', 'last', 'gender', 'street', 'city', 'state', 'job', 'dob',
       'transaction_id', 'is_fraud', 'time', 'hour_of_day'],
      dtype='object')
```

In [30]:
```
columns_to_drop = ['transaction_time', 'credit_card_number', 'merchant', 'category', 'first', 'last
```

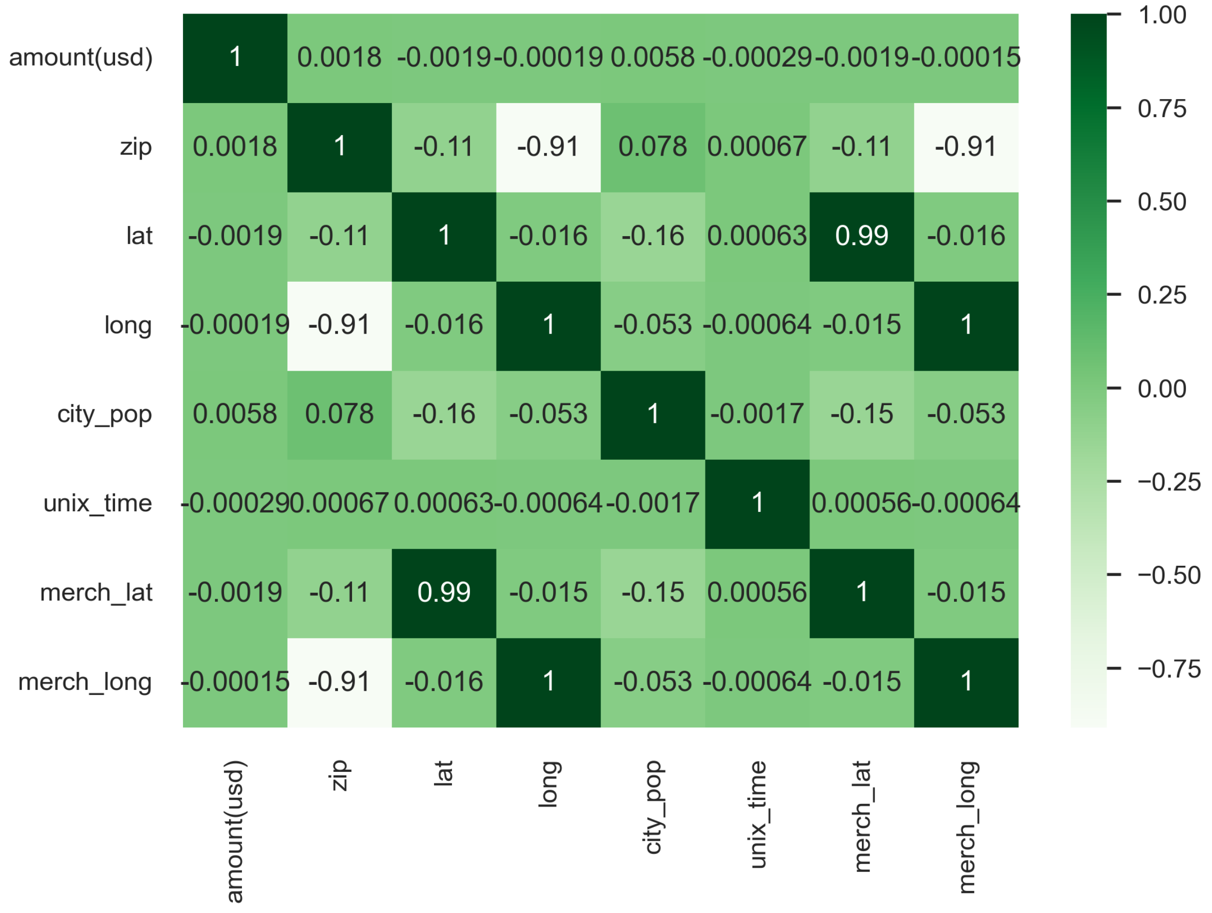
```
new_df_train = df_train.drop(columns=columns_to_drop)
```

In [31]:
```
new_df_train.corr()
```

Out[31]:

	amount(usd)	zip	lat	long	city_pop	unix_time	merch_lat	merch_long
amount(usd)	1.000000	0.001843	-0.001926	-0.000187	0.005818	-0.000293	-0.001873	-0.000151
zip	0.001843	1.000000	-0.114290	-0.909732	0.078467	0.000670	-0.113561	-0.908924
lat	-0.001926	-0.114290	1.000000	-0.015533	-0.155730	0.000632	0.993592	-0.015509
long	-0.000187	-0.909732	-0.015533	1.000000	-0.052715	-0.000642	-0.015452	0.999120
city_pop	0.005818	0.078467	-0.155730	-0.052715	1.000000	-0.001714	-0.154781	-0.052687
unix_time	-0.000293	0.000670	0.000632	-0.000642	-0.001714	1.000000	0.000561	-0.000635
merch_lat	-0.001873	-0.113561	0.993592	-0.015452	-0.154781	0.000561	1.000000	-0.015431
merch_long	-0.000151	-0.908924	-0.015509	0.999120	-0.052687	-0.000635	-0.015431	1.000000

In [32]:
```
sns.heatmap(new_df_train.corr(), cmap="Greens", annot=True)
plt.show()
```



In [33]:
```
features = ['transaction_id','hour_of_day', 'category', 'amount(usd)', 'merchant', 'job']

x = df_train[features].set_index("transaction_id")
y = df_train['is_fraud']

print('X shhape:{}\ny shape:{}'.format(x.shape, y.shape))
```

```
X shhape:(1296675, 5)
y shape:(1296675,)
```

In [34]:
```
from sklearn.preprocessing import OrdinalEncoder

enc = OrdinalEncoder(dtype=np.int64)
enc.fit(x.loc[:,['category', 'merchant', 'job']])
```

```
x.loc[:, ['category', 'merchant', 'job']] = enc.transform(x[['category','merchant','job']])
```

In [35]:
```
x[['category', 'merchant', 'job' ]]
```

Out[35]:

	category	merchant	job
transaction_id			
0b242abb623afc578575680df30655b9	8	514	370
1f76529f8574734946361c461b024d99	4	241	428
a1a22d70485983eac12b5b88dad1cf95	0	390	307
6b849c168bdad6f867558c3793159a81	2	360	328
a41d7549acf90789359a9aa5346dcb46	9	297	116
...	...	...	...
440b587732da4dc1a6395aba5fb41669	0	499	215
278000d2e0d2277d1de2f890067dcc0a	1	2	360
483f52fe67fabef353d552c1e662974c	1	599	308
d667cdcbadaaed3da3f4020e83591c83	1	509	485
8f7c8e4ab7f25875d753b422917c98c9	1	370	467

1296675 rows × 3 columns

In [36]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

def select_features(x_train, y_train, x_test):
    fs = SelectKBest(score_func=chi2, k='all')
    fs.fit(x_train, y_train)
    x_train_fs = fs.transform(x_train)
    x_test_fs = fs.trannsform(x_test)
    return X_train_fs, X_test_fs, fs
```

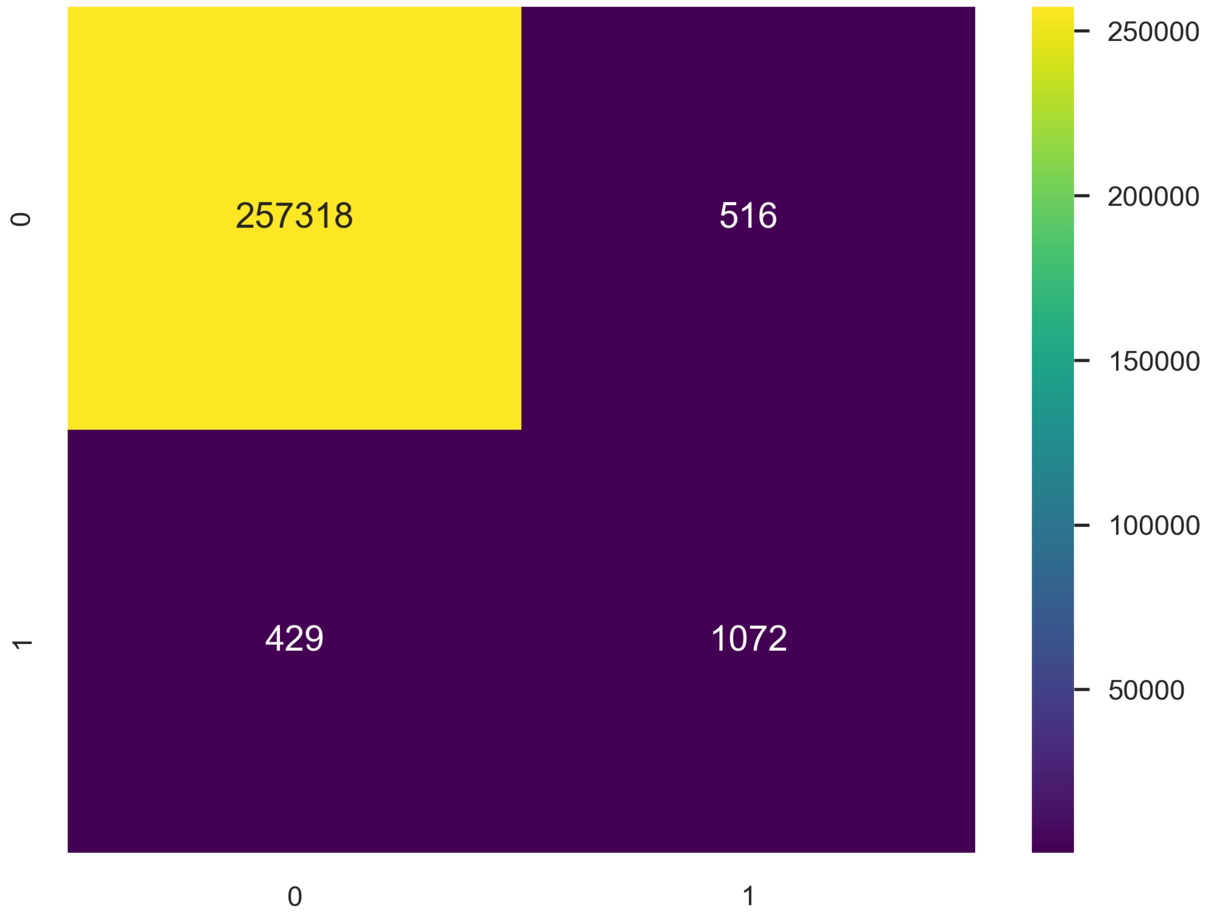
In [37]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y)
print('X_train shape:{}\ny_train shape:{}'.format(x_train.shape,y_train.shape))
print('X_test shape:{}\ny_test shape:{}'.format(y_test.shape,y_test.shape))
```

```
X_train shape:(1037340, 5)
y_train shape:(1037340,)
X_test shape:(259335,)
y_test shape:(259335,)
```

In [38]:
```
from sklearn.tree import DecisionTreeClassifier

dcstree = DecisionTreeClassifier(random_state=42)
dcstree.fit(x_train, y_train)

y_pred = dcstree.predict(x_test)
```

In [39]:
```
fig = plt.figure(figsize=(8,6))
cfs_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cfs_matrix, cmap='viridis', annot=True, fmt='d', annot_kws=dict(fontsize=14))
```

Out[39]:
```
<AxesSubplot:>
```



With Decision Tree we have F1-Score = 0.69 for label 1

SMOTE

In [40]:
```
pip install imblearn
```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.12.0-py3-none-any.whl (257 kB)
     ------------------------------------ 257.7/257.7 kB 659.8 kB/s eta 0:00:00
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\lenovo\anaconda3\lib\site-packages
(from imbalanced-learn->imblearn) (2.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\lenovo\anaconda3\lib\site-packages
(from imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\lenovo\anaconda3\lib\site-packages (from
imbalanced-learn->imblearn) (1.21.5)
Requirement already satisfied: scipy>=1.5.0 in c:\users\lenovo\anaconda3\lib\site-packages (from i
mbalanced-learn->imblearn) (1.9.1)
Collecting joblib>=1.1.1
  Using cached joblib-1.3.2-py3-none-any.whl (302 kB)
Installing collected packages: joblib, imbalanced-learn, imblearn
  Attempting uninstall: joblib
    Found existing installation: joblib 1.1.0
    Uninstalling joblib-1.1.0:
      Successfully uninstalled joblib-1.1.0
Successfully installed imbalanced-learn-0.12.0 imblearn-0.0 joblib-1.3.2
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
    WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\lenovo\anaconda3\lib\site-packages)
```

In [41]:
```
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy={1:48050}, random_state=42)

x_train_smote, y_train_smote = smote.fit_resample(x_train.astype('float'), y_train)
print("Before SMOTE:", Counter(y_train))
print("After SMOTE:", Counter(y_train_smote))
```

```
Before SMOTE: Counter({0: 1031335, 1: 6005})
After SMOTE: Counter({0: 1031335, 1: 48050})
```

In [42]:
```
class test_model:
    from sklearn.metrics import classification_report
    def __init__(self):
        self.metrics = ['prfs', 'auc', 'acc']

    def fit_predict(model, x_train, x_test, y_train, y_test):
        model = model
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)
        return y_pred

    def evaluate(y_pred, metrics):
        results = {}
        for metric in metrics:
            if metric == 'prfs':
                prfs = classification_report(y_test, y_pred)
                results['prfs'] = prfs
            elif metric == 'auc':
                auc_score = roc_auc_score(y_test, y_pred)
                results['auc'] = auc_score
            elif metric == 'acc':
                results['acc'] = accuracy_score(y_test, y_pred)
            else:
                print('Not available metric!')
        return results
```

In [43]:
```
from sklearn.ensemble import RandomForestClassifier

# Specify your metric here
metrics = ['prfs']
print("====================SMOTE====================")
RDForest_eval = test_model.evaluate(y_pred=test_model.fit_predict(RandomForestClassifier(random_sta
                                                                  x_train_smote,
                                                                  x_test,
                                                                  y_train_smote,
                                                                  y_test
                                                                 ),
                                    metrics=metrics
                                    )

print("Random Forest model evaluate:\n", RDForest_eval['prfs'])
```

```
=====================SMOTE=====================
Random Forest model evaluate:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    257834
           1       0.79      0.75      0.77      1501

    accuracy                           1.00    259335
   macro avg       0.90      0.88      0.89    259335
weighted avg       1.00      1.00      1.00    259335
```

With RandomForectClassifier we have better F1-Score = 0.77 for label1

Try tuning some important Hyperparameter for RDF

In [44]:
```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 50, stop = 200, num = 4)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 50, num = 5)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf
              }

print(random_grid)
```

```
{'n_estimators': [50, 100, 150, 200], 'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30,
40, 50, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]}
```

In [45]:
```
from sklearn.metrics import f1_score, make_scorer
f1 = make_scorer(f1_score, greater_is_better=True, pos_label=1)
```

In [46]:
```
rf = RandomForestClassifier()
rf_random = RandomizedSearchCV(estimator = rf,
                               param_distributions = random_grid,
                               n_iter = 50,
                               cv = 5,
                               verbose=2,
                               random_state=42
                              )

#Fit and show the best parameters
rf_random.fit(x_train, y_train)
print(rf_random.best_estimator_)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
50; total time= 1.3min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
50; total time= 1.2min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
50; total time= 1.4min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
50; total time= 1.3min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
50; total time= 1.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 4.9min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.3min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time=89.9min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.6min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.3min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.1min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 4.0min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.4min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.4min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.4min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.4min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.3min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=2
00; total time= 4.7min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=2
00; total time= 4.7min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=
150; total time= 3.4min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=
150; total time= 3.4min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=
150; total time= 3.5min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=
150; total time= 3.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=
150; total time= 3.5min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
```

```
50; total time= 3.5min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.4min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.0min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.0min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.0min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.0min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.0min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 4.7min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=30, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 4.6min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 3.9min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 3.9min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 4.0min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=20
```

```
0; total time= 4.0min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=20
0; total time= 3.9min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=10
0; total time= 2.4min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators
=150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators
=150; total time= 3.4min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators
=150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators
=150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators
=150; total time= 3.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 1.9min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 1.9min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.0min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.0min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.0min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=40, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.4min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
150; total time= 3.4min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
150; total time= 3.5min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
150; total time= 4.2min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
150; total time= 7.7min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
150; total time= 7.5min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 8.7min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 8.6min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 8.7min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 8.7min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=20
0; total time= 8.6min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 5.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 5.1min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 5.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 5.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 5.1min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time=10.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time=10.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time=10.4min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time=10.3min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time=10.5min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=10
0; total time= 5.1min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=10
0; total time= 5.2min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=10
0; total time= 5.2min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=10
0; total time= 5.1min
[CV] END max_depth=30, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=10
0; total time= 5.1min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time=10.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time=10.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time=10.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time=10.2min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time=10.2min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 7.7min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 7.6min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 7.8min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 7.8min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 7.7min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 5.3min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.5min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
```

```
0; total time= 3.5min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=15
0; total time= 3.5min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=15
0; total time= 3.5min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=
200; total time= 4.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=
200; total time= 4.5min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=
200; total time= 4.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=
200; total time= 4.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=
200; total time= 4.5min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=2, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time= 4.5min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time= 4.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time= 4.7min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time= 5.0min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=
200; total time= 5.6min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=10
0; total time= 2.9min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=10
0; total time= 2.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=10
0; total time= 2.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=
150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=
150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=
150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=
150; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=2, n_estimators=
150; total time= 3.4min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.5min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
50; total time= 3.4min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.1min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.1min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.2min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.1min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4, min_samples_split=10, n_estimators
=50; total time= 1.1min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=50, max_features=sqrt, min_samples_leaf=4, min_samples_split=5, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=1
00; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=15
0; total time= 3.5min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=15
0; total time= 3.5min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=15
0; total time= 3.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=2, n_estimators=10
0; total time= 2.3min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
00; total time= 2.0min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
```

```
00; total time= 2.0min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
00; total time= 1.9min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
00; total time= 2.0min
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4, min_samples_split=10, n_estimators=1
00; total time= 1.9min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5
0; total time= 1.1min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=10, n_estimators=5
0; total time= 1.2min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time= 4.7min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=50, max_features=auto, min_samples_leaf=1, min_samples_split=10, n_estimators=2
00; total time= 4.6min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time=368.9min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.3min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.4min
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=5, n_estimators=5
0; total time= 1.3min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.8min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.8min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 4.0min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 4.0min
[CV] END max_depth=40, max_features=auto, min_samples_leaf=4, min_samples_split=5, n_estimators=15
0; total time= 3.5min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=
200; total time= 4.8min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=
200; total time= 5.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=
200; total time= 4.9min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=
200; total time= 5.3min
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2, min_samples_split=5, n_estimators=
200; total time= 5.1min
RandomForestClassifier(max_depth=50, max_features='sqrt', min_samples_split=10,
                       n_estimators=200)
```

In [47]:
```python
rf_random = RandomForestClassifier(max_features='sqrt',
                                   n_estimators=200,
                                   random_state=41
                                   )
8888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888888
rf_random.fit(x_train, y_train)
y_pred = rf_random.predict(x_test)

# Print reprort
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    257834
           1       0.87      0.72      0.79      1501

    accuracy                           1.00    259335
   macro avg       0.93      0.86      0.89    259335
weighted avg       1.00      1.00      1.00    259335
```

After tuning, we have F1-Score = 0.79 for label 1.

In [48]:
```python
df_test = df_test.rename(columns={"trans_date_trans_time":"transaction_time",
                         "cc_num":"credit_card_number",
                         "amt":"amount(usd)",
                         "trans_num":"transaction_id"}
                        )

# Apply function utcfromtimestamp and drop column unix_time
df_test['time'] = df_test['unix_time'].apply(datetime.utcfromtimestamp)

# Add cloumn hour of day
df_test['hour_of_day'] = df_test.time.dt.hour

df_test = df_test[features].set_index("transaction_id")
enc = OrdinalEncoder(dtype=np.int64)
enc.fit(df_test.loc[:, ['category','merchant','job']])

df_test.loc[:, ['category','merchant','job']] = enc.transform(df_test[['category','merchant','job']
```

In [49]:
```python
y_pred  = rf_random.predict(df_test)
y_proba = rf_random.predict_proba(df_test)[:, 1]
```

In [50]:
```python
df_test["Fraud_Proba"] = y_proba
df_test["Fraud_Predict"] = y_pred
```

In [51]:
```python
result = df_test[["Fraud_Proba", "Fraud_Predict"]]
# Store result in a CSV file
result.to_csv(r"./PredictFraud_Result.csv")
```

In [52]:
```python
result
```

Out[52]:

	Fraud_Proba	Fraud_Predict
transaction_id		
2da90c7d74bd46a0caf3777415b3ebd3	0.00	0
324cc204407e99f51b0d6ca0055005e7	0.00	0
c81755dbbbea9d5c77f094348a7579be	0.00	0
2159175b9efe66dc301f149d3d5abf8c	0.00	0
57ff021bd3f328f8738bb535c302a31b	0.00	0
...	...	...
9b1f753c79894c9f4b71f04581835ada	0.00	0
2090647dac2c89a1d86c514c427f5b91	0.00	0
6c5b7c8add471975aa0fec023b2e8408	0.00	0
14392d723bb7737606b2700ac791b7aa	0.00	0
1765bb45b3aa3224b4cdcb6e7a96cee3	0.01	0

555719 rows × 2 columns

In [ ]: