In [1]:
```
#import all the required libraries
import numpy as np
import pandas as pd
import dtsera1
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]: *Kloading the data*
```
df_train = pd.read_csv('C:\Users\Harshith\OneDrive\Documents\sodaPlot\credit card fraudTraindTrain.csv')
df_test = pd.read_csv('C:\Users\Harshith\OneDrive\Documents\sodaPlot\credit card fraudTraindTest.csv')
```

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

Out[3]: (1296675, 23)

In [4]:
```
df_test.shape
```

Out[4]: (555719, 23)

In [5]:
```
df = pd.concat([df_train,df_test])
df.shape
```

Out[5]: (1852394, 23)

In [6]:
```
df.head()
```

Out[6]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	lat	long	city_pop	job	dob
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	...	36.0788	-81.1781	3495	Psychologist, counselling	1988-09-15 00:00:00
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43038 Ring Groves Suite 393	...	48.8878	-118.2105	149	Special educational needs teacher	1978-06-21
2	2	2019-01-01 00:00:51	38998400557661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	42.1808	-112.2620	4654	Nature conservation officer	1962-10-28
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 468	...	46.2306	-112.1138	1939	Patent attorney	1967-12-27
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	...	38.4207	-79.4629	99	Dance movement psychotherapist	1986-03-28

5 rows × 23 columns

In [7]: *#preprocessing the data*
```
df.groupby("is_fraud").count()["cc_num"].plot(kind="bar")
```

In [8]:
```
#kde
sns.kdeplot(df['is_fraud'])
```

Out[8]:

In [9]: *#there can be a overfitting issue since the data is not balanced properly it may lead to poor model performance*
#we gotta make it balanced
```
fraud = df[df['is_fraud']==1]
is_not_fraud = df[df['is_fraud']==0]
print(fraud.shape[0])
print(is_not_fraud.shape[0])
#we can clearly see the imbalance
```

```
9651
1842743
```

In [10]:
```
is_not_fraud = is_not_fraud.sample(fraud.shape[0],random_state = 42)
df_new = pd.concat([fraud,is_not_fraud])
```

In [11]:
```
df_new.groupby("is_fraud").count()["cc_num"].plot(kind="bar")
#now the data is balanced
```

Out[10]:

In [11]: *#let us check the structure of dataset*
```
df_new.info()
#there are many unwanted columns that are not helpful for a model to detect
#other features are basically the unwanted since they are useless in regression or other mathematical detection
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19302 entries, 1440 to 13873
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             19302 non-null  int64
 1   trans_date_trans_time  19302 non-null  object
 2   cc_num                 19302 non-null  int64
 3   merchant               19302 non-null  object
 4   category               19302 non-null  object
 5   amt                    19302 non-null  float64
 6   first                  19302 non-null  object
 7   last                   19302 non-null  object
 8   gender                 19302 non-null  object
 9   street                 19302 non-null  object
 10  city                   19302 non-null  object
 11  state                  19302 non-null  object
 12  zip                    19302 non-null  int64
 13  lat                    19302 non-null  float64
 14  long                   19302 non-null  float64
 15  city_pop               19302 non-null  int64
 16  job                    19302 non-null  object
 17  dob                    19302 non-null  object
 18  trans_num              19302 non-null  object
 19  unix_time              19302 non-null  int64
 20  merch_lat              19302 non-null  float64
 21  merch_long             19302 non-null  float64
 22  is_fraud               19302 non-null  int64
dtypes: float64(5), int64(7), object(11)
memory usage: 3.5+ MB
```

In [12]: *#let us eliminate the unwanted columns*
```
unwanted_cols = ['Unnamed: 0',
                 'cc_num',
                 'merchant',
                 'first',
                 'last',
                 'street',
                 'trans_num',
                 'unix_time',
                 'zip',
                 'merch_lat',
                 'gender',
                 'merch_long',
                 'state',
                 'trans_date_trans_time']
df_new.drop(columns = unwanted_cols,inplace = True)
```

In [13]: *#checking if there any null values in the dataset so that we can continue the process*
```
df_new.isnull().sum().sum()
```

Out[13]: 0

In [14]: *#checking if there any any duplicate values or rows present in the dataset*
```
df_new.duplicated().sum()
```

Out[14]: 2

In [15]: *#eliminating the duplicate values from the dataset*
```
df_new.drop_duplicated()
```

Out[15]:

	category	amt	city	state	lat	long	city_pop	job	dob	is_fraud
1440	grocery_pos	283.06	Collettsville	NC	35.9946	-81.7266	885	1988-09-15	1988-09-15	1
2472	gas_transport	11.52	San Antonio	TX	29.4400	-98.4590	1590767	1960-10-29	1960-10-29	1
2523	grocery_pos	279.71	San Antonio	TX	29.4400	-98.4590	1590767	1960-10-29	1960-10-29	1
2546	gas_transport	7.03	Collettsville	NC	35.9946	-81.7266	885	1988-09-15	1988-09-15	1
2553	grocery_pos	275.73	San Antonio	TX	29.4400	-98.4590	1590767	1960-10-29	1960-10-29	1
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
99709	gas_transport	62.58	Johns Island	SC	32.8357	-79.8217	20476	1987-06-04	1987-06-04	0
274604	misc_net	373.25	Belfast	NY	42.5209	-78.0843	1766	1962-06-04	1962-06-04	0
807093	misc_pos	2.74	Greendale	WI	42.9373	-87.8943	13973	1951-06-08	1951-06-08	0
861157	shopping_net	161.28	Westport	KY	38.4815	-85.4524	564	1996-04-10	1996-04-10	0
135673	kids_pets	163.60	Hinckley	OH	41.2419	-81.7453	7646	1990-09-15	1990-09-15	0

19300 rows × 9 columns

In [16]: *#let us check the structure of the dataframe after performing the data preprocessing*
```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19302 entries, 1440 to 13873
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   category  19302 non-null  object
 1   amt       19302 non-null  float64
 2   city      19302 non-null  object
 3   state     19302 non-null  object
 4   lat       19302 non-null  float64
 5   long      19302 non-null  float64
 6   city_pop  19302 non-null  int64
 7   job       19302 non-null  object
 8   is_fraud  19302 non-null  int64
dtypes: float64(3), int64(2), object(4)
memory usage: 1.5+ MB
```

In [17]: *#let us perform feature engineering*
#to improve the performance of the model we will create new features based on the old ones
```
df_new["dob"] = pd.to_datetime(df_new["dob"])
```

In [18]:
```
from datetime import datetime
df_new["dob"] = df_new["dob"].dt.year
df_new['age'] = datetime.now().year - df_new['dob']
df_new = df_new.rename(columns = {"dob" : "age"})
df_new
```

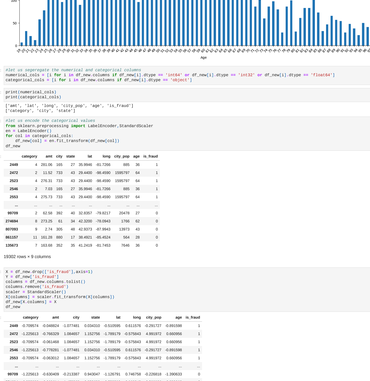
Out[18]:

	category	amt	city	state	lat	long	city_pop	age	is_fraud
1440	grocery_pos	283.06	Collettsville	NC	35.9946	-81.7266	885	36	1
2472	gas_transport	11.52	San Antonio	TX	29.4400	-98.4590	1590767	64	1
2523	grocery_pos	279.71	San Antonio	TX	29.4400	-98.4590	1590767	64	1
2546	gas_transport	7.03	Collettsville	NC	35.9946	-81.7266	885	36	1
2553	grocery_pos	275.73	San Antonio	TX	29.4400	-98.4590	1590767	64	1
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
99709	gas_transport	62.58	Johns Island	SC	32.8357	-79.8217	20476	27	0
274604	misc_net	373.25	Belfast	NY	42.5209	-78.0843	1766	62	0
807093	misc_pos	2.74	Greendale	WI	42.9373	-87.8943	13973	43	0
861157	shopping_net	161.28	Westport	KY	38.4815	-85.4524	564	26	0
135673	kids_pets	163.60	Hinckley	OH	41.2419	-81.7453	7646	36	0

19302 rows × 9 columns

In [19]: *#now we have to understand which age group are maximum involved in the fraud*
#let us make a visualization
```
plt.figure(figsize = (20,7))
sns.countplot(df_new['age'].sort_values(ascending=False).astype(str),hue=df_new['is_fraud'],palette=sns.color_palette("Set1",2))
axis.set_xlabel("Age")
axis.set_ylabel("Count of Fraud cases")
axis.set_title("Count of Fraud cases by the age")
plt.xticks(rotation=45,ha='right')
plt.show()
```

Out[19]:



In [20]: *#let us separate the numerical and categorical values*
```
numerical_cols = df_new.select_dtypes(include=['int64','float64'])
categorical_cols = [ for i in df_new.columns if df_new[i].dtype == 'object']
```

In [21]:
```
print(numerical_cols)
print(categorical_cols)
```

```
['amt', 'lat', 'long', 'city_pop', 'age', 'is_fraud']
['category', 'city', 'state']
```

In [22]: *#let us encode the categorical values*
```
from sklearn.preprocessing import LabelEncoder,StandardScaler
le = LabelEncoder()
for col in categorical_cols:
    df_new[col] = ms.fit_transform(df_new[col])
df_new
```

Out[22]:

	category	amt	city	state	lat	long	city_pop	age	is_fraud
1440	9	283.06	165	27	35.9946	-81.7266	885	36	1
2472	2	11.52	733	43	29.4400	-98.4590	1590767	64	1
2523	9	279.71	733	43	29.4400	-98.4590	1590767	64	1
2546	2	7.03	165	27	35.9946	-81.7266	885	36	1
2553	4	275.73	733	43	29.4400	-98.4590	1590767	64	1
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
99709	2	62.58	392	40	32.8357	-79.8217	20476	27	0
274604	9	373.25	61	34	42.5209	-78.0843	1766	62	0
807093	9	2.74	305	48	42.9373	-87.8943	13973	43	0
861157	11	161.28	980	17	38.4815	-85.4524	564	26	0
135673	7	163.60	352	35	41.2419	-81.7453	7646	36	0

19302 rows × 9 columns

In [23]:
```
X = df_new.drop(['is_fraud',],axis=1)
Y = df_new['is_fraud']
columns = df_new.columns.tolist()[:-1]
scaler = StandardScaler()
scaler = StandardScaler()
X[columns] = scaler.fit_transform(X[columns])
df_new[X.columns] = X
df_new
```

Out[23]:

	category	amt	city	state	lat	long	city_pop	age	is_fraud
1440	-0.708574	-0.040024	-1.077441	-0.034210	-0.510595	-0.611076	-0.291727	-0.691598	1
2472	-0.225812	-0.786039	-1.064007	-1.325756	-1.786179	-0.575043	-4.991072	-0.600596	1
2523	-0.708574	-0.041048	-1.064007	-1.325756	-1.786179	-0.575043	-4.991072	-0.600596	1
2546	-0.225812	-0.779281	-1.077441	-0.034210	-0.510595	-0.611076	-0.291727	-0.691598	1
2553	-0.708574	-0.063012	-1.064007	-1.325756	-1.786179	-0.575043	-4.991072	-0.600596	1
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
99709	-0.225812	-0.630408	-0.211587	-0.940547	-1.126791	-0.746758	-0.229018	-1.390633	0
274604	0.932505	-0.069614	-1.473365	0.552630	0.732260	0.600542	-0.288609	0.930060	0
807093	0.500524	-0.709701	-0.544059	1.502172	0.843095	0.190707	-0.244368	-0.045648	0
861157	1.198564	-0.367073	1.044225	-0.804718	-0.025410	-0.347174	-0.293766	-0.338184	0
135673	0.004495	-0.361195	-0.305050	-0.503253	0.512979	0.010249	-0.269329	-0.691598	0

19302 rows × 9 columns

In [24]: *#splitting train and test set*
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=42)
print("x_train.shape ",x_train.shape)
print("x_test.shape ",x_test.shape)
print("y_train.shape ",y_train.shape)
print("y_test.shape ",y_test.shape)
```

```
x_train.shape  (15441, 8)
x_test.shape  (3861, 8)
y_train.shape  (15441,)
y_test.shape  (3861,)
```

In [25]: *#using randomforest algorithm for detection since it is very effective and suitable when compared to other models*
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier()
rf_model.fit(x_train,y_train)
y_pred = rf_model.predict(x_test)
y_pred
```

Out[25]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [26]: *#as our model has successfully trained let us check the metrics of the models*
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy of the model is: {accuracy*100:.2f}%")
```

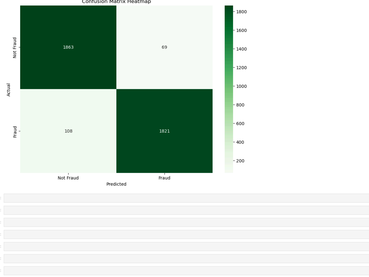
```
Accuracy of the model is: 98.42%
```

In [27]: *#to discover if the model is high let us also provide other metrics like confusion matrix,classification_report*
```
from sklearn.metrics import confusion_matrix,classification_report
confusion_mat = confusion_matrix(y_test,y_pred)
print("Confusion matrix:")
print(confusion_mat)
print("\nClassification report: ")
print(classification_report(y_test,y_pred))
```

```
Confusion matrix:
[[3852   69]
 [  108 1821]]

Classification report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.96      3852
           1       0.96      0.94      0.95      1929

    accuracy                           0.95      3861
   macro avg       0.95      0.95      0.95      3861
weighted avg       0.95      0.95      0.95      3861
```

In [28]: *#let us visualize the confusion matrix*
```
plt.figure(figsize=(8,7))
sns.heatmap(confusion_mat,annot=True,fmt='d',cmap='Greens',xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])
plt.ylabel('Predicted')
plt.xlabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

Out[28]:



In [29]:

In [30]:

In [31]:

In [32]:

In [33]:

In [34]:

In [35]:

In [36]:

In [37]: