Data Visualization

In [117…
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [118…
```python
df = pd.read_csv('creditcardcsvpresent.csv')
```

In [119…
```python
df.head(10)
```

Out[119…

	Merchant_id	Transaction date	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransaction	isHighRiskCountry	Dail
0	3160040998	NaN	100.0	3000.0	N	5	Y	Y	
1	3160040998	NaN	100.0	4300.0	N	5	Y	Y	
2	3160041896	NaN	185.5	4823.0	Y	5	N	N	
3	3160141996	NaN	185.5	5008.5	Y	8	N	N	
4	3160241992	NaN	500.0	26000.0	N	0	Y	Y	
5	3160241992	NaN	500.0	27000.0	N	0	Y	Y	
6	3160272997	NaN	262.5	11287.5	N	0	N	N	
7	3162041996	NaN	185.5	11130.0	Y	20	N	N	
8	3162041996	NaN	185.5	6121.5	Y	20	N	N	
9	3162041996	NaN	185.5	7049.0	Y	20	N	N	

In [120…
```python
df.tail(10)
```

Out[120…

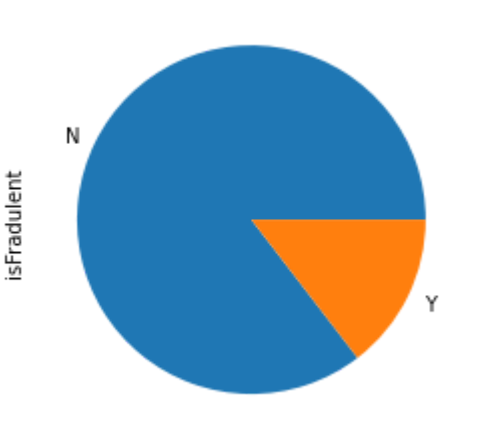
	Merchant_id	Transaction date	Average Amount/transaction/day	Transaction_amount	Is declined	Total Number of declines/day	isForeignTransaction	isHighRiskCountry
3065	6662015632	NaN	240.456831	2164.111484	N	0	N	N
3066	6664866404	NaN	277.746824	7499.164245	N	2	N	N
3067	6665254221	NaN	147.710676	2067.949461	N	0	N	N
3068	6665906072	NaN	695.391196	1390.782391	N	0	N	N
3069	6661273529	NaN	1000.000000	25000.000000	Y	0	Y	N
3070	6661273532	NaN	500.000000	11000.000000	Y	0	N	N
3071	6661273532	NaN	800.000000	0.000000	Y	0	N	N
3072	6661273533	NaN	800.000000	20800.000000	Y	0	N	N
3073	6661273532	NaN	1500.000000	12000.000000	Y	0	Y	Y
3074	6661273533	NaN	1500.000000	36000.000000	Y	0	Y	Y

In [121…
```python
df.shape
```

Out[121… (3075, 12)

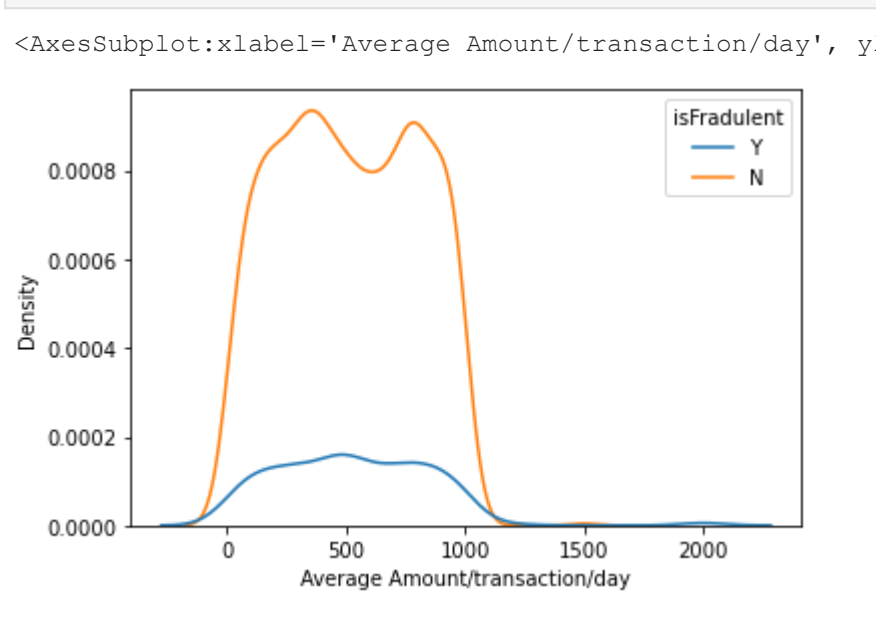
In [122…
```python
df['isFradulent'].value_counts().plot(kind='pie')
```

Out[122… <AxesSubplot:ylabel='isFradulent'>



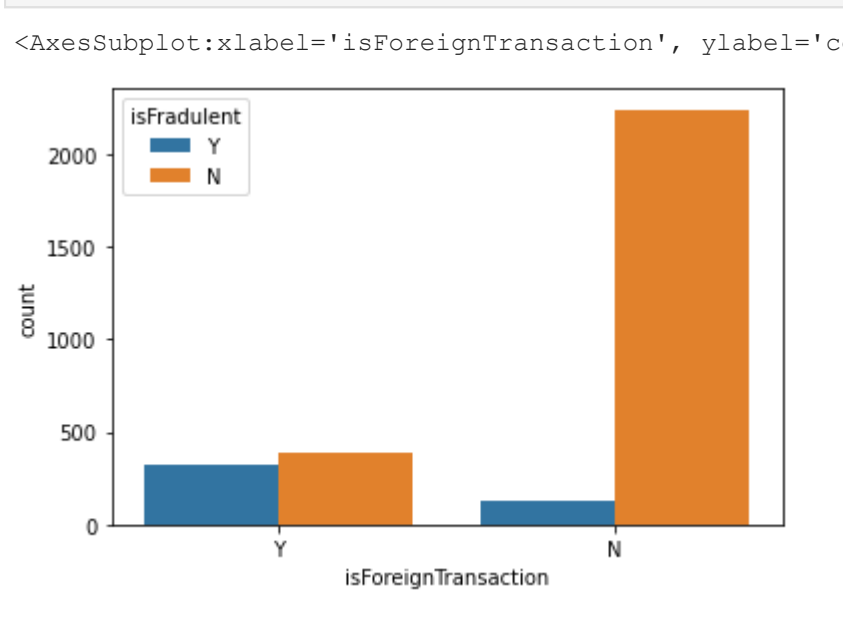
In [123…
```python
sns.kdeplot(x=df['Average Amount/transaction/day'],hue=df['isFradulent'])
```

Out[123… <AxesSubplot:xlabel='Average Amount/transaction/day', ylabel='Density'>



In [124…
```python
sns.countplot(x=df['isForeignTransaction'],hue=df['isFradulent'])
```

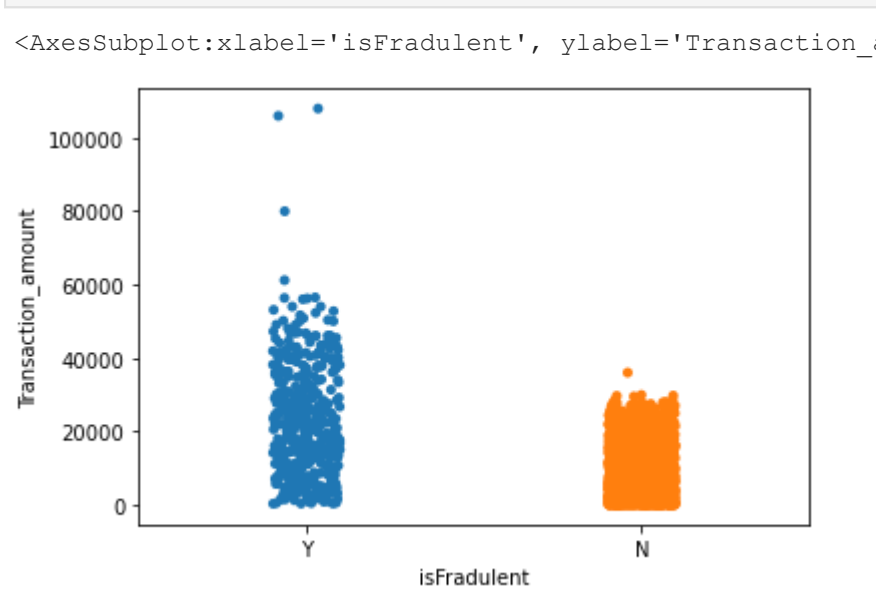
Out[124… <AxesSubplot:xlabel='isForeignTransaction', ylabel='count'>



The number of foreign transaction which have took place among them most of them are fraud

In [125…
```python
sns.stripplot(y=df['Transaction_amount'],x=df['isFradulent'])
```

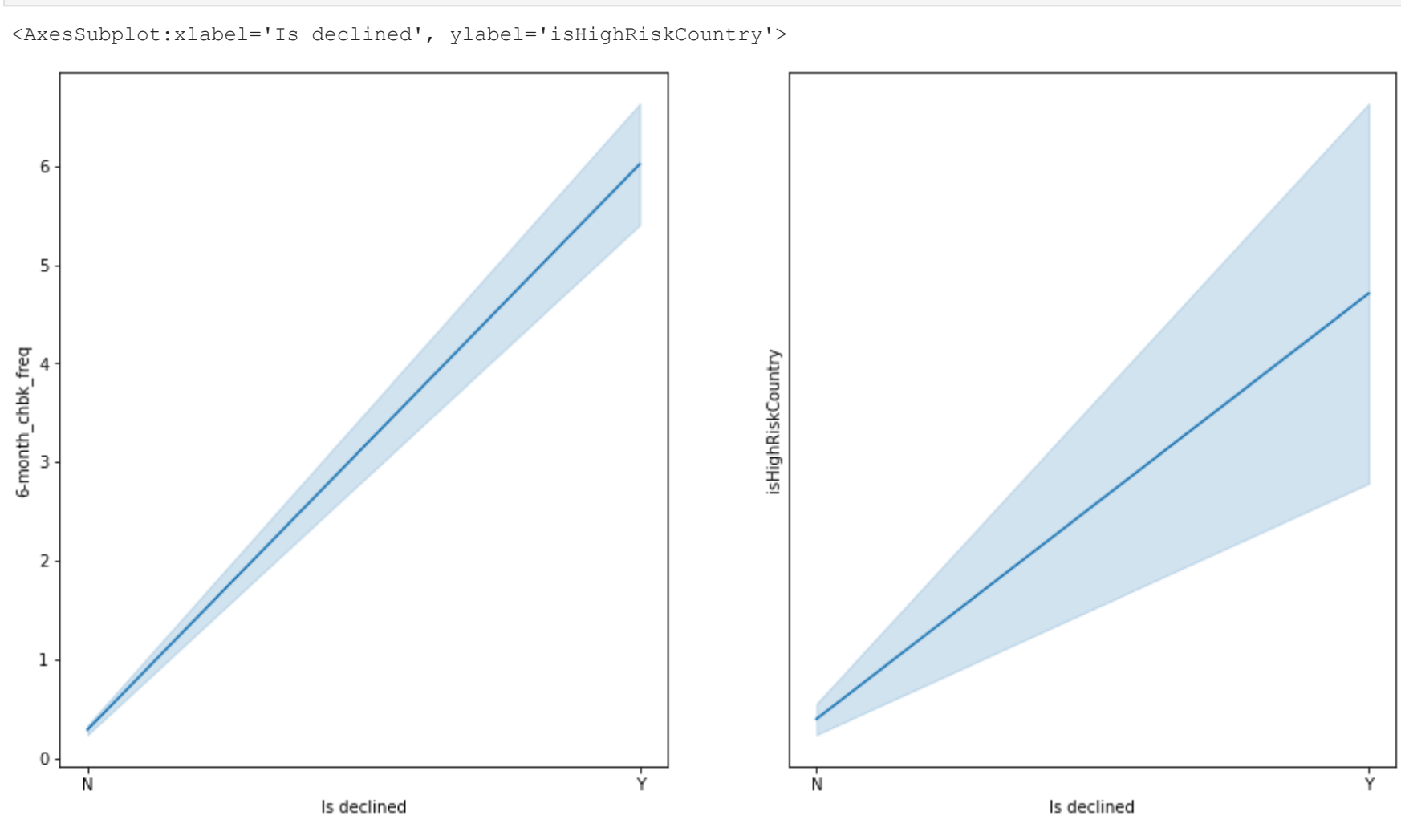
Out[125… <AxesSubplot:xlabel='isFradulent', ylabel='Transaction_amount'>



Looks like big transaction amount are more likely to be a fraud

In [126…
```python
plt.figure(figsize=(15,8))

plt.subplot(1,2,1)
sns.lineplot(x=df['Is declined'],y=df['6-month_chbk_freq'])

plt.subplot(1,2,2)
sns.lineplot(x=df['Is declined'],y=df['isHighRiskCountry'])
```

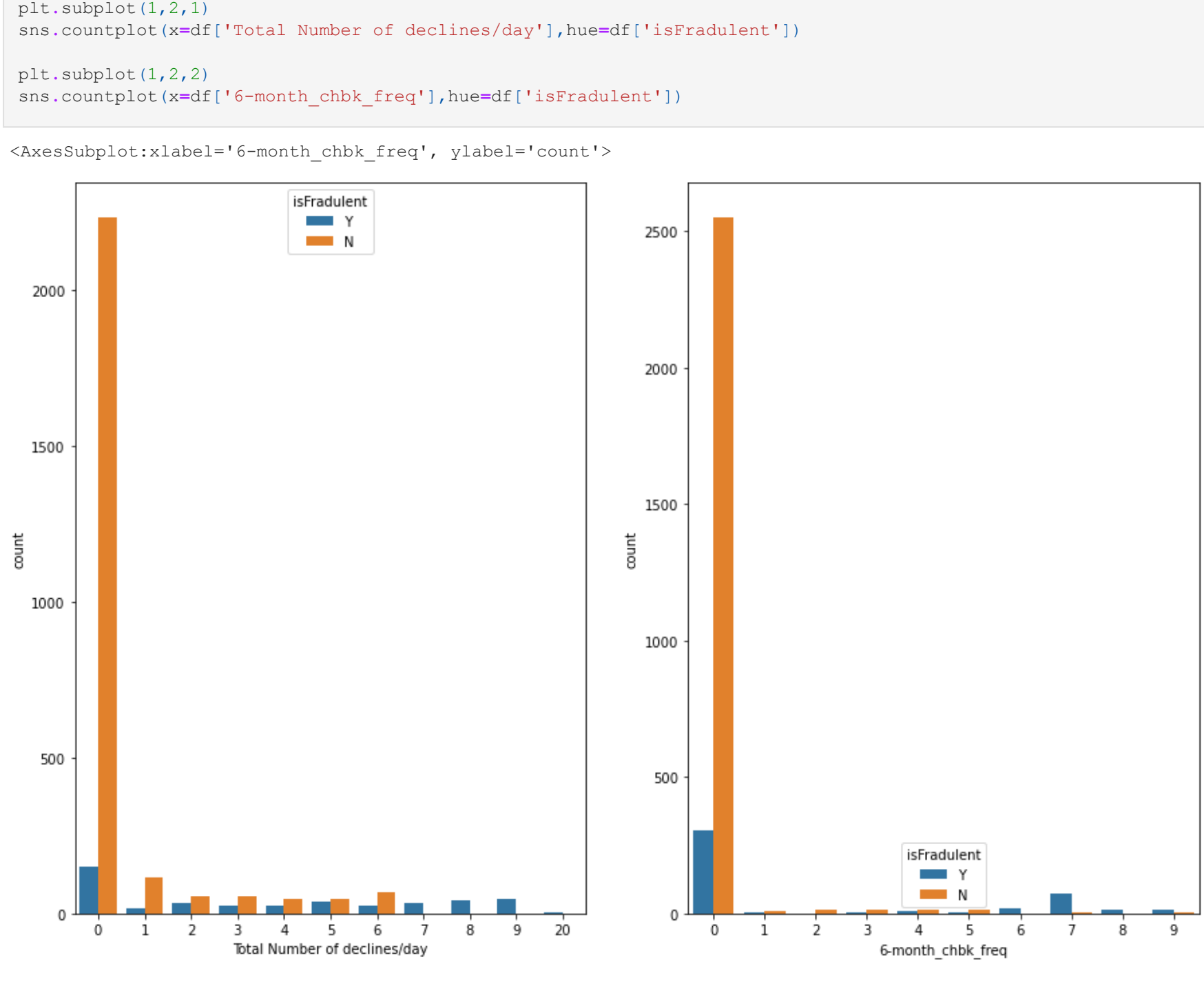
Out[126… <AxesSubplot:xlabel='Is declined', ylabel='isHighRiskCountry'>



> as number of decline increases number of charge back also increases
>
> Similarly is transaction is made to a high risk country chance of getting declined also increases

In [127…
```python
plt.figure(figsize=(15,10))

plt.subplot(1,2,1)
sns.countplot(x=df['Total Number of declines/day'],hue=df['isFradulent'])

plt.subplot(1,2,2)
sns.countplot(x=df['6-month_chbk_freq'],hue=df['isFradulent'])
```

Out[127… <AxesSubplot:xlabel='6-month_chbk_freq', ylabel='count'>



> if we look carefully as the total number of declines increases the chances of being fradulent increases
>
> Similarly is frequency of charge_back increases then changes of being Fraud also increases

Data Cleaning

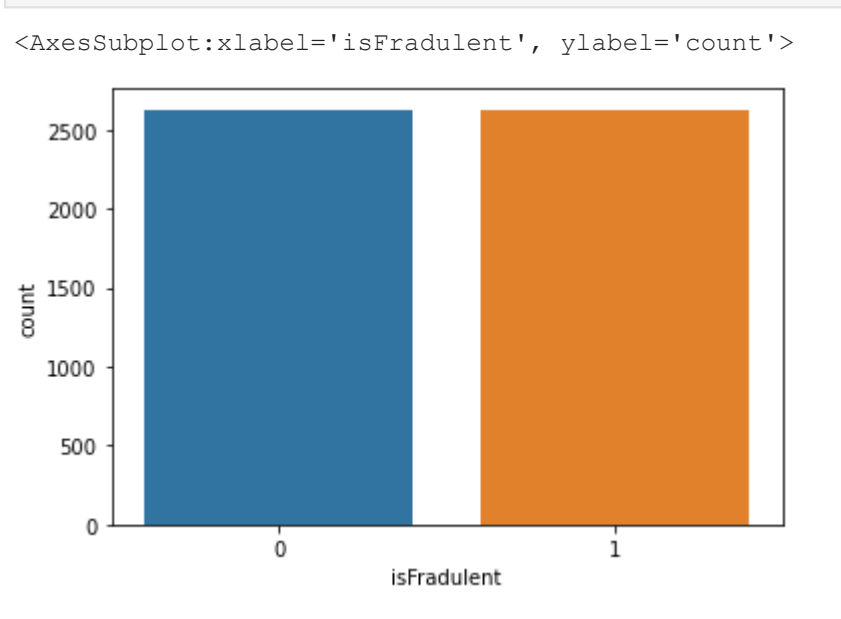
In [128…
```python
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(df['isFradulent'])
df['isFradulent'] = le.transform(df['isFradulent'])

le.fit(df['isHighRiskCountry'])
df['isHighRiskCountry'] = le.transform(df['isHighRiskCountry'])

le.fit(df['Is declined'])
df['Is declined'] = le.transform(df['Is declined'])

le.fit(df['isForeignTransaction'])
df['isForeignTransaction'] = le.transform(df['isForeignTransaction'])
```

In [130…
```python
df.drop(['Merchant_id','Transaction date'],axis=1,inplace=True)
```

In [131…
```python
from sklearn.utils import resample,shuffle

zero =df[df['isFradulent']==0]
one = df[df['isFradulent']==1]

upsampled1 = resample(one, replace=True, n_samples=zero.shape[0])

df = pd.concat([zero,upsampled1])

df = shuffle(df)
```

In [133…
```python
sns.countplot(x=df['isFradulent'])
```

Out[133… <AxesSubplot:xlabel='isFradulent', ylabel='count'>



Model Building

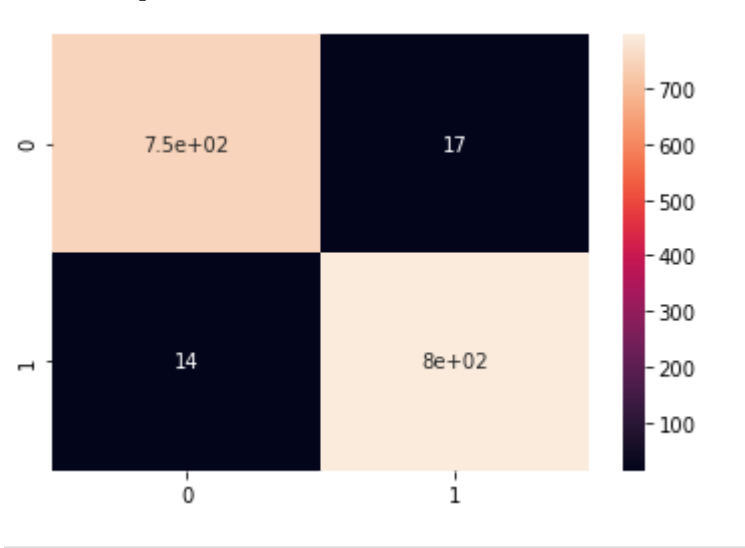
In [134…
```python
x = df.drop(['isFradulent'],axis=1)
y = df['isFradulent']
```

In [135…
```python
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.7,random_state=101)
```

1)Logistic regression

In [136…
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix

lg = LogisticRegression(max_iter=450)
lg.fit(x_train,y_train)
lg_predict = lg.predict(x_test)
lg_cm = confusion_matrix(lg_predict,y_test)
sns.heatmap(lg_cm,annot=True)
```

Out[136… <AxesSubplot:>



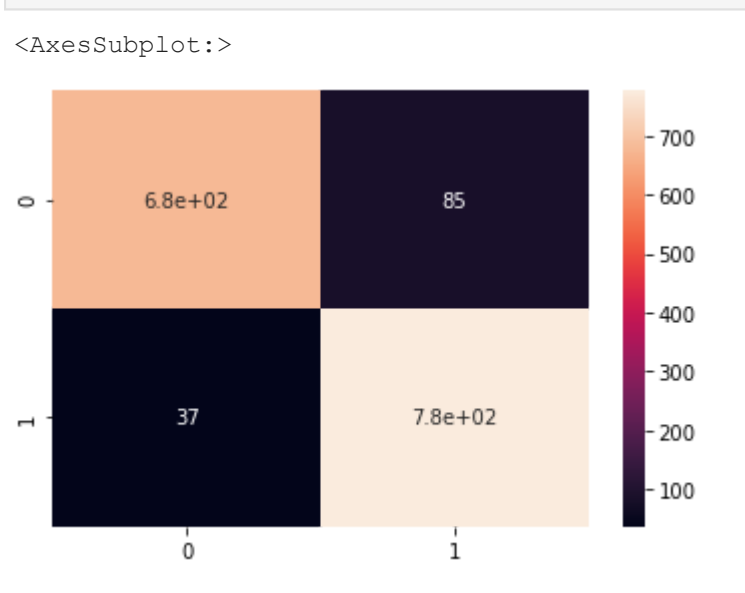
In [137…
```python
accuracy_score(y_test,lg_predict)*100
```

Out[137… 98.03424223208624

2)K-Nearset Neighbours

In [146…
```python
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier(n_neighbors=5)
kn.fit(x_train,y_train)
kn_predict = kn.predict(x_test)
kn_cm = confusion_matrix(y_test,kn_predict)
sns.heatmap(kn_cm,annot=True)
```

Out[146… <AxesSubplot:>



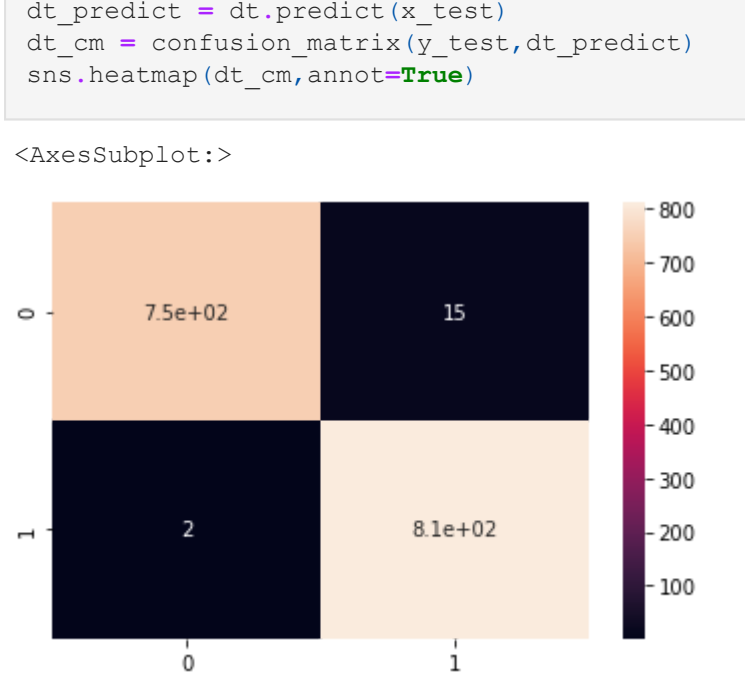
In [147…
```python
accuracy_score(y_test,kn_predict)*100
```

Out[147… 92.26379201014585

3)Decision Tree

In [148…
```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt_predict = dt.predict(x_test)
dt_cm = confusion_matrix(y_test,dt_predict)
sns.heatmap(dt_cm,annot=True)
```

Out[148… <AxesSubplot:>



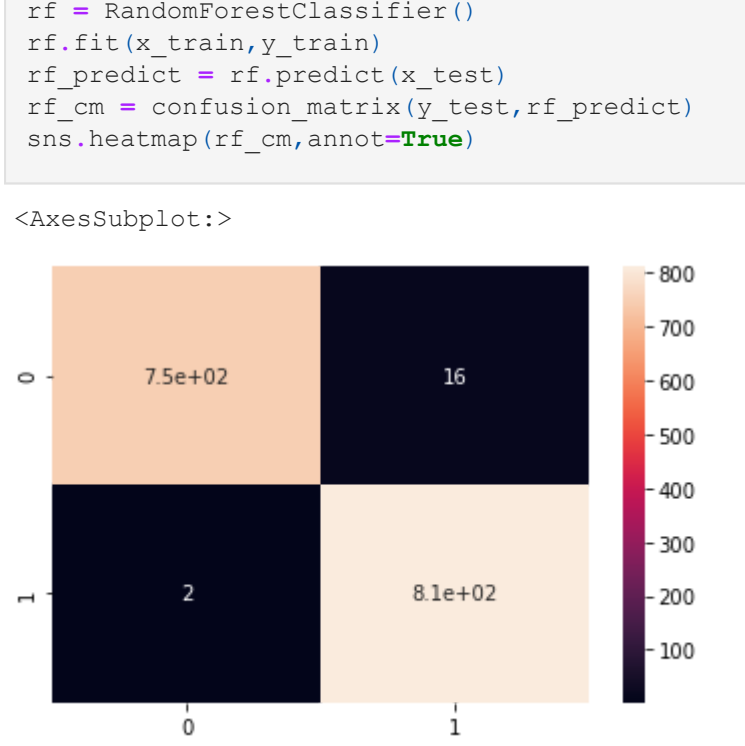
In [149…
```python
accuracy_score(y_test,dt_predict)*100
```

Out[149… 98.92200380469245

4)Random Forest

In [150…
```python
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf.fit(x_train,y_train)
rf_predict = rf.predict(x_test)
rf_cm = confusion_matrix(y_test,rf_predict)
sns.heatmap(rf_cm,annot=True)
```

Out[150… <AxesSubplot:>



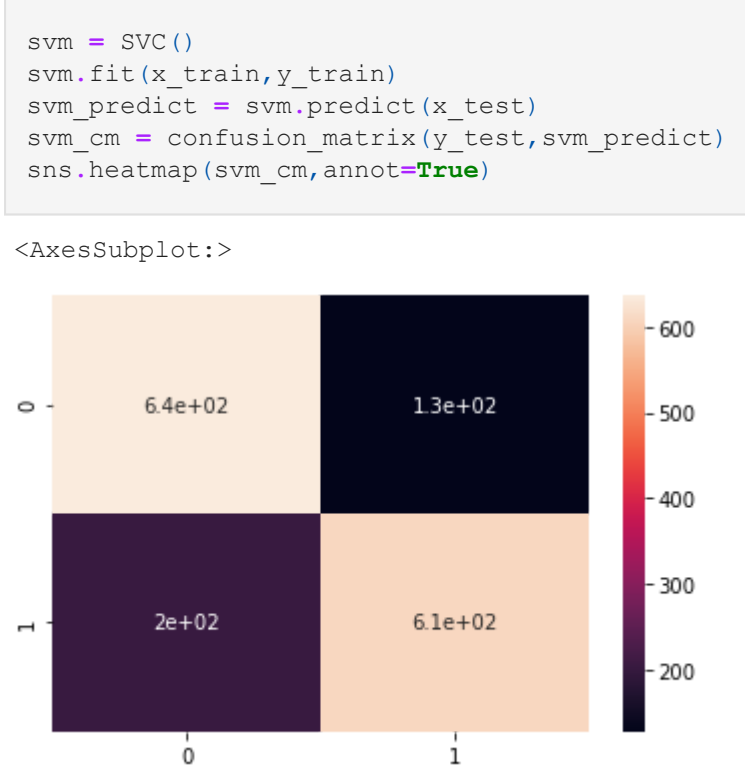
In [151…
```python
accuracy_score(y_test,rf_predict)*100
```

Out[151… 98.858592263792

Support vector machine

In [152…
```python
from sklearn.svm import SVC

svm = SVC()
svm.fit(x_train,y_train)
svm_predict = svm.predict(x_test)
svm_cm = confusion_matrix(y_test,svm_predict)
sns.heatmap(svm_cm,annot=True)
```

Out[152… <AxesSubplot:>



In [153…
```python
accuracy_score(y_test,svm_predict)*100
```

Out[153… 78.94736842105263

Decison Tree has the heighest accuracy of 98.92%

In [ ]: