Task 5

Credit_Card_Fraud_Detection

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Importing the dataset

```
dataset = pd.read_csv('Credit_card.csv')
```

```
dataset.head(3)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.3
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.6

3 rows × 31 columns

```
dataset.shape
```

```
(284807, 31)
```

```
dataset.isnull().sum()
```

	0
Time	0
V1	0
V2	0
V3	0
V4	0
V5	0
V6	0
V7	0
V8	0
V9	0
V10	0
V11	0
V12	0
V13	0
V14	0
V15	0
V16	0
V17	0
V18	0
V19	0
V20	0
V21	0
V22	0
V23	0
V24	0
V25	0
V26	0
V27	0
V28	0
Amount	0
Class	0

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
dataset['Amount']=sc.fit_transform(pd.DataFrame(dataset['Amount']))
dataset.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0

5 rows × 31 columns

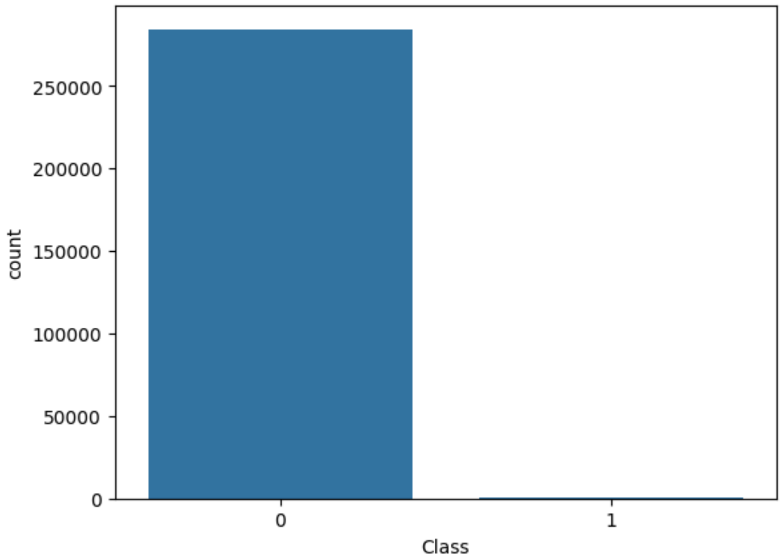
```
data=dataset.duplicated().any()
```

```
data.shape
```

```
()
```

```
import seaborn as sns
sns.countplot(x='Class',data=dataset)
```

```
<Axes: xlabel='Class', ylabel='count'>
```




```
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

```
print(X.shape)
```

```
(284807, 30)
```

```
print(y.shape)
```

```
(284807,)
```

Splitting the dataset into the Training set 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 = 0.20, random_state = 0)
```

```
print(X_train)
```

```
[[ 5.31500000e+04 -1.11504743e+00  1.03558276e+00 ...  2.29736936e-01
  -7.05913040e-02 -3.01454177e-01]
 [ 4.00600000e+04  1.22847256e+00 -1.38826483e-01 ... -6.91781060e-02
   1.89006100e-03 -2.50998329e-01]
 [ 1.54865000e+05  1.92614789e+00 -1.98627962e-01 ...  5.27473010e-02
  -3.26215510e-02 -3.25642598e-01]
 ...
 [ 7.66160000e+04  1.51260229e+00 -9.49435315e-01 ... -1.41537100e-03
   3.66494400e-03 -2.13696185e-01]
 [ 9.72530000e+04  1.79886333e+00 -1.69979073e+00 ... -3.36155800e-02
  -3.24705130e-02  3.31682753e-01]
 [ 7.48870000e+04 -5.89399721e-01  7.47828393e-01 ...  8.65917860e-02
   1.18083774e-01 -2.89299995e-01]]
```

```
print(y_train)
```

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

```
print(X_test)
```

```
[[ 1.25821000e+05 -3.23333572e-01  1.05745525e+00 ...  1.08494430e-01
   1.61139167e-01 -1.93305945e-01]
 [ 1.57235000e+05 -3.49718405e-01  9.32618570e-01 ...  7.68300270e-02
   1.75561960e-01 -3.45313182e-01]
```

```
 [ 1.52471000e+05 -1.61471082e+00 -2.40656973e+00 ...  2.86285101e-01
   4.37321868e-01  3.05868817e-02]
 ...
 [ 1.37149000e+05  1.12540205e+00 -2.28899827e+00 ... -2.40867448e-01
  -7.49435200e-03  1.97849446e+00]
 [ 1.60893000e+05  2.06485724e+00  2.85198054e-01 ... -1.03162057e-01
  -6.17434560e-02 -3.41435039e-01]
 [ 1.53086000e+05  2.35138178e+00 -1.32522606e+00 ...  7.26919800e-03
  -5.16343390e-02 -2.93258100e-01]]
```

```
print(y_test)
```

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

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[-0.87750986 -0.57177938  0.62828369 ...  0.57377804 -0.2196612
  -0.30511629]
 [-1.15308534  0.62679248 -0.08362249 ... -0.17091545  0.00631403
  -0.25423683]
 [ 1.26383162  0.98361214 -0.11987309 ...  0.13283995 -0.10128286
  -0.32950779]
 ...
 [-0.38349503  0.77210803 -0.57499928 ... -0.00209668  0.01184758
  -0.21662151]
 [ 0.05096267  0.91851362 -1.02985151 ... -0.08231776 -0.10081197
   0.33333622]
 [-0.41989457 -0.30294167  0.45385207 ...  0.21715744  0.36857129
  -0.29286006]]
```

```
print(X_test)
```

```
[[ 0.65238668 -0.16686457  0.64154241 ...  0.27172397  0.50280513
  -0.19606009]
 [ 1.31372573 -0.18035885  0.56586861 ...  0.19283776  0.54777109
  -0.34934352]
 [ 1.21343225 -0.82732765 -1.45828848 ...  0.7146577   1.36386022
   0.02971246]
 ...
 [ 0.89086789  0.57407801 -1.38701874 ... -0.5986488  -0.02294377
   1.99397392]
 [ 1.39073528  1.05455377  0.17341371 ... -0.2555804  -0.19207626
  -0.34543281]
 [ 1.22637946  1.20109412 -0.80279701 ...  0.01953937 -0.16055905
  -0.2968514 ]]
```

Training the Logistic Regression model on the Training set

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

Predicting a new result

```
y_pred=classifier.predict(X_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

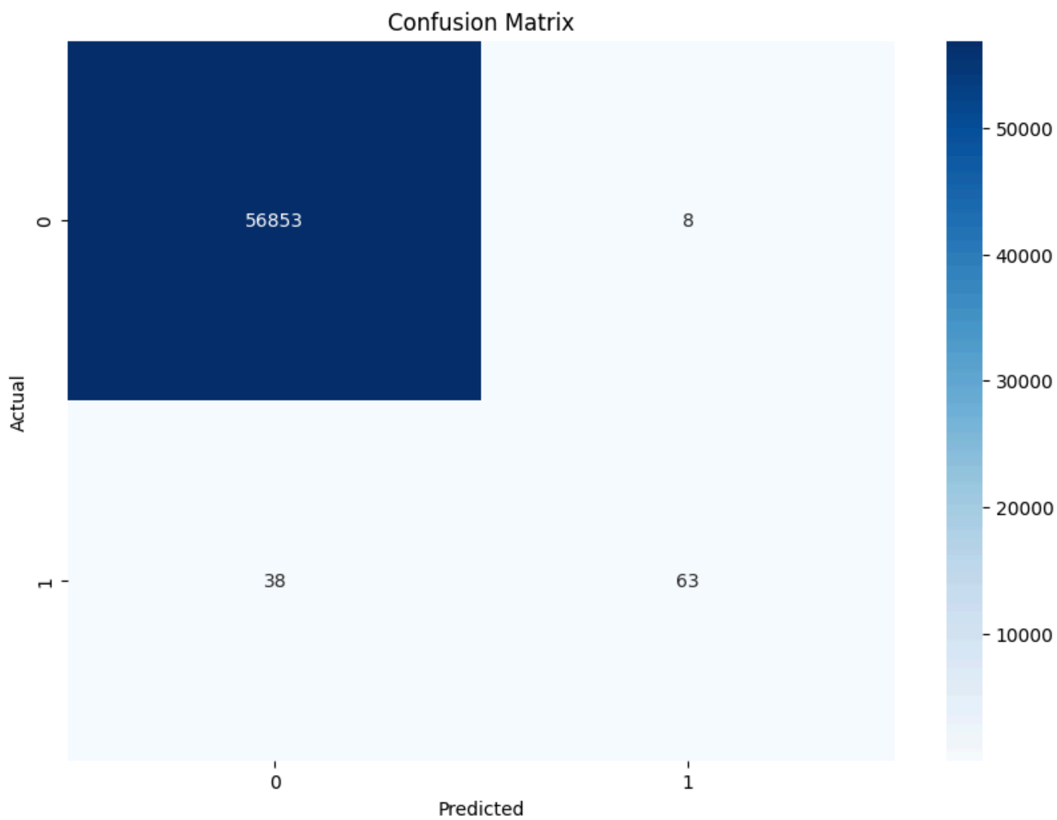
Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[56853     8]
 [   38    63]]
0.9991924440855307
```

Model has 99.91% Accuracy.

```
plt.figure(figsize=(10,7))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

```
accuracy_score(y_test,y_pred)
```

```
0.9991924440855307
```

```
precision_score(y_test,y_pred)
```

```
0.8873239436619719
```

```
recall_score(y_test,y_pred)
```

```
0.6237623762376238
```

```
f1_score(y_test,y_pred)
```

```
0.7325581395348837
```

Random Forest

Training the Random Forest Classification model on the training set

```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0)
classifier.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

```
y_pred=classifier.predict(X_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

Making the Confusion Matrix

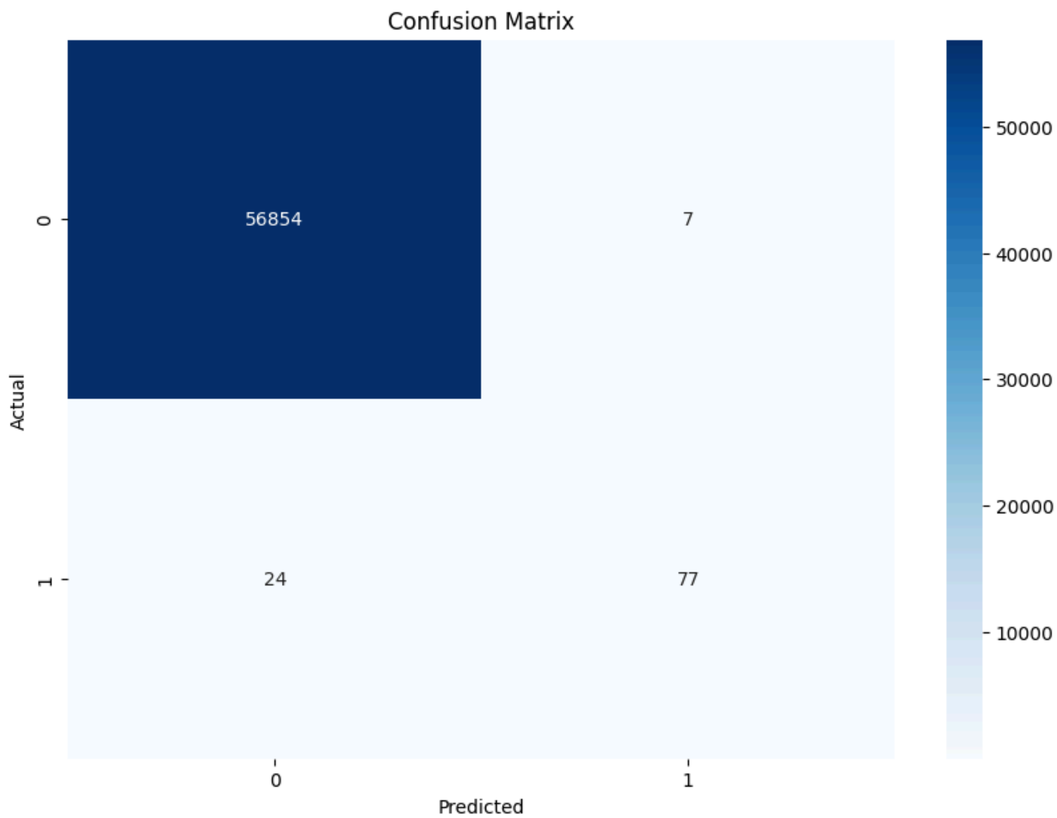
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[56854     7]
 [   24    77]]
0.9994557775359011
```

Model Has 99.94% Acuuracy

```
plt.figure(figsize=(10,7))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
```

```
plt.ylabel('Actual')
plt.show()
```




```
accuracy_score(y_test,y_pred)
```

```
0.9991924440855307
```

```
precision_score(y_test,y_pred)
```

```
0.8873239436619719
```

```
recall_score(y_test,y_pred)
```

```
0.6237623762376238
```

```
f1_score(y_test,y_pred)
```

```
0.7325581395348837
```

Decision Tree

Training The Decision Tree classification Model on the Training set

```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion='entropy',random_state=0)
classifier.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Predicting a new result

```
y_pred=classifier.predict(X_test)
y_pred
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

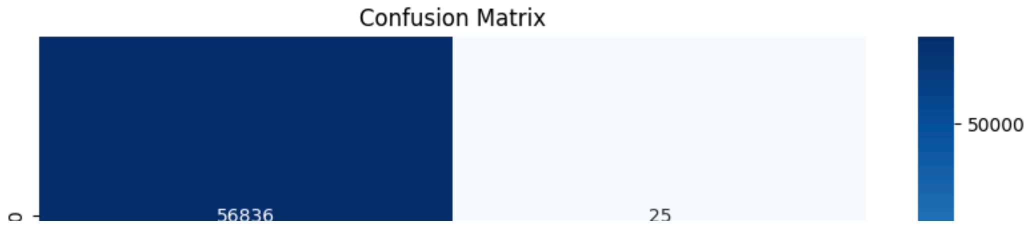
Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[56836    25]
 [   23    78]]
0.9991573329588147
```

Model has 99,91% Accuracy

```
plt.figure(figsize=(10,7))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,fmt='d',cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```


Confusion Matrix
56836
25
0
50000