In [1]:
```
import os
import pandas as pd
```

In [2]:
```
current_directory = os.getcwd()
print("Current Directory:", current_directory)
```

```
Current Directory: /Users/amarachiordor/Documents/Portfolio projects
```

In [3]:
```
files = os.listdir()
print("Files in Directory:", files)
```

```
Files in Directory: ['Credic card fraud detection main.ipynb', 'creditcar
d.csv', 'Creditcard fraud detection.ipynb', '.ipynb_checkpoints']
```

In [4]:
```
#Load CSV File
df = pd.read_csv(r"/Users/amarachiordor/Documents/Portfolio projects/cred
```

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

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.2395
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.0788
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.7914
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.2376
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.5929

5 rows × 31 columns

In [6]:
```
pd.options.display.max_columns = None
```

In [7]:
```
df.shape
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [8]:
```
df.shape
```

Out[8]: (284807, 31)

In [9]:
```
#Checking for missing values
df.isnull().sum()
```

Out[9]:
```
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
dtype: int64
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
```

In [11]:
```
sc = StandardScaler()
df['Amount'] = sc.fit_transform(pd.DataFrame(df['Amount']))
```

In [12]:
```
df = df.drop(['Time'],axis = 1)
```

In [13]:
```
# Check for Duplicates
df.duplicated().sum()
```

Out[13]:
```
9144
```

In [14]:
```
df = df.drop_duplicates()
```

In [15]:
```
df.shape
```

Out[15]:
```
(275663, 30)
```

In [16]:
```
df['Class'].value_counts()
```

Out[16]:
```
Class
0    275190
1       473
Name: count, dtype: int64
```

In [17]:
```
X = df.drop ('Class',axis = 1)
y = df['Class']
```

In [18]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2
```

In [20]:
```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score, re
```

In [21]:
```
classifier = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree Classifier": DecisionTreeClassifier ()
}

for name, clf in classifier.items():
    print (f"\n==========={name}===========")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(f"\n Accuracy: {accuracy_score(y_test, y_pred)} ")
    print(f"\n Precision: {precision_score(y_test, y_pred)} ")
    print(f"\n Recall: {recall_score(y_test, y_pred)} ")
    print(f"\n F1 score: {f1_score(y_test, y_pred)} ")
```

```
===========Logistic Regression===========

 Accuracy: 0.9992563437505668

 Precision: 0.890625

 Recall: 0.6263736263736264

 F1 score: 0.7354838709677419

===========Decision Tree Classifier===========

 Accuracy: 0.998911722561805

 Precision: 0.6631578947368421

 Recall: 0.6923076923076923

 F1 score: 0.6774193548387096
```

In [ ]:
```
#Under sampling
```

In [45]:
```
normal = df[df['Class']==0]
fraud = df[df['Class']==1]
```

In [47]:
```
normal.shape
```

Out[47]: (275190, 30)

In [49]:
```
fraud.shape
```

Out[49]: (473, 30)

In [51]:
```
normal_sample = normal.sample(n = 473)
```

In [53]:
```
normal_sample.shape
```

Out[53]: (473, 30)

In [55]:
```
new_df = pd.concat([normal_sample,fraud], ignore_index = True)
```

In [57]:
```
new_df.head()
```

Out[57]:

	V1	V2	V3	V4	V5	V6	V7
0	-0.734589	0.746812	1.924888	-1.248523	0.453207	-0.143757	0.406034
1	0.652989	-1.138956	0.853234	0.052937	-0.473422	2.044374	-0.847534
2	-0.756901	1.523053	-2.686048	-0.738390	0.294788	-0.640744	0.139390
3	-0.804967	1.128930	-1.057995	-1.930522	1.830317	4.137497	-2.143935
4	1.493342	-0.980063	0.570087	-1.237290	-1.762276	-1.231630	-0.936145

In [59]:
```
new_df['Class'].value_counts()
```

Out[59]:
```
Class
0    473
1    473
Name: count, dtype: int64
```

In [61]:
```
X = new_df.drop ('Class',axis = 1)
y = new_df['Class']
```

In [63]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2
```

In [65]:
```
classifier = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree Classifier": DecisionTreeClassifier ()
}

for name, clf in classifier.items():
    print (f"\n==========={name}===========")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(f"\n Accuracy: {accuracy_score(y_test, y_pred)} ")
    print(f"\n Precision: {precision_score(y_test, y_pred)} ")
    print(f"\n Recall: {recall_score(y_test, y_pred)} ")
    print(f"\n F1 score: {f1_score(y_test, y_pred)} ")
```

```
==========Logistic Regression==========

 Accuracy: 0.9526315789473684

 Precision: 0.979381443298969

 Recall: 0.9313725490196079

 F1 score: 0.9547738693467337

==========Decision Tree Classifier==========

 Accuracy: 0.9210526315789473

 Precision: 0.9393939393939394

 Recall: 0.9117647058823529

 F1 score: 0.9253731343283582
```

In [67]:
```
#Oversampling
```

In [69]:
```
X = df.drop ('Class',axis = 1)
y = df['Class']
```

In [71]:
```
X.shape
```

Out[71]:
```
(275663, 29)
```

In [73]:
```
y.shape
```

Out[73]:
```
(275663,)
```

In [75]:
```
from imblearn.over_sampling import SMOTE
```

In [77]:
```
x_res, y_res = SMOTE().fit_resample(X,y)
```

In [79]:
```
y_res.value_counts()
```

Out[79]:
```
Class
0    275190
1    275190
Name: count, dtype: int64
```

In [81]:
```
X_train, X_test, y_train, y_test = train_test_split(x_res, y_res, test_si
```

In [83]:
```
classifier = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree Classifier": DecisionTreeClassifier ()
}

for name, clf in classifier.items():
    print (f"\n==========={name}===========")
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(f"\n Accuracy: {accuracy_score(y_test, y_pred)} ")
    print(f"\n Precision: {precision_score(y_test, y_pred)} ")
```

```
    print(f"\n Recall: {recall_score(y_test, y_pred)} ")
    print(f"\n F1 score: {f1_score(y_test, y_pred)} ")
```

```
==========Logistic Regression==========

 Accuracy: 0.9443657109633344

 Precision: 0.9730561523721039

 Recall: 0.9139683290002364

 F1 score: 0.9425871411696323

==========Decision Tree Classifier==========

 Accuracy: 0.9982284966750246

 Precision: 0.997458611675864

 Recall: 0.9990000545424795

 F1 score: 0.9982287380439818
```

In [89]:
```
dtc = DecisionTreeClassifier()
dtc.fit(x_res, y_res)
```

Out[89]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [90]:
```
import joblib
```

In [95]:
```
joblib.dump(dtc,"credit_card_model.pkl")
```

Out[95]:
```
['credit_card_model.pkl']
```

In [97]:
```
model = joblib.load("credit_card_model.pkl")
```

In [101…
```
pred = model.predict([[ 1.22965763450793,       0.141003507049326,      0
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarni
ng: X does not have valid feature names, but DecisionTreeClassifier was fi
tted with feature names
  warnings.warn(
```

In [103…
```
pred[0]
```

Out[103…
```
0
```

In [105…
```
if pred == 0:
    print("Normal Transaction")
else:
    print("Fraud Transaction")
```

```
Normal Transaction
```

In [ ]:
```

```