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

In [2]:
```python
data = pd.read_csv(r"C:\Users\Tanushree\Downloads\creditcard.csv")
```

In [3]:
```python
data.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
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	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	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	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	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	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	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	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
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	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

In [4]:
```python
pd.options.display.max_columns = None
```

In [5]:
```python
data.head()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.01
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.22
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.24
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.059647	-0.684093	1.965775	-1.232622	-0.208038	-0.10
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.175121	-0.451449	-0.237033	-0.038195	0.803487	0.408542	-0.00

In [6]:
```python
data.tail()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	4.356170	-1.593105	2.711941	-0.689256	4.626942	-0.924459	1.107641	1.991691	0.510632	-0.682920	1.475
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	-0.975926	-0.150189	0.915802	1.214756	-0.675143	1.164931	-0.711757	-0.025693	-1.221179	-1.545556	0.059
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	-0.484782	0.411614	0.063119	-0.183699	-0.510602	1.329284	0.140716	0.313502	0.395652	-0.577252	0.001
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	-0.399126	-1.933849	-0.962886	-1.042082	0.449624	1.962563	-0.608577	0.509928	1.113981	2.897849	0.127
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	-0.915427	-1.040458	-0.031513	-0.188093	-0.084316	0.041333	-0.302620	-0.660377	0.167430	-0.256117	0.382

In [7]:
```python
data.shape
```

Out[7]:
```
(284807, 31)
```

In [9]:
```python
print("Number of columns: {}".format(data.shape[1]))
print("Number of rows: {}".format(data.shape[0]))
```

```
Number of columns: 31
Number of rows: 284807
```

In [10]:
```python
data.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 [11]:
```python
data.isnull().sum()
```

Out[11]:
```
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 [15]:
```python
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
data['Amount'] = sc.fit_transform(pd.DataFrame(data['Amount']))
```

In [16]:
```python
data.head()
```

Out[16]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.01
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.22
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.24
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.059647	-0.684093	1.965775	-1.232622	-0.208038	-0.10
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.175121	-0.451449	-0.237033	-0.038195	0.803487	0.408542	-0.00

In [19]:
```python
data = data.drop(['Time'], axis=1)
```

In [20]:
```python
data.head()
```

Out[20]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.018307
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.225775
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.247998
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924	-0.631418	-1.059647	-0.684093	1.965775	-1.232622	-0.208038	-0.108300
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670	0.175121	-0.451449	-0.237033	-0.038195	0.803487	0.408542	-0.009431

In [21]:
```python
data.duplicated().any()
```

Out[21]:
```
True
```

In [22]:
```python
data = data.drop_duplicates()
```

In [23]:
```python
data.shape
```

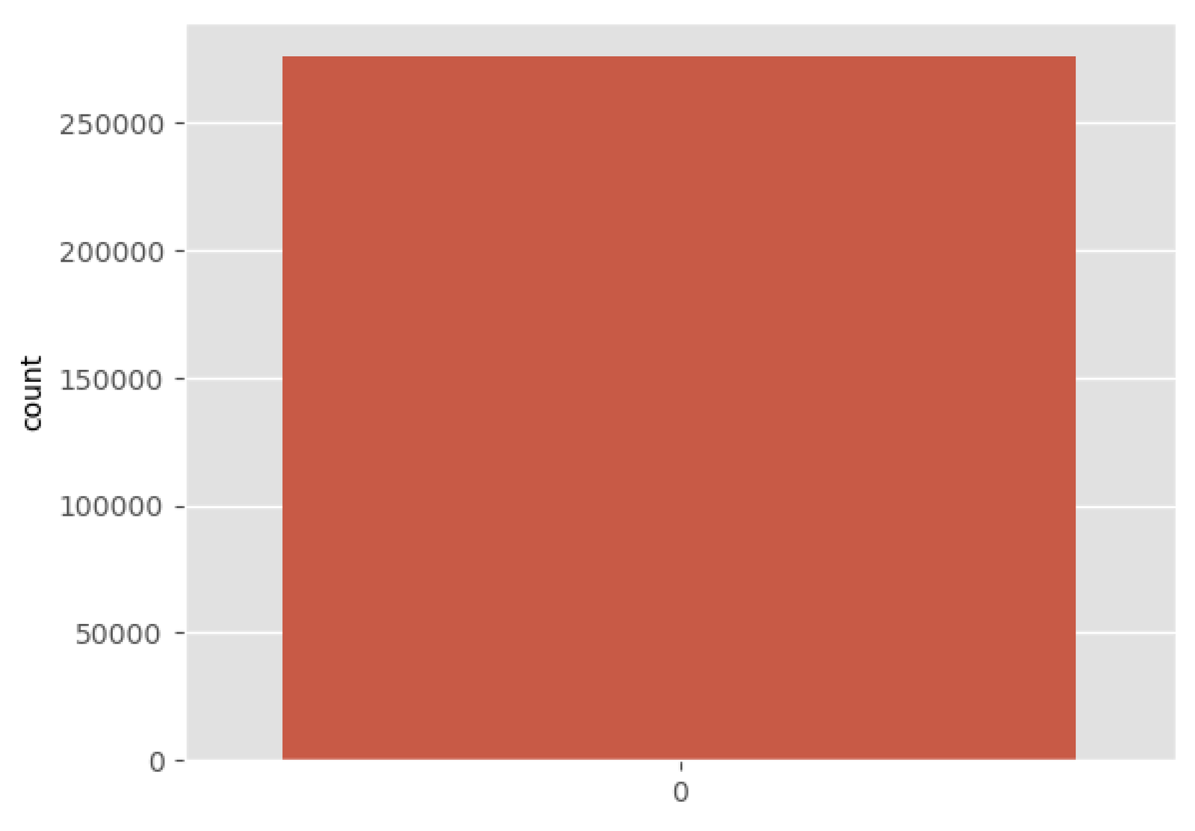
Out[23]:
```
(275663, 30)
```

In [24]:
```python
data['Class'].value_counts()
```

Out[24]:
```
0    275190
1       473
Name: Class, dtype: int64
```

In [26]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [31]:
```python
sns.countplot(data['Class'])
plt.show()
```



In [32]:
```python
x = data.drop('Class', axis = 1)
y = data['Class']
```

In [33]:
```python
from sklearn.model_selection import train_test_split
```

In [34]:
```python
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 42)
```

In [35]:
```python
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, recall_score
```

In [38]:
```python
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)
    accuracy = accuracy_score(y_test, y_pred)
    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: 0.9992200678359603

 Precision: 0.8870967741935484

 Recall: 0.6043956043956044

 F1 score: 0.718954248366013

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

 Accuracy: 0.9988754466471986

 Precision: 0.6435643564356436

 Recall: 0.7142857142857143

 F1 score: 0.6770833333333333
```

In [39]:
```python
#undersampling
```

In [40]:
```python
normal = data[data['Class']==0]
fraud = data[data['Class']==1]
```

In [41]:
```python
normal.shape
```

Out[41]:
```
(275190, 30)
```

In [42]:
```python
fraud.shape
```

Out[42]:
```
(473, 30)
```

In [44]:
```python
normal_sample = normal.sample(n=473)
```

In [45]:
```python
normal_sample.shape
```

Out[45]:
```
(473, 30)
```

In [46]:
```python
new_data = pd.concat([normal_sample, fraud], ignore_index = True)
```

In [47]:
```python
new_data.head()
```

Out[47]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
0	-0.463816	1.393423	2.264570	3.267127	0.265986	0.238790	0.696579	0.118520	-1.505531	0.580911	-1.143055	-0.215209	0.089091	-0.284575	-1.237366	0.525596	-0.383866	-0.571810	-0.951236	-0.099975	-0.205537
1	-0.413186	0.973255	1.131028	-0.172396	0.371011	-0.002820	0.511332	0.259824	-0.562848	-0.376948	1.329781	0.319399	-0.427136	-0.124229	0.524385	0.406829	-0.055192	0.078339	-0.034299	0.084074	-0.208460
2	1.133821	0.155211	0.221398	1.121062	-0.444742	-0.542450	-0.202544	0.063763	0.685889	-0.795513	-0.005848	-0.038721	-1.063191	-1.354665	0.082064	-0.245952	1.578950	-0.420881	-0.493410	-0.168445	-0.140677
3	-0.463737	0.988105	1.770064	-0.141778	0.142103	-0.887991	1.027717	-0.455633	0.191460	0.444588	-0.305467	-0.564806	-0.674099	-0.214968	0.996190	0.041384	-0.571072	-0.345014	-0.003106	0.319162	-0.370588
4	-0.550215	0.704531	-1.332205	-0.367306	2.019752	-1.404882	1.179933	-0.512857	-0.377800	-0.578117	-0.591074	0.060641	0.492745	-1.001815	-0.539646	-0.287412	0.642796	-0.274977	-0.206949	-0.304651	0.092434

In [48]:
```python
new_data['Class'].value_counts()
```

Out[48]:
```
0    473
1    473
Name: Class, dtype: int64
```

In [49]:
```python
x = new_data.drop('Class', axis = 1)
y = new_data['Class']
```

In [50]:
```python
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 42)
```

In [51]:
```python
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)
    accuracy = accuracy_score(y_test, y_pred)
    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: 0.9421052631578948

 Precision: 0.9789473684210527

 Recall: 0.9117647058823529

 F1 score: 0.9441624365482234

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

 Accuracy: 0.9210526315789473

 Precision: 0.9065420560747663

 Recall: 0.9509803921568627

 F1 score: 0.9282296650717703
```

In [52]:
```python
#oversampling
```

In [53]:
```python
x = data.drop('Class', axis = 1)
y = data['Class']
```

In [54]:
```python
x.shape
```

Out[54]:
```
(275663, 29)
```

In [55]:
```python
y.shape
```

Out[55]:
```
(275663,)
```

In [57]:
```python
from imblearn.over_sampling import SMOTE
```

In [58]:
```python
x_res, y_res = SMOTE().fit_resample(x,y)
```

In [59]:
```python
y_res.value_counts()
```

Out[59]:
```
0    275190
1    275190
Name: Class, dtype: int64
```

In [60]:
```python
x_train, x_test, y_train, y_test = train_test_split(x_res, y_res, test_size = 0.2, random_state = 42)
```

In [61]:
```python
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)
    accuracy = accuracy_score(y_test, y_pred)
    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: 0.9444747265525637

 Precision: 0.9732638956110972

 Recall: 0.9139865098267367

 F1 score: 0.9426942694269427

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

 Accuracy: 0.9982375813074603

 Precision: 0.9976393252101908

 Recall: 0.9988364271039761

 F1 score: 0.998237517261429
```

In [62]:
```python
dtc = DecisionTreeClassifier()
dtc.fit(x_res, y_res)
```

Out[62]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [63]:
```python
import joblib
```

In [64]:
```python
joblib.dump(dtc, "credit_card_model.pkl")
```

Out[64]:
```
['credit_card_model.pkl']
```

In [65]:
```python
model = joblib.load("credit_card_model.pkl")
```

In [66]:
```python
pred = model.predict([[-1.359807,-0.072781,2.536347,1.378155,-0.338321,0.462388,0.239599,0.098698,0.363787,0.090794,-0.551600,-0.617801,-0.991390,-0.311169,1.468177,-0.470401,0.20
```

```
C:\Users\Tanushree\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [67]:
```python
pred[0]
```

Out[67]:
```
0
```

In [68]:
```python
if pred[0] == 0:
    print("Normal Transaction")
else:
    print("Fraud Transaction")
```

```
Normal Transaction
```