In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
data = pd.read_csv("creditcard.csv")
```

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

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

5 rows × 31 columns

In [4]:
```
data.tail()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650

5 rows × 31 columns

In [3]:
```
data['Time_taken']=data['Time']-data['Time'].shift(1)
```

In [4]:
```
data['Time_taken'].fillna(0,inplace=True)
```

In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 32 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
 31  Time_taken  284807 non-null  float64
dtypes: float64(31), int64(1)
memory usage: 69.5 MB
```

In [5]:
```
data.drop("Time",axis=1,inplace=True)
```

In [17]:
```
data.head()
```

Out[17]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.09
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.16
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.20
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.05
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.75

5 rows × 31 columns

In [6]:
```
X=data.loc[:,data.columns !='Class']
```

In [7]:
```
Y=data.loc[:,'Class']
```

Decision Tree

In [8]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,stratify=Y,random_state=42)
```

In [25]:
```
y_train.value_counts()
```

Out[25]:
```
0    213236
1       369
Name: Class, dtype: int64
```

In [26]:
```
y_test.value_counts()
```

Out[26]:
```
0    71079
1      123
Name: Class, dtype: int64
```

In [28]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
```

In [29]:
```
from sklearn import metrics
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy :  0.9990168815482711
```

In [30]:
```
from sklearn.model_selection import GridSearchCV
gd = GridSearchCV(clf,{'max_depth':[3,4,5,6,7,8,9],'criterion':['gini','entropy']},cv=4
gd=gd.fit(x_train,y_train)
gd.best_params_
```

Out[30]:
```
{'criterion': 'gini', 'max_depth': 5}
```

In [31]:
```
gd.best_score_
```

Out[31]:
```
0.9994428964318148
```

In [12]:
```
from sklearn.metrics import classification_report
```

In [36]:
```
print(classification_report(y_test,clf.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.72      0.72      0.72       123

    accuracy                           1.00     71202
   macro avg       0.86      0.86      0.86     71202
weighted avg       1.00      1.00      1.00     71202
```

In [31]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,clf.predict(x_test))
```

Out[31]:
```
array([[71077,     2],
       [   33,    90]], dtype=int64)
```

KNN

In [9]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
classifier.fit(x_train,y_train)
y_pred =classifier.predict(x_test)
```

In [14]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.93      0.67      0.78       123

    accuracy                           1.00     71202
   macro avg       0.97      0.84      0.89     71202
weighted avg       1.00      1.00      1.00     71202
```

In [15]:
```
confusion_matrix(y_test,y_pred)
```

Out[15]:
```
array([[71073,     6],
       [   40,    83]], dtype=int64)
```

Naive Bayes

In [16]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

In [17]:
```
model.fit(x_train,y_train)
```

Out[17]:
```
GaussianNB()
```

In [18]:
```
model.score(x_test,y_test)
```

Out[18]:
```
0.9763489789612652
```

In [19]:
```
y_pred = model.predict(x_test)
```

In [21]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support
```

```
              0       1.00      0.98      0.99     71079
              1       0.06      0.82      0.11       123

    accuracy                           0.98     71202
   macro avg       0.53      0.90      0.55     71202
weighted avg       1.00      0.98      0.99     71202
```

In [22]:
```
confusion_matrix(y_test,y_pred)
```

Out[22]:
```
array([[69417,  1662],
       [   22,   101]], dtype=int64)
```

Random Forest

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=20)
clf = clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
```

In [24]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
           1       0.98      0.73      0.84       123

    accuracy                           1.00     71202
   macro avg       0.99      0.87      0.92     71202
weighted avg       1.00      1.00      1.00     71202
```

In [25]:
```
confusion_matrix(y_test,y_pred)
```

Out[25]:
```
array([[71077,     2],
       [   33,    90]], dtype=int64)
```

SVM

In [26]:
```
from sklearn.svm import SVC
model=SVC(kernel='poly')
model.fit(x_train,y_train)
```

Out[26]:
```
SVC(kernel='poly')
```

In [27]:
```
y_pred = model.predict(x_test)
```

In [29]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71079
```

```
           1       0.86      0.05      0.09       123

    accuracy                           1.00     71202
   macro avg       0.93      0.52      0.55     71202
weighted avg       1.00      1.00      1.00     71202
```

In [30]:
```
confusion_matrix(y_test,y_pred)
```

Out[30]:
```
array([[71078,     1],
       [  117,     6]], dtype=int64)
```

In [ ]: