In [1]:

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df = pd.read_csv('../datasets2/BankNote_Authentication.csv')
```

In [3]:

```
df
```

Out[3]:

	variance	skewness	curtosis	entropy	class
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['variance', 'skewness', 'curtosis', 'entropy', 'class'], dtype='object')
```

In [5]:

```
from sklearn.model_selection import train_test_split
```

In [6]:

```
X = df[['variance','skewness','curtosis', 'entropy']]
y = df[['class']]
```

In [7]:

```
X_train, X_test, y_train , y_test = train_test_split(X,y,test_size=0.2,random_state=75 )
```

In [8]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[8]:

```
((1097, 4), (275, 4), (1097, 1), (275, 1))
```

In [9]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [11]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

In [12]:

```
model = KNeighborsClassifier(n_neighbors = 3)
model.fit(X_train_scaled, y_train)
```

Out[12]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [22]:

```
model.score(X_train_scaled,y_train)
```

Out[22]:

```
0.9981768459434822
```

In [23]:

```
model_1 = KNeighborsClassifier(n_neighbors = 5)
model_1.fit(X_train_scaled, y_train)
```

Out[23]:

```
KNeighborsClassifier()
```

In [24]:

```
model_4 = LogisticRegression()
model_4.fit(X_train_scaled,y_train)
```

Out[24]:

```
LogisticRegression()
```

In [25]:

```
model_1.score(X_train_scaled,y_train)
```

Out[25]:

```
0.9981768459434822
```

In [26]:

```
model_4.score(X_train_scaled,y_train)
```

Out[26]:

```
0.96718322698268
```

In [27]:

```
model_2 = KNeighborsClassifier(n_neighbors = 7)
model_2.fit(X_train_scaled, y_train)
```

Out[27]:

```
KNeighborsClassifier(n_neighbors=7)
```

In [29]:

```
model_5 = LogisticRegression()
model_5.fit(X_train_scaled,y_train)
```

Out[29]:

```
LogisticRegression()
```

In [30]:

```
model_2.score(X_train_scaled,y_train)
```

Out[30]:

```
0.9981768459434822
```

In [31]:

```
model_5.score(X_train_scaled,y_train)
```

Out[31]:

```
0.96718322698268
```

In [72]:

```
model_3 = KNeighborsClassifier(n_neighbors = 9)
model_3.fit(X_train_scaled, y_train)
```

Out[72]:

```
KNeighborsClassifier(n_neighbors=9)
```

In [32]:

```
model_6 = LogisticRegression()
model_6.fit(X_train_scaled,y_train)
```

Out[32]:

```
LogisticRegression()
```

In [73]:

```
model_3.score(X_train_scaled,y_train)
```

Out[73]:

```
0.9981768459434822
```

In [33]:

```
model_6.score(X_train_scaled,y_train)
```

Out[33]:

```
0.96718322698268
```

In [36]:

```
model.score(X_test_scaled,y_test)
```

Out[36]:

```
1.0
```

In [37]:

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

In [38]:

```
confusion_matrix(y_test, yp)
```

Out[38]:

```
array([[152,   0],
       [  0, 123]], dtype=int64)
```

In [39]:

```
accuracy_score(y_test,yp)
```

Out[39]:

```
1.0
```

In [40]:

```
recall_score(y_test,yp)
```

Out[40]:

```
1.0
```

In [41]:

```
precision_score(y_test,yp)
```

Out[41]:

```
1.0
```

In [42]:

```
f1_score(y_test,yp)
```

Out[42]:

```
1.0
```

In [43]:

```
model_L = LogisticRegression()
model_L.fit(X_train_scaled,y_train)
```

Out[43]:

```
LogisticRegression()
```

In [45]:

```
L = model_L.predict(X_test_scaled)
L
```

Out[45]:

```
array([1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0], dtype=int64)
```

In [51]:

```
accuracy_score(y_test,L)
```

Out[51]:

```
0.9709090909090909
```

In [52]:

```
recall_score(y_test,L)
```

Out[52]:

```
0.983739837398374
```

In [53]:

```
recall_score(y_test,L)
```

Out[53]:

```
0.983739837398374
```

In [54]:

```
f1_score(y_test,L)
```

Out[54]:

```
0.968
```

In [ ]:

```

```