Data Science February Major Project

Project Description :

Problem statement: Create a classification model to type of the Date Fruits based on external appearance features provided in the dataset.

Context: A great number of fruits are grown around the world, each of which has various types. The factors that determine the type of fruit are the external appearance features such as color, length, diameter, and shape. The external appearance of the fruits is a major determinant of the fruit type. Determining the variety of fruits by looking at their external appearance may necessitate expertise, which is time-consuming and requires great effort. The aim of this study is to classify the types of date fruit, that are, Barhee, Deglet Nour, Sukkary, Rotab Mozafati, Ruthana, Safawi, and Sagai by using different machine learning methods. In accordance with this purpose, 898 images of seven different date fruit types were obtained via the computer vision system (CVS). Through image processing techniques, a total of 34 features, including morphological features, shape, and color, were extracted from these images.

Data Set: https://drive.google.com/drive/folders/1pPMfMjqdb134WlLkTNjCpOtIiuMb93PG?usp=sharing (https://drive.google.com/drive/folders/1pPMfMjqdb134WlLkTNjCpOtIiuMb93PG?usp=sharing)

Predict Type of Date Fruits using 4 algorithms :

```
1) KNN Classification
2) Decision Tree Classification
3) Random Forest Classification
4) Logistic Regression
```

KNN Classification

In [1]:

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

In [2]:

```python
df = pd.read_csv('Date_fruits_dataset.csv') ## import the dataset
df.head()
```

Out[2]:

	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDITY	CC
0	422163	2378.908	837.8484	645.6693	0.6373	733.1539	0.9947	
1	338136	2085.144	723.8198	595.2073	0.5690	656.1464	0.9974	
2	526843	2647.394	940.7379	715.3638	0.6494	819.0222	0.9962	
3	416063	2351.210	827.9804	645.2988	0.6266	727.8378	0.9948	
4	347562	2160.354	763.9877	582.8359	0.6465	665.2291	0.9908	

5 rows × 35 columns

In [3]:

```python
df['Class'].value_counts()
```

Out[3]:

```
DOKOL     204
SAFAVI    199
ROTANA    166
DEGLET     98
SOGAY      94
IRAQI      72
BERHI      65
Name: Class, dtype: int64
```

Check if any null values

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
AREA             0
PERIMETER        0
MAJOR_AXIS       0
MINOR_AXIS       0
ECCENTRICITY     0
EQDIASQ          0
SOLIDITY         0
CONVEX_AREA      0
EXTENT           0
ASPECT_RATIO     0
ROUNDNESS        0
COMPACTNESS      0
SHAPEFACTOR_1    0
SHAPEFACTOR_2    0
SHAPEFACTOR_3    0
SHAPEFACTOR_4    0
MeanRR           0
MeanRG           0
MeanRB           0
StdDevRR         0
StdDevRG         0
StdDevRB         0
SkewRR           0
SkewRG           0
SkewRB           0
KurtosisRR       0
KurtosisRG       0
KurtosisRB       0
EntropyRR        0
EntropyRG        0
EntropyRB        0
ALLdaub4RR       0
ALLdaub4RG       0
ALLdaub4RB       0
Class            0
dtype: int64
```

In [5]:

```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

```
(898, 34)
(898,)
```

Split into training and test data

In [6]:

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

In [7]:

```
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25)
```

In [8]:

```
print(x_tr.shape)
print(x_te.shape)
print(y_tr.shape)
print(y_te.shape)
```

```
(673, 34)
(225, 34)
(673,)
(225,)
```

Build the model

In [13]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [14]:

```
m1 = KNeighborsClassifier(n_neighbors=19)
m1.fit(x_tr,y_tr)
```

Out[14]:

```
KNeighborsClassifier(n_neighbors=19)
```

In [15]:

```
print('Training score',m1.score(x_tr,y_tr))
print('Testing score',m1.score(x_te,y_te))
```

```
Training score 0.7087667161961367
Testing score 0.6888888888888889
```

Prediction of type of Date Fruits

In [18]:

```
ypred1 = m1.predict(x_te)
print(ypred1)
```

```
['ROTANA' 'SAFAVI' 'ROTANA' 'ROTANA' 'ROTANA' 'SAFAVI' 'DOKOL' 'DOKOL'
 'DOKOL' 'SOGAY' 'SAFAVI' 'ROTANA' 'DOKOL' 'DOKOL' 'SOGAY' 'DEGLET'
 'SAFAVI' 'DEGLET' 'ROTANA' 'SAFAVI' 'SAFAVI' 'DOKOL' 'SAFAVI' 'DEGLET'
 'DEGLET' 'SOGAY' 'SAFAVI' 'DOKOL' 'DOKOL' 'SAFAVI' 'ROTANA' 'SAFAVI'
 'ROTANA' 'DOKOL' 'ROTANA' 'SAFAVI' 'DOKOL' 'DEGLET' 'ROTANA' 'DOKOL'
 'DOKOL' 'DOKOL' 'SAFAVI' 'DOKOL' 'DOKOL' 'DOKOL' 'DOKOL' 'DOKOL' 'DOKOL'
 'DOKOL' 'IRAQI' 'ROTANA' 'ROTANA' 'SAFAVI' 'SAFAVI' 'DOKOL' 'IRAQI'
 'ROTANA' 'SOGAY' 'SAFAVI' 'ROTANA' 'SOGAY' 'ROTANA' 'SAFAVI' 'DEGLET'
 'ROTANA' 'IRAQI' 'DOKOL' 'DOKOL' 'DOKOL' 'DEGLET' 'DOKOL' 'DOKOL'
 'DEGLET' 'SAFAVI' 'DOKOL' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA'
 'DOKOL' 'DOKOL' 'ROTANA' 'DOKOL' 'ROTANA' 'SAFAVI' 'SOGAY' 'SAFAVI'
 'SAFAVI' 'ROTANA' 'SAFAVI' 'DOKOL' 'DOKOL' 'DOKOL' 'SAFAVI' 'SOGAY'
 'IRAQI' 'DEGLET' 'ROTANA' 'DOKOL' 'DOKOL' 'ROTANA' 'SAFAVI' 'DEGLET'
 'SOGAY' 'ROTANA' 'SOGAY' 'SAFAVI' 'DOKOL' 'IRAQI' 'SAFAVI' 'ROTANA'
 'DOKOL' 'ROTANA' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'ROTANA'
 'ROTANA' 'DEGLET' 'DOKOL' 'SAFAVI' 'SAFAVI' 'ROTANA' 'DEGLET' 'DEGLET'
 'DOKOL' 'ROTANA' 'SAFAVI' 'SAFAVI' 'DOKOL' 'DOKOL' 'ROTANA' 'ROTANA'
 'SAFAVI' 'ROTANA' 'DOKOL' 'DOKOL' 'ROTANA' 'SAFAVI' 'ROTANA' 'DOKOL'
 'DOKOL' 'ROTANA' 'SAFAVI' 'ROTANA' 'DEGLET' 'DOKOL' 'DOKOL' 'SAFAVI'
 'DOKOL' 'DOKOL' 'DEGLET' 'ROTANA' 'ROTANA' 'DOKOL' 'DOKOL' 'ROTANA'
 'SAFAVI' 'SAFAVI' 'DEGLET' 'DOKOL' 'DOKOL' 'DOKOL' 'DOKOL' 'DOKOL'
 'SAFAVI' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'BERHI' 'DOKOL' 'SOGAY'
 'ROTANA' 'ROTANA' 'ROTANA' 'DEGLET' 'SOGAY' 'SAFAVI' 'SAFAVI' 'IRAQI'
 'SAFAVI' 'DOKOL' 'ROTANA' 'DOKOL' 'SAFAVI' 'DOKOL' 'DOKOL' 'DOKOL'
 'SAFAVI' 'ROTANA' 'SOGAY' 'SAFAVI' 'SAFAVI' 'IRAQI' 'DOKOL' 'IRAQI'
 'IRAQI' 'SAFAVI' 'SAFAVI' 'DEGLET' 'IRAQI' 'SAFAVI' 'SOGAY' 'DEGLET'
 'DOKOL' 'DOKOL' 'DOKOL' 'ROTANA' 'SAFAVI' 'SAFAVI' 'ROTANA' 'SAFAVI'
 'DEGLET' 'ROTANA' 'IRAQI' 'SOGAY' 'ROTANA' 'DOKOL' 'SAFAVI' 'DOKOL']
```

Confusion matrix and Classification report

In [19]:

```
from sklearn.metrics import confusion_matrix,classification_report
```

In [20]:

```
cm1 = confusion_matrix(y_te,ypred1)
print(cm1)
```

```
[[ 0  2  6  4  8  0  1]
 [ 0  7  9  1  4  0  3]
 [ 0  2 42  0  0  3  0]
 [ 0  1  2  6  1  2  0]
 [ 0  2  0  0 40  0  0]
 [ 0  0  3  0  0 50  0]
 [ 1  6  6  0  3  0 10]]
```

In [21]:

```
print(classification_report(y_te,ypred1))
```

```
              precision    recall  f1-score   support

       BERHI       0.00      0.00      0.00        21
      DEGLET       0.35      0.29      0.32        24
       DOKOL       0.62      0.89      0.73        47
       IRAQI       0.55      0.50      0.52        12
      ROTANA       0.71      0.95      0.82        42
      SAFAVI       0.91      0.94      0.93        53
       SOGAY       0.71      0.38      0.50        26

    accuracy                           0.69       225
   macro avg       0.55      0.57      0.54       225
weighted avg       0.63      0.69      0.64       225
```

Accuracy using KNN Classification Model : 0.69

Decision Tree Classification

In [2]:

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

In [3]:

```
df = pd.read_csv('Date_fruits_dataset.csv')
df.head()
```

Out[3]:

	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDITY	CC
0	422163	2378.908	837.8484	645.6693	0.6373	733.1539	0.9947	
1	338136	2085.144	723.8198	595.2073	0.5690	656.1464	0.9974	
2	526843	2647.394	940.7379	715.3638	0.6494	819.0222	0.9962	
3	416063	2351.210	827.9804	645.2988	0.6266	727.8378	0.9948	
4	347562	2160.354	763.9877	582.8359	0.6465	665.2291	0.9908	

5 rows × 35 columns

Check for null values

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
AREA             0
PERIMETER        0
MAJOR_AXIS       0
MINOR_AXIS       0
ECCENTRICITY     0
EQDIASQ          0
SOLIDITY         0
CONVEX_AREA      0
EXTENT           0
ASPECT_RATIO     0
ROUNDNESS        0
COMPACTNESS      0
SHAPEFACTOR_1    0
SHAPEFACTOR_2    0
SHAPEFACTOR_3    0
SHAPEFACTOR_4    0
MeanRR           0
MeanRG           0
MeanRB           0
StdDevRR         0
StdDevRG         0
StdDevRB         0
SkewRR           0
SkewRG           0
SkewRB           0
KurtosisRR       0
KurtosisRG       0
KurtosisRB       0
EntropyRR        0
EntropyRG        0
EntropyRB        0
ALLdaub4RR       0
ALLdaub4RG       0
ALLdaub4RB       0
Class            0
dtype: int64
```

In [5]:

```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

```
(898, 34)
(898,)
```

Split into training and test data

In [7]:

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

In [8]:

```
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25)
print(x_te.shape)
print(x_tr.shape)
print(y_te.shape)
print(y_tr.shape)
```

```
(225, 34)
(673, 34)
(225,)
(673,)
```

Build the model

In [9]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [10]:

```
clf = DecisionTreeClassifier()
```

In [12]:

```
clf = clf.fit(x_tr,y_tr)
```

Prediction

In [13]:

```
y_pred = clf.predict(x_te)
```

In [16]:

```
print(y_pred)
```

```
['ROTANA' 'DEGLET' 'SAFAVI' 'DEGLET' 'DOKOL' 'DOKOL' 'SOGAY' 'DOKOL'
 'DOKOL' 'DEGLET' 'DOKOL' 'SOGAY' 'SAFAVI' 'SAFAVI' 'DEGLET' 'DEGLET'
 'DOKOL' 'BERHI' 'ROTANA' 'ROTANA' 'DOKOL' 'ROTANA' 'SAFAVI' 'DOKOL'
 'ROTANA' 'DOKOL' 'DEGLET' 'DOKOL' 'SOGAY' 'SAFAVI' 'DOKOL' 'SOGAY'
 'SOGAY' 'BERHI' 'DOKOL' 'DEGLET' 'BERHI' 'IRAQI' 'ROTANA' 'ROTANA'
 'SAFAVI' 'SAFAVI' 'DOKOL' 'DEGLET' 'DEGLET' 'IRAQI' 'SOGAY' 'ROTANA'
 'IRAQI' 'BERHI' 'SAFAVI' 'ROTANA' 'DOKOL' 'DEGLET' 'SAFAVI' 'ROTANA'
 'DOKOL' 'BERHI' 'SAFAVI' 'DEGLET' 'ROTANA' 'DEGLET' 'ROTANA' 'SOGAY'
 'DOKOL' 'SAFAVI' 'SAFAVI' 'DOKOL' 'BERHI' 'DOKOL' 'DOKOL' 'IRAQI' 'DOKOL'
 'DOKOL' 'ROTANA' 'SAFAVI' 'ROTANA' 'SOGAY' 'DOKOL' 'BERHI' 'DEGLET'
 'ROTANA' 'SAFAVI' 'BERHI' 'SOGAY' 'DEGLET' 'DOKOL' 'ROTANA' 'IRAQI'
 'DEGLET' 'SOGAY' 'DOKOL' 'DOKOL' 'SOGAY' 'ROTANA' 'DOKOL' 'SAFAVI'
 'IRAQI' 'SAFAVI' 'SAFAVI' 'DOKOL' 'DOKOL' 'DEGLET' 'ROTANA' 'DOKOL'
 'SAFAVI' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'ROTANA' 'DEGLET' 'IRAQI'
 'DOKOL' 'DOKOL' 'DEGLET' 'ROTANA' 'SOGAY' 'DEGLET' 'SOGAY' 'IRAQI'
 'DEGLET' 'SAFAVI' 'BERHI' 'ROTANA' 'SAFAVI' 'DEGLET' 'DEGLET' 'DOKOL'
 'IRAQI' 'SAFAVI' 'DEGLET' 'SOGAY' 'SAFAVI' 'SOGAY' 'SAFAVI' 'DOKOL'
 'SOGAY' 'DOKOL' 'DEGLET' 'IRAQI' 'SAFAVI' 'IRAQI' 'BERHI' 'BERHI' 'DOKOL'
 'DEGLET' 'SAFAVI' 'IRAQI' 'DOKOL' 'DOKOL' 'SOGAY' 'DOKOL' 'SOGAY' 'DOKOL'
 'DEGLET' 'SAFAVI' 'DEGLET' 'BERHI' 'DEGLET' 'DOKOL' 'DEGLET' 'SOGAY'
 'IRAQI' 'ROTANA' 'SAFAVI' 'ROTANA' 'ROTANA' 'SAFAVI' 'SAFAVI' 'DOKOL'
 'DOKOL' 'DEGLET' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'SAFAVI' 'DOKOL' 'DOKOL'
 'DOKOL' 'SAFAVI' 'DOKOL' 'ROTANA' 'ROTANA' 'ROTANA' 'SAFAVI' 'DEGLET'
 'DEGLET' 'DOKOL' 'DEGLET' 'ROTANA' 'ROTANA' 'SOGAY' 'SOGAY' 'ROTANA'
 'ROTANA' 'ROTANA' 'SAFAVI' 'ROTANA' 'SOGAY' 'BERHI' 'ROTANA' 'DOKOL'
 'SAFAVI' 'ROTANA' 'DOKOL' 'BERHI' 'DOKOL' 'DOKOL' 'ROTANA' 'BERHI'
 'BERHI' 'DOKOL' 'DOKOL' 'ROTANA' 'ROTANA' 'SOGAY' 'BERHI' 'DOKOL' 'DOKOL'
 'ROTANA' 'SAFAVI' 'DOKOL' 'DOKOL' 'SAFAVI']
```

Confusion matrix and Classification report

In [17]:

```
from sklearn.metrics import confusion_matrix,classification_report
```

In [19]:

```
cm=confusion_matrix(y_te,y_pred)
print(cm)
```

```
[[10  0  0  2  0  0  0]
 [ 0 13  4  0  1  2  5]
 [ 0  3 48  0  0  0  2]
 [ 6  0  0 11  0  0  1]
 [ 0  2  0  0 37  0  0]
 [ 0  7  0  0  0 41  1]
 [ 1  8  5  0  1  0 14]]
```

In [20]:

```
print(classification_report(y_te,y_pred))
```

```
              precision    recall  f1-score   support

       BERHI       0.59      0.83      0.69        12
      DEGLET       0.39      0.52      0.45        25
       DOKOL       0.84      0.91      0.87        53
       IRAQI       0.85      0.61      0.71        18
      ROTANA       0.95      0.95      0.95        39
      SAFAVI       0.95      0.84      0.89        49
       SOGAY       0.61      0.48      0.54        29

    accuracy                           0.77       225
   macro avg       0.74      0.73      0.73       225
weighted avg       0.79      0.77      0.78       225
```

Accuracy of Decision Tree Classification = 0.77

Random Forest Classification

In [1]:

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

In [2]:

```
df = pd.read_csv('Date_fruits_dataset.csv')
df.head()
```

Out[2]:

```
     AREA  PERIMETER  MAJOR_AXIS  MINOR_AXIS  ECCENTRICITY   EQDIASQ  SOLIDITY  CC
0  422163   2378.908    837.8484    645.6693        0.6373  733.1539    0.9947
1  338136   2085.144    723.8198    595.2073        0.5690  656.1464    0.9974
2  526843   2647.394    940.7379    715.3638        0.6494  819.0222    0.9962
3  416063   2351.210    827.9804    645.2988        0.6266  727.8378    0.9948
4  347562   2160.354    763.9877    582.8359        0.6465  665.2291    0.9908

5 rows × 35 columns
```

Check for null values

In [3]:

```
df.isnull().sum()
```

Out[3]:

```
AREA             0
PERIMETER        0
MAJOR_AXIS       0
MINOR_AXIS       0
ECCENTRICITY     0
EQDIASQ          0
SOLIDITY         0
CONVEX_AREA      0
EXTENT           0
ASPECT_RATIO     0
ROUNDNESS        0
COMPACTNESS      0
SHAPEFACTOR_1    0
SHAPEFACTOR_2    0
SHAPEFACTOR_3    0
SHAPEFACTOR_4    0
MeanRR           0
MeanRG           0
MeanRB           0
StdDevRR         0
StdDevRG         0
StdDevRB         0
SkewRR           0
SkewRG           0
SkewRB           0
KurtosisRR       0
KurtosisRG       0
KurtosisRB       0
EntropyRR        0
EntropyRG        0
EntropyRB        0
ALLdaub4RR       0
ALLdaub4RG       0
ALLdaub4RB       0
Class            0
dtype: int64
```

In [4]:

```
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
print(x.shape)
print(y.shape)
```

```
(898, 34)
(898,)
```

Split into training and test data

In [5]:

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

In [6]:

```
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25)
print(x_te.shape)
print(x_tr.shape)
print(y_te.shape)
print(y_tr.shape)
```

```
(225, 34)
(673, 34)
(225,)
(673,)
```

Build the model

In [7]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [8]:

```
clf=RandomForestClassifier(n_estimators=100)
```

In [10]:

```
clf.fit(x_tr,y_tr)
```

Out[10]:

```
RandomForestClassifier()
```

Prediction

In [11]:

```
ypred=clf.predict(x_te)
```

In [12]:

```
print(ypred)
```

```
['ROTANA' 'SAFAVI' 'IRAQI' 'SAFAVI' 'DOKOL' 'SAFAVI' 'IRAQI' 'SAFAVI'
 'IRAQI' 'IRAQI' 'DOKOL' 'IRAQI' 'DEGLET' 'SAFAVI' 'DOKOL' 'DOKOL' 'SOGAY'
 'SAFAVI' 'ROTANA' 'DOKOL' 'DOKOL' 'SAFAVI' 'SAFAVI' 'DOKOL' 'SOGAY'
 'SAFAVI' 'DEGLET' 'DOKOL' 'SAFAVI' 'DEGLET' 'DEGLET' 'SAFAVI' 'SAFAVI'
 'IRAQI' 'BERHI' 'DOKOL' 'IRAQI' 'ROTANA' 'ROTANA' 'ROTANA' 'DOKOL'
 'ROTANA' 'DOKOL' 'ROTANA' 'ROTANA' 'DOKOL' 'BERHI' 'SAFAVI' 'IRAQI'
 'DOKOL' 'SAFAVI' 'SAFAVI' 'ROTANA' 'DOKOL' 'DOKOL' 'IRAQI' 'SAFAVI'
 'DEGLET' 'DOKOL' 'SOGAY' 'IRAQI' 'IRAQI' 'DOKOL' 'ROTANA' 'SAFAVI'
 'SAFAVI' 'DOKOL' 'DEGLET' 'DOKOL' 'SAFAVI' 'DEGLET' 'DEGLET' 'BERHI'
 'IRAQI' 'DEGLET' 'SOGAY' 'DEGLET' 'ROTANA' 'IRAQI' 'ROTANA' 'DOKOL'
 'BERHI' 'DOKOL' 'DOKOL' 'SAFAVI' 'DEGLET' 'ROTANA' 'SAFAVI' 'SOGAY'
 'ROTANA' 'BERHI' 'IRAQI' 'ROTANA' 'DEGLET' 'SAFAVI' 'DOKOL' 'SOGAY'
 'ROTANA' 'SOGAY' 'ROTANA' 'ROTANA' 'SAFAVI' 'ROTANA' 'DOKOL' 'DOKOL'
 'DEGLET' 'ROTANA' 'SAFAVI' 'SOGAY' 'BERHI' 'ROTANA' 'SAFAVI' 'DOKOL'
 'SOGAY' 'DOKOL' 'ROTANA' 'DOKOL' 'ROTANA' 'SAFAVI' 'SAFAVI' 'IRAQI'
 'IRAQI' 'DOKOL' 'IRAQI' 'SAFAVI' 'SAFAVI' 'BERHI' 'SAFAVI' 'SOGAY'
 'SOGAY' 'DOKOL' 'SAFAVI' 'SAFAVI' 'ROTANA' 'BERHI' 'DOKOL' 'DOKOL'
 'DOKOL' 'DOKOL' 'SAFAVI' 'DOKOL' 'DOKOL' 'DOKOL' 'SAFAVI' 'SAFAVI'
 'IRAQI' 'DOKOL' 'IRAQI' 'DOKOL' 'DOKOL' 'SAFAVI' 'ROTANA' 'SAFAVI'
 'BERHI' 'DOKOL' 'ROTANA' 'SOGAY' 'BERHI' 'BERHI' 'BERHI' 'SAFAVI'
 'SAFAVI' 'DOKOL' 'DOKOL' 'SOGAY' 'DOKOL' 'SAFAVI' 'DOKOL' 'DOKOL'
 'SAFAVI' 'SOGAY' 'IRAQI' 'SOGAY' 'DOKOL' 'DOKOL' 'ROTANA' 'DEGLET'
 'ROTANA' 'SAFAVI' 'SOGAY' 'BERHI' 'IRAQI' 'DOKOL' 'SAFAVI' 'SOGAY'
 'ROTANA' 'DEGLET' 'SAFAVI' 'DOKOL' 'ROTANA' 'SOGAY' 'IRAQI' 'ROTANA'
 'SAFAVI' 'SAFAVI' 'ROTANA' 'IRAQI' 'DOKOL' 'SAFAVI' 'SAFAVI' 'ROTANA'
 'DOKOL' 'DOKOL' 'ROTANA' 'SAFAVI' 'SOGAY' 'DOKOL' 'IRAQI' 'IRAQI' 'SOGAY'
 'DOKOL' 'ROTANA' 'IRAQI' 'DOKOL' 'SAFAVI' 'SAFAVI' 'DOKOL' 'SOGAY'
 'DOKOL' 'DOKOL' 'SAFAVI' 'DOKOL' 'SAFAVI' 'SAFAVI' 'SAFAVI']
```

Confusion matrix and Classification report

In [14]:

```
from sklearn.metrics import confusion_matrix,classification_report
```

In [15]:

```
cm=confusion_matrix(y_te,ypred)
print(cm)
```

```
[[12  0  0  2  0  0  1]
 [ 0  9  4  0  0  0  2]
 [ 0  0 55  0  0  0  0]
 [ 1  0  0 24  0  0  0]
 [ 0  1  0  0 34  0  1]
 [ 0  0  1  0  0 55  0]
 [ 0  5  0  0  1  0 17]]
```

```
In [16]:    print(classification_report(y_te,ypred))
```

```
              precision    recall  f1-score   support

       BERHI       0.92      0.80      0.86        15
      DEGLET       0.60      0.60      0.60        15
       DOKOL       0.92      1.00      0.96        55
       IRAQI       0.92      0.96      0.94        25
      ROTANA       0.97      0.94      0.96        36
      SAFAVI       1.00      0.98      0.99        56
       SOGAY       0.81      0.74      0.77        23

    accuracy                           0.92       225
   macro avg       0.88      0.86      0.87       225
weighted avg       0.92      0.92      0.91       225
```

Accuracy of Random Forest Classification = 0.92

Logistic Regressioin

In [1]:

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

In [2]:

```
df = pd.read_csv('Date_fruits_dataset.csv')
df.head()
```

Out[2]:

	AREA	PERIMETER	MAJOR_AXIS	MINOR_AXIS	ECCENTRICITY	EQDIASQ	SOLIDITY	CC
0	422163	2378.908	837.8484	645.6693	0.6373	733.1539	0.9947	
1	338136	2085.144	723.8198	595.2073	0.5690	656.1464	0.9974	
2	526843	2647.394	940.7379	715.3638	0.6494	819.0222	0.9962	
3	416063	2351.210	827.9804	645.2988	0.6266	727.8378	0.9948	
4	347562	2160.354	763.9877	582.8359	0.6465	665.2291	0.9908	

5 rows × 35 columns

Check for null values

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
AREA             0
PERIMETER        0
MAJOR_AXIS       0
MINOR_AXIS       0
ECCENTRICITY     0
EQDIASQ          0
SOLIDITY         0
CONVEX_AREA      0
EXTENT           0
ASPECT_RATIO     0
ROUNDNESS        0
COMPACTNESS      0
SHAPEFACTOR_1    0
SHAPEFACTOR_2    0
SHAPEFACTOR_3    0
SHAPEFACTOR_4    0
MeanRR           0
MeanRG           0
MeanRB           0
StdDevRR         0
StdDevRG         0
StdDevRB         0
SkewRR           0
SkewRG           0
SkewRB           0
KurtosisRR       0
KurtosisRG       0
KurtosisRB       0
EntropyRR        0
EntropyRG        0
EntropyRB        0
ALLdaub4RR       0
ALLdaub4RG       0
ALLdaub4RB       0
Class            0
dtype: int64
```

In [5]:

```
df.shape
```

Out[5]:

```
(898, 35)
```

```
In [6]:  df.dtypes
```

```
Out[6]: AREA                   int64
        PERIMETER            float64
        MAJOR_AXIS           float64
        MINOR_AXIS           float64
        ECCENTRICITY         float64
        EQDIASQ              float64
        SOLIDITY             float64
        CONVEX_AREA            int64
        EXTENT               float64
        ASPECT_RATIO         float64
        ROUNDNESS            float64
        COMPACTNESS          float64
        SHAPEFACTOR_1        float64
        SHAPEFACTOR_2        float64
        SHAPEFACTOR_3        float64
        SHAPEFACTOR_4        float64
        MeanRR               float64
        MeanRG               float64
        MeanRB               float64
        StdDevRR             float64
        StdDevRG             float64
        StdDevRB             float64
        SkewRR               float64
        SkewRG               float64
        SkewRB               float64
        KurtosisRR           float64
        KurtosisRG           float64
        KurtosisRB           float64
        EntropyRR              int64
        EntropyRG              int64
        EntropyRB              int64
        ALLdaub4RR           float64
        ALLdaub4RG           float64
        ALLdaub4RB           float64
        Class                 object
        dtype: object
```

```
In [7]:  x = df.iloc[:,:-1]
         y = df.iloc[:,-1]
         print(x.shape)
         print(y.shape)
```

```
(898, 34)
(898,)
```

Split data into test and training

```
In [8]:  from sklearn.model_selection import train_test_split
```

In [9]:

```
x_tr,x_te,y_tr,y_te=train_test_split(x,y,test_size=0.25)
print(x_te.shape)
print(x_tr.shape)
print(y_te.shape)
print(y_tr.shape)
```

```
(225, 34)
(673, 34)
(225,)
(673,)
```

Build the Model

In [10]:

```
from sklearn.linear_model import LogisticRegression
```

In [11]:

```
reg=LogisticRegression()
reg.fit(x_tr,y_tr)
```

```
D:\newfolder\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sci
kit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  n_iter_i = _check_optimize_result(
```

Out[11]:

```
LogisticRegression()
```

In [12]:

```
print("Training Score",reg.score(x_tr,y_tr))
print("Testing Score",reg.score(x_te,y_te))
```

```
Training Score 0.5676077265973254
Testing Score 0.5111111111111111
```

Prediction of type of Date Fruits

In [13]:

```
ypred=reg.predict(x_te)
print(ypred)
```

```
['DOKOL' 'ROTANA' 'SAFAVI' 'IRAQI' 'SOGAY' 'IRAQI' 'ROTANA' 'DOKOL'
 'ROTANA' 'ROTANA' 'ROTANA' 'IRAQI' 'ROTANA' 'IRAQI' 'SAFAVI' 'ROTANA'
 'SAFAVI' 'ROTANA' 'ROTANA' 'SAFAVI' 'ROTANA' 'SAFAVI' 'ROTANA' 'ROTANA'
 'DOKOL' 'ROTANA' 'ROTANA' 'DOKOL' 'SAFAVI' 'ROTANA' 'DOKOL' 'ROTANA'
 'DOKOL' 'SAFAVI' 'SOGAY' 'SOGAY' 'ROTANA' 'ROTANA' 'IRAQI' 'SAFAVI'
 'ROTANA' 'DOKOL' 'ROTANA' 'ROTANA' 'SOGAY' 'ROTANA' 'ROTANA' 'IRAQI'
 'ROTANA' 'IRAQI' 'ROTANA' 'SAFAVI' 'SAFAVI' 'ROTANA' 'SAFAVI' 'DOKOL'
 'IRAQI' 'SAFAVI' 'ROTANA' 'ROTANA' 'SAFAVI' 'IRAQI' 'ROTANA' 'SAFAVI'
 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'SAFAVI' 'DOKOL' 'ROTANA' 'DOKOL'
 'DOKOL' 'SAFAVI' 'SOGAY' 'ROTANA' 'ROTANA' 'DOKOL' 'SAFAVI' 'ROTANA'
 'ROTANA' 'ROTANA' 'ROTANA' 'DOKOL' 'DOKOL' 'SOGAY' 'DOKOL' 'ROTANA'
 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'SAFAVI' 'SAFAVI' 'IRAQI' 'SAFAVI'
 'ROTANA' 'ROTANA' 'ROTANA' 'SAFAVI' 'SAFAVI' 'ROTANA' 'ROTANA' 'ROTANA'
 'ROTANA' 'DOKOL' 'ROTANA' 'DOKOL' 'SOGAY' 'SAFAVI' 'SAFAVI' 'SOGAY'
 'SAFAVI' 'ROTANA' 'IRAQI' 'SAFAVI' 'ROTANA' 'SAFAVI' 'SAFAVI' 'SOGAY'
 'SOGAY' 'ROTANA' 'ROTANA' 'DOKOL' 'SOGAY' 'DOKOL' 'SAFAVI' 'DOKOL'
 'ROTANA' 'SAFAVI' 'SAFAVI' 'ROTANA' 'IRAQI' 'SAFAVI' 'ROTANA' 'ROTANA'
 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA' 'ROTANA'
 'IRAQI' 'IRAQI' 'SAFAVI' 'ROTANA' 'ROTANA' 'SOGAY' 'DOKOL' 'ROTANA'
 'ROTANA' 'ROTANA' 'ROTANA' 'SOGAY' 'ROTANA' 'DOKOL' 'SOGAY' 'ROTANA'
 'ROTANA' 'SAFAVI' 'ROTANA' 'IRAQI' 'SAFAVI' 'ROTANA' 'SAFAVI' 'SAFAVI'
 'ROTANA' 'SAFAVI' 'SAFAVI' 'ROTANA' 'SOGAY' 'ROTANA' 'ROTANA' 'SOGAY'
 'ROTANA' 'ROTANA' 'IRAQI' 'DOKOL' 'IRAQI' 'ROTANA' 'ROTANA' 'SOGAY'
 'ROTANA' 'ROTANA' 'SAFAVI' 'IRAQI' 'ROTANA' 'SOGAY' 'ROTANA' 'ROTANA'
 'SAFAVI' 'ROTANA' 'ROTANA' 'ROTANA' 'SOGAY' 'IRAQI' 'SAFAVI' 'IRAQI'
 'ROTANA' 'SAFAVI' 'ROTANA' 'SAFAVI' 'SAFAVI' 'ROTANA' 'ROTANA' 'IRAQI'
 'IRAQI' 'ROTANA' 'SAFAVI' 'SOGAY' 'ROTANA' 'ROTANA' 'DOKOL' 'SAFAVI'
 'ROTANA' 'IRAQI' 'SAFAVI' 'SAFAVI' 'SOGAY' 'ROTANA' 'ROTANA' 'SAFAVI'
 'SAFAVI']
```

In [14]:

```
x_te['Actual_Values']=y_te
x_te['Predicted_Values']=ypred
print(x_te)
```

```
       AREA  PERIMETER  MAJOR_AXIS  MINOR_AXIS  ECCENTRICITY   EQDIASQ  
\
267  151963  1477.1790    551.4623    352.0080        0.7698  439.8696   
109  233279  1812.4000    649.0758    459.3955        0.7064  544.9955   
624  336865  2580.1411    882.9647    488.4337        0.8331  654.9121   
247   86213  1096.8230    384.0147    286.8239        0.6649  331.3152   
828  289876  2087.5200    818.5771    453.4831        0.8325  607.5209   
..      ...        ...         ...         ...           ...       ...   
810  243353  1853.1870    664.4248    469.8410        0.7071  556.6387   
211  207029  1765.3530    704.8253    375.3293        0.8464  513.4175   
453  428711  2570.7129    815.3513    679.2316        0.5532  738.8178   
731  382402  2575.4390    914.6583    536.5853        0.8098  697.7746   
739  325380  2254.8821    863.1747    483.1501        0.8287  643.6511   

     SOLIDITY  CONVEX_AREA  EXTENT  ASPECT_RATIO  ...  KurtosisRG  Kurto
sisRB  \
267    0.9918       153220  0.6908        1.5666  ...      5.0023   
3.4537   
109    0.9865       236470  0.7785        1.4129  ...      1.9759   
2.4983   
624    0.9601       350873  0.7419        1.8077  ...      6.4382   
3.3465   
247    0.9863        87412  0.7599        1.3389  ...      2.6035   
2.2506   
828    0.9878       293458  0.6413        1.8051  ...      4.3011   
2.9179   
..        ...          ...     ...           ...  ...         ...   
...   
810    0.9893       245985  0.7161        1.4141  ...      2.5866   
2.3923   
211    0.9929       208510  0.8048        1.8779  ...      4.7953   
3.1464   
453    0.9745       439942  0.7628        1.2004  ...      2.2800   
2.5466   
731    0.9525       401472  0.7579        1.7046  ...     11.1113   
6.1582   
739    0.9708       335152  0.7249        1.7866  ...     10.1266   
6.7424   

       EntropyRR    EntropyRG    EntropyRB  ALLdaub4RR  ALLdaub4RG  \
267 -33599162368 -33943552000 -25104467968     72.7737     74.0258   
109 -32555407360 -28394031104 -27923378176     56.9895     52.6629   
624 -11477377024 -12436102144 -16732371968     27.8189     30.0402   
247 -14963954688 -15412930560 -14767569920     64.0398     66.0221   
828 -30283675648 -19470006272 -24404664320     51.1169     41.6076   
..           ...          ...          ...         ...         ...   
810 -33449897984 -23056840704 -21356206080     57.1072     48.3514   
211 -26686932992 -22885431296 -28511535104     56.6093     53.3635   
453 -75249565696 -69343453184 -56885456896     64.5790     61.7914   
731  -5097521152  -7803381248  -8578378240     17.9978     22.5063   
739  -4815613440  -7964525056  -9242575872     19.3270     25.6448   
```

```
     ALLdaub4RB  Actual_Values  Predicted_Values
267     63.9547          DOKOL             DOKOL
109     53.9634         DEGLET            ROTANA
624     35.6642         SAFAVI            SAFAVI
247     64.0946          DOKOL             IRAQI
828     46.1993          SOGAY             SOGAY
..          ...            ...               ...
810     45.8750          SOGAY             SOGAY
211     58.8649          DOKOL            ROTANA
453     56.6031         ROTANA            ROTANA
731     24.4442         SAFAVI            SAFAVI
739     27.9815         SAFAVI            SAFAVI

[225 rows x 36 columns]

<ipython-input-14-d7f5e6760c50>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  x_te['Actual_Values']=y_te
<ipython-input-14-d7f5e6760c50>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  x_te['Predicted_Values']=ypred
```

Confusion matrix and cassification report

In [15]:

```
from sklearn.metrics import confusion_matrix,classification_report
```

In [16]:

```
cm=confusion_matrix(y_te,ypred)
print(cm)
```

```
[[ 0  0  2  4 11  0  2]
 [ 0  0  2  0 20  0  4]
 [ 0  0 16  2 33  0  3]
 [ 0  0  2 10  3  3  0]
 [ 0  0  2  0 29  0  0]
 [ 0  0  0  7  1 48  0]
 [ 0  0  0  0  9  0 12]]
```

In [17]:

```
print(classification_report(y_te,ypred))
```

```
              precision    recall  f1-score   support

       BERHI       0.00      0.00      0.00        19
      DEGLET       0.00      0.00      0.00        26
       DOKOL       0.67      0.30      0.41        54
       IRAQI       0.43      0.56      0.49        18
      ROTANA       0.27      0.94      0.42        31
      SAFAVI       0.94      0.86      0.90        56
       SOGAY       0.57      0.57      0.57        21

    accuracy                           0.51       225
   macro avg       0.41      0.46      0.40       225
weighted avg       0.52      0.51      0.47       225


D:\newfolder\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` pa
rameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Accuracy of Logistic regresison model : 0.51

PROJECT REPORT

Accuracy using KNN Classification: 0.69

Accuracy using Decision Tree Classification: 0.77

Accuracy using Random Forest Classification: 0.92

Accuracy using Logistic Regression: 0.51

THEREFORE THE BEST ACCURACY IS SHOWN BY RANDOM FOREST CLASSIFICATION IN THIS CASE.