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

```
df= pd.read_csv('Almond.csv')
df
```

	Unnamed: 0	Length (major axis)	Width (minor axis)	Thickness (depth)	Area	Perimeter	Roundness	Solidity	Compactness	Aspect Ratio	Eccentricity	Extent	Convex hull(convex area)	Type
0	0	NaN	227.940628	127.759132	22619.0	643.813269	NaN	0.973384	1.458265	NaN	NaN	0.681193	23237.5	MAMRA
1	1	NaN	234.188126	128.199509	23038.0	680.984841	NaN	0.957304	1.601844	NaN	NaN	0.656353	24065.5	MAMRA
2	2	NaN	229.418610	125.796547	22386.5	646.943212	NaN	0.967270	1.487772	NaN	NaN	0.683620	23144.0	MAMRA
3	3	NaN	232.763153	125.918808	22578.5	661.227483	NaN	0.965512	1.540979	NaN	NaN	0.685360	23385.0	MAMRA
4	4	NaN	230.150742	107.253448	19068.0	624.842706	NaN	0.951450	1.629395	NaN	NaN	0.714800	20041.0	MAMRA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2798	2798	NaN	192.709366	122.356506	18471.5	653.345233	NaN	0.931000	1.838965	NaN	NaN	0.725739	19840.5	SANORA
2799	2799	NaN	186.254745	118.708961	17213.5	581.688379	NaN	0.952706	1.564234	NaN	NaN	0.714016	18068.0	SANORA
2800	2800	NaN	186.196182	119.147224	17510.5	608.315795	NaN	0.948821	1.681705	NaN	NaN	0.718999	18455.0	SANORA
2801	2801	NaN	188.660828	120.634438	17941.0	630.759446	NaN	0.944810	1.764701	NaN	NaN	0.738191	18989.0	SANORA
2802	2802	269.356903	176.023636	NaN	36683.5	887.310743	0.643761	0.947380	1.707933	1.530231	0.75693	0.722429	38721.0	SANORA

2803 rows × 14 columns

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

```
Unnamed: 0                   0
Length (major axis)        857
Width (minor axis)         942
Thickness (depth)         1004
Area                         0
Perimeter                    0
Roundness                  857
Solidity                     0
Compactness                  0
Aspect Ratio              1799
Eccentricity              1799
Extent                       0
Convex hull(convex area)     0
Type                         0
dtype: int64
```

```
df
```

	Unnamed: 0	Length (major axis)	Width (minor axis)	Thickness (depth)	Area	Perimeter	Roundness	Solidity	Compactness	Aspect Ratio	Eccentricity	Extent	Convex hull(convex area)	Type
0	0	NaN	227.940628	127.759132	22619.0	643.813269	NaN	0.973384	1.458265	NaN	NaN	0.681193	23237.5	MAMRA
1	1	NaN	234.188126	128.199509	23038.0	680.984841	NaN	0.957304	1.601844	NaN	NaN	0.656353	24065.5	MAMRA
2	2	NaN	229.418610	125.796547	22386.5	646.943212	NaN	0.967270	1.487772	NaN	NaN	0.683620	23144.0	MAMRA
3	3	NaN	232.763153	125.918808	22578.5	661.227483	NaN	0.965512	1.540979	NaN	NaN	0.685360	23385.0	MAMRA
4	4	NaN	230.150742	107.253448	19068.0	624.842706	NaN	0.951450	1.629395	NaN	NaN	0.714800	20041.0	MAMRA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2798	2798	NaN	192.709366	122.356506	18471.5	653.345233	NaN	0.931000	1.838965	NaN	NaN	0.725739	19840.5	SANORA
2799	2799	NaN	186.254745	118.708961	17213.5	581.688379	NaN	0.952706	1.564234	NaN	NaN	0.714016	18068.0	SANORA
2800	2800	NaN	186.196182	119.147224	17510.5	608.315795	NaN	0.948821	1.681705	NaN	NaN	0.718999	18455.0	SANORA
2801	2801	NaN	188.660828	120.634438	17941.0	630.759446	NaN	0.944810	1.764701	NaN	NaN	0.738191	18989.0	SANORA
2802	2802	269.356903	176.023636	NaN	36683.5	887.310743	0.643761	0.947380	1.707933	1.530231	0.75693	0.722429	38721.0	SANORA

2803 rows × 14 columns

```
# df.fillna(method= 'ffill', inplace=True)
df.fillna(method= 'bfill', inplace=True)
df
```

	Unnamed: 0	Length (major axis)	Width (minor axis)	Thickness (depth)	Area	Perimeter	Roundness	Solidity	Compactness	Aspect Ratio	Eccentricity	Extent	Convex hull(convex area)	Type
0	0	413.477173	227.940628	127.759132	22619.0	643.813269	0.309009	0.973384	1.458265	1.866195	0.844313	0.681193	23237.5	MAMRA
1	1	413.477173	234.188126	128.199509	23038.0	680.984841	0.309009	0.957304	1.601844	1.866195	0.844313	0.656353	24065.5	MAMRA
2	2	413.477173	229.418610	125.796547	22386.5	646.943212	0.309009	0.967270	1.487772	1.866195	0.844313	0.683620	23144.0	MAMRA
3	3	413.477173	232.763153	125.918808	22578.5	661.227483	0.309009	0.965512	1.540979	1.866195	0.844313	0.685360	23385.0	MAMRA
4	4	413.477173	230.150742	107.253448	19068.0	624.842706	0.309009	0.951450	1.629395	1.866195	0.844313	0.714800	20041.0	MAMRA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2798	2798	269.356903	192.709366	122.356506	18471.5	653.345233	0.643761	0.931000	1.838965	1.530231	0.756930	0.725739	19840.5	SANORA
2799	2799	269.356903	186.254745	118.708961	17213.5	581.688379	0.643761	0.952706	1.564234	1.530231	0.756930	0.714016	18068.0	SANORA
2800	2800	269.356903	186.196182	119.147224	17510.5	608.315795	0.643761	0.948821	1.681705	1.530231	0.756930	0.718999	18455.0	SANORA
2801	2801	269.356903	188.660828	120.634438	17941.0	630.759446	0.643761	0.944810	1.764701	1.530231	0.756930	0.738191	18989.0	SANORA
2802	2802	269.356903	176.023636	NaN	36683.5	887.310743	0.643761	0.947380	1.707933	1.530231	0.756930	0.722429	38721.0	SANORA

2803 rows × 14 columns

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

```
Unnamed: 0              0
Length (major axis)     0
```

```
Width (minor axis)         0
Thickness (depth)          1
Area                       0
Perimeter                  0
Roundness                  0
Solidity                   0
Compactness                0
Aspect Ratio               0
Eccentricity               0
Extent                     0
Convex hull(convex area)   0
Type                       0
dtype: int64
```

```
df['Type'].value_counts()
```

```
Type
SANORA     943
MAMRA      933
REGULAR    927
Name: count, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()

df[['Type']] = df[['Type']].apply(le.fit_transform)
df
```

	Unnamed: 0	Length (major axis)	Width (minor axis)	Thickness (depth)	Area	Perimeter	Roundness	Solidity	Compactness	Aspect Ratio	Eccentricity	Extent	Convex hull(convex area)	Type
0	0	413.477173	227.940628	127.759132	22619.0	643.813269	0.309009	0.973384	1.458265	1.866195	0.844313	0.681193	23237.5	0
1	1	413.477173	234.188126	128.199509	23038.0	680.984841	0.309009	0.957304	1.601844	1.866195	0.844313	0.656353	24065.5	0
2	2	413.477173	229.418610	125.796547	22386.5	646.943212	0.309009	0.967270	1.487772	1.866195	0.844313	0.683620	23144.0	0
3	3	413.477173	232.763153	125.918808	22578.5	661.227483	0.309009	0.965512	1.540979	1.866195	0.844313	0.685360	23385.0	0
4	4	413.477173	230.150742	107.253448	19068.0	624.842706	0.309009	0.951450	1.629395	1.866195	0.844313	0.714800	20041.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2798	2798	269.356903	192.709366	122.356506	18471.5	653.345233	0.643761	0.931000	1.838965	1.530231	0.756930	0.725739	19840.5	2
2799	2799	269.356903	186.254745	118.708961	17213.5	581.688379	0.643761	0.952706	1.564234	1.530231	0.756930	0.714016	18068.0	2
2800	2800	269.356903	186.196182	119.147224	17510.5	608.315795	0.643761	0.948821	1.681705	1.530231	0.756930	0.718999	18455.0	2
2801	2801	269.356903	188.660828	120.634438	17941.0	630.759446	0.643761	0.944810	1.764701	1.530231	0.756930	0.738191	18989.0	2
2802	2802	269.356903	176.023636	NaN	36683.5	887.310743	0.643761	0.947380	1.707933	1.530231	0.756930	0.722429	38721.0	2

2803 rows × 14 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2803 entries, 0 to 2802
Data columns (total 14 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Unnamed: 0                2803 non-null   int64  
 1   Length (major axis)       2803 non-null   float64
 2   Width (minor axis)        2803 non-null   float64
 3   Thickness (depth)         2802 non-null   float64
 4   Area                      2803 non-null   float64
 5   Perimeter                 2803 non-null   float64
 6   Roundness                 2803 non-null   float64
 7   Solidity                  2803 non-null   float64
 8   Compactness               2803 non-null   float64
 9   Aspect Ratio              2803 non-null   float64
 10  Eccentricity              2803 non-null   float64
 11  Extent                    2803 non-null   float64
 12  Convex hull(convex area)  2803 non-null   float64
 13  Type                      2803 non-null   int64  
dtypes: float64(12), int64(2)
memory usage: 306.7 KB
```

```
df.corr()
```

	Unnamed: 0	Length (major axis)	Width (minor axis)	Thickness (depth)	Area	Perimeter	Roundness	Solidity	Compactness	Aspect Ratio	Eccentricity	Extent	Convex hull(convex area)	Type
Unnamed: 0	1.000000	-0.271307	-0.227617	0.014135	-0.146780	-0.223352	0.193454	0.177853	-0.166339	-0.174979	-0.162643	0.148692	-0.162780	0.122121
Length (major axis)	-0.271307	1.000000	0.504808	0.233239	0.583955	0.589105	-0.353089	-0.120866	0.180317	0.450424	0.432509	0.024755	0.600393	-0.233531
Width (minor axis)	-0.227617	0.504808	1.000000	0.391034	0.431200	0.408250	-0.085867	-0.059936	0.106454	-0.045443	-0.048985	-0.034261	0.438680	0.110846
Thickness (depth)	0.014135	0.233239	0.391034	1.000000	0.328950	0.271804	0.220537	0.063740	-0.039865	-0.155422	-0.167686	0.104291	0.327973	0.371933
Area	-0.146780	0.583955	0.431200	0.328950	1.000000	0.793905	0.187532	0.142245	-0.011408	0.137784	0.133940	0.303895	0.996626	-0.013641
Perimeter	-0.223352	0.589105	0.408250	0.271804	0.793905	1.000000	-0.114868	-0.377505	0.561668	0.265586	0.251929	-0.104688	0.834600	-0.123801
Roundness	0.193454	-0.353089	-0.085867	0.220537	0.187532	-0.114868	1.000000	0.364748	-0.391963	-0.433389	-0.445386	0.291055	0.157489	0.326783
Solidity	0.177853	-0.120866	-0.059936	0.063740	0.142245	-0.377505	0.364748	1.000000	-0.866622	-0.276955	-0.266199	0.774073	0.067954	0.277235
Compactness	-0.166339	0.180317	0.106454	-0.039865	-0.011408	0.561668	-0.391963	-0.866622	1.000000	0.256966	0.245431	-0.615750	0.056435	-0.209883
Aspect Ratio	-0.174979	0.450424	-0.045443	-0.155422	0.137784	0.265586	-0.433389	-0.276955	0.256966	1.000000	0.965195	-0.148093	0.160826	-0.547632
Eccentricity	-0.162643	0.432509	-0.048985	-0.167686	0.133940	0.251929	-0.445386	-0.266199	0.245431	0.965195	1.000000	-0.147774	0.155461	-0.544794
Extent	0.148692	0.024755	-0.034261	0.104291	0.303895	-0.104688	0.291055	0.774073	-0.615750	-0.148093	-0.147774	1.000000	0.248131	0.191447
Convex hull(convex area)	-0.162780	0.600393	0.438680	0.327973	0.996626	0.834600	0.157489	0.067954	0.056435	0.160826	0.155461	0.248131	1.000000	-0.034985
Type	0.122121	-0.233531	0.110846	0.371933	-0.013641	-0.123801	0.326783	0.277235	-0.209883	-0.547632	-0.544794	0.191447	-0.034985	1.000000

```
X= df.drop(['Length (major axis)','Area','Perimeter','Compactness', 'Aspect Ratio', 'Eccentricity','Convex hull(convex area)','Type'], axis=1)
# X= df.drop(['Type'], axis=1)
X
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
0	0	227.940628	127.759132	0.309009	0.973384	0.681193
1	1	234.188126	128.199509	0.309009	0.957304	0.656353
2	2	229.418610	125.796547	0.309009	0.967270	0.683620
3	3	232.763153	125.918808	0.309009	0.965512	0.685360
4	4	230.150742	107.253448	0.309009	0.951450	0.714800
...	...	...	...	...	...	...
2798	2798	192.709366	122.356506	0.643761	0.931000	0.725739
2799	2799	186.254745	118.708961	0.643761	0.952706	0.714016
2800	2800	186.196182	119.147224	0.643761	0.948821	0.718999
2801	2801	188.660828	120.634438	0.643761	0.944810	0.738191
2802	2802	176.023636	NaN	0.643761	0.947380	0.722429

2803 rows × 6 columns

```
X.isnull().sum()
```

```
Unnamed: 0            0
Width (minor axis)    0
Thickness (depth)     1
Roundness             0
Solidity              0
Extent                0
dtype: int64
```

```
X.fillna(method= 'ffill', inplace=True)
X
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
0	0	227.940628	127.759132	0.309009	0.973384	0.681193
1	1	234.188126	128.199509	0.309009	0.957304	0.656353
2	2	229.418610	125.796547	0.309009	0.967270	0.683620
3	3	232.763153	125.918808	0.309009	0.965512	0.685360
4	4	230.150742	107.253448	0.309009	0.951450	0.714800
...	...	...	...	...	...	...
2798	2798	192.709366	122.356506	0.643761	0.931000	0.725739
2799	2799	186.254745	118.708961	0.643761	0.952706	0.714016
2800	2800	186.196182	119.147224	0.643761	0.948821	0.718999
2801	2801	188.660828	120.634438	0.643761	0.944810	0.738191
2802	2802	176.023636	120.634438	0.643761	0.947380	0.722429

2803 rows × 6 columns

```
y= df.iloc[:, -1]
y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
2798    2
2799    2
2800    2
2801    2
2802    2
Name: Type, Length: 2803, dtype: int64
```

Logistic Regression

Splitting dataset into Train Set and Test Set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
print(y_test)
```

```
855     2
615     2
70      0
352     2
118     0
       ..
2787    2
1897    1
2694    2
564     2
402     2
Name: Type, Length: 561, dtype: int64
```

```
X_train[0:5]
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
2119	2119	172.156586	105.236511	0.541884	0.933814	0.710378
1203	1203	165.031189	108.674545	0.426295	0.956497	0.796875
452	452	153.442551	132.210663	0.594944	0.988031	0.774955
1761	1761	150.679047	122.900238	0.632317	0.963890	0.765059
2682	2682	118.663330	92.374199	0.371370	0.975980	0.727930

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train[0:5]
```

```
array([[ 0.88961693,  0.00960225, -0.24339255,  0.5542151 , -0.54119414,
        -0.28631389],
       [-0.24535291, -0.22059267, -0.06745256, -0.42017207,  0.02910683,
         1.52632102],
       [-1.17587949, -0.59497823,  1.13699867,  1.00149852,  0.82193723,
         1.06696055],
       [ 0.44603702, -0.68425671,  0.66054147,  1.31654295,  0.21497291,
         0.85959545],
       [ 1.58720211, -1.71856478, -0.90161614, -0.88317757,  0.51895547,
         0.08152341]])
```

```
print(X_test)
```

```
[[-0.67654233 -0.48215625  0.16229978  0.8714193   0.8610247   1.0706117 ]
 [-0.97391434 -0.52223048 -1.67204926 -0.902931    0.47588334 -0.19336168]
 [-1.64919661 -1.95678771 -0.25245836  0.86083389  0.53098569 -0.7701934 ]
 ...
 [ 1.60207071 -1.05693271 -1.2699571  -0.85644468 -0.29110361 -0.60238411]
 [-1.03710589  1.85189513  0.68008489 -0.17034588  0.6053302  -0.48801446]
 [-1.237832   -0.4944249   1.31926755  0.99855937  0.76822769  1.18250853]]
```

Training Logistic Regression model

```
y_train
```

```
2119    1
1203    0
452     2
1761    1
2682    2
       ..
763     2
835     2
1653    1
2607    2
2732    2
Name: Type, Length: 2242, dtype: int64
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

```
y_pred.shape
```

```
(561,)
```

Predict Test results

```
y_test.shape
```

```
(561,)
```

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 0 2 ... 2 2 2]
```

Making Confusion Metrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[136   9  32]
 [  5 162  14]
 [ 56  52  95]]
0.7005347593582888
```

KNN model

```
X
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
0	0	227.940628	127.759132	0.309009	0.973384	0.681193
1	1	234.188126	128.199509	0.309009	0.957304	0.656353
2	2	229.418610	125.796547	0.309009	0.967270	0.683620
3	3	232.763153	125.918808	0.309009	0.965512	0.685360
4	4	230.150742	107.253448	0.309009	0.951450	0.714800
...	...	...	...	...	...	...
2798	2798	192.709366	122.356506	0.643761	0.931000	0.725739
2799	2799	186.254745	118.708961	0.643761	0.952706	0.714016
2800	2800	186.196182	119.147224	0.643761	0.948821	0.718999
2801	2801	188.660828	120.634438	0.643761	0.944810	0.738191
2802	2802	176.023636	120.634438	0.643761	0.947380	0.722429

2803 rows × 6 columns

```
y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
2798    2
2799    2
2800    2
2801    2
2802    2
Name: Type, Length: 2803, dtype: int64
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
X_train
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
2119	2119	172.156586	105.236511	0.541884	0.933814	0.710378
1203	1203	165.031189	108.674545	0.426295	0.956497	0.796875
452	452	153.442551	132.210663	0.594944	0.988031	0.774955
1761	1761	150.679047	122.900238	0.632317	0.963890	0.765059
2682	2682	118.663330	92.374199	0.371370	0.975980	0.727930
...	...	...	...	...	...	...
763	763	186.861343	121.848526	0.431527	0.981826	0.762699
835	835	193.119461	113.428879	0.359518	0.969854	0.737407
1653	1653	137.745041	139.815720	0.524458	0.982817	0.731298
2607	2607	219.880615	130.854446	0.375444	0.981745	0.753098
2732	2732	150.463547	78.389381	0.415667	0.964881	0.717213

2242 rows × 6 columns

```
y_train
```

```
2119    1
1203    0
452     2
1761    1
2682    2
       ..
763     2
835     2
1653    1
2607    2
2732    2
Name: Type, Length: 2242, dtype: int64
```

```
X_test
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
855	855	156.934814	113.164124	0.579513	0.989586	0.775129
615	615	155.694366	77.319206	0.369027	0.974267	0.714813
70	70	111.289436	105.059357	0.578257	0.976459	0.687287
352	352	126.405533	96.050156	0.412739	0.975364	0.718381
118	118	236.790970	99.237144	0.530404	0.986294	0.755431
...	...	...	...	...	...	...
2787	2787	213.362000	120.094116	0.386465	0.933771	0.714018
1897	1897	180.785843	115.628510	0.588775	0.988657	0.760362
2694	2694	139.143326	85.176468	0.374541	0.943761	0.695295
564	564	229.182465	123.282135	0.455931	0.979416	0.700753
402	402	156.555054	135.772369	0.594595	0.985895	0.780469

561 rows × 6 columns

```
y_test
```

```
855     2
615     2
70      0
352     2
118     0
       ..
```

```
2787    2
1897    1
2694    2
564     2
402     2
Name: Type, Length: 561, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train
```

```
array([[ 0.88961693,  0.00960225, -0.24339255,  0.5542151 , -0.54119414,
        -0.28631389],
       [-0.24535291, -0.22059267, -0.06745256, -0.42017207,  0.02910683,
         1.52632102],
       [-1.17587949, -0.59497823,  1.13699867,  1.00149852,  0.82193723,
         1.06696055],
       ...,
       [ 0.31221961, -1.10210607,  1.52618435,  0.40732183,  0.69084101,
         0.15208594],
       [ 1.49427335,  1.55138717,  1.06759484, -0.84883196,  0.66388936,
         0.60893615],
       [ 1.64915461, -0.69121871, -1.61728348, -0.5097687 ,  0.23989943,
        -0.1430791 ]])
```

```
X_test
```

```
array([[-0.67654233, -0.48215625,  0.16229978,  0.8714193 ,  0.8610247 ,
         1.0706117 ],
       [-0.97391434, -0.52223048, -1.67204926, -0.902931  ,  0.47588334,
        -0.19336168],
       [-1.64919661, -1.95678771, -0.25245836,  0.86083389,  0.53098569,
        -0.7701934 ],
       ...,
       [ 1.60207071, -1.05693271, -1.2699571 , -0.85644468, -0.29110361,
        -0.60238411],
       [-1.03710589,  1.85189513,  0.68008489, -0.17034588,  0.6053302 ,
        -0.48801446],
       [-1.237832  , -0.4944249 ,  1.31926755,  0.99855937,  0.76822769,
         1.18250853]])
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'euclidean')
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=3)
```

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 2 0 ... 2 2 2]
```

Making Confusion Metrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[174   0   3]
 [  2 178   1]
 [  2   3 198]]
0.9803921568627451
```

SVM Classifier Model

```
X
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
0	0	227.940628	127.759132	0.309009	0.973384	0.681193
1	1	234.188126	128.199509	0.309009	0.957304	0.656353
2	2	229.418610	125.796547	0.309009	0.967270	0.683620
3	3	232.763153	125.918808	0.309009	0.965512	0.685360
4	4	230.150742	107.253448	0.309009	0.951450	0.714800
...	...	...	...	...	...	...
2798	2798	192.709366	122.356506	0.643761	0.931000	0.725739
2799	2799	186.254745	118.708961	0.643761	0.952706	0.714016
2800	2800	186.196182	119.147224	0.643761	0.948821	0.718999
2801	2801	188.660828	120.634438	0.643761	0.944810	0.738191
2802	2802	176.023636	120.634438	0.643761	0.947380	0.722429

2803 rows × 6 columns

```
y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
2798    2
2799    2
2800    2
2801    2
```

```
2802    2
Name: Type, Length: 2803, dtype: int64
```

Splitting dataset into Train Set and Test Set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
X_train
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
2119	2119	172.156586	105.236511	0.541884	0.933814	0.710378
1203	1203	165.031189	108.674545	0.426295	0.956497	0.796875
452	452	153.442551	132.210663	0.594944	0.988031	0.774955
1761	1761	150.679047	122.900238	0.632317	0.963890	0.765059
2682	2682	118.663330	92.374199	0.371370	0.975980	0.727930
...	...	...	...	...	...	...
763	763	186.861343	121.848526	0.431527	0.981826	0.762699
835	835	193.119461	113.428879	0.359518	0.969854	0.737407
1653	1653	137.745041	139.815720	0.524458	0.982817	0.731298
2607	2607	219.880615	130.854446	0.375444	0.981745	0.753098
2732	2732	150.463547	78.389381	0.415667	0.964881	0.717213

2242 rows × 6 columns

```
y_train
```

```
2119    1
1203    0
452     2
1761    1
2682    2
       ..
763     2
835     2
1653    1
2607    2
2732    2
Name: Type, Length: 2242, dtype: int64
```

```
X_test
```

	Unnamed: 0	Width (minor axis)	Thickness (depth)	Roundness	Solidity	Extent
855	855	156.934814	113.164124	0.579513	0.989586	0.775129
615	615	155.694366	77.319206	0.369027	0.974267	0.714813
70	70	111.289436	105.059357	0.578257	0.976459	0.687287
352	352	126.405533	96.050156	0.412739	0.975364	0.718381
118	118	236.790970	99.237144	0.530404	0.986294	0.755431
...	...	...	...	...	...	...
2787	2787	213.362000	120.094116	0.386465	0.933771	0.714018
1897	1897	180.785843	115.628510	0.588775	0.988657	0.760362
2694	2694	139.143326	85.176468	0.374541	0.943761	0.695295
564	564	229.182465	123.282135	0.455931	0.979416	0.700753
402	402	156.555054	135.772369	0.594595	0.985895	0.780469

561 rows × 6 columns

```
y_test
```

```
855     2
615     2
70      0
352     2
118     0
       ..
2787    2
1897    1
2694    2
564     2
402     2
Name: Type, Length: 561, dtype: int64
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train
```

```
array([[ 0.88961693,  0.00960225, -0.24339255,  0.5542151 , -0.54119414,
        -0.28631389],
       [-0.24535291, -0.22059267, -0.06745256, -0.42017207,  0.02910683,
         1.52632102],
       [-1.17587949, -0.59497823,  1.13699867,  1.00149852,  0.82193723,
         1.06696055],
       ...,
       [ 0.31221961, -1.10210607,  1.52618435,  0.40732183,  0.69084101,
         0.15208594],
       [ 1.49427335,  1.55138717,  1.06759484, -0.84883196,  0.66388936,
         0.60893615],
```

```
       [ 1.64915461, -0.69121871, -1.61728348, -0.5097687 ,  0.23989943,
        -0.1430791 ]])
```

```
X_test
```

```
array([[-0.67654233, -0.48215625,  0.16229978,  0.8714193 ,  0.8610247 ,
         1.0706117 ],
       [-0.97391434, -0.52223048, -1.67204926, -0.902931  ,  0.47588334,
        -0.19336168],
       [-1.64919661, -1.95678771, -0.25245836,  0.86083389,  0.53098569,
        -0.7701934 ],
       ...,
       [ 1.60207071, -1.05693271, -1.2699571 , -0.85644468, -0.29110361,
        -0.60238411],
       [-1.03710589,  1.85189513,  0.68008489, -0.17034588,  0.6053302 ,
        -0.48801446],
       [-1.237832  , -0.4944249 ,  1.31926755,  0.99855937,  0.76822769,
         1.18250853]])
```

Training SVM classifier model on training set

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

```
      SVC
SVC(random_state=0)
```

Predicting Test set result

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 2 0 ... 2 2 2]
```

Making Confusion Metrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[163   1  13]
 [  1 180   0]
 [ 12   0 191]]
0.9518716577540107
```

Training Naive Bayes Classification model

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```

Predicting Test set result

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 0 0 ... 2 2 2]
```

Making Confusion Metrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[118   3  56]
 [  7 171   3]
 [ 38  22 143]]
0.7700534759358288
```

```
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB()
classifier.fit(X_train, y_train)
```

```
BernoulliNB
BernoulliNB()
```

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 0 0 ... 2 2 2]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 84  41  52]
 [ 17 164   0]
 [ 58  42 103]]
0.6256684491978609
```

Training Decision Tree Classification model

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Predicting Test set result

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 2 0 ... 2 2 2]
```

Making Confusion Metrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[177   0   0]
 [  0 181   0]
 [  1   0 202]]
0.9982174688057041
```

Training Random Forest Classification model

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 9, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=9, random_state=0)
```

Predicting Test set result

```
y_pred= classifier.predict(X_test)
print(np.concatenate((y_pred, y_test)))
```

```
[2 2 0 ... 2 2 2]
```

Making Confusion Metrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[177   0   0]
 [  0 181   0]
 [  0   0 203]]
1.0
```

```
my_dict= {
    "Logistic Regression":0.768270944741533,
    "KNN": 0.9893048128342246,
    "SVM": 0.9590017825311943,
    "Naive Bayes (Gaussain)": 0.7789661319073083,
    "Naive Bayes (Bernoulli)": 0.7023172905525846,
    "Decision Tree": 0.9946524064171123,
    "Random Forest": 0.9893048128342246
}

# X on drop target variable
```

```
# my_dict= {
#     "Logistic Regression": 0.7005347593582888,
#     "KNN": 0.9803921568627451,
#     "SVM": 0.9518716577540107,
#     "Naive Bayes (Gaussain)": 0.7700534759358288,
#     "Naive Bayes (Bernoulli)": 0.6256684491978609,
#     "Decision Tree": 0.9982174688057041,
#     "Random Forest": 1
# }

# when X drop many columns
```

```
Series= pd.Series(my_dict)
Series
```

```
Logistic Regression        0.768271
KNN                        0.989305
SVM                        0.959002
Naive Bayes (Gaussain)     0.778966
Naive Bayes (Bernoulli)    0.702317
```

```
Decision Tree              0.994652
Random Forest              0.989305
dtype: float64
```