#This dataset and summary taken from UCI Machine Learning Repository

Dataset Information:

The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment. High quality visualization of the internal kernel structure was detected using a soft X-ray technique. It is non-destructive and considerably cheaper than other more sophisticated imaging techniques like scanning microscopy or laser technology. The images were recorded on 13x18 cm X-ray KODAK plates. Studies were conducted using combine harvested wheat grain originating from experimental fields, explored at the Institute of Agrophysics of the Polish Academy of Sciences in Lublin.

The dataset can be used for the tasks of classification and cluster analysis.

Attribute Information:

To construct the data, seven geometric parameters of wheat kernels were measured: 1. area A, 2. perimeter P, 3. compactness C = 4*pi*A/P^2, 4. length of kernel, 5. width of kernel, 6. asymmetry coefficient 7. length of kernel groove. All of these parameters were real-valued continuous.

In [33]:
```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [34]:
```
# Importing Datasets
df=pd.read_csv("C:/Users/jcadmin/Downloads/seeds.csv")
```

In [35]:
```
df.head()
```

Out[35]:

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	1
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	1
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1

In [36]:
```
df.isnull().sum()
```

Out[36]:
```
Area              0
Perimeter         0
Compactness       0
Kernel.Length     0
Kernel.Width      0
Asymmetry.Coeff   0
Kernel.Groove     0
Type              0
dtype: int64
```

In [37]:
```
df.describe()
```

Out[37]:

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	Asymmetry.Coeff	Kernel.Groove	Type
count	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000	199.000000
mean	14.918744	14.595829	0.870811	5.643151	3.265533	3.699217	5.420653	1.994975
std	2.919976	1.310445	0.023320	0.443593	0.378322	1.471102	0.492718	0.813382
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.519000	1.000000
25%	12.330000	13.470000	0.857100	5.267000	2.954500	2.570000	5.046000	1.000000
50%	14.430000	14.370000	0.873400	5.541000	3.245000	3.631000	5.228000	2.000000
75%	17.455000	15.805000	0.886800	6.002000	3.564500	4.799000	5.879000	3.000000
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.315000	6.550000	3.000000

In [38]:
```
# Splitting Data
X=df.drop('Type',axis=1)
y=df['Type']

print('Shape of X= ', X.shape)
print('Shape of y= ', y.shape)
```

```
Shape of X=  (199, 7)
Shape of y=  (199,)
```

In [39]:
```
#Splitting the data into training 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=10)
print('Shape of X_train= ', X_train.shape)
print('Shape of X_test= ', X_test.shape)
print('Shape of y_train= ', y_train.shape)
print('Shape of y_test= ', y_test.shape)
```

```
Shape of X_train=  (159, 7)
Shape of X_test=  (40, 7)
Shape of y_train=  (159,)
Shape of y_test=  (40,)
```

In [40]:
```
# Feature Scaling

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

In [41]:
```
# Logistics regression

from sklearn.linear_model import LogisticRegression
Lo_r=LogisticRegression()
Lo_r.fit(X_train,y_train)
```

Out[41]:
```
LogisticRegression()
```

In [45]:
```
# Prediction

y_pred=Lo_r.predict(X_test)
```

In [46]:
```
y_pred
```

Out[46]:
```
array([1, 1, 1, 2, 1, 1, 3, 1, 2, 1, 2, 1, 2, 3, 3, 1, 2, 2, 3, 2, 3, 2,
       1, 2, 2, 2, 3, 1, 1, 3, 3, 1, 2, 2, 2, 2, 3, 3, 1, 3], dtype=int64)
```

In [ ]:
```
# Confusion Matrics
```

In [44]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[14  1  0]
 [ 0 14  0]
 [ 0  0 11]]
```

In [47]:
```
Lo_r.score(X_test,y_test)
```

Out[47]:
```
0.975
```

In [ ]:
```

```