In [1]:
```python
from IPython.display import Image
Image(filename='logo.PNG', height=340, width=900)
```

Out[1]:

INMOVIDU
TRANSFORM YOURSELF

BY ANKUR KHANNA

In [2]:
```python
# Importing Libraries

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

In [3]:
```python
# Import Dataset
df = pd.read_csv('seeds.csv')

del df['grain_variety']
```

In [4]:
```python
df.head()
```

Out[4]:

	area	perimeter	compactness	length	width	asymmetry_coefficient	groove_length
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220

	area	perimeter	compactness	length	width	asymmetry_coefficient	groove_length
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175

In [5]:
```
df.tail()
```

Out[5]:

	area	perimeter	compactness	length	width	asymmetry_coefficient	groove_length
205	12.19	13.20	0.8783	5.137	2.981	3.631	4.870
206	11.23	12.88	0.8511	5.140	2.795	4.325	5.003
207	13.20	13.66	0.8883	5.236	3.232	8.315	5.056
208	11.84	13.21	0.8521	5.175	2.836	3.598	5.044
209	12.30	13.34	0.8684	5.243	2.974	5.637	5.063

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 7 columns):
area                     210 non-null float64
perimeter                210 non-null float64
compactness              210 non-null float64
length                   210 non-null float64
width                    210 non-null float64
asymmetry_coefficient    210 non-null float64
groove_length            210 non-null float64
dtypes: float64(7)
memory usage: 11.6 KB
```

In [7]:
```
df.describe()
```

Out[7]:

	area	perimeter	compactness	length	width	asymmetry_coefficient	groo
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	2
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	

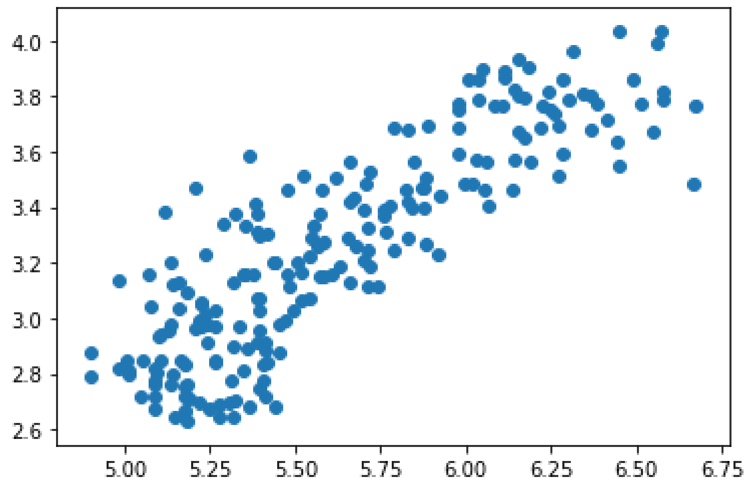
In [8]:

```
X = df.iloc[:,[3,4]].values
```

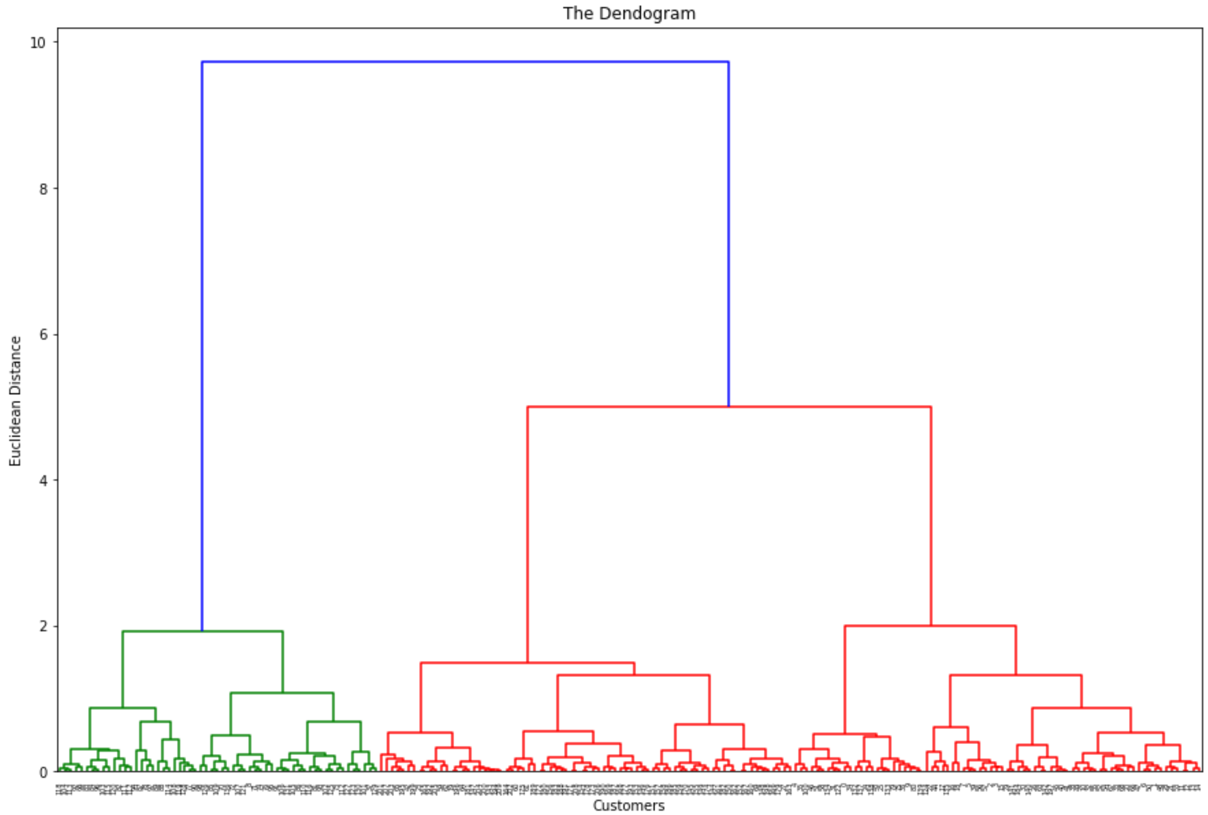
In [9]:

```
# Initial View of the Data
plt.scatter(X[:,0], X[:,1])
```

Out[9]: <matplotlib.collections.PathCollection at 0x16c23609f88>



```
In [10]: # Finding the optimum clusters using the DENDOGRAMS
import scipy.cluster.hierarchy as sch
plt.figure(figsize = (15,10))
dendogram = sch.dendrogram(sch.linkage(X, method='ward'))
plt.title('The Dendogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean Distance')
plt.show()
```



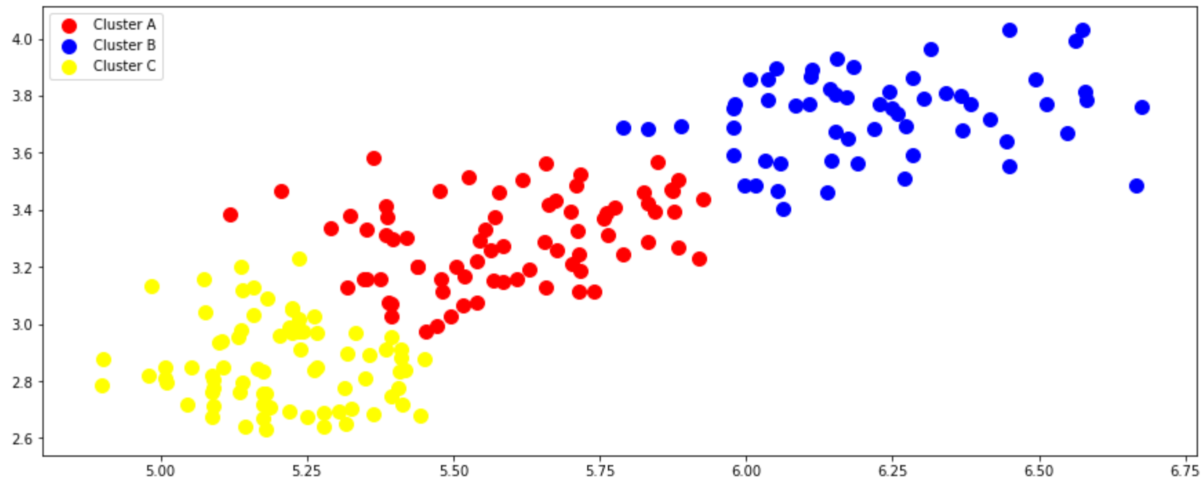

```
In [11]: # Fitting the model
from sklearn.cluster import AgglomerativeClustering
hcluster = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidea
n', linkage = 'ward')
y_clustering = hcluster.fit_predict(X)
```

In [12]:
```
y_clustering
```

Out[12]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
0,
       0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 0, 2,
2,
       0, 0, 0, 2, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
0,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int64)
```

In [13]:
```
# Visualizing Results
plt.figure(figsize=(15,6))
plt.scatter(X[y_clustering==0, 0], X[y_clustering==0, 1], s=100, c='re
d', label = 'Cluster A')
plt.scatter(X[y_clustering==1, 0], X[y_clustering==1, 1], s=100, c='blu
e', label = 'Cluster B')
plt.scatter(X[y_clustering==2, 0], X[y_clustering==2, 1], s=100, c='yel
low', label = 'Cluster C')
#plt.scatter(X[y_clustering==3, 0], X[y_clustering==3, 1], s=100, c='gr
een', label = 'Cluster D')
#plt.scatter(X[y_clustering==4, 0], X[y_clustering==4, 1], s=100, c='pi
nk', label = 'Cluster E')
plt.legend()
```

Out[13]:
```
<matplotlib.legend.Legend at 0x16c27406b88>
```

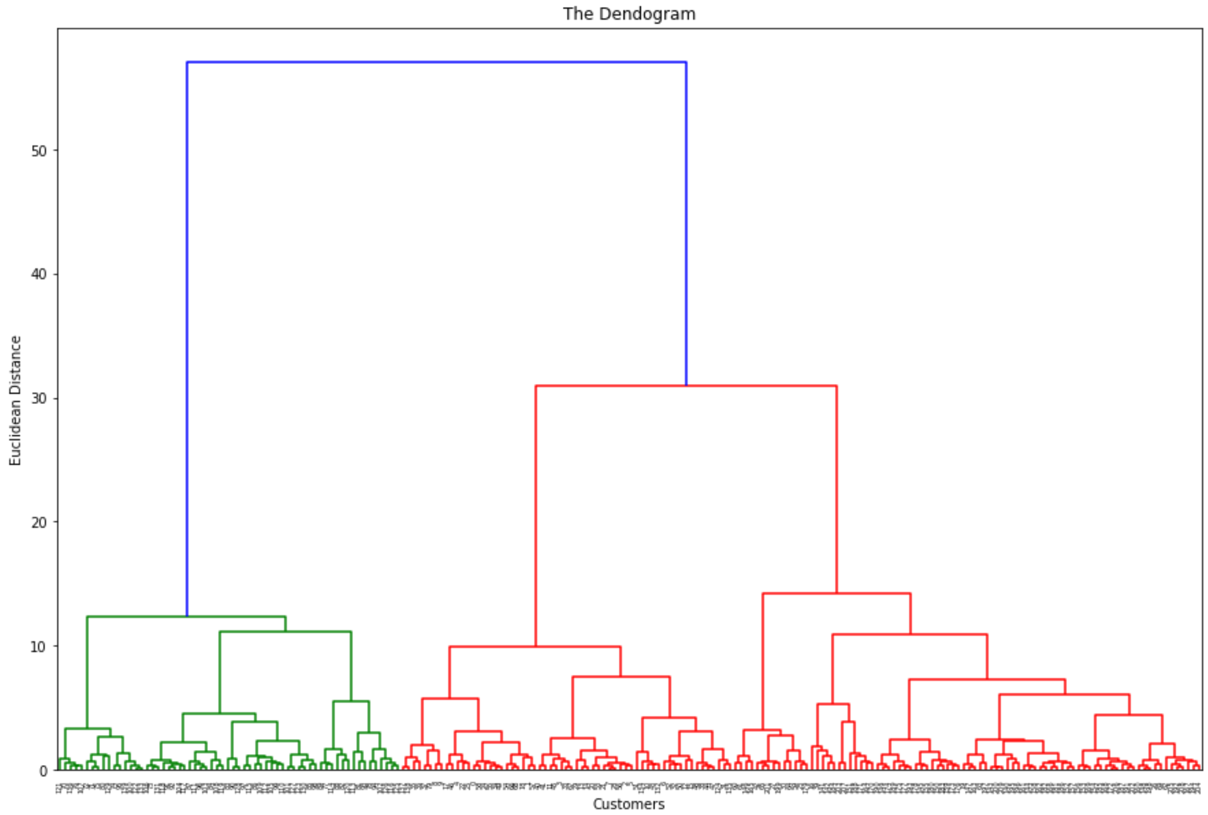


In [ ]:

In [14]:

```
# Using all the Variables now:
X1 = df.iloc[:,:].values

# Finding the optimum clusters using the DENDOGRAMS
import scipy.cluster.hierarchy as sch
plt.figure(figsize = (15,10))
dendogram = sch.dendrogram(sch.linkage(X1, method='ward'))
plt.title('The Dendogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean Distance')
plt.show()
```



In [15]:
```
# Fitting the model
from sklearn.cluster import AgglomerativeClustering
hcluster = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidea
n', linkage = 'ward')
y_clustering1 = hcluster.fit_predict(X1)
```

In [16]:
```
y_clustering1
```

Out[16]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0, 2,
2,
       2, 0, 2, 2, 0, 0, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0,
0,
```

```
       2, 2, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,
1,
       2, 1, 2, 2, 1, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [17]:
```
y_clustering
```

Out[17]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
0,
       0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 0, 2,
2,
       0, 0, 0, 2, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
0,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int64)
```

In [18]:
```
df['Initial Clustering'] = pd.Series(y_clustering)
df['Final Clustering']= pd.Series(y_clustering1)
```

In [19]:
```
df
```

Out[19]:

	area	perimeter	compactness	length	width	asymmetry_coefficient	groove_length	Ini Cluster
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	
...	...	...	...	...	...	...	...	
205	12.19	13.20	0.8783	5.137	2.981	3.631	4.870	
206	11.23	12.88	0.8511	5.140	2.795	4.325	5.003	
207	13.20	13.66	0.8883	5.236	3.232	8.315	5.056	
208	11.84	13.21	0.8521	5.175	2.836	3.598	5.044	
209	12.30	13.34	0.8684	5.243	2.974	5.637	5.063	

210 rows × 9 columns

In [20]:
```
pd.crosstab(df['Initial Clustering'], df['Final Clustering'])
```

Out[20]:

Final Clustering	0	1	2
Initial Clustering			
0	10	6	59
1	0	57	2
2	76	0	0

In [ ]: