In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale

import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
dt = pd.read_csv('wine.csv')
dt
```

Out[3]:

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids	P
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	...	
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 14 columns

In [4]:

```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Type             178 non-null    int64  
 1   Alcohol          178 non-null    float64
 2   Malic            178 non-null    float64
 3   Ash              178 non-null    float64
 4   Alcalinity       178 non-null    float64
 5   Magnesium        178 non-null    int64  
 6   Phenols          178 non-null    float64
 7   Flavanoids       178 non-null    float64
 8   Nonflavanoids    178 non-null    float64
 9   Proanthocyanins  178 non-null    float64
 10  Color            178 non-null    float64
 11  Hue              178 non-null    float64
 12  Dilution         178 non-null    float64
 13  Proline          178 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

In [5]:

```
# Removing unwanted column
```

In [7]:

```
dt2 = dt.iloc[:,1:]
dt2
```

Out[7]:

	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids	Proanthc
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 13 columns

In [8]:

```
dt2.describe()
```

Out[8]:

	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	No
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	

In [9]:

```
# Converting into numpy array
```

In [10]:

```
dt3 = dt2.values
dt3
```

Out[10]:

```
array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

In [11]:

```
## Numerical data
```

In [12]:

```
dt4 = scale(dt3)
dt4
```

Out[12]:

```
array([[ 1.51861254, -0.5622498 ,  0.23205254, ...,  0.36217728,
         1.84791957,  1.01300893],
       [ 0.24628963, -0.49941338, -0.82799632, ...,  0.40605066,
         1.1134493 ,  0.96524152],
       [ 0.19687903,  0.02123125,  1.10933436, ...,  0.31830389,
         0.78858745,  1.39514818],
       ...,
       [ 0.33275817,  1.74474449, -0.38935541, ..., -1.61212515,
        -1.48544548,  0.28057537],
       [ 0.20923168,  0.22769377,  0.01273209, ..., -1.56825176,
        -1.40069891,  0.29649784],
       [ 1.39508604,  1.58316512,  1.36520822, ..., -1.52437837,
        -1.42894777, -0.59516041]])
```

Applying PCA fit transfrom to dataset

In [13]:

```
pca = PCA(n_components=13)

dt_pca = pca.fit_transform(dt4)
dt_pca
```

Out[13]:

```
array([[ 3.31675081e+00, -1.44346263e+00, -1.65739045e-01, ...,
        -4.51563395e-01,  5.40810414e-01, -6.62386309e-02],
       [ 2.20946492e+00,  3.33392887e-01, -2.02645737e+00, ...,
        -1.42657306e-01,  3.88237741e-01,  3.63650247e-03],
       [ 2.51674015e+00, -1.03115130e+00,  9.82818670e-01, ...,
        -2.86672847e-01,  5.83573183e-04,  2.17165104e-02],
       ...,
       [-2.67783946e+00, -2.76089913e+00, -9.40941877e-01, ...,
         5.12492025e-01,  6.98766451e-01,  7.20776948e-02],
       [-2.38701709e+00, -2.29734668e+00, -5.50696197e-01, ...,
         2.99821968e-01,  3.39820654e-01, -2.18657605e-02],
       [-3.20875816e+00, -2.76891957e+00,  1.01391366e+00, ...,
        -2.29964331e-01, -1.88787963e-01, -3.23964720e-01]])
```

In [14]:

```
pca.components_
```

Out[14]:

```
array([[ 0.1443294 , -0.24518758, -0.00205106, -0.23932041,  0.14199204,
         0.39466085,  0.4229343 , -0.2985331 ,  0.31342949, -0.0886167 ,
         0.29671456,  0.37616741,  0.28675223],
       [-0.48365155, -0.22493093, -0.31606881,  0.0105905 , -0.299634  ,
        -0.06503951,  0.00335981, -0.02877949, -0.03930172, -0.52999567,
         0.27923515,  0.16449619, -0.36490283],
       [-0.20738262,  0.08901289,  0.6262239 ,  0.61208035,  0.13075693,
         0.14617896,  0.1506819 ,  0.17036816,  0.14945431, -0.13730621,
         0.08522192,  0.16600459, -0.12674592],
       [-0.0178563 ,  0.53689028, -0.21417556,  0.06085941, -0.35179658,
         0.19806835,  0.15229479, -0.20330102,  0.39905653,  0.06592568,
        -0.42777141,  0.18412074, -0.23207086],
       [-0.26566365,  0.03521363, -0.14302547,  0.06610294,  0.72704851,
        -0.14931841, -0.10902584, -0.50070298,  0.13685982, -0.07643678,
        -0.17361452, -0.10116099, -0.1578688 ],
       [-0.21353865, -0.53681385, -0.15447466,  0.10082451, -0.03814394,
         0.0841223 ,  0.01892002,  0.25859401,  0.53379539,  0.41864414,
        -0.10598274, -0.26585107, -0.11972557],
       [-0.05639636,  0.42052391, -0.14917061, -0.28696914,  0.3228833 ,
        -0.02792498, -0.06068521,  0.59544729,  0.37213935, -0.22771214,
         0.23207564, -0.0447637 ,  0.0768045 ],
       [-0.39613926, -0.06582674,  0.17026002, -0.42797018,  0.15636143,
         0.40593409,  0.18724536,  0.23328465, -0.36822675,  0.03379692,
        -0.43662362,  0.07810789, -0.12002267],
       [ 0.50861912, -0.07528304, -0.30769445,  0.20044931,  0.27140257,
         0.28603452,  0.04957849,  0.19550132, -0.20914487,  0.05621752,
         0.08582839,  0.1372269 , -0.57578611],
       [ 0.21160473, -0.30907994, -0.02712539,  0.05279942,  0.06787022,
        -0.32013135, -0.16315051,  0.21553507,  0.1341839 , -0.29077518,
        -0.52239889,  0.52370587,  0.162116  ],
       [-0.22591696,  0.07648554, -0.49869142,  0.47931378,  0.07128891,
         0.30434119, -0.02569409,  0.11689586, -0.23736257,  0.0318388 ,
        -0.04821201,  0.0464233 ,  0.53926983],
       [-0.26628645,  0.12169604, -0.04962237, -0.05574287,  0.06222011,
        -0.30388245, -0.04289883,  0.04235219, -0.09555303,  0.60422163,
         0.259214  ,  0.60095872, -0.07940162],
       [ 0.01496997,  0.02596375, -0.14121803,  0.09168285,  0.05677422,
        -0.46390791,  0.83225706,  0.11403985, -0.11691707, -0.0119928 ,
        -0.08988884, -0.15671813,  0.01444734]])
```

In [15]:

```
var = pca.explained_variance_ratio_
var
```

Out[15]:

```
array([0.36198848, 0.1920749 , 0.11123631, 0.0706903 , 0.06563294,
       0.04935823, 0.04238679, 0.02680749, 0.02222153, 0.01930019,
       0.01736836, 0.01298233, 0.00795215])
```

In [16]:

```
var1 = np.cumsum(np.round(var,4)*100)
var1
```

Out[16]:

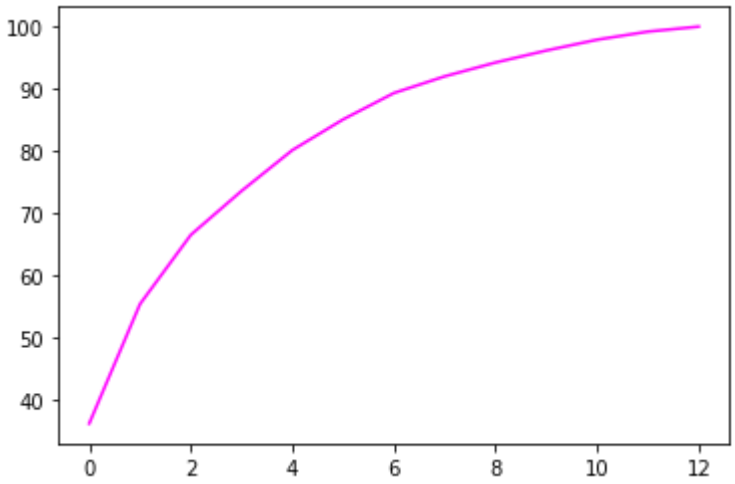
```
array([ 36.2 ,  55.41,  66.53,  73.6 ,  80.16,  85.1 ,  89.34,  92.02,
        94.24,  96.17,  97.91,  99.21, 100.01])
```

In [17]:

```
## PCA PLot
```

In [18]:

```
plt.plot(var1,color='magenta')
plt.show()
```



In [20]:

```
final_data = pd.concat([dt['Type'],pd.DataFrame(dt_pca[:,0:3],columns=['PC1','PC2','PC3'])]
final_data
```

Out[20]:

	Type	PC1	PC2	PC3
0	1	3.316751	-1.443463	-0.165739
1	1	2.209465	0.333393	-2.026457
2	1	2.516740	-1.031151	0.982819
3	1	3.757066	-2.756372	-0.176192
4	1	1.008908	-0.869831	2.026688
...	...	...	...	...
173	3	-3.370524	-2.216289	-0.342570
174	3	-2.601956	-1.757229	0.207581
175	3	-2.677839	-2.760899	-0.940942
176	3	-2.387017	-2.297347	-0.550696
177	3	-3.208758	-2.768920	1.013914

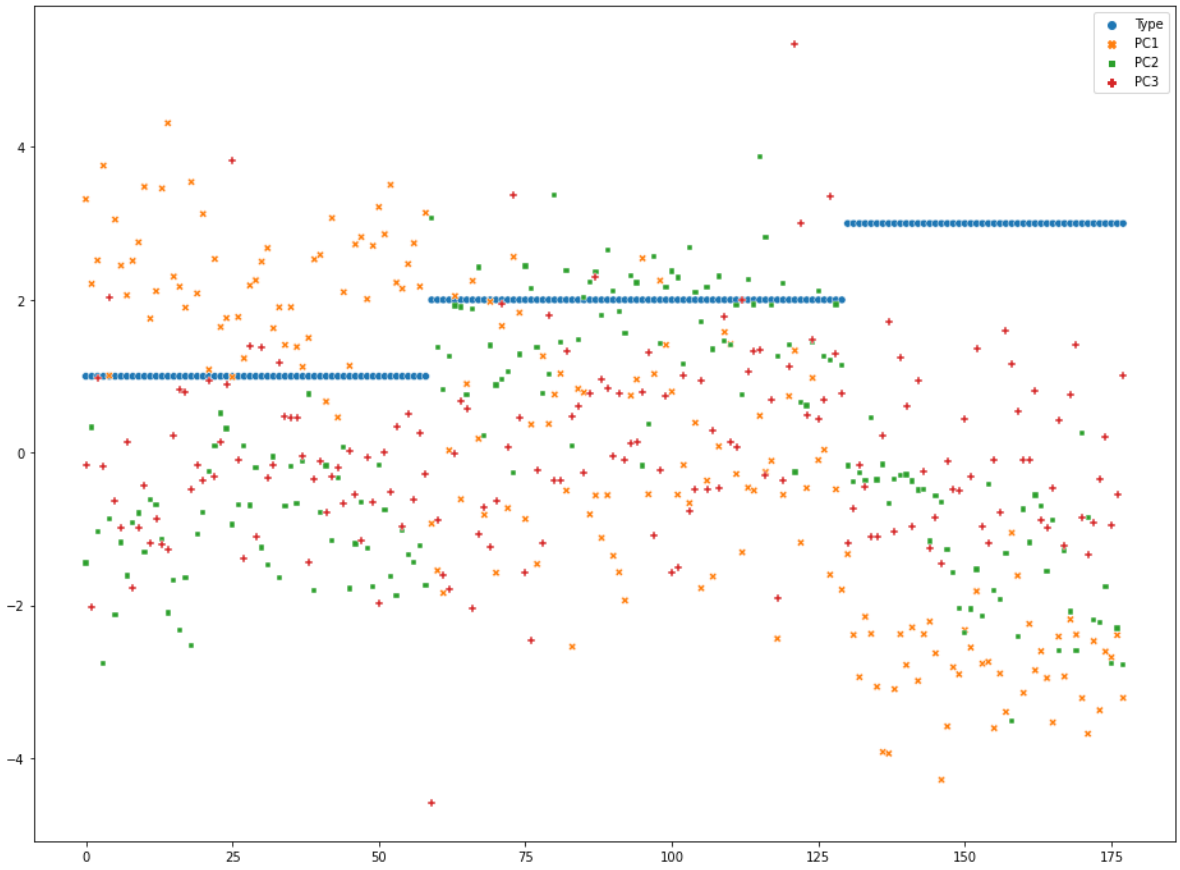
178 rows × 4 columns

In [21]:

```
## Visualisation of PCA's
```

In [22]:

```
plt.figure(figsize=(16,12))
sns.scatterplot(data=final_data)
plt.show()
```



Checking with other Clustering algorithum

In [23]:

```
## 1. Hierarchical Clustering
```

In [24]:

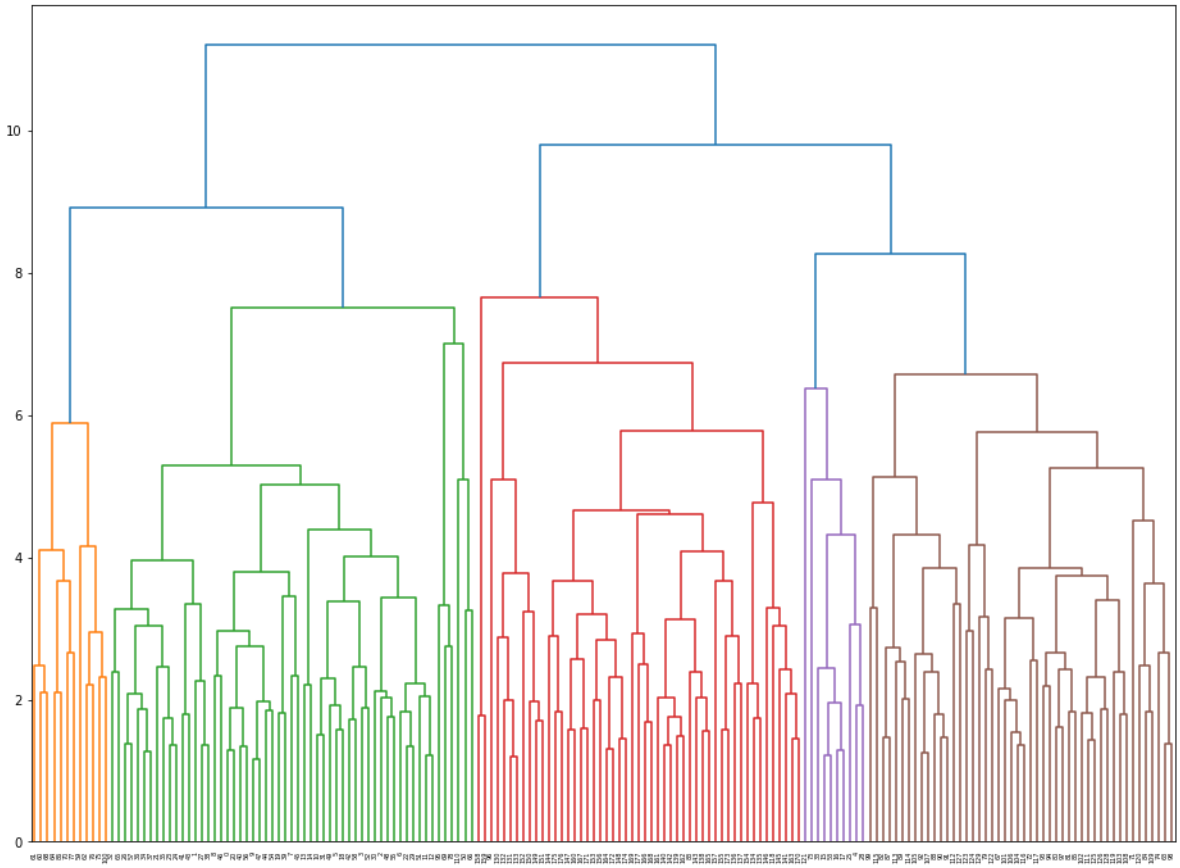
```
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import normalize
```

In [25]:

```
### Dendogram
```

In [26]:

```
plt.figure(figsize=(16,12))
dendogram = sch.dendrogram(sch.linkage(dt4,'complete'))
plt.show()
```



In [27]:

```
### Clusters
```

In [28]:

```
cl = AgglomerativeClustering(n_clusters=3,affinity='euclidean',linkage='ward')
cl
```

Out[28]:

```
AgglomerativeClustering(n_clusters=3)
```

In [30]:

```
y_cl = cl.fit_predict(dt4)
Clusters=pd.DataFrame(y_cl,columns=['Clusters'])
Clusters
```

Out[30]:

	Clusters
0	2
1	2
2	2
3	2
4	2
...	...
173	1
174	1
175	1
176	1
177	1

178 rows × 1 columns

In [31]:

```
wine = dt.copy()
wine['h_clusterid'] = cl.labels_
wine
```

Out[31]:

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids	P
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	...	
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 15 columns

In [32]:

```
## K-measn clustering
```

In [33]:

```
from sklearn.cluster import KMeans
```

In [34]:

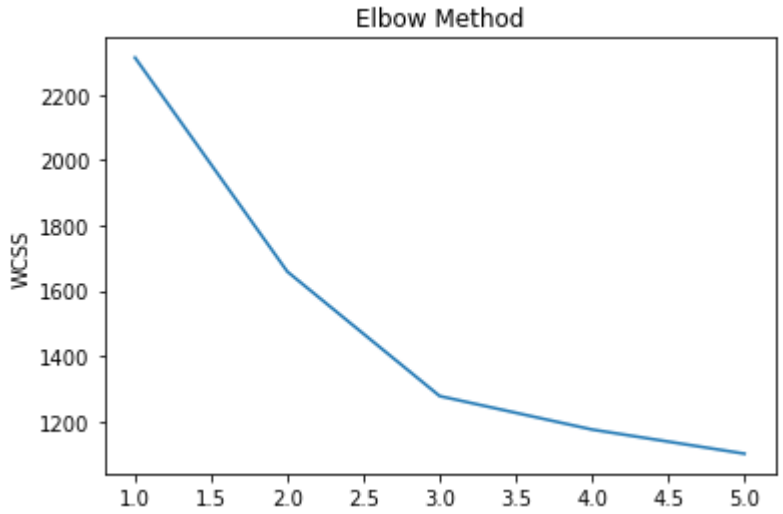
```
wcss = []
for i in range(1,6):
   kmeans = KMeans(n_clusters=i,random_state=2)
   kmeans.fit(dt4)
   wcss.append(kmeans.inertia_)
```

In [36]:

```
## WCSS Plot
```

In [35]:

```
plt.plot(range(1,6), wcss)
plt.title('Elbow Method')
plt.xlabel = ('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



In [37]:

```
## Cluster algorithum
```

In [38]:

```
clusternew = KMeans(3,random_state=30).fit(dt4)
clusternew
```

Out[38]:

```
KMeans(n_clusters=3, random_state=30)
```

In [40]:

```
clusternew.labels_
```

Out[40]:

```
array([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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 1, 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, 1, 2, 2, 2, 2, 2, 2, 2, 2, 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])
```

In [41]:

```
## Assigning Clusters to data set
```

In [43]:

```
wine2=dt.copy()
wine2['k_clustersid']=clusternew.labels_
wine2
```

Out[43]:

	Type	Alcohol	Malic	Ash	Alcalinity	Magnesium	Phenols	Flavanoids	Nonflavanoids	P
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	...	
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 15 columns

In [44]:

```
wine2['k_clustersid'].value_counts()
```

Out[44]:

```
2    65
1    62
0    51
Name: k_clustersid, dtype: int64
```

In [ ]:

```
```