In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import time
import random
```

In [2]:
```
ds = pd.read_csv('/Users/kalyanvikram/Downloads/winequality-red.csv')
ds
```

Out[2]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 12 columns

In [3]:
```
ds.shape
```

Out[3]:
```
(1599, 12)
```

In [4]:
```
t_ds= ds.to_numpy().T
```

In [5]:
```
t_ds.shape
```

Out[5]:
```
(12, 1599)
```

In [6]:
```
covariance_matrix = np.cov(t_ds)
eigen_value, eigen_vector = np.linalg.eig(covariance_matrix)
```

In [7]:
```
eigen_pairs = [(np.abs(eigen_value[i]), eigen_vector[:,i]) for i in range(l
eigen_pairs.sort(key=lambda x: x[0], reverse=True)
```

In [8]:
```
eigen_value_list = [j.reshape(len(ds.columns), 1) for _,j in eigen_pairs[:2
matrix_w = np.hstack(eigen_value_list)
print('Matrix W:\n', matrix_w)
```

```
Matrix W:
 [[ 6.13296554e-03 -2.38646792e-02]
 [-3.84670318e-04 -2.02021707e-03]
 [-1.70762384e-04 -3.02675912e-03]
 [-8.64864277e-03  1.11453593e-02]
 [-6.37476516e-05 -2.37525597e-04]
 [-2.18852809e-01  9.75212313e-01]
 [-9.75669835e-01 -2.18850408e-01]
 [-3.72590009e-06 -2.50439091e-05]
 [ 2.67974074e-04  3.26939011e-03]
 [-2.23244233e-04  6.25945868e-04]
 [ 6.35985376e-03  1.46377527e-02]
 [ 4.31953676e-03  1.15350784e-02]]
```

In [9]:
```
transformed = matrix_w.T.dot(t_ds)
transformed.shape
```

Out[9]:
```
(2, 1599)
```

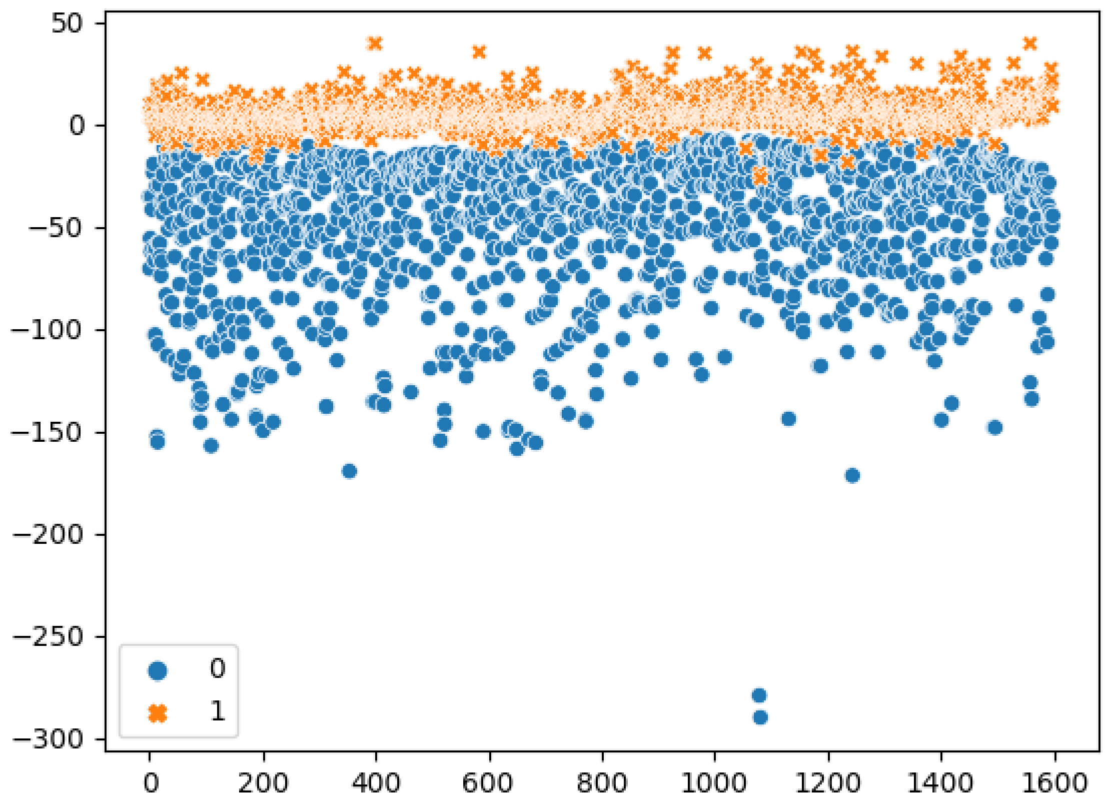
In [10]:
```
new_ds = pd.DataFrame(transformed.T)
new_ds
```

Out[10]:

	0	1
0	-35.469286	3.336638
1	-70.731567	9.770351
2	-55.856674	2.860240
3	-62.119682	3.422528
4	-35.469286	3.336638
...	...	...
1594	-49.823118	21.673219
1595	-58.179579	26.999311
1596	-45.258381	19.642837
1597	-49.826886	21.676253
1598	-44.815392	8.498672

1599 rows × 2 columns

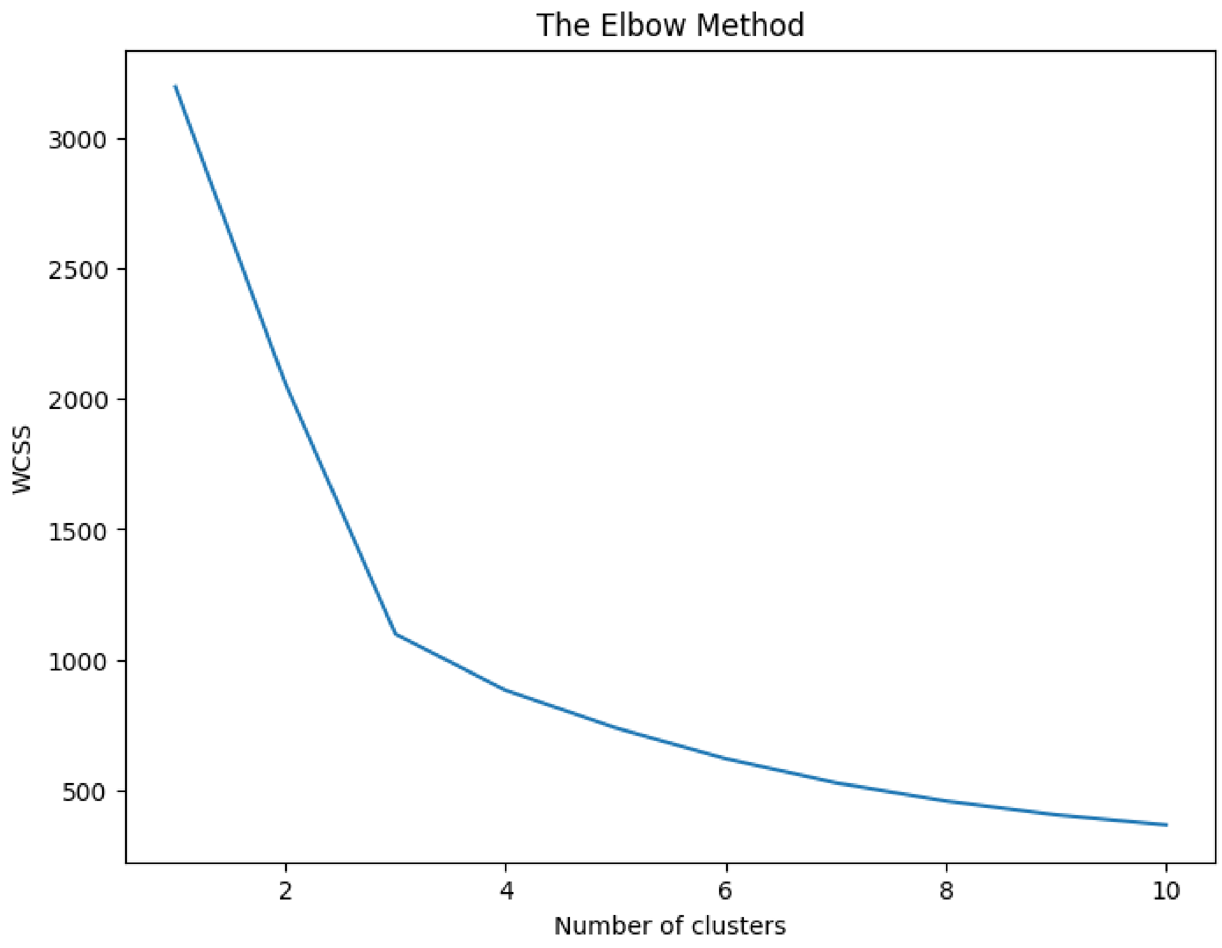
In [11]:
```
sns.scatterplot(new_ds)
plt.show()
```



In [12]:
```
X = new_ds
X_scaler = StandardScaler()
X_scaled = X_scaler.fit_transform(X)
```

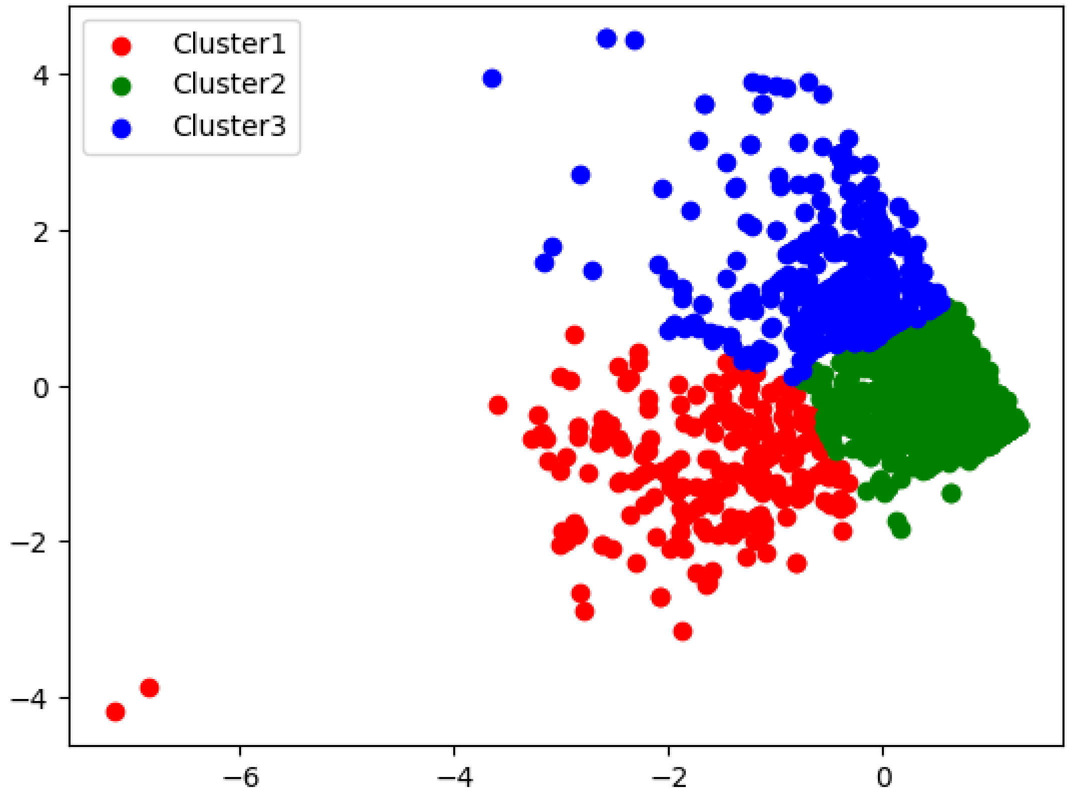
In [13]:
```
wcss = []
for i in range(1,11):
    kmeans_model = KMeans(n_clusters=i,init='k-means++', n_init=12, random_
    kmeans_model.fit(X_scaled)
    wcss.append(kmeans_model.inertia_)
```

In [14]:
```
f3, ax = plt.subplots(figsize=(8, 6))
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [15]:
```
#From above graph, we got to know number of clusters=3. Hence, applying kme
kmeans = KMeans(n_clusters = 3)
clusters = kmeans.fit_predict(X_scaled)
```

```
In [16]: #2D plot of clusters
         colors = 'rgb'
         for i in np.unique(clusters):
             plt.scatter(X_scaled[clusters==i,0],
                         X_scaled[clusters==i,1],
                         color=colors[i], label='Cluster' + str(i+1))
         plt.legend()
         plt.show()
```



```
In [ ]:
```