```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.utils import shuffle
from sklearn.cluster import MeanShift, estimate_bandwidth
from sklearn.cluster import DBSCAN
```

importing stuff

```
data = pd.read_csv("wine.csv")
data
```

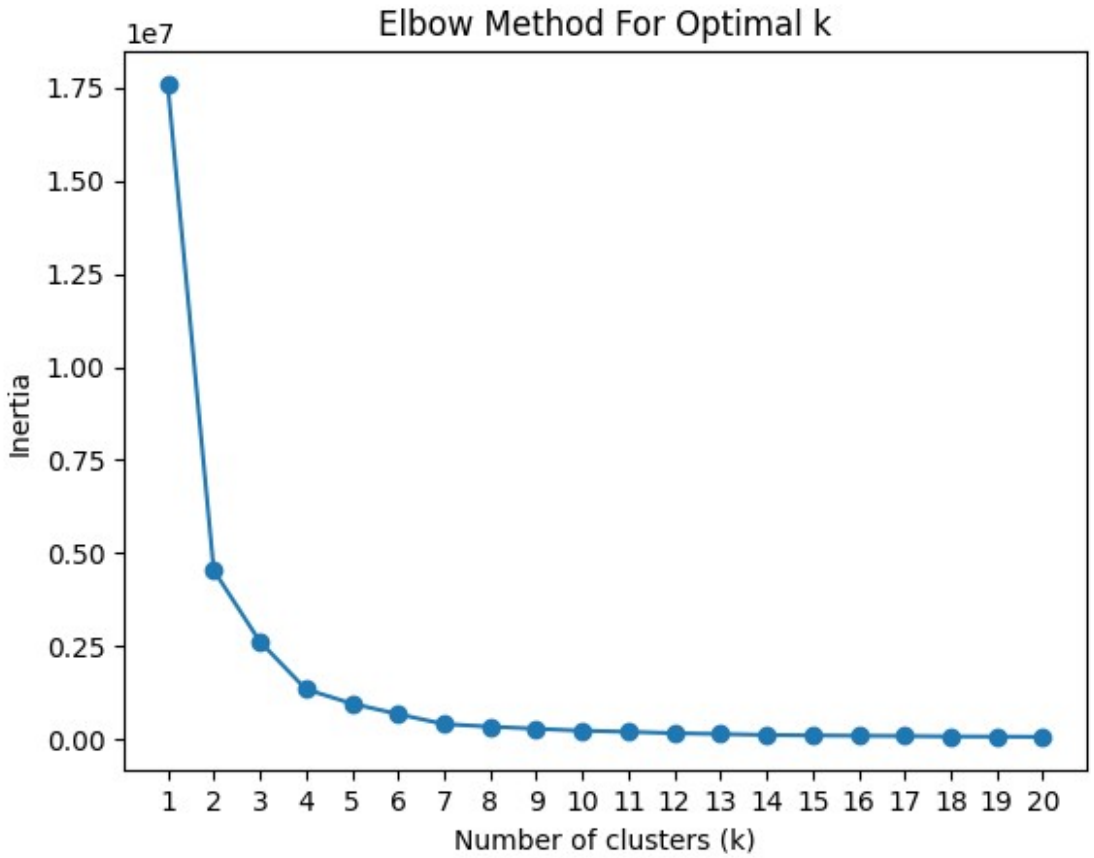
```
     Alcohol  Malic_Acid   Ash  Ash_Alcanity  Magnesium  Total_Phenols
\
0      14.23        1.71  2.43          15.6        127           2.80

1      13.20        1.78  2.14          11.2        100           2.65

2      13.16        2.36  2.67          18.6        101           2.80

3      14.37        1.95  2.50          16.8        113           3.85

4      13.24        2.59  2.87          21.0        118           2.80

..       ...         ...   ...           ...        ...            ...

173    13.71        5.65  2.45          20.5         95           1.68

174    13.40        3.91  2.48          23.0        102           1.80

175    13.27        4.28  2.26          20.0        120           1.59

176    13.17        2.59  2.37          20.0        120           1.65

177    14.13        4.10  2.74          24.5         96           2.05


     Flavanoids  Nonflavanoid_Phenols  Proanthocyanins
Color_Intensity   Hue  \
0           3.06                  0.28             2.29
5.64  1.04
1           2.76                  0.26             1.28
4.38  1.05
2           3.24                  0.30             2.81
5.68  1.03
3           3.49                  0.24             2.18
```

```
7.80  0.86
4          2.69                    0.39              1.82
4.32  1.04
..          ...                     ...               ...                ..
.   ...
173        0.61                    0.52              1.06
7.70  0.64
174        0.75                    0.43              1.41
7.30  0.70
175        0.69                    0.43              1.35
10.20  0.59
176        0.68                    0.53              1.46
9.30  0.60
177        0.76                    0.56              1.35
9.20  0.61

     OD280  Proline
0     3.92     1065
1     3.40     1050
2     3.17     1185
3     3.45     1480
4     2.93      735
..     ...      ...
173   1.74      740
174   1.56      750
175   1.56      835
176   1.62      840
177   1.60      560

[178 rows x 13 columns]
```

```
# helper function that allows us to display data in 2 dimensions an
highlights the clusters
def display_cluster(X,km=[],num_clusters=0):
    color = 'brgcmyk'
    alpha = 0.5
    s = 20
    if num_clusters == 0:
        plt.scatter(X[:,0],X[:,1],c = color[0],alpha = alpha,s = s)
    else:
        for i in range(num_clusters):
            plt.scatter(X[km.labels_==i,0],X[km.labels_==i,1],c =
color[i],alpha = alpha,s=s)
            plt.scatter(km.cluster_centers_[i]
[0],km.cluster_centers_[i][1],c = color[i], marker = 'x', s = 100)
```

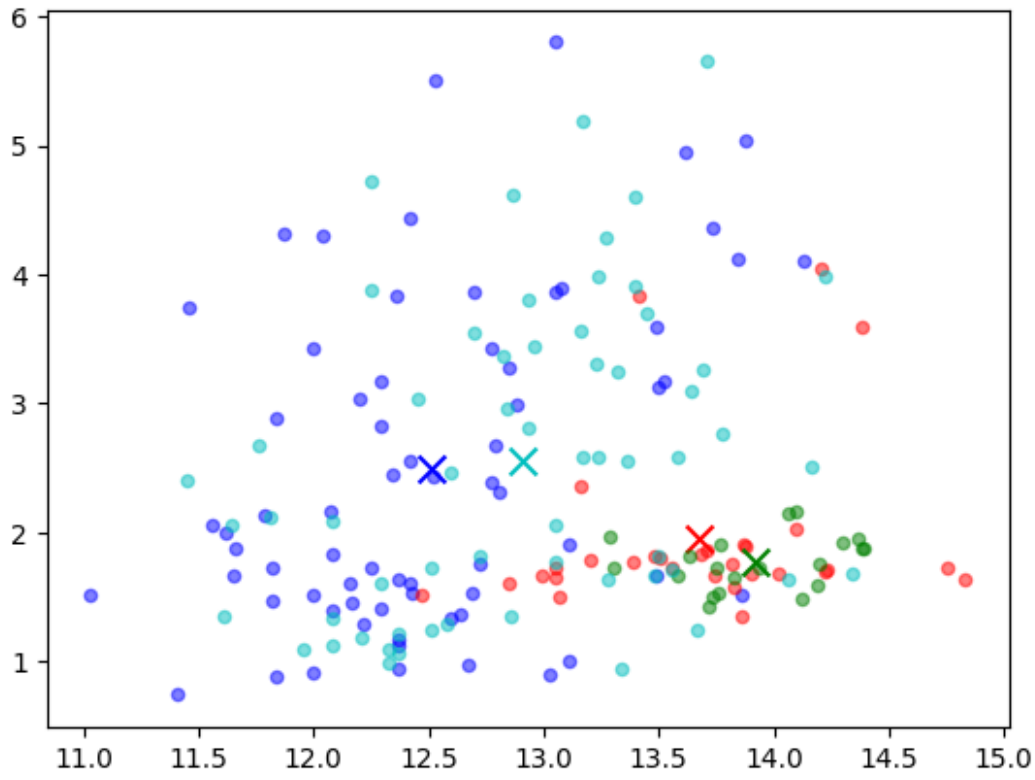
method that helps display the cluster

```
k_values = range(1, 21)
inertia_values = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia_values.append(kmeans.inertia_)

plt.plot(k_values, inertia_values, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.xticks(k_values)
plt.show()
```



using elbow method for finding the optimal number of clusters for kmeans

```
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(data)
clusters = kmeans.predict(data)
display_cluster(X,kmeans,4)
```



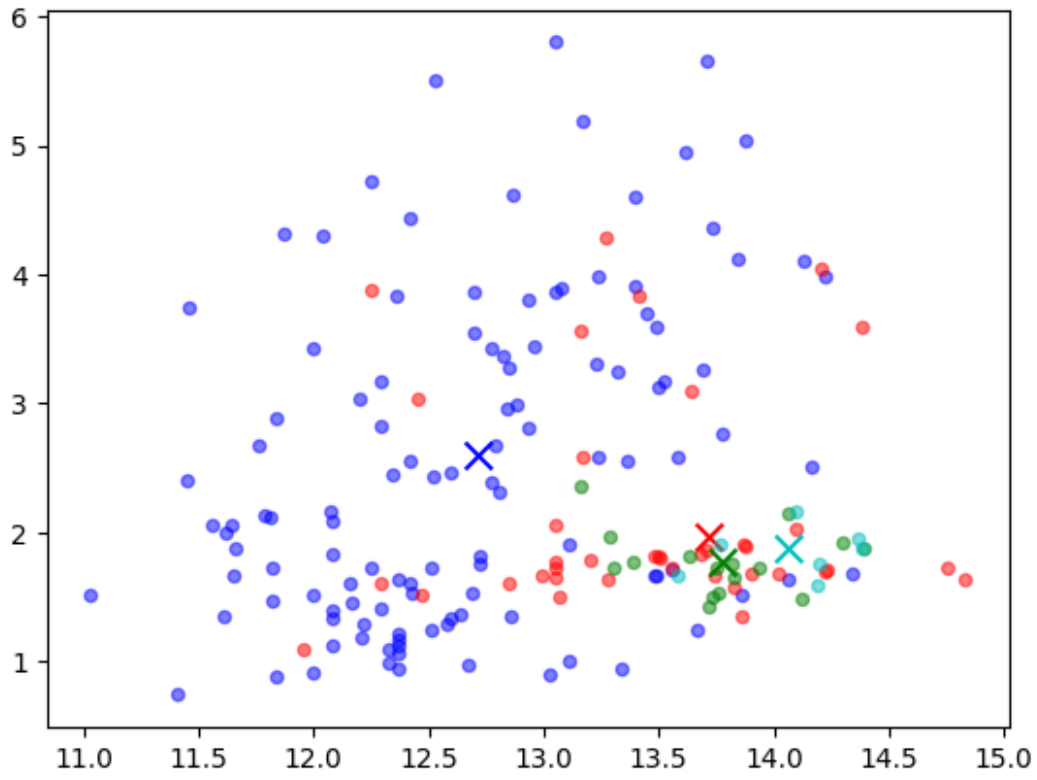
k-means clustering

```
bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=len(data))
meanshift = MeanShift(bandwidth=bandwidth, bin_seeding=True)
meanshift.fit(X)
```

```
MeanShift(bandwidth=125.63925911760262, bin_seeding=True)
```

mean shift

```
labels = meanshift.labels_
num_clusters = len(np.unique(labels))
data['Cluster'] = labels
display_cluster(X,meanshift,num_clusters)
```




```
dbscan = DBSCAN(eps=0.3, min_samples=5)
dbscan.fit(data)

DBSCAN(eps=0.3)
```