Vashu Agarwal

E21CSEU0054 EB06 lab7 q1 ¶

In [1]:

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

In [2]:

```
dataset = pd.read_csv("/Users/vashuagarwal/Downloads/BENNETT  thing
```

In [3]:

```
print(dataset.head())
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
0    842302         M        17.99         10.38          122.80
1001.0
1    842517         M        20.57         17.77          132.90
1326.0
2  84300903         M        19.69         21.25          130.00
1203.0
3  84348301         M        11.42         20.38           77.58
386.1
4  84358402         M        20.29         14.34          135.10
1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave poin
ts_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  texture_worst  perimeter_worst  area_worst  smoothness_wor
st  \
0  ...          17.33           184.60      2019.0            0.16
22
1  ...          23.41           158.80      1956.0            0.12
38
2  ...          25.53           152.50      1709.0            0.14
44
```

```
3  ...          26.50            98.87       567.7            0.20
98
4  ...          16.67           152.20      1575.0            0.13
74

   compactness_worst  concavity_worst  concave points_worst  symme
try_worst  \
0             0.6656           0.7119                0.2654
0.4601
1             0.1866           0.2416                0.1860
0.2750
2             0.4245           0.4504                0.2430
0.3613
3             0.8663           0.6869                0.2575
0.6638
4             0.2050           0.4000                0.1625
0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

In [4]:

```
f = set(["diagnosis"])
dataset["diagnosis"] = dataset["diagnosis"].map({"M":0,'B':1}).asty
print(dataset.head)
```

```
<bound method NDFrame.head of            id  diagnosis  radius_mea
n  texture_mean  perimeter_mean  \
0      842302          0       17.99         10.38          122.8
0
1      842517          0       20.57         17.77          132.9
0
2    84300903          0       19.69         21.25          130.0
0
3    84348301          0       11.42         20.38           77.5
8
4    84358402          0       20.29         14.34          135.1
0
..        ...        ...         ...           ...             ..
.
564    926424          0       21.56         22.39          142.0
0
565    926682          0       20.13         28.25          131.2
0
566    926954          0       16.60         28.08          108.3
0
567    927241          0       20.60         29.33          140.1
0
```

```
568     92751         1         7.76         24.54           47.9
2

     area_mean  smoothness_mean  compactness_mean  concavity_mean
\
0       1001.0          0.11840           0.27760         0.30010
1       1326.0          0.08474           0.07864         0.08690
2       1203.0          0.10960           0.15990         0.19740
3        386.1          0.14250           0.28390         0.24140
4       1297.0          0.10030           0.13280         0.19800
..         ...              ...               ...             ...
564     1479.0          0.11100           0.11590         0.24390
565     1261.0          0.09780           0.10340         0.14400
566      858.1          0.08455           0.10230         0.09251
567     1265.0          0.11780           0.27700         0.35140
568      181.0          0.05263           0.04362         0.00000

     concave points_mean  ...  texture_worst  perimeter_worst  are
a_worst  \
0                0.14710  ...          17.33           184.60
2019.0
1                0.07017  ...          23.41           158.80
1956.0
2                0.12790  ...          25.53           152.50
1709.0
3                0.10520  ...          26.50            98.87
567.7
4                0.10430  ...          16.67           152.20
1575.0
..                   ...  ...            ...              ...
...
564              0.13890  ...          26.40           166.10
2027.0
565              0.09791  ...          38.25           155.00
1731.0
566              0.05302  ...          34.12           126.70
1124.0
567              0.15200  ...          39.42           184.60
1821.0
568              0.00000  ...          30.37            59.16
268.6

     smoothness_worst  compactness_worst  concavity_worst  \
0             0.16220            0.66560           0.7119
1             0.12380            0.18660           0.2416
2             0.14440            0.42450           0.4504
3             0.20980            0.86630           0.6869
4             0.13740            0.20500           0.4000
..                ...                ...              ...
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
```

```
568          0.08996           0.06444          0.0000

     concave points_worst  symmetry_worst  fractal_dimension_worst
\
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
2                  0.2430          0.3613                  0.08758
3                  0.2575          0.6638                  0.17300
4                  0.1625          0.2364                  0.07678
..                    ...             ...                      ...
564                0.2216          0.2060                  0.07115
565                0.1628          0.2572                  0.06637
566                0.1418          0.2218                  0.07820
567                0.2650          0.4087                  0.12400
568                0.0000          0.2871                  0.07039

     Unnamed: 32
0            NaN
1            NaN
2            NaN
3            NaN
4            NaN
..           ...
564          NaN
565          NaN
566          NaN
567          NaN
568          NaN

[569 rows x 33 columns]>
```

In [5]:

```
x = dataset.iloc[:,2:-1].values
print(x)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]
]
```

In [6]:
```
y = dataset.iloc[:,1].values
print(y)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1
0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1
1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1
1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0
0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1
0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0
0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1
1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0
0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1
1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1
1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1
0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 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 0 0 0 0 0 0 1]
```

In [ ]:

In [7]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25
```

In [ ]:

In [8]:
```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train = st_x.fit_transform(x_train)
x_test = st_x.transform(x_test)
```

In [ ]:

```

```

In [14]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5,metric = 'minkowski
classifier.fit(x_train,y_train)
```

Out[14]:

```
KNeighborsClassifier()
```

In [15]:

```
y_pred = classifier.predict(x_test)
```

In [16]:

```
print(y_pred)
```

```
[0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1
0 1 0 1
 0 1 0 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1
1 0 1 1
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 0
1 1 1 1
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 0]
```

In [17]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
```

In [18]:

```
print(cm)
```

```
[[47  6]
 [ 1 89]]
```

In [19]:

```
from sklearn.metrics import accuracy_score
print("Accuracy of model {0}%".format(accuracy_score(y_test,y_pred)
```

```
Accuracy of model 95.1048951048951%
```

In [ ]:

```

```

Vashu Agarwal

E21CSEU0054 EB06 Lab7 Q2

In [8]:
```
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
```

In [9]:
```
dataset = pd.read_csv("/Users/vashuagarwal/Downloads/BENNETT  thing
```

In [10]:
```
print(dataset.head(5))
```

```
  private  apps  accept  enroll  top10perc  top25perc  f_undergrad  
\
0     Yes  1660    1232     721         23         52         2885
1     Yes  2186    1924     512         16         29         2683
2     Yes  1428    1097     336         22         50         1036
3     Yes   417     349     137         60         89          510
4     Yes   193     146      55         16         44          249

   p_undergrad  outstate  room_board  books  personal  phd  termin
al  \
0          537      7440        3300    450      2200   70  
78
1         1227     12280        6450    750      1500   29  
30
2           99     11250        3750    400      1165   53  
66
3           63     12960        5450    450       875   92  
97
4          869      7560        4120    800      1500   76  
72

   s_f_ratio  perc_alumni  expend  grad_rate
0       18.1           12    7041         60
1       12.2           16   10527         56
2       12.9           30    8735         54
3        7.7           37   19016         59
4       11.9            2   10922         15
```

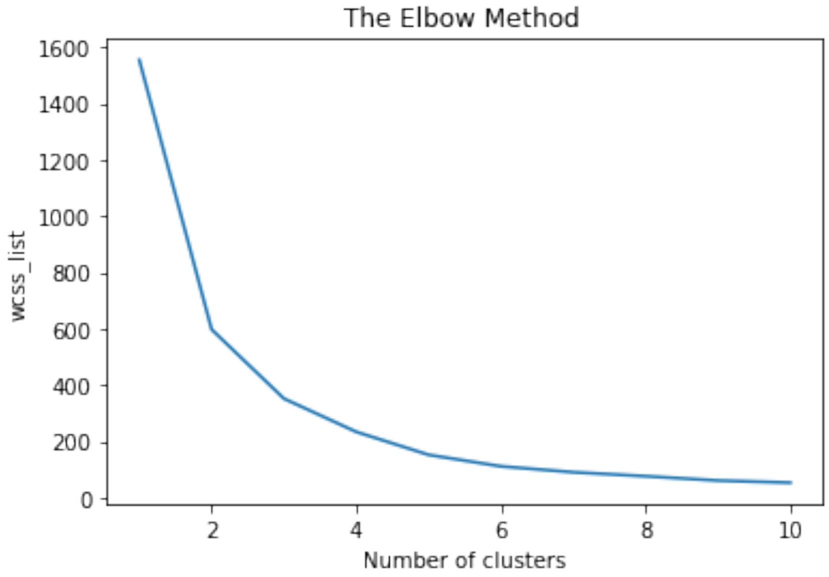
In [57]:
```
x = dataset.iloc[:,[1,2]].values
print(x)
```

```
[[ 1660  1232]
 [ 2186  1924]
 [ 1428  1097]
 ...
 [ 2097  1915]
 [10705  2453]
 [ 2989  1855]]
```

In [58]:
```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x = st_x.fit_transform(x)
```

In [59]:
```
from sklearn.cluster import KMeans
wcss_list = []
```

In [60]:
```
for i in range(1,11):
    kmeans = KMeans(n_clusters = i,init= 'k-means++',random_state =
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1,11),wcss_list)
mtp.title("The Elbow Method ")
mtp.xlabel("Number of clusters")
mtp.ylabel("wcss_list")
mtp.show()
```

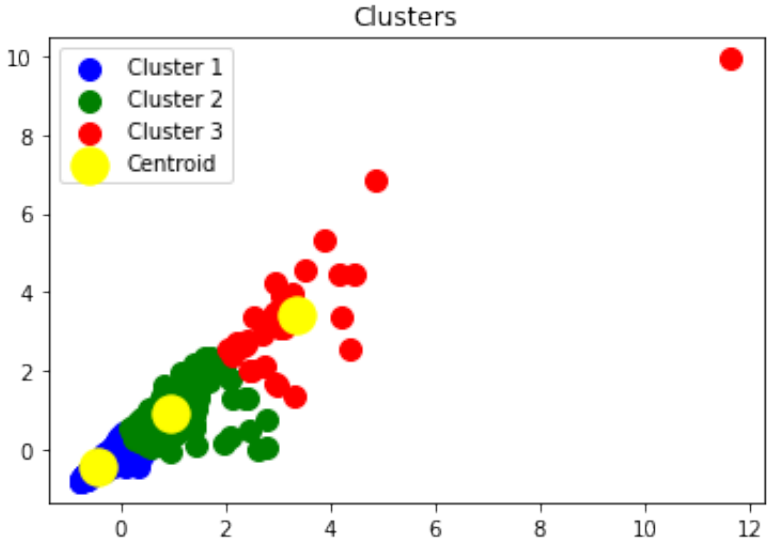


In [ ]:

In [62]:
```
kmeans = KMeans(n_clusters = 3,init = 'k-means++',random_state = 42
y_predict = kmeans.fit_predict(x)
```

In [ ]:

In [63]:
```
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c =
# mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100,
# mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100,
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[
mtp.title("Clusters")
mtp.legend()
mtp.show()
```



In [ ]:

In [ ]:

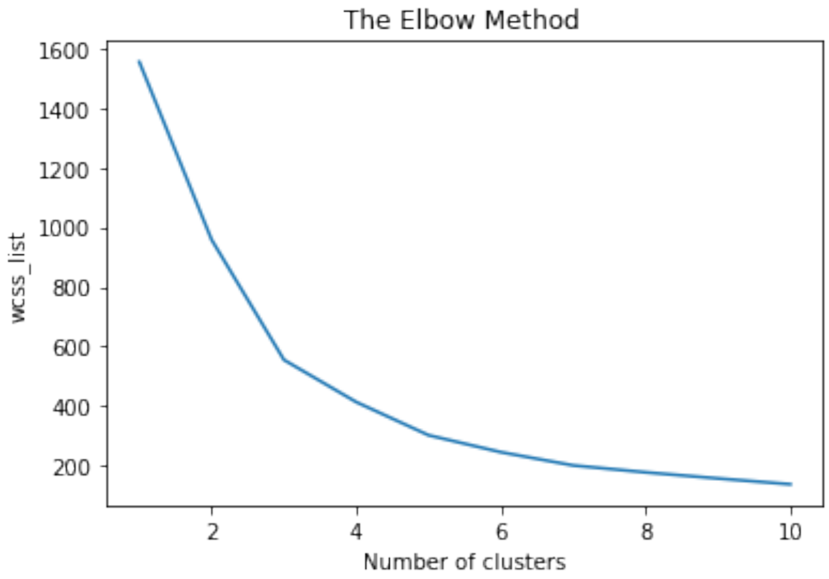
In [74]:
```
x = dataset.iloc[:,[3,4]].values
print(x)
```

```
[[ 721   23]
 [ 512   16]
 [ 336   22]
 ...
 [ 695   34]
 [1317   95]
 [ 691   28]]
```

In [75]:
```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x = st_x.fit_transform(x)
```

In [76]:
```
from sklearn.cluster import KMeans
wcss_list = []
```

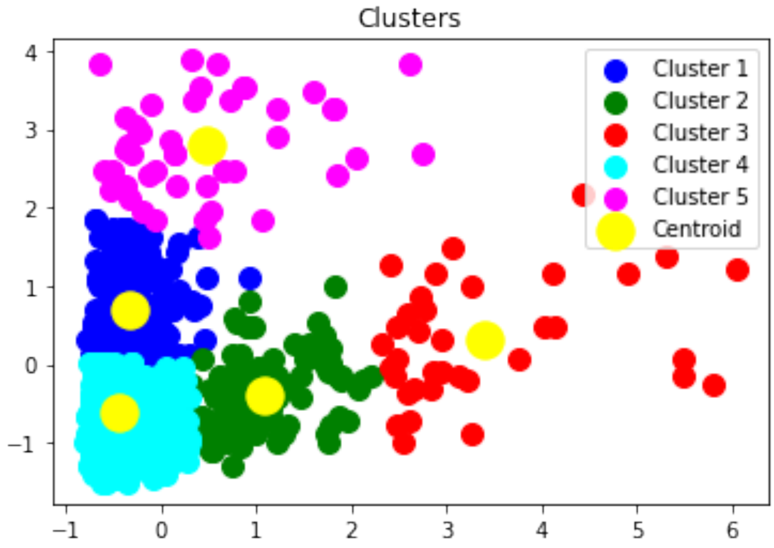
In [77]:
```
for i in range(1,11):
    kmeans = KMeans(n_clusters = i,init= 'k-means++',random_state =
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1,11),wcss_list)
mtp.title("The Elbow Method ")
mtp.xlabel("Number of clusters")
mtp.ylabel("wcss_list")
mtp.show()
```



In [79]:
```
kmeans = KMeans(n_clusters = 5,init = 'k-means++',random_state = 42
y_predict = kmeans.fit_predict(x)
```

In [80]:

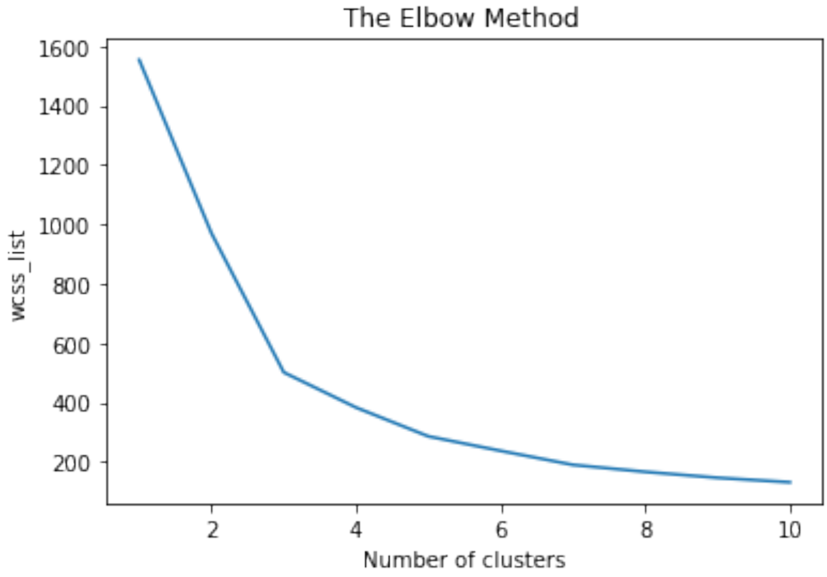
```
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c =
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[
mtp.title("Clusters")
mtp.legend()
mtp.show()
```



In [ ]:

In [81]:

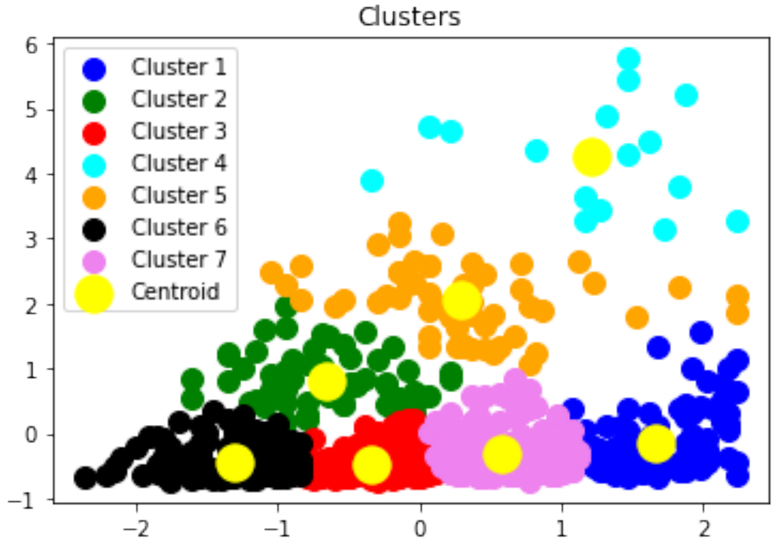
```
x = dataset.iloc[:,[5,6]].values

from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x = st_x.fit_transform(x)
from sklearn.cluster import KMeans
wcss_list = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i,init= 'k-means++',random_state =
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
mtp.plot(range(1,11),wcss_list)
mtp.title("The Elbow Method ")
mtp.xlabel("Number of clusters")
mtp.ylabel("wcss_list")
mtp.show()
```



In [ ]:

In [82]:

```
kmeans = KMeans(n_clusters = 7,init = 'k-means++',random_state = 42
y_predict = kmeans.fit_predict(x)
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c
mtp.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s = 100, c
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c
mtp.scatter(x[y_predict == 5, 0], x[y_predict == 5, 1], s = 100, c
mtp.scatter(x[y_predict == 6, 0], x[y_predict == 6, 1], s = 100, c
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[
mtp.title("Clusters")
mtp.legend()
mtp.show()
```



In [ ]:

In [ ]:

In [ ]: