Prediction Using Unsupervised ML

Author: Jeya Prakash

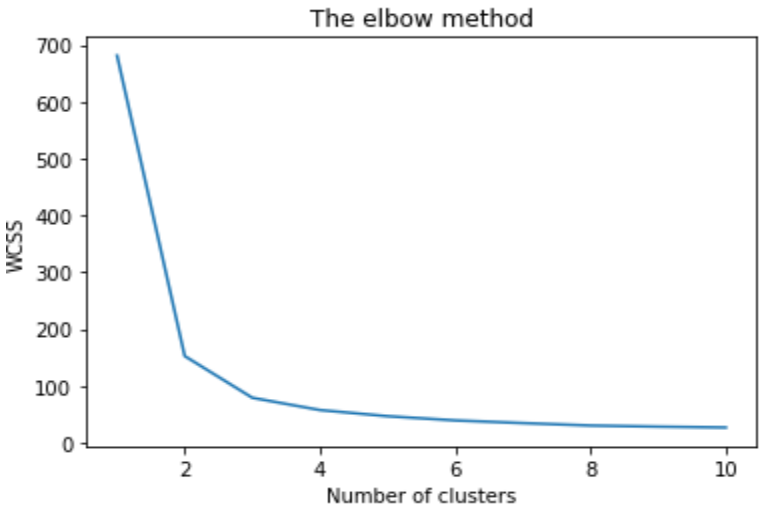
```
In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: import pandas as pd
   ...: from sklearn import datasets
```

Load datasets

```
In [5]: iris = datasets.load_iris()

In [6]: iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)

In [7]: iris_df.head()
Out[7]:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
```

Finding the optimum number of clusters for k-means classification

```
In [8]: x = iris_df.iloc[:, [0, 1, 2, 3]].values
   ...: from sklearn.cluster import KMeans
   ...: wcss = []

In [9]: for i in range(1, 11):
   ...:     kmeans = KMeans(n_clusters = i, init = 'k-means++',
   ...:                     max_iter = 300, n_init = 10, random_state = 0)
   ...:     kmeans.fit(x)
   ...:     wcss.append(kmeans.inertia_)


In [10]: plt.plot(range(1, 11), wcss)
    ...: plt.title('The elbow method')
    ...: plt.xlabel('Number of clusters')
    ...: plt.ylabel('WCSS')
    ...: plt.show()
```



The elbow method' from the above graph, the optimum clusters is where the elbow occurs. This is when the within cluster sum of squares (WCSS) doesn't decrease significantly with every iteration.

Applying kmeans to the dataset

```
In [9]: kmeans = KMeans(n_clusters = 3, init = 'k-means++',
   ...:                 max_iter = 300, n_init = 10, random_state = 0)
   ...: y_kmeans = kmeans.fit_predict(x)
```

```
In [14]:
    ...: plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
    ...:             s = 100, c = 'blue', label = 'Iris-setosa')
    ...: plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
    ...:             s = 100, c = 'yellow', label = 'Iris-versicolour')
    ...: plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
    ...:             s = 100, c = 'green', label = 'Iris-virginica')
    ...: plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1],
    ...:             s = 100, c = 'orange', label = 'Centroids')
    ...:
    ...: plt.legend()
Out[14]: <matplotlib.legend.Legend at 0x57b6c88>
```