

School of Engineering, Technology and Design Assignment Guidelines

**Title of Module**

**Artificial Intelligence Computing**

**Title(s) of Assignment**

**Assignment 1 – Clustering with K-means**

**Report Produced by**

**Marwan Jemal**

**Module Team**

**Dr. Amina Souag**

**Dr. Scott Turner**

**Assessment Type**

**Research Report with practical element**

**Abstract:**

Clustering has long been thought of as an unsupervised data processing technique. in addition to the data instances themselves, information about the issue domain is accessible. We will show how the popular k-means clustering method may be profitably tweaked, to take advantage of this information in this field of work. We see gains in clustering accuracy in trials with artificial limitations on six data sets. Moreover, we use this strategy to solve the real-world problem of automatically classifying flower datasets using Iris data, and consequentially, we see significant improvements. Clustering is the partition of a set into subsets so that the elements in each subset share some common treat. For some entities such as convoys of vehicles, crowds of people, and dust clouds, data clustering is an important procedure, and it is at the core of pattern recognition and classification (Chang and Astolfi, 2011)

**Introduction:**

K-means clustering is one of the most widely used Machine Learning techniques. K-means clustering is the most common autonomous machine learning method. Within the unlabelled dataset, K-means clustering is employed to locate intrinsic groupings and infer the dataset from them. we use K-Means clustering in this kernel to locate intrinsic groupings in the dataset that have the same status type behaviour.



Figure Clustering Concept

Problem to solve

using famous iris flower dataset from Fisher, 1936 we will be going to use k-means clustering to analysis the data.

The ***Iris* flower data set** or **Fisher's *Iris* data set** is a [multivariate](https://en.wikipedia.org/wiki/Multivariate_statistics) [data set](https://en.wikipedia.org/wiki/Data_set) introduced by the British [statistician](https://en.wikipedia.org/wiki/Statistician) and [biologist](https://en.wikipedia.org/wiki/Biologist) [Ronald Fisher](https://en.wikipedia.org/wiki/Ronald_Fisher) in his 1936 paper *The use of multiple measurements in taxonomic problems* as an example of [linear discriminant analysis](https://en.wikipedia.org/wiki/Linear_discriminant_analysis) (Chang, Astolfi, & Astolfi, 2011)

using famous iris flower dataset from Fisher, 1936 we will be going to use k-means clustering to analysis the data.

Steps:

1. Understanding the iris flower dataset.
2. Importing the data set.
3. Apply the k-means algorithm to data analysis.
4. Apply Elbow method to find out optimal value of groups or k and the results graphically.
5. Test different values of the parameter k corresponding to the number of clusters
6. Apply the results graphically

**Iris Plants Database**

Number of cases:

The data set has three classes, each with 50 instances, for a total of 150 instances. Each class refers to a different type of iris plant dataset.

The total number of qualities:

There are three classes.: Iris Setosa

Iris Versicolour

Iris Virginica

The format for the data:

(Id, sepal length in cm, sepal width in cm, petal, length in cm, petal width in cm, target, target names, and Species).

A group of purple flowers

Description automatically generated with medium confidence

Figure iris plants

(Analytics Vidhya, 2021)

Importing Libraries:

**import** pandas **as** pd #data processing/analysis

**import** numpy **as** np # linear algebra

**import** sklearn as sk #

**import** matplotlib.pyplot **as** plt #for data visualisation

**%matplotlib** inline

**from** sklearn.preprocessing **import** MinMaxScaler

**from** sklearn.datasets **import** load\_iris #for data import from sklearndatasets

**from** sklearn.cluster **import** KMeans #using k-means

Loading Data

iris = load\_iris() #as you can see on the last second code we import the iris data from Scikit learn.

Describe Data

diving into the Data

df = pd.DataFrame(iris.data,columns=iris.feature\_names)

df.head()

output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |

Here we are defining our target and predictor using Pandas. Also, we are adding two columns target name which is define type of iris plant and target with a for loop example if the target is 0 the type will equal setosa or if the target is 1 the type will equal versicolor, or if the target is 2 the type will equal virginica.

Input:

df=pd.DataFrame(data=iris.data, columns=['sepal length','sepal width','petal length','petal width'])

df['target']=pd.Series(iris.target)

df['target\_names']=pd.Series(iris.target\_names)

species = []

for i in range(len(df)):

if df.iloc[i]['target'] == 0:

species.append('setosa')

elif df.iloc[i]['target'] == 1:

species.append('versicolor')

elif df.iloc[i]['target'] == 2:

species.append('virginica')

df['Species'] = species

df #print the table to see our results how the data been added to the able.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | sepal length | sepal width | petal length | petal width | target | target\_names | Species |
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | 0 | setosa | setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | 0 | versicolor | setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | 0 | virginica | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | 0 | NaN | setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | 0 | NaN | setosa |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | 2 | NaN | virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | 2 | NaN | virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | 2 | NaN | virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | 2 | NaN | virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | 2 | NaN | virginica |

**Apply the k-means algorithm to data analysis:**

Clustering:

To apply the k algorithm, we will follow the following steps:

1. pick a random number of k clusters.
2. As centroids, choose k-means random points from the data.
3. Allocate all of the points to the cluster centroid that is closest.
4. Calculate the centroids of freshly generated clusters once more.
5. Perform steps three and four unit you find the final k-means.

Taring and prediction

kmeans4 = KMeans(n\_clusters=4,init = 'k-means++', random\_state = 0)

y = kmeans4.fit\_predict(x)

print(y))

output: [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 3 3 3 0 3 0 3 0 3 0 0 0 0 3 0 3 0 0 3 0 3 0 3 3

3 3 3 3 3 0 0 0 0 3 0 3 3 3 0 0 0 3 0 0 0 0 0 3 0 0 2 3 2 2 2 2 0 2 2 2 3

3 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 3 2 2 2 2 2 3 3 2 2 2 3 2 2 2 3 2 2 2 3 3

2 3]

Kmeans4.cluster\_centers\_

output: array ([[5.53214286, 2.63571429, 3.96071429, 1.22857143],

[5.006, 3.428, 1.462, 0.246],

[6.9125, 3.1, 5.846875, 2.13125],

[6.2525, 2.855, 4.815, 1.625]])

Plotting predication

#visualising the data that been clustered:

Input:

plt.scatter(x[y == 0,0], x[y==0,1], s = 15, c= 'yellow', label = 'k1')

plt.scatter(x[y == 1,0], x[y==1,1], s = 15, c= 'blue', label = 'k2')

plt.scatter(x[y == 2,0], x[y==2,1], s = 15, c= 'green', label = 'k3')

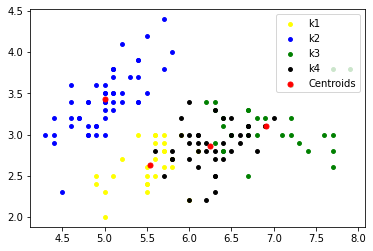
plt.scatter(x[y == 3,0], x[y==3,1], s = 15, c= 'black', label = 'k4')

plt.scatter(kmeans4.cluster\_centers\_[:,0], kmeans5.cluster\_centers\_[:,1], s = 25, c = 'red', label = 'Centroids')

plt.legend()

plt.show()

output:



Centroids:

A centroid is a vector with one digit for each variable, each digit representing the mean of that variable for the observations in that k-means. The k-means multi-dimensional average can be thought of as the centroid.

**Apply Elbow method to find out optimal value of groups or k and the results graphically**

WCSS the Within-Cluster-Sum-of-Squares Within each cluster, is a measure of the variability of the observations. A cluster with a small sum of squares is generally more compact than one with a big number of squares.

A picture containing text

Description automatically generated

(EduPristine, 2018)

Finding the optimum number of clusters

Error =[]

for i in range(1, 11):

kmeans11 = KMeans(n\_clusters = i, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0).fit(x)

kmeans11.fit(x)

Error.append(kmeans11.inertia\_)

Input to build the elbow graph:

plt.plot(range(1, 11), Error)

plt.title('Apply Elbow Method Graph')

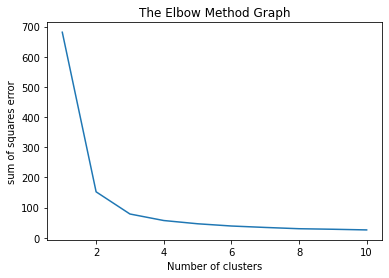
plt.xlabel('Number of clusters’)

plt.ylabel('sum of squares error')

plt.show()

The Elbow method's output graph is presented below

Output:



Our value is k means three as we can see above on the graph its between two and four. Therefore, we are going to apply k means elbow value below to the dataset.

Input:

kmeans3 = KMeans(n\_clusters=3, random\_state=21)

y = kmeans3.fit\_predict(x)

print(y)

output:

[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 0 0 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 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2

2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2

2 0]

Here we will need three centroids as our k means value is 3.

Input:

kmeans3.cluster\_centers\_

output: array ([[5.9016129, 2.7483871, 4.39354839, 1.43387097],

[5.006, 3.428, 1.462, 0.246],

[6.85, 3.07368421, 5.74210526, 2.07105263]])

Applying k means 3 to the graphic and visualising the data that been clustered

Input:

plt.scatter(x[y == 0,0], x[y==0,1], s = 15, c= 'yellow', label = 'k1')

plt.scatter(x[y == 1,0], x[y==1,1], s = 15, c= 'blue', label = 'k2')

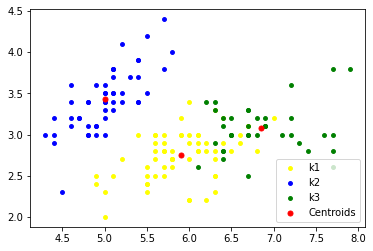
plt.scatter(x[y == 2,0], x[y==2,1], s = 15, c= 'green', label = 'k3')

plt.scatter(kmeans3.cluster\_centers\_[:,0], kmeans3.cluster\_centers\_[:,1], s = 25, c = 'red', label = 'Centroids')

plt.legend()

plt.show()

output:



**Conclusions:**

We used the sklearn dataset to investigate and pre-process the Iris dataset.

This study compares K-Means Clustering on the Iris Dataset, using the dissimilarity measures Euclidean distance and Manhattan Distance, respectively. We can infer that CLARA Clustering using Manhattan distance is superior than K-Means Clustering using Euclidean distance after plotting graphs using the two properties of the dataset, "Petal. Length" and "Petal. Width."

**Reference:**

archive.ics.uci.edu. (n.d.). *UCI Machine Learning Repository: Iris Data Set*. [online] Available at: http://archive.ics.uci.edu/ml/datasets/Iris.

‌

Analytics Vidhya. (2021). *Analyzing Decision Tree and K-means Clustering using Iris dataset.* [online] Available at: https://www.analyticsvidhya.com/blog/2021/06/analyzing-decision-tree-and-k-means-clustering-using-iris-dataset/.

ynpreet (2021). *thesparksfoundation-projects/Unsupervised Machine learning\_Iris data set.ipynb at main · ynpreet/thesparksfoundation-projects*. [online] GitHub. Available at: https://github.com/ynpreet/thesparksfoundation-projects/blob/main/Task2:%20KMeans%20clustering%20%7C%20Iris%20data%20set/Unsupervised%20Machine%20learning\_Iris%20data%20set.ipynb.

EduPristine. (2018). *EduPristine*. [online] Available at: https://www.edupristine.com/blog/beyond-k-means.

Chang, H., Astolfi, A., & Astolfi, A. (2011). Gaussian Based Classification with Application to the Iris Data Set. *IFAC Proceedings Volumes, 44*(1), 14271-14276. Retrieved 4 1, 2022, from https://sciencedirect.com/science/article/pii/s1474667016459203