K-Means Clustering Implementation and Analysis

K-means Clustering

It is one of the simplest algorithm used in the field of data science for unsupervised learning. Here, the task is to group data points in to some number of clusters. K-means clustering is an iterative process of giving output as K number of clusters of the dataset.

Cluster: A group of data points which are collected together because of similarities between them.

Algorithm works as:

1. Select K random centroid from the dataset, where centroid can be imaginary or real data point.
2. Loop until centroids doesn't change or user defined stopping criteria met:

   2.1 Assign data points to respective cluster based on their distance from the centroid of every cluster and choosing the one with minimum value.

   2.2 Update the centroid of each cluster.
3. Return the K clusters.

Q1. K-means Clustering Implementation

Data Files:

We have four files viz. 'animals', 'countries', 'fruits', 'veggies'

DataSet:

All four files have been combined in to single Dataset named ‘wholeData’ which is python list of list implementation.

Shape : 329*301 where last element in every element of wholeData is labled according to following set:

Animals	0
Countries	1
Fruits	2
Veggies	3

External library used:

1. Numpy for array like operations.
2. Sklearn for cosine similarity function only.
3. Matplotlib for plotting various graphs.

Recall:

It gives the answer “How many data points are correctly identified belonging to class out of all which actually belongs to a same class?”

Precision:

It gives the answer “How many data points are correctly identified belonging to a class out of all that the model predict for a class?”

$$recall = \frac{true\ positives}{true\ positives\ + false\ negatives} \qquad precision = \frac{true\ positives}{true\ positives + false\ positives}$$

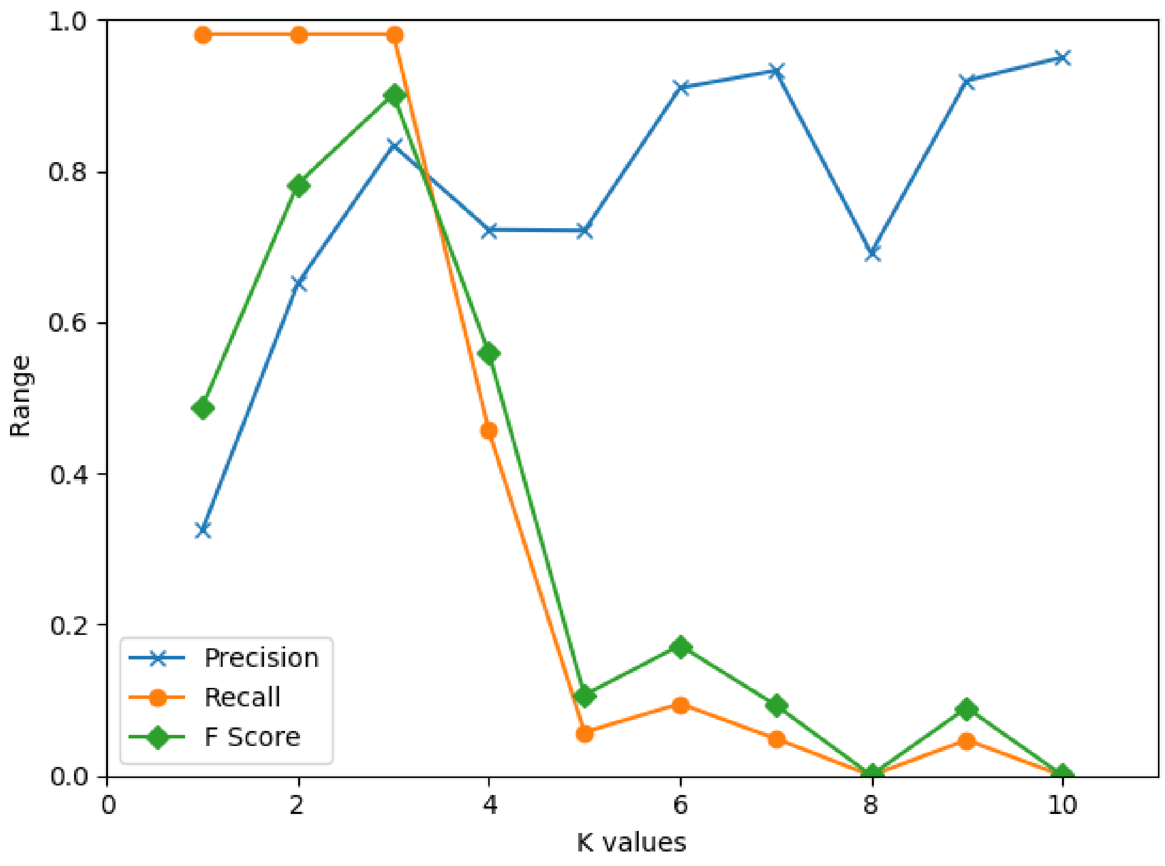
F-score:

It is the harmonic mean of the precision and recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Q2. Precision, Recall and F-score using Euclidean Distance function.

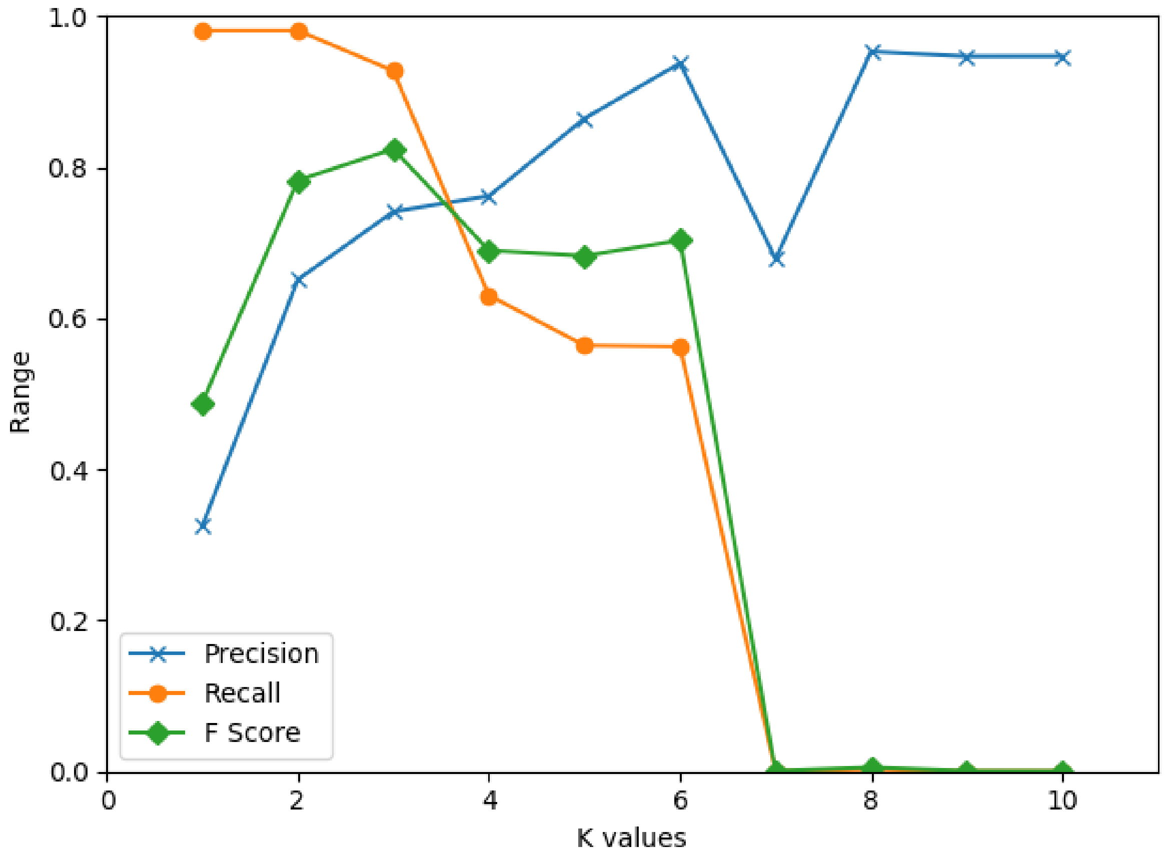
Euclidean Distance = $d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk}\right)^2}$



Observation and Inference:

1. With the increase in the K value, precision increases (except K=8) but recall and F-score goes down very sharp after K =3.
2. K=3 seems to be good setting for this implementation as all the clauses are closer and high in range.

Q3. Precision, Recall and F-score using Euclidian Distance function and L2 normalized data.

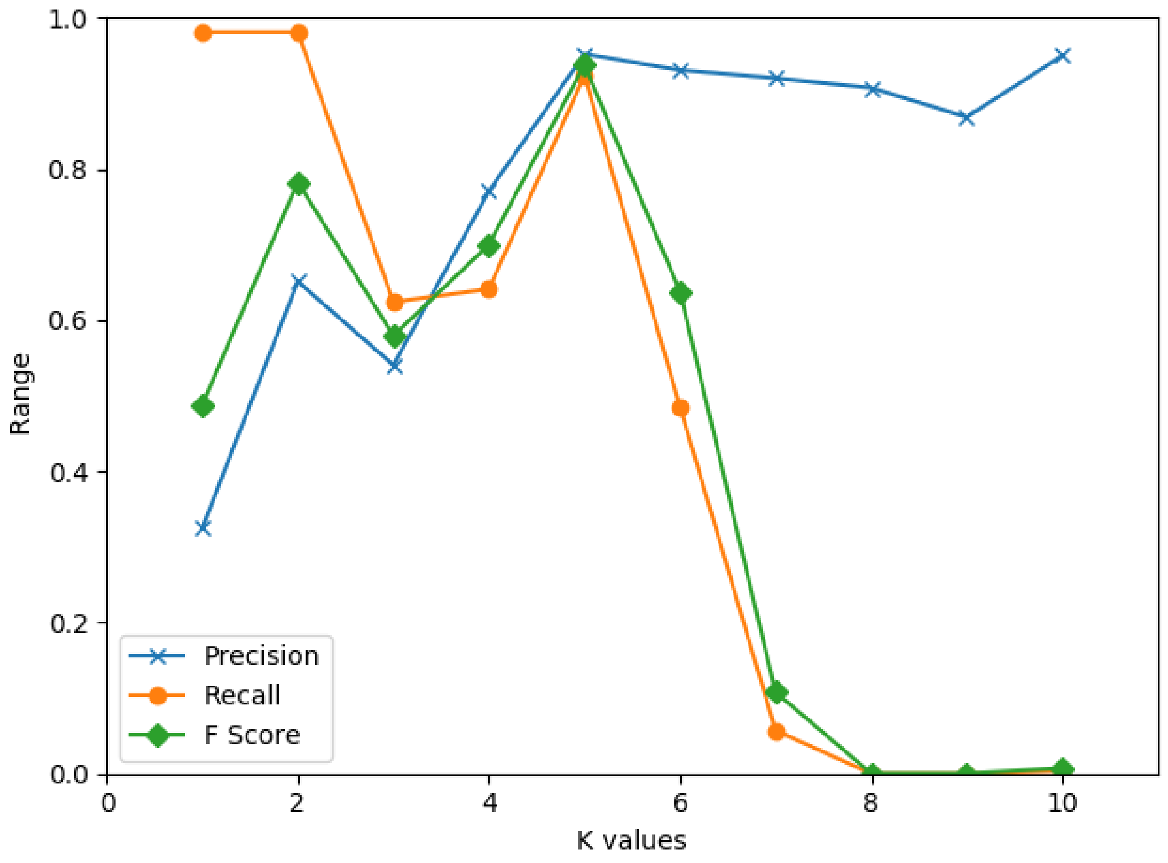


Observation and Inference:

1. F score and recall are inversely proportional to K-value after K=6.
2. Precision is directly proportional to K-value.
3. K=3 is good setting with this implementation.

Q4. Precision, Recall and F-score using Manhattan Distance function.

Manhattan Distance = |x1-y1|\ +\ |x2-y2|\ +\ ...\ +\ |xN-yN|}



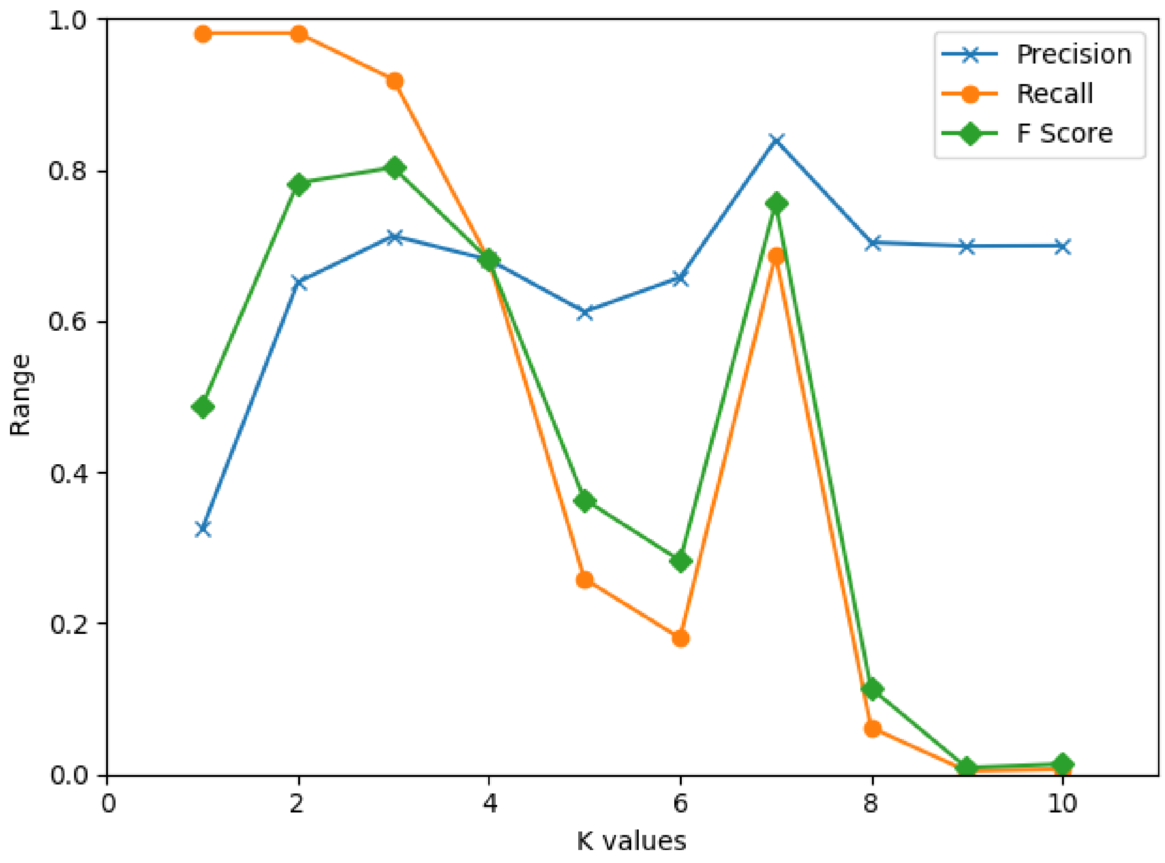
Observation and Inference:

1. There is sharp difference between the precision and recall after K=4 which is not good indicator of our model.
2. K=5 is a really good setting with Manhattan distance as all the three values converge at same point.

Q5. Precision, Recall and F-score using cosine similarity.

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.



Observation and Inference:

1. Most interesting graph is obtained from this implementation using cosine similarity as our model is able to give one best setting and one good setting for the clustering.
2. Also, graph shows that precision is stable after K=4 but recall and F score fluctuate a lot.
3. K=4 is the best setting for this implementation using cosine similarity.

Q6. Comparison of results from Q2 to Q5 and best setting.

From Q2-Q5, results are summarized:

Distance Function	L2 Normalization(Yes/No)	Best K value
Euclidean Distance	No	3
Euclidean Distance	Yes	3
Manhattan Distance	No	5
Cosine Similarity	No	4

Analysis:

When Euclidean function was used with or without normalization, though the best value of K turns out to be 3 with varying precision and recall which means that the model is little better in giving result which actually belong to the class as compared to its ability to find actual result out of all predicted.

When cosine similarity and Manhattan distance functions are used, all the three parameters of performance converge to different value of K giving us the idea that model is equally good in both the function of precision and recall and is a good model.

Best Setting:

According to above observations and inferences, it is pretty much clear that K-Means with K=5 and Manhattan distance function is the best setting because of:

1. Highest F-score value.
2. All three parameter converge at a single point with highest value.

References:

1. Wikipedia
2. https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html