COMP337 Assignment 2 Project Report

Wuwei Zhang, 201522671, sgwzha23@liverpool.ac.uk

1. KMeans and KMedians implementation (Task 1 & 2)

Algorithm 1 KMeans:

Input: train_sample X, number_cluster K
randomly k choose cluster representative $C_1, C_2, ...C_k$ from X
while $C_1, C_2, ...C_k$ are not changing **do**
  initialize $classification = \{\}$, $labels = [\,]$
  for $instance$ **in** X **do**
    apply argmax to all distance between $representatives$ to $instance$ as $class$ by L2 Distance
    append $instance$ to $classification[class]$
  end for
  for C in $ClusterRepresentatives$ **do**
    Calculate mean point $Mean\_Point$ in $classification[C]$
    $ClusterRepresentatives[C] = Mean\_Point$
  end for
end while
return $labels$

K-Means algorithm is a clustering algorithm that partitions n observations by L2 distance into k clusters in which each observation belongs to the cluster with the nearest mean. The above pseudo-code describes the algorithm.

Algorithm 2 KMedians:

Input: train_sample X, number_cluster K
randomly k choose cluster representative $C_1, C_2, ...C_k$ from X
while $C_1, C_2, ...C_k$ are not changing **do**
  initialize $classification = \{\}$, $labels = [\,]$
  for $instance$ **in** X **do**
    apply argmax to all distance between $representatives$ to $instance$ as $class$ by L1 Distance
    append $instance$ to $classification[class]$
  end for
  for C in $ClusterRepresentatives$ **do**
    Calculate median point $Median\_Point$ in $classification[C]$
    $ClusterRepresentatives[C] = Median\_Point$
  end for
end while
return $labels$

K-Medians algorithm is a clustering algorithm that partitions n observations by L1 distance into k clusters in which each observation belongs to the cluster with the nearest median. The above pseudo-code describes the algorithm.

In this project, these two cluster algorithm are respectively implemented into two classes which are K_Means and $K\_Medians$ and they are initialized with specific K and clustering the training data loaded from CA2.data by $.fit()$.

Distance function being used:

L2 distance between x and y: $sqrt(sum(x^2 - y^2))$.

L1 distance between x and y: $sum(abs(x - y))$.

2. Cluster Algorithm Evaluation with B-CUBED and F-Score (Task 3, 4, 5, 6)

In comparison to labelling method, B-CUBED method does not require to compute the label for each cluster representative, we just need to compute the precision and recall for each instance, and we use the mean value for all instances to represent the B-CUBED precision and recall for the data-set and algorithm we evaluated.

The procedure of computing B-CUBED precision and recall is showing as follow:

Algorithm 3 B-CUBED SCORE:

Input: train_label Y, predict_label $Prediction$

$precision\_list = [\,]$, $recall\_list = [\,]$

for $index, p$ **in** $Prediction$ **do**

$precision\_base =$ the list of index in Prediction that $value = p$

$recall\_base =$ the list of index in Y that $value = y[i]$

$dividend =$ the intersection between $precision\_base$ and $recall\_base$

$Precision\_for\_current\_instance = len(dividend)/len(precision\_base)$

$Recall\_for\_current\_instance = len(dividend)/len(recall\_base)$

append current recall and precision to recall[] and precision[]

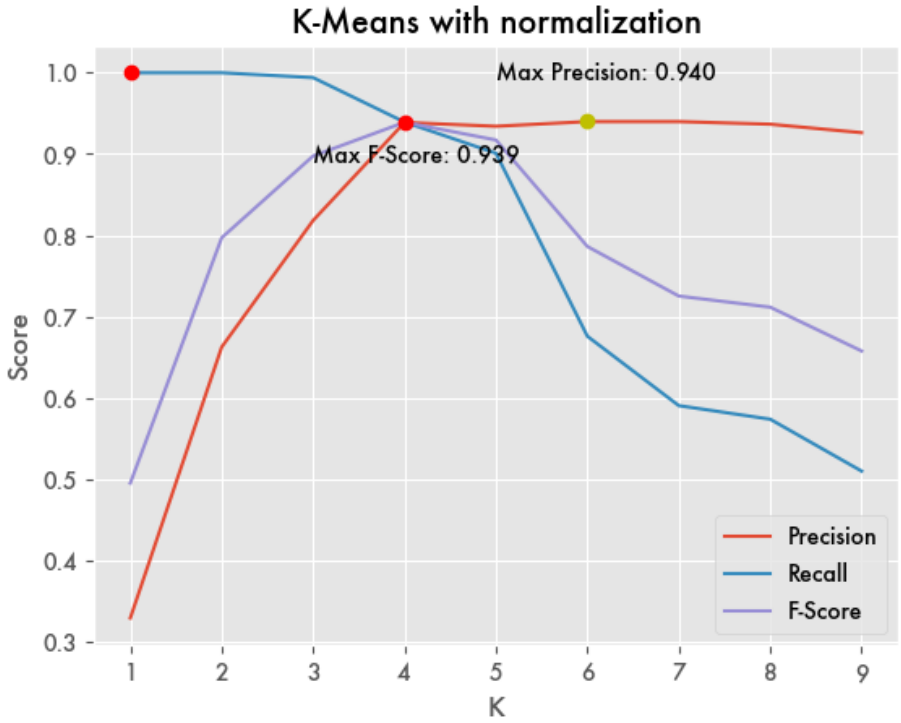
end for

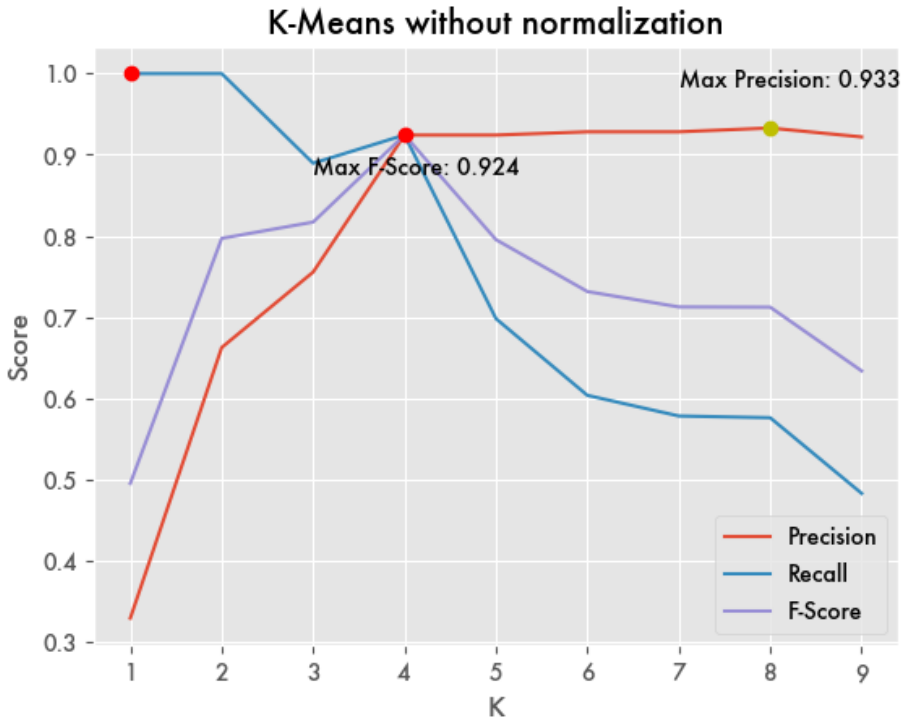
$precision = mean(precision\_list)$, $recall = mean(recall\_list)$,

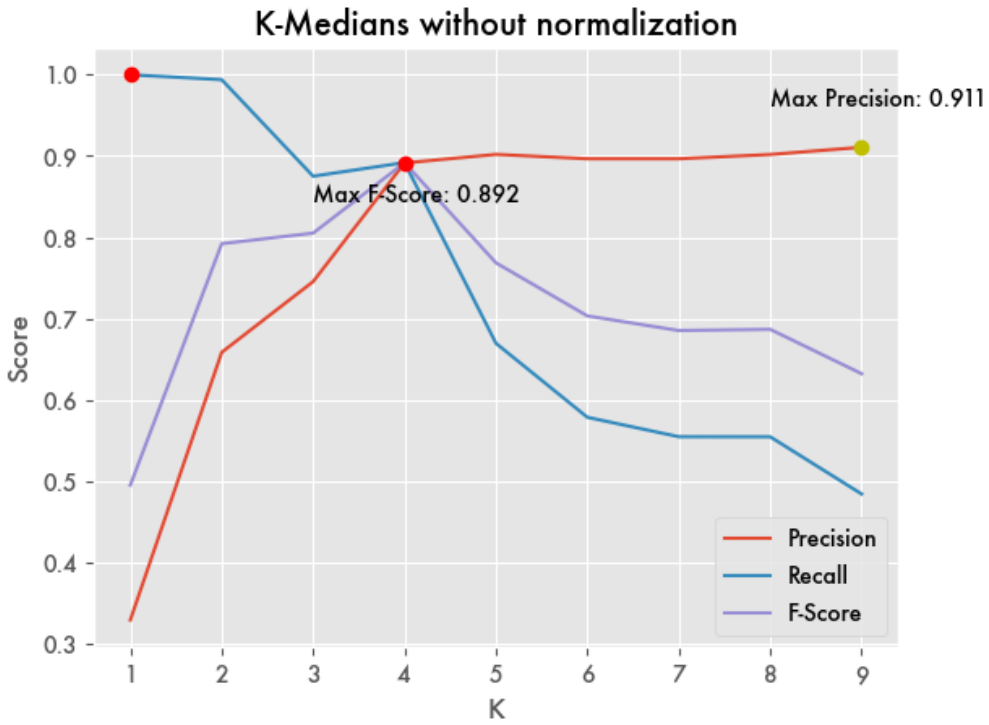
$f\_score = 2 * precision * recall/(precision + recall)$

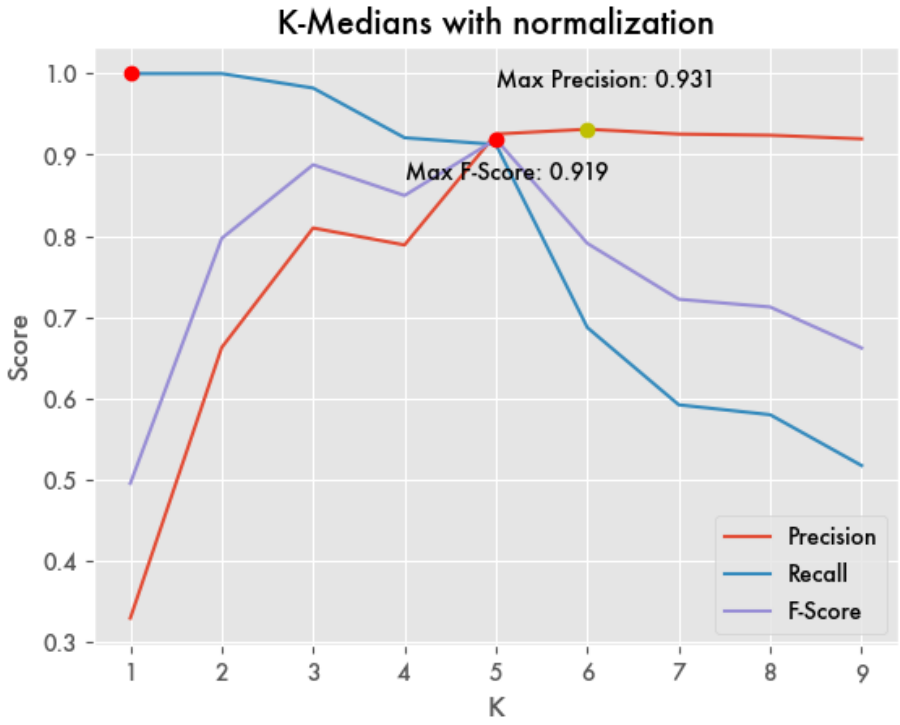
return $precision, recall, f\_score$

In this project, we evaluate 4 configurations(KMeans/KMedians and with/without normalization)for K in 1-9, and we plot the three evaluation scores by using k as the horizontal axis. The plot results are showing as follow:


K-Means with normalization
Max Precision: 0.940
Max F-Score: 0.939
Score
K
Precision
Recall
F-Score


K-Means without normalization
Max Precision: 0.933
Max F-Score: 0.924
Score
K
Precision
Recall
F-Score





Note: These figures is plotting by setting np.random.seed to 1

3. Comparison (Task 7)

Normalization:

For both K-Means and K-Median, using normalized can obtain higher Max Precision and Max F-Score, however, we can find that the best F-Score appears in $K = 5$ in K-Median with normalization but the best F-Score in other three algorithms is in $K = 4$, therefore, although normalization can increase the F-Score and Precision, for K-Median algorithm, the K-Median is likely to use more cluster to get better result which may be caused by L1-distance.

K-Means vs K-Median:

Both K-Means with and without normalization can perform better than K-Median with and without normalization, this may due to that for every instance there are 300 attributes, L2 distance are intend to catch more similarity between to vectors rather than some attributes that have large difference in L1 distance.

K-Value:

Due to that there are four classes in total, clustering algorithms except K-Median with normalization have found the best F-Score in $K = 4$ due to that the recall would rapidly decrease when K greater than 4, which means the some representatives cannot cluster most instances in one class, and the best F-Score K-Median with normalization is in $K = 5$.

Best Setting:

In my experiment, the best setting is K-Means with normalization with $K = 4$, the best $F\_Score = 0.939$. In this setting, the difference between max B-CUBED precision with F-Score and B-CUBED Recall with F-Score are also smaller than other settings.