

Data Classification

Implementing k-means and k-medians clustering algorithms

(COMP527)

***Assignment 2***

**Name:** Daniel Fox

**Student ID:** 201278002

**Student Email:** d.fox4@liverpool.ac.uk

**Date:**07/05/2021

Contents

[Question 1 3](#_Toc71271677)

[Step 1: How to sorting Data 3](#_Toc71271678)

[Step 2: K-Means algorithm: 5](#_Toc71271679)

[Question 2 8](#_Toc71271680)

[Question 3 9](#_Toc71271681)

[Step1: Introducing B-CUBED function: 9](#_Toc71271682)

[Step 2: Plotting functions 11](#_Toc71271683)

[Step 3: Results of K-Means 12](#_Toc71271684)

[Question 4 14](#_Toc71271685)

[Step 1: Introducing L2 Norm 14](#_Toc71271686)

[Step 2: Results of L2 Norm K-Means: 15](#_Toc71271687)

[Question 5 17](#_Toc71271688)

[Step 1: Results of K-Medians 17](#_Toc71271689)

[Question 6 19](#_Toc71271690)

[Step 1: Results of L2 Norm K-Medians 19](#_Toc71271691)

[Question 7 21](#_Toc71271692)

[Step 1: Current seed and data comparison for both K-means and K-Medians 21](#_Toc71271693)

[Step2: Different seed comparisons 22](#_Toc71271694)

[Comparing different seeds for K-Means 23](#_Toc71271695)

[Comparing different seeds for K-Median 23](#_Toc71271696)

[Overall 24](#_Toc71271697)

[Step 3: Error noted: 24](#_Toc71271698)

# Question 1

**(25 marks)** Implement the *k*-means clustering algorithm to cluster the instances into *k* clusters.

## Step 1: How to sorting Data

*Read each data file (animals,countries,fruits,veggies) and convert the data into numpy arrays.The data was read using the csv module to help parse the data.*

****

Figure :ParseData function

Above is the functions used to read each text file and combined the data into one numpy array. Which will then be used with the K clustering algorithm. Below shows the final dataset of the shape of size (329,302) . The function also adds a new category column to the data to help identify what data belongs to what file to help determined the B-Cubed calculations by taking the max amount of expected values for each data file.

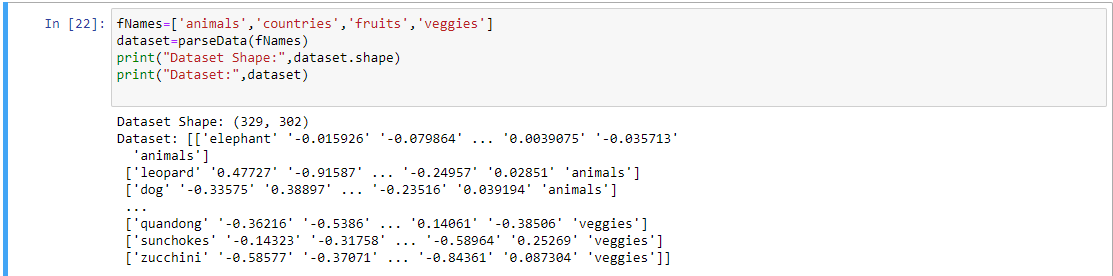
****

Figure :Printing Dataset

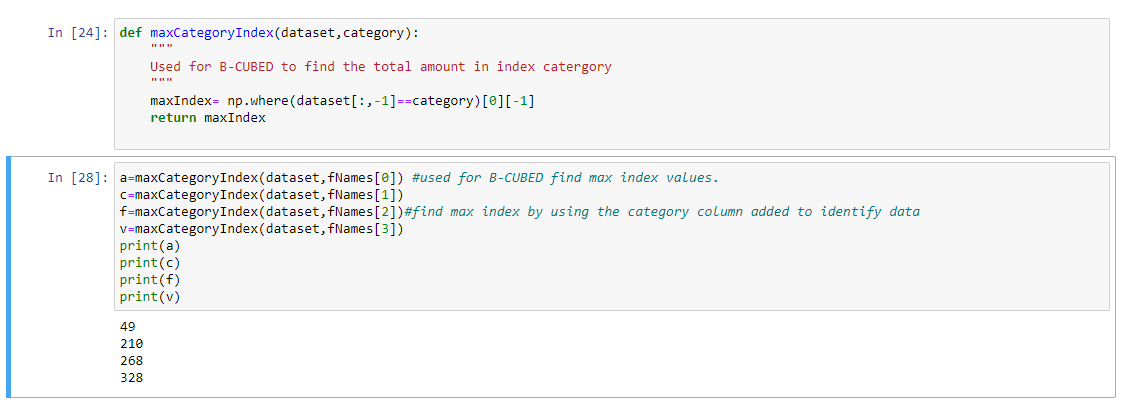


Figure :Finding max index for each category.

Above is the function used to find the max index of each category which was found by using the category column added in the last function to identify the data. After this is stored the column is deleted to pass the data in the K-clustering algorithm to be randomised.

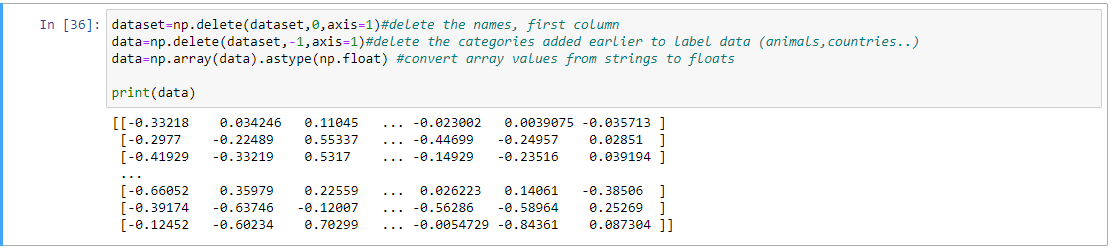
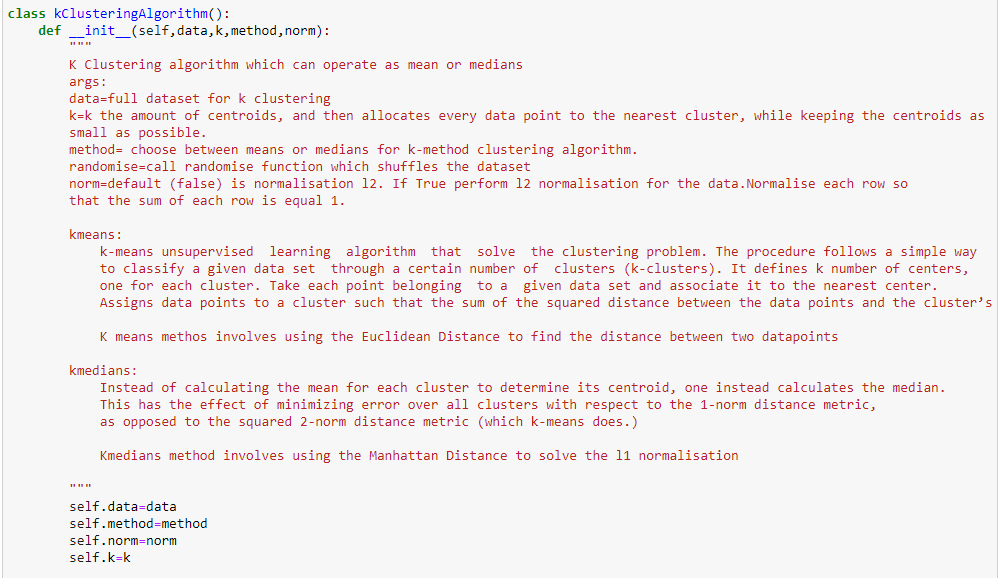
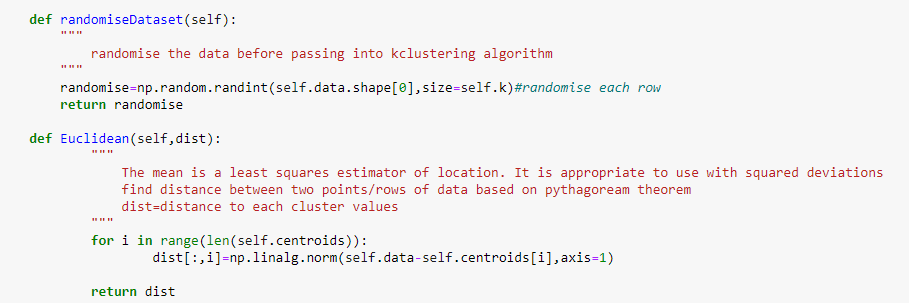


Figure :Removing characters and strings

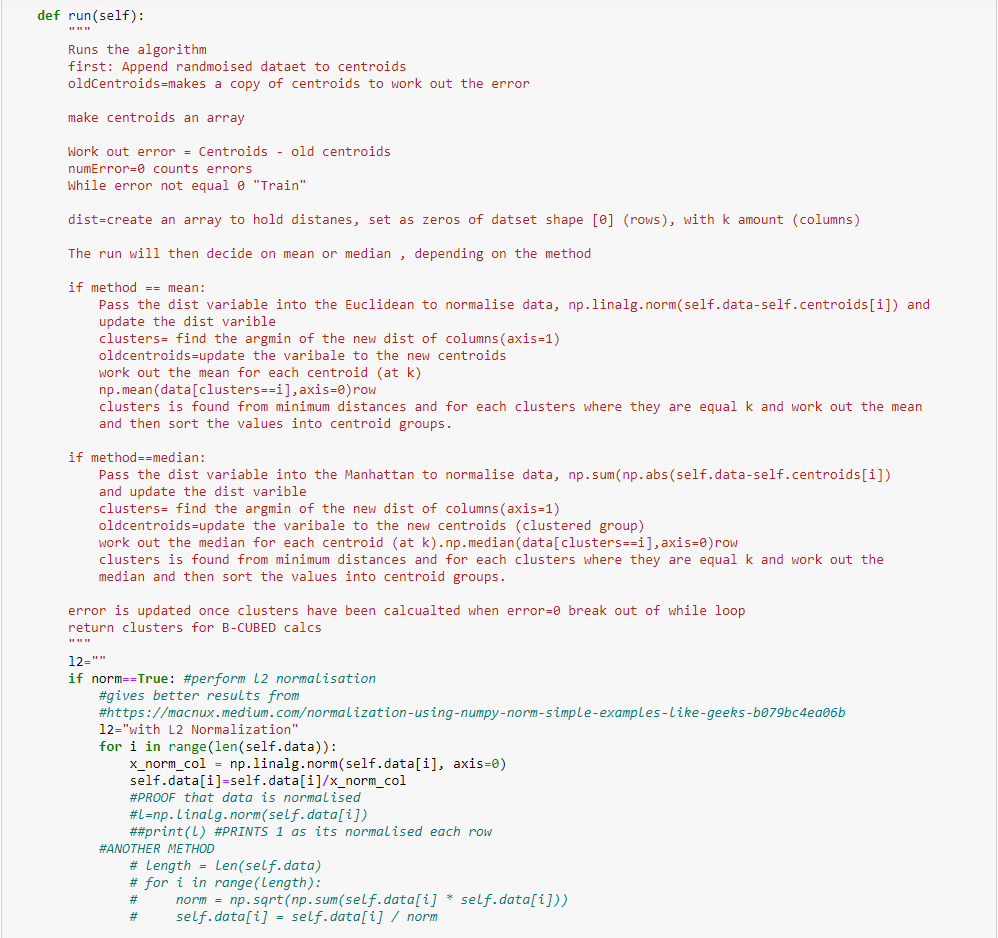
Deleting the string names and category names in the dataset and then convert the data into a numpy array of floats. This is then used to be passed into the algorithm.

## Step 2: K-Means algorithm:





The Euclidean distance method was used to calculate the k-means distances for each cluster. For KMedians it was best to use the Manhattan method to find each cluster.



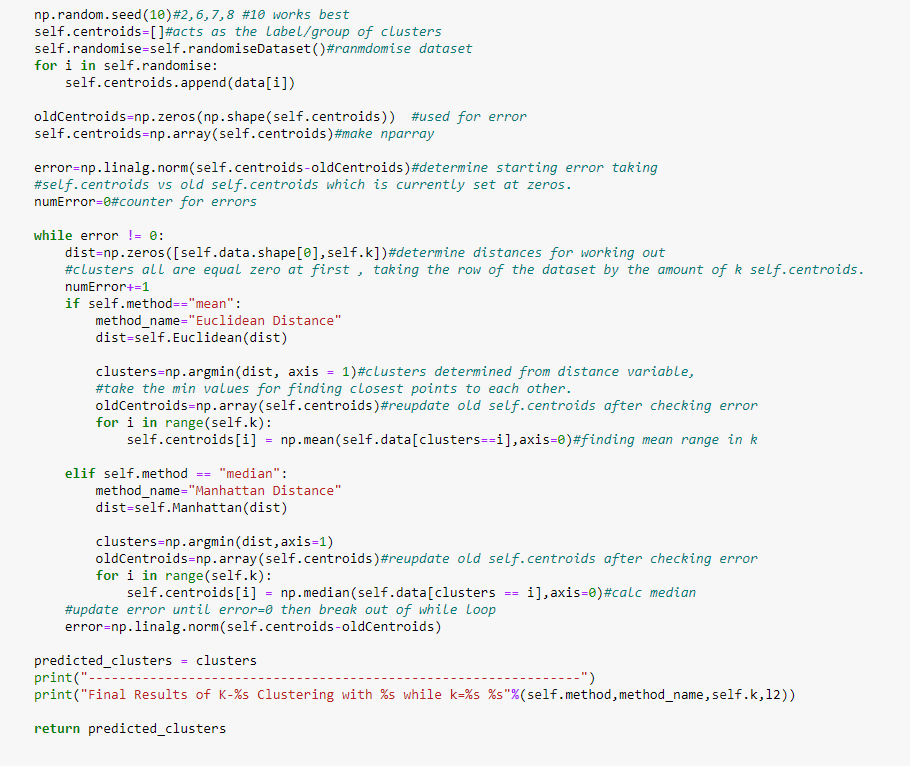


Figure :K-Clustering Algorithm

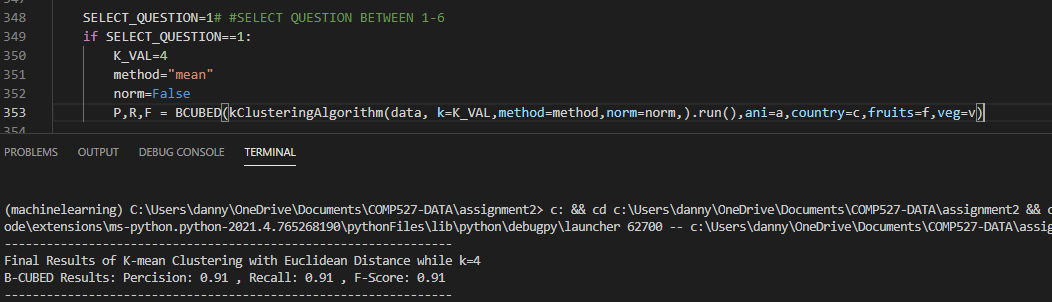


Figure :K-Means Results

Using random seed 10 to replicate results for all data.

# Question 2

**(25 marks)** Implement the *k*-medians clustering algorithm to cluster the instances into *k*

clusters.

The k-medians has been integrated into the KClusteringAlgorithm class so the user selects what type of algorithm they want, means or median. The median also used the Manhattan distance to work out the clusters.

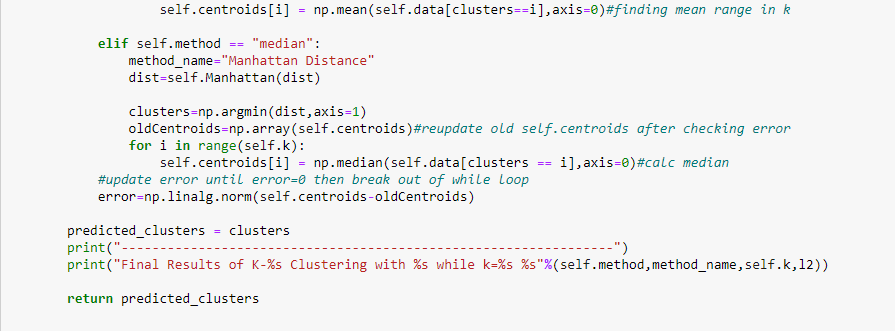


Figure :K-Clustering algorithm - Median method

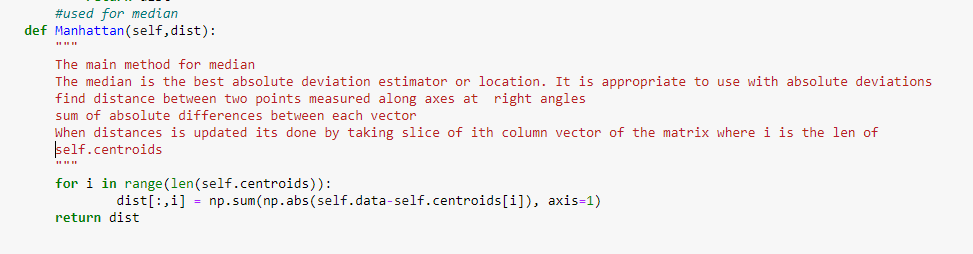


Figure :Manhattan distance for K-Median

Above shows that if the method variable has the value of median then it will perform the k-median clustering algorithm meanwhile if the method is select to be mean then it will perform the k means.

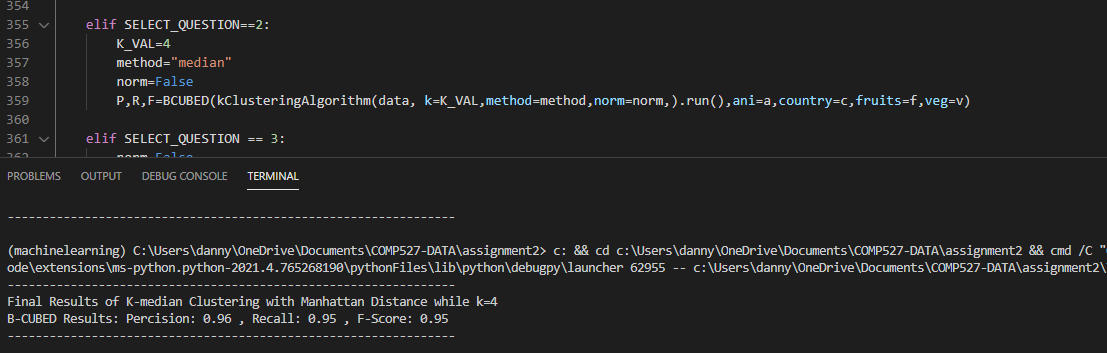
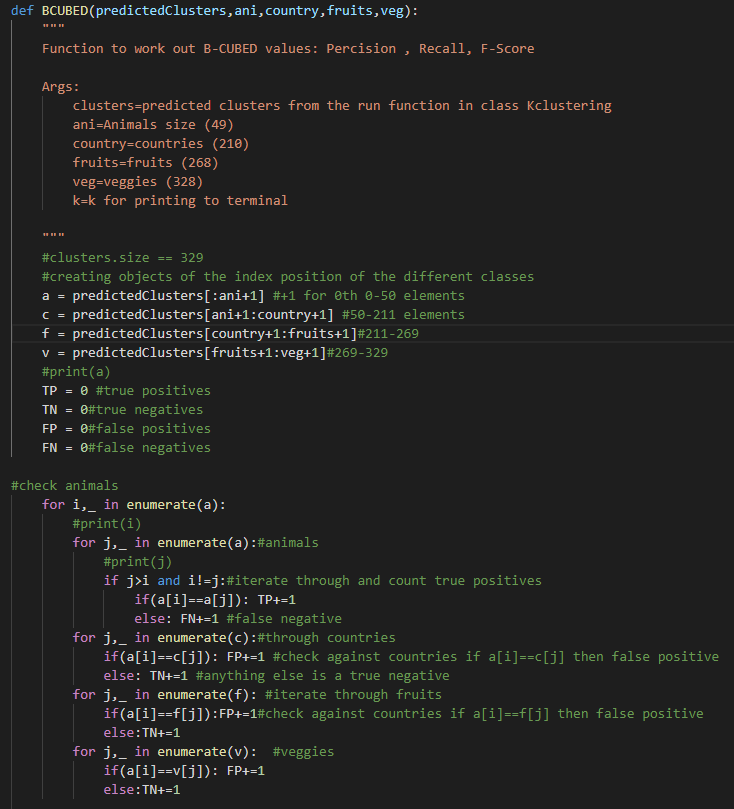


Figure :Printing K-Median results

# Question 3

***(10 marks)*** *Run the k-means clustering algorithm you implemented in part (1) to cluster the given instances. Vary the value of k from 1 to 9 and compute the B-CUBED precision, recall, and F-score for each set of clusters. Plot k in the horizontal axis and the B-CUBED precision, recall and F-score in the vertical axis in the same plot.*

## Step1: Introducing B-CUBED function:



*Continued …*

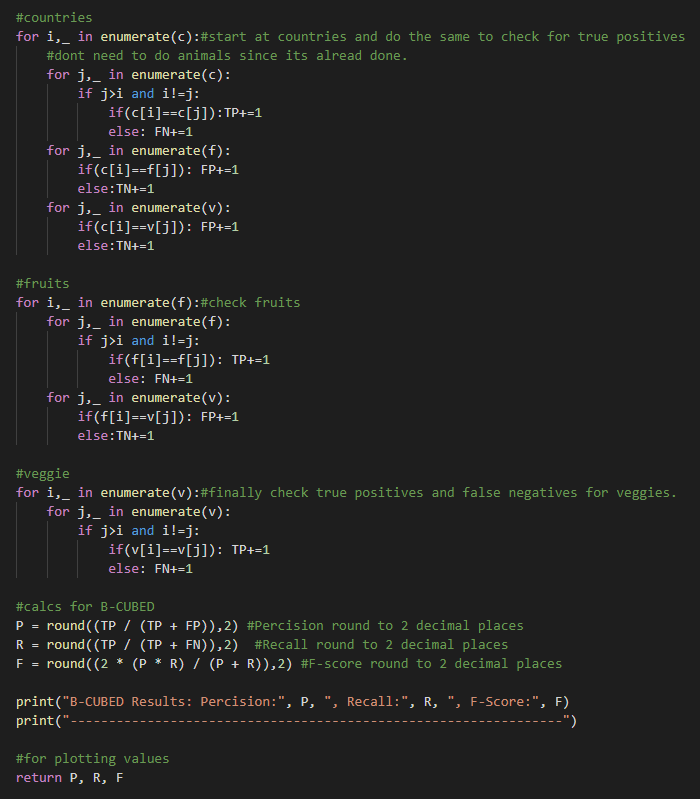
**

Figure :BCUBED Function

## Step 2: Plotting functions

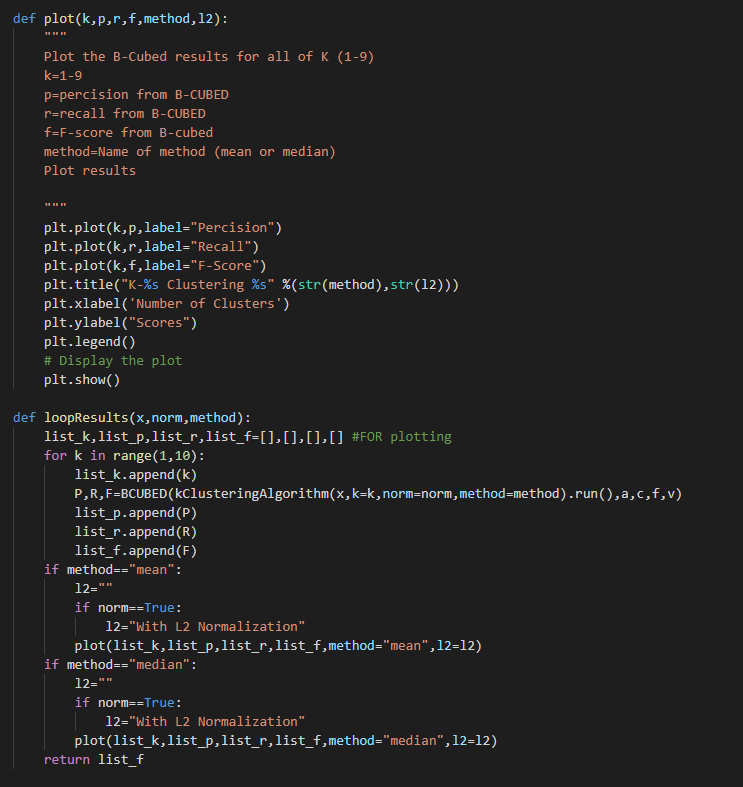


Figure :Plot BCUBED function

Above is showing the plot function which plots the B-CUBED results against the Scores and number of clusters used. The results function is called to be used for questions 3-6.Which creates lists to track the scores and each k used. K will have a value of 1 to 9 as stated in the question.

## Step 3: Results of K-Means

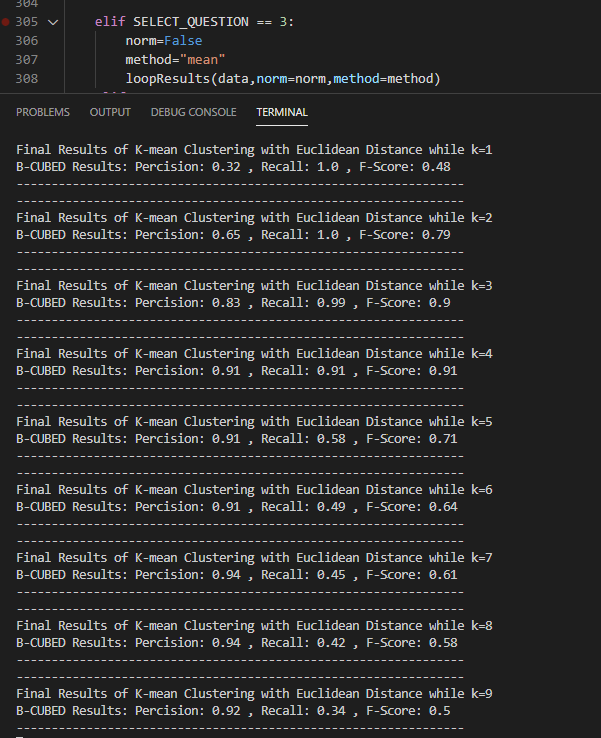


Figure :Results K-Means ranging k= 1 to 9

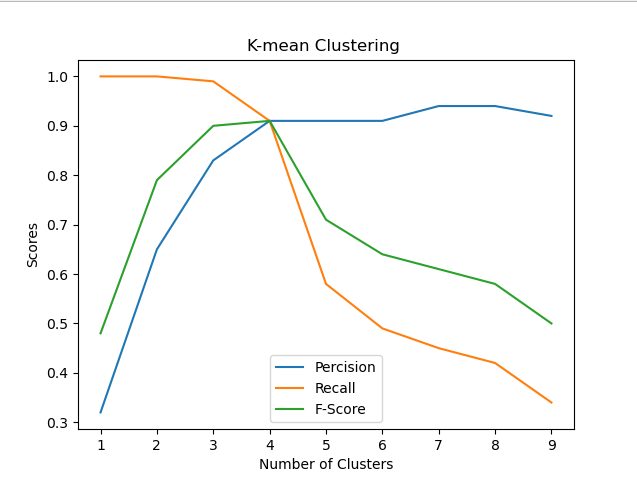


Figure :Graph of K-Mean results

# Question 4

***10 marks)*** *Now re-run the k-means clustering algorithm you implemented in part (1) but normalise each object (vector) to unit l2 length before clustering. Vary the value of k from 1 to 9 and compute the B-CUBED precision, recall, and F-score for each set of clusters. Plot k in the horizontal axis and the B-CUBED precision, recall and F-score in the vertical axis in the same plot.*

## Step 1: Introducing L2 Norm

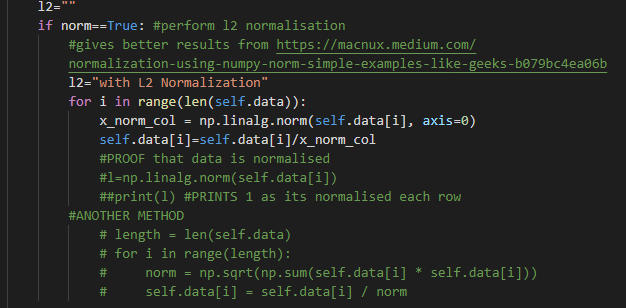


Figure :L2 Normalization implementation

The normalisation is performed as shown above which is at the initialisation of the k clustering class. If norm is set to true, then l2 normalisation will be performed before clustering.

## Step 2: Results of L2 Norm K-Means:

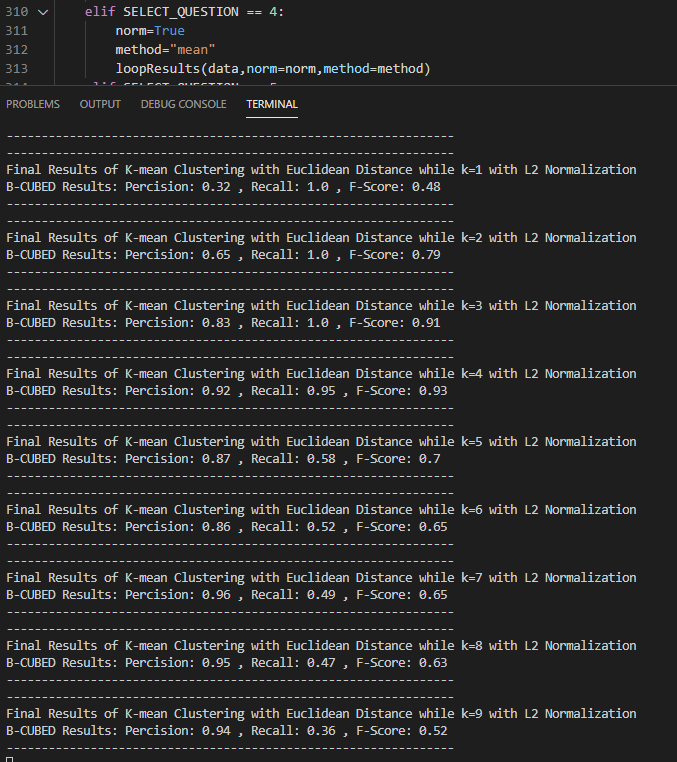


Figure :K-Means L2 Normalised Resutls k=1to9

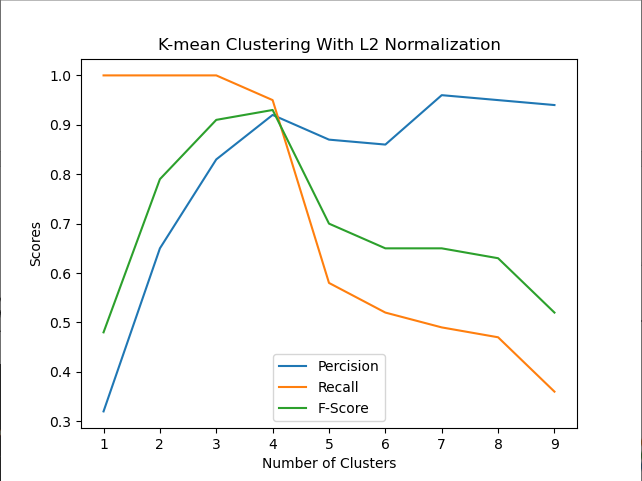


Figure :K-Means clustering graph while L2 Normalised

# Question 5

***(10 marks)*** *Run the k-medians clustering algorithm you implemented in part (2) over the unnormalised objects. Vary the value of k from 1 to 9 and compute the B-CUBED precision, recall, and F-score for each set of clusters. Plot k in the horizontal axis and the B-CUBED precision, recall and F-score in the vertical axis in the same plot.*

## Step 1: Results of K-Medians

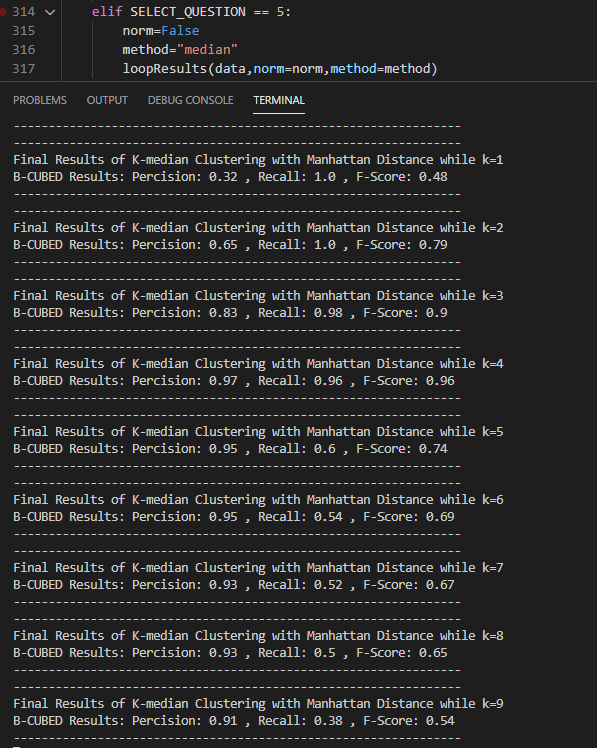


Figure :Results of K-Medians while k=1to9

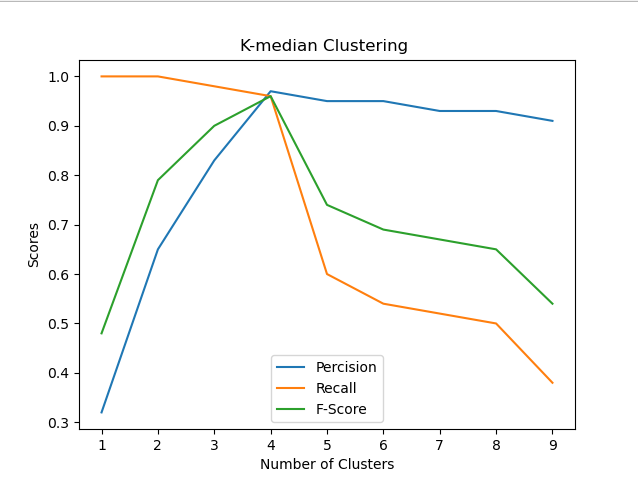


Figure :K-Median Clustering graph

# Question 6

***(10 marks)*** *Now re-run the k-medians clustering algorithm you implemented in part (2) but normalise each object (vector) to unit l2 length before clustering. Vary the value of k from 1 to 9 and compute the B-CUBED precision, recall, and F-score for each set of clusters. Plot k in the horizontal axis and the B-CUBED precision, recall and F-score in the vertical axis in the same plot.*

## Step 1: Results of L2 Norm K-Medians

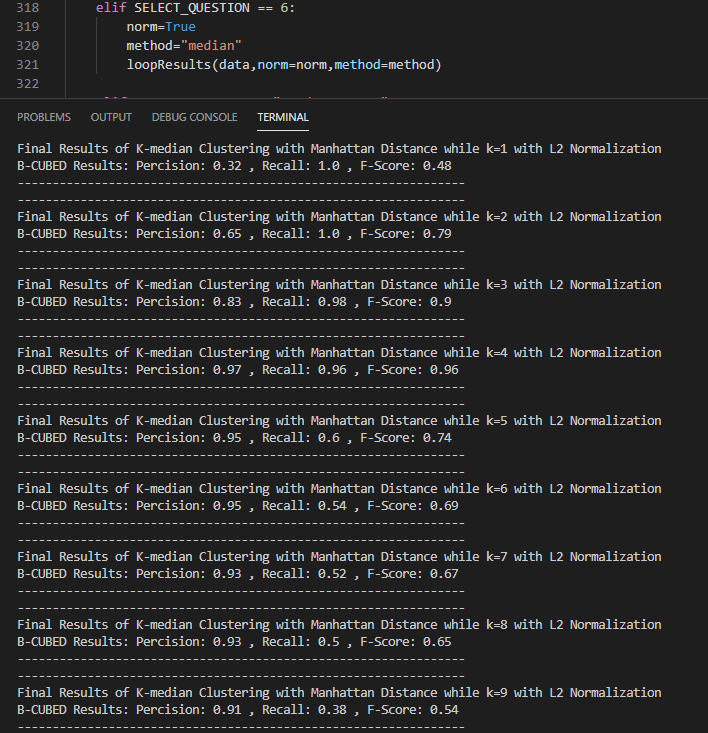


Figure :K-Median scores while L2 Normalised k=1to 9

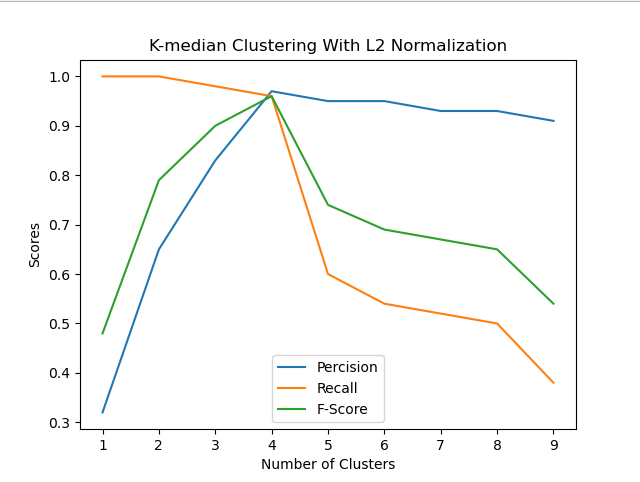


Figure :K-Median graph L2 Normalised

# Question 7

***(10 marks)*** *Comparing the different clusterings you obtained in (3)-(6), discuss in which setting you obtained best clustering for this dataset.*

## Step 1: Current seed and data comparison for both K-means and K-Medians



Figure :Pandas used to print comparison of F-Scores

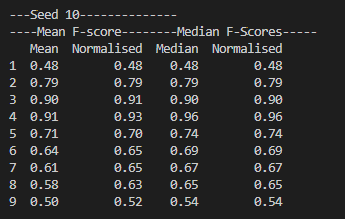
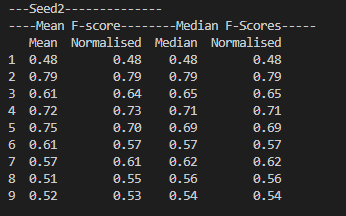
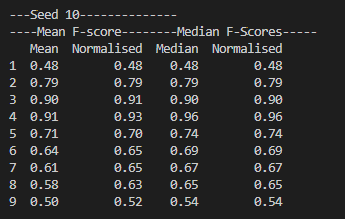


Figure :Comparrison of F-Scores for K-means at seed 10 , K-medians and l2 normalisation for k=1-9

After determining the best k clustering values using the BCUBED method for both algorithms while unnormalized and l2 normalised the results were compared by reviewing the overall F-Score of each cluster ranging from k=1 to 9. When analysing the data the best scores relating to the B-CUBED F-Score calculations are typically when k = 4 this makes sense due to knowing the data should be divided into 4 categories(animals,countries,fruits,veggies).

The best F-score can be shown above, using seed(10) and k=4 for both methods have similar high scores above 0.90 for each. The K-means slightly underperforms the K-Medians algorithm in this case above as the K median has a score of 0.96 whereas the K mean scores 0.91 unnormalized and 0.93 l2 normalised. The K-Median has the highest score out of both algorithms while being at 0.96 for both unnormalized and normalised. We noticed that the median tends to have a higher score for k=4 while normalised and unnormalized however when we further vary between different seeds, we notice that the median F-score can largely be affected and more often than not the K-Means with l2 normalisation performs the best with the highest f-score.

## Step2: Different seed comparisons



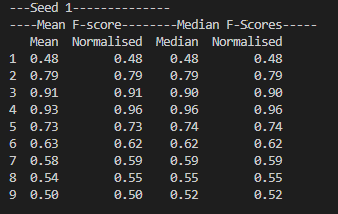
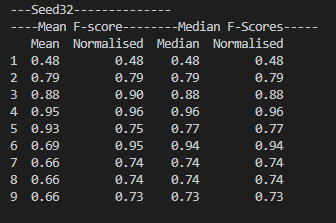
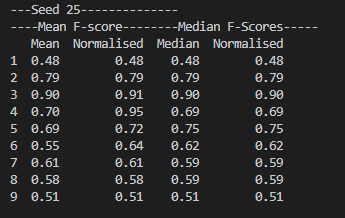
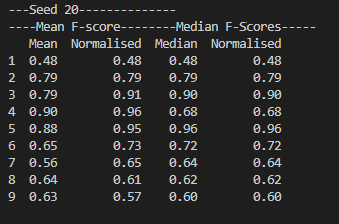
 

Figure :Seed Comparisons of F-Scores k=1to9

When we analysie multiple variations of different seeds we notice that the results can vary largly depending on the randomness of the seed. We can see our seed 10 which is the main seed used for the questions in this report shows that the K-Medians algorithm is best suited for the value of K being equal to 4 or the K-Means l2 Normalised as they all have the same F-Scores. Since we assume 4 is the most optimal K clustering value due to having 4 data classes animals,countries,fruits an veggies. As the K clusters increases the number of centroids the score drops this seems normal as the algorithm will find it difficuilt to differentiate a higher number of k clusters.

When seeds 1 and 32 are set we notice the same results as seed 10 however even with these seeds the F-score’s are similar for all cases apart from K-mean unnormalised we can see that for seed 1 K-means at 4 has value of 0.93 and a normalised value of 0.96, so the best choice would be to normilse the data for k-means. When we look at the K-median for the same seed both of the values are equal 0.96. So in this case either k-means normalised or both versions of K-Medians provides the best F-scores.

### Comparing different seeds for K-Means

What can be seen above apart from the k=4 mean result being the highest on average is in fact that as the data is normalised using the l2 method, we can see that while the k increases past 4 and 5 we notice the data typically stays consistent in comparrison to the unnormalised data, even with the randomness of the seeds it shows that the data remains similar when normalised and often has a higher F-score than the unnormalised data. Typically the normalised k-means always has the better highest F-score out of the rest of the data.

We also notice the score at k=3 seems to hold a high score on some seeds , even as the kmeans is normalised more often when k=3 the score is near 0.90 this is most likely due to it being easier to differentiate between the data using 3 clustered centroids for our dataset. When k passes 4 it should become harder too differentiate due to there being 4 datasets , so most likely the data should be worse when increasing above k = 5 since the k clusters centroids group the data making the algorithm difficuilt to group the datasets together. While when k is less than 4 it should be easier to determine errors.

### Comparing different seeds for K-Median

The data above shows the F-scores found from using the K-Medians method with the Manhattan distance over time. However in regards to K perfoming best at 4 this is not correct for this case, it can be clearly shown on average that K=4 isnt always the best perofming cluster when it comes to the K-median. This is mainly because of instead of calculating the mean of each cluster to determine the centroid it uses the median for each cluster which minimizes the error using 1 norm distance as opposed to squaring it like L2-norm mean. The K-median can run into problems determining clusters using the 1-norm as it struggles to find centers for the centroids as they are more compact. We can see that for seeds 2,20 and 25 when the seed has a low value for k=4 then the highest F-score shows to be typically when k=2 for seed 2 at 0.79, seed 20 and 25 both have the same highest F-score being at K=3 at 0.90. What we notice is that typically the L2 Normalised K-Means still holds the highest score at k=4 out of all the datasets.

In general its best to use K-Median to help identify outliers in the dataset as Mean can massively effect outliers, as an example if the data points where [1,2,3,5,99] its clear that the outlier is 99. The median of the data if 3 whereas the mean would be 22.

### Overall

When analysising seeds 20 and 25 we notice that the value of the F-score has dramitaccly changed apart from the K-mean normlisation value. Which can further show that we can assume the K-Mean l2 normalised holds the best F-score on average as the value is consistently high when varying seeds, at seed 20 and 25 is 0.96 and 0.95 respectively, with this data we can assume that the normalisation has a beneficial effect to the K-means algorithm when the data is randomly choosen. On average using the seeds provided for comparrison the K-Means with L2 Normalisation performs the best. However due to the randomness of choosing our starting points for each centroid there is a large amount of other possibilities where if k is equal 4 centroids out of 329 data points then there is of having different starting positions for the k clustering. So if the algorithm would be used it would most likely work best at K-means l2 Normalised however there is a likelyhood that the randomised data points are not well positioned throughout the entire dataset so there is also a chance that the K-medians could also be another optimal solutioin.

## Step 3: Error noted:

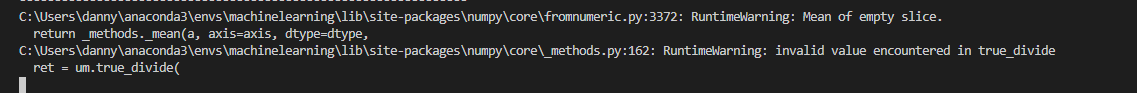


Figure :Notable error found

Note: When changing to other seeds errors occur from the normalised array of data due to the randomness and seed used. Unsure how to fix the error I assume its due to the randomness of the seed. If I ran the data individually most of the time it would compute the results.