MAITREYI PITALE
G01326362

CLUSTERING
HW3

Username : mpitale
PART 1
Rank on miner : 72
Score on miner : 0.90
PART 2
Rank on miner : 64
Score on miner : 0.78

Introduction :

The main goal of this project is to implement K Means algorithm using Iris dataset. As K means is an unsupervised algorithm , no prediction variables are present. Overview of K means is that it finds patterns in the data and will take each data point that is the centroid randomly in the data which will in turn move the centroid of each cluster. The steps will continue until there is no further is no further variation in the centroids.

Dataset Analysis:

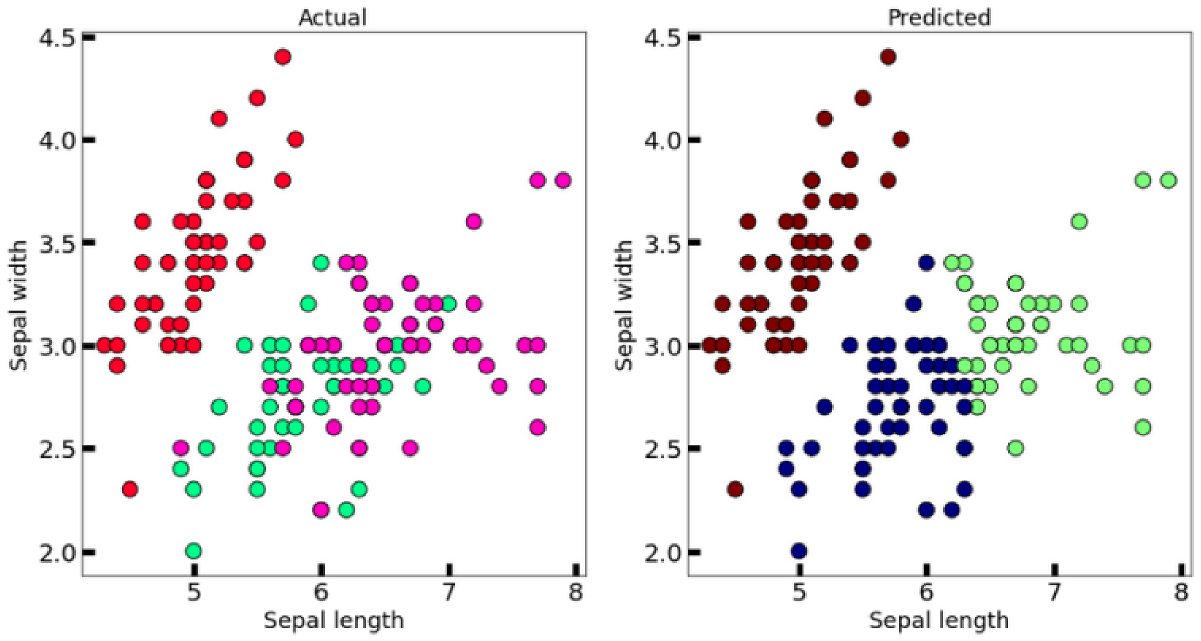
PART 1 -IRIS CLUSTERING
The Iris datasets consists of 3 types of irises - setosa, versicolor and virginica, for petal and sepal length stored in 150*4 numpy array.The rows being the samples and the columns being: Sepal Length, Sepal Width, Petal Length and Petal Width. Using the Elbow method and Inertia which is the sum of squared distances of the samples to their closest cluster centre, I tried to find an Optimal number of clusters required to segment the observations.

PART 2 - IMAGE CLUSTERING
The dataset consists of 784 comma-delimited integers where each row is a record. As the dataset is an image dataset , it requires normalisation and preprocessing.
Preprocessing of dataset includes: PCA and TSNE
Pca and TSNE are responsible for dimensionality reduction and tSNE is computationally expensive but can capture exact features. Also ,Principal component analysis (pca) converts n-dimensions into k-dimensions to perform dimension reductions.

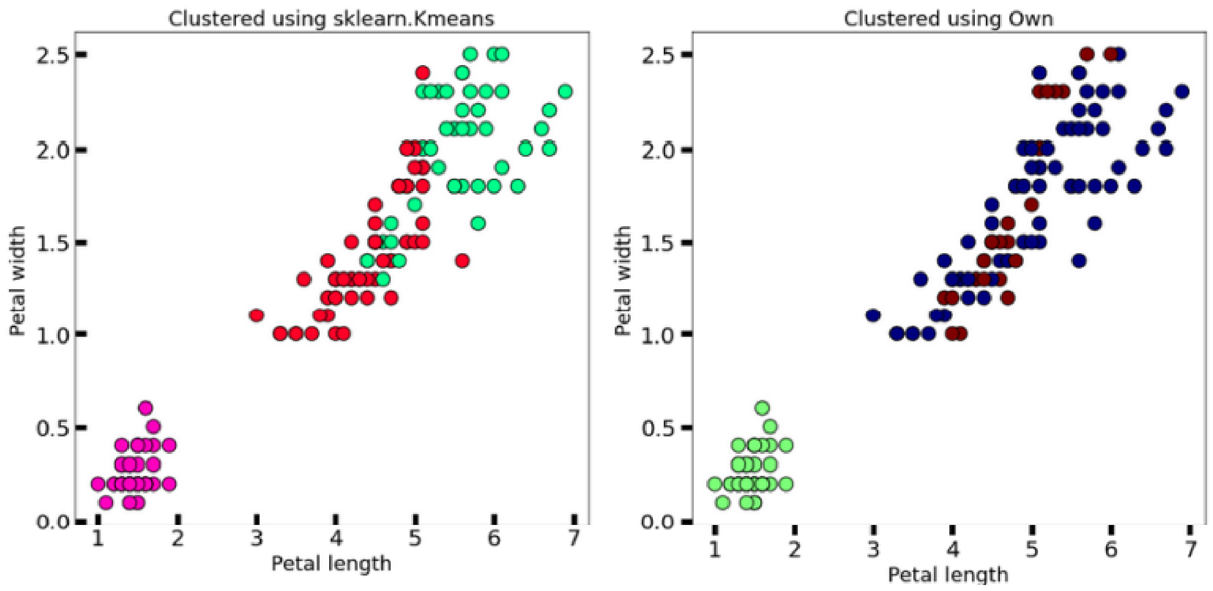
Work Approach

Part 1

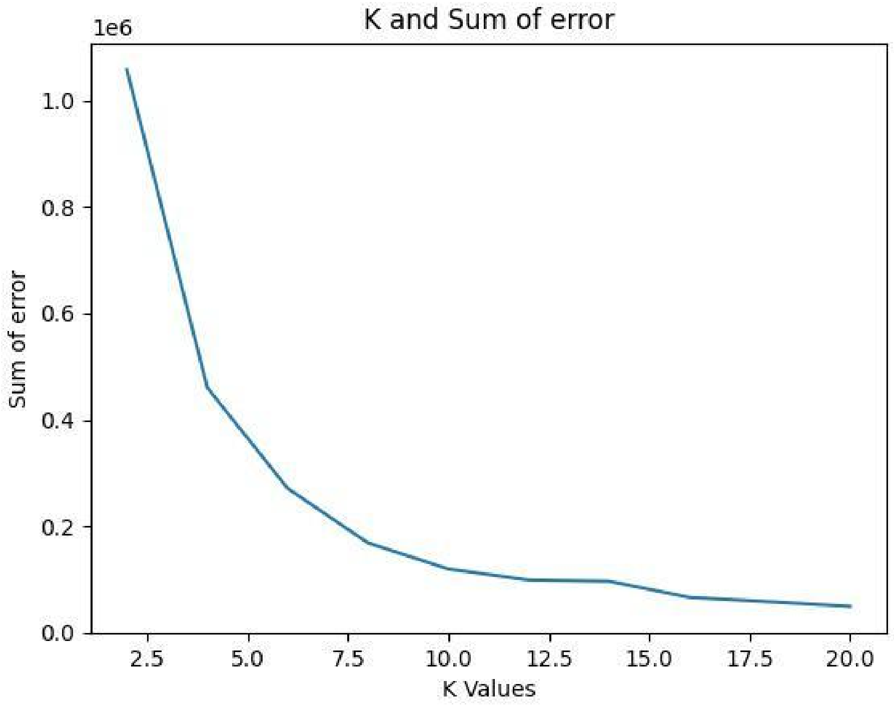
The problem statement is to develop K means Clustering algorithm for iris dataset provided by the professor and cluster the data accordingly. First while approaching the problem, step 1 is to preprocess the data, for this step read the data from the file and convert them into a numpy array for ease. For implementing the K means algorithm at the initial stage we need to assign random labels of centroids around which the clusters can be formed. The random centroids can be generated using random method from numpy package. And tried to implement K means by finding the mean of each point vector and the random centroid formed . Once the algorithm reaches the minimum deviation in centers the clusters will be finalized. By using this directly resulted in a score of 0.90 in minor, After this I tried to check the dataset loading from the sklearn package. I tried implementing the k means from the sklearn package and obtained the below results.



After this I tried to implement the dimensional reduction technique, t- Distributed Stochastic Neighbor Embedding (t-SNE) which calculates the probability similarity of points in high dimensional space as well as in low dimensional space. Once after applying to TSNE I used the data to start clustering. And I compared the clusters formed with my own Kmeans and Kmeans from sklearn for petal length and petal width.



The elbow graph is responsible for visualization of our work of clustering and testing the clusters .I also tried to find the optimal k value by using the elbow graph method and tried to check the v measure scores . Here the below figure gives us the optimal k value to be 3



PART 2:
The input data consists of 10000 rows each consisting of 1x784 vectors.Initial step is to preprocess the data, read the data using pandas read_csv and use separator and put into a dataframe. Try to visualize the data, remove the unnecessary columns and normalise the data. I used PCA for dimensionality reduction to help extract a new set of variables from the existing large set. The same algorithm for the iris dataset is to be used for this image dataset which resulted in an accuracy of 0.55. To further improve the accuracy I tried importing t-SNE for data exploration and visualizing high dimensional data. Basically, to reduce redundant features I used TSNE as well as UMAP which is Approximation and Projection Mapping and got the final results to be 0.78.
Preprocessing :
For the image dataset, clearly, we had to consider more than 10 clusters, since there is more than one possible way to handwrite a particular digit. The first approach is to run the k means algorithm for only 10 clusters which seemed to be incorrect most of the times. This was verified by plotting the initial vectors and viewing the output. Many numbers were repeated. Initial Centroids when k = 10 Running the k means algorithm in this case would give incorrect results, and only 53% of the clusters were identified correctly using this method. To improve this score, an alternative approach had to be adopted. The other approach was to create more than 10 clusters, and then find similar clusters so that they can be grouped or merged so that the resultant was 10 clusters. Multiple variations were tried out. Including 32, 64 and 128 clusters. Optimal performance was found at 64 clusters.

Post processing : Once the clusters were found, the next step was to merge these clusters so we could settle onto 10 final clusters. This involved finding clusters close to each other, and then renaming the cluster id label of all close clusters to one single label between 0 - 10, basically renaming multiple clusters to a single value. My particular approach to do this was to select one datapoint from each cluster and plot it onto a new dimensional space. Then a custom logic was written, to keep checking for close data points for each datapoint in this dimensional space. A small proximity radius was decided, where for each datapoint, if there existed other data points in the proximity radius, then they would be merged. To do this, an optimal proximity radius had to be decided, therefore this process was executed in multiple iterations where the proximity radius slowly increased, so that each of the points would be merged into 10 clusters exactly.

Conclusion:

The K-means algorithm was completely understood, however the implementation has lots of scope for improvement in this project. The final result uploaded to miner does not take post-processing into consideration due to the poor merging logic.The result after some time started to overfit and started to decrease the accuracy rate, this thing needs a bit improvement according to me. For real life datasets, a lot of the final model performance highly depends on the initial centroid selection and the lengths at which post processing is carried out and for that part there is a lot of randomness which can be reduced by using specific methods and that k means model can become more specific and accurate.