IRIS and Image Clustering

REQUIRED INFORMATION

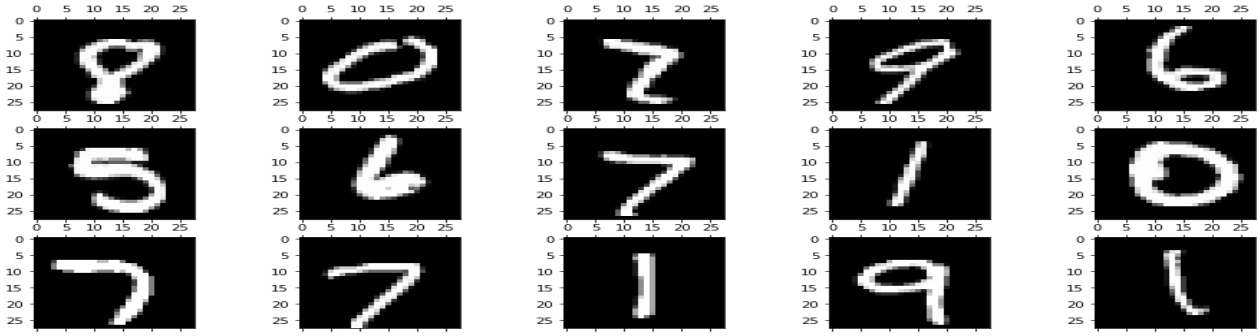
Mason User Id: jbidhan Miner2 Username: flash
Ranks & V scores HW3-IRIS:29, 0.76
Ranks & V scores HW3-Image: 3, 0.81

DESCRIPTION OF APPROACH:

I approached the problem reading the question and making notes for the steps required. The approach I used can be summarized as below:

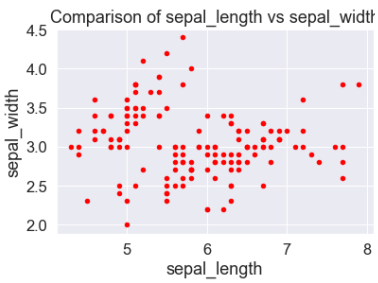
I implemented the K means algorithm and algorithm for initializing centroid as a common code for IRIS dataset to start with and then created different methods for doing various tasks. I created the other file parallel to review the data and understand what techniques are required.

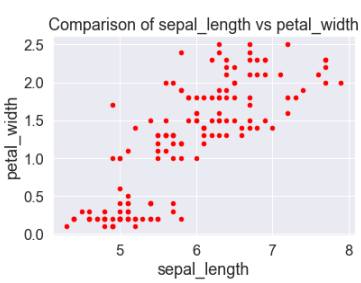
1. **Reading and storing the Training file** using pandas.read_csv : I created method (function: readtestfile()) which needed a parameter for filepath to read the input. Using pandas.read_csv() function read the file and stored as a pandas' DataFrame having column names: ['sepal_length', 'sepal_width','petal_length','petal_width'] for IRIS Dataset and without any column for image data and post that I plot the data to see how the data looks for both the datasets.

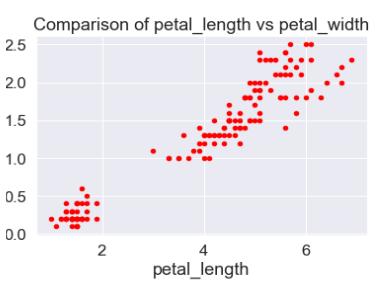


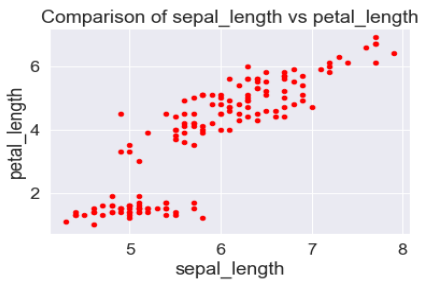
2. **Preprocessing of the data**: I normalized the data and tried dimensionality reduction techniques like PCA, IncrementalPCA, TruncatedSVD and TSNE over the data.
3. **Clustering**: I wrote the logic for initializing clusters from the preprocessed data and then passing it to the k means algorithm and post did some plotting of the data over various metrics and experiments. Flow- evaluate_k_means() is the method that receives the preprocessed data, k value, distance metric and some charting/output attributes. Then, using initailzingCentroid() method, receives the centroid which then feeds the centroid, data, k, and other parameters to K_Means_Algo() which performs to core logic of the algorithm using various internal methods like cluster_change(), update_centroid() and calculate_SSE(); it returns the 'Final Centroids','Clusters','Cluster Assignment', 'Sum of squared errors' and 'runs". This data is used to plot various graphs like the elbow method graph (sse for different k values) or the clustering graph.
4. **Saving the output:** Then using the saveOutput() store the output in the txt file and upload on the miner portal to get the results and then compared which of the following techniques and methods fared well.

*The library modules which were used were wrapped for modularity and the sake of reusability.



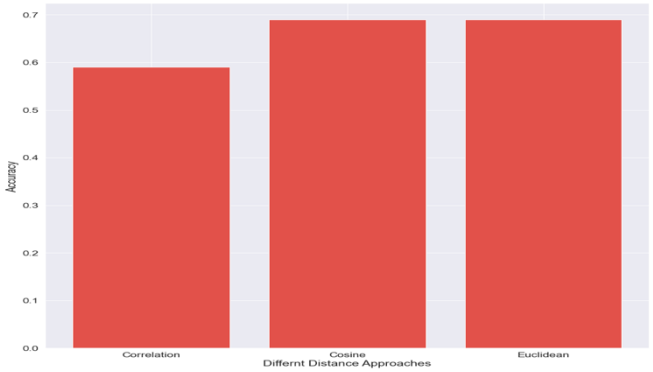






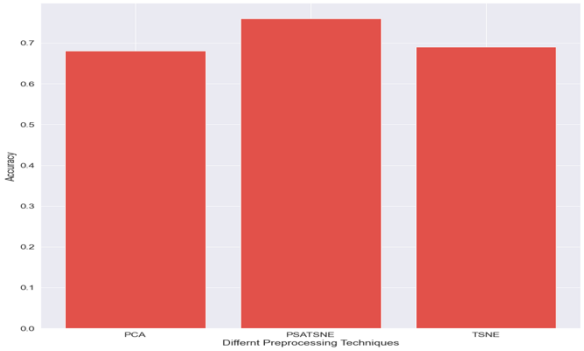
Methodology of Choosing the Approach and Associated Parameters and Graphs

I started writing the code on Jupyter notebook and created two files, one for IRIS Dataset and other for Image Clustering. I tried running the data directly without cleaning, the accuracy was not great and was around 0.37 and then Dimensionality Reduction techniques it improved a lot. There were several factors involved in choosing the approach which are answered in the following points



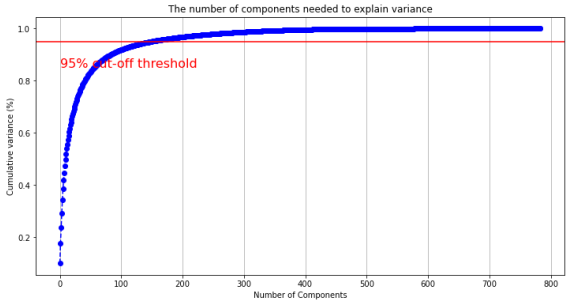
a. **Distance Metric:** I tried with different distance metrics and found that cosine and Euclidean fared much better in terms of V-score but the no of iterations for Euclidean were way less. So, I went on taking Euclidean as the distance metric. (IRIS Dataset).

b. **Preprocessing of Data:**

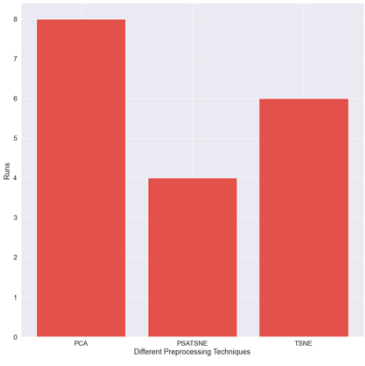


i. **Normalization of Data:** To modify features, I used MinMaxScaler from the sklearn.preprocessing module, which scaled each feature to a range between zero and ones. This estimator scales and transforms each feature separately so that it falls inside the set's range of zero to one. (For both the datasets). I also checked for null values in the dataset but there were no such records.

ii. **Dimensionality Reduction:** I intended to decrease the number of variables, but I was able to discover several that could be totally ignored. I also wanted to make sure that variables are unrelated to one another and that independent variables are more difficult to comprehend. So, I used Principal component analysis (PCA) for Linear Dimensionality reduction. PCA is a method for lowering the number of dimensions in a dataset while keeping most of the data. It makes use of the correlation between some dimensions and seeks to give the smallest number of variables while retaining the most variety or information about the original data's distribution. I set my variance to around 85%, which resulted in a decrease in the number of features from 784 to 55 for the image data. The data from PSA was then fed to T-Distributed Stochastic Neighboring Entities (TSNE) which is another dimensionality reduction technique, it is particularly well adapted to the display of high-dimensional datasets. I performed both one after another to reduce the no of features to 2. This data was ready to be fed to the K-Means algorithm. Prior to this, I tried them individually but did not receive a great score for the either of the datasets. The following graph shows the results obtained for each of the techniques but combining both turned the V-score to increase to 0.76 (IRIS). I tried other techniques as well but none of them fared well.



c. **Implementation of K means Algorithm:**



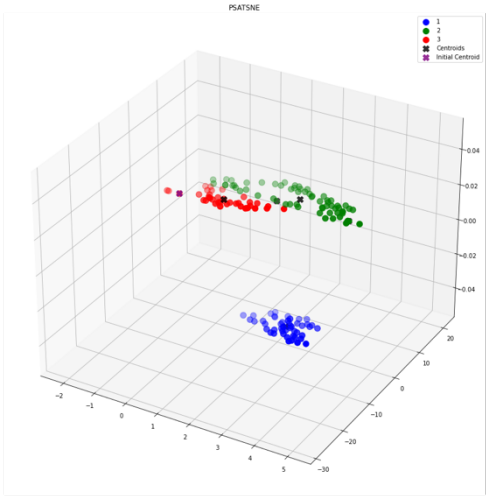
i. **Initializing the initial centroids:** My centroid initialization technique is extremely basic, and it is based on the computation of a composite summation value that represents all the instance's attribute values. Once this composite value has been calculated, it is utilized to order the dataset's occurrences. The dataset is then sharded horizontally into k parts. Finally, the original attributes of each shard are added together, the mean is calculated, and the resultant collection of rows of shard attribute mean values is utilized to create the set of centroids for initialization and I also used a method to set 'k' random centroid using the NumPy library.

ii. **No of iterations:** On analysis of different dimensionality reduction techniques the minimum no of runs was found in PSA output fed to TSNE. The graph shows the details.

iii. K means Algorithm: I created a pseudocode initially to reflect what flow needs to be followed. First step was to calculate the distance from the initial 'k' centroid set in the previous step for each of the record. Then, the distance from each of them was stored in the map with the index of each entry and a cluster(number) was assigned to each one of them. After this step, I had to check if change was required for the centroids and if yes then centroids were reset using the mean of the data points in assigned to a particular cluster(number) and until there was no change in the centroids the loop went on. At the end, I calculated SSE for each of the iteration and returned that, so that it could be used in understanding the algorithms better. Also, the classification was returned after adding 1 to it. Since, the array index started from 0. The classification was done as 0-9 in case of image dataset and 0-2 in case of IRIS dataset.

Important Graphs:

IRIS Data Set:



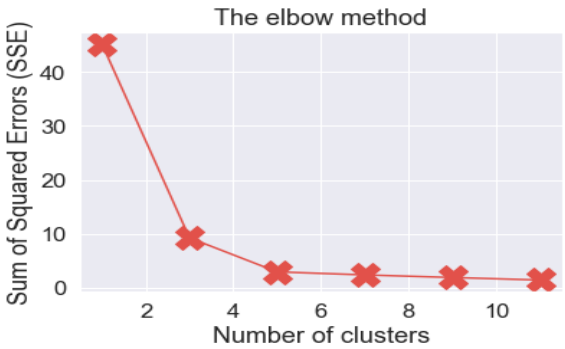
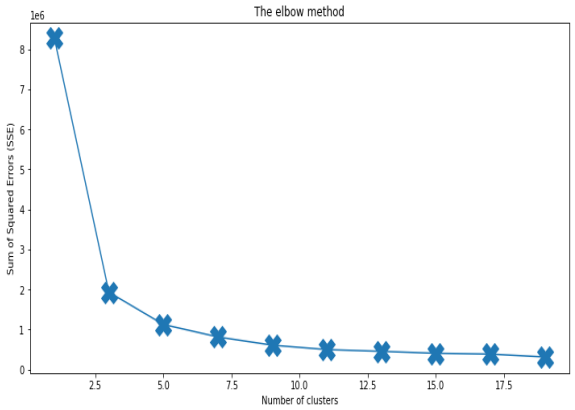
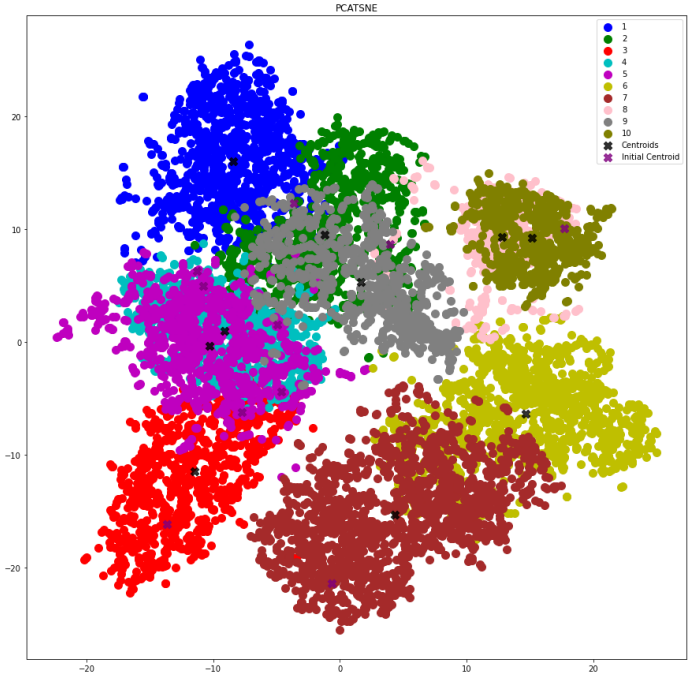


Image Classification Data Set:





CONCLUSION

On seeing the results of the various experiments on various attributes, distance measures on the K-Means implementation gave v-score of 0.81 for Image Classification and 0.76 for IRIS data. Result files are in the output folder.

REFERENCES:

Documentation from https://scikit-learn.org/ , https://www.nltk.org/, https://www.scipy.org , https://matplotlib.org/

Blogs:https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c, https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b