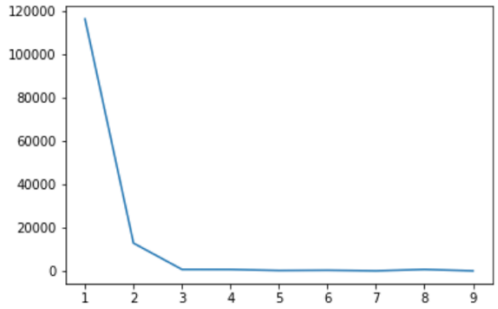
Tuljasree Bonam
G01179672
Username: hulksmash
Part 1 Accuracy: 0.95
Part 2 Accuracy: 0.78

HOMEWORK 3- REPORT

PART 1: Implementation and Importations for Iris Data Set:

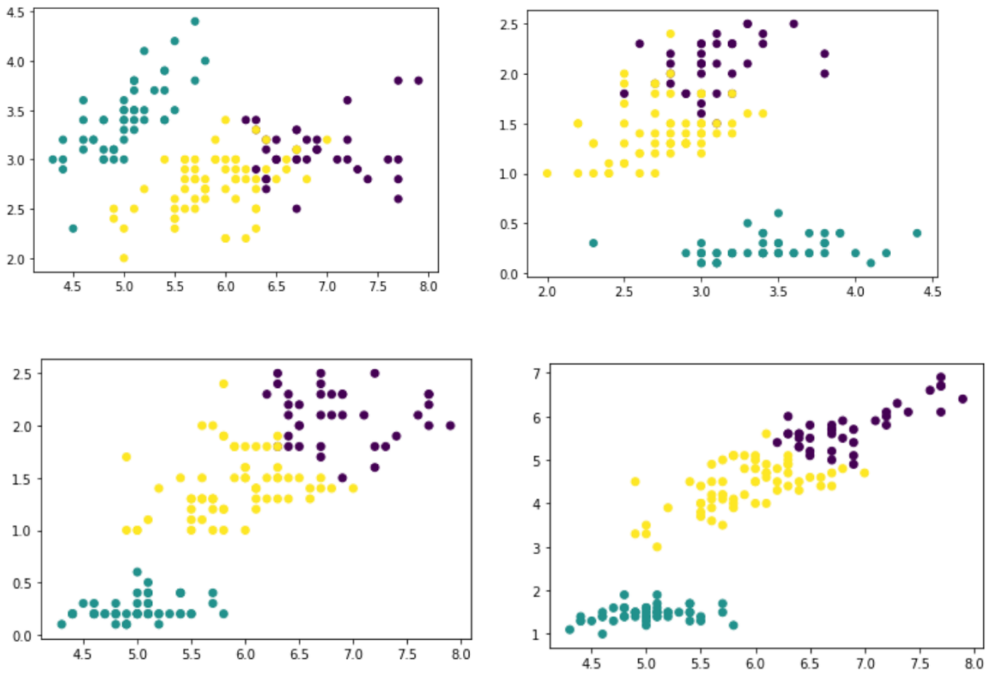
1. I have imported the following packages/ libraries for my assignment- numpy, distance, math, statistics matplotlib.pyplot.
2. I read my test data file dataExtract() function where I appended them to a list.
3. To optimally determine the selection of K-value for KMeans, I made use of **Elbow method**- aids in estimating the optimal K- value in K-Means and plotted a graph against sum of squared errors. I used the k- value obtained from the above Elbow method and used it in the K-Means clustering algorithm that I implemented KMeans_SSE().



From the elbow method, I've taken K=3.

4. **K-Means Algorithm-** a process where we determine the number of clusters **K** and we assume the centroid or center of these clusters. We can take any random objects as the initial centroids or the first **K** objects in sequence can also serve as the initial centroids. In the method **centroidFind()**, I have found the number of centers and passed it to an array which I later converted into a NumPy array. Thus obtained input test data is passed into the function **clustFind()** which measures the distance of each point with respect to the center. I have used Cosine distance function for measuring this. I even imported the distance package from scipy.spatial. Post calculating the distance, I used the index of minimum distance in each row as label for input data. Now, the average of data in each cluster has been assigned to another variable and updated the previous ones to it. The above process is continued till the values don't change anymore and then we can return the labels.

   Visualization: Plotted a scatter plots for the K-Means Iris data set



5. Finally, I have loaded the result in Resultiris.txt file which will print 'Succesful' after it is loaded in the file.

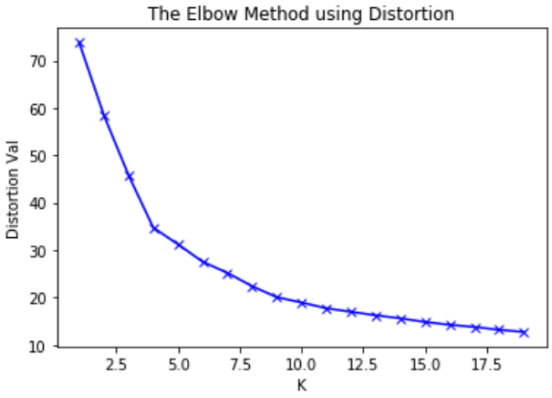
PART 2: Implementation for K-Means Algorithm (Image Clustering)- MNIST Data set:

1. I have implemented all the packages/ libraries necessary for the code- numpy, distance, math, statistics, matplotlib.pyplot, TSNE, PCA, pandas, metrics, cdist.
2. I read the data file using pandas pd.read_csv() method.
3. **Dimensionality Reduction:**
   Used Principal Component Analysis (PCA)- **pca()** and t-Distributed Stochastic Neighbor Embedding (t-SNE)- **tsne()**, the dimensions in the dataset have been reduced to an optimal level without losing any important information.
   Feature Reduction- Used sklearn's PCA- the sklearn.decomposition PCA for this.
   Feature Engineering- Used sklearn's t-SNE- sklearn.manifold for this to get useful features. My scores varied when I changed the number of components:

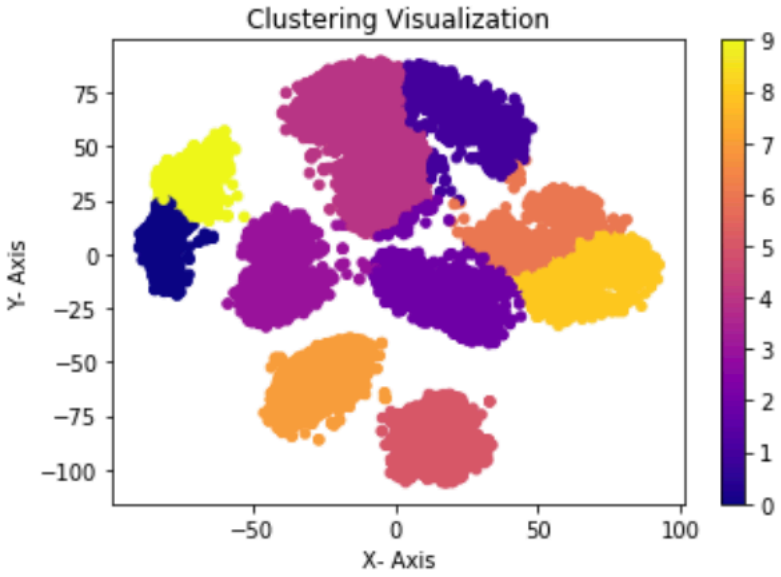
Feature Reduction/ Feature Engineering	**Accuracy Obtained**
sklearnPCA(n_components=350) TSNE(n_components=2, init='random',perplexity=50,n_iter=2000,n_iter_without_progress=1000,early_exaggeration=4) data = tsne.fit_transform(PCA_projected_Data)	**0.76**
sklearnPCA(n_components=50) TSNE(n_components=2, init='random',perplexity=50,n_iter=2000,n_iter_without_progress=1000,early_exaggeration=4)	**0.78**

data = tsne.fit_transform(PCA_projected_Data)	

Elbow Method- Made use of the Elbow Method to determine the optimal value for K by implementing a method for clustering model and calculating distortion and inertia values. Here, I chose K=10.



4. **K-Means Clustering Algorithm:** a process where we determine the number of clusters **K** and we assume the centroid or center of these clusters. We can take any random objects as the initial centroids or the first **K** objects in sequence can also serve as the initial centroids. In the method **centroidFind()**, I have found the number of centers and passed it to an array which I later converted into a NumPy array. Thus obtained input test data is passed into the function **clustFind()** which measures the distance of each point with respect to the center. I have used Manhattan (cityblock) distance function for measuring this. I even imported the distance package from scipy.spatial. Post calculating the distance, I used the index of minimum distance in each row as label for input data. Now, the average of data in each cluster has been assigned to another variable and updated the previous ones to it. The above process is continued till the values don't change anymore and then we can return the labels.
   Here, I tried multiple distance measuring metrics like Cosine, Manhattan and Minkowski and of the all, Manhattan (cityblock) gave me the best accuracy.
5. **Visualizing Clusters:** Scatter plot using K means Mnist dataset.



6. Now finally, the output has been loaded to "resultPartTwo.txt" file and prints "Successful!" at the end.