Task 1:

K means Algorithm:

1. Choose the number of clusters = n
2. Select k random number of points of the data set as the centroid

3. Calculate the distance between the point and the centroid. Assign all the data points to the closest cluster center
4. Recalculate the centroids of newly formed clusters

5. If the centroid locations change, repeat steps 3 & 4, until the Centroids of newly formed clusters do not change, which signals that the points remain in the same cluster.

The objective function of K means is to minimize the loss function J.

$$J = \sum_{i=1}^{k} \sum_{j=1}^{n} r_{ij} (\| x_i - \mu_j \|)^2$$

where, $\| x_i - \mu_j \|$ *is the Eucliedian distance between a point* x_i *and the centroid* μ_j *,*
over all k points in the i^{th} *cluster, for all n clusters.*
Here $r_{ij} = 1$ *for data point* x_i *if it belongs to cluster* j*; otherwise,* $r_{ij} = 0$.

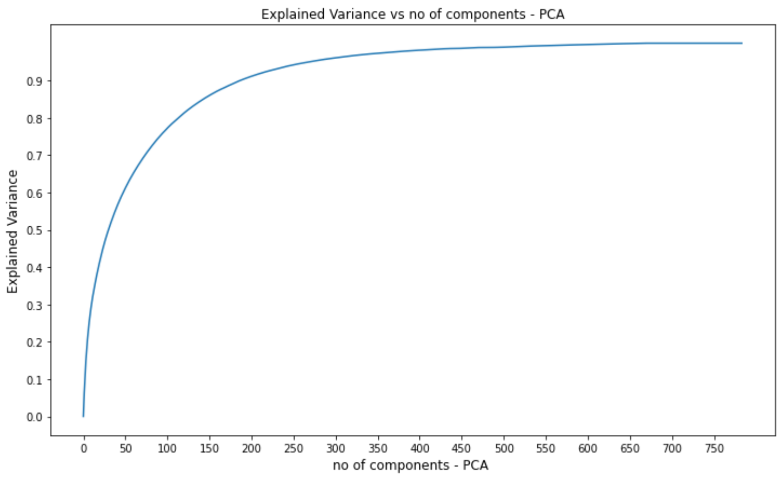
Task 2:

PCA Algorithm:

1. Separate the target data or the label data (if not separated) from the training data set.
2. For the training dataset X, for each column, separate the mean of that column from each entry.
3. Standardize your data X. Let the standardized matrix be Z
4. Get the covariance of Z
   Covariance of Z = Z^T .Z
5. Calculate the Eigenvalues and the Eigenvectors
6. Sort the Eigenvalues and their corresponding Eigenvectors in descending order. Choose k Eigenvectors that correspond to k largest Eigenvalues (Here, k will be the number of dimensions for the new feature subspace k<= d, d is the number of original dimensions)
7. Construct the Projection matrix W from selected k Eigenvalues
8. Transform the original dataset X via W to obtain the new k dimensional feature subspace Y.

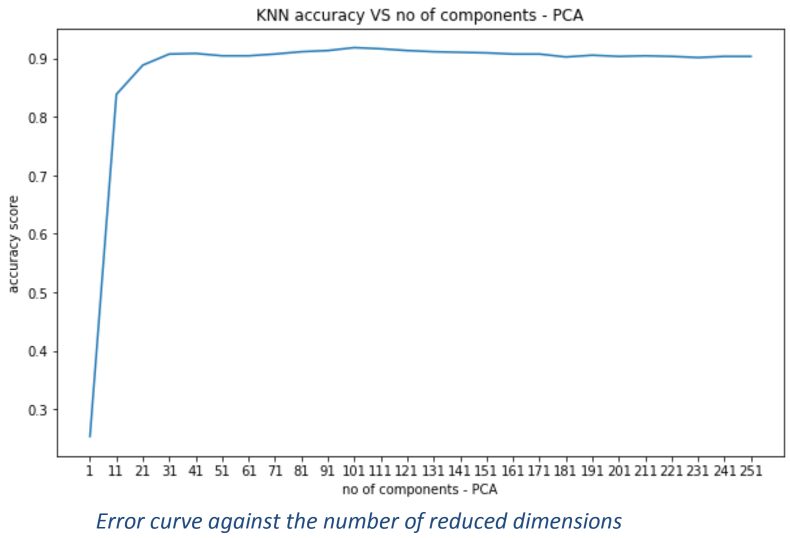
Analyses of implementation:

A graph of variance v/s number of components was plotted to analyze the variance on the number of the components. As you can see in the graph below, the maximum variance is seen with the components between 150 and 200. With a lot of trial and error, the maximum variance was observed with 180 components. So, the 784-dimensional training data can be reduced to 180 dimensions by finding the maximum variance with the help of PCA.



KNN (1 Nearest Neighbour classifier)

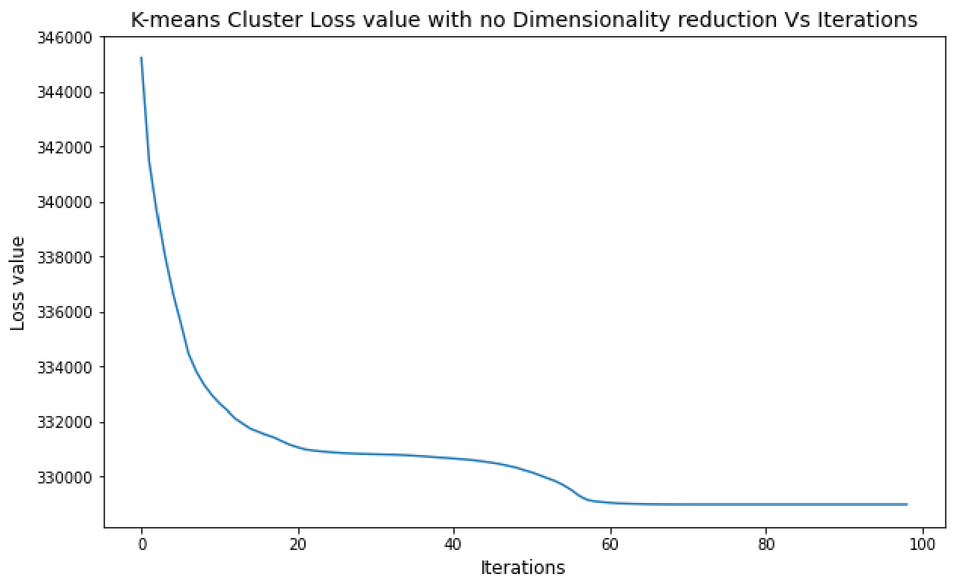
The training data was passed to the implemented PCA for components 1- 256. The projection matrix was then used to get the projected data. On fitting the projected data with the KNN (n=1) classifier the accuracy was plotted against the number of components. As you can see from the graph below, for 10 PCA components and accuracy of around 85% was achieved. And then the accuracy increased to 90% for 30 PCA components. This means that PCA when implemented with KNN (n=1), needs just 30 components to rightly cluster the data. This helps a lot to reduce the complexity of the model and avoid overfitting.



Error curve against the number of reduced dimensions

<u>Task 3:</u>

In this task, k-means clustering was applied to the MNIST training set without dimensionality reduction. The figure below shows the loss value of k-means with respect to the number of iterations. As seen, the loss decreases gradually after 20 iterations, and finally, after around 50 iterations, the loss value becomes almost constant. This means that after iteration number 50, there is no more re clustering taking place. The data points in the clusters are now unchanged.



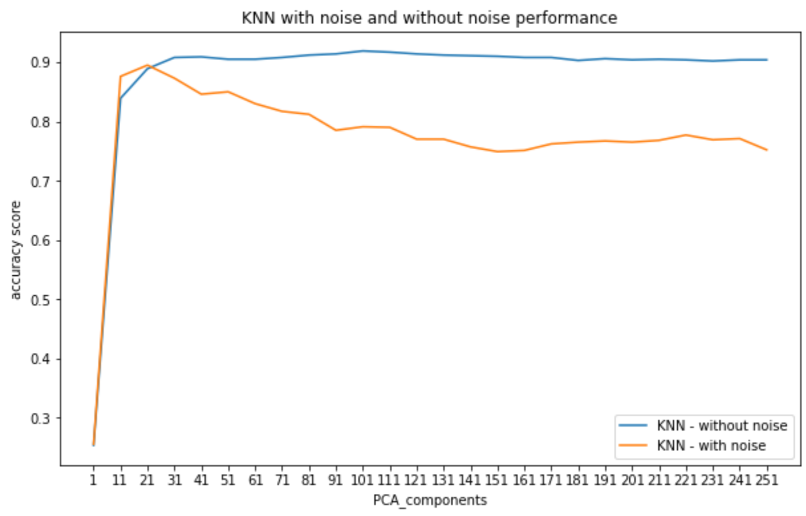
Loss curve - the change of loss value of the k-means algorithm with respect to the number of iterations.

TASK 5:
As required, 256 noisy dimensions were added to the data, to make it a 1024 dimensional dataset. A noisy data with mean = 0 and standard deviation = 0.75, was added to both the training and testing datasets.

PCA with KNN (n=1) as a classifier

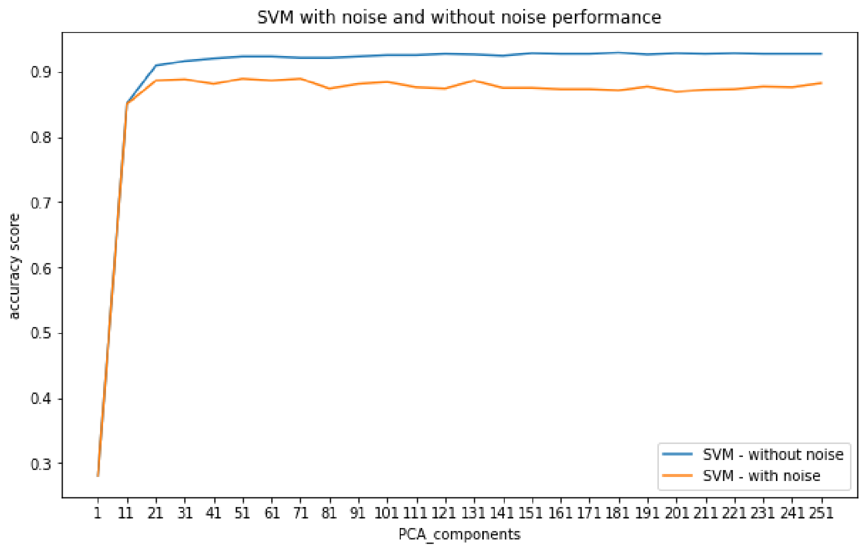
The graph below is the plot of PCA and KNN (n=1) with noisy data and without noisy data. As seen in the graph PCA is sensitive to noisy data.

More experiments with mean = 0.5, 1, 1.5, and deviation = 1.5,2,2.5 respectively, were carried out and the results were on similar lines. KNN was much more sensitive to noise as compared to SVM. This may be because KNN retains all the examples and does not generalize.



PCA with Linear SVM as a classifier

The same experiment was carried out with PCA along with Linear SVM (c = 0.05). The figure below shows the behavior of the model when the data is noisy and also when the data is normal. As seen SVM is less sensitive to the noisy data as compared to KNN. It takes care of outliers better than KNN.



Effect of noise on PCA

As seen above, PCA when used with SVM as a classifier was less sensitive to noise as compares to KNN as a classifier. However, PCA behaved in the same manner till the first 15-20 components, i.e. for the components with the highest variance, it performed the best.
It simply picks up the strongest trends in the dataset and stores it in the components. So, once the components with the highest variance are found, then the rest of the components hardly matter. In this way, PCA also helps for dimensionality reduction. Thus, PCA if the number of components is wisely selected PCA can be very less sensitive to noise.