Dimensionality Reduction and classification for MNIST DataSet

-Varsha Rani Chawan UTA ID#**1001553524**

Problem Statement : Classification of MNIST Images .

MNIST data set is collection of images of numbers from 0-9. Each image size is 28X28 gray scale image.Which makes dimensionality of each image to 28*28 = 784.So do we really need such a huge dimensions or features for classification ? We can still perform better classification with fewer significant features and get better accurate classification results.In scenarios where we have fewer training samples than the dimensions, the model overfits . To avoid this curse of dimensionality , we go for dimensionality reduction techniques to extract most significant features from the huge dimensions, i.e to transform the data from higher dimensions to fewer dimensions.

Approach used : In brief , I have implemented Principal component analysis to reduce and extract the most significant features. I,e used PCA dimensionality reduction technique to reduce the 784 dimensions to lower dimensions.I have experimented with various dimensions(2,5,10,20,50),then used the reduced dimension data set to classify the MNIST Test data using KNN algorithm for various K values[5,10,15,19]. Also, compared the results with other classifiers Logistic regression and random forest classifier. I have plotted the images formed from the [2,5,10,20,50] components eigenvectors . Additionally tried to reconstruct the original image from the reduced dimensions and could able to reconstruct the original data successfully, that I have not included as part of this project.

Algorithm steps in detail :
I have used only 4 classes from MNIST Data set as complete set is taking long time.

1. Read the MNIST Images using scipy, then extract the corresponding labels for each image using split.
2. Flatten the 28x28 shape to 1x784
3. Shuffle and split the data using sklearn library . By default it splits to 80/20 train/test set.
4. The train data is now 303x784 dimensions
5. Applied PCA to reduce the 303x784 data to 303x 2, 303x5, 303x10 , 303x20, 303x50 dimensions respectively.

PCA logic :

1. Calculate the mean for the dataset
2. Subtract the mean from the data to normalise the data(this is to avoid getting wrong features as prominent features and to centre the data and to avoid any errors arising due to scaling)
3. Calculate the covariance matrix for the normalised data.(this matrix contains the variance and covariance for each dimension)

4. Find the eigenvectors and eigenvalues for the covariance matrix using numpy library. Eigenvalue holds the information about the most significant features for the corresponding eigenvectors
5. Sort the eigenvalues in descending order and retrive the indices for most [2,5,10,20,50] components.
6. Using the indices filter the most significant [2,5,10,20,50] eigen vectors
7. Construct new dataset with most significant and reduced dimensions by multiplying normalised data with the filtered Eigenvectors
8. Apply step 1 to 7 for test data as well.
9. Plot the images for the reduced component eigenvectors using matplotlib
10. Now use the reduced dimensional data to classify test set .

KNN algorithm:

For different K[5,10,15,29] values repeat the below steps :

1. For each test data ,find the distance from all the training dataset **.**
2. Sort the distance in descending order
3. Take top K datasets(neighbours) and retrieve their labels
4. Select the label with max count as the predicted label for test set.
5. Repeat step 1 to 4 for remaining test data
6. Calculate accuracy by comparing the predicted values with actual values of test data .

Logistic Regression:

Implemented SKlearn logistic regression with the belows parameters:
solver- “ibfgs”, multiclass = “multinomial”, and calculated accuracy for test data .

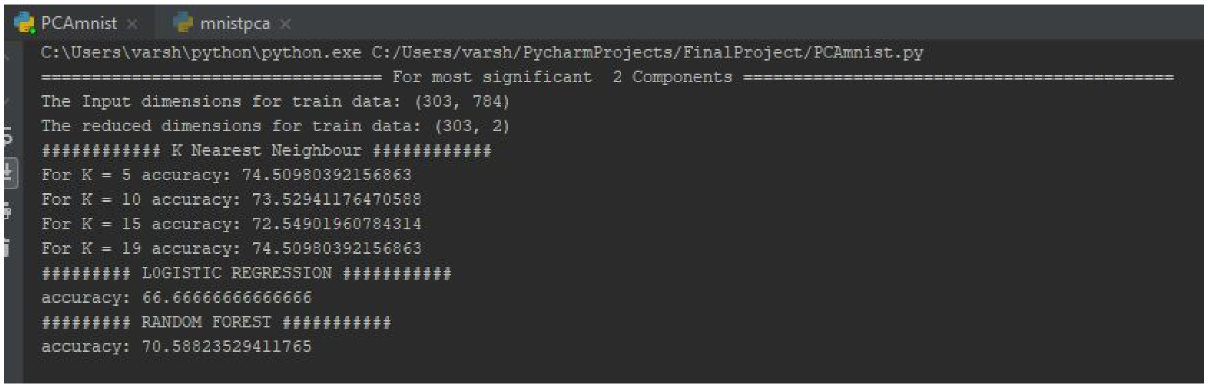
Random Forest Classifier :

Implemented SKlearn Random Forest Classifier : criteria=”Entropy”, estimators = 200

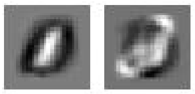
Results :
As I am using random shuffle and split method, the accuracy will vary every time based on the split of data.
Highest accuracy reached was 87%

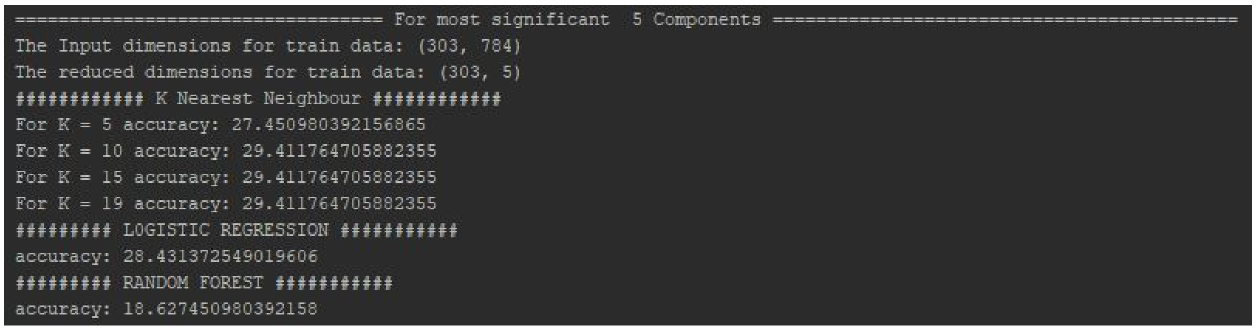
Inference:
The model gives maximum of 87% accuracy and the accuracy is almost same for different classifiers.

The below screenshots shows the accuracy for KNN, Logistic and RandomForest classifier for corresponding reduced component and the image plot is for the corresponding eigenvectors.

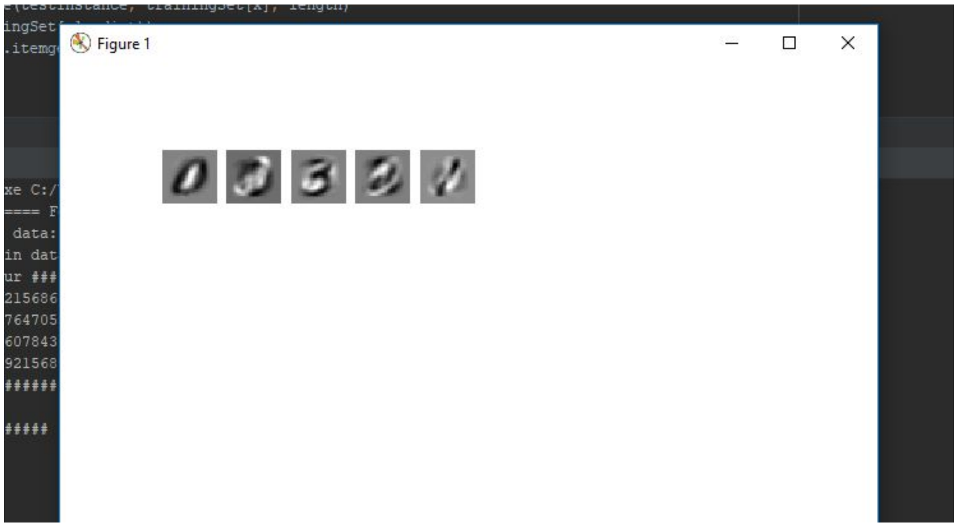


```
C:\Users\varsh\python\python.exe C:/Users/varsh/PycharmProjects/FinalProject/PCAmnist.py
================================== For most significant  2 Components ===========================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 2)
############ K Nearest Neighbour ############
For K = 5 accuracy: 74.50980392156863
For K = 10 accuracy: 73.52941176470588
For K = 15 accuracy: 72.54901960784314
For K = 19 accuracy: 74.50980392156863
######### LOGISTIC REGRESSION ###########
accuracy: 66.66666666666666
######### RANDOM FOREST ###########
accuracy: 70.58823529411765
```

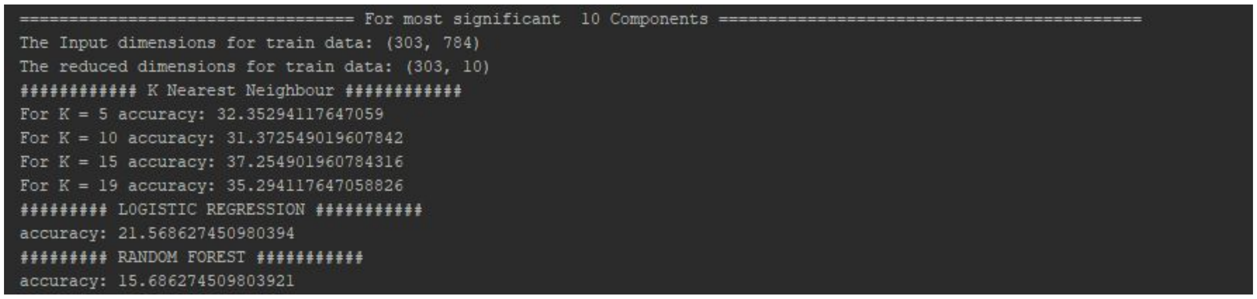




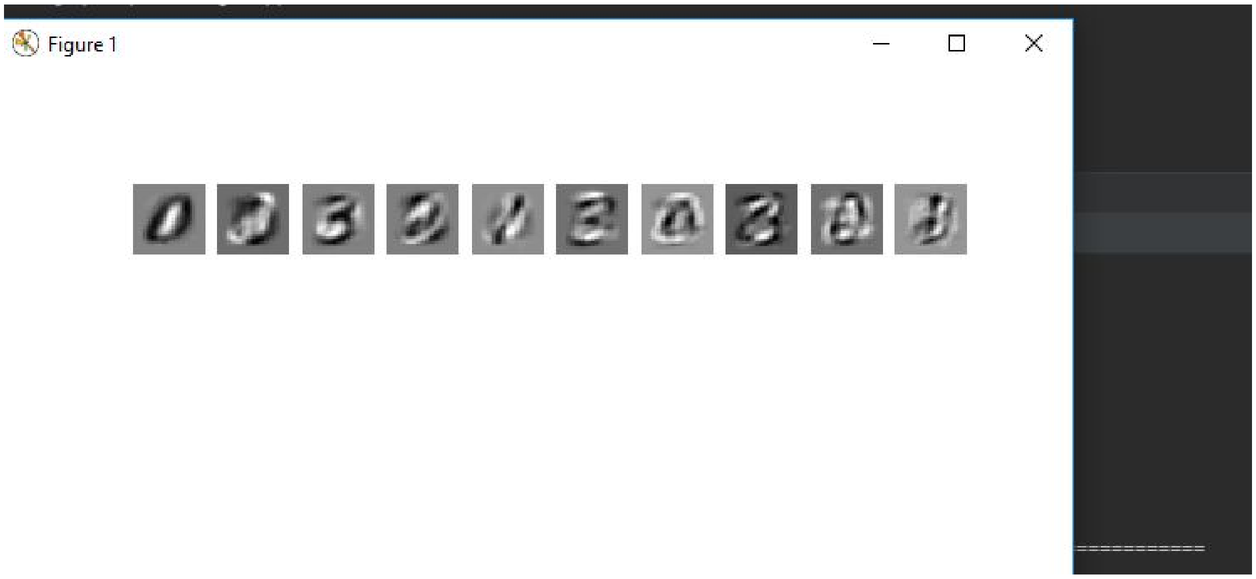
```
================================== For most significant  5 Components ===========================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 5)
############ K Nearest Neighbour ############
For K = 5 accuracy: 27.450980392156865
For K = 10 accuracy: 29.411764705882355
For K = 15 accuracy: 29.411764705882355
For K = 19 accuracy: 29.411764705882355
######### LOGISTIC REGRESSION ###########
accuracy: 28.431372549019606
######### RANDOM FOREST ###########
accuracy: 18.627450980392158
```



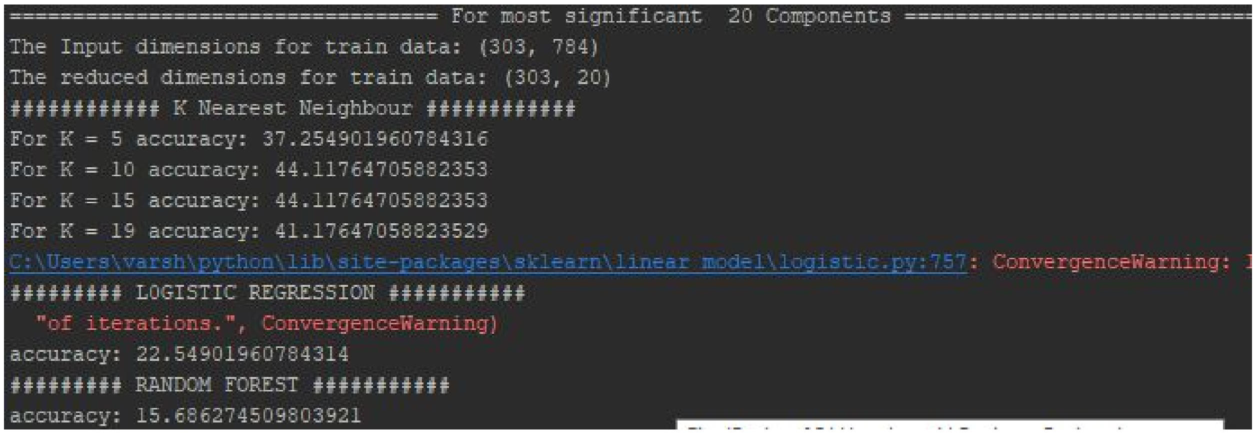
```
e(testInstance, trainingSet[x], length)
ingSet
.itemg
xe C:/
==== F
 data:
in dat
ur ###
215686
764705
607843
921568
######

#####
```



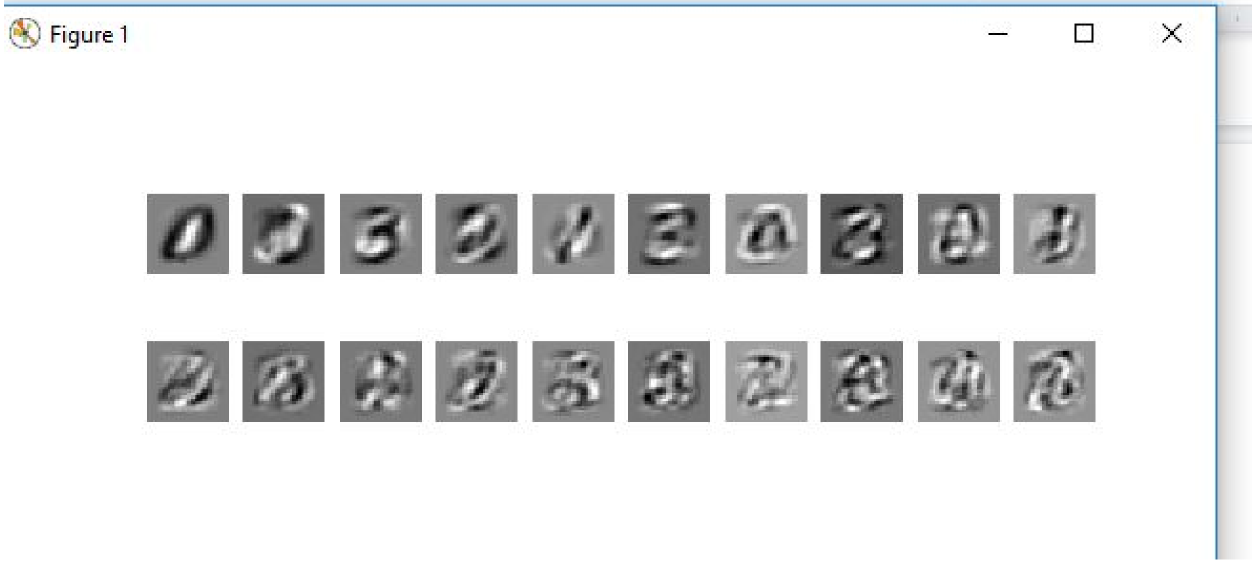
```
================================== For most significant  10 Components ===========================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 10)
############ K Nearest Neighbour ############
For K = 5 accuracy: 32.35294117647059
For K = 10 accuracy: 31.372549019607842
For K = 15 accuracy: 37.254901960784316
For K = 19 accuracy: 35.294117647058826
######### LOGISTIC REGRESSION ###########
accuracy: 21.568627450980394
######### RANDOM FOREST ###########
accuracy: 15.686274509803921
```

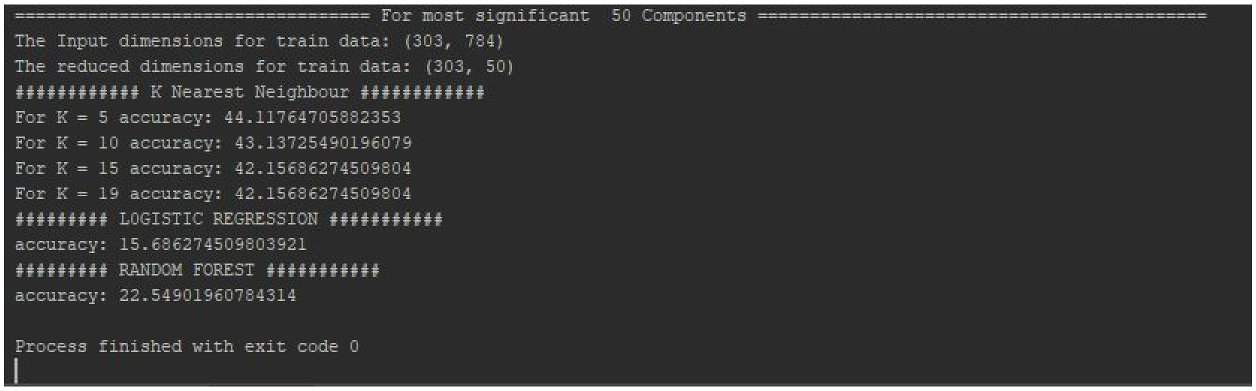


```
===========
```

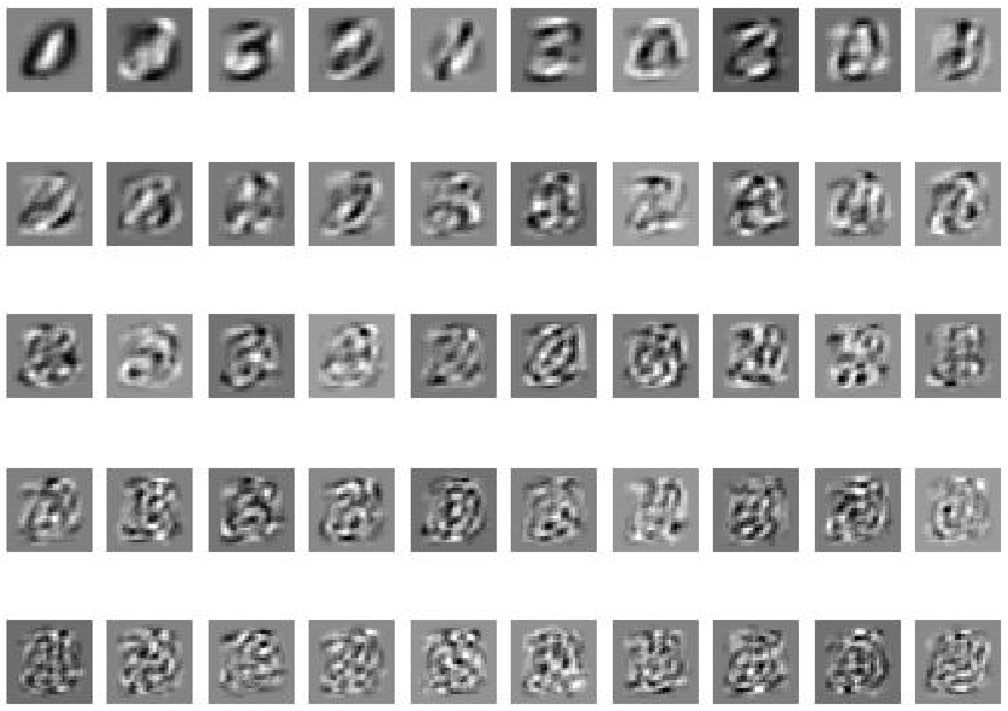


```
=================================== For most significant  20 Components ===========================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 20)
############ K Nearest Neighbour ############
For K = 5 accuracy: 37.254901960784316
For K = 10 accuracy: 44.11764705882353
For K = 15 accuracy: 44.11764705882353
For K = 19 accuracy: 41.17647058823529
C:\Users\varsh\python\lib\site-packages\sklearn\linear_model\logistic.py:757: ConvergenceWarning: l
######### LOGISTIC REGRESSION ###########
  "of iterations.", ConvergenceWarning)
accuracy: 22.54901960784314
######### RANDOM FOREST ###########
accuracy: 15.686274509803921
```

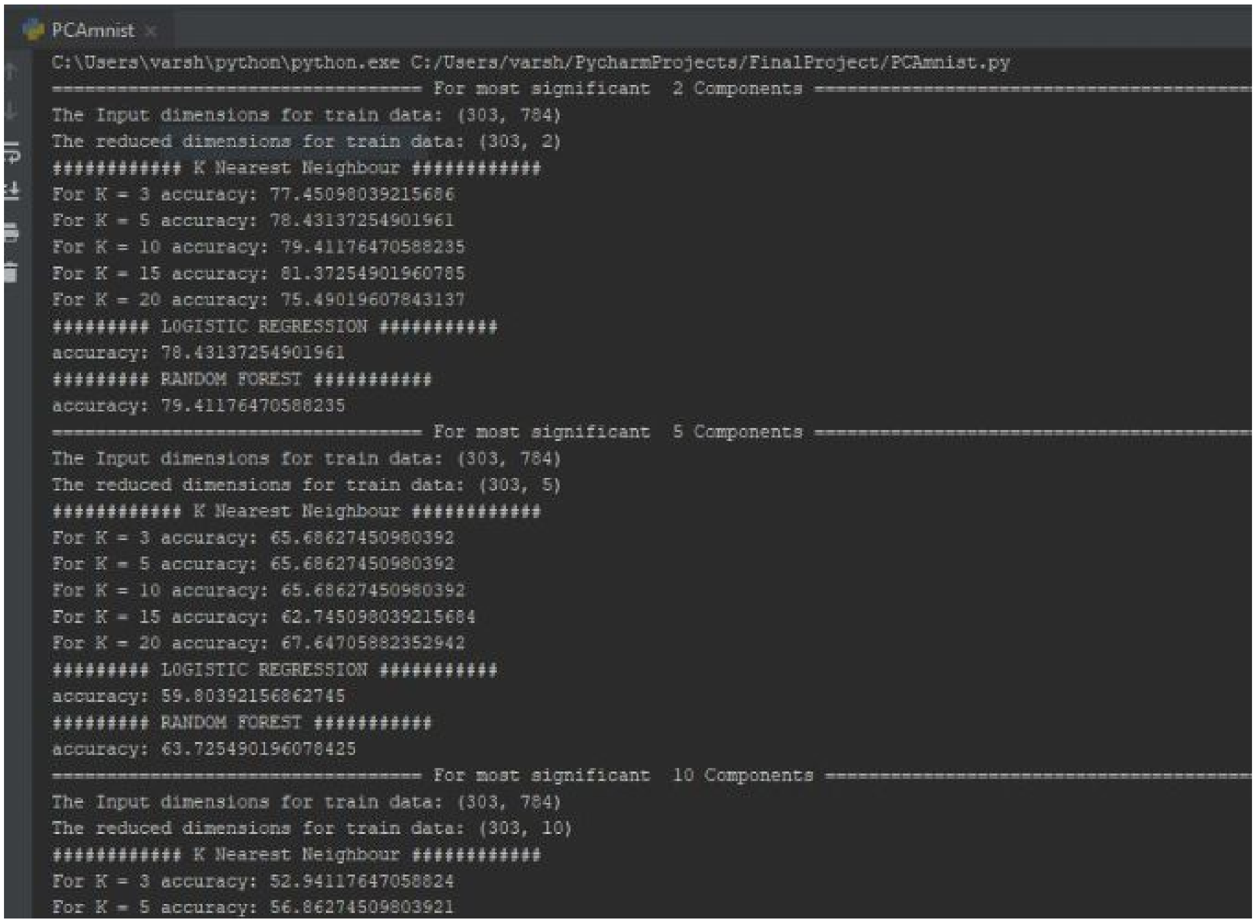




```
=================================== For most significant  50 Components ==========================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 50)
############ K Nearest Neighbour ############
For K = 5 accuracy: 44.11764705882353
For K = 10 accuracy: 43.13725490196079
For K = 15 accuracy: 42.15686274509804
For K = 19 accuracy: 42.15686274509804
######### LOGISTIC REGRESSION ###########
accuracy: 15.686274509803921
######### RANDOM FOREST ###########
accuracy: 22.54901960784314

Process finished with exit code 0
```



Screen shots for 80% accuracy:


PCAmnist

```
C:\Users\varsh\python\python.exe C:/Users/varsh/PycharmProjects/FinalProject/PCAmnist.py
==================================== For most significant  2 Components ============================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 2)
############ K Nearest Neighbour ############
For K = 3 accuracy: 77.45098039215686
For K = 5 accuracy: 78.43137254901961
For K = 10 accuracy: 79.41176470588235
For K = 15 accuracy: 81.37254901960785
For K = 20 accuracy: 75.49019607843137
######### LOGISTIC REGRESSION ###########
accuracy: 78.43137254901961
######### RANDOM FOREST ###########
accuracy: 79.41176470588235
==================================== For most significant  5 Components ============================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 5)
############ K Nearest Neighbour ############
For K = 3 accuracy: 65.68627450980392
For K = 5 accuracy: 65.68627450980392
For K = 10 accuracy: 65.68627450980392
For K = 15 accuracy: 62.745098039215684
For K = 20 accuracy: 67.64705882352942
######### LOGISTIC REGRESSION ###########
accuracy: 59.80392156862745
######### RANDOM FOREST ###########
accuracy: 63.725490196078425
==================================== For most significant  10 Components ===========================================
The Input dimensions for train data: (303, 784)
The reduced dimensions for train data: (303, 10)
############ K Nearest Neighbour ############
For K = 3 accuracy: 52.94117647058824
For K = 5 accuracy: 56.86274509803921
```