PROJECT 1 - MNIST Fashion Classification

Submitted in partial fulfillment of the requirements for the course of

ENEE633 – Statistical Pattern Recognition

By

Adheesh Chatterjee

Course Faculty
Prof. Rama Chellapa

A. JAMES CLARK
SCHOOL OF ENGINEERING

INTRODUCTION

The aim of this project is to analyze the prediction accuracy and the time efficiency of two classifier models: Bayes Classifier with Maximum Likelihood Estimator and K Nearest Neighbour, applied to the Fashion MNIST Dataset. The project also analyzes the effect of two preprocessing methods - Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) applied to both classifier models developed.

1 BAYES CLASSIFIER

The bayes classifier follows the decision rule :

$$\text{Assign } x \text{ to } w_i \text{ if } P(x|w_i)P(w_i) \ > P(x|w_j)P(w_j),\ i \neq j$$

To acquire the parameters which best describe the distribution of the data point given the class label, we use the Maximum Likelihood Estimator.

$$\hat{\theta}_i \ = \ argmax_{\theta_i}\, p(X_i,\theta_i)$$

where X_i is the dataset belonging to class i. In a Gaussian distribution, we have

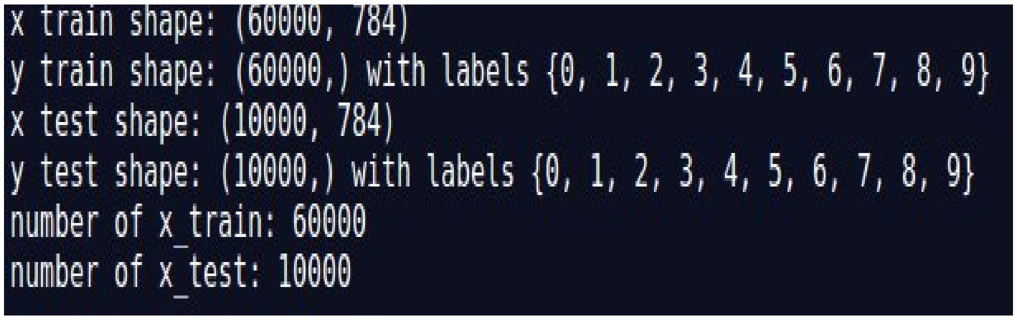
$$\hat{\boldsymbol{\mu}}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} \boldsymbol{x}_{ik}$$
$$\hat{\Sigma}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} (\boldsymbol{x}_{ik} - \hat{\boldsymbol{\mu}}_i)(\boldsymbol{x}_{ik} - \hat{\boldsymbol{\mu}}_i)^T$$

So using the ML estimate and the bayes decision rule, the classification task becomes

$$\text{Assign } x \text{ to } w_i \text{ if } P(x,\mu_i,\Sigma_i) \ > P(x,\mu_j,\Sigma_j),\ i \neq j$$

1.1 APPROACH

First, we load the dataset using the mnist_reader module in the utils folder. We then check the size of our dataset and conduct display a couple of random images from our training data to further understand the dataset.



```
x train shape: (60000, 784)
y train shape: (60000,) with labels {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
x test shape: (10000, 784)
y test shape: (10000,) with labels {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
number of x_train: 60000
number of x_test: 10000
```

Fig. Train and Test Dataset

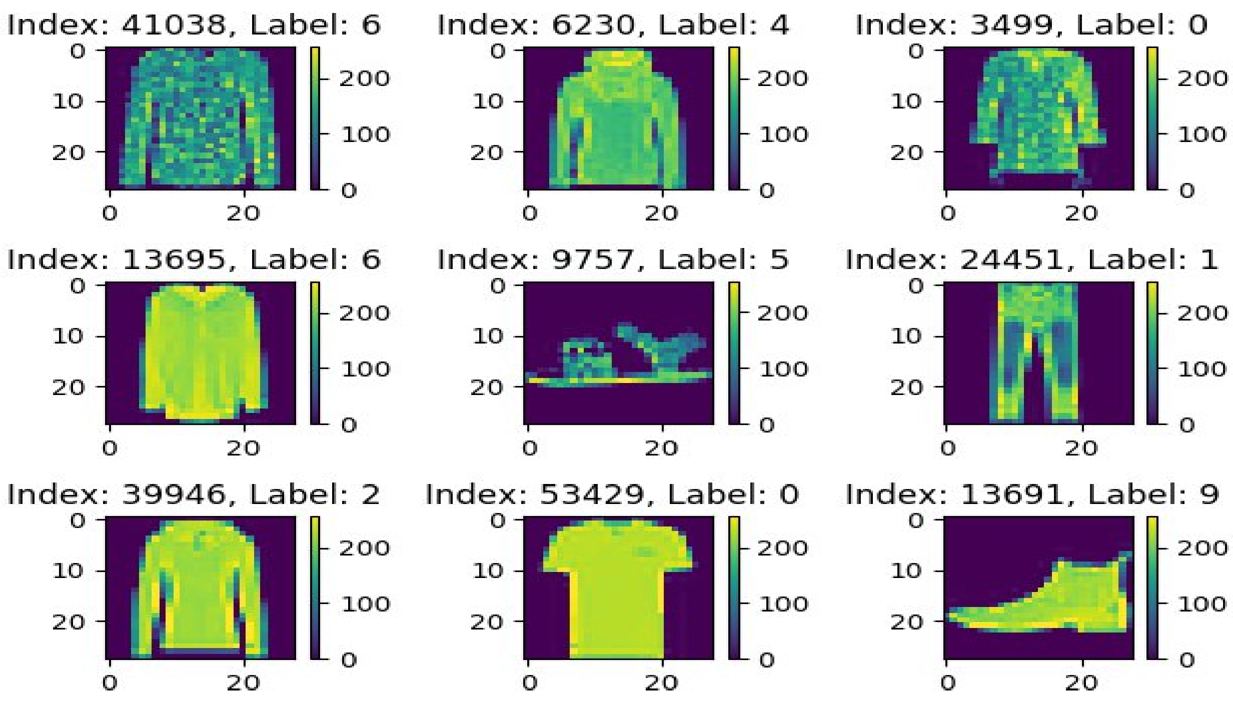


Fig. Training Set Data Visualized

We then split up the images as per their labels into 10 classes, and store them in numpy arrays. The mean and covariance among the classes are then found and stored in mean and covariance arrays respectively.

We now normalize our covariance matrices by dividing every element in the matrix by the largest element in that matrix. We do this to all 10 covariance matrices.But, we notice that some of the Σ are zeros due to the existence of zero rows in the covariance matrices and so our decision rule doesn't work properly. This, we implement a technique called "Regularization" which is ideal in solving such situations by adding suitable constants to the diagonal elements of the Σ matrix

$$\Sigma_{reg} = \Sigma + \alpha I$$

The constant α is a hyperparameter which can potentially be tuned to give better performance. Using trial and error we find that a value of 0.88 gives a good accuracy (~85.39%).

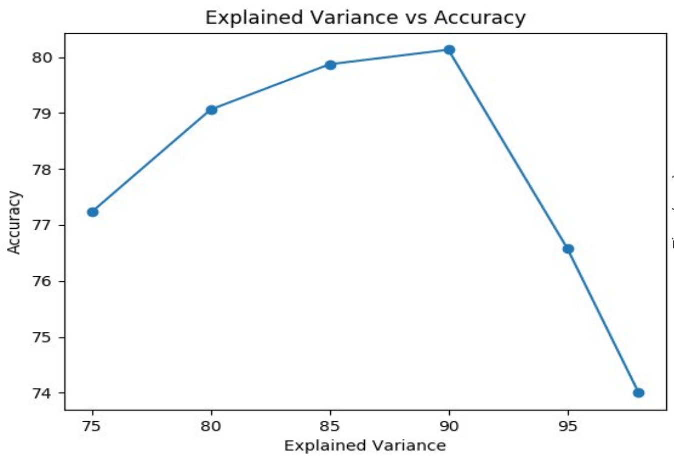
We then fit gaussians using the mean and variances for each data class. Then the every test data point is fit on all the gaussians and the one which best fits the data point is used to classify the point. We create an array of our prediction values and compare them to our ground truth values. The final accuracy of our model comes out to be **85.39%**. However, we notice that it takes a long time for our model to fit and predict the data. Hence, we apply the PCA and LDA techniques to improve speed with a small tradeoff for accuracy.

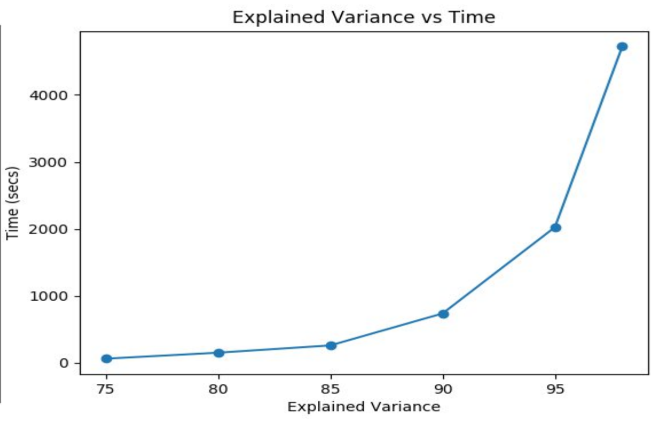
1.2 PCA ANALYSIS

Principal Component Analysis is a standard dimensionality reduction approach. The fundamental idea of PCA is to find several "principle features" in a dataset and use them as axes to describe the dataset. In reality, these axes are possible to be linear combinations of several features. By using Singular Value Decomposition (SVD), we find eigen vectors corresponding to large singular values as the direction of "principle" axes, and omit axes with small singular values. Usually, the threshold of singular value is defined by "Explained variance", which can be view as the percentage of the dataset fullness: by setting a certain threshold, the corresponding percentage of information of the dataset will be maintained. *Reference : https://scikit-learn.org/stable/modules/decomposition.html#pca*

We apply PCA on the original dataset, without normalization or regularization of the covariance matrices, by changing the explained variance and tabulate the change in accuracy as seen below

Explained Variance	No. Of Components	Accuracy	Time Taken (sec)
75	14	77.23%	61.725
80	24	79.06%	151.514
85	43	79.87%	259.594
90	**84**	**80.13%**	**736.744**
95	187	76.58%	2024.279
98	349	74.0%	4722.184





One advantage of PCA is that it converges very quickly, compared to our original approach which takes around 2-3 hours to converge for 10000 datapoints.

We can see that the highest accuracy occurs at the range between **85% and 90%**. Once the percentage is not in the range, the accuracy starts to decrease, which implies that maintaining a reasonable changing tendency of dataset yields a significant result of excluding noisy information as well as decreasing the dimension of the feature space. However, out of this reasonable range causes either losing too much information for describing dataset or leaving dimension of the feature space almost unchanged.

We also observe that the time increases exponentially, so setting a explained variance value of 85%-90% would yield a good accuracy in a reasonable amount of time.

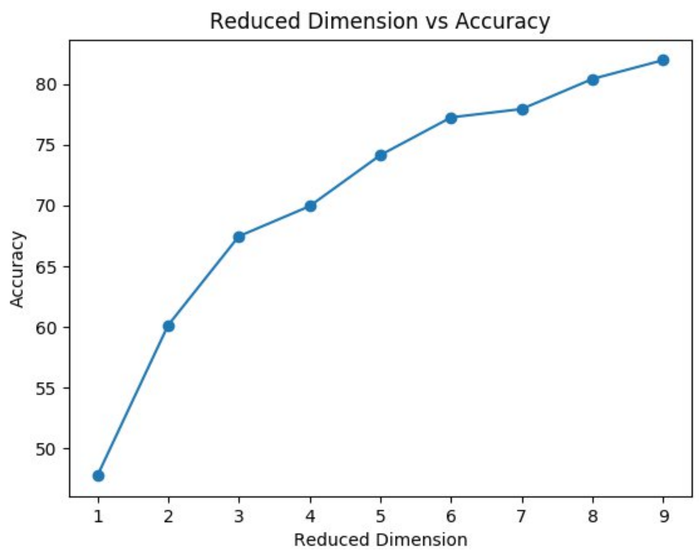
1.3 LDA ANALYSIS

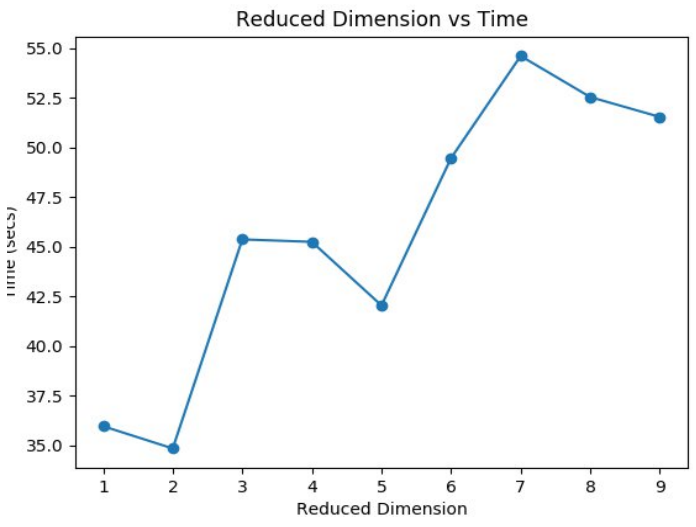
The fundamental idea of Linear Discriminant Analysis (LDA) is to refine features of a dataset in order to achieve successful classification. It can be used as either a classifier or a supervised dimensionality reduction tool. By projecting the input data to a suitable linear subspace, LDA minimizes the within-class scatter and maximizes between-class separation at the same time. The dimension of the output is necessarily less than the number of classes, so this is, in general, a rather strong dimensionality reduction, and is ideal for a multi-class problem. Usually, the optimal reduced result of LDA is given by min{c − 1, d}, where c is the number of classes and d is the original dimension of the feature space.

Reference:https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

We apply LDA by changing the dimension of the projection subspace and tabulate the change in accuracy as seen below

Reduced Dimension	**Accuracy**	**Time Taken (sec)**
1	47.73%	35.961
2	60.13%	34.841
3	67.47%	45.367
4	69.95%	45.243
5	74.16%	42.041
6	77.27%	49.465
7	77.96%	54.603
8	80.44%	52.533
9	**81.99%**	**51.532**





2 K-NEAREST NEIGHBOUR (KNN)

Unlike Bayes classifier, which introduces probability theory into the classification task, kNN employs the use of euclidean distance for classification. For every data point x, it assigns it to a class i if most of the data points of its k nearest neighbors belong to i.

It seems pretty obvious that kNN should achieve better prediction accuracy given large dataset, but the trade-off is a high time complexity O(n^2) when predicting because it has to compare one data point to every other of itself.

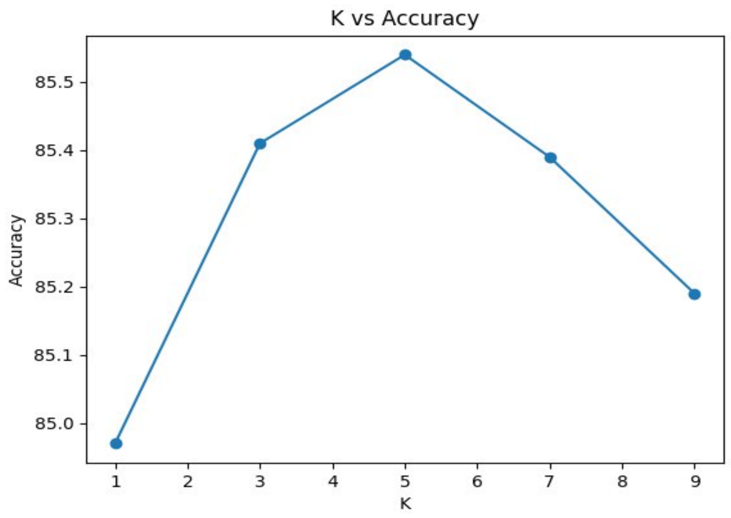
2.1 APPROACH

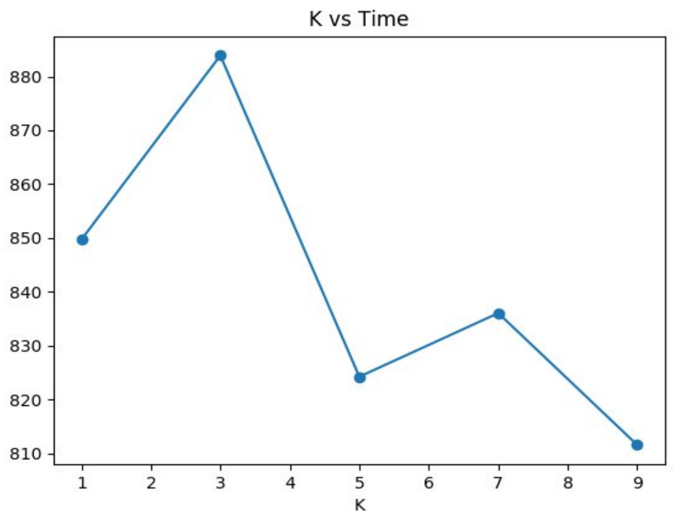
Having already loaded and analyzed our initial data as part of our Bayes Classifier approach, we directly fit the K Neigbours classifier to the training set.

We then predict the label of each image of our test set and compare it to the ground truth values to get accuracy. Again, we apply the PCA and LDA techniques to improve speed with a small tradeoff for accuracy.

We do this for different number of neighbors K.

K	Accuracy	Time Taken (secs)
1	84.97%	849.688
3	85.41%	883.899
5	**85.54%**	**824.215**
7	85.39%	836.036
9	85.19%	811.544



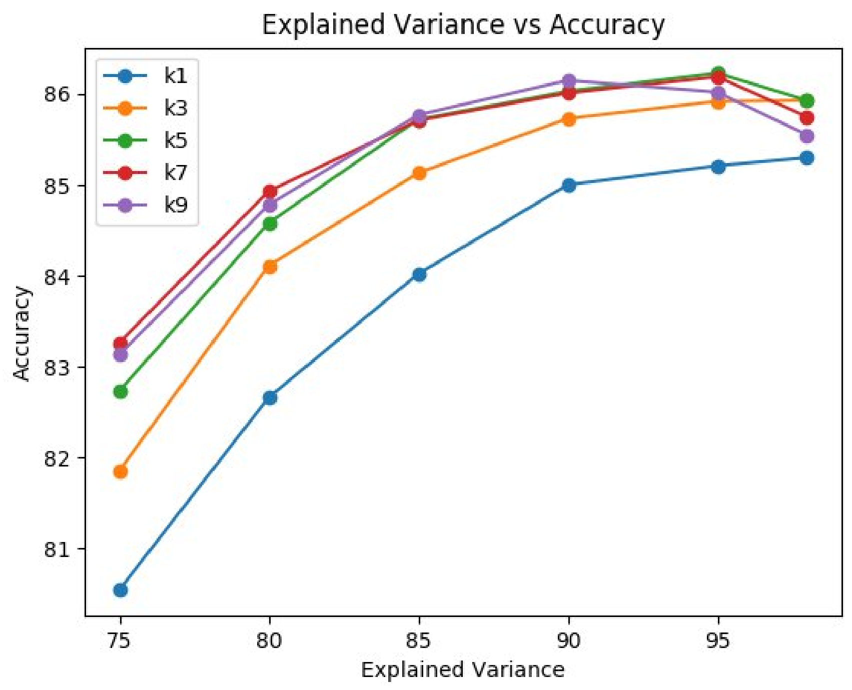


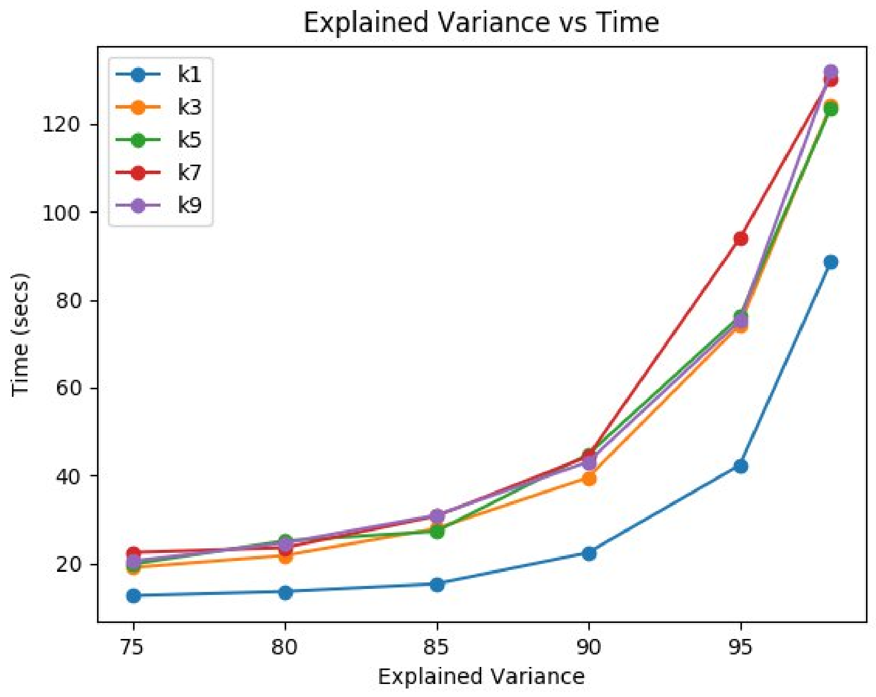
From the plot, we see that the although the accuracy doesn't change significantly depending on the number of neighbors, the highest accuracy is achieved at k=5.

2.2 PCA ANALYSIS

We apply PCA by changing the explained variance and tabulate the change in accuracy as seen below. Assuming a 1% increase in accuracy is considered significant, the best possible solution is highlighted for each k.

k	Explained Variance	No. Of Components	Accuracy	Time Taken (secs)
1	75	14	80.54%	12.686
	80	24	82.66%	13.653
	85	43	84.02%	15.383
	90	**84**	**85.00%**	**22.46**
	95	187	85.21%	42.351
	98	349	85.30%	88.585
3	75	14	81.85%	19.083
	80	24	84.11%	21.823
	85	**43**	**85.13%**	**27.927**
	90	84	85.73%	39.522
	95	187	85.92%	74.175
	98	349	85.93%	124.229
5	75	14	82.73%	19.835
	80	24	84.58%	25.148
	85	43	85.72%	27.308
	90	**84**	**86.03%**	**44.753**
	95	187	86.23%	76.135
	98	349	85.93%	123.548
7	75	14	83.26%	22.541
	80	24	84.93%	23.603
	85	**43**	**85.71%**	**30.750**
	90	84	86.01%	44.337
	95	187	86.19%	93.838
	98	349	85.75%	130.235
9	75	14	83.13%	20.543
	80	24	84.78%	24.667
	85	43	85.77%	30.972
	90	**84**	**86.15%**	**43.068**
	95	187	86.02%	75.129
	98	349	85.55%	131.803





We clearly see that the prediction is almost 200 times faster with negligible decreasing accuracy. Such improvement implies that as long as the information of relative distance between data points is well maintained, dropping out many noisy features will not affect the accuracy of kNN, while the computation time is shortened significantly.

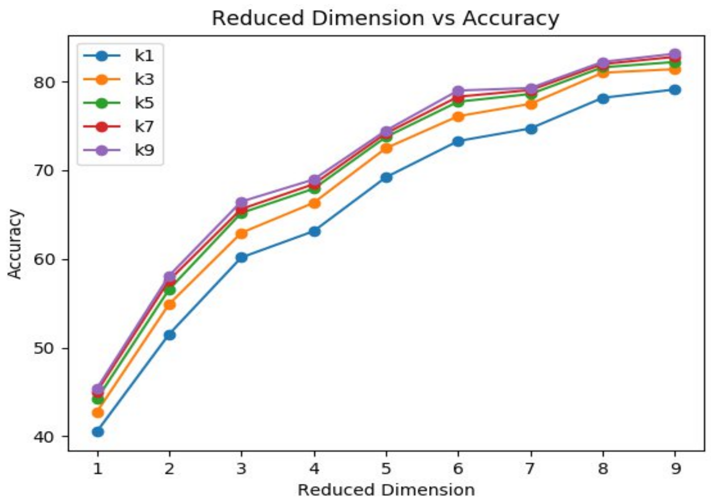
Again, the best possible explained variance is between **85-90%** as it gives a high accuracy in a reasonable amount of time.

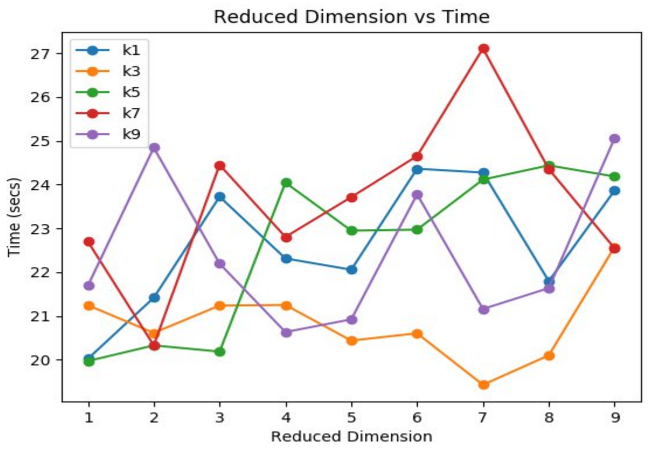
2.3 LDA ANALYSIS

We apply LDA by changing the dimension of the projection subspace and tabulate the change in accuracy as seen below

k	Reduced Dimensions	Accuracy	Time Taken (secs)
1	1	40.52%	20.023
	2	51.51%	21.417
	3	60.17%	23.725
	4	63.13%	22.314
	5	69.21%	22.057
	6	73.30%	24.363
	7	74.72%	24.276
	8	78.17%	21.796
	9	**79.11%**	**23.867**
3	1	42.75%	21.245
	2	54.90%	20.614
	3	62.95%	21.238
	4	66.33%	21.251
	5	72.49%	20.438
	6	76.10%	20.605
	7	77.49%	19.428
	8	81.0%	20.101
	9	**81.40%**	**22.577**
5	1	44.19%	19.971
	2	56.57%	20.328
	3	65.19%	20.187
	4	67.92%	24.047
	5	73.75%	22.949
	6	77.73%	22.972
	7	78.60%	24.116
	8	81.61%	24.439
	9	**82.21%**	**24.186**
7	1	45.03%	22.699
	2	57.59%	20.329
	3	65.63%	24.452
	4	68.43%	22.805
	5	74.16%	23.715
	6	78.30%	24.654
	7	79.04%	27.111
	8	81.97%	24.355
	9	**82.79%**	**22.545**
9	1	45.37%	21.693
	2	58.10%	24.845
	3	66.47%	22.196
	4	68.96%	20.633
	5	74.47%	20.923
	6	78.99%	23.779
	7	79.27%	21.160

	8	82.22%	21.636
9	**9**	**83.14%**	**25.063**





The prediction time is almost the same for all values of k, ranging between 20 and 27 seconds. We see there is no pattern here. But the accuracy as we see increases steadily as we increase the reduced dimensions.This happens due to the projection of data points: what LDA does is changing the scatter pattern within a class and between classes, yet this process also changes the information of distance between data points, which may cause some unwanted result for kNN, for example, overlapping data points in the projecting subspace.

We see that, the time taken is approximately the same no matter the case, hence the highest accuracy is obtained when reduced dimensions are 9 for all k

CONCLUSION

We successfully apply the Bayes Classifier and the k-Nearest Neighbor Classifier on our dataset and then using PCA and LDA dimensionality reduction tools to improve our performance and speed.

We realize that the Bayes Classifier is a parametric approach while kNN is a non-paramatric approach. The fundamental idea behind each model is simple, yet due to the significant improvement in computational speed, these time-consuming processes now becomes significantly faster. However, with the advent of Neural Networks and their innumerous advantages we realize these approaches will soon become redundant in common practice.