“[CMSC 828C/ ENEE 633] Project - 1”

STATISTICAL PATTERN RECOGNITION

PROJECT 1 REPORT

ADITYA VAISHAMPAYAN -116077354

Instructor: Dr. Rama Chellappa

Contents

Introduction

The main aim of this project is to use the methods such as Bayes classifier and K-nearest neighbor classifiers with and without dimensionality reduction techniques such as LDA, various types of PCA on the Fashion-MNIST dataset provided by Zolando Research.

The dataset consists of 70,000 total images divided between 60,000 training images and 10,000 test images. The dataset is similar to the digit MNIST dataset but is considered to be more complicated and is preferred for evaluating the performance of various classifiers. The dataset is 764-dimensional dataset (28 * 28). The main reason behind the dimensionality reduction techniques is to reduce the dimensions of the dataset and thus decrease the training and testing time of the classifiers.

Dimensionality reduction methods:

The dimensionality reduction techniques used are:

1. Principal Component Analysis [PCA]
2. Linear Discriminant Analysis [LDA].

PCA:

First it identifies the hyperplane that lies closest to the data, and then it projects the data onto it. The basic idea behind PCA is that we try to find a axis that preserves the maximum variance in the data. Also, it can be said the we choose a hyperplane defined by the principal components is such a way that it minimizes the distance between the original data and the projected data.

Note: The direction of the principal components is not stable, if you perturb the training set slightly and run PCA again, some of the new PCs may point in the opposite direction of the original PCs. PCA assumes that the dataset is centered around the origin. Scikit learns PCA algorithm does the centering by itself. Instead of arbitrarily choosing the number of principal components, the best way is to choose is such that around 95% of the variance is preserved.

Other options of PCA implementation are as follows:

1. Incremental PCA
2. Randomized PCA
3. Kernel PCA

Incremental PCA:

In incremental PCA one doesn't need to fit the whole dataset in the memory for the PCA algorithm to run. The data is split into smaller batches and then fed to the Scikit-learn algorithm for computing the principal components. This way the memory usage can be kept under control and also it can be applied on the fly.

Randomized PCA:

Yet another type of PCA is randomized PCA which is a is a stochastic algorithm that quickly finds an approximation of the first d principal components. Its complexity is $O(m \times d^2) + O(d^3)$, instead of $O(m \times n^2) + O(n^3)$, where d is much smaller than n. Thus, the randomized PCA is a lot faster.

Kernel PCA:

There is also one more method called kernel PCA. However, I haven't used it for experimentation purposes, as it required grid search algorithm, for determining the best parameters. Grid search or Randomized grid search take a lot of time, and thus have been avoided.

LDA:

LDA is called linear discriminant analysis. It is user for dimensionality reduction. The maximum no. of dimensions for LDA are n-1, where n is the number of classes in the dataset. For e.g. in Fashion-MNIST there are total 10 classes, thus the maximum no. of dimensions for LDA can be 9.

The image given below will give a clear distinction of when to use LDA and when to use PCA. LDA performs well, if the clusters have similar variances but different means, whereas PCA tends to perform better when the clusters in the data have the same mean but different variances. In Fishers' LDA, we try to project the data in such a way that maximizes the interclass distance and minimizes the within class variance.

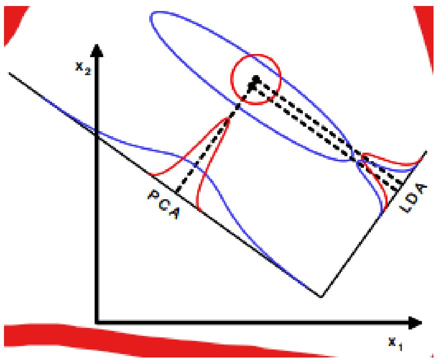


Fig: Plot showing LDA and PCA gaussians for two data clusters

Performance Evaluation methods:

K – fold Cross Validation:

In K – fold cross validation, we randomly split the training set even further into training and validation set. For e.g. if we select the value of k to be 10, the training data gets randomly into 10 portions out of which 9 are used for training and 1 will be used for validation. The subsets of data are called folds. The process of splitting the data happens 10 different times, picking a different fold for evaluation every time and training on other 9 folds. This way we can get 10 different scores, take a mean and thus get a mean performance result for the classifier. Thus, cross validation, not only allows to get an estimate of the model, but it also shows us how precise the estimate is i.e. standard deviation. The only drawback of using cross validation is training the model several times. This leads to increased training time at the cost of better classifier accuracy estimate.

Confusion matrix

Confusion matrix tells us the number of instances where a class is misclassified as another class. I have made the confusion matrix, using the training data only. This way we can evaluate the classifier based on the training data only, thus keeping the testing data untouched. Each row in the confusion matrix is an actual class whereas the columns are the predicted classes. A perfect classifier only has true positives and true negatives, so its confusion matrix, has nonzero values only on its diagonal.

Normalized confusion matrix:

To obtain a normalized confusion matrix, one needs to divide each value in the confusion matrix by the corresponding no. of images in the corresponding class, so we can compare error rates instead of absolute number of errors. Also, the diagonal elements are filled with zeros, so as to only keep the errors

Precision, Recall:

- Precision = TP/ (TP + FP) where TP = True positives, FP = False positives.
- Recall = TP/ (TP +FN).

Recall is the ratio of positive instances that are correctly detected by the classifier

Classification report:

It is a report consisting of precision, recall and f1 score values. Since this is a multiclass problem, It is better to evaluate the P/R/F -measure on individual classes as compared to the whole data, because any imbalance, can make the results better or worse. Thus, individual scores per class are preferred. F1 score is the harmonic mean of precision and recall values. Thus, a classifier has a high F1 score if its precision and recall values are high. Also, the f1 score favors those classifiers which have similar precision and recall values.

Bayes Classifier:

Simple Bayes Classifier:

The standard pipeline for a Bayes classifier is to calculate the mean and covariance for each class. The MLE estimation for mean and covariance are the sample mean and covariances. Then we calculate the sample discriminant functions, and then assign the test data to a given class, provided the discriminant function provides maximum value for it.

Bayes classifier with LDA:

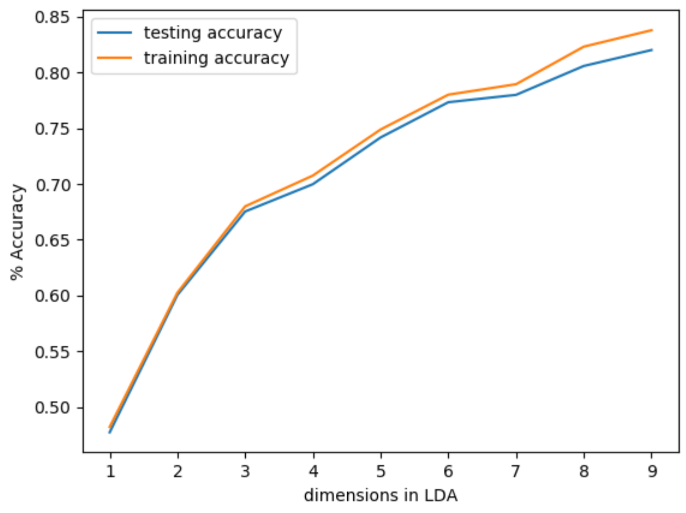


Fig: Train and test accuracy vs LDA dimensions

For LDA, we get the highest accuracy for 9 dimensions in the dataset. Also, we can see that the training and test accuracy increase as the l_components of the dataset increase. Thus, we get a maximum training accuracy of 83.78% and testing accuracy of 82.01% for 9 l-components in the dataset.

Bayes classifier with incremental PCA:

In the plot below we can see that the dimensions in PCA, vary from around 75, all the way up to 185. The highest, variance is obtained for 187 dimensions in PCA. This way we are preserving the variance in the range of 90% to 99%. Preserving the variance prevents loss of helpful information and maximizes the data retention. The PCA algorithm used here is incremental PCA. According the table given above, normal PCA algorithm gives an accuracy of about 22%, whereas the incremental PCA gives an accuracy of about 77.8%.

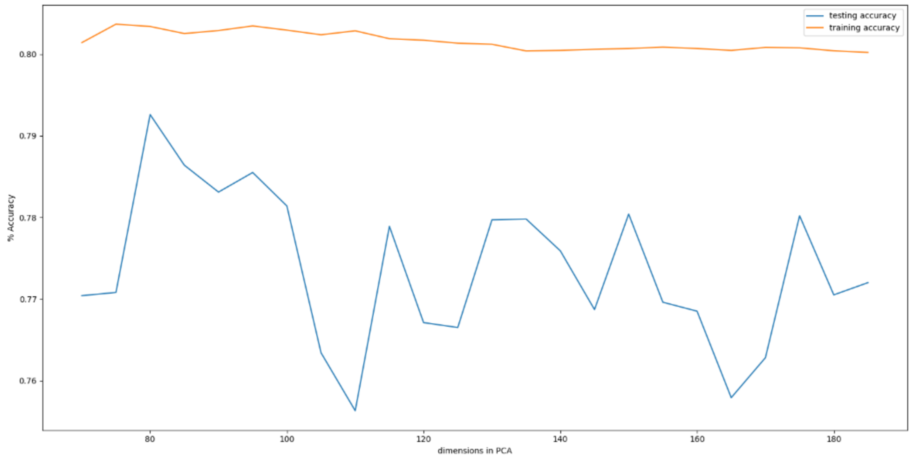


Fig: Plot of test and train accuracy vs incremental PCA dimensions

	Training time (s)	**Training accuracy**	**Testing time (s)**	**Testing accuracy**	**Average accuracy**
Bayes	0.68	78%	3.61	75.80%	**75.80%**
Bayes + Simple PCA (d = 500)	0.49	79%	1.42	22.73%	**22.70%**
Bayes + Incremental PCA (batch = 50, d =187)	0.18	80%	0.4	77.85%	**77.8**
Bayes + LDA	0.03	83.78	0.028	82.01%	**82.01**

Table: Testing and Training accuracies, time for Bayes classifiers with and w/o dimensionality reduction

From the above table it can be seen that for Bayes Classifier, we obtain highest test and train accuracy after applying LDA on the Fashion MNIST dataset. If we implement the normal PCA with as high as 500 dimensions, even then we get a very low average accuracy of 22.7%. However, this can be compensated by applying an incremental PCA with 187 dimensions and a batch size of 50. Thus, getting an average classifier accuracy of 77.8%. For a simple Bayes classifier, we get an average of 75.80% which is pretty good, even without performing dimensionality reduction.

Thus, we can infer that LDA gives much better results as compared to incremental or Simple PCA. Also, we can say that LDA carves up the hyperspace to suit the data distribution of individual classes whereas PCA collapses the hyperspace.

KNN (K-nearest neighbors) classifier

KNN stands for k-nearest neighbor. The way this algorithm works is, it calculates the distance of the test point from all the other points, and then sorts them all according to their increasing distance. K is a parameter, that needs to be experimented with. This is k = 5, it means, the algorithm considers 5 nearest neighbors to the unclassified point, and then the class that is in majority becomes the class of the point.

	Training time(s)	**Training accuracy**	**Testing time (s)**	**Testing accuracy**	**average accuracy**
KNN	13.79	87.2%	678.26	85.54%	85.5
KNN + simple PCA	1.99	86.4%	696	55.7%	55.8
KNN + randomized PCA	1.05	86.4%	468.8	56.06%	56.1
KNN + incremental PCA	0.76	86.4%	313.8	84.19%	84.2
KNN + LDA	0.05	84%	6.41	82.21%	82.21

Table: Testing and Training accuracies, time for KNN classifier with and w/o dimensionality reduction

Given below are various graphs, related to PCA and LDA dimensionality reduction techniques applied on KNN algorithm. I have plotted two different kinds of graphs.

1. Plot of testing and training accuracy vs no of dimensions for a constant k.
2. Plot of testing accuracy vs k for a constant number of dimensions in the dataset.

KNN with LDA:

Here the KNN algorithm has been applied after dimensionality reduction using LDA.

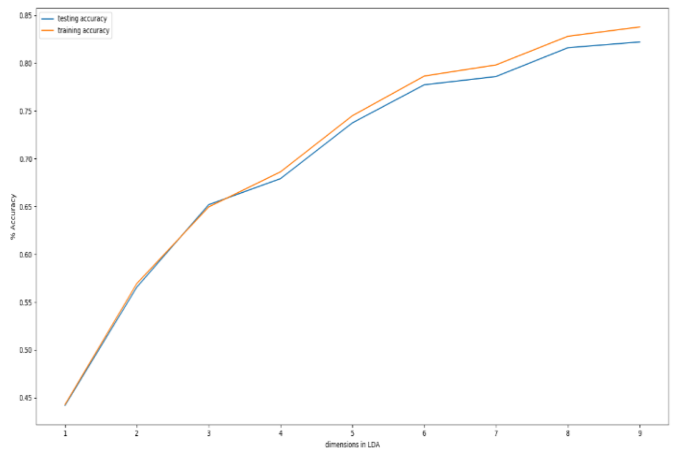


Fig: train and test accuracy for k = 5 and L_components from 1 to 9

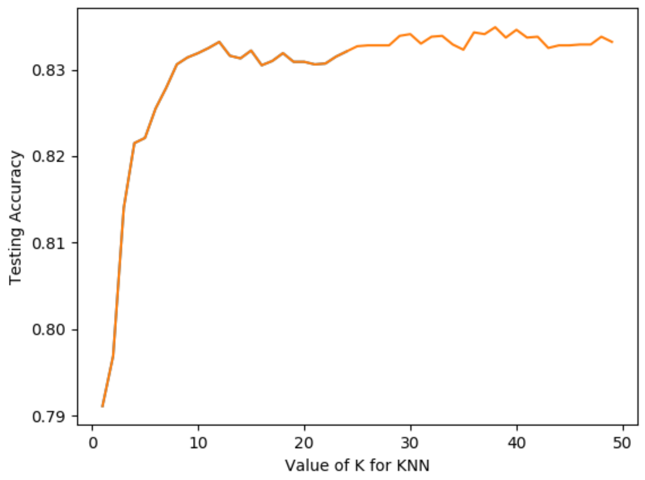


Fig: Change in testing accuracy for various k with constant L_components = 9

From the above plots we can conclude that the best parameters to get high classification accuracies are k = 5 and L_components = 9 when LDA dimensionality reduction is applied on the dataset.

KNN with incremental PCA:

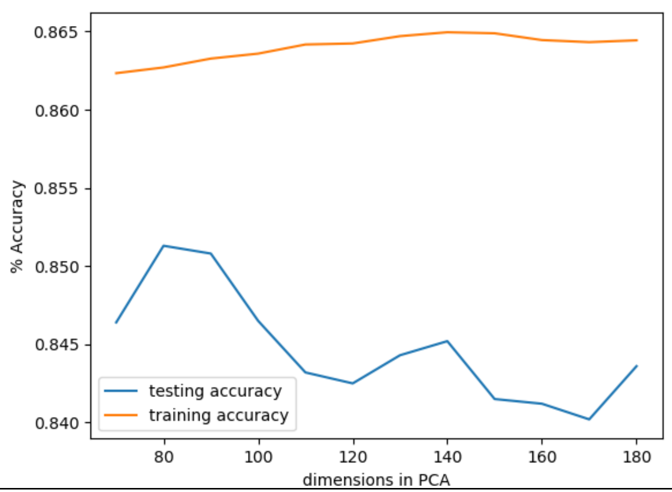
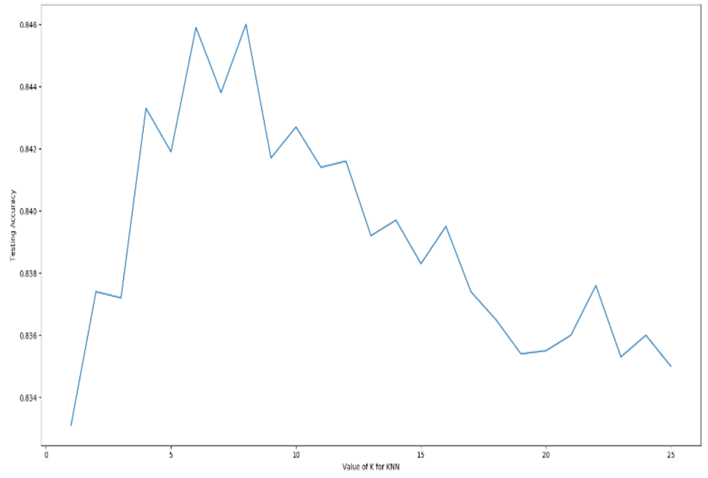
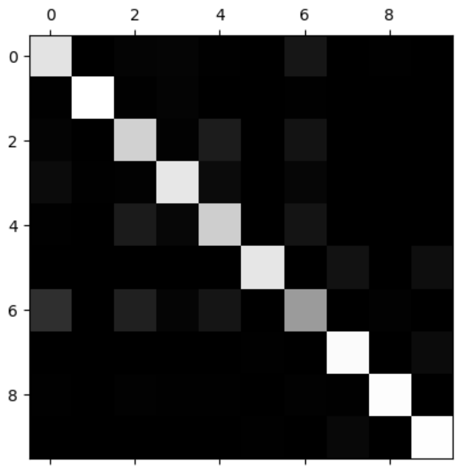
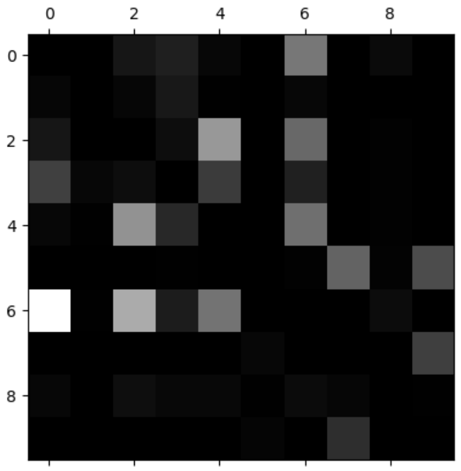
	

Fig: Plot of test and train accuracy vs PCA dimensions for k =5	Fig: Accuracy vs nearest neighbors for incremental PCA dimensions = 187

Thus, we infer from the graphs that for k= 5, we get highest of around 85% and for 187 dimensions, we get an accuracy of around 84.5%. Even though we get a high accuracy of 85% in the earlier case for k = 85, I wouldn't go for that method as, it leads to a large drop in the variance of the dataset. Most of the times we want to preserve the variance, and ideally it should be greater than 95%, to retain maximum amount of data. Thus, for k = 187, I found out that it preserved maximum amount of variance. Also, we can see that the training accuracy increases with an increase in the number of dimensions whereas the testing accuracy decreases for a fixed k = 5.

Analysis of confusion matrix for incremental PCA:





Fig(L to R): Confusion matrix, Normalized confusion matrix, Miss classified images

The left most matrix is the confusion matrix, obtained for the dataset. The one in center is a normalized confusion matrix. As seen in the normalized confusion matrix, the column for class 0 is bright, which tells us that class gets misclassified more often. Conversely, some rows are pretty dark. Which means they, those classes are generally classified correctly. Thus, analyzing the confusion matrix gives us better insights. The two 5×5 blocks on the left show images classified as class 6, and the two 5×5 blocks on the right show images classified as class 2. The confusion is justified as Class 6 is shirt whereas class 2 is a pullover. They both tend to have a similar structure. Analyzing confusion matrix gives better insights about improving the classifier. Ways to improve the classifier are, increase the data, preprocess the image, or engineer a new algorithm.

Conclusion

We conclude that, dimensionality reduction, not only improves the classifier's accuracy but it also reduces the computation time. Also, we observed that PCA is more suitable for KNN algorithm. And also, incremental PCA is better as compared to the simple PCA. The major observation for the project is that LDA dimensionality reduction method, performs equally well for both Bayes and KNN classifiers.

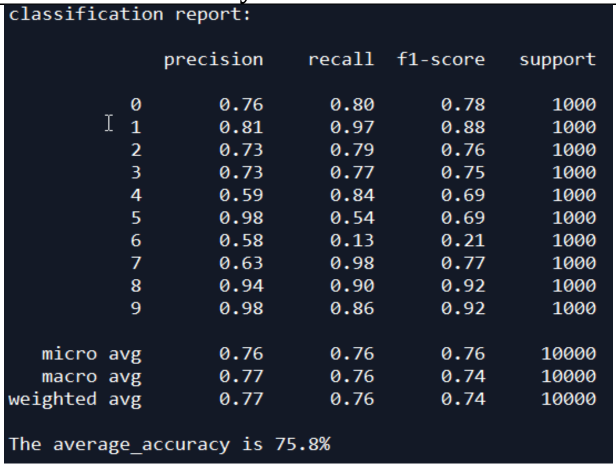
References:

1. FASHION-MNIST: https://github.com/zalandoresearch/fashion-mnist
2. SCIKITLEARN LDA: https://scikit-learn.org/0.16/modules/generated/sklearn.lda.LDA.html
3. BAYES CLASSIFIER : https://lazyprogrammer.me/bayes-classifier-and-naive-bayes-tutorial-using/
4. https://towardsdatascience.com/linear-discriminant-analysis-in-python-76b8b17817c2
5. https://github.com/Zolza09/Digit_fashion_MNIST/blob/master/Digit_MNIST_assignment1.ipynb
6. Class notes and medium, towards data science articles

Appendix

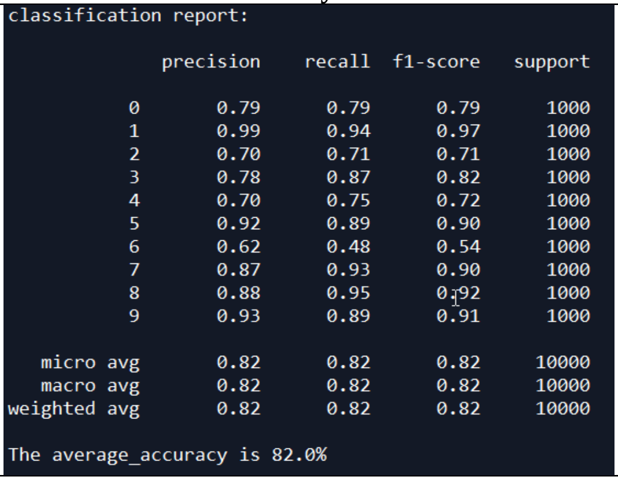
Bayes Classification reports:

Bayes Classifier



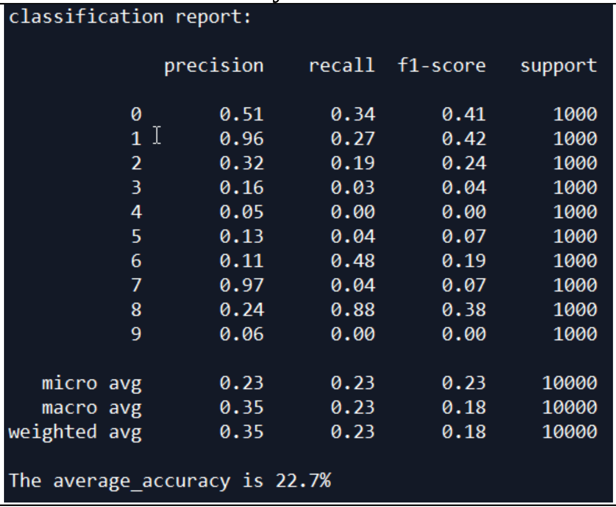
```
classification report:

              precision    recall  f1-score   support

           0       0.76      0.80      0.78      1000
           1       0.81      0.97      0.88      1000
           2       0.73      0.79      0.76      1000
           3       0.73      0.77      0.75      1000
           4       0.59      0.84      0.69      1000
           5       0.98      0.54      0.69      1000
           6       0.58      0.13      0.21      1000
           7       0.63      0.98      0.77      1000
           8       0.94      0.90      0.92      1000
           9       0.98      0.86      0.92      1000

   micro avg       0.76      0.76      0.76     10000
   macro avg       0.77      0.76      0.74     10000
weighted avg       0.77      0.76      0.74     10000

The average_accuracy is 75.8%
```

Bayes LDA



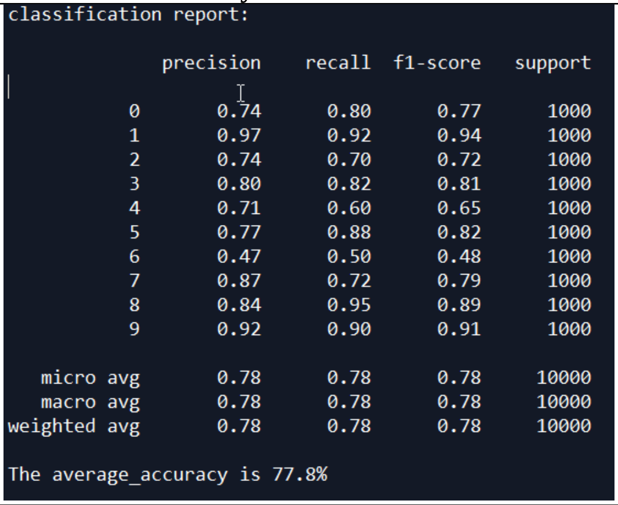
```
classification report:

              precision    recall  f1-score   support

           0       0.79      0.79      0.79      1000
           1       0.99      0.94      0.97      1000
           2       0.70      0.71      0.71      1000
           3       0.78      0.87      0.82      1000
           4       0.70      0.75      0.72      1000
           5       0.92      0.89      0.90      1000
           6       0.62      0.48      0.54      1000
           7       0.87      0.93      0.90      1000
           8       0.88      0.95      0.92      1000
           9       0.93      0.89      0.91      1000

   micro avg       0.82      0.82      0.82     10000
   macro avg       0.82      0.82      0.82     10000
weighted avg       0.82      0.82      0.82     10000

The average_accuracy is 82.0%
```

Bayes PCA



```
classification report:

              precision    recall  f1-score   support

           0       0.51      0.34      0.41      1000
           1       0.96      0.27      0.42      1000
           2       0.32      0.19      0.24      1000
           3       0.16      0.03      0.04      1000
           4       0.05      0.00      0.00      1000
           5       0.13      0.04      0.07      1000
           6       0.11      0.48      0.19      1000
           7       0.97      0.04      0.07      1000
           8       0.24      0.88      0.38      1000
           9       0.06      0.00      0.00      1000

   micro avg       0.23      0.23      0.23     10000
   macro avg       0.35      0.23      0.18     10000
weighted avg       0.35      0.23      0.18     10000

The average_accuracy is 22.7%
```

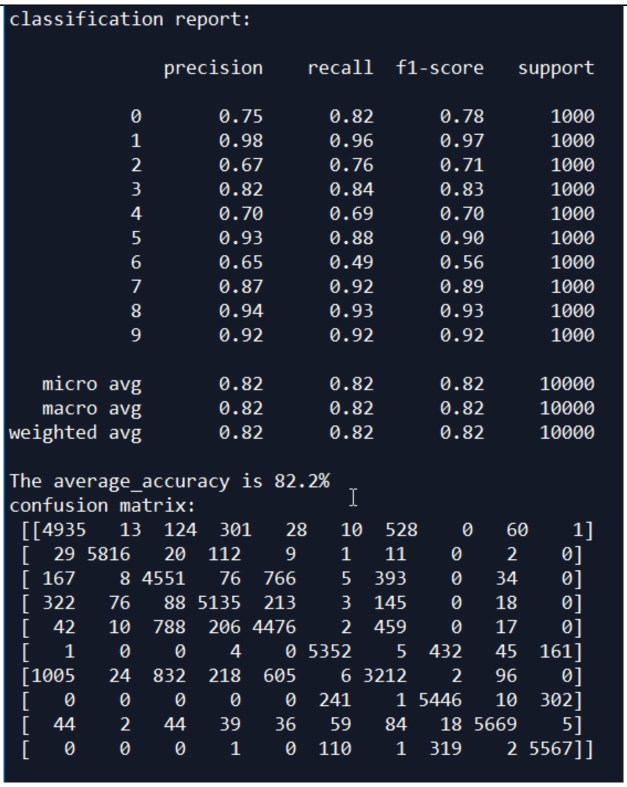
Bayes Incremental PCA



```
classification report:

              precision    recall  f1-score   support

           0       0.74      0.80      0.77      1000
           1       0.97      0.92      0.94      1000
           2       0.74      0.70      0.72      1000
           3       0.80      0.82      0.81      1000
           4       0.71      0.60      0.65      1000
           5       0.77      0.88      0.82      1000
           6       0.47      0.50      0.48      1000
           7       0.87      0.72      0.79      1000
           8       0.84      0.95      0.89      1000
           9       0.92      0.90      0.91      1000

   micro avg       0.78      0.78      0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.78      0.78      0.78     10000

The average_accuracy is 77.8%
```

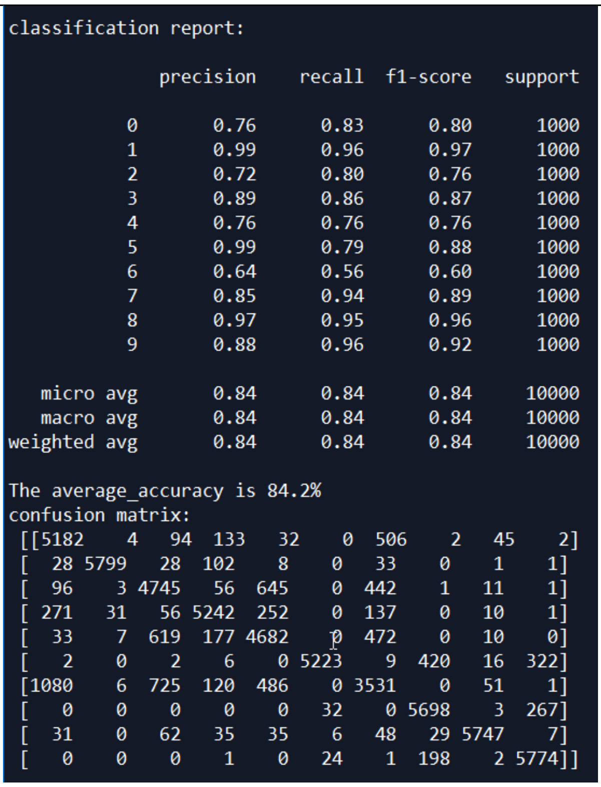
KNN Classification reports:

KNN_LDA



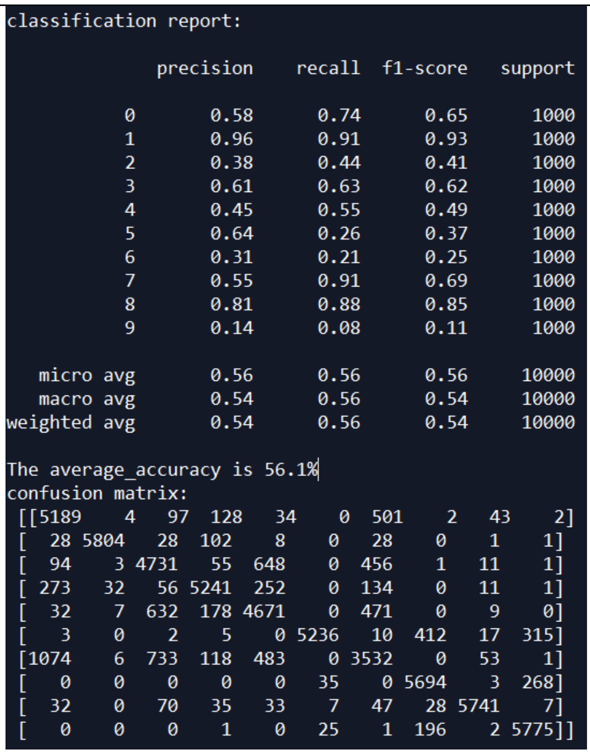
```
classification report:

              precision    recall  f1-score   support

           0       0.75      0.82      0.78      1000
           1       0.98      0.96      0.97      1000
           2       0.67      0.76      0.71      1000
           3       0.82      0.84      0.83      1000
           4       0.70      0.69      0.70      1000
           5       0.93      0.88      0.90      1000
           6       0.65      0.49      0.56      1000
           7       0.87      0.92      0.89      1000
           8       0.94      0.93      0.93      1000
           9       0.92      0.92      0.92      1000

   micro avg       0.82      0.82      0.82     10000
   macro avg       0.82      0.82      0.82     10000
weighted avg       0.82      0.82      0.82     10000

The average_accuracy is 82.2%
confusion matrix:
 [[4935   13  124  301   28   10  528    0   60    1]
 [  29 5816   20  112    9    1   11    0    2    0]
 [ 167    8 4551   76  766    5  393    0   34    0]
 [ 322   76   88 5135  213    3  145    0   18    0]
 [  42   10  788  206 4476    2  459    0   17    0]
 [   1    0    0    4    0 5352    5  432   45  161]
 [1005   24  832  218  605    6 3212    2   96    0]
 [   0    0    0    0    0  241    1 5446   10  302]
 [  44    2   44   39   36   59   84   18 5669    5]
 [   0    0    0    1    0  110    1  319    2 5567]]
```

KNN_INCREMENTAL_PCA



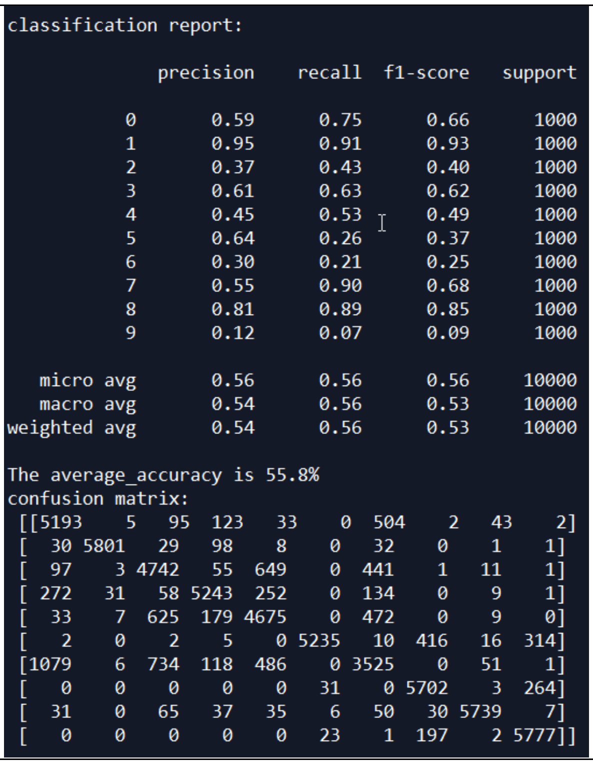
```
classification report:

              precision    recall  f1-score   support

           0       0.76      0.83      0.80      1000
           1       0.99      0.96      0.97      1000
           2       0.72      0.80      0.76      1000
           3       0.89      0.86      0.87      1000
           4       0.76      0.76      0.76      1000
           5       0.99      0.79      0.88      1000
           6       0.64      0.56      0.60      1000
           7       0.85      0.94      0.89      1000
           8       0.97      0.95      0.96      1000
           9       0.88      0.96      0.92      1000

   micro avg       0.84      0.84      0.84     10000
   macro avg       0.84      0.84      0.84     10000
weighted avg       0.84      0.84      0.84     10000

The average_accuracy is 84.2%
confusion matrix:
 [[5182    4   94  133   32    0  506    2   45    2]
 [  28 5799   28  102    8    0   33    0    1    1]
 [  96    3 4745   56  645    0  442    1   11    1]
 [ 271   31   56 5242  252    0  137    0   10    1]
 [  33    7  619  177 4682    0  472    0   10    0]
 [   2    0    2    6    0 5223    9  420   16  322]
 [1080    6  725  120  486    0 3531    0   51    1]
 [   0    0    0    0    0   32    0 5698    3  267]
 [  31    0   62   35   35    6   48   29 5747    7]
 [   0    0    0    1    0   24    1  198    2 5774]]
```

KNN RANDOMIZED PCA



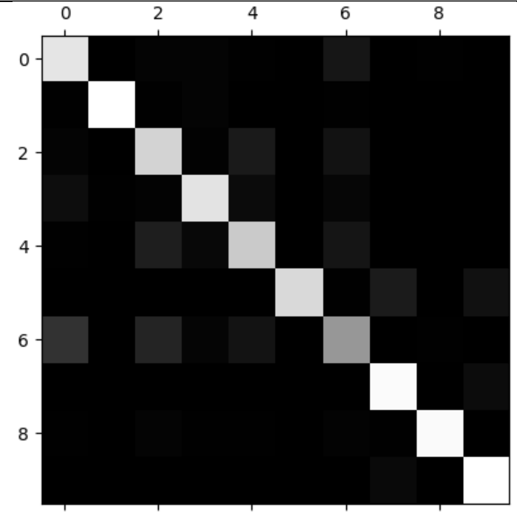
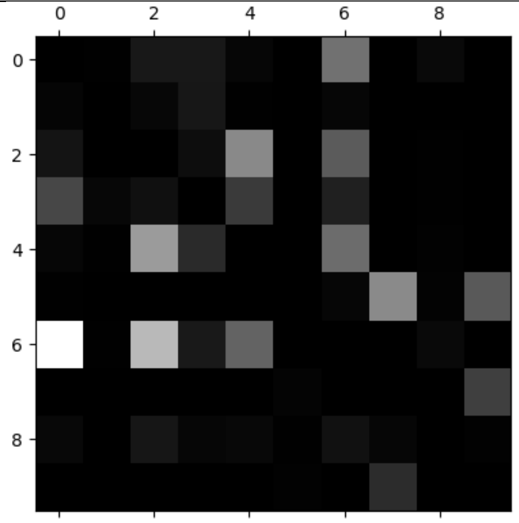
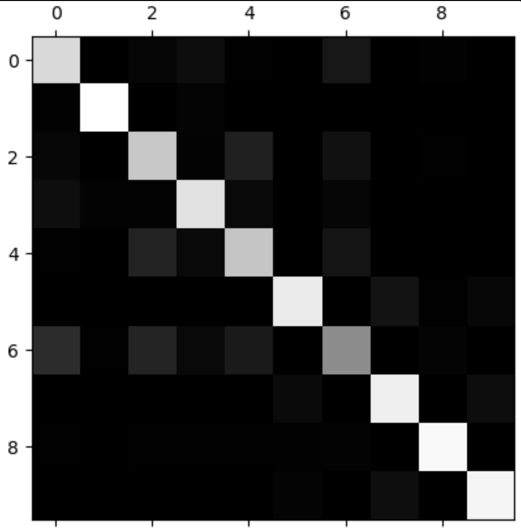
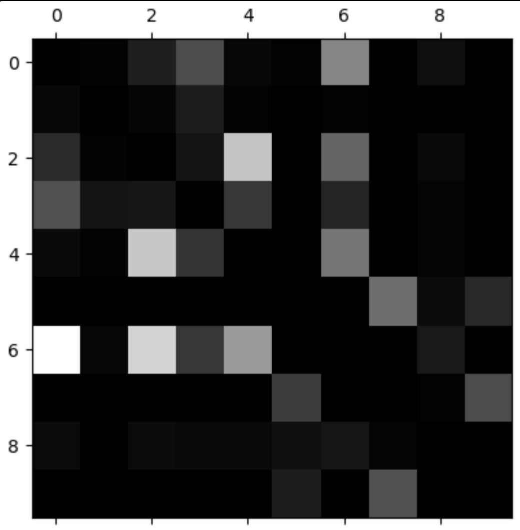
```
classification report:

              precision    recall  f1-score   support

           0       0.58      0.74      0.65      1000
           1       0.96      0.91      0.93      1000
           2       0.38      0.44      0.41      1000
           3       0.61      0.63      0.62      1000
           4       0.45      0.55      0.49      1000
           5       0.64      0.26      0.37      1000
           6       0.31      0.21      0.25      1000
           7       0.55      0.91      0.69      1000
           8       0.81      0.88      0.85      1000
           9       0.14      0.08      0.11      1000

   micro avg       0.56      0.56      0.56     10000
   macro avg       0.54      0.56      0.54     10000
weighted avg       0.54      0.56      0.54     10000

The average_accuracy is 56.1%
confusion matrix:
 [[5189    4   97  128   34    0  501    2   43    2]
 [  28 5804   28  102    8    0   28    0    1    1]
 [  94    3 4731   55  648    0  456    1   11    1]
 [ 273   32   56 5241  252    0  134    0   11    1]
 [  32    7  632  178 4671    0  471    0    9    0]
 [   3    0    2    5    0 5236   10  412   17  315]
 [1074    6  733  118  483    0 3532    0   53    1]
 [   0    0    0    0    0   35    0 5694    3  268]
 [  32    0   70   35   33    7   47   28 5741    7]
 [   0    0    0    1    0   25    1  196    2 5775]]
```

KNN SIMPLE PCA



```
classification report:

              precision    recall  f1-score   support

           0       0.59      0.75      0.66      1000
           1       0.95      0.91      0.93      1000
           2       0.37      0.43      0.40      1000
           3       0.61      0.63      0.62      1000
           4       0.45      0.53      0.49      1000
           5       0.64      0.26      0.37      1000
           6       0.30      0.21      0.25      1000
           7       0.55      0.90      0.68      1000
           8       0.81      0.89      0.85      1000
           9       0.12      0.07      0.09      1000

   micro avg       0.56      0.56      0.56     10000
   macro avg       0.54      0.56      0.53     10000
weighted avg       0.54      0.56      0.53     10000

The average_accuracy is 55.8%
confusion matrix:
 [[5193    5   95  123   33    0  504    2   43    2]
 [  30 5801   29   98    8    0   32    0    1    1]
 [  97    3 4742   55  649    0  441    1   11    1]
 [ 272   31   58 5243  252    0  134    0    9    1]
 [  33    7  625  179 4675    0  472    0    9    0]
 [   2    0    2    5    0 5235   10  416   16  314]
 [1079    6  734  118  486    0 3525    0   51    1]
 [   0    0    0    0    0   31    0 5702    3  264]
 [  31    0   65   37   35    6   50   30 5739    7]
 [   0    0    0    0    0   23    1  197    2 5777]]
```

KNN Confusion matrices

	Confusion matrix	Normalized confusion matrix
SIMPLE KNN		
KNN + LDA		
KNN + INC PCA