Machine Learning Assignment 1 - Group: 5

Tutor: Iwan Budiman

Anshu Kumar (490517666)
Sanna Nazir (490517677)
Samarth Sehgal (490528857)

October 2019

Contents

1 ABSTRACT **3**

2 INTRODUCTION **3**

2.1 Aim (The goals to be achieved) And Motivation (Understanding the why) 3

2.2 Dataset Overview 3

2.2.1 Displaying random images from the dataset 4

2.2.2 Displaying samples across each class 4

2.2.3 Representing data in 2D (Using PCA) 5

2.2.4 Representing data in 3D (Using PCA) 5

3 METHODS **6**

3.1 Data Pre-processing 6

3.1.1 Standardizing 6

3.1.2 Principal Component Analysis 6

3.2 Classifier Creation 7

3.2.1 Naive Bayes Classifier 8

3.2.2 Logistic Regression Classifier 8

3.2.3 K-Nearest Neighbours Classifier 8

3.2.4 Neural Network Classifier 9

4 EXPERIMENTATION AND RESULTS **9**

4.1 Experimentation 9

4.1.1 Naive Bayes Algorithm: 10

4.1.2 Logistic Regression Algorithm: 10

4.1.3 K-Nearest Neighbours Algorithm: 11

4.1.4 Neural Network Algorithm: 12

4.2 Results 13

5 DISCUSSION **14**

5.1 Impact of Pre-processing: 14

5.2 Impact of Fine-tuning parameters of algorithms: 14

6 CONCLUSION **15**

7 BIBLIOGRAPHY **16**

8 APPENDIX **17**

8.1 Hardware and Software Specifications 17

8.2 Instructions for running the code 17

1 ABSTRACT

A task of multi-class image classification is assigned. Four different algorithms/classifiers are designed from scratch, analyzed and evaluated. The report discusses the significance of each classifier, impact of changing the hyper-parameters of a specific classifier and their results post experimentation. Further, the report also explores the impact of data pre-processing techniques on the classifiers. The choice of the final classifier is made after extensive analysis of each classifier based on various parameters such as computational time, ease of understanding of algorithm as well as resulting accuracy without over-/under-fitting.

2 INTRODUCTION

The purpose of this report is to evaluate various classifier models for a machine learning task of multi-class classification problem. The data used for training, testing and evaluating the model is the publicly available dataset titled "Fashion MNIST Dataset." The report is backed by a set of experiments and methods which can be explored in the corresponding code file attached with the same. The key aspects of the report are discussed below:

2.1 Aim (The goals to be achieved) And Motivation (Understanding the why)

The aim of this project and the report may be divided into two separate levels:

Microscopically, at a student level, the project is aimed at inculcating machine learning skills in the student. By performing each sub-step of the project, ranging from data cleaning to final predictions using the classifiers built; the student is expected to gain diverse skills required for implementing the classroom skills on real-life problems. Furthermore, by designing of various separate classifier, the goal of critical thinking and analysis at the student level is hoped to be achieved. Summarizing the experiments, their results and final conclusion of choice of a classifier based on various trade-off factors is aimed at developing essential analytical skills.

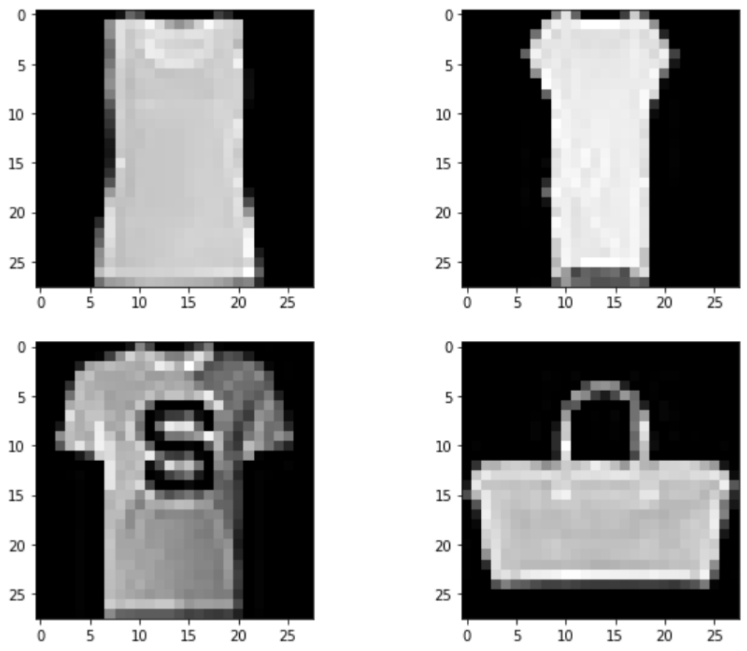
Macroscopically, the study is aimed at understanding the growing need for image-based classifiers in the realm of technology. With the explosion of data in the modern world coupled with the growing propensity of consumers for digital shopping trends, e-commerce has seen an exponential growth. A key task in building such e-commerce platforms would be the segregation of garments in various classes of human-understandable sections for instance, a section for pants and a separate one for pants. By executing this project and assignment, the student can become efficient in such a task of specific textile classification based on images.

More specifically, the microscopic aims of this project may be defined as: Analysing the raw dataset. Dividing the annotated dataset into training and validation set. Pre-processing the data for better performance. Studying different Machine Learning and Deep Learning algorithms and selecting those that solve the given problem domain with high accuracy. Creating the required classifiers and carry out the experimentation Comparison of techniques and draw educated inferences from the same. Concluding the study with closing remarks and personal reflection.

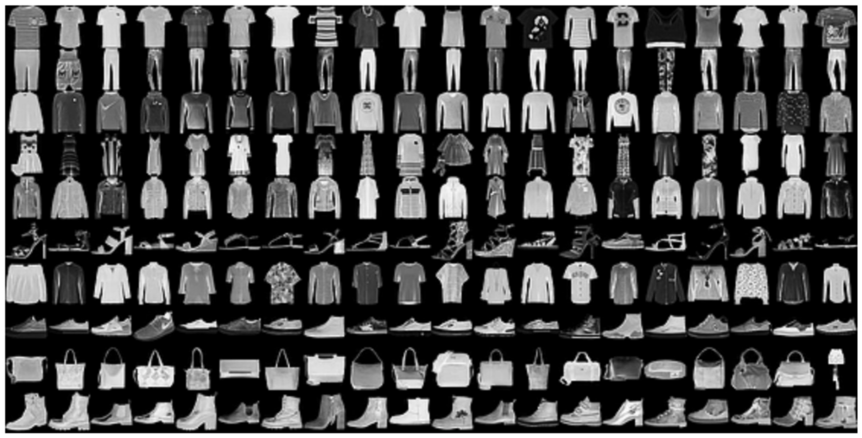
2.2 Dataset Overview

The Dataset has been provided by the unit lecturer. The same is a publicly available dataset titled as 'Fashion MNIST Data.' The original dataset contains 60,000 example images with a test set of 10,000 example images. For our use in this project, the dataset has been modified to contain a training dataset of 30,000 labelled images and a testing dataset of 2000 labelled images. Each image is a $(28 * 28)$ pixel matrix in grayscale.

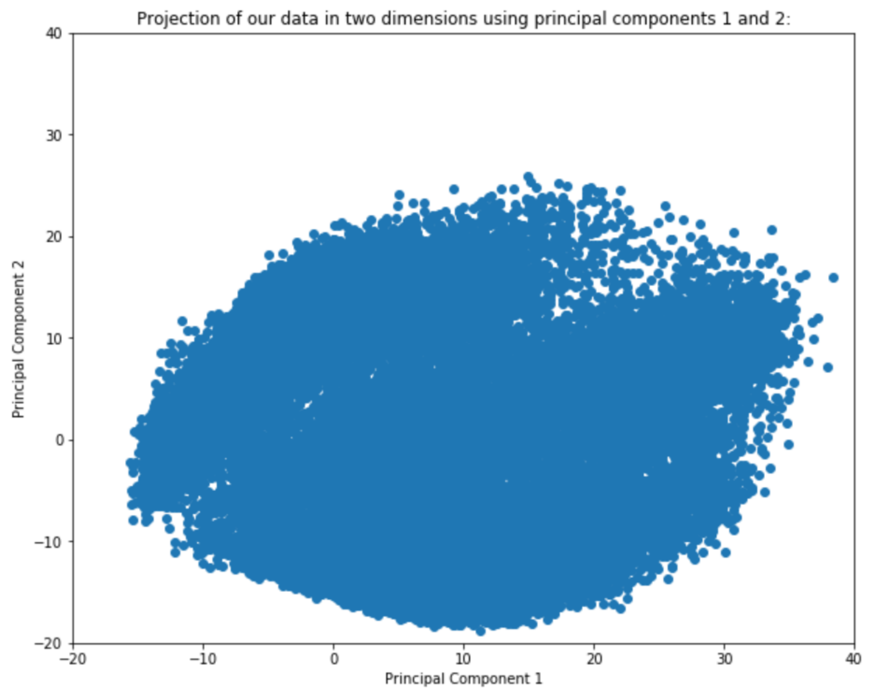
2.2.1 Displaying random images from the dataset



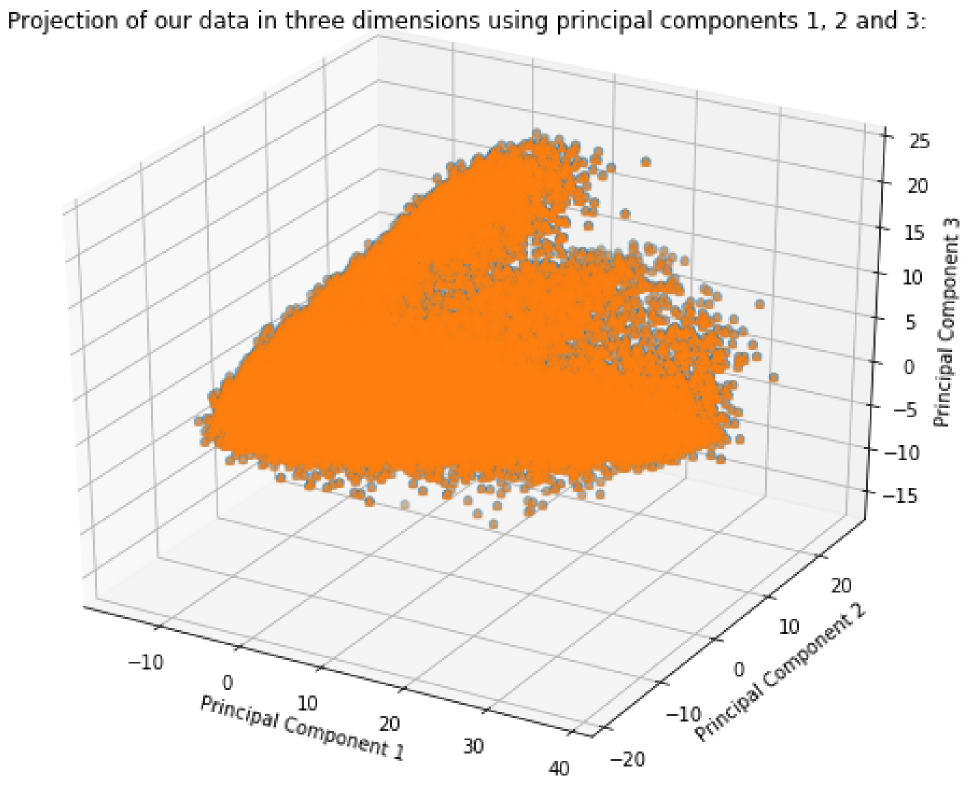
2.2.2 Displaying samples across each class



2.2.3 Representing data in 2D (Using PCA)



2.2.4 Representing data in 3D (Using PCA)



3 METHODS

The methods used for our task may be broadly classified into two parts:

PART A: Data Pre-processing and PART B: Classifier Creation

3.1 Data Pre-processing

"Pre-processing data is an essential step to enhance data efficiency." [2] Data pre-processing is the first step in creating any data or machine learning model. As the famous saying goes, 'Garbage in, Garbage out.' A machine learning model is only as good as the data provided to it. Biased, raw, un-clean data will result in a poorly performing model. Furthermore, raw data often consists of co-related features which can only aid in increasing the complexity and computation time of the algorithm. Data pre-processing also helps in eliminating such highly co-related features. Some of the popular methods used for data pre-processing include standardizing the data (across a specific parameter), normalizing the data (to have values within a specific range only), eliminating null values or manually filling them, dimensionality reduction techniques etc.

For our project, we have used the following two techniques:

3.1.1 Standardizing

Standardization or Normalization is a feature scaling technique used in data pre-processing step of creating any data model or machine learning algorithm. Various methods for standardization are used including the Min-Max normalization and the Z-score Normalization. For our project, we have used the Z-score Normalization. This technique is one of the most popular ones for processing and scaling your data before inputting it into an algorithm. The method makes use of the mean and variance of each feature in the dataset.

Put mathematically, a z-score is defined as:

$$z = \frac{x_i - x_{mean}}{\sigma}$$

where:
x_i is a data point
x_{mean} is the mean of the data points across a feature
σ is the variance of that feature.

3.1.2 Principal Component Analysis

Principal Component Analysis is a popular, reliable technique for dimensionality reduction of large datasets. The relevance of PCA as a dimension reduction technique is increasing even more with the boom in information experienced in modern age. Furthermore, PCA is also used a s a data visualization technique for datasets represented in more than 2 or 3 dimensions. It allows projection of all dimensions into a 2D or 3D space.

"The description of Principal Component Analysis is made by means of the explanation of eigenvalues and eigenvectors of a matrix" [3] PCA makes use of the covariance matrix of any data matrix to deduce the eigen value, eigen vector pairs. It then uses these eigen pairs to understand the spread of the data across it's principal components. Principal Components may be defined as the axes along which the projection of entire dataset reserves the maximum information. The steps involved in a PCA dimensionality reduction may be described as follows:

1. Find the covariance matrix (say, C) of the required data set (say, X).

Covariance matrix of a matrix C with n rows and n columns may be defined as:

$$C^{n\times n} = (c_{i,j}, c_{i,j} = cov(Dim_i, Dim_j))\ [5]$$

2. Calculate the eigen value and eigen vector pairs from the covariance matrix.

3. Check various eigen pair combinations to understand the information retention percentage.

Retention Percentage of information may be calculated by:

$$R = \left(\frac{sum\ of\ largest\ k\ eigen\ values}{sum\ of\ all\ eigen\ values}\right) \times 100$$

4. Select appropriate 'k' value for your purpose and create the new projection matrix, W.

5. Apply the classifier on the newly created, feature-reduced matrix.

$$X' = dot\ product\ of\ X\ and\ W$$

3.2 Classifier Creation

Classifier is an umbrella term used here for the various image multi-class classifiers designed for this project. A classifier, in lay man's terms, may be defined simply as an algorithm that classifies an unknown data point (here, an image) into a pre-defined set of classes or labels.

The problem at hand is a multi-class classification problem. This signifies that the output labels or classes are more than 2. Two output classes is referred to as a binary classification problem. Consequently, classifier approach to a multi-class problem is changed. This is done in two different ways:
1. One-versus-One: Each class is compared to each different class in a collection of many binary classifications. Thus, increasing the number of classifications to be done.
2. One-versus-All: Each class is compared to the rest of the classes treated collectively as one class. Thus, reducing the number of classifications to be done.

We have used four different types of classifiers for the purpose of training the Fashion MNIST dataset provided to us. These are discussed below in detail:

3.2.1 Naive Bayes Classifier

The Naive Bayes Classification model is a probabilistic model that assumes conditional independence of all feature vectors of a certain class. This model is based on the tenets of conditional probability and joint distributions. "In the learning process of this classifier with the known structure, class probabilities and conditional probabilities are calculated using training data, and then values of these probabilities are used to classify new observations." [4]

Because the classifier assumes a complete independence between features, it is termed as 'Naive.' Naive refers to an artless, inexperienced individual. In terms of this algorithm, naive means that the classifier is making assumptions that may not turn out to be true for majority real-life situation. Thus, it's inexperienced and naive.

The Naive Bayes classifier is founded on the "Bayes Theorem of Probability" which states:

$$P(A|B)P(\dot{B}) = P(B|A)P(\dot{A}) \tag{1}$$

The Naive Bayes classifier describes the joint probability distribution for a binary class problem as below:

$$P(X = x, Y = c) = P(Y = c)P(X = \dot{x}|Y = c) \tag{2}$$

3.2.2 Logistic Regression Classifier

The Logistic Regression Model is an extension of the Naive Bayes Classification Model. It works on the fundamental principle of creating a boundary between various classes. This boundary may or may not be linear. For binary logistic classification problem, the sigmoid function is used to delineate the class boundary. Simply put, if the classification equation function yields a value less than a certain threshold, the input data point is classified as one class and if it yields an output above the specified threshold, then its classified into the other class.

The popular function used for probability estimation in the logistic model is the "sigmoid function." This function may be mathematically defined as:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{3}$$

For the purpose of the logistic regression classifier, 'a' is equivalent to the dot product of the feature (X) and the transpose of the weights for these features i.e.

$$a = w^T \dot{x}$$

3.2.3 K-Nearest Neighbours Classifier

kNN is an extension of the Nearest Neighbours algorithm. For KNN, the 'k' refers to the number of neighbours considered during the execution of the classification algorithm. As the name clearly suggests, this algorithm considers the closest 'k' neighbours in an unsupervised technique of classification. Simply put, it garners votes from k-nearest neighbours of the unknown data point, checks and evaluates the votes and which class these neighbours belonged to; and then classifies the data point based on certain confidence level.

Popular technique for evaluating the distance of any data point from its neighbours is the Euclidean distance method. The Euclidean distance (say D) between two data points defined as X and Y may be given mathematically as:

$$D = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4}$$

3.2.4 Neural Network Classifier

Neural Networks are the most human-emulating classifiers. Originally, 'neural' refers to the neurons and the nervous system within a human being. The neurons represent the simplest mathematical operation performed within our nerves wherein an input is taken, a function performed and then an output is generated. This output then serves on to be an input for the next connected neuron. So on the system of billions of neurons make up the human nervous system.

Neural Networks in technology aim to imitate this property of the human body. A digital neural network is comprised of 'perceptrons.' A perceptron is analogous to a neuron, just digital! It takes an input, applies some function on that input and then gives an output based on some threshold function. This function is known as the 'Activation Function.' Quite simply put, it is the function that activates (or not) the perceptron.

Neural Classification is a black-box technique for unsupervised classification. This means, that the internal functioning of the neural layers is imperceptible to a human being. It just learns patterns based on input data, then reduces errors by back propagation and eventually comes up with a fairly accurate classifier. These are the most popular classification method used nowadays. This is because they require high computation resources which are available due to increasing improvements in hardware.

4 EXPERIMENTATION AND RESULTS

4.1 Experimentation

This section describes the experiments carried out using the algorithms described above, on our required dataset. And the impact of fine-tuning various algorithmic parameters on the final predictions. Furthermore, the time taken by each algorithm in performing the test was also tracked.

The training data used is a $(30,000 \times 784)$ matrix titled 'data train.' The testing dataset is a $(10,000 \times 784)$ matrix which is then sliced into a $(2,000 \times 784)$ matrix for the purposes of our algorithm's accuracy measurements. The 2,000 labels have been provided by the lecturer for the task execution while the remaining 8,000 are to be tested discreetly.

NOTE: As per the assignment instructions, it was mentioned that the test set only contains 5,000 images, however, it contains 10,000 images on inspection. From here-on, it is assumed that test set is a $(10,000 \times 784)$ matrix.

4.1.1 Naive Bayes Algorithm:

The Naive Bayes algorithm was built from scratch using the numpy, pandas and allowed required libraries. The same was executed with dimensionality reduction using PCA and without PCA as well. This algorithm was trained on the data of 30,000 train images and tested on the test data of 2,000 images.

The resulting performance of the algorithm is as under:

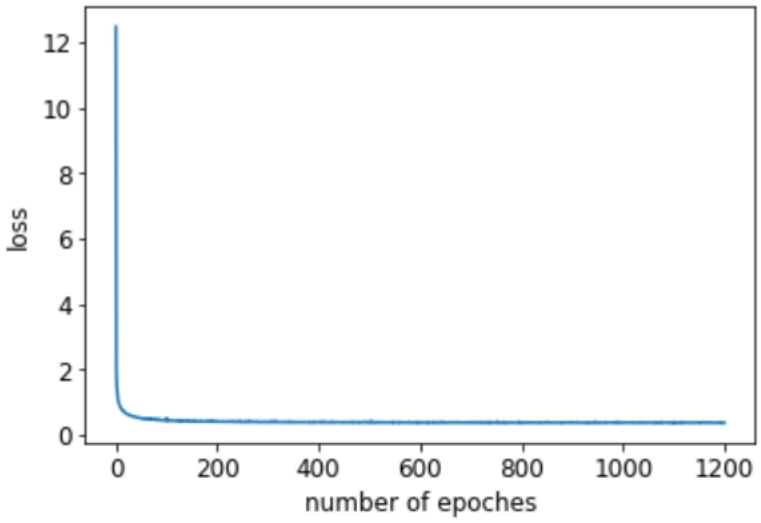
Parameter	With PCA	Without PCA
Accuracy (training)	9.87	66.65
Accuracy(testing)	10.5	66.64
Time Taken	0.013	0.013

4.1.2 Logistic Regression Algorithm:

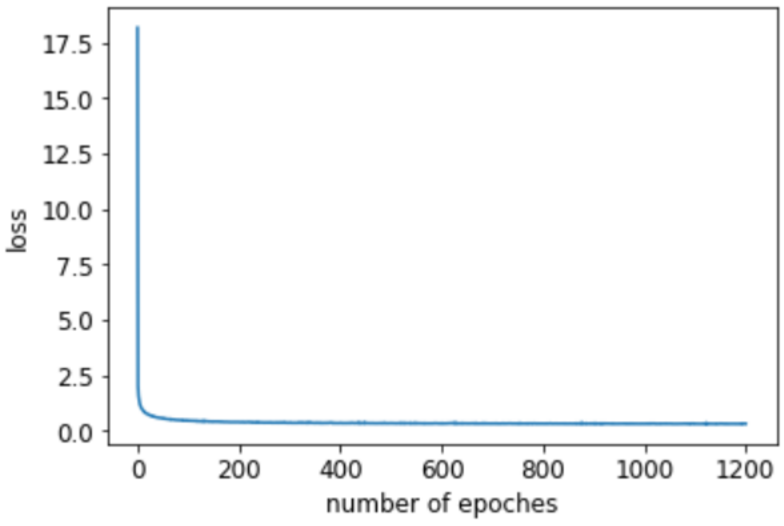
The Logistic Regression algorithm was built from scratch using the numpy, pandas and allowed required libraries. The same was executed with dimensionality reduction using PCA and without PCA as well. This algorithm was trained on the data of 30,000 train images and tested on the test data of 2,000 images.

The resulting performance of the algorithm is as under:

Parameter	With PCA	Without PCA
Accuracy (training)	87.11	88.63
Accuracy(testing)	81.75	82.05
Time Taken (min)	3.74	8.6



Logistic Regression with PCA: Epochs Vs Loss graph



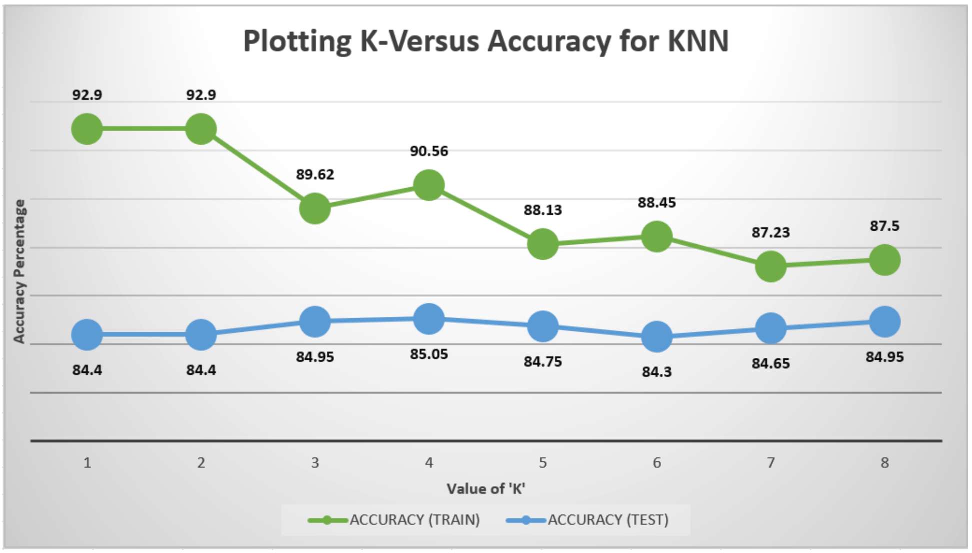
Logistic Regression without PCA: Epochs Vs Loss graph

4.1.3 K-Nearest Neighbours Algorithm:

The K-Nearest Neighbours algorithm was built from scratch using the numpy, pandas and allowed required libraries. The same was executed with dimensionality reduction using PCA and without PCA as well. This algorithm was trained on the data of 30,000 train images and tested on the test data of 2,000 images.

The resulting performance of the algorithm is as under:

Parameter	With PCA	Without PCA
Accuracy (training)	90.02	90.56
Accuracy(testing)	85.2	85.05
Time Taken	2.6	2.51



Impact of changing 'k' for KNN Classifier: Accuracy Vs 'K' graph for the test and train sets

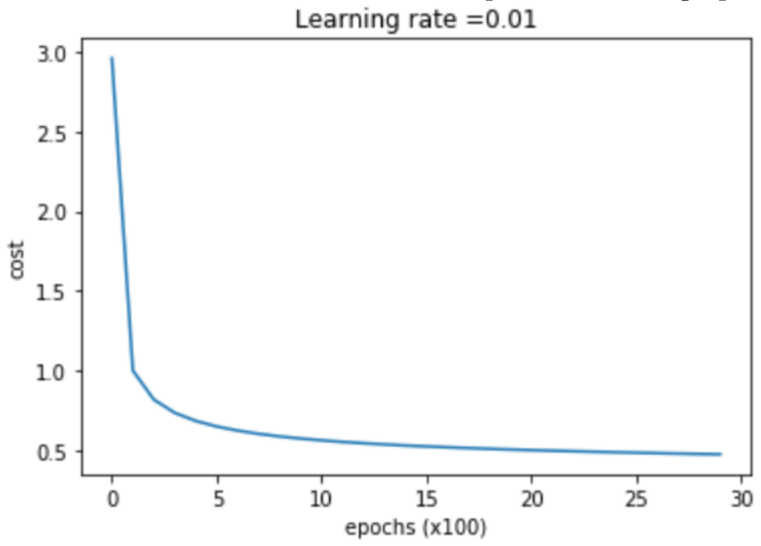
4.1.4 Neural Network Algorithm:

The Neural Network algorithm was built from scratch using the numpy, pandas and allowed required libraries. The same was executed with dimensionality reduction using PCA and without PCA as well. This algorithm was trained on the data of 30,000 train images and tested on the test data of 2,000 images.

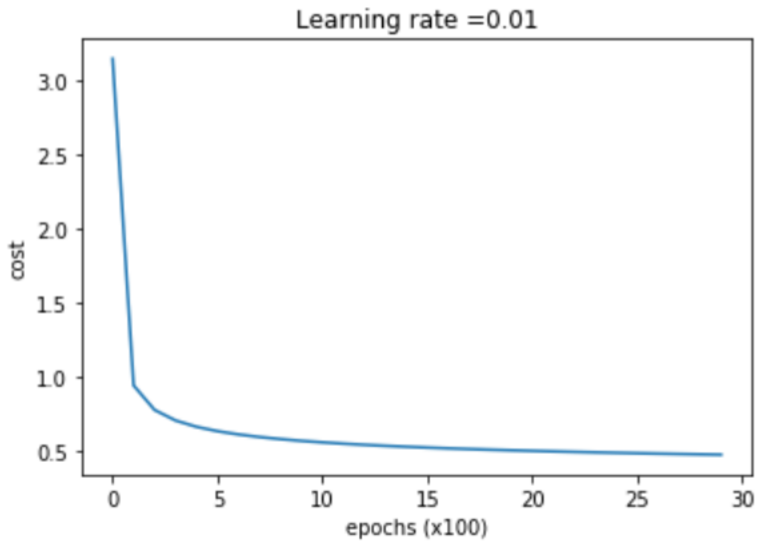
The resulting performance of the algorithm is as under:

Parameter	With PCA	Without PCA
Accuracy (training)	83.56	83.98
Accuracy(testing)	82.65	81.9
Time Taken (min)	9.35	16.2

Neural Network without PCA: Epochs Vs Cost graph



Neural Network with PCA: Epochs Vs Cost graph



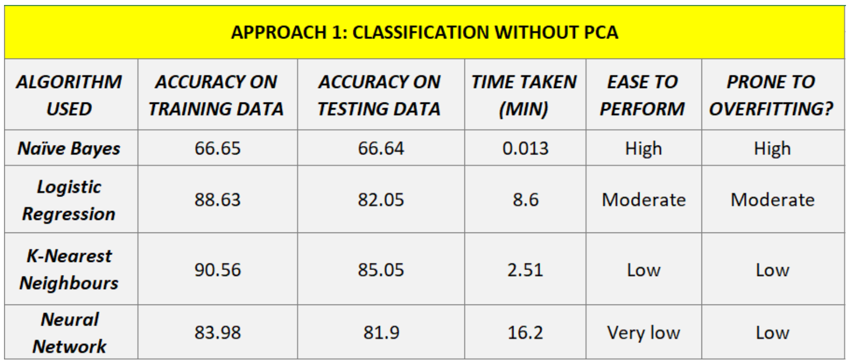
4.2 Results

The results from conducting the above task of classification using various classifiers on our dataset has yielded promising results. These results can broadly be studied under two categories: QUALITATIVE and QUANTITATIVE.

Qualitatively, we have understood the grass-root, mathematical working of all the above four used classifiers. We understand the the efficiency of each classifier when faced with a multi-class classification problem. Furthermore, we can also understand the impact of fine tuning of parameters on the accuracy, computation cost and computation time of each algorithm.

Quantitatively, the results of the experimentation can be summarized as under:

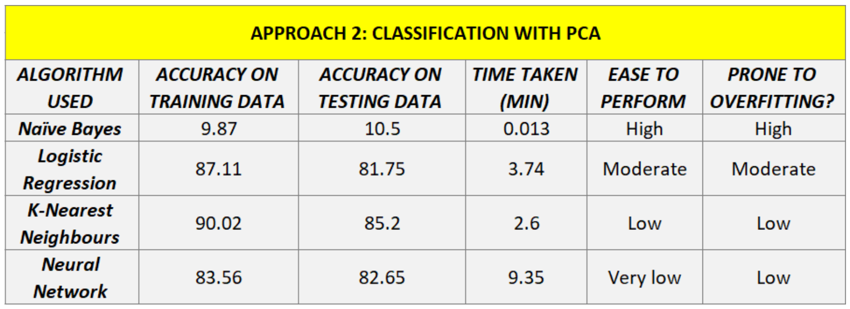
FIRST APPROACH: WITHOUT PCA



APPROACH 1: CLASSIFICATION WITHOUT PCA

ALGORITHM USED	***ACCURACY ON TRAINING DATA***	***ACCURACY ON TESTING DATA***	***TIME TAKEN (MIN)***	***EASE TO PERFORM***	***PRONE TO OVERFITTING?***
Naïve Bayes	66.65	66.64	0.013	High	High
Logistic Regression	88.63	82.05	8.6	Moderate	Moderate
K-Nearest Neighbours	90.56	85.05	2.51	Low	Low
Neural Network	83.98	81.9	16.2	Very low	Low

SECOND APPROACH: WITH PCA



APPROACH 2: CLASSIFICATION WITH PCA

ALGORITHM USED	***ACCURACY ON TRAINING DATA***	***ACCURACY ON TESTING DATA***	***TIME TAKEN (MIN)***	***EASE TO PERFORM***	***PRONE TO OVERFITTING?***
Naïve Bayes	9.87	10.5	0.013	High	High
Logistic Regression	87.11	81.75	3.74	Moderate	Moderate
K-Nearest Neighbours	90.02	85.2	2.6	Low	Low
Neural Network	83.56	82.65	9.35	Very low	Low

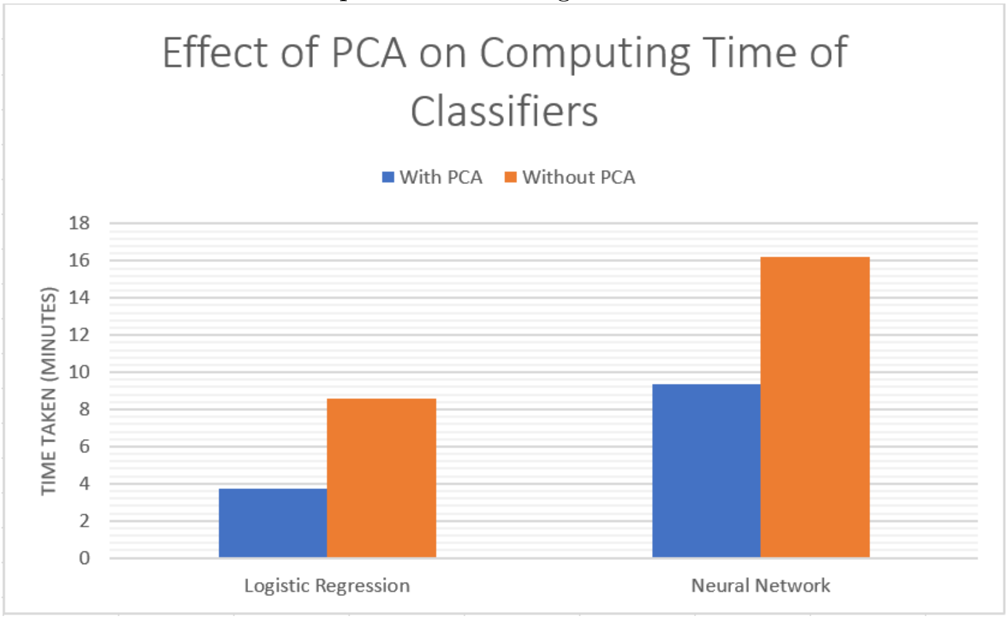
5 DISCUSSION

As seen above, the post experimentation results show a range of performances. From the least performing algorithm being Naive Bayes after PCA, to the best performing algorithm being K-Nearest Neighbours, both with and without PCA.

5.1 Impact of Pre-processing:

The impact of pre-processing and reduction in dimensionality can be clearly seen in the reduction of computational times of the classifiers. For the Logistic Regression Model, five minutes of computing time are saved immediately with just a trade-off of one percent in the accuracy. Similarly, for the Neural Classifier, PCA reduces the computational time by almost half. Reducing the running time from approximately sixteen minutes to nine minutes. The trade-off in accuracy is negligible here.

Reduction in computation time of given classifiers after PCA



In case of Naive Bayes, it is seen that PCA dramatically reduces the accuracy of the algorithm. This may be because PCA is essentially a technique to map and evaluate the variance of features. However, it doesn't take into account the impact of the features on the resulting labels. This may be causing the problem as seen. For such an issue, we hypothesize that LDA or Linear Discriminant Analysis may be a more appropriate technique for dimensionality reduction. Since LDA considers the inter-class variance, it studies the impact of features on classes as well as their variance. However, the hypothesis remains to be tested in another report.

5.2 Impact of Fine-tuning parameters of algorithms:

As seen above, certain changes in the algorithmic parameters results in impact on the accuracy and run-time of the same. These can be summarized as below:

- For Naive Bayes, we experimented the changing of learning rates for classification impact. However, the impact was negligible and hence, discarded.
- For Logistic Regression, tuning of number of epochs was analyzed with impact on the loss function. As can be seen in the graph above (under section 4.1.2.), the loss function decreases sharply at the beginning and then appears to smoothen over number of iterations. While the loss function keeps decreasing even after the sharp decline, the change is quite slow. This effect is seen in both PCA and without PCA approaches.
- For the K-Nearest Neighbours Algorithm, we have fine tuned the parameters 'K' and the batch size for processing. While the batch size has no impact on the end results, 'K' has a significant impact on the final accuracy. As seen in the chart above (under section 4.1.3.), accuracy is maximized for 'K = 1 and 2' and then decreases at 'K=3', again increases for 'K=4' and then keeps decreasing for higher 'K.' This trend is replicated for the testing data as well. And can be seen in both PCA and non-PCA approaches.
- For the Neural Network classifier, we have fine tuned the number of iterations versus the cost (loss) function for a learning rate of 0.01. As can be seen in the graphs above (under section 4.1.4.), the cost function decreases monotonically with increasing epochs. That is, they are inversely proportional. After about 1500 epochs, the decrease is less sharp and almost imperceptible. This effect is seen in both PCA and without PCA approaches.

6 CONCLUSION

After extensive experimentation and analysis, we conclude that for the purposes of our dataset, K Nearest Neighbours serves as the best classification algorithm. As can be seen below, KNN performs equally well (with a 1 variation in accuracy) for PCA and non PCA approaches. Furthermore, it takes a comparatively small computation time in running and provides a very high accuracy. Having close to 90 percent accuracy for the training data and above 85 percent on the training set.

The below table is represented for the K value of 4. The choice of this value is based on the analysis of (K versus Accuracy) graph of the KNN classifier. While accuracy achieved at less than four K values is higher, the difference between training and testing accuracy is high. This signifies that the K values are prone to overfitting. For K value of four, the difference (or gap) between training and testing accuracy is minimized and an above 85 percent accuracy is still retained in the testing set. Thus, choice of K at four is justified.

Reflecting on the future work for such a project, we believe using higher end Neural Networks would be the best approach. K-NN is limited in its efficiency by tweaking of parameter 'k.' There is only so much that the classifier can do.On the other hand, Neural Classification is a promising, developing new field. The current Neural Network we designed is a single layer network which is fully connected. However, modern day research has achieved immensely powerful accuracy results using multi-layered neural networks, convolutional neural networks and so on. Thus, we propose the building of a better neural netowrk as a further step to this project.

7 BIBLIOGRAPHY

REFERENCES

[1] Ian T. Joliffe, Jorge Cadima. Principal component analysis: a review and recent developments. PMC [Internet]. 2012 April [cited October 2019]; PMC4792409: 26953178. Available from: cbi.nlm.nih.gov/pmc/articles/PMC4792409/

[2] Wesam S. Bhaya. Review of Data Preprocessing Techniques in Data Mining. Journal of Engineering and Applied Sciences [Internet]. 2017 September [cited October 2019]; 12. 4102-4107. 10.3923. Available from: https://www.researchgate.net/publication/319990923_Review_of_Data_Preprocessing_Techniques_in_Data_Mining

[3] Espirito Santo, Rafael. Principal Component Analysis applied to digital image compression. Einstein (São Paulo, Brazil). 10. 135-9. 10.1590/S1679-45082012000200004.

[4] Sona Taheri, Musa A. Mammadov. Learning the naive Bayes classifier with optimization models. Applied Mathematics and Computer Science. 2013 [cited October 2019]; DOI:10.2478/amcs-2013-0059

[5] Lindsay I. Smith. A tutorial on Principal Component Analysis [Internet]. February 26, 2002. Available from: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

8 APPENDIX

8.1 Hardware and Software Specifications

The code was created using Google Colabratory Open Source Computing Platform. However, we used a laptop for running the same online, which has below features:

- Processor: Intel i7
- OS: Windows 10
- RAM: 8 GB
- GPU: 4 GB
- Model: HP Pavilion 360

8.2 Instructions for running the code

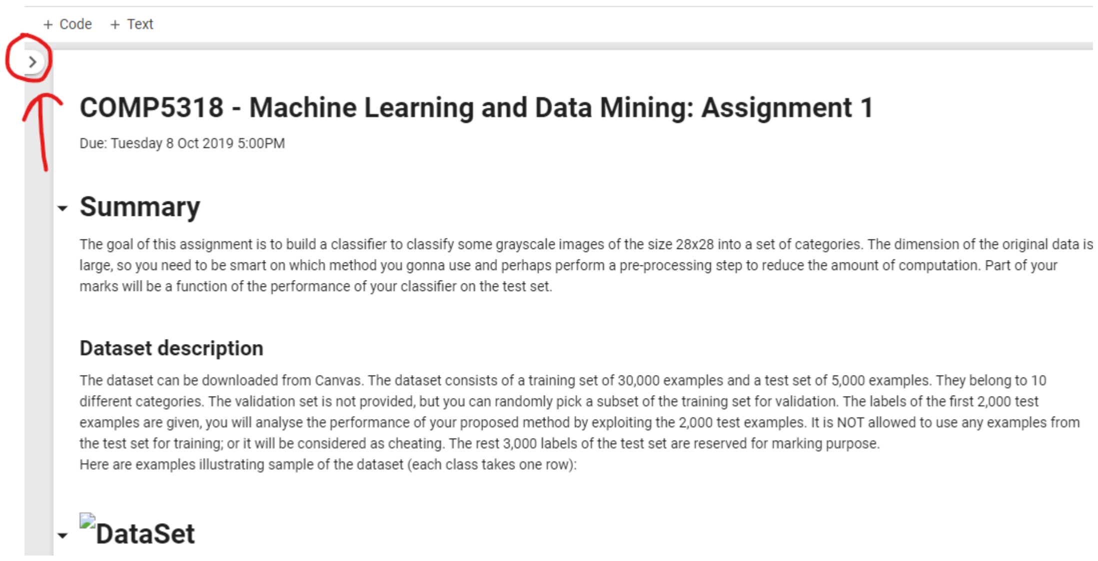
Given below are the instructions for running the attached code file for the project. Please follow the same for successful implementation of the classification task.

KEY NOTES:

- **Our most recommended classifier is KNN (default value of K is 4) with PCA. For choosing the same, please follow all code instructions and finally choose option 3 at user input prompt.**
- **Best accuracy is for KNN with least computing time.**
- **Accuracy of NN, KNN as well as LR classifiers are comparable. Our recommendation of KNN is based on less computation time.**
- **You may test any algorithm by choosing required user input at the prompt.**
- **The output file is created at the end and contains 10,000 predicted labels for the entire given test set (assignment instructions said it's 5000, however, we found to be 10,000 after running code and thus, output file has 10,000 predicted labels.)**

PLEASE FOLLOW THESE INSTRUCTIONS FOR RUNNING THE CODE:

1. Open the .ipynb notebook in Google Colab.
2. Upload the given files using the steps below
    - Click on side arrow



COMP5318 - Machine Learning and Data Mining: Assignment 1

Due: Tuesday 8 Oct 2019 5:00PM

Summary

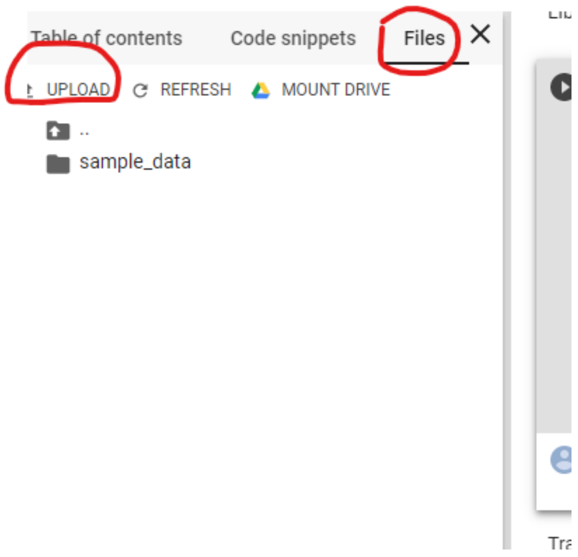
The goal of this assignment is to build a classifier to classify some grayscale images of the size 28x28 into a set of categories. The dimension of the original data is large, so you need to be smart on which method you gonna use and perhaps perform a pre-processing step to reduce the amount of computation. Part of your marks will be a function of the performance of your classifier on the test set.

Dataset description

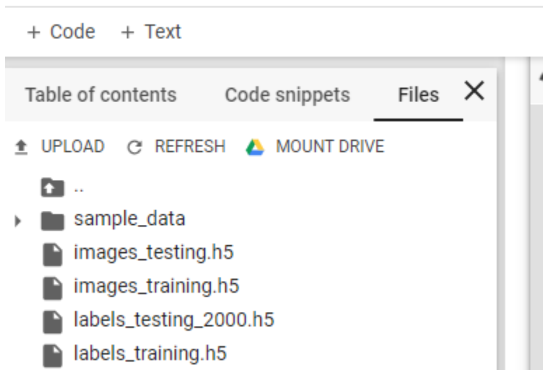
The dataset can be downloaded from Canvas. The dataset consists of a training set of 30,000 examples and a test set of 5,000 examples. They belong to 10 different categories. The validation set is not provided, but you can randomly pick a subset of the training set for validation. The labels of the first 2,000 test examples are given, you will analyse the performance of your proposed method by exploiting the 2,000 test examples. It is NOT allowed to use any examples from the test set for training; or it will be considered as cheating. The rest 3,000 labels of the test set are reserved for marking purpose.
Here are examples illustrating sample of the dataset (each class takes one row):

DataSet

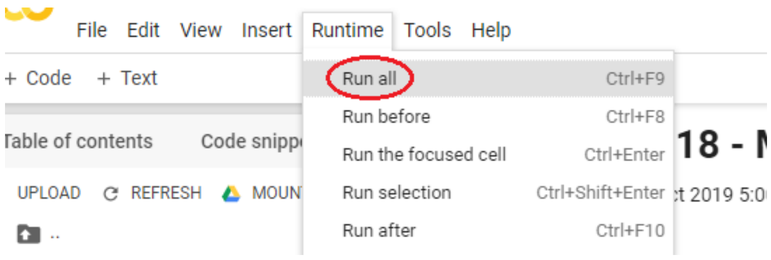
- On the side bar, click on Files option



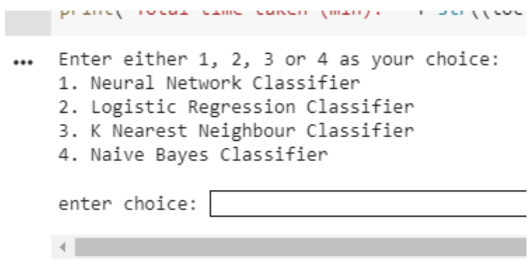
- Select the files to be uploaded and press ok.



3. Now from the menu above, click on "Runtime" and then click "Run All". Alternatively, you can press Ctrl+F9 (or Ctrl+Fn+F9) on your keyboard. **Please Note, it is necessary to run all snippets of code on each try to ensure no error is encountered.**



4. The program will ask for a user input. Enter required input (see key notes above)



```
Enter either 1, 2, 3 or 4 as your choice:
1. Neural Network Classifier
2. Logistic Regression Classifier
3. K Nearest Neighbour Classifier
4. Naive Bayes Classifier

enter choice:
```

5. Wait for a few minutes (can take up to 2-10 minutes depending upon your choice of classifier) and an output file "predicted label.h5" will be generated. Please export as required.