Machine Learning (CS60050) - Assignment 3

Rajat Bachhawat (19CS10073)
Kaushal Banthia (19CS10039)

REPORT

- The code is available on Google Colab using this [link]
- The code is also added in the zip file as a .ipynb file.
- We begin with analysing the dataset

0. Analysing the Dataset

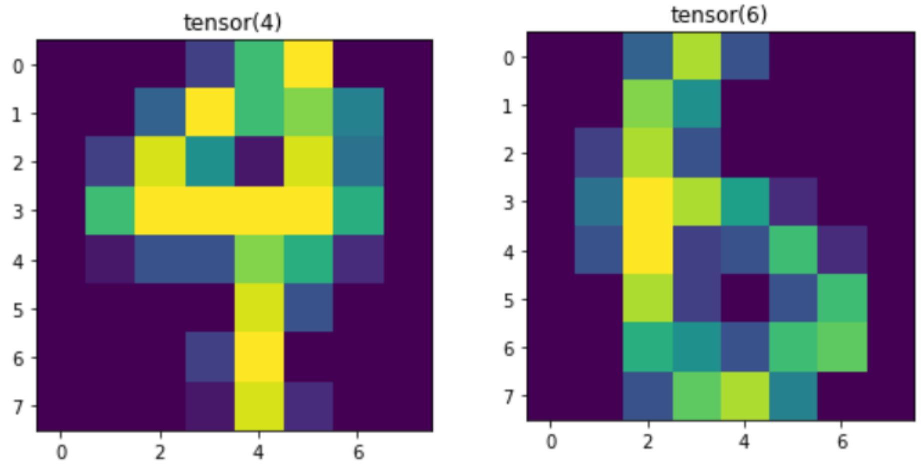
0.1. About the Dataset

- The original form of this dataset was as a 32x32 bitmap (or consider it like an image) of handwritten digits.
- The data is already divided into a training and a testing set. From a total of 43 people, 30 contributed to the training set and different 13 to the test set.
- 32x32 bitmaps were divided into non-overlapping blocks of 4x4 and the number of “on pixels” were counted in each block. This generated an input matrix of 8x8 where each element is an integer in the range 0..16 representing the number of “on pixels” in the mapped 4x4 block. This reduces dimensionality and gives invariance to small distortions.
- All input attributes are integers in the range 0..16. The last attribute is the class code 0..9 which represents the digit that the sample represents.
- There are no missing attribute values in this dataset.

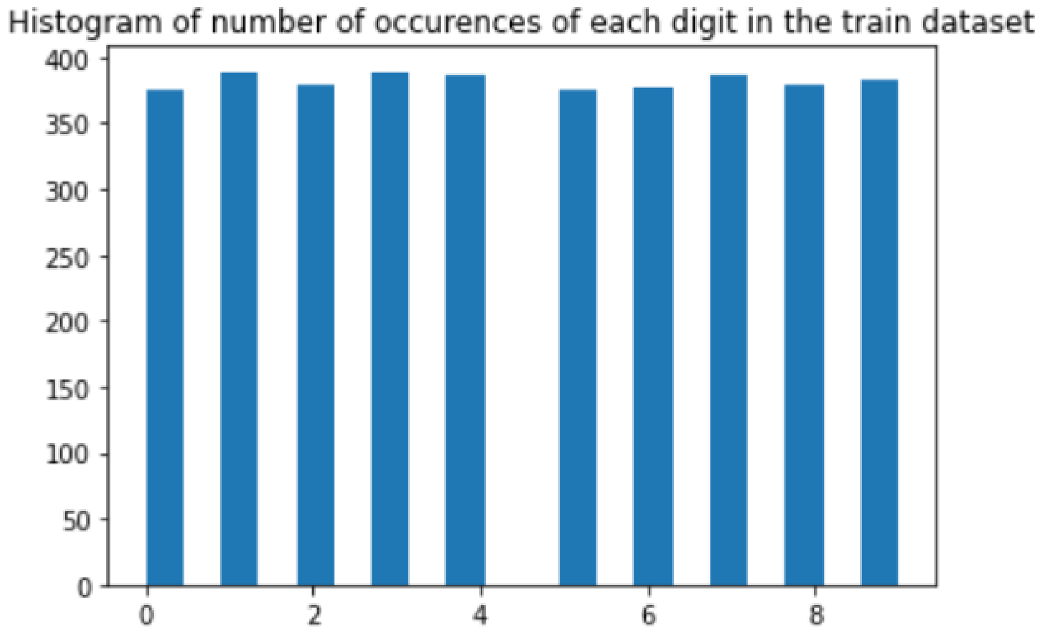
- We normalised the feature values to lie between 0 and 1. This is done so that the data points have values in the same range. This helps in training, as it helps in converging faster.

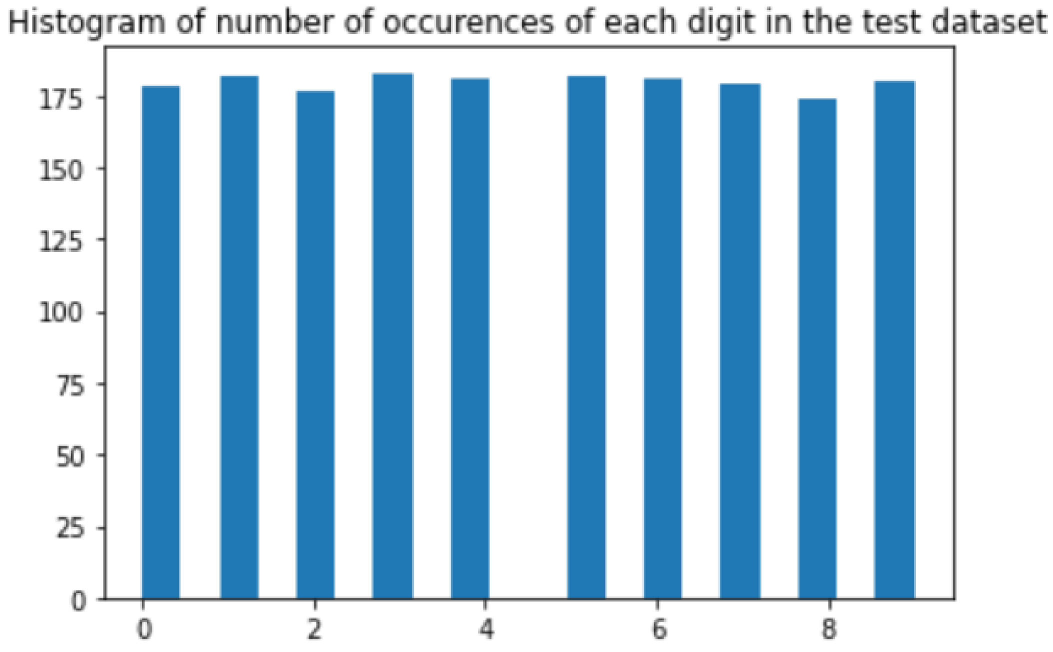
0.2. Visualizations of the Dataset

- We couldn't do any analysis using heatmaps, pairplots etc. because this time, the dataset had columns whose correlation with each other was kind of irrelevant.
- We also did not use 2 columns at a time to plot the variations in the digits using a scatterplot, because there are a total of 64 columns and the total number of graphs would have been 64C2, which would have made the analysis very cumbersome.
- We plot the images to see how they look. Here, 2 digits have been plotted



- Histogram of the number of occurrences of each digit (class)





1. Hyperparameters

Number of Nodes in Input Layer = 8*8 = 64
Justification: In our dataset, there are 64 features which each represent a pixel in a 8x8 image of a handwritten digit.

Number of Nodes in Output Layer = 10

Justification: In our dataset, there are 10 possible values for the class attribute ranging from 0 to 9, each representing a digit. So we wish to classify each sample as 0 or 1 or … or 9.

Other Hyperparameters:
- **Learning Rates** = [10, 1, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001]
- **Justification:** We are using all the learning rates that were asked to be used and additionally using 10, 1 and 0.5 to explore potentially better results.
- **Number of Iterations** = 50,000
- **Number of Epochs** = Number of Iterations / (Number of Samples / Batch Size)

2. MLP Classifier using various Architectures

We implemented an MLP Classifier using pytorch packages. We designed the following three classes all of which inherited from the base class torch.nn.Module.
- ANN_0_layers : For neural network with 0 hidden layers
- ANN_1_layers : For neural network with 1 hidden layer
- ANN_2_layers : For neural network with 2 hidden layers

Each of these classes has two functions:
- init() : For initializing the the NN using the input, output and hidden layer dimensions
- forward() : For doing the forward propagation

Using these classes we have implemented the following 5 architectures for MLP Classifiers:

2.1. 0 Hidden Layers (Model A)

Accuracies on learning rates [10, 1, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001] =

```
[94.37952142459655, 95.54813578185865, 95.43683917640512,
95.10294936004452, 93.93433500278242, 91.15191986644408,
86.19922092376183, 17.139677239844186]
```
respectively

2.2. 1 Hidden Layer with 2 Nodes (Model B)

Accuracies on learning rates [10, 1, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001] =

```
[9.961046188091263, 9.961046188091263, 64.6076794657763,
72.00890372843628, 66.83361157484697, 29.827490261547023,
23.20534223706177, 9.961046188091263]
```
respectively

2.3. 1 Hidden Layer with 6 Nodes (Model C)

Accuracies on learning rates [10, 1, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001] =

```
[9.961046188091263, 79.79966611018364, 92.32053422370618,
93.21090706733445, 93.09961046188091, 83.30550918196995,
40.40066777963272, 14.746800222593212]
```
respectively

2.4. 2 Hidden Layers with 2 and 3 Nodes respectively (Model D)

Accuracies on learning rates [10, 1, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001] =

```
[9.961046188091263, 9.961046188091263, 66.7779632721202, 86.0879243183083,
84.75236505286588, 56.14913745130774, 22.982749026154703,
17.250973845297718]
```
respectively

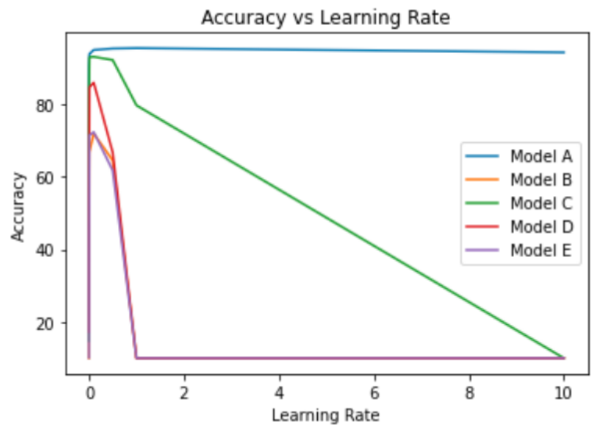
2.5. 2 Hidden Layers with 3 and 2 Nodes respectively (Model E)

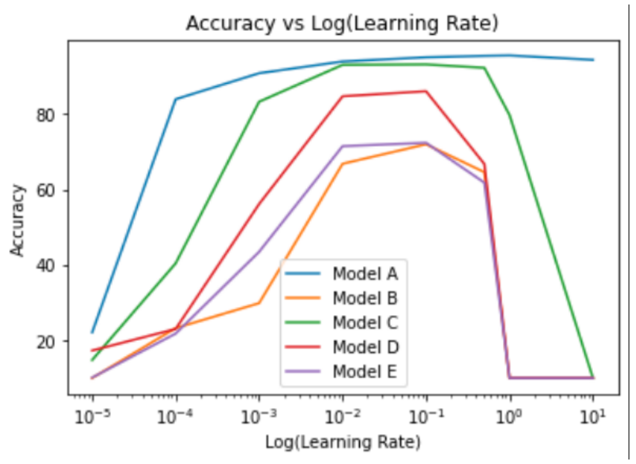
Accuracies on learning rates [10, 1, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001] =

```
[9.961046188091263, 9.961046188091263, 61.82526432943795,
72.39844184752366, 71.50806900389539, 43.4613244296049,
21.758486366165833, 10.072342793544797]
```
respectively

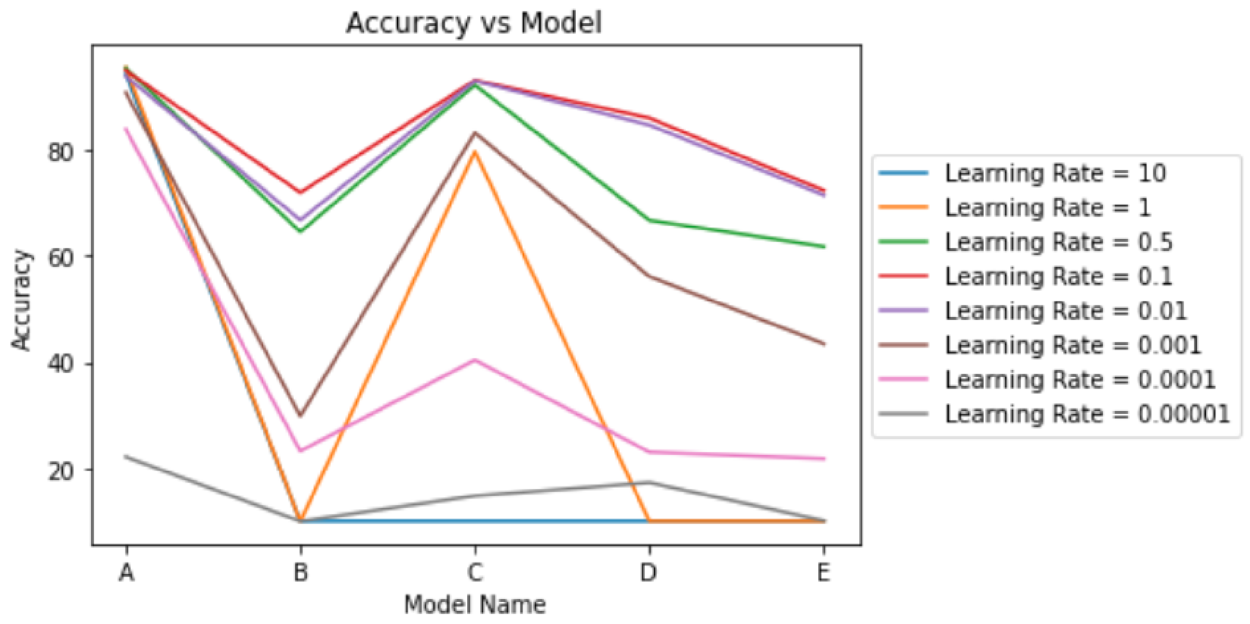
3. Graphical Comparisons of Models and Learning Rates

Observed Plot of Accuracy vs Learning Rate for all 5 Architectures





Observed Plot of Accuracy vs Model for all Learning Rates



Inference

From the above graphs, we can clearly see that model A is the best model. As we increase the complexity of the model, by adding more hidden layers and nodes, the accuracy drops. This can be attributed to the fact that the dataset is very small and thus a complex neural network model is not required to learn it. A complex model would only overfit to the dataset and thus cause a decrease in the accuracy on the test dataset.

Also, we can infer from the graphs, that the best learning rate for the models is 0.1
Even though it is not the best for model A, it is chosen, because it is the best for all the other models and performs very well on model A too. The higher learning rates are not preferred

because they reduce the accuracy of the models. This is due to the fact that a higher learning rate tends to result in weight updates that will be too large and thus, the performance of the model (such as its loss on the training dataset) will oscillate over training epochs. This is not desirable.

Also if the learning rate is too small, then it would take forever to train the model, and thus we would not get the desired accuracy, as the training wouldn't converge (as it evident with learning rate = 0.00001)

4. Best Model

From the plots above we find that the best model to use is the neural network with:

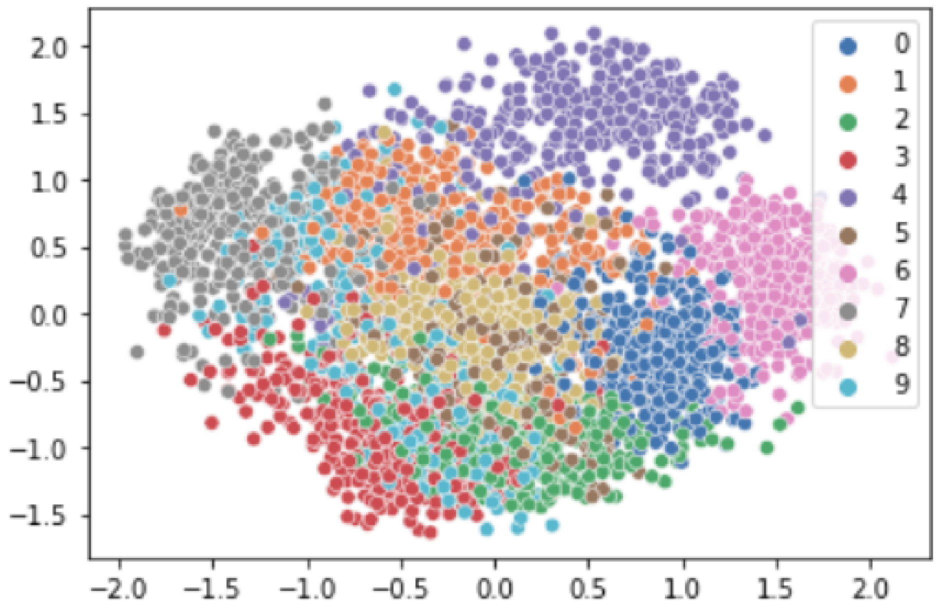
- **0 Hidden Layers**
- **Learning Rate = 0.1**

The accuracy for this architecture is `95.10294936004452%.`
Justification: As discussed before, the best model is model A, which gives us an accuracy of >95%. Some other models also perform considerably well, but the model with the least complexity performs the best here.

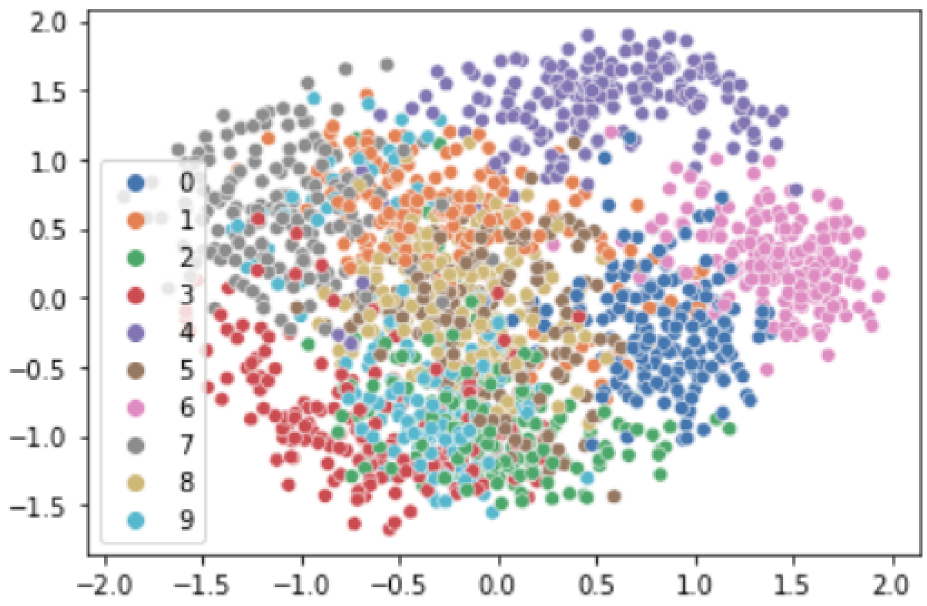
5. Dimensionality Reduction using PCA

We transformed the 64-feature dataset to a 2-feature dataset using Principal Component Analysis. We used the pca module from the sklearn.decomposition module for implementing PCA with the number of components = 2.

Scatter Plot of the Dimensionally Reduced Training Dataset in a 2D plane



Scatter Plot of the Dimensionally Reduced Testing Dataset in a 2D plane



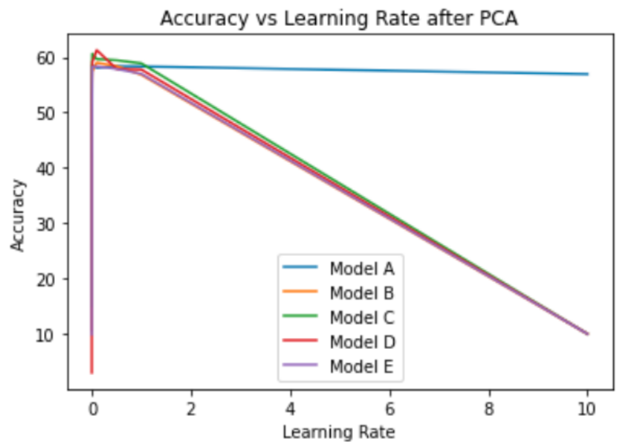
Inference

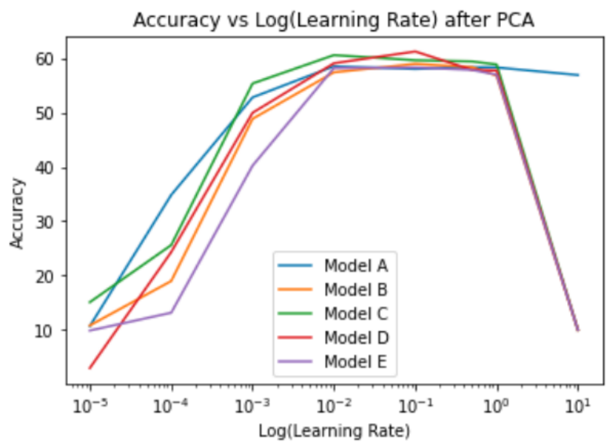
We observe that the training data is fairly clustered but the labels 2,5 and 9 which are vastly scattered on reducing to 2 dimensions. Thus, we don't expect very high accuracy on reducing dimensions. Now the number of nodes in the input layer is 2 (because the number of features in the dataset has been reduced to 2) and the number of nodes in the output layer remain the same.

6. Implementing MLP on **reduced feature space**

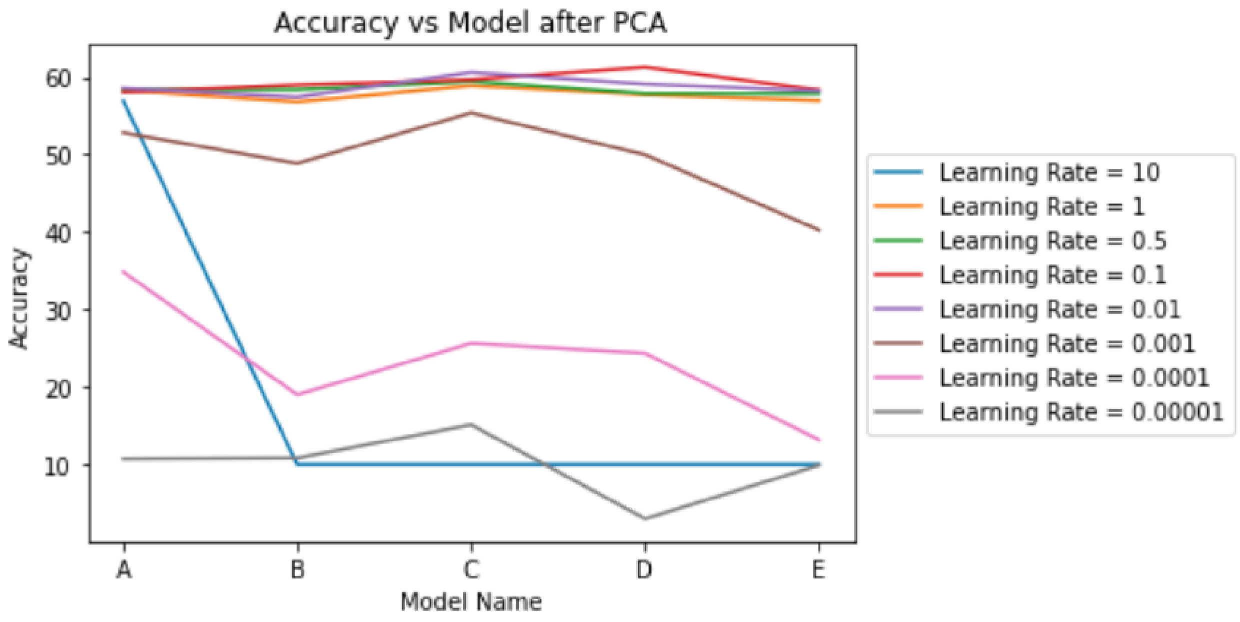
6.1. MLP Results using the 5 architectures

Observed Plot of Accuracy vs Learning Rate for all 5 Architectures





Observed Plot of Accuracy vs Model for all Learning Rates



Inference

We can see that the accuracy has taken a significant hit after applying PCA. This is because of the fact that we now have a reduced dataset with a very less number of features. This causes insufficient information to the model and thus the model cannot train adequately, giving us subpar accuracies.

From the above graphs, it is quite evident that all the models perform more or less similarly. This happens due to the loss of information caused by performing PCA on the dataset. This makes the architecture of the model irrelevant.

6.2. Best Learning Rate from Step 3

Best learning rate from step 3 was 0.1 as can be clearly seen from the plot. All the models achieved fairly high accuracies on 0.1. Thus, we use this learning rate to compare the classification reports before and after applying PCA for each of the 5 architectures.

6.3. Comparison of Classification Output

6.3.1. 0 Hidden Layers (Model A)

```
Before PCA
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       178
           1       0.91      0.96      0.93       182
           2       0.97      0.97      0.97       177
           3       0.99      0.91      0.95       183
           4       0.96      0.97      0.96       181
           5       0.92      0.97      0.94       182
           6       0.99      0.98      0.98       181
           7       0.98      0.91      0.94       179
           8       0.92      0.90      0.91       174
           9       0.89      0.94      0.91       180

    accuracy                           0.95      1797
   macro avg       0.95      0.95      0.95      1797
weighted avg       0.95      0.95      0.95      1797

After PCA
              precision    recall  f1-score   support

           0       0.71      0.86      0.78       178
           1       0.47      0.70      0.56       182
           2       0.45      0.56      0.50       177
           3       0.58      0.67      0.62       183
           4       0.85      0.90      0.88       181
           5       0.23      0.10      0.14       182
           6       0.90      0.87      0.88       181
           7       0.81      0.67      0.73       179
           8       0.41      0.29      0.34       174
           9       0.20      0.18      0.19       180

    accuracy                           0.58      1797
   macro avg       0.56      0.58      0.56      1797
weighted avg       0.56      0.58      0.56      1797
```

6.3.2. 1 Hidden Layer with 2 Nodes (Model B)

```
Before PCA
              precision    recall  f1-score   support
```

```
           0       0.75      0.79      0.77       178
           1       0.47      0.47      0.47       182
           2       0.86      0.85      0.85       177
           3       0.88      0.80      0.84       183
           4       0.89      0.94      0.92       181
           5       0.60      0.50      0.54       182
           6       0.86      0.84      0.85       181
           7       0.96      0.74      0.84       179
           8       0.59      0.58      0.59       174
           9       0.50      0.73      0.60       180

    accuracy                           0.72      1797
   macro avg       0.74      0.72      0.73      1797
weighted avg       0.74      0.72      0.73      1797

After PCA
              precision    recall  f1-score   support

           0       0.72      0.81      0.77       178
           1       0.45      0.69      0.54       182
           2       0.44      0.59      0.51       177
           3       0.58      0.69      0.63       183
           4       0.86      0.91      0.88       181
           5       0.25      0.08      0.12       182
           6       0.87      0.90      0.88       181
           7       0.80      0.71      0.75       179
           8       0.41      0.29      0.34       174
           9       0.17      0.14      0.15       180

    accuracy                           0.58      1797
   macro avg       0.55      0.58      0.56      1797
weighted avg       0.56      0.58      0.56      1797
```

6.3.3. 1 Hidden Layer with 6 Nodes (Model C)

```
Before PCA
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       178
           1       0.88      0.93      0.90       182
           2       0.92      0.94      0.93       177
           3       0.95      0.91      0.93       183
           4       0.95      0.98      0.96       181
           5       0.92      0.95      0.93       182
           6       0.94      0.94      0.94       181
           7       0.99      0.85      0.92       179
           8       0.88      0.82      0.85       174
           9       0.82      0.92      0.87       180

    accuracy                           0.92      1797
   macro avg       0.92      0.92      0.92      1797
weighted avg       0.92      0.92      0.92      1797

After PCA
```

```
              precision    recall  f1-score   support

           0       0.74      0.81      0.78       178
           1       0.48      0.71      0.58       182
           2       0.45      0.51      0.48       177
           3       0.56      0.66      0.61       183
           4       0.90      0.93      0.92       181
           5       0.24      0.12      0.16       182
           6       0.91      0.90      0.91       181
           7       0.75      0.76      0.75       179
           8       0.40      0.45      0.43       174
           9       0.33      0.16      0.21       180

    accuracy                           0.60      1797
   macro avg       0.58      0.60      0.58      1797
weighted avg       0.58      0.60      0.58      1797
```

6.3.4. 2 Hidden Layers with 2 and 3 Nodes respectively (Model D)

```
Before PCA
              precision    recall  f1-score   support

           0       0.91      0.84      0.88       178
           1       0.60      0.54      0.57       182
           2       0.80      0.80      0.80       177
           3       0.80      0.82      0.81       183
           4       0.78      0.82      0.80       181
           5       0.65      0.54      0.59       182
           6       0.82      0.86      0.84       181
           7       0.88      0.70      0.78       179
           8       0.51      0.57      0.54       174
           9       0.53      0.69      0.60       180

    accuracy                           0.72      1797
   macro avg       0.73      0.72      0.72      1797
weighted avg       0.73      0.72      0.72      1797

After PCA
              precision    recall  f1-score   support

           0       0.73      0.79      0.76       178
           1       0.46      0.71      0.56       182
           2       0.44      0.61      0.51       177
           3       0.55      0.68      0.61       183
           4       0.89      0.92      0.90       181
           5       0.28      0.10      0.15       182
           6       0.87      0.92      0.90       181
           7       0.79      0.72      0.75       179
           8       0.40      0.45      0.42       174
           9       0.25      0.07      0.11       180

    accuracy                           0.60      1797
   macro avg       0.57      0.60      0.57      1797
weighted avg       0.57      0.60      0.57      1797
```

6.3.5. 2 Hidden Layers with 3 and 2 Nodes respectively (Model E)

```
Before PCA
              precision    recall  f1-score   support

           0       0.82      0.77      0.79       178
           1       0.64      0.57      0.60       182
           2       0.82      0.84      0.83       177
           3       0.90      0.79      0.84       183
           4       0.82      0.91      0.86       181
           5       0.58      0.58      0.58       182
           6       0.83      0.88      0.86       181
           7       0.91      0.75      0.83       179
           8       0.52      0.57      0.55       174
           9       0.52      0.61      0.56       180

    accuracy                           0.73      1797
   macro avg       0.74      0.73      0.73      1797
weighted avg       0.74      0.73      0.73      1797

After PCA
              precision    recall  f1-score   support

           0       0.71      0.84      0.77       178
           1       0.46      0.66      0.55       182
           2       0.44      0.59      0.50       177
           3       0.61      0.63      0.62       183
           4       0.87      0.94      0.90       181
           5       0.26      0.10      0.14       182
           6       0.91      0.91      0.91       181
           7       0.78      0.73      0.75       179
           8       0.43      0.30      0.36       174
           9       0.21      0.19      0.20       180

    accuracy                           0.59      1797
   macro avg       0.57      0.59      0.57      1797
weighted avg       0.57      0.59      0.57      1797
```

Inference

From the above classification reports, we can observe that the dataset before PCA yielded much better accuracies than the one after PCA. This is due to the presence of more information initially, which helped in finding correlations in the dataset. When the dataset was subjected to PCA, then a lot of the information was lost, which caused difficulties in identifying patterns. Thus, the new dataset yielded bad results.