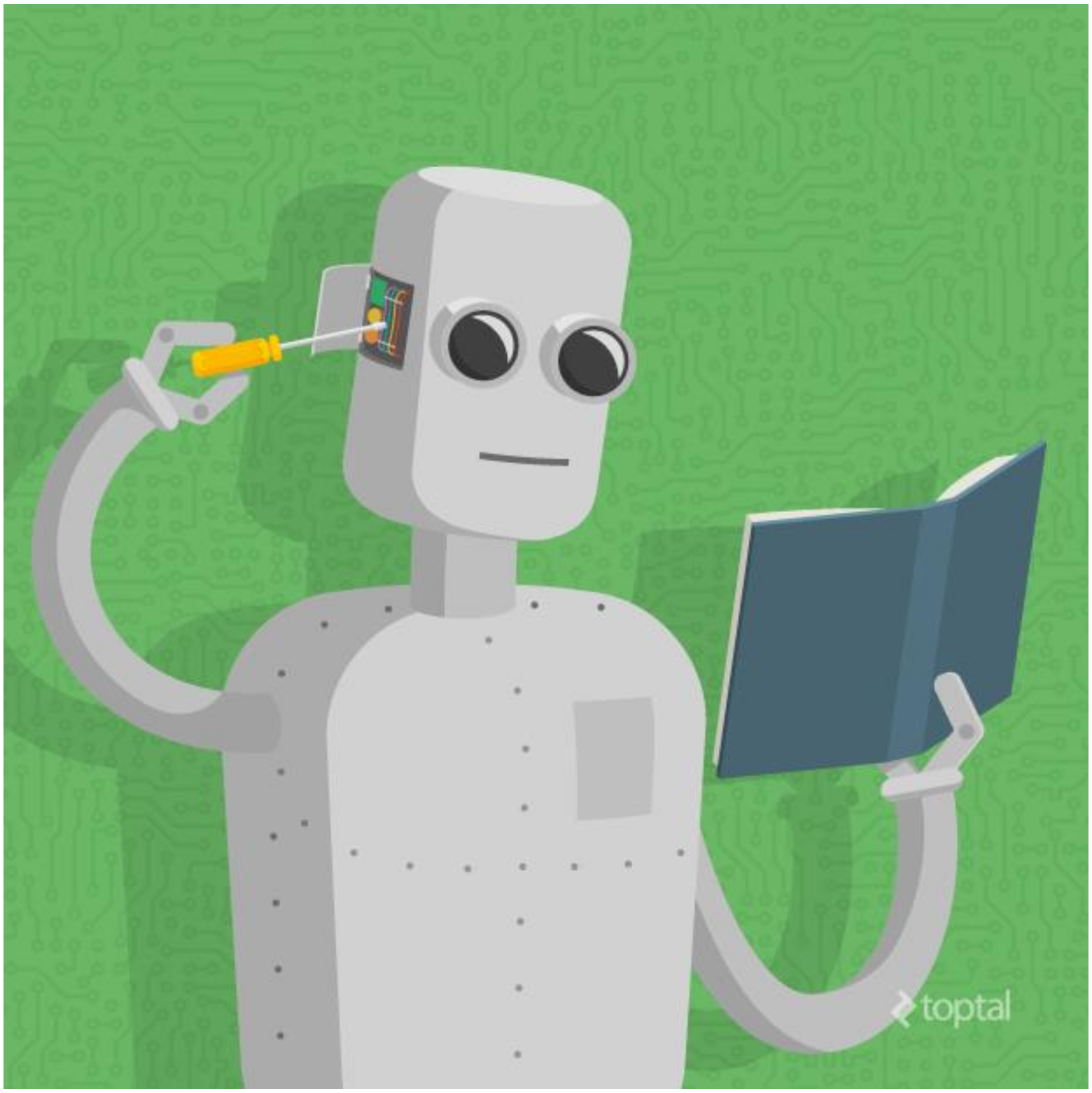


PROJECT -1 REPORT

ABSTRACT

Apply various classifiers on the MNIST dataset and observe the behavior of each classifier.

Jigar Agrawal
(011469796)
EE258

Table of Contents

Table of Figure

Abstract

There are plenty of different classifiers available in the market. But it is very important to have an intuition about which one of them to be used on which data. For that we have to learn the behavior of the data first and then we can decide which classifier is best suited for that application to get the more accuracy from the model.

Description Of data

For this project, we are going to use the MNIST dataset. MNIST data set is one of the most popular datasets. It is often considered to be the hello world of the machine learning. Let's have a look at the data set first.

The data set contains 70000 of total handwritten digit images. Each of them are collected from very random places and they are very different from each other (even the same number digit). Each image is 28 x 28-pixel image. So, there will be total of 784 features in the dataset. The pixels are serialized row wise in the data set given. Each digit image looks like the one shown in figure 1.

Dataset is already divided into the training set and test set. Training set contains 60000 images and the test set contains 10000 images. Training set is well shuffled already so that the model wont overfit to the data.

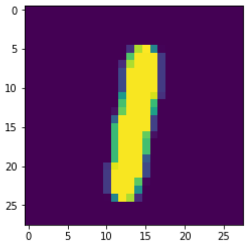


Figure 1 Digit image from training dataset

Methodologies Used

Cross Validation

Cross validation technique is necessary to avoid the overfitting to the data while training the model. It assures the good generalization. The cross-validation method used in this project is k-fold cross validation. In k-fold technique training data is divided into k splits. Then one of them is used to validate the data and the rest are used to estimate the model. Then same is performed for every split and the error is monitored through the validation set. The final error is the average of all the errors gotten by k-estimations.

There is one drawback of the k-fold method that it requires more computing time as it performs the model estimation k-times. So, they are slow sometimes.

Sklearn provides cross_val_score function which can be used for this purpose.

```
# Cross Validation for predicting training error
#No of K = 3;
score = cross_val_score(ML_model, Train_X, Train_y, cv= 3)
print('Traning Accuracy is = ' + str(score.mean() * 100) + '%')
```

Early Stopping

When we start training the model both training and the test errors start decreasing up to certain point. If we continue to train our model it will start over fitting to the training data, so the training error continues to decrease but the generalization error starts increasing due to overfitting as shown in the following figure. So, to prevent this we should stop training the model after this point called the early stopping point.

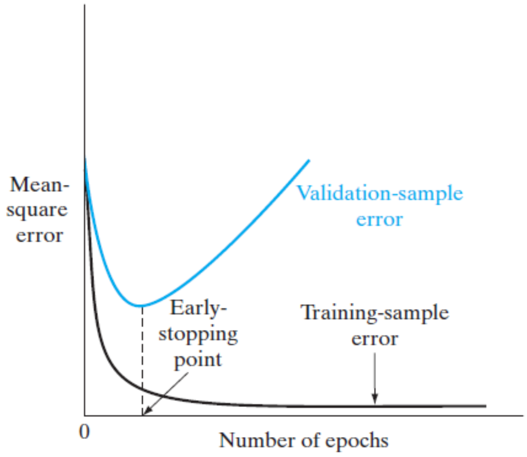


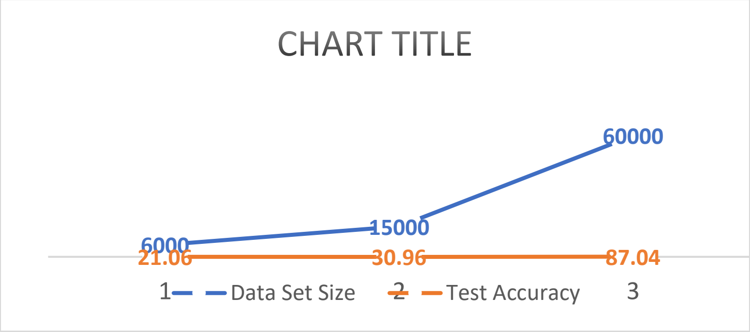
Figure 2 Early Stopping Point

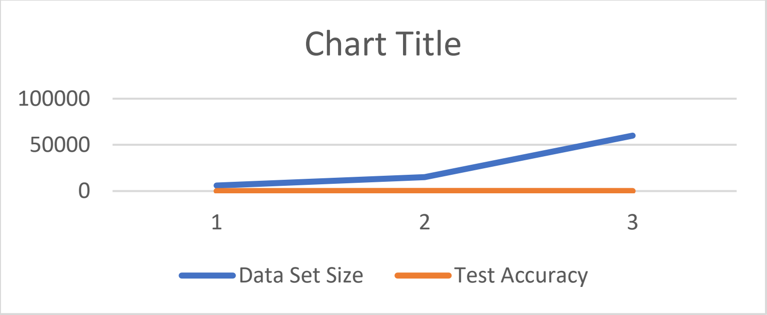
Classifiers

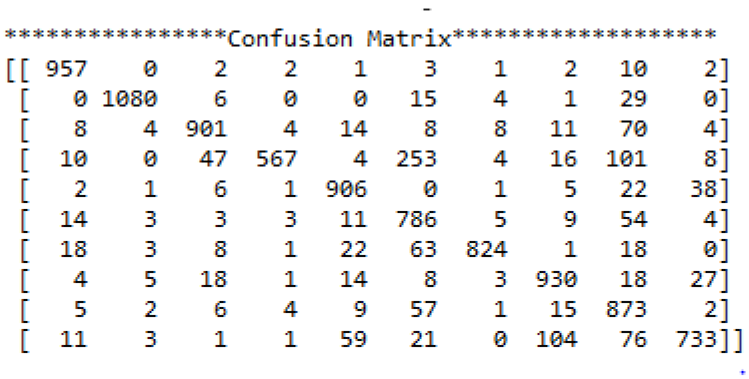
Perceptron

Perceptron is the first simple most classifier. Perceptron is a linear classifier. It is a binary classifier. It is an iterative method. Perceptron needs data to be linearly separable to be classified. Depending on the data separation we can get many possible models classifying the given data. In real world, the problems are more complicated than perceptron can handle, so it is rarely used.

Data Set Size	**Training Accuracy**	**Test Accuracy**
6000	99.933	21.06
15000	98	30.96
60000	87.99	87.04





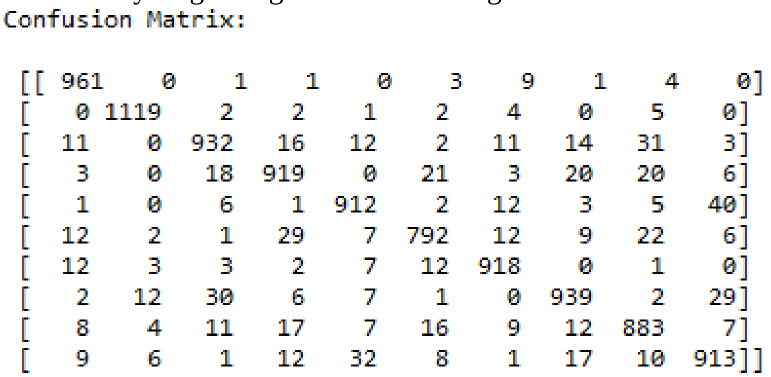


```
****************Confusion Matrix*******************
[[ 957    0    2    2    1    3    1    2   10    2]
 [   0 1080    6    0    0   15    4    1   29    0]
 [   8    4  901    4   14    8    8   11   70    4]
 [  10    0   47  567    4  253    4   16  101    8]
 [   2    1    6    1  906    0    1    5   22   38]
 [  14    3    3    3   11  786    5    9   54    4]
 [  18    3    8    1   22   63  824    1   18    0]
 [   4    5   18    1   14    8    3  930   18   27]
 [   5    2    6    4    9   57    1   15  873    2]
 [  11    3    1    1   59   21    0  104   76  733]]
```

Figure 3 Perceptron Confusion Matrix

K Nearest Neighbors

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). K nearest neighbor search for the k similar data point near to it in a space according to a distance. It is basically a clustering algorithm. That also means that the algorithm it is unsupervised learning. So, we must pay a cost of accuracy. It takes a lot time to train the model because it searches every single neighbor for clustering.



```
Confusion Matrix:

[[ 961    0    1    1    0    3    9    1    4    0]
 [   0 1119    2    2    1    2    4    0    5    0]
 [  11    0  932   16   12    2   11   14   31    3]
 [   3    0   18  919    0   21    3   20   20    6]
 [   1    0    6    1  912    2   12    3    5   40]
 [  12    2    1   29    7  792   12    9   22    6]
 [  12    3    3    2    7   12  918    0    1    0]
 [   2   12   30    6    7    1    0  939    2   29]
 [   8    4   11   17    7   16    9   12  883    7]
 [   9    6    1   12   32    8    1   17   10  913]]
```

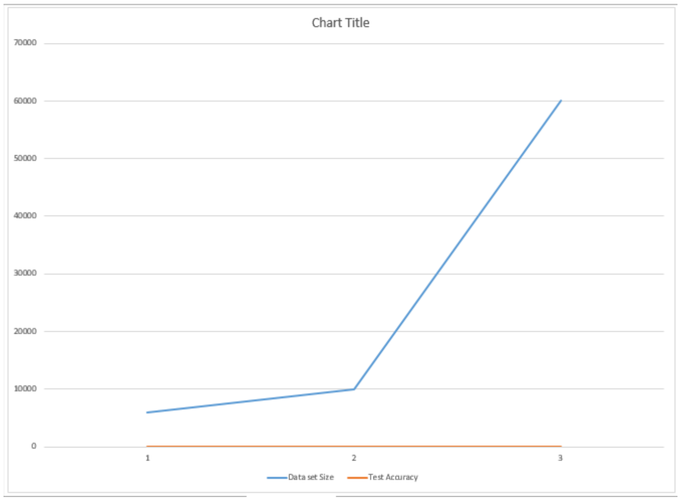


Figure 4Test vs epochs

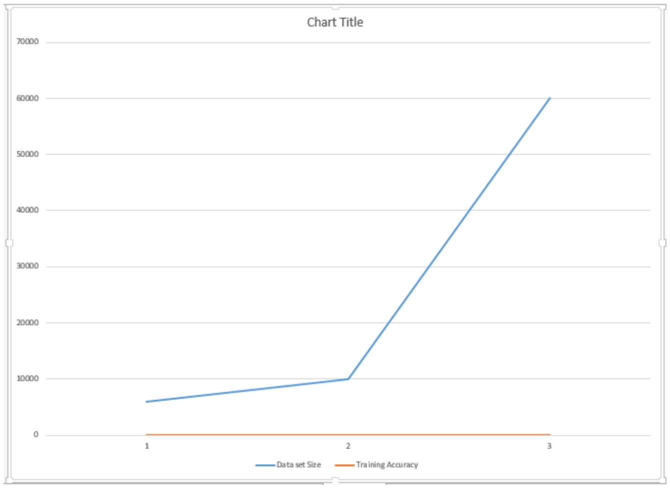
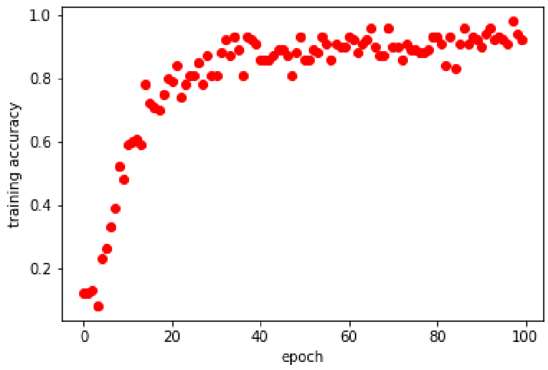
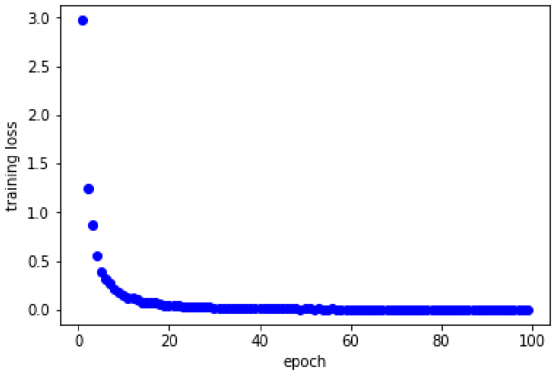


Figure 5 Training accuracy vs epochs

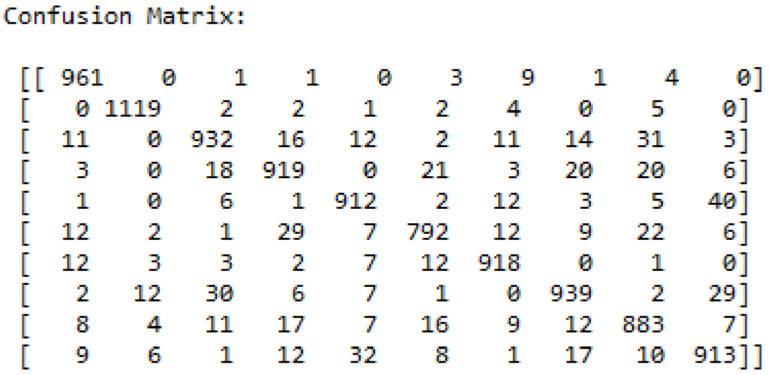
Radial Basis Function (RBF) neural network

Radial basis function neural network classifier is the unsupervised classifier. It uses clustering technique to classify the data. As the name indicates the radial basis function means that the classifier uses radial basis function instead of the linear combiner as in the case of perceptron. It creates centers in the given data set and then clusters the nearby points which may highly likely to belong to the same class. As it is the unsupervised method we must pay the price of accuracy using this classifier.





Confusion Matrix:

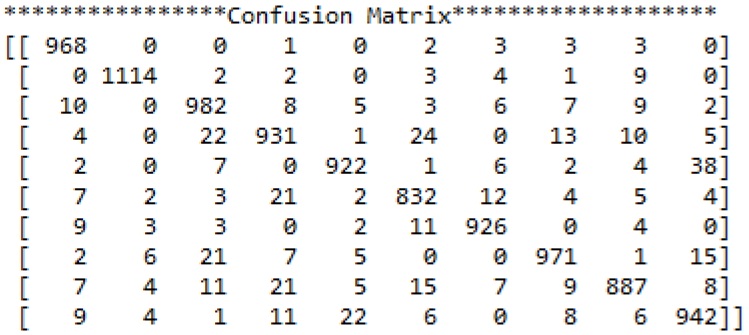


```
Confusion Matrix:

[[ 961    0    1    1    0    3    9    1    4    0]
 [   0 1119    2    2    1    2    4    0    5    0]
 [  11    0  932   16   12    2   11   14   31    3]
 [   3    0   18  919    0   21    3   20   20    6]
 [   1    0    6    1  912    2   12    3    5   40]
 [  12    2    1   29    7  792   12    9   22    6]
 [  12    3    3    2    7   12  918    0    1    0]
 [   2   12   30    6    7    1    0  939    2   29]
 [   8    4   11   17    7   16    9   12  883    7]
 [   9    6    1   12   32    8    1   17   10  913]]
```

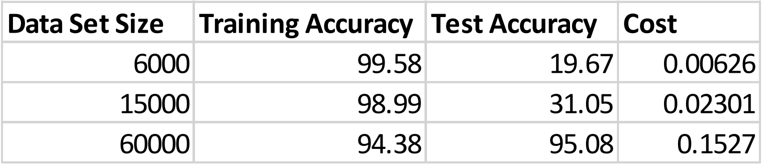
Multilayer Perceptron

Multilayer perceptron is the classifier based the linear perceptron. It is extension of the basic linear perceptron. In MLP there are more than one layer. The extra layers are called hidden layers. Adding hidden layer make the classifier more magical in the sense that hidden neurons can perform variety of work and pass their information to the next layer. This make Neural Network more accurate than other classifiers. It performs well on the nonlinear data as well.



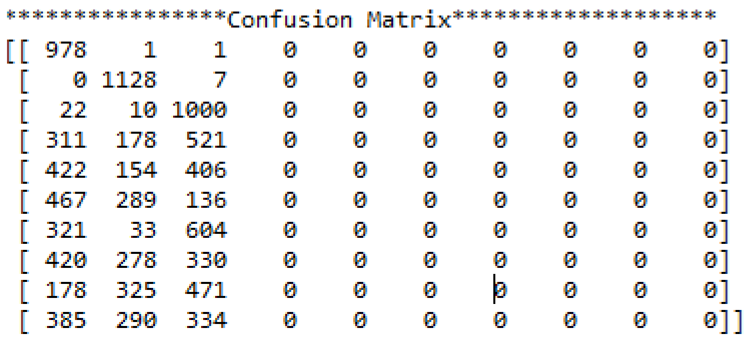
```
****************Confusion Matrix*******************
[[ 968    0    0    1    0    2    3    3    3    0]
 [   0 1114    2    2    0    3    4    1    9    0]
 [  10    0  982    8    5    3    6    7    9    2]
 [   4    0   22  931    1   24    0   13   10    5]
 [   2    0    7    0  922    1    6    2    4   38]
 [   7    2    3   21    2  832   12    4    5    4]
 [   9    3    3    0    2   11  926    0    4    0]
 [   2    6   21    7    5    0    0  971    1   15]
 [   7    4   11   21    5   15    7    9  887    8]
 [   9    4    1   11   22    6    0    8    6  942]]
```

Figure 6 One Hidden Layer MLP Confusion Matrix



Data Set Size	Training Accuracy	Test Accuracy	Cost
6000	99.58	19.67	0.00626
15000	98.99	31.05	0.02301
60000	94.38	95.08	0.1527

Figure 7MLP 1 Hidden Layer Table



```
****************Confusion Matrix******************
[[ 978    1    1    0    0    0    0    0    0    0]
 [    0 1128    7    0    0    0    0    0    0    0]
 [   22   10 1000    0    0    0    0    0    0    0]
 [ 311  178  521    0    0    0    0    0    0    0]
 [ 422  154  406    0    0    0    0    0    0    0]
 [ 467  289  136    0    0    0    0    0    0    0]
 [ 321   33  604    0    0    0    0    0    0    0]
 [ 420  278  330    0    0    0    0    0    0    0]
 [ 178  325  471    0    0    0    0    0    0    0]
 [ 385  290  334    0    0    0    0    0    0    0]]
```

Figure 8 MLP 2 Layer Confusion Matrix

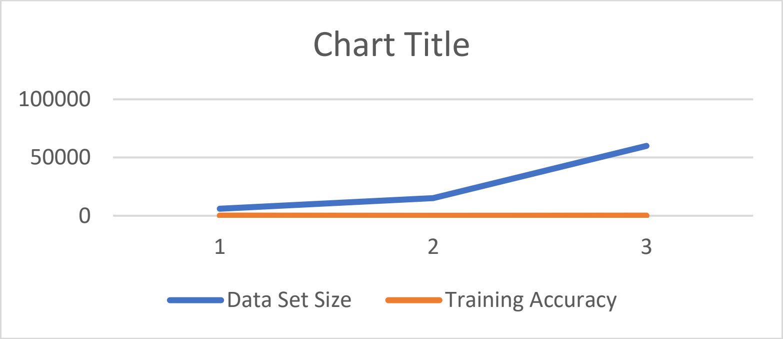


Figure 9MLP 1 Hidden Layer Training Accuracy

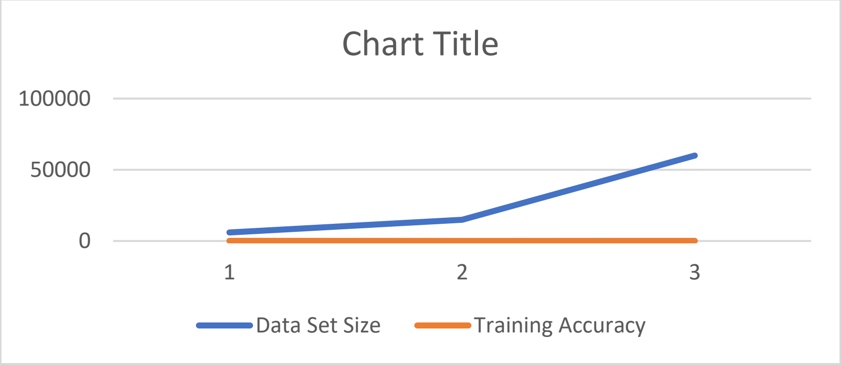


Figure 10Hidden Layer Test Accuracy

Data Set Size	Training Accuracy	Test Accuracy	Cost
6000	98.71	9.8	0.047
15000	98.6	31.06	0.0249
60000	99.93	9390.00%	0.1851

Figure 11 MLP 2 Hidden Layer Table

Stochastic gradient descent

Stochastic gradient descent is a simple yet very efficient approach to fit linear models. It is particularly useful when the number of samples (and the number of features) is very large. As the name tells it randomizes the training.

Data set Size	Training Accuracy	Test Accuracy
6000	99.91	21.06
36000	92.93	56.17
60000	86.72	87.59

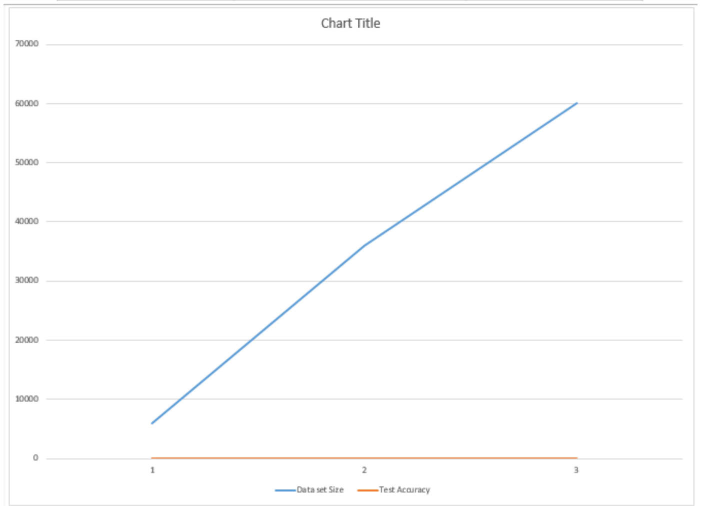


Figure 12 Test Vs Epochs

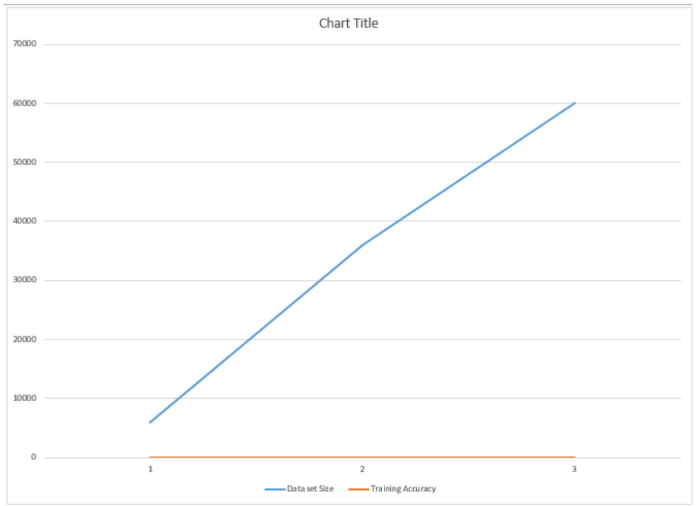


Figure 13 Training vs Epochs

Ridge

Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares.

Data set Size	Training Accuracy	Test Accuracy
6000	99.7	20.19
36000	92.78	55.71
60000	85.5	85.65

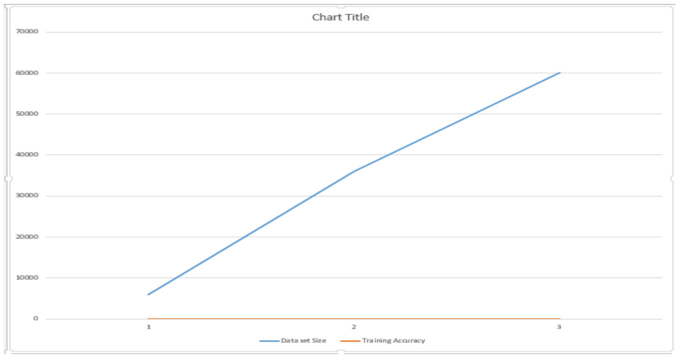


Figure 14Training Accuracy Vs Epochs

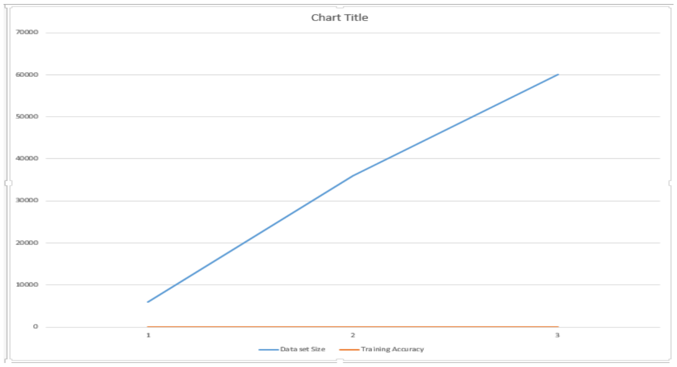


Figure 15Test Accuracy Vs Epochs

Random Forest

It uses decision tree algorithm, to train the model. It is ensemble learning method.

Data set Size	Training Accuracy	Test Accuracy
6000	99.88	20.47
36000	96.65	58.54
60000	93.89	94.47

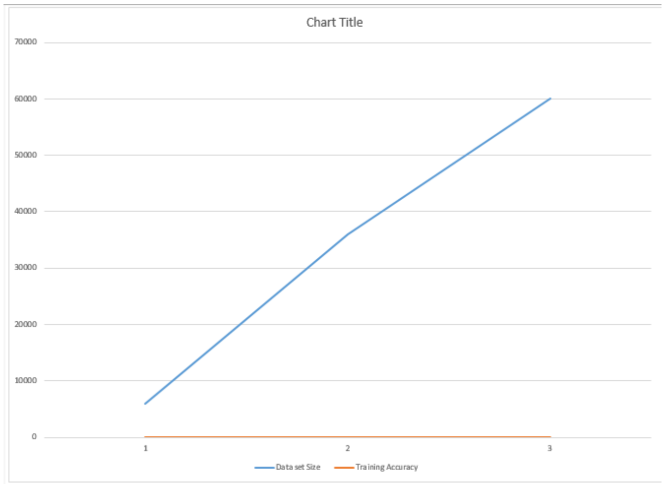


Figure 16 Training Accuracy vs epochs

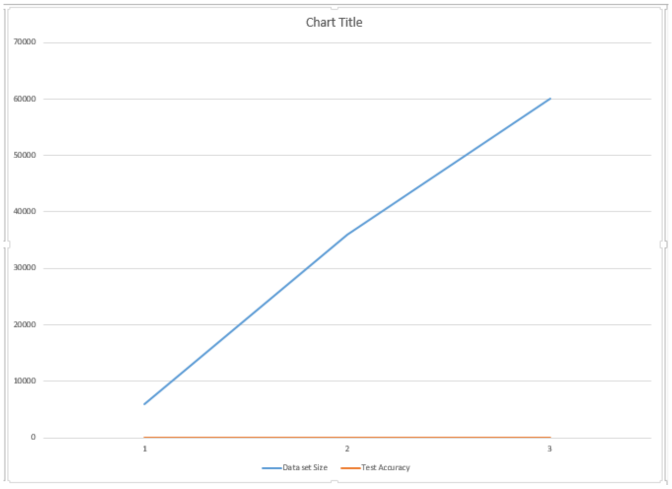


Figure 17 Test Accuracy vs epochs

Result

These project uses number of classifiers on the MNIST dataset and are compared based on performance in terms of accuracy and errors. It was observed that classifiers based on supervised learning returns returned higher accuracy as the data is provided with desired output also. Linear Classifier had an accuracy of 97% approximately with unsupervised learning classifiers returned with the accuracy of 78-80% as desired output was not provided. Every classifier were highly influenced with learning rate and the number of epochs provided.