**EE258 PROJECT**

**FALL 2017**

**CLASSIFICATION OF HANDWRITTEN DIGITS**

Methodology:

As per given in the project questions, we have used five different types of classifiers and three extra classifiers for extra credit. The classifiers are as follows:

1. Linear classifier
2. K-nearest neighbour classifier
3. Radial Basis Function (RBF) neural network
4. One-hidden layer fully connected multilayer neural network
5. Two-hidden layer fully connected multilayer neural network
6. Random forest classifier
7. Decision tree classifier
8. RBM classifier
9. Linear Classifier: Linear classifier is another word for binary classifier. Here we are identifying just one digit to find the accuracy and confusion metrics. We are using Stochastic Gradient Descent (SGD) classifier for simplicity of operation. Output is either true or false.
10. K-nearest neighbour classifier: The k-Nearest Neighbour classifier is by far the most simple image classification algorithm. In fact, it’s so simple that it doesn’t actually “learn” anything! Instead, this algorithm simply relies on the distance between feature vectors. Simply put, the k-NN algorithm classifies unknown data points by finding the most common class among the kclosest examples. Each data point in the k closest data points casts a vote, and the category with the highest number of votes wins! Or in plain english: “Tell me who your neighbours are, and I’ll tell you who you are”.
11. Radial Basis Function (RBF) neural network: Support Vector Machines (SVMs) is a group of powerful classifiers and we are using RBF as a kernel here. In the field of mathematical modelling, a radial basis function network is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Radial basis function networks have many uses, including function approximation, time series prediction, classification, and system control.
12. One-hidden layer fully connected multilayer neural network: This is a supervised type of learning algorithm and uses back-propagation for training. Each node in neural network uses a non-linear activation function. In this there is only one layer between input and output layer.
13. Two-hidden layer fully connected multilayer neural network – This is same as one-hidden layer just the difference is that now there are two layers between input and output layer.
14. Decision Trees (DTs) - are a supervised learning algorithm that can learn complex decision boundaries for handling both classiﬁcation and regression problems. The algorithm works by constructing a tree from the training data in which interior nodes correspond to one of the input features and the leaf nodes contain a prediction of the output value or category. Each interior node also contains a cutoﬀ value, and in a binary DT like in this project where classification of normal and abnormal (malicious) packets have implemented, a left and a right subtree.
15. Random Forests - is a technique for reducing DT test error due to overﬁtting. Instead of training a single DT on the entire set of features, in this project instead training an ensemble of n trees, each considering only a random subset of m of the features. In making a prediction for a test example, the n predictions from each tree are generated and then output the ﬁnal classiﬁcation as the mode of these predictions.
16. Support Vector Machine (SVM) - performs regression and classification tasks by constructing nonlinear decision boundaries. Because of the nature of the feature space in which these boundaries are found, Support Vector Machines can exhibit a large degree of flexibility in handling classification and regression tasks of varied complexities.

Data

The MNIST dataset is one of the most well studied datasets in the computer vision and machine learning literature. In many cases, it’s a benchmark and a standard to which machine learning algorithms are ranked.

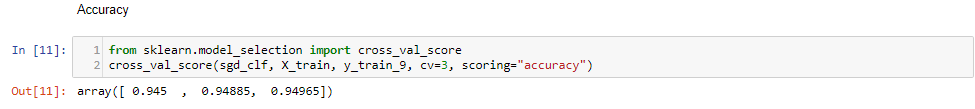
The goal of this dataset is to correctly classify the handwritten digits *0-9*. Our data consists of 60,000 training images and 10,000 testing images. We’ll be using a data provided by the scikit-learn library. Each image in the dataset from scikit-learn is represented as a 28 \* 28-dim raw pixel intensity feature vector. This means that each image is actually a 28 x 28 grayscale image, but scikit-learn “flattens” the image into a list.

All digits are placed on a black background with the foreground being shades of white and grey.

Simulations

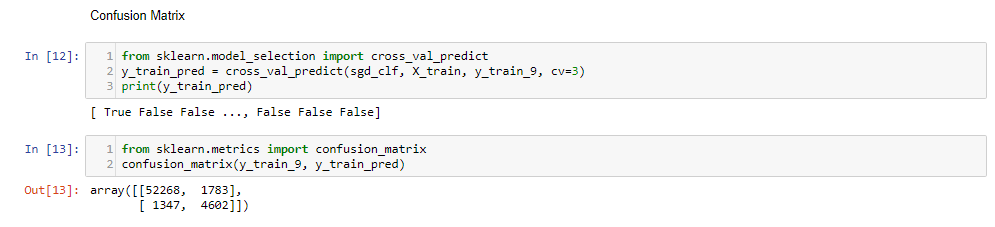
In simulations, we measure and compare the accuracy, confusion metrics, precision, recall, f1 score etc for different classifier on MNIST dataset.

1. Linear classifier:



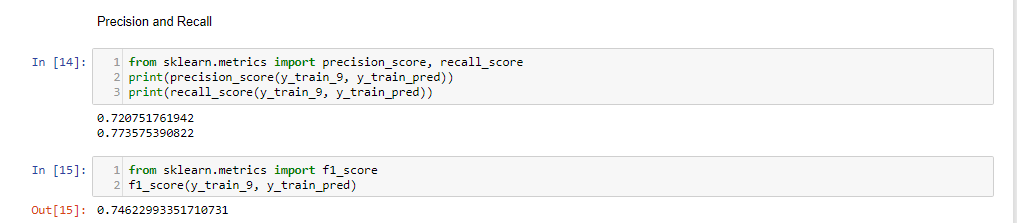
We are getting around 94% of accuracy but this is just for one number and its either true or false. This result will change drastically if we use the larger data and that’s why accuracy is not useful in most performance measure applications.

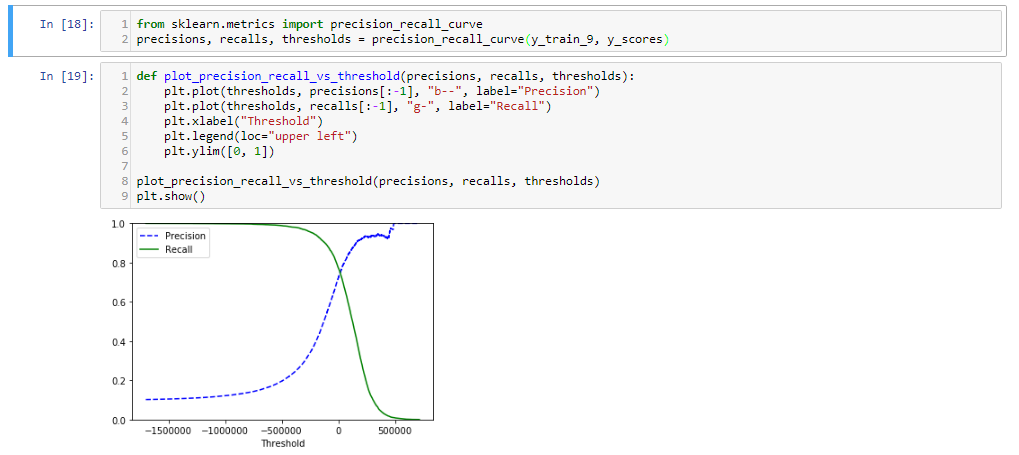
We are using SGD classifier for this binary operation and predicting that number is 9 or not. SGD classifier is very much capable of handling the large dataset and that’s one of the reason of using it.



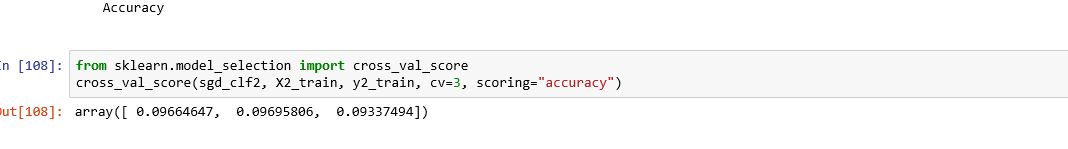
Confusion metrics is much better way to measure the performance. It simply gives the value of correct and incorrect predictions of given number. In our result, we have 52,268 images which are not number 9 and called true negative. We have 4602 images as number 9 and called true positive. Other two parameters are of wrong prediction and says 1783 are false positive and 1347 are false negative. Means either they are true and considered false or other way around.

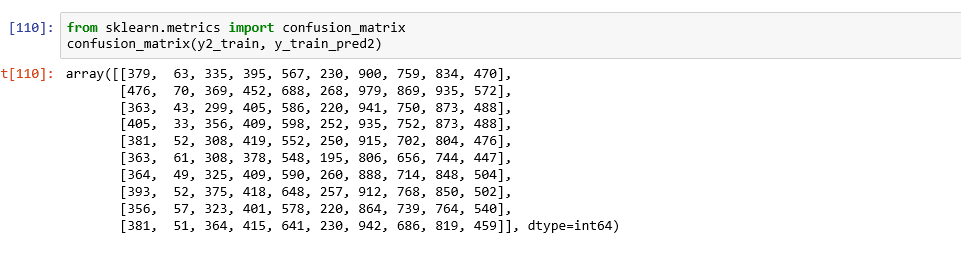
Precision and Recall is more advance and gives us clear results and good understanding of our classifier. We are calculating precision and recall scores importing from sklearn’s metrics function and using precision\_score and recall\_score. Precision score is the calculated by dividing true positive values with addition of true positives and false positives. Recall is measured by dividing True positives values by addition of true positives and false negatives. True positives are when we are expecting 1 and we get 1 as output which means it is correctly classified. False positives are when the actual output was 0 but we got 1 as output which means it was wrongly classified which is same as false negative just the difference is we get 0 instead of 1.

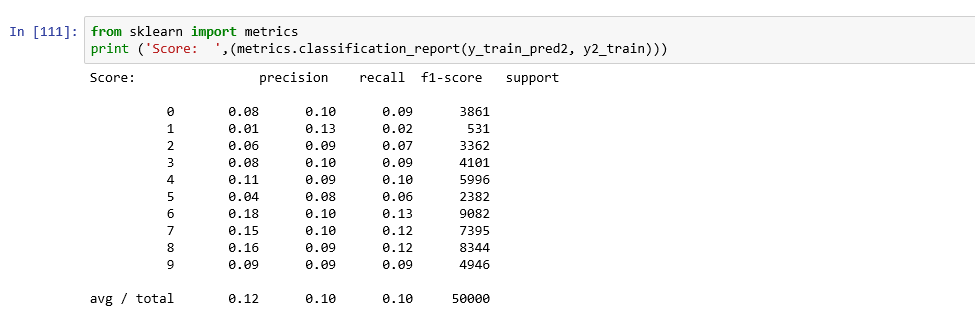




From PR (precision-recall) curve we can see that as we increase recall precision decreases. So precision and recall curves are inversely proportional to each other and called precision vs recall trade-off.

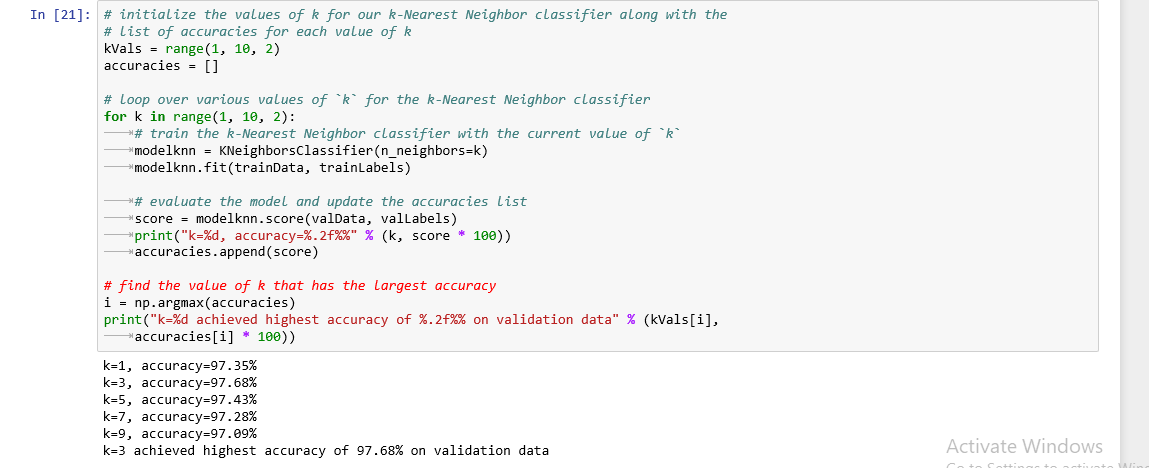




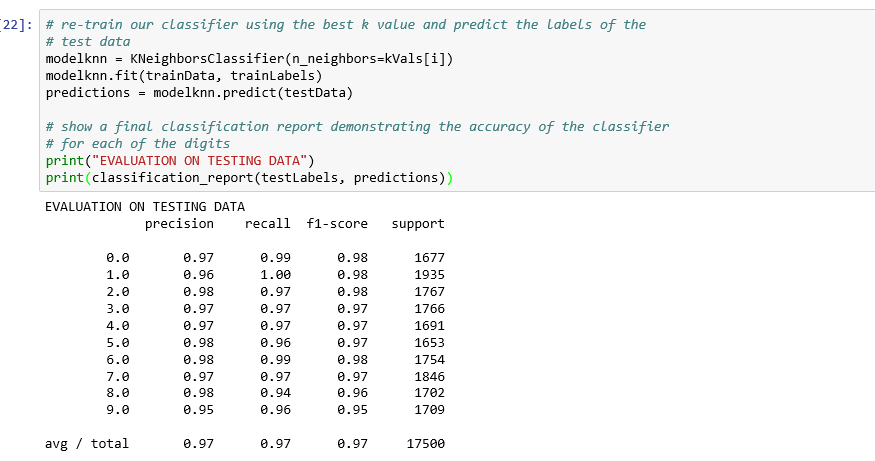


1. K-nearest neighbour classifier

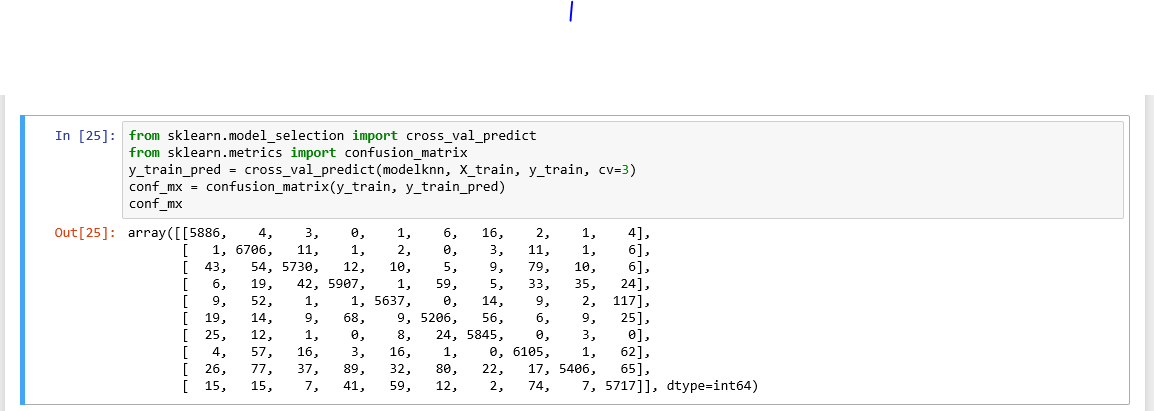
We have divided training and test data by a factor of 0.75:0.25. Then again, we divide training into validation set by a dividing factor of 0.9:0.1. Then we have run a loop of knn classification from 1 to 10 with a step of 2 and compare the accuracy. We see that for k = 3 we get the highest accuracy which is 97.68%. We are running k = 3 on whole data set again after the individual run through.



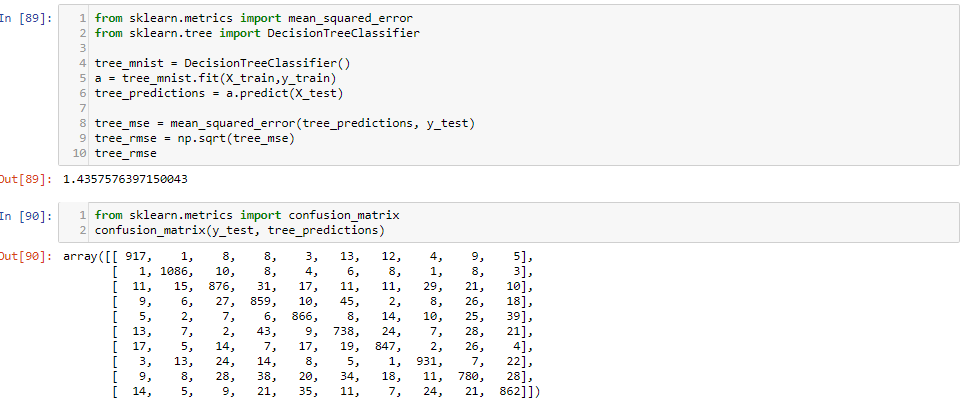
Then the precision and recall is calculated from 0 to 10 range.

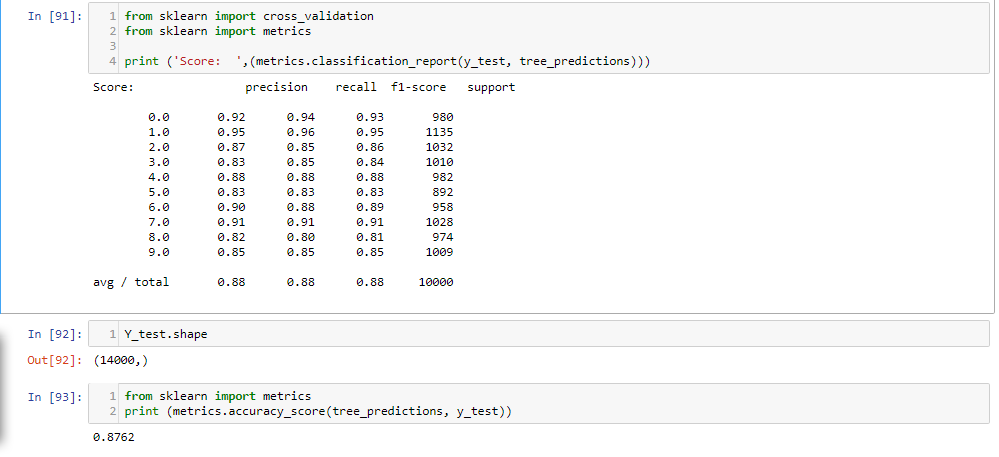


F1 score is calculated by dividing 2 by addition of inverses of precision and recall.

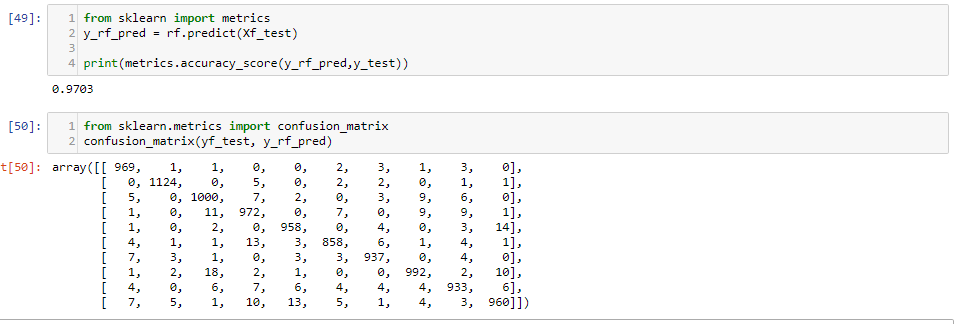


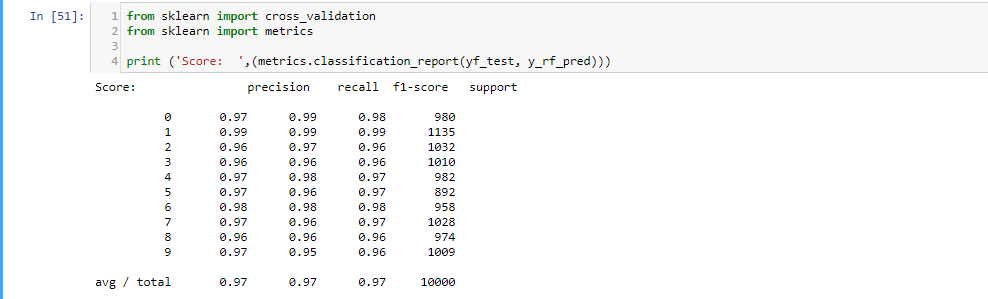
Decision tree:



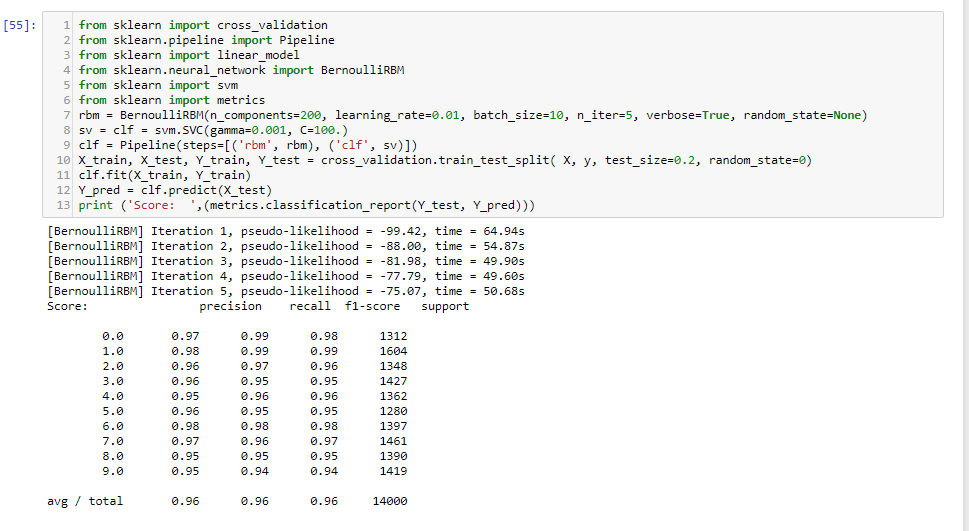


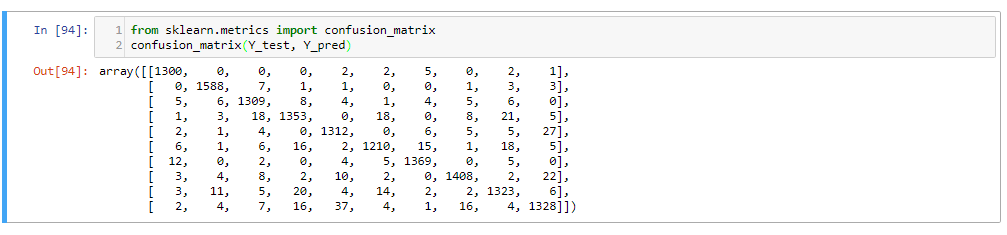
RBM





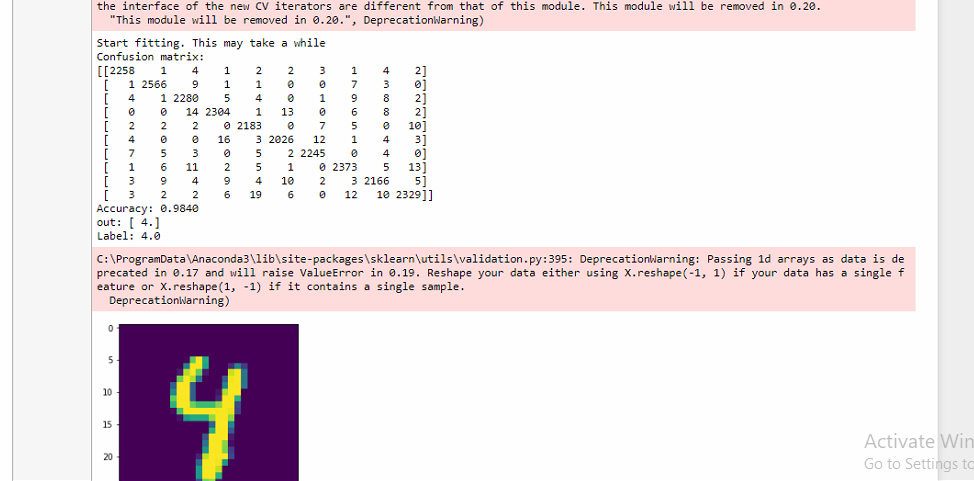
RBM



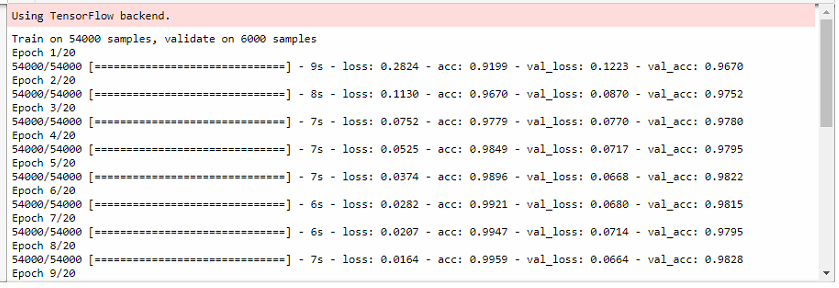


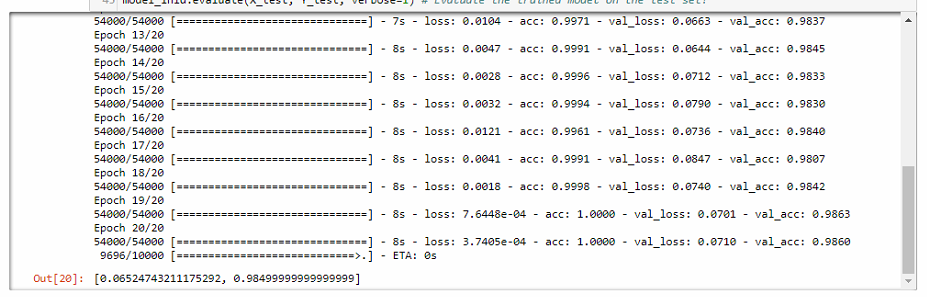
RBF

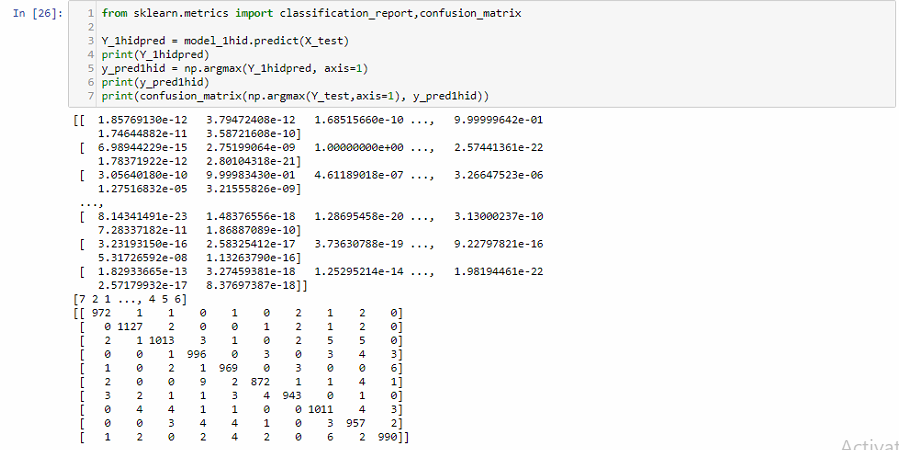
We have calculated the confusion matrix as below and with a accuracy of 0.9840.



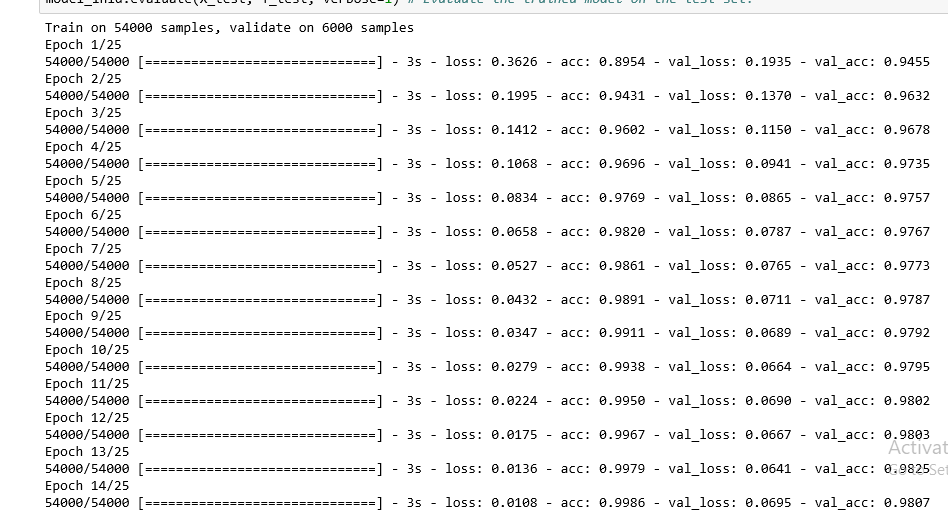
One hidden layer Neural Network:

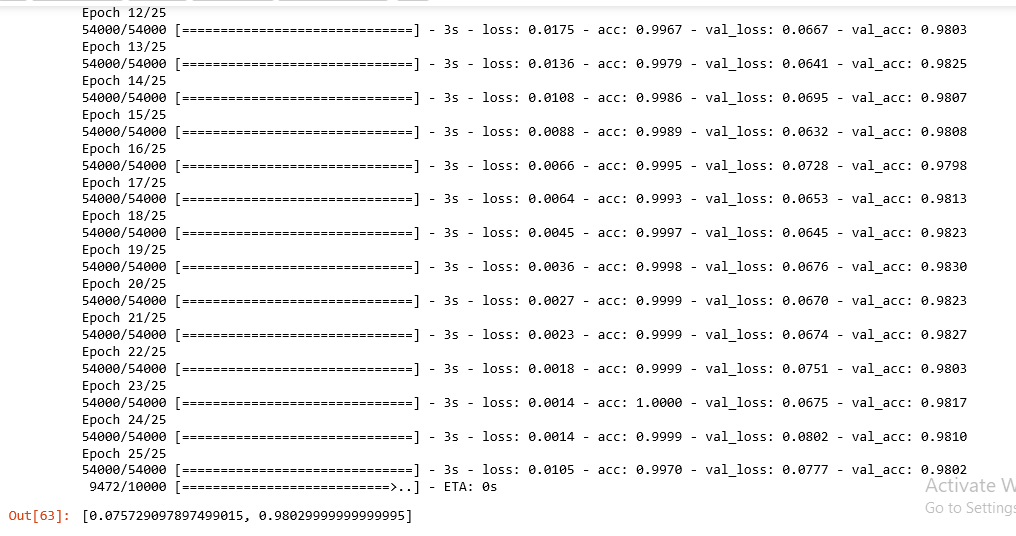


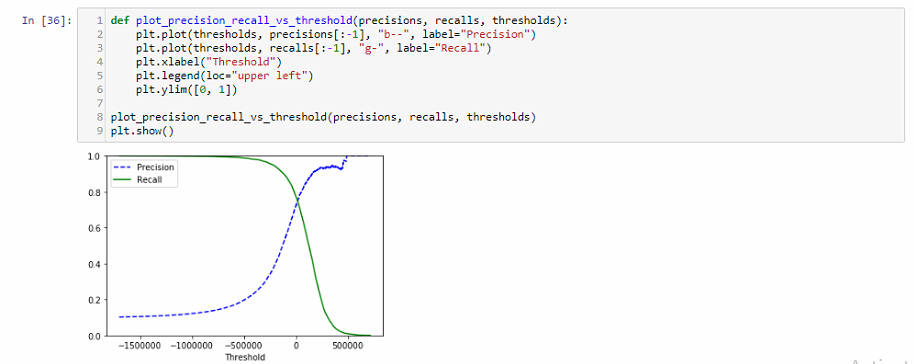




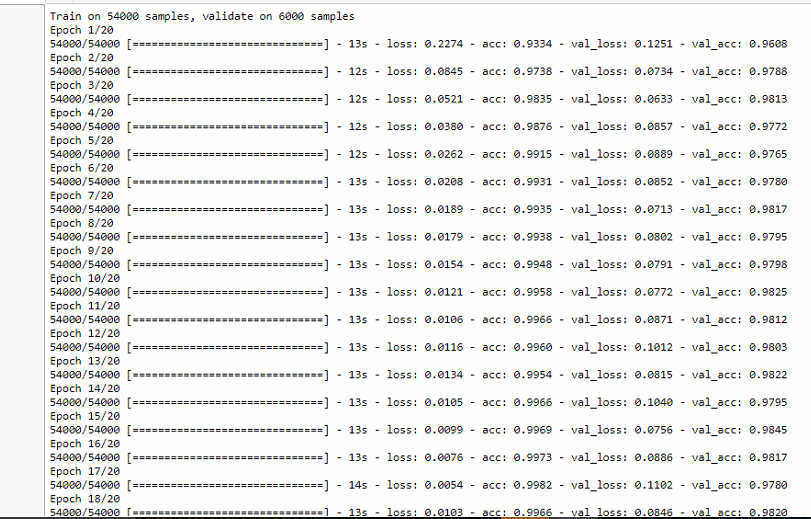
Changed the parameters

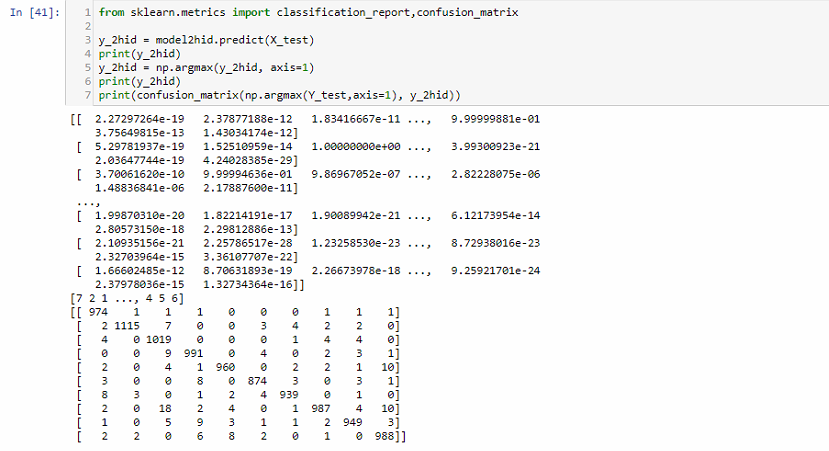




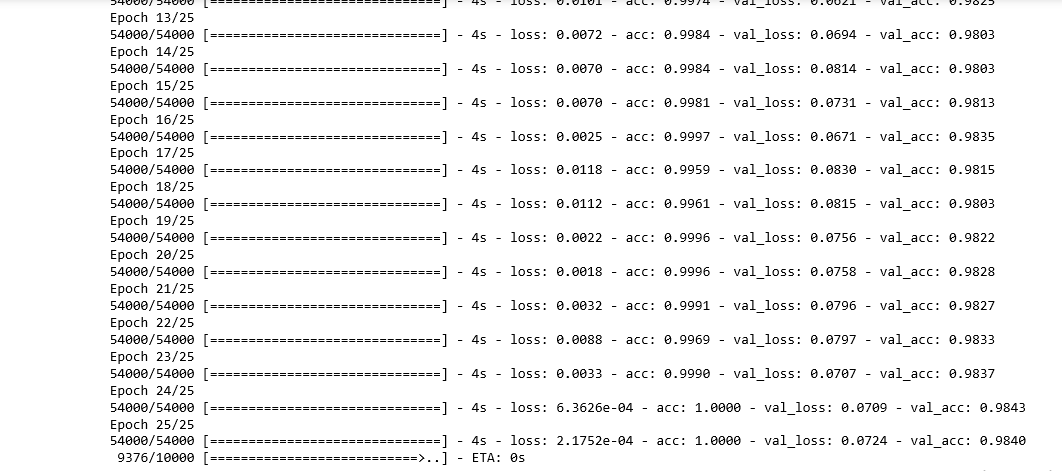


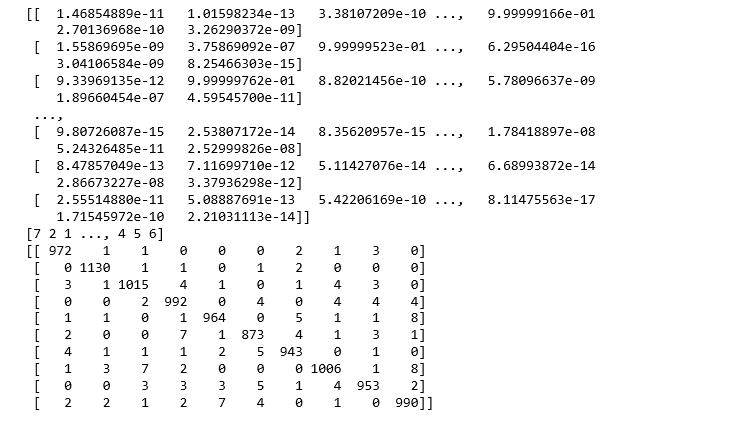
Two hidden layer Neural network





Now we have changed the training sample size to 50000 images. As we can see that computational time has decreased and it has not affected the precision and recall score much.

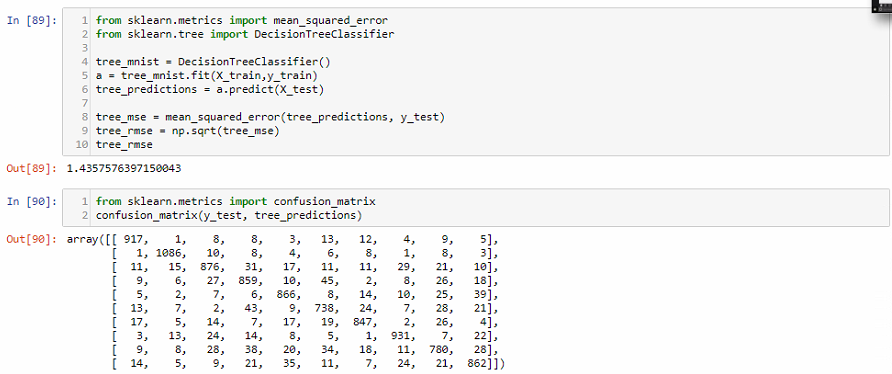


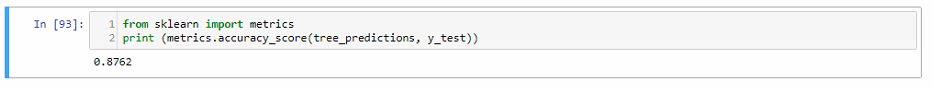
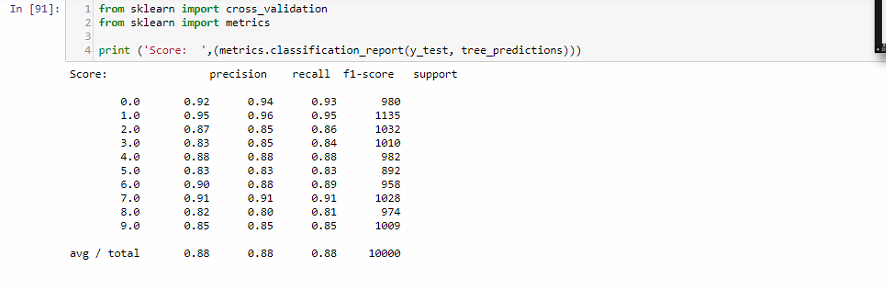


Decision Tree:

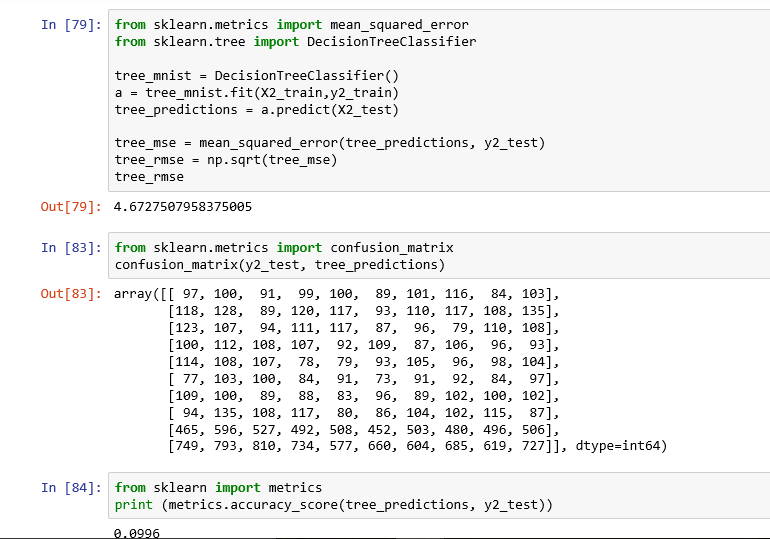
In decision tree, we ask questions like after this how it will be divided. Then a node is created and tree is formed.

RMSE



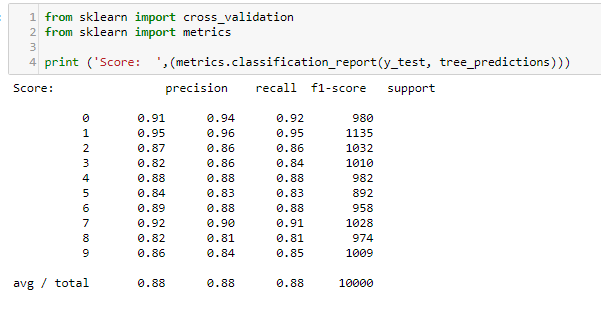


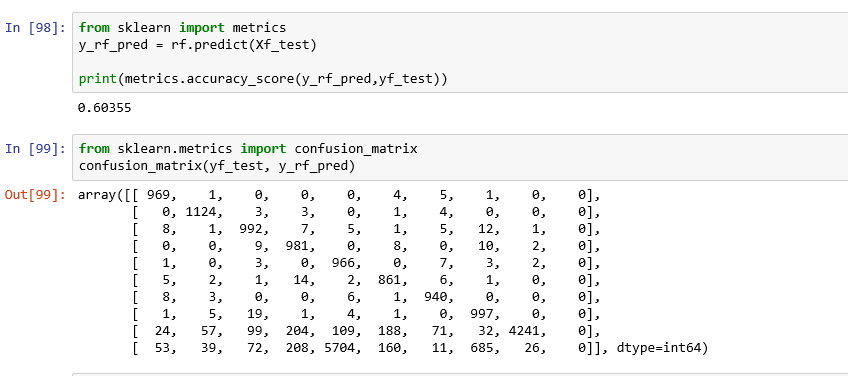
Changed training sample size, accuracy dropped.



Random forest



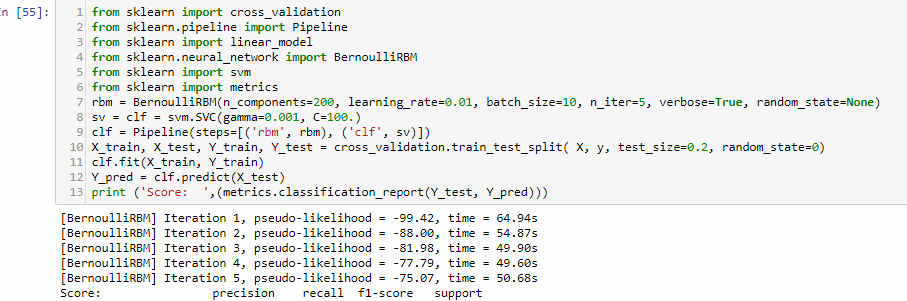




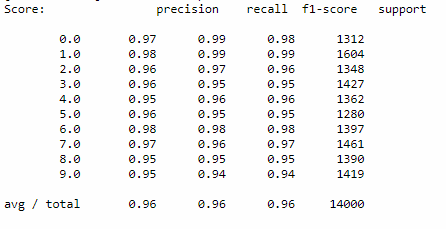
Decreasing training sample size decreases accuracy.

RBM (Restricted Boltzmann Machine):

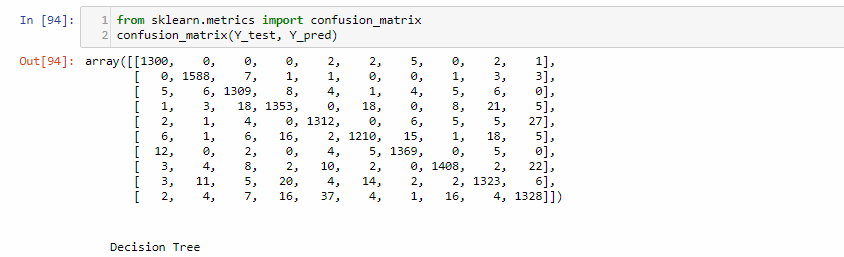
Time taken and pseudo -likelihood



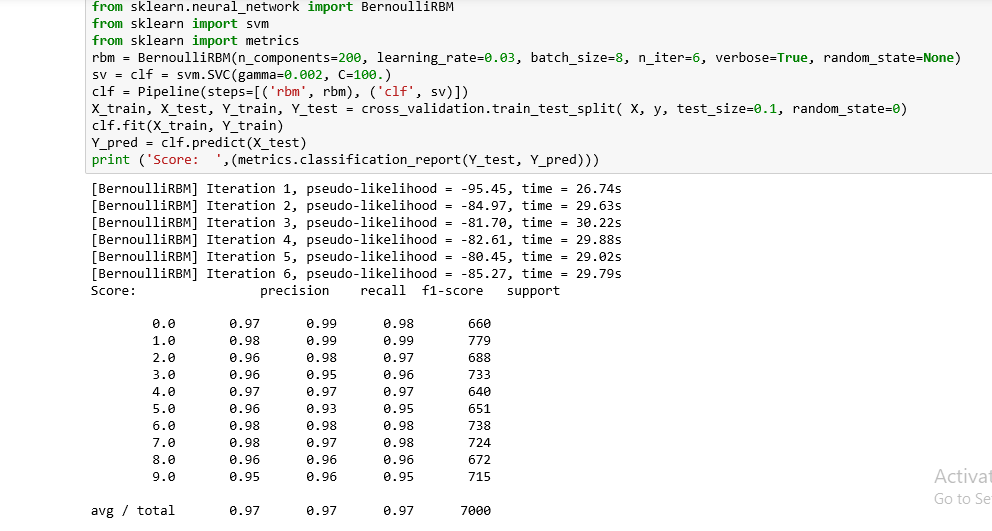
Precision Recall table:



Confusion Matrix



Now we have changed the training sample size to 50000 images. As we can see that computational time has decreased and it has not affected the precision and recall score much.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr No. | Name | Accuracy | Precision, recall | Run Time | F1 Score | Error |
| 1 | Linear | 94.5% (Best) | 72.075%, 77.357% | 300 msec | 74.622% | Around 6.5% |
| 2 | KNN | 97.68% (For k = 3) | 97%, 97% | 70 Minutes | 97% | - |
| 3 | RBF | 98.40% | - | 30 minutes |  | - |
| 4 | 1 Hidden Layer MLP | 98.63% | - | 7 secs for every 20 iterations | - | 7.6448e-04 |
| 5 | 2 Hidden Layer MLP | 97.93% | - | 13 secs for every 20 iterations | - | 0.0055 |
| 6 | Decision Tree (for score = 2) | 87.62% | 87%, 85% | 3 mins | 86% | - |
| 7 | Random Forest (for score = 2) | 97.03% | 96%, 97% | 2 mins | 96% | - |
| 8 | RBM | - | 96%, 97% | 1 min | 96% | - |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr No. | Name | Accuracy | Precision, recall | Run Time | F1 Score | Error |
| 1 | Linear | 96.64% (Best) | 6%, 9% | 300 msec | 7% | Around 90.5% |
| 2 | KNN | 97.68% (For k = 3) | 97%, 97% | 70 Minutes | 97% | - |
| 3 | RBF | 98.32% | - | 30 minutes |  | - |
| 4 | 1 Hidden Layer MLP | 99.7% | - | 3 secs for every 20 iterations | - | 0.0105 |
| 5 | 2 Hidden Layer MLP | 100% | - | 4 secs for every 20 iterations | - | 2.1752e-04 |
| 6 | Decision Tree (for score = 2) | 9.96% | - | 5 mins | 97% | - |
| 7 | Random Forest (for score = 2) | 97.03% | 96% | 3 mins | 96% | - |
| 8 | RBM | - | 96%, 98% | 1.5 min | 96% | - |

Conclusion:

Changing parameters like number of epochs, iterations, training and test data size, activation function etc changes the accuracy and PR curve. The confusion matrix and its graph are suffice for the output.