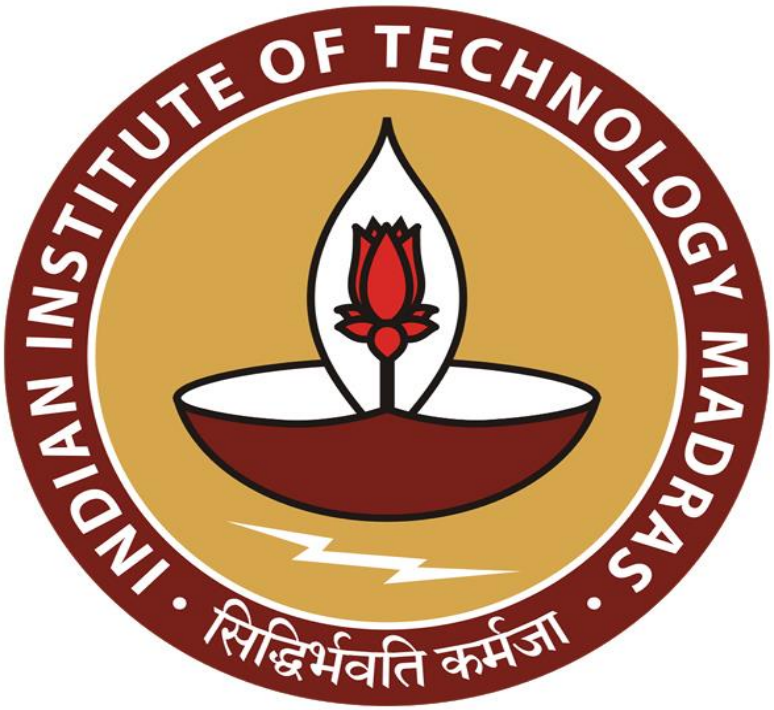
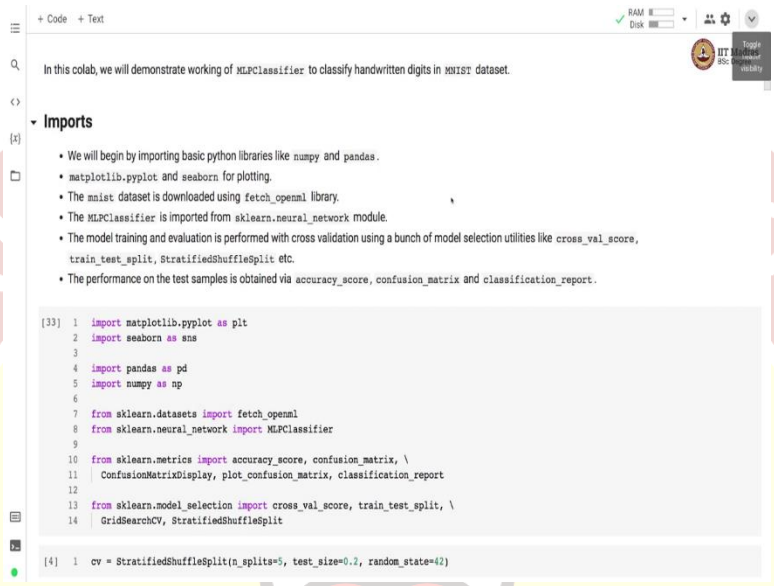


IIT Madras

ONLINE DEGREE

Neural Networks
Professor. Doctor. Ashish Tendulkar
Indian Institute of Technology, Madras
Multilayer Perceptron Classifier on MNIST

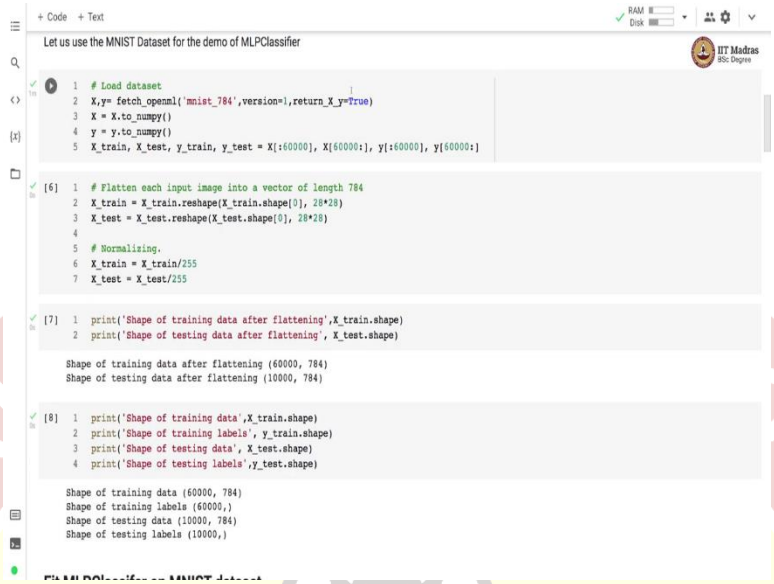
(Refer Slide Time: 00:11)



Namaste! Welcome to the next video of Machine Learning Practice Course. In this collab will demonstrate working of MLPClassifier to classify handwritten digits in this dataset. We will begin by importing basic Python libraries like pandas and numpy, matplotlib and seaborn for plotting. The mnist dataset is downloaded using fetch _openml library. The MLPClassifier is imported from sklearn.neural network module.

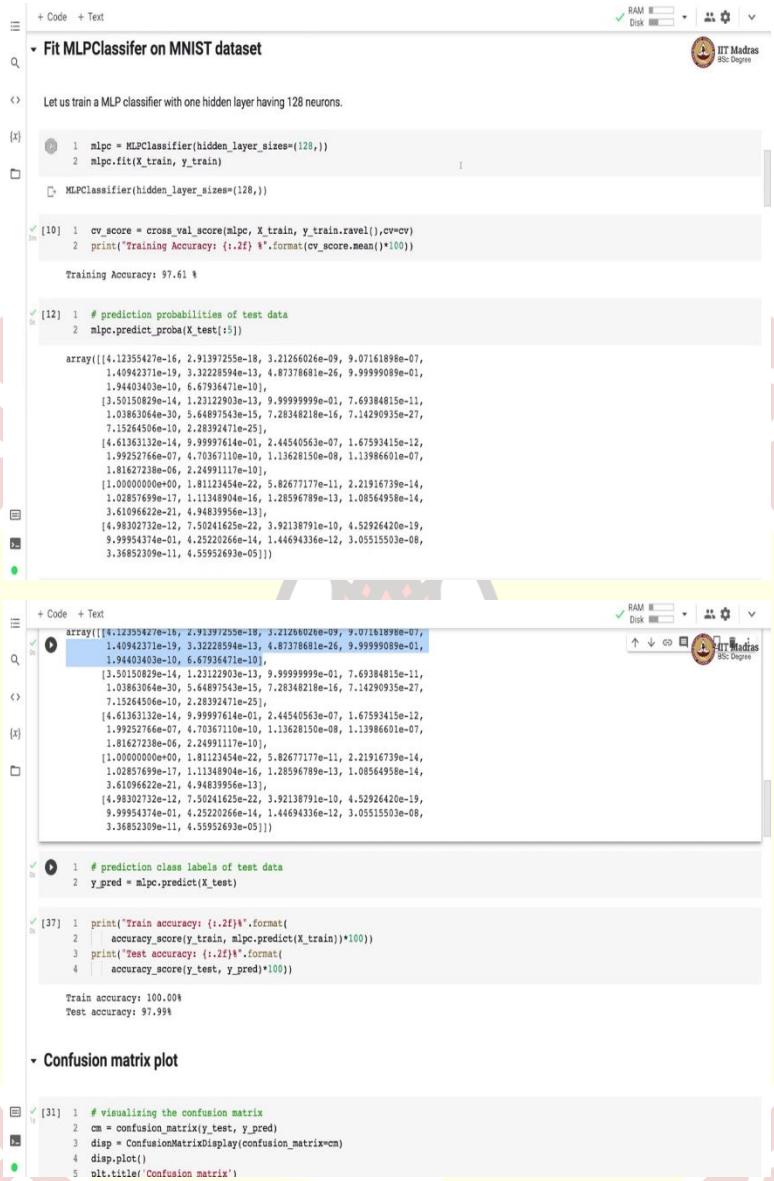
The model training and evaluation is performed with cross-validation using a bunch of model selection utilities like cross _val _score, train _test _split GridSearchCV and StratifiedShuffleSplit. The performance on the test sample is obtained via accuracy _score, confusion _matrix and classification _report, which are imported from sklearn.metrics module. We create StratifiedShuffleSplit with number of splits = 5 and setting a size 20% examples as test. We will be using this particular cross-validation strategy for training the model.

(Refer Slide Time: 01:27)



Let us use the MNIST dataset and that MNIST dataset is fetched using fetch _openml appear by specifying mnist _784 as the dataset string. As usual, we use first 60,000 examples as studying example, and remaining 10,000 examples as test examples. You flatten each input image into a vector of length 784. As we have been doing throughout this course, we normalize each image by dividing the pixel value by 255 in both training and test set. After flattening, we have training examples, there are 60,000 training examples, each represented with 784 features. And there are 10,000 test examples and each example is represented with 784 features.

(Refer Slide Time: 02:20)



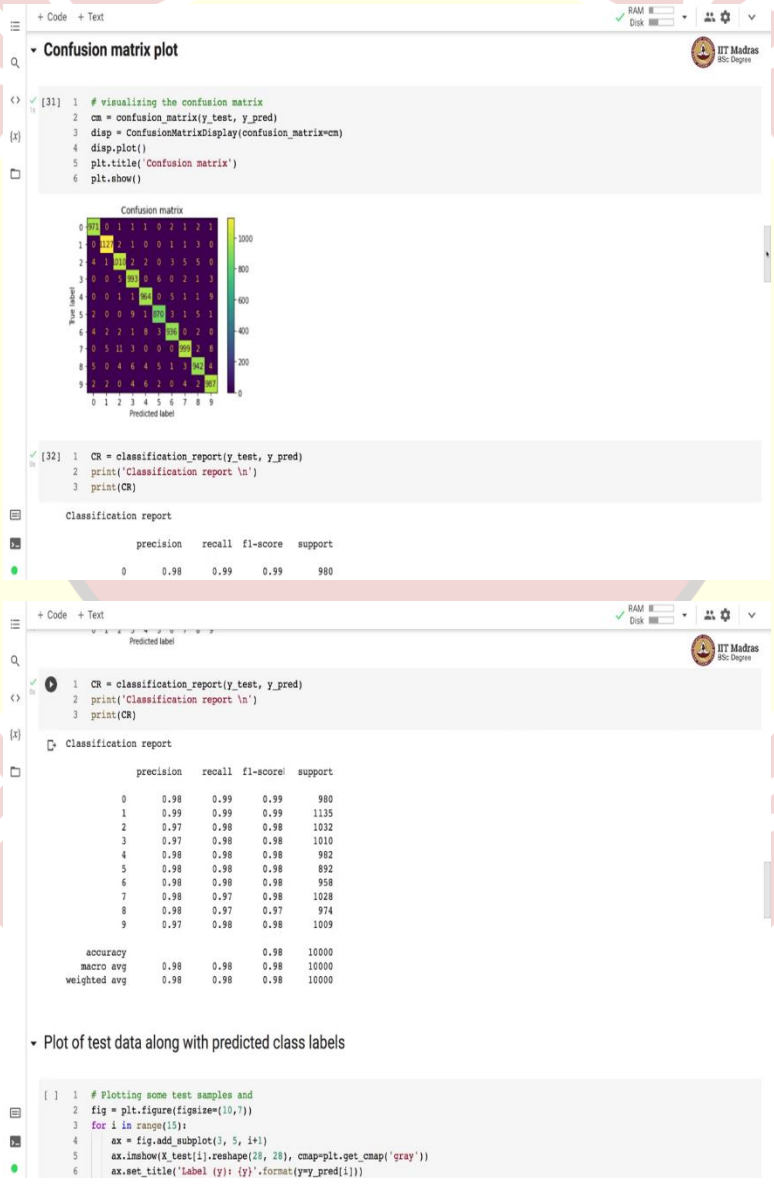
Let us train MLPClassifier with 1 hidden layer having 128 neurons. So, we instantiate MLPClassifier by specifying hidden _layer _sizes, and we used 1 hidden layer with 128 neurons. And we call the fit method by supplying the training feature matrix and training labels as input. After training the MLPClassifier, we obtained cross valid score using the cross-validation which is StratifiedShuffleSplit as specified earlier.

And this cross-validation uses five-fold cross-validation. So, we obtained accuracy of 97.61 % in the cross validated region, we can predict probability on the test data using this model by calling predict _proba method on the MLPClassifier object. And here we have shown probability vectors

for first five test examples. So, each vector that you see here is a probability distribution of that example, over 10 different classes in handwritten digit dataset.

We can use predict method to get class labels of test set. So, now we have predictions for test set in y _pred variable. We use this particular value or these predictions for calculating test accuracy, we also obtain train accuracy and compare them. So, we obtain almost 100% training accuracy and the test accuracy is close to 98%.

(Refer Slide Time: 04:05)

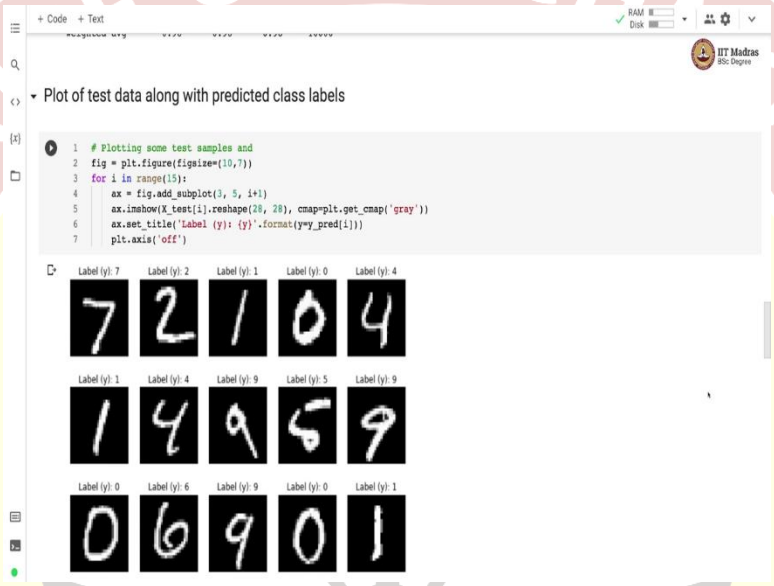


You plot the confusion _matrix for the test examples and you can see that this is almost a perfect confusion _matrix with very little confusion among different classes. So, our diagonal entries are

very small, if you compare this confusion _matrix with other confusion _matrix that we had obtained with other classifiers in the course. The classification _report shows that the test accuracy average test accuracy is 9 %.

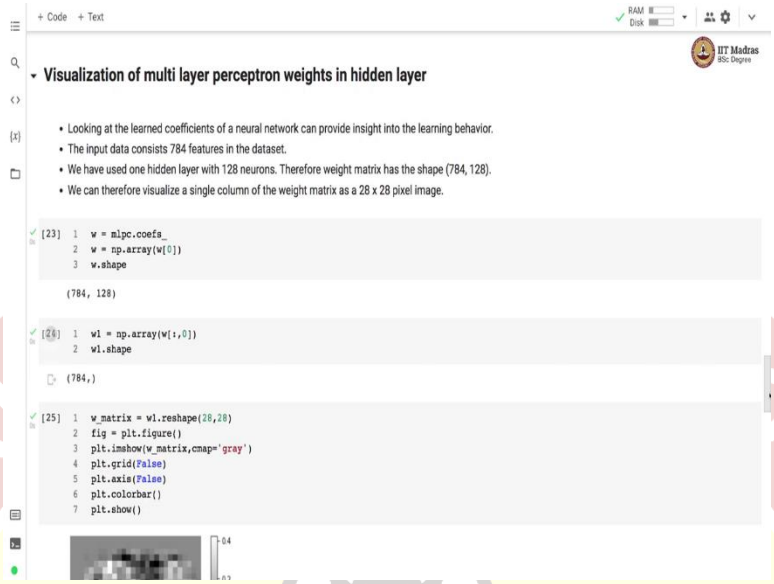
And for most of the classes the accuracy is > or = 97%. The highest accuracy that was obtained is 0.99 or 99% on class 0 and 1.

(Refer Slide Time: 04:50)



Here we have plotted the test data with their with the predicted labels, and we have plotted 15 examples over here and you can see that most of the examples have been correctly predicted using our MLPClassifier.

(Refer Slide Time: 05:13)

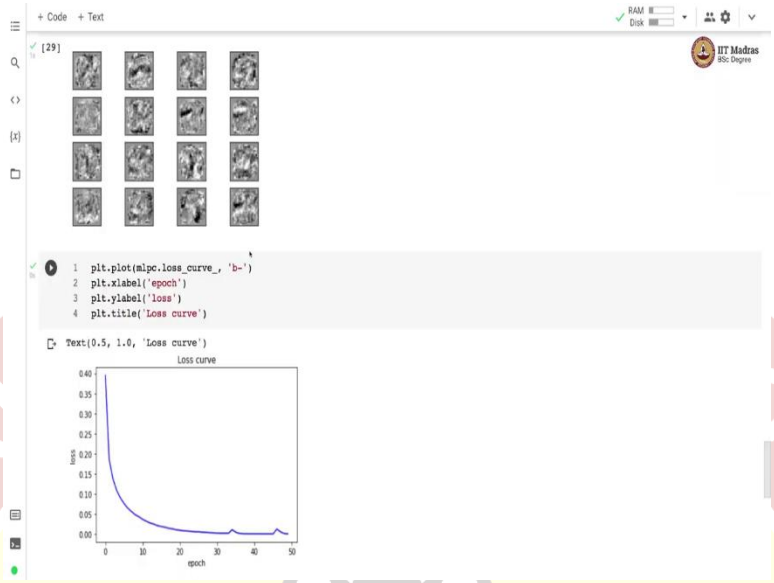


Finally, we will visualize the weights that are learned by multi layer perceptron or MLPClassifier. Since you have used 1 hidden layer with 128 neurons, the weight matrix will have shape of 784 by 128. We visualize a single column of weight matrix as 28 × 28 pixel image. So, you can see that the shape of the weight matrix for the hidden layer is 784 by 128. And for w1 the shape is 784.

(Refer Slide Time: 05:51)

And this is a sample matrix the weight matrix that is learned by our MLPClassifier.

(Refer Slide Time: 05:59)



Finally, we look at loss curve of the train classifier. And you can see that the training loss goes down consistently as we train the model for more epochs, which is a sign of well-trained classifier. So, in this video, we demonstrated how MLPClassifier can be used for training, a model for recognizing handwritten digits from MNIST dataset, you can use MLPClassifier for many such kind of task, not only in the image recognition but it can also be used for structured datasets.