INDIAN INSTITUTE OF TECHNOLOGY MADRAS

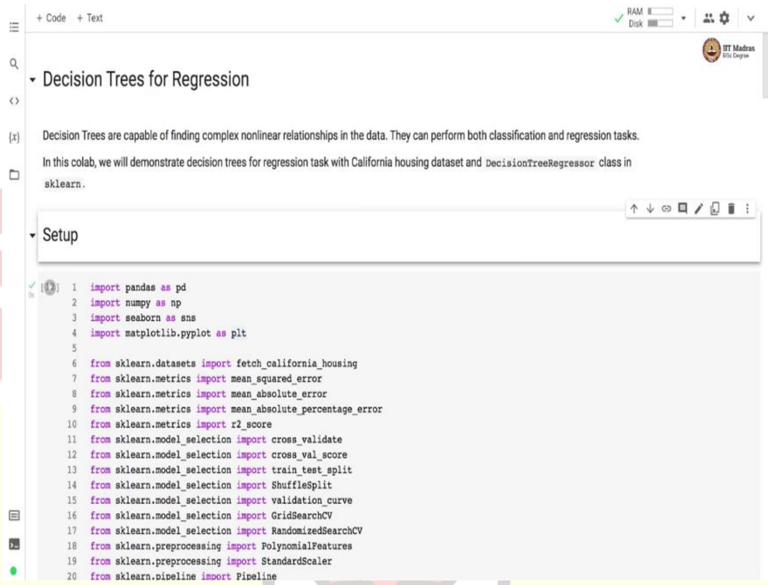
सिद्धिर्भवति कर्मजा

IIT Madras

ONLINE DEGREE

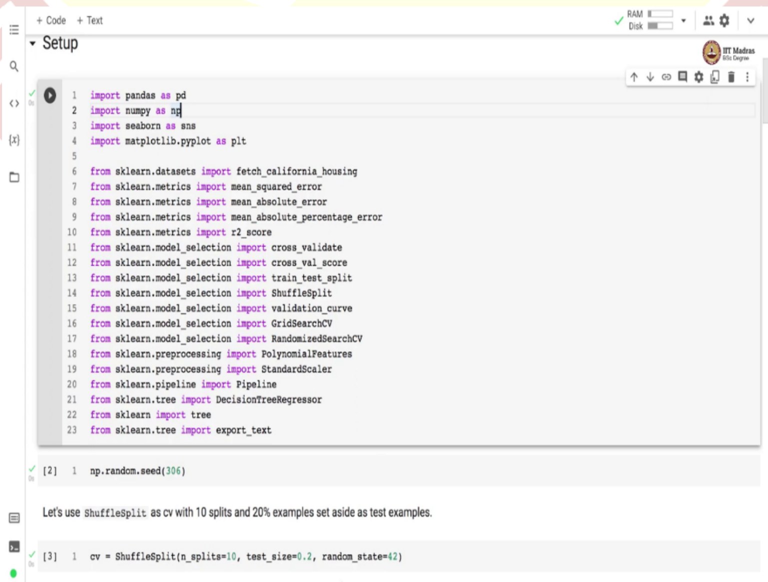
Machine Learning Practice
Professor Dr. Ashish Tendulkar
B. Sc in Programming and Data Science
Indian Institute of Technology, Madras
Decision Trees for Regression

(Refer Slide Time: 0:11)



Namaste! welcome to the next video of Machine Learning Practice Course. In this video, we will demonstrate decision trees for regression. As you know, decision trees are capable of finding complex nonlinear relationships in the data, they can perform both classification and regression task. In this collab will demonstrate decision trees for regression tasks with California housing dataset, and DecisionTreeRegressor class in sklearn.

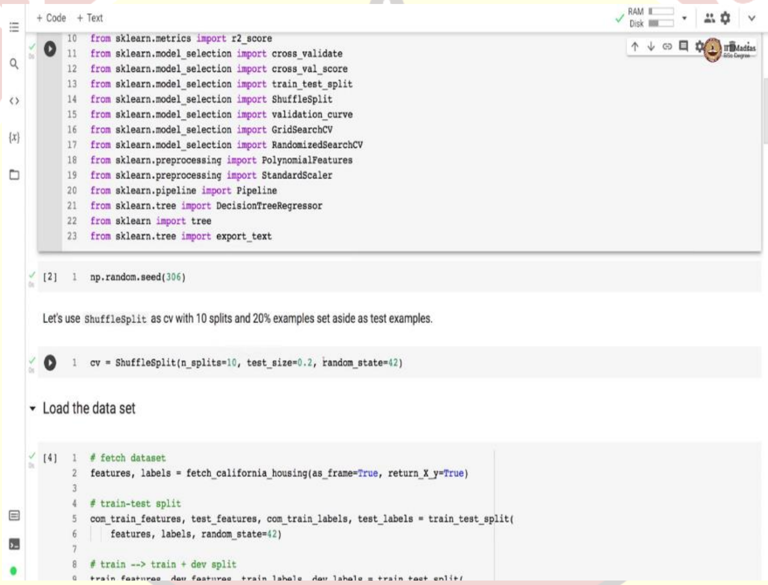
(Refer Slide Time: 0:42)



Let us begin by importing all necessary libraries, we import a bunch of basic Python libraries for data handling and plotting. Since we are going to use California housing dataset, we import fetch _California _housing API from sklearn . dataset then bunch of metrics like mean _squared _error mean _absolute _error, and mean _absolute _percentage _error and r2 _score.

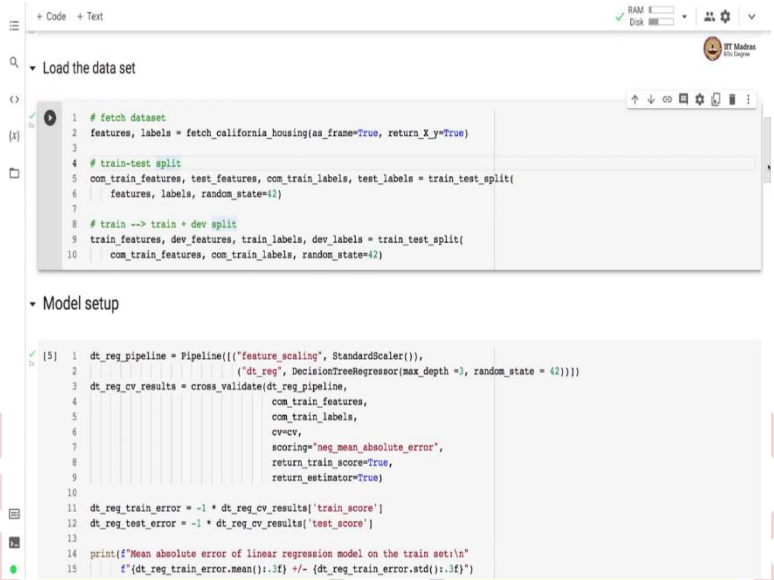
Then there are a bunch of model selection utilities that are imported, followed by the pre-processing utilities, the pipeline utility, and here we are going to use DecisionTreeRegressor as a class, and there is export tree and tree API's that are imported for visualising the tree in graphical format as well as in the text format.

(Refer Slide Time: 01:35)



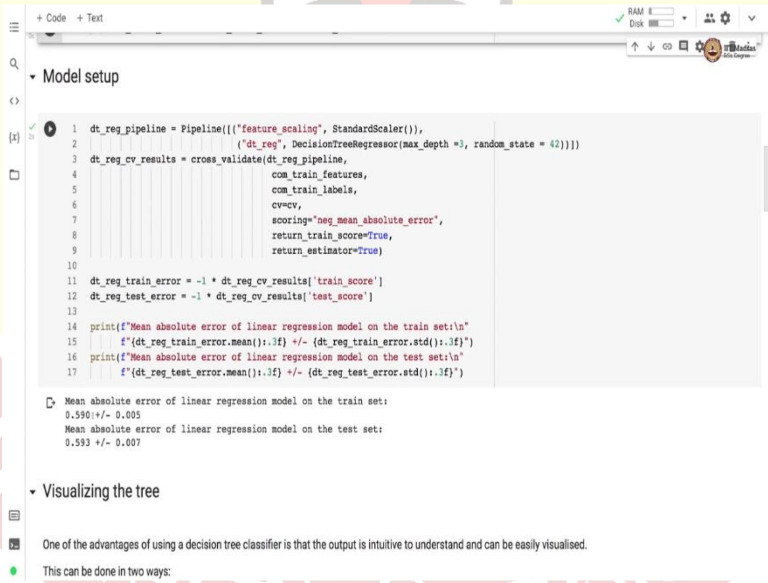
As usual, it is a good practice to set the seed and we set the random seed to 306. In this case, we use ShuffleSplit CV as a cross-validation with 10 splits, and 20% example set aside as test examples.

(Refer Slide Time: 01:54)



Next, we load the dataset using fetch _California _housing API, we perform training test split, then we perform the further split of the training set into train and development sets.

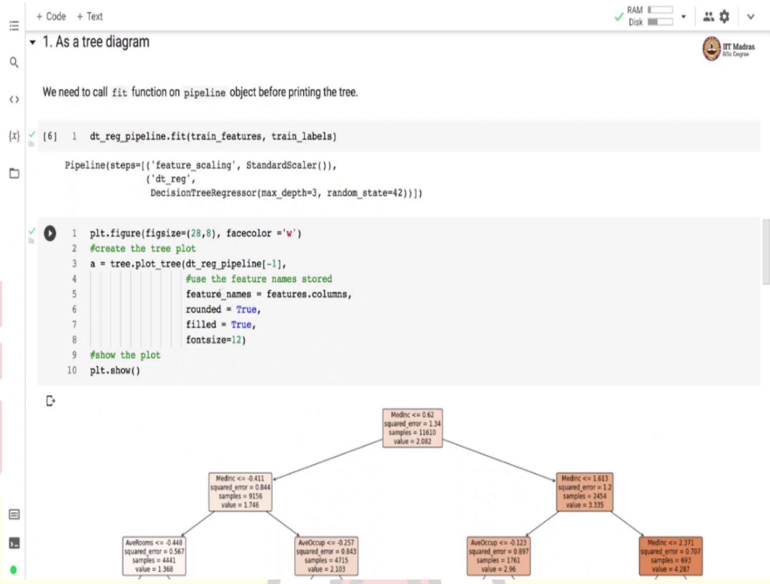
(Refer Slide Time: 02:10)



We set up a pipeline model, where we are using feature scaling as a pre-processing step. And for feature scaling, we are using StandardScaler. And then we are defining the DecisionTreeRegressor with max _depth = 3, we perform the training with cross _validate. And here we use all of the training data, we use ShuffleSplit CV and negative mean _absolute _error as a scoring mechanism.

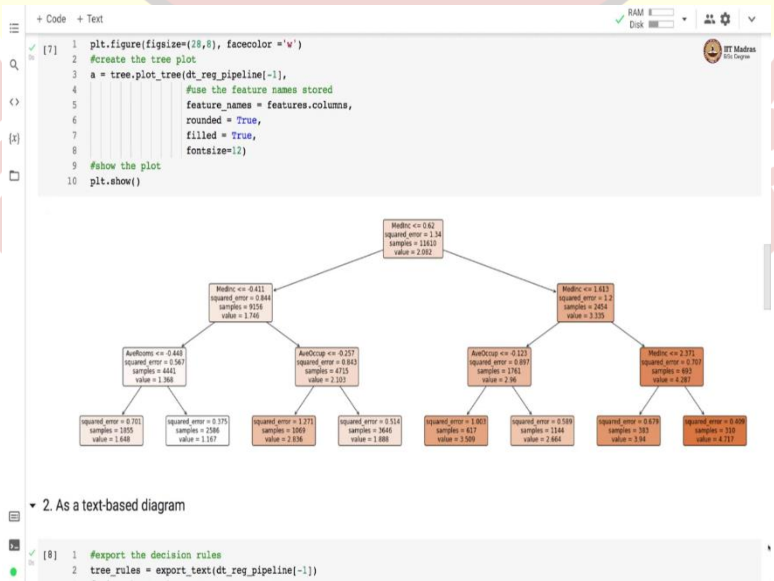
So, after performing the cross _validate regression based training, what we obtain is the matrix on the train set and test set. So, we have the mean _absolute _error of 0.59 on the training set and 0.593 on the test set, the standard deviation is pretty small, which indicates that the mean _absolute _error is consistent across different validation sets.

(Refer Slide Time: 03:12)



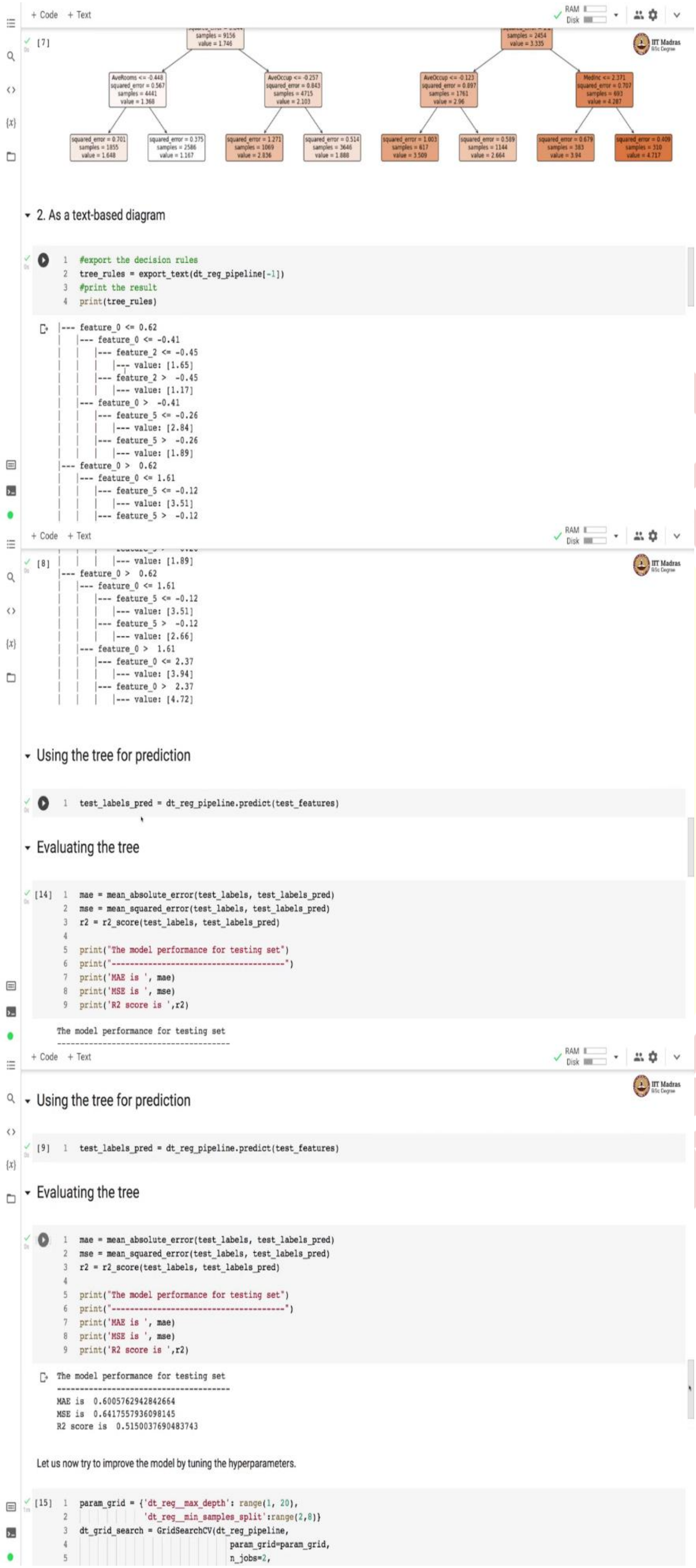
Let us visualize the tree that we have learned through this process. So there are two ways to visualize a tree, using a tree diagram and second is a text based diagram. So we need to call a fit function on the pipeline object before printing the tree otherwise, the print function returns errors. So, here after calling the fit on the pipeline, we use the estimator or specifically, we specify the tree for plot _tree. And here we also specify the feature _names.

(Refer Slide Time: 03:51)



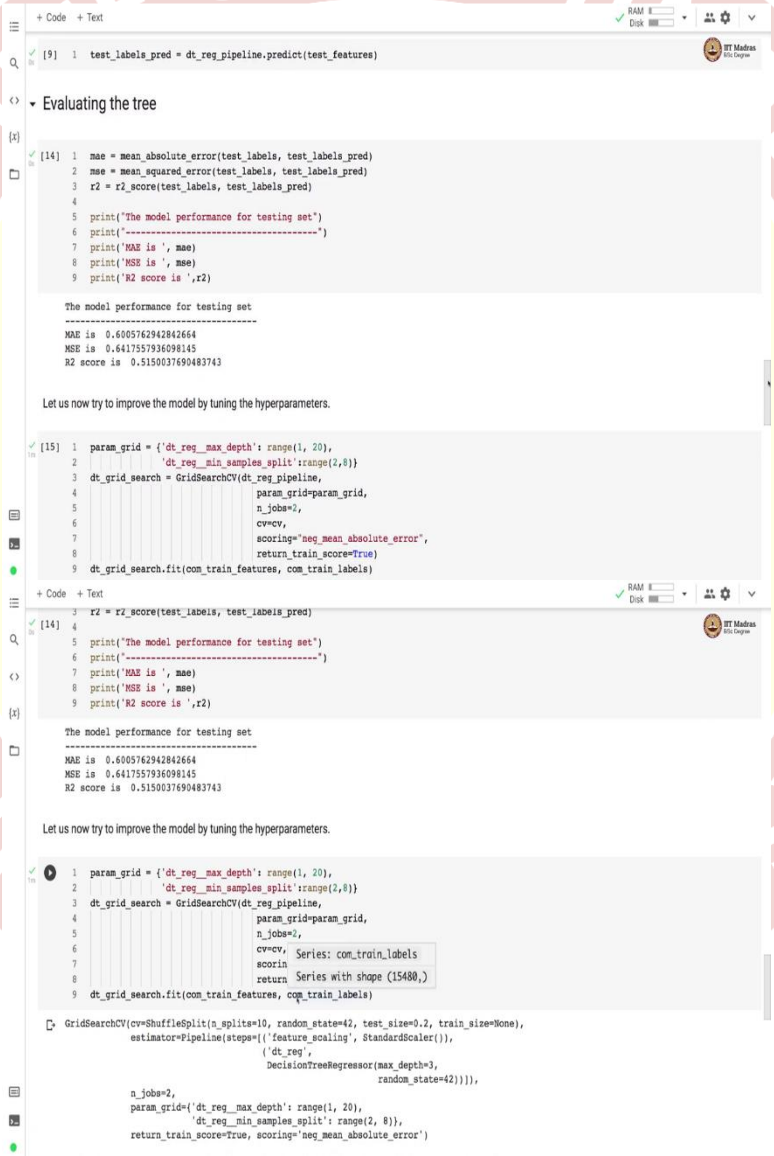
So, what you see on your screen is a tree with depth of 3. And in this tree we see the split criterion which is mid income or median income less than 0.62, there is a split on median income also in the, at the second level of the tree. Then the third level splits are different for different branches, here we are splitting based on average room size, here based on average occupancy in these two cases. And here again based on median income. And we also print the squared error for each of the node and number of samples and values.

(Refer Slide Time: 04:38)



We can also convert this visual representation into text representation by calling export _test function. And in export _test, we basically get the rule sets. So this is more like, if else conditions as you can convert this tree into this rule set. Then we use this tree for prediction by calling the predict function and passing the test feature matrix as input. Based on the predicted labels, we print a bunch of evaluation metrics like mean _absolute _error, mean _squared _error and r _score. The mean _absolute _error is 0.60, mean _squared _error is 0.64 and r2 _score is 0.51.

(Refer Slide Time: 05:27)

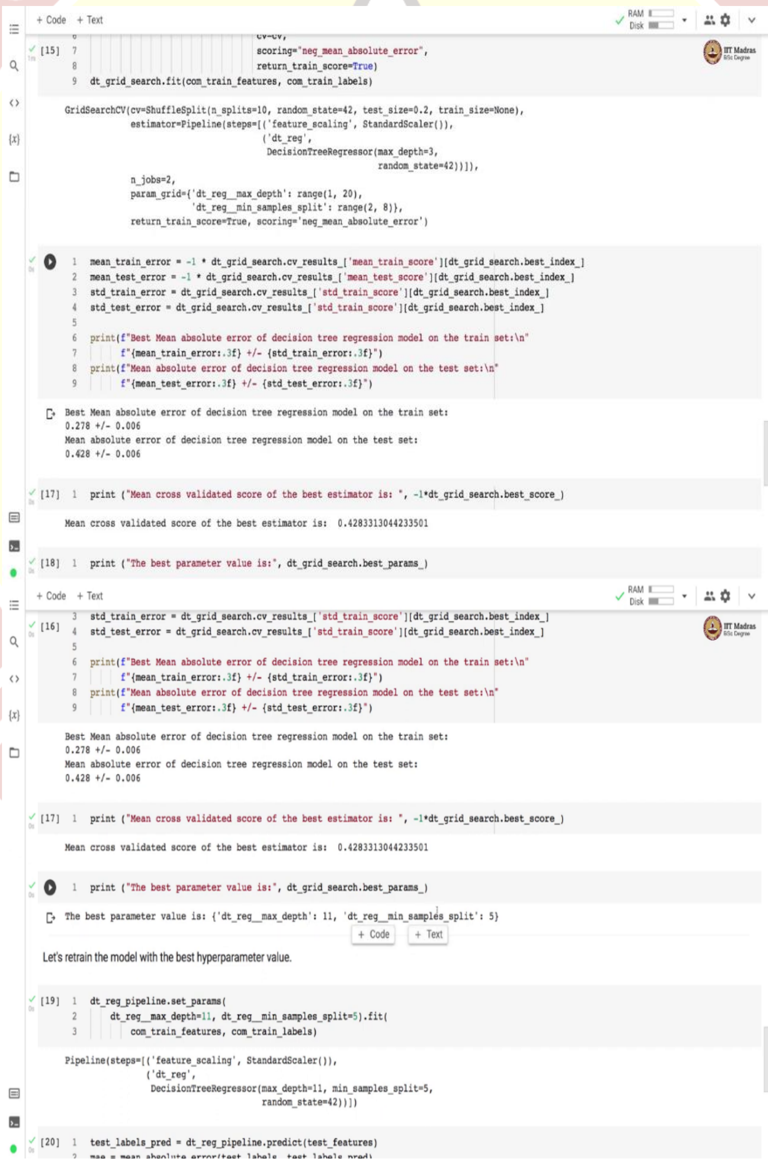


Now, let us try to improve this model by tuning the hyper-parameter. There are two hyper-parameters in case of trees, one is max _depth and second is min _samples _split. And these two hyper-parameters are of our interest for this particular exercise and we will try to tune them.

So here we define the parameter grid where we want to try max _depth from 1 to 20 and min _samples _split from 2 to 8. We define a GridSearchCV object. So here we are using GridSearchCV for hyper-parameter tuning, we specify the tree estimator and the parameter grid which is specified over here. And we are going to use again the ShuffleSplit CV as a way of performing the cross-validation.

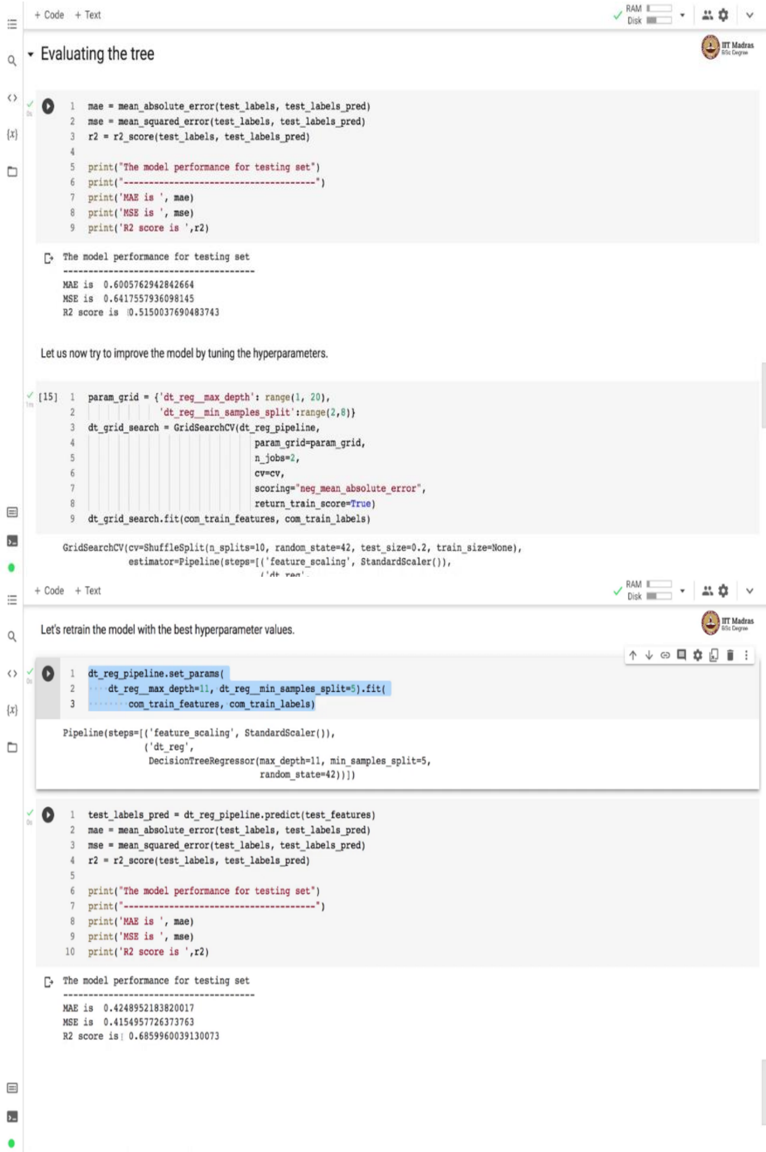
We are using negative mean _absolute _error for scoring. We fit the GridSearchCV object with all of the training data, which includes the combined training features and combined training labels. So we are using the combined training data which has got training as well as development set.

(Refer Slide Time: 06:40)



After performing the hyper-parameter search, we found out the best mean _absolute _error obtained on the training set as well as on the test set. So you can see that on the training set, the best mean _absolute _error that is obtained is 0.278. And on test set, it is 0.428. And the best parameter values for the tree are the max _depth = 11 and mean _samples _split = 5.

(Refer Slide Time: 07:07)



We retrain our model with the best hyper-parameter values, these best hyper-parameter values are set over here, and then we call the fit function on our pipeline. And then we obtain the retrain pipeline with the best hyper-parameter values. We perform the prediction on the test set and calculate various metrics on the test set in order to evaluate the retrained pipeline.

Here we see that the mean _absolute _error has gone down to 0.42. And the r2 _score has improved to 0.68. So if we compare this with the values that we were having over here, here the mean _absolute _error was 0.60 that is now reduced to 0.42. And r2 _score has jumped from 0.51 to 0.68.

So, in this video, we use decision trees for regression problem with California Housing dataset. We also demonstrate how to perform hyper-parameter search, and saw that after performing hyper-parameter search and retraining the pipeline, some of the metrics like r2 _score, mean _absolute _error, and mean _squared _errors improved.