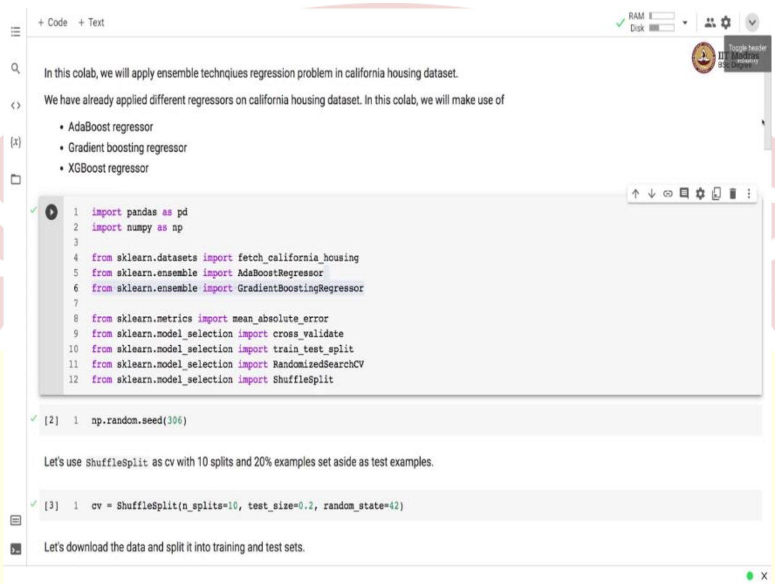
IIT Madras

ONLINE DEGREE

Machine Learning Practice
Professor. Ashish Tendulkar
Indian Institute of Technology, Madras
AdaBoost and GradientBoost Regressor on California Housing

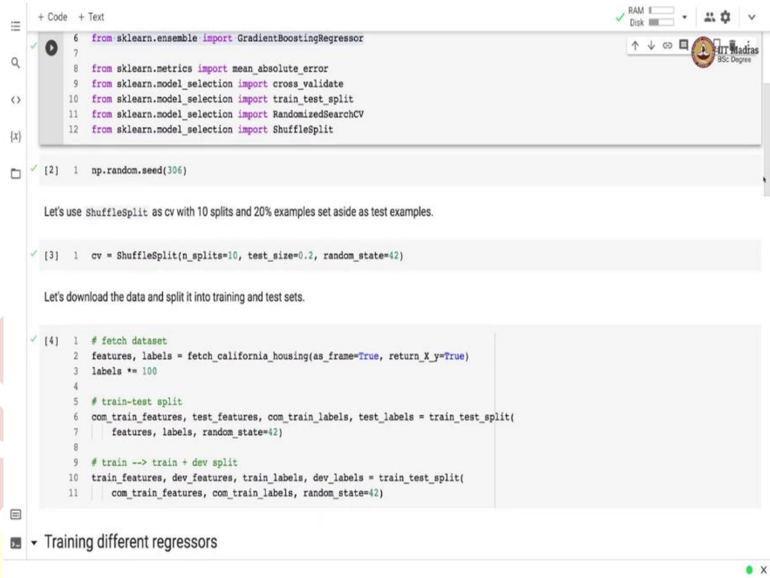
(Refer Slide Time: 00:10)



Namaste! Welcome to the next video of Machine Learning Practice Course. In this video, will apply boosting technique for California housing dataset, which is a dataset with regression problem. In this problem, we tried to predict the price of the house based on various attributes. We have already applied different regressions on California Housing dataset.

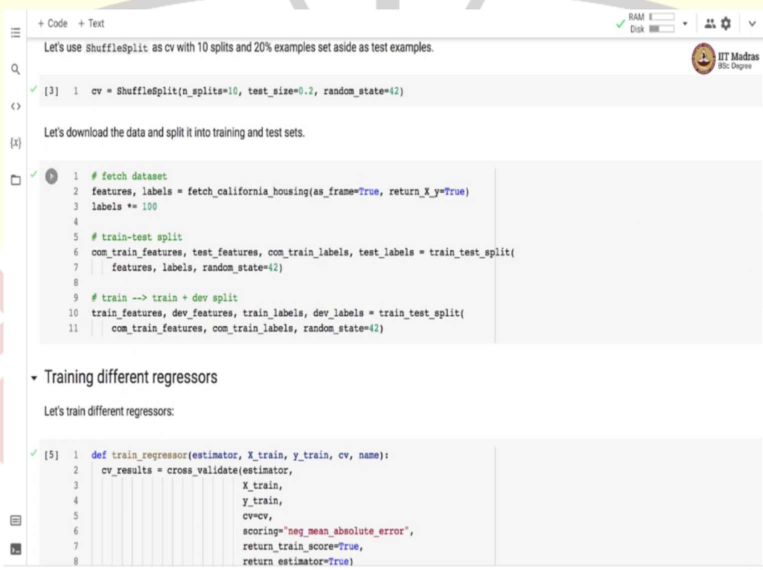
In this collab will make use of boosting regressors mainly AdaBoost, gradient boosting and XGBoost, AdaBoost and GradientBoostingRegressor are available in sklearn.ensemble module. Whereas, XGBoost regressor is not available directly in sklearn. But there is a separate library for XGBoost. And we will talk about XGBoost towards end of this collab.

(Refer Slide Time: 01:07)



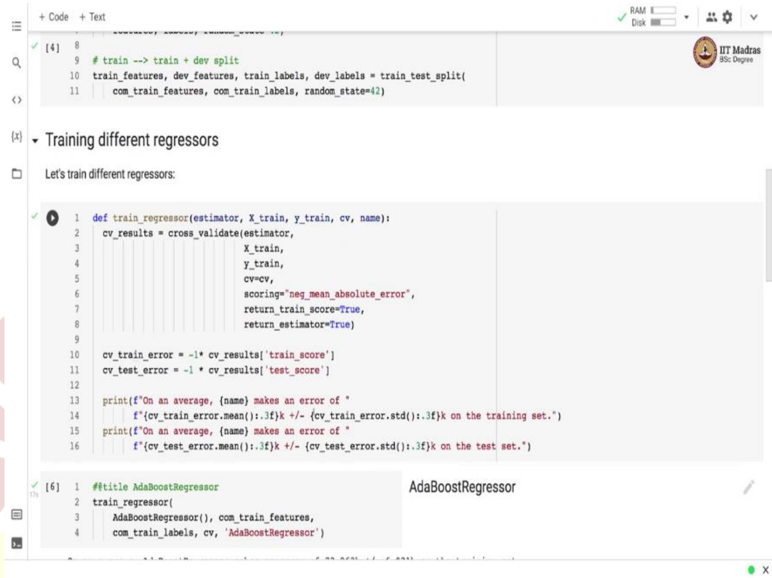
We start by assigning a random seed of 306. And this random seed helps us to get the reproducible result in the collab. We will use ShuffleSplit as a cross-validation with 10 splits and 20% examples set aside as test examples. Let us download the data and split it into training and test sets.

(Refer Slide Time: 01:32)



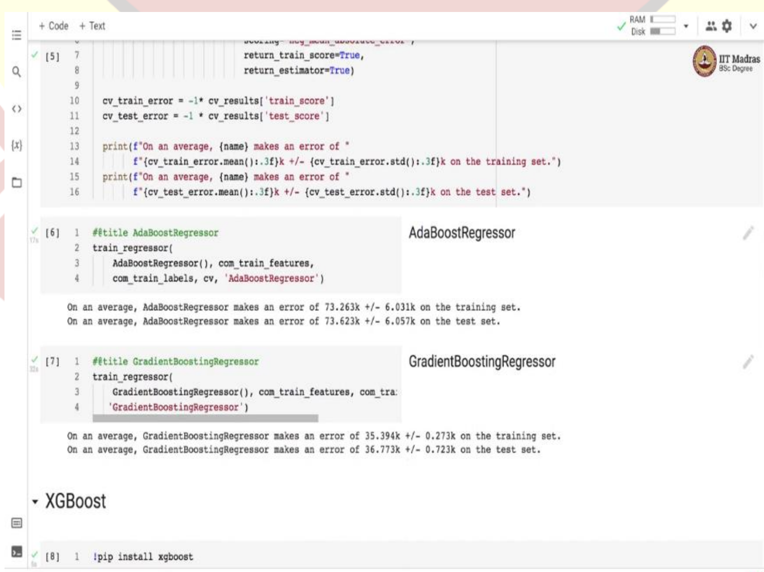
So, we fetch California Housing dataset and divide that into training and test as well as the training data is divided into train and dev.

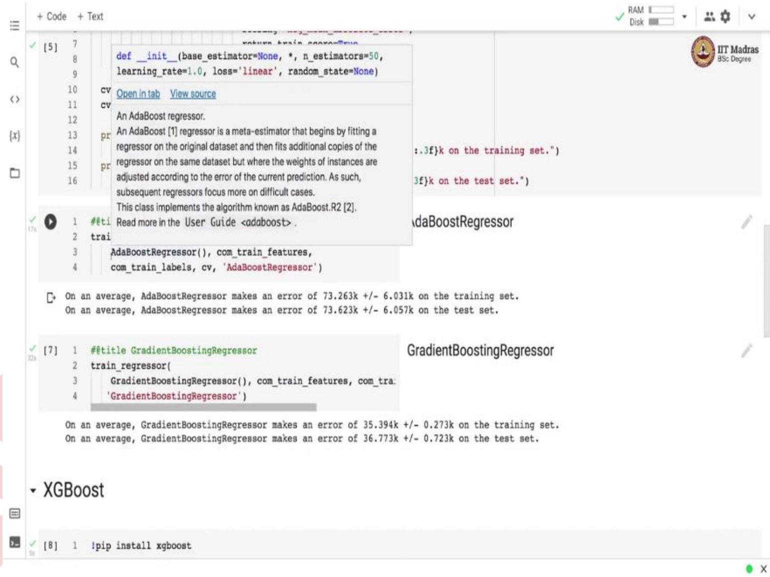
(Refer Slide Time: 01:43)



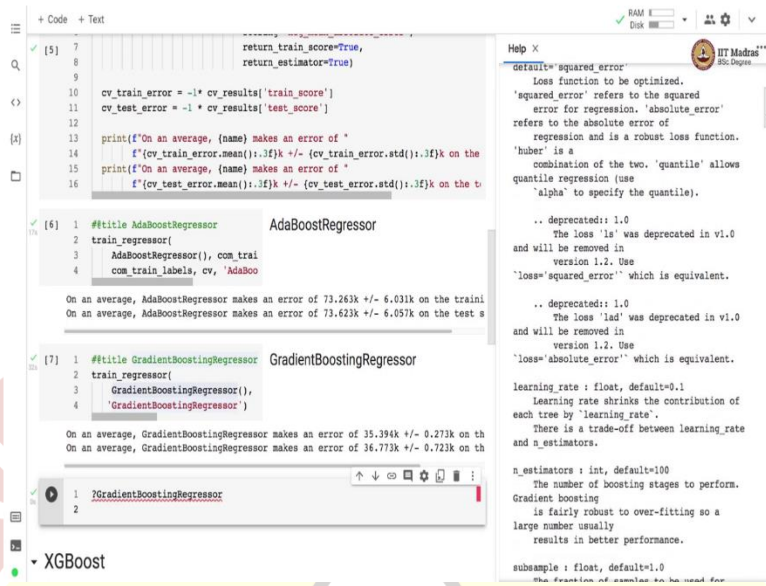
Then we trained different regressors and for training different regressors we define a function called train _regressor that takes the regression estimator, the feature matrix label vector cross-validation strategy and name of the regressor as inputs. It performs the cross-validation based training of the estimator using the negative mean absolute error as a scoring function. It uses cross-validation strategy as specified by **cv** parameter of the function argument. After the model is trained, we calculate the training error under test _error and print it out in these statements.

(Refer Slide Time: 02:28)

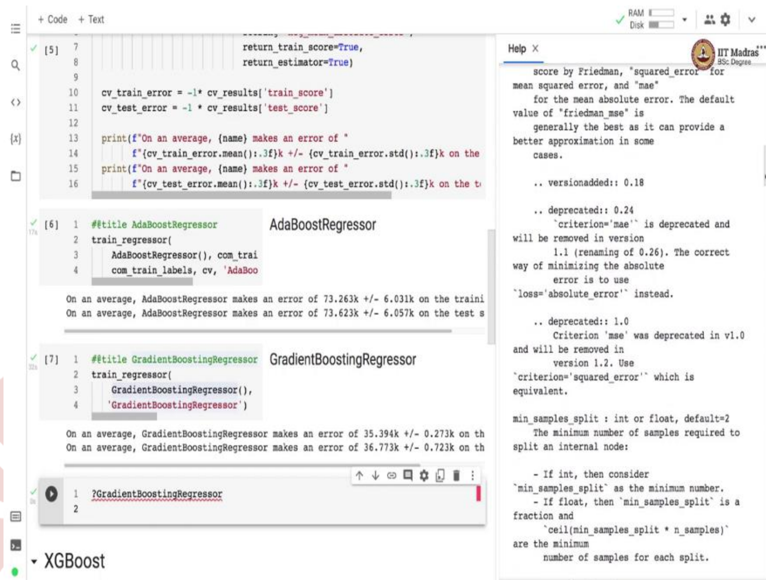




सिद्धिर्भवति कर्मजा



INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •



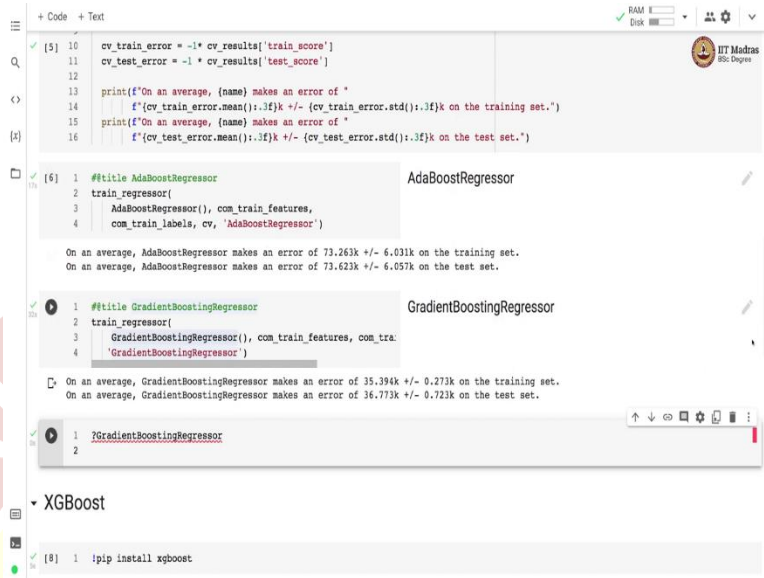
Let us run the AdaBoostRegressor and after running the AdaBoostRegressor we find that AdaBoostRegressor makes an error of 73.26k on the training set, and almost similar type of error on the test set the standard deviation is 6.03 for training and 6.05 for the test set. So, AdaBoostRegressor is defined using AdaBoostRegressor.

This is the instantiation of AdaBoostRegressor and we have made instantiation with default parameters and default parameters you can see here, number of estimators is = 50 and learning _rate is 1. And it uses linear loss. Next we define our GradientBoostingRegressor we instantiate GradientBoostingRegressor again with a bunch of default parameters.

And in case of GradientBoostingRegressor if you want to find out what are the default parameters, we can simply type the ? GradientBoostingRegressor and run this particular cell. And we can see the documentation for GradientBoostingRegressor. So, the default loss in case of GradientBoostingRegressor is squared error loss.

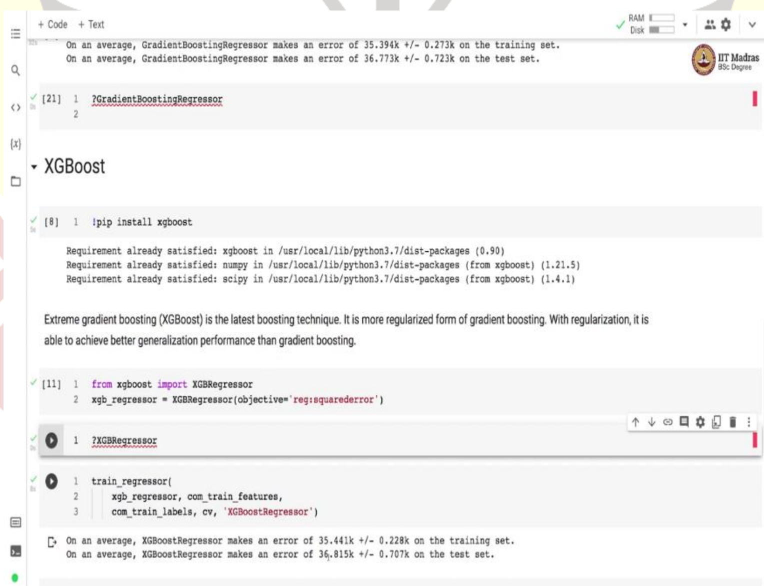
The default learning _rate is 0.1, and by default it trains 100 estimators. It uses friedman _mse as a criterion for making split in the decision tree. So, this is how you can access the documentation of the API that you are interested in.

(Refer Slide Time: 04:14)



So, you can see that after training the model with GradientBoostingRegressor it makes an error of 35.39k with the standard deviation of 0.273, whereas on the test set, it makes an error of 36.77 with standard deviation of 0.72. So, you can see that using gradient boosting has improved the performance of the regressor compared to the AdaBoostRegressor.

(Refer Slide Time: 04:42)



And finally, we trained XGBoost regressor model. So, we installed xgboost regressor and xgboost is Extreme gradient boosting it is the latest boosting technique. It is more regularized for gradient boosting it is very similar to gradient boosting except that it is more regularized form and with regularization, it is able to achieve better generalization performance than gradient boosting.

So, xgboost regressor is available in xgboost module. So, from xgboost we import XGBRegressor we instantiate XGBRegressor with objective function as squared error loss. And here you can again, check out the documentation of XGBRegressor. And when we train our model with xgboost regressor the performance is very similar to gradient boosting regressor.

So, remember that we have trained xgboost regressor with default parameters and with default parameters, it is giving performance very similar to gradient boosting. So, what we have not done is we have not tried to fine tune the parameters of these different regressors. So, another exercise that I am leaving you with is to basically perform hyper-parameter tuning of these regressors and see what kind of accuracy or what kind of errors you can get on the training set.

The idea is to reduce the error as much as possible, so that our regressor is able to make better predictions on the unseen data. So, in this video, we use boosting techniques for solving regression problem. We demonstrated the boosting regressor on California Housing dataset task. And we found that the gradient boosting and xgboost regressor in with the default parameters obtained very comparable performance, whereas AdaBoostRegressor was having worst of the performance among these three regressors.