INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

IIT Madras

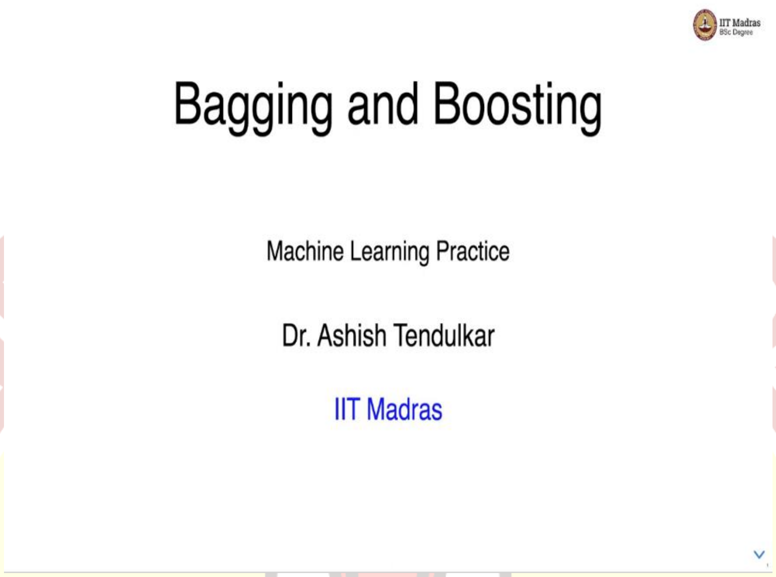
ONLINE DEGREE

Machine Learning Practice

Professor. Doctor Ashish Tendulkar
Indian Institute of Technology, Madras

Voting, Bagging and Random Forest

(Refer Slide Time: 00:11)



Bagging and Boosting

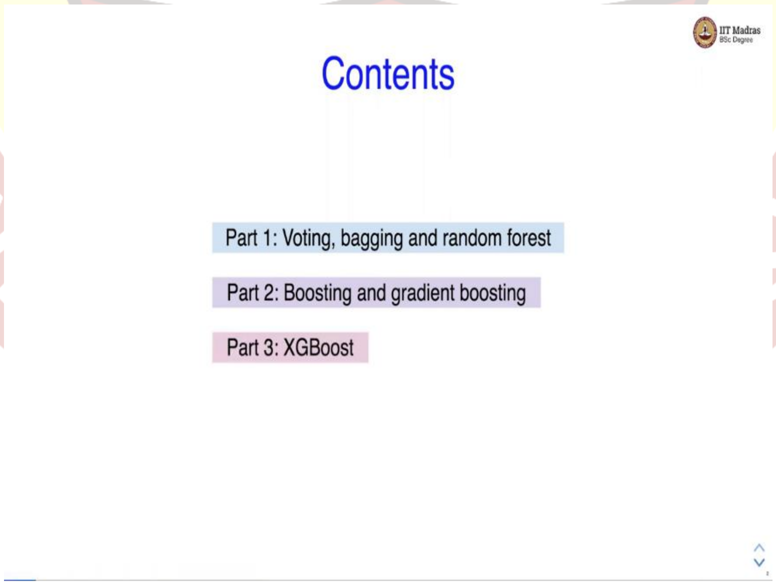
Machine Learning Practice

Dr. Ashish Tendulkar

IIT Madras

Namaste! Welcome to the next video of Machine Learning Practice Course. In this video we will discuss about bagging and boosting.

(Refer Slide Time: 00:22)



Contents

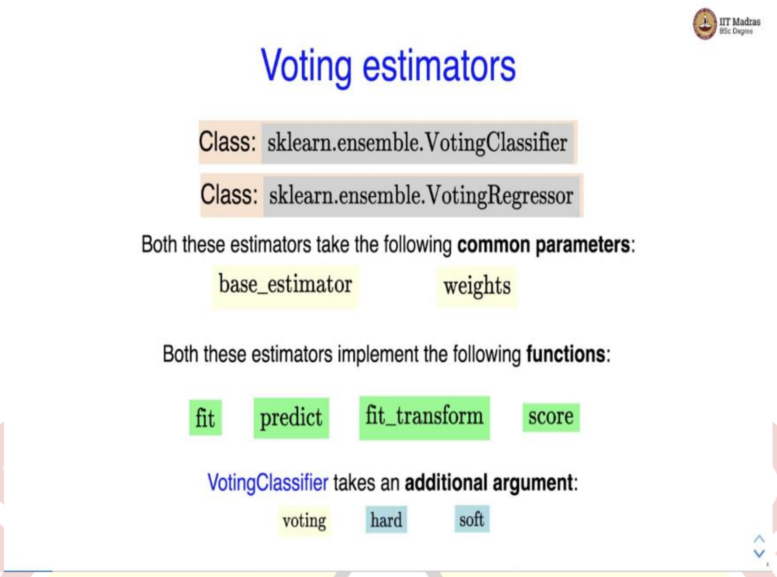
Part 1: Voting, bagging and random forest

Part 2: Boosting and gradient boosting

Part 3: XGBoost

We will be recording this slide deck in three parts, in the first part, we will record voting bagging and random forest. In the second part, we will record boosting and gradient boosting. And in the third part, we will record the XGBoost functionality in sklearn.

(Refer Slide Time: 00:39)



Voting estimators

Class: sklearn.ensemble.VotingClassifier

Class: sklearn.ensemble.VotingRegressor

Both these estimators take the following **common parameters**:

base_estimator weights

Both these estimators implement the following **functions**:

fit predict fit_transform score

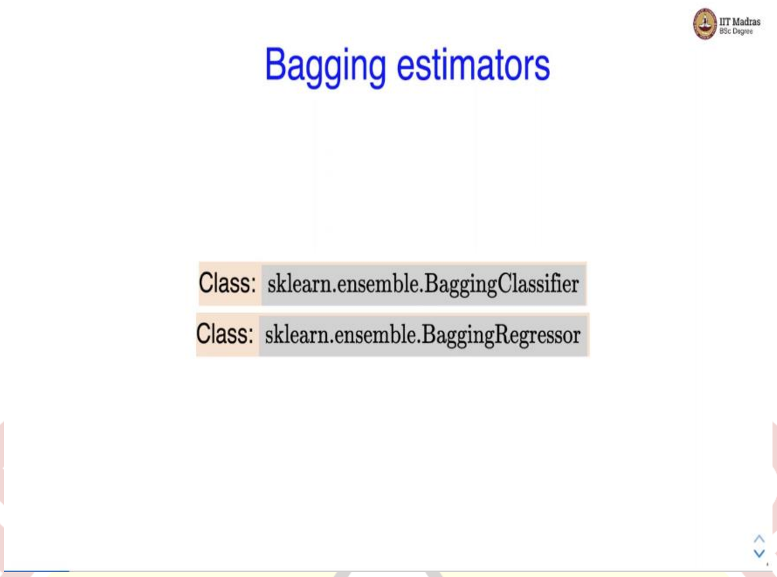
VotingClassifier takes an **additional argument**:

voting hard soft

Let us first discuss about voting estimators. There are two voting estimators, voting classifier, and voting regressor. They are implemented as part of a sklearn**.**ensemble module. Both these estimators take the following common parameters, base _estimator and weights. And implement the following functions: fit, predict, fit _transform and score.

Fit is used to train the estimators. Predict is used to obtain the prediction for new samples. Fit _transform performs transformation as well as the fitment. And a score is used to obtain the performance of the estimator. Voting classifier takes an additional argument which is voting, and there are two types of voting either a hard voting or a soft voting.

(Refer Slide Time: 01:38)

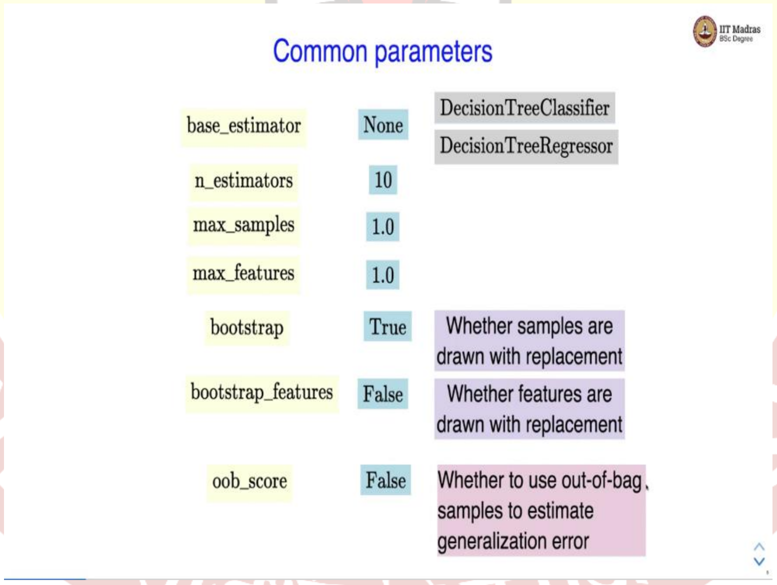


Bagging estimators

Class:	sklearn.ensemble.BaggingClassifier
Class:	sklearn.ensemble.BaggingRegressor

Then there are bagging estimators, two bagging estimators, again, bagging classifier and bagging regressor. They are also implemented as part of sklearn.ensemble module.

(Refer Slide Time: 01:48)



Common parameters

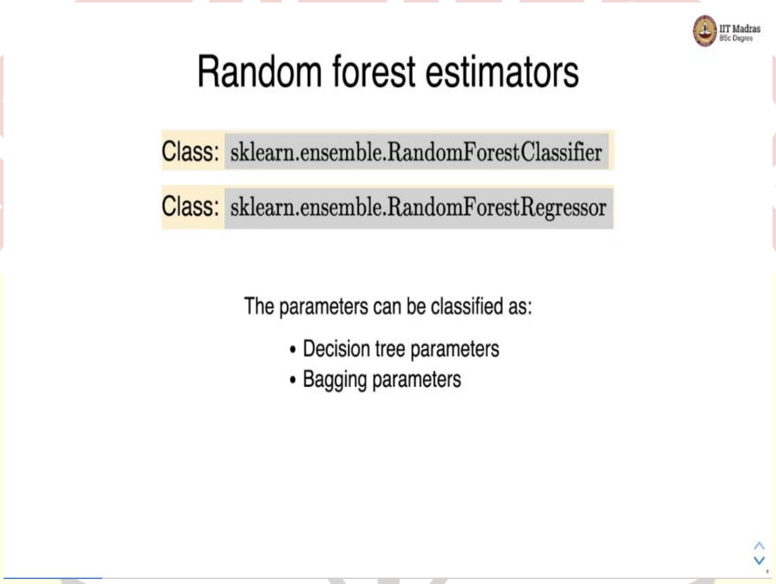
base_estimator	None	DecisionTreeClassifier DecisionTreeRegressor
n_estimators	10	
max_samples	1.0	
max_features	1.0	
bootstrap	True	Whether samples are drawn with replacement
bootstrap_features	False	Whether features are drawn with replacement
oob_score	False	Whether to use out-of-bag samples to estimate generalization error

They take the following common parameters base _estimator, by default it is none. In case of classification, we use DecisionTreeClassifier as the default base _estimator, whereas in case of bagging regressor we use DecisionTreeRegressor as a default base _estimator. Default number of estimators is 10. Max _samples can be integer or float, whenever it is integer, we take those many samples.

And when is float, we basically multiply the max _samples with the total number of samples and take the resulting samples as the max _samples for sampling. Max _features is also either an integer or float, and if it is integer, we select those many features. And if it is float, then we select number of features = the total number of features into max _features.

Bootstrap is true by default, and it specifies whether samples are drawn with replacement. Bootstrap _features and oob _score, both of them are false by default. And Bootstrap _features denotes or indicates whether features are drawn with the placement and oob _score denotes whether to use out of bag samples to estimate generalization error or test error.

(Refer Slide Time: 03:22)



Random forest estimators

Class: sklearn.ensemble.RandomForestClassifier

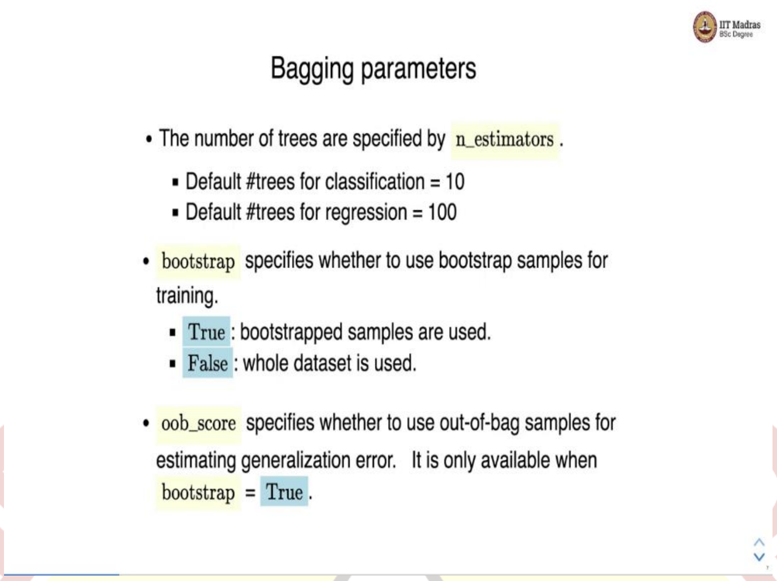
Class: sklearn.ensemble.RandomForestRegressor

The parameters can be classified as:

- Decision tree parameters
- Bagging parameters

Then we have random forests estimators, again two estimators one for classification, and second for regression, RandomForestClassifier and RandomForestRegressor respectively. They are also implemented as part of sklearn**.**ensemble module. The parameters of random forests can be classified as decision tree parameters and bagging parameters.

(Refer Slide Time: 03:43)

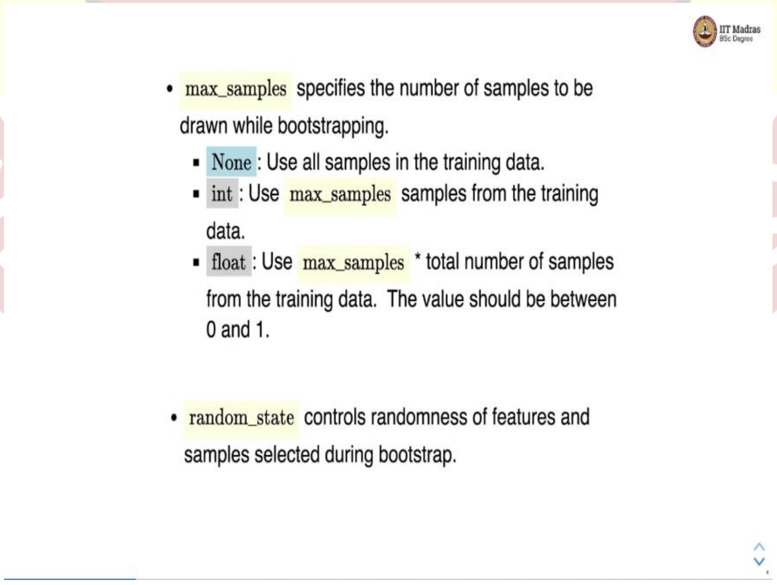


Bagging parameters

- The number of trees are specified by n_estimators.
  - Default #trees for classification = 10
  - Default #trees for regression = 100
- bootstrap specifies whether to use bootstrap samples for training.
  - True : bootstrapped samples are used.
  - False : whole dataset is used.
- oob_score specifies whether to use out-of-bag samples for estimating generalization error. It is only available when bootstrap = True.

Let us first look at bagging parameters. The number of trees are specified by n _estimators. Default number of trees used for classification instead, whereas default number of trees used for regression are 100. Then we have bootstrap flag that specifies whether to use Bootstrap samples for training when it is true bootstrap samples are used. And when Bootstrap is false whole dataset is used for training the estimators. oob _score specifies whether to use out of bag samples for estimating generalization error. It is only available when Bootstrap is = true.

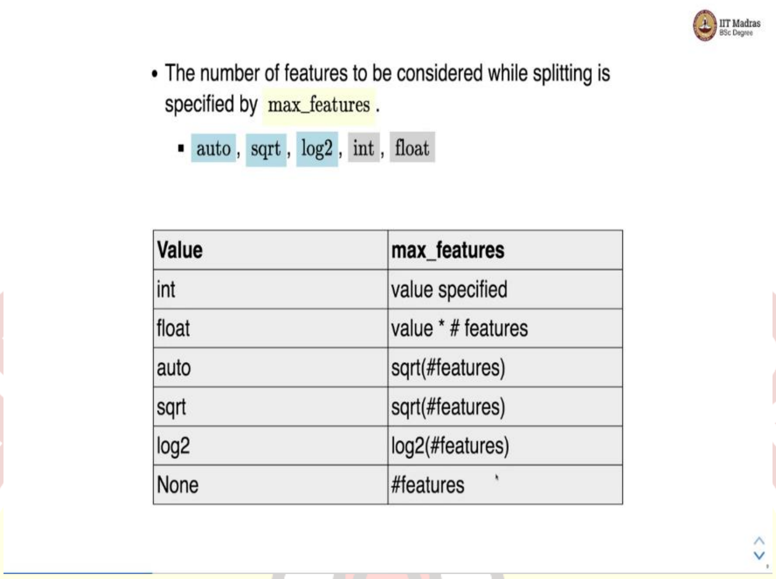
(Refer Slide Time: 04:26)



- max_samples specifies the number of samples to be drawn while bootstrapping.
  - None : Use all samples in the training data.
  - int : Use max_samples samples from the training data.
  - float : Use max_samples * total number of samples from the training data. The value should be between 0 and 1.
- random_state controls randomness of features and samples selected during bootstrap.

Then max _samples specifies the number of samples to be drawn while bootstrapping. If it is specified as none, we use all samples in the training data. If it is specified as integer, then we use max _samples from training data. If it is float, we use max _samples into total number of samples from the training data. If it is float, the value should always between 0 and 1. And random _state controls randomness of features and sample selected during bootstrap.

(Refer Slide Time: 05:04)



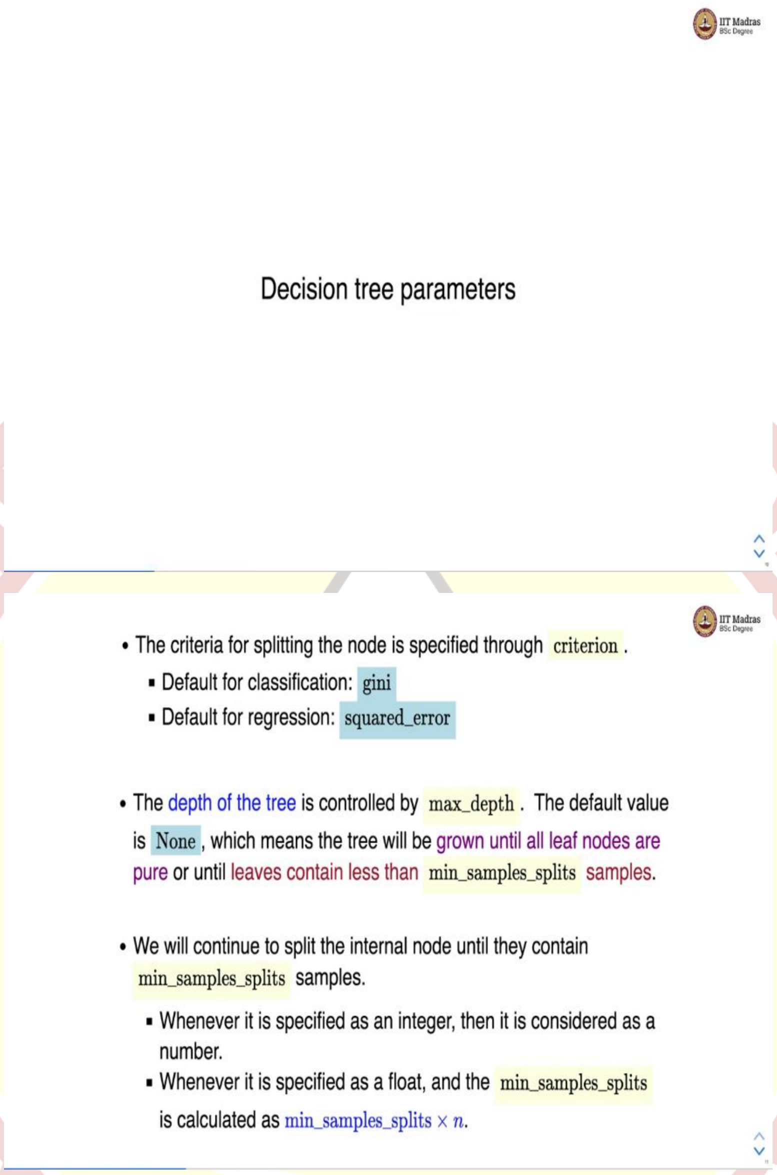
- The number of features to be considered while splitting is specified by max_features .
  - auto , sqrt , log2 , int , float

Value	max_features
int	value specified
float	value * # features
auto	sqrt(#features)
sqrt	sqrt(#features)
log2	log2(#features)
None	#features

The number of features to be considered while splitting the decision tree is specified by max _features. There could be five different values either auto, square root, log2, integer and float. Whenever we specify an integer value, we use the max _features as the value specified in the in the parameter.

If it is float, then the total number of features used are value into total number of features. If it is auto, then we use square root of number of features. If it is square root, sqrt, we use square root of number of features while making the splitting decision. If it is log2, we take number of features, log of number of features to the base 2. And if it is none, then we take all the features for deciding on the split in the decision tree.

(Refer Slide Time: 06:04)



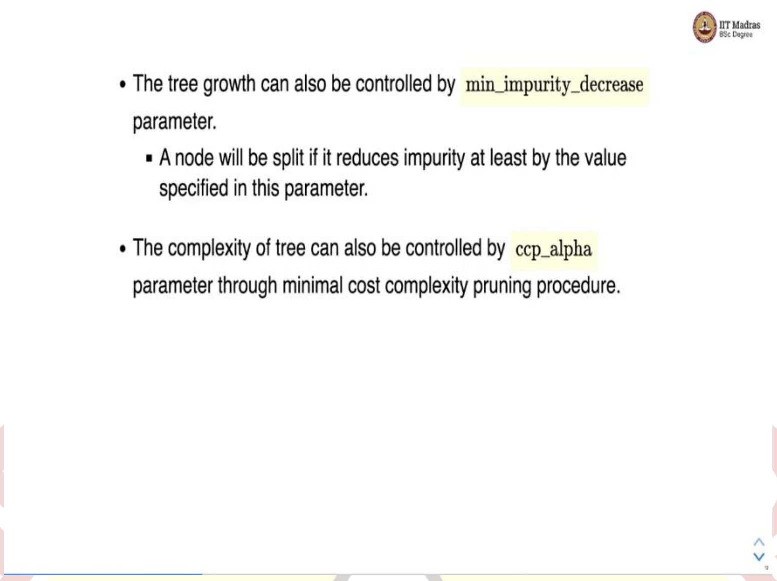
Decision tree parameters

- The criteria for splitting the node is specified through criterion.
  - Default for classification: gini
  - Default for regression: squared_error
- The depth of the tree is controlled by max_depth. The default value is None, which means the tree will be grown until all leaf nodes are pure or until leaves contain less than min_samples_splits samples.
- We will continue to split the internal node until they contain min_samples_splits samples.
  - Whenever it is specified as an integer, then it is considered as a number.
  - Whenever it is specified as a float, and the min_samples_splits is calculated as min_samples_splits $\times n$.

Let us look at some of the decision tree parameters. We are the criteria for splitting the node and it is specified to criterion default for classification is gini and default for regression is squared _error. The depth of the tree is controlled by a max _depth parameter, the default value is none which means the tree will be grown until all leaf nodes are pure or until leaves content less than min _samples _splits samples.

We will continue to split internal load until they contain min _samples _split samples. Whenever it is specified as an integer then it is considered as a number and whenever it is specified as a float then the min _samples _splits is calculated as the min _samples _splits into total number of samples.

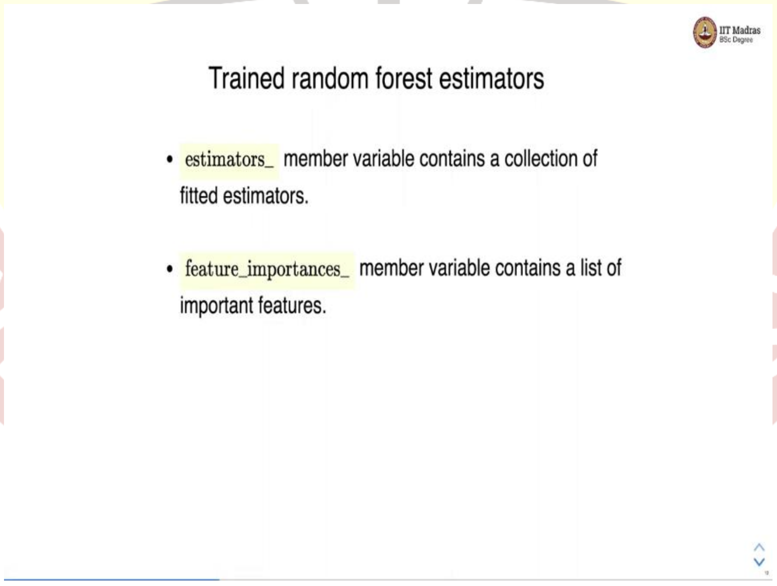
(Refer Slide Time: 07:00)



- The tree growth can also be controlled by min_impurity_decrease parameter.
  - A node will be split if it reduces impurity at least by the value specified in this parameter.
- The complexity of tree can also be controlled by ccp_alpha parameter through minimal cost complexity pruning procedure.

The tree growth can be controlled by min _impurity _decrease parameter. A node will be split if it reduces impurity, at least by the value specified in this parameter. The complexity of the tree can also be controlled by ccp _alpha parameter through minimal cost complexity pruning procedure. So, ccp _alpha is a parameter used in minimal cost complexity pruning.

(Refer Slide Time: 07:30)

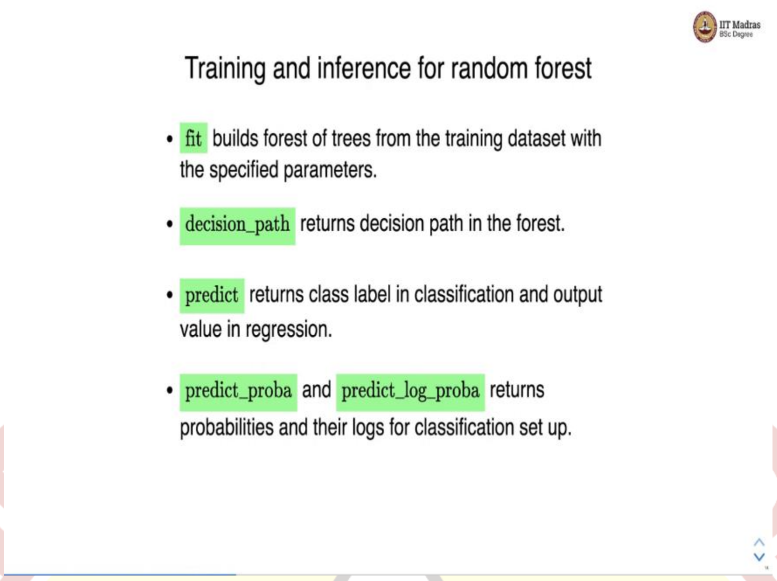


Trained random forest estimators

- estimators_ member variable contains a collection of fitted estimators.
- feature_importances_ member variable contains a list of important features.

Estimator _member variable contains a collection of fitted estimators. Feature _importances _member variable contains a list of important features.

(Refer Slide Time: 07:42)



Training and inference for random forest

- fit builds forest of trees from the training dataset with the specified parameters.
- decision_path returns decision path in the forest.
- predict returns class label in classification and output value in regression.
- predict_proba and predict_log_proba returns probabilities and their logs for classification set up.

We use fit method for building forest of trees from training data with the specified parameters. Decision _path returns decision path in the forest. Predict returns class labor in classification, and outputs a value in regression. And predict _proba and predict _log _proba are functions available in classification and the return probabilities and the log of probabilities.

In this video, we studied a sklearn functionality for Voting, Bagging and Random Forest. We looked at how to set up classification and regression with Voting, Bagging and Random Forest.