# DecisionForest

## Classification

**Classification** **(nClasses, seed=777, nTrees = 100, observationsPerTreeFraction = 1,featuresPerNode=0,maxTreeDepth=0,minObservationsInLeafNodes=1,impurityThreshold=0,varImportance=None,resultsToCompute=0):**

Constructor to set decision forest classification training parameters. The best split for classification uses Gini impurity criteria

*parameters:*

**seed:** default: 777

The seed for random number generator, which is used to choose the bootstrap set, split features in every split node in a tree, and generate permutation required in computations of MDA variable importance.

**nTrees:** default**:** 100

The number of trees in the forest.

**observationsPerTreeFraction**: default: 1

Fraction of the training set S used to form the bootstrap set for a single tree training, 0 < observationsPerTreeFraction <= 1. The observations are sampled randomly with replacement.

**featuresPerNode**:default: 0

The number of features tried as possible splits per node. If the parameter is set to 0, the library uses the square root of the number of features for classification and (the number of features)/3 for regression.

**maxTreeDepth** : default: 0

Maximal tree depth. Default is 0 (unlimited).

**minObservationsInLeafNodes**: default: 1

Minimum number of observations in the leaf node.

**impurityThreshold**: default: 0

The threshold value used as stopping criteria: if the impurity value in the node is smaller than the threshold, the node is not split anymore.

**varImportance**: default: none

The variable importance computation mode.

Possible values:

* + None or 0– variable importance is not calculated
  + ‘MDI’ or 1- also known as the Gini importance or Mean Decrease Gini
  + ‘MDA\_Raw’ or 2 - Mean Decrease of Accuracy (permutation importance)
  + ‘MDA\_Scaled’ or 3 - the MDA\_Raw value scaled by its standard deviation

**resultsToCompute**: default: 0

Possible values

* None or 0- no OOB error calculated
* 1 - OOB error calculated
* 2 - OOB error per observation calculated
* 3 - OOB error and OOB error per observation calculated

## Methods:

1. **training(trainData, trainDependentVariables):**

*parameters:*

train data feature values(type nT), train data target values(type nT)

*returns*:

training results object with attributes to get variance scores, OOB error and OOB error per observation numeric tables

ex: trainingResult.getTable (training.outOfBagErrorPerObservation)

rainingResult.getTable (training.outOfBagError)

trainingResult.getTable (training.varImportance)

1. **predict(trainingResult, testData):**

*parameters:*

training result object, test data feature values(type nT)

*returns*:

predicted values of type nT

1. **compress(arrayData):**

*parameters:*

serialized numpy array

*returns*:

Compressed numpy array

1. **decompress(arrayData):**

*parameters:*

deserialized numpy array

*returns*:

decompressed numpy array

1. **serialize(data, fileName=None, useCompression=False):**

*parameters:*

Method 1: data(type nT/model)

-Returns serialized numpy array

Method 2: data(type nT/model), fileName(.npy file to save serialized array to disk)

- Saves serialized numpy array as "fileName" argument

Method 3: data(type nT/model), useCompression = True

-Returns compressed numpy array

Method 4: data(type nT/model), fileName(.npy file to save serialized array to disk), useCompression = True

-Saves compressed numpy array as "fileName" argument

1. **deserialize(serialObjectDict=None, fileName=None, useCompression=False):**

*parameters:*

serialized/ compressed numpy array or serialized/ compressed .npy file

*returns*:

deserialized/ decompressed numeric table/model

1. **qualityMetrics(predictResults, testGroundTruth)**

*parameters:*

prediction values(type nT), test data actual target values(type nT)

*returns* :

qualityMetrics object having get method to retrieve one of the following metrics

BinarySVM: Accuracy, Precision, Recall, F1-score, Specificity, AUC

MultiSVM: Accuracy, error rate, Micro precision,Micro recall,Micro F-score,Macro precision,Macro recall,Macro F-score

1. **predictWithQualityMetrics(trainingResult, testData,testGroundTruth):**

*parameters*:

training result object, test data feature values of type nT, test data actual target values(type nT).

*returns*:

predicted values(type nT), quality metrics object for binary classifier.

1. **printAllQualityMetrics(qualityMetrics):**

*parameters*:

quality metrics object for respective classifier

Prints Accuracy, Precision, Recall, F1-score, Specificity, AUC for BinarySVM

Prints Accuracy, error rate, Micro precision,Micro recall,Micro F-score,Macro precision,Macro recall,Macro F-score for MultiSVM

# Regression:

**Regression** **(nClasses, seed=777, nTrees = 100, observationsPerTreeFraction = 1,featuresPerNode=0,maxTreeDepth=0,minObservationsInLeafNodes=5,impurityThreshold=0,varImportance=None,resultsToCompute=0):**

Constructor to set decision forest regression training parameters. The best split for regression uses variance

*parameters:*

**seed:** default: 777

The seed for random number generator, which is used to choose the bootstrap set, split features in every split node in a tree, and generate permutation required in computations of MDA variable importance.

**nTrees:** default**:** 100

The number of trees in the forest.

**observationsPerTreeFraction**: default: 1

Fraction of the training set S used to form the bootstrap set for a single tree training, 0 < observationsPerTreeFraction <= 1. The observations are sampled randomly with replacement.

**featuresPerNode**:default: 0

The number of features tried as possible splits per node. If the parameter is set to 0, the library uses the square root of the number of features for classification and (the number of features)/3 for regression.

**maxTreeDepth** : default: 0

Maximal tree depth. Default is 0 (unlimited).

**minObservationsInLeafNodes**: default: 5

Minimum number of observations in the leaf node.

**impurityThreshold**: default: 0

The threshold value used as stopping criteria: if the impurity value in the node is smaller than the threshold, the node is not split anymore.

**varImportance**: default: none

The variable importance computation mode.

Possible values:

* + None or 0– variable importance is not calculated
  + ‘MDI’ or 1- also known as the Gini importance or Mean Decrease Gini
  + ‘MDA\_Raw’ or 2 - Mean Decrease of Accuracy (permutation importance)
  + ‘MDA\_Scaled’ or 3 - the MDA\_Raw value scaled by its standard deviation

**resultsToCompute**: default: 0

Possible values

* None or 0- no OOB error calculated
* 1 - OOB error calculated
* 2 - OOB error per observation calculated
* 3 - OOB error and OOB error per observation calculated

## Methods

1. **predict(trainingResult, testData)**

*parameters:*

training result object, test data feature values(type nT)

*returns*:

predicted values of type nT

1. **compress(arrayData)**

*parameters:*

serialized numpy array

*returns*:

Compressed numpy array

1. **decompress(arrayData)**

*parameters:*

deserialized numpy array

*returns*:

decompressed numpy array

1. **serialize(data, fileName=None, useCompression=False)**

*parameters:*

Method 1: data(type nT/model)

-Returns serialized numpy array

Method 2: data(type nT/model), fileName(.npy file to save serialized array to disk)

- Saves serialized numpy array as "fileName" argument

Method 3: data(type nT/model), useCompression = True

-Returns compressed numpy array

Method 4: data(type nT/model), fileName(.npy file to save serialized array to disk), useCompression = True

-Saves compressed numpy array as "fileName" argument

1. **deserialize(serialObjectDict=None, fileName=None, useCompression=False)**

*parameters:*

serialized/ compressed numpy array or serialized/ compressed .npy file

*returns*:

deserialized/ decompressed numeric table/model