**Random Forest in Apache Ignite**

Random Forest is an ensemble learning method for finding solutions to classification and regression problems. Random forest training builds a model composition (ensemble) of one type and uses an aggregation algorithm from several model answers. Each model is trained on a part of the training dataset. The part is defined according to bagging and feature subspace methods. More information about these concepts may be found here: [1](https://en.wikipedia.org/wiki/Random_forest), [2](https://en.wikipedia.org/wiki/Bootstrap_aggregating) and [3](https://en.wikipedia.org/wiki/Random_subspace_method).

There are several implementations of the aggregation algorithm in Apache Ignite:

* MeanValuePredictionsAggregator - computes answer of random forest as the mean value of predictions from all models in the given composition. Often such computations are used for regression tasks
* OnMajorityPredictionsAggegator - gets [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of predictions from all models in the given composition. Such computations can be useful for classification tasks. NOTE: This aggregator supports multi-classification tasks

**Model**

The random forest algorithm is implemented in Ignite ML as a special case of model composition with specific aggregators for different problems (MeanValuePredictionsAggregator for regression, OnMajorityPredictionsAggegator for classification).

An example of model usage would be as follows:

ModelsComposition randomForest = ….

double prediction = randomForest.apply(featuresVector);

**Trainer**

The random forest training algorithm is implemented with RandomForestRegressionTrainer and RandomForestClassifierTrainer trainers with the following parameters:

* meta - features meta, list of feature type description such as:
  + featureId - index in features vector
  + isCategoricalFeature-flag having a true value if the feature is categorical
  + feature name

This meta-information is important for the random forest training algorithm because it builds feature histograms and categorical features should be represented in histograms for all feature values.

* featuresCountSelectionStrgy - sets strategy defining count of random features for learning one tree. There are several strategies: SQRT, LOG2, ALL and ONE\_THIRD strategies implemented in the FeaturesCountSelectionStrategies class
* maxDepth - sets maximum tree depth
* minInpurityDelta - node in decision tree is split into two nodes if impurity values on these nodes is less than the not split node on minImpurityDecrease value
* subSampleSize - value lies in the [0; MAX\_DOUBLE]-interval. This parameter defines the count of sample repetitions in uniformly sampling with replacement
* seed - seed value used in random generators

Random forest training may be used as follows:

RandomForestClassifierTrainer trainer = new RandomForestClassifierTrainer(featuresMeta)

.withCountOfTrees(101) .withFeaturesCountSelectionStrgy(FeaturesCountSelectionStrategies.ONE\_THIRD)

.withMaxDepth(4)

.withMinImpurityDelta(0.)

.withSubSampleSize(0.3)

.withSeed(0);

ModelsComposition rf = trainer.fit(  
 datasetBuilder,  
 featureExtractor,  
 labelExtractor  
);

**Example**

To see how Random Forest Classifier can be used in practice, try this [example](https://github.com/apache/ignite/blob/master/examples/src/main/java/org/apache/ignite/examples/ml/tree/randomforest/RandomForestClassificationExample.java) that is available on GitHub and delivered with every Apache Ignite distribution. In this example Wine recognition dataset was used. Description of this dataset and data are available by [this link](https://archive.ics.uci.edu/ml/datasets/wine).