Random Forests

Lesson Objectives

- After completing this lesson, you should be able to:
  - Understand the Pipelines API for Random Forests and Gradient-Boosted Trees
  - Describe default's Input and Output columns
  - Perform classification and regression with RFs
  - Understand and use RF’s parameters

Ensemble Method

- Learning algorithm which creates a model composed of a set of other base models
- ‘Random Forests’ and ‘Gradient-Boosted Trees’ are ensemble algorithms based on decision trees
- Among top performers for classification and regression problems

Random Forests (RFs)

- Ensembles of Decision Trees
- One of the most successful machine learning models for classification and regression
- Combine many decision trees in order to reduce the risk of overfitting
- Supports binary and multiclass classification
- Supports regression
- Supports continuous and categorical features

RF: Basic Algorithm

- RF trains a bunch of decision trees separately
- RF Injects randomness into the training process
  - –bootstrapping: subsamples the original dataset on each iteration to get a different training set
  - –considers different random subsets of features to split on at each tree node
- Combined predictions from several trees reduces the variance of the predictions and improves the performance on test data
  - –classification: majority vote - each tree's prediction is counted as a vote for one class and the predicted label is the class with largest number of votes
  - –regression: average - each tree predicts a real value and the predicted label is equal to the average of all predictions

Random Forest Parameters (1)

- Most important parameters in Spark.ml implementation
- Parameters that CAN be tuned to improve performance:
  - **numTrees**: number of trees in the forest. If it increases:
    - the variance of predictions decreases, improving test-time accuracy
    - training time increases roughly linearly
  - **maxDepth**: maximum depth of each tree in the forest. If it increases:
    - model gets more expressive and powerful
    - takes longer to train
    - is more prone to overfitting

Random Forest Parameters (2)

- Parameters that DO NOT require tuning but CAN be tuned to speed up training:
  - **subsamplingRate**: specifies the fraction of size of the original dataset to be used for training each tree in the forest
    - default = 1.0, but decreasing it can speed up training
  - **featureSubsetStrategy**: specifies the fraction of total number of features to use as candidates for splitting at each tree node
    - decreasing it will speed up training
    - if too low, can also impact performance

Inputs and Outputs

Param name	Type(s)	Default	Description
labelCol	Double	"label"	Label to predict
featuresCol	Vector	"features"	Feature vector

Param name	Type(s)	Default	Description	Notes
predictionCol	Double	"prediction"	Predicted label	
rawPredictionCol	Vector	"rawPrediction"	Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction	Classification only
probabilityCol	Vector	"probability"	Vector of length # classes equal to rawPrediction normalized to a multinomial distribution	Classification only

RF Classification (1)

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import RandomForestClassificationModel

labelIndexer = StringIndexer().setInputCol("label") \
                              .setOutputCol("indexedLabel").fit(data)

labelConverter = IndexToString().setInputCol("prediction") \
                              .setOutputCol("predictedLabel") \
                              .setLabels(labelIndexer.labels)

featureIndexer = VectorIndexer().setInputCol("features") \
                              .setOutputCol("indexedFeatures") \
                              .setMaxCategories(4).fit(data)

rfC = RandomForestClassifier().setLabelCol("indexedLabel") \
                              .setFeaturesCol("indexedFeatures") \
                              .setNumTrees(3)
```

RF Classification (2)

```
trainingData, testData = data.randomSplit([0.7, 0.3])

pipelineRFC = Pipeline().setStages([labelIndexer, featureIndexer, rfC, labelConverter])

modelRFC = pipelineRFC.fit(trainingData)

predictionsRFC = modelRFC.transform(testData)
```

RF Classification (3)

```
predictionsRFC.select("predictedLabel", "label", "features").show(5)
```

```
+--------------+-----+--------------------+
|predictedLabel|label|            features|
+--------------+-----+--------------------+
|           1.0|  1.0|(692,[123,124,125...|
|           1.0|  1.0|(692,[123,124,125...|
|           0.0|  0.0|(692,[124,125,126...|
|           0.0|  0.0|(692,[126,127,128...|
|           0.0|  0.0|(692,[151,152,153...|
+--------------+-----+--------------------+
only showing top 5 rows
```

RF Classification (4)

```
rfModelC = modelRFC.stages[2]
rfModelC.featureImportances
```

```
SparseVector(692, {319: 0.0317, 455: 0.2749, 463: 0.2692, 490: 0.3017, 517: 0.0641, 540: 0.0585})
```

```
print rfModelC.toDebugString
```

```
RandomForestClassificationModel (uid=rfc_1ab9a4d10b3e) with 3 trees
  Tree 0 (weight 1.0):
    If (feature 463 <= 0.0)
     If (feature 517 <= 116.0)
      Predict: 1.0
     Else (feature 517 > 116.0)
      Predict: 0.0
    Else (feature 463 > 0.0)
     Predict: 0.0
  Tree 1 (weight 1.0):
    If (feature 490 <= 31.0)
     If (feature 319 <= 253.0)
      Predict: 1.0
     Else (feature 319 > 253.0)
      Predict: 0.0
    Else (feature 490 > 31.0)
     Predict: 0.0
```

RF for regression

```
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.regression import RandomForestRegressionModel

rfR = RandomForestRegressor().setLabelCol("label").setFeaturesCol("indexedFeatures")

pipelineRFR = Pipeline().setStages([featureIndexer, rfR])

modelRFR = pipelineRFR.fit(trainingData)

predictionsRFR = modelRFR.transform(testData)
```

RF for regression

```
predictionsRFR = modelRFR.transform(testData)

predictionsRFR.select("prediction", "label", "features").show(5)
```

```
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       1.0|  1.0|(692,[123,124,125...|
|       1.0|  1.0|(692,[123,124,125...|
|       0.1|  0.0|(692,[124,125,126...|
|       0.0|  0.0|(692,[126,127,128...|
|      0.05|  0.0|(692,[151,152,153...|
+----------+-----+--------------------+
only showing top 5 rows
```

Lesson Summary

- Having completed this lesson, you should be able to:
  - Understand how to run a random forest in Spark
  - Grasp most of the parameters and their effects
  - Understand inputs and outputs
  - Understand how to use RF for regression and categorization