AAS_Ch4 & DM_6 My Extensions

Alexander Spivey:

Different Hyperparameters for Decision Tree

```
val paramGrid = new ParamGridBuilder().
    addGrid(classifier.maxMemoryInMB, Seq(1024)).
    addGrid(classifier.impurity, Seq("entropy")). //use both impurity measures
    addGrid(classifier.maxDepth, Seq(10, 20, 30)).  //depth of 10 to 30
    addGrid(classifier.maxBins, Seq(20, 40, 60, 80, 100)).  //20 to 100 values for rules
    addGrid(classifier.minInfoGain, Seq(0.0, 0.01, 0.02)).  //ranges from no changes to impurity to improve impurity by 0.02
    //addGrid(classifier.minInstancesPerNode, Seq(500, 1000, 2000))
    //the first time I ran this, I tried to use 500, 1000, 2000 for minInstance, but ended up drastically dropping the success of the function
    build()
```

Results

```
//-How well did the data do on the cv and test-
//print("CV Results")
validatorModel.validationMetrics.max
//0.9408314546635589

//print("UnencTestData")
multiclassEval.evaluate(bestModel.transform(unencTestData))
//0.9422549869904596

//print("UnencTrainingData")
multiclassEval.evaluate(bestModel.transform(unencTrainData))
//0.9994898368624394
//Does this even fall into the category of overfitting?
```

```
0.9402965114057544
{
    dtc_e253ce313ef8-impurity: entropy,
    dtc_e253ce313ef8-maxBins: 60,
    dtc_e253ce313ef8-maxDepth: 30,
    dtc_e253ce313ef8-maxMemoryInMB: 1024,
    dtc_e253ce313ef8-minInfoGain: 0.0
}

0.9396278323334989
{
    dtc_e253ce313ef8-impurity: entropy,
    dtc_e253ce313ef8-maxBins: 100,
    dtc_e253ce313ef8-maxDepth: 30,
    dtc_e253ce313ef8-maxMemoryInMB: 1024,
    dtc_e253ce313ef8-minInfoGain: 0.0
}

0.9387489969813916
{
    dtc_e253ce313ef8-impurity: entropy,
    dtc_e253ce313ef8-maxBins: 100,
    dtc_e253ce313ef8-maxDepth: 30,
    dtc_e253ce313ef8-maxMemoryInMB: 1024,
    dtc_e253ce313ef8-minInfoGain: 0.01
}

0.9370104313935271
{
    dtc_e253ce313ef8-impurity: entropy,
    dtc_e253ce313ef8-maxBins: 40,
    dtc_e253ce313ef8-maxDepth: 30,
    dtc_e253ce313ef8-maxMemoryInMB: 1024,
    dtc_e253ce313ef8-minInfoGain: 0.0
}

0.936876695579076
{
    dtc_e253ce313ef8-impurity: entropy,
    dtc_e253ce313ef8-maxBins: 60,
    dtc_e253ce313ef8-maxDepth: 30,
    dtc_e253ce313ef8-maxMemoryInMB: 1024,
    dtc_e253ce313ef8-minInfoGain: 0.01
}
```

Different Hyperparameters for Decision Tree v2

```
val paramGrid = new ParamGridBuilder().
    addGrid(classifier.impurity, Seq("entropy")). //use both impurity measures
    addGrid(classifier.maxDepth, Seq(30)).  //use max max depth
    addGrid(classifier.maxBins, Seq(40, 60, 80, 100)).  //20 to 100 values for rules
    addGrid(classifier.minInfoGain, Seq(0.0)).  //0.0 since it doesn't seem to affect the data
    addGrid(classifier.minInstancesPerNode, Seq(1, 2, 3)).
    build()
```

Results

0.9306456536670992
{
dtc_fd915673ac55-impurity: entropy,
dtc_fd915673ac55-maxBins: 100,
dtc_fd915673ac55-maxDepth: 30,
dtc_fd915673ac55-minInfoGain: 0.0,
dtc_fd915673ac55-minInstancesPerNode: 5
}

0.9301114248645348
{
dtc_fd915673ac55-impurity: entropy,
dtc_fd915673ac55-maxBins: 80,
dtc_fd915673ac55-maxDepth: 30,
dtc_fd915673ac55-minInfoGain: 0.0,
dtc_fd915673ac55-minInstancesPerNode: 5
}

0.9274593604518049
{
dtc_fd915673ac55-impurity: entropy,
dtc_fd915673ac55-maxBins: 40,
dtc_fd915673ac55-maxDepth: 30,
dtc_fd915673ac55-minInfoGain: 0.0,
dtc_fd915673ac55-minInstancesPerNode: 5
} ...

CV Results
res103: Double = 0.9308936884682897

UnencTest
res105: Double = 0.9325177721029747

UnencTrain
res107: Double = 0.9705002905998593

Speeding up the program with maxMemoryInMB

```
//ITERATION 0
print("ITERATION 0\n")
val paramGrid = new ParamGridBuilder().
    addGrid(classifier.maxMemoryInMB, Seq(128)).
    addGrid(classifier.impurity, Seq("entropy")). //use both impurity measures
    addGrid(classifier.maxDepth, Seq(30)).  //depth of 1 to 20
    addGrid(classifier.maxBins, Seq(20, 40, 60, 80, 100)).  //40 to 300 values for rules
    addGrid(classifier.minInfoGain, Seq(0.0)).  //ranges from no changes to impurity to improve impurity by 0.05
    //addGrid(classifier.minInstancesPerNode, Seq(500, 1000, 2000))
    //the first time I ran this, I tried to use 500, 1000, 2000 for minInstance, but ended up drastically dropping the success of the function
    build()
val multiclassEval = new MulticlassClassificationEvaluator().
    setLabelCol("Cover_Type").
    setPredictionCol("prediction").
    setMetricName("accuracy")   //kinda self explanotary what happening here

val validator = new TrainValidationSplit().
    setSeed(Random.nextLong()).
    setEstimator(pipeline).
    setEvaluator(multiclassEval).
    setEstimatorParamMaps(paramGrid). //HYPER PARAMETERS CAN STILL OVER FIT
    setTrainRatio(0.9)  //take another 10 percent and set it aside
//the left out 10% is used as a crossvalidation set (evaluate parameters that fit to training set)
//the original 10% left out to evaluate hyperparameters that fit the CV^^ (examples that arent in CV but has not been trained on [real world data])

val start = Calendar.getInstance()
val sh = start.get(Calendar.HOUR)
val sm = start.get(Calendar.MINUTE)
val ss = start.get(Calendar.SECOND)
printf("Start Time: %02d:%02d:%02d\n", sh, sm, ss);

val validatorModel = validator.fit(unencTrainData)  //returns best overall pipeline

val end = Calendar.getInstance()
val eh = end.get(Calendar.HOUR)
val em = end.get(Calendar.MINUTE)
val es = end.get(Calendar.SECOND)
printf("End Time: %02d:%02d:%02d\n", eh, em, es);

val dh = eh-sh
val dm = em-sm
val ds = es-ss
printf("Total Time: %02d:%02d:%02d\n", dh, dm, ds);
```

Results

ITERATION 0
Start Time: 08:34:15
End Time: 08:36:24
Total Time: 00:02:09

ITERATION 1
Start Time: 08:03:28
End Time: 08:05:43
Total Time: 00:02:15

ITERATION 2
Start Time: 08:05:44
End Time: 08:07:56
Total Time: 00:02:12

ITERATION 3
Start Time: 08:07:57
End Time: 08:10:04
Total Time: 00:02:07

ITERATION 4
Start Time: 08:10:05
End Time: 08:12:16
Total Time: 00:02:11

ITERATION 5
Start Time: 08:12:17
End Time: 08:14:28
Total Time: 00:02:11

Weka Random Decision Forest

```
=== Evaluation result ===

Scheme: RandomForest
Options: -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation: covtype-weka.filters.unsupervised.attribute.NumericToNominal-R11-55

=== Summary ===

Correctly Classified Instances      187798               95.0664 %
Incorrectly Classified Instances      9746                4.9336 %
Kappa statistic                          0.9206
Mean absolute error                      0.0402
Root mean squared error                  0.114
Relative absolute error                 22.5566 %
Root relative squared error             38.192  %
Total Number of Instances           197544
```

66%

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.938    0.022    0.961      0.938   0.950      0.922    0.994     0.991     1
                 0.971    0.053    0.946      0.971   0.958      0.917    0.993     0.993     2
                 0.956    0.004    0.936      0.956   0.946      0.942    0.999     0.987     3
                 0.837    0.000    0.910      0.837   0.872      0.872    0.999     0.938     4
                 0.773    0.001    0.931      0.773   0.845      0.846    0.998     0.942     5
                 0.885    0.002    0.922      0.885   0.903      0.900    0.999     0.964     6
                 0.946    0.001    0.976      0.946   0.961      0.959    1.000     0.993     7
Weighted Avg.    0.951    0.034    0.951      0.951   0.950      0.920    0.995     0.990

=== Confusion Matrix ===

     a     b     c     d     e     f     g   <-- classified as
 67558  4245     1     0    32     9   141 |     a = 1
  2333 93269   217     3   135   116    24 |     b = 2
     1   214 11789    42    10   278     0 |     c = 3
     0     0   118   759     0    30     0 |     d = 4
    41   637    46     0  2502    11     0 |     e = 5
     7   207   430    30     8  5247     0 |     f = 6
   326    54     0     0     0     0  6674 |     g = 7
```

```
=== Summary ===

Correctly Classified Instances      110973               95.5001 %
Incorrectly Classified Instances      5229                4.4999 %
Kappa statistic                          0.9276
Mean absolute error                      0.0382
Root mean squared error                  0.1102
Relative absolute error                 21.4352 %
Root relative squared error             36.8873 %
Total Number of Instances           116202
```

80%

```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.944    0.020    0.964      0.944   0.954      0.928    0.995     0.992     1
                 0.973    0.048    0.951      0.973   0.962      0.925    0.994     0.994     2
                 0.962    0.004    0.942      0.962   0.952      0.949    0.999     0.990     3
                 0.833    0.000    0.920      0.833   0.874      0.875    0.999     0.939     4
                 0.795    0.001    0.931      0.795   0.857      0.858    0.999     0.949     5
                 0.900    0.002    0.931      0.900   0.916      0.913    0.999     0.971     6
                 0.945    0.001    0.979      0.945   0.962      0.961    1.000     0.995     7
Weighted Avg.    0.955    0.031    0.955      0.955   0.955      0.927    0.995     0.991
```