SparkSession and SparkContext

In [1]:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("classification").getOrCreate()
sc = spark.sparkContext
```

Schema

In [2]:

```
from pyspark.sql.types import StructType, StructField, IntegerType, Doub
leType
```

In [3]:

```
colNames = ["Elevation", "Aspect", "Slope",
"Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology",
"Horizontal_Distance_To_Roadways",
"Hillshade_9am", "Hillshade_Noon", "Hillshade_3pm",
"Horizontal_Distance_To_Fire_Points"]

for i in range(4):
    colNames += ["Wilderness_Area_"+str(i),]

for i in range(40):
    colNames += ["Soil_Type_"+str(i),]

colNames += ["Cover_Type",]
```

In [4]:

```
schema = StructType()
for name in colNames:
    if name == "Cover_Type":
        schema.add(StructField(name, DoubleType(), True))
    else:
        schema.add(StructField(name, IntegerType(), True))
```

Read CSV with the prepared schema

In [5]:

```
data = spark.read.csv('../Dropbox/pj_ss/covtype/covtype.data', header=Fa
lse, schema=schema)
```

In [6]:

```
data.printSchema()
```

```
root
 |-- Elevation: integer (nullable = true)
 |-- Aspect: integer (nullable = true)
 |-- Slope: integer (nullable = true)
 |-- Horizontal_Distance_To_Hydrology: integer (nullable = true)
 |-- Vertical_Distance_To_Hydrology: integer (nullable = true)
 |-- Horizontal_Distance_To_Roadways: integer (nullable = true)
 |-- Hillshade_9am: integer (nullable = true)
 |-- Hillshade_Noon: integer (nullable = true)
 |-- Hillshade_3pm: integer (nullable = true)
 |-- Horizontal_Distance_To_Fire_Points: integer (nullable = true)
 |-- Wilderness_Area_0: integer (nullable = true)
 |-- Wilderness_Area_1: integer (nullable = true)
 |-- Wilderness_Area_2: integer (nullable = true)
 |-- Wilderness_Area_3: integer (nullable = true)
 |-- Soil_Type_0: integer (nullable = true)
 |-- Soil_Type_1: integer (nullable = true)
 |-- Soil_Type_2: integer (nullable = true)
 |-- Soil_Type_3: integer (nullable = true)
 |-- Soil_Type_4: integer (nullable = true)
 |-- Soil_Type_5: integer (nullable = true)
 |-- Soil_Type_6: integer (nullable = true)
 |-- Soil_Type_7: integer (nullable = true)
 |-- Soil_Type_8: integer (nullable = true)
 |-- Soil_Type_9: integer (nullable = true)
 |-- Soil_Type_10: integer (nullable = true)
 |-- Soil_Type_11: integer (nullable = true)
 |-- Soil_Type_12: integer (nullable = true)
 |-- Soil_Type_13: integer (nullable = true)
 |-- Soil_Type_14: integer (nullable = true)
 |-- Soil_Type_15: integer (nullable = true)
 |-- Soil_Type_16: integer (nullable = true)
 |-- Soil_Type_17: integer (nullable = true)
 |-- Soil_Type_18: integer (nullable = true)
 |-- Soil_Type_19: integer (nullable = true)
 |-- Soil_Type_20: integer (nullable = true)
 |-- Soil_Type_21: integer (nullable = true)
 |-- Soil_Type_22: integer (nullable = true)
 |-- Soil_Type_23: integer (nullable = true)
 |-- Soil_Type_24: integer (nullable = true)
 |-- Soil_Type_25: integer (nullable = true)
 |-- Soil_Type_26: integer (nullable = true)
 |-- Soil_Type_27: integer (nullable = true)
 |-- Soil_Type_28: integer (nullable = true)
 |-- Soil_Type_29: integer (nullable = true)
 |-- Soil_Type_30: integer (nullable = true)
 |-- Soil_Type_31: integer (nullable = true)
 |-- Soil_Type_32: integer (nullable = true)
 |-- Soil_Type_33: integer (nullable = true)
 |-- Soil_Type_34: integer (nullable = true)
 |-- Soil_Type_35: integer (nullable = true)
 |-- Soil_Type_36: integer (nullable = true)
 |-- Soil_Type_37: integer (nullable = true)
 |-- Soil_Type_38: integer (nullable = true)
 |-- Soil_Type_39: integer (nullable = true)
 |-- Cover_Type: double (nullable = true)
```

In [7]:
```
data.take(1)
```

Out[7]:
```
[Row(Elevation=2596, Aspect=51, Slope=3, Horizontal_Distance_To_Hydrolo
gy=258, Vertical_Distance_To_Hydrology=0, Horizontal_Distance_To_Roadwa
ys=510, Hillshade_9am=221, Hillshade_Noon=232, Hillshade_3pm=148, Horiz
ontal_Distance_To_Fire_Points=6279, Wilderness_Area_0=1, Wilderness_Are
a_1=0, Wilderness_Area_2=0, Wilderness_Area_3=0, Soil_Type_0=0, Soil_Ty
pe_1=0, Soil_Type_2=0, Soil_Type_3=0, Soil_Type_4=0, Soil_Type_5=0, Soi
l_Type_6=0, Soil_Type_7=0, Soil_Type_8=0, Soil_Type_9=0, Soil_Type_10=
0, Soil_Type_11=0, Soil_Type_12=0, Soil_Type_13=0, Soil_Type_14=0, Soil
_Type_15=0, Soil_Type_16=0, Soil_Type_17=0, Soil_Type_18=0, Soil_Type_1
9=0, Soil_Type_20=0, Soil_Type_21=0, Soil_Type_22=0, Soil_Type_23=0, So
il_Type_24=0, Soil_Type_25=0, Soil_Type_26=0, Soil_Type_27=0, Soil_Type
_28=1, Soil_Type_29=0, Soil_Type_30=0, Soil_Type_31=0, Soil_Type_32=0,
Soil_Type_33=0, Soil_Type_34=0, Soil_Type_35=0, Soil_Type_36=0, Soil_Ty
pe_37=0, Soil_Type_38=0, Soil_Type_39=0, Cover_Type=5.0)]
```

Assembler

In [8]:
```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

In [9]:
```
(trainData, testData) = data.randomSplit([0.9, 0.1])
```

In [10]:
```
trainData
```

Out[10]:
```
DataFrame[Elevation: int, Aspect: int, Slope: int, Horizontal_Distance_
To_Hydrology: int, Vertical_Distance_To_Hydrology: int, Horizontal_Dist
ance_To_Roadways: int, Hillshade_9am: int, Hillshade_Noon: int, Hillsha
de_3pm: int, Horizontal_Distance_To_Fire_Points: int, Wilderness_Area_
0: int, Wilderness_Area_1: int, Wilderness_Area_2: int, Wilderness_Area
_3: int, Soil_Type_0: int, Soil_Type_1: int, Soil_Type_2: int, Soil_Typ
e_3: int, Soil_Type_4: int, Soil_Type_5: int, Soil_Type_6: int, Soil_Ty
pe_7: int, Soil_Type_8: int, Soil_Type_9: int, Soil_Type_10: int, Soil_
Type_11: int, Soil_Type_12: int, Soil_Type_13: int, Soil_Type_14: int,
Soil_Type_15: int, Soil_Type_16: int, Soil_Type_17: int, Soil_Type_18:
int, Soil_Type_19: int, Soil_Type_20: int, Soil_Type_21: int, Soil_Type
_22: int, Soil_Type_23: int, Soil_Type_24: int, Soil_Type_25: int, Soil
_Type_26: int, Soil_Type_27: int, Soil_Type_28: int, Soil_Type_29: int,
Soil_Type_30: int, Soil_Type_31: int, Soil_Type_32: int, Soil_Type_33:
int, Soil_Type_34: int, Soil_Type_35: int, Soil_Type_36: int, Soil_Type
_37: int, Soil_Type_38: int, Soil_Type_39: int, Cover_Type: double]
```

In [11]:
```
inputCols = trainData.drop('Cover_Type').columns
```

```
In [12]: inputCols
```

```
Out[12]: ['Elevation',
          'Aspect',
          'Slope',
          'Horizontal_Distance_To_Hydrology',
          'Vertical_Distance_To_Hydrology',
          'Horizontal_Distance_To_Roadways',
          'Hillshade_9am',
          'Hillshade_Noon',
          'Hillshade_3pm',
          'Horizontal_Distance_To_Fire_Points',
          'Wilderness_Area_0',
          'Wilderness_Area_1',
          'Wilderness_Area_2',
          'Wilderness_Area_3',
          'Soil_Type_0',
          'Soil_Type_1',
          'Soil_Type_2',
          'Soil_Type_3',
          'Soil_Type_4',
          'Soil_Type_5',
          'Soil_Type_6',
          'Soil_Type_7',
          'Soil_Type_8',
          'Soil_Type_9',
          'Soil_Type_10',
          'Soil_Type_11',
          'Soil_Type_12',
          'Soil_Type_13',
          'Soil_Type_14',
          'Soil_Type_15',
          'Soil_Type_16',
          'Soil_Type_17',
          'Soil_Type_18',
          'Soil_Type_19',
          'Soil_Type_20',
          'Soil_Type_21',
          'Soil_Type_22',
          'Soil_Type_23',
          'Soil_Type_24',
          'Soil_Type_25',
          'Soil_Type_26',
          'Soil_Type_27',
          'Soil_Type_28',
          'Soil_Type_29',
          'Soil_Type_30',
          'Soil_Type_31',
          'Soil_Type_32',
          'Soil_Type_33',
          'Soil_Type_34',
          'Soil_Type_35',
          'Soil_Type_36',
          'Soil_Type_37',
          'Soil_Type_38',
          'Soil_Type_39']
```

In [16]:
```
assembler = VectorAssembler(
    inputCols=inputCols,
    outputCol="featureVector")
```

In [17]:
```
assembler
```

Out[17]:
```
VectorAssembler_4bb3bb34a9cf0b6fd419
```

In [18]:
```
assembledTrainData = assembler.transform(trainData)
```

In [19]:
```
assembledTrainData
```

Out[19]:
```
DataFrame[Elevation: int, Aspect: int, Slope: int, Horizontal_Distance_
To_Hydrology: int, Vertical_Distance_To_Hydrology: int, Horizontal_Dist
ance_To_Roadways: int, Hillshade_9am: int, Hillshade_Noon: int, Hillsha
de_3pm: int, Horizontal_Distance_To_Fire_Points: int, Wilderness_Area_
0: int, Wilderness_Area_1: int, Wilderness_Area_2: int, Wilderness_Area
_3: int, Soil_Type_0: int, Soil_Type_1: int, Soil_Type_2: int, Soil_Typ
e_3: int, Soil_Type_4: int, Soil_Type_5: int, Soil_Type_6: int, Soil_Ty
pe_7: int, Soil_Type_8: int, Soil_Type_9: int, Soil_Type_10: int, Soil_
Type_11: int, Soil_Type_12: int, Soil_Type_13: int, Soil_Type_14: int,
Soil_Type_15: int, Soil_Type_16: int, Soil_Type_17: int, Soil_Type_18:
int, Soil_Type_19: int, Soil_Type_20: int, Soil_Type_21: int, Soil_Type
_22: int, Soil_Type_23: int, Soil_Type_24: int, Soil_Type_25: int, Soil
_Type_26: int, Soil_Type_27: int, Soil_Type_28: int, Soil_Type_29: int,
Soil_Type_30: int, Soil_Type_31: int, Soil_Type_32: int, Soil_Type_33:
int, Soil_Type_34: int, Soil_Type_35: int, Soil_Type_36: int, Soil_Type
_37: int, Soil_Type_38: int, Soil_Type_39: int, Cover_Type: double, fea
tureVector: vector]
```

In [20]:
```
assembledTrainData.select('featureVector').show(3, truncate=False)
```

```
+----------------------------------------------------------------------
------------------------------+
|featureVector
                              |
+----------------------------------------------------------------------
------------------------------+
|(54,[0,1,2,3,4,5,6,7,8,9,13,15],[1863.0,37.0,17.0,120.0,18.0,90.0,217.
0,202.0,115.0,769.0,1.0,1.0]) |
|(54,[0,1,2,3,4,5,6,7,8,9,13,18],[1879.0,28.0,19.0,30.0,12.0,95.0,209.
0,196.0,117.0,778.0,1.0,1.0])  |
|(54,[0,1,2,3,4,5,6,7,8,9,13,15],[1888.0,33.0,22.0,150.0,46.0,108.0,20
9.0,185.0,103.0,735.0,1.0,1.0])|
+----------------------------------------------------------------------
------------------------------+
only showing top 3 rows
```

Decision Tree

In [21]:
```
from pyspark.ml.classification import DecisionTreeClassifier
```

```
In [22]: classifier = DecisionTreeClassifier(labelCol="Cover_Type",
                                             featuresCol="featureVector",
                                             predictionCol="prediction")
```

```
In [23]: model = classifier.fit(assembledTrainData)
```

```
In [24]: model
```

```
Out[24]: DecisionTreeClassificationModel (uid=DecisionTreeClassifier_44ac953bdb2
         a9ad6e065) of depth 5 with 63 nodes
```

In [25]:

```
print(model.toDebugString)
```

```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_44ac953bdb2
a9ad6e065) of depth 5 with 63 nodes
  If (feature 0 <= 3034.5)
   If (feature 0 <= 2479.5)
    If (feature 3 <= 15.0)
     If (feature 13 <= 0.5)
      If (feature 17 <= 0.5)
       Predict: 6.0
      Else (feature 17 > 0.5)
       Predict: 6.0
     Else (feature 13 > 0.5)
      If (feature 23 <= 0.5)
       Predict: 4.0
      Else (feature 23 > 0.5)
       Predict: 3.0
    Else (feature 3 > 15.0)
     If (feature 16 <= 0.5)
      If (feature 9 <= 576.5)
       Predict: 3.0
      Else (feature 9 > 576.5)
       Predict: 3.0
     Else (feature 16 > 0.5)
      If (feature 9 <= 1295.0)
       Predict: 3.0
      Else (feature 9 > 1295.0)
       Predict: 4.0
   Else (feature 0 > 2479.5)
    If (feature 17 <= 0.5)
     If (feature 15 <= 0.5)
      If (feature 0 <= 2942.5)
       Predict: 2.0
      Else (feature 0 > 2942.5)
       Predict: 2.0
     Else (feature 15 > 0.5)
      If (feature 9 <= 1380.5)
       Predict: 3.0
      Else (feature 9 > 1380.5)
       Predict: 3.0
    Else (feature 17 > 0.5)
     If (feature 0 <= 2692.5)
      If (feature 0 <= 2635.5)
       Predict: 3.0
      Else (feature 0 > 2635.5)
       Predict: 3.0
     Else (feature 0 > 2692.5)
      If (feature 5 <= 1200.5)
       Predict: 5.0
      Else (feature 5 > 1200.5)
       Predict: 2.0
  Else (feature 0 > 3034.5)
   If (feature 0 <= 3309.5)
    If (feature 7 <= 238.5)
     If (feature 0 <= 3101.5)
      If (feature 3 <= 191.0)
       Predict: 1.0
      Else (feature 3 > 191.0)
       Predict: 2.0
     Else (feature 0 > 3101.5)
      If (feature 5 <= 998.0)
       Predict: 1.0
      Else (feature 5 > 998.0)
       Predict: 1.0
    Else (feature 7 > 238.5)
     If (feature 3 <= 333.0)
      If (feature 0 <= 3186.5)
       Predict: 1.0
      Else (feature 0 > 3186.5)
       Predict: 1.0
     Else (feature 3 > 333.0)
      If (feature 0 <= 3206.5)
       Predict: 2.0
      Else (feature 0 > 3206.5)
       Predict: 1.0
   Else (feature 0 > 3309.5)
    If (feature 12 <= 0.5)
     If (feature 3 <= 290.0)
      If (feature 6 <= 207.5)
       Predict: 1.0
      Else (feature 6 > 207.5)
       Predict: 7.0
     Else (feature 3 > 290.0)
      If (feature 10 <= 0.5)
       Predict: 1.0
      Else (feature 10 > 0.5)
       Predict: 1.0
    Else (feature 12 > 0.5)
     If (feature 45 <= 0.5)
      If (feature 0 <= 3369.5)
       Predict: 7.0
      Else (feature 0 > 3369.5)
       Predict: 7.0
     Else (feature 45 > 0.5)
      If (feature 5 <= 998.0)
       Predict: 7.0
      Else (feature 5 > 998.0)
       Predict: 1.0
```

In [26]:
```
print(model.featureImportances)
```

```
(54,[0,3,5,6,7,9,10,12,13,15,16,17,23,45],[0.8133742673793523,0.0309786
0477331104,0.014747043247755682,0.0023999635668687604,0.025523595174575
104,0.00646850931723158,0.0031668991984998784,0.011192823508553537,0.00
27142714212239146,0.03035888049891708,0.0033263230701791355,0.038074558
83308248,0.0009625115515200479,0.01671174845892951])
```

In [27]:
```
predictions = model.transform(assembledTrainData)
```

In [28]:

```
predictions.select(["Cover_Type", "prediction", "probability"]).show(3,
truncate=False)
```

```
+----------+----------+-----------------------------------------------------
-------------------------------------------------------------------+
|Cover_Type|prediction|probability
                                                                   |
+----------+----------+-----------------------------------------------------
-------------------------------------------------------------------+
|6.0       |3.0       |[0.0,5.213492518638236E-5,0.0570356081539023,0.5
660288827485532,0.025024764089463532,0.0,0.35185861008289454,0.0]|
|6.0       |3.0       |[0.0,5.213492518638236E-5,0.0570356081539023,0.5
660288827485532,0.025024764089463532,0.0,0.35185861008289454,0.0]|
|6.0       |3.0       |[0.0,5.213492518638236E-5,0.0570356081539023,0.5
660288827485532,0.025024764089463532,0.0,0.35185861008289454,0.0]|
+----------+----------+-----------------------------------------------------
-------------------------------------------------------------------+
only showing top 3 rows
```

Evaluation

In [29]:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

In [30]:

```
evaluator = MulticlassClassificationEvaluator(labelCol="Cover_Type",
                                              predictionCol="prediction"
)
```

In [31]:

```
evaluator
```

Out[31]:

```
MulticlassClassificationEvaluator_4a96ab9498d109c86b86
```

In [32]:

```
evaluator.setMetricName("accuracy").evaluate(predictions)
```

Out[32]:

```
0.7022924849787542
```

In [33]:

```
evaluator.setMetricName("f1").evaluate(predictions)
```

Out[33]:

```
0.6857631073030778
```

Pipeline

In [34]:

```
from pyspark.ml import Pipeline
```

In [35]:

```
inputCols = trainData.columns[:-1]
```

In [36]:

```
inputCols
```

Out[36]:

```
['Elevation',
 'Aspect',
 'Slope',
 'Horizontal_Distance_To_Hydrology',
 'Vertical_Distance_To_Hydrology',
 'Horizontal_Distance_To_Roadways',
 'Hillshade_9am',
 'Hillshade_Noon',
 'Hillshade_3pm',
 'Horizontal_Distance_To_Fire_Points',
 'Wilderness_Area_0',
 'Wilderness_Area_1',
 'Wilderness_Area_2',
 'Wilderness_Area_3',
 'Soil_Type_0',
 'Soil_Type_1',
 'Soil_Type_2',
 'Soil_Type_3',
 'Soil_Type_4',
 'Soil_Type_5',
 'Soil_Type_6',
 'Soil_Type_7',
 'Soil_Type_8',
 'Soil_Type_9',
 'Soil_Type_10',
 'Soil_Type_11',
 'Soil_Type_12',
 'Soil_Type_13',
 'Soil_Type_14',
 'Soil_Type_15',
 'Soil_Type_16',
 'Soil_Type_17',
 'Soil_Type_18',
 'Soil_Type_19',
 'Soil_Type_20',
 'Soil_Type_21',
 'Soil_Type_22',
 'Soil_Type_23',
 'Soil_Type_24',
 'Soil_Type_25',
 'Soil_Type_26',
 'Soil_Type_27',
 'Soil_Type_28',
 'Soil_Type_29',
 'Soil_Type_30',
 'Soil_Type_31',
 'Soil_Type_32',
 'Soil_Type_33',
 'Soil_Type_34',
 'Soil_Type_35',
 'Soil_Type_36',
 'Soil_Type_37',
 'Soil_Type_38',
 'Soil_Type_39']
```

In [37]:
```
assembler = VectorAssembler(inputCols=inputCols, outputCol="featureVector")
```

In [38]:
```
classifier = DecisionTreeClassifier(labelCol="Cover_Type",
                                    featuresCol="featureVector",
                                    predictionCol="prediction")
```

In [39]:
```
classifier
```

Out[39]:
```
DecisionTreeClassifier_41c58c49ca4ad38a5bc6
```

In [40]:
```
pipeline = Pipeline(stages=[assembler, classifier])
```

In [41]:
```
pipeline
```

Out[41]:
```
Pipeline_4b6aad06f5c5ac75d01b
```

In [42]:
```
from pyspark.ml.tuning import ParamGridBuilder
```

In [46]:
```
paramGrid = ParamGridBuilder()\
    .addGrid(classifier, ["gini", "entropy"])\
    .addGrid(classifier, [1, 20])\
    .addGrid(classifier, [40, 300])\
    .addGrid(classifier, [0.0, 0.05])\
    .build()
```

In [47]:
```
multiclassEval = MulticlassClassificationEvaluator(
    labelCol="Cover_Type",
    predictionCol="prediction",
    metricName="accuracy")
```

In [48]:
```
multiclassEval.evaluate(predictions)
```

Out[48]:
```
0.7022924849787542
```

In [49]:
```
from pyspark.ml.tuning import TrainValidationSplit
```

In [50]:
```
validator = TrainValidationSplit(
    estimator=pipeline,
    estimatorParamMaps=paramGrid,
    evaluator=multiclassEval,
    trainRatio=0.9)
```

In [51]:
```
validatorModel = validator.fit(trainData)
```

In [52]:
```
bestModel = validatorModel.bestModel
```

In [53]:
```
bestModel
```

Out[53]:
```
PipelineModel_44b3ad4e4f4671d6a89b
```

In [54]:
```
bestModel.stages[-1].extractParamMap()
```

Out[54]:
```
{Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='cach
eNodeIds', doc='If false, the algorithm will pass trees to executors to
match instances with nodes. If true, the algorithm will cache node IDs
for each instance. Caching can speed up training of deeper trees.'): Fa
lse,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='chec
kpointInterval', doc='set checkpoint interval (>= 1) or disable checkpo
int (-1). E.g. 10 means that the cache will get checkpointed every 10 i
terations. Note: this setting will be ignored if the checkpoint directo
ry is not set in the SparkContext'): 10,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='feat
uresCol', doc='features column name'): 'featureVector',
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='impu
rity', doc='Criterion used for information gain calculation (case-insen
sitive). Supported options: entropy, gini'): 'gini',
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='labe
lCol', doc='label column name'): 'Cover_Type',
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='maxB
ins', doc='Max number of bins for discretizing continuous features.  Mu
st be >=2 and >= number of categories for any categorical feature.'): 3
2,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='maxD
epth', doc='Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 lea
f node; depth 1 means 1 internal node + 2 leaf nodes.'): 5,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='maxM
emoryInMB', doc='Maximum memory in MB allocated to histogram aggregatio
n.'): 256,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='minI
nfoGain', doc='Minimum information gain for a split to be considered at
a tree node.'): 0.0,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='minI
nstancesPerNode', doc='Minimum number of instances each child must have
after split.  If a split causes the left or right child to have fewer t
han minInstancesPerNode, the split will be discarded as invalid. Should
be >= 1.'): 1,
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='pred
ictionCol', doc='prediction column name'): 'prediction',
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='prob
abilityCol', doc='Column name for predicted class conditional probabili
ties. Note: Not all models output well-calibrated probability estimate
s! These probabilities should be treated as confidences, not precise pr
obabilities'): 'probability',
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='rawP
redictionCol', doc='raw prediction (a.k.a. confidence) column name'):
'rawPrediction',
 Param(parent='DecisionTreeClassifier_41c58c49ca4ad38a5bc6', name='see
d', doc='random seed'): 7750783964360596788}
```

In [55]:
```
paramsAndMetrics = validatorModel.validationMetrics
```

In [56]:
```
paramsAndMetrics
```

Out[56]:
```
[0.7012882447665056, 0.7012882447665056]
```

```
In [57]: multiclassEval.evaluate(bestModel.transform(testData))

Out[57]: 0.7040250662810316
```

Undoing Onehot Coding

```
In [58]: wildernessCols = []
         for i in range(4):
             wildernessCols += ["Wilderness_Area_"+str(i),]
```

```
In [63]: wildernessAssembler = VectorAssembler(
             inputCols=wildernessCols,
             outputCol="wilderness")
```

```
In [64]: from pyspark.sql.functions import udf
         from pyspark.sql.types import ArrayType, DoubleType, StructType
```

```
In [65]: unhotudf = udf(lambda x: float(x.toArray().nonzero()[0]), DoubleType())
```

```
In [66]: withWilderness = wildernessAssembler.transform(data)
```

```
In [67]: withWilderness.take(1)

Out[67]: [Row(Elevation=2596, Aspect=51, Slope=3, Horizontal_Distance_To_Hydrolo
         gy=258, Vertical_Distance_To_Hydrology=0, Horizontal_Distance_To_Roadwa
         ys=510, Hillshade_9am=221, Hillshade_Noon=232, Hillshade_3pm=148, Horiz
         ontal_Distance_To_Fire_Points=6279, Wilderness_Area_0=1, Wilderness_Are
         a_1=0, Wilderness_Area_2=0, Wilderness_Area_3=0, Soil_Type_0=0, Soil_Ty
         pe_1=0, Soil_Type_2=0, Soil_Type_3=0, Soil_Type_4=0, Soil_Type_5=0, Soi
         l_Type_6=0, Soil_Type_7=0, Soil_Type_8=0, Soil_Type_9=0, Soil_Type_10=
         0, Soil_Type_11=0, Soil_Type_12=0, Soil_Type_13=0, Soil_Type_14=0, Soil
         _Type_15=0, Soil_Type_16=0, Soil_Type_17=0, Soil_Type_18=0, Soil_Type_1
         9=0, Soil_Type_20=0, Soil_Type_21=0, Soil_Type_22=0, Soil_Type_23=0, So
         il_Type_24=0, Soil_Type_25=0, Soil_Type_26=0, Soil_Type_27=0, Soil_Type
         _28=1, Soil_Type_29=0, Soil_Type_30=0, Soil_Type_31=0, Soil_Type_32=0,
         Soil_Type_33=0, Soil_Type_34=0, Soil_Type_35=0, Soil_Type_36=0, Soil_Ty
         pe_37=0, Soil_Type_38=0, Soil_Type_39=0, Cover_Type=5.0, wilderness=Spa
         rseVector(4, {0: 1.0}))]
```

```
In [68]: withWilderness = withWilderness\
             .drop(*wildernessCols)\
             .withColumn("wilderness", unhotudf(withWilderness['wilderness']))
```

```
In [69]: withWilderness.take(1)
```

```
Out[69]: [Row(Elevation=2596, Aspect=51, Slope=3, Horizontal_Distance_To_Hydrolo
         gy=258, Vertical_Distance_To_Hydrology=0, Horizontal_Distance_To_Roadwa
         ys=510, Hillshade_9am=221, Hillshade_Noon=232, Hillshade_3pm=148, Horiz
         ontal_Distance_To_Fire_Points=6279, Soil_Type_0=0, Soil_Type_1=0, Soil_
         Type_2=0, Soil_Type_3=0, Soil_Type_4=0, Soil_Type_5=0, Soil_Type_6=0, S
         oil_Type_7=0, Soil_Type_8=0, Soil_Type_9=0, Soil_Type_10=0, Soil_Type_1
         1=0, Soil_Type_12=0, Soil_Type_13=0, Soil_Type_14=0, Soil_Type_15=0, So
         il_Type_16=0, Soil_Type_17=0, Soil_Type_18=0, Soil_Type_19=0, Soil_Type
         _20=0, Soil_Type_21=0, Soil_Type_22=0, Soil_Type_23=0, Soil_Type_24=0,
         Soil_Type_25=0, Soil_Type_26=0, Soil_Type_27=0, Soil_Type_28=1, Soil_Ty
         pe_29=0, Soil_Type_30=0, Soil_Type_31=0, Soil_Type_32=0, Soil_Type_33=
         0, Soil_Type_34=0, Soil_Type_35=0, Soil_Type_36=0, Soil_Type_37=0, Soil
         _Type_38=0, Soil_Type_39=0, Cover_Type=5.0, wilderness=0.0)]
```

```
In [70]: soilCols = []
         for i in range(40):
             soilCols += ["Soil_Type_"+str(i),]
```

```
In [71]: soilAssembler = VectorAssembler(
                 inputCols=soilCols,
                 outputCol="soil")
```

```
In [72]: withWilderness = soilAssembler.transform(withWilderness)
```

```
In [73]: unencodedData = withWilderness\
             .drop(*soilCols)\
             .withColumn("soil", unhotudf(withWilderness['soil']))
```

```
In [74]: unencodedData.take(2)
```

```
Out[74]: [Row(Elevation=2596, Aspect=51, Slope=3, Horizontal_Distance_To_Hydrolo
         gy=258, Vertical_Distance_To_Hydrology=0, Horizontal_Distance_To_Roadwa
         ys=510, Hillshade_9am=221, Hillshade_Noon=232, Hillshade_3pm=148, Horiz
         ontal_Distance_To_Fire_Points=6279, Cover_Type=5.0, wilderness=0.0, soi
         l=28.0),
          Row(Elevation=2590, Aspect=56, Slope=2, Horizontal_Distance_To_Hydrolo
         gy=212, Vertical_Distance_To_Hydrology=-6, Horizontal_Distance_To_Roadw
         ays=390, Hillshade_9am=220, Hillshade_Noon=235, Hillshade_3pm=151, Hori
         zontal_Distance_To_Fire_Points=6225, Cover_Type=5.0, wilderness=0.0, so
         il=28.0)]
```

Decision Tree with Unencoded Data

```
In [75]: (unencTrainData, unencTestData) = unencodedData.randomSplit([0.9, 0.1])
```

```
In [76]: from pyspark.ml.feature import VectorIndexer
```

```
In [77]: inputCols = unencTrainData.drop('Cover_Type').columns
```

In [78]:

```
inputCols
```

Out[78]:

```
['Elevation',
 'Aspect',
 'Slope',
 'Horizontal_Distance_To_Hydrology',
 'Vertical_Distance_To_Hydrology',
 'Horizontal_Distance_To_Roadways',
 'Hillshade_9am',
 'Hillshade_Noon',
 'Hillshade_3pm',
 'Horizontal_Distance_To_Fire_Points',
 'wilderness',
 'soil']
```

In [79]:

```
assembler = VectorAssembler(
    inputCols=inputCols,
    outputCol="featureVector")
```

In [80]:

```
indexer = VectorIndexer(
    maxCategories=40,
    inputCol="featureVector",
    outputCol="indexedVector")
```

In [81]:

```
classifier = DecisionTreeClassifier(
    seed=42,
    labelCol="Cover_Type",
    featuresCol="indexedVector",
    predictionCol="prediction")
```

In [82]:

```
pipeline = Pipeline(stages=[assembler, indexer, classifier])
```

Random Forrest Classifier

In [83]:

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

In [84]:

```
classifier = RandomForestClassifier(
    seed=42,
    maxBins=40,
    labelCol="Cover_Type",
    featuresCol="indexedVector",
    predictionCol="prediction")
```

In [85]:

```
pipeline = Pipeline(stages=[assembler, indexer, classifier])
```

In [86]:

```
paramGrid = ParamGridBuilder()\
    .addGrid(classifier.minInfoGain, [0.0, 0.05])\
    .addGrid(classifier.numTrees, [1, 10])\
    .build()
```

In [87]:

```
multiclassEval = MulticlassClassificationEvaluator(
    labelCol="Cover_Type",
    predictionCol="prediction",
    metricName="accuracy")
```

In [88]:

```
validator = TrainValidationSplit(
    seed=42,
    estimator=pipeline,
    evaluator=multiclassEval,
    estimatorParamMaps=paramGrid,
    trainRatio=0.9)
```

In [89]:

```
validatorModel = validator.fit(unencTrainData)
```

In [90]:

```
bestModel = validatorModel.bestModel
```

In [91]:

```
bestModel
```

Out[91]:

```
PipelineModel_4cdf8601e7c081412e4a
```

```
In [92]: forestModel = bestModel.stages[-1]
         print(forestModel.extractParamMap())
```

```
{Param(parent='RandomForestClassifier_4fa2ae5257b23789060c', name='cach
eNodeIds', doc='If false, the algorithm will pass trees to executors to
match instances with nodes. If true, the algorithm will cache node IDs
for each instance. Caching can speed up training of deeper trees.'): Fa
lse, Param(parent='RandomForestClassifier_4fa2ae5257b23789060c', name
='checkpointInterval', doc='set checkpoint interval (>= 1) or disable c
heckpoint (-1). E.g. 10 means that the cache will get checkpointed ever
y 10 iterations. Note: this setting will be ignored if the checkpoint d
irectory is not set in the SparkContext'): 10, Param(parent='RandomFore
stClassifier_4fa2ae5257b23789060c', name='featureSubsetStrategy', doc
='The number of features to consider for splits at each tree node. Supp
orted options: auto, all, onethird, sqrt, log2, (0.0-1.0], [1-n].'): 'a
uto', Param(parent='RandomForestClassifier_4fa2ae5257b23789060c', name
='featuresCol', doc='features column name'): 'indexedVector', Param(par
ent='RandomForestClassifier_4fa2ae5257b23789060c', name='impurity', doc
='Criterion used for information gain calculation (case-insensitive). S
upported options: entropy, gini'): 'gini', Param(parent='RandomForestCl
assifier_4fa2ae5257b23789060c', name='labelCol', doc='label column nam
e'): 'Cover_Type', Param(parent='RandomForestClassifier_4fa2ae5257b2378
9060c', name='maxBins', doc='Max number of bins for discretizing contin
uous features.  Must be >=2 and >= number of categories for any categor
ical feature.'): 40, Param(parent='RandomForestClassifier_4fa2ae5257b23
789060c', name='maxDepth', doc='Maximum depth of the tree. (>= 0) E.g.,
depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf node
s.'): 5, Param(parent='RandomForestClassifier_4fa2ae5257b23789060c', na
me='maxMemoryInMB', doc='Maximum memory in MB allocated to histogram ag
gregation.'): 256, Param(parent='RandomForestClassifier_4fa2ae5257b2378
9060c', name='minInfoGain', doc='Minimum information gain for a split t
o be considered at a tree node.'): 0.0, Param(parent='RandomForestClass
ifier_4fa2ae5257b23789060c', name='minInstancesPerNode', doc='Minimum n
umber of instances each child must have after split.  If a split causes
the left or right child to have fewer than minInstancesPerNode, the spl
it will be discarded as invalid. Should be >= 1.'): 1, Param(parent='Ra
ndomForestClassifier_4fa2ae5257b23789060c', name='numTrees', doc='Numbe
r of trees to train (>= 1)'): 1, Param(parent='RandomForestClassifier_4
fa2ae5257b23789060c', name='predictionCol', doc='prediction column nam
e'): 'prediction', Param(parent='RandomForestClassifier_4fa2ae5257b2378
9060c', name='probabilityCol', doc='Column name for predicted class con
ditional probabilities. Note: Not all models output well-calibrated pro
bability estimates! These probabilities should be treated as confidence
s, not precise probabilities'): 'probability', Param(parent='RandomFore
stClassifier_4fa2ae5257b23789060c', name='rawPredictionCol', doc='raw p
rediction (a.k.a. confidence) column name'): 'rawPrediction', Param(par
ent='RandomForestClassifier_4fa2ae5257b23789060c', name='seed', doc='ra
ndom seed'): 42, Param(parent='RandomForestClassifier_4fa2ae5257b237890
60c', name='subsamplingRate', doc='Fraction of the training data used f
or learning each decision tree, in range (0, 1].'): 1.0}
```

```
In [93]: forestModel.getNumTrees
```

```
Out[93]: 1
```

```
In [94]: sorted(list(zip(inputCols, forestModel.featureImportances)), key=lambda
         x: x[1], reverse=True)
```

```
Out[94]: [('Elevation', 0.7822498415213323),
          ('soil', 0.11297264686002277),
          ('wilderness', 0.037105461899423466),
          ('Horizontal_Distance_To_Roadways', 0.03535837398448177),
          ('Horizontal_Distance_To_Fire_Points', 0.012732741797356662),
          ('Hillshade_Noon', 0.011420066167183551),
          ('Horizontal_Distance_To_Hydrology', 0.005068419955167024),
          ('Vertical_Distance_To_Hydrology', 0.003092447814683044),
          ('Aspect', 0.0),
          ('Slope', 0.0),
          ('Hillshade_9am', 0.0),
          ('Hillshade_3pm', 0.0)]
```

```
In [95]: testAccuracy = multiclassEval.evaluate(bestModel.transform(unencTestData
         ))
```

```
In [96]: testAccuracy
```

```
Out[96]: 0.6989563173901023
```