实验九　使用决策树算法预测森林植被

（一）实验目的

1. 了解模型构造和调优的相关概念：特征、向量、训练和交叉检验等；
2. 理解决策树和决策森林算法的思想；
3. 掌握大数据分析实验平台中Spark MLlib组件的使用方法。

（二）实验要求

1. 基于Covtype数据集构造决策树和决策森林并调优；
2. 在Spark MLlib中实现决策树和决策森林算法；
3. 根据地理位置、土壤类型等信息预测森林植被的类型。

（三）实验环境

1. 大数据分析实验平台（FSDP）；
2. GraphX；
3. Scala XML；
4. Spark MLlib。

（四）实验步骤

1. Covtype数据集的获取与准备；
2. 基于Spark MLlib构造决策树；
3. 对所构造的决策树进行调优；
4. 构造随机决策森林进行森林植被预测。

1、Covtype数据集的获取与准备

本实验使用Covtype数据集，该数据集可以在线下载（http://t.cn/R2wmlsl），包含一个CSV格式的压缩数据文件covtype.data.gz，并附带一个描述数据文件的信息文件covtype.info。

该数据集记录了不同地块的森林植被类型，每个样本包含了描述每块土地的若干特征，包括海拔、坡度、到水源的距离、遮阳情况和土壤类型等，并且随同给出了地块的已知森林植被类型。我们需要总共54个特征中的其余各项来预测森林植被类型。

该数据集为CSV格式，解压covtype.data文件并复制到大数据分析实验平台的HDFS中，本实验中，数据文件存放在/user/ds/目录下。

2、基于Spark Mllib构造决策树

首先将数据输入改为LabeledPoint对象格式，可以从数据集中取出部分数据，用以评估所得到的模型。为了评测保留的收听数据和模型预测之间的一致性，采用AUC指标，将评价指标改为精确度指标。同时将数据分成完整的三部分：训练集、交叉检验集（CV）和测试集，其中训练集占80%，交叉检验集和测试集各占10%，主要代码如下：

```
val Array(trainData, cvData, testData) =
data.randomSplit(Array(0.8, 0.1, 0.1))
trainData.cache() cvData.cache()
testData.cache()
```

DecisionTree实现也有几个超参数，需要为它们选择值。训练集和CV集用于给这些超参数选择一个合适值，测试集用于对基于选定超参数的模型期望准确度做无偏估计。模型在交叉检验集上的准确度往往不是无偏差的。本实验将在测试集上评估最终模型。

在训练集上构造一个 DecisionTreeModel 模型，参数采用默认值，并用CV集来计算结果模型的指标，主要代码如下：

```
import org.apache.spark.mllib.evaluation._
import org.apache.spark.mllib.tree._
import org.apache.spark.mllib.tree.model._
import org.apache.spark.rdd._
def getMetrics(model: DecisionTreeModel, data:
RDD[LabeledPoint]):
    MulticlassMetrics = {
  val predictionsAndLabels = data.map(example
      =>(model.predict(example.features),
      example.label))
  new MulticlassMetrics(predictionsAndLabels)
}
  val model = DecisionTree.trainClassifier(
    trainData,7,Map[Int,Int](),"gini",4,100)
  val metrics = getMetrics(model,cvData)
```

用trainClassifier指示每个LabeledPoint里的目标都应该当作不同的类别标号，同时必须指明数据集中目标的取值个数为7。Map保存类别型特征的信息，MulticlassMetrics以不同方式计算分类器预测质量的标准指标，分类器运行在CV集上。

理想情况下，分类器对CV集中每个样本的目标类别应该都能做出正确预测，这里用准确度、精确度、召回率等指标度量这种正确性。

按照类别在训练集中出现的比例来预测类别，每次分类的正确度将和一个类型在CV集中出现的次数成正比。比如，一个类别在训练集中占20%，在CV集中占

实验九　使用决策树算法预测森林植被

10%，那么该类别将贡献10%的20%，即2%的总体准确度。

通过按20%的时候将样本猜测为该类，CV集样本中有10%的样本会被猜对。将所有类别在训练集和CV集出现的概率相乘，然后把结果相加，就可以得到对准确度的评估：

```
import org.apache.spark.rdd._
def classProbabilities(data: RDD[LabeledPoint]):
Array[Double] = {
    val countsByCategory = data.map(_.label)
                               .countByValue()
    val counts =countsByCategory.toArray.sortBy
                (_._1).map(_._2)
    counts.map(_.toDouble / counts.sum) }
val trainPriorProb = classProb(trainData)
val cvPriorProb = classProb(cvData)
trainPriorProb.zip(cvPriorProb).map {
    case(trainProb,cvProb)=>trainProb*cvProb
}.sum
```

先构造超参数取不同值时的不同组合的模型，然后用某个指标评估每个组合结果的质量，通过这种方式来选择超参数值。本实验的指标为多元分类准确度，控制决策树选择过程的超参数为最大深度、最大桶数和不纯性度量。

3、对所构造的决策树进行调优

用Spark将不同不纯性度量所得到决策树的准确度值的多种组合并报告结果：

```
val evaluations =
  for (impurity <- Array("gini", "entropy");
       depth    <- Array(1, 20);
       bins     <- Array(10, 300))
    yield {
      val model = DecisionTree.trainClassifier(
        trainData, 7, Map[Int,Int](), impurity,
        depth, bins)
      val predictLabels=cvData.map(example=>
        (model.predict(example.features),
         example.label))
      val accuracy =
        new MulclassMetric(predictLabels).preci
      ((impurity, depth, bins), accuracy)
    }
evaluations.sortBy(_._2).reverse.foreach(print)
```

前面的测试表明，目前为止超参数的最佳选择是：不纯性度量采用熵，最大深度为20，桶数为300，这时得到的准确度为91.2%。

最后，用得到的超参数同时在训练集和CV集上构造模型并且像前面那样进行评估：

```
val model = DecisionTree.trainClassifier
(trainData.union(cvData), 7, Map[Int,Int](),
"entropy", 20, 300)
```

重复将数据集拆分成训练集/CV集/测试集和模型评估的过程，指定两个新的类别型特征的不同取值个数。由于地块（soil）特征有40个不同的取值，决策树需要桶数目最少增加到40。考虑前面的结果，增加决策树的深度至30——当前决策树能支持的最大深度。

最后，在训练集和 CV 集上的准确度报告如下：

```
for(impurity <- Array("gini", "entropy");
       depth    <- Array(10, 20, 30);
       bins     <- Array(40, 300))
val model = DecisionTree.trainClassifier (
        trainData, 7, Map(10 -> 4, 11 -> 40),
        impurity, depth, bins)
val trAccur = getMetri(model,trData).precision
val cvAccur = getMetri(model,cvData).precision
((impurity, depth, bins), (trAccur, cvAccur))
```

4、构造随机决策森林进行森林植被预测

随机决策森林是由多个决策树独立构造而成，每一棵都能对正确目标值给出合理、独立且互不相同的估计。这些树的集体平均预测比任一个体预测更接近正确答案。通过RandomForest，Spark MLlib可以构建随机决策森林：

```
val forest = RandomForest.trainClassifier (
    trainData, 7, Map(10 -> 4, 11 -> 40), 20,
    "auto", "entropy", 30, 300)
```

这里出现了两个新参数。第一个代表要构建多少棵树，这里是20。第二个新参数是特征决策树每层的评估特征选择策略，这里设为auto。随机决策森林在实现过程中决策规则不会考虑全部特征，而只考虑全部特征的一个子集。特征选择策略参数控制算法如何选择该特征子集。只检查少数特征速度明显要快，并且由于速度快，随机决策森林才得以构造多棵决策树。

随机决策森林的预测只是所有决策树预测的加权平均。对于类别型目标，这就是得票最多的类别，或有决策树概率平均后的最大可能值。随机决策森林和决策树一样也支持回归问题，这时森林作出的预测就是每棵树预测值的平均。训练集由LabeledPoint类型实例组成，每个实例包含一个Vector和目标值，它们分别是输入和已知的输出。

现在已经展现了DecisionTree和RandomForest训练的结果，它们分别是DecisionTreeModel和RandomForestModel对象。这两个模型对象本质上都只有一个方法predict()。和LabeledPoint的特征向量部分一样，predict()方法接受一个Vector。因此通过把每个新样本转换成一个特征向量，同样可以对它进行分类并预测它的目标类别：

```
val input =
"2709,125,28,67,23,3224,253,207,61,6094,0,29"
val vector =
Vectors.dense(input.split(',').map(_.toDouble))
forest.predict(vector)
```