## 大数据分布式计算-期末作业

## 王源禾-应用统计专硕

### 一、背景介绍

课上课下所学分布式计算工具，使我们如今终于可以处理超大量的数据，现有的统计工具正一步步实现“分布式”化，在大数据上可用的工具越来越多。但同时我们应注意到，分布式平台上提供的方法更多的并非是统计计算方法。对大量的数据进行分析，会消耗大量的计算资源。

本文选取了kaggle上经典的美国49州的车祸数据集，收集了2016年2月至2020年6月共约350万条事故记录。[数据链接](https://www.kaggle.com/sobhanmoosavi/us-accidents)为https://www.kaggle.com/sobhanmoosavi/us-accidents

本文使用的计算工具为学校提供的搭建了Hadoop与Spark平台的云服务器，计算力有限，且同学也一起使用占用较多。故本文在本地使用数据子集进行了预测试，测试完毕后再放入HDFS上对全数据运行，再测试结果。HDFS上的作业地址为：/home/devel/students/2020210995wangyuanhe/spark

本文的研究思路如下：

首先，读取数据，并对数据进行清洗。数据量共49列，本文使用了其中29列进行研究。对其中部分变量做了变换、调整，将定性变量转变为0-1变量。同时，本文对三水平及以上的定性变量进行拆分与合并，并且对缺失超50%的列进行了删除，得到新的数据框。

第二，对数据进行了列联表分析与数据透视分析，掌握数据的性质与特征，观察变量分组比较结果。

第三，将数据转化为RDD形式后，将变量按7:3分为训练集与测试集，使用随机森林模型进行分类拟合，并且对结果进行评价。

### 二、读取数据与清洗

具体的变量介绍见表1。

采用的变量分为三类，共26个自变量+1个因变量。

第一类为车祸详细信息。包括作为因变量的“严重等级”，以及作为自变量的车祸发生时的车道、影响距离、所处州、所处时区四个变量。其中，对车道信息进行分析后，将该列数据转化为“是否为右车道”的定性变量数据。所处州进行分析后，转化为“是否为加利福尼亚州(CA)”的定性变量数据。

第二类为车祸附近气象数据。由于部分变量缺失较多，本文选择了事故发生地的华氏温度、湿度、大气压强、能见度以及风速。其中，对于大气压强由于变化不大，使用了30英寸作为分界将该变量转化为“是否大于30inches”的定性变量。

第三类为车祸附近环境信息。包括周围是否存在便利设施、是否存在减速带、是否存在十字路口、是否存在错车道、是否存在路口、是否存在禁止出口标志、是否存在铁路、是否存在环状交叉路、是否存在车站、是否存在公交车站、是否存在限速标志、是否存在交通信号灯、是否存在转弯车道（国外特有的圆环）。

第四类为时间信息，主要包括四种日夜的区分方式，基于日出日落的昼夜、民用曙暮光、航海用曙暮光以及天文用曙暮光。

因变量严重等级根据是否大于2级进行二元化，变成2水平定型变量。

在对数据列处理结束后，将缺失数据行删除。

表1 变量介绍表

|  |  |  |  |
| --- | --- | --- | --- |
| 变量名 | 含义 | 变量类型 | 取值范围 |
| Distance\_mi | 事故影响距离（米） | 定量 | 0,24.04 |
| Temperature\_F | 华氏温度 | 定量 | 5,106 |
| Humidity | 湿度 | 定量 | 8,100 |
| Visibility\_mi | 能见度（米） | 定量 | 0.2,80 |
| Wind\_Speed\_mph | 风速 | 定量 | 1.2,33.4 |
| Amenity\_int | 是否存在便利店 | 定性 | 0,1 |
| Higher\_pressure\_int | 气压是否高于30 | 定性 | 0,1 |
| Side\_R\_int | 路右侧 | 定性 | 0,1 |
| Is\_US\_Pacific\_int | 在太平洋时区 | 定性 | 0,1 |
| Is\_CA\_int | 在加州 | 定性 | 0,1 |
| Bump\_int | 是否存在减速带 | 定性 | 0,1 |
| Crossing\_int | 是否存在十字路口 | 定性 | 0,1 |
| Give\_Way\_int | 是否存在错车道 | 定性 | 0,1 |
| Junction\_int | 是否存在路口 | 定性 | 0,1 |
| No\_Exit\_int | 是否存在禁止离开标志 | 定性 | 0,1 |
| Railway\_int | 是否存在铁路 | 定性 | 0,1 |
| Roundabout\_int | 是否存在环状交叉路 | 定性 | 0,1 |
| Station\_int | 是否存在火车站 | 定性 | 0,1 |
| Stop\_int | 是否存在汽车站 | 定性 | 0,1 |
| Traffic\_Calming\_int | 是否存在 | 定性 | 0,1 |
| Traffic\_Signal\_int | 是否存在交通信号灯 | 定性 | 0,1 |
| Turning\_Loop\_int | 是否存在转盘 | 定性 | 0,1 |
| Sunrise\_Sunset\_int | 日出日落的昼夜 | 定性 | 0,1 |
| Civil\_Twilight\_int | 民用曙暮光 | 定性 | 0,1 |
| Nautical\_Twilight\_int | 航海用曙暮光 | 定性 | 0,1 |
| Astronomical\_Twilight\_int | 天文用曙暮光 | 定性 | 0,1 |
| Severe\_condition\_int | 事故严重性 | 定性-因变量 | 0,1 |

### 三、探索性分析

我们使用分组统计工具，对于变量中感兴趣的列，构建对应数据透视表。将小样本结果展示如下：

**1.白天晚上-严重程度-能见度 均值**

由结果可知，白天晚上对于能见度本身即有一定影响，晚上能见度稍低于白天。发生事故时，晚上的严重事故对应的能见度较高，但差异并不明显。

+------------------+-----------------+-----------------+  
|Sunrise\_Sunset\_int| 0| 1|  
+------------------+-----------------+-----------------+  
| 1|9.233051574886797|9.198871490178403|  
| 0|8.965545832201112|8.968837346750709|  
+------------------+-----------------+-----------------+

**2.时区-严重程度-能见度 均值**

由结果可知，时区对事故发生严重性影响明显。在US\_Pacific时区（即西海岸太平洋时区），事故发生时能见度较高，在其他时区平均能见度较低。说明在西海岸发生的事故与能见度关系不大，而其他时区发生的事故可能受到能见度影响较大。

+-----------------+-----------------+-----------------+  
|Is\_US\_Pacific\_int| 0| 1|  
+-----------------+-----------------+-----------------+  
| 1|9.188526880444936| 9.19591528495979|  
| 0|9.163683622734286|9.118213268678888|  
+-----------------+-----------------+-----------------+

**3.车道位置-严重程度-能见度 均值**

由结果可知，在道路右侧发生严重事故的时候，能见度较高，而在道路左侧发生严重事故时，能见度平均较低。发生不严重事故时，路左路右能见度差异不大。路右侧发生事故时，平均能见度稍高于道路左侧。几个数据几乎无差别。

+----------+-----------------+-----------------+  
|Side\_R\_int| 0| 1|  
+----------+-----------------+-----------------+  
| 1|9.169556061333106|9.133147684697581|  
| 0| 9.17524828241871|9.182424423195016|  
+----------+-----------------+-----------------+

**4.路左路右-严重程度-昼夜 数量**

由结果可知，在路右侧发生事故的可能性最大，占了全部事件的95%以上，但事故严重程度对道路左右区别并不明显。晚上发生事故的比例稍高，但白天发生严重事故的数量稍高。

+----------+------+------+  
|Side\_R\_int| 0| 1|  
+----------+------+------+  
| 1|241032|644835|  
| 0| 16147| 40848|  
+----------+------+------+

**5.日出日落-严重程度-风速 均值**

由结果可知，日出时的风速均值普遍较高，而其中严重事故发生时的平均风速最高。

+------------------+-----------------+-----------------+  
|Sunrise\_Sunset\_int| 0| 1|  
+------------------+-----------------+-----------------+  
| 1|8.477338902080852|8.932200302471946|  
| 0|6.758330039607091|7.472274952465167|  
+------------------+-----------------+-----------------+

根据以上探索性数据分析大概可知，事故的严重等级与多方面有关，受到风速、昼夜、能见度等变量的影响。其中有些情况与我们认知相符，有些情况则超出我们的预计。

### 四、建立分类模型

**1.模型结果**

本文建立的模型为随机森林模型。随机森林是一种投票模型，将多棵树的分类结果进行投票，最终选取一棵树，本文设定一共10棵树，树最大深度为4，树最大装箱数为32。最大装箱数是为了近似统计变量，可以把数据近似分为32段去研究。

使用的信息增益为“Gini不纯度”。以信息增益来度量属性选择，即看特征能够为系统带来多少信息，带来的信息越多，则该特征越重要。

首先对需要的变量进行初始化，先按照7:3随机划分训练集与测试集。训练集共2125620数据，测试集共907923数据。下面是模型的核心代码。

featuresArray = changedTypedf.columns[:-1] # 最后一列是因变量（标签）   
assembler = VectorAssembler().setInputCols(featuresArray).setOutputCol("features")   
featureIndexer = VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").setMaxCategories(3)   
# 设置maxCategories，使具有>3个不同水平的特征，就被视为连续变量   
rf = RandomForestRegressor(maxDepth=4).setLabelCol('Severe\_condition\_int').setFeaturesCol("indexedFeatures").setNumTrees(10)

上述代码经过运行，带入测试集数据进行测试，得到一系列预测结果。其中数值表示对于严重程度的概率，可以粗略认为，大于等于0.5的预测即认为多种现象造成的事故等级严重，小于0.5的预测即认为造成的事故等级不严重。模型基本结果展示如下（包含了一棵树的形式，省略了10棵树）：

+-------------------+--------------------+--------------------+   
| prediction|Severe\_condition\_int| features|   
+-------------------+--------------------+--------------------+   
|0.12938209168761905| 0|(26,[1,2,3,4],[19...|   
|0.37756768929679596| 0|(26,[1,2,3,4,6,7,...|   
|0.06986661156578408| 0|(26,[1,2,3,4,22,2...|   
|0.12938209168761905| 0|(26,[1,2,3,4],[31...|   
|0.06887701440715391| 0|(26,[1,2,3,4,6,22...|   
|0.11539020792101147| 0|(26,[1,2,3,4,6],[...|   
| 0.3589968649259682| 0|(26,[1,2,3,4,7,24...|   
|0.29386467380140746| 0|(26,[1,2,3,4,6,7,...|   
|0.35762211454410997| 0|(26,[1,2,3,4,6,7,...|   
|0.35762211454410997| 0|(26,[1,2,3,4,6,7,...|   
+-------------------+--------------------+--------------------+   
only showing top 10 rows   
   
VectorIndexerModel: uid=VectorIndexer\_7d9478729964, numFeatures=26, handleInvalid=error   
训练好的随机森林模型:   
RandomForestRegressionModel: uid=RandomForestRegressor\_7fe25057cac4, numTrees=10, numFeatures=26   
 Tree 0 (weight 1.0):   
 If (feature 7 in {0.0})   
 If (feature 0 <= 0.0205)   
 If (feature 4 <= 6.95)   
 If (feature 25 in {1.0})   
 Predict: 0.04641368011756351   
 Else (feature 25 not in {1.0})   
 Predict: 0.061121815457075575   
 Else (feature 4 > 6.95)   
 If (feature 2 <= 17.5)   
 Predict: 0.032933478735005454   
 Else (feature 2 > 17.5)   
 Predict: 0.06213332260715168   
 Else (feature 0 > 0.0205)   
 If (feature 24 in {1.0})   
 If (feature 0 <= 0.2594999952315)   
 Predict: 0.3114456267843239   
 Else (feature 0 > 0.2594999952315)   
 Predict: 0.5190986534762297   
 Else (feature 24 not in {1.0})   
 If (feature 0 <= 0.11899999865899999)   
 Predict: 0.41819160387513454   
 Else (feature 0 > 0.11899999865899999)   
 Predict: 0.6121468926553673   
 Else (feature 7 not in {0.0})   
 If (feature 25 in {1.0})   
 If (feature 8 in {1.0})   
 If (feature 1 <= 60.05)   
 Predict: 0.2495663155557406   
 Else (feature 1 > 60.05)   
 Predict: 0.303119331035031   
 Else (feature 8 not in {1.0})   
 If (feature 6 in {0.0})   
 Predict: 0.36328317710902813   
 Else (feature 6 not in {0.0})   
 Predict: 0.384098554191688   
 Else (feature 25 not in {1.0})   
 If (feature 8 in {1.0})   
 If (feature 4 <= 3.25)   
 Predict: 0.1633423623081343   
 Else (feature 4 > 3.25)   
 Predict: 0.37058192323565825   
 Else (feature 8 not in {1.0})   
 If (feature 0 <= 0.11899999865899999)   
 Predict: 0.42499702132729655   
 Else (feature 0 > 0.11899999865899999)   
 Predict: 0.6517679349873526

使用测试集上的测试结果，可以得到模型的准确率与AUC值，模型的准确率为0.7004，召回率（TP/（TP+TN））为0.0444，AUC值为0.6729。由上面的树可以知道，根据不同的特征（feature）取值，在不同的节点进行了分叉，最终得到了全部的预测值。

同时，综合了全部的树结果，本文发现树的深度并不需要4层，徒增大量的计算难度，且并没有有效改变结果。也可以改变树的数量和最大装箱数，以此进行模型的参数调优。同时，可以改变验证策略，改用k折交叉验证，可以提高验证效率和精度。

根据模型结果，可以得到一个通过现有环境等情况推测事故发生严重程度的随机森林模型。该模型可以帮助预警可能到来的严重事故，在事故发生后，根据现有条件在处理前粗略推测事故的严重程度，并据此配置不同程度的资源进行处理。其实该数据集还有更多的用法，包括研究车祸热点位置，人员伤亡分析以及提取因果规则以预测车祸，以及研究降雨或其他环境刺激对事故发生的影响。

**2.与单机的对比及思考**

本文在分布式平台上对数据进行处理， Spark里也可以做线性回归与逻辑回归，也可以实现如本文中的随机森林模型。对比单机实现结果上看无异，我们利用spark更好性能的云服务器，输入自己的交互命令，就能在Spark上操作大量数据。对于原本在单机上由于内存不足，计算性能不足等问题，根本无法处理的大量数据，有了很好的解决方案。有了Spark的DataFrame，就不再依赖于pandas的DataFrame，节约了大量的内存和算力。而这样的数据在今后将逐渐成为分析的主流。

Spark也有缺点，虽然它能够完成一些基本的深度学习算法，只是现在Spark中的算法类型不能做到像R一样全面，更新虽快，更多的也并非统计计算方法。对于数据方向的工作者而言，Spark还无法做到像python与R那样使用起来得心应手。

本文在进行单机测验时，对于单机上pyspark的速度也十分感慨，在数据透视时，如果使用pandas则会十分缓慢，但Spark只要将数据存入内存，就能飞速运行。不过对应模型的估算时间比较方面，小样本上时间差距并不大，因为电脑内存都能支持存储全部数据。可是单机上处理时，不方便在于在运行程序时不稳定，且电脑作为终端应该解放出生产力进行其他工作，不如服务器的代码托管工作。

单机上也有multipleprocessing，也可以实现伪分布式，当前计算机也大多拥有一定的内存，足够进行一些小规模的数据处理。但我们应该明白，Spark提供的不仅仅是代码托管，是一种全新的数据存储形式和处理形式，分布式改变了编程的思路，我们从面向对象的编程转变为面向平台的编程，哪怕是其他工具，都有分布式的思想贯穿始终。而这，和Spark、Hadoop一样，都为我们在处理海量数据时提供工具与抓手，这是Spark这些工具最大的功劳。

### 附录 对本课程的小建议

分布式计算是一门专业性很强的课程。分布式平台作为统计计算的载体，显得尤为重要。在课程中，我学习了分布式平台的使用方法，加上每周的任务驱动，给我很大的动力去学习。虽然有时任务有些困难，需要很长时间完成，但是总的来说确实学到了很多，是很有质量的投入。其实同学们上讲台展示自己的代码，我偶尔会有跟不上他们思路的时候。不仔细看代码的话，能学到的东西有限。我建议以后的展示可以将代码共享，这样我们就算上课展示没能跟上，课后自己钻研也能省时省力。

不过大家的Github和博客我也有关注，老师的主页我也是常客（当然都有点star和fork），我自己也写了这门课的课程笔记博客，老师还鼓励了我，让我备受鼓舞。这门课给我思想带来了前所未有的启示，是研一上学期最好的一门课。统计与计算机的界限不是那么分明，在计算机领域，统计学还应该得到进一步发展。

还有一个小建议，学校的服务器不是很行，这门课的期末作业如果可以多人完成，那样可能在服务器方面能轻松一些，哈哈，只不过有可能有人搭便车，是个问题。听说学校实验室有新的设备，后面的同学或许有福了。