Spark 实习报告

薛飞跃 1700017831

一．实习目标

a). Spark RDD：对给出的莎士比亚文集 Shakespere.txt 进行 wordcount（注：文件中包含特殊字符，请先进行过滤操作仅留下英文字符）
b). Spark SQL：在 tmdb 数据上实现查询功能。
c). Spark MLlib：使用 TitanicTrainTest.zip 中的训练集训练一个分类模型(比如决策树)，并且给出在测试集上的正确率。

二．算法步骤

wordcound:

1. 将文本读入，用空格切词
2. 用正则表达式过滤掉含非英文字母的单词
3. 将 word 映射成（word，1）的形式再用 reduceByKey 实现计数
4. sortBy 函数按计数从大到小排列，并 take 前 100 个。

```
import re
def f(word):
    if re.match(r'[a-z]+',word,re.I):
        return True
    else:
        return False

import pyspark
spark = pyspark.sql.SparkSession.builder.appName("SimpleApp").getOrCreate()
sc = spark.sparkContext
textFile = sc.textFile("file:///home/xuefeiyue/下载/Shakespeare.txt")
wordcount = textFile.flatMap(lambda line : line.split(" ")).filter(lambda x:f(x)).\
           map(lambda word : (word,1)).reduceByKey(lambda a, b : a + b).sortBy(lambda x: x[1],False)

print(wordcount.take(100))
#wordcount.collect()
```

```
================== RESTART: /home/xuefeiyue/文档/sparktest.py ==================
[('the', 43557), ('I', 34738), ('and', 32887), ('of', 28633), ('to', 27410), ('a
', 22469), ('my', 18898), ('in', 17123), ('you', 16242), ('is', 13892), ('that',
 13103), ('not', 12086), ('And', 11869), ('with', 11687), ('your', 10939), ('his
', 10787), ('be', 10697), ('for', 10283), ('have', 9338), ('it', 8682), ('he', 8
516), ('this', 8508), ('me', 8354), ('The', 7976), ('as', 7619), ('will', 7344),
 ('thou', 7266), ('but', 6454), ('thy', 6005), ('To', 5533), ('shall', 5444), ('
him', 5404), ('so', 5280), ('are', 5178), ('all', 5157), ('her', 5079), ('scene'
, 5030), ('by', 5024), ('do', 4987), ('our', 4825), ('That', 4657), ('we', 4616)
, ('no', 4335), ('on', 4208), ('at', 3988), ('from', 3967), ('what', 3915), ('Ac
t', 3910), ('Scene', 3878), ('good', 3792), ('if', 3668), ('A', 3639), ('But', 3
625), ('would', 3580), ('am', 3548), ('was', 3408), ('Enter', 3378), ('they', 33
75), ('their', 3322), ("I'll", 3134), ('or', 3047), ('she', 3019), ('thee', 2906
), ('you,', 2900), ('hath', 2832), ('KING', 2826), ('more', 2824), ('For', 2742)
, ('an', 2742), ('What', 2731), ('My', 2725), ('Shakespeare', 2658), ('homepage'
, 2652), ('like', 2649), ('You', 2580), ('make', 2576), ('than', 2562), ('As', 2
538), ('Next', 2527), ('If', 2494), ('Previous', 2458), ('let', 2436), ('one', 2
413), ('should', 2399), ('upon', 2377), ('were', 2366), ('may', 2342), ('must',
2336), ('which', 2314), ('me,', 2308), ('know', 2294), ('them', 2252), ('had', 2
218), ('did', 2200), ('such', 2155), ('when', 2124), ('now', 2113), ('He', 2024)
, ('love', 1965), ('come', 1936)]
```

tmdb 查询:

1.读取文件转化为 dataframe 格式

```
import csv
import pyspark
spark = pyspark.sql.SparkSession.builder.appName("SimpleApp").getOrCreate()
file="/home/xuefeiyue/下载/tmdb-5000-movie-dataset(1)/tmdb-5000-movie-dataset/tmdb_5000_movies.csv"
with open(file) as f:
    reader=csv.DictReader(f)
    df=spark.createDataFrame(reader)
```

2.查询投票数超过 100 的电影中分数最高的前二十名电影

```
>>> df.select('title', 'vote_average', 'vote_count').filter(df.vote_count>100).sort('vote_average',ascending=False).show()
```

```
+--------------------+------------+----------+
|               title|vote_average|vote_count|
+--------------------+------------+----------+
|The Shawshank Red...|         8.5|      8205|
|       The Godfather|         8.4|      5893|
|    Schindler's List|         8.3|      4329|
|The Godfather: Pa...|         8.3|      3338|
|        Pulp Fiction|         8.3|      8428|
|          Fight Club|         8.3|      9413|
|            Whiplash|         8.3|      4254|
|       Spirited Away|         8.3|      3840|
|      The Green Mile|         8.2|      4048|
|     The Dark Knight|         8.2|     12002|
|Once Upon a Time ...|         8.2|      1069|
|              Psycho|         8.2|      2320|
|        Forrest Gump|         8.2|      7927|
|          GoodFellas|         8.2|      3128|
|Howl's Moving Castle|         8.2|      1991|
|   Princess Mononoke|         8.2|      1983|
|  American History X|         8.2|      3016|
|       Seven Samurai|         8.2|       878|
|The Empire Strike...|         8.2|      5879|
|One Flew Over the...|         8.2|      2919|
+--------------------+------------+----------+
only showing top 20 rows
```

3. 查询投票数超过一千的电影中，收益预算比最高的二十部电影。

```
>>> df.select('title', 'vote_average', 'vote_count',df.revenue / df.budget).filter(df.vote_count>1000).sort('(revenue / budget)',ascending=False).show()
```

```
+--------------------+------------+----------+-----------------+
|               title|vote_average|vote_count|(revenue / budget)|
+--------------------+------------+----------+-----------------+
| Paranormal Activity|         5.9|      1316|12890.386666666667|
|The Blair Witch P...|         6.3|      1055| 4133.333333333333|
|               Bambi|         6.8|      1405|  311.709965034965|
|            Mad Max|         6.6|      1213|            250.0|
|          Halloween|         7.4|      1035|233.33333333333334|
|Snow White and th...|         6.9|      1914|124.24256142239135|
|               Rocky|         7.5|      1791|        117.235147|
|                Saw|         7.2|      2184|        86.5930575|
|E.T. the Extra-Te...|         7.3|      3269| 75.51529085714286|
|           Star Wars|         8.1|      6624| 70.49072790909091|
|                Jaws|         7.5|      2542| 67.23628571428571|
|          Insidious|         6.8|      1737| 64.67276666666666|
|        Unfriended|         5.5|      1047|          62.88209|
|       The Exorcist|         7.5|      2005|      55.163268125|
|      The Godfather|         8.4|      5893|40.844401833333336|
|             Psycho|         8.2|      2320|39.655591190510414|
|          Annabelle|         5.6|      1517|39.272894307692304|
|              Saw II|         6.3|      1251|       38.23127325|
|One Flew Over the...|         8.2|      2919| 36.32709166666667|
|       Pretty Woman|         7.0|      1746| 33.07142857142857|
+--------------------+------------+----------+-----------------+
only showing top 20 rows
```

titanic 分类预测:

1. 导入数据集，查看前二十条记录，查看数据总数

```
import pyspark
spark = pyspark.sql.SparkSession.builder.appName("SimpleApp").getOrCreate()

df = spark.read.csv(r"file:///home/xuefeiyue/下载/Titanic/TitanicData.csv", encoding='gbk', header=True, inferSchema=True)
```

```
>>> df.show()
```

```
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|PassengerId|Survived|Pclass|                Name|   Sex| Age|SibSp|Parch|          Ticket|   Fare|Cabin|Embarked|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
|          1|       0|     3|Braund, Mr. Owen ...|  male|22.0|    1|    0|       A/5 21171|   7.25| null|       S|
|          2|       1|     1|Cumings, Mrs. Joh...|female|38.0|    1|    0|        PC 17599|71.2833|  C85|       C|
|          3|       1|     3|Heikkinen, Miss. ...|female|26.0|    0|    0|STON/O2. 3101282|  7.925| null|       S|
|          4|       1|     1|Futrelle, Mrs. Ja...|female|35.0|    1|    0|          113803|   53.1| C123|       S|
|          5|       0|     3|Allen, Mr. Willia...|  male|35.0|    0|    0|          373450|   8.05| null|       S|
|          6|       0|     3|    Moran, Mr. James|  male|null|    0|    0|          330877| 8.4583| null|       Q|
|          7|       0|     1|McCarthy, Mr. Tim...|  male|54.0|    0|    0|           17463|51.8625|  E46|       S|
|          8|       0|     3|Palsson, Master. ...|  male| 2.0|    3|    1|          349909| 21.075| null|       S|
|          9|       1|     3|Johnson, Mrs. Osc...|female|27.0|    0|    2|          347742|11.1333| null|       S|
|         10|       1|     2|Nasser, Mrs. Nich...|female|14.0|    1|    0|          237736|30.0708| null|       C|
|         11|       1|     3|Sandstrom, Miss. ...|female| 4.0|    1|    1|         PP 9549|   16.7|   G6|       S|
|         12|       1|     1|Bonnell, Miss. El...|female|58.0|    0|    0|          113783|  26.55| C103|       S|
|         13|       0|     3|Saundercock, Mr. ...|  male|20.0|    0|    0|       A/5. 2151|   8.05| null|       S|
|         14|       0|     3|Andersson, Mr. An...|  male|39.0|    1|    5|          347082| 31.275| null|       S|
|         15|       0|     3|Vestrom, Miss. Hu...|female|14.0|    0|    0|          350406| 7.8542| null|       S|
|         16|       1|     2|Hewlett, Mrs. (Ma...|female|55.0|    0|    0|          248706|   16.0| null|       S|
|         17|       0|     3|Rice, Master. Eugene|  male| 2.0|    4|    1|          382652| 29.125| null|       Q|
|         18|       1|     2|Williams, Mr. Cha...|  male|null|    0|    0|          244373|   13.0| null|       S|
|         19|       0|     3|Vander Planke, Mr...|female|31.0|    1|    0|          345763|   18.0| null|       S|
|         20|       1|     3|Masselmani, Mrs. ...|female|null|    0|    0|            2649|  7.225| null|       C|
+-----------+--------+------+--------------------+------+----+-----+-----+----------------+-------+-----+--------+
only showing top 20 rows
```

```
>>> df.count()
```

```
891
```

2. Cabin 列大多是 null，去掉此列。计算所有列的 null 所占的百分比。age 是连续型，可以用均值填充。embarked 是离散型，用众数填充。检验填充后的所有列，null 占比都是 0

```
>>> df = df.drop('cabin')

>>> import pyspark.sql.functions as fn

>>> df.agg(*[(1-(fn.count(c) /fn.count('*'))).alias(c) for c in df.columns]).show()
```

```
+-----------+--------+------+----+---+-------------------+-----+-----+------+----+--------------------+
|PassengerId|Survived|Pclass|Name|Sex|                Age|SibSp|Parch|Ticket|Fare|            Embarked|
+-----------+--------+------+----+---+-------------------+-----+-----+------+----+--------------------+
|        0.0|     0.0|   0.0| 0.0|0.0|0.19865319865319864|  0.0|  0.0|   0.0| 0.0|0.002244668911335568|
+-----------+--------+------+----+---+-------------------+-----+-----+------+----+--------------------+
```

```
>>> agg_mean = round(df.select(fn.mean('age')).collect()[0][0],0)

>>> df.groupby('embarked').count().sort('count',ascending=False).show()
```

```
+--------+-----+
|embarked|count|
+--------+-----+
|       S|  644|
|       C|  168|
|       Q|   77|
|    null|    2|
+--------+-----+
```

```
>>> df = df.fillna({'age':agg_mean,'embarked':'S'})

>>> df.agg(*[(1-(fn.count(c) /fn.count('*'))).alias(c) for c in df.columns]).show()
```

```
+-----------+--------+------+----+---+---+-----+-----+------+----+--------+
|PassengerId|Survived|Pclass|Name|Sex|Age|SibSp|Parch|Ticket|Fare|Embarked|
+-----------+--------+------+----+---+---+-----+-----+------+----+--------+
|        0.0|     0.0|   0.0| 0.0|0.0|0.0|  0.0|  0.0|   0.0| 0.0|     0.0|
+-----------+--------+------+----+---+---+-----+-----+------+----+--------+
```

3. 将数据划分为训练集和验证集。Sex 和 Embarked 属性都是字符串的类别数据，转化为数值型的 index

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer


(df,dftest)=df.randomSplit((0.8,0.2))
indexers = [StringIndexer(inputCol=column, outputCol=column+"_index") for column in ['Sex','Embarked']]


pipeline = Pipeline(stages=indexers).fit(df)
df = pipeline.transform(df)
dftest=pipeline.transform(dftest)

df=df.drop('Sex','Embarked')
dftest=dftest.drop('Sex','Embarked')
```

4. 将 dataframe 格式转化为 labeledpoint 格式，导入决策树模型，在训练记上拟合模型，在测试集上给出预测，计算准确率。

```
from pyspark.mllib.regression import LabeledPoint
volume = df.rdd.map(lambda row: [e for e in row])
train = volume.map(lambda row: LabeledPoint(row[0], row[1:]))

volume = dftest.rdd.map(lambda row: [e for e in row])
test = volume.map(lambda row: LabeledPoint(row[0], row[1:]))

from pyspark.mllib.tree import DecisionTree

model = DecisionTree.trainClassifier(train, numClasses=2,categoricalFeaturesInfo={});
y_pred = model.predict(test.map(lambda row: row.features)).collect()
y_ture = test.map(lambda row: row.label).collect()

acc=len([1 for i in range (len(y_ture)) if y_pred[i]==y_ture[i]])/len(y_ture)
print("accuracy:",acc)
```

```
accuracy: 0.8206521739130435
```

三．心得体会

1. tmdb 数据尝试使用 spark.read_csv 函数读取，发现该函数仅以逗号作为分隔符分割，而原始的 csv 中的数据项存在列表，列表元素也是以逗号分隔，spark.read_csv 会把列表分割开来造成错误。
2. spark 中 rdd，dataframe 和 labeledpoint 格式各有其使用方法，应该注意区分。
3. spark 中的 dataframe 和 pandas 中的 dataframe 有很多相似之处，但也有很多不同，可以互相转换。MLlib 也和 sklearn 有很多相似之处。