关于使用 Spark 预测用户流失的分析

项目介绍

Spark 是美国数字音乐服务公司，大量用户在服务器上听歌，无论是免费用户还是付费 用户，只要在平台上进行了任何听歌、点赞、交友互动，注销用户的活动，都会产生数据。 为了可以预测分析用户的流失情况，会运用这个完整的数据集的迷你子集，用于进行可视化 分析和预测。

问题概述

本次项目想通过 Spark 来处理数据，由于预测用户流失的问题属于二元分类的问题，所以计划用机器学习分类的算法对用户流失进行预测，打算用逻辑回归、随机森林 和 GBT 三种模型对数据进行分类，然后通过 F1 值的大小来判断模型。

模型选择

预期解决方案：探索性分析主要打算根据已有的数据，定义流失客户，在根据对于数据的理解，结合商业活动中找出与之觉得与之相关的变量，建立特征工程。同时选择模型对数据进行训练，并通过性能评价指标选择合适的模型预测客户流失。

评价指标：

	Positive	Negative
True	TP	FP
False	FN	TN

模型常用的评价指标是准确率（Accuracy）、召回率（Recall）、精确率（Precision）和 F1 Score。

准确率（Accuracy）：所有的样本中预测正确的比例，准确率 =(TP+TN)/(TP+TN+FP+FN)

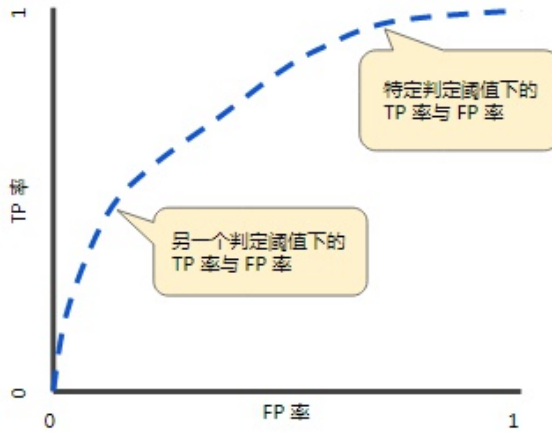
召回率（Recall）：也称查全率、敏感度：所有正样本中预测正确的比例，即正样本的准确率，即在所有确实为真的样本中，被判为的 “真” 的占比。召回率= TP/(TP+FN)

精确率（Precision）：也称查准率：所有预测为正样本的集合中预测正确的比例。即所有系统判定的“真”的样本中，确实是真的的占比。精确率=TP/(TP+FP)

F1 Score：综合精确率和召回率指标。2*precision*recall / (precision + recall)

同时

ROC 曲线中的主要两个指标就是真正率和假正率，上面也解释了这么选择的好处所在。其中横坐标为假正率（FPR），纵坐标为真正率（TPR），下面就是一个标准的 ROC 曲线图。



为了计算 ROC 曲线上的点，我们可以使用不同的分类阈值多次评估逻辑回归模型，但这样做效率非常低。幸运的是，有一种基于排序的高效算法可以为我们提供此类信息，这种算法称为**曲线下面积（Area Under Curve）**。

比较有意思的是，如果我们连接对角线，它的面积正好是 0.5。对角线的实际含义是：**随机判断响应与不响应，正负样本覆盖率应该都是 50%，表示随机效果。**ROC 曲线越陡越好，所以理想值就是 1，一个正方形，而最差的随机判断都有 0.5，所以一般 AUC 的值是介于 0.5 到 1 之间的。

根据这些指标的定义，考虑到 F1 的分数是用以衡量二次分类模型精确度的指标，同时也反应了模型的精确率和召回率。这些指标可以帮我们更加精确的预测流失用户，减少将留存用户预测为流失用户的概率，因此选择用 F1 分数为主来对模型指标进行判断。

加载和清洗数据

在拿到数据后，由于样本内存过于庞大，肉眼是无法分析理解数据的。因此我们可以使 用变成语言进行分析，了解数据库大致的情况，比如由多少记录，有多少变量，有哪些明显 的错误，是否存在空值等等。

```
spark = SparkSession.builder.appName("Sparkify").getOrCreate()
df = spark.read.json('mini_sparkify_event_data.json')

df.printSchema()
```

```
root
 |-- artist: string (nullable = true)
 |-- auth: string (nullable = true)
 |-- firstName: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- itemInSession: long (nullable = true)
 |-- lastName: string (nullable = true)
 |-- length: double (nullable = true)
 |-- level: string (nullable = true)
 |-- location: string (nullable = true)
 |-- method: string (nullable = true)
 |-- page: string (nullable = true)
 |-- registration: long (nullable = true)
 |-- sessionId: long (nullable = true)
 |-- song: string (nullable = true)
 |-- status: long (nullable = true)
 |-- ts: long (nullable = true)
 |-- userAgent: string (nullable = true)
 |-- userId: string (nullable = true)
```

在现实中，有问题的数据是非常普遍的，这要求分析数据的相关人员，用很强的耐心去 研究和探索。比如如下观察:

清洗数据

- 查看是否有空值，将有空值的所有行都删除
- location是城市和州组成的，需要分开
- registration是注册时间，ts是时常，应该将其改成时间类型

根据这些观察，并提出的问题，然后运用编程语言对数据库进行修正。 1. 处理空值: 用 for 的条件语句，自动查找存在空值的列。

```
# 查看是否有空值
for col in df.columns:
    missing_count = df.filter((df[col] == "") | df[col].isNull() | isnan(df[col])).count()
    print('{}: '.format(col), missing_count)
```

```
artist:  58392
auth:  0
firstName:  8346
gender:  8346
itemInSession:  0
lastName:  8346
length:  58392
level:  0
location:  8346
method:  0
page:  0
registration:  8346
sessionId:  0
song:  58392
status:  0
ts:  0
userAgent:  8346
userId:  8346
```

这里由于样本数量巨大，所以对这里的空值所在的行直接删除，但有的时候由于一些变量之间可能存在很大的相关性，所以可以用均值或编写条件语句进行填充。

1. 行列数据唯一性

为了后续更好的对数据进行分析，所以如果一个单元格包含多种信息，是需要进行提取分开的。因此这里的 location，城市和州的信息在一个单元格里，不利于分析，因此将其分开成两列。

```
# 将location里的城市和州的名字分开，并删除location一列
split_col = fn.split(dfx['location'], ',')

dfx = dfx.withColumn('City', split_col.getItem(0))
dfx = dfx.withColumn('State', split_col.getItem(1))
dfx = dfx.drop('location')
```

2. 调整数据类型

Registration 是属性的不符合，如果变成时间属性，可以方便后面的加减，如果后续涉及到时间顺序的分析其他分析，可以更好的进行分析。

```
#更改registeration的数据类型
get_hour = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).hour)
get_day = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).day)
get_month = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).month)
get_weekday = udf(lambda x: datetime.datetime.fromtimestamp(x / 1000.0).strftime('%w'))
```

探索性数据分析

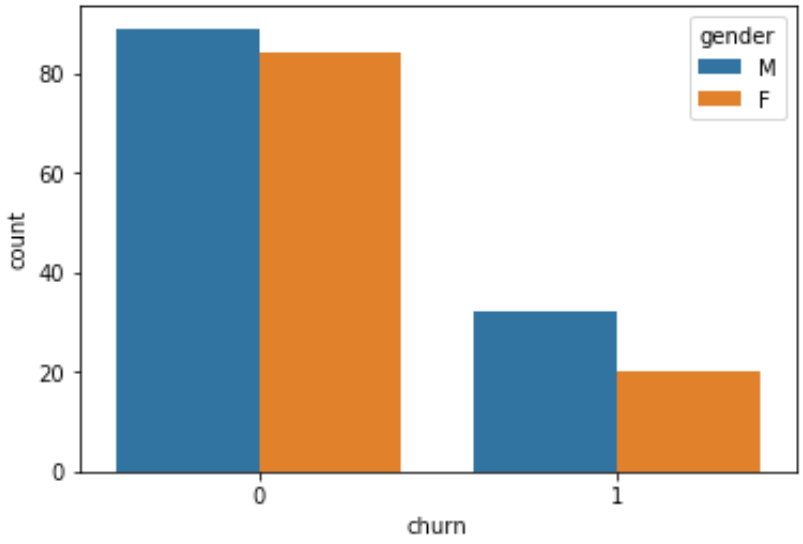
在清理好数据之后，我们就可以开始对数据进行进一步的探索。

1. 定义流失用户

用数据中网页访问 “Cancellation Confirmation” 页面的用户 ID 定义为流失用户。同时为了方便后续的数据计算，将其变成 0 和 1

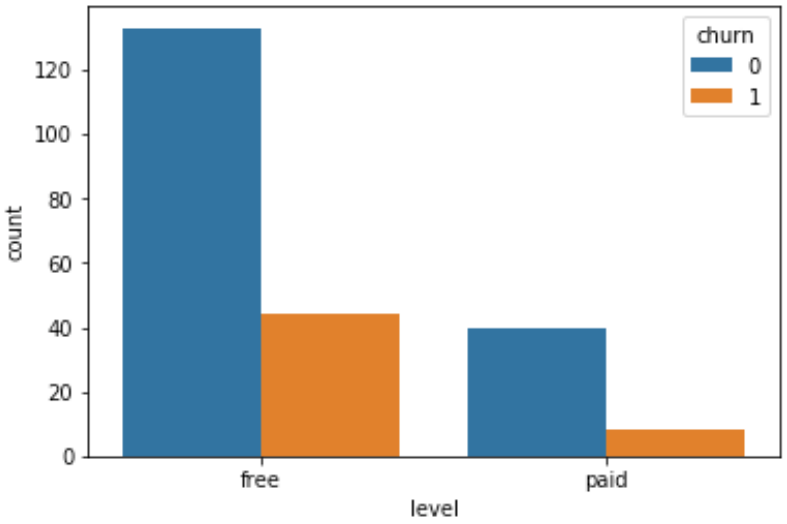
```
# 将访问过cancellation的用户定义客户流失。为了后续的计算，将true 和false变成0和1

churn = df_clean.filter(df.page=="Cancellation Confirmation").select("userId").dropDuplicates()

churn_list = [user["userId"] for user in churn.collect()]

churn_event = udf(lambda x:1 if x in churn_list else 0, IntegerType())

df_clean = df_clean.withColumn("churn", churn_event("userId"))
```

2. 用户流失的性别分布



男性用户比女性用户删除账户的人数更多

3. 用户付费之间和注销用户的关系



免费用户中，注销用户的人数比较多，付费用户的流失用户量比较少

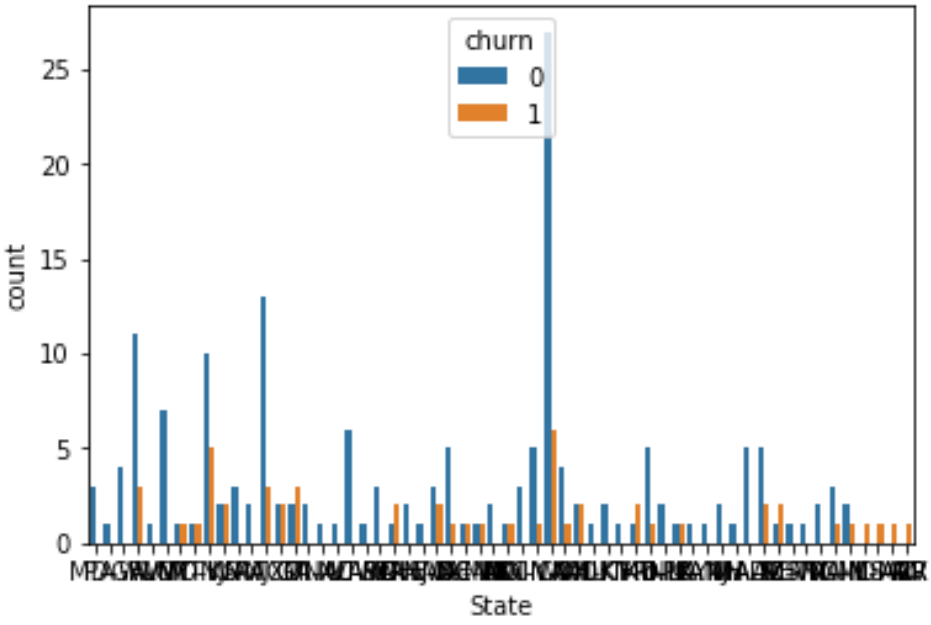
4. 取消收费订阅和免费订阅的数量

```
print(df_clean.filter((df_clean.churn == 1) & (df.level=='paid')).count())

print(df_clean.filter((df_clean.churn == 1) & (df.level=='free')).count())
```

```
32476
12388
```

可以发现，取消收费订阅的用户要远远取消免费订阅的用户

5. 所在州之间的注销用户和留存用户之间的关系



从图表里可得知，CA 这个州的客户流失量最大，其次是 NY 和 FL 这两个州

6. 查看浏览页的分布情况

绝大部分的用户选择 ‘NextSong’ ，其次是点赞或者将歌曲加入各单。从数据层面上可以发现，大部分用户还是对 Spark 是喜爱的。

根据对数据的理解，因为是要预测客户流失，所以选择保留了数据库里的 10 个变量，如下：

在大致的对数据进行分析后，决定保留或者综合数据内容得出如下相关元素：

- 1.userId
- 2.churn
- 3.gender
- 4.level
- 5.ThumbsUp
- 6.ThumbsDown
- 7.AddtoPlaylist
- 8.AddFriend
- 9.songs
- 10.artist

为了后面更好的进行分析，将其都变成数字变量。

因为预测属于二分类模型，因此打算采用逻辑回归、随机森林以及 Gradient Boosted Trees 进行分类预测。

并且分别得出他们的准确率和 F1 score，如下：

	逻辑回归	Random Forest	GBT
Accuracy	0.625	0.5	0.375
F-1 Score	0.48	0.42	0.34

我们最终选择用该逻辑回归模型进行预测，因为 F-1 分越高，说明相比之下假阴性和假阳性更少。另外处理大数据以改进模型时，其实我们也可以考虑添加更加多的指标。 比如滚动广告，注销用户的情况也可以添加到功能中

总结：

此次项目主要是为了预测流失用户，因为是二元分类问题，选用了逻辑回归、随机森立和 GBT 模型进行运算。经过对 F1 分数的测算，逻辑回归效果比较好。这个过程，首选是对数据集进行了分析，然后根据对数据集的探索，寻找了与预测流失用户相关的特征，并进行特征工程。最后建立机器学习模型进行预测。根据结果，其实准确率并不算特别高，分析的原因，可能一是选区变量分布不均衡，在前期处理的时候不细致，导致一些关键数据流失，同时一些参数的设置还需要进一步调整。同时训练集中的用户量较小，其实可以尝试其他的模型进行探索。

在后面的进一步分析，可以加入其他特征进入模型，同时测试不同的参数范围进行多次的测试。同时因为数据集里涉及到时间序列的，也可以在拆分训练集的时候，按照时间进行筛选拆分等方式，对数据进行多维度的探索。

困难：在探索性数据分析方面对于我来说是极大的挑战，如何寻找变量之间的关联性，变量之间是否有较强的关联性，正常用户和流失用户在行为方面的差异等等，都是需要大量的数据去显示，大量的时间去分析。同时在模型选择中，选择什么样的模型，用怎样的方式去检验都是需要查找文献去学习和研究的。但是整体下来，所得到的收获也是令人满意的。

Source：

1. https://blog.csdn.net/Cheese_pop/article/details/78228156
2. https://zhuanlan.zhihu.com/p/46714763
3. https://www.silect.is/blog/2019/4/2/random-forest-in-spark-ml
4. https://zhuanlan.zhihu.com/p/46831267