Big Data Analytics Techniques and Applications

Homework 4

309652022 黃伯丞

A. 環境設定&資料預處理

我在 colab 上使用 pyspark 實作，因為這次作業需要建立模型預測，所以需要調高記憶體的大小。

```
# Import SparkSession
from pyspark.sql import SparkSession
# Create SparkSession
spark=SparkSession.builder.config('spark.executor.memory','8g').config('spark.driver.memory', '20g') .getOrCreate()
```

作業 4 提供的資料集包含 2003~2005 年的航班資料，主要的目的是要由 2003~2004 年的資料去預測 2005 年的每班班機是否發生延遲。在進行預測之前，首先需要把全部的資料串在一起，觀察原本的資料集紀錄了什麼樣的資訊以及是否有資料缺失的情形。當航班的 Cancelled=1 時，代表這趟班機被取消所以不會有起飛、降落延遲的發生，所以要把這些航班去除掉。當航班的 Diverted=1 時，代表這趟班機的起飛地點改變，所以跟起飛有關的資料並未記錄，只有記錄降落延遲(DepDelay)，這類的航班只有 39193 筆，佔不到 1%。整理資料集還需要把多餘的特徵刪除，我設定預測是根據 2005 年的這些班機實際起飛降落時間為未知的，所以把'DepTime', 'ArrTime'去除，另外取消的班機對延遲沒有影響，所以 'Cancelled', 'CancellationCode'要刪掉，與延遲原因有關的這五個特徵 'CarrierDelay', 'WeatherDelay', 'NASDelay', 'SecurityDelay', 'LateAircraftDelay'也不需要。

```
df03=spark.read.csv('drive/MyDrive/HW4/2003.csv',header=True,inferSchema=True)
df04=spark.read.csv('drive/MyDrive/HW4/2004.csv',header=True,inferSchema=True)
df05=spark.read.csv('drive/MyDrive/HW4/2005.csv',header=True,inferSchema=True)

total_data=df03.union(df04).union(df05)

total_data.count()

total_data.filter(total_data.Cancelled==1).show()

total_data.filter(total_data.Diverted==1).count()

#刪除取消沒有資訊的班機、刪除Delay詳細資料
total_data=total_data.filter(total_data.Cancelled==0).drop('DepTime','ArrTime','Cancelled','CancellationCode',\
                             'CarrierDelay','WeatherDelay','NASDelay','SecurityDelay','LateAircraftDelay')
```

接下來要定義出問題明確的目標，不論是起飛發生延遲或降落發生延遲我都歸類在有發生延遲的類別，新增 delay 的 feature，根據延遲於否給定 0(未發生),1(發生)的標籤。預計起飛降落時間的格式是 hhmm，我將小時取出來當作新的特徵(CRSDepH, CRSArrH)。當 Diverted=1 時部分特徵被紀錄為 NA，我將這些 NA 用 0 代替，TailNum 中有缺失值，用 na.fill 填入 unknown。

```
total_data=total_data.withColumn('delay',F.when((total_data.ArrDelay>0)|(total_data.DepDelay>0),1).\
                                 otherwise(0)).\
                      withColumn('CRSDepH',(total_data.CRSDepTime/100).cast('int')).\
                      withColumn('CRSArrH',(total_data.CRSArrTime/100).cast('int')).\
                      drop('ArrDelay','DepDelay','CRSArrTime','CRSDepTime')

total_data=total_data.withColumn('ActualElapsedTime', F.regexp_replace('ActualElapsedTime', 'NA', '0')\
                                 .cast(IntegerType()))\
.withColumn('CRSElapsedTime', F.regexp_replace('CRSElapsedTime', 'NA', '0').cast(IntegerType()))\
.withColumn('AirTime', F.regexp_replace('AirTime', 'NA', '0').cast(IntegerType()))

total_data=total_data.na.fill({'TailNum': 'unknown'})
```

將字串型態的特徵用 StringIndexer 轉成數字 index，並將原本的特徵去除，完成資料預處理。

```
s1=StringIndexer(inputCol='UniqueCarrier',outputCol='UniqueCarrierIndex')
s2=StringIndexer(inputCol='Origin',outputCol='OriginIndex')
s3=StringIndexer(inputCol='Dest',outputCol='DestIndex')
s4=StringIndexer(inputCol='TailNum',outputCol='TailNumIndex')

total_data=Pipeline(stages=[s1,s2,s3,s4]).fit(total_data).transform(total_data)

total_data.printSchema()

total_data=total_data.drop('UniqueCarrier','Origin','Dest','TailNum')
```

B. 問題

I. Q1

在選擇特徵時，我先計算 delay 與其他特徵的相關係數(預設的方法為 Pearson, PCCs)，從中挑出比較好的作為模型使用的特徵。

```
#相關係數
train=total_data.filter(total_data.Year<2005)
for i in range(1,len(train.columns)):
  print(train.columns[i],train.corr('delay',train.columns[i]))
```

```
Month 0.020397243906215008
DayofMonth 0.007798284728275292
DayOfWeek 0.004609385801791708
FlightNum -0.035842038555943335
ActualElapsedTime 0.13586446542081576
CRSElapsedTime 0.0619709343291983
AirTime 0.06649013540585919
Distance 0.060125646525764014
TaxiIn 0.022132798174829556
TaxiOut 0.22925952836995062
Diverted -0.0020212781994442 94
delay 1.0
CRSDepH 0.10953276648022327
CRSArrH 0.11449876049764114
UniqueCarrierIndex 0.009260979018298846
OriginIndex -0.07814842934953298
DestIndex 0.0009411289041247588
TailNumIndex 0.022457517055998806
```

根據相關係數的絕對值挑出 6 個特徵 (TaxiOut, CRSDepH, CRSArrH, ActualElapsedTime, AirTime, OriginIndex)，將這 6 個特徵轉成 mllib 所接受的向量格式作為 features 輸入。接著把整理完成的資料拆成 train 以及 test 兩部分。

```
#Take features with higher corr
total_data=total_data.withColumn('features',F.array(total_data.TaxiOut,total_data.CRSDepH,total_data.CRSArrH,\
                                                    total_data.ActualElapsedTime, total_data.AirTime, total_data.OriginIndex))
list_to_vector_udf=F.udf(lambda l: Vectors.dense(l), VectorUDT())
total_data=total_data.withColumn('features',list_to_vector_udf(total_data['features']))\
            .withColumn('label',total_data.delay)
```

```
train=total_data.select('features','label').filter(total_data.Year<2005)
test=total_data.select('features','label').filter(total_data.Year==2005)
```

我使用 ml 的 LogisticRegression 作為模型，它是一種分類的方法，想要找出一條直線能夠將所有數據分開並做分類，預設的懲罰項為 L2 norm penalty。

II. Q2

我利用 ml 的 CrossValidator 進行 cross-validation，這個套件使用的方法是 K-fold，將訓練集分成 K 等分，利用任意 K-1 份做為訓練，剩下的 1 份驗證，比較模型在這 K 種組合的表現。套件預設的 K=3。

III. Q3

除了預測正確(accuracy)以外，還有其他的 metrics 可以拿來參考。像是 confusion matrix 將預測的標籤與本來的標籤比較，分成 4 類情形(tp, fp, fn, tn)，在這個問題上的 positive 代表發生延遲，negative 代表未發生延遲。從 confusion matrix 得到的四樣數據可以用來計算其他的指標，像是 sensitivity (tp/(tp+fn)), specificity (tn/(tn+fp)) 等。

在參數的選擇上，我利用 BinaryClassificationEvaluator 決定標準來選擇模型，它所預設的標準是 AUC，也就是 ROC 曲線底下的面積。 ROC 曲線的座標空間將偽陽性率 (FPR=fp/(fp+np)) 當作 X 軸，真陽性率 (TPR，與 sensitivity 相同) 當作 Y 軸，FPR 跟 TPR 會隨著 positive, negative 的分布而同時改變，把同一模型的 (FPR, TPR) 座標都畫在 ROC 空間裡，即為該模型的 ROC 曲線。

IV. Q4

調整參數 maxIter 以及 regParam，經過 CrossValidator 測試三次 (K=3) 後得到由 BinaryClassificationEvaluator 所產生的 avgMetrics (平均的 AUC)，選擇最高的作為模型的參數選擇。

```
lr=LogisticRegression()
grid=ParamGridBuilder().addGrid(lr.maxIter,[10,50]).addGrid(lr.regParam,[0.1,0.05,0.01]).build()
evaluator=BinaryClassificationEvaluator()
cv=CrossValidator(estimator=lr,estimatorParamMaps=grid,evaluator=evaluator,parallelism=2)
cvModel=cv.fit(train)

cvModel.avgMetrics
```

maxIter\regParam	0.1	0.05	0.01
10	0.6765471908	0.6817442859	0.6843694784

50	0.6772324041	0.6817087916	0.6838576850

最後模型(maxIter=10, regParam=0.01)所預測的結果如下：

	AOC	tp	fp	fn	tn
Train	0.684	3182274	1655217	3227405	5323688
Test	0.685	1763221	796550	1831581	2615514

可以發現這個模型對於 negative 樣本的判斷比 positive 樣本的準確(特異度高、敏感度低)。

	acc	specificity	sensitivity
Train	63.5%	76.3%	49.6%
Test	62.5%	76.7%	49.0%

C. 討論

資料清理時，有的時候可以從原本的資料中擷取出一部分訊息做為新的特徵(例如從時間中取出小時)，或者將多個特徵合併成一個新的特徵，有機會能讓預測更加準確。

選擇模型時，MLlib 有相當多 Classification 的模型，除了 Logistic regression 之外，還有 Decision tree classifier, Multilayer perceptron classifier 等，可以根據不同演算法嘗試做參數調整。

在 metrics 的部分，如果我們希望壓低偽陽性或是偽陰性的情形，那麼指標的優先度會因此而改變，實際的應用上不一定 accuracy 越高代表越好。另外，如果樣本的比例有懸殊狀況發生時反而 accuracy 不能反映出對較少樣本的預測準確度，此時 sensitivity, specificity 的重要性會變高。