



به نام خداوند بخشنده و مهربان

تمرین چهارم: لاجستیک رگرسیون
درس: تحلیل سیستم داده‌های حجیم

استاد: محمدعلی نعمت‌بخش
دستیاران: فاطمه ابراهیمی، پریسا لطیفی، امیر سرتیپی

نام و نام خانوادگی: مهناز توحیدی‌مهر

آدرس گیت: <https://github.com/mtohidimehr/BigData-hw4-mTohidimehr.git>

در این تمرین هدف کار با کتابخانه‌ی `pyspark` و همچنین کتابخانه‌ی یادگیری ماشین آن است.

برای این منظور دیتاستی در اختیار شما قرار گرفته است. اطلاعات کاربران شرکتی در اختیار شما قرار داده شده است. این شرکت اطلاعات چند ماه از کاربرانش را برچسب گذاری کرده است. این برچسب به معنای این است که آیا مشتری شرکت را ترک کرده و دیگر از خدمات آن استفاده می‌کند یا خیر. انتظار می‌رود با بررسی دقیق مجموعه‌ی داده و تحلیل داده‌گان آن در نهایت مدل پیشبینی کننده‌ای برای این شرکت طراحی کنید.

قدم اول: دیتاست داده شده را پیش پردازش کنید. مقادیر `NA` را مقدار دهی کنید تحلیل داده اکتشافی (`EDA`) را به خوبی انجام دهید. این ستونها براساس ماهیت خود میتواند تولید کننده ویژگیهای بیشتری باشند که ممکن است دقت مدل شما را بالاتر ببرند. در این مرحله همبستگی و ارتباط بین تمام ویژگی هایی که میتوانید استخراج کنید را بررسی کنید. (نمودارهای لازم برای تحلیل دادگان ترسیم شود).

بعد از نصب `pyspark`، با استفاده از `SparkSession` یک برنامه به نام `BigData-HW4` ایجاد می‌کنیم.

✓
8s

```
import numpy as np
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql import Row
spark = SparkSession \
    .builder \
    .appName("BigData-HW4") \
    .getOrCreate()
```

فایل data.csv مورد نظر را از آدرسی که در آن آپلود شده است، خوانده و در دیتافریم CustomerInfo قرار می دهیم. در شکل زیر قسمتی از این دیتافریم به همراه ستون ها نشان داده شده است.

```
✓ 1s ▶ CustomerInfo = spark.read.option("header",True) \
        .csv('/content/sample_data/data.csv')
CustomerInfo.show()
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
5331-RGTT	Male	1.0	Yes	No	54.0	Yes	Yes
5161-XEUVX	Male	0.0	Yes	No	37.0	Yes	Yes
0336-PIKEI	Male	1.0	Yes	No	72.0	Yes	No
3345-PBBFH	Male	0.0	Yes	No	8.0	Yes	No
5067-XJQFU	Male	1.0	Yes	Yes	66.0	Yes	Yes
4056-QHXHZ	Female	0.0	Yes	Yes	72.0	Yes	Yes
8028-PNXHQ	Male	0.0	Yes	Yes	62.0	Yes	Yes
8181-YHCFM	Female	0.0	Yes	Yes	68.0	No	No phone service
6734-PSBAW	Male	0.0	Yes	No	72.0	Yes	Yes
3655-SNQYZ	Female	0.0	Yes	Yes	69.0	Yes	Yes
9508-ILZDG	Female	1.0	No	No	34.0	Yes	Yes
3620-EHIMZ	Female	0.0	Yes	Yes	52.0	Yes	No
4178-EGMON	Male	0.0	Yes	No	70.0	Yes	Yes
0036-IHMOT	Female	0.0	Yes	Yes	55.0	Yes	No

با استفاده از دستور زیر تعداد کل رکوردها مشخص می شود.

```
✓ 0s [19] CustomerInfo.count()


228605
```

ستون های این دیتا فریم به صورت زیر است:

```
✓ 0s [10] CustomerInfo.columns

['customerID',
 'gender',
 'SeniorCitizen',
 'Partner',
 'Dependents',
 'tenure',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod',
 'MonthlyCharges',
 'TotalCharges',
 'Label']
```

با دستور زیر تعداد missing-value ها در هر ستون مشخص می شود. همانطور که پیداست، ۲۰۷ رکورد فاقد label می باشند.

10s 

CustomerInfo.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in CustomerInfo.columns]).show()

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
253	232	264	224	242	224	268	260	229	230	241	251	262
StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Label					
247	218	230	256	245	242	251	207					

برای انجام تغییرات روی دیتافریم یک کپی از آن می گیریم تا دیتافریم اصلی بدون تغییر باقی بماند. با استفاده از دستور printSchema نوع داده ای هر ستون مشخص خواهد شد. با توجه به خروجی تمام داده ها به صورت string هستند.

```
MyCustomerInfo = CustomerInfo
MyCustomerInfo.printSchema()

root
  |-- gender: string (nullable = true)
  |-- SeniorCitizen: string (nullable = true)
  |-- Partner: string (nullable = true)
  |-- Dependents: string (nullable = true)
  |-- tenure: string (nullable = true)
  |-- PhoneService: string (nullable = true)
  |-- MultipleLines: string (nullable = true)
  |-- InternetService: string (nullable = true)
  |-- OnlineSecurity: string (nullable = true)
  |-- OnlineBackup: string (nullable = true)
  |-- DeviceProtection: string (nullable = true)
  |-- TechSupport: string (nullable = true)
  |-- StreamingTV: string (nullable = true)
  |-- StreamingMovies: string (nullable = true)
  |-- Contract: string (nullable = true)
  |-- PaperlessBilling: string (nullable = true)
  |-- PaymentMethod: string (nullable = true)
  |-- MonthlyCharges: string (nullable = true)
  |-- TotalCharges: string (nullable = true)
  |-- Label: string (nullable = true)
```

می‌خواهیم نوع داده‌ای ستون‌های tenure، MonthlyCharges و TotalCharges را به float تبدیل نماییم.

```
0s ▶ MyCustomerInfo = MyCustomerInfo.withColumn("tenure",MyCustomerInfo['tenure'].cast('float'))
MyCustomerInfo = MyCustomerInfo.withColumn("MonthlyCharges",MyCustomerInfo['MonthlyCharges'].cast('float'))
MyCustomerInfo = MyCustomerInfo.withColumn("TotalCharges",MyCustomerInfo['TotalCharges'].cast('float'))
MyCustomerInfo.printSchema()

root
|-- customerId: string (nullable = true)
|-- gender: string (nullable = false)
|-- SeniorCitizen: string (nullable = false)
|-- Partner: string (nullable = false)
|-- Dependents: string (nullable = false)
|-- tenure: float (nullable = true)
|-- PhoneService: string (nullable = false)
|-- MultipleLines: string (nullable = false)
|-- InternetService: string (nullable = false)
|-- OnlineSecurity: string (nullable = false)
|-- OnlineBackup: string (nullable = false)
|-- DeviceProtection: string (nullable = false)
|-- TechSupport: string (nullable = false)
|-- StreamingTV: string (nullable = false)
|-- StreamingMovies: string (nullable = false)
|-- Contract: string (nullable = false)
|-- PaperlessBilling: string (nullable = false)
|-- PaymentMethod: string (nullable = false)
|-- MonthlyCharges: float (nullable = true)
|-- TotalCharges: float (nullable = true)
|-- Label: string (nullable = false)
```

برخورد با مقادیر Null :

همانطور که دیدیم در همه ستون‌ها مقادیر Null وجود داشت. این ستون‌ها دو حالت دارند:

حالت اول : ستون‌های Categorical

اغلب ستون‌های دیتافریم مورد نظر از این نوع هستند. برای پر کردن مقادیر Null در این ستون‌ها، مقداری که بیشترین تکرار دارد را به عنوان mode حساب کرده و در فیلدهای بدون مقدار آن ستون قرار می‌دهیم.

حالت دوم: ستون‌های Numerical

در این دیتافریم سه ستون با این ویژگی وجود دارد، که برای پر کردن مقادیر Null در آن‌ها از مقدار میانگین عددی مقادیر ستون استفاده می‌کنیم. ['TotalCharges' , 'MonthlyCharges' , 'tenure']

```

from pyspark.sql.types import IntegerType

mean_Tenure = MyCustomerInfo.agg({'tenure': 'mean'})
mean_Tenure = str((float(int(mean_Tenure.collect()[0][0]))))

mean_MonthlyCharges = MyCustomerInfo.agg({'MonthlyCharges': 'mean'})
mean_MonthlyCharges = mean_MonthlyCharges.withColumn("avg(MonthlyCharges)" \
, round(mean_MonthlyCharges["avg(MonthlyCharges)"], 2))
mean_MonthlyCharges = str(float(mean_MonthlyCharges.collect()[0][0]))

mean_TotalCharges = MyCustomerInfo.agg({'TotalCharges': 'mean'})
mean_TotalCharges = mean_TotalCharges.withColumn("avg(TotalCharges)" \
, round(mean_TotalCharges["avg(TotalCharges)"], 2))
mean_TotalCharges = str(float(mean_TotalCharges.collect()[0][0]))

MyCustomerInfo = MyCustomerInfo.fillna({'tenure': mean_Tenure \
, 'MonthlyCharges': mean_MonthlyCharges \
, 'TotalCharges': mean_TotalCharges})

for c in MyCustomerInfo.columns:
    if( c not in ['customerID', 'tenure', 'MonthlyCharges', 'TotalCharges'] ):
        mode_c = MyCustomerInfo.groupBy(c).count()
        mode_c = mode_c.orderBy(col('count').desc())
        mode_c = mode_c.collect()[0][0]
        MyCustomerInfo = MyCustomerInfo.fillna({c : mode_c})

```

همانطور که در جدول زیر مشخص است به جز ستون customerID ، مقادیر null در ستون‌های دیگر با مقدار مناسب پر شدند.

```

MyCustomerInfo.select([count(when(isnan(c) | col(c).isNull() , c)).alias(c) for c in MyCustomerInfo.columns]).show()

```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
253	0	0	0	0	0	0	0	0	0	0

DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Label
0	0	0	0	0	0	0	0	0	0

تحلیل ستون‌ها:

قبل از تحلیل ستون‌ها برای استفاده در مدل، ستون‌های Categorical را به نوع عددی تبدیل می‌کنیم. اینکار با استفاده از stringIndexer صورت می‌گیرد. بعد از تبدیل این ستون‌ها به index برای استفاده از آن‌ها در مدل با OneHotEncoder آن‌ها را به فرمت برداری درمی‌آوریم. در این قسمت توجه داریم که ستون Label جدا از دیگر ستون‌ها index شود. چون در قسمت تبدیل به بردار و ترکیب بردارها به features برای آموزش مدل، این ستون نباید حضور داشته باشد.

```
[88] from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler

num_col = ['tenure', 'MonthlyCharges', 'TotalCharges']
cat_col = ['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines', 'InternetService', \
           'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', \
           'Contract', 'PaperlessBilling', 'PaymentMethod']

indexers = [StringIndexer(inputCol=column, outputCol="{0}_indexed".format(column)).fit(MyCustomerInfo) for column in cat_col ]
label_indexer = StringIndexer(inputCol='Label', outputCol= "Label_indexed")
encoders = [OneHotEncoder(droptLast=False, inputCol = indexer.getOutputCol() \
                          , outputCol= "{0}_encoded".format(indexer.getOutputCol())) for indexer in indexers]

assembler = VectorAssembler(inputCols=[encoder.getOutputCol() for encoder in encoders]+ num_col, outputCol="features")

pipeline = Pipeline(stages=indexers + encoders+[assembler]+ [label_indexer])
model = pipeline.fit(MyCustomerInfo)
MyCustomerInfo_tr = model.transform(MyCustomerInfo)
#MyCustomerInfo_tr = MyCustomerInfo_tr.withColumn('Label_indexed',)
MyCustomerInfo_tr.show()
```

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
5331-RGMTT	Male	1.0	Yes	No	54.0	Yes	Yes	Fiber optic	No	No
5161-XEUVX	Male	0.0	Yes	No	37.0	Yes	Yes	Fiber optic	No	No
0336-PIKEI	Male	1.0	Yes	No	72.0	Yes	No	DSL	Yes	Yes
3345-PBBFH	Male	0.0	Yes	No	8.0	Yes	No	DSL	No	Yes
5067-XJOFU	Male	1.0	Yes	Yes	66.0	Yes	Yes	Fiber optic	No	Yes

ستون اول: customerID

این ستون، شناسه یکتایی است که در آموزش مدل از آن استفاده نمی‌کنیم زیرا در نتیجه پیش بینی اثر مثبتی نخواهد داشت.

ستون دوم: gender

با توجه به نتایج زیر و مقدار همبستگی، جنسیت تاثیر چندانی در مقدار لیبل ندارد پس می‌توانیم در مراحل بعد آن را حذف نماییم.

```
gender_dependency = MyCustomerInfo_tr.groupBy('gender', 'Label').count()
gender_dependency.show()
print('correlation between gender and Label:')
MyCustomerInfo_tr.corr('gender_indexed', 'Label_indexed', 'pearson')
```

```
+-----+-----+-----+
|gender|Label|count|
+-----+-----+-----+
|Male|No|98693|
|Male|Yes|17845|
|Female|No|97393|
|Female|Yes|16059|
+-----+-----+-----+
```

```
correlation between gender and Label:
-0.016326515705542194
```

```
[47] MyCustomerInfo_tr.drop('gender', 'gender_indexed', 'gender_indexed_end=coded').show()
```

customerID	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	
5331-RGMTT	1.0	Yes	No	54.0	Yes	Yes	Fiber optic	
5161-XEUVX	0.0	Yes	No	37.0	Yes	Yes	Fiber optic	

ستون سوم: SeniorCitizen

برای این ستون نیز وابستگی با Label را محاسبه کرده و با استفاده از نتایج و جداول زیر، تصمیم به حفظ آن در آموزش مدل می‌گیریم. در این ستون تعدادی داده پرت وجود دارد که آنها را تبدیل به ۱ می‌کنیم.

```
MyCustomerInfo_tr.groupBy('SeniorCitizen', 'Label').count().show()
```

```
+-----+-----+-----+
|SeniorCitizen|Label| count|
+-----+-----+-----+
|          14.0|No|    548|
|           1.0|Yes|  10046|
|          17.0|No|     6|
|          14.0|Yes|   246|
|           1.0|No|  28139|
|           0.0|Yes|  23612|
|           0.0|No| 167393|
+-----+-----+-----+
```

```
[59] MyCustomerInfo_tr = MyCustomerInfo_tr.withColumn("SeniorCitizen", when(col("SeniorCitizen")>=1, 1).otherwise(0))
MyCustomerInfo_tr.groupBy('SeniorCitizen', 'Label').count().show()
print('corrolation between SeniorCitizen and Label:')
MyCustomerInfo_tr.corr('SeniorCitizen_indexed', 'Label_indexed', 'pearson')
```

```
+-----+-----+-----+
|SeniorCitizen|Label| count|
+-----+-----+-----+
|           1|No|  28693|
|           0|No| 167393|
|           0|Yes|  23612|
|           1|Yes|  10292|
+-----+-----+-----+
```

corrolation between SeniorCitizen and Label:
0.14788531816558

ستون چهارم: Partner

```
MyCustomerInfo_tr.groupBy('Partner').count().show()
MyCustomerInfo_tr.groupBy('Partner', 'Label').count().show()
print('corrolation between Partner and Label:')
MyCustomerInfo_tr.corr('Partner_indexed', 'Label_indexed', 'pearson')
```

```
+-----+-----+
|Partner| count|
+-----+-----+
|No|  85648|
|Yes| 144342|
+-----+-----+
```

```
+-----+-----+-----+
|Partner|Label| count|
+-----+-----+-----+
|Yes|Yes|  18044|
|No|No|  69788|
|Yes|No| 126298|
|No|Yes|  15860|
+-----+-----+-----+
```

corrolation between Partner and Label:
0.08204854679463729

ستون پنجم: Dependents

با توجه به مقادیر زیر اینطور برداشت می شود که dependents در داده های زیادی تاثیر مستقیم در مقدار لیبیل خواهد گذاشت.

2s



```
MyCustomerInfo_tr.groupBy('Dependents').count().show()
MyCustomerInfo_tr.groupBy('Dependents','Label').count().show()
print('corrolation between Dependent and Label:')
MyCustomerInfo_tr.corr('Dependents_indexed','Label_indexed','pearson')
```

```
+-----+-----+
|Dependents| count|
+-----+-----+
|          No|148232|
|          Yes| 81758|
+-----+-----+
```

```
+-----+-----+-----+
|Dependents|Label| count|
+-----+-----+-----+
|          Yes|  Yes|  7264| |
|          No|   No|121592|
|          Yes|   No| 74494|
|          No|  Yes| 26640||
+-----+-----+-----+
```

```
corrolation between Dependent and Label:
-0.12269058784498946
```

ستون ششم: tenure

در این ستون نیز تعدادی داده پرت وجود دارد که آن ها را با میانگین ستون جایگزین می کنیم.



```
MyCustomerInfo_tr.groupBy('tenure','Label').count().show()
```



```
+-----+-----+-----+
|tenure|Label|count|
+-----+-----+-----+
|  24.0|  Yes|  552|
|-592.0|  No|   16|
|  69.0|  No| 6003|
|-585.0|  No|   30|
|  51.0|  Yes|  408|
|-593.0|  Yes|   14|
|  47.0|  No| 2538|
|-584.0|  No|   32|
|  15.0|  No|  915|
|  21.0|  Yes|  357|
|  41.0|  Yes|  574|
|  58.0|  No| 3306|
|  63.0|  No| 4284|
|  63.0|  Yes|  252|
```



```
✓ [133] MyCustomerInfo_tr = MyCustomerInfo_tr.withColumn("tenure", when(col("tenure")<=0, mean_Tenure).otherwise(col("tenure")))
```

حال میزان همبستگی را محاسبه می‌کنیم. مشخص است که این ویژگی در پیش‌بینی تاثیرگذار است.

```
1s ▶ MyCustomerInfo_tr.groupBy('tenure','Label').count().show()  
print('corrolation between tenure and Label:')  
MyCustomerInfo_tr.corr('tenure','Label_indexed','pearson')
```

```
↵ +-----+-----+-----+  
|tenure|Label|count|  
+-----+-----+-----+  
| 24.0| Yes| 552|  
| 69.0| No| 6003|  
| 51.0| Yes| 408|  
| 47.0| No| 2538|  
| 15.0| No| 915|  
| 21.0| Yes| 357|  
| 41.0| Yes| 574|  
| 58.0| No| 3306|  
| 63.0| No| 4284|  
| 63.0| Yes| 252|  
| 59.0| No| 3068|  
| 14.0| Yes| 336|  
| 39.0| No| 1638|  
| 36.0| Yes| 360|  
| 66.0| No| 5019|  
| 18.0| Yes| 432|  
| 34.0| No| 1802|  
| 68.0| Yes| 680|  
| 22.0| Yes| 594|  
| 14.0| No| 714|  
+-----+-----+-----+  
only showing top 20 rows
```

```
corrolation between tenure and Label:  
-0.2561566998567579
```

ستون هفتم: PhoneService

```
▶ MyCustomerInfo_tr.groupBy('PhoneService','Label').count().show()  
print('corrolation between PhoneService and Label:')  
MyCustomerInfo_tr.corr('PhoneService_indexed','Label_indexed','pearson')
```

```
↵ +-----+-----+-----+  
|PhoneService|Label| count|  
+-----+-----+-----+  
| Yes| Yes| 31196|  
| No| No| 19109|  
| Yes| No| 176977|  
| No| Yes| 2708|  
+-----+-----+-----+
```

```
corrolation between MultipleLines and Label:  
-0.021268995159031242
```

ستون هشتم: MultipleLines

```
[141] MyCustomerInfo_tr.groupBy('MultipleLines','Label').count().show()
print('corrolation between MultipleLines and Label:')
MyCustomerInfo_tr.corr('MultipleLines_indexed','Label_indexed','pearson')
```

MultipleLines	Label	count
No phone service	No	19110
Yes	Yes	22182
No	No	73492
Yes	No	103484
No phone service	Yes	2707
No	Yes	9015

```
corrolation between MultipleLines and Label:
-0.07725986644163231
```

ستون نهم: internetServices

با توجه به مقدار همبستگی این ستون باید نگه داشته شود.

```
✓ [142] MyCustomerInfo_tr.groupBy('InternetService','Label').count().show()
1s print('corrolation between InternetService and Label:')
MyCustomerInfo_tr.corr('InternetService_indexed','Label_indexed','pearson')
```

InternetService	Label	count
DSL	Yes	6513
Fiber optic	Yes	26459
No	No	45985
Fiber optic	No	76491
No	Yes	932
DSL	No	73610

```
corrolation between InternetService and Label:
-0.27517341751687435
```

بقیه ستون ها در فایل notebook بررسی شده اند.

قدم دوم: عملیات feature engineering را به خوبی برای داده‌گان خود انجام دهید و دلیل انتخاب هریک از ستون‌ها یا عدم انتخاب آن‌ها را به صورت منطقی بیان کنید. (با نمودار و تحلیل آن، با کمک EDA انجام شده)

قدم سوم: الگوریتم Logistic Regression را بر روی داده‌های خود اعمال کنید.

```

MyCustomerInfo2 = MyCustomerInfo_tr.select('customerID','features','Label_indexed')
MyCustomerInfo2 = MyCustomerInfo2.withColumnRenamed('Label_indexed','label')
MyCustomerInfo2.na.drop()
MyCustomerInfo2.show(50)

```

customerID	features	label
5331-RGMTT	(48,[0,3,6,8,10,1...]	0.0
5161-XEUVX	(48,[0,2,6,8,10,1...]	0.0
0336-PIKEI	(48,[0,3,6,8,10,1...]	0.0
3345-PBBFH	(48,[0,2,6,8,10,1...]	0.0
5067-XJOFU	(48,[0,3,6,9,10,1...]	0.0

داده‌ها را با نسبت ۷ به ۳ به دو قسمت برای آموزش و تست تقسیم می‌کنیم. با استفاده از مدل آماده logisticRegression مدل جدیدی برای آموزش و تست ایجاد نموده و داده‌های مربوط را به آن می‌دهیم. نتیجه در دیتافریم result قابل مشاهده است.

```

from pyspark.ml.classification import LogisticRegression

train_data , test_data = MyCustomerInfo2.randomSplit([0.7,.3],seed=200)
log_reg = LogisticRegression(featuresCol='features',labelCol='label')
fit_model = log_reg.fit(train_data)
results = fit_model.transform(test_data)

results.show()

```

customerID	features	label	rawPrediction	probability	prediction
null	(47,[0,2,6,8,10,1...]	0.0	[4.20240678706489...	[0.98526096003936...	0.0
null	(47,[0,2,6,8,10,1...]	0.0	[5.64276210134641...	[0.99646944045063...	0.0
null	(47,[0,2,6,8,10,1...]	0.0	[5.75521478992064...	[0.99684376688432...	0.0
null	(47,[0,2,6,8,10,1...]	0.0	[5.82122475886831...	[0.99704478640371...	0.0
null	(47,[0,2,6,8,10,1...]	0.0	[5.86071506324827...	[0.99715888931656...	0.0

قدم چهارم: دقت مدل خود را ارزیابی کنید. (در این مرحله شما باید مراحل آزمایش، تعداد دادگان ترین و تست، احتمال صحیح بودن یک برچسب که مدل پیشبینی کرده است، را تعیین کنید)

در این قسمت دقت مدل با استفاده از تفاوت بین label پیش‌بینی شده و label موجود در داده‌ها محاسبه می‌شود.

✓
7s

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
my_eval = BinaryClassificationEvaluator(rawPredictionCol='prediction',
                                       labelCol='label')

results.select('label','prediction')
accuracy = my_eval.evaluate(results)
print("accuracy is : ",accuracy)
```

📄 accuracy is : 0.618722297478557