ml_with_spark

November 24, 2019

In [92]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, col, concat,count, desc, explode, lit, min, max,
from pyspark.sql.types import IntegerType

from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.feature import CountVectorizer, IDF, Normalizer, PCA, RegexTokenizer, S
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

import re
```

In [2]:
```
stack_overflow_data = "s3n://juditl/Train_onetag_small.json"
```

In [3]:
```
df = spark.read.json(stack_overflow_data)
df.persist()
```

Out[3]:
```
DataFrame[Body: string, Id: bigint, Tags: string, Title: string, oneTag: string]
```

In [4]:
```
df.head()
```

Out[4]:
```
Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpeg
```

In [5]:
```
regexTokenizer = RegexTokenizer(inputCol="Body", outputCol="words", pattern="\\W")
df = regexTokenizer.transform(df)
df.head()
```

Out[5]:
```
Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpeg
```

In [6]:
```
body_length = udf(lambda x: len(x), IntegerType())
df = df.withColumn("BodyLength", body_length(df.words))
```

In [7]:
```
df.head()
```

Out[7]:
```
Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpeg
```

In [8]:
```
number_of_paragraphs = udf(lambda x: len(re.findall("</p>", x)), IntegerType())
number_of_links = udf(lambda x: len(re.findall("</a>", x)), IntegerType())
```

```
In [9]: df = df.withColumn("NumParagraphs", number_of_paragraphs(df.Body))
        df = df.withColumn("NumLinks", number_of_links(df.Body))

In [10]: df.head(2)

Out[10]: [Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jp
          Row(Body=u'<p>In my favorite editor (vim), I regularly use ctrl-w to execute a certain

In [11]: assembler = VectorAssembler(inputCols=["BodyLength", "NumParagraphs", "NumLinks"], outp
         df = assembler.transform(df)

In [12]: df.head()

Out[12]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [13]: scaler = Normalizer(inputCol="NumFeatures", outputCol="ScaledNumFeatures")
         df = scaler.transform(df)

In [14]: df.head(2)

Out[14]: [Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jp
          Row(Body=u'<p>In my favorite editor (vim), I regularly use ctrl-w to execute a certain

In [15]: scaler2 = StandardScaler(inputCol="NumFeatures", outputCol="ScaledNumFeatures2", withSt
         scalerModel = scaler2.fit(df)
         df = scalerModel.transform(df)

In [16]: df.head(2)

Out[16]: [Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jp
          Row(Body=u'<p>In my favorite editor (vim), I regularly use ctrl-w to execute a certain
```

Quiz time

```
In [28]: df.where(df.Id == 1112).show()

+--------------------+----+--------------------+--------------------+------+--------------------
|                Body|  Id|                Tags|               Title|oneTag|               words
+--------------------+----+--------------------+--------------------+------+--------------------
|<p>I submitted my...|1112|iphone app-store ...|iPhone app releas...|iphone|[p, i, submitted,...
+--------------------+----+--------------------+--------------------+------+--------------------



In [30]: df = df.withColumn("Desc", concat(col("Title"), lit(' '), col("Body")))

In [31]: regexTokenizer2 = RegexTokenizer(inputCol="Desc", outputCol="words2", pattern="\\W")
         df = regexTokenizer2.transform(df)
         df = df.withColumn("DescLength", body_length(df.words2))

In [33]: df.where(df.Id == 5123).collect()
```

```
Out[33]: [Row(Body=u"<p>Here's an interesting experiment with using Git. Think of Github's \u201

In [34]: assembler2 = VectorAssembler(inputCols=["DescLength"], outputCol="DescVec")
         df = assembler2.transform(df)

In [35]: scaler_q1 = Normalizer(inputCol="DescVec", outputCol="DescVecNormalizer")
         df = scaler_q1.transform(df)

In [36]: df.where(df.Id == 512).collect()

Out[36]: [Row(Body=u"<p>I'd like to have a better understanding of what optimizations HotSpot mi

In [44]: scaler_q2 = StandardScaler(inputCol="DescVec", outputCol="DescVecStandardScaler4", with
         scalerModel_q2 = scaler_q2.fit(df)
         df = scalerModel_q2.transform(df)

In [45]: df.where(df.Id == 512).collect()

Out[45]: [Row(Body=u"<p>I'd like to have a better understanding of what optimizations HotSpot mi

In [46]: from pyspark.ml.feature import MinMaxScaler
         scaler_q3 = MinMaxScaler(inputCol="DescVec", outputCol="DescVecMinMaxScaler")
         scalerModel_q3 = scaler_q3.fit(df)
         df = scalerModel_q3.transform(df)

In [47]: df.where(df.Id == 512).collect()

Out[47]: [Row(Body=u"<p>I'd like to have a better understanding of what optimizations HotSpot mi

In [48]: cv = CountVectorizer(inputCol="words", outputCol="TF", vocabSize=1000)
         cvmodel = cv.fit(df)
         df = cvmodel.transform(df)
         df.take(1)

Out[48]: [Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jp

In [49]: cvmodel.vocabulary[-10:]

Out[49]: [u'customer',
          u'desktop',
          u'buttons',
          u'previous',
          u'master',
          u'math',
          u'000',
          u'blog',
          u'comes',
          u'wordpress']
```

```
In [50]: idf = IDF(inputCol="TF", outputCol="TFIDF")
         idfModel = idf.fit(df)
         df = idfModel.transform(df)
         df.head()

Out[50]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [51]: indexer = StringIndexer(inputCol="oneTag", outputCol="label")
         df = indexer.fit(df).transform(df)

In [52]: df.head()

Out[52]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [ ]: pca = PCA(k=100, inputCol="TFIDF", outputCol="pcaTFIDF")
        model = pca.fit(df)
        df = model.transform(df)

In [ ]: df.head()

In [53]: number_of_tags = udf(lambda x: len(x.split(" ")), IntegerType())
         df = df.withColumn("NumTags", number_of_tags(df.Tags))

In [54]: df.head()

Out[54]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [55]: df.groupby("NumTags").count().orderBy("NumTags").show()

+-------+-----+
|NumTags|count|
+-------+-----+
|      1|13858|
|      2|26540|
|      3|28769|
|      4|19108|
|      5|11725|
+-------+-----+



In [56]: df.groupby("NumTags").agg(avg(col("BodyLength"))).orderBy("NumTags").show()

+-------+------------------+
|NumTags|   avg(BodyLength)|
+-------+------------------+
|      1|135.41311877615817|
|      2|153.82456669178598|
|      3|172.73704334526747|
|      4|192.67050450073268|
|      5|218.54251599147122|
+-------+------------------+
```

```
In [57]: assembler = VectorAssembler(inputCols=["BodyLength"], outputCol="LengthFeature")
         df = assembler.transform(df)

In [58]: df.head()

Out[58]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [62]: lr = LinearRegression(maxIter=5, regParam=0.0, fitIntercept=False, solver="normal")

In [63]: data = df.select(col("NumTags").alias("label"), col("LengthFeature").alias("features"))
         data.head()

Out[63]: Row(label=5, features=DenseVector([83.0]))

In [64]: lrModel = lr.fit(data)

In [65]: lrModel.coefficients

Out[65]: DenseVector([0.0079])

In [66]: lrModel.intercept

Out[66]: 0.0

In [67]: lrModelSummary = lrModel.summary

In [68]: lrModelSummary.r2

Out[68]: 0.42481762576079773
```

Quiz time

```
In [69]: df.groupby("NumTags").agg(avg(col("DescLength"))).orderBy("NumTags").show()

+-------+------------------+
|NumTags|   avg(DescLength)|
+-------+------------------+
|      1|143.68776158175783|
|      2| 162.1539186134137|
|      3|181.26021064340088|
|      4|201.46530249110322|
|      5|227.64375266524522|
+-------+------------------+



In [72]: data = df.select(col("NumTags").alias("label"), col("DescVec").alias("features"))
         data.head()

Out[72]: Row(label=5, features=DenseVector([96.0]))
```

```
In [73]: lrModel_q1 = lr.fit(data)

In [74]: lrModel_q1.summary.r2

Out[74]: 0.44551495963084176
```

Back to Logistic Regression

```
In [ ]: data2 = df.select(col("label").alias("label"), col("TFIDF").alias("features"))
        data2.head()

In [ ]: lr2 = LogisticRegression(maxIter=10, regParam=0.0)

In [ ]: lrModel2 = lr2.fit(data2)

In [ ]: lrModel2.coefficientMatrix

In [ ]: lrModel2.interceptVector

In [ ]: lrModel2.summary.accuracy

In [ ]: 1/301.0
```

Quiz - Clustering

```
In [75]: from pyspark.ml.clustering import KMeans

In [76]: df.head()

Out[76]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [88]: df.agg(min("DescLength")).show()

+---------------+
|min(DescLength)|
+---------------+
|             10|
+---------------+



In [89]: df.agg(max("DescLength")).show()

+---------------+
|max(DescLength)|
+---------------+
|           7532|
+---------------+
```

```
In [93]: df.agg(avg("DescLength"), stddev("DescLength")).show()

+---------------+-----------------------+
|avg(DescLength)|stddev_samp(DescLength)|
+---------------+-----------------------+
|      180.28187|     192.10819533505128|
+---------------+-----------------------+
```

```
In [78]: kmeans = KMeans().setParams(featuresCol="DescVec", predictionCol="DescGroup", k=5, seed
         model = kmeans.fit(df)
         df = model.transform(df)

In [79]: df.head()

Out[79]: Row(Body=u"<p>I'd like to check if an uploaded file is an image file (e.g png, jpg, jpe

In [85]: df.groupby("DescGroup").agg(avg(col("DescLength")), avg(col("NumTags")), count(col("Des

+---------+------------------+------------------+-----------------+
|DescGroup|   avg(DescLength)|      avg(NumTags)|count(DescLength)|
+---------+------------------+------------------+-----------------+
|        4| 92.75317245164402| 2.732166913366707|            60127|
|        0|224.90495069296375| 3.068663379530917|            30016|
|        2| 457.1547183613753|3.2275054864667156|             8202|
|        3| 989.9467576791809| 3.279180887372014|             1465|
|        1| 2634.815789473684|3.3684210526315788|              190|
+---------+------------------+------------------+-----------------+
```

Pipelines

```
In [ ]: print type(lr2)

In [ ]: print type(lrModel2)

In [59]: df2 = spark.read.json(stack_overflow_data)
         df2.persist()

Out[59]: DataFrame[Body: string, Id: bigint, Tags: string, Title: string, oneTag: string]

In [60]: regexTokenizer = RegexTokenizer(inputCol="Body", outputCol="words", pattern="\\W")
         cv = CountVectorizer(inputCol="words", outputCol="TF", vocabSize=10000)
         idf = IDF(inputCol="TF", outputCol="features")
         indexer = StringIndexer(inputCol="oneTag", outputCol="label")

         lr =  LogisticRegression(maxIter=10, regParam=0.0, elasticNetParam=0)

         pipeline = Pipeline(stages=[regexTokenizer, cv, idf, indexer, lr])
```

```
In [ ]: plrModel = pipeline.fit(df2)

In [ ]: df3 = plrModel.transform(df2)

In [ ]: df3.head()

In [ ]: df3.filter(df3.label == df3.prediction).count()
```