notebooks_jupyter_zoey

March 12, 2022

[2]:
```
import os
import sys
import pyspark
from pyspark.sql import SQLContext
import pandas as pd
from pyspark import SparkContext, SparkConf
import pyspark.sql.functions as F
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('GCSFilesRead').getOrCreate()
spark.conf.set("spark.sql.debug.maxToStringFields", 1000)
```

[3]:
```
'''
from google.cloud import storage

gcs_client = storage.Client()
bucket = gcs_client.bucket('datasetsbdp')
'''
```

[3]:
```
"\nfrom google.cloud import storage\n\ngcs_client = storage.Client()\nbucket =
gcs_client.bucket('datasetsbdp')\n"
```

[4]:
```
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/Users/xiantang/Desktop/
↪BigDataPlatform/Project/yelp_dataset/iconic-being-343500-f1cb28ff8582.json'
```

[5]:
```
#spark._jsc.hadoopConfiguration().set("google.cloud.auth.service.account.json.
↪keyfile","/Users/xiantang/Desktop/BigDataPlatform/Project/yelp_dataset/
↪iconic-being-343500-f1cb28ff8582.json")
```

[5]:
```
!hdfs dfs -ls 'gs://datasetsbdp/'
```

```
ls: Error reading credential file from environment variable
GOOGLE_APPLICATION_CREDENTIALS, value
'/Users/xiantang/Desktop/BigDataPlatform/Project/yelp_dataset/iconic-
being-343500-f1cb28ff8582.json': File does not exist.
```

[6]:
```
PROJECT_ID="BDPYelpFinalProject"
bucket_name = "datasetsbdp"
path_review = f"gs://{bucket_name}/dataset_review.json"
path_tip = f"gs://{bucket_name}/dataset_tip.json"
path_user = f"gs://{bucket_name}/dataset_user.json"
path_checkin = f"gs://{bucket_name}/dataset_checkin.json"
path_business = f"gs://{bucket_name}/dataset_business.json"
#gs://bdpproject/yelp_academic_dataset_review.json
```

0.1 For df_review: stars column only have 1~5: 5 distinct numbers.

[7]:
```
df_review=spark.read.json(path_review)
df_review.show(5)
```

```
+--------------------+----+--------------------+-----+--------------------+-----+
--------------------+------+--------------------+
|         business_id|cool|                date|funny|           review_id|stars|
text|useful|             user_id|
+--------------------+----+--------------------+-----+--------------------+-----+
--------------------+------+--------------------+
|XQfwVwDr-v0ZS3_Cb…|   0|2018-07-07 22:09:11|    0|KU_O5udG6zpxOg-Vc…|
3.0|If you decide to …|     0|mh_-eMZ6K5RLWhZyI…|
|7ATYjTIgM3jUlt4UM…|   1|2012-01-03 15:28:18|    0|BiTunyQ73aT9WBnpR…|
5.0|I've taken a lot …|     1|OyoGAe7OKpv6SyGZT…|
|YjUWPpI6HXG530lwP…|   0|2014-02-05 20:30:30|    0|saUsX_uimxRlCVr67…|
3.0|Family diner. Had…|     0|8g_iMtfSiwikVnbP2…|
|kxX2SOes4o-D3ZQBk…|   1|2015-01-04 00:01:03|    0|AqPFMleE6RsU23_au…|
5.0|Wow!  Yummy, diff…|     1|_7bHUi9Uuf5__HHc_…|
|e4Vwtrqf-wpJfwesg…|   1|2017-01-14 20:54:15|    0|Sx8TMOWLNuJBWer-0…|
4.0|Cute interior and…|     1|bcjbaE6dDog4jkNY9…|
+--------------------+----+--------------------+-----+--------------------+-----+
--------------------+------+--------------------+
only showing top 5 rows
```

[17]:
```
df_review.count()
```

[17]: 6990280

[18]:
```
df_review.select('stars').distinct().show()
```

```
[Stage 7:=================================================>  (38 + 2) / 40]
+-----+
```

```
|stars|
+-----+
|  1.0|
|  4.0|
|  3.0|
|  2.0|
|  5.0|
+-----+
```

[8]:
```
df_tip = spark.read.json(path_tip)
df_tip.show(5)
```

```
+--------------------+---------------+-------------------+--------------------+
--------------------+
|         business_id|compliment_count|               date|                text|
user_id|
+--------------------+---------------+-------------------+--------------------+
--------------------+
|3uLgwr0qeCNMjKenH…|               0|2012-05-18 02:17:21|Avengers time
wit…|AGNUgVwnZUey3gcPC…|
|QoezRbYQncpRqyrLH…|               0|2013-02-05 18:35:10|They have lots
of…|NBN4MgHP9D3cw--Sn…|
|MYoRNLb5chwjQe3c_…|               0|2013-08-18 00:56:08|It's open even
wh…|-copOvldyKh1qr-vz…|
|hV-bABTK-glh5wj31…|               0|2017-06-27 23:05:38|Very decent
fried…|FjMQVZjSqY8syIO-5…|
|_uN0OudeJ3Zl_tf6n…|               0|2012-10-06 19:43:09|Appetizers..
plat…|ld0AperBXk1h6Ubqm…|
+--------------------+---------------+-------------------+--------------------+
--------------------+
only showing top 5 rows
```

[20]:
```
df_tip.select('compliment_count').distinct().show()
```

```
+----------------+
|compliment_count|
+----------------+
|               0|
|               5|
|               6|
|               1|
```

```
|               3|
|               2|
|               4|
+----------------+
```

[9]:
```
df_user = spark.read.json(path_user)
df_user.show(5)
```

```
+-------------+---------------+---------------+----------------+-------------+-
-------------+---------------+---------------+----------------+--------------
-+-----------------+----------------+-----+--------------------+----+---------
-----------+-----+------+------------+------+--------------------+-------------
-----+
|average_stars|compliment_cool|compliment_cute|compliment_funny|compliment_hot|c
ompliment_list|compliment_more|compliment_note|compliment_photos|compliment_plai
n|compliment_profile|compliment_writer| cool|               elite|fans|
friends|funny|  name|review_count|useful|             user_id|
yelping_since|
+-------------+---------------+---------------+----------------+-------------+-
-------------+---------------+---------------+----------------+--------------
-+-----------------+----------------+-----+--------------------+----+---------
-----------+-----+------+------------+------+--------------------+-------------
-----+
|         3.91|            467|             56|             467|           250|
18|             65|            232|             180|             844|
55|              239| 5994|               2007| 267|NSCy54eWehBJyZdG2…|
1259|Walker|         585|  7217|qVc8ODYU5SZjKXVBg…|2007-01-25 16:47:26|
|         3.74|           3131|            157|            3131|          1145|
251|            264|           1847|             1946|            7054|
184|
1521|27281|2009,2010,2011,20…|3138|ueRPE0CX75ePGMqOF…|13066|Daniel|
4333| 43091|j14WgRoU_-2ZE1aw1…|2009-01-25 04:35:42|
|         3.32|            119|             17|             119|            89|
3|             13|             66|               18|              96|
10|               35| 1003|2009,2010,2011,20…|  52|LuO3Bn4f3rlhyHIaN…| 1010|
Steph|         665|  2086|2WnXYQFK0hXEoTxPt…|2008-07-25 10:41:00|
|         4.27|             26|              6|              26|            24|
2|              4|             12|               9|              16|
1|              10|  299|      2009,2010,2011|  28|enx1vVPnfdNUdPho6…|  330|
Gwen|         224|   512|SZDeASXq7o05mMNLs…|2005-11-29 04:38:33|
|         3.54|              0|              0|               0|             1|
0|              1|             1|               0|               1|
0|                0|    7|                    |   1|PBK4q9KEEBHhFvSXC…|   15|
Karen|          79|    29|hA5lMy-EnncsH4JoR…|2007-01-05 19:40:59|
+-------------+---------------+---------------+----------------+-------------+-
```

```
-------------+--------------+--------------+-----------------+--------------
-+-----------------+----------------+-----+--------------------+----+---------
----------+-----+------+-----------+------+--------------------+-------------
-----+
only showing top 5 rows
```

[10]:
```
df_checkin = spark.read.json(path_checkin)
df_checkin.show(5)
```

```
+--------------------+--------------------+
|         business_id|                date|
+--------------------+--------------------+
|---kPU91CF4Lq2-Wl…|2020-03-13 21:10:…|
|--0iUa4sNDFiZFrAd…|2010-09-13 21:43:…|
|--30_8IhuyMHbSOcN…|2013-06-14 23:29:…|
|--7PUidqRWpRSpXeb…|2011-02-15 17:12:…|
|--7jw19RH9JKXgFoh…|2014-04-21 20:42:…|
+--------------------+--------------------+
only showing top 5 rows
```

[ ]:
```
```

[11]:
```
df_business = spark.read.json(path_business)
df_business.show(5)
```

```
+--------------------+--------------------+--------------------+---------------
----+-------------+--------------------+-------+----------+-----------+--------
-----------+----------+------------+-----+-----+
|             address|          attributes|         business_id|
categories|         city|               hours|is_open|  latitude|   longitude|
name|postal_code|review_count|stars|state|
+--------------------+--------------------+--------------------+---------------
----+-------------+--------------------+-------+----------+-----------+--------
-----------+----------+------------+-----+-----+
|1616 Chapala St, …|{null, null, null…|Pns2l4eNsfO8kk83d…|Doctors,
Traditio…|Santa Barbara|                null|
0|34.4266787|-119.7111968|Abby Rappoport, L…|      93101|           7|  5.0|
CA|
|87 Grasso Plaza S…|{null, null, null…|mpf3x-BjTdTEA3yCZ…|Shipping
Centers,…|       Affton|{8:0-18:30, 0:0-0…|      1| 38.551126|  -90.335695|
The UPS Store|      63123|          15|  3.0|   MO|
|5255 E Broadway Blvd|{null, null, null…|tUFrWirKiKi_TAnsV…|Department
Stores…|       Tucson|{8:0-23:0, 8:0-22…|      0| 32.223236| -110.880452|
Target|      85711|          22|  3.5|   AZ|
```

```
|         935 Race St|{null, null, u'no…|MTSW4McQd7CbVtyjq…|Restaurants,
Food…| Philadelphia|{7:0-21:0, 7:0-20…|      1|39.9555052| -75.1555641|  St
Honore Pastries|      19107|          80|  4.0|   PA|
|       101 Walnut St|{null, null, null…|mWMc6_wTdE0EUBKIG…|Brewpubs,
Breweri…|   Green Lane|{12:0-22:0, null,…|      1|40.3381827|
-75.4716585|Perkiomen Valley …|      18054|          13|  4.5|   PA|
+--------------------+--------------------+--------------------+----------------
----+------------+--------------------+-------+----------+-----------+--------
-----------+-----------+-----------+-----+-----+
only showing top 5 rows
```

0.2 cleaning df_business

[24]:
```
df_business.printSchema()
```

```
root
 |-- address: string (nullable = true)
 |-- attributes: struct (nullable = true)
 |    |-- AcceptsInsurance: string (nullable = true)
 |    |-- AgesAllowed: string (nullable = true)
 |    |-- Alcohol: string (nullable = true)
 |    |-- Ambience: string (nullable = true)
 |    |-- BYOB: string (nullable = true)
 |    |-- BYOBCorkage: string (nullable = true)
 |    |-- BestNights: string (nullable = true)
 |    |-- BikeParking: string (nullable = true)
 |    |-- BusinessAcceptsBitcoin: string (nullable = true)
 |    |-- BusinessAcceptsCreditCards: string (nullable = true)
 |    |-- BusinessParking: string (nullable = true)
 |    |-- ByAppointmentOnly: string (nullable = true)
 |    |-- Caters: string (nullable = true)
 |    |-- CoatCheck: string (nullable = true)
 |    |-- Corkage: string (nullable = true)
 |    |-- DietaryRestrictions: string (nullable = true)
 |    |-- DogsAllowed: string (nullable = true)
 |    |-- DriveThru: string (nullable = true)
 |    |-- GoodForDancing: string (nullable = true)
 |    |-- GoodForKids: string (nullable = true)
 |    |-- GoodForMeal: string (nullable = true)
 |    |-- HairSpecializesIn: string (nullable = true)
 |    |-- HappyHour: string (nullable = true)
 |    |-- HasTV: string (nullable = true)
 |    |-- Music: string (nullable = true)
 |    |-- NoiseLevel: string (nullable = true)
 |    |-- Open24Hours: string (nullable = true)
 |    |-- OutdoorSeating: string (nullable = true)
 |    |-- RestaurantsAttire: string (nullable = true)
```

```
 |    |-- RestaurantsCounterService: string (nullable = true)
 |    |-- RestaurantsDelivery: string (nullable = true)
 |    |-- RestaurantsGoodForGroups: string (nullable = true)
 |    |-- RestaurantsPriceRange2: string (nullable = true)
 |    |-- RestaurantsReservations: string (nullable = true)
 |    |-- RestaurantsTableService: string (nullable = true)
 |    |-- RestaurantsTakeOut: string (nullable = true)
 |    |-- Smoking: string (nullable = true)
 |    |-- WheelchairAccessible: string (nullable = true)
 |    |-- WiFi: string (nullable = true)
 |-- business_id: string (nullable = true)
 |-- categories: string (nullable = true)
 |-- city: string (nullable = true)
 |-- hours: struct (nullable = true)
 |    |-- Friday: string (nullable = true)
 |    |-- Monday: string (nullable = true)
 |    |-- Saturday: string (nullable = true)
 |    |-- Sunday: string (nullable = true)
 |    |-- Thursday: string (nullable = true)
 |    |-- Tuesday: string (nullable = true)
 |    |-- Wednesday: string (nullable = true)
 |-- is_open: long (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- name: string (nullable = true)
 |-- postal_code: string (nullable = true)
 |-- review_count: long (nullable = true)
 |-- stars: double (nullable = true)
 |-- state: string (nullable = true)
```

[25]:
```
df_business=df_business.
↪select('business_id','name','city','stars','review_count','categories')
```

[26]:
```
df_business.show(2)
```

```
+--------------------+--------------------+------------+-----+------------+----
----------------+
|         business_id|                name|        city|stars|review_count|
categories|
+--------------------+--------------------+------------+-----+------------+----
----------------+
|Pns2l4eNsfO8kk83d…|Abby Rappoport, L…|Santa Barbara|  5.0|
7|Doctors, Traditio…|
|mpf3x-BjTdTEA3yCZ…|       The UPS Store|       Affton|  3.0|
15|Shipping Centers,…|
+--------------------+--------------------+------------+-----+------------+----
----------------+
```

```
only showing top 2 rows
```

1 Cleaning df_review

[12]:
```
df_review=df_review.select('business_id','date','stars','text','user_id')
```

[13]:
```
df_review.show(2)
```

```
+--------------------+-------------------+-----+--------------------+-----------
---------+
|         business_id|               date|stars|                text|
user_id|
+--------------------+-------------------+-----+--------------------+-----------
---------+
|XQfwVwDr-v0ZS3_Cb…|2018-07-07 22:09:11|  3.0|If you decide to …|mh_-
eMZ6K5RLWhZyI…|
|7ATYjTIgM3jUlt4UM…|2012-01-03 15:28:18|  5.0|I've taken a lot
…|OyoGAe7OKpv6SyGZT…|
+--------------------+-------------------+-----+--------------------+-----------
---------+
only showing top 2 rows
```

[14]:
```
from pyspark.ml.feature import StringIndexer, OneHotEncoder
from pyspark.ml.feature import VectorAssembler

#varIdxer = StringIndexer(inputCol='strVar',outputCol='varIdx').fit(factors)
#factors = varIdxer.transform(factors)
```

[30]:
```
# What is the average number of reviews per business?


total_review_count=df_business.agg(F.sum("review_count")).collect()[0][0]

total_business=df_business.select('business_id').distinct().count()

avg_review=total_review_count/total_business

print('the average number of reviews per business is :',avg_review)
```

```
[Stage 32:>                                                         (0 + 2) / 2]
the average number of reviews per business is : 44.86656113232144
```

[31]:
```
#What is the stars distribution?
#x: star 1 to 5 ; y: num of reviews

star_dist_plot=df_review.groupby('stars').count().toPandas()

star_dist_plot=star_dist_plot.sort_values(by=['count'])
```

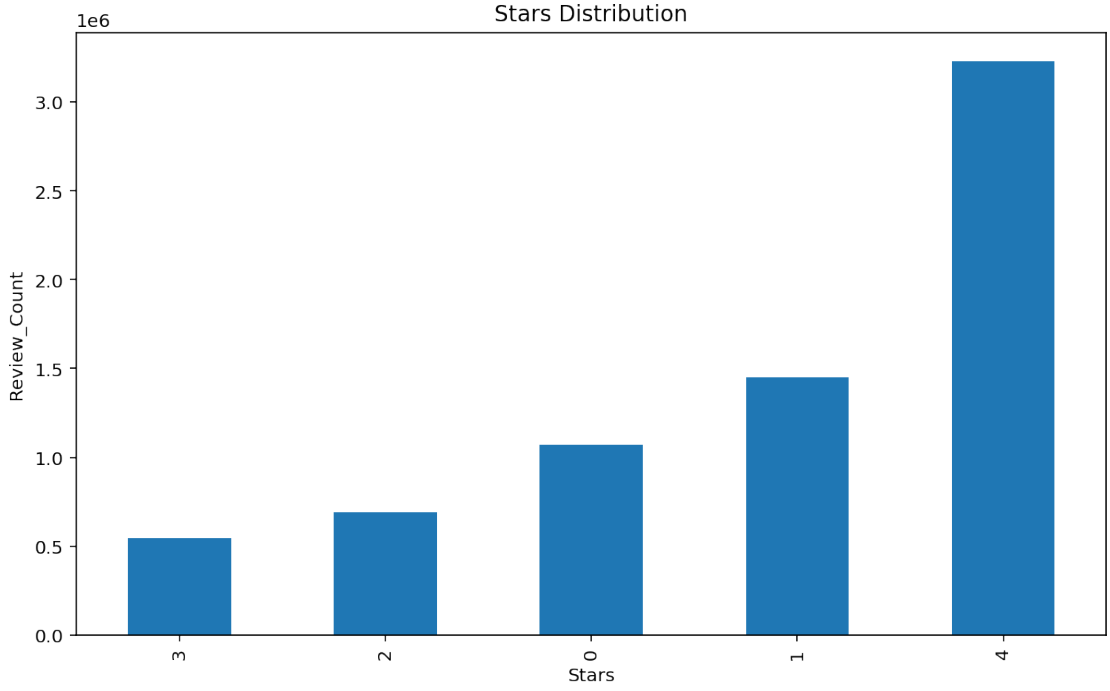
[32]:
```
#Plotting

# Set the figure size - handy for larger output
from matplotlib import pyplot as plt
plt.rcParams["figure.figsize"] = [10, 6]
# Set up with a higher resolution screen (useful on Mac)
%config InlineBackend.figure_format = 'retina'

from matplotlib import pyplot as plt
star_dist_plot['count'].plot(kind="bar", title="stars distribution")

plt.title("Stars Distribution")
plt.xlabel("Stars")
plt.ylabel("Review_Count")
```

[32]:
```
Text(0, 0.5, 'Review_Count')
```



2 Text Preprocessing

[33]:
```
## remove special character: keeps only letters, space and numbers
## PipeLine: Tokenization, stopword, word2vec,
```

[15]:
```
df_review.show(5)
```

```
+--------------------+--------------------+-----+--------------------+----------
---------+
|         business_id|                date|stars|                text|
user_id|
+--------------------+--------------------+-----+--------------------+----------
---------+
|XQfwVwDr-v0ZS3_Cb…|2018-07-07 22:09:11|  3.0|If you decide to …|mh_-
eMZ6K5RLWhZyI…|
|7ATYjTIgM3jUlt4UM…|2012-01-03 15:28:18|  5.0|I've taken a lot
…|OyoGAe7OKpv6SyGZT…|
|YjUWPpI6HXG530lwP…|2014-02-05 20:30:30|  3.0|Family diner.
Had…|8g_iMtfSiwikVnbP2…|
|kxX2SOes4o-D3ZQBk…|2015-01-04 00:01:03|  5.0|Wow!  Yummy,
diff…|_7bHUi9Uuf5__HHc_…|
|e4Vwtrqf-wpJfwesg…|2017-01-14 20:54:15|  4.0|Cute interior
and…|bcjbaE6dDog4jkNY9…|
+--------------------+--------------------+-----+--------------------+----------
---------+
only showing top 5 rows
```

[16]:
```
# remove special characters
from pyspark.sql.functions import regexp_replace, col

df_review=df_review.withColumn("text", regexp_replace(col("text"), "'/[^a-z0-9␣
 ↪]+/i'", " "))
```

2.0.1 Build pipeline for Tokenization, stopword, word2vec

[17]:
```
from pyspark.ml import Pipeline
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml.feature import StopWordsRemover
from pyspark.ml.feature import Word2Vec
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml.feature import HashingTF, IDF


tokenizer = Tokenizer(inputCol="text", outputCol="words")  #tokenize words
```

```
remover = StopWordsRemover(inputCol="words", outputCol="filtered")  #remove␣
↪stop words
word2Vec = Word2Vec(vectorSize=100, inputCol="filtered", outputCol="features")
```

2.0.2 Implement Pipeline

[ ]:
```
pipeline = Pipeline(stages=[tokenizer, remover, word2Vec])
df_review_nlp = pipeline.fit(df_review).transform(df_review)
```

```
22/03/12 10:56:48 WARN com.github.fommil.netlib.BLAS: Failed to load
implementation from: com.github.fommil.netlib.NativeSystemBLAS
22/03/12 10:56:48 WARN com.github.fommil.netlib.BLAS: Failed to load
implementation from: com.github.fommil.netlib.NativeRefBLAS
```

[ ]:
```
df_review_nlp.show(5)
```

```
[Stage 17:>                                                         (0 + 1) / 1]
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+
|         business_id|                date|stars|                text|
user_id|               words|            filtered|            features|
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+
|XQfwVwDr-v0ZS3_Cb…|2018-07-07 22:09:11|  3.0|If you decide to …|mh_-
eMZ6K5RLWhZyI…|[if, you, decide,…|[decide, eat, her…|[0.00980463082312…|
|7ATYjTIgM3jUlt4UM…|2012-01-03 15:28:18|  5.0|I've taken a lot
…|OyoGAe7OKpv6SyGZT…|[i've, taken, a, …|[taken, lot,
spin…|[0.06728812187725…|
|YjUWPpI6HXG530lwP…|2014-02-05 20:30:30|  3.0|Family diner.
Had…|8g_iMtfSiwikVnbP2…|[family, diner., …|[family, diner.,
…|[-0.0068525894545…|
|kxX2SOes4o-D3ZQBk…|2015-01-04 00:01:03|  5.0|Wow!  Yummy,
diff…|_7bHUi9Uuf5__HHc_…|[wow!, , yummy,, …|[wow!, , yummy,,
…|[0.03081508976174…|
|e4Vwtrqf-wpJfwesg…|2017-01-14 20:54:15|  4.0|Cute interior
and…|bcjbaE6dDog4jkNY9…|[cute, interior, …|[cute, interior,
…|[0.03522218355248…|
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

[ ]:

[20]:
```
df_review_nlp.count()
```

[20]: 6990280

[21]:
```
df_review_nlp.select('features').show(5)
```

```
[Stage 21:>                                                         (0 + 1) / 1]

+--------------------+
|            features|
+--------------------+
|[0.00980463082312…|
|[0.06728812187725…|
|[-0.0068525894545…|
|[0.03081508976174…|
|[0.03522218355248…|
+--------------------+
only showing top 5 rows
```

[ ]:
```
## Identify if a customer like or dislike the restaurant
```

[ ]:
```
##Step 1: remove star3, convert 1,2 to 0; 4,5 to 1.
```

[22]:
```
#remove star 3
df_review_nlp=df_review_nlp.filter(df_review_nlp.stars != 3)
```

[23]:
```
#categorize stars to bad(0) and good (1)
from pyspark.sql.functions import regexp_replace
df_review_nlp=df_review_nlp.withColumn('stars', regexp_replace('stars', '1.0',␣
 ↪'0'))

df_review_nlp=df_review_nlp.withColumn('stars', regexp_replace('stars', '2.0',␣
 ↪'0'))

df_review_nlp=df_review_nlp.withColumn('stars', regexp_replace('stars', '4.0',␣
 ↪'1'))

df_review_nlp=df_review_nlp.withColumn('stars', regexp_replace('stars', '5.0',␣
 ↪'1'))
```

[24]:
```
df_review_nlp.select('stars').distinct().show()
```

```
[Stage 22:=============================================================>  (38 + 2) / 40]

+-----+
|stars|
```

```
+-----+
|    0|
|    1|
+-----+
```

[26]:
```
from pyspark.sql.types import IntegerType

df_review_nlp = df_review_nlp.withColumn("stars", df_review_nlp["stars"].
 ↪cast(IntegerType()))
```

2.1 Split test and train data

[27]:
```
train_df, test_df = df_review_nlp.randomSplit([.8,.2],seed=666)
```

2.2 Model 1: Logistic Regression Model

[ ]:
```

```

[28]:
```
df_review_nlp.dtypes
```

[28]:
```
[('business_id', 'string'),
 ('date', 'string'),
 ('stars', 'int'),
 ('text', 'string'),
 ('user_id', 'string'),
 ('words', 'array<string>'),
 ('filtered', 'array<string>'),
 ('features', 'vector')]
```

[29]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.classification import LogisticRegression
```

[30]:
```
#Logistic Regression Model
lr = LogisticRegression(featuresCol = 'features', labelCol='stars', maxIter=10)␣
 ↪# maxIter=10, regParam=0.01

# Predict each point's label and show the results.
lrm = lr.fit(train_df)
predictions = lrm.transform(test_df)
```

[31]:
```
predictions.show(5)
```

```
[Stage 51:>                                                         (0 + 1) / 1]
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+-------
------------+--------------------+----------+
|         business_id|                date|stars|                text|
user_id|               words|            filtered|            features|
rawPrediction|         probability|prediction|
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+-------
------------+--------------------+----------+
|-02xFuruu85XmDn2x…|2014-09-11 01:12:16|    1|I have been
going…|7jOaJw3txVFlkHB7Y…|[i, have, been, g…|[going, family,
v…|[0.06484698632777…|[-5.2822425956671…|[0.00505533703999…|       1.0|
|-0Ym1Wg3bXd_TDz8J…|2018-07-09 01:29:36|    1|Neat and trendy
l…|a_qVF8ybTqqTkWrhf…|[neat, and, trend…|[neat, trendy,
li…|[0.11797223139613…|[-5.2221648397870…|[0.00536667914084…|       1.0|
|-0Ym1Wg3bXd_TDz8J…|2018-07-10 02:58:15|    1|I walked in on a
…|JHDxkyjPwuEfC3fkX…|[i, walked, in, o…|[walked, monday,
…|[0.06850417284466…|[-2.8055676331607…|[0.05702405197689…|       1.0|
|-0Ym1Wg3bXd_TDz8J…|2018-08-05 14:42:42|    1|I'm a little late…|Wa-
DgCDkaB3O0xP3c…|[i'm, a, little, …|[little, late,
pa…|[0.04197621637307…|[-6.2599998230808…|[0.00190760024118…|       1.0|
|-0Ym1Wg3bXd_TDz8J…|2018-08-09 21:46:59|    1|Delicious soft
se…|8ag1jJ6yuhJ5YmROr…|[delicious, soft,…|[delicious,
soft,…|[0.10347018678118…|[-3.0765789396712…|[0.04408375552538…|
1.0|
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+-------
------------+--------------------+----------+
only showing top 5 rows
```

[ ]:
```
#print evaluation metrics
#evaluator = BinaryClassificationEvaluator(labelCol="stars",␣
↪rawPredictionCol="prediction")
```

[ ]:
```
#evaluator.evaluate(predictions)
```

[ ]:
```
#evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderPR"})
```

[ ]:
```
#predictions.show(5)
```

[32]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

[33]:
```
#print evaluation metrics
evaluator = MulticlassClassificationEvaluator(labelCol="stars",␣
 ↪predictionCol="prediction")

print('Logistic Regression')
print('accuracy:',evaluator.evaluate(predictions, {evaluator.metricName:␣
 ↪"accuracy"}))
print('f1:',evaluator.evaluate(predictions, {evaluator.metricName: "f1"}))
```

```
Logistic Regression


accuracy: 0.9474397070550382
[Stage 54:=========================================================> (39 + 1) / 40]
f1: 0.9472945457939297
```

2.3 K Fold Cross validation for Logistic Regression

[ ]:
```
df_review_nlp.show()
```

[34]:
```
df_review_nlp=df_review_nlp.withColumnRenamed('stars','label')
```

[35]:
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.linalg import Vectors
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder,␣
 ↪CrossValidatorModel
import tempfile
```

[36]:
```
lr = LogisticRegression()
```

[37]:
```
grid = ParamGridBuilder().addGrid(lr.maxIter, [0, 1]).build()
```

[38]:
```
evaluator = BinaryClassificationEvaluator()
```

[39]:
```
cv = CrossValidator(estimator=lr, estimatorParamMaps=grid, evaluator=evaluator,
    parallelism=2)
```

[ ]:
```
cvModel = cv.fit(df_review_nlp)
```

[ ]:
```
cvModel.getNumFolds()
```

[ ]:
```
3
```

[ ]:
```
# It confirms the function call returns the metrics on the test dataset.
cvModel.avgMetrics[0]
```

[ ]:
```
0.5
```

[ ]:
```
#area under ROC
evaluator.evaluate(cvModel.transform(df_review_nlp))
```

[ ]:
```
0.9277334328910288
```

[ ]:
```
cvModel.avgMetrics
```

[ ]:
```
[0.5, 0.9277328212711858]
```

[ ]:
```

```

[ ]:
```

```

2.4 Random Forest

[45]:
```
from pyspark.ml.classification import RandomForestClassifier
```

[46]:
```
# Set parameters for the Random Forest.
rfc = RandomForestClassifier(maxDepth=5, numTrees=15, impurity="gini",␣
↪labelCol="stars", predictionCol="prediction")

# Fit the model to the data.
rfcm = rfc.fit(train_df)

# Given a dataset, predict each point's label, and show the results.
predictions = rfcm.transform(test_df)
```

[48]:
```
#print evaluation metrics

evaluator = MulticlassClassificationEvaluator(labelCol="stars",␣
↪predictionCol="prediction")

print('Random Forest')
print('accuracy:',evaluator.evaluate(predictions, {evaluator.metricName:␣
↪"accuracy"}))
print('f1:', evaluator.evaluate(predictions, {evaluator.metricName: "f1"}))
```

```
Random Forest


accuracy: 0.8615612887356632
[Stage 188:=====================================================> (39 + 1) / 40]
f1: 0.8471563765287868
```

[ ]:

[49]:
```
from pyspark.ml.classification import LinearSVC
```

[50]:
```
lsvc = LinearSVC(labelCol="stars", maxIter=50,predictionCol="prediction")
```

[51]:
```
rfcm = lsvc.fit(train_df)
```

[52]:
```
predictions = rfcm.transform(test_df)
```

[53]:
```
predictions.show(3)
```

```
[Stage 400:>                                                        (0 + 1) / 1]
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+-------
------------+----------+
|         business_id|                date|stars|                text|
user_id|               words|            filtered|            features|
rawPrediction|prediction|
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+-------
------------+----------+
|-02xFuruu85XmDn2x…|2014-09-11 01:12:16|    1|I have been
going…|7jOaJw3txVFlkHB7Y…|[i, have, been, g…|[going, family,
v…|[0.06484698632777…|[-3.0411404093057…|       1.0|
|-0Ym1Wg3bXd_TDz8J…|2018-07-09 01:29:36|    1|Neat and trendy
l…|a_qVF8ybTqqTkWrhf…|[neat, and, trend…|[neat, trendy,
li…|[0.11797223139613…|[-3.2168363690869…|       1.0|
|-0Ym1Wg3bXd_TDz8J…|2018-07-10 02:58:15|    1|I walked in on a
…|JHDxkyjPwuEfC3fkX…|[i, walked, in, o…|[walked, monday,
…|[0.06850417284466…|[-1.5997814116237…|       1.0|
+--------------------+--------------------+-----+--------------------+----------
---------+--------------------+--------------------+--------------------+-------
------------+----------+
only showing top 3 rows
```

[54]:
```
predictions=predictions.withColumnRenamed('stars','label')
```

[ ]:
```
#print evaluation metrics
print('LinearSVC')
evaluator=MulticlassClassificationEvaluator(metricName="accuracy")
acc = evaluator.evaluate(predictions)
print("Prediction Accuracy: ", acc)


print('accuracy:',evaluator.evaluate(predictions, {evaluator.metricName:␣
↪"accuracy"}))
print('f1:', evaluator.evaluate(predictions, {evaluator.metricName: "f1"}))
```

```
LinearSVC


Prediction Accuracy:  0.9482490349558241


accuracy: 0.9482490349558241
[Stage 405:=====================================================> (39 + 1) / 40]
f1: 0.9481117266482638
```

[ ]:

[ ]: