PySpark MLIB

May 23, 2023

0.1 Installations

[1]:
```
import pyspark
sc = spark.sparkContext

print('Original spark.driver.maxResultSize: ' + sc._conf.get('spark.driver.
 ↪maxResultSize'))
print('Original spark.driver.maxMemory: ' + sc._conf.get('spark.driver.memory'))

spark = SparkSession.builder.getOrCreate()
spark.conf.set("spark.sql.repl.eagerEval.enabled",True)
```

```
Original spark.driver.maxResultSize: 1920m
Original spark.driver.maxMemory: 3840m
```

[2]:
```
import pandas as pd
import numpy as np
pd.set_option('display.max_colwidth', None)
pd.reset_option('display.max_rows')
from itertools import compress
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings(action='ignore')
warnings.simplefilter('ignore')


!pip uninstall -y nltk
!pip install nltk --upgrade --no-cache-dir

import re
from pyspark.ml.feature import MinHashLSH
from pyspark.ml.feature import CountVectorizer,  IDF, CountVectorizerModel,␣
 ↪Tokenizer, RegexTokenizer, StopWordsRemover
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql import Row
```

```python
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords


import os
import shutil
import pandas as pd
# import sh
#from pyspark.sql.functions import *
from pyspark.sql import functions as F
from pyspark.sql.types import *
from google.cloud import storage
import datetime
from pyspark.mllib.linalg import Vector, Vectors
from pyspark.mllib.clustering import LDA, LDAModel
```

```
Found existing installation: nltk 3.6.4
Uninstalling nltk-3.6.4:
  Successfully uninstalled nltk-3.6.4
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv

Collecting nltk
  Downloading nltk-3.8.1-py3-none-any.whl (1.5 MB)
                           1.5/1.5 MB
29.8 MB/s eta 0:00:00a 0:00:01
Requirement already satisfied: click in
/opt/conda/miniconda3/lib/python3.8/site-packages (from nltk) (7.1.2)
Requirement already satisfied: joblib in
/opt/conda/miniconda3/lib/python3.8/site-packages (from nltk) (1.2.0)
Collecting regex>=2021.8.3 (from nltk)
  Downloading
regex-2023.5.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (771 kB)
                         771.9/771.9 kB
234.1 MB/s eta 0:00:00
Requirement already satisfied: tqdm in
/opt/conda/miniconda3/lib/python3.8/site-packages (from nltk) (4.65.0)
Installing collected packages: regex, nltk
  Attempting uninstall: regex
    Found existing installation: regex 2021.4.4
    Uninstalling regex-2021.4.4:
      Successfully uninstalled regex-2021.4.4
Successfully installed nltk-3.8.1 regex-2023.5.5
```

```
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[41]:
```
Edtweet_org2 = spark.read.parquet("gs://msca-bdp-students-bucket/shared_data/
 ↪imajumd0/Edtweet_org")
```

[6]:
```
twitter_raw_full = spark.read.json("gs://msca-bdp-tweets/final_project/")
```

```
23/05/23 12:59:44 WARN
org.apache.spark.sql.execution.datasources.SharedInMemoryCache: Evicting cached
table partition metadata from memory due to size constraints
(spark.sql.hive.filesourcePartitionFileCacheSize = 262144000 bytes). This may
impact query planning performance.
23/05/23 13:05:32 WARN
org.apache.spark.scheduler.cluster.YarnSchedulerBackend$YarnSchedulerEndpoint:
Requesting driver to remove executor 13 for reason Container marked as failed:
container_1684846236830_0001_01_000013 on host: hub-msca-bdp-dphub-students-
backup-imajumd0-sw-3jgm.c.msca-bdp-students.internal. Exit status: -100.
Diagnostics: Container released on a *lost* node.
23/05/23 13:05:32 WARN
org.apache.spark.scheduler.cluster.YarnSchedulerBackend$YarnSchedulerEndpoint:
Requesting driver to remove executor 12 for reason Container marked as failed:
container_1684846236830_0001_01_000012 on host: hub-msca-bdp-dphub-students-
backup-imajumd0-sw-3jgm.c.msca-bdp-students.internal. Exit status: -100.
Diagnostics: Container released on a *lost* node.
23/05/23 13:05:32 ERROR org.apache.spark.scheduler.cluster.YarnScheduler: Lost
executor 13 on hub-msca-bdp-dphub-students-backup-imajumd0-sw-3jgm.c.msca-bdp-
students.internal: Container marked as failed:
container_1684846236830_0001_01_000013 on host: hub-msca-bdp-dphub-students-
backup-imajumd0-sw-3jgm.c.msca-bdp-students.internal. Exit status: -100.
Diagnostics: Container released on a *lost* node.
23/05/23 13:05:32 ERROR org.apache.spark.scheduler.cluster.YarnScheduler: Lost
executor 12 on hub-msca-bdp-dphub-students-backup-imajumd0-sw-3jgm.c.msca-bdp-
students.internal: Container marked as failed:
container_1684846236830_0001_01_000012 on host: hub-msca-bdp-dphub-students-
backup-imajumd0-sw-3jgm.c.msca-bdp-students.internal. Exit status: -100.
Diagnostics: Container released on a *lost* node.
```

[ ]:
```
#write
Edtweet_org_ML.write.format('parquet').save('gs://msca-bdp-students-bucket/
 ↪shared_data/imajumd0/Edtweet_org_ML')
```

```
[ ]: #read
     Edtweet_org_ML = spark.read.parquet("gs://msca-bdp-students-bucket/shared_data/
      ↪imajumd0/Edtweet_org_ML")
```

```
[3]: #if edTweet_org has no retweeted, then use the retweeted subset dataset
     #read
     retweeted_subset = spark.read.parquet("gs://msca-bdp-students-bucket/
      ↪shared_data/imajumd0/retweeted_subset")
     #add org_type to retweeted status
```

```
23/05/23 12:56:42 WARN org.apache.spark.sql.catalyst.util.package: Truncated the
string representation of a plan since it was too large. This behavior can be
adjusted by setting 'spark.sql.debug.maxToStringFields'.
```

0.2 Data Transformations

```
[45]: #Get back the retweeted variable from the original dataset
      Edtweet_org_retweeted2 = Edtweet_org2.join(twitter_raw_full['retweeted', 'id'],␣
       ↪F.col('id') == F.col('tweet_id'), 'left')
```

```
[60]: Edtweet_org_Null = Edtweet_org_retweeted2.filter(F.col('retweeted') == '').
       ↪sample(fraction=0.05, seed=0)
```

```
[62]: Edtweet_org_Null = Edtweet_org_Null.select('org_type', 'user.statuses_count',␣
       ↪'tweet_id', 'tweet_text', 'retweeted') \
                                    .withColumn('retweeted_encoded', F.
       ↪when(Edtweet_org_Null.retweeted == 'RT', 1).otherwise(0)) \
                                    .persist()
```

```
[47]: #Get back the retweeted variable from the original dataset
      Edtweet_org_retweeted = retweeted_subset.join(twitter_raw_full['retweeted',␣
       ↪'id'], F.col('id') == F.col('tweet_id'), 'left')
      #then we take the retweeted subset to develop Edtweet_org, which includes␣
       ↪org_type
```

```
[64]: #We got Edtweet_org from Edtweet_org_retweeted by creating the organization␣
       ↪type feature as below
      Edtweet_org_RT = Edtweet_org.filter(F.col('retweeted') == 'RT').
       ↪sample(fraction=0.0007, seed=0)
```

```
[65]: Edtweet_org_RT = Edtweet_org_RT.select('org_type', 'user.statuses_count',␣
       ↪'tweet_id', 'tweet_text', 'retweeted') \
                                    .withColumn('retweeted_encoded', F.
       ↪when(Edtweet_org_RT.retweeted == 'RT', 1).otherwise(0)) \
                                    .persist()
```

```
23/05/23 14:42:57 WARN org.apache.spark.sql.execution.CacheManager: Asked to
cache already cached data.
```

[66]:
```
#we merge these two datasets to create a balanced sample for ML
Edtweet_org_ML = Edtweet_org_RT.unionAll(Edtweet_org_Null)
```

[68]:
```
Edtweet_org_ML.cache()
```

[68]:
```
DataFrame[org_type: string, statuses_count: bigint, tweet_id: bigint,
tweet_text: string, retweeted: string, retweeted_encoded: int]
```

[69]:
```
retweeted_twt = Edtweet_org_ML.filter(F.col('retweeted_encoded') == 1).count()
origin_twt = Edtweet_org_ML.filter(F.col('retweeted_encoded') == 0).count()

print(f"Tweets that are retweets: {retweeted_twt}")
print(f"Tweets that are original: {origin_twt}")
```

```
[Stage 342:================================================>  (5771 + 18) / 6049]

Tweets that are retweets: 25992
Tweets that are original: 30922
```

0.3 Develop a feature for organization type

[48]:
```
# This code was run in a previous analysis, but is placed here for reference.␣
↪The function defines the rules to categorize user account types i.e.␣
↪'org_type'
def tag_user(profile):
    try:
        followers_count = profile['followers_count']
        verified_status = profile['verified']
        user_description = profile['description']
        user_name = profile['name']
        url = profile['url']

        org_type = 'The Masses'

        if followers_count > 12500 and verified_status == 0 and any(keyword in␣
↪user_description.lower().split() for keyword in ['influencer', 'coach',␣
↪'blogger', 'author', 'certified', 'consultant', 'edtech', 'speaker',␣
↪'trainer', 'founder', 'professor', 'attorney', 'writer', 'entrepreneur',␣
↪'curriculum developer', 'specialist', 'researcher', 'analyst', 'teacher',␣
↪'educator', 'faculty', 'professor']):
            org_type = 'EdTech Influencer'
        elif url is not None and '.gov' in url:
            org_type = 'Govt/State-affiliated'
        elif verified_status == 1 and any(keyword in user_description.lower().
↪split() for keyword in ['nominee', 'state', 'rep', 'democratic',␣
↪'republican', 'independent', 'senator', 'representative', 'candidate',␣
↪'fellow', 'board', 'state', 'governor', 'mayor', 'congressman',␣
↪'congresswoman', 'parliamentarian', 'diplomat', 'activist', 'elected',␣
↪'political', 'organizer', 'cabinet', 'policy', 'justice', 'civil', 'policy',␣
↪'constituency', 'election', 'legislation', 'bipartisan', 'progressive',␣
↪'conservative', 'liberal', 'libertarian', 'grassroots', 'campaign',␣
↪'constituent', 'public']):
            org_type = 'Politicians'
        elif 'jobs' in user_name.lower():
            org_type = 'Job-Boards'
        elif verified_status == 1 and any(keyword in user_description.lower().
↪split() for keyword in ['journalism', 'media', 'news']):
            org_type = 'News Orgs'
        elif any(keyword in user_description.lower().split() for keyword in␣
↪['school', 'schools', 'university', 'community college', 'board',␣
↪'official']) and ('.edu' in url or '.org' in url):
            org_type = 'Edu-Institution'
        elif any(keyword in user_description.lower().split() for keyword in␣
↪['president', 'member', 'director', 'superintendent', 'chancellor',␣
↪'provost', 'dean', 'headmaster', 'headmistress', 'academic director', 'chief␣
↪academic officer', 'chief learning officer', 'coordinator', 'chair',␣
↪'administrator', 'director', 'manager']):
            org_type = 'Leaders in Ed'
        elif followers_count > 150000:
            org_type = 'Other Influencer'

        return org_type
    except Exception as e:
        row_number = profile['id']
        print("NoneType error at user_id:", row_number)
        return None
```

[49]:
```
# Define the UDF wrapper for the tag_user function
tag_user_udf = F.udf(tag_user, StringType())

# Apply the tag_user function using withColumn
Edtweet_org = Edtweet_org_retweeted.filter(F.col('user')['description'].
↪isNotNull()) \
                                   .withColumn('org_type', tag_user_udf(F.
↪col('user')))
```

[50]:
```
# Deal with null values
Edtweet_org = Edtweet_org.withColumn('org_type', F.when(F.col('org_type').
↪isNull(), 'The Masses').otherwise(F.col('org_type')))
```

0.4 Develop a feature to assess originality of a Tweet. We will use Jaccard Similarity

[70]:
```
#create a function that takes as inputs - J.sim threshold, a single column␣
↪named 'text' - and produces duplicates, uniques
def jsim_by_org(df_text_raw, max_dist_threshold):

    rdd_text = df_text_raw.select("id", "text").rdd.filter(lambda x: x['text']␣
↪is not None)

    StopWords = stopwords.words("english")
    tokens = rdd_text \
    .map(lambda x: (x['id'], x['text'].strip().lower().split())) \
    .map(lambda x: (x[0], [word for word in x[1] if len(word) > 1])) \

    row = Row('text')
    df_tokens = spark.createDataFrame(tokens, ["id", "list_of_words"])
    df_tokens = df_tokens.where(F.col('list_of_words').getItem(0).isNotNull())
    vectorize = CountVectorizer(inputCol="list_of_words", outputCol="features",␣
↪minDF=1.0)
    df_vectorize = vectorize.fit(df_tokens).transform(df_tokens)

    mh = MinHashLSH(inputCol="features", outputCol="hashes", numHashTables=5)
    model = mh.fit(df_vectorize)
    df_hashed = mh.fit(df_vectorize).transform(df_vectorize).cache()
    df_hashed_text = df_text_raw.join(df_hashed, "id", how = 'left')

    df_dups_text = model.approxSimilarityJoin(df_hashed_text, df_hashed_text,␣
↪max_dist_threshold).filter("datasetA.id < datasetB.id").select(
                F.col("distCol"),
                F.col("datasetA.id").alias("id_A"),
                F.col("datasetB.id").alias("id_B"),
                F.col('datasetA.text').alias('text_A'),
                F.col('datasetB.text').alias('text_B'))
    df_dups_text.cache()
    records = df_hashed_text.count()
    dups = df_dups_text.select('id_A').distinct().count()
    uniques = records - dups
    return df_dups_text
```

[72]:
```
Jaccard_text_raw = Edtweet_org_ML.select('tweet_text', 'tweet_id') \
                                 .withColumnRenamed('tweet_text', 'text') \
                                 .withColumnRenamed('tweet_id', 'id')


#Jaccard_text_raw.show()
```

[ ]:
```
df_dups_edtweet60  = jsim_by_org(Jaccard_text_raw, .60)
```

```
23/05/23 14:48:54 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 14:49:39 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 14:49:39 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 14:50:21 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:41:52 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:41:54 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
```

[75]:
```
# For each tweet, we take the min(distCol). Two identical tweets will have␣
 ↪DistCol = 0, since it is a measure of how far apart two tweets are.
# Even if a tweet is identical to one other tweet, we consider it "not␣
 ↪original". So we take the min(Distcol)
dups_id_distcol = df_dups_edtweet60.select('id_A', 'distCol') \
                                   .groupBy('id_A') \
                                   .agg(F.min('distCol'))
```

[76]:
```
#for the right tweet_id, add back the distCol variable from df_dups_edtech60 to␣
 ↪the original Edtweet_org_ML dataset.
#Take the minimum distCol for each tweet_id

Edtweet_org_ML = Edtweet_org_ML.join(dups_id_distcol, F.col('id_A') == F.
 ↪col('tweet_id'), 'left')
```

[77]:
```
Edtweet_org_ML = Edtweet_org_ML.withColumnRenamed('min(distCol)',␣
 ↪'Jac_distance')
```

[78]:
```
Edtweet_org_ML.printSchema()
#check the name of statuses_count and check the type for the target variable␣
 ↪(integer?)
```

```
root
 |-- org_type: string (nullable = true)
 |-- statuses_count: long (nullable = true)
 |-- tweet_id: long (nullable = true)
 |-- tweet_text: string (nullable = true)
 |-- retweeted: string (nullable = true)
 |-- retweeted_encoded: integer (nullable = false)
 |-- id_A: long (nullable = true)
 |-- Jac_distance: double (nullable = true)
```

```
[33]: #unique tweets
      Edtweet_org_ML.filter(F.col('Jac_distance').isNull()).count()
```

```
23/05/23 01:41:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1895.9 KiB
23/05/23 01:41:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1895.8 KiB
```

```
[33]: 26590
```

```
[79]: #make the Jac_distance column = 1 if the tweet was identified as an original by␣
       ↪our function
      Edtweet_org_ML = Edtweet_org_ML.withColumn('Jac_distance', F.when(F.
       ↪col('Jac_distance').isNull(), 1).otherwise(F.col('Jac_distance')))
```

```
[34]: # all tweets
      Edtweet_org_ML.count()
```

```
23/05/23 01:43:10 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1889.4 KiB
23/05/23 01:43:13 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1892.6 KiB
```

```
[34]: 30922
```

```
[36]: #duplicate tweets
      df_dups_edtweet60.select('id_A').distinct().count()
```

```
23/05/23 01:44:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1889.1 KiB
23/05/23 01:44:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1896.1 KiB
```

```
[36]: 4332
```

```
[38]: #Did it work?
      Edtweet_org_ML.filter(F.col('Jac_distance').isNull()).count()
```

```
23/05/23 01:48:42 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1896.0 KiB
23/05/23 01:48:46 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1903.2 KiB
```

```
[38]: 0
```

1 Preparing the data for Pyspark MLIB ingestion

1.1 One-hot encoding of categorical variable, org_type

[82]:
```
#VectorAssembler needs to understand org_type as a categorical variable, so we␣
↪one-hot encode it.
from pyspark.ml.feature import StringIndexer, OneHotEncoder

# StringIndexer to convert 'org_type' to numeric indices
stringIndexer = StringIndexer(inputCol='org_type', outputCol='org_type_index')
indexedData = stringIndexer.fit(Edtweet_org_ML).transform(Edtweet_org_ML)

# OneHotEncoder to one-hot encode the indexed column
oneHotEncoder = OneHotEncoder(inputCols=['org_type_index'],␣
↪outputCols=['org_type_encoded'])
encodedData = oneHotEncoder.fit(indexedData).transform(indexedData)
```

```
23/05/23 15:43:12 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:14 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:14 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:37 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
```

[50]:
```
encodedData.show()
```

```
23/05/23 02:14:06 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1896.0 KiB
23/05/23 02:14:09 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 1895.8 KiB
[Stage 264:=================================================>(8492 + 11) / 8573]

+-----------+--------------+-------------------+--------------------+------------
-----+-------------------+------------------+--------------+----------------+---
------------------+
|  org_type|statuses_count|           tweet_id|
tweet_text|retweeted_encoded|                id_A|
Jac_distance|org_type_index|org_type_encoded|            features|
+-----------+--------------+-------------------+--------------------+------------
-----+-------------------+------------------+--------------+----------------+---
------------------+
|The Masses|        176361|1529251476236689408|Thoughts and pray…|
0|                 null|               1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          8066|1530178720228593664|@dwp4401 Police i…|
0|                 null|               1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|           112|1529573741252030471|From first day to…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|        347325|1598166883110641664|@dreamitnowdoit I…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|           983|1529253192009142273|@AndyRichter We k…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|         56833|1529299186931453960|@MikeJVivian @Pop…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          1682|1529256052646064128|This can't contin…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|         14140|1529256341793132545|I couldn't help b…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          1018|1529256366384226309|Praying for the v…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|         30993|1529300396036046849|"Look on the brig…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|         13945|1529230450027560960|@GOP Screw you. T…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|           648|1529230710308995074|Children used to …|
0|1529230710308995074|0.2222222222222222|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|         12755|1529302329236901894|I wonder what els…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          1291|1593249773418864646|    ?…|                 0|
null|               1.0|           0.0|   (8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          5826|1529239484952346624|these kids almost…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          1125|1529219410329620481|I can't stop cryi…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|           552|1529254379533967361|Sending prayers t…|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          9126|1529255033832873985|Another massacre …|
0|                null|                1.0|           0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|          3915|1517847377767538688|    Awesome school!!|
0|                null|                 1.0|            0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
|The Masses|             3|1531385360755859458|I'm so excited, s…|
0|                null|                 1.0|            0.0|
(8,[0],[1.0])|(10,[0,8,9],[1.0,…|
+----------+-------------+------------------+-------------------+------------
-----+-------------------+-----------------+-------------+---------------+---
-----------------+
only showing top 20 rows
```

[83]:
```python
from pyspark.ml.feature import VectorAssembler
```

[84]:
```python
# Select the desired features and use VectorAssembler
features = ['org_type_encoded', 'statuses_count', 'Jac_distance']
assembler = VectorAssembler(inputCols=features, outputCol='features')
encodedData = assembler.transform(encodedData)
#Edtweet_org_ML[['org_type', 'statuses_count', 'distCol', 'features']].show()
```

2 Running Logistic Regression to Predict Whether a Tweet will be Retweeted

[ ]:
```python
from pyspark.ml.classification import LogisticRegression

train, test = encodedData.randomSplit([0.7, 0.3], seed=7)

lr = LogisticRegression(featuresCol='features', labelCol='retweeted_encoded')
model = lr.fit(train)
```

```
23/05/23 15:43:47 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:49 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:50 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:51 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 15:43:52 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:44:17 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:44:18 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:44:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
```

```
large task binary with size 3.0 MiB
23/05/23 15:44:43 WARN com.github.fommil.netlib.BLAS: Failed to load
implementation from: com.github.fommil.netlib.NativeSystemBLAS
23/05/23 15:44:43 WARN com.github.fommil.netlib.BLAS: Failed to load
implementation from: com.github.fommil.netlib.NativeRefBLAS
23/05/23 15:44:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:45:04 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:45:05 WARN org.apache.spark.storage.BlockManager: Asked to remove
block broadcast_162_piece0, which does not exist
23/05/23 15:45:05 WARN org.apache.spark.storage.BlockManager: Asked to remove
block broadcast_162, which does not exist
23/05/23 15:45:05 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:45:29 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:45:30 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:45:51 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:45:52 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:46:12 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:46:13 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:46:54 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:46:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:47:15 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:47:16 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:47:36 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:47:38 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:47:58 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:47:59 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:48:19 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:48:20 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:48:41 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
```

```
large task binary with size 3.0 MiB
23/05/23 15:48:42 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:49:02 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:49:03 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:49:23 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:49:24 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:49:44 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:49:45 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:50:06 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:50:07 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:50:27 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:50:27 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:50:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:50:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:51:08 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:51:09 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:51:29 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:51:30 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:51:50 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:51:51 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:52:11 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:52:12 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:52:32 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:52:33 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:52:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
```

```
large task binary with size 3.0 MiB
23/05/23 15:52:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:53:13 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:53:14 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:53:34 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:53:35 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:53:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:53:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:54:15 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:54:16 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:54:36 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:54:37 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:54:57 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:54:58 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:55:18 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:55:18 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:55:38 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:55:39 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:55:59 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:56:00 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:56:22 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:56:23 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:56:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:56:44 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:57:04 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
```

```
large task binary with size 3.0 MiB
23/05/23 15:57:05 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:57:25 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:57:25 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:57:46 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:57:46 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:58:07 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:58:07 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:58:27 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:58:28 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:58:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:58:49 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:59:09 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:59:10 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:59:30 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:59:30 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:59:50 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 15:59:51 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:00:11 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:00:12 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:00:32 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:00:32 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:00:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:00:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:01:13 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
```

```
large task binary with size 3.0 MiB
23/05/23 16:01:14 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:01:35 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:01:36 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:01:56 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:01:57 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:02:17 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:02:17 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:02:38 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:02:38 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:02:59 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:02:59 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 16:03:01 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 16:03:03 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 16:03:05 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
```

```
[ ]: # Training Summary Data
     trainingSummary = model.summary
     evaluationSummary = model.evaluate(test)
```

```
23/05/23 16:03:06 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 16:03:08 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 16:03:09 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
23/05/23 16:03:11 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 2.9 MiB
```

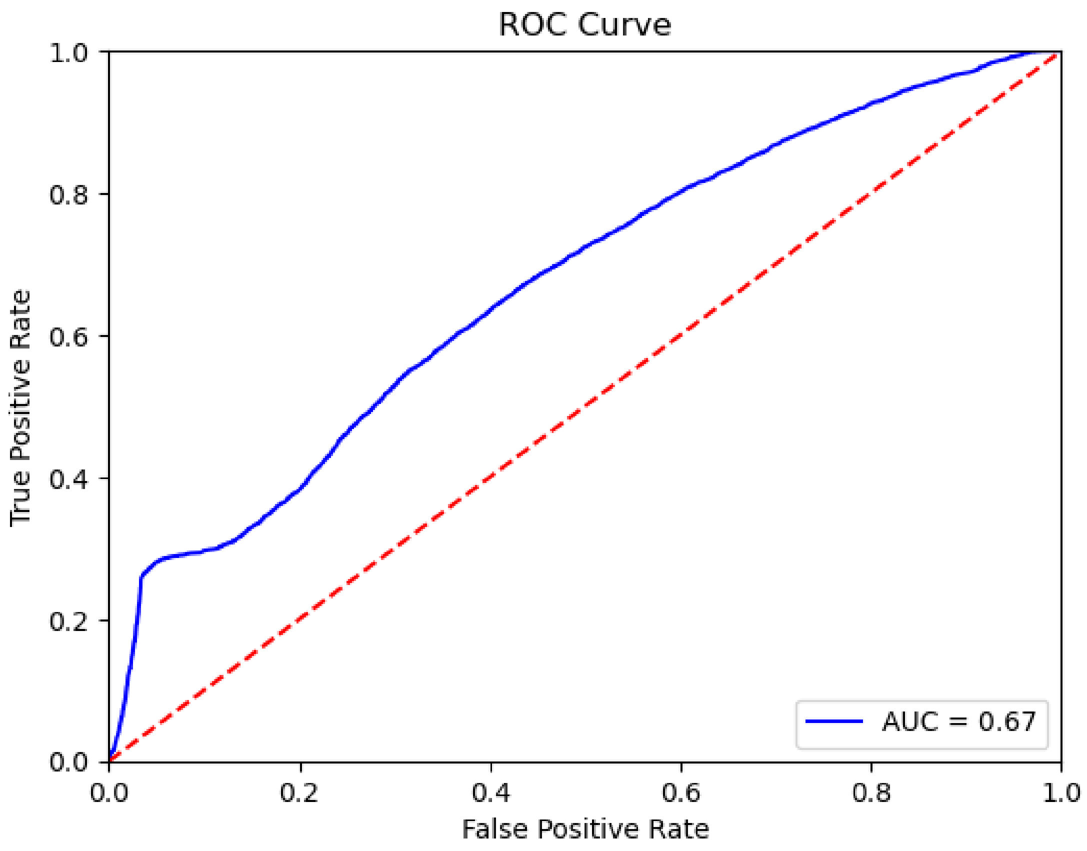
```
[ ]: print("Training AUC: " + str(trainingSummary.areaUnderROC))
     print("Test AUC: ", str(evaluationSummary.areaUnderROC))

     print("\nFalse positive rate by label (Training):")
```

```python
for i, rate in enumerate(trainingSummary.falsePositiveRateByLabel):
    print("label %d: %s" % (i, rate))

print("\nTrue positive rate by label (Training):")
for i, rate in enumerate(trainingSummary.truePositiveRateByLabel):
    print("label %d: %s" % (i, rate))

print("\nTraining Accuracy: " + str(trainingSummary.accuracy))
print("Test Accuracy: ", str(evaluationSummary.accuracy))
```

```
23/05/23 16:03:12 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
23/05/23 16:03:50 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB

Training AUC: 0.6728374087056163

23/05/23 16:04:19 ERROR org.apache.spark.scheduler.AsyncEventQueue: Dropping
event from queue eventLog. This likely means one of the listeners is too slow
and cannot keep up with the rate at which tasks are being started by the
scheduler.
23/05/23 16:04:19 WARN org.apache.spark.scheduler.AsyncEventQueue: Dropped 1
events from eventLog since the application started.

Test AUC:  0.6709072096827704

False positive rate by label (Training):

23/05/23 16:04:23 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB

label 0: 0.64336434193868
label 1: 0.1731892389054955

True positive rate by label (Training):
label 0: 0.8268107610945045
label 1: 0.35663565806131997

Training Accuracy: 0.6127981559430747

23/05/23 16:04:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting
large task binary with size 3.0 MiB
[Stage 758:========================================>          (4737 + 48) / 6049]

Test Accuracy:  0.6090459945888719
```

[88]:
```python
# Get ROC curve and send it to Pandas so that we can plot it
roc_df = evaluationSummary.roc.toPandas()
```

[89]:
```
# Close previous plots; otherwise, will just overwrite and display again
plt.close()

plt.plot(roc_df.FPR, roc_df.TPR, 'b',
         label = 'AUC = %0.2f' % evaluationSummary.areaUnderROC)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve')
plt.show()

%matplot plt
```



```
UsageError: Line magic function `%matplot` not found.
```

[ ]: