l1

January 16, 2024

[1]:
```
# Import other modules not related to PySpark
import os
import sys
import pandas as pd
from pandas import DataFrame
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import matplotlib
from mpl_toolkits.mplot3d import Axes3D
import math
from IPython.core.interactiveshell import InteractiveShell
from datetime import *
import statistics as stats
# This helps auto print out the items without explixitly using 'print'
InteractiveShell.ast_node_interactivity = "all"
%matplotlib inline
```

[2]:
```
# Import PySpark related modules
import pyspark
from pyspark.rdd import RDD
from pyspark.sql import Row
from pyspark.sql import DataFrame
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql import functions
from pyspark.sql.functions import lit, desc, col, size, array_contains\
, isnan, udf, hour, array_min, array_max, countDistinct
from pyspark.sql.types import *

MAX_MEMORY = '15G'
# Initialize a spark session.
conf = pyspark.SparkConf().setMaster("local[*]") \
        .set('spark.executor.heartbeatInterval', 10000) \
        .set('spark.network.timeout', 10000) \
        .set("spark.core.connection.ack.wait.timeout", "3600") \
        .set("spark.executor.memory", MAX_MEMORY) \
        .set("spark.driver.memory", MAX_MEMORY)
def init_spark():
    spark = SparkSession \
        .builder \
        .appName("COVID") \
        .config(conf=conf) \
        .getOrCreate()
    return spark

spark = init_spark()
filename_data = 'the-reddit-covid-dataset-comments.csv'
# Load the main data set into pyspark data frame
df = spark.read.csv(filename_data, header=True, inferSchema=True)
print('Data frame type: ' + str(type(df)))
```

```
Data frame type: <class 'pyspark.sql.dataframe.DataFrame'>
```

[3]:
```
#
df = df.sample(withReplacement=False, fraction=0.01, seed=42)
df = df.limit(2000)
```

[4]:
```
print('Columns overview')
pd.DataFrame(df.dtypes, columns = ['Column Name','Data type'])
```

```
Columns overview
```

[4]:
```
      Column Name Data type
0            type    string
1              id    string
2    subreddit.id    string
3  subreddit.name    string
4  subreddit.nsfw    string
5     created_utc    string
6       permalink    string
7            body    string
8       sentiment    string
9           score    string
```

[5]:
```
#
new_column_names = ['type', 'id', 'subreddit_id', 'subreddit_name',␣
 ↪'subreddit_nsfw', 'created_utc', 'permalink', 'body', 'sentiment', 'score']
#                               ,
df = df.toDF(*new_column_names)

pd.DataFrame(df.dtypes, columns = ['Column Name','Data type'])
df.printSchema()
```

[5]:
```
      Column Name Data type
0            type    string
1              id    string
2    subreddit_id    string
3  subreddit_name    string
4  subreddit_nsfw    string
5     created_utc    string
6       permalink    string
7            body    string
8       sentiment    string
9           score    string
```

```
root
 |-- type: string (nullable = true)
 |-- id: string (nullable = true)
 |-- subreddit_id: string (nullable = true)
 |-- subreddit_name: string (nullable = true)
 |-- subreddit_nsfw: string (nullable = true)
 |-- created_utc: string (nullable = true)
 |-- permalink: string (nullable = true)
 |-- body: string (nullable = true)
 |-- sentiment: string (nullable = true)
 |-- score: string (nullable = true)
```

[6]:
```
#
selected_columns = ['id', 'subreddit_name', 'subreddit_nsfw', 'sentiment',␣
 ↪'score']
df = df.select(selected_columns)

#
df.show(4)
```

```
+-------+--------------+--------------+---------+-----+
|     id|subreddit_name|subreddit_nsfw|sentiment|score|
+-------+--------------+--------------+---------+-----+
|   NULL|          NULL|          NULL|     NULL| NULL|
|hi1v4vl|        canada|         false|  -0.7269|    1|
|hi1uyme|  conservative|         false|     NULL| NULL|
|hi1udht|       jontron|         false|     NULL| NULL|
+-------+--------------+--------------+---------+-----+
only showing top 4 rows
```

[7]:
```
df = df.na.drop(subset=['id', 'subreddit_name', 'subreddit_nsfw', 'sentiment',␣
 ↪'score'])
df.show(2)
```

```
+-------+--------------+--------------+---------+-----+
|     id|subreddit_name|subreddit_nsfw|sentiment|score|
+-------+--------------+--------------+---------+-----+
|hi1v4vl|        canada|         false|  -0.7269|    1|
|hi1satq|           nrl|         false|  -0.7506|   46|
+-------+--------------+--------------+---------+-----+
only showing top 2 rows
```

[8]:
```
#               "sentiment"
df = df.withColumn("sentiment", col("sentiment").cast(FloatType()))

#               "sentiment"
df = df.withColumn("score", col("score").cast(FloatType()))
```

[9]:
```
#
quantiles = df.stat.approxQuantile(["sentiment", "score"], [0.25, 0.75], 0.05)

#                    (IQR)
IQR_sentiment = quantiles[0][1] - quantiles[0][0]
IQR_score = quantiles[1][1] - quantiles[1][0]

#
lower_bound_sentiment = quantiles[0][0] - 1.5 * IQR_sentiment
upper_bound_sentiment = quantiles[0][1] + 1.5 * IQR_sentiment

lower_bound_score = quantiles[1][0] - 1.5 * IQR_score
upper_bound_score = quantiles[1][1] + 1.5 * IQR_score

#
df_filtered = df.filter((col("sentiment").between(lower_bound_sentiment,␣
 ↪upper_bound_sentiment)) &
                        (col("score").between(lower_bound_score,␣
 ↪upper_bound_score)))

#
df_filtered.show(3)
```

```
+-------+--------------+--------------+---------+-----+
|     id|subreddit_name|subreddit_nsfw|sentiment|score|
+-------+--------------+--------------+---------+-----+
|hi1v4vl|        canada|         false|  -0.7269|  1.0|
|hi1q4qb|toiletpaperusa|         false|   0.4815|  3.0|
|hi1pi1o|    ukpolitics|         false|  -0.9432|  1.0|
+-------+--------------+--------------+---------+-----+
only showing top 3 rows
```

[10]:
```
#
summary = df_filtered.describe()

#
summary.show(3)
```

```
+-------+----+--------------+--------------+--------------------+---------------
---+
|summary|  id|subreddit_name|subreddit_nsfw|           sentiment|
score|
+-------+----+--------------+--------------+--------------------+---------------
---+
|  count| 246|           246|           246|                 246|
246|
|   mean|NULL|          NULL|
NULL|-0.00225569022349…|2.0853658536585367|
| stddev|NULL|          NULL|          NULL|
0.5579037134764742|1.9369735672689639|
+-------+----+--------------+--------------+--------------------+---------------
---+
only showing top 3 rows
```

[11]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import mean
#
mean_score = df_filtered.select(mean("score").alias("mean_score")).
  ↪collect()[0]["mean_score"]

#             (25%, 50%, 75%)         "score"
quantiles_score = df_filtered.stat.approxQuantile("score", [0.25, 0.5, 0.75], 0.
  ↪05)

#
print(f"Mean Score: {mean_score}")
print(f"25th Percentile: {quantiles_score[0]}")
print(f"50th Percentile (Median): {quantiles_score[1]}")
print(f"75th Percentile: {quantiles_score[2]}")
```

```
Mean Score: 2.0853658536585367
25th Percentile: 1.0
50th Percentile (Median): 1.0
75th Percentile: 2.0
```

[12]:
```
#
mean_score = df_filtered.select(mean("sentiment").alias("mean_sentiment")).
  ↪collect()[0]["mean_sentiment"]
```

```
#              (25%, 50%, 75%)         "sentiment"
quantiles_score = df_filtered.stat.approxQuantile("sentiment", [0.25, 0.5, 0.
 ↪75], 0.05)

#
print(f"Mean Score: {mean_score}")
print(f"25th Percentile: {quantiles_score[0]}")
print(f"50th Percentile (Median): {quantiles_score[1]}")
print(f"75th Percentile: {quantiles_score[2]}")
```

```
Mean Score: -0.002255690223499527
25th Percentile: -0.4018999934196472
50th Percentile (Median): 0.0
75th Percentile: 0.3400000035762787
```

[13]:
```
from pyspark.sql.functions import col, count, lit, when, sum
#                       subreddit
subreddit_counts = df_filtered.groupBy("subreddit_name").agg(count("*").
 ↪alias("count"))

#
total_count = df_filtered.count()

#
subreddit_percentages = subreddit_counts.withColumn(
    "percentage",
    (col("count") / total_count * 100).cast("double")
)

#                            Other (    ,     ,                      )
threshold = 1

#                        Other
subreddit_percentages_filtered = (
    subreddit_percentages
    .withColumn("subreddit_grouped",
                when(col("count") > threshold, col("subreddit_name"))
                .otherwise(lit("Other")))
    .groupBy("subreddit_grouped")
    .agg(sum("count").alias("total_count"), sum("percentage").
 ↪alias("total_percentage"))
    .orderBy("total_percentage", ascending=False)
)

#                  Pandas DataFrame
subreddit_percentages_filtered_pd = subreddit_percentages_filtered.toPandas()
```

```
#                    truncate=False
print(subreddit_percentages_filtered_pd)
```

```
        subreddit_grouped  total_count  total_percentage
0                   Other          128         52.032520
1               askreddit           12          4.878049
2              conspiracy           12          4.878049
3          hermancainaward           12          4.878049
4              newzealand            5          2.032520
5               worldnews            5          2.032520
6       whitepeopletwitter            4          1.626016
7                 antiwork            4          1.626016
8                 politics            4          1.626016
9                   soccer            4          1.626016
10      anarcho_capitalism            3          1.219512
11             libertarian            3          1.219512
12               byebyejob            3          1.219512
13           amitheasshole            3          1.219512
14                facepalm            3          1.219512
15                joerogan            3          1.219512
16                 science            3          1.219512
17       louderwithcrowder            3          1.219512
18              ukpolitics            2          0.813008
19                  movies            2          0.813008
20         unpopularopinion            2          0.813008
21          politicalhumor            2          0.813008
22                 ireland            2          0.813008
23          debatevaccines            2          0.813008
24      lockdownskepticism            2          0.813008
25                  tinder            2          0.813008
26               singapore            2          0.813008
27                 sanjose            2          0.813008
28            stopdrinking            2          0.813008
29   politicalcompassmemes            2          0.813008
30          fantasyfootball            2          0.813008
31           askanamerican            2          0.813008
32                   holup            2          0.813008
33               teenagers            2          0.813008
```

[21]:
```
nsfw_column = "subreddit_nsfw"

#
df_filtered = df_filtered.withColumn(nsfw_column, col(nsfw_column).
 ↪cast("boolean"))

#              NSFW
```

```python
grouped_df = df_filtered.groupBy(nsfw_column).count()

#
nsfw_count = grouped_df.filter(col(nsfw_column) == True).select("count").
 ↪first()[0]
sfw_count = grouped_df.filter(col(nsfw_column) == False).select("count").
 ↪first()[0]

#
total_count = nsfw_count + sfw_count
nsfw_percentage = (nsfw_count / total_count) * 100
sfw_percentage = (sfw_count / total_count) * 100

#
print(f"NSFW: {nsfw_count}      , {nsfw_percentage:.2f}%")
print(f"SFW: {sfw_count}      , {sfw_percentage:.2f}%")

#
plt.show()
```

```
NSFW: 1      , 0.41%
SFW: 245      , 99.59%
```

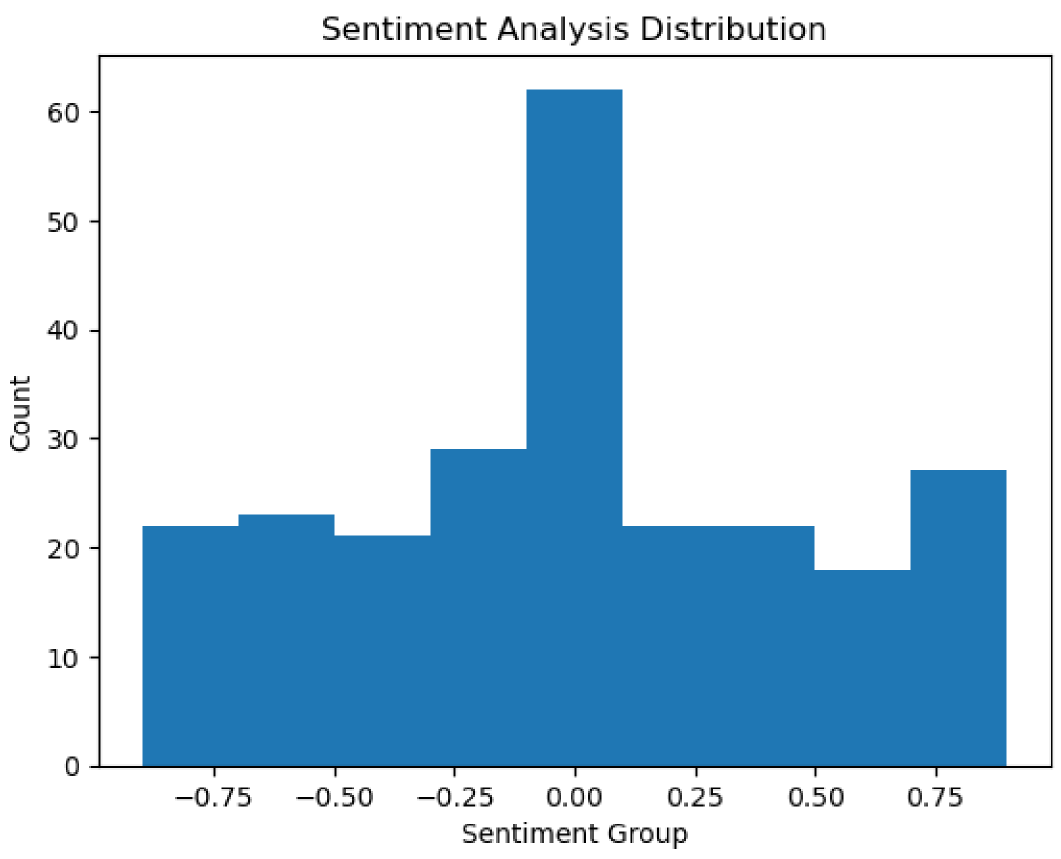
[18]:
```python
#
grouped_df = df_filtered.groupBy(((col("sentiment") / 0.2).cast("int") * 0.2).
 ↪alias("sentiment_group")).count()

#
grouped_df = grouped_df.sort("sentiment_group")

#          Pandas DataFrame                 matplotlib
pandas_df = grouped_df.toPandas()

#
plt.bar(pandas_df["sentiment_group"], pandas_df["count"], width=0.2)
plt.xlabel("Sentiment Group")
plt.ylabel("Count")
plt.title("Sentiment Analysis Distribution")
plt.show()
```

[18]:
```
<BarContainer object of 9 artists>
```

[18]:
```
Text(0.5, 0, 'Sentiment Group')
```

[18]:
```
Text(0, 0.5, 'Count')
```

[18]:
```
Text(0.5, 1.0, 'Sentiment Analysis Distribution')
```



[19]:
```
#          subreddit_name             score         subreddit_name
subreddit_scores = df_filtered.groupBy("subreddit_name").sum("score")

#                      score        -10
top_subreddits = subreddit_scores.sort(col("sum(score)").desc()).limit(10)

#
top_subreddits.show()
```

```
+--------------------+----------+
|      subreddit_name|sum(score)|
+--------------------+----------+
|     hermancainaward|      46.0|
|          conspiracy|      38.0|
|          newzealand|      21.0|
|           askreddit|      18.0|
|  whitepeopletwitter|      13.0|
|           worldnews|      12.0|
|              soccer|      12.0|
|  lockdownskepticism|      10.0|
|             ireland|       8.0|
|lockdownskepticismau|       8.0|
+--------------------+----------+
```

[20]:
```
#
correlation = df_filtered.corr("sentiment", "score")

#
print(f"Correlation between sentiment and score: {correlation}")
```

```
Correlation between sentiment and score: -0.018082291461601126
```