PredictSessionLengthURLVisits

October 26, 2017

```
In [3]: from pyspark.sql import SparkSession

        # Build the SparkSession
        spark = SparkSession.builder \
            .master("local") \
            .appName("Predict Session Length for given IP") \
            .config("spark.executor.memory", "1gb") \
            .getOrCreate()
        sc = spark.sparkContext

        # Load the data by creating rdd
        rdd = sc.textFile('/home/hassan/Side_Projects/WeblogChallenge/data/2015_07_22_mktplace_s
        # split the data into columns
        rdd = rdd.map(lambda line: line.split(" "))

        # =====================================
        # Manipulating data
        # =====================================
        from pyspark.sql import Row
        from pyspark.sql.types import *
        from pyspark.sql.functions import *

        #Map the RDD to a DF for better performance
        mainDF = rdd.map(lambda line: Row(timestamp=line[0], ipaddress=line[2].split(':')[0],url
        # convert timestamps from string to timestamp datatype
        mainDF = mainDF.withColumn('timestamp', mainDF['timestamp'].cast(TimestampType()))
        # sessionizing data based on 15 min fixed window time
        # assign an Id to each session
        SessionDF = mainDF.select(window("timestamp", "15 minutes").alias('FixedTimeWindow'),'ti
        SessionDF = SessionDF.withColumn("SessionId", monotonically_increasing_id())
        # join the time stamps and url to the Sessionized DF
        dfWithTimeStamps = mainDF.select(window("timestamp", "15 minutes").alias('FixedTimeWindo
        SessionDF = dfWithTimeStamps.join(SessionDF,['FixedTimeWindow','ipaddress'])
        # Finding the first hit time of each ip for each session and join in to our session df
        FirstHitTimeStamps = SessionDF.groupBy("SessionId").agg(min("timestamp").alias('FristHit
        SessionDF = FirstHitTimeStamps.join(SessionDF,['SessionId'])
        timeDiff = (unix_timestamp(SessionDF.timestamp)-unix_timestamp(SessionDF.FristHitTime))
        SessionDF = SessionDF.withColumn("timeDiffwithFirstHit", timeDiff)
        tmpdf = SessionDF.groupBy("SessionId").agg(max("timeDiffwithFirstHit").alias("SessionDur
        SessionDF = SessionDF.join(tmpdf,['SessionId'])


        # for any given IP if we don't have any previous log from that IP
        # the prediction for it's session length is the average session length
        meandf = SessionDF.groupBy().avg('SessionDuration')
        meandf.show()
```

```
+--------------------+
|avg(SessionDuration)|
+--------------------+
|  141.58578161415625|
+--------------------+
```

```
In [4]: # if the given ip has a record in the following table
        # the prediction for it's session length is the it's previous session's average
        meanSessionIP = SessionDF.groupBy("ipaddress").agg(avg("SessionDuration").alias('Average
        meanSessionIP.show(20)
```

```
+---------------+--------------------------+
|      ipaddress|AverageSessionDurationForIP|
+---------------+--------------------------+
|   27.62.30.188|                      33.0|
|  120.63.59.185|                       0.0|
|  115.69.247.81|                      30.0|
|  59.95.113.108|                      10.0|
|122.175.225.152|         226.5769230769230 7|
|   14.139.60.13|                       0.0|
|117.232.164.217|                       1.5|
| 59.160.110.163|                     165.5|
| 101.62.250.135|                      27.0|
| 107.167.99.177|                      53.0|
| 121.246.85.180|                       4.0|
|  14.98.247.140|                       5.0|
|    1.23.208.26|        10.833333333333334|
|  59.177.37.135|                       0.0|
|122.175.153.217|                       0.0|
| 223.176.174.84|                       2.0|
| 223.225.252.41|                     118.0|
|122.181.181.211|                     173.0|
|   103.15.63.34|        46.666666666666664|
|123.238.121.215|                       3.0|
+---------------+--------------------------+
only showing top 20 rows
```

```
In [6]: # For Predicting the number of unique url visits for a given IP
        # again if we don't have the IP in our logs, the predicted value is the
        # average unique url visit by any IP
        SessionDF.groupBy("ipaddress","url").count().distinct().groupBy().agg(avg("count")).show
```

```
+------------------+
|        avg(count)|
+------------------+
|1.3430700599135612|
+------------------+
```

```
In [8]: # if we have the give ip in the records
        # we can find the average previous unique url visits of that IP in the following table
        SessionDF.groupBy("ipaddress","url").count().distinct().groupBy("ipaddress").agg(avg("co
```

```
+---------------+------------------+
|      ipaddress|        avg(count)|
+---------------+------------------+
| 59.160.110.163|1.4285714285714286|
| 117.241.152.20|1.5588235294117647|
|  202.174.92.10|               1.0|
|  61.16.142.162|            1.0625|
| 117.205.39.248|            1.3125|
|117.203.181.144|               1.0|
|115.112.250.108|               1.0|
|  202.53.89.132|1.2272727272727273|
| 117.247.188.13|               1.0|
|  14.139.82.134|2.0555555555555554|
|   120.61.47.36|1.1428571428571428|
|   27.63.186.72|               1.0|
| 113.193.114.25|               1.0|
|123.136.182.137|               1.2|
|  27.34.244.251|1.1696428571428572|
| 124.125.22.218|               1.0|
| 117.207.97.173|               1.0|
|   61.0.225.164|1.3333333333333333|
|117.218.161.174|1.4210526315789473|
|   61.2.172.171|               1.0|
+---------------+------------------+
only showing top 20 rows
```

```
In [ ]:
```