IBM **Watson Studio** Projects Tools Community Services Manage

Assignment3.1

VERSION AUTHOR	LAST UPDATED	LANGUAGE
Mariusz Jaskowski	4 Jul 2018, 3:20 PM	Python 2 with Spark 2.1

Assignment 3

Welcome to Assignment 3. This will be even more fun. Now we will calculate statistical measures on the test data you have created.

YOU ARE NOT ALLOWED TO USE ANY OTHER 3RD PARTY LIBRARIES LIKE PANDAS. PLEASE ONLY MODIFY CONTENT INSIDE THE FUNCTION SKELETONS Please read why: https://www.coursera.org/learn/exploring-visualizing-iot-data/discussions/weeks/3/threads/skjCbNgeEeapeQ5W6suLkA (https://www.coursera.org/learn/exploring-visualizing-iot-data/discussions/weeks/3/threads/skjCbNgeEeapeQ5W6suLkA) . Just make sure you hit the play button on each cell from top to down. There are seven functions you have to implement. Please also make sure than on each change on a function you hit the play button again on the corresponding cell to make it available to the rest of this notebook. Please also make sure to only implement the function bodies and DON'T add any additional code outside functions since this might confuse the autograder.

So the function below is used to make it easy for you to create a data frame from a cloudant data frame using the so called "DataSource" which is some sort of a plugin which allows ApacheSpark to use different data sources.

In [1]:

```
#Please don't modify this function
def readDataFrameFromCloudant(database):
    cloudantdata=spark.read.load(database, "org.apache.bahir.cloudant")

    cloudantdata.createOrReplaceTempView("washing")
    spark.sql("SELECT * from washing").show()
    return cloudantdata
```

All functions can be implemented using DataFrames, ApacheSparkSQL or RDDs. We are only interested in the result. You are given the reference to the data frame in the "df" parameter and in case you want to use SQL just use the "spark" parameter which is a reference to the global SparkSession object. Finally if you want to use RDDs just use "df.rdd" for obtaining a reference to the underlying RDD object.

Let's start with the first function. Please calculate the minimal temperature for the test data set you have created. We've provided a little skeleton for you in case you want to use SQL. You can use this skeleton for all subsequent functions. Everything can be implemented using SQL only if you like.

In [2]:

```
def minTemperature(df,spark):
    return spark.sql("SELECT MIN(temperature) AS mintemp FROM washing")
.first().mintemp
```

Please now do the same for the mean of the temperature

In [3]:

```
def meanTemperature(df,spark):
    return spark.sql("SELECT AVG(temperature) AS avgtemp FROM washing")
.first().avgtemp
```

Please now do the same for the maximum of the temperature

```
In [4]: def maxTemperature(df,spark):
            return spark.sql("SELECT MAX(temperature) AS maxtemp FROM washing")
        .first().maxtemp
```

Please now do the same for the standard deviation of the temperature

```
In [5]: def sdTemperature(df,spark):
            return spark.sql("SELECT STDDEV(temperature) AS sdtemp FROM washin
        g").first().sdtemp
```

Please now do the same for the skew of the temperature. Since the SQL statement for this is a bit more complicated we've provided a skeleton for you. You have to insert custom code at four position in order to make the function work. Alternatively you can also remove everything and implement if on your own. Note that we are making use of two previously defined functions, so please make sure they are correct. Also note that we are making use of python's string formatting capabilitis where the results of the two function calls to "meanTemperature" and "sdTemperature" are inserted at the "%s" symbols in the SQL string.

```
In [6]: def skewTemperature(df,spark):
            return spark.sql("""
            SELECT
                (
                    1/Float(COUNT(temperature))
                ) *
                SUM (
                    POWER((temperature - %s), 3) / POWER(%s, 3)
                ) AS skwtemp FROM washing"""
                             %(meanTemperature(df,spark),sdTemperature(df,spark
        ))).first().skwtemp
```

Kurtosis is the 4th statistical moment, so if you are smart you can make use of the code for skew which is the 3rd statistical moment. Actually only two things are different.

```
In [7]: def kurtosisTemperature(df,spark):
            return spark.sql("""
            SELECT
                SUM (
                    POWER((temperature - %s), 4) / POWER(%s, 4)
                ) /
                (
                    Float(COUNT(temperature))
                )
                AS krttemp FROM washing"""
                             %(meanTemperature(df,spark),sdTemperature(df,spark
        ))).first().krttemp
```

Just a hint. This can be solved easily using SQL as well, but as shown in the lecture also using RDDs.

```
In [19]: def meanHardness(df,spark):
             return spark.sql("SELECT AVG(hardness) as avghard from washing").fi
         rst().avghard
```

```
def sdHardness(df,spark):
    return spark.sql("SELECT STDDEV(hardness) as sdhard from washing").
first().sdhard
def correlationTemperatureHardness(df,spark):
    return spark.sql("""
SELECT
    (
        SUM((temperature-%s) * (hardness-%s)) / Float(COUNT(temperatur
e))
    ) /
    (
        %s * %s
    )
AS cortemphrd FROM washing
                    """ %(meanTemperature(df,spark),meanHardness(df,spa
rk),sdTemperature(df,spark),sdHardness(df,spark))).first().cortemphrd
```

PLEASE DON'T REMOVE THIS BLOCK - THE FOLLOWING CODE IS NOT GRADED

axx

PLEASE DON'T REMOVE THIS BLOCK - THE FOLLOWING CODE IS NOT GRADED

Now it is time to connect to the cloudant database. Please have a look at the Video "Overview of end-to-end scenario" of Week 2 starting from 6:40 in order to learn how to obtain the credentials for the database. Please paste this credentials as strings into the below code

TODO Please provide your Cloudant credentials here

In [9]:

```
hostname = "f17dd5c1-740d-4923-91a1-e13b46ac747e-bluemix.cloudant.com"
user = "f17dd5c1-740d-4923-91a1-e13b46ac747e-bluemix"
pw = "dd5d752a6306ff1738fd83ea17e6487ac787346965891b356573b732dd7e81e2"
database = "washing" #as long as you didn't change this in the NodeRED
 flow the database name stays the same
```

In [10]:

```
spark = SparkSession\
    .builder\
    .appName("Cloudant Spark SQL Example in Python using temp tables")\
    .config("cloudant.host",hostname)\
    .config("cloudant.username", user)\
    .config("cloudant.password",pw)\
    .getOrCreate()
cloudantdata=readDataFrameFromCloudant(database)
```

```
+--------------------+--------------------+-----+--------+----------+--
-------+--------+-----+-----------+-------------+-------+
|                 _id|                _rev|count|flowrate|fluidlevel|fr
equency|hardness|speed|temperature|           ts|voltage|
+--------------------+--------------------+-----+--------+----------+--
-------+--------+-----+-----------+-------------+-------+
|28e0a6047fbb61bc1...|1-0516b5694f9cde5...|    1|      11|acceptable|
   null|      72| null|         96|1530406897515|   null|
|28e0a6047fbb61bc1...|1-e0d4def254349ef...|    1|    null|      null|
   null|    null| 1056|        null|1530406901519|   null|
|28e0a6047fbb61bc1...|1-4b63d38095a17e4...|    9|      11|acceptable|
   null|      76| null|          99|1530406905525|   null|
|28e0a6047fbb61bc1...|1-6306e41a8743c5f...|    2|    null|      null|
   null|    null| 1033|        null|1530406906520|   null|
|28e0a6047fbb61bc1...|1-fdfed10e51e00a5...|   13|      11|acceptable|
   null|      76| null|          86|1530406909534|   null|
|28e0a6047fbb61bc1...|1-6daa45c1e6e8c47...|    3|    null|      null|
   null|    null| 1031|        null|1530406911522|   null|
|28e0a6047fbb61bc1...|1-0abe6486f53bb03...|   17|      11|acceptable|
   null|      74| null|          96|1530406913540|   null|
|28e0a6047fbb61bc1...|1-d71c991a062ff5e...|   24|      11|acceptable|
   null|      79| null|          89|1530406920547|   null|
|28e0a6047fbb61bc1...|1-fd2035da60487ea...|   35|      11|acceptable|
   null|      70| null|          97|1530406931572|   null|
|28e0a6047fbb61bc1...|1-4d616ab1c995cf9...|   54|      11|acceptable|
   null|      70| null|          90|1530406950615|   null|
|28e0a6047fbb61bc1...|1-988eb68f432da53...|   64|      11|acceptable|
   null|      71| null|          92|1530406960633|   null|
|28e0a6047fbb61bc1...|1-d7bd3b8754dd584...|   70|      11|acceptable|
   null|      74| null|          95|1530406966643|   null|
|28e0a6047fbb61bc1...|1-a0044025cea1578...|   75|      11|acceptable|
   null|      80| null|          87|1530406971649|   null|
|28e0a6047fbb61bc1...|1-09e963537e45973...|   76|      11|acceptable|
   null|      73| null|          92|1530406972652|   null|
|28e0a6047fbb61bc1...|1-c72148411906d3f...|   84|      11|acceptable|
   null|      77| null|          84|1530406980664|   null|
|28e0a6047fbb61bc1...|1-79943ce50a55c94...|   95|      11|acceptable|
   null|      72| null|         100|1530406991684|   null|
|28e0a6047fbb61bc1...|1-76437369291ad0e...|  123|      11|acceptable|
   null|     110| null|          83|1530407019738|   null|
|28e0a6047fbb61bc1...|1-c60795fbbbd2cbd...|  149|      11|acceptable|
   null|      79| null|          85|1530407045789|   null|
|28e0a6047fbb61bc1...|1-f126164c620852a...|  156|      11|acceptable|
   null|      78| null|          85|1530407052804|   null|
|28e0a6047fbb61bc1...|1-69b968fbc8c9945...|  158|      11|acceptable|
   null|      72| null|          94|1530407054806|   null|
+--------------------+--------------------+-----+--------+----------+--
-------+-------+-----+----------+------------+-------+
only showing top 20 rows
```

```
In [11]: minTemperature(cloudantdata,spark)
```

```
Out[11]: 80
```

```
In [12]: meanTemperature(cloudantdata,spark)
```

```
Out[12]: 90.20837209302326
```