DSCI 617 – HW 01 Instructions

General Instructions

Navigate to the **Homework** folder inside of your user directory in the Databricks workspace. Create a notebook named **HW_01** inside the **Homework** folder.

Any set of instructions you see in this document with an orange bar to the left will indicate a place where you should create a markdown cell. For each new problem, create a markdown cell that indicates the title of that problem as a level 2 header. Any set of instructions you see with a blue bar to the left will provide instructions for creating a single code cell.

Add a markdown cell that displays the following text as a level 1 header: **DSCI 617 – Homework 01**. Within the same cell, on the line below the header, add your name in bold.

Add a code cell that imports that **`math`** package as well as the **`SparkSession`** class. Also include the following import statement: **`from pyspark.mllib.random import RandomRDDs`**

Add another code cell to create **`SparkSession`** and **`sparkContext`** objects named **`spark`** and **`sc`**.

Problem 1: Terminology

Create a markdown containing a numbered list with answers to the following questions. You only need to provide the answers and are not required to retype the questions. Each question requires only a 1 – 3 word answer.

1. Spark provides APIs for Python, Scala, Java, and R. But it was written in what language?
2. What is the type of object that represents the primary entry point for accessing Spark functionality?
3. What is the type of object that provides tools for working with RDDs?
4. What does the acronym RDD stand for?
5. When an RDD is created, it is broken into smaller pieces that can be distributed over the cluster. What are these pieces called?
6. Does a transformation or an action return a new RDD?
7. What type of method triggers the evaluation of an RDD, a transformation or an action?
8. Is **`sample()`** a transformation or an action?
9. Is **`take()`** a transformation or an action?
10. Is **`map()`** a transformation or an action?
11. Is **`reduce()`** a transformation or an action?
12. What data type is returned by the **`collect()`** method?
13. What is the term used for the node that manages the other nodes in a cluster?
14. What is the term used for the other nodes in a cluster?
15. What is the term used for the process that manages tasks in a Spark application?
16. What is the term used for the processes that perform tasks in a Spark application?

Problem 2: Working with a Numerical RDD

In the first code cell for this problem, we will create a randomly generated RDD and will then calculate some descriptive statistics for this RDD.

Complete the following steps in a single code cell:

1. Use the following line of code to create an RDD containing 1.2 million elements randomly selected from the interval `[0,1]`: **`random_rdd = RandomRDDs.uniformRDD(sc, size=1200000, seed=1)`**
2. Use built-in RDD methods to calculate the sum, mean, standard deviation, minimum, and maximum of the values stored in **`random_rdd`**. Display the results in the format shown below, replacing the `xxxx` strings with the appropriate values. Add spacing to ensure that the numerical values are left-aligned.

```
Sum:      xxxx
Mean:     xxxx
Std Dev:  xxxx
Minimum:  xxxx
Maximum:  xxxx
```

In the second cell in Problem 2, we will explore how the RDD we created has been partitioned.

Complete the following steps in a single code cell:

1. Use the **`getNumPartitions()`** RDD method to determine the number of partitions used for **`random_rdd`**.
2. Use **`glom()`** and **`map()`** along with the built-in Python function **`len()`** to create a list that contains the number of elements contained within each of the partitions of **`random_rdd`**. Try to complete this task with a single line of code by chaining together RDD transformations, followed by **`collect()`**.
3. Print the results in the format shown below, replace the `xxxx` strings with the appropriate values.

```
Number of Partitions: xxxx
Size of Partitions:
[xxxx, xxxx, xxxx, xxxx]
```

Problem 3: Transformations

In the first code cell in Problem 3, we will scale the values in our randomly generated RDD to be in the range 0 to 10 rather than the range 0 to 1.

Complete the following steps in a single code cell:

1. Use the **`map()`** transformation to multiply each element of **`random_rdd`** by 10, storing the results in a variable named **`scaled_rdd`**.
2. Calculate the sum, mean, standard deviation, minimum, and maximum of the values stored in **`scaled_rdd`**. Display the results using the same format as in the first cell of Problem 2.

We will now transform the values in our scaled RDD by applying the natural logarithm to each value.

Complete the following steps in a single code cell:

1. Use the **`map()`** transformation along with the function **`math.log()`** to take the natural logarithm of each element in **`scaled_rdd`**. Store the new RDD in a variable named **`log_rdd`**.
2. Calculate the sum, mean, standard deviation, minimum, and maximum of the values stored in **`log_rdd`**. Display the results using the same format as in the previous cell.

Problem 4: Calculating SSE

In this problem, we will read in and process a data file containing two decimal values in each line. The first value is intended to represent an observed value of some random variable, while the second value is intended to represent a prediction generated by some model. We will use these values to calculate the sum of squared errors score for the predictions.

The path for the data file we will be using in this problem is **`/FileStore/tables/pairs_data.txt`**. We will start by reading the data file and counting the number of records.

Complete the following steps in a single code cell:

1. Read the contents of the data file into and RDD named **`pairs_raw`**.
2. Display the number of elements contained in the **`pairs_raw`** RDD

We will now display the first few elements of this RDD.

Use a **`for`** loop and the **`take()`** method to display the first 5 elements of **`pairs_raw`**. Note that these elements are stored as strings.

We will now process each of the elements of the RDD by tokenizing each string and coercing the individual values to floats.

Complete the following steps in a single code cell:

1. Write a function named **`process_line()`**. The function should accept a single parameter named **`row`**. This parameter is intended to take on string values of the type stored in **`pairs_raw`**. The function should split the string at the space character, coerce each of the two tokens into **`float`** values, and return a tuple containing these two **`float`** values**.**
2. Use the **`map()`** transformation to apply **`process_line()`** to **`pairs_raw`** storing the resulting RDD in **`pairs`**.
3. Use a **`for`** loop and the **`take()`** method to display the first 5 elements of **`pairs`**.

We will now calculate the sum of squared errors score for the values stored in the RDD we have created.

Complete the following steps in a single code cell:

1. Use **`map()`** with a **`lambda`** function to calculate the squared difference of each pair of values stored in **`pairs`**. Then call the **`sum()`** method of the resulting RDD, storing the result in a variable named **`SSE`**.
2. Print the value of **`SSE`**.

Problem 5: Calculating r-Squared

We will continue the analysis started in Problem 4 by calculating the r-squared score for our predictions. The first step in this process in to calculate the mean of the observed values.

Complete the following steps in a single code cell:

1. Use the **`map()`** transformation along with a **`lambda`** function to select the first element of each tuple in the **`pairs`** RDD. Call the **`mean()`** method of the resulting RDD, storing the result in a variable named **`mean`**.
2. Print **`mean`**.

Note that this calculation might take a couple of minutes to complete.

We will now calculate the sum of the squared deviations between each observed value and their mean. This quantity is sometimes referred to as SST, or "total sum of squared deviations".

Complete the following steps in a single code cell:

1. Use the **`map()`** transformation along with a **`lambda`** function to calculate the square of the difference between each observed value in **`pairs`** and **`mean`**. Call the **`sum()`** method of the resulting RDD, storing the result in a variable named **`SST`**.
2. Print **`SST`**.

We will now calculate the r-squared score for the predictions. The formula for this value is given as follow: $r^2 = 1 - \frac{SSE}{SST}$

Complete the following steps in a single code cell:

1. Use **`SSE`** and **`SST`** to calculate r-squared, storing the result in a variable named **`r2`**.
2. Print **`r2`**.

Problem 6: NASA Server Logs

In this problem, we will explore one month of server logs for the NASA.gov website. These logs are from August 1995 and contain information about every request received by the server during that month. The path for the file within which these logs are stored is **`/FileStore/tables/NASA_server_logs_Aug_1995.txt`**.

Complete the following steps in a single code cell:

1. Read the contents of the data file into an RDD named **`nasa`**.
2. Display the number of elements contained in the **`nasa`** RDD

We will now display the first few elements of this RDD.

Use a **`for`** loop and the **`take()`** method to display the first 5 elements of **`nasa`**.

Each line in the log file contains 6 pieces of information separated by spaces. These pieces of information are, in order:

- The IP or DNS hostname of the server making the request.
- The data and time of the request.
- The request type. This will be **GET, POST**, or **HEAD**.
- The location of the requested resource.
- The HTTP state code the server sent back to the client.
- The number of bytes transferred to the client.

We will now determine the number of requests of each type.

Complete the following steps in a single code cell:

1. Use the **`lambda`** function shown below to map each line of the server log to a Boolean value indicating whether or not the line contains the string **`'GET'`**. Call the **`sum()`** method of the resulting RDD to count the number of **GET** requests.

   `lambda x : 'GET' in x`

2. Modify the process used in step one to determine the number of **POST** requests and the number of **HEAD** requests.
3. Print the results in the format shown below.

   ```
   Number of GET requests:  xxxx
   Number of POST requests: xxxx
   Number of HEAD requests: xxxx
   ```

Submission Instructions

When you are done, click **Clear State and Run All**. If any cell produces an error that you are unable to correct, then manually run every cell after that one, in order. Export the notebook as an HTML file and then upload this file to Canvas. Do not alter your notebook on Databricks after submitting unless you intend to resubmit a new HTML file. The notebook on Databricks should match the HTML file on Canvas.