Assignment 3: Hadoop, Spark and Flink

By

Rachael Joan Dias

0651897

Addressed to,

Professor Brian Srivastava

Trent University

AMOD-5410H-A-2019GW-PTBO Big Data

Objective: Hadoop's Mapreduce programming model was implemented on a dataset to answer a specific set of questions. Using Spark's machine learning capabilities, linear regression was performed on a dataset to predict values of life expectancy. Flink's real-time streaming was used to process text data.

1. Data Gathering

Hadoop: For the Hadoop part of the assignment I used the UCI Heart Disease dataset from Kaggle. The data consisted of 14 attributes, there were 303 records totally. [1]

Spark: I used the (WHO) Life Expectancy dataset from Kaggle and applied linear regression to predict the average life expectancy based on a few factors. [2]

Flink: I created a text file implemented some of the DataStream and DataSet API functions on it.

2. Hadoop

Note: Since Hadoop was not working on my system, I used my friend's laptop to execute my code, Prof Sri has given us permission to do so.

Prior to executing MapReduce below commands were run to start Hadoop, create input and output directories

Command 1: Start-all.cmd

This will start Hadoop the namenode, datanode, resource manager and node manager

Command 2: hadoop fs -mkdir /input

This command will create an input directory by the name of input

Command 3: hadoop fs -put C:\Users\17059\Desktop\BigData\heart.txt /input

Once the input directory is created, we need to place the file in the input directory, the above command will put the file in the input directory

Command 4: hadoop fs -mkdir /output

Now, that we have the file in the input directory, the next step is to create an output directory where we can see the output of the MapReduce program

I used the "sex" and "age" columns from UCI Heart Disease dataset. Using, a mapper and reducer program I was able to answer 3 specific questions related to the gender and age.

a. Average age among males and females in the dataset

I used next(infile) to skip the first line of the file since this is the header.

Below is a screenshot of the mapper script. The mapper script reads from stdin and prints to stdout line by line, since we only need the "sex" and "age" attributes we can select them by indexing line[1] and line[2]. These are printed out as key-value pairs

```
#!/usr/bin/python

import sys
infile = sys.stdin
next(infile) # skip first line of input file
for line in infile:

    line = line.strip()
    line = line.split(",")

    if len(line) >=2:
        sex = line[1]
        age = line[2]

        print '%s\t%s' % (sex, age)
```

An empty dictionary is initialized by specifying sex_age{}, which will store sex and age as key-value pairs.

The reducer script will sum all the ages present in the dictionary and divide by the total number of elements present in the dictionary. This will return the average age

```
#!/usr/bin/python
#Reducer.py
import sys

sex_age = {}

#Partitoner
for line in sys.stdin:
    line = line.strip()
    sex, age = line.split('\t')

    if sex in sex_age:
        sex_age[sex].append(int(age))
    else:
        sex_age[sex] = []
        sex_age[sex].append(int(age))

#Reducer
for sex in sex_age.keys():
    ave_age = sum(sex_age[sex])*1.0 / len(sex_age[sex])
    print '%s\t%s'% (sex, ave_age)
```

Below Hadoop streaming command was executed to run the MapReduce job

```
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$  hadoop jar /usr/local/Cellar/hadoop/3.1.1/libexec/share/hadoop/tools/lib/
hadoop-*streaming*.jar -input /user/ShaimaShoukat/heart.txt -output /user/ShaimaShoukat/rach -mapper "python /Users
/ShaimaShoukat/Downloads/map1.py" -reducer "python /Users/ShaimaShoukat/Downloads/red1.py"
2019-04-09 10:30:47,542 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
2019-04-09 10:30:48,825 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-04-09 10:30:48,896 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-09 10:30:48,896 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-09 10:30:48,921 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-09 10:30:49,431 INFO mapred.FileInputFormat: Total input files to process : 1
2019-04-09 10:30:49,546 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-09 10:30:49,782 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1229253940_0001
2019-04-09 10:30:49,784 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-09 10:30:49,964 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2019-04-09 10:30:49,968 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2019-04-09 10:30:49,969 INFO mapreduce.Job: Running job: job_local1229253940_0001
```

The MapReduce job gives the following output, the average age among males is 53.75 years while the average age among females is 55.67 years.

```
3.1.1 — -bash — 102×23
~ — -bash    ...    /usr/local/Cellar/hadoop/3.1.1 — -bash
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2019-04-09 10:31:32,227 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
cat: `/output/*': No such file or directory
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$ hadoop dfs -cat /rach/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2019-04-09 10:31:41,354 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
cat: `/rach/*': No such file or directory
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$ hadoop dfs -cat /user/ShaimaShoukat/rach/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2019-04-09 10:32:20,800 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
M       53.7584541063
F       55.6770833333
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$
```

Scripts: map.py and red1.py

b. Maximum age among males and females in the dataset

In a similar way, the maximum age among males and females can be found using the same mapper script and changing the reducer by replacing it with the max() function. The max() function will return the largest age among males and females in the dictionary.

```python
#!/usr/bin/python
#Reducer.py
import sys

sex_age = {}

#Partitoner
for line in sys.stdin:
    line = line.strip()
    sex, age = line.split('\t')

    if sex in sex_age:
        sex_age[sex].append(int(age))
    else:
        sex_age[sex] = []
        sex_age[sex].append(int(age))

#Reducer
for sex in sex_age.keys():
    m = max(sex_age[sex])
    print '%s\t%s'% (sex, m)
```

Below Hadoop Streaming command was executed to run the MapReduce job

```
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$  hadoop jar /usr/local/Cellar/hadoop/3.1.1/libexec/share/hadoop/tools/lib/
hadoop-*streaming*.jar -input /user/ShaimaShoukat/heart.txt -output /user/ShaimaShoukat/rach -mapper "python /Users
/ShaimaShoukat/Downloads/map1.py" -reducer "python /Users/ShaimaShoukat/Downloads/red1.py"
2019-04-09 10:30:47,542 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
2019-04-09 10:30:48,825 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-04-09 10:30:48,896 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-09 10:30:48,896 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-09 10:30:48,921 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-09 10:30:49,431 INFO mapred.FileInputFormat: Total input files to process : 1
2019-04-09 10:30:49,546 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-09 10:30:49,782 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1229253940_0001
2019-04-09 10:30:49,784 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-09 10:30:49,964 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2019-04-09 10:30:49,968 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2019-04-09 10:30:49,969 INFO mapreduce.Job: Running job: job_local1229253940_0001
```

The maximum age among males and females is 77 and 76 years respectively

```
M       77
F       76
```

Scripts: map.py and red2.py

c. Count of males and females in the dataset

The reducer script is modified to count the total number of males and females in the dataset, by applying the len() function to the dictionary we can count the total number of males and females present

```python
#!/usr/bin/python
#Reducer.py
import sys

sex_age = {}

#Partitoner
for line in sys.stdin:
    line = line.strip()
    sex, age = line.split('\t')

    if sex in sex_age:
        sex_age[sex].append(int(age))
    else:
        sex_age[sex] = []
        sex_age[sex].append(int(age))

#Reducer
for sex in sex_age.keys():
    count = len(sex_age[sex])
    print '%s\t%s'% (sex, count)
```

Command for MapReduce job

```
3.1.1 — -bash — 115×23
~ — -bash        /usr/local/Cellar/hadoop/3.1.1 — -bash
2019-04-09 10:59:30,072 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-09 10:59:30,072 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-09 10:59:30,087 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-09 10:59:30,221 ERROR streaming.StreamJob: Error Launching job : Output directory hdfs://localhost:8020/use
r/ShaimaShoukat/rach3 already exists
Streaming Command Failed!
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$  hadoop jar /usr/local/Cellar/hadoop/3.1.1/libexec/share/hadoop/tools/lib/
hadoop-*streaming*.jar -input /user/ShaimaShoukat/heart.txt -output /user/ShaimaShoukat/rach4 -mapper "python /User
s/ShaimaShoukat/Downloads/map1.py" -reducer "python /Users/ShaimaShoukat/Downloads/red3.py"
2019-04-09 10:59:47,740 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
2019-04-09 10:59:48,751 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-04-09 10:59:48,817 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-09 10:59:48,817 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-09 10:59:48,835 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-09 10:59:49,304 INFO mapred.FileInputFormat: Total input files to process : 1
2019-04-09 10:59:49,391 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-09 10:59:49,537 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local160932310_0001
2019-04-09 10:59:49,539 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-09 10:59:49,712 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2019-04-09 10:59:49,714 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2019-04-09 10:59:49,715 INFO mapreduce.Job: Running job: job_local160932310_0001
2019-04-09 10:59:49,716 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
```

From the output below we can see that there are 207 and 96 females.

```
3.1.1 — -bash — 115×23
~ — -bash    /usr/local/Cellar/hadoop/3.1.1 — -bash
                GC time elapsed (ms)=13
                Total committed heap usage (bytes)=610271232
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=12435
        File Output Format Counters
                Bytes Written=11
2019-04-09 10:59:50,736 INFO streaming.StreamJob: Output directory: /user/ShaimaShoukat/rach4
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$ hadoop dfs -cat /user/ShaimaShoukat/rach4/*
WARNING: Use of this script to execute dfs is deprecated.
WARNING: Attempting to execute replacement "hdfs dfs" instead.

2019-04-09 11:00:01,224 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
 builtin-java classes where applicable
M       207
F       96
Shaimas-MacBook-Pro:3.1.1 ShaimaShoukat$
```

Scripts: map.py and red3.py

Hadoop is especially useful when performing computation on large datasets, because the MapReduce programming model stores and distributes the data across a cluster of computers.

3. Spark

Regression and Visualization:

Linear regression predicts the value of dependent variable 'Y' based on variable 'X' assuming there is a strong correlation between 'X' and 'Y'. The life expectancy dataset consists of the "LifeExpectancy" attribute which is the average number of years an individual is expected to live for different countries between 2000 and 2015. Using, linear regression variables such as adult mortality, BMI, income and schooling can be used to predict the life expectancy for different countries.

We start our analysis by importing the libraries below to visualize the relationships and then, build a linear model to predict the life expectancy of an individual. First, we read the .csv file using the pandas function read_csv(). Then, since we want to simplify our analysis we drop all rows containing Null values using the dropna() function.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt   #Data visualisation libraries
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

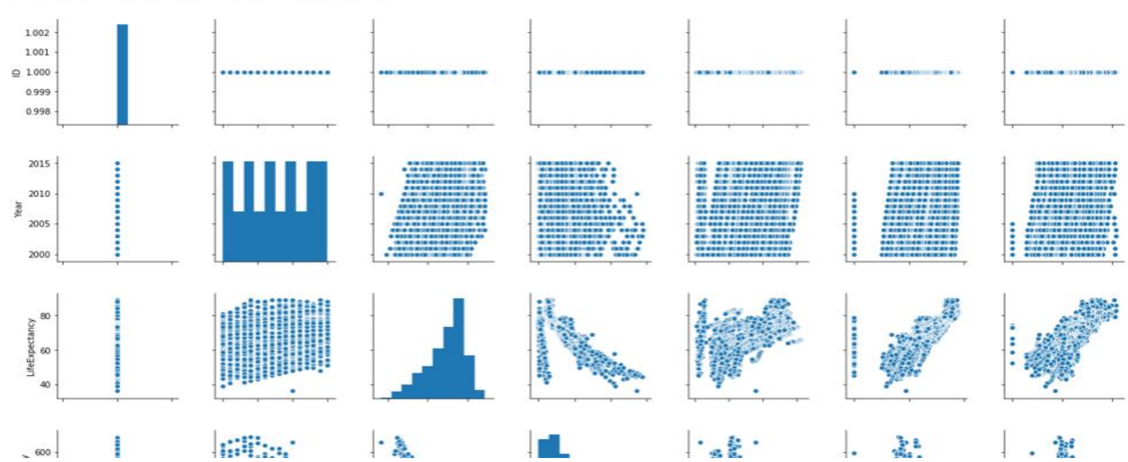
```python
df1 = pd.read_csv('LifeExpectancyData.csv')
df=df1.dropna()
df.head()
df.info()
df.describe()
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2736 entries, 0 to 2937
Data columns (total 9 columns):
ID                                  2736 non-null int64
Country                             2736 non-null object
Year                                2736 non-null int64
Status                              2736 non-null object
LifeExpectancy                      2736 non-null float64
Adult Mortality                     2736 non-null float64
BMI                                 2736 non-null float64
Income composition of resources     2736 non-null float64
Schooling                           2736 non-null float64
dtypes: float64(5), int64(2), object(2)
memory usage: 213.8+ KB

Index(['ID', 'Country', 'Year', 'Status', 'LifeExpectancy', 'Adult Mortality',
       'BMI', 'Income composition of resources', 'Schooling'],
      dtype='object')
```

From, the correlation plots we observe that there is a strong linear relationship between the “LifeExpectancy” variable and Adult mortality, BMI, income and schooling. Therefore, we can use these variables to build a linear regression model.

```python
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x29857035320>
```



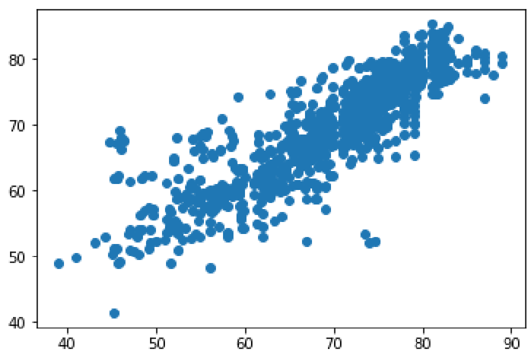
```
df.corr()
```

	ID	Year	LifeExpectancy	Adult Mortality	BMI	Income composition of resources	Schooling
ID	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Year	NaN	1.000000	0.171609	-0.077084	0.101722	0.244376	0.218844
LifeExpectancy	NaN	0.171609	1.000000	-0.681017	0.563736	0.719335	0.749688
Adult Mortality	NaN	-0.077084	-0.681017	1.000000	-0.375359	-0.447458	-0.442386
BMI	NaN	0.101722	0.563736	-0.375359	1.000000	0.509299	0.558363
Income composition of resources	NaN	0.244376	0.719335	-0.447458	0.509299	1.000000	0.791651
Schooling	NaN	0.218844	0.749688	-0.442386	0.558363	0.791651	1.000000

First, we define the outcome variable "y" and the independent variables "X", then we split the dataset into test and train. Using the LinearRegression() function a linear model is fit with the data which can be used to make predictions.

```
X = df[['Adult Mortality','BMI','Income composition of resources','Schooling']]
y= df['LifeExpectancy']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
lm = LinearRegression()
lm.fit(X_train,y_train)

predictions = lm.predict(X_test)
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x2986731df98>
```



A scatter of the actual value and predictions based on the data.

4. Machine Learning using Spark

```
#################################### REGRESSION WITH PYSPARK #############################################
```

```
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

```
spark=SparkSession.builder.appName('LifeExpectancy').getOrCreate()
data1=spark.read.csv('LifeExpectancyData.csv',inferSchema=True,header=True)
data=data1.dropna(thresh=5,subset=('Adult Mortality','BMI','Income composition of resources','Schooling','LifeExpectancy'))
data.show()
data.printSchema()
```

```
+-----------+----+----------+--------------+---------------+-------------+-------+----------------------+-----------+--------+-
---+-----------------+-----+-----------------+----------+--------+-----------+-----------+--------------------+------------
-------+------------------------------+---------+
|    Country|Year|    Status|LifeExpectancy|Adult Mortality|infant deaths|Alcohol|percentage expenditure|Hepatitis B|Measles |
BMI|under-five deaths |Polio|Total expenditure|Diphtheria | HIV/AIDS|       GDP| Population| thinness  1-19 years| thinness 5
-9 years|Income composition of resources|Schooling|
+-----------+----+----------+--------------+---------------+-------------+-------+----------------------+-----------+--------+-
---+-----------------+-----+-----------------+----------+--------+-----------+-----------+--------------------+------------
-------+------------------------------+---------+
|Afghanistan|2015|Developing|          65.0|            263|           62|   0.01|           71.27962362|         65|    1154|1
9.1|               83|    6|             8.16|         65|     0.1|  584.25921|3.3736494E7|                17.2
|               17.3|                          0.479|     10.1|
|Afghanistan|2014|Developing|          59.9|            271|           64|   0.01|           73.52358168|         62|     492|1
8.6|               86|   58|             8.18|         62|     0.1| 612.696514|   327582.0|                17.5
|               17.5|                          0.476|     10.0|
|Afghanistan|2013|Developing|          59.9|            268|           66|   0.01|           73.21924272|         64|     430|1
```

We import SparkSession then intialize the session by using the command “spark=SparkSession.builder.appName('LifeExpectancy').getOrCreate()”

Then, we import the LinearRegression model from the machine learning library. We read the dataset using the read.csv() command inferSchema=True insures that the datatype is maintained while reading it from the csv. In order to make the analysis easier, we drop all rows having Null values.

data.show() will print the data frame and data.printSchema() will print the datatypes of the columns in the dataframe.

```
################################### REGRESSION WITH PYSPARK ############################################
```

```python
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

```python
spark=SparkSession.builder.appName('LifeExpectancy').getOrCreate()
data1=spark.read.csv('LifeExpectancyData.csv',inferSchema=True,header=True)
data=data1.dropna(thresh=5,subset=('Adult Mortality','BMI','Income composition of resources','Schooling','LifeExpectancy'))
data.show()
data.printSchema()
```

```
+---+-----------+----+----------+--------------+---------------+----+-------------------------------+---------+
| ID|    Country|Year|    Status|LifeExpectancy|Adult Mortality| BMI|Income composition of resources|Schooling|
+---+-----------+----+----------+--------------+---------------+----+-------------------------------+---------+
|  1|Afghanistan|2015|Developing|          65.0|            263|19.1|                          0.479|     10.1|
|  1|Afghanistan|2014|Developing|          59.9|            271|18.6|                          0.476|     10.0|
|  1|Afghanistan|2013|Developing|          59.9|            268|18.1|                           0.47|      9.9|
|  1|Afghanistan|2012|Developing|          59.5|            272|17.6|                          0.463|      9.8|
|  1|Afghanistan|2011|Developing|          59.2|            275|17.2|                          0.454|      9.5|
|  1|Afghanistan|2010|Developing|          58.8|            279|16.7|                          0.448|      9.2|
|  1|Afghanistan|2009|Developing|          58.6|            281|16.2|                          0.434|      8.9|
|  1|Afghanistan|2008|Developing|          58.1|            287|15.7|                          0.433|      8.7|
|  1|Afghanistan|2007|Developing|          57.5|            295|15.2|                          0.415|      8.4|
|  1|Afghanistan|2006|Developing|          57.3|            295|14.7|                          0.405|      8.1|
|  1|Afghanistan|2005|Developing|          57.3|            291|14.2|                          0.396|      7.9|
|  1|Afghanistan|2004|Developing|          57.0|            293|13.8|                          0.381|      6.8|
|  1|Afghanistan|2003|Developing|          56.7|            295|13.4|                          0.373|      6.5|
|  1|Afghanistan|2002|Developing|          56.2|              3|13.0|                          0.341|      6.2|
|  1|Afghanistan|2001|Developing|          55.3|            316|12.6|                           0.34|      5.9|
|  1|Afghanistan|2000|Developing|          54.8|            321|12.2|                          0.338|      5.5|
|  1|    Albania|2015|Developing|          77.8|             74|58.0|                          0.762|     14.2|
|  1|    Albania|2014|Developing|          77.5|              8|57.2|                          0.761|     14.2|
|  1|    Albania|2013|Developing|          77.2|             84|56.5|                          0.759|     14.2|
|  1|    Albania|2012|Developing|          76.9|             86|55.8|                          0.752|     14.2|
+---+-----------+----+----------+--------------+---------------+----+-------------------------------+---------+
only showing top 20 rows

root
 |-- ID: integer (nullable = true)
 |-- Country: string (nullable = true)
 |-- Year: integer (nullable = true)
 |-- Status: string (nullable = true)
 |-- LifeExpectancy: double (nullable = true)
 |-- Adult Mortality: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- Income composition of resources: double (nullable = true)
 |-- Schooling: double (nullable = true)
```

Pyspark requires the input variables to be represented as a vector, the VectorAssembler() groups all the independent variables as a vector, all the inputcols are and outputcol have to be provided to the VectorAssembeler()

From the screenshot we can see that the independent features have been transformed into a vector for corresponding values of LifeExpectancy.

```
featureassembler=VectorAssembler(inputCols=['Adult Mortality','BMI','Income composition of resources','Schooling'],outputCol="In
```

```
output=featureassembler.transform(data)
output.show()
```

```
+---+-----------+----+----------+-------------+---------------+----+-------------------------------+---------+----------------
----+
| ID|    Country|Year|    Status|LifeExpectancy|Adult Mortality| BMI|Income composition of resources|Schooling|Independent Feat
ures|
+---+-----------+----+----------+-------------+---------------+----+-------------------------------+---------+----------------
----+
|  1|Afghanistan|2015|Developing|         65.0|            263|19.1|                          0.479|     10.1|[263.0,19.1,0.47
9...|
|  1|Afghanistan|2014|Developing|         59.9|            271|18.6|                          0.476|     10.0|[271.0,18.6,0.47
6...|
|  1|Afghanistan|2013|Developing|         59.9|            268|18.1|                           0.47|      9.9|[268.0,18.1,0.4
7,...|
|  1|Afghanistan|2012|Developing|         59.5|            272|17.6|                          0.463|      9.8|[272.0,17.6,0.46
3...|
|  1|Afghanistan|2011|Developing|         59.2|            275|17.2|                          0.454|      9.5|[275.0,17.2,0.45
4...|
```

```
finalized_data=output.select("Independent Features","LifeExpectancy")
finalized_data.show()
```

```
+--------------------+--------------+
|Independent Features|LifeExpectancy|
+--------------------+--------------+
|[263.0,19.1,0.479...|          65.0|
|[271.0,18.6,0.476...|          59.9|
|[268.0,18.1,0.47,...|          59.9|
|[272.0,17.6,0.463...|          59.5|
|[275.0,17.2,0.454...|          59.2|
|[279.0,16.7,0.448...|          58.8|
|[281.0,16.2,0.434...|          58.6|
|[287.0,15.7,0.433...|          58.1|
|[295.0,15.2,0.415...|          57.5|
|[295.0,14.7,0.405...|          57.3|
|[291.0,14.2,0.396...|          57.3|
|[293.0,13.8,0.381...|          57.0|
|[295.0,13.4,0.373...|          56.7|
|[3.0,13.0,0.341,6.2]|          56.2|
|[316.0,12.6,0.34,...|          55.3|
|[321.0,12.2,0.338...|          54.8|
|[74.0,58.0,0.762,...|          77.8|
|[8.0,57.2,0.761,1...|          77.5|
|[84.0,56.5,0.759,...|          77.2|
|[86.0,55.8,0.752,...|          76.9|
+--------------------+--------------+
only showing top 20 rows
```

Next, we divide the data into test and training instances. We select the LinearRegression() model and specify the independent and dependent features. Then, we fit the data to the model.

```
train_data,test_data=finalized_data.randomSplit([0.75,0.25])
regressor=LinearRegression(featuresCol="Independent Features",labelCol="LifeExpectancy")
regressor=regressor.fit(train_data)
regressor.coefficients
```

```
DenseVector([-0.0303, 0.0504, 9.713, 0.9877])
```

```
regressor.intercept
```

```
54.30813568940057
```

We can check the evaluate the prediction based on the test_data by using the evaluate function. The table below shows the Independent Feature, the actual value of life expectancy and the prediction based on the model

```
pred_results=regressor.evaluate(test_data)
pred_results.predictions.show()
```

```
+--------------------+--------------+-----------------+
|Independent Features|LifeExpectancy|       prediction|
+--------------------+--------------+-----------------+
|[1.0,52.6,0.713,1...|          75.3|75.70814476895163|
|[1.0,56.1,0.856,1...|          78.0|82.90354843037976|
|[2.0,15.4,0.563,1...|          66.0|70.86343149682182|
|[2.0,43.8,0.482,9.0]|          67.6|70.02723533667036|
|[3.0,13.0,0.341,6.2]|          56.2|64.30864532611494|
|[3.0,36.2,0.687,1...|          63.4|75.06181602405105|
|[4.0,23.3,0.456,8.3]|          52.8|67.98890283225019|
|[6.0,27.1,0.493,9.1]|          45.9| 69.2694258989168|
|[6.0,61.0,0.869,1...|          81.8|81.84126394695605|
|[6.0,64.6,0.895,1...|          82.2| 81.8802529434724|
|[6.0,65.4,0.874,1...|          82.4|83.19818942249036|
|[7.0,22.9,0.86,14.7]|          81.8|78.12315237278158|
|[7.0,57.0,0.931,1...|          84.0|83.29791938755267|
|[7.0,62.3,0.854,1...|          81.3|81.43448697485861|
|[8.0,58.5,0.867,1...|          79.4|81.04249631468429|
|[8.0,58.6,0.826,1...|          76.5|78.18002856470002|
|[9.0,56.4,0.873,1...|          79.1| 82.0510366237727|
|[9.0,57.3,0.809,1...|          79.3|80.58585057568357|
| [11.0,6.8,0.0,13.7]|          73.9|67.84912339276286|
|[11.0,45.0,0.656,...|          72.6|73.18399820518252|
+--------------------+--------------+-----------------+
only showing top 20 rows
```

Scripts name: Assg#3_Spark.ipynb

5. Flink

Flink is capable of processing real-time streaming data.

To start Flink, we navigate to the Flink bin directory and initiate the cluster by running the ./start-cluster.sh command

MINGW64:/c/BigData/flink-1.7.2/bin

```
17059@DESKTOP-RA3RPFF MINGW64 /
$ cd C:\BigData\flink-1.7.2\bin
bash: cd: C:BigDataflink-1.7.2bin: No such file or directory

17059@DESKTOP-RA3RPFF MINGW64 /
$ ./start-cluster.sh
bash: ./start-cluster.sh: No such file or directory

17059@DESKTOP-RA3RPFF MINGW64 /
$ cd C:/Users/17059/Desktop/BigData

17059@DESKTOP-RA3RPFF MINGW64 ~/Desktop/BigData
$ cd C:/BigData/flink-1.7.2/bin

17059@DESKTOP-RA3RPFF MINGW64 /c/BigData/flink-1.7.2/bin
$ ./start-cluster.sh local
Starting cluster.
Starting standalonesession daemon on host DESKTOP-RA3RPFF.
Starting taskexecutor daemon on host DESKTOP-RA3RPFF.

17059@DESKTOP-RA3RPFF MINGW64 /c/BigData/flink-1.7.2/bin
$ |
```

Then, we start Scala-Shell by running the command ./start-scala-shell.sh local

Below programs can be run in the shell line by line

In the first command read and save the sample.txt file to a variable text, benv means that we have set the execution environment to batch, similarly senv means that it is a streaming environment. In the next step, we map each word, group them and then sum them this will give us the number of occurrences of each word in the text file

```
Flink 5

################ PROGRAM 1 Aggregation DataSet Api##########################################
val text = benv.readTextFile("C:/Users/17059/Desktop/BigData/sample.txt")
val count = text.flatMap { _.toLowerCase.split("\\W+") } .map { (_, 1) }.groupBy(0).sum(1)
count.print()
senv.execute("count")


################ PROGRAM 2 Aggregation DataStream Api#######################################
val text = senv.readTextFile("C:/Users/17059/Desktop/BigData/sample.txt")
val count = text.flatMap { _.toLowerCase.split("\\W+") } .map { (_, 1) }.keyBy(0).sum(1)
count.print()
senv.execute("count")
```

Below are the screenshots for the output

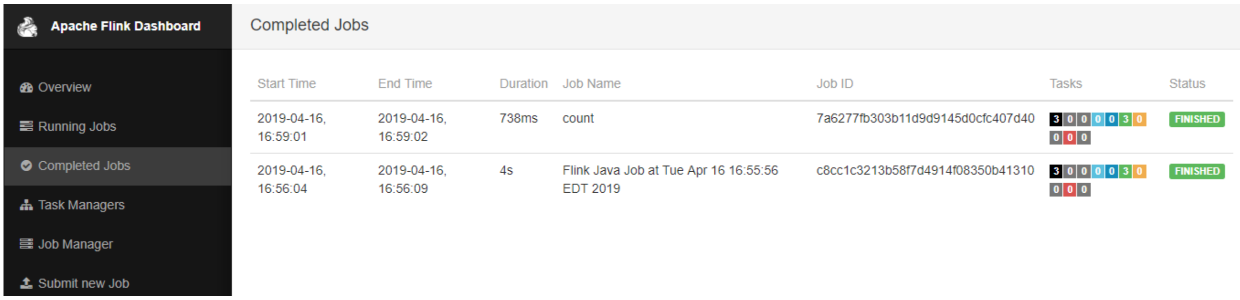
MINGW64:/c/BigData/flink-1.7.2/bin

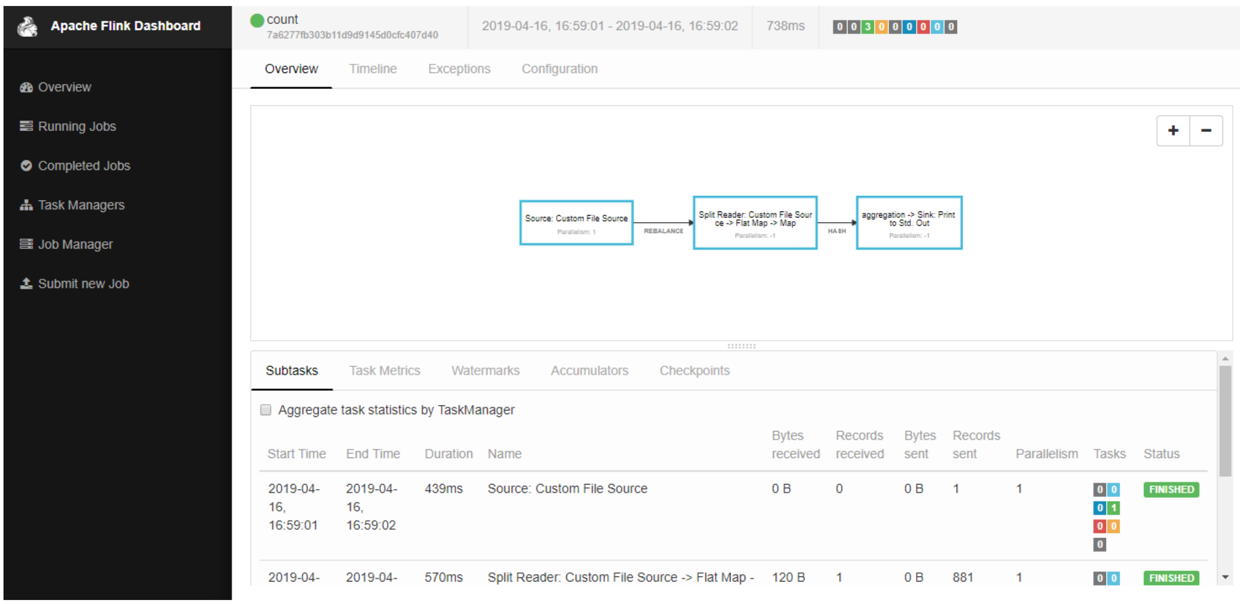
```
scala> val text = benv.readTextFile("C:/Users/17059/Desktop/BigData/sample.txt")
text: org.apache.flink.api.scala.DataSet[String] = org.apache.flink.api.scala.DataSet@39685204

scala> val count = text.flatMap { _.toLowerCase.split("\\W+") } .map { (_, 1) }.groupBy(0).sum(1)
count: org.apache.flink.api.scala.AggregateDataSet[(String, Int)] = org.apache.flink.api.scala.AggregateDataSet@5a02fca5

scala> count.print()
(,27)
(161,1)
(2001,1)
(24,1)
(50,1)
(78,1)
(a,30)
(about,2)
(above,1)
(act,1)
(actually,1)
(ade,1)
(air,1)
(airplane,1)
(airport,2)
(all,1)
(already,1)
(an,2)
(ancient,1)
```

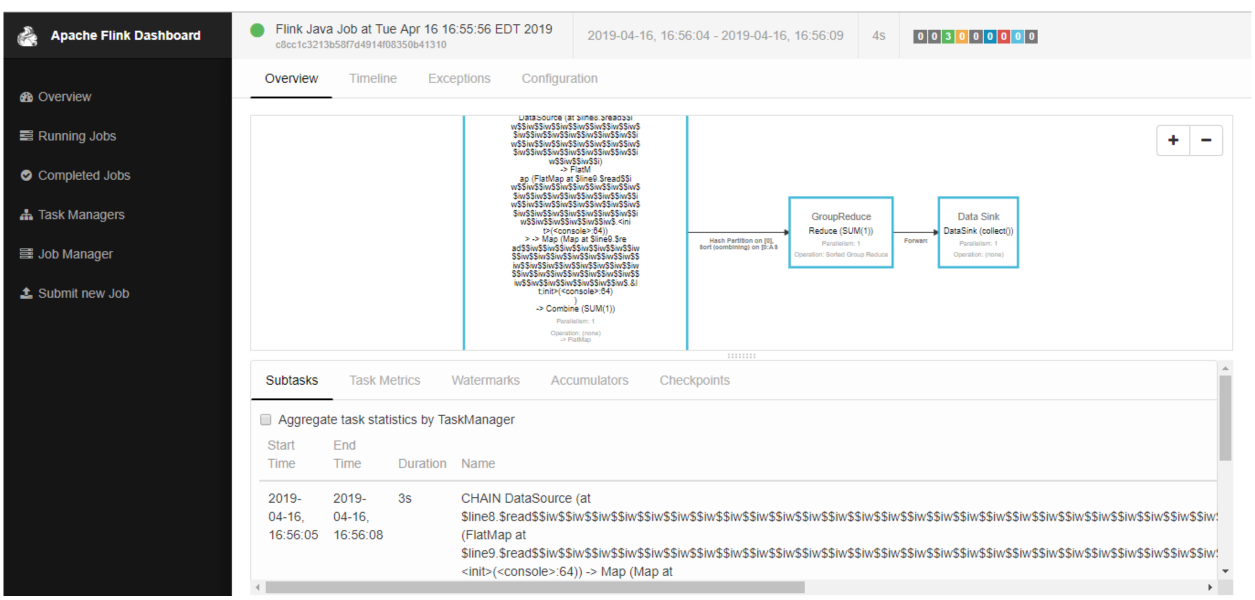
MINGW64:/c/BigData/flink-1.7.2/bin

```
scala> senv.execute("count")
java.lang.IllegalStateException: No operators defined in streaming topology. Cannot execute.
  at org.apache.flink.streaming.api.environment.StreamExecutionEnvironment.getStreamGraph(StreamExecutionEnvironment.java:1535)
  at org.apache.flink.streaming.api.environment.RemoteStreamEnvironment.execute(RemoteStreamEnvironment.java:174)
  at org.apache.flink.streaming.api.scala.StreamExecutionEnvironment.execute(StreamExecutionEnvironment.scala:654)
  ... 30 elided

scala> val text = senv.readTextFile("C:/Users/17059/Desktop/BigData/sample.txt")
text: org.apache.flink.streaming.api.scala.DataStream[String] = org.apache.flink.streaming.api.scala.DataStream@24456c9e

scala> val count = text.flatMap { _.toLowerCase.split("\\W+") } .map { (_, 1) }.groupBy(0).sum(1)
<console>:64: error: value groupBy is not a member of org.apache.flink.streaming.api.scala.DataStream[(String, Int)]
       val count = text.flatMap { _.toLowerCase.split("\\W+") } .map { (_, 1) }.groupBy(0).sum(1)
                                                                                 ^

scala> val count = text.flatMap { _.toLowerCase.split("\\W+") } .map { (_, 1) }.keyBy(0).sum(1)
count: org.apache.flink.streaming.api.scala.DataStream[(String, Int)] = org.apache.flink.streaming.api.scala.DataStream@6d19e557

scala> count.print()
res2: org.apache.flink.streaming.api.datastream.DataStreamSink[(String, Int)] = org.apache.flink.streaming.api.datastream.DataStreamSink@1963b057

scala> senv.execute("count")
(it,1)
(was,1)
(the,1)
(hunter,1)
(s,1)
(first,1)
(time,1)
(outside,1)
(montana,1)
(he,1)
(woke,1)
(stricken,1)
(still,1)
(with,1)
(the,2)
(hours,1)
(old,1)
(vision,1)
```

We can check the output of the jobs on the Flink dashboard to see if they have completed successfully, from the screenshots below we can see that both the jobs completed successfully.







References:

[1] https://www.kaggle.com/ronitf/heart-disease-uci

[2] https://www.kaggle.com/kumarajarshi/life-expectancy-who