

BIG DATA HADOOP & SPARK TRAINING

Case Study III from
session 19 on
Sensor data

CASE STUDY III

Case study on Sensor data

For this data analysis, you can download the necessary dataset from this link.

In the above link there are two datasets; building.csv contains the details of the top 20 buildings all over the world and HVAC.csv contains the target temperature and the actual temperature along with the building Id.

HVAC (heating, ventilating/ventilation, and air conditioning) is the technology of indoor and vehicular environmental comfort. Its goal is to provide thermal comfort and acceptable indoor air quality. Through the HVAC sensors, we will get the temperature of the buildings.

Here are the columns that are present in the datasets:

Building.csv – BuildingID, BuildingMgr, BuildingAge, HVACproduct, Country

HVAC.csv – Date, Time, TargetTemp, ActualTemp, System, SystemAge, BuildingID

Tasks:

- Load HVAC.csv file into temporary table
- Add a new column, tempchange - set to 1, if there is a change of greater than +/-5 between actual and target temperature
- Load building.csv file into temporary table
- Figure out the number of times, temperature has changed by 5 degrees or more for each country:
 - Join both the tables.
 - Select tempchange and country column
 - Filter the rows where tempchange is 1 and count the number of occurrence for each country

Load HVAC.csv, building.csv file to spark:

To load csv file to spark we have to perform the following steps:

- Create a manual schema for both csv files which would provide the schema while loading both CSV's as shown below

```
val Manual_schema_HVAC = new StructType(Array(new StructField("Date", StringType, true),
  new StructField("Time", StringType, false),
  new StructField("TargetTemp", LongType, true),
  new StructField("ActualTemp", LongType, false),
  new StructField("System", LongType, false),
  new StructField("SystemAge", LongType, false),
  new StructField("BuildingID", LongType, false)))

val Manual_schema_Building = new StructType(Array(new StructField("BuildingID", LongType, true),
  new StructField("BuildingMgr", StringType, false),
  new StructField("BuildingAge", LongType, true),
  new StructField("HVACproduct", StringType, false),
  new StructField("Country", StringType, false)))
```

Note:

StructType is a built-in data type used for Schema definition in Spark SQL, to represent a collection of StructFields that together define a schema or its part.

<schema-name> = new

StructType<array_of_columns><Struct_field>(<column_name>,<data_type_of_column>,<nullable_or_not_nullable(true/false)>)

- Now we load the CSV files from local file system to spark as shown below:

```
val HVAC = spark.read.format("CSV")
  .option("header", true)
  .schema(Manual_schema_HVAC)
  .load("E:\\casestudies\\sensorcasestudy\\HVAC.csv")

HVAC.show()
HVAC.registerTempTable("HVAC_table")
println("HVAC table registered!")

val buildings = spark.read.format("CSV")
  .option("header", true)
  .schema(Manual_schema_Building)
  .load("E:\\casestudies\\sensorcasestudy\\buliding.csv")
buildings.show()

buildings.registerTempTable("building_table")
println("buildings table registered!")
```

- We are using the CSV file read format, this provides various options of which we have used a few of them, which are as follows:
 - We have used the option to remove the header from the input file.
 - We have given the manual schema that we have created in the previous step
 - We are giving the path where the CSV file is saved in the local file system.
- Once loading is done we can register the respective tables to TempTable as shown above i.e. `HVAC.registerTempTable("HVAC_table")`.

```

18/05/13 16:01:11 INFO CodeGenerator: Code generated in 36.124469 ms
+-----+-----+-----+-----+-----+-----+
| Date| Time|TargetTemp|ActualTemp|System|SystemAge|BuildingID|
+-----+-----+-----+-----+-----+-----+
| 6/1/2013| 0:00:01| 66| 58| 13| 20| 4|
| 6/2/2013| 1:00:01| 69| 68| 3| 20| 17|
| 6/3/2013| 2:00:01| 70| 73| 17| 20| 18|
| 6/4/2013| 3:00:01| 67| 63| 2| 23| 15|
| 6/5/2013| 4:00:01| 68| 74| 16| 9| 3|
| 6/6/2013| 5:00:01| 67| 56| 13| 28| 4|
| 6/7/2013| 6:00:01| 70| 58| 12| 24| 2|
| 6/8/2013| 7:00:01| 70| 73| 20| 26| 16|
| 6/9/2013| 8:00:01| 66| 69| 16| 9| 9|
| 6/10/2013| 9:00:01| 65| 57| 6| 5| 12|
| 6/11/2013| 10:00:01| 67| 70| 10| 17| 15|
| 6/12/2013| 11:00:01| 69| 62| 2| 11| 7|
| 6/13/2013| 12:00:01| 69| 73| 14| 2| 15|
| 6/14/2013| 13:00:01| 65| 61| 3| 2| 6|
| 6/15/2013| 14:00:01| 67| 59| 19| 22| 20|
| 6/16/2013| 15:00:01| 65| 56| 19| 11| 8|
| 6/17/2013| 16:00:01| 67| 57| 15| 7| 6|
| 6/18/2013| 17:00:01| 66| 57| 12| 5| 13|
| 6/19/2013| 18:00:01| 69| 58| 8| 22| 4|
| 6/20/2013| 19:00:01| 67| 55| 17| 5| 7|
+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

18/05/13 16:01:12 INFO ContextCleaner: Cleaned accumulator 2
18/05/13 16:01:12 INFO BlockManagerInfo: Removed broadcast_1_piece0 on 192.168.56.1:62246 in memory (size: 5.2 KB, free: 902.7 MB)
18/05/13 16:01:12 INFO BlockManagerInfo: Removed broadcast_0_piece0 on 192.168.56.1:62246 in memory (size: 14.7 KB, free: 902.7 MB)
18/05/13 16:01:12 INFO ContextCleaner: Cleaned accumulator 1
18/05/13 16:01:12 INFO ContextCleaner: Cleaned accumulator 0
18/05/13 16:01:12 INFO SparkSqlParser: Parsing command: HVAC_table
HVAC table registered!
18/05/13 16:01:12 INFO FileSourceStrategy: Pruning directories with:

```

```
NewProj1 [C:\Users\Laptop\IdeaProjects\NewProj1 - ...src\main\scala\casestuyll.scala [newproj1] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

NewProj1 src main scala casestuyll.scala
Run: casestuyll x
18/05/13 16:01:12 INFO DAGScheduler: Job 1 finished: show at casestuyll.scala:57, took 0.068446 s
18/05/13 16:01:12 INFO CodeGenerator: Code generated in 26.117151 ms
+-----+
|BuildingID|BuildingMgr|BuildingAge|HVACproduct|Country|
+-----+
| 1| M1| 25| AC1000| USA|
| 2| M2| 27| FN39TG| France|
| 3| M3| 28| JDNS77| Brazil|
| 4| M4| 17| GG1919| Finland|
| 5| M5| 3| ACMAX22| Hong Kong|
| 6| M6| 9| AC1000| Singapore|
| 7| M7| 13| FN39TG|South Africa|
| 8| M8| 25| JDNS77| Australia|
| 9| M9| 11| GG1919| Mexico|
| 10| M10| 23| ACMAX22| China|
| 11| M11| 14| AC1000| Belgium|
| 12| M12| 26| FN39TG| Finland|
| 13| M13| 25| JDNS77|Saudi Arabia|
| 14| M14| 17| GG1919| Germany|
| 15| M15| 19| ACMAX22| Israel|
| 16| M16| 23| AC1000| Turkey|
| 17| M17| 11| FN39TG| Egypt|
| 18| M18| 25| JDNS77| Indonesia|
| 19| M19| 14| GG1919| Canada|
| 20| M20| 19| ACMAX22| Argentina|
+-----+

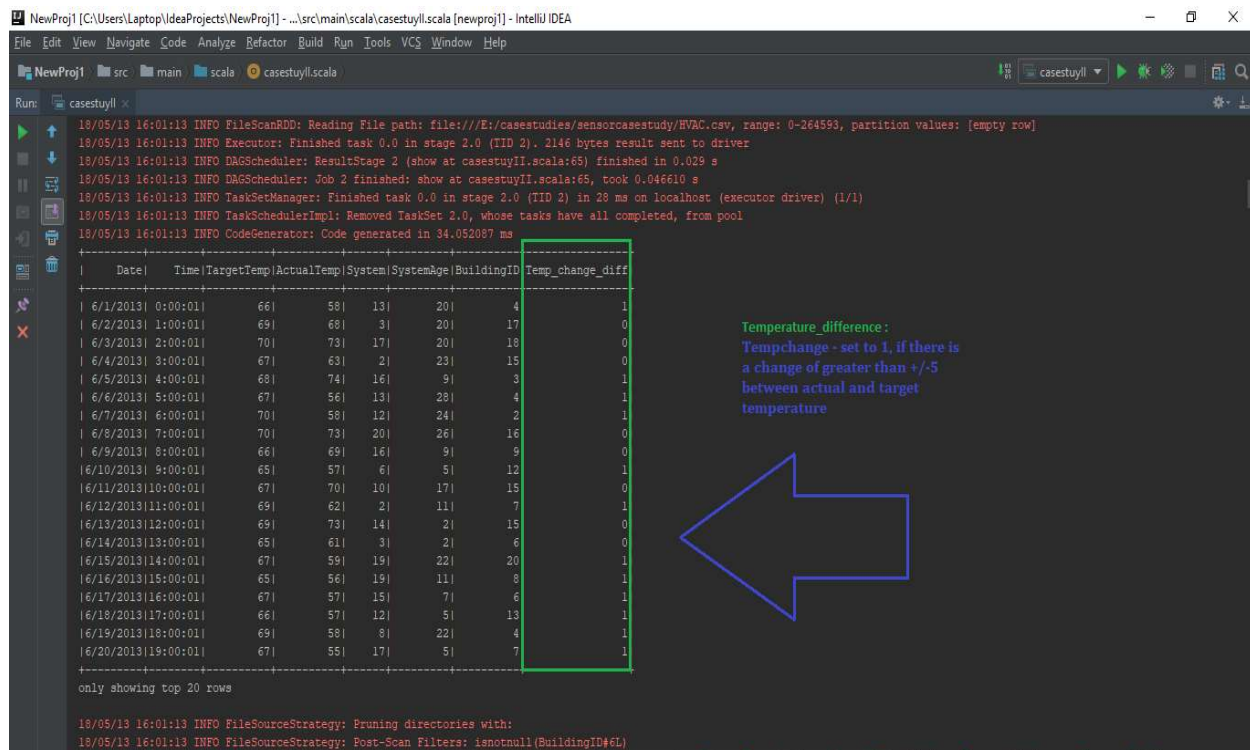
Building.csv file
contents
←

18/05/13 16:01:12 INFO SparkSqlParser: Parsing command: building_table
buildings table registered!
18/05/13 16:01:12 INFO SparkSqlParser: Parsing command: select *, IF((TargetTemp-ActualTemp)> 5 , '1', IF((TargetTemp-ActualTemp)< -5 , '1',0)) as Temp_change_diff from HVAC_table
18/05/13 16:01:13 INFO FileSourceStrategy: Pruning directories with:
18/05/13 16:01:13 INFO FileSourceStrategy: Post-Scan Filters:
18/05/13 16:01:13 INFO FileSourceStrategy: Output Data Schema: struct<Date: string, Time: string, TargetTemp: bigint, ActualTemp: bigint, System: bigint ... 5 more fields>
18/05/13 16:01:13 INFO FileSourceStrategy: Pushed Filters:
18/05/13 16:01:13 INFO CodeGenerator: Code generated in 31.742739 ms
```

- Now we will add a new column, tempchange - set to 1, if there is a change of greater than +/-5 between actual and target temperature

```
val filterHVAC = spark.sql("""select *, IF((TargetTemp-ActualTemp)> 5 , '1',
IF((TargetTemp-ActualTemp)< -5 , '1', 0)) as Temp_change_diff from HVAC_table""")
filterHVAC.show()
```

- Here we are filtering based on the difference between the Target temperature and Actual temperature, i.e. if the difference is between +/- 5 then give '1' as output if not give output as '0'.



Run: casestudyll

```
18/05/13 16:01:13 INFO FileScanRDD: Reading File path: file:///E:/casestudies/sensorcasestudy/HVAC.csv, range: 0-264593, partition values: [empty row]
18/05/13 16:01:13 INFO Executor: Finished task 0.0 in stage 2.0 (TID 2). 2146 bytes result sent to driver
18/05/13 16:01:13 INFO DAGScheduler: ResultStage 2 (show at casestudyll.scala:65) finished in 0.029 s
18/05/13 16:01:13 INFO DAGScheduler: Job 2 finished: show at casestudyll.scala:65, took 0.046610 s
18/05/13 16:01:13 INFO TaskSetManager: Finished task 0.0 in stage 2.0 (TID 2) in 28 ms on localhost (executor driver) (1/1)
18/05/13 16:01:13 INFO TaskSchedulerImpl: Removed TaskSet 2.0, whose tasks have all completed, from pool
18/05/13 16:01:13 INFO CodeGenerator: Code generated in 34.052087 ms
```

Date	Time	TargetTemp	ActualTemp	System	SystemAge	BuildingID	Temp_change_diff
6/1/2013	0:00:01	66	58	13	20	4	1
6/2/2013	1:00:01	69	68	3	20	17	0
6/3/2013	2:00:01	70	73	17	20	18	0
6/4/2013	3:00:01	67	63	2	23	15	0
6/5/2013	4:00:01	68	74	16	9	3	1
6/6/2013	5:00:01	67	56	13	28	4	1
6/7/2013	6:00:01	70	58	12	24	2	1
6/8/2013	7:00:01	70	73	20	26	16	0
6/9/2013	8:00:01	66	69	16	9	9	0
6/10/2013	9:00:01	65	57	6	5	12	1
6/11/2013	10:00:01	67	70	10	17	15	0
6/12/2013	11:00:01	69	62	2	11	7	1
6/13/2013	12:00:01	69	73	14	2	15	0
6/14/2013	13:00:01	65	61	3	2	6	0
6/15/2013	14:00:01	67	59	19	22	20	1
6/16/2013	15:00:01	65	56	19	11	8	1
6/17/2013	16:00:01	67	57	15	7	6	1
6/18/2013	17:00:01	66	57	12	5	13	1
6/19/2013	18:00:01	69	58	8	22	4	1
6/20/2013	19:00:01	67	55	17	5	7	1

only showing top 20 rows

18/05/13 16:01:13 INFO FileSourceStrategy: Pruning directories with:
18/05/13 16:01:13 INFO FileSourceStrategy: Post-Scan Filters: isNotNull(BuildingID#6L)

Temperature difference:
Tempchange - set to 1, if there is
a change of greater than +/-5
between actual and target
temperature

- Now we will Join both the tables based on BuildingID, i.e. we will join HVAC and Buildings tables and register the output of join to a temptable called "HVACJBUILD"

```
val joinExpression = filterHVAC.col("BuildingID") ===
buildings.toDF().col("BuildingID")
val HVACJOBUILD = filterHVAC.join(buildings, joinExpression)
HVACJOBUILD.show()
HVACJOBUILD.registerTempTable("HVACJBUILD")
```

```

18/05/13 16:01:14 INFO ContextCleaner: Cleaned accumulator 103
18/05/13 16:01:14 INFO ContextCleaner: Cleaned accumulator 104
18/05/13 16:01:14 INFO BlockManagerInfo: Removed broadcast_4_piece0 on 192.168.56.1:62246 in memory (size: 14.7 KB, free: 902.7 MB)
18/05/13 16:01:14 INFO BlockManagerInfo: Removed broadcast_5_piece0 on 192.168.56.1:62246 in memory (size: 6.5 KB, free: 902.7 MB)
18/05/13 16:01:14 INFO CodeGenerator: Code generated in 80.956654 ms

```

Date	Time	TargetTemp	ActualTemp	System	SystemAge	BuildingID	Temp_change_diff	BuildingID	BuildingMtr	BuildingAge	HVACProduct	Country
6/1/2013	0:00:01	66	58	13	20	4	1	4	M4	17	GG1919	Finland
6/2/2013	1:00:01	69	68	3	20	17	0	17	M17	11	FN39TG	Egypt
6/3/2013	2:00:01	70	73	17	20	18	0	18	M18	25	JDNST77	Indonesia
6/4/2013	3:00:01	67	63	2	23	15	0	15	M15	19	ACMAX22	Israel
6/5/2013	4:00:01	68	74	16	9	3	1	3	M3	28	JDNST77	Brazil
6/6/2013	5:00:01	67	56	13	28	4	1	4	M4	17	GG1919	Finland
6/7/2013	6:00:01	70	58	12	24	2	1	2	M2	27	FN39TG	France
6/8/2013	7:00:01	70	73	20	26	16	0	16	M16	23	AC1000	Turkey
6/9/2013	8:00:01	66	69	16	9	9	0	9	M9	11	GG1919	Mexico
6/10/2013	9:00:01	65	57	6	5	12	1	12	M12	26	FN39TG	Finland
6/11/2013	10:00:01	67	70	10	17	15	0	15	M15	19	ACMAX22	Israel
6/12/2013	11:00:01	69	62	2	11	7	1	7	M7	13	FN39TG	South Africa
6/13/2013	12:00:01	69	73	14	2	15	0	15	M15	19	ACMAX22	Israel
6/14/2013	13:00:01	65	61	3	2	6	0	6	M6	9	AC1000	Singapore
6/15/2013	14:00:01	67	59	19	22	20	1	20	M20	19	ACMAX22	Argentina
6/16/2013	15:00:01	65	56	19	11	8	1	8	M8	25	JDNST77	Australia
6/17/2013	16:00:01	67	57	15	7	6	1	6	M6	9	AC1000	Singapore
6/18/2013	17:00:01	66	57	12	5	13	1	13	M13	25	JDNST77	Saudi Arabia
6/19/2013	18:00:01	69	58	8	22	4	1	4	M4	17	GG1919	Finland
6/20/2013	19:00:01	67	55	17	5	7	1	7	M7	13	FN39TG	South Africa

only showing top 20 rows

Joined data resulted as HVAC and Buildings tables were joined

- Now let's select tempchange and country column, then filter the rows where tempchange is 1 and count the number of occurrence for each country

```

val selective = spark.sql("""select Temp_change_diff, Country from HVACJBUILD WHERE
Temp_change_diff = 1""").toDF()
// we saving the selected fields to variable "selective"
selective.registerTempTable("newselective")
// we will register the above joined table to temptable "newselective"
spark.sql("""select Country, count(Temp_change_diff) from newselective group by
Country""").show()
//we will count the temperature difference across each country

```



```
18/05/13 16:01:19 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
18/05/13 16:01:19 INFO Executor: Finished task 59.0 in stage 15.0 (TID 206). 3470 bytes result sent to driver
18/05/13 16:01:19 INFO TaskSetManager: Finished task 59.0 in stage 15.0 (TID 206) in 9 ms on localhost (executor driver) (75/75)
18/05/13 16:01:19 INFO TaskSchedulerImpl: Removed TaskSet 15.0, whose tasks have all completed, from pool
18/05/13 16:01:19 INFO DAGScheduler: ResultStage 15 (show at casestuyll.scala:77) finished in 0.869 s
18/05/13 16:01:19 INFO DAGScheduler: Job 10 finished: show at casestuyll.scala:77, took 0.924789 s
```

```
+-----+
| Country|count(Temp_change_diff)|
+-----+
| Singapore|      230|
| Turkey|      243|
| Germany|      196|
| France|      251|
| Argentina|     230|
| Belgium|     199|
| Finland|     473|
| China|     241|
| Hong Kong|     248|
| Israel|     232|
| USA|     213|
| Mexico|     228|
| Indonesia|     243|
| Saudi Arabia|     233|
| Canada|     232|
| Brazil|     226|
| Australia|     225|
| Egypt|     236|
| South Africa|     237|
+-----+
```

Select tempchange and country column, then filter
the rows where tempchange is 1 and count the
number of occurrence for each country

