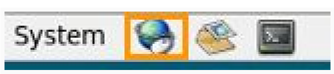
Handling Missing Values in Spark

By the end of this activity, you will be able to perform the following in Spark:

1. Remove rows containing missing values from a DataFrame.
2. Impute missing values with the average value.

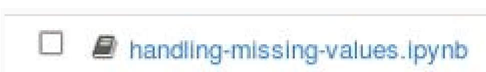
In this activity, you will be programming in a Jupyter Python Notebook. If you have not already started the Jupyter Notebook server, see the instructions in the Reading *Instructions for Starting Jupyter*.

Step 1. **Open Jupyter Python Notebook.** Open a web browser by clicking on the web browser icon at the top of the toolbar:



Navigate to *localhost:8889/tree/Downloads/big-data-4*:

Open the handling missing values notebook by clicking on *handling-missing-values.ipynb:*



Step 2. **Load classes and weather data.** Run the first cell in the notebook to load the *SQLContext* class, create an instance of *SQLContext*, and read the weather data into a DataFrame.

```
In [1]: from pyspark.sql import SQLContext

        sqlContext = SQLContext(sc)

        df = sqlContext.read.load('file:///home/cloudera/Downloads/big-data-4/daily_weather.csv',
                                  format='com.databricks.spark.csv',
                                  header='true',inferSchema='true')
```

Step 3. **Print summary statistics.** We can print the summary statistics for all the columns using *describe():*

```
In [2]: df.describe().toPandas().transpose()
```

Out[2]:

	0	1	2	3	4
summary	count	mean	stddev	min	max
number	1095	547.0	316.24357700987383	0	1094
air_pressure_9am	1092	918.8825513138097	3.1841611803868353	907.9900000000024	929.320000
air_temp_9am	1090	64.93300141287075	11.175514003175877	36.752000000000685	98.9059999
avg_wind_direction_9am	1091	142.23551070057584	69.13785928889183	15.500000000000046	343.4
avg_wind_speed_9am	1092	5.50828424225493	4.552813465531715	0.69345139999974	23.5549781
max_wind_direction_9am	1092	148.9535179651692	67.23801294602951	28.89999999999991	312.199999
max_wind_speed_9am	1091	7.019513529175272	5.59820917078096	1.1855782000000479	29.8407795
rain_accumulation_9am	1089	0.20307895225211126	1.5939521253574904	0.0	24.0199999
rain_duration_9am	1092	294.1080522756142	1598.078778660148	0.0	17704.0
relative_humidity_9am	1095	34.24140205923539	25.472066802250044	6.090000000001012	92.6200000
relative_humidity_3pm	1095	35.34472714825902	22.52407945358728	5.3000000000006855	92.2500000

Let's just look at the statistics for the air temperature at 9am:

```
In [3]: df.describe(['air_temp_9am']).show()

+-------+------------------+
|summary|      air_temp_9am|
+-------+------------------+
|  count|              1090|
|   mean| 64.93300141287075|
| stddev|11.175514003175877|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+------------------+
```

This says that there are 1090 rows. The total number of rows in the DataFrame is 1095:

```
In [4]: df.count()
Out[4]: 1095
```

This means that 5 of the rows in the *air_temp_9am* column are missing values.

Step 4. **Remove missing values.** We can drop all the rows missing a value in any calling using *na.drop():*

```
In [5]: removeAllDF = df.na.drop()
```

Let's look at the summary statistics for *air_temp_9am* with the missing values dropped:

```
In [6]: removeAllDF.describe(['air_temp_9am']).show()

+-------+-----------------+
|summary|     air_temp_9am|
+-------+-----------------+
|  count|             1064|
|   mean| 65.02260949558739|
| stddev|11.168033449415699|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+
```

We can see that the mean and standard deviation is close to the original values: mean is 64.933 vs. 65.022, and standard deviation is 11.175 vs. 11.168.

The count is 1064, which means that 1095 - 1064 = 31 rows were dropped. We can see this agrees with the total number of rows in the new DataFrame:

```
In [7]: removeAllDF.count()
Out[7]: 1064
```

Step 5. **Impute missing values.** Instead of removing rows containing missing values, let's replace the values with the mean value for that column. First, we'll load the *avg* function and make a copy of the original DataFrame:

```
In [8]: from pyspark.sql.functions import avg

        imputeDF = df
```

Next, we'll iterate through each column in the DataFrame: compute the mean value for that column and then replace any missing values in that column with the mean.

```
In [9]: for x in imputeDF.columns:
            meanValue = removeAllDF.agg(avg(x)).first()[0]
            print(x, meanValue)
            imputeDF = imputeDF.na.fill(meanValue, [x])

number 545.0018796992481
air_pressure_9am 918.9031798641055
air_temp_9am 65.02260949558739
avg_wind_direction_9am 142.30675564934032
avg_wind_speed_9am 5.485793050713691
max_wind_direction_9am 148.48042413321312
max_wind_speed_9am 6.9997136588756925
rain_accumulation_9am 0.18202347650615522
rain_duration_9am 266.3936973996038
relative_humidity_9am 34.07743985327712
relative_humidity_3pm 35.14838093290537
```

The *agg()* function performs an aggregate calculation on the DataFrame and *avg(x)* specifies to compute the mean on column *x*. The *agg()* function returns a DataFrame, *first()* returns the first *Row*, and *[0]* gets the first value.

The last line of code uses *na.fill()* to replace the missing values with the mean value (first argument) in column *x*(second argument).

The output of executing this cell prints the mean values for each column and we can see the mean value for *air_temp_9am* is the same as the mean when we removed all the missing values in step 4, *i.e.,* 65.022.

Step 6. **Print imputed data summary statistics.** Let's call *describe()* to show the summary statistics for the original and imputed *air_temp_9am:*

```
In [10]: df.describe(['air_temp_9am']).show()
         imputeDF.describe(['air_temp_9am']).show()
```

```
+-------+-----------------+
|summary|     air_temp_9am|
+-------+-----------------+
|  count|             1090|
|   mean| 64.93300141287075|
| stddev|11.175514003175877|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+

+-------+-----------------+
|summary|     air_temp_9am|
+-------+-----------------+
|  count|             1095|
|   mean| 64.93341058219822|
| stddev|11.149948199920226|
|    min|36.752000000000685|
|    max| 98.90599999999992|
+-------+-----------------+
```

The count for the imputed data is larger since the 5 rows with missing data have replaced with real values. Additionally, we can see that the means are close, but not equal, and this is probably due to round-off error.