École Polytechnique de Montréal
Department of Software Engineering
LOG8415 - Advanced Concepts of Cloud Computing

Spark on EMR Cluster

Elias Jambari	1842489
Charles-Olivier Favreau	1789479
Christian Bukasa Kabulo	1735918

Lab instructor:
S. Amirhossein Abtahizadeh

Lab 2

Montréal, April 3rd, 2020

Table of Contents

1. Introduction

Several cloud computing services are available nowadays in the cloud industry. Each service is tailored for specific needs depending on the task that need to be performed. In this lab, we explore the MapReduce paradigm along with Machine Learning Algorithms deployed on an AWS EMR Cluster.

2. Setup Spark and S3 Bucket

The following section describes how to install Spark and the required dependencies on an EC2 instance on AWS.

2.1 Procedure to setup Spark on EC2

Disclaimer : The following instructions were tested on an **t2.2xlarge** instance. The results and the instructions may change on other types of instances, but the general procedure remains the same.

2.1.1 Install Python 3.7+ and pip

Install Python 3.7 and the pip installer using the following commands :

1. `sudo apt-get update && sudo apt-get upgrade`
2. `sudo apt-get install python3.7`
3. `sudo apt install python3-pip`

Note : Python 3 is already installed on the instance (but it is Python 3.6 instead of Python 3.7). We don't want to remove the existing version that may be used by other programs, that is why we need to install Python 3.7.

2.1.2 Install Java 8

Install Java 8 using the following command :

1. `sudo apt install openjdk-8-jre-headless`

We will need to link Java with Spark later in the process.

2.1.3 Install newest version of Spark and Hadoop

The most recent version can be found here : `https://spark.apache.org/downloads.html`. Download and extract Spark and Hadoop using the following commands :

1. `wget http://www.gtlib.gatech.edu/pub/apache/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz`
2. `sudo tar -zxvf spark-2.4.5-bin-hadoop2.7.tgz`

2.1.4 Link all dependencies

Set the following environment variables to link PySpark with its dependencies.

1. `export SPARK_HOME='/home/ubuntu/spark-2.4.5-bin-hadoop2.7'`
2. `export PATH=$SPARK_HOME:$PATH`
3. `export PYSPARK_PYTHON=python3.7`
4. `export 'JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64'`

2.1.5 Install Python libraries

Install the following libraries in order to run the script properly.

1. `python3.7 -m pip install boto3`
2. `python3.7 -m pip install pyspark`
3. `python3.7 -m pip install stop_words`

2.1.6 Configure AWS CLI

Configure AWS CLI in order to use boto3 in a python script on this EC2 instance.

1. `sudo apt install awscli`
2. `aws configure` (follow steps online below)

More information can be found here on how to configure AWS CLI : `https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html`

2.2 Word Count script and top 20 most frequent words

See the results of the script after running `word_count.py`

3. Build Machine Learning on AWS EMR Cluster

The following section presents an analysis of the dataset provided for TP along with three machine learning models that predict the variable of interest.

3.1 Summary & analysis of the input dataset

3.1.1 Analysis of the input dataset

In this sub-section, we will give a short description of the input data set. The dataset contains variables from different sites (locations) in Beijing which all contain air pollutant variables and meteorological variables. Meteorological data from each air quality site are associated with the nearest meteorological station. The data runs from March 1, 2013 to February 28, 2017, and we noticed that missing information is replaced by NA in the original dataset. Some pre-processing will be applied to the dataset to fulfill missing values.

3.1.2 Data exploration

In order to better understand the dataset, we explored it using various methods. First of all, we studied the individual distribution of each variable. The following table (split in two) shows some statistics about the attributes of the dataset.

Table 3.1: Part 1 - Distribution of variables

	No	year	month	day	hour	PM2.5	PM10	SO2
count	385 325	385 325	385 325	385 325	385 325	377 282	379 362	376 973
mean	17 524.85	2 014.66	6.52	15.72	11.49	79.31	103.91	15.71
std	10 120.70	1.17	3.44	8.80	6.92	80.33	91.18	21.43
min	1	2013	1	1	0	2	2	0.2856
25%	8 759	2014	4	8	5	20	36	3
50%	17 531	2015	7	16	11	55	81	7
75%	26 289	2016	10	23	18	110	145	19
max	35 064	2017	12	31	23	957	999	500

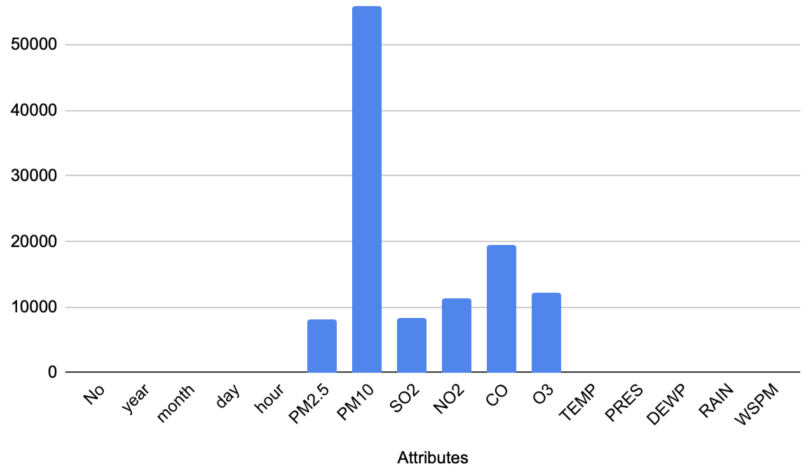
Table 3.2: Part 2 - Distribution of variables

	NO2	CO	O3	TEMP	PRES	DEWP	RAIN	WSPM
count	373 964	365 922	373 133	385 325	385 323	385 320	385 319	385 321
mean	50.20	1 217.89	57.48	13.51	1010.67	2.47	0.0644	1.72
std	35.03	1 153.62	56.63	11.44	10.46	13.80	0.8231	1.24
min	1.02	100	0.2142	-19.9	982.4	-43.4	0	0
25%	23	500	11	3.1	1 002.2	-9	0	0.9
50%	43	900	45	14.5	1 010.3	3	0	1.4
75%	71	1 500	82	23.2	1 019	15.1	0	2.2
max	290	10 000	1 071	41.6	1 042.8	29.1	72.5	12.9

Using these tables we can see that we have a total of 385 325 entries. As we may notice some variables have missing values (e.g. PM2.5, PM10, SO2, NO2, etc.) because their count is lower. We will explain later on how we processed these missing values to make the dataset usable.

The following bar chart shows how many values are missing for each variable.

Table 3.3: Part 2 - Distribution of variables



Variables that measure concentrations are the ones with the most missing values (more than 55 000 for PM10 which corresponds to about 17 % of the total number of entries). What is important here is that we need to take in account that the variable PM10 might not be as precise as the others when building our ML models.

With the following boxplot, it is possible to see that the pressure distribution is normal. This means that the pressure is quite high in the majority of cases.

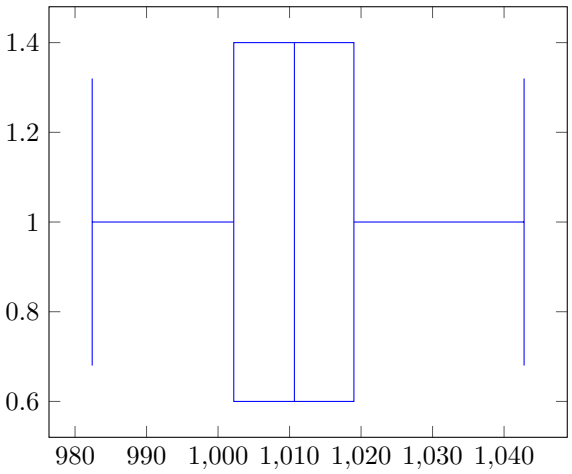


Figure 3.1: pression boxplot

With this second boxplot, we can see that most of the SO2 and O3 values are not higher than the median. We can say that the SO2 and O3 concentrations are relatively low.

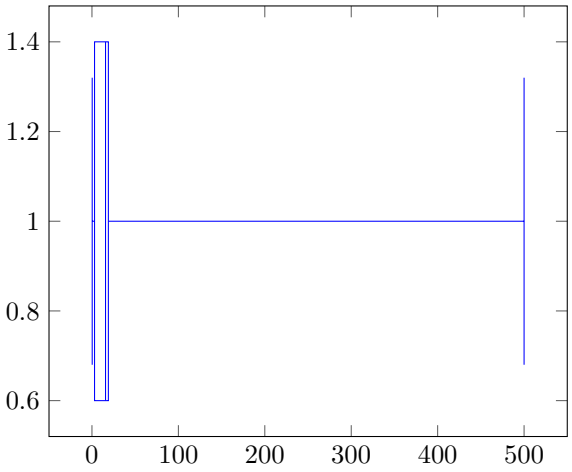


Figure 3.2: SO2 boxplot

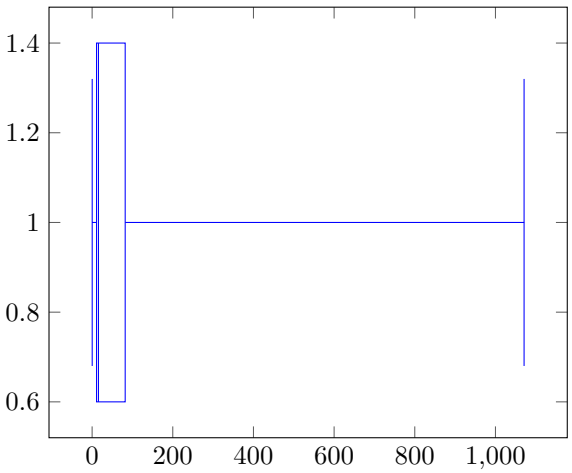


Figure 3.3: O3 boxplot

3.2 Description of ML Methods (validity & results)

3.2.1 Pre-processing

Note : We exported our pipeline into a pickle file to make our code more clear, but the original code can be found in our notebook that we included in the files of this TP : original_notebook.pdf

We performed many pre-processing steps to make the dataset processable by a machine learning algorithm. First of all, we removed *null* values. Missing numerical values were replaced by the average value of the involved feature. For textual features (strings), missing values were replaced by the most frequent string.

The second transformation that we performed was to convert temperatures to categorical values as instructed in the TP indications. The categories are represented by an integer value ranging from 0 (coldest) to 4 (hottest).

The third transformation that we performed is converting the wind direction values to one hot encoded values. We did this transformation because a categorization would have introduced a bias (wind direction values would have been given an order of importance which is not which is

wrong). We applied the same transformation for air-quality monitoring stations. These values were encoded in one hot.

3.2.2 Other transformations

Two more transformations were applied before using the dataset. First, the different variables were **normalized**. This process is very important to ensure that all variables are on the same scale. As we can see in Figures 3.1, 3.2 & 3.3, every variable has a different range and normalizing variables gives them all the same weight so that no bias is introduced. With no normalization, attributes with bigger values could discriminate attributes with smaller values.

Secondly, another transformation that was applied is using **PCA** (Principal Component Analysis). This transformation is basically a **dimension reduction** method which aims to reduce the number of dimensions of the dataset while preserving only the **most relevant components**. We set the number of components to 10 (compared to 45 dimensions in the original transformed dataset). The PCA reduction will provide better performance during training (because of the reduced dataset dimensions) and it will also reduce the noise, because the dataset is "compressed" and it preserves only the essential information.

3.2.3 Choosing attributes

Choosing the most relevant attributes to use is a hard task which is essentially a whole domain itself. There are several methods for doing so including some methods that are very powerful and time consuming. These methods won't be covered in this report because they are beyond the objectives of this work.

While PCA helped us reduce the dimensions of the dataset, it also helped us **identify the most important attributes** that participate to the prediction of the variable of interest. The percentage/number of missing values of each attribute was also taken into account (see Table 3.3) when choosing which **attribute to keep**. An attribute with a lot of missing values should be deprioritized compared to an attribute with less or no missing values. Here are the attributes that were kept the build the ML models.

3.2.4 Splitting the dataset

Data splitting is one of the first decisions to take when starting a prediction problem in artifical intelligence. In our case, this technique allowed us to split the dataset in two groups: a training set and a test set. For this lab, 15% of the dataset will be used as a test set and 85% will be used as a training set.

3.2.5 Model 1 : SVM

Note : Although this was not required, we implemented cross-validation in distributed mode. This problem took us several days to be solved.

Our choice of ML algorithms was based on general AI knowledge provided by the community online and in other courses at Polytechnique Montreal.

First, as instructed in the TP, our first model was Support Machine Vector (SVM) which provided a baseline for our prediction problem. The results obtained using this algorithm were not so great but served as a baseline as shown on the following table.

Table 3.4: Accuracy calculated using distributed cross-validation

0.49498984	0.49434104	0.49369936	0.49380028	0.49727501

Average accuracy using cross-validation : 0.4948

If we take a closer look to SVM, we see that this method is intended for classification problems like ours, but it seems to not be enough to get a good accuracy.

3.2.6 Model 2 : Random Forest

Secondly, Random Forest was used as a second approach. This predicting method is widely used in the AI field [1]. It is a good starting point especially for predictions of multi-class [2]. The results obtained are very impressive as we can see in the following table.

Table 3.5: Accuracy calculated using distributed cross-validation

0.96291757	0.96371055	0.9620808	0.96291704	0.96235474

	precision	recall	f1-score	support
0	0.97	0.97	0.97	5872
1	0.97	0.97	0.97	9742
2	0.96	0.97	0.97	9063
3	0.98	0.97	0.98	11587
4	0.95	0.97	0.96	2269
accuracy			0.97	38533
macro avg	0.97	0.97	0.97	38533
weighted avg	0.97	0.97	0.97	38533

Figure 3.4: Detailed results for Random Forest

Average accuracy using cross-validation : 0.9627

This model will be kept as our final model since it gave us the best accuracy.

3.2.7 Model 3 : Gradient Boosting

Thirdly, "boosting is a method of converting weak learners into strong learners" [3]. Gradient-Boosting trains many models in gradual, additive and sequential manner. And it allows us to improve the learning process of a machine learning model. It uses ensemble learning to boost accuracy model [3]. The accuracy obtained with this method was average.

Table 3.6: Accuracy calculated using distributed cross-validation

0.76794648	0.76894131	0.76572277	0.7658237	0.76585253

Average accuracy using cross-validation : 0.7668

3.3 How to run the program

Please download the file `elasticmapreduce/Gradient_Boosting.pckl` from Google drive here :
`https://drive.google.com/drive/folders/1dEwdzpvw_1RpGBdWL1CjIIjYmHrKzOEO?usp=sharing`

To run the program you need to run the file `predict_method.py`, you need to run the bash file `extra/scriptInstalation.sh` before. After you install all the requirements, `predict_method.py` requires the following arguments in order :

- `aws-logs-862438390833-us-east-1`
- `csv_input_file_path`
- `output_name`
- `elasticmapreduce/SVM.pckl` **[Model 1]**
- `elasticmapreduce/Random_Forest.pckl` **[Model 2]**
- `elasticmapreduce/Gradient_Boosting.pckl` **[Model 3]**

4. Conclusion

In conclusion, this TP was the opportunity for us to learn more about the deployment of ML algorithms over the cloud. We were able to notice that ML algorithms can consume a lot of resources and that it is better to deploy them remotely for large scale applications instead of locally. We also had the chance to familiarize ourselves with the AWS environment using new tools and managing accesses for users for example when using S3. It would be interesting in another TP to discover other services that are available on AWS.

Bibliography

[1] Tony Yiu. Understanding random forest, 2019. https://towardsdatascience.com/understanding-random-forest-58381e0602d2. 8

[2] lanenok. When to use random forest over svm and vice versa?, 2015. https://datascience.stackexchange.com/questions/6838/when-to-use-random-forest-over-svm-and-vice-versa. 8

[3] intellipaat. Gradient boosting in machine learning, 2020. https://intellipaat.com/blog/gradient-boosting-in-machine-learning/. 8