Project Report for the Cloud Computing course project

Groups members: Schirinà Alice, Colarusso Giusy, Antonini Valerio, Ceccaroni Riccardo, Giannotta Gabriele
Project title: *Sentimental analysis of tweets during Covid-19 emergency*

The code, the data and the results of the project are available in the following Github repository.

Deviations from the Project proposal

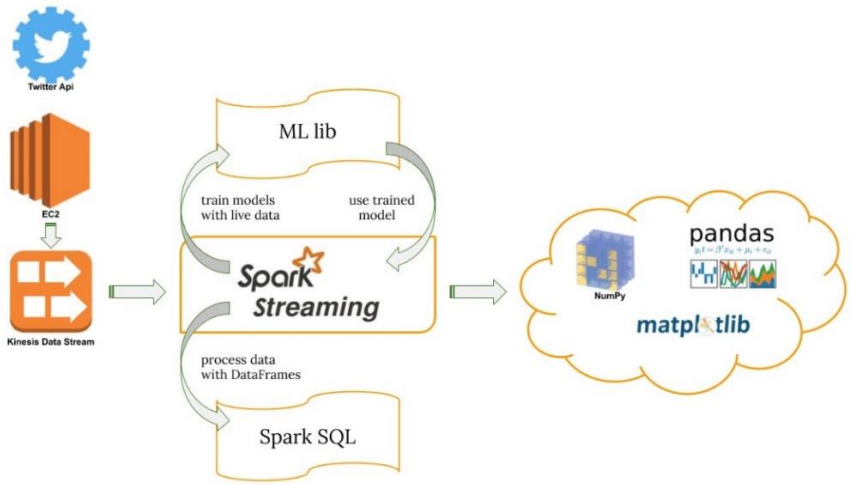
Due to some account Amazon limitations, it was no possible to create EC2 autoscaling group. Thus, we used Elastic Map Reduce (EMR) to automatically create the cluster. The cluster was composed by a master and two principal nodes and by the m5.xlarge instances (4 core and 16 giga of memory). EMR allowed us to directly use Spark without any initial configurations.
Furthermore, we used Bucket S3 in order to save trainset data and the final results.
An other change with respect to the proposal is that we obtained the tweets by Twitter API library (instead of Tweepy).

Description of the problem addressed

The aim of our project is to implement a sentimental analysis of tweets published during the Covid-19 emergency. We used these data to analyze the sentiment of Twitter's users throughout this period. At the end we made a live classifications of some tweets concerned Covid-19.

Design of the solution

In order to achieve this goal, we used different tools. This is the schema of the implementation design:



Description of the Implementation of the solution

First we exploited the based EC2 instance (t2.micro), where we executed the code to obtain tweets by Twitter API and generate a Kinesis flow. From EMR we read the flow, stored the data in Spark SQL , processed the data and made a classification on the tweets exploiting ML Spark library.

Description of the deployment of your solution

As first step we used a dataset from Kaggle about tweets (all concerning Covid-19 emergency) already labeled in the range [-1, 1]. Here the link. The dataset was composed by: tweet id and sentiment expressed. By Twitter API we obtained the content of the tweet associated at that id. We downloaded only the ones having sentiment exactly equal to -1, 0 or 1 representing negative, neutral and positive respectively. In order to have a balanced dataset, we kept the same number of elements for each class (100000). Thus we created a new dataset (stored in S3) structured in the following way: content of the tweet and sentiment. The next step was the processing of the text: we deleted the links, the username, the punctuation, the stop-words, the special characters and the words having less than 3 letters and finally we stemmed the text.
Then we implemented the following pipeline: we tokenized the text, we computed the tf-idf metric for each word and we trained a Logistic Regression model.
The last step was: reading the tweets flow from Kinesis, processing the text by the function defined previously and making a sentiment classification. In addition we draw an interactive pie chart showing the live predictions.

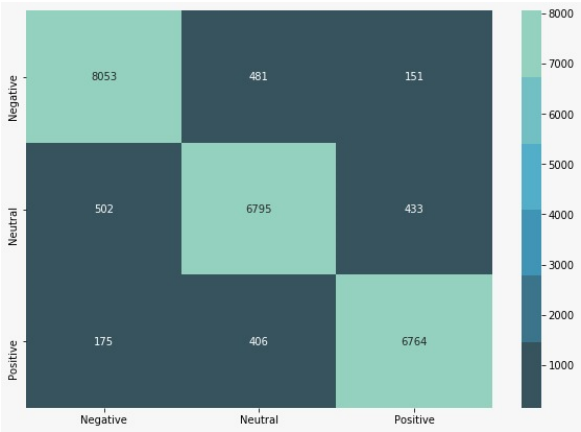
Test/validation design

The solution implemented to test the design of the application is a table (recorded during the simulation) showing the live read tweets and the relative classification. A new row is added to the table as soon as another tweet is obtained (here the link). All works correctly!
To validate our results on a large enough amount of data, we collected tweets for about half an hour. Below we shows the utilized resources in this interval of time.

As we can see in the time needed for the operation we used both master node and the principal nodes. Two VMs have been exploited (at the beginning of the operation), then they have been scaled to three when the workload is increased.

Experimental results

After testing different machine learning models we selected Logistic Regression, the one reaching best performances. On the trainset we obtained an accuracy of about 91%. The execution time was about 30 seconds. To have a better comprehension of the results we computed the following confusion matrix:



As expected, the diagonal shows the great results achieved by the model, i.e. the tweets that the model correctly classified.
Furthermore in order to have a better visualization of the classifications live flow we have realized an interactive pie chart explaining the percentage of each class (here the link).