**CSP 554 BIG DATA TECHNOLOGIES**

**PROJECT REPORT**

**Topic:** Sentiment analysis of Amazon food reviews

**Dataset:**

Amazon Fine Food Reviews from Kaggle ( <https://www.kaggle.com/snap/amazon-fine-food-reviews>)

**Introduction:**

Amazon is an online shopping website where users are encouraged to post product reviews that they purchase. The review comments provided by the customers on the product have substantial information about sentiments towards the products they purchased and the use of products. The review comments are useful to both other vendors and customers. The sentiment analysis of customer reviews helps the vendor to understand the user’s perspectives. They can further use the review comments to improve their products. These reviews are used as a dataset for sentiment analysis in this project.

A typical method to obtain valuable information is to extract the sentiment or opinion from a message. Sentiment analysis of product reviews, an application problem, has recently become very popular in text mining and computational linguistics research. Machine learning technologies are widely used in sentiment classification because of their ability to learn from the training dataset to predict or support decision making with relatively high accuracy. Apache Spark MLlib is one of the most prominent platforms for big data analysis which offers a set of excellent functionalities for different machine learning tasks ranging from regression, classification, and dimension reduction to clustering and rule extraction. Apache Spark MLlib is one of the most highly demanded platform-independent and open-source libraries for big data machine learning which benefits from distributed architecture and automatic data parallelization. This project utilizes Apache Spark MLlib for sentiment analysis on Amazon food reviews.

**Apache Spark Mllib 2.0:**

Apache Spark is a highly scalable, fast, and in-memory big data processing engine and it offers an ability to develop distributed applications using Java, Python, Scala, and R programming languages. It comes with four major libraries, including Apache Spark Streaming, Apache Spark SQL, Apache Spark GraphX, and Apache Spark ML-lib. Apache Spark MLlib is a big data analytics library which provides more than 55 scalable machine learning algorithms that benefit from both data and process parallelization. The library includes implementation of a variety of machine learning strategies, such as classification, clustering, regression, dimension reduction, and rule extraction which enables easy and fast in-practice development of large-scale machine learning applications.

Apache Spark MLlib offers fast, flexible, and scalable implementations of a variety of machine learning components, ranging from ensemble learning and principal component analysis (PCA) to optimization and clustering analysis. Apache Spark MLlib also offer options for distributed processing by parallel processing and support of big data tools that utilize distributed architectures. These criteria will decrease the processing time required and, at the same time, increase the time available to interpret analytics results. This becomes very important when the machine learning task has many predictions to calculate. The distributed architecture can also take advantages of some of the big data tool sets available to help break apart the machine learning component to improve overall running time. Integration is another advantage of Apache Spark MLlib, meaning that the MLlib gains from several software components available in the Spark ecosystem, such as Spark GraphX, Spark SQL, and Spark Streaming, and a wide range of well-organized documentations, including code samples are publicly and freely available to the machine learning community. Given that, Apache Spark is well-suited for querying and trying to make sense of very, very large data sets. The software offers many advanced machine learning and econometrics tools, although these tools are used only partially because very large data sets require too much time when the data sets get too large. The software is not well-suited for projects that are not big data in size. The graphics and analytical output are subpar compared to other tools.

**Alternatives:**

**1. Pytorch:** PyTorch is a python-based library built to provide flexibility as a deep learning development platform. The workflow of PyTorch is as close as we can get to python’s scientific computing library like numpy. It allows interactive debugging and significantly easier to debug as well as to visualize. Instead of predefined graphs with specific functionalities, PyTorch provides a framework for us to build computational graphs as we go, and even change them during runtime. This is valuable for situations where we don’t know how much memory is going to be required for creating a neural network. High level and low-level APIs is also supported. When compared to other alternatives PyTorch is less mature and it has limited references / resources outside of the official documentation which makes it difficult to use.

**2. Tensorflow:** TensorFlow is a library developed by Google that helps in improving the performance of numerical computation and neural networks and generating data flow as graphs. Essentially a machine learning framework, it helps people create deep learning models without the need for rigorous skill sets of a machine learning specialist. It is a software to define and execute machine learning models, in an implementation-independent fashion (i.e., the same code can run on a CPU, on a GPU, parallelized over machines, etc). It comes with many machine learning primitives and features, suitable for building sophisticated ML models. A Google API enabling computation on deep learning and machine learning, TensorFlow gives a graphical representation (Tensorboard) computation flow. The API helps user to write complex neural network design and tune it according to activation values. In summary, it could be said that Apache Spark is a data processing framework, whereas TensorFlow is used for custom deep learning and neural network design.

**3. H2O:** H2O is invented by a Silicon Valley start-up H2O and this new framework (Sparkling Water = H20 + Apache Spark) claims to be the new generation ML tech stack and offers the best high-performance open source machine learning opportunity in the Hadoop eco-system right now. Spark’s MLLib is an open source library built by the Apache community who built Spark. H2O on the other hand is a free standalone library invented by the company H2O which can be integrated with Spark with the 'Sparkling Water' connector. The goal of H2O is to allow simple horizontal scaling to a given problem in order to produce a solution faster. The conceptual paradigm MapReduce (AKA “divide and conquer and combine”), along with a good dependency is required to use it. It's very simple to use. If you want to modify or tweak your ML algorithm then H2O concurrent application structure, enable this type of scaling in H2O. H2O claims better performance than Spark MlLib in terms of convergence speed. H2O is also solely concentrating on ML algorithms, while Spark is a whole framework. H2O is most suited if in little time you wanted to build and train a model. It has support with R, Python and Java, so no programming is not suitable. You can't develop a model from scratch.

**Proposed Approach:**

1. Load data into the Hadoop environment: this requires AWS S3 and Hadoop clusters.
2. Clean and transform the dataset: collect the relevant columns, clean, and tokenize the reviews for sentiment analysis.
3. Use Machine Learning libraries in PySpark for sentiment analysis.

**DataSet Source:**

This dataset consists of reviews of Fine Foods from Amazon. The food reviews are from Oct 1999 to Oct 2012 (approx. 10 years). There is total of 568,454 food reviews from 256,059 users on 74,258 products. There are 260 users with more than 50 reviews. Reviews include product and user information, ratings, and a plain text review. The ratings given by the user for each product is on a scale of 1 to 5, 1 being the lowest. The data is present in the CSV format. Some significant columns of the dataset are as follows:

* Text: complete review text from the user
* Score: the rating of product ranging from 1 to 5
* Summary: Brief text of the review
* Helpfulness Numerator: number of users who found the review helpful
* Helpfulness Denominator: number of users who indicated whether they found the review helpful or not.
* Product Id: unique identifier for each product.

**Implementation Details:**

The project implementation is done in the following phases:

* **Data Cleaning:**

The initial dataset was a comma-separated CSV format. And, to retain the ‘Text’ column’s contents which consists of commas, the dataset was converted to a tab separated file.

* **Data Selection:**

The data is read into RDD using SparkContext and is split with a tab separator. The data is then converted into a dataframe. Filtered out the data which has Score=3 as it is a neutral score (i.e. neither comes under a positive nor negative review). Further, every word in the Text column is changed to lower case for the better analysis. Mainly, ‘Score’ and ‘Text’ columns will be used for performing the analysis.

* **Text Mining:**

All review texts are in lower case and all words that are not alphabets are removed. StopWords in Text column are removed and words are tokenized using built-in libraries StopWordsRemover and Tokenizer from spark MLlib. Removal of stopwords (i.e. frequently used words) enables to focus on the more important words of the review text provided by the user.

* **Storage:**

The data is stored in the Spark SQL table and then cached for enhancing the performance of analysis.

* **Data Analysis:**

Segregate words into positive word list and negative words list from most positive review (Score=5) and the most negative review (Score=1). Sort the words with respect to its frequency to know most used words in positive and negative reviews. Plot a word cloud of the positive and negative words list. Choose specific set of negative and positive words. Count the frequency of these words appearing in the reviews with Score 5 and 1. Select word ‘bad’ as a typical negative words and word ‘great’ as a typical positive. Find the frequency of ‘bad’ in the most negative reviews and the frequency of ‘great’ in the most positive reviews.





**Word Cloud for most positive reviews:**

**Positive Words Selected: ['awesome', 'amazing', 'best', 'good', 'great', 'love', 'loves','like','wonderful','tasty','fresh','organic','happy']**



**Word Cloud for most negative reviews:**

**Negative Words Selected: ['bad', 'hate', 'horrible', 'terrible','worst', 'dislike','disappointed','disappointing','never','waste','awful']**

****

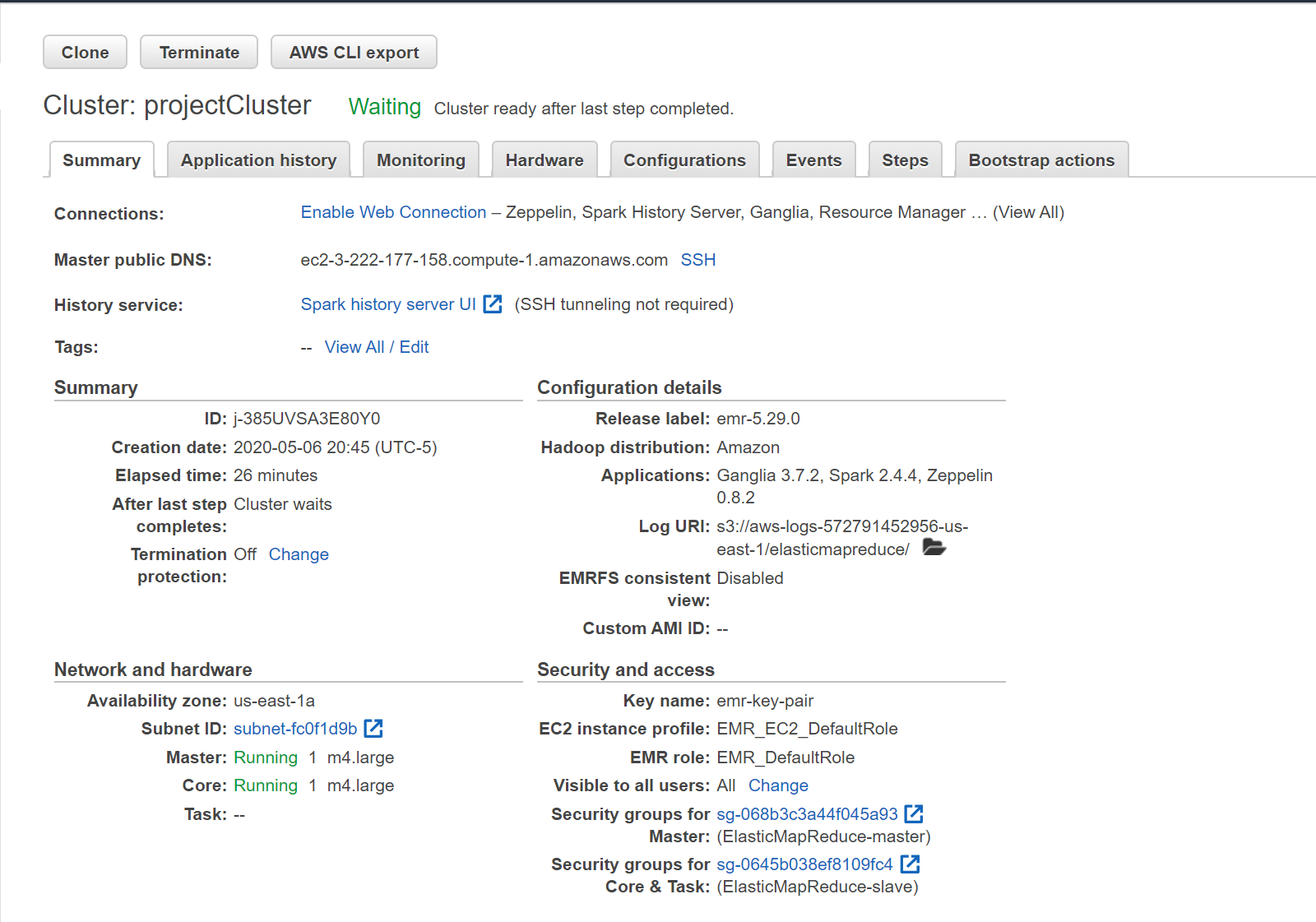
* **Deployment and Execution:**

Create an EMR cluster on AWS by selecting Spark configuration with two m4.large instances in which one acts as a master node and other acts as slave node. Simultaneously, create an S3 bucket and upload the dataset and code files to S3. As transferring the larger dataset through SCP takes longer time than usual, uploading and downloading files to and from S3 is faster. Once, the cluster is up and running, ssh to the master node and download the required files from S3. Also, install the necessary packages for the python code to run on the master node. After downloading files from S3, copy the dataset to Hadoop file system using following command **hadoop fs** **-put filepath /user/hadoop/**

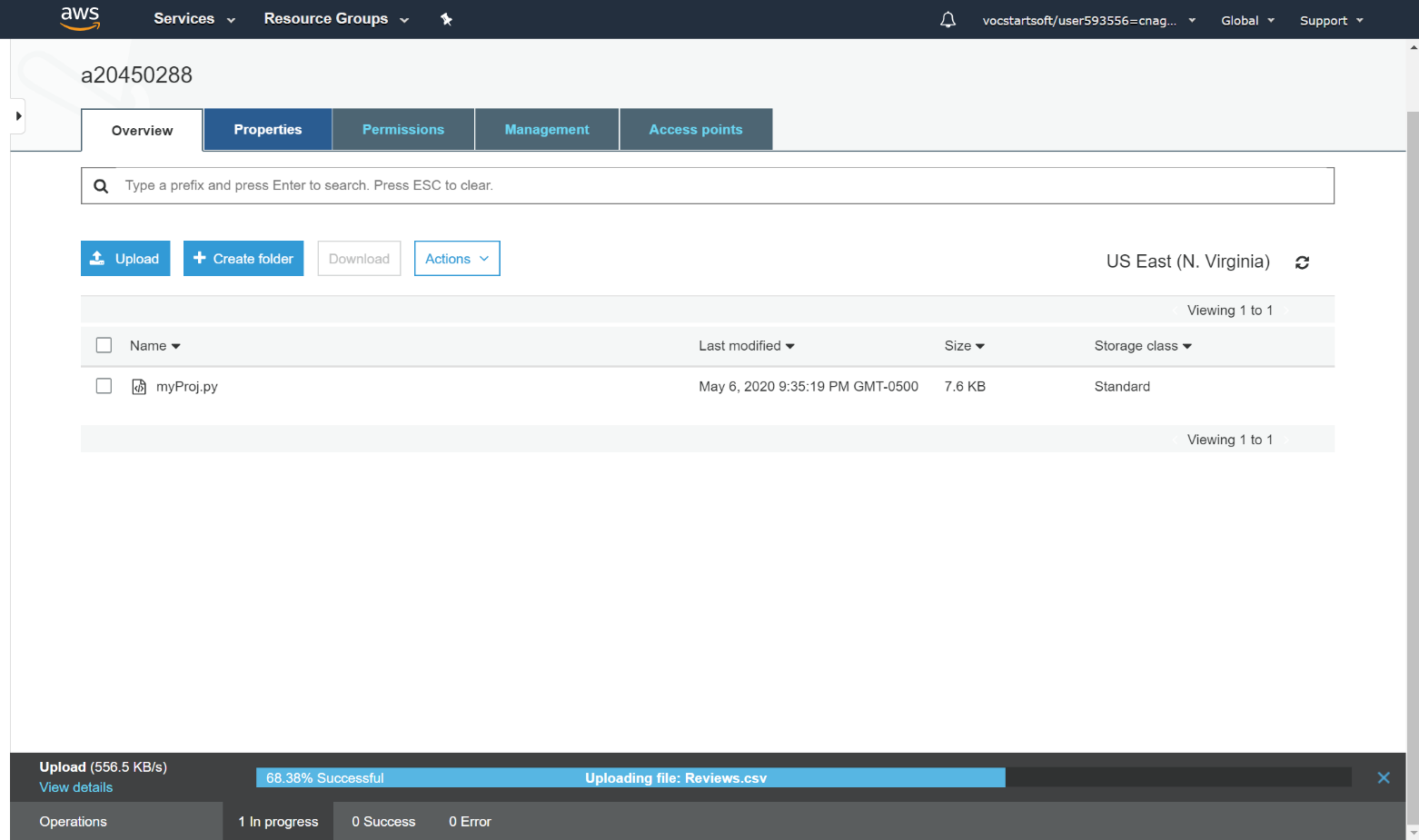
Use **spark-submit filename.py** commandto execute the written code or we can use **execfile(‘filename.py’)** command in the pyspark shell as well (Note: remember to remove the spark context setup from the code before executing the code in pyspark).

**Screenshots of the execution and output:**

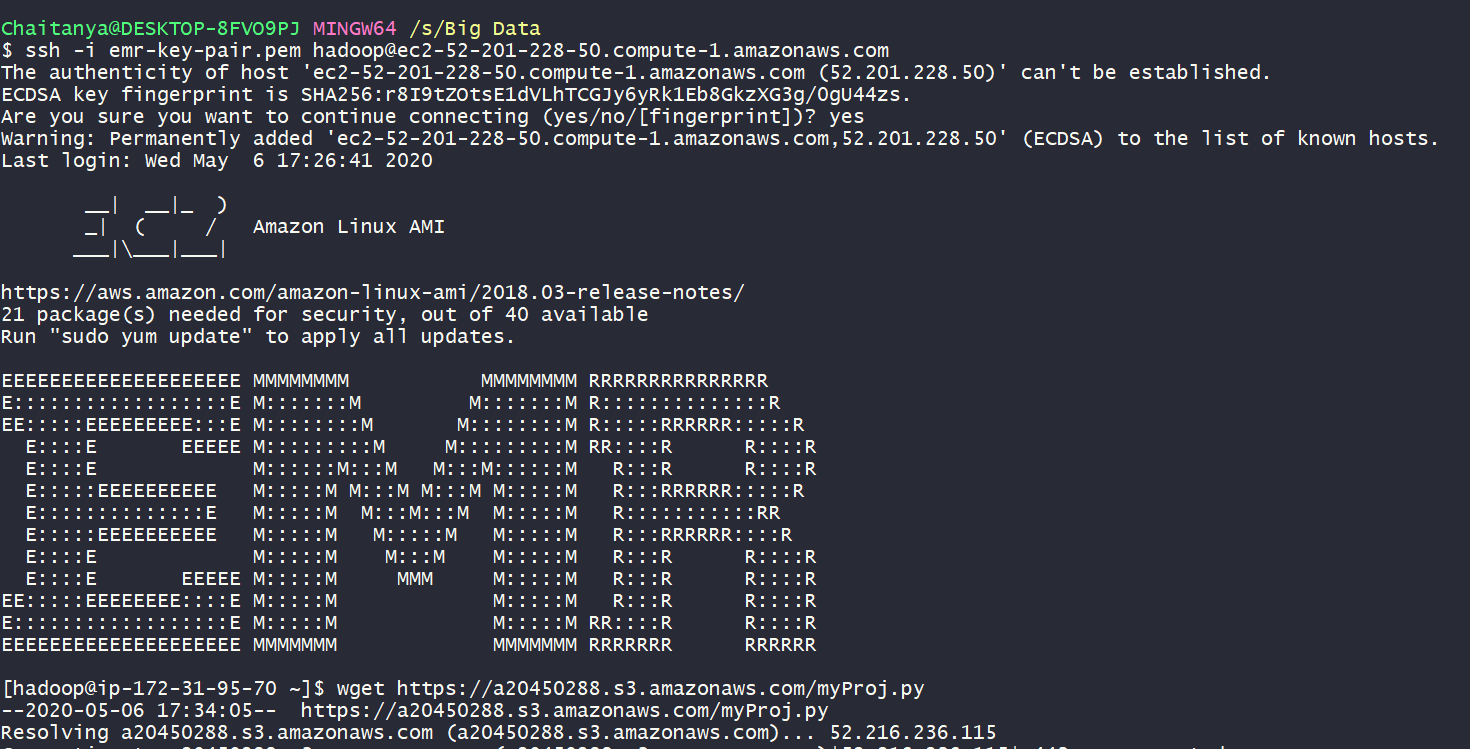
**Creation of Spark cluster on AWS:**



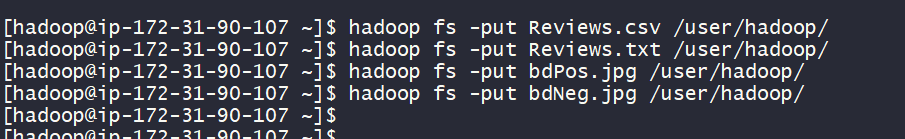
**S3 Bucket creation and files upload:**

****

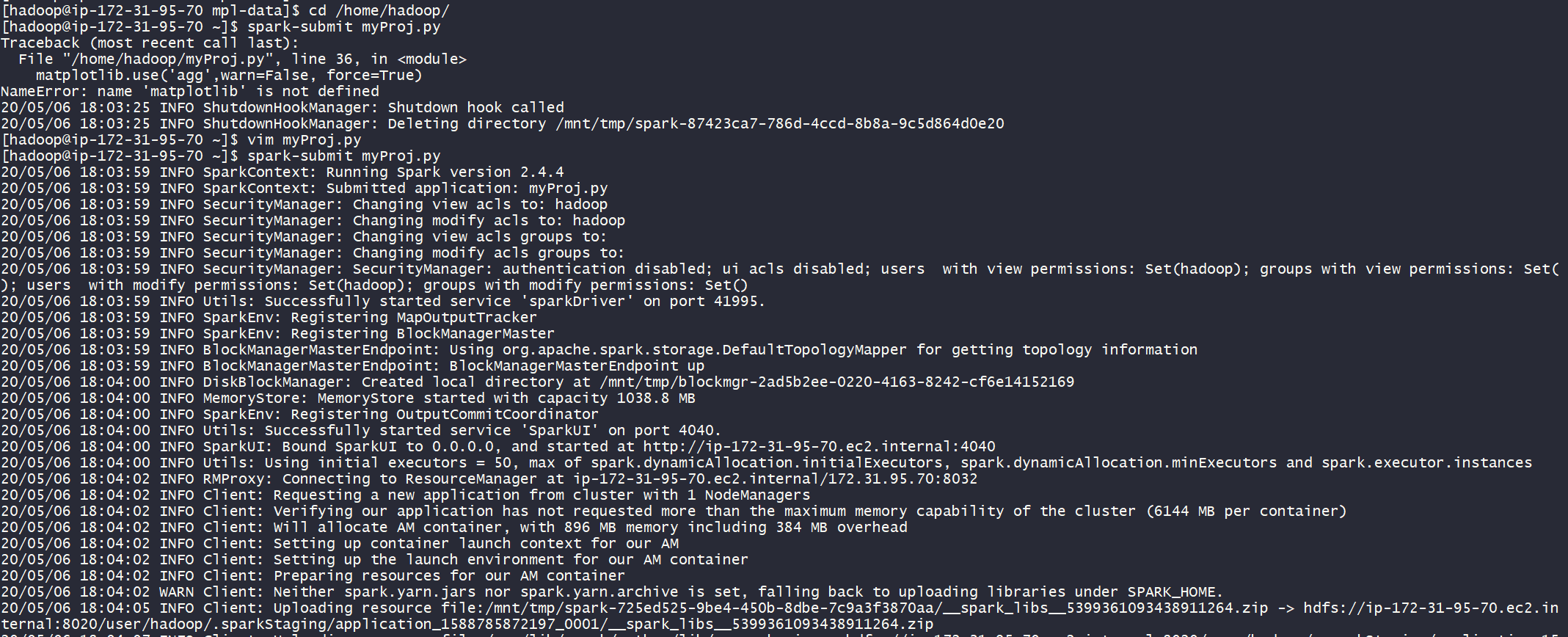
**SSH to EMR Cluster and downloading files from S3 using wget:**



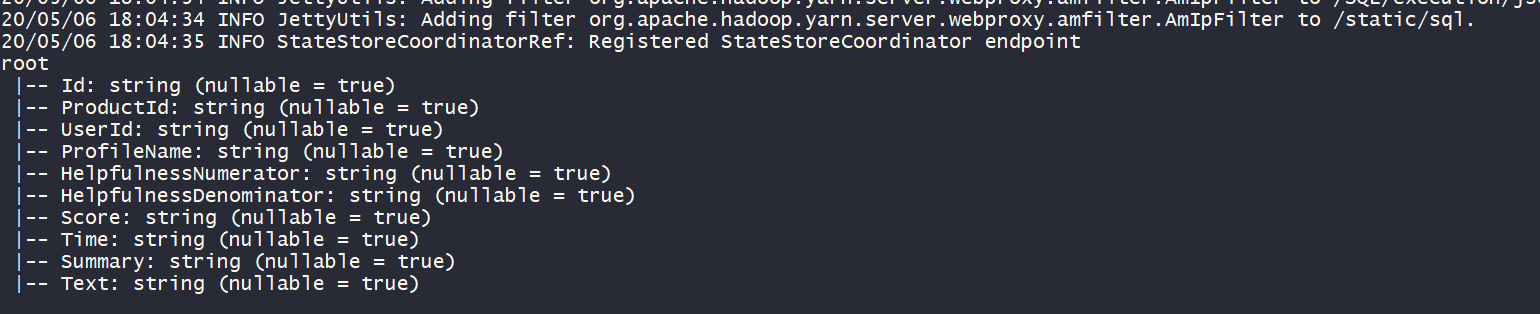
**Put files in Hadoop Cluster:**

****

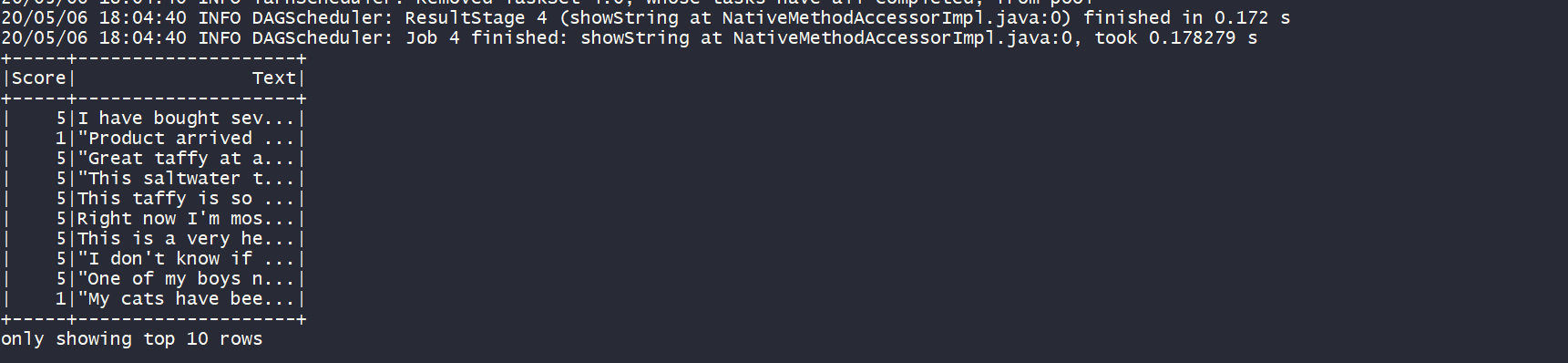
**Code execution using spark-submit:**



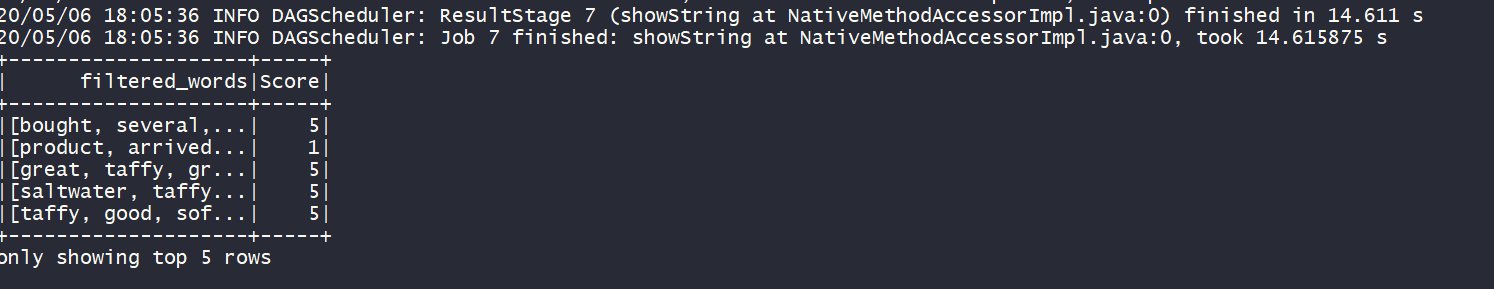
**dataRDD schema:**

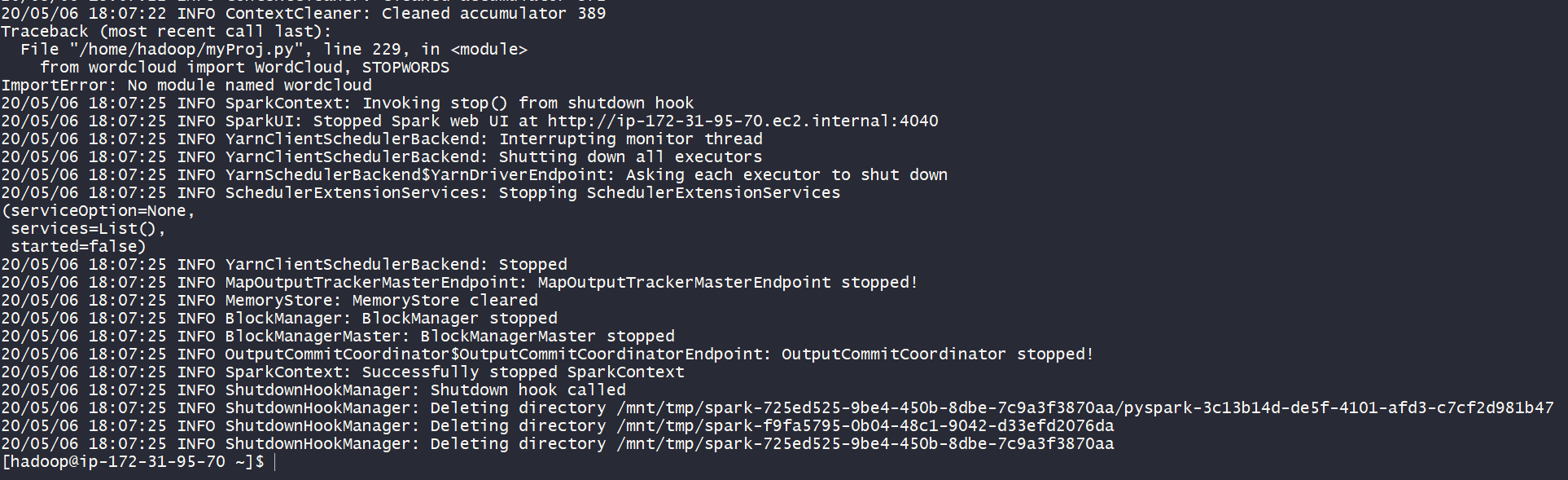


**Review Text and Score columns: Data Selection:**



**Filtered text with Score: Text Mining:**





**Conclusion:**

By executing this project successfully, we understand that Spark is capable of:

* Handling of data of large scale
* Faster computations through Distributed computing.
* Data frame and Tables manipulation.
* Easy to use APIs
* Application of MLlib for data science research and experiments.

As Spark comes packaged with libraries including support for SQL queries, streaming data, machine learning and graph processing. Spark can be used without relying on external packages, modules or software for any part of the Big Data analysis.