

# Big Data Mining Using Spark: Natural Language Processing, Streaming Data Analysis

Most Husne Jahan

mosthusne.jahan@ryerson.ca

## Abstract

Mining massive data is a process of finding hidden knowledge and patterns from large volume of data. Well known Big data mining tool, Hadoop has been designed to perform batch processing in distributed computing platforms. Non-batch processing jobs on Bigdata, like iterative jobs/Machine learning, streaming data analysis are exponentially growing. Apache Spark designed on top of resilient distributed computing framework to alleviate these needs by offering lower latency queries. In this research, we have revealed Text Data Mining(TM) using Spark. We have used SMS collection data to build a Machine Learning application based on Natural Language Processing(NLP). Spark-API is fault-tolerant in large-scale cluster and suitable in-memory computation. We have shown the performance difference between various Machine Learning algorithm on Databricks cloud. We have build model using NLP and TM to identify Spam email. In addition, we used distributed computing such as AWS cloud and Databricks cloud for spark- streaming data analysis.

Keywords—Bigdata Analytics; Natural Language Processing; Spark; Naive Bayes, Decision Tree, Random Forest Classifier, Databricks, AWS.

## I. INTRODUCTION

Natural Language Processing (NLP) is a growing technology, it involves the study and development algorithms to enable computers to comprehend and accomplish operations involving human language [3]. NLP helps analyzing human language to extract commands or useful information. The purpose of our research is to be intimated with NLP on Spark ecosystem and spark-streaming API. Hadoop MapReduce designed to perform: Map and Reduce while Spark explored more than 80 high-level operations. Spark offers 100x faster operation than Hadoop MapReduce in memory and 10x faster on disk. Apache Spark is skilled of in-memory caching which makes iterative algorithms faster on cluster. Spark-streaming is well established API for real-time data processing along with distributed processing. Spark-MLlib consists of in built library for machine learning algorithm. Spark has the ability to make schema from structured and unstructured data, then SQL-queries can be accomplished.

In our research, firstly, we have accomplished NLP to analyze the performance of different Machine Learning model like Naïve Bayes, Decision tree, Random forests by creating a Databricks cluster. In addition, a case study performed for collecting Spark-streaming data from twitter on EMR cluster and storing, analyzing the collected tweets using Dynamo Db and Databricks cluster.

## II. RELATED WORK

In the paper [2], authors have shown an approach to detect targeted phishing email attacks using machine learning. They applied (NLP) techniques to parse each sentence and identify the semantic roles of important words in the sentence in relation to the predicate. Another research [4], author presents a survey, Student Directed Discussion Surveys (SDDS). The improvement of the design, administration, and analysis of student surveys can be done by creating assessments that consist of few general discussion questions. They have analyzed the result with Natural Language Processing (NLP) and shown comparison of the Likert scaled survey results with the SDDS textual data. Using Natural Language Processing and Text Mining, unstructured text of resumes can be analyzed [6]. Researcher proposed a way to identify knowledge profiles for Software Engineering Positions. They invented the KP GENERATOR, which successfully bring out the technical knowledge from the resumes. Business process models can also be generated using NLP [11]. The documents existing in an organization can be used for that. They applied Sentence Level Analysis and Text Level Analysis for their model. Another researcher [12] proposed a context; they used Kafka for data ingestion and spark for stream processing, and data visualization. Kafka has a strong message brokering system to import tweets, and to distribute it based on topics. Kafka make the streaming data available over consumers' nodes for analytical tools. Spark Streaming is related to these consumers directly and analyze data.

Researcher performs a comparative analysis Hadoop and spark for meta-actions generating algorithm [15]. Their system for distributed action rule mining reduced time.

### III. SPARK-STREAMING (A CASE STUDY)

#### Spark Ecosystem:

Databricks and apache Spark is open source. Databricks cluster makes spark suitable and quicker distributed computing platform. Spark has following advantages-

- APIs: Java, Scala, Python and R,
- Library: Spark-SQL structured data processing, MLlib for machine learning, GraphX for graph processing, Spark Streaming for real-time data analysis.
- Data sources: HDFS, Cassandra, HBase, and S3.
- Spark run on Hadoop, Mesos, standalone, or in the cloud.

#### Spark Framework

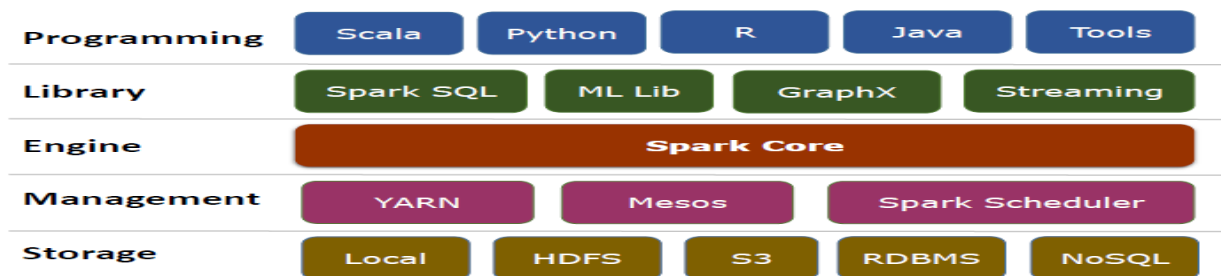


Fig-01: Spark ecosystem

Resilient Distributed Datasets (RDD) is a primary data structure of Spark [2]. It is an irreversible distributed collection of objects which provides backtracking to complete a task instead of starting everything from the scratch.

Each dataset in RDD is shared into logical partitions, which may be computed on different nodes of the cluster. It is a fault-tolerant collection of elements that can be operated on in parallel. There are two ways to create RDDs:

- parallelizing an existing collection in the driver program or
- referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop Input Format.

RDD supports two types of operations-

- a) Transformations: we can create a new dataset(RDD) from an existing one(RDD), which is called transformation
- b) Actions: transformation functions like map, filter, flatMap, groupByKey, reduceByKey, aggregateByKey, [5] etc execute when action happened. It returns a value to the driver program after running a computation on the dataset.



Fig-02: The core Spark engine

In our research work, we have implemented the following-

- Spark-MLib
- Spark- SQL
- Spark-Streaming

### Cluster Setup:

We have performed the experiments on AWS Cluster and transferred database to Databricks cluster. The cluster information is as shown below:

- EMR-5.13.0
- Hadoop distribution: Amazon 2.8.3
- Applications: Hive 2.3.2, Hue 4.1.0, Tez 0.8.4, HBase 1.4.2, Zeppeline 0.7.3, Spark 2.3.0
- Instance type: m4.large(1master,2 core)
- 4 vcore,8GB memory

Work flow for Streaming data analysis:

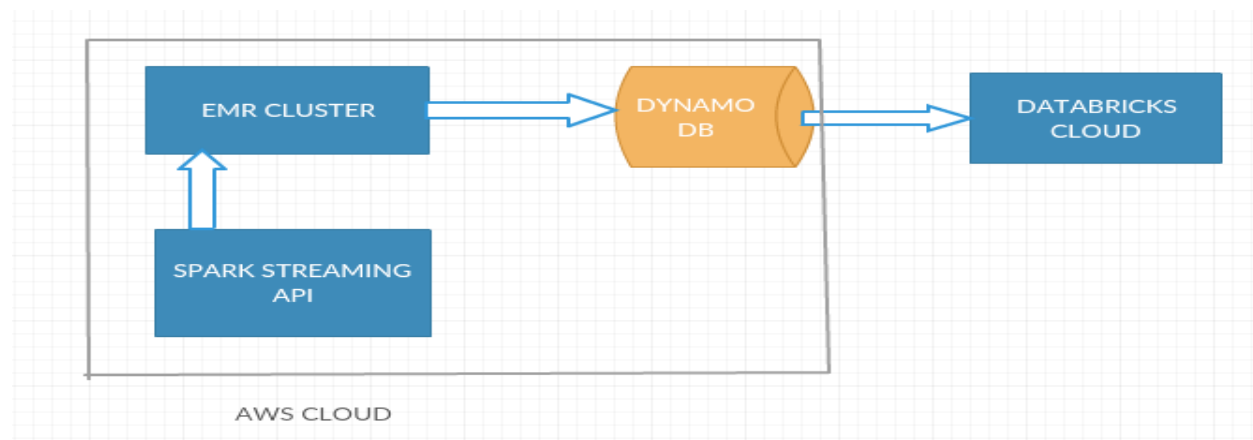


Fig-03: Spark-Streaming

### EMR: Elastic MapReduce

In this experiment, we have used EMR cluster, which provided a managed Hadoop framework. EMR cluster is convenient to process vast amounts of data across dynamically scalable Amazon EC2 instances. In EMR we can run Apache Spark, HBase, Presto, and Flink ,Zeppelin etc. It can co-operate with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB [8].

Summary	Configuration details
<b>ID:</b> j-3IMGP8CNOUCZ6	<b>Release label:</b> emr-5.13.0
<b>Creation date:</b> 2018-04-23 22:02 (UTC-4)	<b>Hadoop</b> Amazon 2.8.3
<b>Elapsed time:</b> 44 minutes	<b>distribution:</b>
<b>Auto-terminate:</b> No	<b>Applications:</b> Hive 2.3.2, Hue 4.1.0, Tez 0.8.4, HBase 1.4.2, Zeppelin 0.7.3, Spark 2.3.0
<b>Termination protection:</b> Off <a href="#">Change</a>	<b>Log URI:</b> s3://aws-logs-132617414309-us-east-1/elasticmapreduce/



```

1 data.groupBy("_c4").count().show(5)

(4) Spark Jobs
+-----+-----+
|_c4|count|
+-----+-----+
|location (S)|1|
|Don Mueang, Bangkok|1|
|None|26|
|toronto|1|
|France|2|
+-----+-----+
only showing top 5 rows

Command took 0.63 seconds -- by mosthusne.jahan@ryerson.ca at 4/25/2018, 12:12:47 PM on spark-project

```

Fig-05: Analysis collected tweets on Databricks cluster

## IV. NATURAL LANGUAGE PROCESSING

Applying NLP, we have build 3-classifier using Spark-Mlib. The stages are as below-

Stage 1: Load Dataset on Databricks cluster

In our research, we have used the dataset that contains one set of 5,574 SMS messages in English. Which are labeled as ham and spam. The document contains over 90,000 words.

```

1 import pyspark
2 from pyspark.sql import SparkSession
3 spark = SparkSession.builder.appName('nlp').getOrCreate()
4 data = spark.read.csv("/FileStore/tables/SMSspamCollection", inferSchema=True, sep='\t')
5 data = data.withColumnRenamed('_c0', 'class').withColumnRenamed('_c1', 'text')
6 data.show()
7

(3) Spark Jobs
  Job 126 View (Stages: 1/1)
  Job 127 View (Stages: 1/1)
  Job 128 View (Stages: 1/1)

data: pyspark.sql.dataframe.DataFrame
class: string
text: string

+-----+-----+
|class|text|
+-----+-----+
|ham|Go until jurong p...|
|ham|Ok lar... Joking ...|
|spam|Free entry in 2 a...|
|ham|U dun say so earl...|
+-----+-----+

```

Fig-06: Dataset load on Databricks cluster

Stage 2: tokenizer to break down the file into words.

In other words, breaking up the given text into units/tokens is called tokenization. Then we can find tokens as words or number or punctuation mark. Tokenization accomplish this task by locating word boundaries. Word boundaries are defined as the ending point of a word and beginning of the next word. However, tokenization is also known as word segmentation.

Stage 3: stop word remover.

In this stage, we have removed stop words commonly used words (such as “the”, “and”, “if”). Stop words are ignored by search engine, both when indexing entries and retrieving them as the result of a search query.

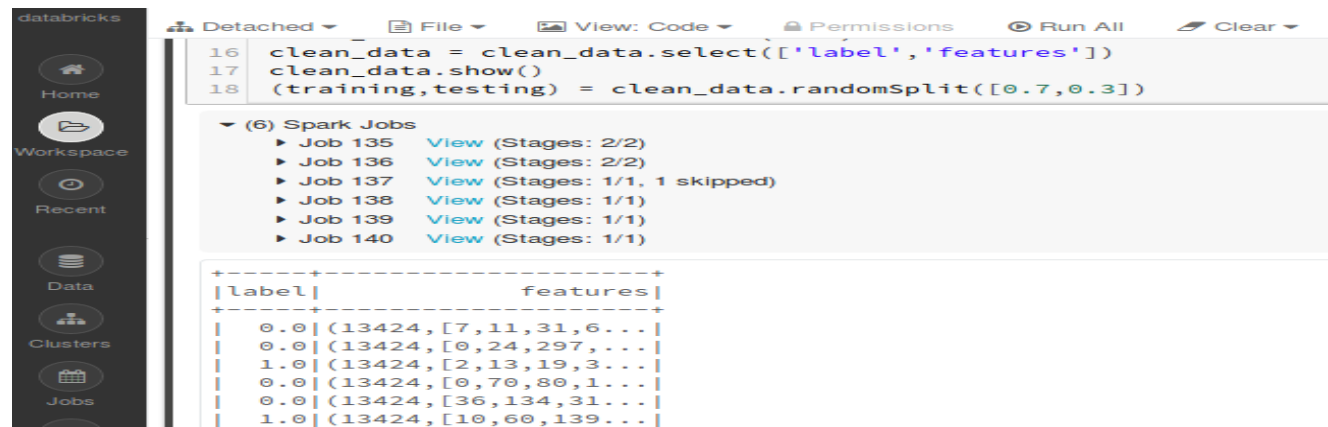
Sample text with stop words	Without stop words
Can special flooring reduce fall injuries for elderly?	special flooring reduces fall injuries elderly
Painkiller prescribed by the doctor, may not cure	Painkiller prescribed doctor cure

#### Stage 4: Count Vectorizer

Count Vectorizer helps if the vocabulary/bag of words is huge as it speeds up the process which in turn reduced the computational time.

#### Stage 5: Feature extraction using TF-IDF

Feature extraction technique is familiar using term frequency-Inverse document frequency. It uses entirely the tokens in the dataset as vocabulary. Frequency of occurrence of a token from vocabulary in each document consists of the term frequency and number of documents in which token occurs determines the Inverse document frequency. Both these TF and IDF matrices for a document are multiplied and normalized to form TF-IDF of a document.



```
16 clean_data = clean_data.select(['label','features'])
17 clean_data.show()
18 (training,testing) = clean_data.randomSplit([0.7,0.3])
```

▼ (6) Spark Jobs

- ▶ Job 135 View (Stages: 2/2)
- ▶ Job 136 View (Stages: 2/2)
- ▶ Job 137 View (Stages: 1/1, 1 skipped)
- ▶ Job 138 View (Stages: 1/1)
- ▶ Job 139 View (Stages: 1/1)
- ▶ Job 140 View (Stages: 1/1)

label	features
0.0	(13424, [7, 11, 31, 6...]
0.0	(13424, [0, 24, 297, ...]
1.0	(13424, [2, 13, 19, 3...]
0.0	(13424, [0, 70, 80, 1...]
0.0	(13424, [36, 134, 31...]
1.0	(13424, [10, 60, 139...]

Fig-07: Clean data after feature extraction

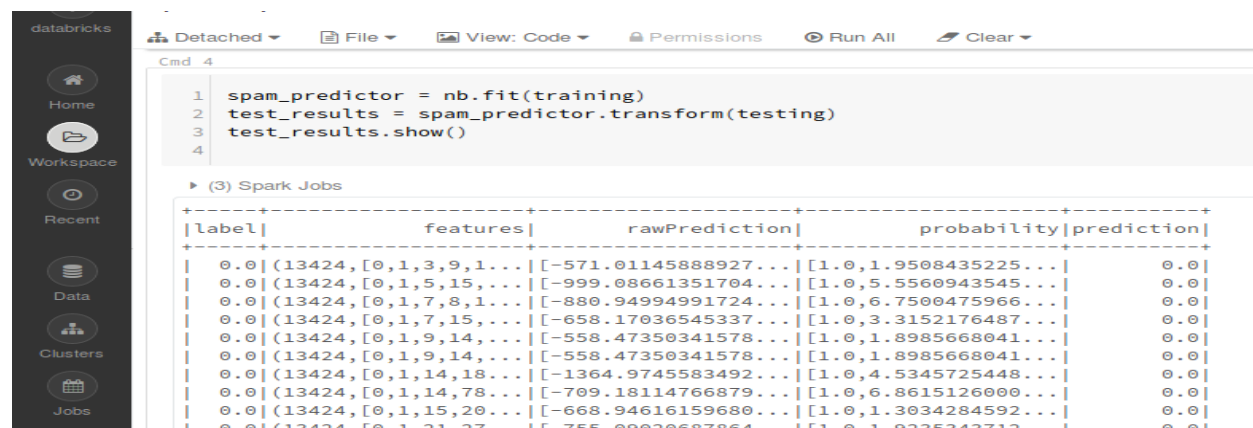
We have split our clean data as 70/30 for training and testing respectively.

#### Stage 5: Machine learning algorithm for classification

##### Algorithm-1: Naïve Bayes

At first, we have applied Naïve Bayes multiclass classification algorithm. Training is very efficient for this algorithm.

If the feature has given label in training data, it can calculate conditional probability distribution. For prediction, it applies Bayes' theorem to calculate the conditional probability distribution.



```
1 spam_predictor = nb.fit(training)
2 test_results = spam_predictor.transform(testing)
3 test_results.show()
4
```

▶ (3) Spark Jobs

label	features	rawPrediction	probability	prediction
0.0	(13424, [0, 1, 3, 9, 1...]	[-571.01145888927...]	[1.0, 1.9508435225...]	0.0
0.0	(13424, [0, 1, 5, 15, ...]	[-999.08661351704...]	[1.0, 5.5560943545...]	0.0
0.0	(13424, [0, 1, 7, 8, 1...]	[-880.94994991724...]	[1.0, 6.7500475966...]	0.0
0.0	(13424, [0, 1, 7, 15, ...]	[-658.17036545337...]	[1.0, 3.3152176487...]	0.0
0.0	(13424, [0, 1, 9, 14, ...]	[-558.47350341578...]	[1.0, 1.8985668041...]	0.0
0.0	(13424, [0, 1, 9, 14, ...]	[-558.47350341578...]	[1.0, 1.8985668041...]	0.0
0.0	(13424, [0, 1, 14, 18...]	[-1364.9745583492...]	[1.0, 4.5345725448...]	0.0
0.0	(13424, [0, 1, 14, 78...]	[-709.18114766879...]	[1.0, 6.8615126000...]	0.0
0.0	(13424, [0, 1, 15, 20...]	[-668.94616159680...]	[1.0, 1.3034284592...]	0.0
0.0	(13424, [0, 1, 21, 27...]	[-755.09020687864...]	[1.0, 1.9235343712...]	0.0

Fig-08: Prediction using Naïve Bayes on test dataset

The screenshot shows a Databricks workspace for a notebook named 'spark-nlp-nb (Python)'. The interface includes a sidebar with navigation options like Home, Workspace, Recent, Data, and Clusters. The main area displays a code editor with the following Python code:

```

1 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
2 acc_eval = MulticlassClassificationEvaluator()
3 acc = acc_eval.evaluate(test_results)
4 print("Accuracy of NaiveBayes model at predicting spam was: {}".format(acc))

```

Below the code, the execution results are shown for three Spark jobs (144, 145, 146), all of which completed successfully. The final output of the script is:

```

Accuracy of NaiveBayes model at predicting spam was: 0.921220047603338

```

The command took 2.31 seconds to execute.

Fig-09: Accuracy of Naïve Bayes classifier Model

#### Algorithm-2: Decision tree classifier

We have used decision tree which is a supervised learning algorithm that is mostly used in classification problems. It is suitable for both categorical and continuous variables. Decision trees have root node, that represents entire population. Sub-node creates by splitting a node, when a sub-node splits into further sub-nodes then it is defined as decision node. When a node can not split further then it is defined as terminal node. Sub-tree can be assumed by considering a sub-section of the entire tree. Parent node can be divided into sub-nodes, where as sub-nodes are the child of parental node.

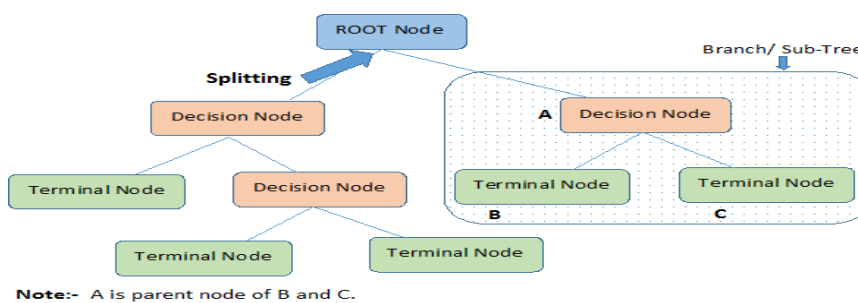


Fig-10: Decision tree classifier

The screenshot shows a Databricks workspace for a notebook named 'Spark-nlp-dct (Python)'. The interface includes a sidebar with navigation options like Home, Workspace, Recent, Data, Clusters, Jobs, and Search. The main area displays a code editor with the following Python code:

```

1 from pyspark.ml.classification import DecisionTreeClassifier
2 dct = DecisionTreeClassifier()
3 spam_predictor = dct.fit(training)
4 test_results = spam_predictor.transform(testing)
5 test_results.show()

```

Below the code, the execution results are shown for nine Spark jobs (114, 115, 116, 117, 118, 119, 120, 121, 122), all of which completed successfully. The final output of the script is a table showing the results of the prediction:

label	features	rawPrediction	probability	prediction
0.0	(13424, [0,1,2,13,...]	[2542.0,35.0]	[0.98641831587116...	0.0
0.0	(13424, [0,1,3,9,1...]	[2542.0,35.0]	[0.98641831587116...	0.0
0.0	(13424, [0,1,14,18...]	[3.0,0.0]	[1.0,0.0]	0.0

Fig-11: Prediction using Decision tree classifier on test dataset



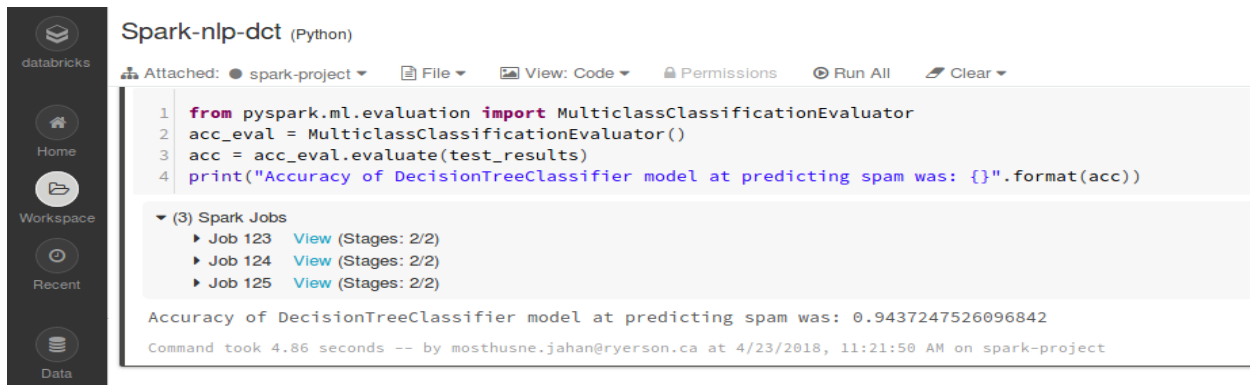


Fig-12: Accuracy of Decision tree classifier Model

### Algorithm-3: Random forests

Finally, we used Random Forest algorithm, popular supervised classification algorithm which can solve overfitting if there are enough trees in the forest. There are two stages in Random Forest algorithm, random forest creation and make a prediction. At first it randomly selects features and calculate the node using best split point. Then daughter nodes are explored by best split. This process continues repeating to build forest until “1” number of node has been reached. By calculating the high voted predicted target, the final prediction from the random forest algorithm performed.

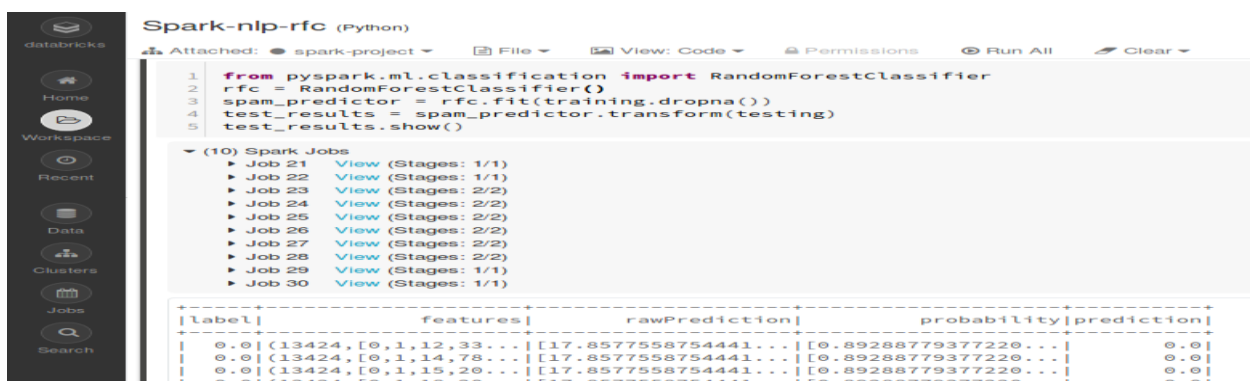


Fig-13: Prediction using Random forests classifier on test dataset

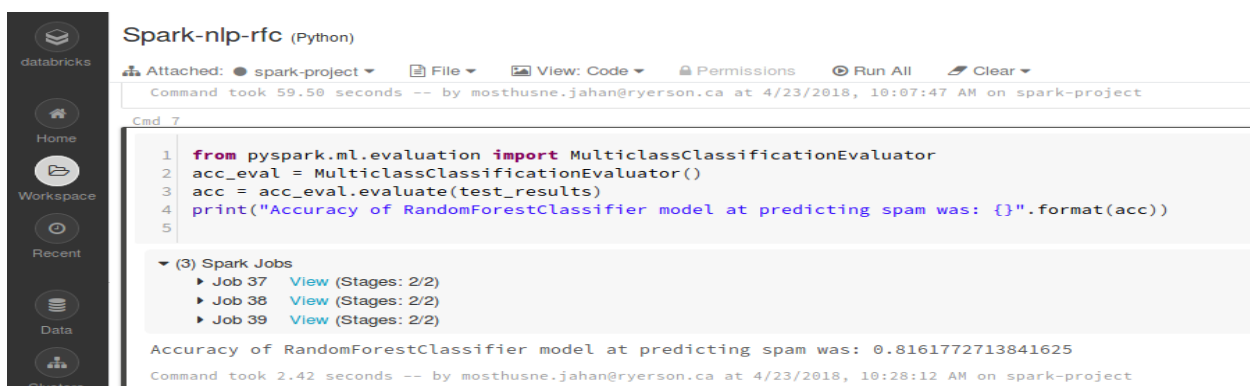


Fig-14: Accuracy of Random forests classifier Model



Overall, our models showed the following accuracy-

Naïve Bayes classifier Model	Decision tree classifier Model	Random forests classifier Model
92.12%	94.37%	81.6%

## V. CONCLUSION AND FUTURE WORK:

Spark computational framework designed for distributed cluster which can cache in memory storage, it is quite faster for iterative queries. We have found best performance from Decision tree classifier Model. Our case study showed which area generates more tweets of the occurrence and also the reaction of the users. We will explore our work on collecting millions of tweets to implement NLP and train model for detecting user's reaction on multi-node cluster considering large volume of dataset.

## REFERENCES

- [1] Apache Hadoop. <http://hadoop.apache.org/>.
- [2] "Detecting Phishing Attacks Using Natural Language Processing and Machine Learning"- Tianrui Peng, Ian G. Harris, Yuki Sawa. 2018 12th IEEE International Conference on Semantic Computing
- [3] D. Jurafsky and J. H. Martin, *Speech and language processing*, vol. 3. Pearson, 2014.
- [4] "Improving Student Surveys with Natural Language Processing"- Karoline M. Hood, Patrick K. Kuiper. 2018 Second IEEE International Conference on Robotic Computing
- [5] "Big Data Management Processing with Hadoop MapReduce and Spark Technology: A Comparison"- Ankush Verma, Ashik Hussain Mansuri, Dr. Neelesh Jain. 2016 Symposium on Colossal Data Analysis and Networking (CDAN).
- [6] "Natural Language Processing and Text Mining to Identify Knowledge Profiles for Software Engineering Positions"- Rogelio Valdez-Almada, Oscar M. Rodriguez-Elias. 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)
- [7] <https://aws.amazon.com/dynamodb/>
- [8] <https://aws.amazon.com/emr/>
- [9] <https://spark.apache.org/docs/latest/ml-guide.html>
- [10] <http://spark.apache.org/docs/latest/sql-programming-guide.html>
- [11] "Extracting Business Process Models using Natural Language Processing (NLP) Techniques"- Konstantinos Sintoris, Kostas Vergidis. 2017 IEEE 19th Conference on Business Informatics
- [12] "Developing a Real-time Data Analytics Framework for Twitter Streaming Data"- Babak Yadrangjaghdam, Seyedfaraz Yasrobi, Nasseh Tabrizi. 2017 IEEE 6th International Congress on Big Data
- [13] "Action Rules for Sentiment Analysis on Twitter Data using Spark"- Jaishree Ranganathan, Allen S. Irudayaraj, Angelina A. Tzacheva. 2017 IEEE International Conference on Data Mining Workshops