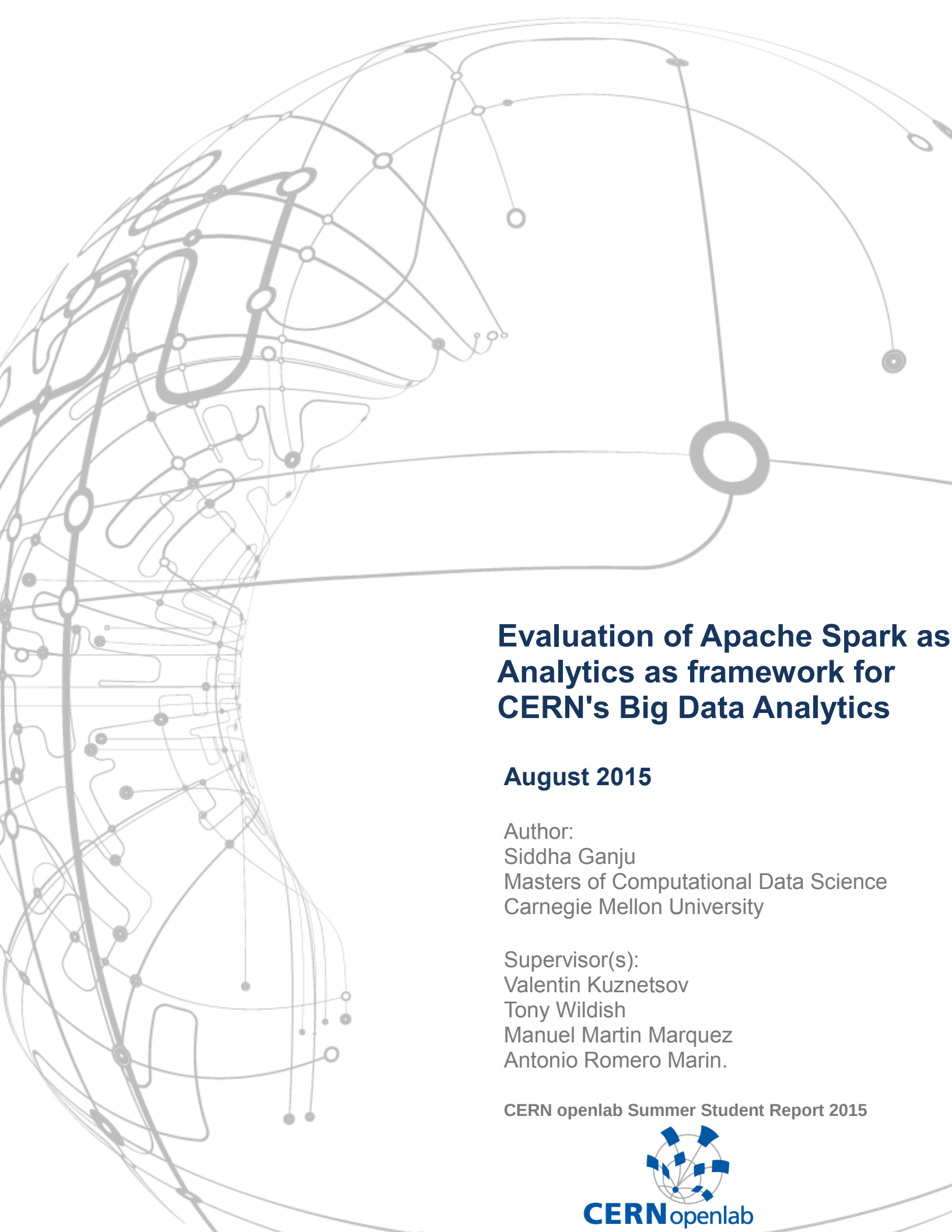


Evaluation of Apache Spark as Analytics as framework for CERN's Big Data Analytics

August 2015

Author:
Siddha Ganju
Masters of Computational Data Science
Carnegie Mellon University

Supervisor(s):
Valentin Kuznetsov
Tony Wildish
Manuel Martin Marquez
Antonio Romero Marin.

CERN openlab Summer Student Report 2015

CERN openlab

Project Specification

The goal of this openlab summer student project is to analyse Apache Spark as a framework for the big data analytics framework of CERN. It utilizes MLlib and Spark Streaming along with Python and its libraries such as scikit-learn and scipy. The objective was to determine if the metadata of CMS can be analysed efficiently, in terms of CPU usage and time for completion of the job using Apache Spark. Apache Spark is a framework providing speedy and parallel processing of distributed data in real time. Additionally it provides powerful cache and persistence capabilities. Because of these features Apache Spark is applied to data analytics problems. Components of Spark like Spark Streaming and MLlib (Spark native machine learning library) make analysis possible.

Software Specifications:

1. Apache Spark Release 1.4.0
2. Python 2.7.5
    a. scikit-learn package
        i. ensemble
        ii. cross_validation
        iii. neighbors
        iv. decomposition
    b. pandas package
        i. The Pandas acronym comes from a combination of panel data and Python data analysis. It targets five typical steps in the processing and analysis of data, regardless of the data origin: load, prepare, manipulate, model, and analyze.
        ii. Pandas places much emphasis on flexibility, for example, in handling disparate cell separators. Moreover, it reads directly from the cache or loads Python objects serialized in files by the Python pickle module.
    c. Numpy package: manipulating arrays
    d. Optparse package

Abstract

I present an evaluation of Apache Spark for streamlining predictive models which use information from CMS data -services. The models are used to predict which datasets will become popular over time. This will help to replicate the datasets that are most heavily accessed, which will improve the efficiency of physics analysis in CMS. The evaluation will cover implementation on Apache Spark framework to evaluate quality of individual models, make ensembles and choose best predictive model(s) for new set of data.

The task in this project is to predict popular datasets. Finding the popular datasets is helpful in a two-fold way. Firstly, it helps in providing expeditious access to datasets that might be required and secondly, it helps in finding which might become the 'hot topics' in high energy physics. It is also necessary to define what a popular dataset is. Based on the data collected, some parameters such as nusers, naccesses and tot_cpu can be said to define popularity because the curve between all the parameters and popularity is mostly dependent on them. To find the numerical value of the threshold limit beyond which a dataset is termed popular, a graph is plotted. This graph is plotted on the log scale so that all values can be plotted within the region represented by the graph.

After calculating the threshold values, transformation into a classification problem is done. Now, a rolling forecast is performed. This helps us to predict binary popularity values for each week. Each week's data is added to the existing data and a new model is created. This follows the notion, more data leads to better data analysis. Prediction can be done in various ways following implementation of several machine learning algorithms, mainly, Naive Bayes, Stochastic Gradient Descent and Random Forest. Their models are then combined into an ensemble to check which algorithm offers the best true positive, true negative, false positive or false negative value.

This project also includes plotting the results obtained against the time scale to get a notion of how accuracy scores change with each week. These include sensitivity, specificity, precision, and recall and fallout rate against time scale.

Table of Contents

1 Introduction

Intext is present an evaluation of Apache Spark for streamlining predictive models which use information from CMS data- services. The models are used to predict which datasets will become popular over time. This will help to replicate the datasets that are most heavily accessed, which will improve the efficiency of physics analysis in CMS. Analysing this data leads to useful information about the physical processes. Reproducibility is necessary so that any process can be simulated just like the original in software at different times. Besides this, some process may be researched more by users and hence it needs to be made easily accessible to all the users. Accessibility is possible using replicas of data at some specified places. The user can then obtain the data from the nearest replica. Creating numerous replicas of every dataset is not feasible because the all datasets are vast. Maintaining and mirroring such vast datasets is an expensive job hence, the need of predicting which dataset might become popular is necessary.

The aim is to solve a classification problem which finds if a dataset will become popular or not. It includes calculating binary values of popular (1 / TRUE) or unpopular (0 / FALSE) of the 2013, 2014 and the currently producing 2015 data.

The process of predicting popularity of datasets, hence which datasets replicas should be created, is a machine learning problem. Hence, the aim of the problem is to use machine learning to predict which dataset will become popular and when.

After finding which dataset is popular, it must be found out which machine learning algorithm suits the procedure the best. For this purpose three algorithms are employed, namely, Naive Bayes, stochastic gradient descent and random forest. Additionally, these models are combined into an ensemble to check which algorithm offers the best true positive, true negative, false positive or false negative value.

All this will lead to analysis of Apache Spark as a framework for parallel, real time processing of the distributed data that is abundantly available in CMS.

2 Goals

1. Define the popularity index.
    a. The target goal is to find the popular datasets, but what is a popular dataset?
    b. Which parameters define popularity of the dataset?
2. Apply machine learning algorithms for prediction of popularity.
3. Find the hardware configuration that gives the relatively best run of a machine learning algorithm.
4. Benchmarking of algorithms and hardware configuration that they support.
5. Generalize machine learning algorithms so that they can streamline CMS data without much data formatting. Create a pipeline or command line program that performs complete analysis with just the minimum configuration changes.
6. Develop ensemble of used algorithms.
7. Evaluate Apache Spark as an alternate framework for the complete analysis procedure.
8. Plot graphs displaying accuracy against time scale.

3 Dataset

A dataset describes a process completely. A process is any interaction taking place in the LHC. An example would be proton-proton collision in the LHC, taking place at a single vertex. A single process may be composed of many collisions taking place at the same vertex. The weekly collection of data is uniquely represented by the name of the dataset. It describes which weeks data it contains, 20140101 - 20140107 will describe the first week of year 2014.

A datasets format is defined by three distinct parts process/software/tier, where:

- **process**: is a process type, examples include Higgs Process, TopQuark, ttbq
- **software**: is the software used and its version/release number. It is important for reproducing results.
- **tier**: defines the tier. There are many different tiers but the most important ones are: {AOD, AODSIM, MINIAOD, MINIAODSIM, USER}

4 Dataframe

Dataframe is the input file in comma separated values format. This is the file on which the machine learning algorithms are run upon. It contains many datasets, and each dataset is uniquely represented by one row in the dataframe. The access to files from all CMS analysis centres is recorded in a central database (PopularityDB). This data is pulled out in week-sized chunks for analysis, and stored in a dataframe as the input to the ML algorithms. Currently used popularity metrics as reported by the PopularityDB are:

- naccess - number of accesses to a dataset
- totcpu - number of CPU hours utilized to access a dataset
- nusers - number of users times days when the dataset was accessed

5 Data Acquisition

CMS produces nearly 300 Mb of data per second. That is a lot of data that might need to be processed. However, only a part of this data requires analysis. The important data that requires processing is stored for future reference. This data consists of relatively important information

that is new and has not been obtained before. The data from one week is recorded as a dataframe. Each data frame name is unique and is recorded with the week in which it is produced in. One dataframe is defined by its attributes. These are approximately 84 features. After selecting relevant features, the feature list comes down to 26. Relevant features are selected by removing sparse columns. The final list of the 26 features comes out to be: id, cpu, creator, dataset, dbs, dtype, era, naccess, nblk, nevt, nfiles, nlumis, nrel, nsites, nusers, parent, primds, proc_evts, naccess, rnusers, rtotcpu, size, tier, totcpu and wct. Feature extraction is essential because not all the columns contribute towards making a dataset popular. Features remain consistent irrespective of frames. Using these features a dataframe can be identified uniquely. It is necessary to note that for this analysis metadata of the CMS data is being used.

6 Procedure

There are three main algorithms that are to be implemented on Spark platform and checked against other implementations on the Hadoop Distributed File System and Python packages like pandas and scikit-learn.

6.1 Transforming data to classification problem

Based on the values of naccess and nusers, a classification problem can be developed. The target column gets a value of 1 (popular) when naccess is greater than 10 and nusers is greater than 5.

```
def convert(def):

        threshold_naccess = 10
        threshold_nusers = 5
        return df['naccess'] > threshold_naccess and df['nusers'] >
        threshold_nusers
```

6.2 Machine Learning

Machine learning is a branch of artificial intelligence which is concerned with the construction and study of systems that are able to learn from data. Hence, basically it is a type of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.

An abstract definition of machine learning is often expressed as:

A Machine (Computer Program) Learns with experience E for Some Task T and Performance Measure P, if P keeps on increasing with increase in E.

The types of machine learning are:

1. Supervised Learning

Supervised learning methods provide labels in the training data set and the computer has to learn from these examples. It needs to create its own hypothesis based on the training data set and test it on the test data set. In supervised learning, the training patterns are given as input and the corresponding correct outputs are also available.

2. Unsupervised Learning

Learning happens automatically and the structures hidden in the data are automatically recognized by the system. All the interesting and/or significant patterns in the data are extracted without any feedback as to what is right and necessary to the calculation.

3. Semi-Supervised Learning

This is not based on any major distinction, but it actually depends on the type and size of the data. Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training. It consists of a small amount of labeled data with a large amount of unlabeled data.

The machine learning models created for each additional week are similar because the data produced in the weeks is similar. This is advantageous for a machine learning problem because it means that if the data has some similar patterns then a model trained on one year can be used to predict another week. Similar patterns are repeated throughout the year and this fact is advantageous for using machine learning algorithms for prediction and analysis.

6.2.1 Naïve Bayes

It is a probabilistic classifier based on the Bayes Theorem. The Bayes Theorem states that given a class variable as ‘y’ and dependent features ranging from x1 to xn,

$P(y| x1, x2….., xn) = P(y) . P(x1, ….xn | y) / P(x1,….,xn)$. It is based on the following naïve assumption that $P(xi | y , x1,….xn) = P(xi | y)$. Calculating this value over all values of ‘I’ the following is attained: $P(y | x1, ….xn) = P(y) \Sigma i=1, n P(xi | y) / P(x1,…,xn)$ From this the following classification rule is obtained: $Y = \arg\max P(y) \Sigma P(xi | y)$.

6.2.2 Random Forest

It was originally developed by Leo Breiman and Adele Cutler. It operates by constructing a various decision trees while working with the training data set and outputting the class that is the mode of the classes output by individual trees. The random forest classifier is the easiest to apply and also obtains high accuracy from the cross validation score.

6.2.3 Stochastic Gradient Descent

Stochastic Gradient Descent like other gradient descent problems works towards optimizing a function. This objective function is a sum of different differentiable functions. It must be ensured that the route taken to optimize the objective function does not fall into a local maxima or minima because then the function would not be able to reach the global minima and maxima.

6.2.4 Bagging and Boosting

Bagging and boosting approach for ensembles is being applied. The computer program derives a rule of thumb or finds any structure in the data. Then this rule is applied to the subsets of the training data set. With each subset, some modifications are performed on the rule and a different rule is created. Then this new rule is applied to the next subset. This process is repeated for all subsets. In the ensemble both bagging and boosting is applied.

6.3 Rolling Forecast

Rolling forecasts are prediction machine learning techniques which built upon a pre-existing model each time a new dataset is added. These are used to make future predictions based on past and present analysis trends. Incremental learning is suited for this problem as each new week's data is produced it has to be added to the model. This is incorporated in the problem because each week's data is added onto the existing data. A new model is generated; which has combined features from the previous year's model and the learning's from the recently added week. This leads to the formation of a new and enhanced model. The window of the model can either be kept consistent or can be perpetually increasing with each new week. In this problem, the window size increases by one week. It starts with 52 weeks (2013), hence of one year. Then one week is added to it, adding to the existing model.

The procedure is described in detail in the diagram below. 2013.csv is the complete data of 2013, on this a machine learning algorithm is applied which helps to generate a model. Predictions are done for a week of 2014 and then cross validated against the calculated target values. The learnings that are obtained from cross validation are applied to the next model. Hence with every week the model learns more and performs better for the upcoming weeks. The figures have in depth description below:

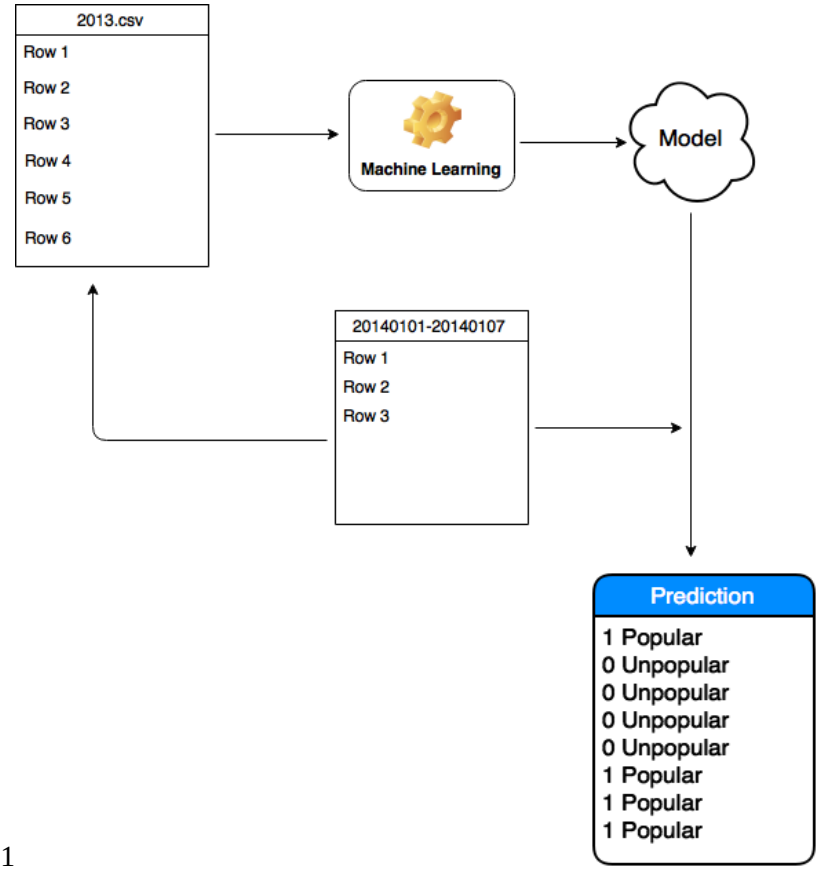

2013.csv
Row 1
Row 2
Row 3
Row 4
Row 5
Row 6
Machine Learning
Model
20140101-20140107
Row 1
Row 2
Row 3
Prediction
1 Popular
0 Unpopular
0 Unpopular
0 Unpopular
0 Unpopular
1 Popular
1 Popular
1 Popular

Fig 6.1

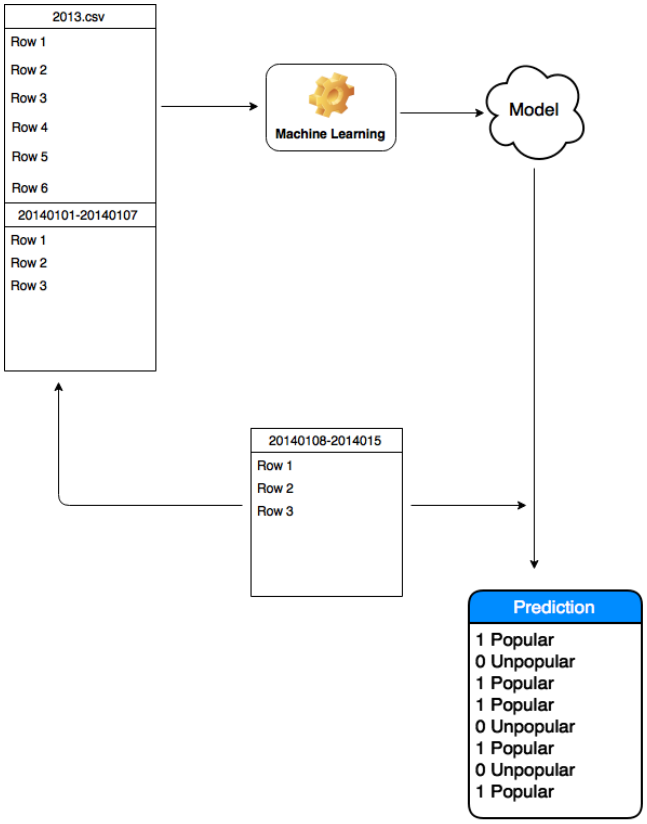


Fig 6.2

Figure 6.2 explains the incremental growth taking place in a rolling forecast. Week 20140107 is added to the complete 2013 dataset. The model is generated by applying machine learning algorithms basically, Naive Bayes, Stochastic Gradient Descent, and Random Forest. Then the following week, which is 20140108, is predicted and cross validated with the target values obtained by computing the naccess and nusers. Based on the cross-validation, some learning is done whcih is updated to the model and this process continues each week.

6.4 Popularity metric

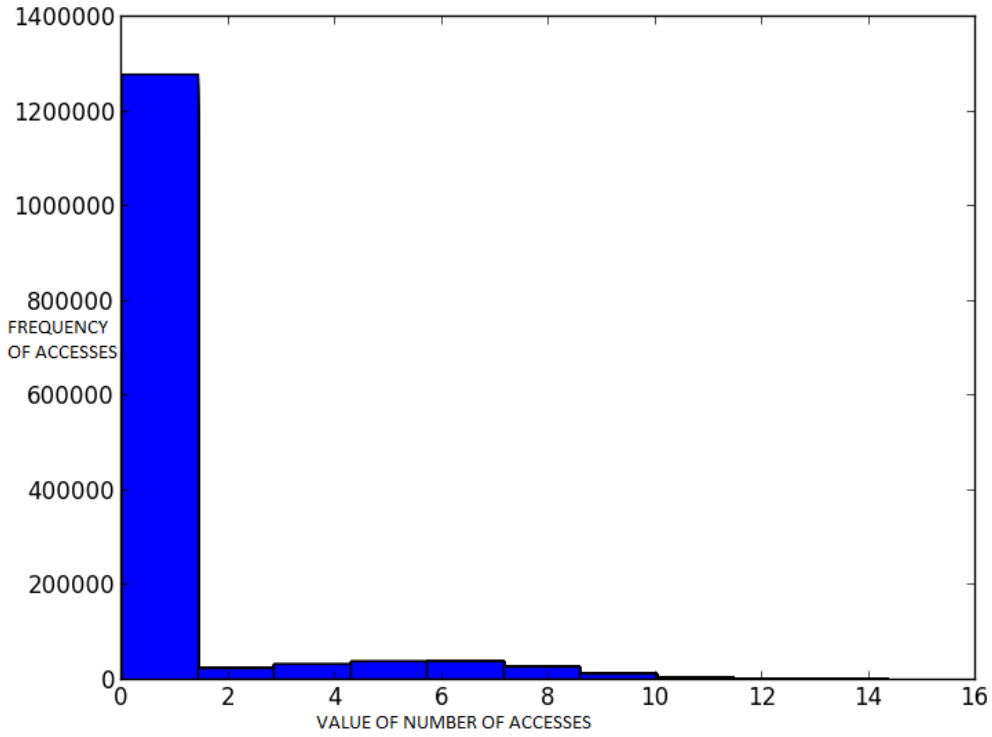
Define numeric values for naccess, nusers, totcpu for popularity metrics. Popularity is defined as which datasets are used for research most of the time. These are considered “popular” because they are demanded by many agencies and hence must be replicated at various data centres around the world. Popularity can be found by making plots for naccess, nusers, totcpu distributions. These explain the distribution of naccess, nusers and totcpu with the remaining parameters. Plots are made on the log scale so that all values can be plotted on a small scale Y axis. Plots are superimposed to know the correlation between different parameters. After calculating the value of the cut, apply that cut to facilitate conversion to a classification problem. Cut will help in deciding which is popular and which not popular category is. Cuts need to be applied to naccess,

nusers, totcpu. Cuts act as threshold values and indicate a set is popular if its value is above the threshold cut value and not popular otherwise.

Conclusion for cuts:

1. threshold_nusers = 5
2. threshold_naccess = 10

Both these cuts are applied together. In the code they are implemented with a logical AND. The plot below describe the popularity metric "naccess"



6.5 Spark

A pipeline of Naive Bayes, Stochastic Gradient Descent and Random Forest is applied on Apache Spark platform, using pyspark, the Python binding of Apache Spark. Spark requires a cluster manager and a distributed storage system. For cluster management, Mesos is being used while for interfacing distributed storage; Spark is interfaced with Hadoop Distributed File System (HDFS). MLlib is a distributed machine learning framework on top of Spark that, because of the distributed memory-based Spark architecture, is nine times as fast as the Hadoop disk-based version of Apache Mahout. Apache Spark is a fast and general processing engine compatible with Hadoop data. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can

process data in HDFS. It is designed to perform batch processing (training one year's data) and processing for new workloads like streaming (live prediction of each week), interactive queries, and machine learning.

Data is being read from the csv file into a pyspark dataframe. In order for parallelization to occur the algorithms must take the input of RDD format. To enable this, the target values and dataframe are being converted to RDD using LabeledPoint functions. Apache Spark allows parallel analysis on distributed data and this offers a much less total execution time for the whole algorithm or ensemble to execute.

This project allowed testing of several features and functionalities of Spark. These include streaming data and Spark Streaming, because each week's data is being added to the entire dataset. The cache persistence functionalities of Spark allow an RDD once created to persist in memory and for the next iteration of the algorithm the new week can be added straight to the RDD rather than undergoing the complete procedure once again. Machine Learning is performed using the MLlib which has defined Random Forest, Naïve Bayes and Stochastic Gradient Descent.

6.6 Ensembles

Machine learning ensemble methods use multiple learning algorithms, like Naive Bayes, Random Forest, and Stochastic Gradient Descent to obtain better predictive performance that could be obtained from any of the constituent learning algorithms. This ensemble refers only to a concrete finite set of alternative models and allows a flexible structure that enhances user's manipulation of the chosen features and parameters as well as to choose amongst different alternatives.

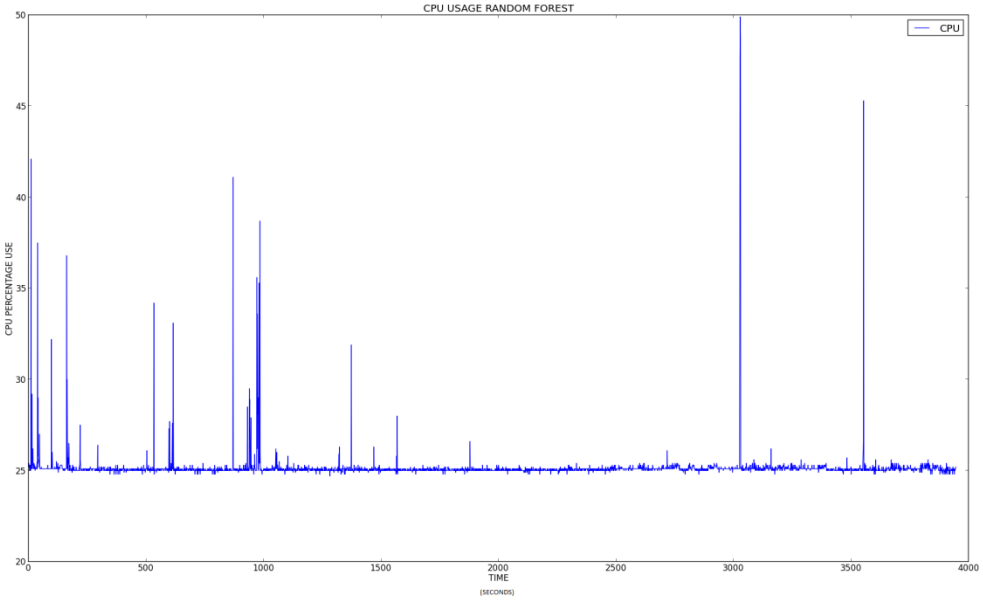
Bagging and boosting approach for ensembles is being applied. The computer program derives a rule of thumb or finds any structure in the data. Then this rule is applied to the subsets of the training data set. With each subset, some modifications are performed on the rule and a different rule is created. Then this new rule is applied to the next subset. This process is repeated for all subsets. AdaBoost is a great technique as it combines bagging and boosting. Hence, all the advantages of a random forest with gradient boosting are possible in a bagging and boosting technique.

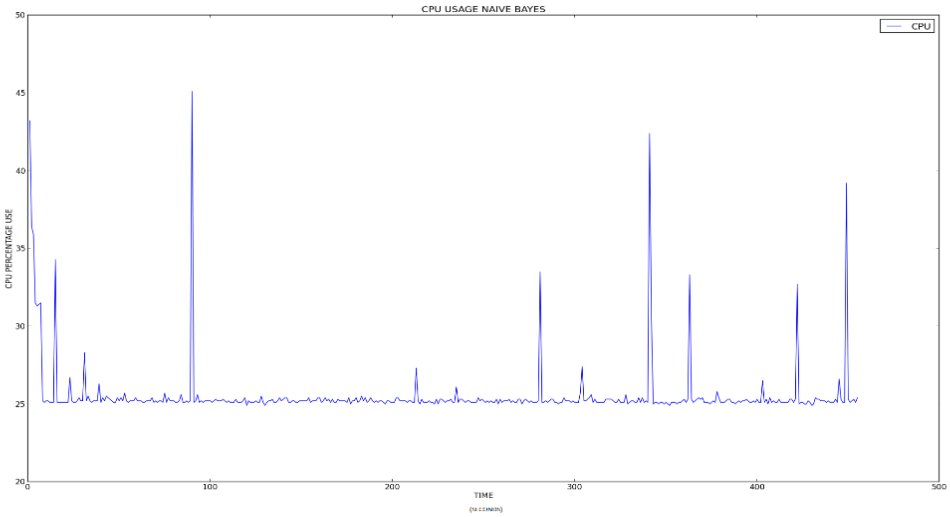
7 Performance Metrics

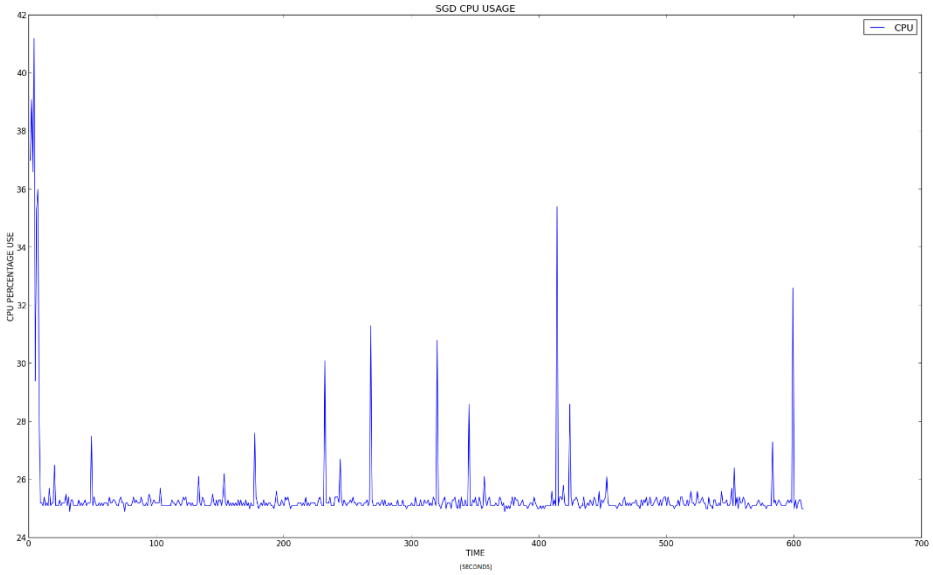
The performance metrics describe how good one algorithm is when it comes to the time for training the model and how accurate the predictions are.

7.1 CPU Time

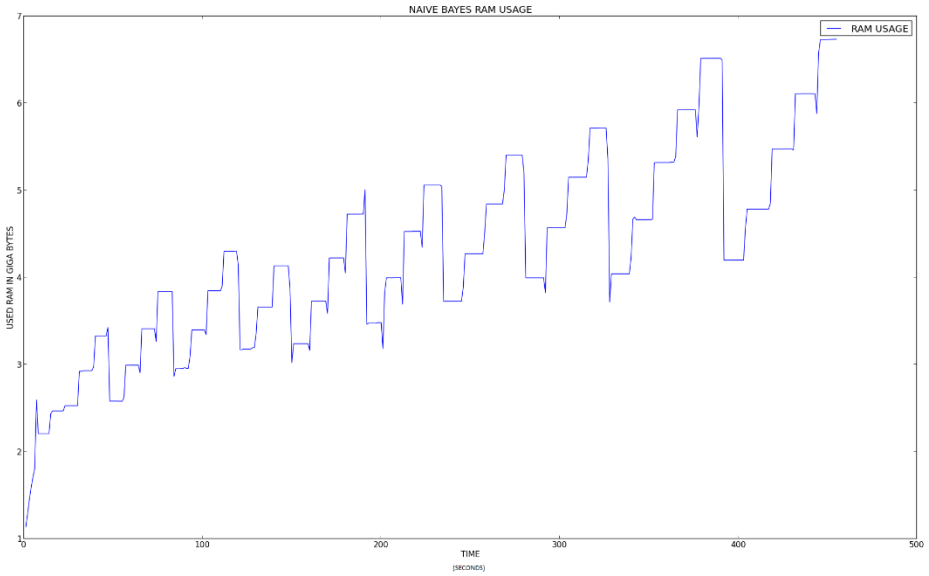
Using psutil (python system and process utilities) and the wide range of functions that it provides, the cpu time utilized can be used to ascertain how long one process might have taken. Psutil is a cross-platform Python library for retrieving information on running processes and system utilization (CPU, memory, disks, and network). It is also used for system monitoring, profiling and limiting process resources and management of running processes. Approximately 25% of the CPU is always being used by the algorithms.

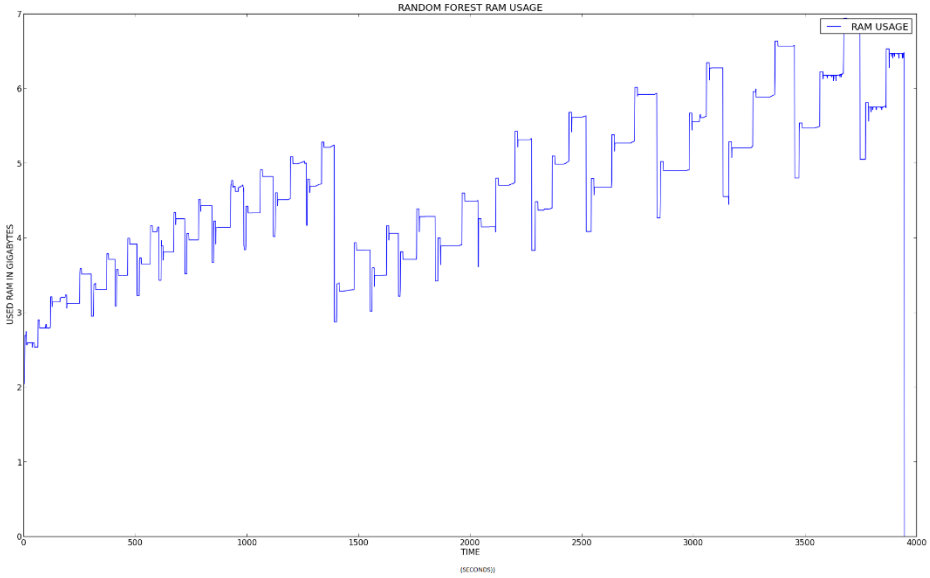


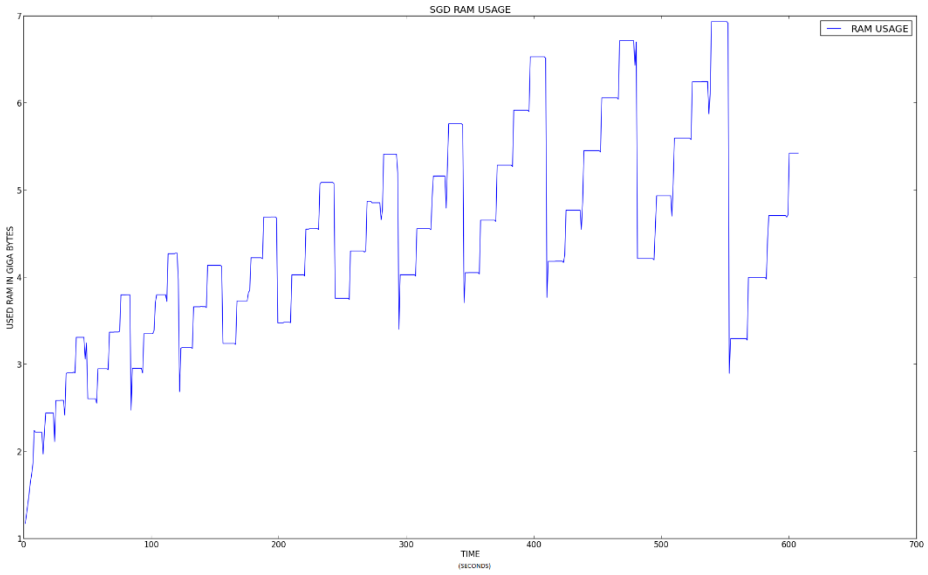




It can be gauged from the RAM usage graphs which depict the incremental learning how the memory is being used. With each iteration more RAM is utilized because the amount of data being processed is continually increased. The graph does not indicate memory leak, instead it indicates the rolling approach, hence the incremental learning part.


NAIVE BAYES RAM USAGE
RAM USAGE
USED RAM IN GIGA BYTES
TIME
(SECONDS)
0
100
200
300
400
500
1
2
3
4
5
6
7


RANDOM FOREST RAM USAGE
RAM USAGE
USED RAM IN GIGABYTES
TIME
(SECONDS))
0
500
1000
1500
2000
2500
3000
3500
4000
1
2
3
4
5
6
7



7.2 Accuracy

Accuracy is measured in terms of true positive, true negative, false negative and false positives. Combining these values, different parameters such as precision, recall, F1, true negative rate, true positive rate and fall-out rate are obtained. These offer a greater degree of insight along with the plots. Systems with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labelled correctly. Underfitting and overfitting need to be checked using cross-validation. Underfitting is when the features are not learnt properly because the linear function learning the features is not sufficient to fit all the training samples. Overfitting learns the noise in the data and considered noise as feature as well. Overfitting can be approximated as a high order polynomial whereas underfitting would resemble a low degree polynomial such as a linear function. To gauge the accuracy of the model there are several methods. Using the cross validation scoring method as:

from sklearn.metrics import accuracy_score

score(clf, test, target_test)

print score.max()

Alternatively, using the confusion matrix we can infer the number of true positives, false negatives, true negatives and false positives. These are used to calculate the true positive rate or recall and accuracy.

True Positive Rate (TPR) =TP / P = TP /(TP+FN)

Accuracy (ACC) = (TP + TN) / (P + N)

1. **True Positive**
    a. A correct hit
    b. The number of datasets that were predicted as popular and were popular.
2. **True Negative**
    a. A correct rejection
    b. The number of datasets that were predicted as not popular and were not popular.
3. **False Positive**
    a. A false alarm
    b. The number of datasets that were wrongly predicted as popular, but were not popular.
4. **False Negative**
    a. A miss
    b. The number of datasets that were wrongly predicted as not popular, but were popular.
5. **Precision**
    a. Precision is the ratio TP / (TP + FP) where TP is the number of true positives and FP the number of false positives.
    b. Intuitively precision is the ability of the classifier to not wrongly label an item
    c. Best value is 1 and the worst value is 0.
    d. Precision of the positive class in binary classification or weighted average of the precision of each class for the multiclass task.
6. **Confusion matrix**
    a. Accuracy is not a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced. This leads to the use of a confusion matrix. A confusion matrix is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives. This allows more detailed analysis than mere proportion of correct guesses (accuracy).
    b. Based on the confusion matrix, it can be seem that some algorithms have higher sensitivity to true positives, hence will be able to detect true positives more. This sensitivity depends on the development of the algorithm and can exist for either true positive, true negative, false positive or false negative. The sensitivity of an algorithm depends on which region is it biased towards.
    c. The aim of this project was to check if Apache Spark can serve as a framework for and not for optimizing for time and space constraints. As a future goal, it is necessary that optimization be considered as an important factor.

RF	**1**	**0**
1	570374	0
0	2	6438
NB	**1**	**0**
1	568415	6
0	12783	6432
SGD	**1**	**0**
1	568040	3351
0	2336	3042

8 Conclusion

Apache Spark is a framework providing speedy and parallel processing of distributed data in real time. It provides powerful cache and persistence capabilities. Because of these features Apache Spark is applied to data analytics problems. Components of Spark like Spark Streaming and MLlib (Spark native machine learning library) make analysis possible, which is useful for CMS data analysis. Spark provides a solution to the many features that are required for CMS data analyses, like streaming and distributed data.

9 References

1. Programming Collective Intelligence, Toby Segaran, O' Reilly
2. Advanced Analytics with Spark, Patterns for Learning from Data at Scale, Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills, O' Reilly
3. Learning Spark, Lightning-Fast Big Data Analysis, Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia, O'Reilly
4. Wikipedia, http://www.wikipedia.com
5. Apache Spark Documentation http://spark.apache.org/
6. Apache Spark Programming Guide http://spark.apache.org/docs/latest/programming-guide.html
7. Scikit Documentation, http://scikit-learn.org/stable/
8. Pandas Documentation, http://**pandas**.pydata.org/