ISBN 978 -987-633-574-4

A Comparative Study of the Performance of the Classification Algorithms of the Apache Spark ML Library

Genaro Camele[1,2], Waldo Hasperué[1,3], Ronchetti Franco[1,4] and Quiroga Facundo Manuel[1]

[1] Instituto de investigación en informática (III-LIDI), Facultad de informática - Universidad Nacional de La Plata
[2] Becario UNLP
[3] Investigador asociado - Comisión de Investigaciones Científicas (CIC-PBA)
[4] Investigador asistente - Comisión de Investigaciones Científicas (CIC-PBA)
{gcamele,whasperue}@lidi.info.unlp.edu.ar
http://weblidi.info.unlp.edu.ar/wp/

Abstract. Classification algorithms are widely used in several areas: finance, education, security, medicine, and more. Another use of these algorithms is to support feature extraction techniques. These techniques use classification algorithms to determine the best subset of attributes that support an acceptable prediction. Currently, a large amount of data is being collected and, as a result, databases are becoming increasingly larger and distributed processing becomes a necessity. In this sense, Spark, and in particular its Spark ML library, is one of the most widely used frameworks for performing classification tasks in large databases. Given that some feature extraction techniques need to execute a classification algorithm a significant number of times, with a different subset of attributes in each run, the performance of these algorithms should be known beforehand so that the overall feature extraction process is carried out in the shortest possible time. In this work, we carry out a comparative study of four Spark ML classification algorithms, measuring predictive power and execution times as a function of the number of attributes in the training dataset.

Keywords: Big Data, Machine Learning, Classification Models, Apache Spark, Spark ML

1 Introduction

As technology becomes faster and more accessible, it is possible to collect much more information; however, that comes with increasing database volume. This phenomenon can be observed in various research areas. Astronomy, marketing, and social, economic, biological and medical sciences, among others, now have the possibility of storing and analyzing large volumes of information [9] [20] [12]

[8] [6]. This growing data size gives rise to the need for algorithms that allow extracting useful information in a reasonable time.

Among the tasks carried out in data mining, classification is one of the most commonly used; there are various works where it is used for some purpose, from predicting market behavior [7], to image classification [16] and the detection of pathologies in functional medicine [11].

Training a classification model on large volumes of data is a computationally expensive task, and training time is critical in techniques that require performing this task multiple times. Attribute or feature selection techniques are some examples. These techniques consist of selecting a subset of attributes from the available datasets to obtain with this subset the same predictive power that is available when using the entire database.

Selecting a subset of attributes that obtain a predictive power similar to that achieved with the full set of attributes is not a trivial task. There are many techniques that allow selecting an optimal subset of attributes [10] [5] [3] [18] [22]. Most of the techniques for this task consist of an iterative process during which several subsets of attributes are ranked and the best of them is selected as the final result. To measure the predictive power of a subset of attributes, a classification algorithm must be executed with that subset of attributes. Therefore, feature selection techniques must run these ranking algorithms a significant number of times until the optimal attribute subset is reached. This is why the execution time required by a classification algorithm is a critical factor for feature selection techniques, and it becomes increasingly critical as the number of attributes in the database increases.

To process large volumes of data, tools are available that allow distributing computation tasks among different nodes in a cluster of computers, so that the workload is balanced and processing times decrease. In this regard, tools such as Apache Hadoop or Apache Spark allow algorithms to be run in a distributed paradigm, abstracting the developer from all the inconveniences that this entails, such as synchronization, data transfer, fault tolerance, and so forth. In particular, Apache Spark has the Spark ML library, which contains the implementation of several machine learning algorithms such as neural networks, decision trees, Random Forest, regressions, SVM, and others. It also provides the functionalities of transformation, filtering and other utilities to pre-process the information that will be used to train the models.

In this work, a comparison between four of the classification algorithms implemented in Spark ML is presented: Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB) and MultiLayer Perceptron (MLP). Different predictive metrics from the models are measured and compared along with the running time required for training. In the experiments carried out, an ovarian cancer classification database [14] with 15154 attributes was used. The different experiments carried out have variations in the number of attributes selected to measure classification algorithm performance based on the number of attributes in the database used for training.

The remaining sections of this article are organized as follows: in Section 2, some similar publications that evaluate some of the features of Spark or Spark ML are mentioned. In Section 3, the experiments carried out are described, and in Section 4 the results obtained are presented. Finally, in Section 5, conclusions and possible future works are presented.

2 State of the Art

Several studies have been carried out comparing the performance of the classification algorithms in MLlib and Spark ML libraries. These investigations focus on the number of rows in the databases or on specific problems. No studies have been found that carry out a thorough study on the performance of classification algorithms based on the number of columns in the database, which is the objective proposed in this work.

In [15] a comparison of the NB, RF, Decision Tree (DT), SVM and Logistic Regression (LR) classifiers implemented in MLlib (version 1.6.2) is carried out. The only performance metric evaluated is accuracy. These authors run trials with different sizes (number of rows) of the training data set. In addition, since the classification is carried out with Amazon product reviews, they also evaluate the performance of the classifiers by varying the number of n-grams extracted from each review. The conclusion of the work in general lines is that logistic regression is the classifier that achieves the best results.

The experiments carried out in [21] do not focus on the characteristics of the database, but rather measure the execution time of the algorithms with 128 different configurations of a 16-node Spark cluster. They evaluate LR, SVM, RF and Gradient Boosted Trees (GBT) for Decision Trees under different settings for their hyperparameters. They use variants of the ”Higgs” database from the UCI repository. The conclusions of the work detail the RAM and node vCPUs configuration that achieves the best execution time with each algorithm, but the predictive power of the models obtained is not evaluated.

In [13], the authors carry out experiments to detect which classification algorithm yields the best classification results with databases of patients suffering from a mental illness. The algorithms studied are LR, DT, RF and MLP using several sets of values for their respective hyperparameters. F1-measure, accuracy, recall, and precision are measured using two different datasets. The results obtained show that the RF algorithm is the one that yields the best classification models for the two databases analyzed.

Very similar to the previous case, but with the aim of predicting stock price fluctuations in the stock market, the authors of [19] study NB, RF, DT and LR measuring accuracy, ROC curves and PR curves together with the execution time with various configurations of a Spark cluster where the number of worker nodes is varied. The database used contains information on the United States market for the last 20 years, with a total volume of information of 1.7 GB. They determined that RF and DT are the algorithms with the best predictive power.

As for computation time, NB was the fastest algorithm followed by DT, while RF and LR were the algorithms that needed the most time to obtain a model.

In [1], the authors analyze tweets to determine patients at risk of heart attacks. They use several hyperparameter settings from the DT, SVM, RF, and LR algorithms. Since the focus of the work is on the online detection of patients at risk, only the accuracy to get the best model used in production was measured. Two experiments were carried out, one with the entire database and another one with a database with a smaller number of columns as a result of selecting the best characteristics, resulting in half the attributes. DT and RF turned out to be the best models using the database with half the characteristics. In the case of the entire database, RF was the best model, followed by LR and SVM and lastly DT.

Recently, the authors of [17] measured the performance of LR, DT and SVM in a case study of intrusion detection in computer networks. The area under the ROC and PR curves was measured, as well as accuracy and training times. LR turned out to be the algorithm that obtained the best models when measuring the area under the ROC and PR curves, while DT was the one that obtained the models with the best accuracy. LR also turned out to be the fastest algorithm, followed by DT and lastly SVM.

In [2], the performance of the LR, RF, SVM and PM algorithms in natural language processing applied to posts about the COVID-19 pandemic on Twitter is compared. The variability of the experiments lies in the different number of records used to train the models. As regards precision, recall, F1-measure, accuracy and execution time, LR was the fastest algorithm and SVM the slowest. On average, SVM and LR had better models than RF and PM.

In general terms, it can be seen that RF and SVM are the algorithms that achieve the best models in terms of prognostic power, while RF and LR are the fastest ones. In most works, the performance of Spark stands out – it scales very well in terms of number of rows, thanks to the power offered by its internal structure (RDD - Resilient Distributed Datasets). Therefore it is interesting to see how Spark performs when the parameter that is modified is the number of columns in the dataset, as is discussed below.

3 Experiment

As previously mentioned, the objective of this work is to evaluate the performance, in terms of execution time, of four classification algorithms of the Spark ML library of Spark, Naïve Bayes, Random Forest, Support Vector Machine and MultiLayer Perceptron. In the experiments carried out, different numbers of columns in the dataset used for training were used, measuring performance as the number of columns in the dataset increased. In each experiment, the training time and accuracy, precision, recall and F1-measure metrics were measured. Since our goal was not finding the best model by tuning the corresponding hyperparameters of the classification algorithms, the default values of the library itself were used, with the exception of the multilayer perceptron whose structure

ISBN 978 -987-633-574-4

was defined as two 4- and 5-neuron hidden layers. We used predictive power metrics with the simple purpose of comparing one model against another, being aware that neither of them might be the best model to allows solve the real problem.

All the experiments were carried out in a cluster made up of a single master node and three worker nodes. All four nodes had Ubuntu 20.04 LTS, an Intel(R) Core(TM) i3-4160 CPU @ 3.60GHz, and 8GB of RAM. As regards the software, the Hadoop and Spark versions used were 3.2.2 and 3.1.1, respectively. Spark ML version 3.1.1 was used.

The database used corresponds to expression data of 15,154 genes in 253 patients [14]. The inference class for this dataset corresponds to *Normal* if the patient does not present evidence of an ovarian tumor, or *Cancer* if the presence of the tumor has been detected. Each gene corresponds to a column in the dataset. Experiments were carried out with different numbers of attributes: 10, 50, 100, 500, 750, 1000, 2000, 3000, 4000, 5000 and 10000. For each of the experiments, attributes were selected at random, since as already mentioned, our goal was not obtaining the best model to solve the problem of tumor detection, but rather to measure execution time with different numbers of dataset features.

To avoid any bias, a five-fold cross-validation step was performed. To obtain these folds, stratisfied sampling [4] was used. Additionally, the entire process of randomly selecting attributes, dividing into folds, training and evaluating the models was performed 30 times with each classification algorithm.

The Spark class CrossValidator[1] was used to carry out the measurements. However, since this class only allows evaluating models with a single metric at a time, a custom cross-validation executor was developed. This executor ensures that the four analyzed algorithms share the same random sample of characteristics extracted from the dataset, in addition to the same samples generated in each fold of the cross-validation.

The algorithm used is detailed below and the full code, along with the evaluated dataset, can be found in a public Github repository [2].

Pseudocode for the custom cross-validation executor

```
program StratisfiedCrossValidator
    parameters
        num_of_folds, models, metrics, spark_df
    begin
        for k in 1 .. num_of_folds do
            train, validation = generate_stratisfied_subsets(spark_df)

            for model in models do
                start_time = get_current_time()
                trained_model = train_model(model, train)
                store_training_time(model, start_time)
```

[1] https://spark.apache.org/docs/3.1.1/api/scala/org/apache/spark/ml/tuning/CrossValidator.html
[2] https://github.com/midusi/classification-models-spark

```
                for metric in metrics do
                    store_metric_for_model(model, metric, validation)
                end
            end
        end

        {Returns, for each model, the best result obtained for each metric
        and the training time in all folds}
        return get_best_cross_results()
    end
end
```

Pseudocode for the main program

```
program ClassificationModelComparison
    {The previous definition for the models and metrics variables is assumed};
    df = read_local_ovarian_dataframe()
    ncols = [10, 50, 100, 500, 750, 1000, 2000, 3000, 4000, 5000, 10000]
    for n_features in ncols do
        x_and_y = extract_random_features(df, n_features)
        spark_df = convert_spark_dataframes(x_and_y)

        all_results = []
        for i in 1..30 do
            result = StratisfiedCrossValidator(spark_df, models, metrics)
            store_result(result)
        end

        {Reports the 30 separate values for each model and each metric,
        and training time}
        report_results(all_results)
    end
end
```

4 Results

Figure 1 shows the average and standard deviation of the execution times for each algorithm for the different datasets used. As it can be seen, the SVM and MLP algorithms require the longest execution time as the number of attributes used for training increases. The RF algorithm also shows increased execution time in experiments that include thousands of attributes, but it is approximately 10 times smaller than SVM and MLP. Even though this cannot be seen in the figure, Naïve Bayes also has an increase in execution time, reaching 17 seconds for the dataset with 10,000 attributes.

Figure 2 shows the average and standard deviation of the 30 separate runs for the accuracy (figure 2a), precision (figure 2b), recall (figure 2c) and F1-measure (figure 2d) metrics achieved by each of the algorithms. It can be seen that NB was the algorithm with the worst models, followed by RF, which achieved very good results in experiments with large numbers of attributes. SVM and MLP are the algorithms that best model datasets with fewer attributes. In fact, the results obtained by both models in the datasets with 10, 50 and 100 attributes are very similar to each other. In datasets with a larger number of attributes, SVM begins to differ from MLP and, although RF improves the metrics as the number of attributes increases, it can be seen that the models obtained with SVM are the best in all experiments.

The values for the four metrics that were obtained with the MLP executions using 500 attributes are intriguing. Even though a detailed study of the case was not done, we believe that these values are the product of the 500 randomly selected attributes. Apparently, those 500 attributes make up a scenario where the MLP algorithm finds local minima from which it cannot escape. Out of the 30 runs, it achieved accuracy values higher than 0.96, only in eight runs, the remaining 22 are below 0.91.

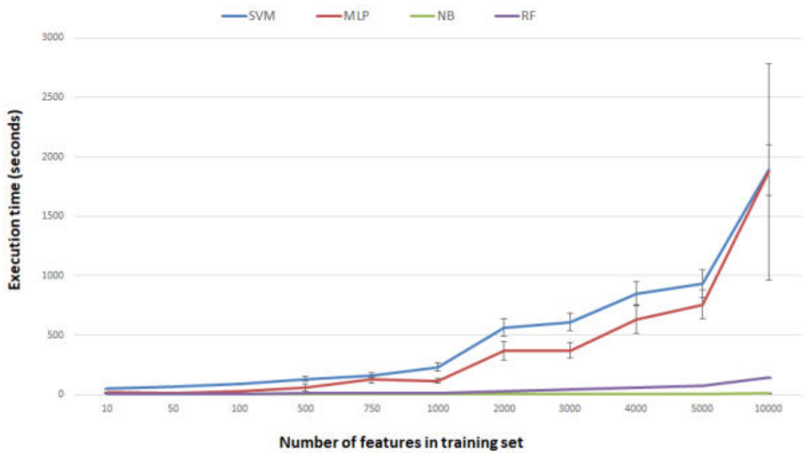
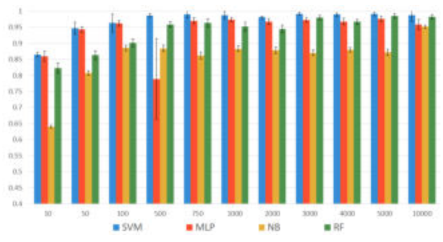


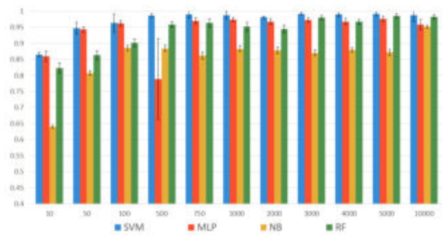
Fig. 1: Average and standard deviation of the execution times of the four algorithms studied for the different subsets of attributes used.

5 Conclusions and future work

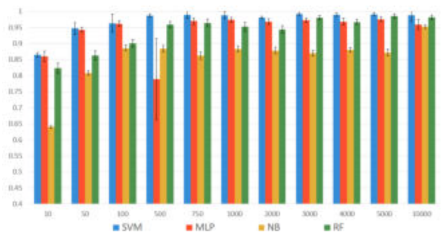
In this work, execution time, accuracy, precision, recall and F1-measure of four Spark ML algorithms were measured by varying the number of attributes in the training dataset. The results obtained show that the algorithms that require the



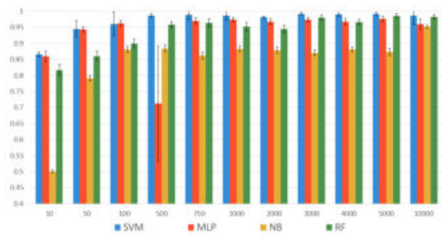
(a) Accuracy



(b) Precision



(c) Recall



(d) F1-Score.

Fig. 2: Average and standard deviation of the accuracy, precision, recall and F1-measure metrics of the four algorithms studied for the different subsets of attributes used.

most computing time are SVM and MLP. However, in experiments with a low number of attributes, these two algorithms obtained the best models. RF turned out to be the algorithm that required the shortest execution time to achieve models with a high prediction rate.

In attribute selection algorithms, where a classification algorithm has to be run hundreds or thousands of times with different numbers of attributes, the dichotomy presented in this work opens the door to an interesting challenge, especially considering that, when selecting a subset of attributes with high predictive power, it is generally expected that the resulting subset is the smallest possible.

The results obtained show the need for tradeoff between an algorithm with low execution time while achieving good models with few attributes. On the one hand, SVM yields very good models, regardless of the number of attributes in the dataset, but it requires a lot of computation time as the number of attributes to be analyzed grows. On the other hand, Random Forest requires little execution time, but to obtain models that resemble those obtained by SVM, it needs thousands of attributes.

The dataset used in the experiments has a significant number of attributes ($> 15{,}000$), but only a few samples (253). In the future, the same experiments will be carried out using datasets with more samples, and the performance of the classification algorithms in scenarios where the volume of data is greater will be studied. In these scenarios, measuring the volume of data transmitted by the

ISBN 978 -987-633-574-4

cluster during the execution of the algorithms is also of interest, since in the experiments carried out for this work, these times were negligible.

References

1. Hager Ahmed, Eman MG Younis, Abdeltawab Hendawi, and Abdelmgeid A Ali. Heart disease identification from patients' social posts, machine learning solution on spark. *Future Generation Computer Systems*, 111:714–722, 2020.
2. Wafaa S Albaldawi and Rafah M Almuttairi. Comparative study of classification algorithms to analyze and predict a twitter sentiment in apache spark. In *IOP Conference Series: Materials Science and Engineering*, volume 928, page 032045. IOP Publishing, 2020.
3. Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review. *Data Clustering*, pages 29–60, 2018.
4. Zdravko Botev and Ad Ridder. *Variance Reduction*, pages 1–6. American Cancer Society, 2017.
5. Jie Cai, Jiawei Luo, Shulin Wang, and Sheng Yang. Feature selection in machine learning: A new perspective. *Neurocomputing*, 300:70–79, 2018.
6. Ashutosh Kumar Dubey, Abhishek Kumar, and Rashmi Agrawal. An efficient aco-pso-based framework for data classification and preprocessing in big data. *Evolutionary Intelligence*, 14(2):909–922, 2021.
7. Uma Gurav and Nandini Sidnal. Predict stock market behavior: Role of machine learning algorithms. In *Intelligent Computing and Information and Communication*, pages 383–394. Springer, 2018.
8. Eslam M Hassib, Ali I El-Desouky, Labib M Labib, and El-Sayed M El-kenawy. Woa+ brnn: An imbalanced big data classification framework using whale optimization and deep neural network. *soft computing*, 24(8):5573–5592, 2020.
9. Gerardo Hernández, Erik Zamora, Humberto Sossa, Germán Téllez, and Federico Furlán. Hybrid neural networks for big data classification. *Neurocomputing*, 390:327–340, 2020.
10. Gang Kou, Pei Yang, Yi Peng, Feng Xiao, Yang Chen, and Fawaz E Alsaadi. Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. *Applied Soft Computing*, 86:105836, 2020.
11. Konstantina Kourou, Themis P Exarchos, Konstantinos P Exarchos, Michalis V Karamouzis, and Dimitrios I Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17, 2015.
12. SK Lakshmanaprabu, K Shankar, M Ilayaraja, Abdul Wahid Nasir, V Vijayakumar, and Naveen Chilamkurti. Random forest for big data classification in the internet of things using optimal features. *International journal of machine learning and cybernetics*, 10(10):2609–2618, 2019.
13. Dorin Moldovan, Marcel Antal, Claudia Pop, Adrian Olosutean, Tudor Cioara, Ionut Anghel, and Ioan Salomie. Spark-based classification algorithms for daily living activities. In *Computer Science On-line Conference*, pages 69–78. Springer, 2018.
14. Elnaz Pashaei and Nizamettin Aydin. Binary black hole algorithm for feature selection and classification on biological data. *Applied Soft Computing*, 56, 03 2017.

ISBN 978 -987-633-574-4

15. Tomas Pranckevičius and Virginijus Marcinkevičius. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221, 2017.
16. Franco Ronchetti, Facundo Quiroga, Genaro Camele, Waldo Hasperué, and Laura Lanzarini. Un estudio de la generalización en la clasificación de peatones. *Revista Cubana de Transformación Digital*, 2(1):33–45, 2021.
17. S Saravanan et al. Performance evaluation of classification algorithms in the design of apache spark based intrusion detection system. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 443–447. IEEE, 2020.
18. Saúl Solorio-Fernández, J Ariel Carrasco-Ochoa, and José Fco Martínez-Trinidad. A review of unsupervised feature selection methods. *Artificial Intelligence Review*, 53(2):907–948, 2020.
19. Jiang Xianya, Hai Mo, and Li Haifeng. Stock classification prediction based on spark. *Procedia Computer Science*, 162:243–250, 2019.
20. Wenchao Xing and Yilin Bei. Medical health big data classification based on knn classification algorithm. *IEEE Access*, 8:28808–28819, 2019.
21. Seyedfaraz Yasrobi, Jakayla Alston, Babak Yadranjiaghdam, and Nasseh Tabrizi. Performance analysis of sparks machine learning library. *Trans. MLDM*, 10(2):67–77, 2017.
22. Rizgar Zebari, Adnan Abdulazeez, Diyar Zeebaree, Dilovan Zebari, and Jwan Saeed. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70, 2020.