- ORIGINAL ARTICLE -

Statistical Analysis of the Performance of Four Apache Spark ML Algorithms

Análisis Estadístico del Rendimiento de Cuatro Algoritmos de Apache Spark ML

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

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

Abstract

Feature selection (FS) techniques generally require repeatedly training and evaluating models to assess the importance of each feature for a particular task. However, due to the increasing size of currently available databases, distributed processing has become a necessity for many tasks. In this context, the Apache Spark ML library is one of the most widely used libraries for performing classification and other tasks with large datasets. Therefore, knowing both the predictive performance and efficiency of its main algorithms before applying a FS technique is crucial to planning computations and saving time. In this work, a comparative study of four Spark ML classification algorithms is carried out, statistically measuring execution times and predictive power based on the number of attributes from a colon cancer database. Results were statistically analyzed, showing that, although Random Forest and Naïve Bayes are the algorithms with the shortest execution times, Support Vector Machine obtains models with the best predictive power. The study of the performance of these algorithms is interesting as they are applied in many different problems, such as classification of pathologies from epigenomic data, image classification, prediction of computer attacks in network security problems, among others.

Keywords: Big Data, Machine Learning, Classification Models, Apache Spark, Spark ML, Wilcoxon Test, Student's T Test

Resumen

Las técnicas de selección de características suelen requerir el entrenamiento y la evaluación repetida de modelos con el fin de evaluar la importancia de cada característica para una tarea concreta. Sin embargo, debido al aumento del tamaño de las bases de datos disponibles actualmente, el procesamiento distribuido se ha convertido en una necesidad para muchas tareas. En este contexto, la librería Apache Spark ML es una de las más utilizadas para realizar clasificación y otras tareas con grandes conjuntos de datos. Por ello, conocer tanto el rendimiento predictivo como la eficiencia de sus principales algoritmos antes de aplicar una técnica de selección de características es crucial para planificar los cálculos y ahorrar tiempo. En este trabajo se realiza un estudio comparativo de cuatro algoritmos de clasificación de Spark ML, midiendo estadísticamente los tiempos de ejecución y el poder predictivo en función del número de atributos de una base de datos de cáncer de colon. Los resultados fueron analizados estadísticamente, mostrando que, aunque Random Forest y Naïve Bayes son los algoritmos con menores tiempos de ejecución, Support Vector Machine obtiene modelos con el mejor poder predictivo. El estudio de la performance de estos algoritmos resulta interesante ya que los mismos son utilizados en problemas muy diversos, como por ejemplo, la clasificación de diferentes patologías a partir de datos epigenómicos, clasificación de imágenes, la predicción de ataques informáticos en problemas de seguridad en redes, entre otros.

Palabras claves: Big Data, aprendizaje automático, modelos de clasificación, Apache Spark, Spark ML, Test de Wilcoxon, Test T-Student

1 Introduction

The advancement in speed and accessibility of technology in recent years has allowed an increase in data collection, generating an increase in the volume of available databases. This phenomenon can be observed in various research areas such as astronomical social science, economic science, biological science and medical science, among others. These now have the possibility of storing and analyzing large volumes of information. For example, [1] employs several datasets in order to evaluate the ability of a neural network model to perform transfer learning, including both small and

large scale datasets. [2] use public health data repositories to test an improved K-Nearest Neighbours model that reduces the time needed for inference so that it is suitable for large volumes of data. [3] also use a large dataset consisting of records from 100,000 patients per day from Internet of Things sensors. They adapt neural network and random forest classification models for the Apache Hadoop distributed computing framework. [4] improve the area under the curve (AUC) metric by applying a novel dimensionality reduction technique based on Whale Optimization Algorithm before training the final classifier with large imbalanced datasets. As in the previous work, [5] also evaluates a metaheuristic, Particle Swarm Optimization, for dimensionality reduction with various datasets from the UCI public repository [1]..This growing data volume leads to a need for algorithms that allow extracting useful information in a reasonable time.

In data mining, classification is one of the most commonly used tasks; there are various works where it is used for some purpose, from predicting market behavior [6], to image classification [7] and the detection of pathologies in functional medicine [8].

Training a classification model using large volumes of data has a significant computationally cost, and training time can be critical in techniques that require performing this task multiple times. FS techniques are typical examples of this problem, since they involve selecting a subset of attributes from the available datasets to train and evaluate a model, in order to find the subset that leads to the best performance. Therefore, the already costly task of training a classifier using big data is compounded by the need to perform these task multiple times. Indeed, for a dataset with N features, a brute force FS technique can require evaluating the $2^N - 1$ possible combinations of features, which is computationally impossible for datasets with, for example, genomic data, which can include up to thousands of features.

Feature selection techniques selecting an optimal subset of attributes without exploring the full set of possibilities. For example, [9] evaluate ten different FS methods with nine different metrics for a text classification task using a Multi-Criteria Decision-Making framework. [10] discusses various evaluation metrics typically used for FS in supervised, unsupervised and semi-supervised problems. [11] performs a generalized study of FS techniques such as Filter, Wrapper and other hybrid approaches for clustering algorithms. [12] further reviews various FS techniques, including as well other unsupervised models. Finally, [13] analyses Feature Extraction techniques as well as FS methods for various well-known datasets, emphasizing wrapper techniques that require repeatedly evaluating a model.

Of the aforementioned techniques, most employ an iterative process during which several subsets of attributes are tested and the best of them is selected as the final result. With each subset of selected attributes, a classification algorithm is run to obtain a classifier model and measure its predictive power. Therefore, FS techniques have to run these classification algorithms a significant number of times to reach the optimal subset of attributes. As a result of this, the execution time required by the classification algorithm is a critical factor for FS techniques, and it grows in relevance as the number of attributes in the database increases.

Currently, there are tools that, by distributing computation between different nodes that make up a cluster of computers, allow processing large volumes of data. They balance the workload, reducing processing times. In this regard, tools such as Apache Hadoop or Apache Spark allow algorithms to be run in a distributed paradigm, abstracting the developer from the complexity that this entails using technologies such as intracluster 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, Support Vector Machines, and others. It also provides transformation and filtering functionalities, as well as other utilities to pre-process the information that will be used to train the models.

This work is an extension of [14], in which the necessary experiments were carried out to obtain two new classification metrics (sensitivity and AUC) using an ovarian cancer classification dataset [15] which contains 15,154 attributes (genes). Each of the rows of the dataset contains the expression of the genes of a particular patient. Experiments with more configurations in the number of features used were also carried out. Additionally, all results – those published in [14] and the ones presented in this paper – are analyzed by performing a hypothesis test to measure the level of significance for the metrics obtained with each algorithm.

This article is 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. All the works cited in this section measure performance as a function of the vertical (number of rows) increase in datasets, in order to study the efficiency of vertical splitting in the Spark framework. Spark stores data in an internal structure called Resilient Distributed Datasets (RDD). RDD is

[1] https://archive.ics.uci.edu/ml/index.php

formed by a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel (Figure 1). Subsets of these partitions are assigned to the computing nodes of the cluster in order to be processed. The choice of the partitioning algorithm has great impact in the performance of the algorithms that execute in the computing model proposed by the framework, since they involve different schedules of data distribution and synchronization among the computing elements.

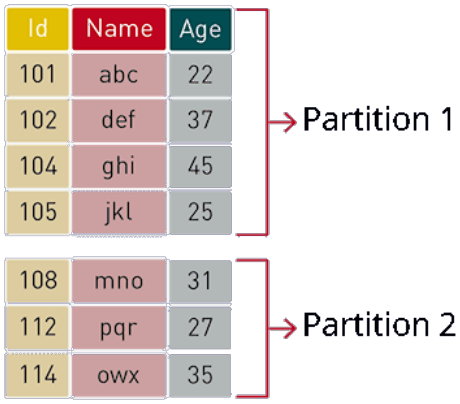

Figure 1: Spark generates partitions by splitting the dataset in a vertical fashion. Partitions are distributed among the nodes of the cluster to increase amount of paralellism.

Given the framework's reliance on an adequate vertical partitioning scheme to achieve higher performance, many works focus on the scaling of algorithms implemented in Spark as a function of the number of rows, but there has been little work focusing on horizontal data growth, disregarding the impact of the increase in dimensionality on tasks such as classification.

In [16], a comparison of the Naïve Bayes (NB), Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM) and Logistic Regression (LR) classifiers implemented in MLlib (version 1.6.2) is carried out with a database of $2,638,274$ Amazon product reviews [2]. These authors run trials with 7 different training set sizes, ranging from $25,000$ to $375,000$. The infrastructure of the data-processing cluster consists of a master with 4 vCPU and two workers with 2 vCPU. The only performance metric evaluated is accuracy. In addition, they also evaluate the performance of the classifiers by varying the number of n-grams extracted from each review. Their main conclusion of the work indicates that logistic regression is the classifier that achieves the best results of accuracy. The authors focus on the negligible impact of increasing the training set size, but disregard any measure or analysis of execution times.

In [17], the authors analyze tweets to determine patients at risk of heart attacks (303 samples and 13 features). They use several hyper parameter 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. The authors explore different feature selection approaches, resulting in a dataset with nearly half the features that allows achieving similar performance to the full dataset for most models. In this case, RF models achieved the best accuracies, followed by LR, SVM and DT.

In [18], 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 models are trained and evaluated using the the ADLs Binary Sensors Dataset and the DaLiAc Dataset, with $4,686,842$ samples. They evaluate the Spark implementations of LR, DT, RF and Multilayer Perceptron (MLP) using several configurations 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 better classification models for the two databases analyzed.

Recently, the authors of [19] measure the performance of LR, DT and SVM. They employ a χ^2 features selection technique in a case study of intrusion detection in computer networks. The dataset contains $140,000$ rows and 41 columns). The area under the ROC and PR curves is measured, as well as accuracy and training times. LR turns out to be the algorithm that obtains the best models when measuring the area under the ROC and PR curves, while DT is the one that obtains the models with the best accuracy. LR also turns out to be the fastest algorithm, followed by DT and lastly SVM. Both in this work and the previous ones, the authors did not include any details on the cluster configuration used to run the experiments. The results therefore appear to come from a single experiments, with no execution time information.

With the aim of predicting stock price fluctuations in the stock market, the authors of [20] study NB, RF, DT and LR measuring accuracy, Receiver Operating Characteristic (ROC) curves and Precision-Recall (PR) curves together with the execution time with various configurations of a Spark cluster with various numbers of worker nodes 3, 6, 8 y 11 nodes. 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 ($20,855,395$ rows). In conclusion, it was determined that RF and DT are the algorithms with the best predictive power. As for computation time, NB is the fastest algorithm followed by DT, while RF and LR were the algorithms that needed the most time to obtain a model. With regards to the execution time, the authors verify that these increase as a function of the number of nodes. In this way, they highlight the large differences in execution time between different algorithms in the low node count regime. Conversely,

[2] https://jmcauley.ucsd.edu/data/amazon/

the execution times of the different algorithms become more similar as the number of nodes increases, even if LR is the quickest model in all experiments. However, the authors do not vary the size of the dataset, which in turn impacts in the load-balancing strategy and therefore the execution speed of the training and testing phases of the model.

In [21], the performance of the LR, RF, SVM and PM algorithms in natural language processing applied to posts about the COVID-19 pandemic on Twitter (more than $2,000$ millions of rows) is compared. The variability of the experiments lies in the different numbers of records used to train the models. As regards, precision, recall, F1-measure, accuracy and execution time, LR is the fastest algorithm and SVM the slowest. On average, SVM and LR have better models than RF and PM. The main conclusion of the work indicates that the execution time for training classification algorithms increases with the size of the dataset.

The experiments carried out in [22] 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 (8 gigabytes, 28 attributes, and 11 million samples) from the UCI repository [3]. The conclusions of the work detail the RAM and node vCPUs configuration that achieve the best execution time with each algorithm, but the predictive power of the models obtained is not evaluated. In this work, the authors also conclude that the execution speed of all algorithms improves as more nodes are added to the cluster. In terms of vCPU, the optimal number per node is not well defined, with values ranging from 2 to 8. Agan, they do not perform any experiment varying the size of the dataset.

In [14], a comparison between four of the classification algorithms implemented in Spark ML is presented: RF, SVM, NB and MLP. Different predictive metrics (accuracy, F1-score, precision and recall) of the models are measured and compared, along with the execution time required for training. In the experiments, an ovarian cancer classification database [15] with $15,154$ attributes is used. The different experiments consist in measuring the classification algorithms' performance based on the number of attributes in the database used for training. Except for this work, no other contributions have been found that carry out a comprehensive study on the performance of classification algorithms based on the number of columns in the database.

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 ones that demand less computation time. In most of the works, the performance of Spark stands out – it scales very well in terms of the number of rows, thanks to the power offered by the Spark RDDs structures.

From this review, we can conclude that many previous works do not perform any sort of measurement and/or comparison of execution times, perhaps because the dataset size is too small to justify the use of a framework such as Spark. On the other hand, most of the works that do measure execution times focus on its dependence on dataset size as a function of the number of rows, not columns. Indeed, none of the previously cited works has measured the performance of Spark as a function of the number of features in the dataset.

In the context of feature selection this type of dependence is of critical importance, since many techniques such as those based on metaheuristics require repeatedly executing a training task with different feature subsets. That is why it is interesting to see how Spark performs when the parameter that is modified is the number of columns in the dataset, which is discussed below.

3 Experiments

As previously mentioned, the objective of this work is to extend the evaluation of the execution times of four classification algorithms of the Spark ML library of Spark: Naïve Bayes, Random Forest, Support Vector Machine, and Multilayer Perceptron. For this purpose, we use a binary classification databse for the task of ovarian cancer prediction. The dataset includes $15,154$ features/genes, and 243 samples. Each row in the dataset contains gene expressions of a specific patient, where each column corresponds to a different gene. In the experiments carried out, different numbers of features in the training dataset were used, measuring performance as the number of attributes in the dataset increased (from now on called the ω parameter). In this work, experiments were carried out with $\omega = 7,500$, $12,500$ and $15,000$, in order to extend the execution time curve presented at [14] with new data. In each experiment, in addition to the training time and the accuracy, precision, recall and F1-measure metrics, two additional new metrics were also measured: sensitivity and area under the curve. Since our goal was not about finding the best model by tuning the corresponding hyper parameters of the classification algorithms, the default values of the library itself were used, except for the Multilayer Perceptron whose structure was established in two 4- and 5-neuron hidden layers. Predictive power metrics were used simply to compare one model versus another, being aware that neither of them might be the best model that allows solving 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

[3] https://archive.ics.uci.edu/ml/datasets/HIGGS

Spark versions used were 3.2.2 and 3.1.1, respectively. Spark ML version 3.1.1 was used.

To avoid any bias, a five-fold cross-validation step was performed. To obtain these folds, stratified sampling [23] was used. Additionally, the entire process of randomly selecting attributes, dividing them into folds, training and evaluating the models was performed 30 times with each classification algorithm.

Each of the experiments carried out consists of the evaluation of a classification algorithm with a dataset with ω_i characteristics. In total, between the four algorithms studied and the configuration of 14 values for ω, a total of 56 experiments were obtained. As each of the experiments was run 30 times independently, measuring the seven metrics mentioned above in each run, the final result is a total of $56 * 7 = 392$ samples of 30 values each.

The results obtained for the metrics with the four algorithms were analyzed statistically. All metrics were compared in pairs, comparing the result of one algorithm versus all others, giving a total of six pairs of comparisons.

To perform statistical analysis, first, it was established whether the series of values corresponds to a normal distribution. For this, the Shapiro test was used, and the result was that only 228 out of the 392 experiments (58%) had a normal distribution. For this reason, it was decided to determine whether the mean values of each series were statistically significant or not; to do this, the Wilcoxon non-parametric test was carried out with a statistical significance of 95% ($\alpha = 0.05$)

4 Results

With the new values for ω used in this work and those used in [14], it was possible to "smooth" the time curves, and it was observed that the SVM and MLP algorithms are the algorithms that require the longest execution time as the number of training attributes increases. Figure 2 shows the average and standard deviation of the execution times for each algorithm for the different values for ω.

Figure 3 and 4 show the average and standard deviation of the 30 separate runs for the specificity and AUC metrics achieved by each of the algorithms. Based on the results shown in [14], it can be seen that NB was the algorithm that resulted on the worst models for all cases (except for the evaluation with 10 features for specificity). RF achieved very good results in the experiments with a large number of attributes, being on the same level as SVM and MLP, which were the algorithms that best evaluated the rest of the metrics. As the size of the evaluated set increases, MLP maintains its performance, while RF and SVM achieve excellent results.

For each of the seven metrics, the series of values corresponding to the 30 independent runs was compared. Each of the four classification algorithms was compared versus the others, comparing the 14 configurations of ω. Table 1 lists the number of times one model was significantly better than another. The results are shown grouped by metric; the triplet of values of each cell shows: first, the number of times that the M_i model of the corresponding row was statistically better than the M_c model, represented by the column; then, the number of times that the Wilcoxon test did not turn out to be significant, preventing the rejection of the null hypothesis that both models evaluated the same; and finally, the number of times M_i is statistically worse than M_c. In total, the values of each triplet add up to 14, this being the number of experiments evaluated with different values of ω.

Table 1: Statistical significance table. The values in the rows represent, in respective order, the times the model on the left performed better, the same, and worse than the model in the column for each given metric. For each metric and model, we highlight in bold the best result.

	RF	SVM	MLP
Time			
NB	**14**, 0, 0	**14**, 0, 0	**14**, 0, 0
RF	-	14, 0, 0	14, 0, 0
SVM	-	-	0, 1, 13
AUC			
NB	0, 0, **14**	0, 0, **14**	0, 0, **14**
RF	-	2, 4, **8**	**9**, 2, 3
SVM	-	-	**9**, 4, 1
Specificity			
NB	3, 0, **11**	2, 1, **11**	3, 1, **10**
RF	-	1, 4, **9**	**10**, 2, 2
SVM	-	-	**12**, 1, 1
Precision			
NB	0, 0, **14**	0, 1, **13**	1, 0, **13**
RF	-	2, 3, **9**	**6**, 2, **6**
SVM	-	-	**8**, 5, 1
Recall			
NB	0, 0, **14**	0, 0, **14**	0, 1, **13**
RF	-	1, 2, **11**	**6**, 2, **6**
SVM	-	-	**9**, 5, 0
Accuracy			
NB	0, 0, **14**	0, 0, **14**	0, 1, **13**
RF	-	1, 2, **11**	**6**, 2, **6**
SVM	-	-	**9**, 5, 0
F1-Score			
NB	0, 0, **14**	0, 1, **13**	1, 0, **13**
RF	-	2, 1, **11**	**6**, 2, **6**
SVM	-	-	**10**, 3, 1

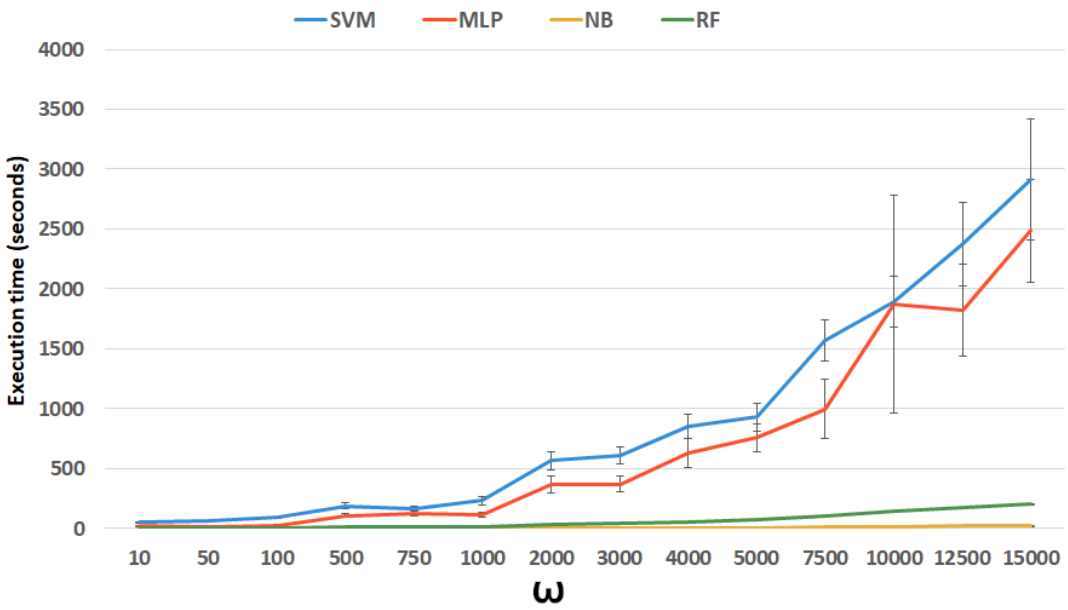

Figure 2: Average and standard deviation of the execution times of the four algorithms studied for the 14 values for ω.

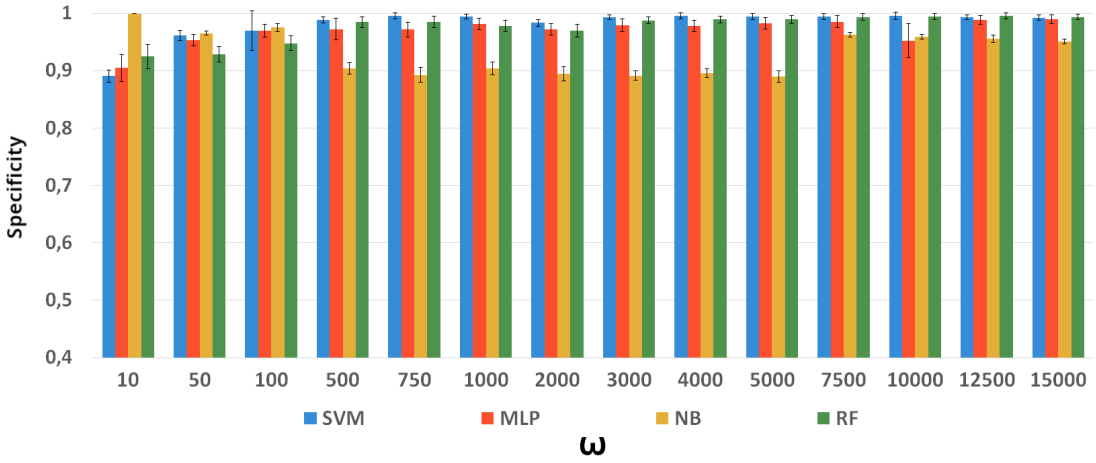

Figure 3: Average and standard deviation of the specificity of the four algorithms studied for the 14 values for ω.

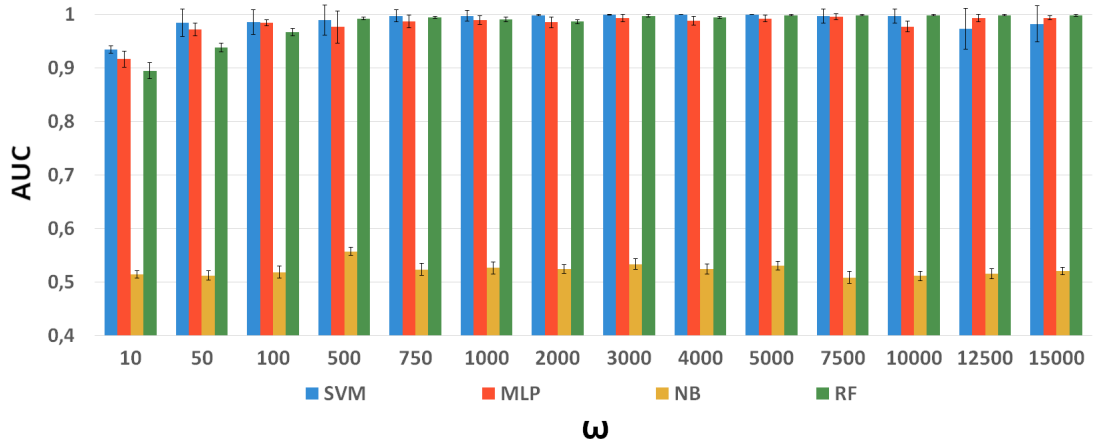

Figure 4: Average and standard deviation of the Area Under the Curve (AUC) of the four algorithms studied for the 14 values for ω.

Table 1 confirms the assumptions from the previous work: all the execution speeds measured were significantly different, including NB and RF which, for a small sample size, seemed similar in performance. NB is the fastest model for all ω values and models evaluated. Taking into account the new metrics (specificity and AUC), SVM and RF continue to be the models that yield better results, in some cases being very similar to MLP. The greatest difference is observed with specificity, where SVM yields better results in 12 out of the 14 cases, while RF vs. MLP yields better results in 10 out of the 14 total cases. NB continues to present very poor results for both metrics, evaluating worse for all cases and models.

5 Conclusions and future work

In this work, execution time, specificity and AUC were measured for four Spark ML algorithms varying the number of attributes in the training dataset; this was an extension of the work previously carried out and published in [14]. All comparisons between the metrics obtained by the algorithms were compared using a hypothesis test to measure whether their differences are statistically significant or not. The results obtained show that the algorithm that performed the worst for all the metrics is NB, although it managed to outperform the rest of the models presented in execution time. The algorithms that require the most computing time are SVM and MLP; yet, with a low number of attributes, these two algorithms are the ones that 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.

This performance-evaluation separation is interesting to evaluate in the field of feature selection, where a classification algorithm needs to be executed hundreds or thousands of times with varying numbers of attributes. The results obtained show the need to follow up with an algorithm with low execution time while achieving good models with few attributes.

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. In contrast, RF requires little execution time, but thousands of attributes to achieve a model that resembles the one achieved by SVM.

Even though the dataset used has a significant number of attributes (more than 15,000), it has few rows (253). A natural extension of this work would imply carrying out the same experiments using datasets with more rows and studying the performance of the classification algorithms in scenarios where the volume of data is greater. Furthermore, in this work we did not perform specific hyper parameter tuning for each model because of computational limitations. In future work, it would be useful to include the tuning to evaluate how it affects both the quality of models and their computational requirements.

It would also be interesting to measure the performance of intracluster data transmission in the Spark framework, via the evaluation of metaheuristics with the same kind of data. In this way, we can measure the load-balancing efficiency in feature selection tasks alongside the training of classifiers.

Competing interests

The authors have declared that no competing interests exist

Authors' contribution

WH and GC wrote the algorithm, performed the experiments and drafted the manuscript. RF and QF analyzed the results and did the literature review. All authors reviewed, edited and approved the final manuscript.

References

[1] G. Hernández, E. Zamora, H. Sossa, G. Téllez, and F. Furlán, "Hybrid neural networks for big data classification," *Neurocomputing*, vol. 390, pp. 327–340, 2020.

[2] W. Xing and Y. Bei, "Medical health big data classification based on knn classification algorithm," *IEEE Access*, vol. 8, pp. 28808–28819, 2019.

[3] S. Lakshmanaprabu, K. Shankar, M. Ilayaraja, A. W. Nasir, V. Vijayakumar, and N. Chilamkurti, "Random forest for big data classification in the internet of things using optimal features," *International journal of machine learning and cybernetics*, vol. 10, no. 10, pp. 2609–2618, 2019.

[4] E. M. Hassib, A. I. El-Desouky, L. M. Labib, and E.-S. M. El-kenawy, "Woa+ brnn: An imbalanced big data classification framework using whale optimization and deep neural network," *soft computing*, vol. 24, no. 8, pp. 5573–5592, 2020.

[5] A. K. Dubey, A. Kumar, and R. Agrawal, "An efficient aco-pso-based framework for data classification and preprocessing in big data," *Evolutionary Intelligence*, vol. 14, no. 2, pp. 909–922, 2021.

[6] U. Gurav and N. Sidnal, "Predict stock market behavior: Role of machine learning algorithms," in *Intelligent Computing and Information and Communication*, pp. 383–394, Springer, 2018.

[7] F. Ronchetti, F. Quiroga, G. Camele, W. Hasperué, and L. Lanzarini, "Un estudio de la generalización en la clasificación de peatones," *Revista Cubana de Transformación Digital*, vol. 2, no. 1, pp. 33–45, 2021.

[8] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[9] G. Kou, P. Yang, Y. Peng, F. Xiao, Y. Chen, and F. E. Alsaadi, "Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods," *Applied Soft Computing*, vol. 86, p. 105836, 2020.

[10] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[11] S. Alelyani, J. Tang, and H. Liu, "Feature selection for clustering: A review," *Data Clustering*, pp. 29–60, 2018.

[12] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 907–948, 2020.

[13] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction," *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020.

[14] G. Camele, W. Hasperué, F. Ronchetti, and F. Quiroga, "A comparative study of the performance of four classification algorithms from the apache sparkml library," *Congreso Argentino de Ciencias de la Computación*, 2020.

[15] E. Pashaei and N. Aydin, "Binary black hole algorithm for feature selection and classification on biological data," *Applied Soft Computing*, vol. 56, 03 2017.

[16] T. Pranckevičius and V. 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*, vol. 5, no. 2, p. 221, 2017.

[17] H. Ahmed, E. M. Younis, A. Hendawi, and A. A. Ali, "Heart disease identification from patients' social posts, machine learning solution on spark," *Future Generation Computer Systems*, vol. 111, pp. 714–722, 2020.

[18] D. Moldovan, M. Antal, C. Pop, A. Olosutean, T. Cioara, I. Anghel, and I. Salomie, "Spark-based classification algorithms for daily living activities," in *Computer Science On-line Conference*, pp. 69–78, Springer, 2018.

[19] 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)*, pp. 443–447, IEEE, 2020.

[20] J. Xianya, H. Mo, and L. Haifeng, "Stock classification prediction based on spark," *Procedia Computer Science*, vol. 162, pp. 243–250, 2019.

[21] W. S. Albaldawi and R. 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*, vol. 928, p. 032045, IOP Publishing, 2020.

[22] S. Yasrobi, J. Alston, B. Yadranjiaghdam, and N. Tabrizi, "Performance analysis of sparks machine learning library.," *Trans. MLDM*, vol. 10, no. 2, pp. 67–77, 2017.

[23] Z. Botev and A. Ridder, *Variance Reduction*, pp. 1–6. American Cancer Society, 2017.

Citation: G. Camele, W. Hasperué, F. Ronchetti and F.M. Quiroga. *Statistical Analysis of the Performance of Four Apache Spark ML Algorithms*. Journal of Computer Science & Technology, vol. 22, no. 2, pp. 175–182, 2022.
DOI: 10.24215/16666038.22.e14
Received: March 3, 2022 **Accepted:** May 17, 2022.