HAL
archives-ouvertes.fr

Ensemble Learning for Large Scale Virtual Screening on Apache Spark

Karima Sid, Mohamed Batouche

▶ To cite this version:

Karima Sid, Mohamed Batouche. Ensemble Learning for Large Scale Virtual Screening on Apache Spark. 6th IFIP International Conference on Computational Intelligence and Its Applications (CIIA), May 2018, Oran, Algeria. pp.244-256, 10.1007/978-3-319-89743-1_22 . hal-01913905

HAL Id: hal-01913905
https://hal.inria.fr/hal-01913905

Submitted on 7 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Ensemble Learning for Large Scale Virtual Screening on Apache Spark

Karima SID and Mohamed BATOUCHE

University of Constantine 2– Abdelhamid Mehri
Computer Science Department, Constantine, Algeria
sidk.karima@gmail.com, mohamed.batouche@univ-constantine2.dz

Abstract. Virtual screening (VS) is an *in-silico* tool for drug discovery that aims to identify the candidate drugs through computational techniques by screening large libraries of small molecules. Various ligand and structure-based virtual screening approaches have been proposed in the last decades. Machine learning (ML) techniques have been widely applied in drug discovery and development process, predominantly in ligand based virtual screening approaches. Ensemble learning is a very common paradigm in ML field, where many models are trained on the same problem's data, to combine in the end the results in one improved prediction. Applying VS to massive molecular libraries (Big Data) is computationally intensive; so the split of these data to chunks to parallelize and distribute the task became necessary. For many years, MapReduce has been successfully applied on clusters to solve the problems with very large datasets, but with some limitations. Apache Spark is an open source framework for Big Data processing, which overcomes the shortcomings of MapReduce. In this paper, we propose a new approach based on ensemble learning paradigm in Apache Spark to improve in terms of execution time and precision the large-scale virtual screening. We generate a new training dataset to evaluate our approach. The experimental results show a good predictive performance up to 92% precision with an acceptable execution time.

Keywords: Virtual Screening, Big Data, Apache Spark, Machine Learning, Ensemble Learning.

1 Introduction

The discovering of new drug is a very expensive and long process. High Throughput Screening (HTS) is a widely used experimental tool in the drug discovery process, where large molecular libraries are screened in fully automated environments [1]. However, with the very fast increase in the size of these libraries, HTS will be expensive and provides a small number of hits with a high false positive and false-negative rate [1, 2]. As an alternative, Virtual Screening (VS) is a pre-screening technique, cheaper and faster than HTS, successfully applied to decrease (filter) the number of compounds to be screened by generating new drug leads [2, 3]. There are two strategies for Virtual Screening: Ligand based (LBVS) and Structure based (SBVS) [3]. In LBVS, the existing information about the ligands is used to find compounds that best match a given

query; this strategy can work in the absence of structural information of the target [4]. However, in SBVS strategy, the structural information of the target (generally proteins) is required [4].

Machine learning (ML) is a very active branch in artificial intelligence domain; it aims to build models that can predict the output value of input data. The application of ML in VS process is not recent; various methods have been developed in this context. Generally, there are two main applications of ML techniques in VS process. Firstly, in LBVS, where the common task is to distinguish between active and inactive compounds in a given dataset [3, 4, 5]; secondly, in SBVS, as scoring functions to improve structure-based binding affinity prediction [6, 7]. Artificial Neural Networks (ANN), Support Vector Machines (SVM), Decision Trees (DT) are the most popular used ML techniques. Ensemble learning is a recent paradigm in machine learning area, where the basic concept is to train a set of base learners on the same dataset, and combine their predictions into a single output prediction that should have better performance [8].

Recently, the number of compounds in the molecular libraries has increased considerably. Applying machine learning techniques on massive libraries (Big Data) in VS process is computationally expensive [9]. The need to sophisticated frameworks for efficient Big Data analytics is becoming more important [9]. Apache Hadoop [10] is one of the most popular used platforms for Big Data analytics; it includes Google's MapReduce model [11] as processing tool and Hadoop Distributed File System (HDFS) as storage system. Some limitations have been known with Google's MapReduce, such as acyclic data flow model, the absence of some features such as in-memory data caching (cache reusable data), and broadcast variables (reusable data), make it inappropriate for some applications [1]. Apache Spark [12, 13] is an open source framework for large-scale datasets processing on clusters, which overcomes the shortcomings of MapReduce, while providing similar scalability and fault tolerance properties [13]. It includes a set of libraries to support a variety of compute intensive tasks for instance *Spark MLlib* for Big Data machine learning [14, 15].

The rest of this paper is organized as follows. Section 2 discusses briefly some works carried out in the same context. Section 3 explains a set of methods and techniques that are used to develop our work. Section 4 details the proposed approach and the main sub-workflows. Section 5 presents and discusses the obtained results, while the paper is concluded in Section 6.

2 Related Works

Lately, research works concerning machine learning on Big Data in VS process is quite active. In [16] the authors used Spark and MapReduce programming model to implement SVM based virtual screening. The work showed how HDFS and Spark could be used in combination to distribute and process data in parallel, with a satisfactory scaling behavior. In the study [17], the author developed a general pipeline to perform machine learning on big datasets to derive predictive models using Apache Spark. These models are generated by learning from already tested chemical substances; the results showed the effectiveness of Spark to create pipelines based on machine learning techniques with a good scaling behavior in a distributed environment. Dries Harnie and his teamwork [18] re-implemented the Chemogenomics pipeline using Apache Spark (S-

CHEMO). The Chemogenomics project attempts to derive new candidate drugs from existing experiments through a set of machine learning predictor programs. The S-CHEMO aims to scale the existing pipeline to a multi-node cluster without changes. The authors benchmarked S-CHEMO pipeline against the original; the results showed almost linear speedup up to eight nodes. Where in the study [19], the author used large unbalanced dataset to train a homogeneous ensemble learning based on Support Vector Machine techniques. The evaluation was for two metrics, the predictive performance of the ensemble model, and the scalability (weak and strong scaling). The results proved that Apache Spark is a very powerful tool for Big Data machine learning.

This synthesis allowed us to see the effectiveness of Apache Spark, machine learning, and ensemble learning paradigm to improve the virtual screening process in terms of execution time and predictive performance. This paper aims to exploit this effectiveness to propose a novel approach for large scale VS, where the originality of the approach and the main contributions can be summarized in the following points:

- Propose a process to generate a new training dataset to be used for performance evaluation;
- Construct a heterogeneous ensemble learning model in Apache Spark using a different type of classifiers ;

3 Methods and Materials

In this section, we will explain the methods and techniques used to develop the proposed approach, which is based principally on *MLlib* library for Big Data machine learning in Apache Spark, and heterogeneous ensemble learning using three different classifiers SVM, DT and MLP.

3.1 Machine learning algorithms

In VS context, the common task of machine learning algorithms is the assignment of molecular descriptors[1] vector (sample) to a class. Generally, ML algorithms are divided into two main categories:

- *Supervised learning*, which requires the vector of class-labels, both the input and the target (class) values for each sample are used in the training to derive classification/regression models [20]. The most common algorithms in this category are; Artificial Neural Networks (ANN), Support Vector Machines (SVMs), and Decision Trees (DT) [20];
- *Unsupervised learning*: which is used when the vector of class-labels is unknown [20].

Support Vector Machines. Support vector machine (SVM) form a class of supervised machine learning algorithms, which train the classifier model using pre-labeled data [21]. The SVM algorithm in LBVS intended to separate compounds that are represented

[1] http://www.moleculardescriptors.eu/

by vectors of molecular descriptors in the training dataset into active and inactive compounds [20]. Each vector is represented as a *support vector* composed by its attributes (molecular descriptors), SVM search for one target known as the optimal hyperplane ρ that separates the support vectors. The optimal hyperplane ρ maximizes the margin of separation between the hyperplane and the closest data points (support vectors) on both sides of the hyperplane [21, 22].

Decision Trees. Decision Trees (DTs) is one of the most popular used supervised machine learning technique in LBVS. The general reason of using DT is to create a training model which can be used to predict classes of input compounds by learning decision rules inferred from prior data (training dataset) [23]. DT algorithm attempts to solve the problem by using tree representation. Each leaf node corresponds to a class label, whereas non-leaf node (root or internal node) corresponds to molecular descriptor, and branch represents a test on corresponding molecular descriptor [20, 23]. The class of unknown compound is a leaf node that it achieved over a series of questions (nodes) and answers (deciding which branches to take) from the root node [20]. Random Forests (RF) or Decision Forests (DF) is an ensemble classifier including many DTs based on bagging technique [8]. For each DT classifier the training set is constructed by random sampling with replacement from the original dataset [20].

Multi-Layer Perceptron. Multi-Layer Perceptron (MLP) is a feedforward Artificial Neural Network (feedforward ANN) consists of multiple layers (input layer, hidden layers and output layer) [20]. Each layer consists of a variable number of neurons, where the output layer represents the class-labels (two neurons for two class-labels: active - inactive). Each neuron has multiple inputs associated with weights and one output, and related to an activation function. During the training, MLP model seeks to solve an optimization problem by optimizing the weights of each neuron, through back propagation and gradient descent techniques. The optimization problem aims to minimize the mean-square error that represents the difference between the model outputs and the correct answers (correct outputs) [20].

Ensemble learning. In the classification context, the main idea behind the ensemble learning paradigm is to weight several base classifiers (weak learners), and combine them to obtain a more efficient classifier (strong learner) [8, 24]. To derive an ensemble learning model, we need to follow two main steps. Firstly, many base classifiers are generated and trained in a parallel (e.g. bagging) or in a sequential (e.g. Boosting) manner [8]. Secondly, the results of base classifiers are combined, the most popular technique of combination for classification is majority voting [8], according to equations (1) and (2) [24].

$$class\ (x) = \arg \max_{L_i \in dom(c)} (\sum_k g(c_k(x), L_i)) \quad (1)$$

Where *k* is the number of classifiers, $c_k(x)$ is the classification result of the *k'th* classifier and g (c, *L)* defined as [24]:

$$\begin{cases} 1 \text{ if } c = L \\ 0 \text{ if } c \neq L \end{cases} \quad (2)$$

3.2 Apache Spark for Big Data Machine Learning

For many years, Apache Hadoop [10] was the de facto standard for Big Data analytics [25], because of its ecosystem structure that includes a set of modules appropriate to manage this type of data; principally ,Google's MapReduce [11] for the processing and Hadoop Distributed File System (HDFS) for the storage [25]. The Hadoop MapReduce provides many benefits such as flexibility, scalability and fault-tolerance, but at the same time, it has some limitations that make it not suitable for some applications such as iterative jobs (e.g. Machine learning algorithms), and interactive analytics [1, 13, 22].

Apache Spark [12] is a highly scalable, fast and in-memory Big Data processing engine [9]; it overcomes the shortcomings of Hadoop MapReduce model, while retaining scalability and fault tolerance [13]; it offers an ability to develop distributed applications using Java, Python, Scala, and R programming languages [9]. It consists of a set of libraries for different compute intensive tasks, including Apache Spark Streaming, Apache Spark SQL, Apache Spark GraphX, and Apache Spark MLlib [9]. In cluster mode, Spark supports three cluster managers, standalone, YARN of Hadoop and Mesos. It can access diverse data sources including HDFS, Cassandra, HBase, and S3 [12, 22]. The main abstraction in Spark is the Resilient Distributed Dataset (RDD) [25], which is defined as an immutable collection of objects partitioned across the nodes in the cluster, it can be cached in memory (as reusable data) and rebuilt if a partition is lost through a notion of *lineage* [13, 22].

Spark MLlib is a distributed machine learning library; it consists of fast and scalable implementations of standard learning algorithms, including classification, regression, collaborative filtering, clustering, and dimensionality reduction [15]. It also provides a variety of underlying statistics, linear algebra, and optimization primitives. As part of Spark ecosystem, *MLlib* provides a high-level API to simplify the development of machine learning pipelines [25, 15].

3.3 The Chemistry Development Kit

The Chemistry Development Kit (CDK)[2] is an open source toolkit implemented in Java for Structural Chemo-and Bio- informatics. It provides methods for many common tasks in molecular informatics such as the calculation of molecular descriptors [26]. In this paper, we used CDK version 2.0 to generate a vector of molecular descriptors for each molecule for succeeding machine learning steps [17].

4 Proposed Approach

Our proposed approach is based on Spark's master-worker architecture as shown in the figure below (see fig. 1), using Standalone cluster manager to acquire resources on the cluster, and HDFS as storage system; where the master node acts as NameNode of HDFS, and each worker node acts as DataNode of HDFS.

[2] http://sourceforge.net/projects/cdk/

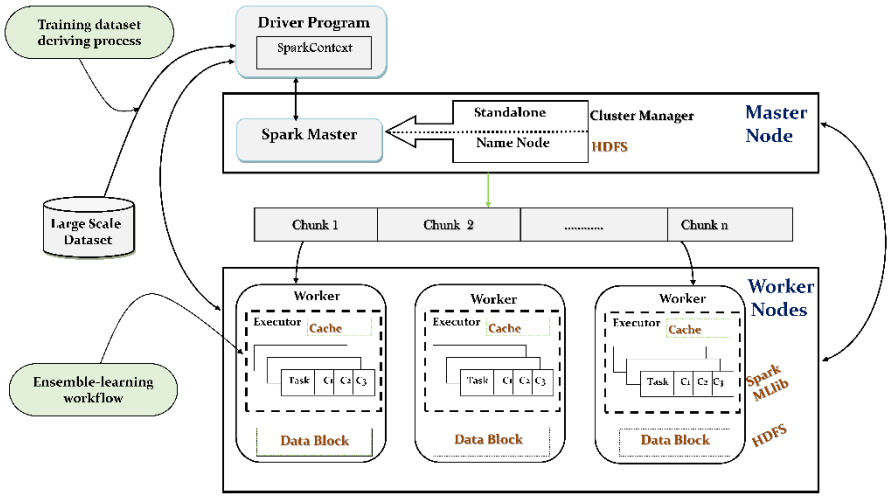

Fig. 1. Schematic representation of the proposed approach on Spark Cluster.

Firstly, we create SparkContext via "`new SparkContext (conf)`" instance, which allows accessing the cluster through a resources manager (Standalone). "`conf`" represents SparkConf instance created via the instruction, "`new SparkConf().setAppName().set("parameters")` that stores the configuration parameters to pass it to SparkContext such as the number of cores and memory size used by the executors on worker nodes.

When loading a dataset from HDFS to an RDD via "`sc.textFile("hdfs://...")`" method, Spark normally splits the input data into chunks or partitions. Next, the data partitions are distributed in the cluster (over the worker nodes) and conserved at DataNode; while NameNode only contains the metadata about the dataset kept at each DataNode. Then, in each worker the dataset is processed, a task for each partition is launched.

To transfer our application, we create a JAR file using Apache Maven[3], which includes all the dependencies. This JAR file is then submitted to a Spark cluster through the command line spark-submit as follows:

"`bin>spark-submit --master spark://master URL --class Main_class / path/to/JARfile.jar`".

One JAR file includes all the dependencies ensure the availability of all these dependencies in each worker node, which reduce the overall processing time. The worker nodes process the dataset stored at DataNode (Data Block) using Ensemble-learning workflow (see fig. 2). When all the worker nodes complete the processing, the master node gets back the final results.

[3] https://maven.apache.org/

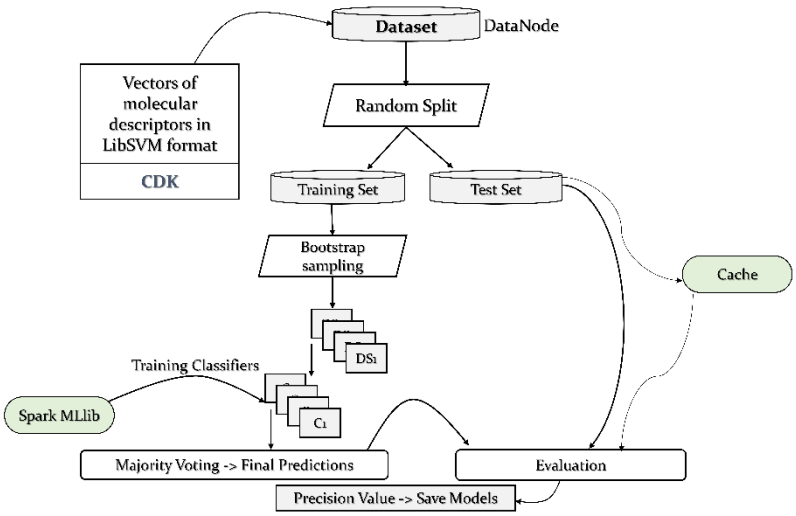

Fig. 2. Ensemble-learning workflow.

5 Experimental Results

This section describes the proposed process to generate our training dataset to be used to evaluate the predictive performance and the scalability, in the first section. Next, in the second section we will present and discuss the obtained results.

5.1 Dataset Generation

In this paper, we chose to study HIV/AIDS disease, which still does not have a practical drug. The main obstacle in the treatment of HIV is the ability of the virus to mutate rapidly into drug-resistant variants [27]. A major target in HIV disease is its protease (HIV protease receptor). Many studies have been developed to discover new HIV protease inhibitors that aims principally to prevent viral replication by selectively binding to viral proteases [28]. The following steps are used to derive our training dataset.

Step 1: HIV protease inhibitors preparation. Firstly, we extract a set of HIV protease inhibitors in SDF format from ChEBI [4] database (ID = CHEBI: 35660) that contains 23 inhibitors. Next, we generate the vectors of molecular descriptors of these inhibitors using CDK toolkit.

Step 2: Dataset preparation. The dataset used in this study was established from the open chemical database ChEMBL[5]. ChEMBL is a structured database, which is containing more than one million bioactive drug-like substances. From ChEMBL version

[4] https://www.ebi.ac.uk/chebi/
[5] https://www.ebi.ac.uk/chembl/

23, we extract randomly 50E+4 compounds. Next, we generate a vector of molecular descriptors for each compound using CDK toolkit.

Step 3: Training dataset generation. To derive our training dataset (see fig. 3) we used Tanimoto index [29] to measure the similarity between two compounds (*i: compound, j: inhibitor*). Tanimoto coefficient *sim(i,j)* can be calculated as follows:

$$sim\ (i,j) = \frac{\sum_{k=1}^{l} x_{ki} x_{kj}}{\sum_{k=1}^{l} (x_{ki})^2 + \sum_{k=1}^{l} (x_{kj})^2 - \sum_{k=1}^{l} x_{ki} x_{kj}} \quad (3)$$

Where *l* represents the number of descriptors and *x* represents the molecular descriptors vector. The threshold values for similarity compounds are typically in the range of 0.8 to 0.9 [29]. We selected 0.9 as threshold to classify the compounds as inhibitors (actives) and non-inhibitors (inactive).

```
Inputs:
IN: a set of 23 inhibitors
CM: a set of compounds
s: dataset size
Output:
L: vector of class-labels
Begin
  for i =1 to s do
    for j=1 to 23 do
      Calculate sim(CM(i), IN(j)); according to eq. (3)
      if sim>=0,9 then
        L(i)=1
      else
        L(i)=0
      end
    end
  end
End
```

Fig. 3. Pseudo code of training dataset deriving process.

5.2 Results and Discussion

We implemented the proposed approach using Spark 1.6 version, Hadoop 2.6 version and Scala as programming language with Scala IDE for eclipse 4.3 version. The configuration of the computer used for experiments is a local machine Intel Core i7 with 2.10 GHz speed and 8 GB of RAM.

We created Spark Standalone cluster locally (see fig. 4), we launched firstly one spark master using the command line "`bin>spark-class org.apache.spark.deploy.master.Master`".

Next, we launched a set of workers using the command "`bin >spark-class org.apache.spark.deploy.worker.Worker spark://master URL`". Where master `URL` takes format *@IP:port* (ex. 192.168.223.1:7077).

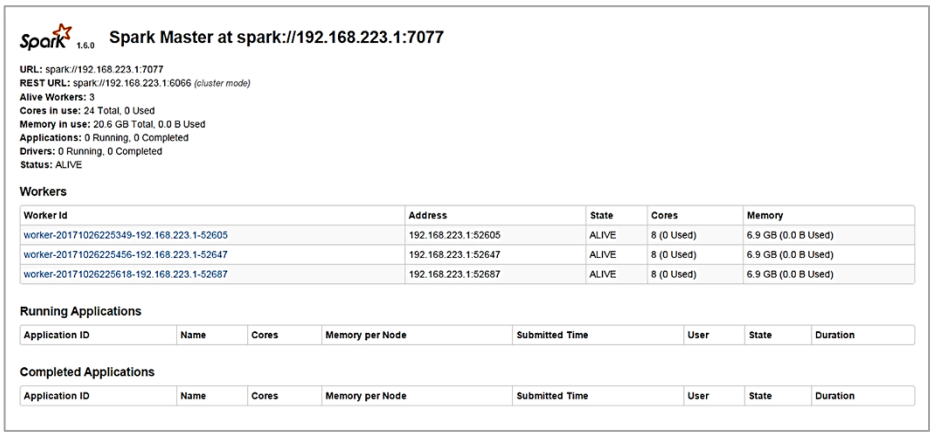

Fig. 4. Example of locally Standalone cluster with three workers.

Precision evaluation. To evaluate the predictive performance of our approach, we used 10-fold cross validation technique, where the dataset is randomly divided into ten subsets, nine of them are used for training and the last one is used as a test set to validate the classifier [5]. We selected the precision as metric evaluation that represents the fraction of correctly identified positives over the total amount of instances classified as positive [19]. The results are illustrated in the following figure (see fig. 5).

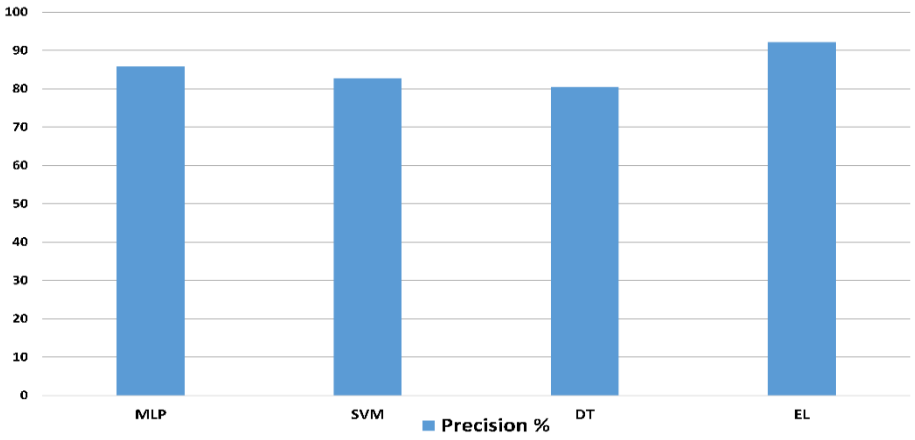

Fig. 5. Precision results for MLP, SVM and DT classifiers compared with ensemble learning model (EL).

The results show that Multi-Layer Perceptron (MLP) classifier gave higher precision on our dataset than the other classifiers SVM and DT, while the ensemble learning model (EL) made a good improvement in the results with precision up to 92%.

Scalability evaluation. In order to evaluate the scalability of our approach, we carried out a scalability test on two aspects: scaling with training dataset size (weak scaling) and scaling with cluster size (strong scaling).

The figure below (see fig.6) shows the effect of training dataset size on the execution time. With more instances, ensemble learning model takes longer to train. These results are obtained for a number of worker nodes equal to 5.

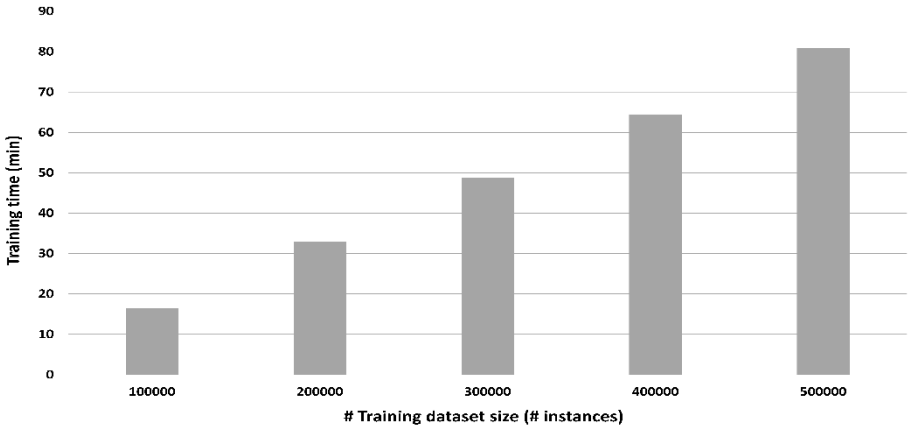

Fig. 6. Scaling with training dataset size (# instances).

The figure below (see fig.7) illustrate the obtained results where scaling the approach with cluster size. We started with 2 workers, and the number of workers increases in each test with 2 nodes. The results show that the time is decreasing linearly, where with 2 workers the workflow takes average of 102 minutes to return the results and approximately 11 minutes with 16 workers. We can say that ensemble learning model is significantly faster when using more workers.

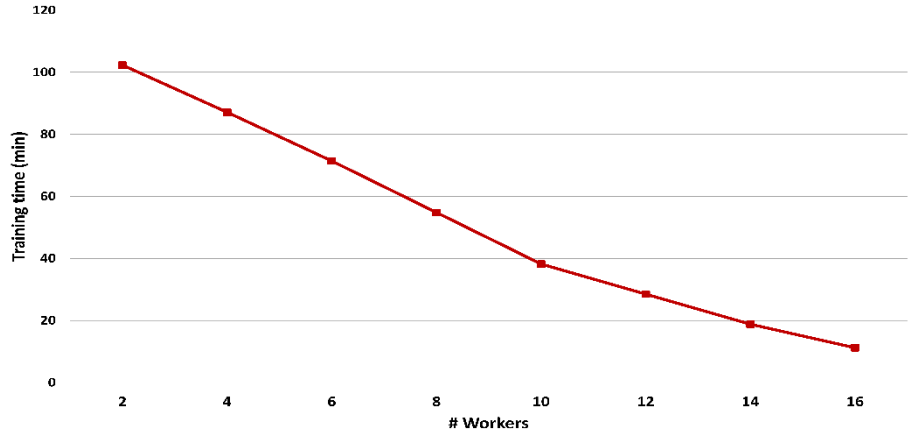

Fig. 7. Scaling with cluster size (# workers).

6 Conclusion and Future Work

In this paper, we have presented a new approach based on ensemble learning paradigm and Apache Spark to enhance the performance of large-scale virtual screening process. We have used three classifiers which are multi-layer perceptron, decision trees and support vector machines in combination to establish our ensemble learning model, where the technique of aggregation was majority voting. The approach has been based on master-worker Spark architecture, using standalone cluster manager and HDFS as storage system. The implementation of ensemble learning in Spark has been via Spark MLlib library. To evaluate the approach, we have generated a new training dataset. The process of generation has been consisted of three steps using CDK toolkit and similarity index Tanimoto. The obtained results have shown the effectiveness of our approach with precision up to 92% in few minutes. As a future work, we plan to use other machine learning models for instance deep learning architecture, with other Big Data frameworks such as Sparkling Water (H2O) and DeepLearning4j to implement the approach in Big Data context.

References

1. Capuccini, M., Ahmed, L. Schaal, W., Laure, E., Spjuth, O.: Large scale virtual screening on public cloud resources with Apache Spark. J. Cheminformatics 9(15), 1-6 (2017). doi: 10.1186/s13321-017-0204-4.
2. Pradeep, P., Struble, C., Neumann, T., Sem, D.S., Merrill, S.J.: A Novel Scoring Based Distributed Protein Docking Application to Improve Enrichment. J. IEEE/ACM Transactions on Computational Biology and Bioinformatics 12(6), 1-8 (2015). doi: 10.1109/TCBB.2015.2401020.
3. Fang, X., Bagui, S., Bagui, S.: Improving Virtual Screening Predictive Accuracy of Human Kallikrein 5 inhibitors using Machine Learning Models. J. Computational Biology and Chemistry 69, 110-119 (2017). doi: 10.1016/j.compbiolchem.2017.05.007.
4. Preeja, M.P., Hemant, P., Soman, K.P., Prashant, S. K.: Ligand-Based Virtual Screening using Random Walk Kernel and Empirical Filters. J. Procedia Computer Science 57, 418–427 (2015). doi: 10.1016/j.procs.2015.07.508.
5. Upul, S., Rahal, P., Roshan, R.: Machine Learning based Search Space Optimisation for Drug Discovery. In: IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), pp. 68-75. IEEE Press, Singapore (2013). doi: 10.1109/CIBCB.2013.6595390.
6. Ain, Q. U., Aleksandrova, A., Roessler, F. D., Ballester, P. J.: Machine-learning scoring functions to improve structure-based binding affinity prediction and virtual screening. WIREs Comput Mol Sci 5, 405–424 (2015). doi:10.1002/wcms.1225.
7. Boff de Ávila, M., et al. : Supervised machine learning techniques to predict binding affinity. A study for cyclin-dependent kinase 2. J. Biochemical and Biophysical Research Communications 494 (1–2), 305-310 (2017). doi: 10.1016/j.bbrc.2017.10.035.
8. Yun, Y.: Ensemble Learning. Chapter 4 - In Temporal Data Mining Via Unsupervised Ensemble Learning, Elsevier, 35-56 (2017). doi: 10.1016/B978-0-12-811654-8.00004-X.
9. Mehdi, A., Ehsun, B., Liu, G., Ahmad, P. T.: Big Data Machine Learning using Apache Spark MLlib. In: IEEE International Conference on Big Data (BIGDATA), pp. 3492- 3498. IEEE Press, Boston, USA (2017). doi:10.1109/BigData.2017.8258338.
10. Apache Hadoop, http://hadoop.apache.org, last accessed 07/02/2018.

11. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: the 6th Symposium on Operating Systems Design and Implementation, pp.137-149. San Francisco (2004).
12. Apache Spark™, http://spark.apache.org, last accessed 07/02/2018.
13. Zaharia, M. et al.: Spark: Cluster computing with working sets. In: the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, pp. 1-7. USA (2010).
14. Wei, H. et al.: In-Memory Parallel Processing of Massive Remotely Sensed Data Using an Apache Spark on Hadoop YARN Model. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 10(1), 3-19 (2017). doi: 10.1109/JSTARS.2016.2547020.
15. Meng, X. et al.: MLlib: Machine Learning in Apache Spark. J. Machine Learning Research 17(34), 1−7 (2016).
16. Ahmed, L., Edlund, A., Laure, E., Spjuth, O.: Using Iterative MapReduce for Parallel Virtual Screening. In: IEEE International Conference on Cloud Computing Technology and Science, pp. 27-32. IEEE Press, Bristol, UK (2013). doi: 10.1109/CloudCom.2013.99.
17. Staffan, A.: Automating model building in ligand-based predictive drug discovery using the Spark framework. Degree Project in Bioinformatics, Masters Programme in Molecular Biotechnology Engineering, Uppsala University School of Engineering (2015).
18. Harnie, D. et al.: Scaling machine learning for target prediction in drug discovery using Apache Spark. J. Future Generation Computer Systems 67, 409-417 (2017) .doi: 10.1016/j.future.2016.04.023.
19. Simon, L.: Distributed Ensemble Learning with Apache Spark. Degree Project in Bioinformatics, Masters Programme in Molecular Biotechnology Engineering, Uppsala University School of Engineering (2016).
20. Antonio, L.: Machine-learning approaches in drug discovery: methods and applications. J. Drug Discovery Today 20 (3), 318- 331 (2015). doi: 10.1016/j.drudis.2014.10.012.
21. Bissan, G., Joe, N.S.: High dimensional data classification and feature selection using support vector machines. J. European Journal of Operational Research 265 (3), 993-1004 (2018). doi : 10.1016/j.ejor.2017.08.040.
22. Karima, S., Mohamed, B.: Big Data Analytic Techniques in Virtual Screening for Drug Discovery. In: the 2nd international Conference on Big Data, Cloud and Applications (BDCA), Article 9, 7 pages. ACM, Morocco (2017). doi 10.1145/3090354.3090363.
23. Introduction to Decision Tree Algorithm, http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/, last accessed 07/02/2018.
24. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33 (1-2), 1-39 (2010).
25. Bilal, A, Ying, Z., Uwe, R. On the Usability of Hadoop MapReduce, Apache Spark & Apache Flink for Data Science. In: IEEE International Conference on Big Data (BIGDATA), pp. 303 – 310. IEEE Press, Boston, USA (2017). doi: 10.1109/BigData.2017.8257938.
26. Christoph, S. et al.: The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo-and Bioinformatics. J. Chemical Information and Computer Sciences 43(2), 493–500 (2003). doi: 10.1021/ci025584y.
27. Maris, L. et al.: Proteochemometric modeling of HIV protease susceptibility. J. BMC Bioinformatics 9:181 (2008). doi: 0.1186/1471-2105-9-181.
28. Protease inhibitor (pharmacology), https://en.wikipedia.org/wiki/Protease_inhibitor_(pharmacology), last accessed 07/02/2018.
29. Han, B., et al.: Development and experimental test of support vector machines virtual screening method for searching Src inhibitors from large compound libraries. J. Chemistry Central Journal, 6:139 (2012). doi: 10.1186/1752-153X-6-139.