Cloud Implementation of Classifier Nominal Concepts using DistributedWekaSpark

Rawia Fray, Nida Meddouri, and Mondher Maddouri

University of Tunis El Manar
Faculty of Sciences of Tunis
LIPAH, Tunis, Tunisia

Abstract. In this article, we are interested in the Cloudification of a classification method based on Formal Concept Analysis named Classifier Nominal Concepts. The basic idea is to create a distributed version for this existing method, named Distributed Classifier Nominal Concepts, and implement it on Cloud Computing. Implementation of a classification method on cloud is one of Distributed/Big Data Mining methods. The latter generally depends on four requirements: a Big Data Framework to support the distribution of applications, a Distributed Data Mining tool, a parallel programming model, e.g. MapReduce, and the distributed system Cloud Computing used as an execution environment. In our work, we chose Spark as a Big Data Framework, DistributedWekaSpark as a Distributed Data Mining tool, and Amazon Web Services Cloud as environment for implementation. We implemented our approach on a Cluster of five virtual machines by using large data samples for testing. This Cloudified version is compared to the sequential single-node executed version. The evaluation of the results demonstrate the effectiveness of our work.

Keywords: Formal Concept Analysis · Cloud Computing · Big Data Mining · Classifier Nominal Concepts · DistributedWekaSpark · Amazon Web Services.

1 Introduction

Data mining is the process of discovering interesting patterns and knowledge from large amounts of data [8]. With the continuous and rapid growth of data size, extracting knowledge from these data using traditional data mining tools and algorithms has become difficult.

Big Data Frameworks are created to make the possibility of large data processing in general. However, extracting knowledge from this large data depends on specific tools, called Big Data Mining tools. Typically, these tools rely on

a distributed environment such as Cloud Computing to prove their effectiveness, so they are also called distributed data mining tools. Several distributed data mining tools were created. This has allowed the development of various distributed versions for data mining methods and the implementation of these methods on distributed environments. Classifier Nominal Concepts(CNC) is one of data mining algorithms. CNC is a classification method based on Formal Concept Analysis (FCA)[10]. A distributed version of CNC can be made using the Big Data Minig tool DistributedWekaSpark[17], and implemented on Cloud Computing.

This paper is organized as follows. In Section 2, we give an overview of most popular Big Data frameworks, the parallel programminig model MapReduce and the Big Data mining system. In Section 3, we recall some basics of FCA, we present the principle of the classification method called Classifier Nominal Concepts(CNC), and we introduce a distributed version for this method based on the unified paradigm of DistributedWekaSpark tool. Section 4 is devoted to the implementation of our proposed method on Cloud, and the presentation of experimental results allowing the evaluation of the performance of our proposed approach.

2 Preliminaries

Distributed data mining attempts to improve the performance of traditional data mining systems, and recently it has garnered much attention from the data mining community. Distributed data mining is mentioned with parallel data mining in the literature[3]. There are several tools created to scale existing Data Mining algorithms. These tools depend on Big Data framework used.

2.1 Big Data frameworks

With the emergence of cloud computing and other distributed computing systems, the amount of data generated is increasing every day. So, these sets of large volumes of complex data that can not be processed using traditional data processing software are called Big Data. Big Data concern large-volume, complex and growing data sets with multiple and autonomous sources. There are many Big Data techniques that can be used to store data, perform tasks faster, make the system distributed, increase processing speed, and analyze data. To perform these tasks, we need Big Data frameworks. A Big Data framework is the set of functions or structure that defines how to perform processing, manipulation and representation of large data, it manages both structured, unstructured and semi-structured data[14]. The best-known Big Data frameworks are: Apache Spark, Apache Hadoop, Apache Storm, Apache Flink and Apache Samza. In [19, 16, 11, 12], you find a survey study on these frameworks. We focus on the two appreciated and most used, Apache Hadoop and Apache Spark.

Apache Spark is a framework characterized by its speed. It aims to accelerate batch workloads, this is done by the complete computation in memory and optimization of the processing[6]. Spark can be integrated with Hadoop, and it is

more advantageous compared to other big data frameworks. Spark is characterized by Resilient Distributed Data set(RDD)[7], which is a collection of objects partitioned across a cluster(set of computing machines)[6].

Apache Hadoop is an open source, scalable and fault-tolerant framework. It is a processing framework that provides only batch processing and effectively handles large volumes of data on a core hardware cluster. The two main components of Apache Hadoop are Hadoop Distributed File System(HDFS) and MapReduce. HDFS provides a distributed file system that allows large files to be stored on distributed machines reliably and efficiently[9]. MapReduce is the native batch processing engine of Hadoop.

2.2 MapReduce Programming Model

The MapReduce model is a programming paradigm that allows the computing of huge amounts of data on clusters of physical or virtual computers[5]. These benefits include scalability, crash tolerance, ease of use and cost reduction. There are two basic steps in MapReduce.

The Map function: is the first step of the model, this function takes input and creates key and value pairs (k, v). Then, it transforms them into a list of intermediate pairs of keys and values: List (Ki, Vi). Intermediate values that belong to the same intermediate key are fixed and then transmitted to the Minimize function.
Map (k, v) ⇒List(Ki, Vi)

The Reduce function: follows the Map function, it return a final model by merging values that possess the same key.
Reduce (Ki, List(Vi)) ⇒ List (Vo)

2.3 Big Data Mining system

Data Mining and Machine Learning allow to use the different aspects of Big Data technologies(such Big Data frameworks mentioned previously), to scale up existing algorithms and solve some of the related problems[15]. A scalable solution for Big Data Mining depends on many relied components that forms a Big Data Mining system. The first component is the user interface that allows the user to interact with the Big Data Mining system. The second component is the application that contains our code with all the dependencies. The third component is the big data framework that corresponds to our application. The fourth component is the distributed storage layer where data is stored, the latter encapsulates the local storage of data in a large-scale logical environment. Finally, the infrastructure layer that contains a set of virtual or physical machines, these machines form a cluster [17].

3 Distributed Classifier Nominal Concepts

3.1 Basics of Formal Concept Analysis

Definition 1. ***Formal context***
A formal context is a triplet ($\mathcal{G}$,$\mathcal{M}$,$\mathcal{I}$). The elements of $\mathcal{G}$ are called objects, the elements of $\mathcal{M}$ are called properties (binary attributes)and $\mathcal{I}$ is a binary relation defined between $\mathcal{G}$ and $\mathcal{M}$, such that $\mathcal{I} \subseteq \mathcal{G} \times \mathcal{M}$. For $g \in \mathcal{G}$ and $m \in \mathcal{M}$, the notation (g,m) $\in \mathcal{I}$ means that the object g verifies the property m[4].

Suppose that X $\subseteq \mathcal{G}$ *and* Y $\subseteq \mathcal{M}$ *two finite sets. The operators $\varphi(X)$ and $\delta(Y)$ are denoted as follows [4]:*

- $\varphi(X) =$ { m$\in \mathcal{M}$ | g $\in$ X *and (*g,m*)* $\in \mathcal{I}$ }.
- $\delta(Y) =$ { g$\in \mathcal{G}$ | m $\in$ Y *and (*g,m*)* $\in \mathcal{I}$ }.

The operator φ maps the attributes shared by all the elements of X*. The operator δ maps the objects which share the same attributes of the set* Y*. The two operators φ and δ define the Galois Connection between the two sets* X *and* Y*[4].*

Definition 2. ***Closing***
For both sets X *and* Y *mentioned previously, closure operators are defined by:*

- *X" = $\delta \circ \varphi(X)$*
- *Y" = $\varphi \circ \delta(Y)$*

So, a set is closed if it is equal to its closure. Thus, X is closed if X=X" and Y is closed if Y=Y" [4].

Definition 3. ***Formal Concept***
*A formal concept of the context $< \mathcal{G},\mathcal{M},\mathcal{I} >$ is a pair of the form (*X,Y*) for which* X$\subseteq \mathcal{G}$ *is the extent (domain) and* Y$\subseteq \mathcal{M}$ *is the intent (co-domain) with $\varphi(X) = Y$ and $\delta(Y) = X$.*

Definition 4. ***Many-Valued Context***
A Many-valued context allows a different representation of the data than a formal context(mono-valued context). It is a quadruple ($\mathcal{G}$, $\mathcal{M}$, $\mathcal{W}$, $\mathcal{I}$), where $\mathcal{G}$ is a set of objects, $\mathcal{M}$ is a set of attributes, $\mathcal{W}$ is a set of attribute values, and $\mathcal{I}$ is a ternary relation satisfying the condition that the same object-attribute pair can be related to at most one value. An object may have at most one value for each attribute. So, every attribute m *may be treated as a function what maps an object to an attribute value.*

Proposition 1: From a multi-valued context, the δ operator is set by:

$$\delta(AN^* = v_j) = \{\ g \in \mathcal{G} \mid AN^*(g) = v_j\}[10]. \tag{1}$$

Proposition 2: From a multi-valued context, the φ operator is set by:

$$\varphi(B) = \{v_j | \forall g, g \in B,\ \exists AN_l \in AN \mid AN_l(g) = v_j\}[10]. \tag{2}$$

	Outlook	*Temperature*	*Humidity*	*Windy*	*Play*
g1	sunny	hot	high	false	No
g2	sunny	hot	high	true	No
g3	overcast	hot	high	false	Yes
g4	rainy	mild	high	false	Yes
g5	rainy	cool	normal	false	Yes
g6	rainy	cool	normal	true	No
g7	overcast	cool	normal	true	Yes
g8	sunny	mild	high	false	No
g9	sunny	cool	normal	false	Yes
g10	rainy	mild	normal	false	Yes
g11	sunny	mild	normal	true	Yes
g12	overcast	mild	high	true	Yes
g13	overcast	hot	normal	false	Yes
g14	rainy	mild	high	true	No

Table 1. Illustration of the Many-Valued Context (Weather.nominal).

Illustrative example : Considering the training set Weather.nominal described by a set of nominal attributes AN.This data set is selected from UCI Machine Learning Repository [1].

$$AN = \{AN_l \mid l = \{1, .., L\},\ \exists g \in \mathcal{G},\ \exists m \in M,\ AN_l(g) = m\}. \tag{3}$$

Assuming that the chosen attribute AN^* from this many-valued context is 'Outlook' . According to the proposition 1, we extract the associated objects for each value v_j from this attribute. we get these 3 sets of objects ({g1,g2,g8,g9,g11},{g3,g7,g12,g13}, {g4,g5,g6,g10,g14}). According to the proposition 2, we look for the other attributes describing all the extracted objects. In this example, $\varphi(AN^* = v_j)$=({Outlook = sunny },{Outlook = overcast },{Outlook = rainy }). As result, we obtain 3 formal concepts:({g1,g2,g8,g9,g11 }, {Outlook = sunny }),({ g3,g7,g12,g13}, {Outlook = overcast }),({g4,g5,g6,g10,g14 }, {Outlook = rainy }).

3.2 Classifier of Nominal Concepts

Classifier Nominal Concepts(CNC) is a classifier based on Formal Concept Analysis that can handle nominal data. Calculating the formal concept from the multivalue Context by Conceptual Scaling is expensive (RAM consumptions, CPU time). So, CNC calculates it directly using Nomial Scaling. A nominal context is a many-valued context whose attribute values are of the nominal type [2]. From the nominal training instances $\mathcal{G}$ described by L nominal attributes AN, CNC select the attribute AN^* that maximizes the Information Gain[1]. The latter is

[1] http:// archive.ics.uci.edu/ml/

calculated from the Entropy function ($E()$).

$$Gain.Info(AN, \mathcal{G}) = E(\mathcal{G}) - \sum_{j=1}^{V^{an}} \frac{S(Val_j^{an})}{n} E(Val_j^{an}) \tag{4}$$

Once the relevant nominal attribute AN^* is chosen, proposition 1 is used to extract the associated objects for each value v_j from this attribute. The next step is the search for the most relevant value v^* and the objects associated with this value. Then, the attributes checked by this set of objects are selected according to proposition 2 and using the closure operator ($\delta \circ \varphi(AN^* = v^*)$)[4]. So, the pertinent formal concept is constructed from selected objects and attributes ($\delta(AN^* = v^*),\ \delta \circ \varphi(AN^* = v^*)$). The classification rule is obtained by looking for the majority class corresponding to the extent of this concept ($\delta(AN^* = v^*)$). The condition part is formed by the conjunction of the attributes of the intent of the concept ($\delta \circ \varphi(AN^* = v^*)$). The conclusion part is formed by the majority class[10]. In [10], authors proposed the method named CNC Dagging (DNC). DNC is a parallel set method that improves the performance of CNC [18]. The cloudification of the DNC method is one of our future perspectives.

Illustrative example : Considering the same training set (table 1). First, we calculate the Information Gain of each attribute, the attribute "Outlook" is chosen with a Gain Information value of 0.37. It is characterized by 3 different values: "sunny", "overcast" and "rainy". The most relevant value is "rainy" (or "sunny"). According to proposition 1, the associated objects with this value are {g4,g5,g6,g10,g14 }. We use the closure operator with proposition 2 to select the attributes verified by these objects, we get {Outlook = rainy }. So, the relevant concept obtained is ({g4,g5,g6,g10,g14 }, {Outlook = rainy }). The associated majority class is "$Play = Yes$", the following classification rule generated is : "$If\ Outlook = rainy,\ then\ Play = Yes$".

Data: n nominal instances $\mathcal{G} = \{(g_1, y_1), ..., (g_n, y_n)\}$ with labels $y_i \in \mathcal{Y}$.
Result: The classification rule h_{CNC}
begin
From $\mathcal{G}$, determine AN^*: the attribute that maximizes the Information Gain;
From AN^*, determine the most relevant value v^*;
Calculate the associated closure of this relevant value;
Generate the relevant concept;
Define the majority class y^*;
Induce and return h_{CNC}: the classification rule;
end

Algorithm 1: Algorithm of Classifier Nominal Concept[10]

3.3 Distributed CNC : a distributed version of CNC algorithm

The implementation of a classification method using DistributedWekaSpark tool is based on 4 requirements: RDD generation from raw data, creation of headers using RDD, model creation and model evaluation [17]. The transformation of HDFS data into RDD is not sufficient because these RDD objects created by Spark are raw data (string objects) and this type of object is not supported by Weka. So this RDD format created previously must be transformed into a second format that is the format Instances. The second step is to create a header that contains attribute types and names, and other statistics, including minimum and maximum values, to form the ARFF format supported by Weka [17]. The creation of the model and its evaluation is done through the unified framework provided by DistributedWekaSpark, this framework allows the personalized distributed implementation of each classification algorithm [17].

Training phase of Distributed CNC: The master node divides data in parts and distributes the task(code) and data partitions to the slave nodes. A set of partitions is assigned to each slave node, each one applies the CNC method to eatch data partition using the Map function of the MapReduce parallel programming model, and returns the result to the master node. So, we get a list of classifiers. Each time, we apply an aggregation test to the first two classifiers in the list. Two classifiers are aggregable, are two homogeneous classifiers that have the same class. Thus, if they are aggregable, they are replaced directly by a single classifier. Otherwise, an average vote is used to merge these two classifiers. In both cases, each time the first two classifiers are replaced by a single classifier, until a single classifier is obtained at the end.

Data: Dataset in HDFS storage.
Result: The classification rule h_{DCNC}
begin
  Divide the input data into partitions ;
  Distribute training task and data partitions to the slave nodes;
  Map step: create a CNC model for eatch partition;
  Reduce Step: merge the models;
  Return the result model h_{DCNC};
end

Algorithm 2: Distributed CNC: Training Step

Evaluation phase of Distributed CNC: The evaluation phase of the model requires a new MapReduce step. The master node distributes the CNC model formed to the slave nodes. During a new Map phase, each slave node initiates a model evaluation procedure using its set of partitions, and reviews the local evaluation results. The Reduce function produce the final result by aggregating the intermediate results.

Data: h_{DCNC} in HDFS storage.
Result: Evaluation results
begin
  Distribute evaluation tasks to the slave nodes;
  Distribute the trained model h_{DCNC} to the slave nodes;
  Map step: each slave node uses its partitons to evaluate the model;
  Reduce Step: merge the evaluation results;
  Return the merged evaluation results;
end

Algorithm 3: Distributed CNC: Evaluation Step

Illustrative example: We propose to consider a dataset composed of 12 objects as an example, and a cluster composed of 3 nodes: 1 master node and 2 slave nodes. Each node has only 2 cores, so the number of partitions in this case will be 4 (2 * 2), the dataset will be partitioned into 4 partitions, each partition composed of 3 objects. Each slave node applies the CNC classification method on its partitions in parallel. So, we obtain 4 classification rules. Only one classification rule is returned using the Reduce function. In the evaluation step, a copy of the classification rule will be distributed on all slave nodes, where each one evaluates the model on each of its partitions. Finally the evaluation results will be aggregated.

4 Implementation and experimental study

4.1 Implementation

The second step of Cloudification is to create a cluster of virtual machines in Cloud, and implement distributed CNC on this cluster, and conduct an experimental study to compare our results with those of the sequential version. Our architechture contains five Amazon EC2 instances[2], one master node and four slave nodes. We chose the Big Data framework Apache Spark for its speed, and the Big Data Mining tool DistributedWekaSpark. The latter uses the tool Weka [3] as user interface.

Each instance is operated with Linux Ubuntu 16.04, and equipped with a 4 CPU, 16 GB of main memory and Amazon Elastic Block Store (Amazon EBS) storage [4], wich provides persistent block storage volumes for use with Amazon EC2 instances in the AWS cloud. All nodes are configured with Hadoop 2.7.6[5] and Spark 2.3.1[6].

[2] The number of instances/virtual machines for our AWS educationale account is limited to 5
[3] https://www.cs.waikato.ac.nz/ml/weka
[4] https://aws.amazon.com/fr/ebs
[5] https://hadoop.apache.org
[6] https://spark.apache.org

Launching Application with spark-submit: After creating and configuring our cluster, and before starting execution using the spark-submit script, we need to create a jar file which gathers other projects on which our code depends, for that we used Maven [7]. Then we send our Jar file from our local machine to the master node of our cluster, and we upload our data files into HDFS. The last step is to launch our application using the spark-submit script(see in figure 1 and figure 2). This script depends on a set of parameters. The first parameter is the class, which is the entry point for the application. The second parameter is the master URL for the cluster. The third parameter is the deploy-mode(cluster mode or local mode). The fourth parameter is the path to the jar file that include all dependencies. The fifth parameter is the URL for the dataset in HDFS, the rest of the parameters are the application arguments. They are the number of attributes, class index, task(build classifier/evaluation), and the URL for the classification method.

```
>spark-submit  --master  <spark://URL-to-master:7077> --class <path-to-main-class>
  <path-to-DistributedWekaSpark.jar>  -hdfs-dataset-path  <path-to-dataset>
  -num-of-attributes <num> -class-index  <class index>  -task buildClassifier
  -classifier <path-to-CNC-classifier>
```

Fig. 1. Spark-Submit script: build classifier step.

```
>spark-submit  --master  <spark://URL-to-master:7077> --class <path-to-main-class>
  <path-to-DistributedWekaSpark.jar>  -hdfs-dataset-path  <path-to-dataset>
  -num-of-attributes <num> -class-index  <class index>  -task buildClassifierEvaluation
  -classifier <path-to-CNC-classifier>
```

Fig. 2. Spark-Submit script: evaluation step.

4.2 Experimental study

Five data sets with large scale and high dimensionality are used in the experiments as shown in Table 2. The two first data sets from the UCI machine learning repository [8], the three last data sets from the Open Machine Learning repository [9]. They are generated by the Bayesian Network (BNG)[13]. Data sets generated by the Bayesian Netwok are a collection of artificially generated datasets. These datasets have been generated to meet the need for a large heterogeneous set of large datasets [13].

[7] http://maven.apache.org/
[8] http://archive.ics.uci.edu/ml/
[9] http://www.openml.org

Table 2. Datasets characteristics.

Data	Objects	Attributes	Classes	Data size in MB
Letter	20 000	16	26	0.77
Covertype	110 393	55	7	15.25
BNG(kr-vs-kp)	1 000 000	37	2	76.13
BNG(ionosphere)	1 000 000	35	2	200.2
BNG(spambase)	1 000 000	58	2	336.91

To evaluate our approach, we must compare our results with those of the sequential method Classifier Nominal Concepts(CNC), we can classify these results according to two axes of performance: the error rate(see in table 3), and the execution time (see in table 4).

Table 3. Error rate comparison of CNC and Distributed CNC.

Data	*CNC*	*Distributed CNC*
Letter	20,96%	20,96%
Covertype	4.6%	4.6%
BNG(kr-vs-kp)	33.8%	33.8%
BNG(ionosphere)	14.96%	14.96%
BNG(spambase)	39.35%	39.35%

For evaluation, we used the 10-fold cross-validation scheme(for each partition). The experiments were conducted multiple times automatically. So, the error rate in the table 3 is the mean of the error rates of these experiments. The results show that after the implementation of CNC method on cloud, the CNC error rate is not changed. So, we can conclude that the Cloudification of the CNC method does not affect its accuracy performance.

Table 4 presents the result of our work, after comparing the execution time of the two methods, we can notice that the execution time of Distributed CNC is lower than the execution time of CNC method, this can be remarkable from a certain size. We can conclude that Distributed CNC is more efficient than CNC in terms of speed. so, now we conduct experiments on very large data files which is impossible on only one local machine. To show superiority even more, we think about conducting experiment on larger datasets.

5 Conclusion

Distributed Data Mining tools are created to scale the existing data mining algorithms. These tools depend on a Big Data framework used, which is designed

Table 4. Execution time comparison of CNC and Distributed CNC (second).

Data	*CNC*	*Distributed CNC*
Letter	**1,21**	1,37
Covertype	**3,96**	6,4
BNG(kr-vs-kp)	38,53	**29,49**
BNG(ionosphere)	40,4	**28,85**
BNG(spambase)	64,39	**35,09**

to solve the problems of processing of large datasets. These framewoks usually rely on a parallel programming paradigm, often the MapReduce model is used. In this article, we have proposed a distributed version of the classification method based on Formal Concept Analysis. We implemented this release on the Amazon Web Services Cloud by creating a cluster composed of five virtual machines. Preparatory experiments have shown that Distributed CNC is faster than the single-node sequential version(CNC).

So, after the Cloudification of the classification method named CNC, we were be able to overcome the runtime problems and limitations of our material resources. Now, we can use this method with large datasets without worrying about time and without thinking about acquiring other more powerful machines.

Our work allowed us to discover several future perspectives that can be considered in the context of Big Data Mining. In future work, we will conduct experiments on parallel algorithms to improve the efficiency of the use of computing resources. In a second perspective and always in the context of Big Data Mining, we will propose a new method of data distribution on slave nodes, this method will be inspired by the principle of stratified sampling. Also, we will create big data solutions for improving benefits of algorithms based on Formal Concept Analysis.

Acknowledgments

We would like to thank the managers of RosettaHUB(www.rosettahub.com), the platform through which we had access to Amazon Web Services Educate, which allowed us to have all the services of the AWS Cloud.

References

1. Quinlan, J. R.: Induction of decision trees. Journal of Machine Learning, **1**(1), 81–106 (1986)
2. Kuznetsov, S. O.: Mathematical aspects of concept analysis. Journal of Mathematical Sciences, **80**(2),1654–1698 (1996)
3. Fu, Y.: Distributed Data Mining: An Overview. IEEE TCDP newsletter, Springer (2001)

4. Ganter, B., Stumme, G., Wille, R.: Formal concept analysis: foundations and applications. Springer (2005)
5. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Com ACM, pp. 107–113, (2008)
6. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. In Proceedings of the 2Nd USENIX, Conference on Hot Topics in Cloud Computing, HotCloud'10, pp. 10–10, Berkeley, CA, USA, (2010)
7. Zaharia, M., Chowdhury, M., Franklin, M. J., Tathagata, D., Ankur D., Justin Ma., Murphy M., Scott, S., Ion S.: Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, Berkeley, CA, USA, (2010)
8. Han, J., Kamber, M., Pei, J.: Data Mining. Concepts and Techniques, 3rd edn. Elsevier, (2011)
9. White, T.: Hadoop : The definitive guide. O'Reilly Media, Inc, (2012)
10. Meddouri, N., Khoufi, H., Maddouri, M.: Parallel learning and classification for rules based on formal concepts. In 18th International Conference on Knowledge-Based and Intelligent Information Engineering Systems - KES2014 , pp. 358–367, Poland (2014)
11. Liu, X., Iftikhar, N., Xie, X.: Survey of real-time processing systems for big data. In Proceedings of the 18th International Database Engineering and Applications Symposium 2014, pp. 356–361. ACM, (2014)
12. Singh, D., Reddy, C. K.: A survey on platforms for big data analytics. Journal of Big Data, **2**(1):8 (2014)
13. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Algorithm Selection on Data Streams. In Discovery Science. Lecture Notes in Computer Science, vol 8777. Springer, Cham (2014)
14. Shukla, R. K.,Pandey, P., Kumar, V.: Big data frameworks : At a glance. International Journal of Innovations Advancement in Computer Science IJIACS (2015)
15. Al-Jarrah, O.Y., Yoo, P.D., Muhaidat, Karagiannidis, S. G.K., Taha, K.: Efficient machine learning for big data : a review. Big Data Res, 87–93, (2015)
16. Landset, S., Khoshgoftaar, T. M., Richter, A. N., Hasanin, T.: A survey of open source tools for machine learning with big data in the hadoop ecosystem. Journal of Big Data, **2**(1):1 (2015)
17. Koliopoulos, A., Yiapanis, P., Tekiner, F., Nenadic, G., Keane, J.: A parallel distributed weka framework for big data mining using spark. In IEEE International Congress on Big Data 2015,CA USA (2015)
Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. In Proceedings of the 2Nd USENIX, Conference on Hot Topics in Cloud Computing, HotCloud'10, pp. 10–10, Berkeley, CA, USA, (2010)
18. Trabelsi, M., Meddouri, N., Maddouri, M.: A new feature selection method for nominal classifier based on Formal Concept Analysis. In: Proceedings of the 21th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2017 [B]), Procedia Computer Science, Vol. 112, pp. 186-194. Elsevier, Marseille, France (2017)
19. Inoubli, W., Aridhi, S., Mezni, H., Maddouri, M., Mephu Nguifo, E.: An experimental survey on big data frameworks, Future Generation Computer Systems, (2018)