2015 IEEE 16th International Conference on Information Reuse and Integration

A Multi-Dimensional Comparison of Toolkits for Machine Learning with Big Data

Aaron N. Richter, Taghi M. Khoshgoftaar, Sara Landset, Tawfiq Hasanin
Florida Atlantic University
{arichter, khoshgof, slandset, thasanin2013}@fau.edu

***Abstract*—Big data is a big business, and effective modeling of this data is key. This paper provides a comprehensive multi-dimensional analysis of various open source tools for machine learning with big data. An evaluation standard is proposed along with detailed comparisons of the frameworks discussed, with regard to algorithm availability, scalability, speed, and more. The major tools profiled are Mahout, MLlib, H_2O, and SAMOA, along with the big data processing engines they utilize, including Hadoop MapReduce, Apache Spark, and Apache Storm. There is not yet one framework that "does it all", but this paper provides insight into each tool's strengths and weaknesses along with guidance on tool choice for specific needs.**

***Keywords*—big data, machine learning, data mining, Hadoop, Mahout, Spark**

I. Introduction

Data is permeating our culture and industries, and techniques are needed to harness the power of this data. Machine learning (ML) and data mining algorithms have the capability to provide key insights and predictive modeling for various kinds of data spanning many industries and sectors. Historically, these algorithms have primarily been studied and utilized in research settings. With the recent explosion of data, along with the tools to process it, machine learning has seen a growing wave of enterprise and research applications.

Additionally, the explosion of big data provides new challenges for data processing, let alone machine learning. Recent years have seen efforts to scale data processing methods across large parallel computing clusters. Consequently, traditional machine learning algorithms, developed to work on "small" data, cannot hold up when applied to these massive datasets. New methods and tools are needed to dig deep into the big data at hand.

This paper profiles several major distributed machine learning frameworks for big data. The goal is to provide engineers and researchers familiar with small-scale machine learning tools a comprehensive overview of options available for modeling big data. Various criteria are presented to evaluate each toolkit, such as processing speed, fault tolerance, usability, scalability, and extensibility. Algorithm availability is analyzed across each library, and guidance is given for different machine learning scenarios.

In Section II we offer various selection criteria and considerations when evaluating ML toolkits. In Sections III through VI we discuss prominent open source frameworks for machine learning with big data, and in Section VII we briefly mention proprietary tools along with up-and-coming open source tools. Finally, we offer a comprehensive comparison of these tools in Section VIII.

II. Selecting Toolkits

Selecting a suite of tools for any job can be a daunting task, and the choice of machine learning tools for big data is no exception. The new buzz around big data, data analytics, data mining, and machine learning makes it difficult to identify useful tools. There are different needs across different application domains, and there may be platform and programming language restrictions due to existing infrastructure. Additionally, algorithm availability is essential for those wishing to utilize ML with big data. These considerations and more are evaluated for each tool discussed, along with examples of industry adoption and research projects using the tool. Table I provides an overview of the tools that are analyzed. Each tool is presented with its processing platform and available algorithms in Sections III through VI. While performance considerations are noted in each section, a detailed comparison of the tools using these measures is provided in Section VIII.

A. Processing Paradigms

Traditional machine learning methods perform offline, or batch, model building on a training dataset before scoring new test data. This is helpful when there is a mass of historical data ready to be leveraged for a machine learning task. Sometimes historical data is not readily available, and streaming, or online machine learning is required. In online learning, ML models are built and evaluated as the data comes into the system, requiring efficient algorithms that can process data at the speed that it is coming in. Batch learning, however, requires iterations over the entire dataset, which may not be time efficient for big data. These two paradigms are referred to as "batch" and "streaming" in this paper.

B. Algorithms

Several families of machine learning tasks are covered by the tools discussed; below is a brief introduction to each group.

1) Classification: Classification is a supervised learning task that seeks to discriminate a class label from a set of inputs. Examples include labeling email as spam, predicting patient response to a drug [6], and sentiment analysis of text data [23]. Commonly used algorithms for classification include Logistic Regression, Decision Trees, Support Vector Machines, k-Nearest Neighbor, and ensembles of these methods.

978-1-4673-6656-4/15 $31.00 © 2015 IEEE
DOI 10.1109/IRI.2015.12

2) Regression: Regression is also a supervised learning task that fits a model to a group of data, allowing for numeric predictions [15]. Traditional algorithms for regression include Linear Regression and tree models.

3) Recommendation: Recommendation engines predict meaningful relationships between entities. Recommending books, music, movies, and other products to users based on their relationships with other users is a prime example of these systems. Collaborative Filtering (CF) methods are widely used to create recommender systems [30].

4) Clustering: Clustering is an unsupervised learning task that groups elements in a population together by examining their various features [35]. Common clustering algorithms are k-Means Clustering and Expectation-Maximization (EM) Clustering.

5) Deep Learning: Deep learning utilizes Artificial Neural Networks that simulate the human brain to model mathematical relationships between data points. Deep learning is a promising field of machine learning and artificial intelligence [21], and has been used to detect objects in videos and beat human players in video games [18][20]. While deep learning is used for tasks like classification and regression, we consider it separate from the other categories to illustrate coverage by the various tools.

6) Association Rules: Association rule models, such as Frequent Pattern Growth, detect relationships between instances in a dataset. These rules can be used to impute missing features or select sets of items that are related to each other [31].

7) Dimensionality Reduction: Dimensionality reduction minimizes data size by combining, transforming, and removing features. Notable algorithms for dimensionality reduction are Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Feature selection is a technique in dimensionality reduction that chooses specific features based on a statistic or evaluation of how useful the feature is to the overall modeling task [7].

C. Platform & Performance

The Hadoop ecosystem is often the go-to platform for big data processing, and all the tools covered in this paper are integrated with Hadoop, being able to access and store data in the Hadoop Distributed File System (HDFS). Many of these machine learning libraries also support other big data platforms, storage systems, and processing engines. Platform options, along with the following performance considerations, will be presented for each tool.

1) Speed: Speed refers to the processing and execution time needed to train and make predictions from models. The speed of ML toolkits is often tied to their underlying processing engines, and the bulk of the processing time for ML workflows occurs during model building. While speed can be an important factor for certain situations, in scenarios where models do not have to be updated frequently, a slower algorithm or tool may be used without consequence. Once a model is built, it is relatively fast and less computationally intensive to score new instances.

2) Fault Tolerance: Fault tolerance is important to consider in distributed systems, as a single node failure can have an adverse effect on the entire workflow. Various tools have different fault tolerance mechanisms, from disk caching to data re-computation.

3) Usability: Usability covers the overall process of using the tool for model building and evaluation. Ease of use, programming language interfaces, documentation, and user/developer community all contribute to the usability of a toolkit.

4) Scalability: Data sizes, with regard to number of instances, features, and complexity of the data should be considered when deciding how "big" the data is. A framework is scalable when it can meet ML needs as data size grows.

5) Extensibility: Machine learning models often have to be tuned or even extended beyond their existing capabilities to be applied to a specific domain or dataset. The developer community, languages models are built in (not interface languages), and parameter tunings options are all criteria for determining the extensibility of a toolkit.

TABLE I. BIG DATA ML TOOLS COVERED

Toolkit	**Current Version (as of 06/24/15)**	**Processing Paradigms**	**Processing Engines**
Mahout	0.10.0	Batch	MapReduce, Spark, H_2O
MLlib	1.4.0	Batch, Streaming	Spark
H_2O	3.0.0.22	Batch	H_2O
SAMOA	0.2.0	Streaming	Storm, Samza, S4

III. MAHOUT

Mahout is the oldest tool for distributed machine learning that is discussed in this paper. The library was initially built upon Hadoop and MapReduce [22] as a batch machine learning framework and has seen widespread enterprise implementation[1] and usage in academia, as will be presented in the following section. While the MapReduce algorithms require Hadoop, several components of Mahout (such as the non-distributed collaborative filtering engine, formerly a separate project called Taste[2]) run on single machines that do not require Hadoop. In April 2015, Mahout 0.10 was released, signaling a new direction for the project. It introduced Samsara, an environment for math operations built on Scala, which allows users to customize and build their own algorithm implementations. Additionally, the project moved away from MapReduce in favor of in-memory computation via Spark and H_2O, even including an interactive shell for distributed computations in Spark. Compatibility with the Flink processing engine is also in development. Spark and H_2O will be introduced and explained in Sections IV and V.

A. MapReduce

Version 0.9 and below offer ML algorithms implemented on MapReduce, which are known to be slow due to the nature of MapReduce on-disk processing. MapReduce writes all intermediate computations to disk, which allows for robust fault-tolerance, but at a tradeoff of speed. This allows algorithms

[1]http://mahout.apache.org/general/powered-by-mahout.html

[2]https://mahout.apache.org/users/recommender/recommender-documentation.html

to scale to very large datasets, but the overhead becomes significant when using smaller datasets. Additionally, writing pure MapReduce programs takes very skilled programming to translate tasks into concepts involving mappers and reducers. Mahout version 0.10 seeks to solve many of these issues by using the in-memory processing engines of Spark and H_2O, but its effectiveness is not yet known since it was released very recently.

B. *Algorithms*

Mahout offers a wide variety of machine learning algorithms implemented in Java and Scala for use on a single machine or a cluster via MapReduce[3], mainly focusing on classification, clustering, and collaborative filtering.

Classification is possible through Logistic Regression, Naïve Bayes, Random Forest, Hidden Markov Models, and Multilayer Perceptron implementations, though only Naïve Bayes and Random Forest are available for parallelized learning at scale. The Naïve Bayes model is based on [24], though it also offers a Complementary Naïve Bayes learner for situations where Bayes' independence assumptions do not hold up. Mahout 0.10 includes in-memory implementations of the Naïve Bayes models using Spark, though we have not seen any studies comparing it to the MapReduce algorithms. Mahout's Random Forest has seen many applications due to its accuracy and speed (distribution is easy as tree building can be spread across different compute nodes) [28]. Several studies have used this model in the field of bioinformatics and healthcare [16][19][36].

Mahout's collaborative filtering tools are some of the best-known algorithms for recommendation systems. Distributed MapReduce algorithms are available for user-based recommendations and variations of matrix factorization. There are many options for computing similarity, such as Pearson correlation coefficient, Euclidean distance, cosine similarity, Tanimoto coefficient, and log-likelihood. Mahout 0.10 introduced both item and user-based collaborative filtering, along with row similarity calculations, implemented on Spark. In addition to the algorithms, Mahout provides for hold-out testing when feedback from actual users is not readily available [22]. Sparks et al. compared the Mahout MapReduce implementations to collaborative filtering techniques in other frameworks and found that the Mahout algorithms require more code and performed slower than the other frameworks [29]. The team behind the research-sharing site Mendeley found that tweaking the parameters of Mahout's MapReduce collaborative filtering algorithms produced better results in less time [12].

Clustering in parallel is available via k-Means, Fuzzy k-Means, Streaming k-Means and Spectral Clustering, as well as topic modeling via Latent Dirichlet Allocation (LDA). Esteves et al. [8] found that the k-Means algorithm scales well for large datasets, while Esteves and Rong [9] compared the performance and efficiency of fuzzy and traditional k-Means algorithms by clustering documents from Wikipedia. There was high variation in many of the different experiments, but they concluded that the traditional algorithm provided more meaningful results than the fuzzy implementation for noisy datasets. The creator of the Streaming k-Means implementation, Dan Filimon, found the Streaming k-Means algorithm to be up to 8 times faster than the traditional algorithm [10]. Gao et al., after adding local sensitivity hashing in preprocessing, found Mahout's Spectral Clustering algorithm to run significantly faster than other implementations [11].

Mahout offers several algorithms for dimensionality reduction with MapReduce, and Mahout-Samsara offers the ability to use Scala's built-in reduction methods as well. Singular Value Decomposition (SVD), Stochastic SVD, Principal Component Analysis, and QR Decomposition are available for MapReduce as well as Spark and H_2O by using the new Scala & Spark bindings in version 0.10.

IV. MLlib

MLlib is a machine learning library that is shipped with Spark, an in-memory distributed data processing engine that has gained rapid popularity and adoption for big data uses. Spark supports both batch and stream processing, and MLlib can be used to learn from data using both paradigms.

A. *Spark*

Spark was created at the University of California, Berkeley, with the goal of solving many problems inherent with the Hadoop MapReduce processing architecture [33]. The project introduced the concept of Resilient Distributed Datasets (RDD) which store and process data in-memory across nodes in a cluster [32]. Fault tolerance is provided by creating a Directed Acyclic Graph (DAG) of operations and evaluating actions in a lazy manner. When a node fails, results are re-computed using the actions stored in the DAG [13]. This significantly reduces the number of read/write operations typically used in MapReduce programs, resulting in greater time efficiency. Spark set the Sort Benchmark[4] record in October 2014, sorting 100TB of data in 23 minutes using 206 nodes. The previous record, held by Hadoop MapReduce, sorted the data in 72 minutes with 2,100 nodes. Additionally, Spark sorted one petabyte in 234 minutes with 190 nodes[5], though not part of the official competition.

Spark is compatible with Hadoop elements such as HDFS and Hive, and can run on top of a Hadoop cluster using YARN. Additionally, the package can be downloaded and run on a single machine, and the same code can be deployed to a Spark standalone or Mesos cluster. Spark is written in Scala and uses the Java Virtual Machine (JVM), but the engine is accessible through Scala, Java, SQL, Python, and R. MLlib is also supported in these languages, allowing for adoption by many users that are familiar with different languages. Additionally, MLlib comes shipped with Spark, alleviating the versioning and deployment issues seen with other ML libraries [2].

B. *Spark Streaming*

Spark offers data stream processing through Spark Streaming, which groups incoming data into micro-batches for processing by the Spark engine [34]. This is accomplished through a Discretized Stream (DStream), which represents a set of

[3]http://mahout.apache.org/users/basics/algorithms.html

[4]http://sortbenchmark.org/

[5]https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html

RDDs. This makes the processing options for batch Spark jobs available for streaming as well, allowing MLlib and GraphX (Spark's graph processing engine) to operate with streaming data.

C. Algorithms

MLlib was developed to allow users to extend and create their own algorithms using the library. The spark.ml package provides a unified API for building, extending, and applying machine learning algorithms with MLlib[6]. MLlib also supports many mathematical and statistical methods that are useful for data preprocessing and model evaluation. Many models now come out-of-the-box with MLlib, including algorithms for classification, regression, recommendation, clustering, and dimensionality reduction. Spark, and subsequently MLlib, is rather young compared to Mahout, so there are not as many academic studies applying and evaluating these algorithms.

Classification with MLlib is available through Support Vector Machines (SVM), Logistic Regression, Naïve Bayes, and decision tree methods (including ensembles via Random Forest and Gradient-Boosted Trees). Regression is available through Linear Regression (linear least squares, Lasso, and ridge regression), Streaming Linear Regression (using ordinary least squares), Isotonic Regression, and the same tree models used for classification. MLlib offers collaborative filtering via Alternating Least Squares (ALS), which is used in the recommendation engines of OpenTable and Spotify[7].

MLlib has several algorithms for clustering, including traditional and streaming k-Means, Gaussian Mixture, Power Iteration Clustering (PIC), and topic modeling with Latent Dirichlet allocation (LDA). Dimensionality reduction can be accomplished with Singular Value Decomposition (SVD) and Principal Component Analysis (PCA). MLlib also offers feature selection using the Chi-Squared statistic.

V. H_2O

H_2O, while open source, was created and is supported by a private company of the same name (formerly known as 0xdata). The tool operates much like a product rather than a project, which has its benefits and drawbacks. Benefits include a consistent support team, including paid enterprise support options. The product has not seen much use in academia, however, despite being about the same age as Spark. While H_2O itself performs batch training of ML models, the project can make online predictions using Storm[8]. Distributed computation is accomplished using Distributed Fork/Join, a divide-and-conquer technique, which parallelizes jobs across data nodes for in-memory computation and then combines the results. It should be noted that H_2O's data processing tools can be used separately from its machine learning capabilities, much like the distinction between Spark's Core API and MLlib.

H_2O is easy to download and test locally, and it offers a Web Graphical User Interface (GUI) for building and evaluating models. Models can also be customized and deployed using Java, R, Python, and Scala. There is seamless integration with R and RStudio to appeal to those experienced with these tools. H_2O Flow[9], part of H_2O project, is an interactive web-based notebook for manipulating and learning from data using a hybrid command-line and point-and-click approach. It uses Coffeescript[10], a scripting language that compiles into Javascript, but is also compatible with R, Python, and Scala. H_2O even has macros for training models from Excel. The H_2O team claims that the product's integration with Spark, Sparkling Water, is the "killer app for Spark"[11]. It offers an H2ORDD, which can be converted to and from native Spark RDDs, allowing Spark's data manipulation capabilities to be used in tandem with H_2O's machine learning models, some of which have capabilities beyond those of MLlib.

A. Algorithms

H_2O offers a selection of models from various families, along with distributed deep learning algorithms. In addition to machine learning models for regression, classification, clustering, and dimensionality reduction, it offers tools for profiling data, creating features, and model validation and scoring plans with various statistical measures.

The project provides Generalized Linear Model (GLM) algorithms for regression using Gaussian (normal), Poisson, Gamma, and Tweedie (zero-inflated Poisson) distributions, and a binomial GLM for classification (Logistic Regression). Gradient Boosted Models (GBM), also known as Gradient Boosted Trees, are offered for classification and regression tasks. Kejela et al. found the GBM model to have better accuracy than the GLM model for regression, while both models were comparable for classification [14]. Naïve Bayes and Random Forest models are also available for classification.

Clustering is possible via the k-Means algorithm, with several choices for initial centroid selection. Dimensionality reduction is available with Principal Component Analysis (PCA). In addition, time-to-event analysis is implemented using the Cox Proportional Hazards statistical model [4].

Deep Learning in H_2O is implemented with multilayer feed-forward neural networks trained by Stochastic Gradient Descent (SGD) with back-propagation. These models are highly tunable, with options for choosing activation functions, number and size of hidden layers, number of iterations, loss function, regularization methods, and more. These models can be used for large-scale distributed unsupervised learning and anomaly detection.

VI. SAMOA

SAMOA stands for Scalable Advanced Massive Online Analysis, and was developed at Yahoo! Labs Barcelona specifically for machine learning from data streams. The project has been in Apache project incubation since late 2014. SAMOA provides real-time online building and evaluation of machine learning models with data streams. According to its creators, it is the "Mahout for streaming".

[6]https://spark.apache.org/docs/latest/ml-guide.html

[7]https://cwiki.apache.org/confluence/display/SPARK/Powered+By+Spark

[8]not to be confused with online model building, which H_2O does not support

[9]http://h2o.ai/product/flow/

[10]http://coffeescript.org/

[11]http://h2o.ai/product/sparkling-water/

SAMOA accepts input streams from Storm[12], S4[13], and Samza[14], all of which are Apache projects for processing big data streams. Model building can be also be accomplished with local files for development and testing. The algorithms implemented in SAMOA are similar to some found in MOA[15], the Massive Online Analysis ML project from the creators of Weka, and native MOA models can be used in SAMOA with a plugin[16]. Models can be created and extended through SAMOA's Java API.

A. Algorithms

The model selection in SAMOA is rather sparse, due to the age of the project and nature of data streaming algorithms. The Vertical Hoeffding Tree (VHT), or Very Fast Decision (VFDT), is implemented for classification. The Adaptive Model Rules (AMRules) algorithm is used for regression in SAMOA with implementations of Vertical and Horizontal Adaptive Model Rules Regressors. These classification and regression algorithms are used with Prequential Evaluation to perform online model training and testing.

Clustering is accomplished via an implementation of the CluStream [1] algorithm. Ensembles of various models, including those from MOA using the SAMOA-MOA plugin, can be accomplished with Bagging and Boosting. Frequent Itemset Mining (association rule mining) is also available through an implementation of the PARMA [25] algorithm with a time biased sampling approach.

Though relatively young, several studies have used or extended SAMOA for various purposes. The clustering algorithm implemented in the project was developed by Severien in [27]. Romsaiyud built and evaluted a topic modeling system for streaming text data, and found SAMOA to have much higher throughput than MOA with the same algorithm [26]. Rahnama used VHT with Bagging as part of a system to perform large-scale real-time analysis of Twitter streams [17]. Mauro and Sarno used VHT with Preqential Evaluation for internet traffic detection with Skype network traffic [5].

VII. Other tools

We evaluated four major open source tools for machine learning with big data due to their maturity and usage in the industry and academia. There are other tools that exist to learn from big data that were left out of the deeper analysis. Flink-ML[17] is a new ML library for the Flink (formerly known as Stratosphere) streaming and iterative big data processing engine [3]. Oryx[18] is another tool originally built on MapReduce, but now offers ML on top of Spark and Kafka. Vowpal Wabbit[19] is an open source tool for big data ML developed by Yahoo! and Microsoft Research. The Weka[20] tookit, while not built for distributed machine learning, offers connectors for

[12]https://storm.apache.org/

[13]http://incubator.apache.org/s4/

[14]http://samza.apache.org/

[15]http://moa.cms.waikato.ac.nz/

[16]https://github.com/samoa-moa/samoa-moa

[17]https://github.com/apache/flink/tree/master/flink-staging/flink-ml

[18]https://github.com/OryxProject/oryx

[19]http://hunch.net/˜vw/

[20]http://www.cs.waikato.ac.nz/˜ml/weka/

TABLE II. Detailed Comparison of ML Tools for Big Data

	Mahout MapReduce	**Mahout Samsara**	**MLlib**	**H2O**	**SAMOA**
PROCESSING PARADIGM					
Batch	✓	✓	✓	✓	
Streaming			✓		✓
PROCESSING ENGINES					
MapReduce	✓				
Spark		✓	✓	✓	
H2O		✓		✓	
Storm					✓
S4					✓
Samza					✓
INTERFACE					
Java	✓	✓	✓	✓	✓
Scala		✓	✓	✓	
Python			✓	✓	
R			✓	✓	
Other Languages				✓	
Graphical User Interface				✓	
PERFORMANCE*					
Speed	1	4	4	4	5
Fault Tolerance	5	4	4	3	1
Usability	1	3	4	5	3
Scalability	4	3	3	3	5
Extensibility	1	4	5	5	2
ALGORITHMS					
Classification					
Naïve Bayes	✓	✓	✓	✓	
Random Forest	✓		✓	✓	
Support Vector Machine			✓		
Logistic Regression			✓	✓	
Gradient Boosted Trees			✓	✓	
Vertical Hoeffding Tree					✓
Regression					
Linear Regression			✓	✓	
Streaming Linear Regression			✓		
Generalized Linear Models				✓	
Random Forest			✓	✓	
Gradient Boosted Trees			✓	✓	
Adaptive Model Rules					✓
Recommendation					
User-Based CF		✓			
Item-Based CF	✓	✓			
CF with Alternating Least Squares	✓		✓		
Clustering					
k-Means	✓		✓	✓	
Fuzzy k-Means	✓				
Streaming k-Means	✓		✓		
Spectral Clustering	✓				
Latent Dirichlet Allocation	✓		✓		
Gaussian Mixture			✓		
Power Iteration Clustering			✓		
CluStream					✓
Association					
Frequent Itemset Mining					✓
FP-Growth	✓		✓		
Deep Learning					
Artificial Neural Networks				✓	
Dimensionality Reduction					
Singular Value Decomposition	✓	✓	✓		
Stochastic SVD	✓	✓			
Principal Component Analysis	✓	✓	✓	✓	
QR Decomposition	✓	✓			
Feature Selection					
Chi-Squared			✓		

*On scale from 1-5; 1 being the worst, 5 being the best.

Hadoop and Spark to distribute some of its native algorithms across a cluster. Deeplearning4j[21] is an open source distributed deep learning tool for Java. Proprietary tools for machine learning with big data include Dato[22] (formerly GraphLab) and Skytree Infinity[23].

VIII. Comparison

A detailed comparison of the tools discussed in this paper is provided in Table II. The scores for each element in the performance category are relative rankings based on available literature and online documentation. Some tools have identical rankings, meaning they are tied or nearly equivalent for that consideration. These are rankings that we have applied based on comprehensive literature review, not our own experimental results. Due to the fundamental differences between Mahout version 0.9 (MapReduce) and Mahout version 0.10 (Samsara), it is split into two categories for the analysis. Mahout's non-distributed algorithms are not included.

A. Processing Paradigm

Both versions of Mahout, along with H_2O, only support batch model building. Note that H_2O does support live predictions with Storm integration, but cannot train the models online. SAMOA, on the other hand, only supports online learning, as that is the goal of the project. MLlib is the only tool that supports both batch and stream model building, though it only has two online algorithms: Streaming k-Means and Streaming Linear Regression.

B. Platform & Performance

Figure 1 illustrates the performance considerations, outlined in Section II, for each framework. Due to the nature of the Hadoop MapReduce system, Mahout with MapReduce has high marks for both scalability and fault tolerance, but is the slowest tool on the list. Additionally, the MapReduce algorithm implementations are no longer supported, warranting its extensibility score. Mahout Samsara, however, improves on many of the problems of its ancestor. Though very young, it shows promise for both speed and extensibility, being built in Scala and using in-memory computation engines.

MLlib and H_2O have several similar marks, due to the nature of their in-memory processing engines. Both have high extensibility ratings, due to their packages supporting ML algorithm development and deployment. Additionally, MLlib has a growing user and developer base, and both are largely supported by for-profit companies (Databricks[24], founded by the creators of Spark, is a large supporter of the project). MLlib's models are written in Scala, while H_2O's models are written in Java. H_2O is the leader in usability, as it is the only tool that comes with a GUI and has the largest base of compatible languages. MLlib and H_2O receive lower scalability marks relative to Mahout MapReduce and SAMOA, since problems may arise when datasets are too large to fit in memory.

[21]http://deeplearning4j.org/
[22]https://dato.com/
[23]http://www.skytree.net/products/skytree-infinity/
[24]http://databricks.com/

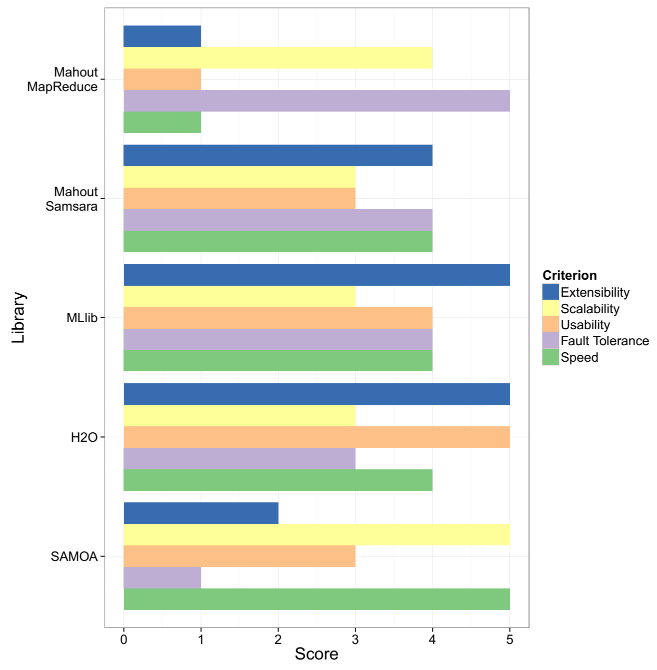

Fig. 1. Performance considerations for each tool.

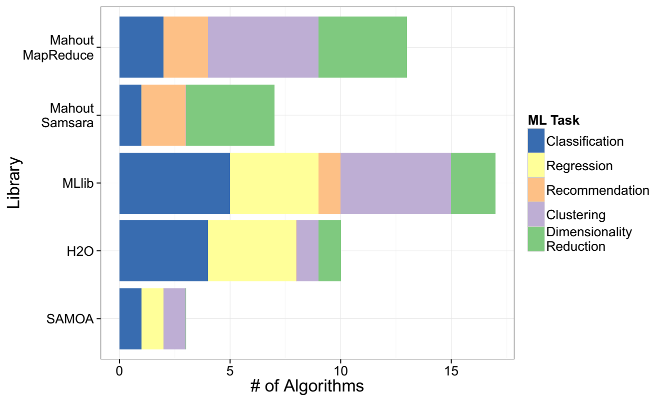

Fig. 2. Machine learning algorithm implementations for each tool.

SAMOA is fundamentally different than the other tools on this list, since it is only designed for learning from streaming data. This affords it high scores for scalability and speed, as its models handle incoming data immediately rather than having to compute on large datasets in batch. Consequently, it has a low fault tolerance rank, and the extensibility is limited since ML pipeline support functions are lacking. The project is still relatively new, and there is much future work to be done to extend its functionality and abilities.

C. Algorithms

Figure 2 outlines available models across various algorithm families. Refer to Table II for a detailed breakdown of individual algorithms. Although not included in the figure, H_2O is the only tool to offer deep learning and MLLib is the only tool to offer feature selection. Additionally, Mahout MapReduce, MLlib, and SAMOA each include a single algorithm for association mining.

MLlib and Mahout MapReduce offer the largest coverage for ML algorithms, though the Mahout MapReduce algorithms are no longer supported as development has moved to in-memory algorithm implementations. Both versions of Mahout offer the most options (and same algorithms) for dimensionality reduction. MLlib and H_2O both offer large selections of algorithms for classification and regression, largely due to their decision tree algorithms (Random Forest and GBT can be used for classification and regression). H_2O, unlike MLlib, has very flexible Generalized Linear Models. Mahout is well-known for its collaborative filtering tools for recommendation, and MLlib implements one CF algorithm, while H_2O and SAMOA do not support recommenders (though CF is on the H_2O algorithm roadmap[25]). Mahout Samsara and SAMOA are both lacking in algorithm implementations, but many more algorithms are expected as the projects mature. SAMOA's algorithms are not offered by any other tool on the list, as learning from data streams requires fundamentally different processing techniques than for batch.

IX. Conclusion

This paper provides an in-depth look at four prominent open source frameworks for machine learning with big data. Though in various maturity stages, each project has its own strengths and weaknesses. Therefore, an individual or team must carefully evaluate their desired ML task and choose a toolkit accordingly.

The choice of tools will largely depend on the desired application domain and existing infrastructure. For example, e-commerce or social networking applications should consider Mahout and MLlib for their recommendation algorithms. Domains such as social media or sensor networks may require real-time model building and evaluation, which would require MLlib or SAMOA. Government and healthcare organizations often produce disparate datasets that require extensive data cleaning or hybrid batch and streaming implementations, warranting the use of Spark/MLlib or H_2O.

MLlib and H_2O currently offer the most comprehensive and readily available algorithm implementations, along with tools for task pipelining and data manipulation. H_2O has a web GUI, and MLlib a very large support community. Many groups will find either one of these tools sufficient for their ML needs, and they can also be used together with the Sparkling Water integration. SAMOA caters to specific streaming ML needs, and Mahout has the largest selection of collaborative filtering algorithms. Mahout should also be carefully watched as it is heading in a new direction with in-memory computing.

Acknowledgment

The authors gratefully acknowledge partial support by the National Science Foundation, under grant number CNS-1427536. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[25]http://docs.0xdata.com/resources/algoroadmap.html

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[2] M. Alber, "Big Data and Machine Learning: A Case Study with Bump Boost," Ph.D. dissertation, 2014.

[3] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke, "The Stratosphere platform for big data analytics," *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 23, no. 6, pp. 939–964, May 2014.

[4] P. K. Andersen and R. D. Gill, "Cox's regression model for counting processes: a large sample study," *The annals of statistics*, pp. 1100–1120, 1982.

[5] M. Di Mauro and C. Di Sarno, "A framework for Internet data real-time processing: A machine-learning approach," *2014 International Carnahan Conference on Security Technology (ICCST)*, Oct. 2014.

[6] D. Dittman, T. M. Khoshgoftaar, R. Wald, and A. Napolitano, "Random forest: A reliable tool for patient response prediction," in *Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on*. IEEE, 2011, pp. 289–296.

[7] D. J. Dittman, T. M. Khoshgoftaar, R. Wald, and J. Van Hulse, "Feature selection algorithms for mining high dimensional dna microarray data," in *Handbook of Data Intensive Computing*. Springer New York, 2011, pp. 685–710.

[8] R. M. Esteves, R. Pais, and C. Rong, "K-means Clustering in the Cloud - A Mahout Test," in *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*. Ieee, Mar. 2011, pp. 514–519.

[9] R. M. Esteves and C. Rong, "Using Mahout for Clustering Wikipedia's Latest Articles: A Comparison between K-means and Fuzzy C-means in the Cloud," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. Athens: IEEE, Nov. 2011, pp. 565–569.

[10] D. Filimon, "Clustering of Real-time Data at Scale," in *Berlin Buzzwords*, 2013.

[11] F. Gao, W. Abd-Almageed, and M. Hefeeda, "Distributed approximate spectral clustering for large-scale datasets," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing - (HPDC '12)*. New York, New York, USA: ACM Press, 2012, pp. 223–234.

[12] K. Jack, "Mahout becomes a researcher: Large Scale Recommendations at Mendeley," in *Big Data Week, Hadoop User Group UK*, London, 2012.

[13] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, *Learning Spark: Lightning-Fast Big Data Analysis*. " O'Reilly Media, Inc.", 2015.

[14] G. Kejela, R. M. Esteves, and C. Rong, "Predictive Analytics of Sensor Data Using Distributed Machine Learning Techniques," in *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. Ieee, Dec. 2014, pp. 626–631.

[15] T. M. Khoshgoftaar and N. Seliya, "Fault prediction modeling for software quality estimation: Comparing commonly used techniques," *Empirical Software Engineering*, vol. 8, no. 3, pp. 255–283, 2003.

[16] K. D. Ko, T. El-Ghazawi, D. Kim, and H. Morizono, "Predicting the severity of motor neuron disease progression using electronic health record data with a cloud computing Big Data approach," in *Computational Intelligence in Bioinformatics and Computational Biology, 2014 IEEE Conference on*. IEEE, May 2014.

[17] N. Kourtellis, G. D. F. Morales, F. Bonchi, G. De, and F. Morales, "Scalable Online Betweenness Centrality in Evolving Graphs," *arXiv preprint arXiv:1401.6981*, p. 16, 2014.

[18] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[19] L. Li, S. Bagheri, H. Goote, A. Hasan, and G. Hazard, "Risk Adjustment of Patient Expenditures: A Big Data Analytics Approach," in *2013 IEEE International Conference on Big Data*. IEEE, Oct. 2013, pp. 12–14.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[21] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, , and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, pp. 1–21, 2015. [Online]. Available: http://link.springer.com/article/10.1186%2Fs40537-014-0007-7

[22] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Manning, 2011.

[23] J. D. Prusa, T. M. Khoshgoftaar, and D. J. Dittman, "Impact of feature selection techniques for tweet sentiment classification," in *Proceedings of the 28th International FLAIRS conference*, May 2015, pp. 299–304.

[24] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the Poor Assumptions of Naive Bayes Text Classifiers," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, no. 1973, 2003.

[25] M. Riondato, J. a. DeBrabant, R. Fonseca, and E. Upfal, "PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce," *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, pp. 85–94, 2012.

[26] W. Romsaiyud, "Automatic Extraction of Topics on Big Data Streams Through Scalable Advanced Analysis," in *2014 International Computer Science and Engineering Conference (ICSEC)*, 2014, pp. 255–260.

[27] A. L. Severien, "Scalable Distributed Real-Time Clustering for Big Data Streams," Ph.D. dissertation, Universitat Politècnica de Catalunya, 2013.

[28] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests," *Information Sciences*, vol. 278, pp. 488–497, Sep. 2014.

[29] E. R. Sparks, A. Talwalkar, V. Smith, J. Kottalam, X. Pan, J. Gonzalez, M. J. Franklin, M. I. Jordan, and T. Kraska, "MLI: An API for Distributed Machine Learning," in *2013 IEEE 13th International Conference on Data Mining*. Ieee, Dec. 2013, pp. 1187–1192.

[30] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[31] J. Van Hulse and T. M. Khoshgoftaar, "Class noise detection using frequent itemsets," *Intelligent Data Analysis*, vol. 10, no. 6, pp. 487–507, 2006.

[32] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

[33] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets."

[34] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, "Discretized Streams: A Fault-Tolerant Model for Scalable Stream Processing," *University of California at Berkeley Technical Report No. UCB/EECS-2012-259*, 2012.

[35] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Clustering-based network intrusion detection," *International Journal of reliability, Quality and safety Engineering*, vol. 14, no. 02, pp. 169–187, 2007.

[36] K. Zolfaghar, N. Meadem, A. Teredesai, S. B. Roy, S.-C. Chin, and B. Muckian, "Big data solutions for predicting risk-of-readmission for congestive heart failure patients," in *2013 IEEE International Conference on Big Data*. Ieee, Oct. 2013, pp. 64–71.