Selection of Machine Learning Algorithms and Libraries for Big Data Applications

Boris Zibitsker, PhD, BEZNext; Alex Lupersolsky, PhD, BEZNext; Dominique Heger, PhD, Data Analytica; Yuri Balasanov, PhD, University of Chicago; Students of University of Chicago: Jianghui Wen (Cherish); Minghao Bian; Zhiyin Shi, Ziyuan Li

Abstract
There are many Machine Learning (ML) algorithms and libraries that can be used for solving the same business problem. Each one provides different response times, accuracy, scalability and resource usage. Selection of the appropriate ML algorithm and ML library for a new Big Data application is one of the most challenging tasks for data scientists and Big Data application developers [2].   Our team Benchmarked and compared responsiveness, accuracy and scalability of several ML algorithms

Benchmarking of ML algorithms and libraries can take a long time, so we decided to develop a methodology and example of the benchmark test [5]. We invited several universities and research organizations to collaborate with us on this project. We quickly discovered that when data scientists review the results of benchmarks, they realize that both their data set sizes and their hardware and software configurations are different from benchmark environment.

To address this issue, we applied ML prediction algorithms to expand the results of the benchmark tests. We also developed algorithm of recommender, which takes into consideration business requirements to response time, accuracy, scalability and cost, size of the data set and modeling results based on benchmark results to determine the most appropriate ML algorithm and ML library.

The objective of this paper is to present a methodology for benchmarking, example of collecting and aggregating data, application of modeling for evaluating results and selecting the appropriate ML algorithms and ML library that will satisfy specific business requirements.

Results of the benchmarks will be stored in a knowledge base accessible by data scientists and Big Data application developers.

1 Introduction
Big Data application developers incorporate descriptive, diagnostic, predictive and prescriptive analytics. Analytics use different implementations of regression, classification, and clustering algorithms stored in different ML Libraries.

In this paper we will review a collaborative project for conducting parallel benchmarks of ML algorithms and libraries

We will present the methodology data collection during benchmark tests of variations of Least-Squared Method (LSM): Ordinary Least Squares (OLS), Ridge, and Random Forest (RF) Regression algorithms.

Because the number of benchmarks tests is limited, we applied ML models to evaluate scenarios where we did not perform tests nor collect measurement data.

Finally, we will review the example of selecting the most appropriate ML algorithm and ML library for specific requirements.

2 Organization of Collaborative Process
Organizers of this project, BEZNext, Data Analytica, IBM, University of Chicago and Protiviti, developed a methodology and conducted a benchmark study for several regression algorithms, which we then shared with several collaborators.

Figure 1 shows tasks planned for organizers and collaborators.

The results of the benchmark tests performed by collaborators will be transformed to the common format and stored in a knowledge based repository, shared with data scientists and application developers concerned with selection of the ML algorithms and ML libraries for new Big Data projects.

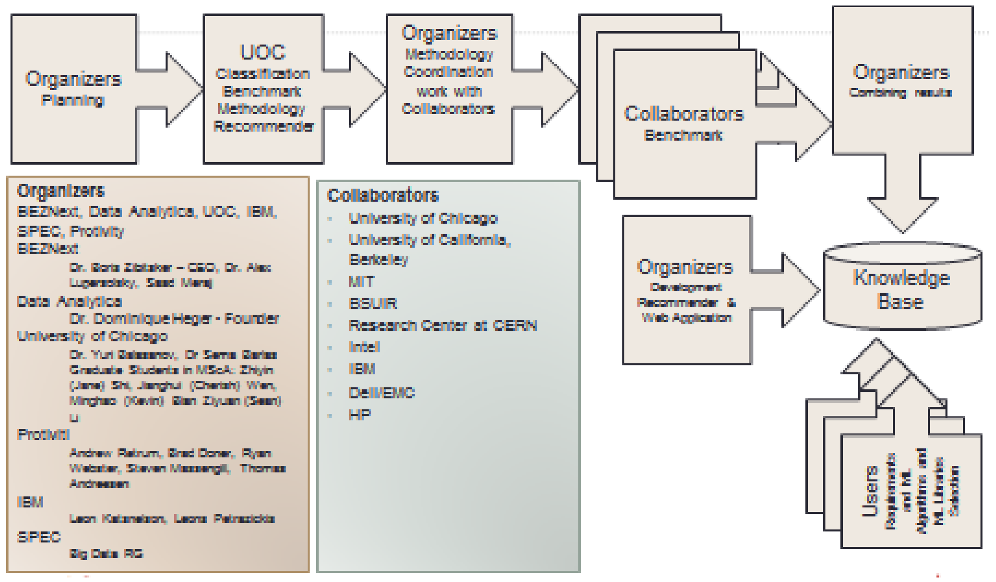

Organizers Planning
UOC Classification Benchmark Methodology Recommender
Organizers Methodology Coordination work with Collaborators
Collaborators Benchmark
Organizers Combining results
Organizers Development Recommender & Web Application
Knowledge Base
Users Requirements and ML Algorithms and ML Libraries Selection
Organizers
BEZNext, Data Analytica, UOC, IBM, SPEC, Protivity
BEZNext
Dr. Boris Zibitsker – CEO, Dr. Alex Lupersolsky, Saad Meraj
Data Analytica
Dr. Dominique Heger - Founder
University of Chicago
Dr. Yuri Balasanov, Dr Sema Barlas Graduate Students in MScA: Zhiyin (Jane) Shi, Jianghui (Cherish) Wen, Minghao (Kevin) Bian Ziyuan (Sean) Li
Protiviti
Andrew Ratnum, Brad Doner, Ryan Webster, Steven Massengill, Thomas Andreasen
IBM
Leon Katsnelson, Leons Petrazickis
SPEC
Big Data RG
Collaborators
University of Chicago
University of California, Berkeley
MIT
BSUIR
Research Center at CERN
Intel
IBM
Dell/EMC
HP

Figure 1. Tasks performed by Organizers and Collaborators of the Project.

Benchmarking process includes the following steps:

- Preparing the Data Sets with different numbers of observations and predictors
- Preparing and running the Benchmark test:
    - Writing Python programs
    - Creating the benchmark environment (Figures 2 and 3)
    - Collecting measurement data about response time, accuracy, CPU, memory usage, I/O rate and network utilization.
- Modeling:
    - Building models to predict performance characteristics of different ML algorithms and ML Libraries for different sizes of the data sets not included into the benchmarks
- Model validating:
    - Comparing the prediction results with actual measurement data for data set with different number of observations and predictors.

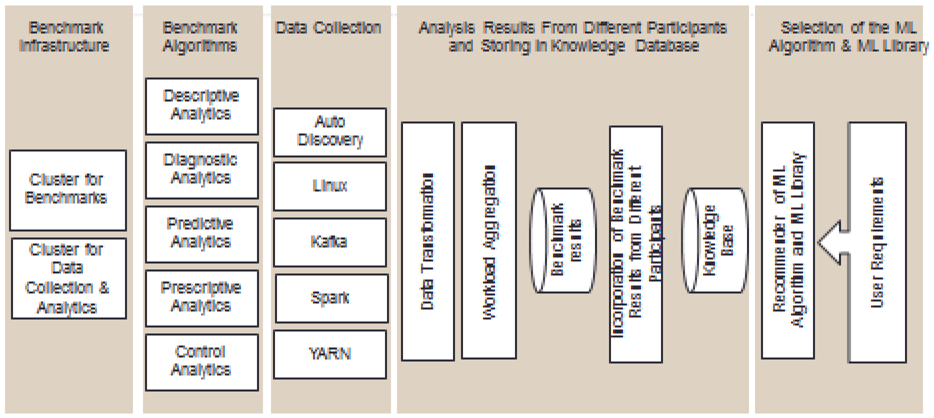


Figure 2. During this project, the Python programs ran standalone and in Spark environment**.**

Infrastructure for the benchmark tests provided by IBM is shown on Figure 3. It includes 4 Data nodes Spark Cluster and 4 Control nodes used for management, data collection, and analysis.

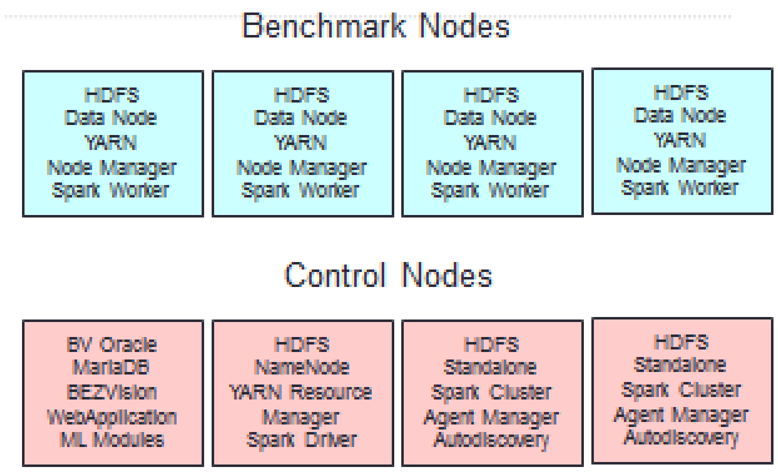

Benchmark Nodes
HDFS Data Node YARN Node Manager Spark Worker
HDFS Data Node YARN Node Manager Spark Worker
HDFS Data Node YARN Node Manager Spark Worker
HDFS Data Node YARN Node Manager Spark Worker
Control Nodes
BV Oracle MariaDB BEZVision WebApplication ML Modules
HDFS NameNode YARN Resource Manager Spark Driver
HDFS Standalone Spark Cluster Agent Manager Autodiscovery
HDFS Standalone Spark Cluster Agent Manager Autodiscovery

Figure 3. Infrastructure for Benchmarking ML Algorithms and ML Libraries

Results of the benchmark tests performed by collaborators are transformed to common format and infrastructure using modeling:

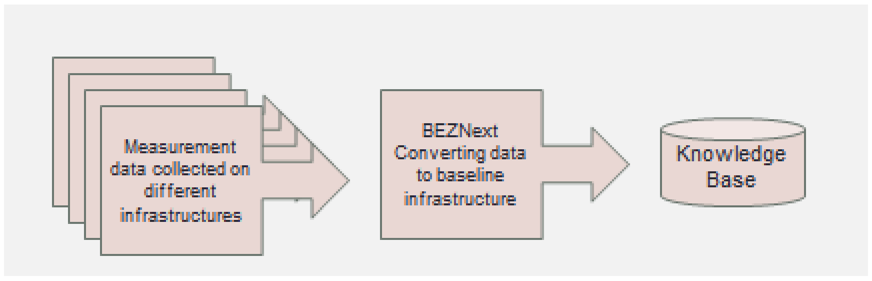

Measurement data collected on different infrastructures
BEZNext Converting data to baseline infrastructure
Knowledge Base

Figure 4. Queueing network models (QNM) are used to convert measurement data collected by different collaborators on different clusters into baseline configuration with four Data Nodes

1 Measurement Data Collected During the benchmark test

During benchmarking OLS, Ridge, RF algorithms implemented in Python were run as standalone Python application and as Spark Application. Each algorithm was tested for 22 dataset sizes with number of predictors changing from 2 to 500 incremented by 50 with 500 and 5000 observations.

Each benchmark test was run for 2 hours.

Measurement data include Accuracy, Response time, CPU utilization, Memory usage, I/O rate and Network throughput changed during the benchmark tests with the increase of the number of observations and number of predictors.

As an *example,* the Memory usage by OLS Algorithms in Spark MLlib library increases with the increase in number of observations *is shown below*.

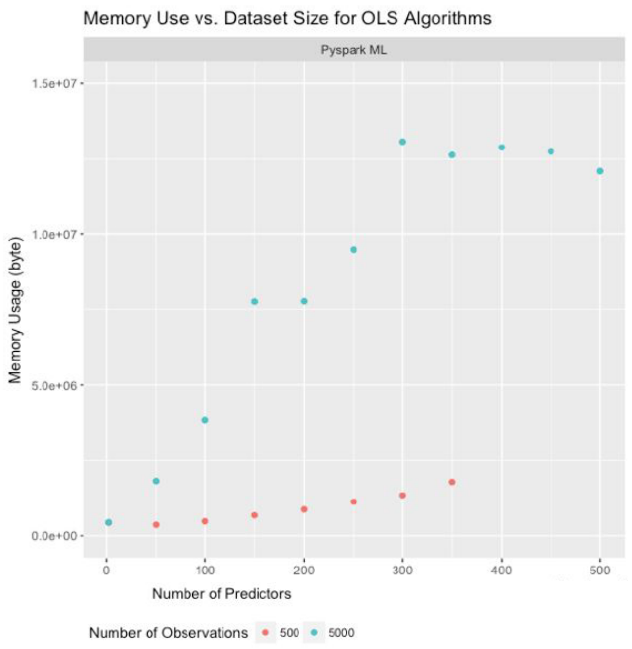


Figure 5. The memory usage of Ridge algorithms for 500 observations and small number of predictors in Spark MLlib library is significantly lower comparing with 5000 observations and increased number of predictors.

2 Use of Machine Learning Algorithms to Expand Data Collected During Limited Number of Tests

Number of benchmarks tests is limited, so we applied ML models to fill gaps and enable evaluation of scenarios where tests were not performed and measurement data were not collected. These models will be used to evaluate algorithms and recommend one which will be most appropriate based on size of the data set and performance requirements of new application [3].

We use different ML algorithms to predict the algorithms' metrics and estimate the accuracy by comparing prediction results with measurement data.

Measurement data were split into two data sets: 80% of data were used for model training and 20% of data were used to compare the actual measurement data with prediction results. Trained models are used to predict Response Time, CPU Utilization, I/O rate, Memory Utilization, and Accuracy for each ML Algorithm and ML Library [4].

2.1 Prediction of Accuracy for Different ML Algorithms

Prediction results for OLS and Ridge show that accuracy depends on data set sizes and have a linear relationship when ratio between the number of predictors and number of observations is in [0.01, 0.6].
To predict accuracy for each ML algorithm we built OLS and Random Forest models using Python Libraries.

2.1.1 Prediction of Accuracy for LSM Algorithms:

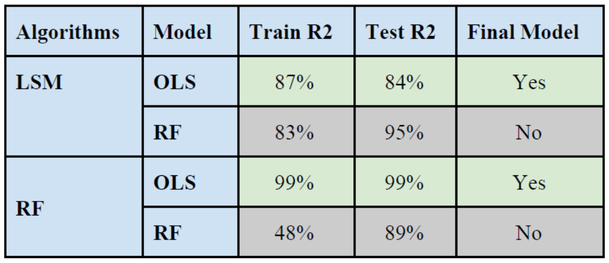
The model: **Accuracy = f(Algorithm, Library, Number of Observations, log(log(Number of Features)).**

2.1.2 Accuracy Prediction for Random Forest Algorithm:

Building model with original dataset didn't explain much variance, therefore we considered data transformation. We log-transformed the feature - number of features. Our model is represented in the equation below:

Accuracy = f(Algorithm, Number of Observations, log(Number of Features)).

Model Selection:



Algorithms	Model	Train R2	Test R2	Final Model
LSM	OLS	87%	84%	Yes
	RF	83%	95%	No
RF	OLS	99%	99%	Yes
	RF	48%	89%	No

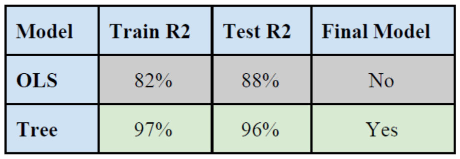
Table 1. For LSM and RF the difference between Train and Test lower for OLS

2.2 Prediction of CPU Utilization for Different ML Algorithms:

The models we used include OLS Regression and Regression Tree. For each model, the variables involved are shown below:

CPU Utilization= f(Algorithm, Library, Number of Observations, Number of Features)

Model Selection:



Model	Train R2	Test R2	Final Model
OLS	82%	88%	No
Tree	97%	96%	Yes

Table 2. Regression Tree model is the final model since it has higher accuracy and stability (the difference between train and test results is smaller).

2.3 Prediction of Memory Usage for Different ML Algorithms:

Memory Usage Model:

The models we used include OLS Regression, Regression Tree and Random Forest. For each model, the variables involved are shown below:

log(Memory Usage) = f(Algorithm, Library, Number of Observations, Number of Features)

Model Selection:

Model	Train R2	Test R2	Final Model
OLS	78%	87%	No
Tree	87%	81%	Yes
RF	60%	62%	No

Table 3. Regression Tree model is the final model.

2.4 Prediction of Response Time for Different ML Algorithms:

Response Time Model:
We built a single model for LSM algorithms and another model for Random Forest algorithm.

2.4.1 Response Time Model for LSM Algorithms:

The models we used include OLS Regression, Random Forest and Regression Tree.
Building model with original dataset didn't explain much variance, therefore we considered data transformation. We log-transformed the response - Average Response Time to make the response variable more Gaussian-like and the relationship between response and dataset size more linear. The model formula is expressed below:

log(Response Time) = f(Algorithm, Library, Number of Observations, Number of Features)

Model Comparison:

Model	Train R2	Test R2	Final Model
OLS	89%	91%	Yes
RF	70%	96%	No
Tree	80%	88%	No

Table 4. OLS regression outperforms tree and random forest models in predicting the response time.

OLS regression outperforms tree and random forest models in predicting the response time since R2 in both train and test dataset are higher and the accuracy is more stable. Therefore, OLS Regression Model is selected as the final model to predict response time and will be incorporated in our recommender

2.4.2 Response Time Model for RF Algorithm:

Building a model with original dataset didn't explain much variance, therefore we considered data transformation. We log-transformed the response - Average Response Time to make the response variable more Gaussian-like and the relationship between response and dataset size more linear. The model formula is expressed below:

log(Response Time) = f(Library, Number of Observations, Number of Features)

Model Comparison:
The results of OLS Regression, Random Forest Models in train and test dataset are shown in the following.

Model	Train R2	Test R2	Final Model
OLS	73%	85%	Yes
RF	50%	91%	No

Table 5. OLS regression outperforms random forest models in predicting the response time for RF Algorithm.

To select a best fitted model for Accuracy, Response time, Memory usage and CPU utilization we compared the train and test R2 among all models. For accuracy and response time prediction OLS models are selected for both LSM and Random Forest algorithms. For memory usage and CPU utilization we selected tree models which obtain train R2 0.87 and 0.97 respectively. All the selected models' performance is listed below:

Measure	Algorithms	Model	Train R2	Test R2
Accuracy	LSM	OLS	87%	84%
	RF	OLS	99%	99%
Response Time	LSM	OLS	89%	91%
	RF	OLS	73%	85%
Memory Usage	LSM & RF	Tree	87%	81%
CPU Utilization	LSM & RF	Tree	97%	96%

Table 6. Summary of comparing different algorithms: OLS should be used to predict Accuracy and Response Time. Regression Tree algorithm should be used to predict CPU and Memory utilization.

3 Recommender of ML Algorithms and ML Libraries Selection

The Score takes into consideration the type of ML algorithm, Number of Observations and Features / Predictors in Data Set, the relative importance of the different criteria, like response time, Accuracy, CPU Utilization, Memory utilization, Number of I/O operations, and other parameters:

Score = w1 * Accuracy + w2 * Response Time + w3 * CPU Utilization + w4 * Memory Utilization,

where the weighting coefficients *wi* represent business priorities between 0 and 1.

Type of Requirement	Relative Weight
Accuracy	0.2
Response Time	0.4
CPU Utilization	0.2
Memory Utilization	0.2
Total	1

Table 7 Example of the relative importance of the different requirements to the new application

Response Time can vary between 0 and infinity. We transform the response time as 1 / (1 + RT) to make it as a number between 0 and 1, where 1 is better. In addition to calculating the score we check if predicted CPU Utilization and Memory Usage are less than 1.

Value of score is used to recommend the appropriate ML algorithm and ML Library.

Algorithm	library	pred_score	pred_rank	true_score	true_rank
OLS	Python Sklearn	0.962057911	1	0.936165261	1
OLS	Pyspark ML	0.876712666	2	0.753752225	2
Ridge	Python Sklearn	0.781980143	3	0.725268522	3
Ridge	Pyspark ML	0.722426161	4	0.659234146	4
RF	Python Sklearn	0.476284999	5	0.429752013	5
RF	Pyspark ML	0.465422159	6	0.415271967	6

Table 8. ML OLS Algorithm using Python Sklearn ML library is the most appropriate algorithm to satisfy business requirements presented in Table 7.

4 Summary

In this presentation we reviewed the methodology of conducting benchmarks of the different ML algorithms and ML libraries in parallel by different collaborators. It includes data preparation for benchmark, collection of performance measurement data, workload characterization, performance analysis and comparison of the different algorithms and libraries.

We reviewed an example of the algorithm to select the most appropriate algorithm and library based on requirements to performances, use of resources, and accuracy.
This project is a foundation for organizing collaboration between Universities and Research organizations in benchmarking of ML algorithms and Libraries.

Selection of the appropriate ML algorithms and libraries during design and development of Big Data application is a part of Performance Engineering [1], which primary goal is to reduce risk of performance surprises.

Related Work
Several papers presented at ACM and IEEE conferences discuss Big Data ML algorithms benchmarking, but they are not focus on development recommendation how to select appropriate algorithm and ML library for a specific project.,

Future Work
A future work focus is on benchmarking different types of ML algorithms and incorporation of Prescriptive Analytics to develop Recommenders

Key Words
Performance Assurance, Performance Engineering, Benchmarking, Machine Learning Algorithms and Machine Learning Benchmark, Machine Learning Algorithm Selection and Machine Learning Library Selection

Acknowledgement
Special thanks go out to our colleagues from BEZNext, Leon Katsnelson, CTO, IBM and Leons Petrazickis, Platform Architect, IBM and Dr. Sema Barlas, Director of Master of Science in Analytics program at University of Chicago.

References
1. Daniel A. Menasce “Software, Performance, or Engineering?” WOSP '02 Proceedings of the 3rd international workshop on Software and performance Pages 239-242.
2. B. Zibitsker, IEEE Conference, Delft, Netherlands, March 2016, Big Data Performance Assurance.
3. Dominique A. Heger “Big Data Predictive Analytics, Applications, Algorithms and Cluster Systems” ISBN:978-1-61422-951-3.
4. Yuri Balasanov, Linear Regression with Large Number of Predictors, Lecture materials for: "Machine Learning and Predictive Analytics", MScA, University of Chicago, 2016
5. Benchmark using Reuters Data: https://github.com/BIDData/BIDMach/wiki/Benchmarks#reuters-data.