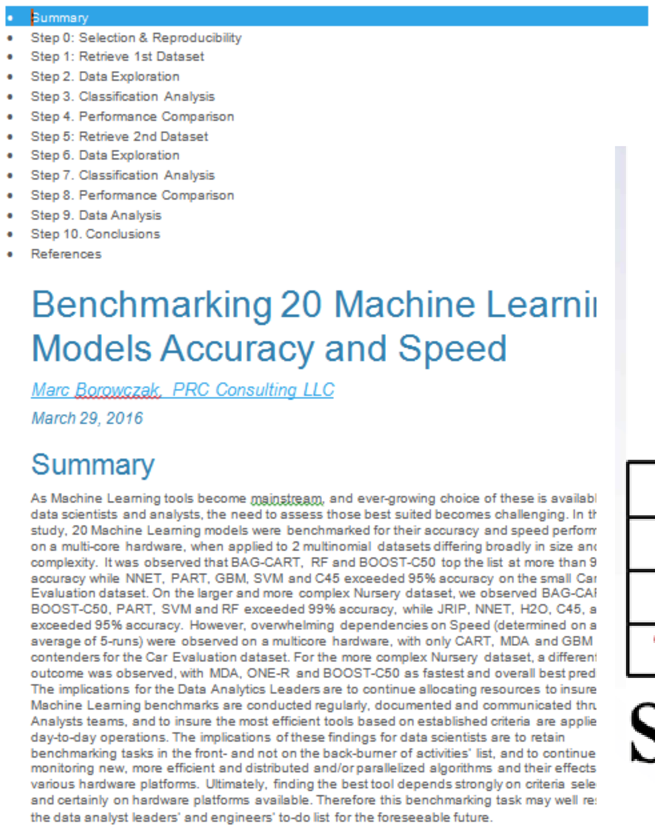
ML ALGORITHMS AND ML LIBRARIES SELECTION

BEZNext, Data Analytica, University of Chicago, Protiviti, IBM, SPEC

Motivation

- Selection of ML Algorithms and ML Libraries affects performance, scalability, accuracy and cost
- It is one of the most challenging tasks for application developers and data scientists
- There are many benchmarks of the individual ML algorithms are performed, but they are done on different environments and there is no methodology for comparing algorithms and ML Libraries

Publications



Benchmarking 20 Machine Learni
Models Accuracy and Speed

Marc Borowczak, PRC Consulting LLC

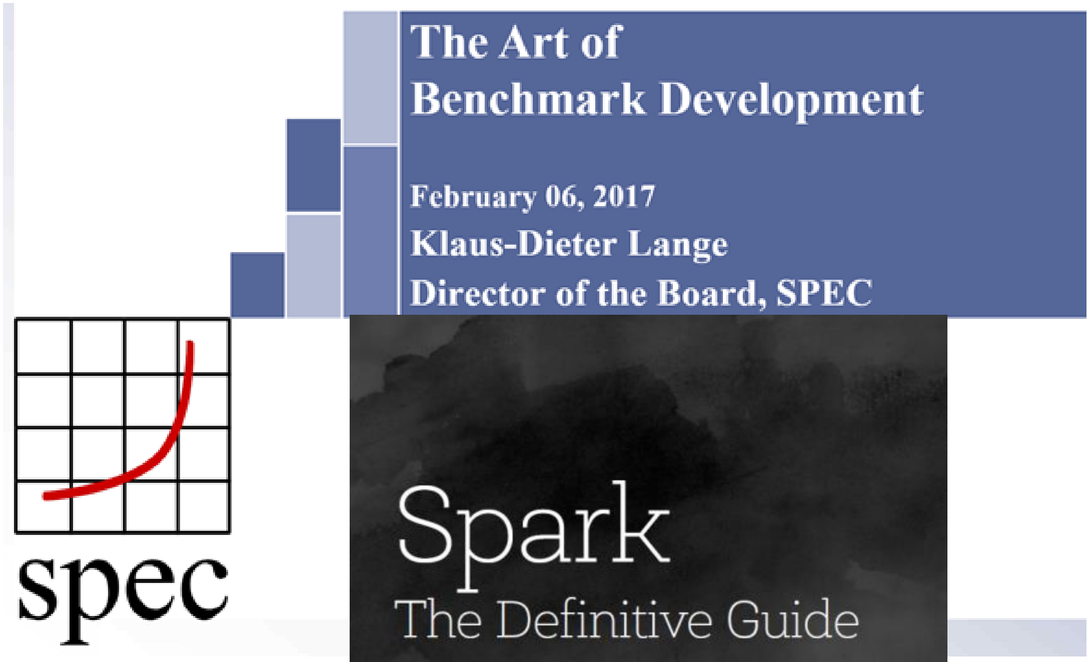
March 29, 2016

Summary

As Machine Learning tools become mainstream, and ever-growing choice of these is availabl
data scientists and analysts, the need to assess those best suited becomes challenging. In th
study, 20 Machine Learning models were benchmarked for their accuracy and speed perform
on a multi-core hardware, when applied to 2 multinomial datasets differing broadly in size anc
complexity. It was observed that BAG-CART, RF and BOOST-C50 top the list at more than 9
accuracy while NNET, PART, GBM, SVM and C45 exceeded 95% accuracy on the small Car
Evaluation dataset. On the larger and more complex Nursery dataset, we observed BAG-CA
BOOST-C50, PART, SVM and RF exceeded 99% accuracy, while JRIP, NNET, H2O, C45, a
exceeded 95% accuracy. However, overwhelming dependencies on Speed (determined on a
average of 5-runs) were observed on a multicore hardware, with only CART, MDA and GBM
contenders for the Car Evaluation dataset. For the more complex Nursery dataset, a differen
outcome was observed, with MDA, ONE-R and BOOST-C50 as fastest and overall best pred
The implications for the Data Analytics Leaders are to continue allocating resources to insure
Machine Learning benchmarks are conducted regularly, documented and communicated thru
Analysts teams, and to insure the most efficient tools based on established criteria are applie
day-to-day operations. The implications of these findings for data scientists are to retain
benchmarking tasks in the front- and not on the back-burner of activities' list, and to continue
monitoring new, more efficient and distributed and/or parallelized algorithms and their effects
various hardware platforms. Ultimately, finding the best tool depends strongly on criteria sele
and certainly on hardware platforms available. Therefore this benchmarking task may well re
the data analyst leaders' and engineers' to-do list for the foreseeable future.

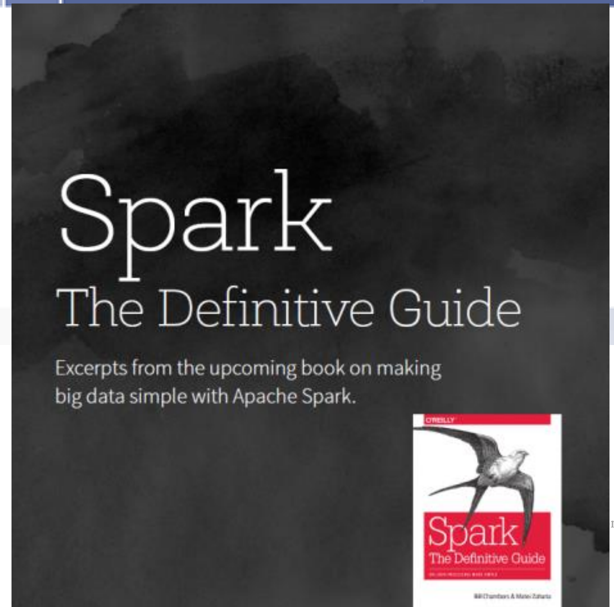
Step 0: Selection & Reproducibility

As Machine Learning gains attention, more applications and models are being used, and often
speed or accuracy of the predicted models lack comparisons. In this analysis, we'll compare the
accuracy and speed of 20 Machine learning models commonly selected. We'll exercise these
models on two multinomial UCI reference datasets, differing in size, predictor levels and number of
dependent variables. The first and smallest dataset is Car Evaluation and we'll compare results



The Art of
Benchmark Development

February 06, 2017
Klaus-Dieter Lange
Director of the Board, SPEC



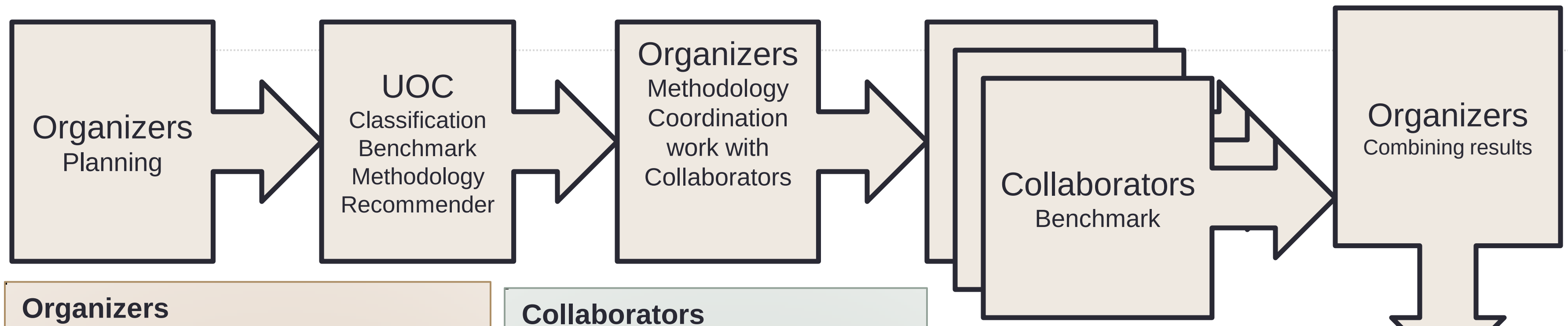
Spark
The Definitive Guide

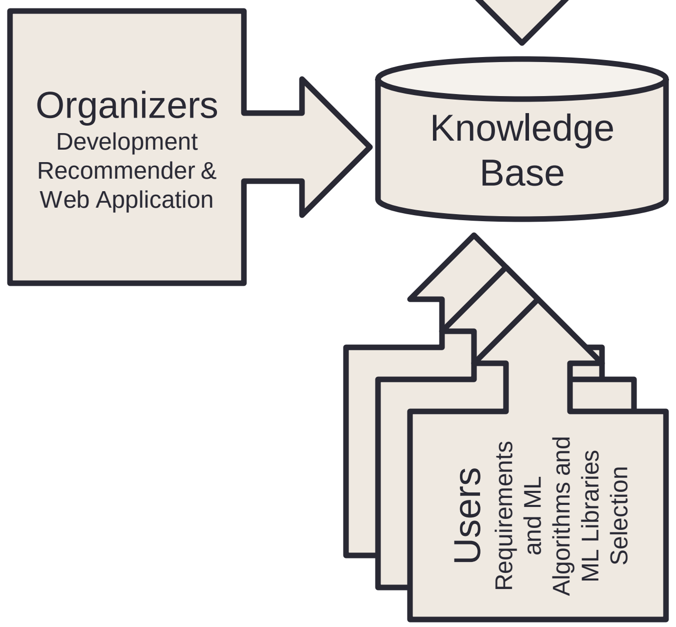
Excerpts from the upcoming book on making
big data simple with Apache Spark.

Objective

- Develop an example of the benchmark for testing several Regression Algorithms and create a methodology for collaborators for benchmarking of ML algorithms and ML libraries
- Conduct the benchmark tests in parallel by Collaborators
- Use models to convert benchmark's results to the common format, store results in a knowledge database
- Develop Recommender assisting with selection of ML Algorithms and ML Libraries for specific requirements

Roles of Organizers and Collaborators

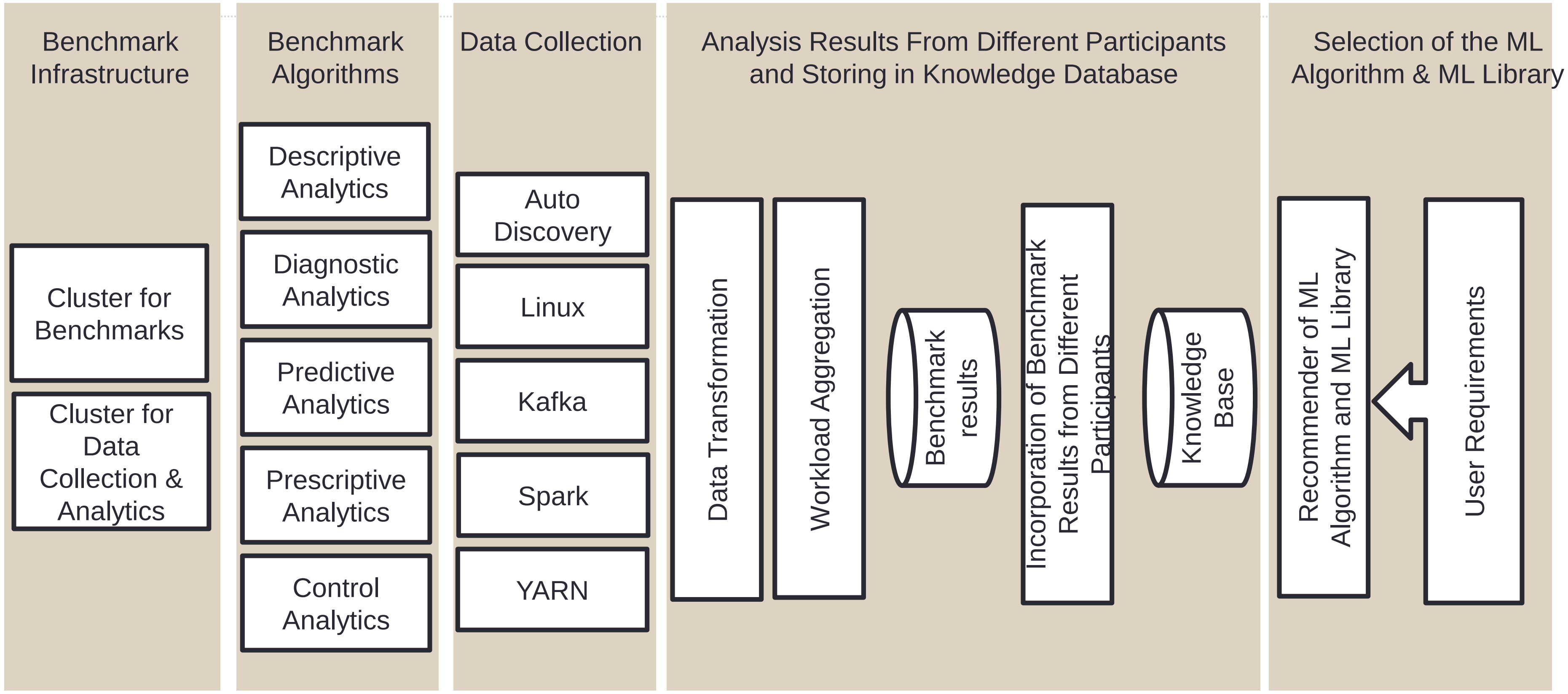

Organizers
Planning
UOC
Classification Benchmark Methodology Recommender
Organizers
Methodology Coordination work with Collaborators
Collaborators
Benchmark
Organizers
Combining results


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 Retrum, Brad Doner, Ryan Webster, Steven Massengill, Thomas Andreesen
IBM
Leon Katsnelson, Leons Petrazickis
SPEC
Big Data RG

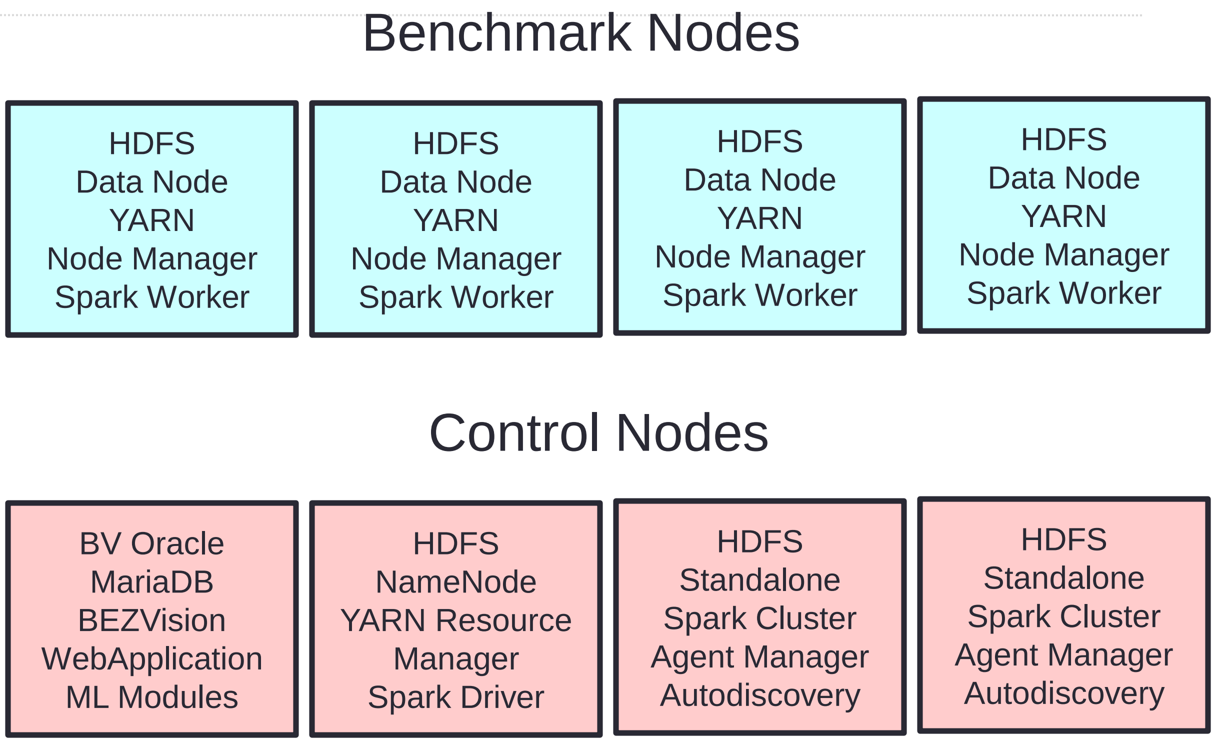
Collaborators
- University of Chicago
- University of California, Berkeley
- MIT
- BSUIR
- Research Center at CERN
- Intel
- IBM
- Dell/EMC
- HP

Process


Benchmark Infrastructure
Cluster for Benchmarks
Cluster for Data Collection & Analytics
Benchmark Algorithms
Descriptive Analytics
Diagnostic Analytics
Predictive Analytics
Prescriptive Analytics
Control Analytics
Data Collection
Auto Discovery
Linux
Kafka
Spark
YARN
Analysis Results From Different Participants and Storing in Knowledge Database
Data Transformation
Workload Aggregation
Benchmark results
Incorporation of Benchmark Results from Different Participants
Knowledge Base
Selection of the ML Algorithm & ML Library
Recommender of ML Algorithm and ML Library
User Requirements

Reference Infrastructure for ML Algorithms and ML Libraries Benchmarks Provided by IBM BDU

- HDFS DataNode, YARN NodeManager and Spark Worker on each of the 4 nodes.
- Two nodes are used for BV Oracle, WebApp MariaDB, HDFS NameNode, YARN ResourceManager and Spark Client / Driver for benchmark cluster, BEZVision and Analytics WebApp including Python modules.
- Two nodes are used for the processing cluster: Spark standalone cluster, HDFS NameNode / DataNode, Agent Manager and Autodiscovery.


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

Input

Benchmark

- Input Data Set
- Infrastructure
- Data Collection Technology
- Instructions on how to run benchmark

Recommender

Users' Requirements for Selecting ML Algorithms and Libraries

- Accuracy
- Performance
  - Response Time, Throughput
- Resource Utilization
  - CPU, Memory, SSD, HDD, Network
- Scalability

Data Set

Infrastructure

Output

Measurement Data Collected During Benchmark Tests

- ML Algorithm Type
- ML Algorithm Name
- ML Library Name
- Performance Measurement Type (RT, Throughput)
- Usage of Resources (CPU, I/O, Memory, Network)
- Accuracy Measurement Type (mse, …)
- Scalability (sensitivity of performance to changes in #predictors, volume of data, increase in # concurrent jobs …)
- Cost (#nodes required to support business SLGs for expected increase in volume of data and # of concurrent jobs)

Business Requirements

Infrastructure

Resources

- BEZNext Data Collection, Workload Characterization and Performance Prediction Software
- IBM BDU Clusters
- Information from Spec organization
- Protiviti Python and Spark expertise

Deliverables

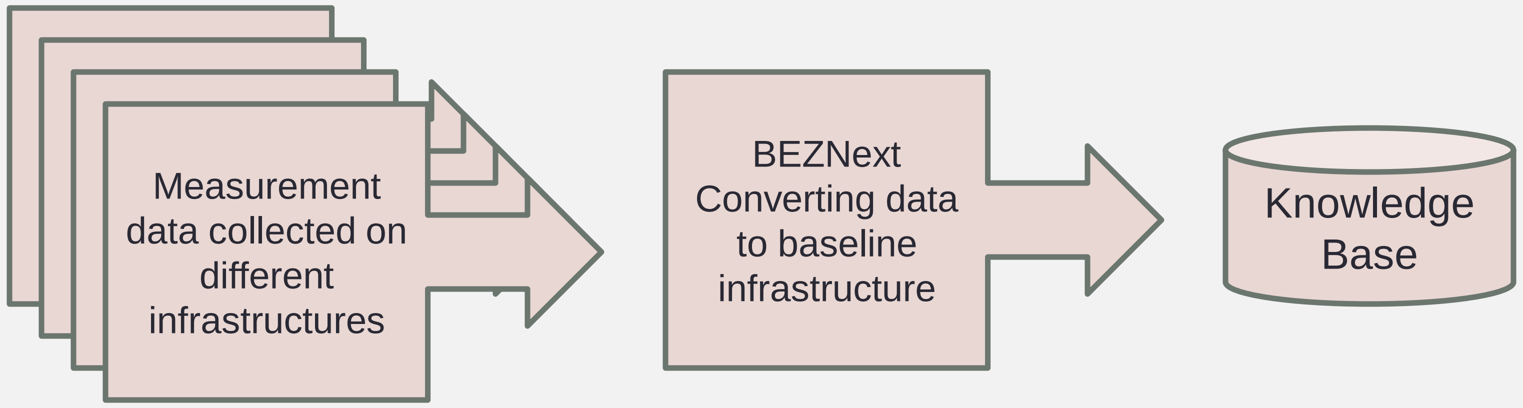
- Methodology, Recommendation for selection of the Algorithm and ML library
- Results of testing ML algorithms and ML Libraries
- Recommendations

Types of ML Algorithms and ML Libraries

- Classification
- Cluster Analysis
- Correlation Analysis
- Regression Analysis
- Optimization Analysis

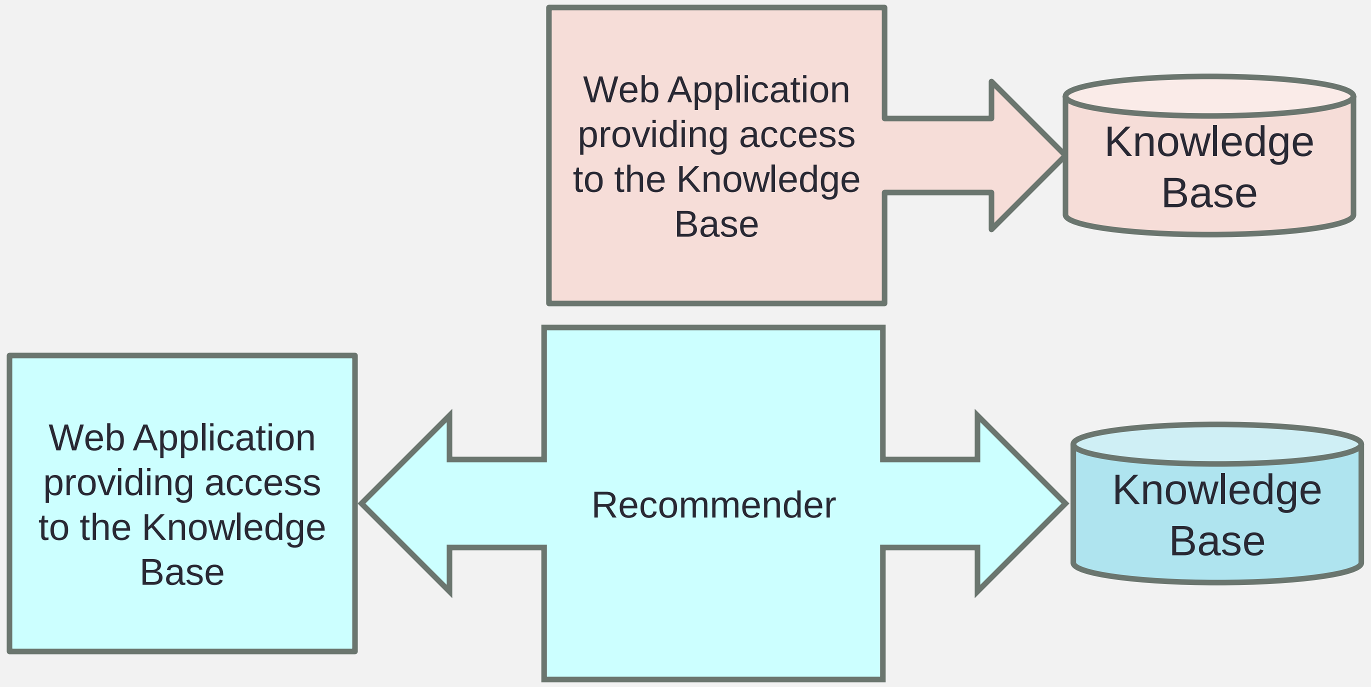
- Descriptive
- Diagnostic
- Predictive (Regression, Time Series, ANN, QNM),
- Prescriptive

Use BEZVision to convert measurement collected by collaborators in different environment into 4 nodes baseline configuration


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

Develop Web Site

- User Requirements for ML application
- Develop algorithm and implement Recommender for selecting ML algorithm and ML Library


Web Application providing access to the Knowledge Base
Knowledge Base
Web Application providing access to the Knowledge Base
Recommender
Knowledge Base

Implementation Progress

Report 1

- Built benchmark tests for OLS, Ridge and Lasso Regression in sklearn in Python.
- Collected measurement data for 60s for the three algorithms.

Report 2:

- Built benchmark tests for SVM and Random Forest in sklearn in Python.
- Built benchmark test for OLS linear regression in spark considering one node.
- Started build prediction model for accuracy, response time, cpu utilization and memory usage.

Report 3:

- Revise benchmark test in Python, considering cross-validation to tune parameters in SVM, Ridge,
- Lasso and Random Forest.
- Recollect measurement data for Python benchmark tests.
- Revise our models with new measurement data.

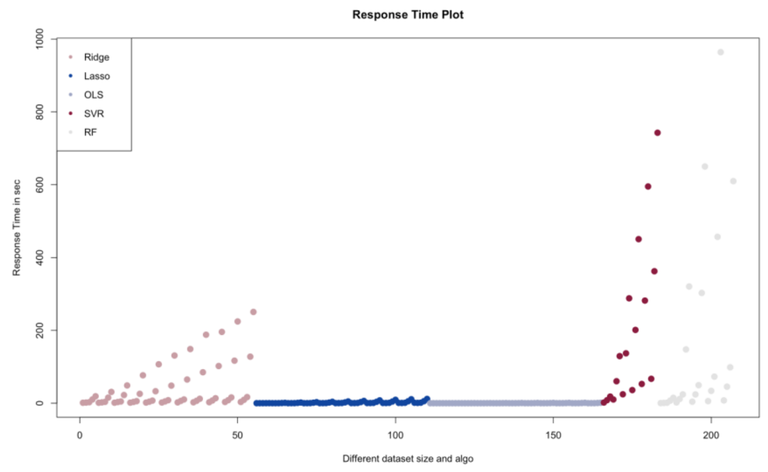
Implementation Report Part 1 & 2 Benchmark Test:

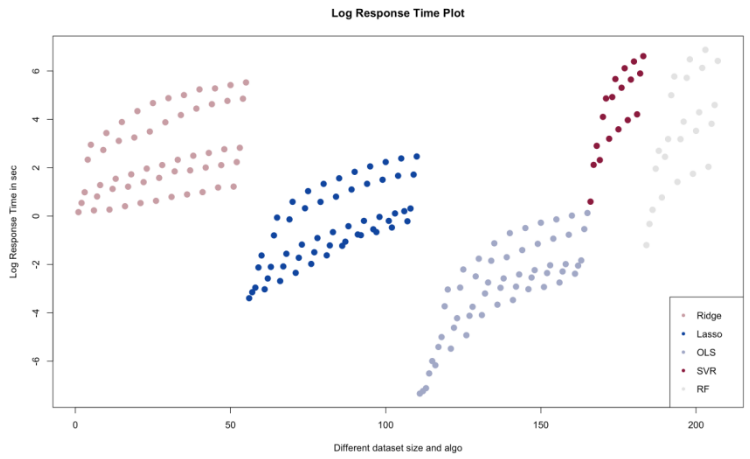
- The benchmark test scripts for OLS, Ridge and Random Forest in Python sklearn and Spark MLlib considering one node is in the capstone Dropbox.
- Collecting measurement data with every combination of number of predictors and observations can be time consuming. Therefore we are benchmarking a selected number of dataset sizes and build models later to predict the performance of other dataset sizes.

Implementation Report Part 3 Modeling:

Part 3.1 Response Time:

- The plot of response time and against dataset size with and without log transformation are shown below. Each color represents an algorithm. Within each algorithm, we can see there are several levels (data of different slopes), the levels varying in vertical direction represents different number of observations and the variation in horizontal direction represents different number of features. The response time increases exponentially for all algorithms. This provides us a reason to build models on the log-transformed response time.

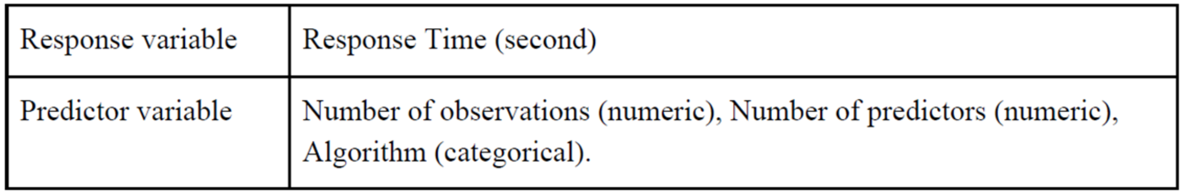




Implementation Report Part 3 Modeling:

Part 3.1 Response Time:

1. Simple linear regression and single regression tree were considered to model response time. The measurement dataset is randomly separated to train and test. Model was built using train dataset and test error was computed.
2. For the regression tree, cv suggests a tree with 12 nodes as final model.
3. For each model, the variables involved are shown below:



Response variable	Response Time (second)
Predictor variable	Number of observations (numeric), Number of predictors (numeric), Algorithm (categorical).

OLS Regression Model:
OLS Regression Model was trained to predict the response time. The model yielded satisfactory results in both train and test dataset with R2 greater than 90%. The residual plot, q-q plot and residual histogram are shown in Figure. 3, 4, 5 respectively. Except for few outliers that came from random forest and SVR category, the residuals are normal and meet our assumption.

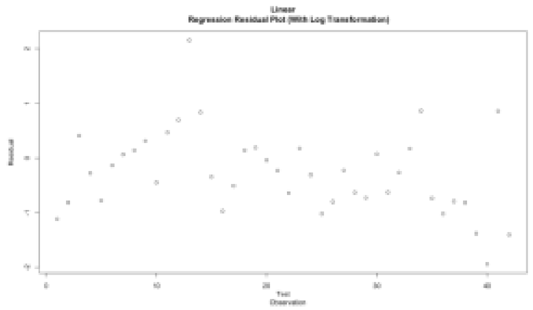


Figure.2.3. Residual Plot of OLS Regression Model for Response Time

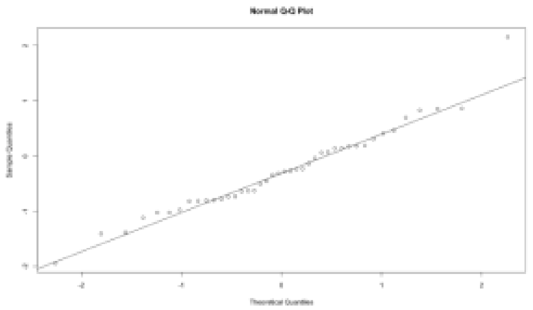


Figure.2.4. Q-Q Plot of OLS Regression Model for Response Time

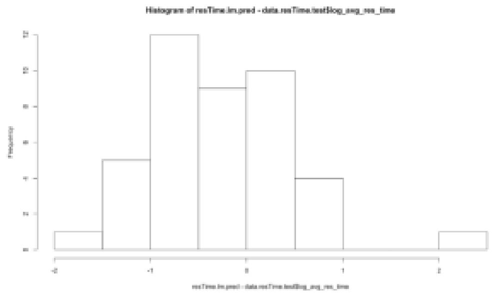


Figure.2.5. Residual Histogram of OLS Regression Model for Response Time

Regression Tree Model:
Regression Tree Model was trained to predict the response time. The R2 in training dataset is greater than 90% while the in test dataset the R2 drops to 80%.

Model Comparison:
The results of OLS Regression and Regression Tree Models in train and test dataset are shown in Figure.2.6.

	Model Accuracy R2
ResTime.lm.Train	0.92
ResTime.lm.Test	0.93
ResTime.tree.Train	0.9
ResTime.tree.Test	0.81

Figure.2.6. Results of OLS and Tree for Response Time

Overall, OLS regression outperforms tree model in predicting the response time since the R2 in both train and test dataset are higher and the out-of-bag performance is more stable. Therefore, OLS Regression Model is selected as the final model to predict response time and will be assembled in our recommender prototype in later section.

Accuracy Model:

Data Exploration:
The plot of accuracy and against dataset size is shown Figure.1.1. Each color represents an algorithm. Within each algorithm, we can see there are several levels (data points at different slopes), the levels varying in vertical direction represents 5 different number of observations and the variation in horizontal direction represents 11 different number of features.

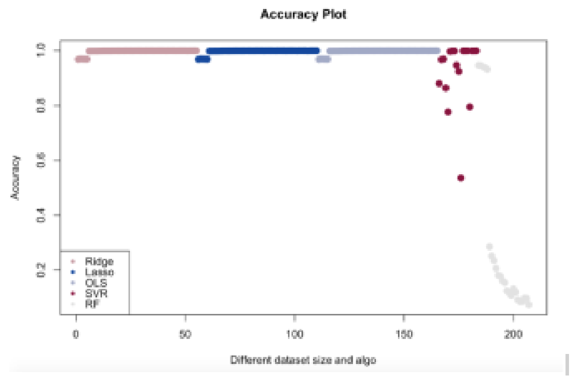


Figure.1.1 Accuracy v.s. Dataset Size

From the figure above, we can see the accuracy behaves differently among least-square method (OLS, Lasso and Ridge), SVR, Random Forest. Therefore, we analyzed the data and build model for each of the three categories separately. Finally we choose a random forest model to predict accuracy of LSM group, OLS model to predict Random Forest group. The SVR group is highly volatile and unpredictable, thus we are unable to model the relationship. Details are shown below.

Accuracy Model:
Models were built separately for each of the three algorithm categories: LSMs, SVR and Random Forest. Simple linear model and random forest model were applied to predict accuracy from dataset size. The data is randomly splitted to train and test dataset by the ratio of 0.8:0.2. Model build use the train set, and evaluate and compared by test errors.

Accuracy Model for LSM Algorithms:
Building model with original dataset didn't explain much variance, therefore we considered data transformation and the models discussed in this section is based on transformed data where the number of features is taken of log twice. This transformation makes sense since in general, the accuracy increases as the number of features increases, but the rate of increase in accuracy is gradually decreasing, thus a log transformation make the relationship more linear. Our model is represented in the equation below:

Accuracy = Algorithm + Number of Observations + log(log(Number of Features)).

OLS Regression Model for LSM Algorithms:
OLS regression model resulted in R2 = 96.6% in train dataset and R2 = 95.6% in test dataset. The model residual plots are shown below, as we can see the residual distribution is skewed to the right and it violates our assumption on normality.

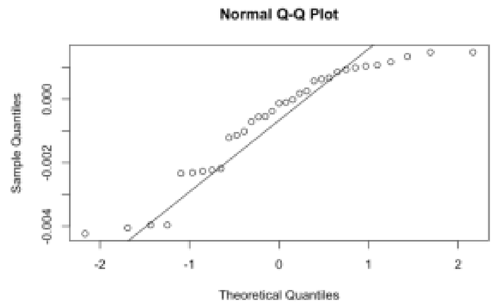
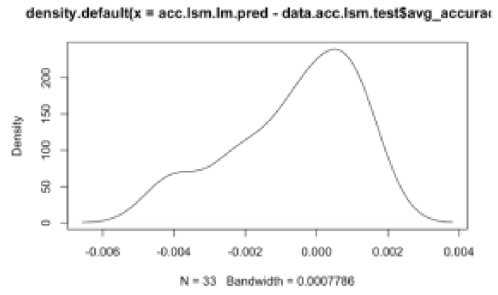


Figure.1.2 Q-Q Plot of OLS Regression Model to Predict Accuracy for LSM Algorithms



Random Forest Model for LSM Algorithms:
Random Forest model has train R2 = 98.9% and test R2 = 99.8%. The resulting R2 are almost perfect in both train and test dataset. In addition to this, random forest adopts a binary split algorithm therefore there is no assumption on the residual, which make this model more favorable than the OLS Regression.

Accuracy Model for SVR Algorithm:
Both OLS and tree regression were used to model the accuracy for SVR algorithm. However, scatter plot of response and pairwise plots suggest no pattern or relationship. Different transformations on both response and features are tried but the test R2 is still below 5%. The accuracy of SVR algorithm is highly volatile since SVR required extensive efforts and experience to cross-validate and tune the parameters, but our benchmarking process only considered less than 10 possibilities of parameter tuning. Therefore we would recommend other algorithms where the accuracy performance is more stable.

Accuracy Model for Random Forest Algorithm:
Building model with original dataset didn't explain much variance, therefore we considered data transformation and the models discussed in this section is based on transformed data where the number of features is taken of log twice. Our model is represented in the equation below:

Accuracy = Algorithm + Number of Observations + log(Number of Features).

OLS Regression Model for Random Forest Algorithm:
OLS regression model has train r2 = test r2 = 98.1%. This model's residual plots are shown below. Since we only have 5 data points in the test dataset, our plots can be misleading, but overall the residual is normal.

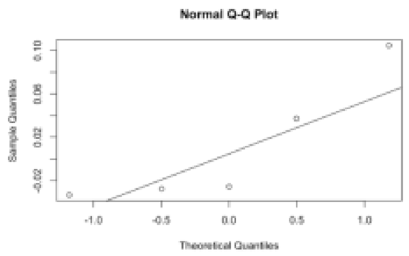


Figure.1.4. Q-Q Plot of OLS Regression Model to Predict Accuracy for RF Algorithms

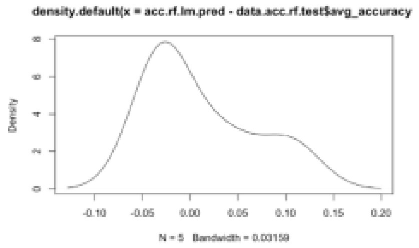


Figure.1.4. Residual Density Plot of OLS Regression Model to Predict Accuracy for RF Algorithms

Random Forest Model for Random Forest Algorithm:
Random Forest model has train r2 = 90.8% and test r2 = 94.6%. The results are not as competitive as the OLS Regression Model.

Model Comparison:
The results of all models in train and test dataset are shown in Figure.1.5.

	Model Accuracy R2
acc.lsm.lm.Train	0.97
acc.lsm.lm.Test	0.96
acc.lsm.rf.Train	0.99
acc.lsm.rf.Test	1
acc.rf.lm.Train	0.98
acc.rf.lm.Test	0.98
acc.rf.rf.Train	0.91
acc.rf.rf.Test	0.95

Considering model accuracy, assumption hold and stability of model accuracy in train and test, finally we choose a random forest model to predict accuracy of LSM group, OLS model to predict Random Forest group. The SVR group is highly volatile and required prior experience to tune the model parameters, and our benchmarking process did not capture the best parameter set therefore we are unable to model the relationship.

Memory Usage Model:

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

Response variable	Memory Usage
Predictor variable	Number of observations (numeric), Number of predictors (numeric), Algorithm (categorical).

OLS Regression Model:

OLS Regression Model was trained to predict the memory usage. The model does not yielded satisfactory results in both train dataset with R2 50% and test dataset with R2 only 22%. The residual plot, q-q plot and residual histogram are shown in Figure. 3, 4, 5 respectively.

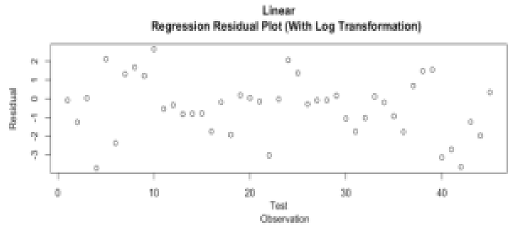
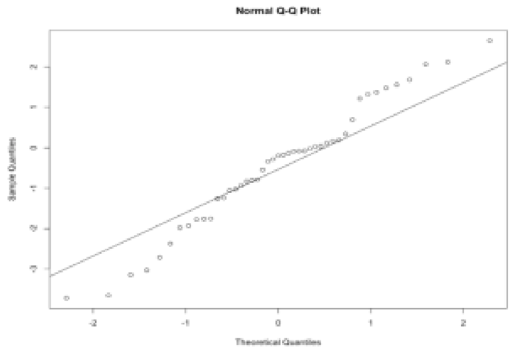


Figure.3.3. Residual Plot of OLS Regression Model for Memory Usage



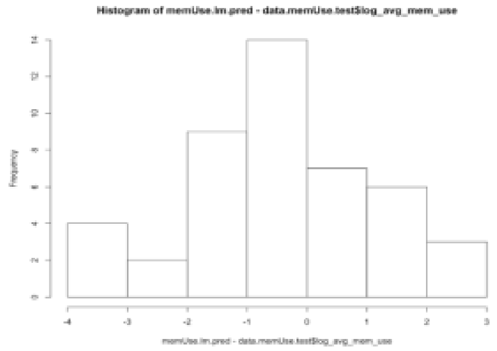


Figure.3.5. Residual Histogram of OLS Regression Model for Memory Usage

Regression Tree Model:
Regression Tree Model was trained to predict the response time. The R2 in training dataset is 50% while the in test dataset the R2 drops to 19%. The residual plot, q-q plot and residual histogram are shown in Figure. 6, 7, 8 respectively.

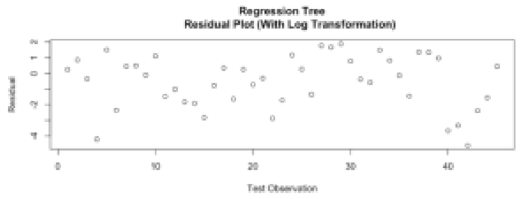
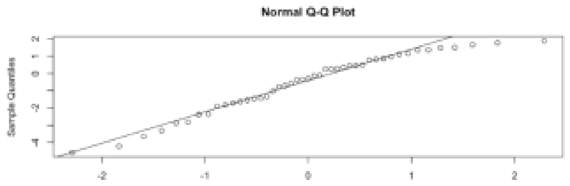
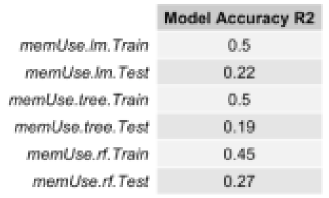


Figure.3.6. Residual Plot of Regression Tree Model for Memory Usage



Model Comparison:

The results of OLS Regression and Regression Tree Models in train and test dataset are shown in Figure.3.12.



	Model Accuracy R2
memUse.lm.Train	0.5
memUse.lm.Test	0.22
memUse.tree.Train	0.5
memUse.tree.Test	0.19
memUse.rf.Train	0.45
memUse.rf.Test	0.27

Figure.3.12. Results of OLS, Tree and Random Forest for Memory Usage

Overall, none of the three models did a good job in predicting the memory usage since the R2 in all of th train and test dataset are extremely low. Therefore, we begin to seek answers on why it is hard to predict memory usage at this stage.

Caveat:
We had a meeting with our client BEZNext and the Founder and CTO Dominique Heger from Data Analytica. He owns a lot of experience in this area. He offered us the feedback that it is indeed very hard to predict memory usage. The memory usage measurement data we currently has is not accurate. It will b more accurate to use an individual separate cluster to run the benchmark. The proper procedure to measure memory usage includes doing one iteration, get the measurement data, bring the cluster down, reboot ; second iteration: bring the cluster down, get the measurement data, bring the cluster down, reboot; keep iterating. In this way, the collected memory usage measurement data is far more accurate to be utilized for building prediction model.

As IBM has already provided us a separate cluster, we can follow the steps to collect the measurement data and build the model.

Final recommander
After we get each performance for each type of algorithm, we can calculate scores for the each algorithm based on the equation below:

$$Score = w_1 * Accuracy + w_2 * Response\ Time + w_3 * Memory\ Usage$$

Note that :

1. Weight selection
   The weighting coefficients w_1, w_2, w_3 are given by our clients based on their needs.
2. Scaling
   Accuracy is in [0, 1],
   Response Time is in [0, inf] and we take inverse and logit of it to transfer to [0, 1],
   Memory Usage is in [0, 1]
3.

Finally, we can rank those algorithms using score and take first 3 of them to make recommendation.

Model Validation

We will repeat the above steps but run Spark benchmark for four algorithms, including OLS, Lasso, Ridge, and RF on IBM cluster. Our client, BEZNext will collaborate with measurement data collection. With the new measurement data in hand, we will build new models and final recommender.

Reference

[1] Yuri, Balasanov. "Week 1 Workshop: Linear Regression With Large Number Of Predictors". *MScA, Machine Learning and Predictive Analytics (31009)*. Retrieved 3 April 2017, from http://ilykei.com/api/fileProxy/documents%2FAdvanced%20Data%20Mining%20and%20Predictive%20Analytics%2031009%2FLecture%201%2FMScA_AdvancedDataMining_Week1_Workshop1.html

Summary

Value for Collaborators

- Free service
- Reduce uncertainty and risk
- Framework for conducting own benchmarks
- Option of obtaining additional services related to application development, performance management and capacity planning

Value for Customers

- Free service
- Reduce uncertainty and risk
- Framework for conducting own benchmarks
- Option of obtaining additional services related to application development, performance management and capacity planning

THANK YOU!
ARE ANY QUESTIONS?

Boris Zibitsker, bzibitsker@beznext.com