XGBoost -AK-47 in your machine learning models arsenal

Mateusz Łuczko, SER XIX

pwc

BIO

- Maths, Computer Science & Econometrics graduate (Warsaw University)
- PwC Data Analytics since 2015
- 2013 – 2015: PZU (insurance risk management)
- Main focus on financial sector and geospatial modeling

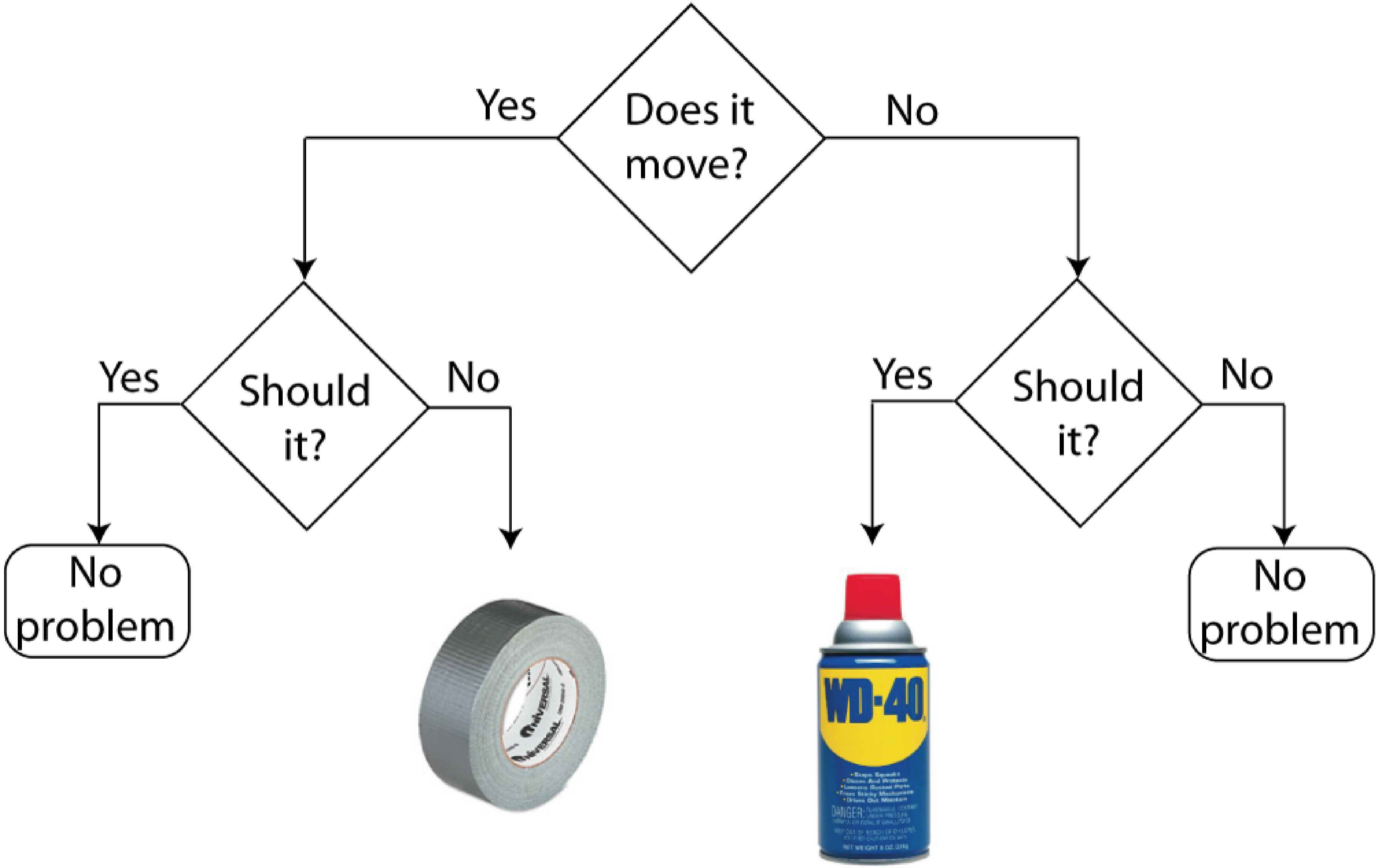
- Nerd by heart

pwc

Agenda

- Decision trees & ensembling
- XGBoost – Genesis
- Algorithm explanation
- Benchmarks
- R demo

Laboratory Troubleshooting Flowchart



Decision trees

Pros:

- **interpretable!!!**
- can handle missing data
- very fast at testing time: O(depth)

Cons:

- prone to overfitting
- unstable

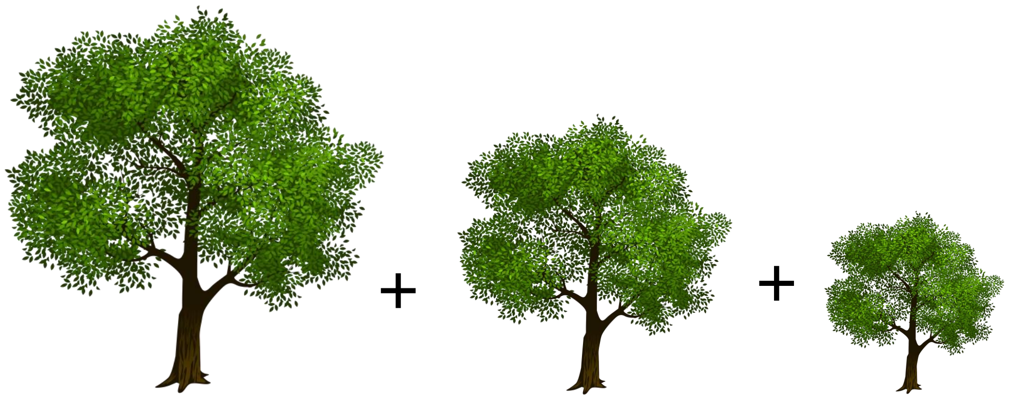
Ensembling!!!

pwc

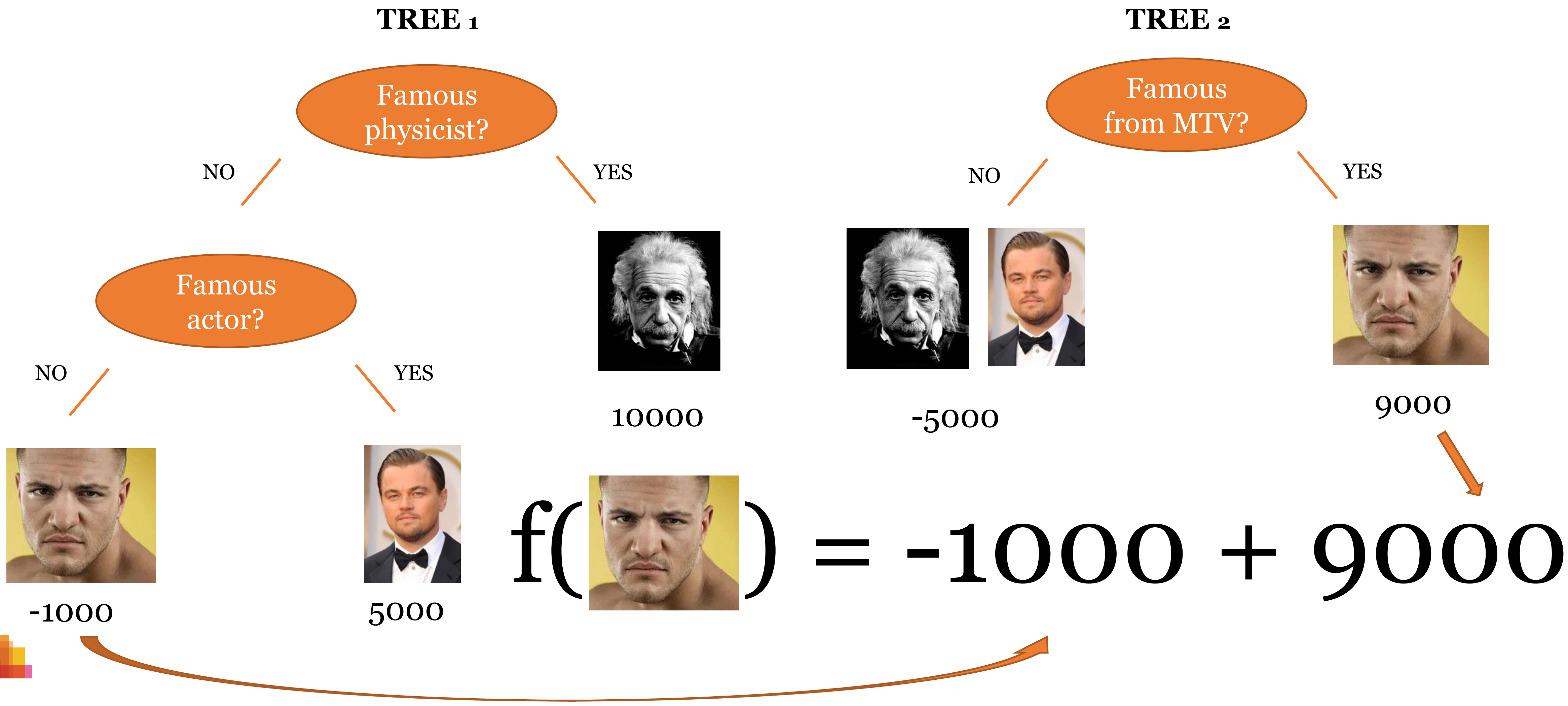
In ensembling we trust

Random Forest

Gradient boosted Regression/Decision Trees



Boosting - example



kaggle

pwc

2012-...

Chucking everything into a Random Forest: Ben Hamner on Winning The Air Quality Prediction Hackathon

Kaggle Team | 05.01.2012

Random Forest of 'Give Me Some Credit' Survey Results

Margit Zwemer | 04.02.2012

pwc

...-2016

Profiling Top Kagglers: Gilberto Titericz, New #1 in the World

Triskelion | 11.09.2015

What are your favorite machine learning libraries?

I have many favourite libraries, but I will try to rank them according the ones I use most:

- XGBoost - Fast and optimized GBM implementation.
- Vowpal Wabbit - Very fast with lots of parameters and linear algorithms.
- LibFM - Factorization Machines.
- scikit-learn - Lots of functions and algorithms.
- Lasagne - Very good and fast NN implementation (but unfortunately I don't have a GPU).
- Matlab Neural Network Toolbox - Classic.
- R GBM, randomForest and glm - Classic.

Profiling Top Kagglers: KazAnova, New #1 in the World

Triskelion | 02.10.2016

What are your favorite machine learning libraries?

1. Scikit for forests.
2. XGBoost for GBM.
3. LibLinear for linear models.
4. Weka for all.
5. Encog for neural nets.
6. Lasagne for nets, although I learnt it very recently.
7. RankLib for functions like NDCG.

It used to be random forest that was the big winner, but over the last six months a new algorithm called Xgboost has cropped up, and it's winning practically every competition in the structured data category.

Anthony Goldbloom, CEO Kaggle (12.2015)

pwc

In theory…

XGBoost (eXtreme Gradient Boosting) is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

pwc

In practice...

- Speed!!!
- State-of-the-art results on a wide range of problems:
  - store sales prediction,
  - web text classification,
  - customer behavior prediction,
  - ad click through rate prediction,
  - malware classification,
  - product categorization
- Interfaces in R, Python, C++, Julia, Java, Scala, Spark, Flink

- Large number of hyperparameters to tune ☹

Algorithm (in 3 „easy” steps)

$$Obj = \sum_{i=1}^{n} l(y_i, \sum_{k=1}^{K} f_k(x_i)) + \sum_{k=1}^{K} \Omega(f_k) \qquad f_k \in \mathcal{F}$$

Loss functions:

- RMSE
- binary / multiclass log-loss
- binary / multiclass misclassification rate
- auc
- customized functions

Regularization

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

- **number of leaves (T)**
- **L2 – norm of leaf scores**
- learning rate
- maximum tree depth
- minimum number of instances in node
- columns subsampling
- observations subsampling

Algorithm (in 3 „easy” steps)

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$$

$$\simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) + constant$$

$$= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) + constant$$

...

$$w_j^* = -\frac{G_j}{H_j+\lambda} \qquad Obj = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j+\lambda} + \gamma T$$

Algorithm (in 3 „easy" steps)

We grow each tree in greadily manner:

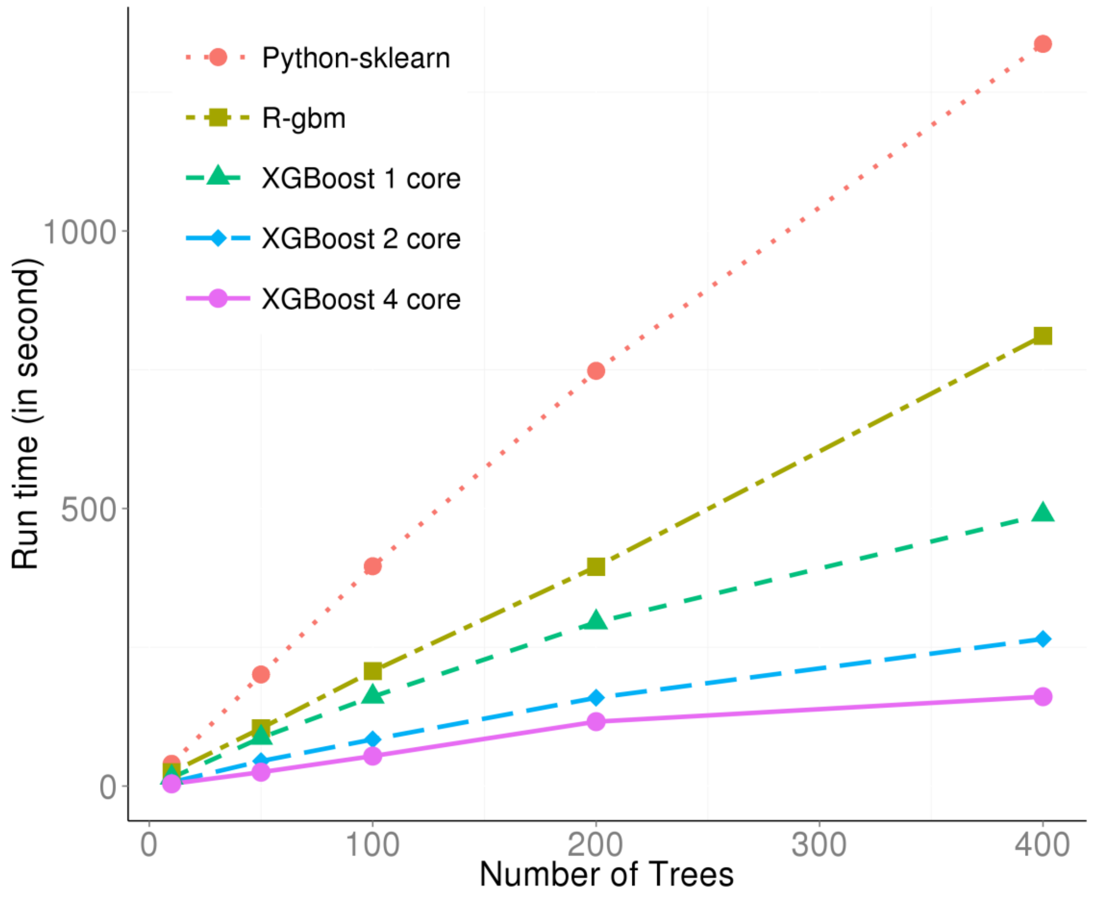
- Start with one node
- For each node make a split (using one of the variable); the change of objective is equal to:

$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

- For given node, take the variable with highest gain
- Repeat until zero/negative gain for every node

pwc

Benchmarks



data from: Higgs Boson Competition (250000 observations, 30 features)
source: http://www.r-bloggers.com/an-introduction-to-xgboost-r-package/

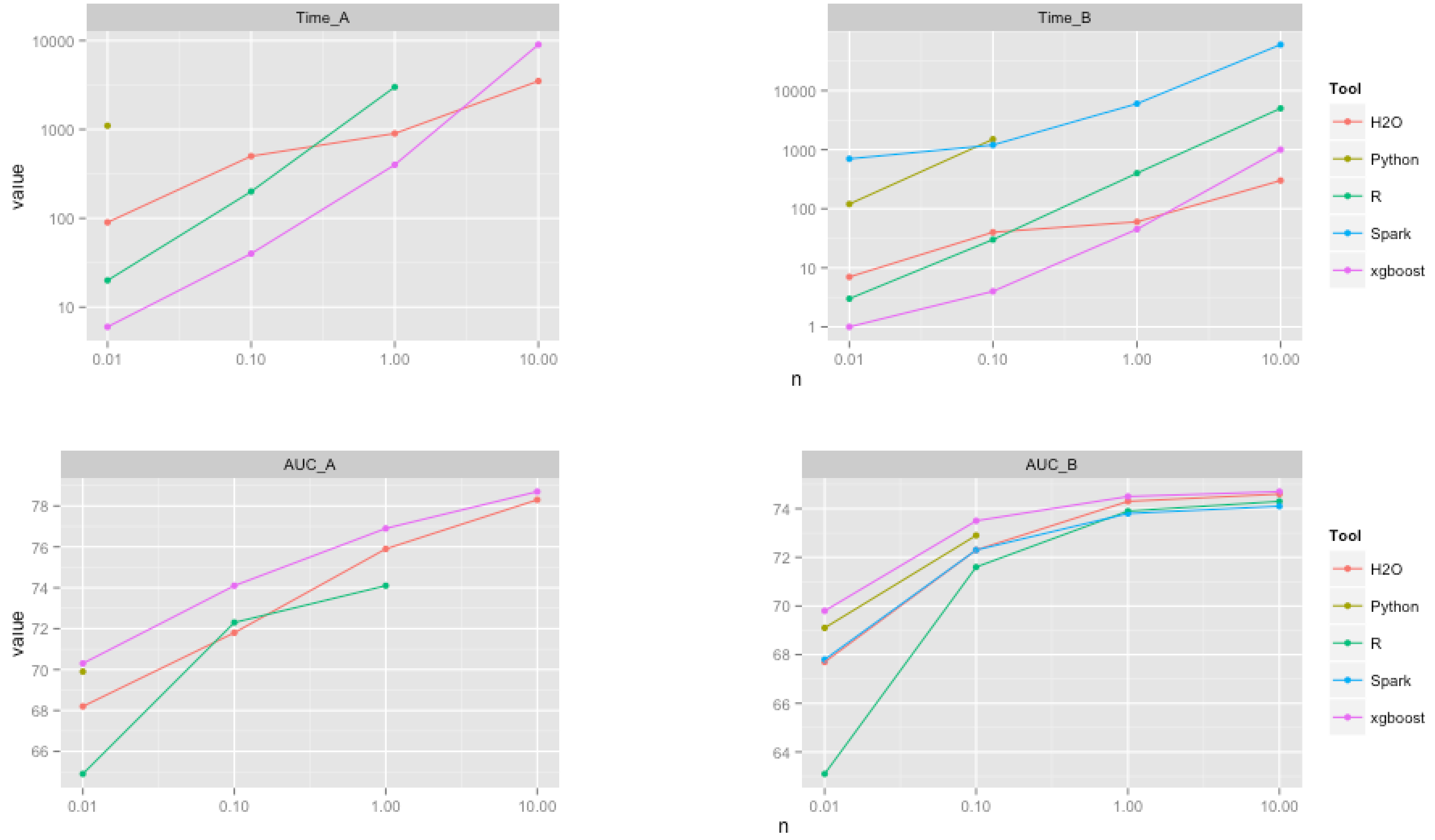
Benchmarks

More objective benchmark (https://github.com/szilard/benchm-ml)

- Training datasets from well – known airline dataset (predict whether a flight will be delayed by more than 15 minutes)
- Sample sizes: 10K, 100K, 1M, 10M (ca 1000 features)
- Setup: Amazon EC2 c3.8xlarge instance (32 cores, 60GB RAM)
- Open source implementations:
  - R (gbm)
  - Python (GradientBoostingClassifier)
  - Spark Mllib (GradientBoostedTrees)
  - H2O (h2o.gbm)
  - XGBoost

pwc

Benchmarks



Time_A
Time_B
AUC_A
AUC_B
value
n
0.01
0.10
1.00
10.00
10000
1000
100
10
1
78
76
74
72
70
68
66
64
Tool
H2O
Python
R
Spark
xgboost

R

References

- http://xgboost.readthedocs.io/en/latest/
- https://cran.r-project.org/web/packages/xgboost/xgboost.pdf
- http://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf
- https://github.com/dmlc/xgboost/

That's all Folks!