Boosting

Machine Learning for Data Science 1

bagging

bootstrap aggregating

Ensembles reduce the loss on the test data

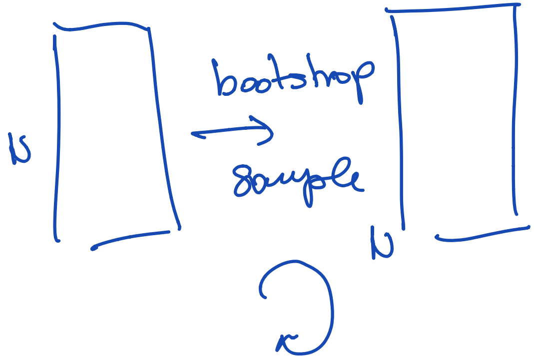
improve accuracy

~~interpretability~~

no covariance

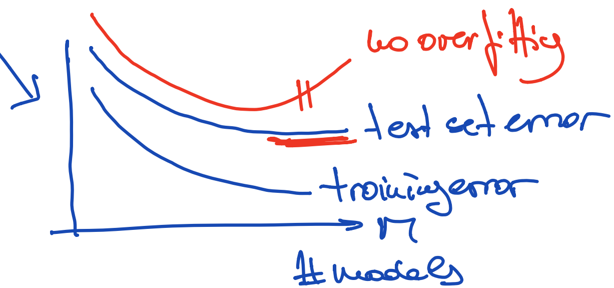
0

$$\sigma_{\bar{x}}^2 = \frac{\sum \sigma_i^2 + 2\sum\sum \mathrm{cov}(x_i, y_i)}{n^2}$$



$$\frac{\sum_{i=1}^{M} f_i(x)}{M}$$

: average



AdaBoost.M1 : Shapire 1997 : classification, binary

$G_m(x) \in \{-1, 1\}$

1. Initialize training data weights
   $w_i = 1/N \quad i = 1..N$

2. for $m = 1$ to M

   fit $G_m(x)$ to the weighted training data

   compute $err_m = \dfrac{\sum w_i I(y_i \neq G_m(x_i))}{\sum w_i}$

   compute $\alpha_m = \log \dfrac{1 - err_m}{err_m}$ → accuracy → error

   set $w_i \leftarrow w_i \, e^{\alpha_m I(y_i \neq G_m(x_i))}$

   not independent

   for correct predictions, weights are not changes

3. output

   → $G(x) = \text{sign}\left[\sum_{i=1}^{M} \alpha_m G_m(x)\right]$

   for incorrect predictions

   $\dfrac{1 - err}{err}$

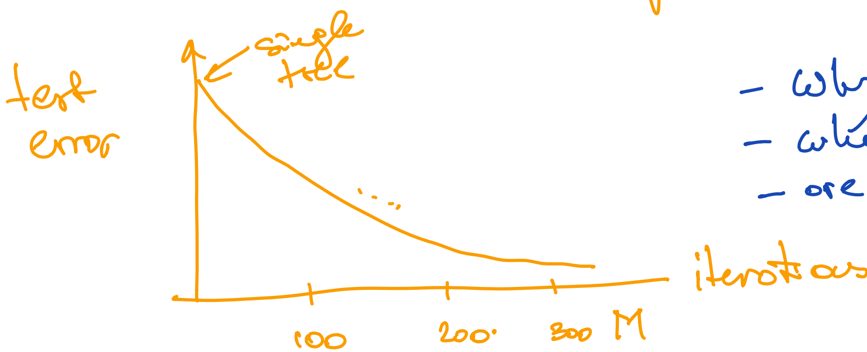
   similar to bagging

$$\alpha_1 G_1(x) + \alpha_2 G_2(x) + \alpha_3 G_3(x) + \alpha_4 G_4(x) = G(x)$$

training data $\longrightarrow$ weighted sample $\longrightarrow$ weighted sample $\longrightarrow$ - - -

AdaBoost.M1 : increases the weights of misclassified data instances

dramatic increase of performance $\rightarrow$ accuracy



- why?
- what are we optimizing? $\rightarrow$ L
- are there any other algorithms of this kind?

$\Downarrow$

<u>powerfull</u>

Boosting & Additive models

$G_1, G_2, G_3 \dots$

$$G(x) = \text{sign}\left(\sum \alpha_m G_m(x)\right)$$

boosting, we expand the approximation one model at a time

basis function expansion

$$f(x) = \sum_{m=1}^{M} \beta_m b(x, \gamma_m)$$

β_m: a weight

γ_m: a set of parameters

$$* \min_{\{\beta_m, \gamma_m\}_1^M} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{M} \beta_m b(x_i, \gamma_m)\right)$$

$M = 1000$

computationally intensive optimization

(optimizing all parameters at once)

(PROBLEMATIC)

ALL AT ONCE

Forward Stagewise Additive Modelling

Approximates the solution to $\min_{\beta_m, \gamma_m} L\left(\sum \ldots\right)$
by adding a new function
to the expansion
w/o adjusting the coefficients
already inferred

1. Initialize $f_0(x) = 0$
2. for $m = 1$ to M

   compute $\beta_m, \gamma_m = \arg\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta\, b(x_i, \gamma))$

   set $f_m(x) = f_{m-1}(x) + \beta_m b(x, \gamma_m)$

   ↳ correct the output of the previous model (expansion)

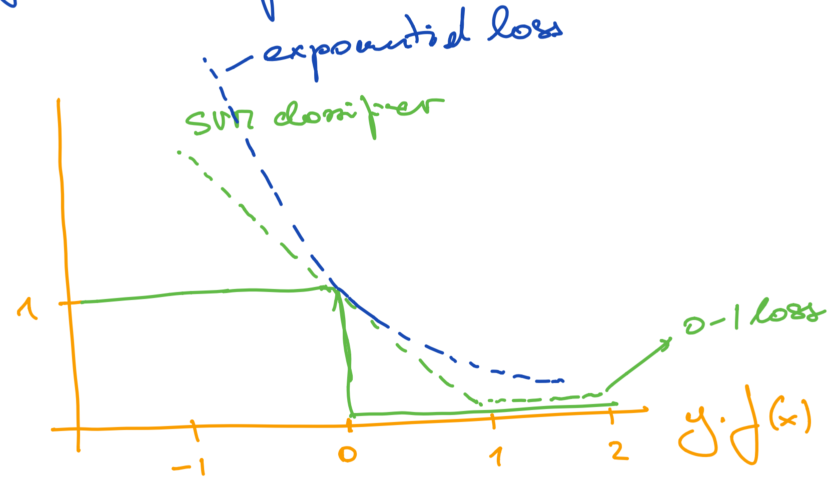
3. output $f_M(x)$

AdaBoost.M1 and Loss Function

What is the loss function optimized by AdaBoost?
Can similar procedures be used for other loss functions?

AdaBoost ≡ forward stepwise additive modelling that uses exponential loss — | +5 years

$$L(y, f(x)) = \exp(-y f(x))$$



AdaBoost.M1: $G_m(x) = \{-1, 1\}$

For exponential loss, we have to solve:

$$\beta_m, G_m = \arg\min_{\beta, G} \sum_{i=1}^{N} \exp[-y_i (f_{m-1}(x_i) + \beta G(x_i))]$$

$$= \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta \underbrace{y_i G(x_i)}_{=1})$$

$e^{-\beta}$

$$w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$$

a weight applied to each observation changes with each iteration

$$= \arg\min e^{-\beta} \sum_{y_i = G(x_i)} w_i^{(m)} + e^{+\beta} \sum_{y_i \neq G(x_i)} w_i^{(m)}$$

$$= \arg\min_{\beta} \left[(e^{\beta} + e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G_m(x_i)) + e^{-\beta} \sum_{i=1}^{N} w_i^{(m)} \right]$$

E

$G_m = ?$

$$G_m = \arg\min_{G_m} \sum w_i^{(m)} I(y_i \neq G_m(x_i))$$

classifier that minimizes weighted error rate what is β?

$$\frac{\partial E}{\partial \beta} = \frac{d\left(\sum_{y=G(x_i)} w_i^{(m)} e^{-\beta} + \sum_{y \neq G(x_i)} w_i^{(m)} e^{\beta}\right)}{d\beta} = 0$$

$$\beta_m = \overbrace{\frac{1}{2} \log \frac{1-\varepsilon}{\varepsilon}}^{\dots}$$

$$\varepsilon = \frac{\sum w_i^{(m)} I(y_i \neq G_m(x_i))}{\sum w_i^{(m)}}$$

looks familiar

$$2\beta_m = \alpha_m$$

The update of approximation

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{-\beta_m y_i G_m(x_i)}$$

$$-y_i G_m(x_i) = 2 I(y_i \neq G_m(x_i)) - 1$$

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\underbrace{2\beta_m}_{\alpha_m} I(y_i \neq G_m(x_i))} \cancel{e^{-\beta_m}}$$

exactly the weight update for AdaBoost M1

AdaBoost Recap

At each iteration

choose classifier that minimises weighted error

we use this to estimate error rate

we use this to compute the weights

improve the overall classifier with the new model

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x)$$

→ forward stepwise additive model that minimises

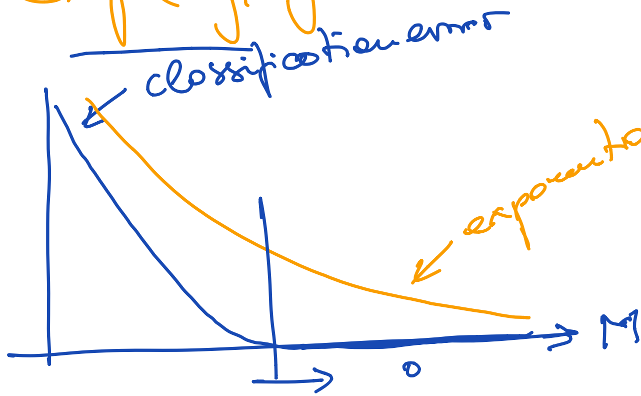
$$\boxed{L = \sum_i \exp(-y_i f(x_i))}$$

→ under change of loss function

↳ what is the training data

↳ what are the predictions

training error



Boosting Trees ↓ formalism

trees: portion feature space to disjoint regions R_j where a constant is assigned

$$x \in R_j \Rightarrow f(x) = \gamma_j$$

Formally

$$T(x;\Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

Θ

$$\Theta = \arg\min \underbrace{\sum_{j=1}^{J}}_{\text{all regions}} \underbrace{\sum_{i=1}^{N}}_{\text{all examples}} L(y_i, \gamma_j)$$

optimization problem

heuristics

find γ_j given R_j : $\hat{\gamma}_j = \bar{y}_i$ the mean

find R_j : greedy, top-down recursive partitioning

Boosted Tree Model

looks like bagging

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

induced in forward stepwise manner

at each stage

$$\Theta_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

given the regions

$$\gamma_{jm} = \arg\min_{\gamma_{jm}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma_{jm})$$

finding regions is difficult

but in some cases,
there is heuristic approach

$$\boxed{L(y, f(x)) = (y - f(x))^2}$$

$$L(y_i, f_{m-1}(x_i) + T_M(x_i)) =$$

$$= (\underbrace{(y_i - f_{m-1})}_{\text{residual of the current model } \varepsilon} - \underline{T_M(x_i)})^2$$

to minimize this

the tree should be fitted to the residual

$f_0(x) = \bar{y}$ ↑ training data → residual of f_0 ↓ fit T_1

$f_1 = f_0 + T_1$ ↑ residual of f_0 → residual of f_1 ↓ fit T_2

$\underline{f_0 + f_1 + T_2}$ ↑ residual of f_1

. . .

Numerical Optimization

$$L(f) = \sum_{i=1}^{N} L(y_i, f(x_i))$$

$f(x)$: is a sum of trees

Goal: $\min_{f} L(f)$

$$\hat{f} = \arg\min_{f} L(f)$$

Numerical optimizat

$$\vec{f}_M = \sum_{m=0}^{M} \vec{f}_m \qquad f_m \in \mathbb{R}^N$$

f_0: initial guess

f_m: induced based on f_{m-1}, which is the sum of previously induced vectors

Steepest descent

$$\vec{f}_m = \rho_m \cdot \vec{g}_m \quad : \text{gradient}$$

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i) = f_{m-1}(x)}$$

Loss	$-\partial L(y_i, f(x_i)) / \partial f(x_i)$
regression $\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$ = residual
regression $\lvert y_i - f(x_i) \rvert$	$\mathrm{sign}[y_i - f(x_i)]$
classif. deviance $\sum_{k=1}^{K} p_{ki} \log p_{ki}$	k-th component $I(y_i = G_k) - p_k(x_i)$
⋮	⋮

We fit the regression trees

Gradient Tree Boosting Algorithm (L)

1. Initialize $f_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

2. for $m=1$ to M

   compute $r_{im} = -\left[\dfrac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$

   fit regression tree to r_{im}

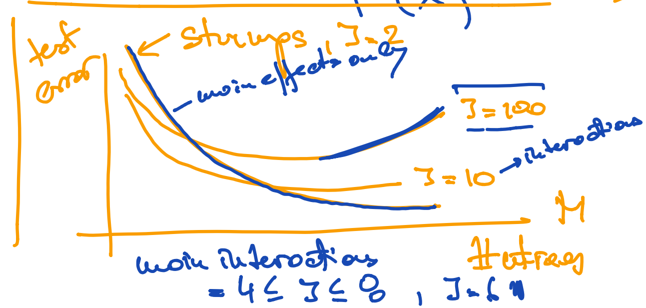
   for $j=1..J$ (regions) compute

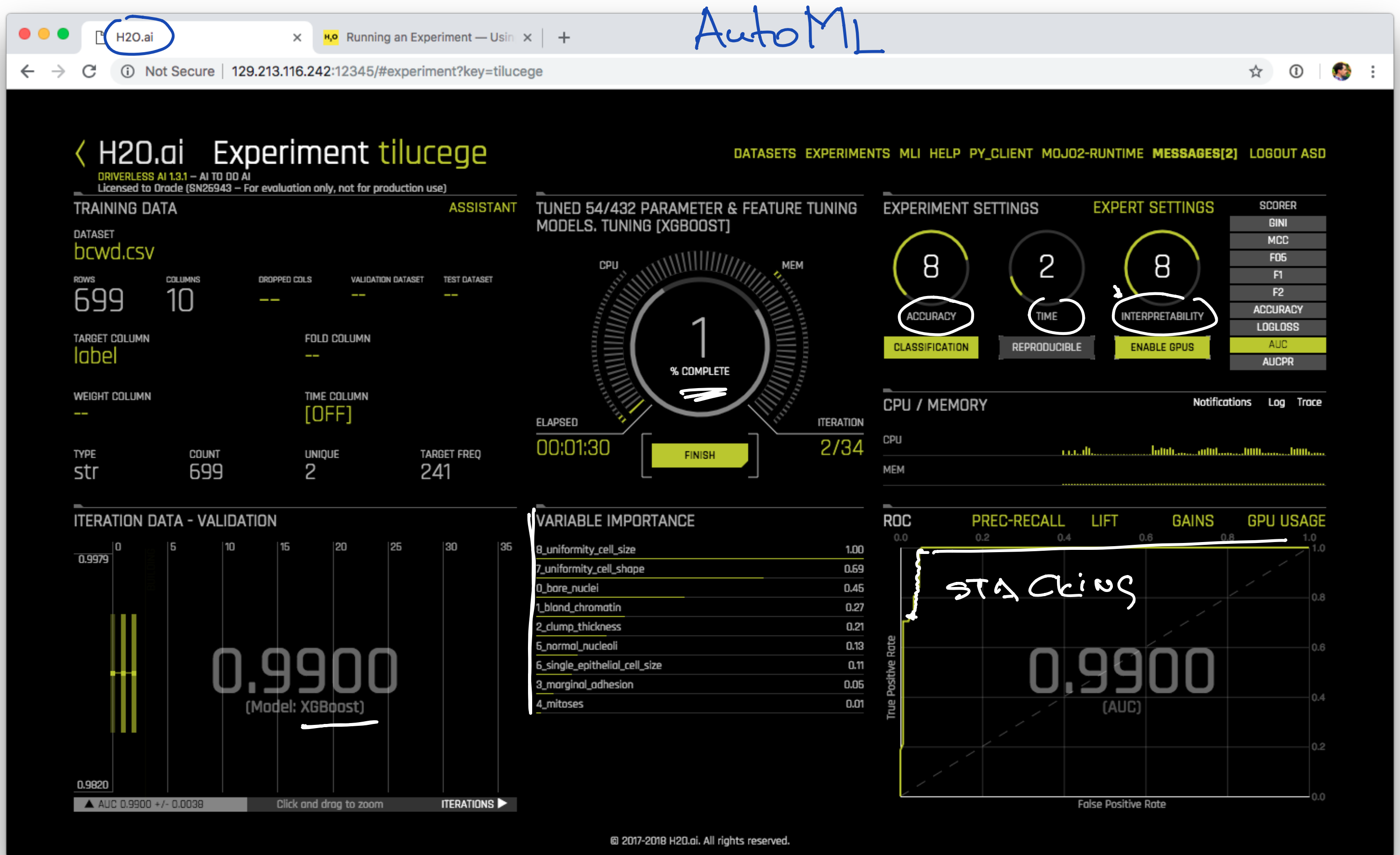
   $$\gamma_{jm} = \arg\min_{\gamma} \sum_{x\in R_{jm}} L(y_i, f_{m-1}(x_i)+\gamma)$$

   update $f_m(x) = f_{m-1}(x) + \underbrace{\sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_j)}_{T(x)}$

3. output $\hat{f}(x) = f_M(x)$

choice of the size of the trees




AutoML
H2O.ai
Running an Experiment — Usin
Not Secure
129.213.116.242:12345/#experiment?key=tilucege
H2O.ai Experiment tilucege
DRIVERLESS AI 1.3.1 – AI TO DO AI
Licensed to Oracle (SN26943 – For evaluation only, not for production use)
DATASETS EXPERIMENTS MLI HELP PY_CLIENT MOJO2-RUNTIME MESSAGES[2] LOGOUT ASD
TRAINING DATA
ASSISTANT
DATASET
bcwd.csv
ROWS 699
COLUMNS 10
DROPPED COLS --
VALIDATION DATASET --
TEST DATASET --
TARGET COLUMN label
FOLD COLUMN --
WEIGHT COLUMN --
TIME COLUMN [OFF]
TYPE str
COUNT 699
UNIQUE 2
TARGET FREQ 241
TUNED 54/432 PARAMETER & FEATURE TUNING MODELS. TUNING [XGBOOST]
CPU
MEM
1
% COMPLETE
ELAPSED
00:01:30
FINISH
ITERATION
2/34
EXPERIMENT SETTINGS
EXPERT SETTINGS
8 ACCURACY
2 TIME
8 INTERPRETABILITY
CLASSIFICATION
REPRODUCIBLE
ENABLE GPUS
SCORER
GINI
MCC
F05
F1
F2
ACCURACY
LOGLOSS
AUC
AUCPR
CPU / MEMORY
Notifications
Log
Trace
CPU
MEM
ITERATION DATA - VALIDATION
0.9979
0.9820
0.9900
(Model: XGBoost)
AUC 0.9900 +/- 0.0038
Click and drag to zoom
ITERATIONS
VARIABLE IMPORTANCE
8_uniformity_cell_size 1.00
7_uniformity_cell_shape 0.59
0_bare_nuclei 0.45
1_bland_chromatin 0.27
2_clump_thickness 0.21
5_normal_nucleoli 0.13
6_single_epithelial_cell_size 0.11
3_marginal_adhesion 0.05
4_mitoses 0.01
ROC
PREC-RECALL
LIFT
GAINS
GPU USAGE
STACKING
0.9900
(AUC)
True Positive Rate
False Positive Rate
© 2017-2018 H2O.ai. All rights reserved.

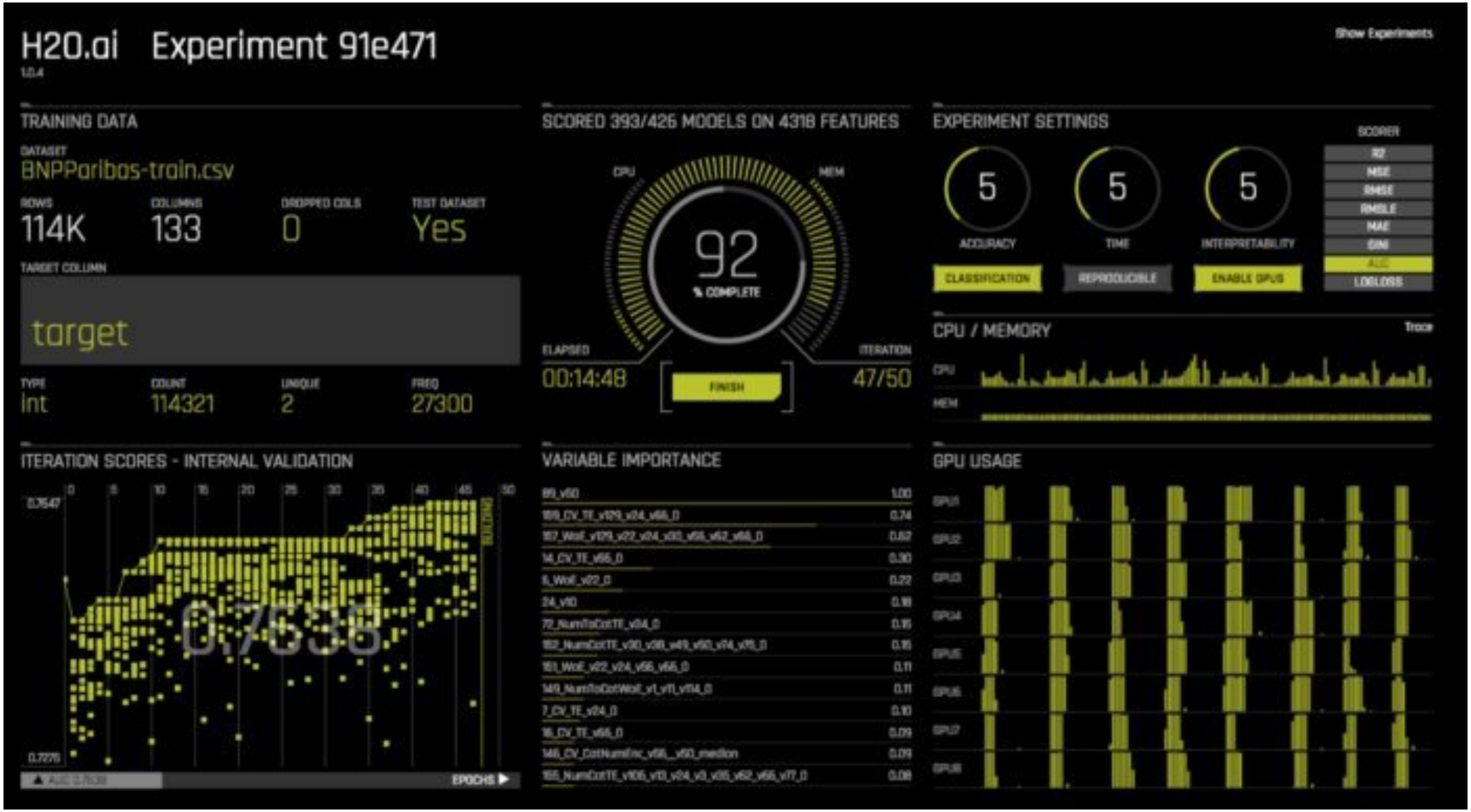

H2O.ai Experiment 91e471
Show Experiments
TRAINING DATA
DATASET
BNPParibas-train.csv
ROWS
114K
COLUMNS
133
DROPPED COLS
0
TEST DATASET
Yes
TARGET COLUMN
target
TYPE
int
COUNT
114321
UNIQUE
2
FREQ
27300
SCORED 393/426 MODELS ON 4318 FEATURES
CPU
MEM
92
% COMPLETE
ELAPSED
00:14:48
FINISH
ITERATION
47/50
EXPERIMENT SETTINGS
5
ACCURACY
5
TIME
5
INTERPRETABILITY
CLASSIFICATION
REPRODUCIBLE
ENABLE GPUS
SCORER
R2
MSE
RMSE
RMSLE
MAE
GINI
AUC
LOGLOSS
CPU / MEMORY
Trace
CPU
MEM
ITERATION SCORES - INTERNAL VALIDATION
0.7547
0.7636
0.7275
EPOCHS
VARIABLE IMPORTANCE
GPU USAGE
GPU1
GPU2
GPU3
GPU4
GPU5
GPU6
GPU7
GPU8

TABLE OF CONTENTS

Search...

Get Started with XGBoost

KAGGLE

This is a quick start tutorial showing snippets for you to quickly try out XGBoost on the demo dataset on a binary classification task.

Links to Other Helpful Resources

- See Installation Guide on how to install XGBoost.
- See Text Input Format on using text format for specifying training/testing data.
- See Tutorials for tips and tutorials.
- See Learning to use XGBoost by Examples for more code examples.

Python

```
import xgboost as xgb
# read in data
dtrain = xgb.DMatrix('demo/data/agaricus.txt.train')
dtest = xgb.DMatrix('demo/data/agaricus.txt.test')
# specify parameters via map
param = {'max_depth':2, 'eta':1, 'objective':'binary:logistic' }
num_round = 2
bst = xgb.train(param, dtrain, num_round)
# make prediction
preds = bst.predict(dtest)
```

v: latest