MODEL TUNING

HOW TO WIN A KAGGLE COMPETION

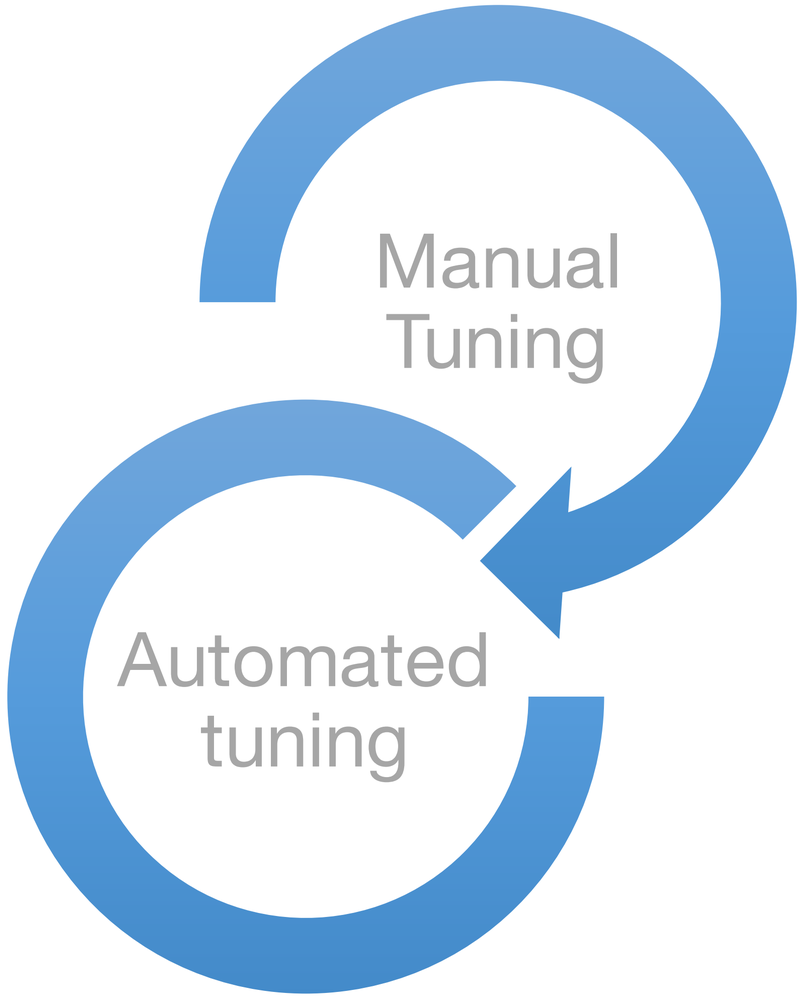
MODEL TUNING

TUNING A MACHINE LEARNING ALGORITHM IS OFTEN A “BLACK ART” THAT REQUIRES EXPERT EXPERIENCE, UNWRITTEN RULES OF THUMB, OR SOMETIMES BRUTE-FORCE SEARCH

Jasper Snoek

Author of “Practical Bayesian Optimization of Machine Learning Algorithms “

MANUAL TUNING VS AUTOMATED TUNING



Manual tuning

- Greedy search
- Tune one parameter at a time
- Use cross validation if possible
- Otherwise use holdout dataset

Automated tuning

- Tune multiple parameters at the same time
- Grid search, or
- Bayesian Optimization
- Findings discovered by manual tuning can be leveraged to determine parameter space for automated tuning

XGBOOST

THE NAME XGBOOST, THOUGH, ACTUALLY REFERS TO THE ENGINEERING GOAL TO PUSH THE LIMIT OF COMPUTATIONS RESOURCES FOR BOOSTED TREE ALGORITHMS. WHICH IS THE REASON WHY MANY PEOPLE USE XGBOOST.

Tianqi Chen

XGBOOST OVERVIEW

- State of art accuracy thanks to its enhanced
  - object function - **second order derivation** (GBDT uses only first order)
  - regularization - tree **complexity as regularization**
- Faster processing
  - Parallel processing
    - **Feature based**, as compared to RF which is tree based
    - Fit **Residual** between trees, vs Gradient (traditional GBDT, e.g. GBM)
    - Optimized memory usage - **pre-sorted blocks**
  - Smartly deal with **sparse features** and **missing values**
- Supports both **Tree** booster (gbtree) and **Linear** Booster (gblinear)

XGBOOST HYPER PARAMETERS – TREE COMPLEXITY

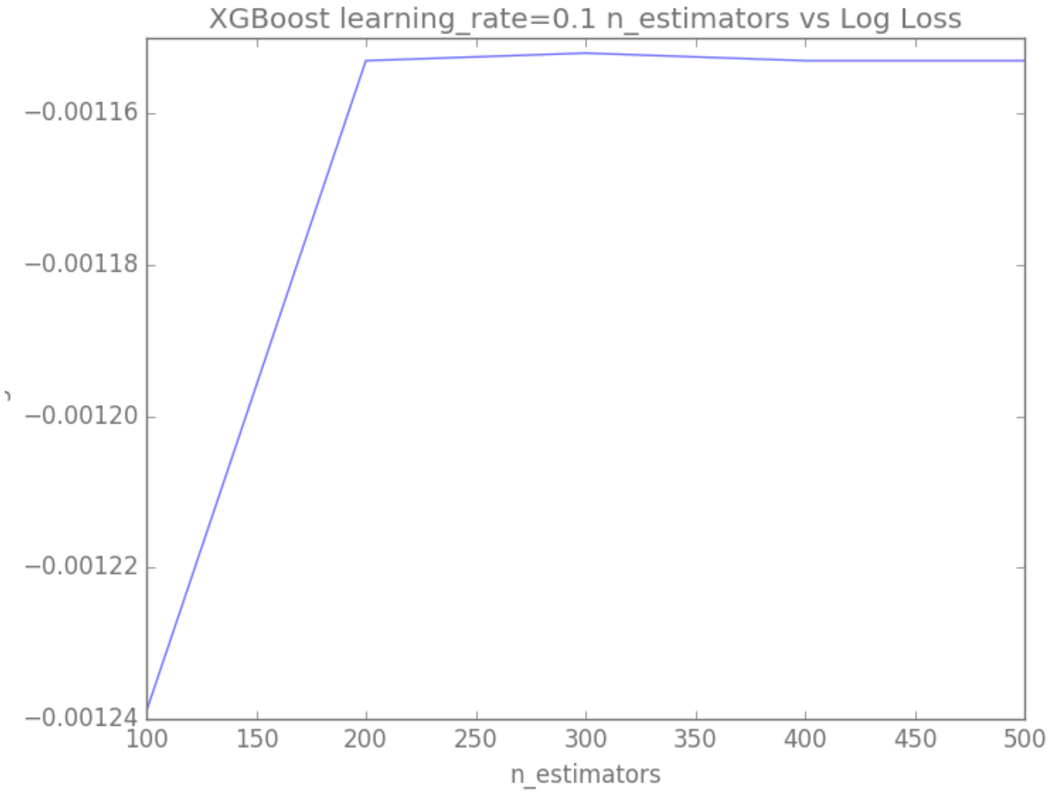
- **max depth** of the tress (max_depth)
  - Controls over-fitting as higher depth will allow model to learn relations very specific to a particular sample however can lead to over-fitting
  - Typical value range: 3 - 10
- **min child weight** of a leaf (min_child_weight)
  - Defines the minimum sum of weights (hessian) of all observations required in a child which is used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
  - Typical value: 1-20
- **gamma**
  - minimum loss reduction (number of leaves + L2 norm of leaf scores) required to make a further partition on a leaf node of the tree. the larger, the more conservative the algorithm will be.
  - **better** than min_child_weight in terms of control overfitting
  - Typical values: 0-2

XGBOOST HYPER PARAMETERS – STOCHASTIC GRADIENT BOOSTING

- **subsample**
  - Denotes the fraction of observations to be randomly samples for each tree.
  - Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting.
  - Typical values: 0.7-1
- **column sample by tree** (colsample_bytree)
  - Denotes the fraction of columns to be randomly samples for each tree.
  - Typical values: 0.3-0.9

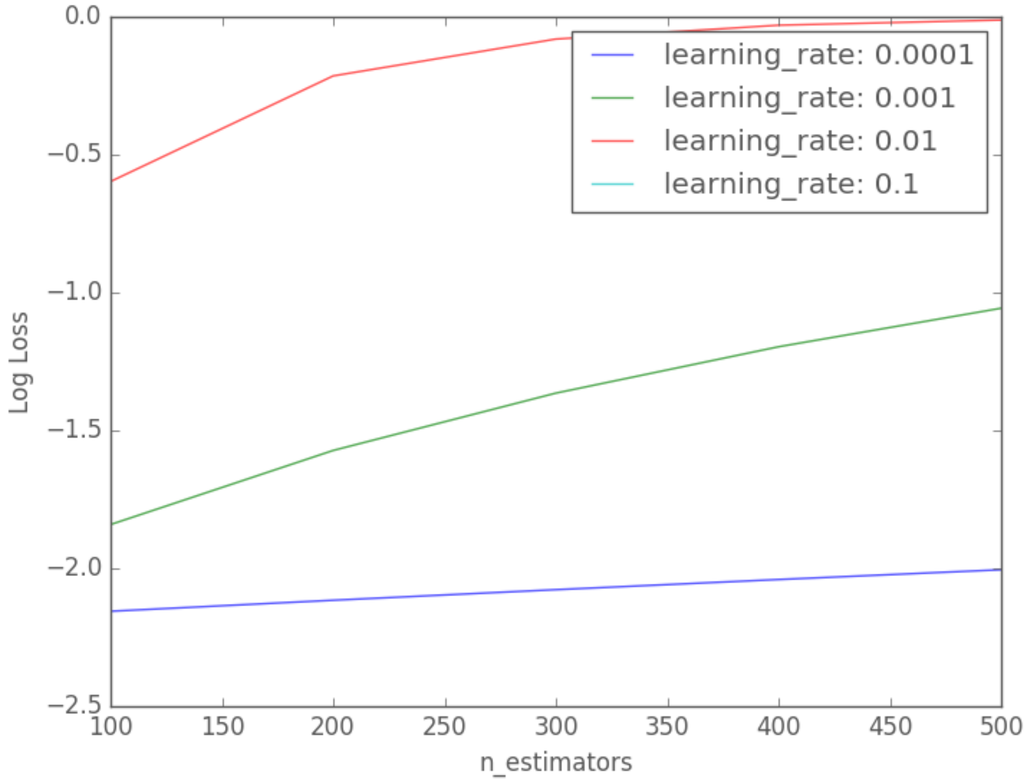
XGBOOST HYPER PARAMETERS – BOOSTING

- **number of trees** (n_estimators, num_of_round)
  - Typically the larger the better, however there's a turn-around point where the model will go overfitting afterwards.
  - The optimal number depends on other parameters, particularly on eta
  - Use xgb.cv and early stopping or for tuning



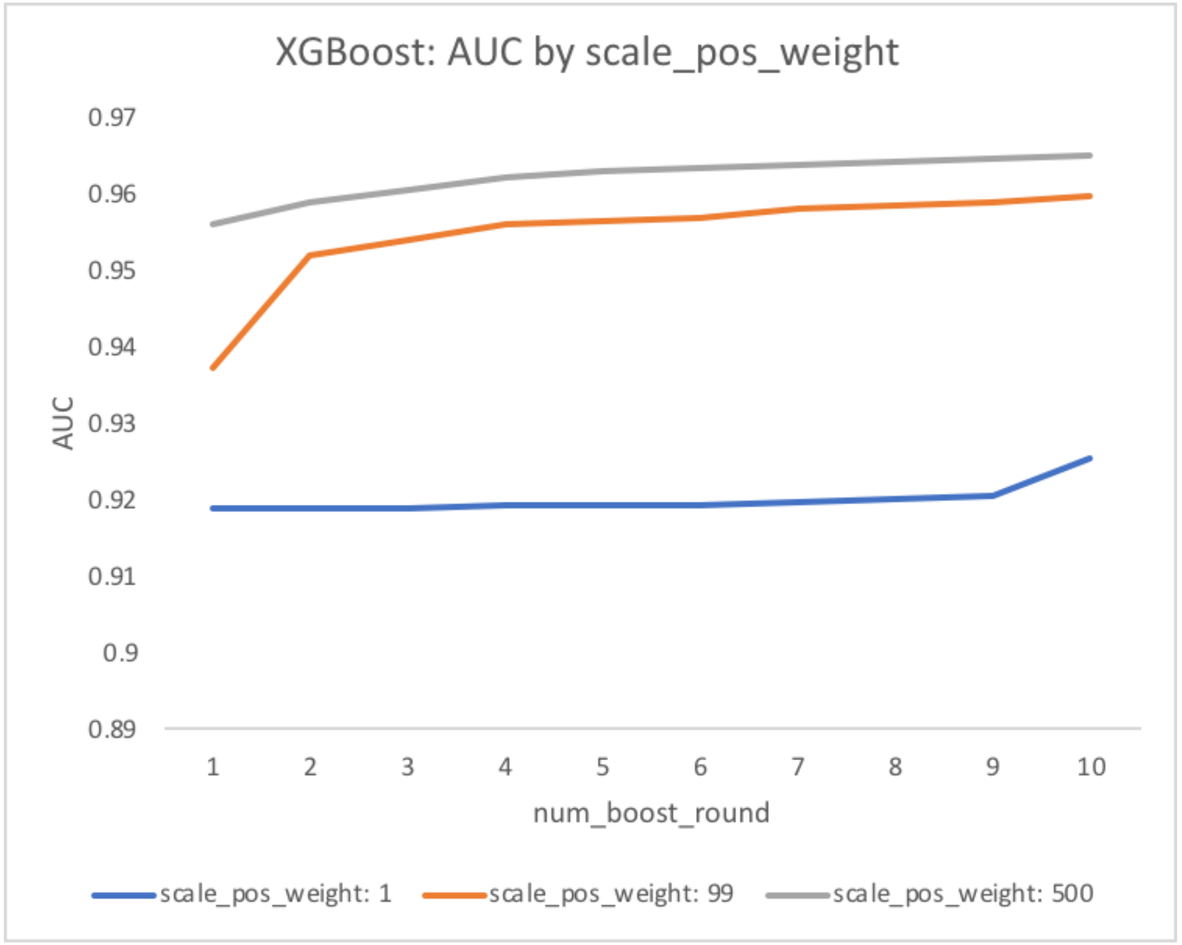
XGBOOST HYPER PARAMETERS – BOOSTING

- **learning rate** (learning_rate, eta)
  - **Smaller** learning rate yields **better** results but may needs more boosting rounds(trees) to converge.
  - Typical range: 0.01-0.1
  - Use **bigger** eta (0.1) to **tune** other parameters to save time
  - Use **smaller** eta to **train** the actual model once other parameter have been tuned.



IMBALANCED DATASET – SCALE_POS_WEIGHT

- **scale_pos_weight** , as a parameter, is available in XGBoost and LightGBM
- Recommended for imbalanced dataset, as compared to resampling approaches.
  - Avoid modifying training data, yet change the weight of positive observations.
  - Similar to cost-sensitive approach where the model learns to classify the rare event better by penalizing its incorrect prediction.
- Suggested to be set to **#of negative/#of positive records (**0.98/0.02≈500 in our case)



ADDITIONAL READINGS

- **XGBoost Parameters:**

  https://github.com/dmlc/xgboost/blob/master/doc/parameter.md

- **XGBoost official tuning guide**

  http://xgboost.readthedocs.io/en/latest/how_to/param_tuning.html

- **Tianqi Chen's guide on XGBoost in Chinese**

  http://www.52cs.org/?p=429

- **MachineLearningMastery's XGBoost tutorials**

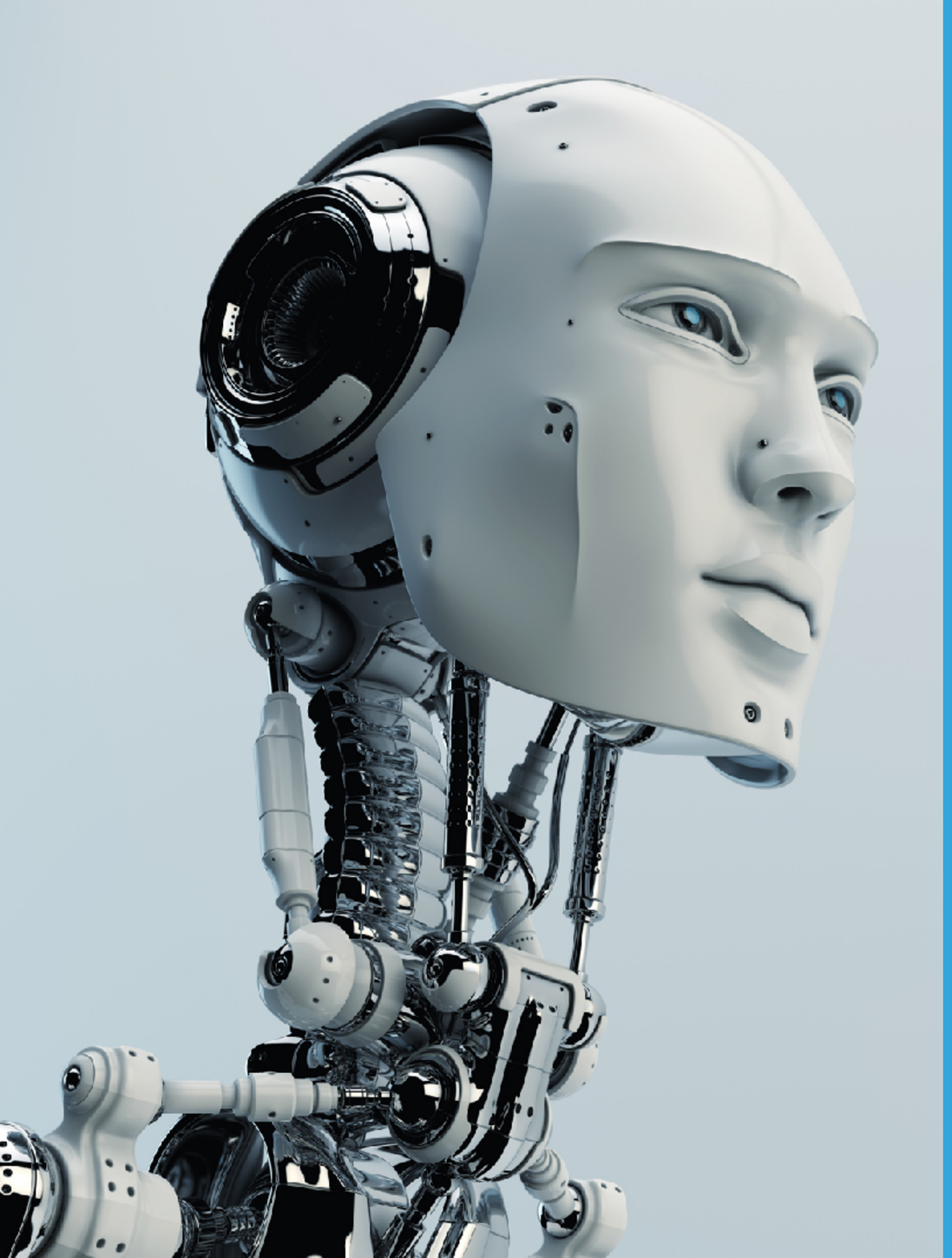
  http://machinelearningmastery.com/category/xgboost/

- **Complete Guide to Parameter Tuning in XGBoost**

  https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

- **What to optimize first? Gamma or Depth? What's up with min_child_weight?**

  https://medium.com/data-design/xgboost-hi-im-gamma-what-can-i-do-for-you-and-the-tuning-of-regularization-a42ea17e6ab6



LIGHTGBM

Best alternative to XGBoost

4-7x faster than XGBoost (with comparable parameters).

Subproject of Microsoft’s DMTK project.

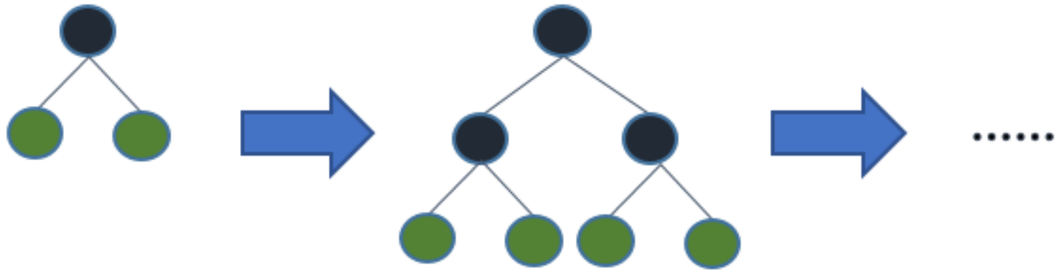
Developed by the winning team of Didi Data Science Competition

https://github.com/Microsoft/LightGBM

HIGHLIGHTS

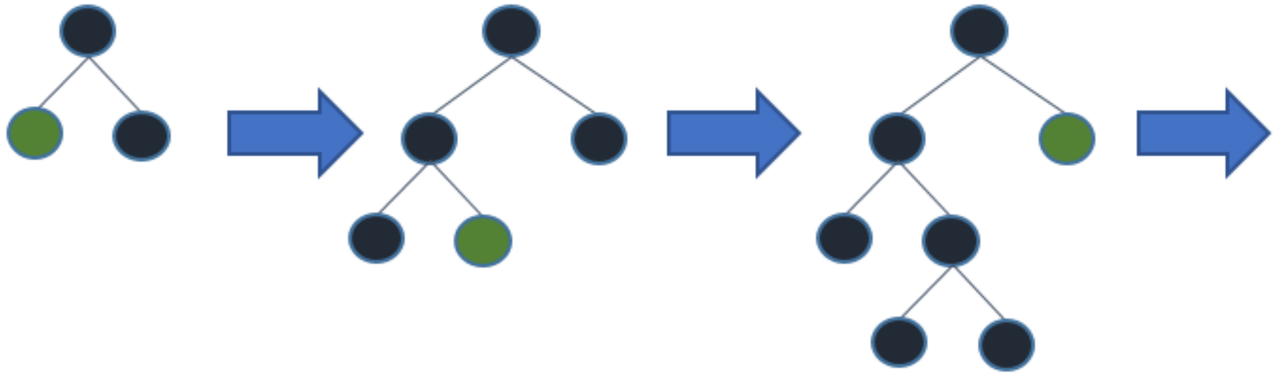
- Faster training speed and higher efficiency
- Lower memory usage
- Better accuracy
- Parallel learning supported
- Capable of handling large-scale data

LEVEL-WISE TREE VS LEAF-WISE TREE



Level-wise tree growth

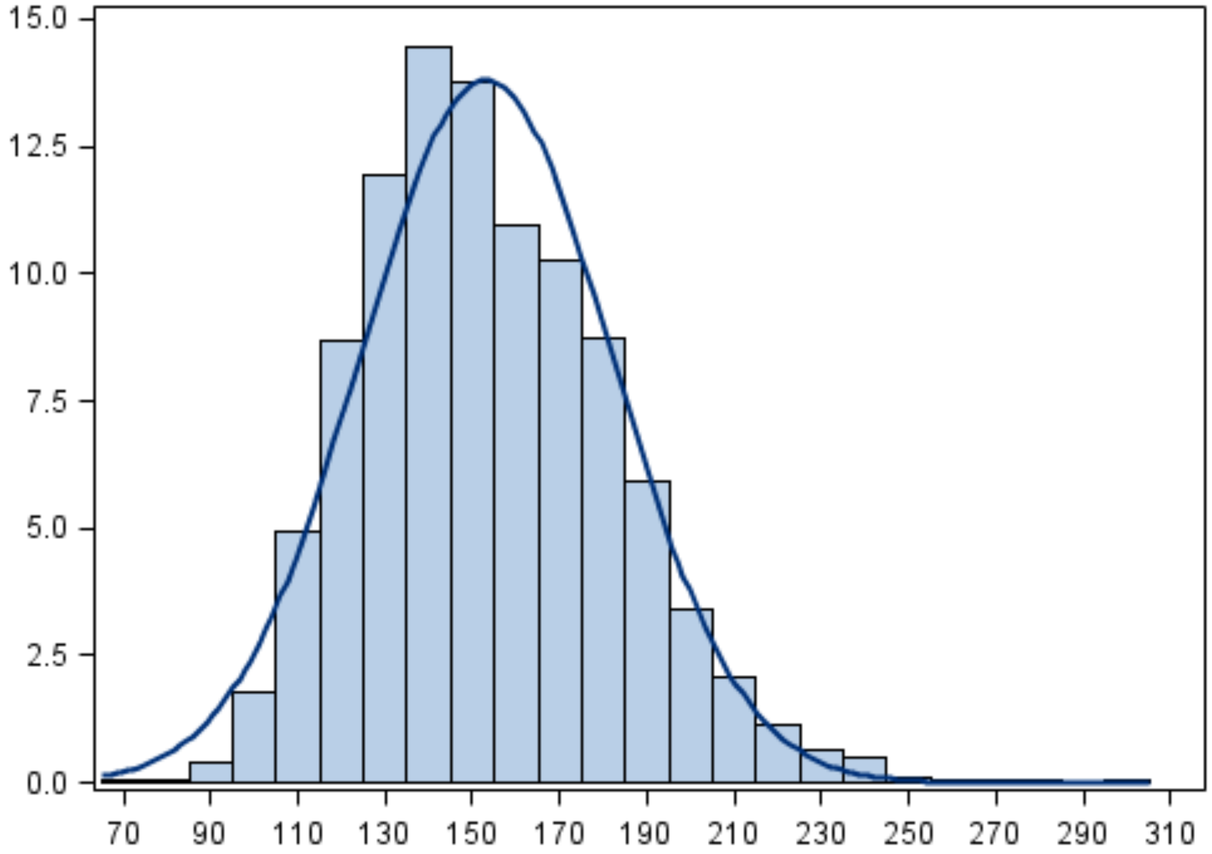
MAX_DEPTH



Leaf-wise tree growth

NUM_LEAVES

CONTINUOUS VS DISCRETE



DT/GBDT: no preprocessing on continuous data

XGBoost: pre-sort continuous data

LightGBM/XGBoost w/ hist: discretize continuous data into histograms

PARAMETERS

- learning_rate: shrinkage rate.
- num_iterations: number of boosting trees.
- max_bin: max number of bin that feature values will bucket in
- num_leaves: number of leaves in one tree
- min_sum_hessian_in_leaf (min_child_weight): minimal sum hessian in one leaf.
- feature_fraction(colsample_by_tree): fraction of features used for each iteration
- bagging_fraction(sample): fraction of samples used for each iteration
- bagging_freq: frequency for bagging.
- min_gain_to_split: the minimal gain to perform split. (Similar as gamma in XGB)

https://github.com/Microsoft/LightGBM/blob/master/docs/Parameters.md

XGBOOST/ LIGHTGBM GPU ACCELERATION

- GPU acceleration is available for both XGBoost and LightGBM
- Typically GPU versions are faster(2-10x depending on the datasets) but may sacrifice a bit of accuracy.


QUESTIONS?