Gradient Boosting

Boosting algorithm

1. Initialize $f(x) = 0, r = y$

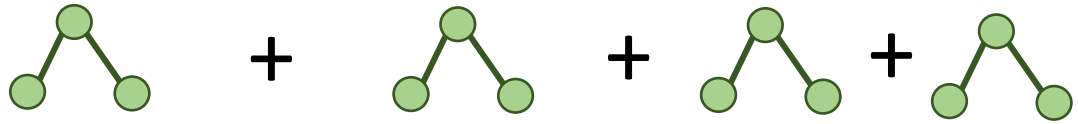
2. For $b = 1,2, \ldots, B$, repeat

   a) Fit a tree $f_b(x)$ to the training data (X, r)

   b) $f(x) \leftarrow f(x) + \lambda f_b(x)$

   c) $r \leftarrow r - \lambda f_b(x)$

3. output $\sum_{b=1}^{B} \lambda f_b(x)$

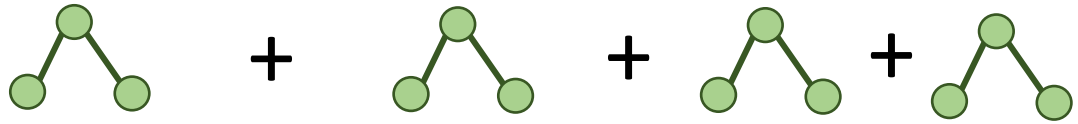


Gradient Boosting

$y - f(x)$

$L(x, y, y_p^{f(x)}) = \frac{1}{2}(y - f(x))^2$

1. Initialize $f(x) = 0, r = -g$
2. For $b = 1,2, \ldots, B$, repeat
   a) Fit a tree $f_b(x)$ to the training data (X, r)
   b) $f(x) \leftarrow f(x) + \lambda f_b(x)$
   c) $r \leftarrow r - \lambda f_b(x)$
3. output $\sum_{b=1}^{B} \lambda f_b(x)$



Gradient Boosting

1. Initialize $f_0(x_i) = \underset{\theta_0}{\operatorname{argmin}} \sum_{i=1}^{N} L(x_i, y_i; \theta_0)$

2. For $b = 1,2, \ldots, B$, repeat

   a) Calculate the negative gradient $r_{ib} = -\left.\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right|_{f=f_{b-1}}$

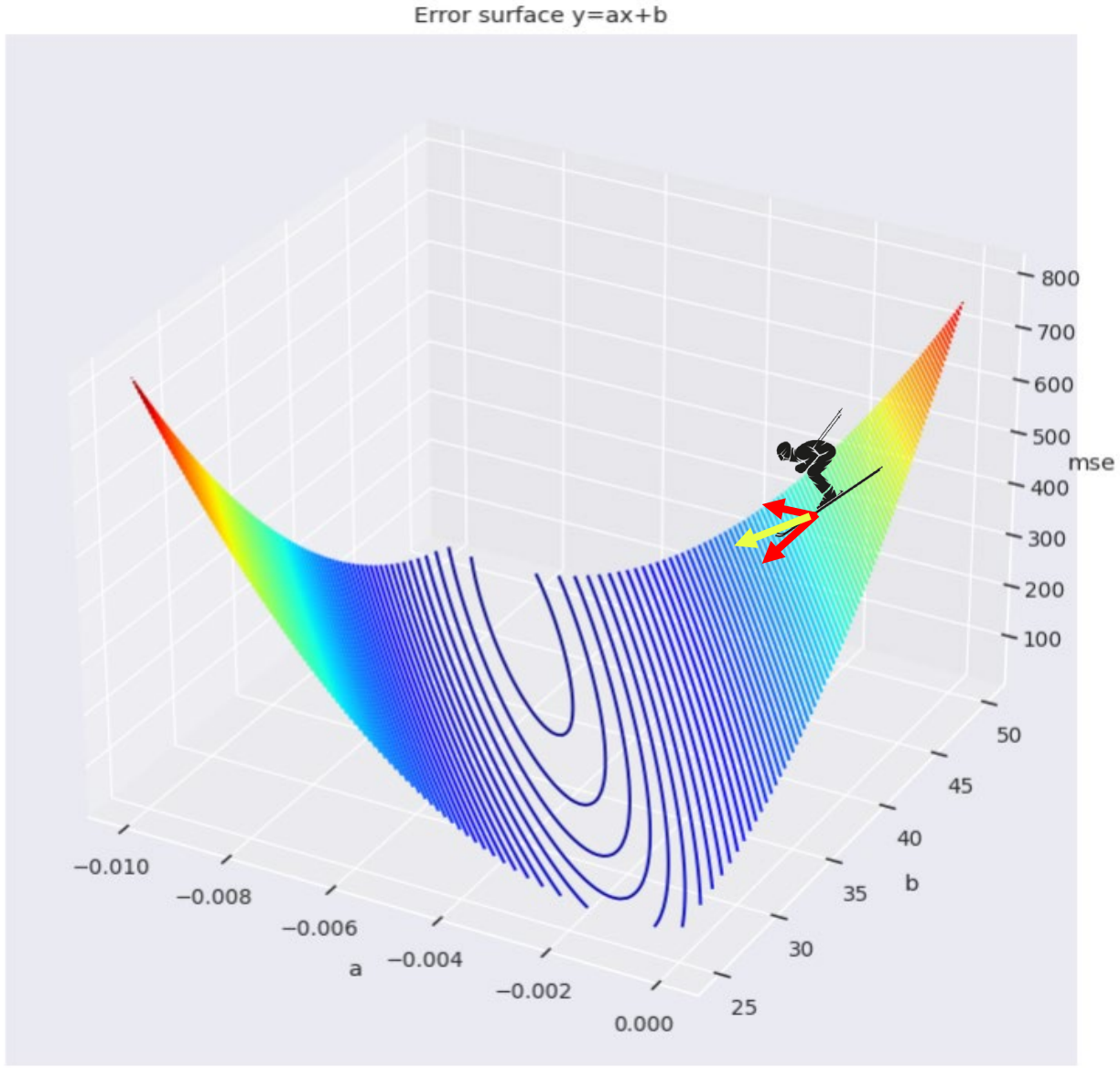
   b) Fit a tree $f_b(x_i)$ to the training data (x_i, r_{ib}) $\rightarrow \theta_b$

   c) Update loss $L(x, y; f_b)$

   d) Update function $f(x) \leftarrow f(x) + \lambda f_b(x)$

3. output $\sum_{b=1}^{B} \lambda f_b(x)$

Why Gradient?



Steepest descent

Performance Comparison

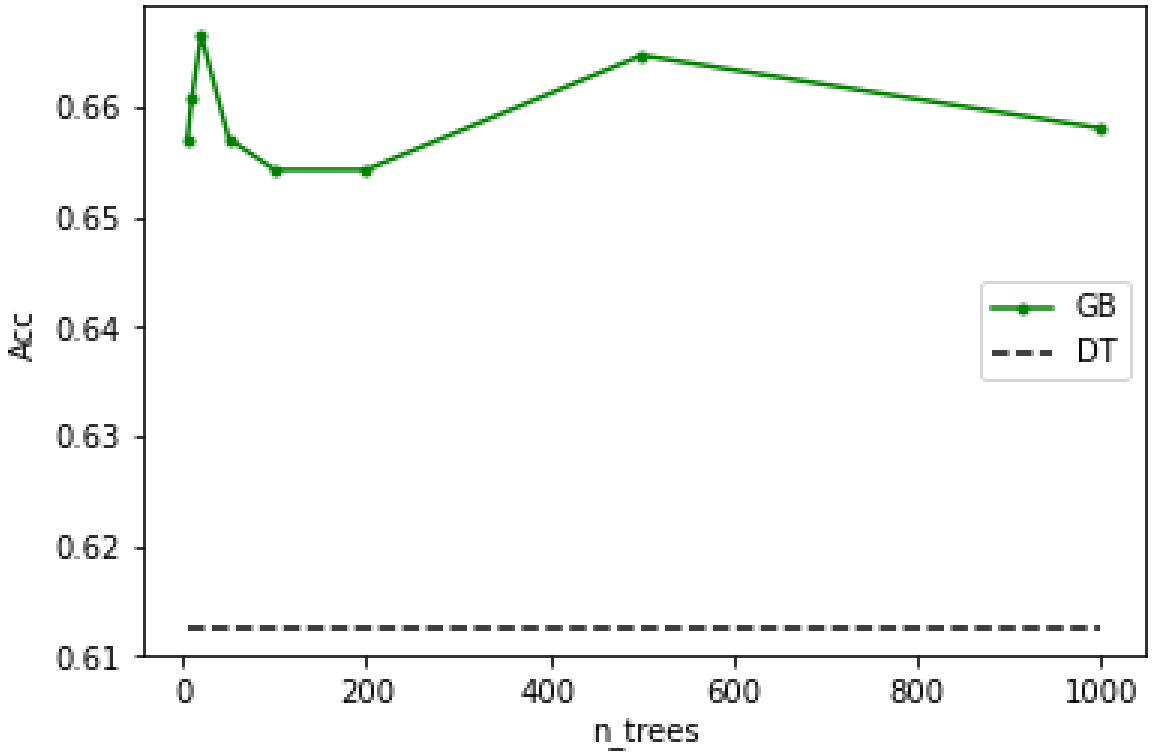
Data 1.

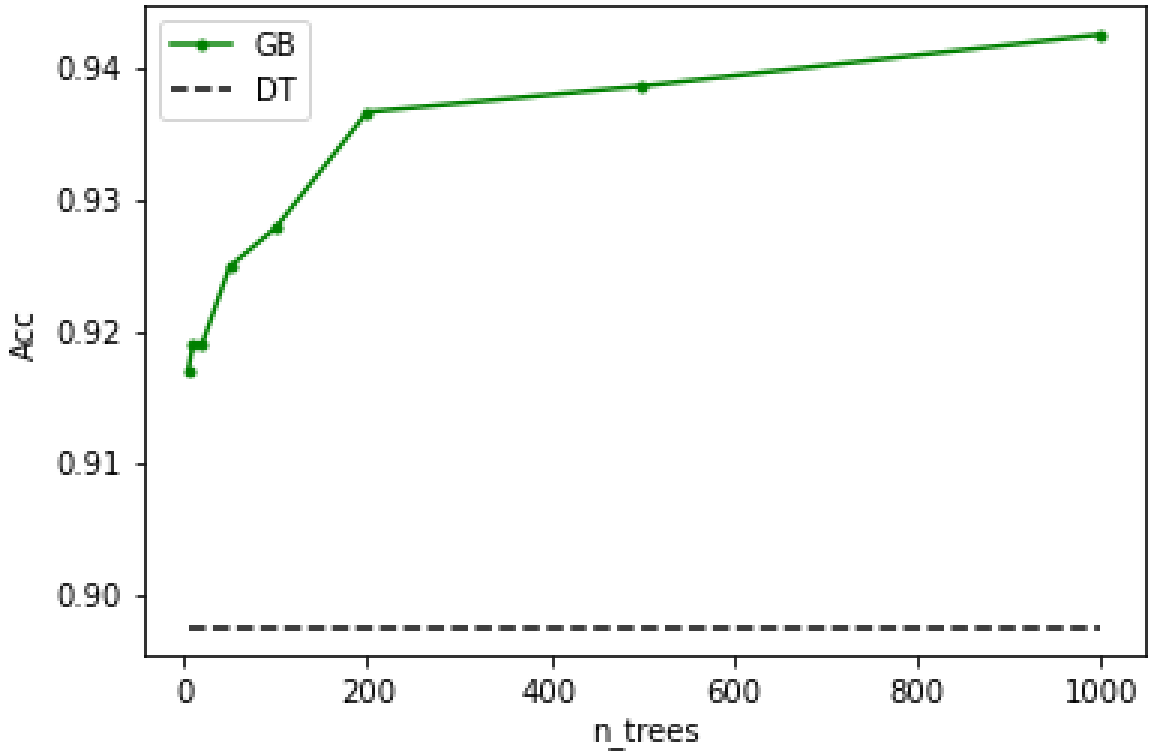
- 13 features,
- 5200+ samples
- DT performance 0.61

Data 2.

- 20 features,
- 5100+ samples
- DT performance 0.89

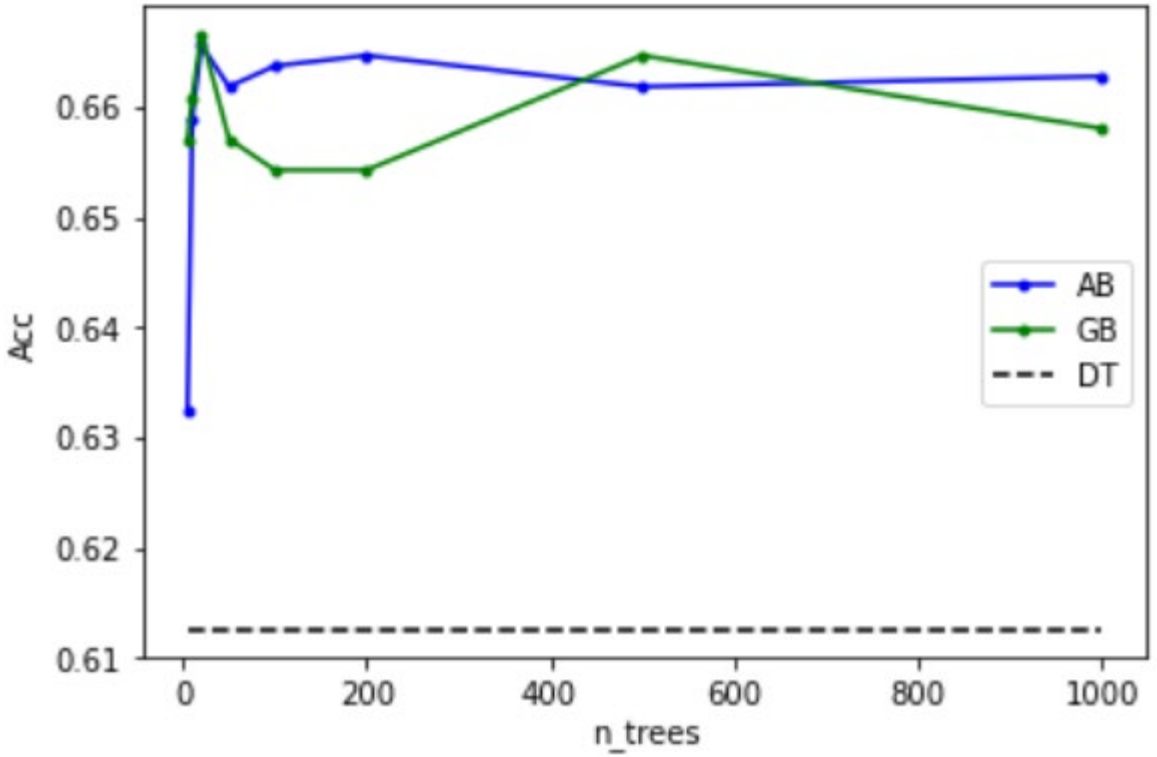
Performance Comparison

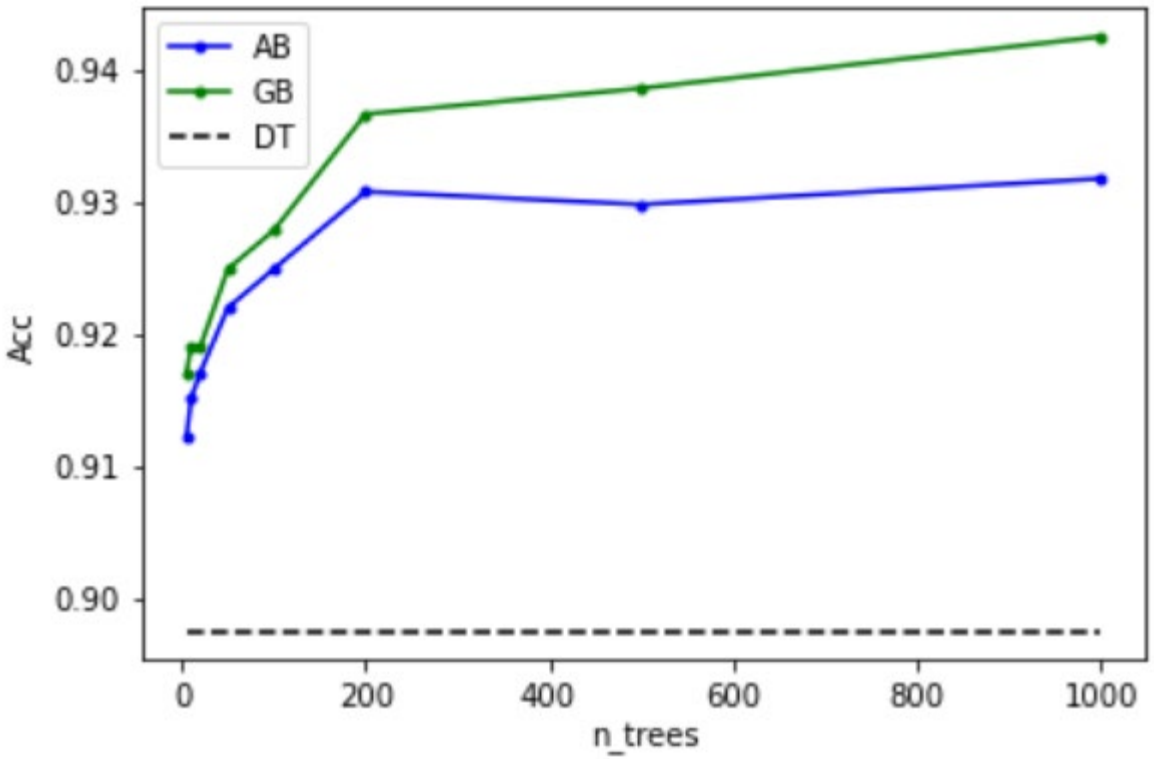




Performance Comparison /w AdaBoost

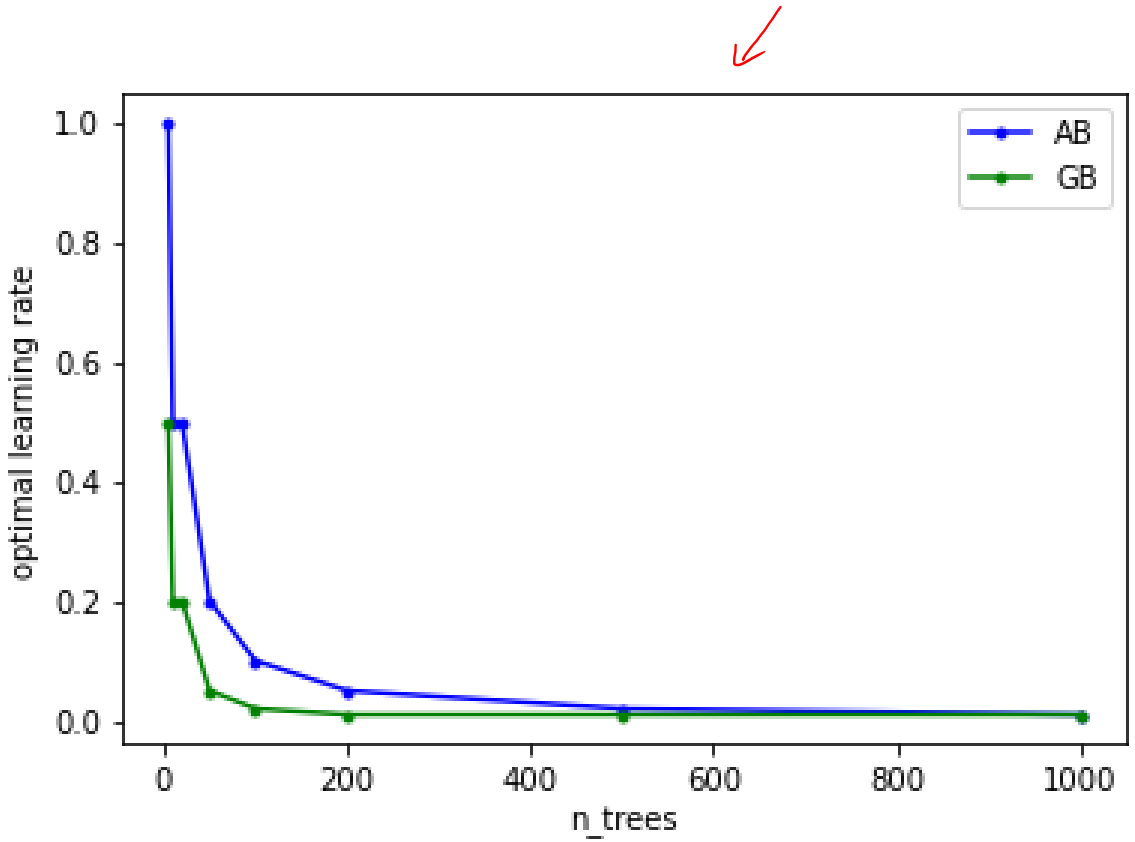
Data 2

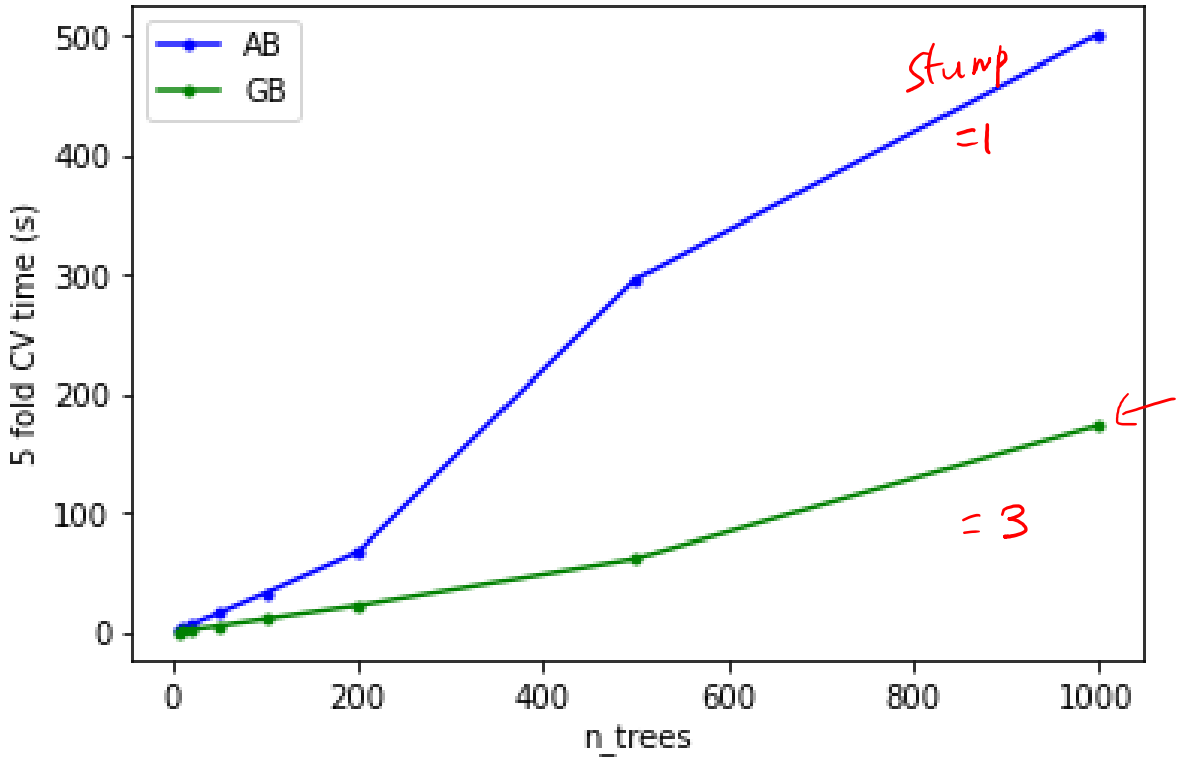




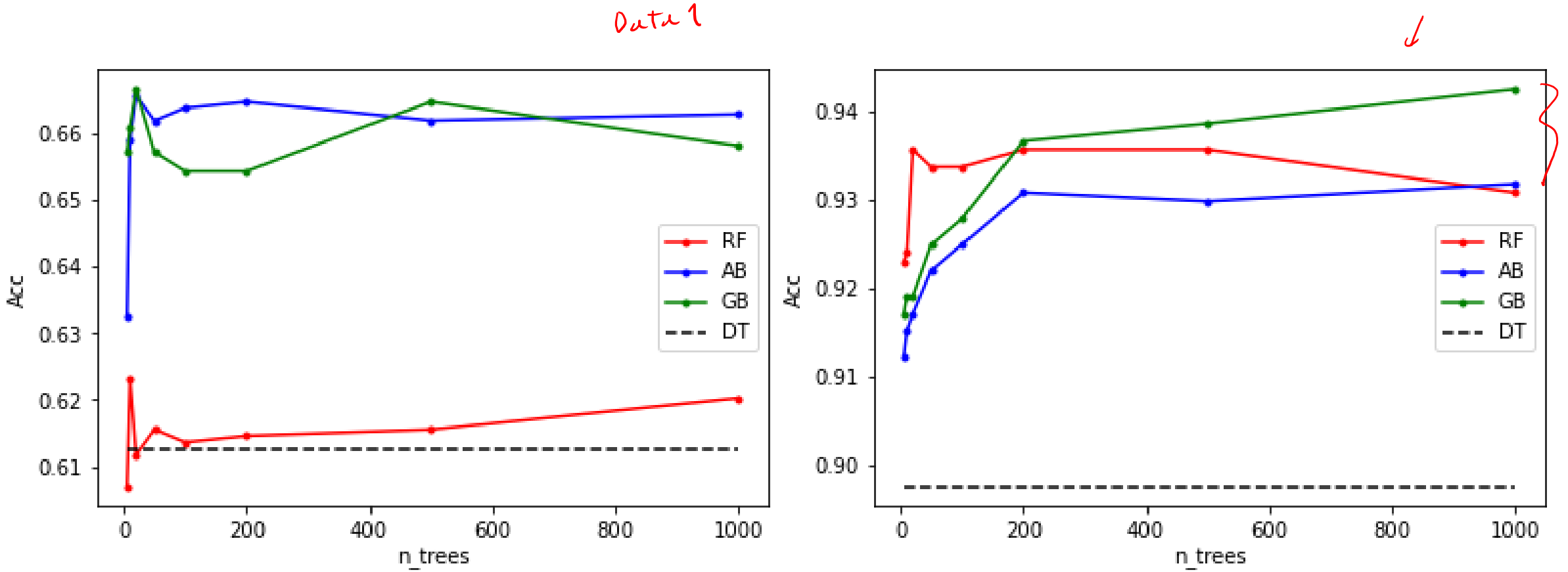
Performance Comparison /w AdaBoost

Data 1.





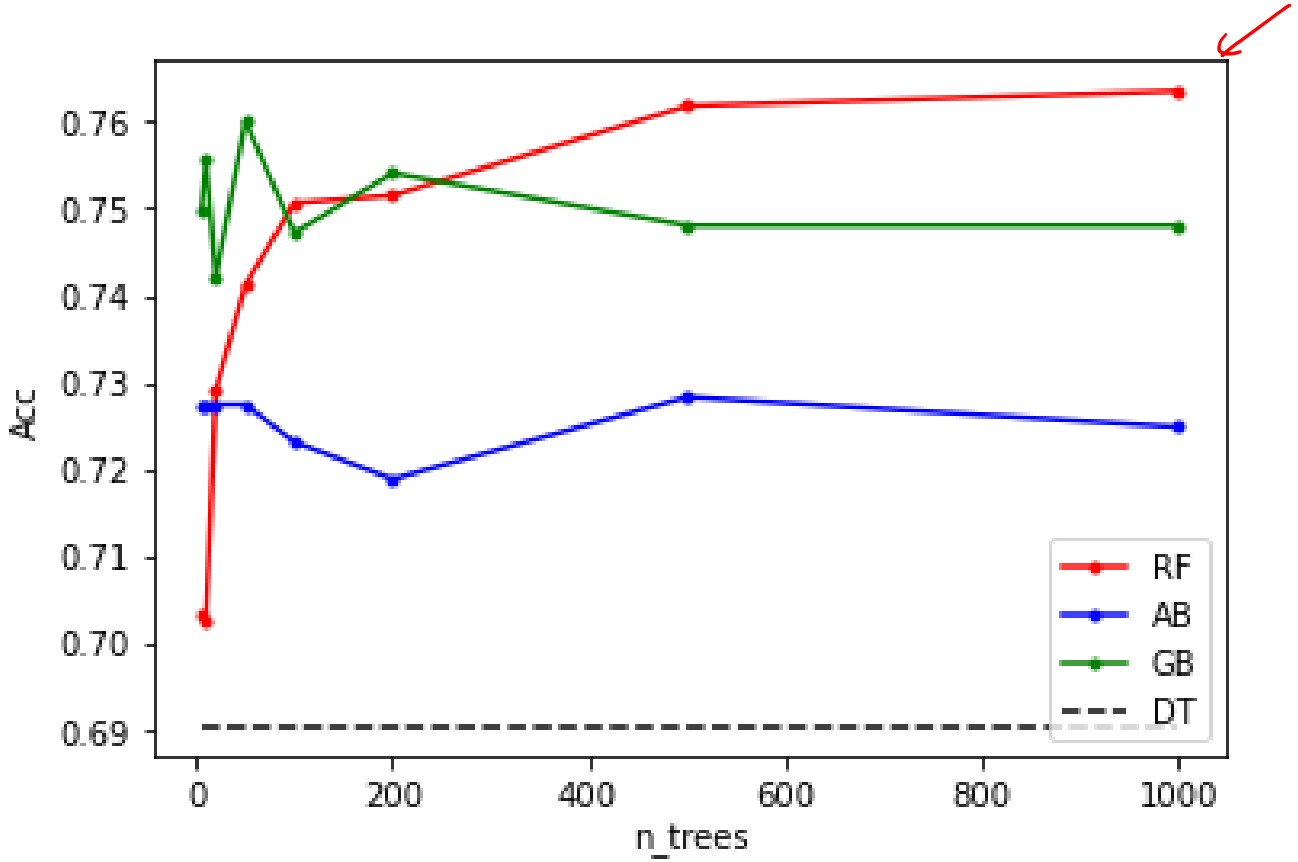
Performance Comparison /w Random Forest



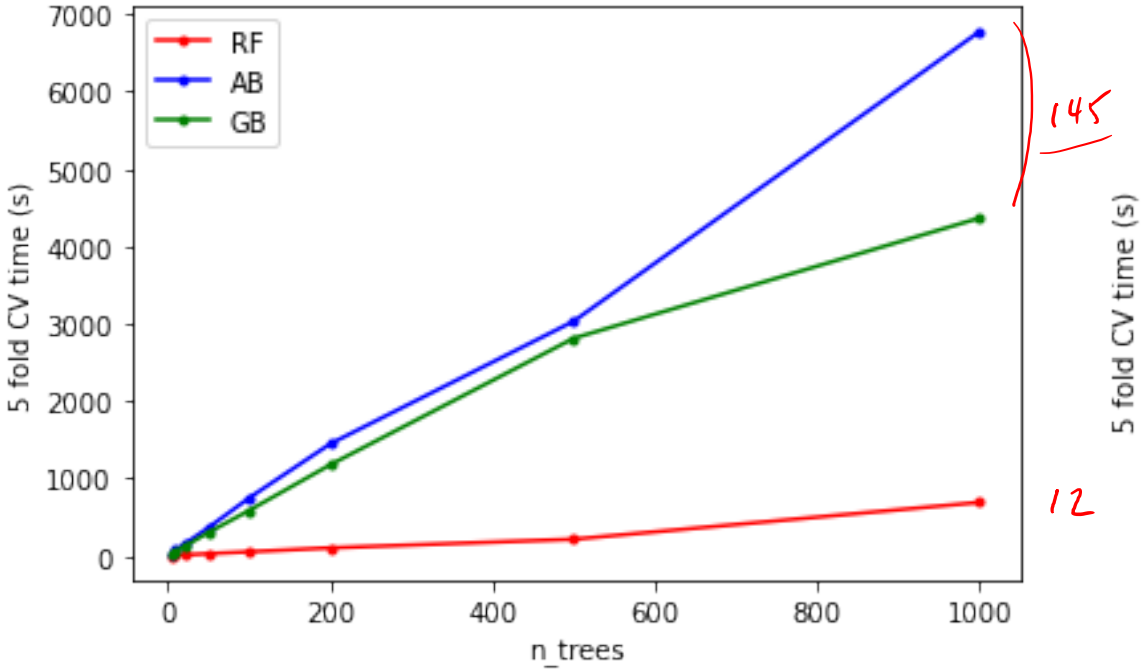
Performance Comparison /w Random Forest

Data 3.

- 145 features,
- ~3000 samples
- DT performance 0.69



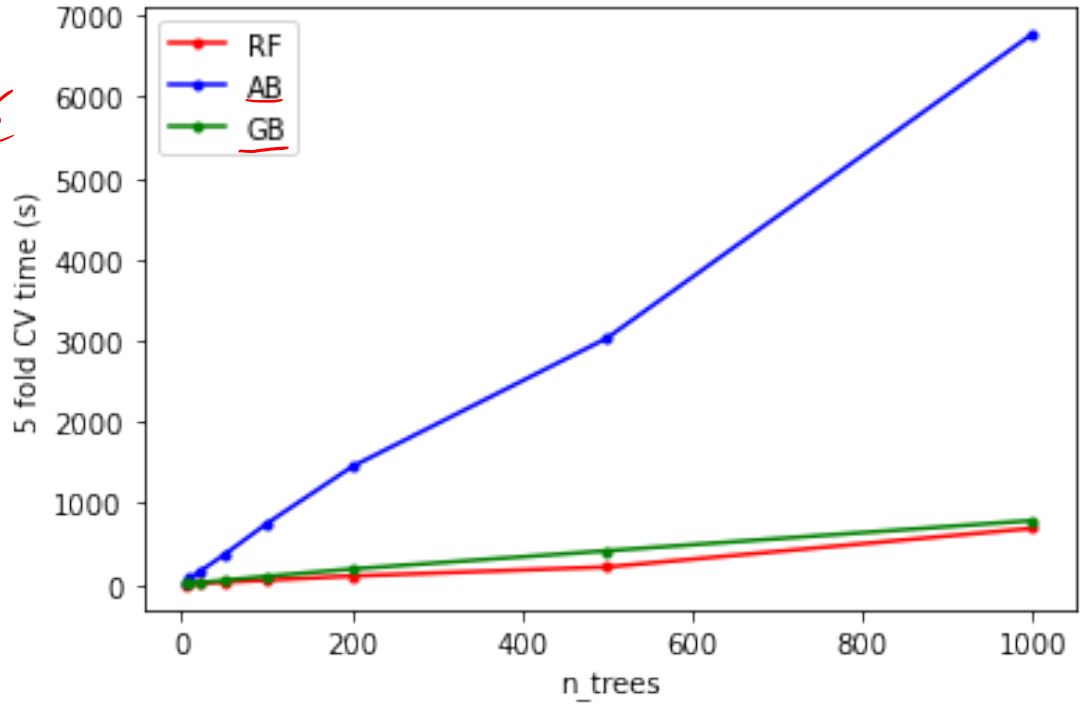
Performance Comparison /w Random Forest



145

12

max_features = 'sqrt'



Other useful packages

- XGBoost →
- lightGBM (Histogram-based)
- ExtraTree

No bagging,

Other useful packages

User guide: See the Ensemble methods section for further details.

ensemble.AdaBoostClassifier([...])	An AdaBoost classifier.
ensemble.AdaBoostRegressor([base_estimator, ...])	An AdaBoost regressor.
ensemble.BaggingClassifier([base_estimator, ...])	A Bagging classifier.
ensemble.BaggingRegressor([base_estimator, ...])	A Bagging regressor.
ensemble.ExtraTreesClassifier([...])	An extra-trees classifier.
ensemble.ExtraTreesRegressor([n_estimators, ...])	An extra-trees regressor.
ensemble.GradientBoostingClassifier(*[, ...])	Gradient Boosting for classification.
ensemble.GradientBoostingRegressor(*[, ...])	Gradient Boosting for regression.
ensemble.IsolationForest(*[, n_estimators, ...])	Isolation Forest Algorithm.
ensemble.RandomForestClassifier([...])	A random forest classifier.
ensemble.RandomForestRegressor([...])	A random forest regressor.
ensemble.RandomTreesEmbedding([...])	An ensemble of totally random trees.
ensemble.StackingClassifier(estimators[, ...])	Stack of estimators with a final classifier.
ensemble.StackingRegressor(estimators[, ...])	Stack of estimators with a final regressor.
ensemble.VotingClassifier(estimators, *[, ...])	Soft Voting/Majority Rule classifier for unfitted estimators.
ensemble.VotingRegressor(estimators, *[, ...])	Prediction voting regressor for unfitted estimators.
ensemble.HistGradientBoostingRegressor([...])	Histogram-based Gradient Boosting Regression Tree.
ensemble.HistGradientBoostingClassifier([...])	Histogram-based Gradient Boosting Classification Tree.

Recap

RF (parallel) vs. Boosting (serial)

M large

$x + x + x + x = f(x)$