



# Gradient Boosting

# Boosting algorithm

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

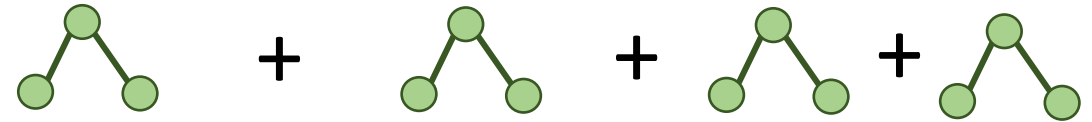
2. For  $b = 1, 2, \dots, B$ , repeat

a) Fit a tree  $f_b(x)$  to the training data  $(X, \underline{r})$

b)  $f(x) \leftarrow f(x) + \underline{\lambda f_b(x)}$

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

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



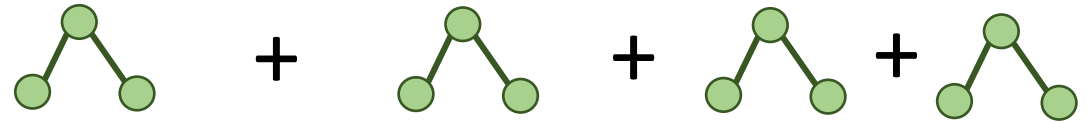
# Gradient Boosting

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

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

2. For  $b = 1, 2, \dots, B$ , 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  $\underline{f_0(x_i)} = \operatorname{argmin}_{\theta_0} \sum_{i=1}^N L(x_i, y_i; \theta_0)$

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

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

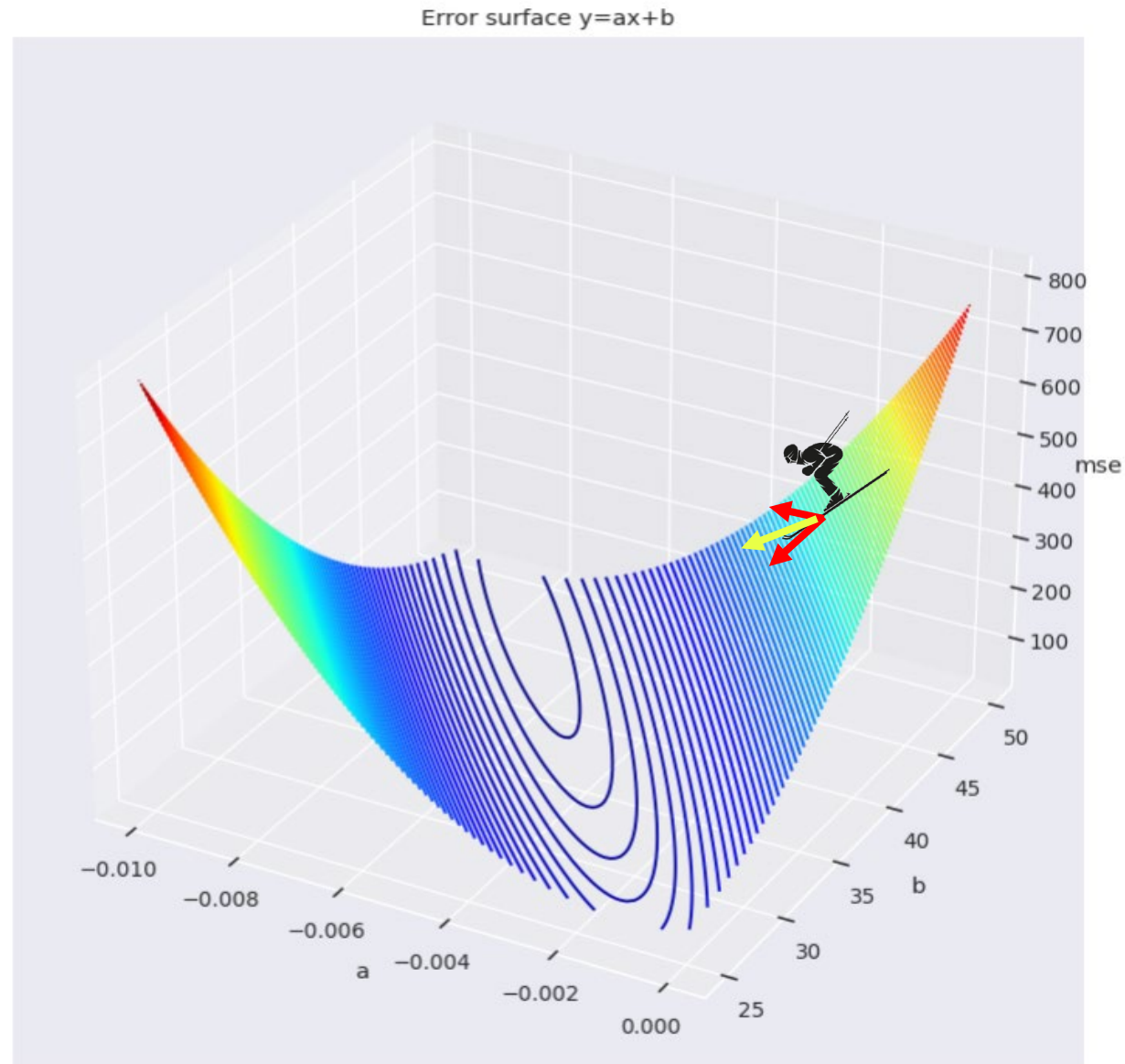
b) Fit a tree  $f_b(x_i)$  to the training data  $(x_i, r_{ib}) \rightarrow \underline{\theta_b}$

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

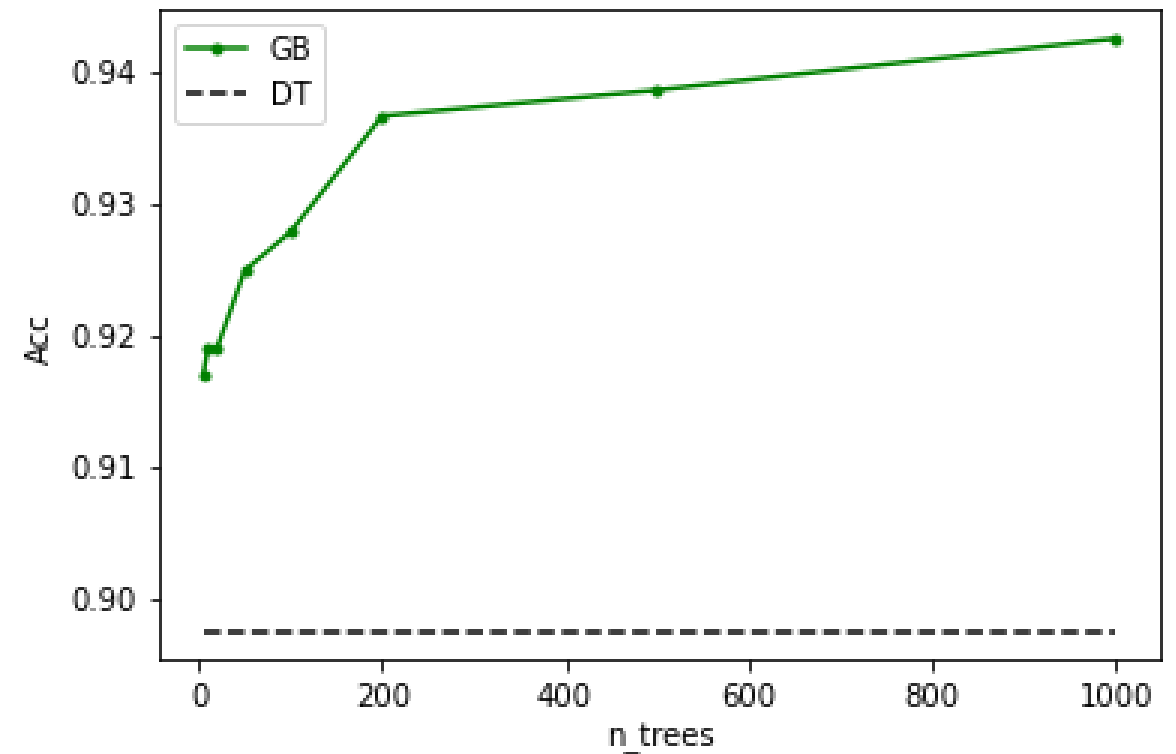
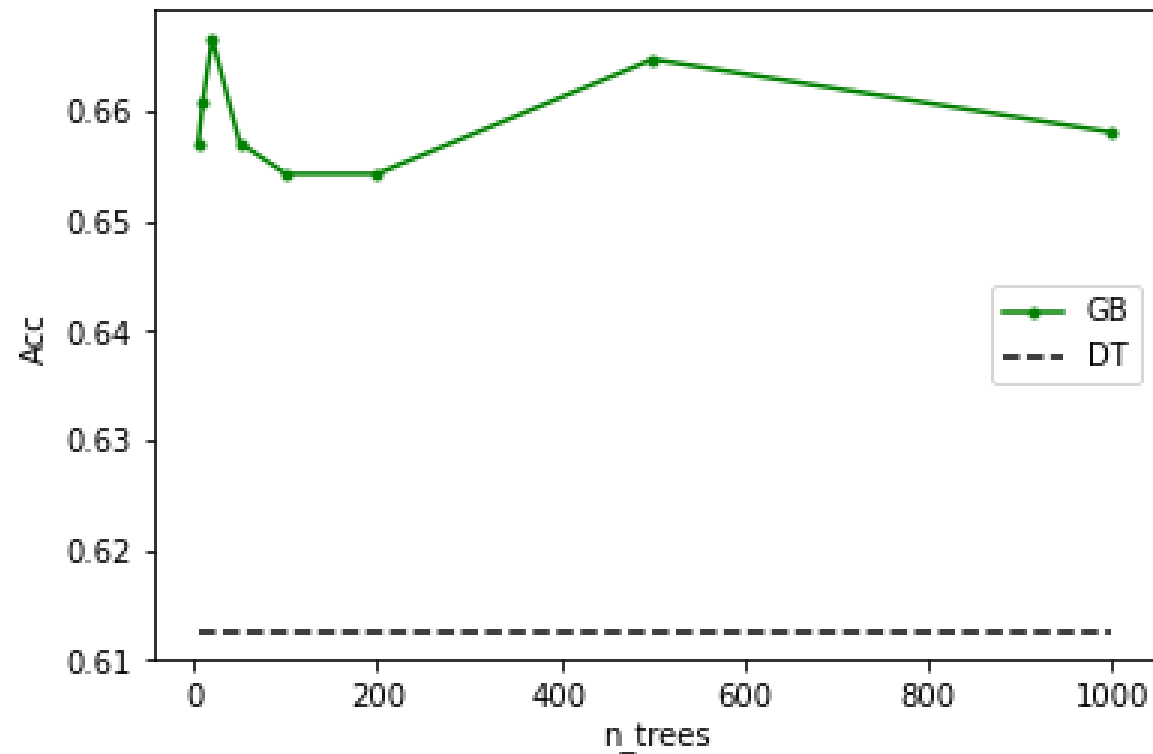
## Data 1.

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

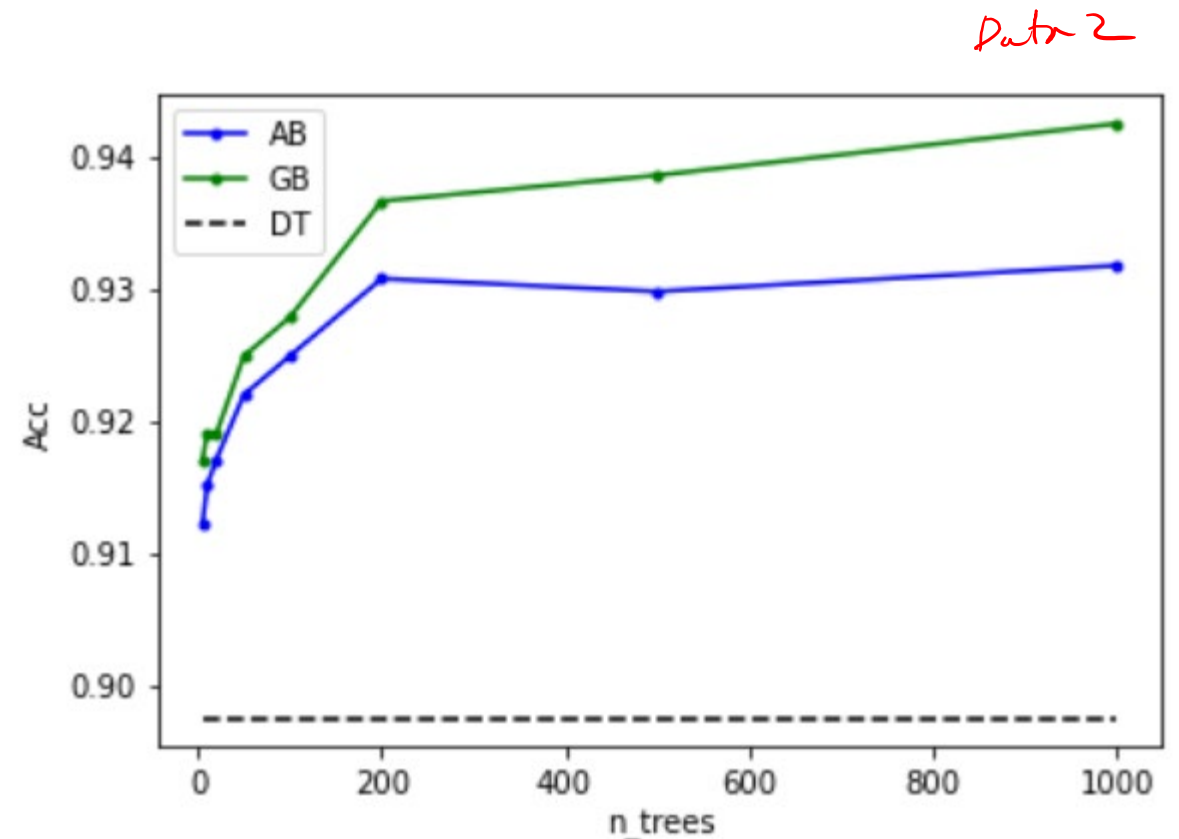
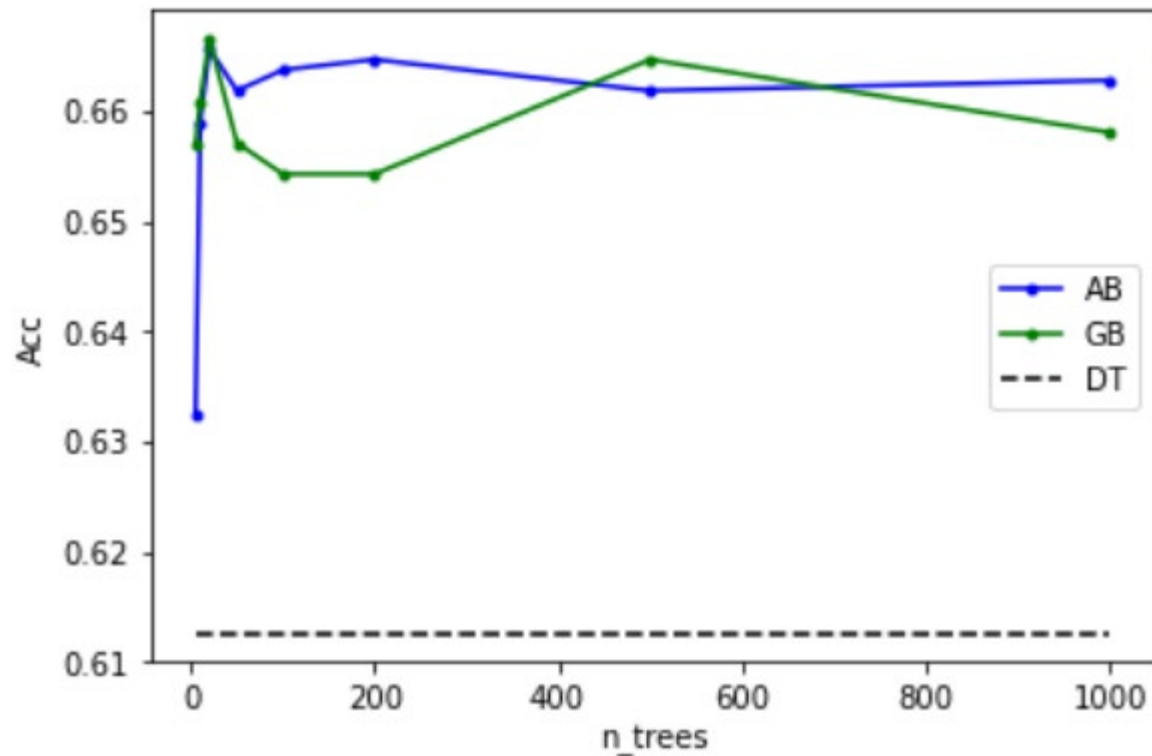
## Data 2.

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

# Performance Comparison



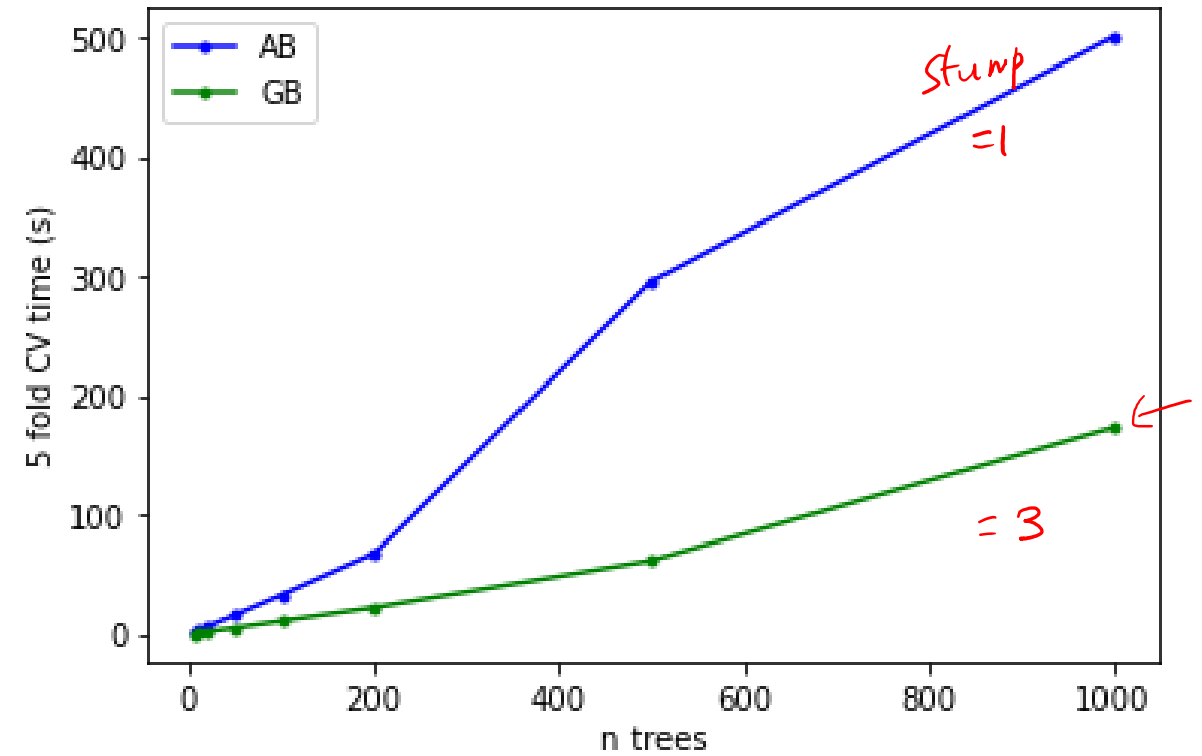
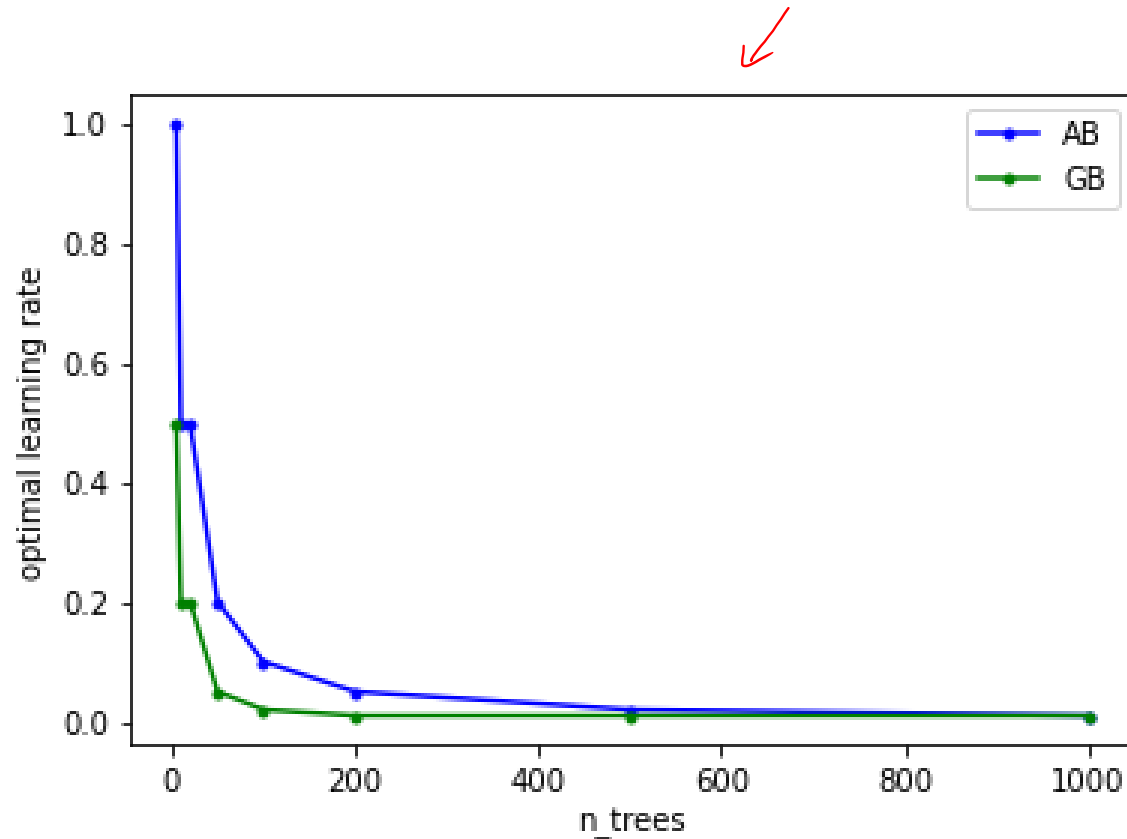
# Performance Comparison /w AdaBoost





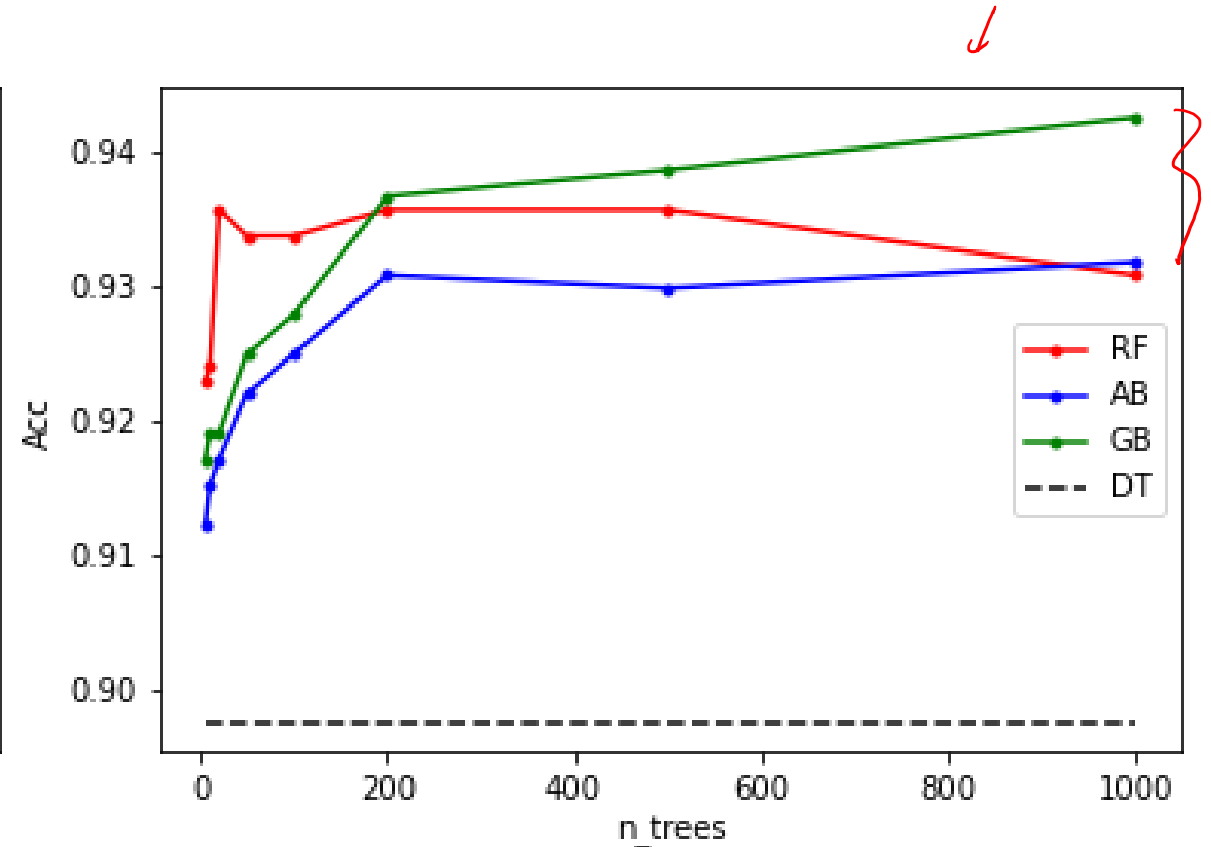
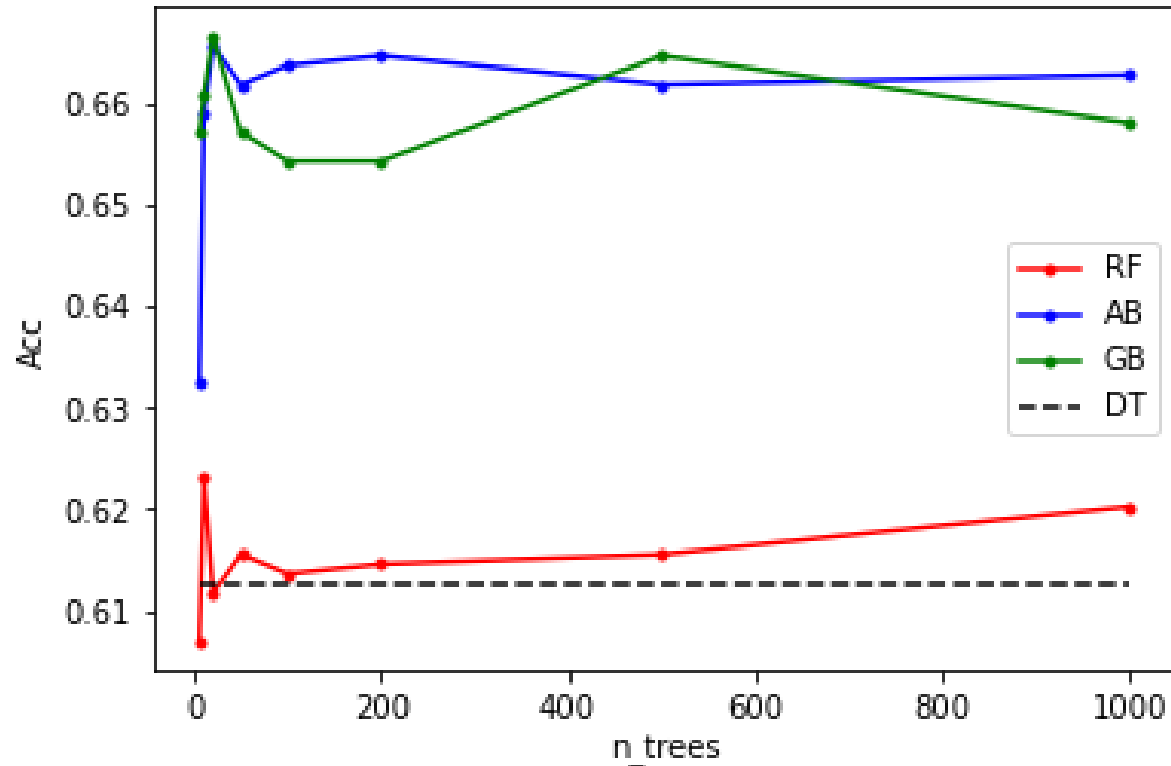
# Performance Comparison /w AdaBoost

Data 1.



# Performance Comparison /w Random Forest

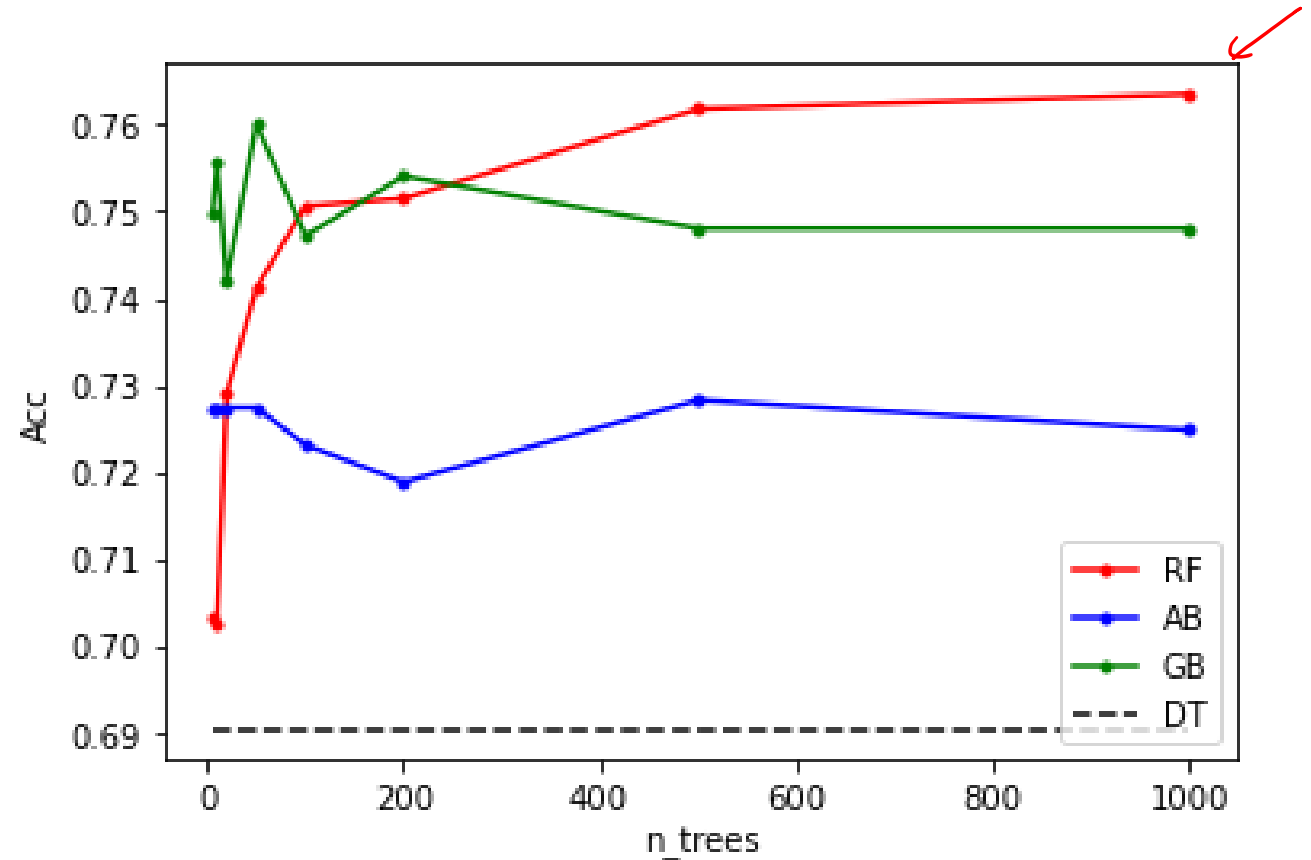
Data 1



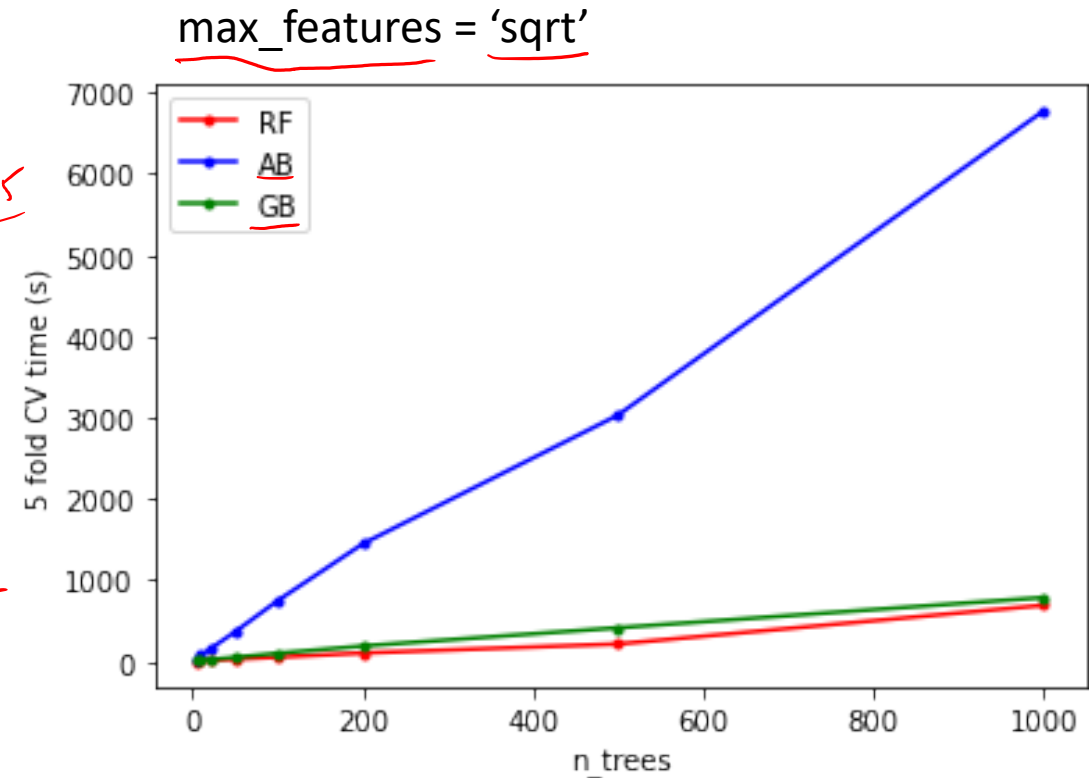
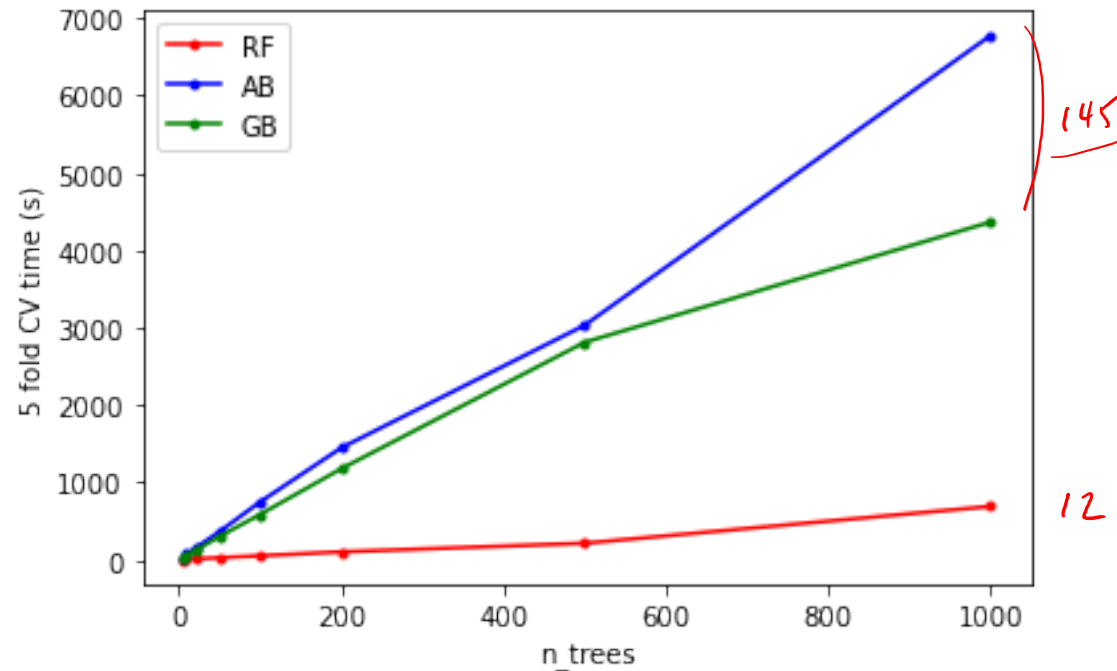
# Performance Comparison /w Random Forest

Data 3.

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



# Performance Comparison /w Random Forest



# Other useful packages

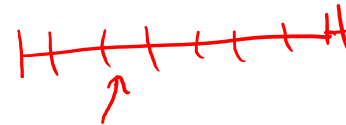
- XGBoost →

- lightGBM (Histogram-based)



- ExtraTree

No bagging ,



# Other useful packages

**User guide:** See the [Ensemble methods](#) section for further details.

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



# Recap

RF (parallel)

$$\frac{f_1 + f_2 + \dots + f_M}{M}$$

M large

vs.

Boosting (serial)

$$f_1 + f_2 + f_3 + \dots = f(x)$$