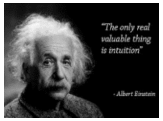
Boosting Models



Boosting Models: Intuition

- Like Bootstrap, **Boosting** is a statistical technique that can be used for **various** types of **models**
- With **Bagging**, each tree is generated from **random samples** generated from the data with replacement
- **Boosting** works similarly, except that the **trees** are **grown** **"sequentially"** with each tree generated using **information** from the **prior tree**
- Boosting does **not** use **bootstrap**; each tree is **fit** on a **modified version** of the previously tree
- **Bagging** tries trees **randomly**, whereas **Boosting learns slowly** by using the **residuals** from the prior model, rather than the actual outcome values
- Think of it as placing more emphasis on the part of the data that remains unexplained (i.e., the residuals), thus improving the fit

Boosting Models: Details

- Boosting is done by generating multiple trees and allowing the **model** to **learn** **"slowly"** from the **prior** model
- The **Boosting** estimation **algorithm** is explained in the textbook, and is beyond the scope of this class, but it **works like this**:
  1. We estimate the **first tree** and **"shrink"** it by a tuning parameter λ which controls the **speed of learning** → the **smaller** the λ the **less** we **learn** from the prior model (because we are shrinking it)
  2. We then **compute** the **residual errors** from this model
  3. We then **fit another tree**, but this time we **predict** the **residual errors** (i.e., the unexplained part of the model), **shrink** it by λ and **add** this prediction to the **prior model**.
  4. Go **back** to 2 and **repeat** until the desired number of trees has been generated
- In other words, the tree is **grown slowly** (controlled by λ) in each **iteration** by **adding** the subsequent (shrunken) trees, **rather** than by **averaging**.

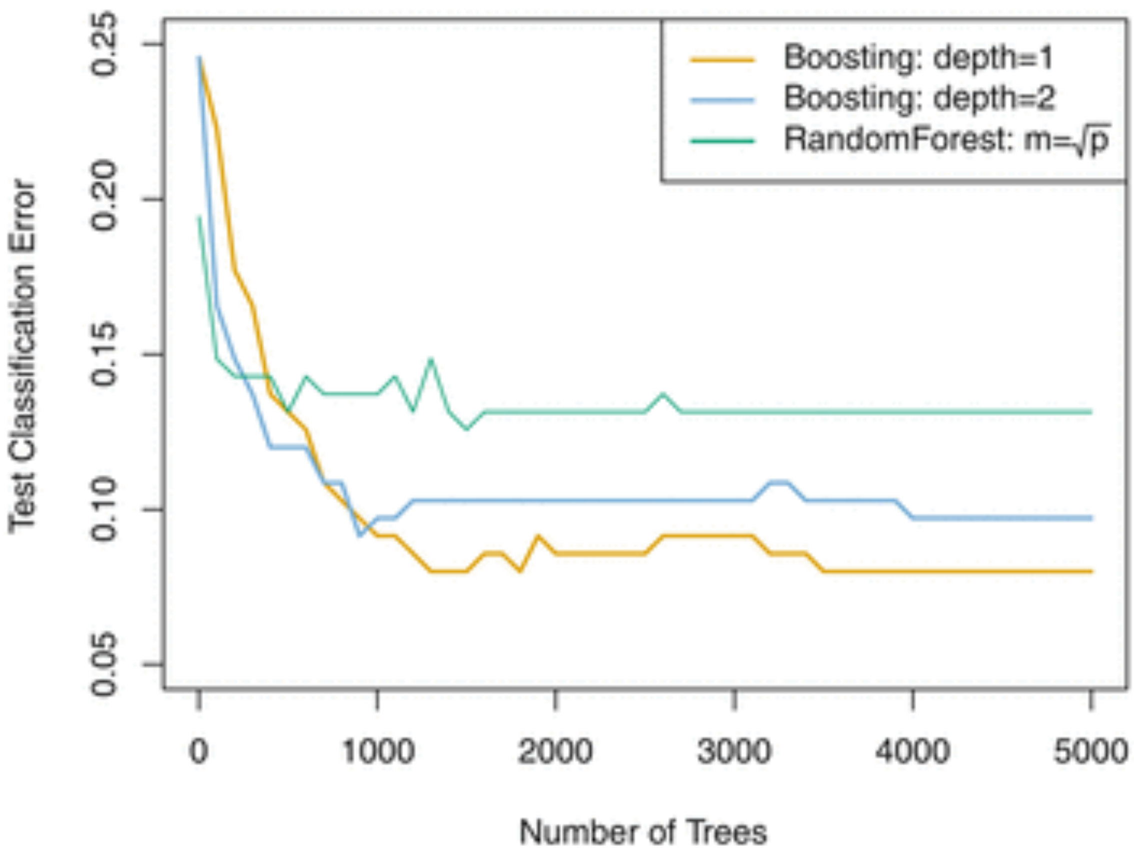
Boosting Models: Tuning

- Boosting has **3 tuning parameters**:
  1. The number of **tree iterations B** – the number of trees (unlike Boosting) improve training fit but also increase over-fitting
  2. The **shrinkage** parameter λ – **Small** values are better → models that **learn slowly** tend to perform **better**, but this depends on the selected value of B for the number of trees ($\lambda = 0.01$ to 0.001 are typical)
  3. **Tree depth** – i.e., number of **tree splits** (typically small, from 1 to 6) – since predictors change in each iteration, there is no need to have deep trees with too many splits.

Boosting Illustration: Tuning

Observations

1. The boosted models were generated with learning **shrinkage** $\lambda = 0.01$
2. In this example **Boosting outperformed Random** Forest
3. A **depth** of **1** (split) **outperformed** a depth of **2** (splits)



R Tips

`gbm(){gbm}` → Function in the `{gbm}` "Generalized Boosting Package" used to Boosting models

```
boost.fit=gbm(y~x1+x2+etc.,data=dataName,
              distribution="gaussian",
              n.trees=5000,
              interaction.depth=4)
```
→ Use `distribution="gaussian"` for regression models and `distribution="bernoulli"` for classification models; fits a Boosting model using 5000 trees each of depth 4.

`summary(boost.fit)` → Provides various statistics and plot

Kogod School
of
Business