Lecture inf620 2025 - Boost

Depto de Informática - UFV

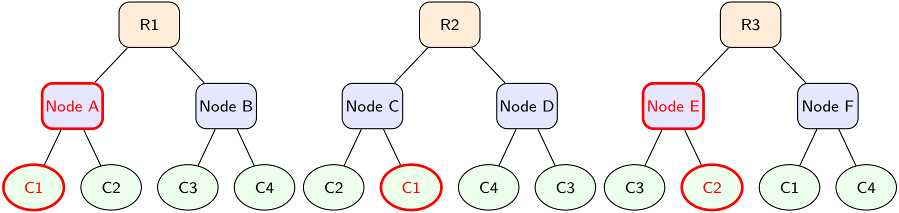
Ricardo Ferreira
ricardo@ufv.br

2024

Introduction

- Class Material (click here for the Colab)
- **Review**: Supervised Learning with Decision Trees
- **Problems**: Classification and Regression
- TODAY's class: Ensemble Techniques
  - Boost: Adaboost, Gradient Boost
  - XGBoost

Review: Example of a Random Forest (3 Trees)



Gradient Boosting (GBM)

- **Basic Principle**
  - Sequential construction of trees
  - Focus on previous errors
  - Gradient descent
- **Characteristics**
  - Shallow trees (3–5 levels)
  - Gradual learning
  - High predictive power

Modern Implementations

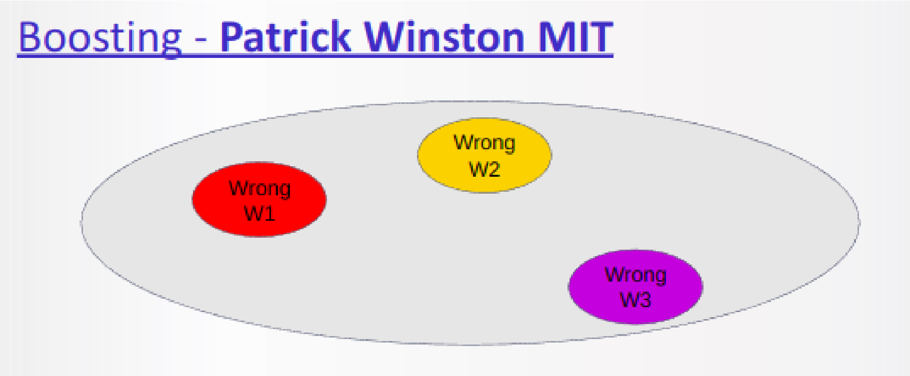
XGBoost

- Regularization
- Optimized for sparse data
- Efficient parallelization

LightGBM

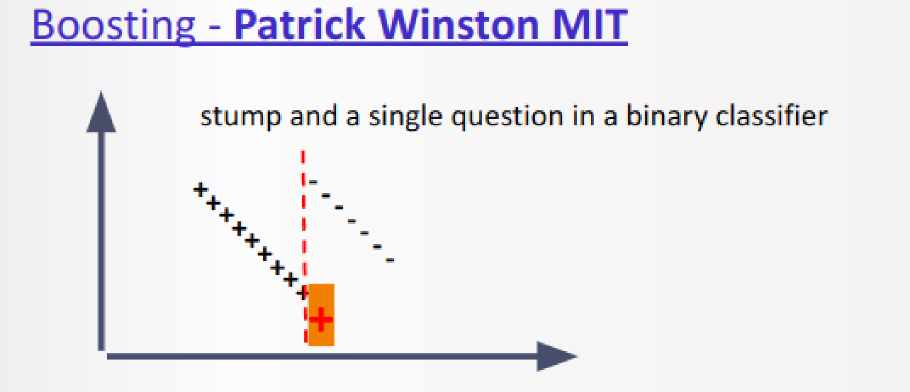
- Leaf-wise growth
- Native categorical support
- Lower memory usage

Weak and Strong



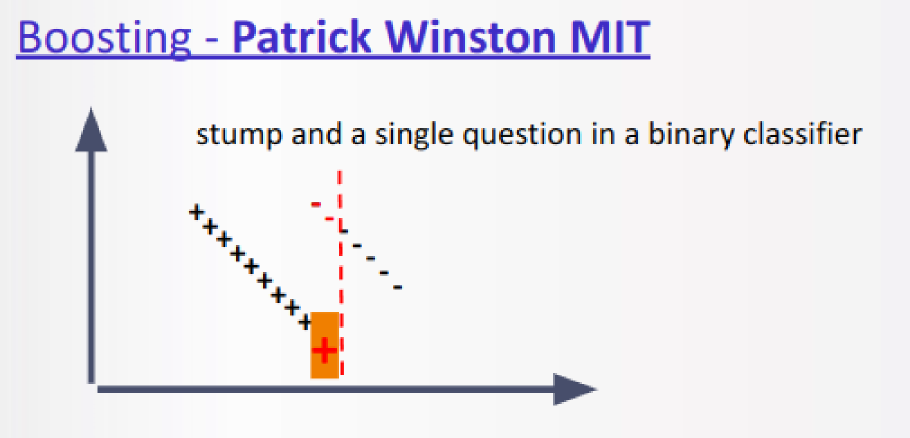
Click here MIT Lecture

Weak and Strong



Click here MIT Lecture

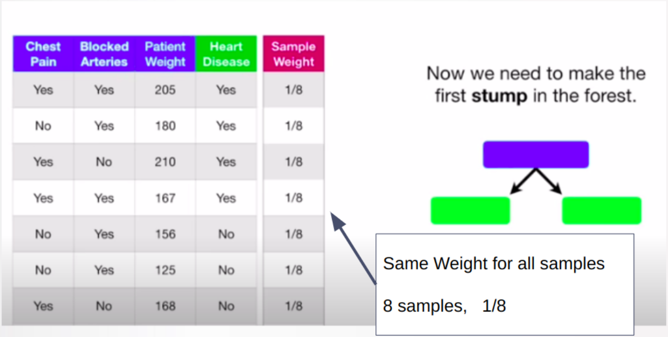
Weak and Strong



Click here MIT Lecture

Ada Boost

AdaBoost (Statquest) - Create First Stump



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8

Click here Stat Quest Lecture
Click here for Wiki
Paper: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting

Ada Boost

AdaBoost (Statquest) - Select smaller Gini

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8

Click here Stat Quest Lecture

Ada Boost

AdaBoost (Statquest) - Compute the error

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8

The **Total Error** for a stump is the sum of the weights associated with the *incorrectly* classified samples.

Thus, in this case, the **Total Error** is **1/8**.

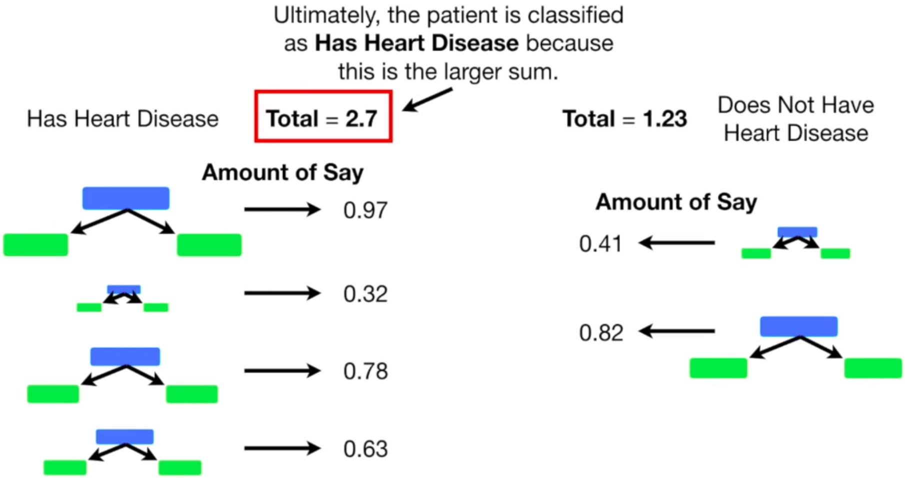
Click here Stat Quest Lecture

Ada Boost

AdaBoost (Statquest) - new weights

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Weight	Norm. Weight
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

Ada Boost



Click here Stat Quest Lecture

Gradient Boost

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for

(B) Fit a regression tree to the r_{im} values and (
regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_m I(x \in R_{jm})$

Step 3: Output $F_M(x)$

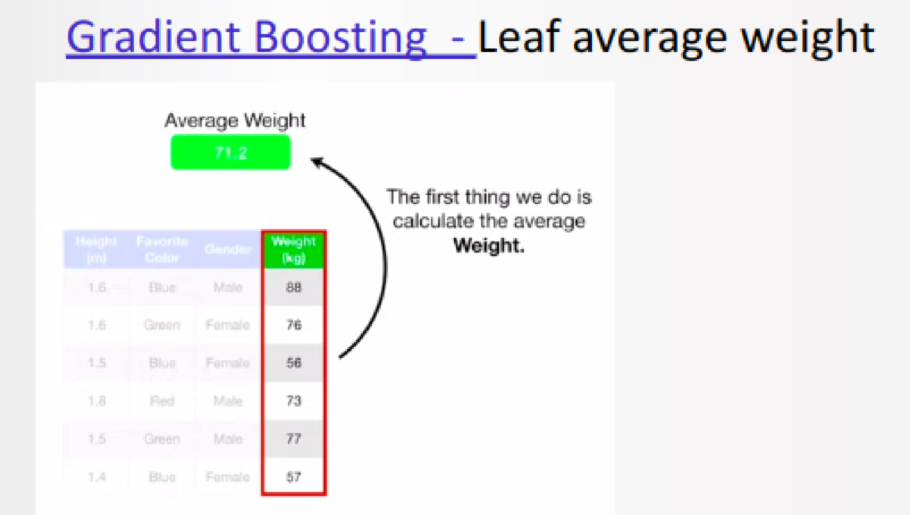
NOTE: The **Gradient Boost** algorithm looks complicated because it was designed to be configured in a wide variety of ways...

Click here Stat Quest Lecture

paper: Greedy function approximation: a gradient boosting machine

Wiki

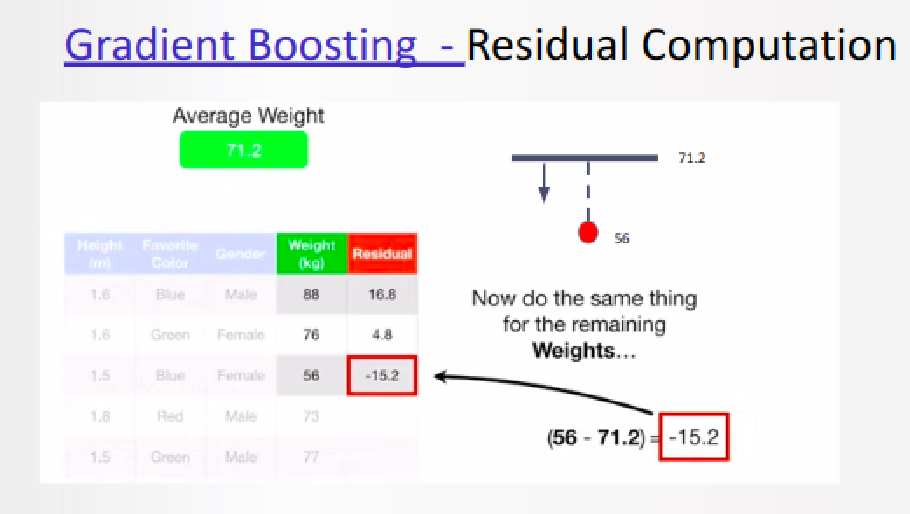
Gradient Boost



Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Click here Stat Quest Lecture

Gradient Boost

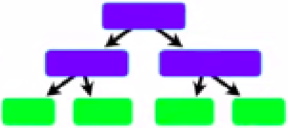


Click here Stat Quest Lecture

Gradient Boost

Gradient Boosting - Build Tree to predict residual !

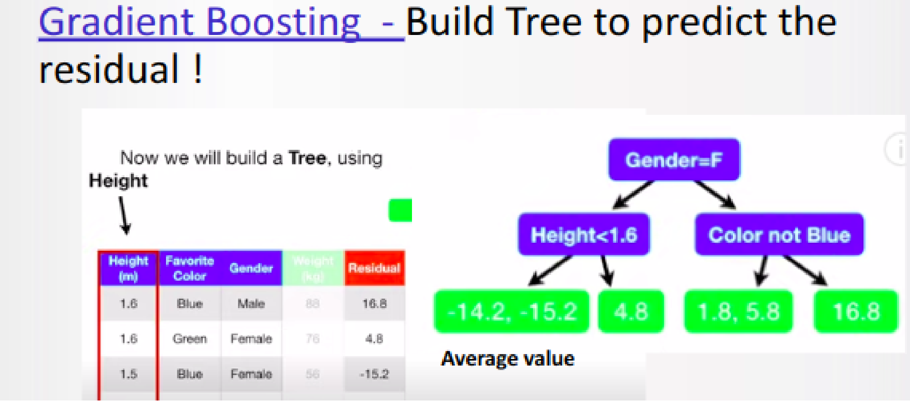
Now we will build a **Tree**, using **Height**



Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8

Click here Stat Quest Lecture

Gradient Boost

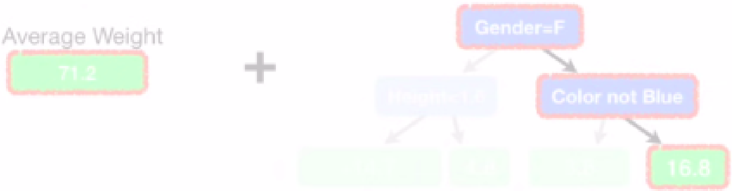


Click here Stat Quest Lecture

Gradient Boost

Click here Stat Quest Lecture

Gradient Boost



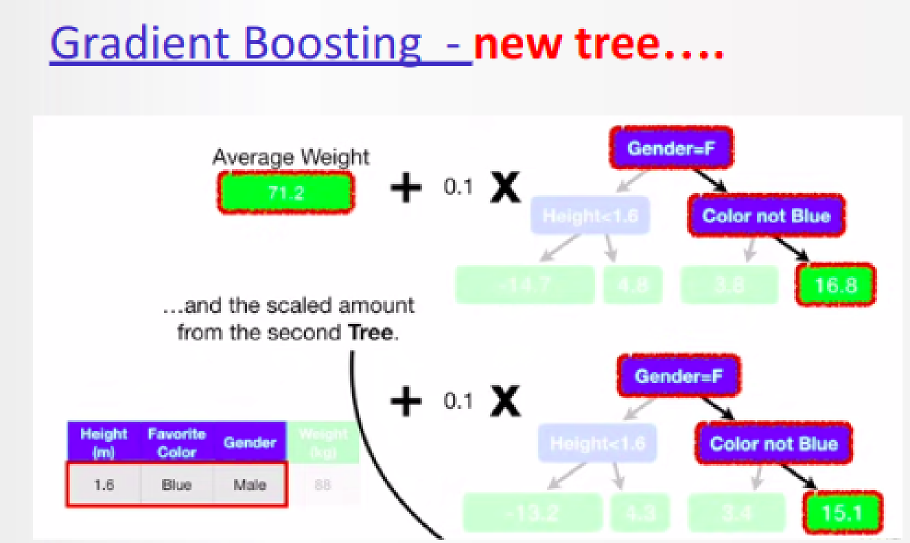
Predicted Weight = 71.2 + 16.8 = 88

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

In other words, we have low **Bias**, but probably very high **Variance**.

Click here Stat Quest Lecture

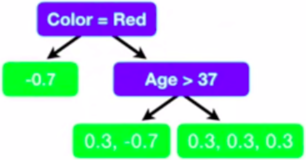
Gradient Boost



Click here Stat Quest Lecture

Gradient Boost Classification

Gradient Boosting Classification - New Tree

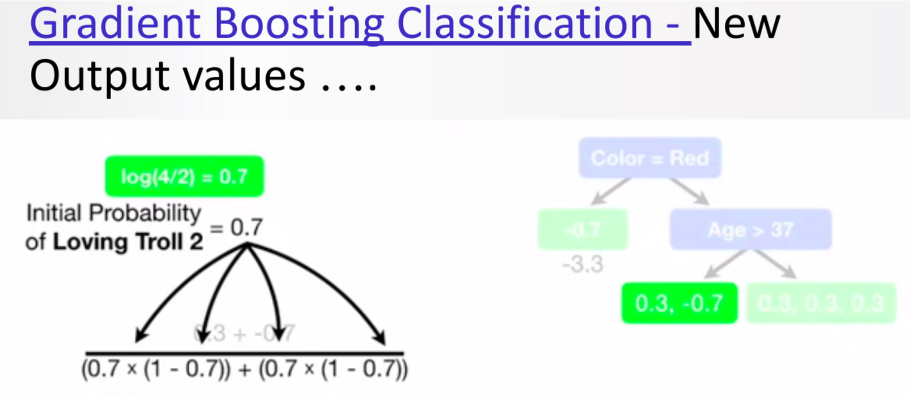


Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3

NOTE: Just like when we used **Gradient Boost for Regression**, we are limiting the number of leaves that we will allow in the tree.

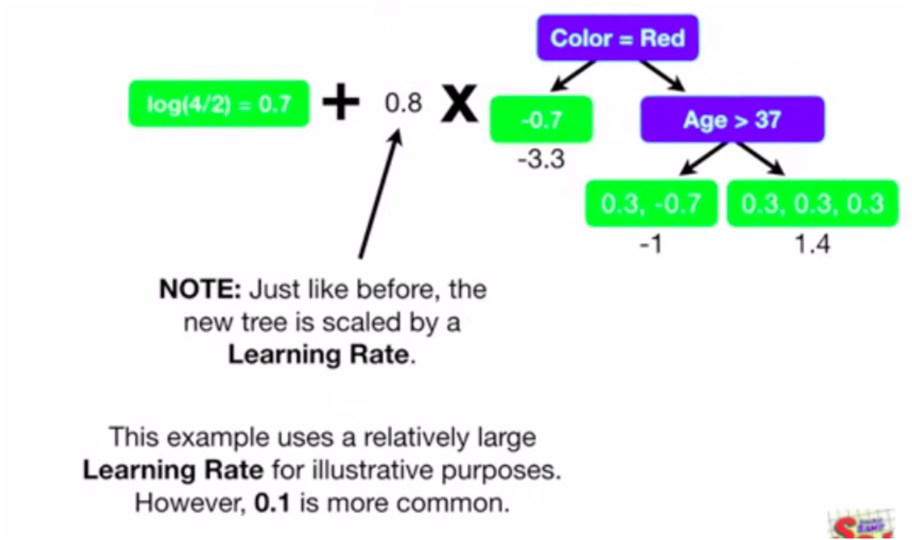
Click here Stat Quest Lecture

Gradient Boost Classification



Click here Stat Quest Lecture

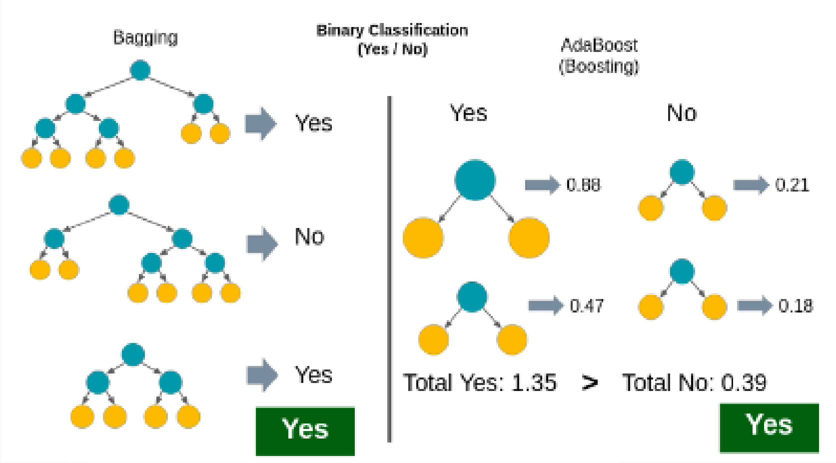
Gradient Boost Classification



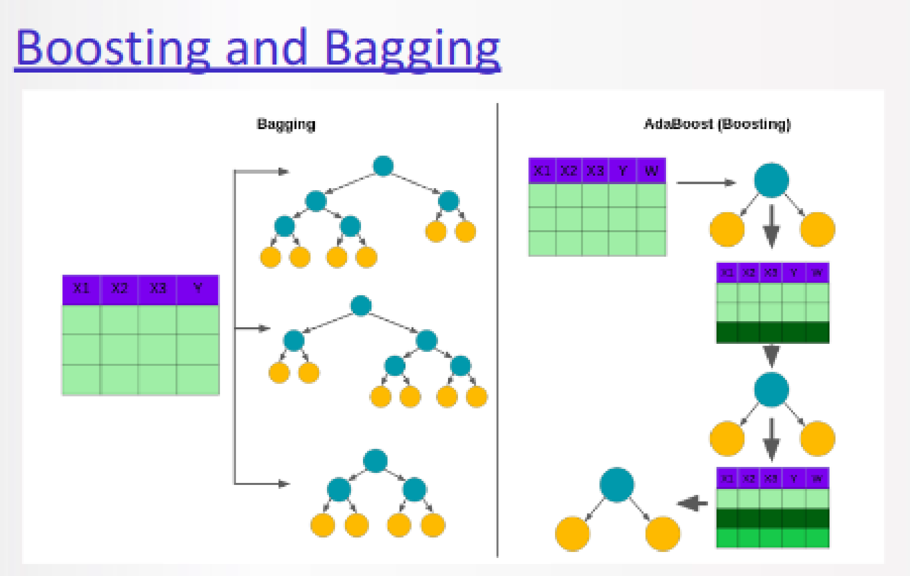
Click here Stat Quest Lecture

Bag or Boost

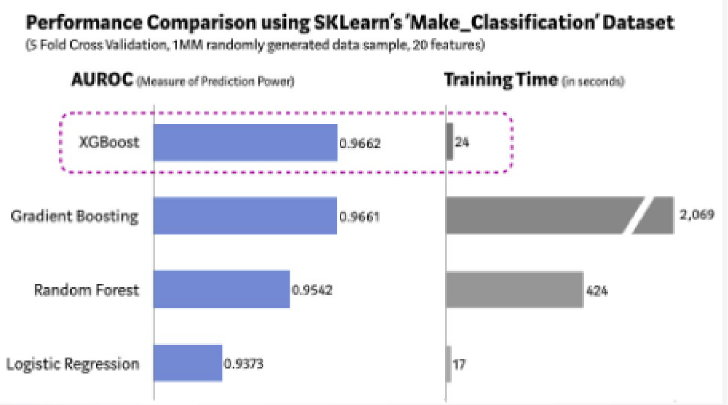
Boosting and Bagging



Bag or Boost


Boosting and Bagging
Bagging
X1
X2
X3
Y
AdaBoost (Boosting)
X1
X2
X3
Y
W

XGBoost



Paper

XGBoost: A scalable tree boosting system

Wiki

What is XGBoost?

- **XGBoost** = eXtreme Gradient Boosting
- Efficient and scalable implementation of gradient boosted trees
- Designed for speed and performance
- **Core Idea:** Add trees to minimize loss, one at a time
- Handles missing values and categorical features natively
- Widely used in Kaggle competitions and industry

Key Features of XGBoost

- **Regularization**: Prevents overfitting (L1 and L2)
- **Parallelization**: Tree construction is parallelized
- **Tree Pruning**: Uses a max-depth parameter (not max leaf nodes)
- **Early Stopping**: Stops training when no improvement
- **Cross-validation**: Built-in support
- **Sparsity-aware**: Efficient handling of sparse data

What is CatBoost?

- **CatBoost** = Categorical Boosting
- Gradient boosting algorithm developed by Yandex
- Released in 2017 "CatBoost: unbiased boosting with categorical features"
- **Core Idea:** Superior handling of categorical features with minimal preprocessing
- Implements Ordered Boosting to fight prediction shift
- Uses oblivious decision trees for better generalization
- Built-in GPU acceleration for faster training
- click here - How create Cat Trees

CatBoost: Advantages & Limitations

Advantages

- Handles categorical features automatically
- Robust against overfitting
- Often performs well out-of-the-box
- Minimal hyperparameter tuning needed
- Built-in feature importance

Limitations

- Can be slower than alternatives on large datasets
- More memory-intensive than some competitors
- Fewer configuration options than XGBoost
- Less established ecosystem
- Limited interpretability tools

What is LightGBM?

- **LightGBM** = Light Gradient Boosting Machine
- Developed by Microsoft
- Released in 2017 "LightGBM: A Highly Efficient Gradient Boosting Decision Tree"
- **Core Idea:** Gradient-based One-Side Sampling (GOSS) to focus on informative samples
- Exclusive Feature Bundling (EFB) to handle high-dimensional sparse features
- Leaf-wise tree growth strategy instead of level-wise
- Extremely fast training and low memory usage

LightGBM: Advantages & Limitations

Advantages

- Extremely fast training speed
- Low memory consumption
- Better accuracy than many alternatives
- Handles large datasets efficiently
- Native categorical feature support
- Parallel, distributed, and GPU learning

Limitations

- Can overfit on small datasets
- Leaf-wise growth needs careful max_depth limiting
- More hyperparameters to tune than CatBoost
- Less robust with default parameters
- May require more preprocessing for optimal results

AdaBoost

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

data = load_iris() # Carregar o dataset
X = data.data
y = data.target

# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

ada = AdaBoostClassifier(n_estimators=100, random_state=42) # Cri
ada.fit(X_train, y_train) # Treinar
y_pred = ada.predict(X_test) # Prever no conjunto de teste

accuracy = accuracy_score(y_test, y_pred) # Calcular a acurácia
```

Gradient Boost

```python
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

data = load_iris() # Carregar o dataset
X = data.data
y = data.target

# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

gb = GradientBoostingClassifier(n_estimators=100, random_state=42
gb.fit(X_train, y_train) # Treinar
y_pred = gb.predict(X_test) # Prever no conjunto de teste

accuracy = accuracy_score(y_test, y_pred) # Calcular a acurácia
```

XGBoost

```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

data = load_iris() # Carregar o dataset
X = data.data
y = data.target

# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

xgb = XGBClassifier(n_estimators=100, random_state=42) # Criar o
xgb.fit(X_train, y_train) # Treinar
y_pred = xgb.predict(X_test) # Prever no conjunto de teste

accuracy = accuracy_score(y_test, y_pred) # Calcular a acurácia
```