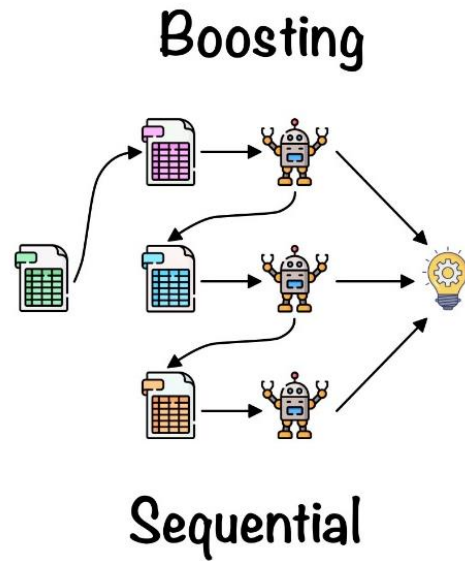


# Class -22

## Boosting algorithms (AdaBoost, GBM, XGBoost)

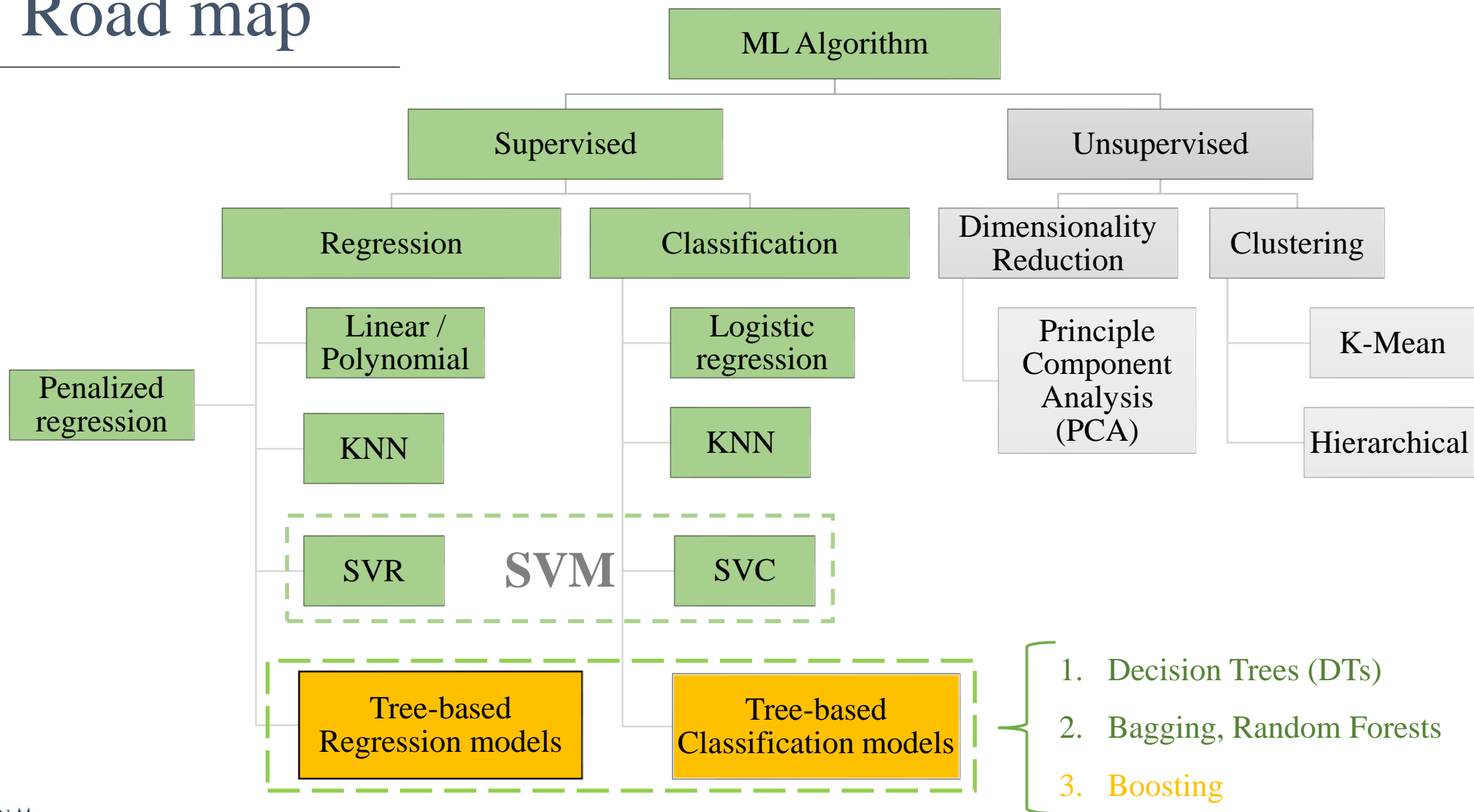
---

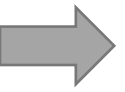
Prof. Pedram Jahangiry





# Road map





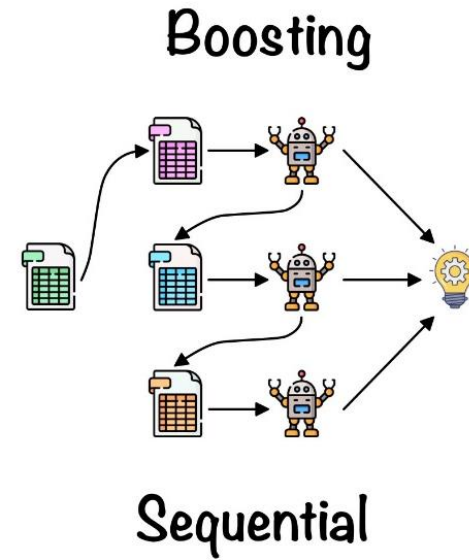
# Topics

## **Part I**

1. Bagging vs Boosting
2. AdaBoost
3. Gradient Boosting Machine (GBM)
4. XGBoost

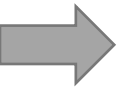
## **Part II**

Pros and Cons



# Part I

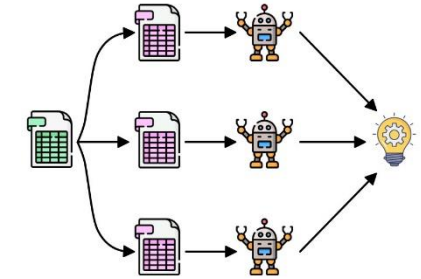
1. Bagging vs Boosting
2. AdaBoost
3. Gradient Boosting Machine (GBM)
4. XGBoost



# Bagging vs Boosting

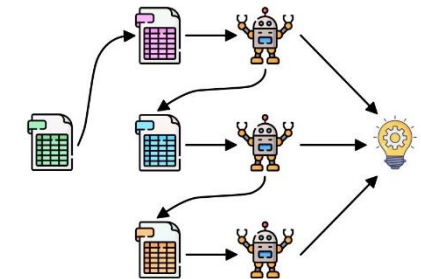
- **Bagging** consists of creating many “copies” of the training data (each copy is slightly different from another) and then apply the weak learner to each copy to obtain multiple weak models and then combine them.
- In bagging, the bootstrapped trees are **independent** from each other.
- **Boosting** consists of using the “original” training data and **iteratively** creating multiple models by using a weak learner. Each new model would be different from the previous ones in the sense that the weak learner, by building each new model tries to “fix” the **errors** which previous models make.
- In boosting, each tree is grown using information from **previous** tree.

## Bagging



## Parallel

## Boosting



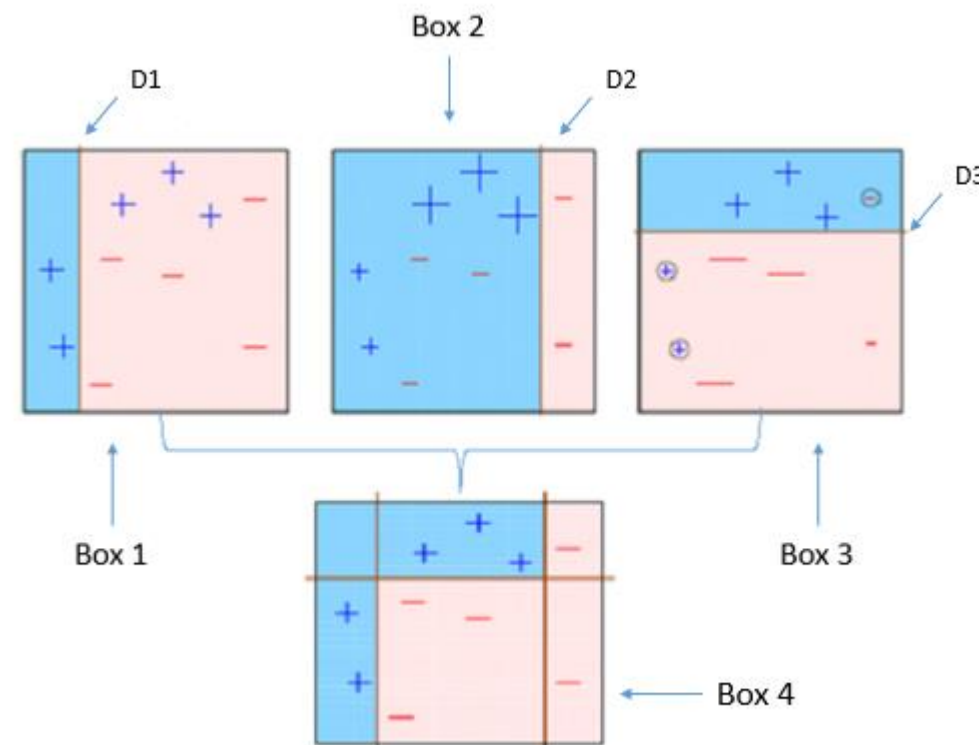
## Sequential



# AdaBoost (Adaptive Boosting)

- Forest of weak learners (trees with only 1 feature; **stumps**).
- Each tree (stump) depends on the previous tree's **errors** rather than being independent.

- 1) Starting with usual splitting criteria!
- 2) Each tree (stump) gets **different weight** based on its prediction accuracy.
- 3) Each **observation** gets a weight inversely related to its predicted outcome. (ex, misclassified ones get more weight).
- 4) **Aggregation** is done based on each weak learner's weight.



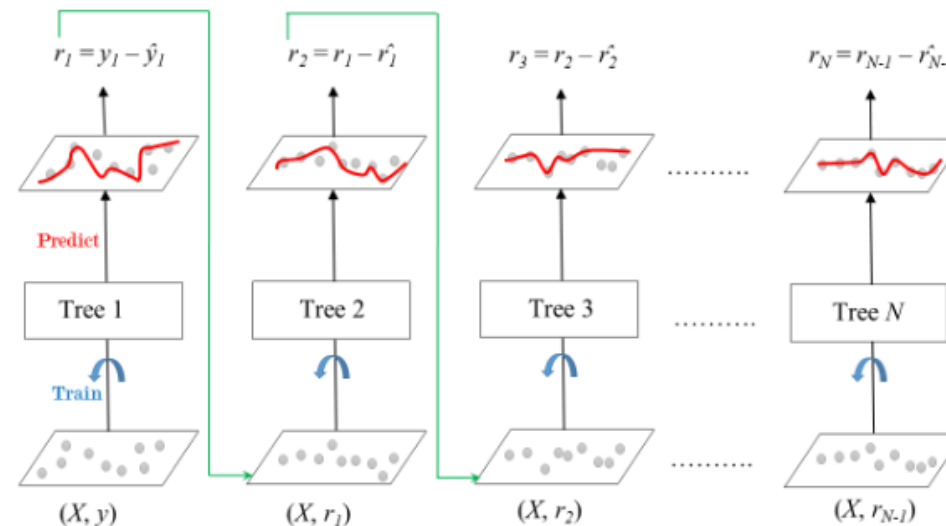
Source: [Towards data science](#)



# Gradient Boosting Machine (GBM)

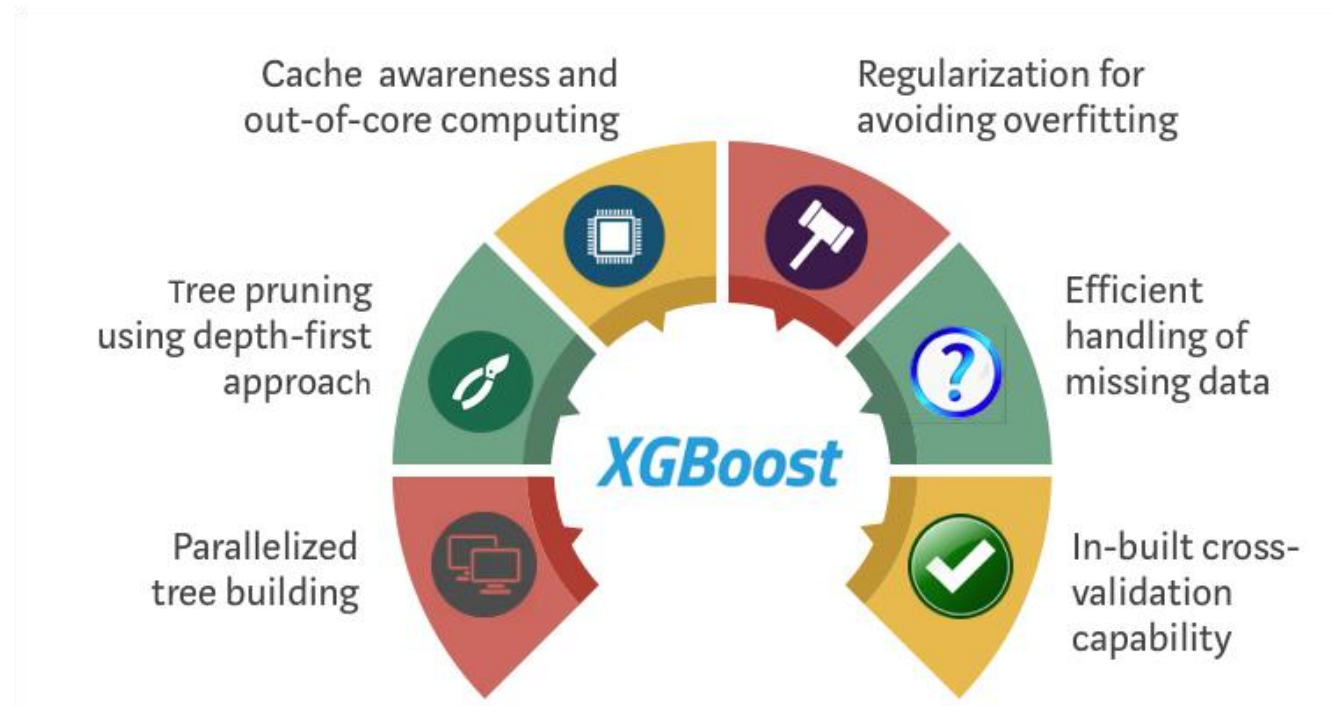
Source: [Geeksforgeeks](https://www.geeksforgeeks.org/gradient-boosting-machine/)

- In **gradient boosting**, each weak learner corrects its predecessor's error.
- Unlike AdaBoost, the **weights** of the training instances are not tweaked, instead, each predictor is trained using the **residual errors** of predecessor as labels.
- Unlike AdaBoost, each tree can be **larger than a stump**. However, the trees are still small. By fitting a small tree to the residuals, the GBM slowly improve  $\hat{f}$  in areas where it does not perform well.
- **Learning rate** shrinks the contribution of each tree. There is a trade-off between **learning rate** and **number of trees**. Learning rate slows down the process even further, allowing for more and different shaped trees to attack the residuals.
- **Aggregation** is done by adding the first tree predictions and a scaled (shrunk) version of the following trees.



# ➔ Extreme Gradient Boosting (XGBoost)

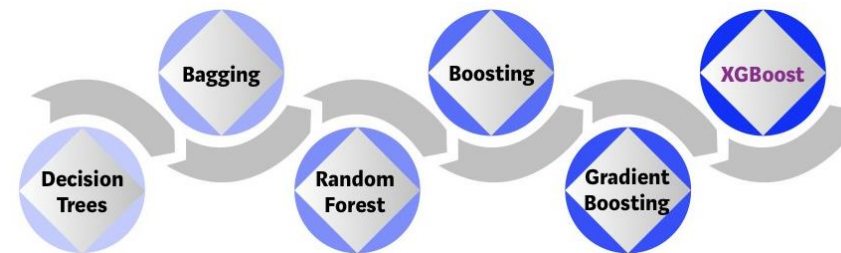
- **XGBoost** is a refined and customized version of a **gradient boosting** decision tree system, created with **performance** and **speed** in mind.
- **Extreme** refers to the fact that the algorithms and methods have been customized to push the limit of what is possible for gradient boosting algorithms.





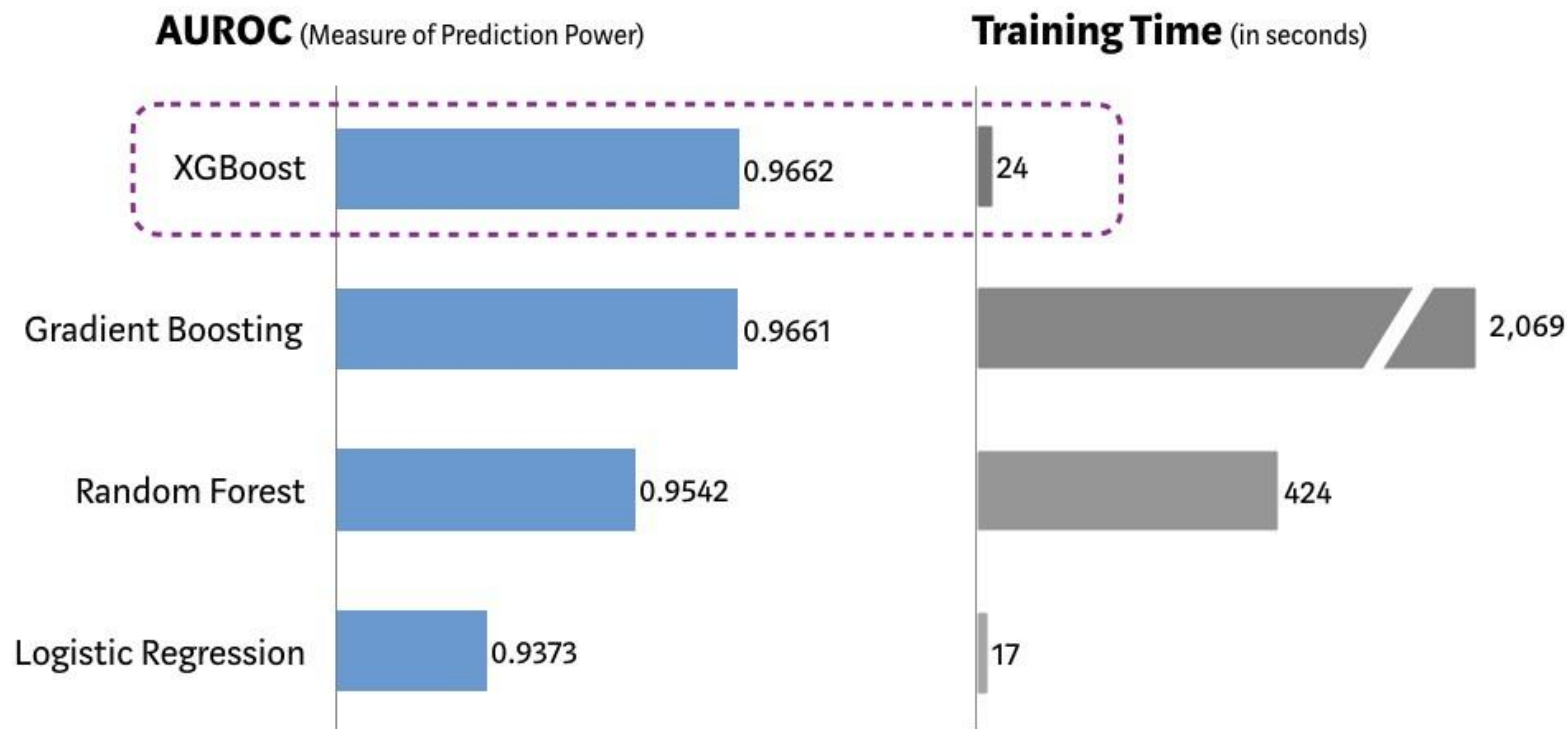


# Put it all together!



## Performance Comparison using SKLearn's 'Make\_Classification' Dataset

(5 Fold Cross Validation, 1MM randomly generated data sample, 20 features)

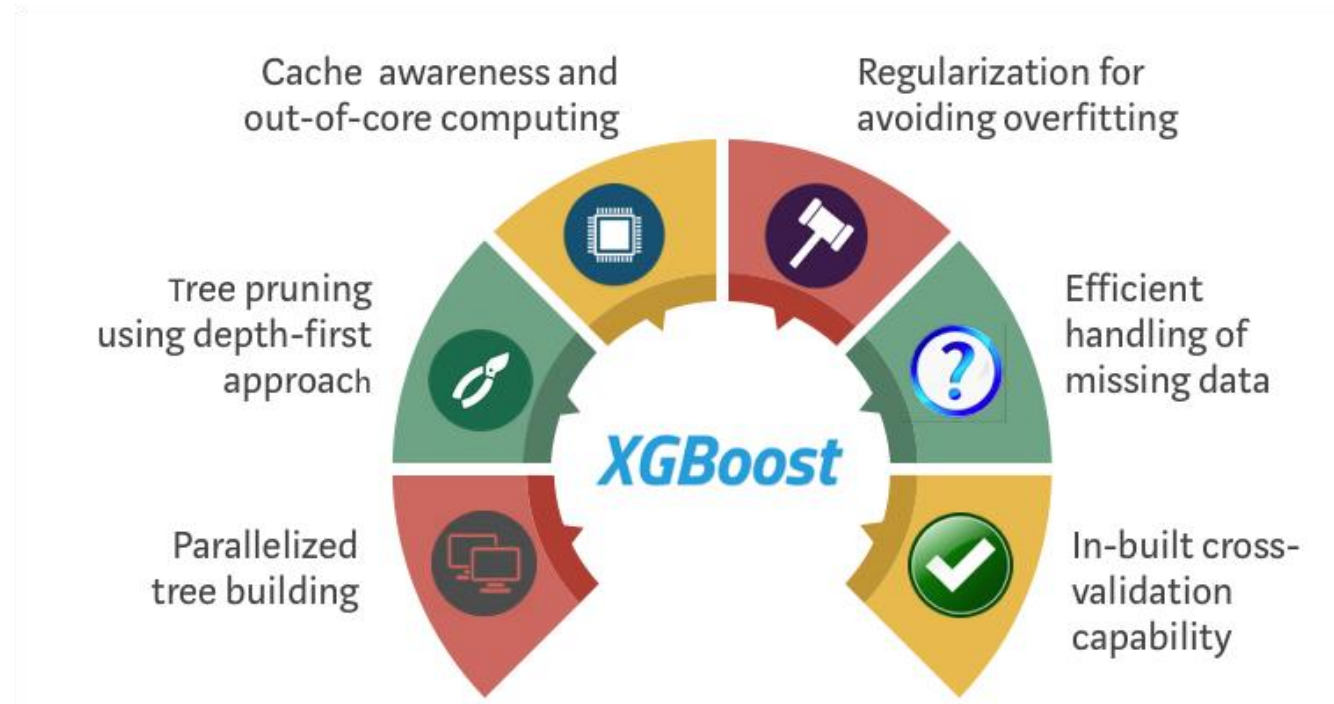


# Part II

## Pros and Cons

# ➔ XGBoost's Pros and Cons

## Pros:



## Cons:

- XGBoost is more difficult to understand, visualize and to tune compared to AdaBoost and random forests. There is a multitude of hyperparameters that can be tuned to increase performance.

# Students' questions

---

1. test