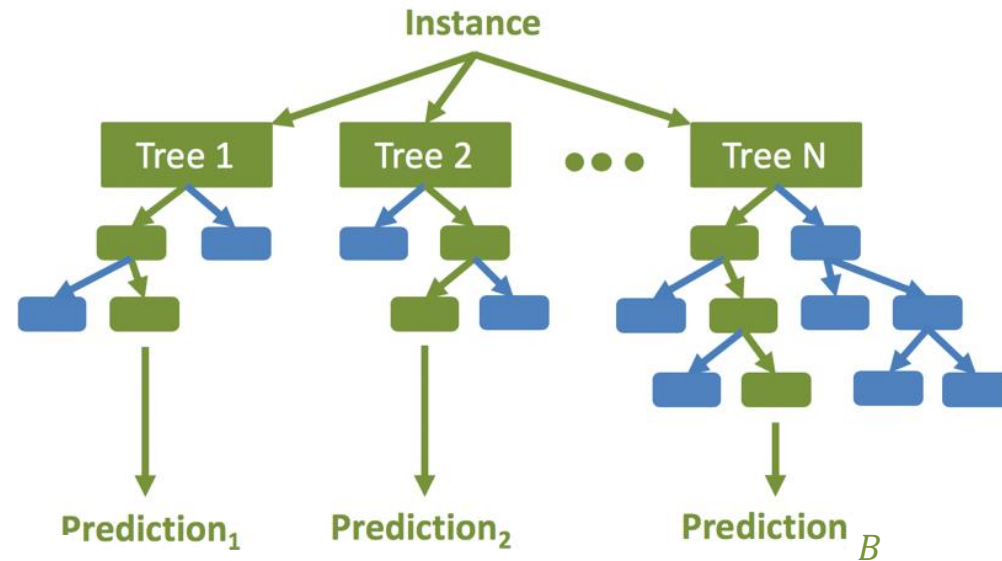


Class -20

Bagging and Random Forests

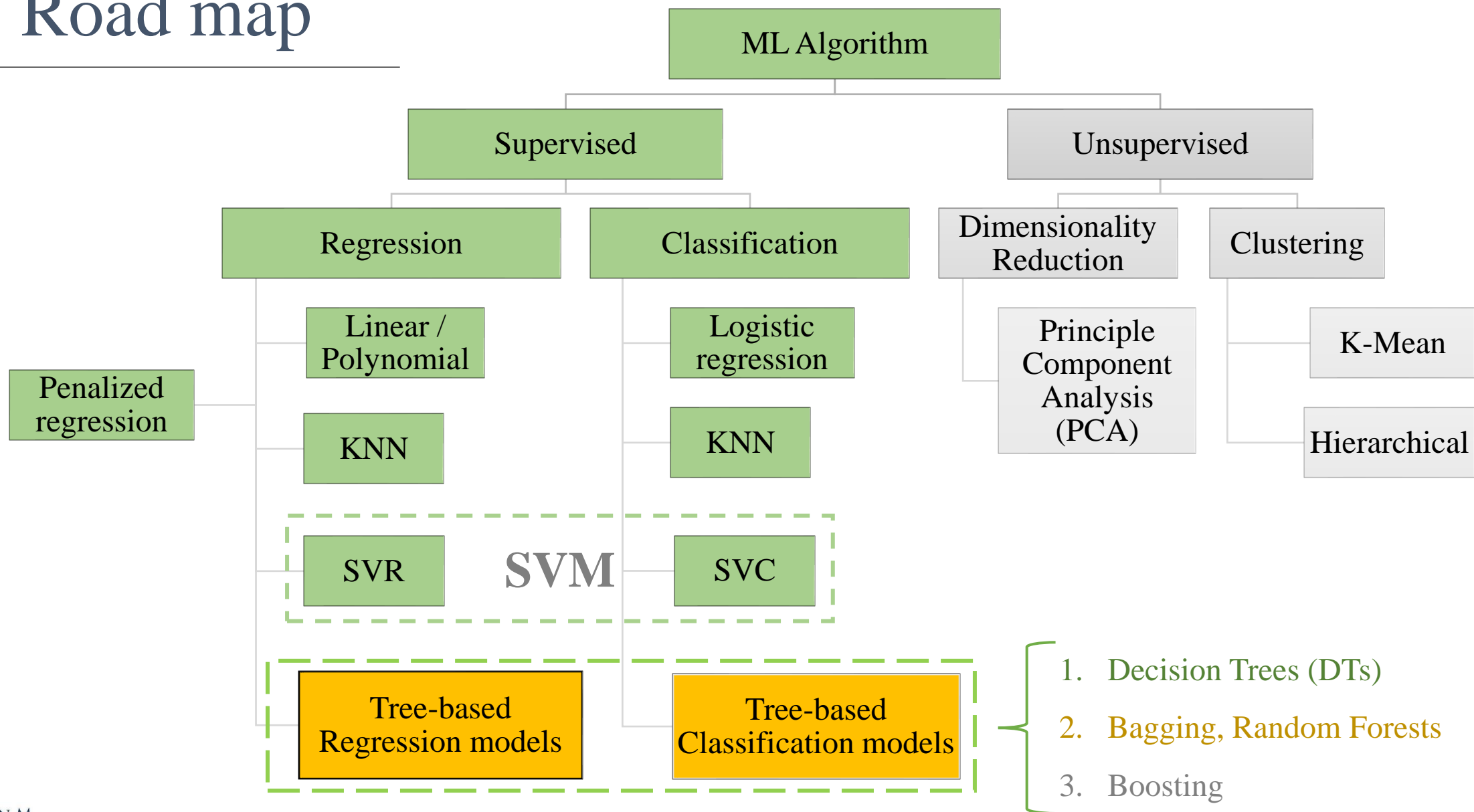


Prof. Pedram Jahangiry





Road map





Topics

Part I

1. Why not a simple tree?
2. Ensemble Learning

Part II

1. Bootstrap Aggregation (Bagging)
2. Random Forests

Part III

1. Hyperparameters
2. Feature importance

Part IV

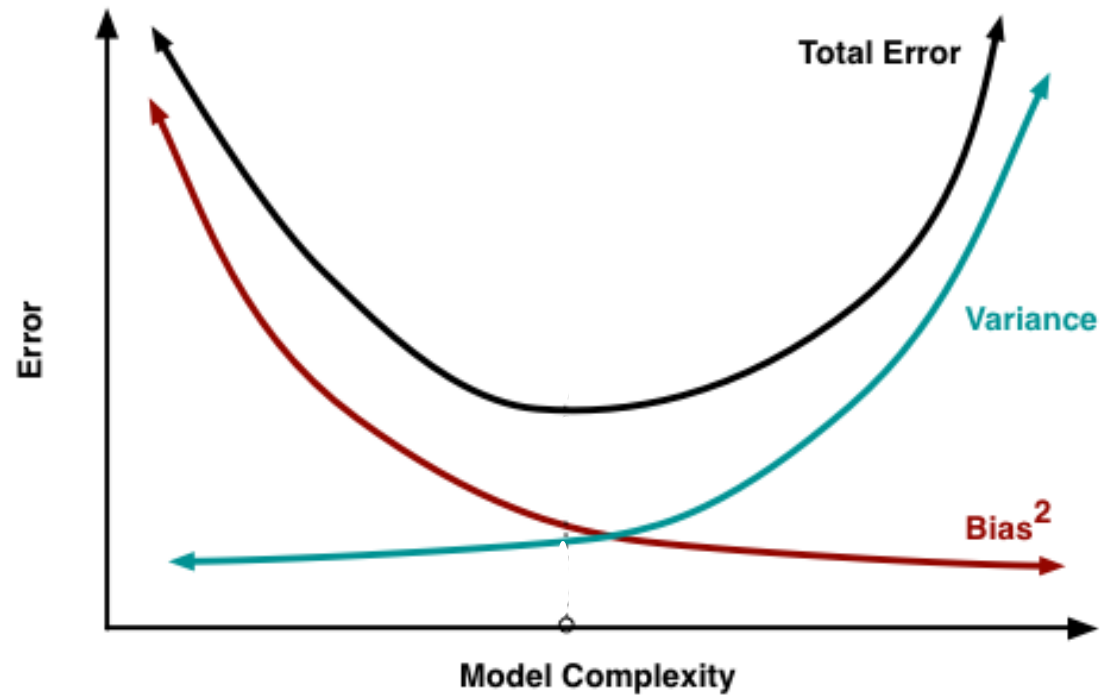
1. Pros and Cons
2. Applications in Finance

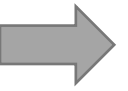
Part I

Motivation

➔ Why not a simple tree?

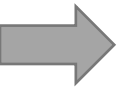
- Goal: Reduce the bias and variance at the same time?!





Ensemble Learning

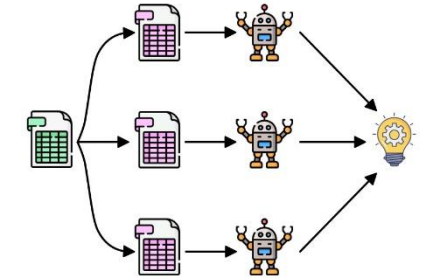
- Why not use the predictions of a group (an ensemble) of models?
- **Ensemble Learning**: Combining the predictions from a collection of models
- **Idea**: instead of trying to learn one super-accurate model, focusing on training many low-accuracy models and then combining the predictions given by those **weak** models.
- Ensemble learning typically produces **more accurate** and **more stable** predictions than the best single model.
- Ensemble learning methods are based on:
 1. Aggregation of **heterogeneous** learners (voting classifiers / average predictions)
 2. Aggregation of **homogeneous** learners (**bagging** and **boosting**)



Bagging vs Boosting

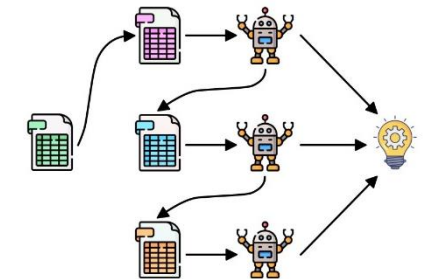
- **Bagging** consists of creating many “copies” of the training data (each copy is slightly different from another) and then apply the weak learner to each copy to obtain multiple weak models and then combine them.
- **Boosting** consists of using the “original” training data and iteratively creating 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.

Bagging



Parallel

Boosting



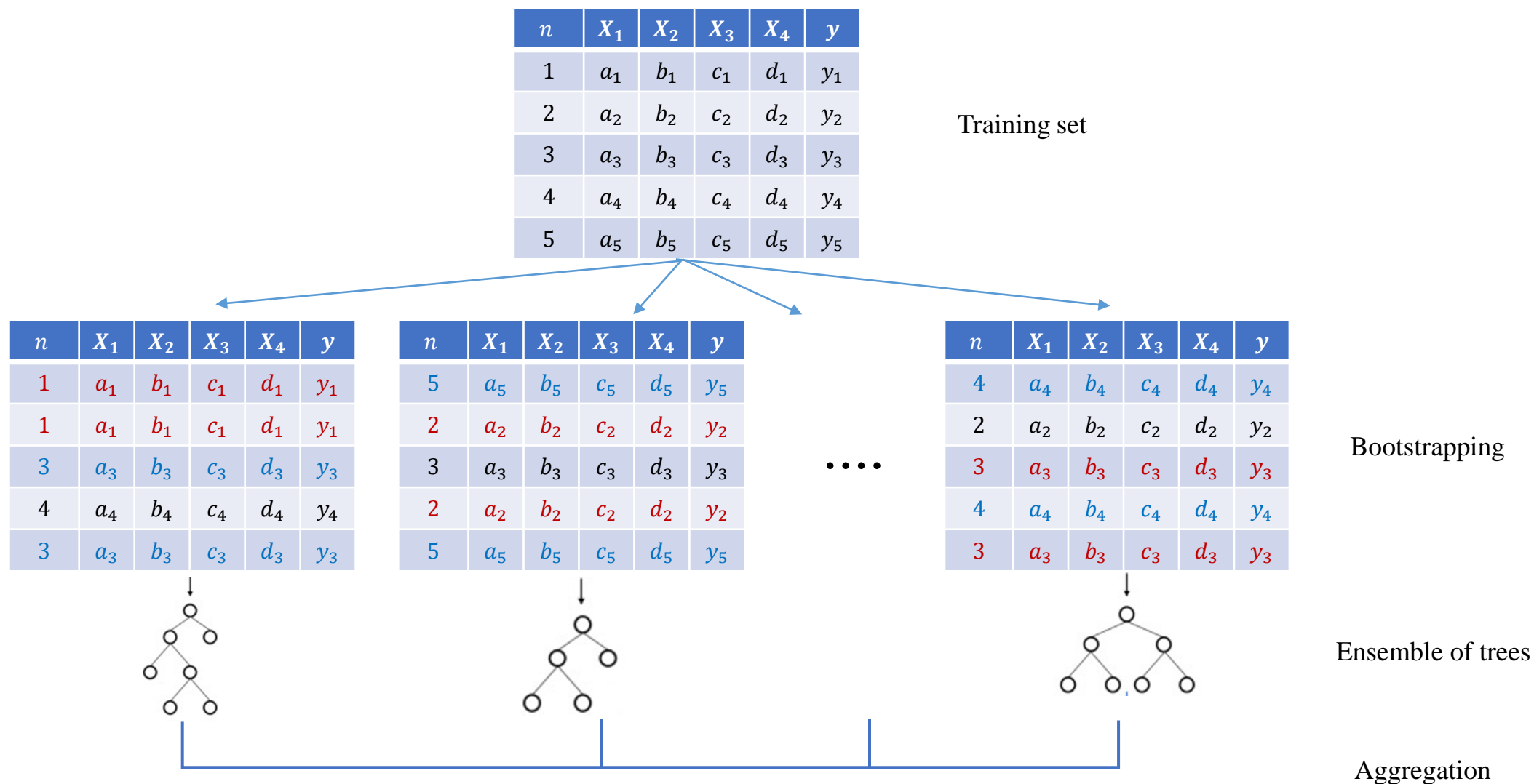
Sequential

Part II

Bagging and Random Forest



Bootstrap Aggregation (Bagging)



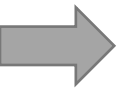


Bagging procedure

- Bootstrap aggregating (or bagging) is a general-purpose procedure for reducing the variance of a learning method.
- Given a training set, first we create B random samples (with replacement) of the training set.
- For each sample b , build a decision tree model f_b
- After training, we have B decision trees. The predictions for a new test observation x is obtained as the **average** of B predictions (for regression) or a **majority vote** (for classification).

$$y \leftarrow \hat{f}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{1}{B} \sum_{b=1}^B f_b(\mathbf{x}) \quad \text{or} \quad \text{most frequent } f_b(X)$$

- Bagging is a very useful technique because it helps to improve the **stability** of predictions and protects against **overfitting** the model. (why?)



Random Forests

- **Random forests** provide an **improvement** over The “vanilla” bagging algorithm by using a small trick that **decorrelates** the trees.
- RF uses a modified tree learning algorithm that at each split, inspects a **random subset of the features** (instead of inspecting the entire feature space!)
- The reason for doing this is to **avoid the correlation of the trees** in our forest:

*“Suppose there is one **very strong feature** in the data set. When using bagged trees, **most of the trees** will use that feature as the top split, resulting in an ensemble of similar trees that **are highly correlated**”*

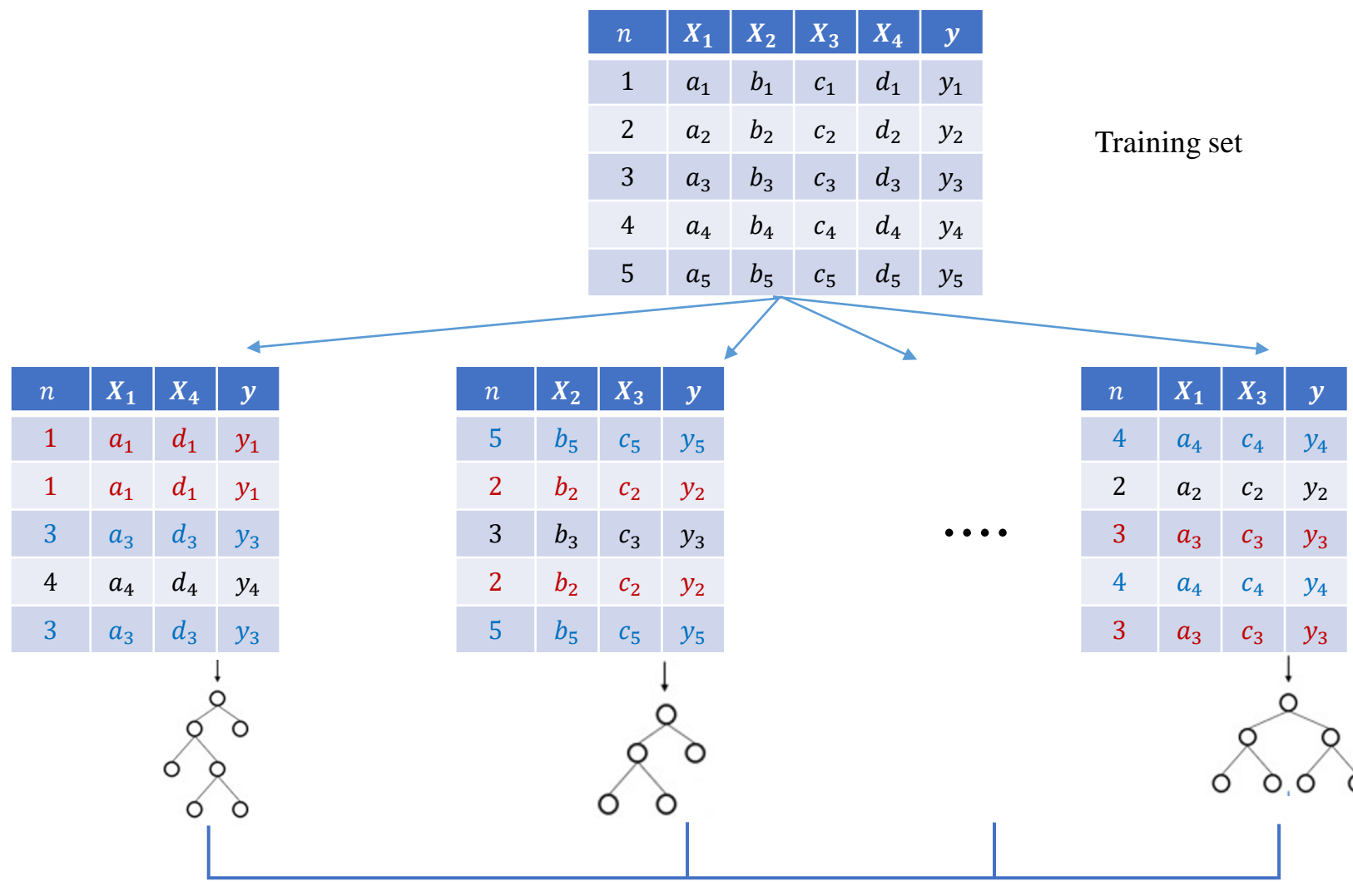




Random Forests

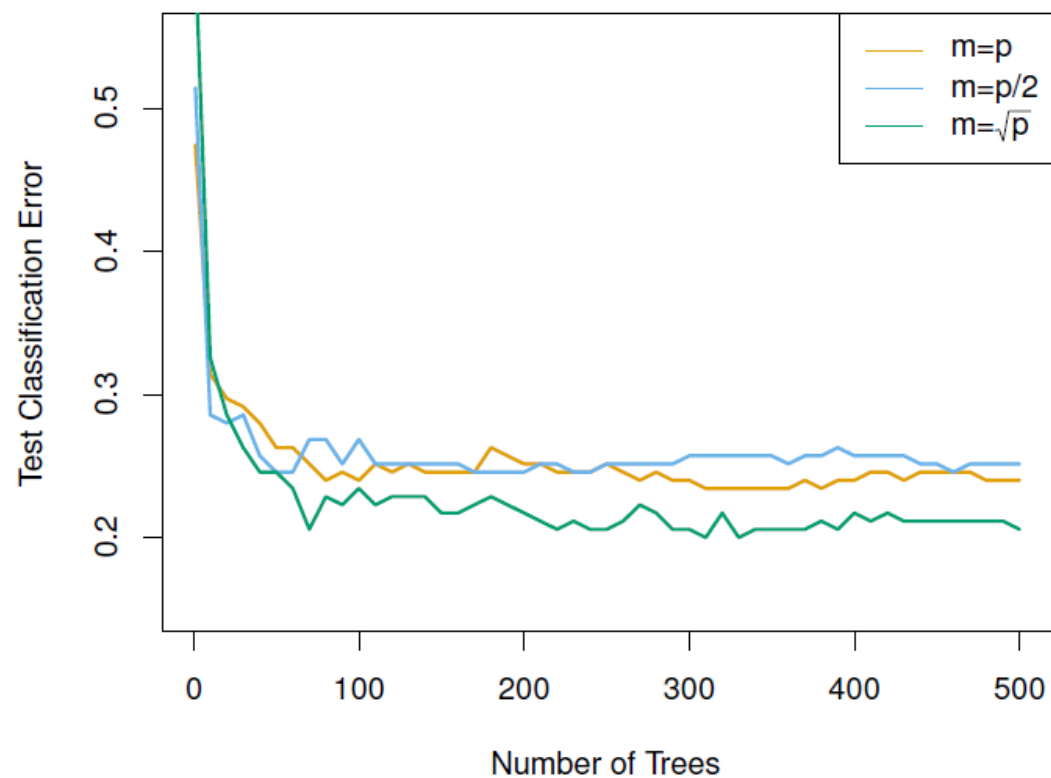


Random subset of features happens at each split!



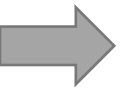
➔ Random Forests vs Bagging

- A random subset m , can be any subset of p features like $\frac{p}{2}$ or \sqrt{p} or $\log(p)$ etc.



Part III

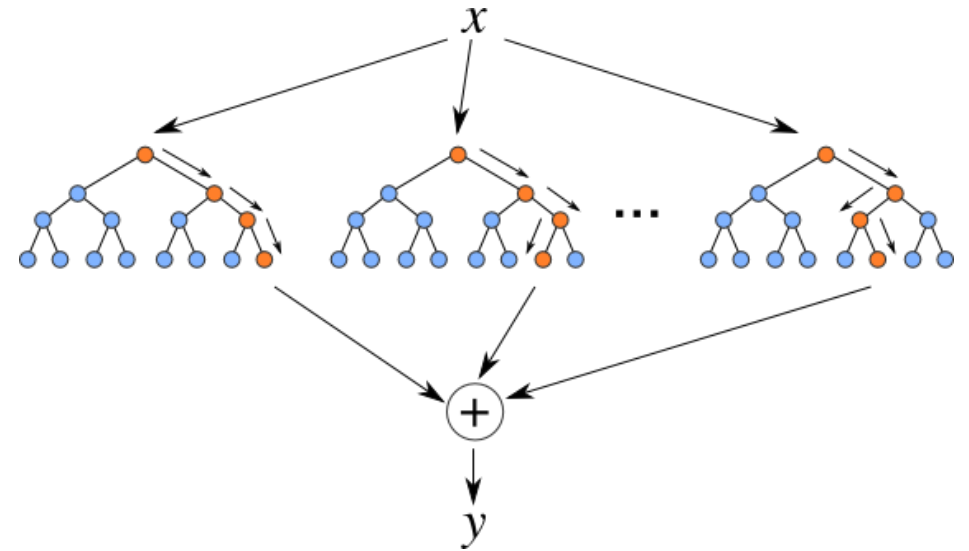
Hyper parameters
Feature importance



Hyperparameters

- The most important hyper parameters are:

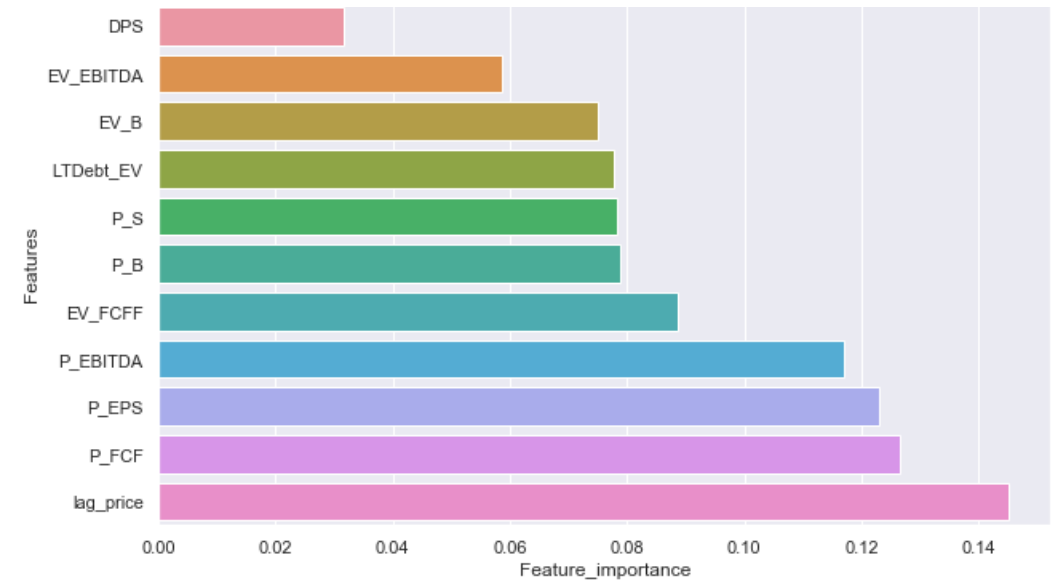
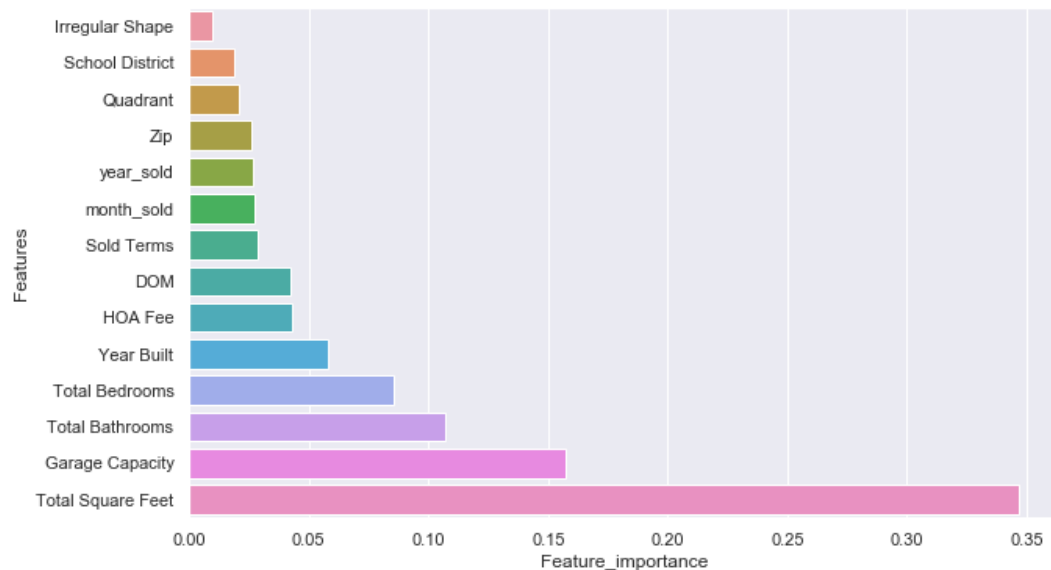
- ✓ The number of subset features (**m**)
- ✓ The number of trees to use (**B**)
- ✓ The minimum **size** of each node (or leaf)
- ✓ The maximum **number** of leaf nodes
- ✓ The maximum **depth** of each tree
- ✓ Criterion: gini, entropy



- **Grid search CV** could be used to tune a combination of hyper parameters.

→ Feature importance

- **Feature importance** refers to techniques that assign a **score** to input features based on how **useful** they are at predicting a target variable.
- For RF , the total amount that the **RSS is decreased** / **Gini index or entropy is decreased** due to splits over a given predictor, averaged over all B trees. A **large** value indicates an **important** predictor.



Part IV

Pros and Cons

Applications in finance

➔ Random Forests' Pros and Cons

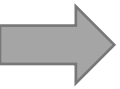
Pros:

- One of the most widely used ensemble learning algorithms. Because on top of all the advantages of a Decision Tree model (handling categorical data, learning nonlinear patterns, no preprocessing needed, nonparametric), it can:
- Avoid overfitting by reducing the model variance.
- Parallelizable!
- Great with high dimensionality
- Quick training and prediction speed (why?)



Cons:

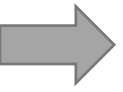
- Lacks the ease of interpretability of individual trees! Relatively **black box** type of algorithm.
- Can overfit without tuning the hyperparameters.



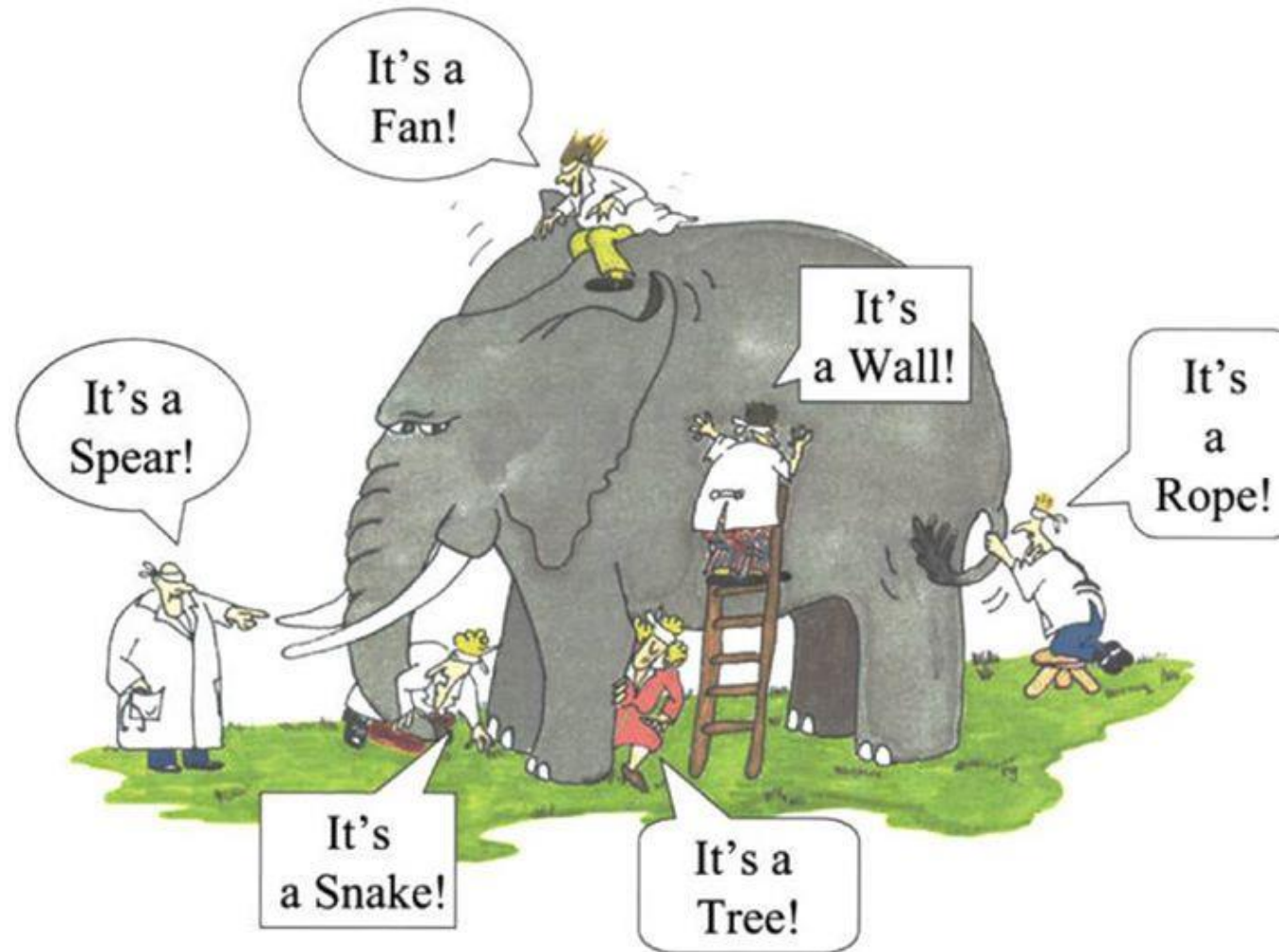
Random Forests' Applications in finance

- Factor based investment strategies for **asset allocation** or **investment selection**
- Predicting whether an IPO will be **successful** based on some ranked factors including:
 - Percent oversubscribed,
 - First trading day close price
 - First trading day volume
 - ...
- Fundamental factor modeling!
 - P/E, EV/EBITDA, P/S, P/B, ...





Question of the day: Wisdom of Crowds



Students' questions

1. I've heard the term bootstrapping in an econometric context before. Are the two concepts the same?
2. Are random forest trees allowed to be deeper with little cost to over-fitting when compared to decision trees?
3. I didn't really get what the disadvantages of random forest. I also don't really understand what bootstrapping does and how it works.
4. I don't know when you would use decision trees over random forest.
5. I don't get what's meant by a 'weaker learner' that's being applied to the copies of the training data.
6. When making a random forest, does the code make ALL possible decision trees with the data? If not, how do we decide which, and if yes, would that not take too long with large data sets?
7. Do we set the number of nodes for the tree or is this truly a "black box" that we just look at the output?
8. Is bootstrapping attempting to accomplish the same thing as splitting the data into test sets?