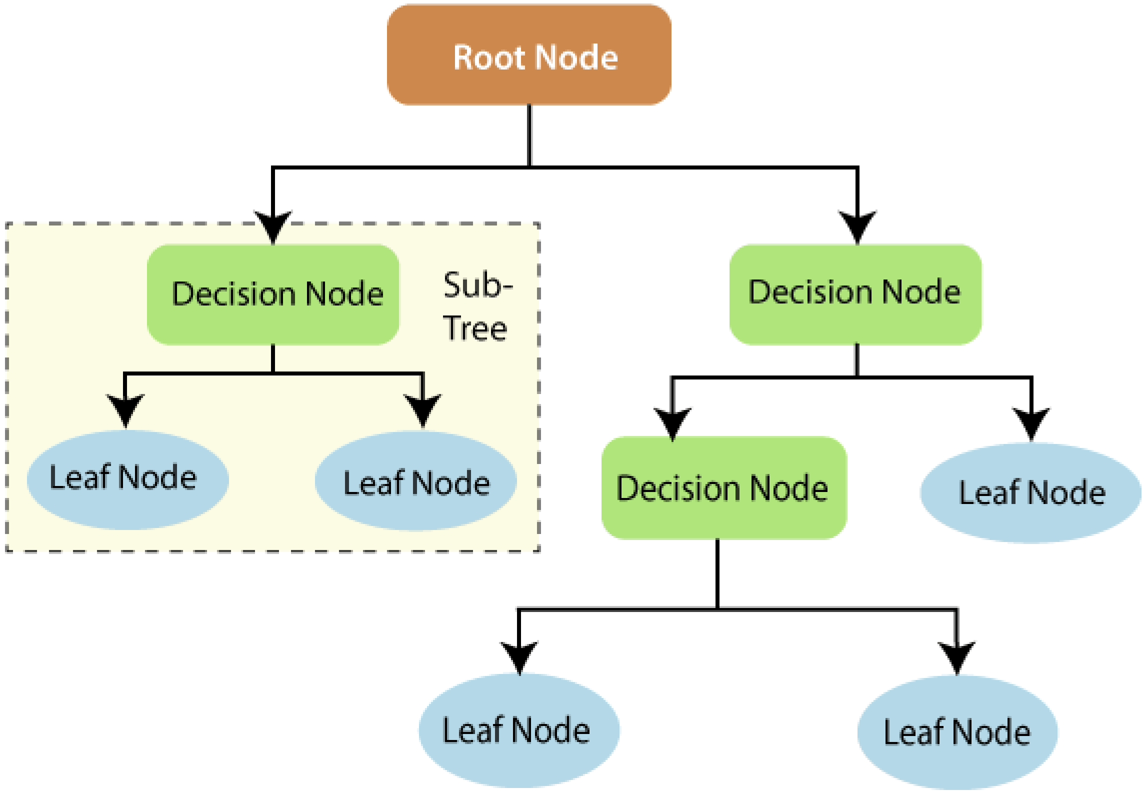
Module 10 – Part III
Machine Learning Boosting models

Decision Trees Fundamental questions

- Four fundamental questions to be answered:

1) What feature and cut off to start with?
2) How to split the samples?
3) How to grow a tree?
4) How to combine trees?

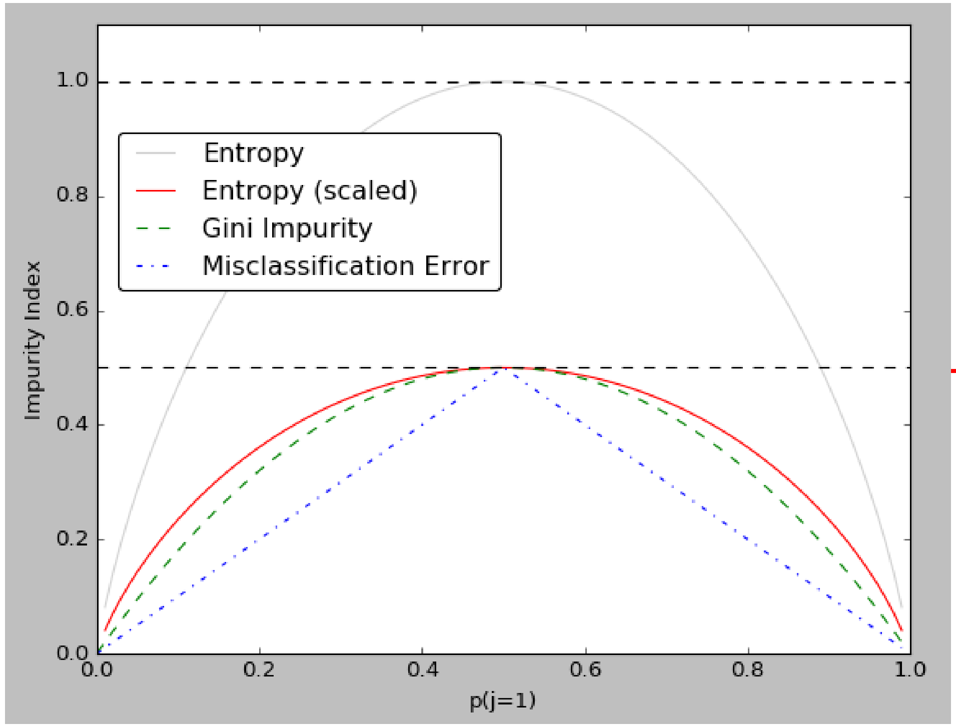


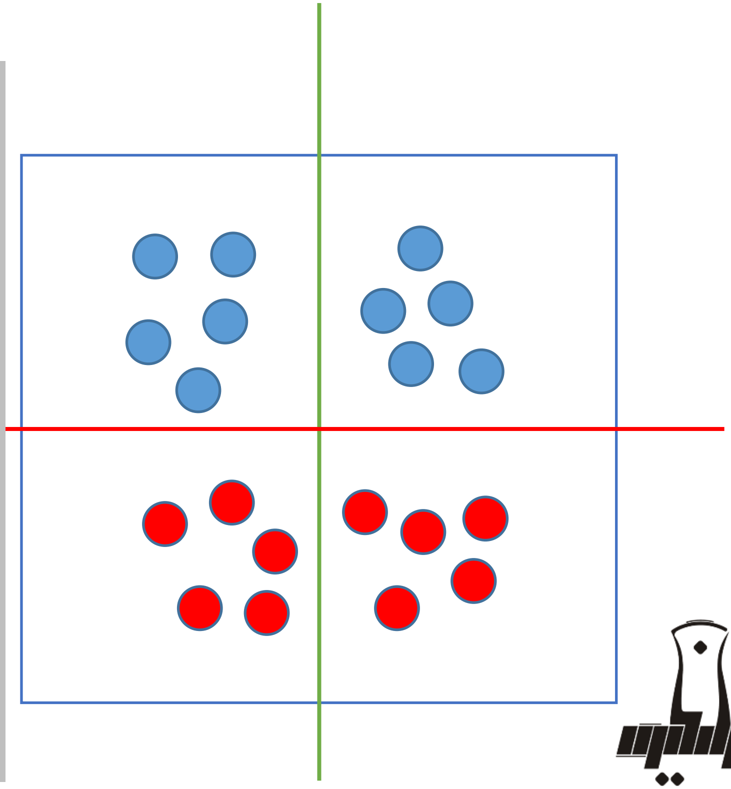
What feature and cut off to start with?

- Which feature and cut off adds the most information gain (minimum impurity)?

- Regression trees: MSE
- Classification trees:
  1. Error rate
  2. Entropy
  3. Gini Index

Control how a Decision Tree decides to split the data

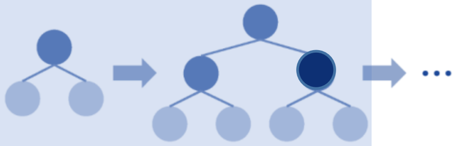
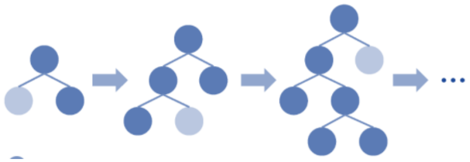
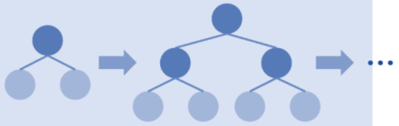




How to split the samples?

Method	Description
Pre-sorted and histogram based	This method sorts the data and creates histograms of the values before splitting the tree. This allows for faster splits but can result in less accurate trees.
GOSS (Gradient-based One-Side Sampling)	This method uses gradient information as a measure of the weight of a sample for splitting. Keeps instances with large gradients while performing random sampling on instances with small gradients.
Greedy method	This method selects the best split at each step without considering the impact on future splits. This method May result in suboptimal trees

How to grow a tree?

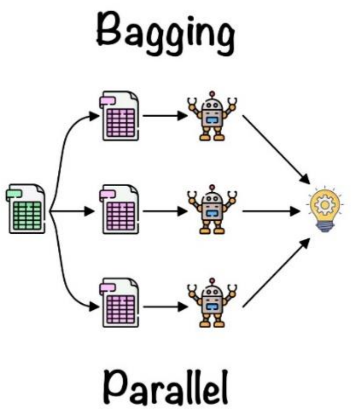
Algorithm	Description
Depth-Wise Level-Wise	Repeatedly splitting the data along the feature with the highest information gain, until a certain maximum depth is reached. Resulting in a tree with a balanced structure, where all leaf nodes are at the same depth. 
Leaf-wise	Repeatedly splitting the data along the feature with the highest information gain, until all leaf nodes contain only a single class. Resulting in a tree with a highly unbalanced structure, where some branches are much deeper than others. 
Symmetric	Builds the tree by repeatedly splitting the data along the feature with the highest information gain, until a certain stopping criterion is met (e.g. a minimum number of samples per leaf node). Resulting in a more balanced tree structure than leaf-wise growth. 

Jon M. Huntsman School of Business
Utah State University

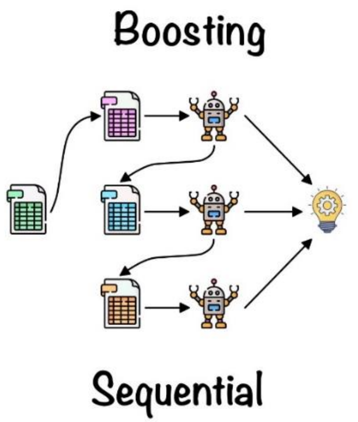
Pedram Jahangiry

How to combine trees?

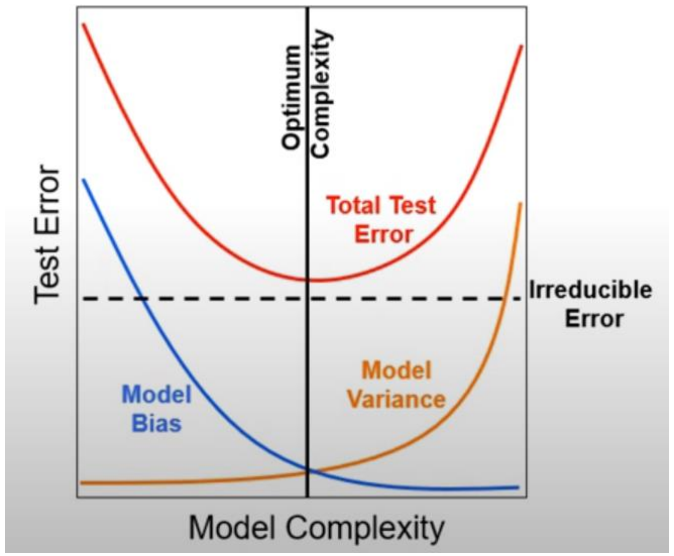
- Bagging consists of creating many "copies" of the training data (each copy is slightly different from another) and then apply the weak learner to each copy to obtain multiple weak models and then combine them.
- In bagging, the bootstrapped trees are independent from each other.

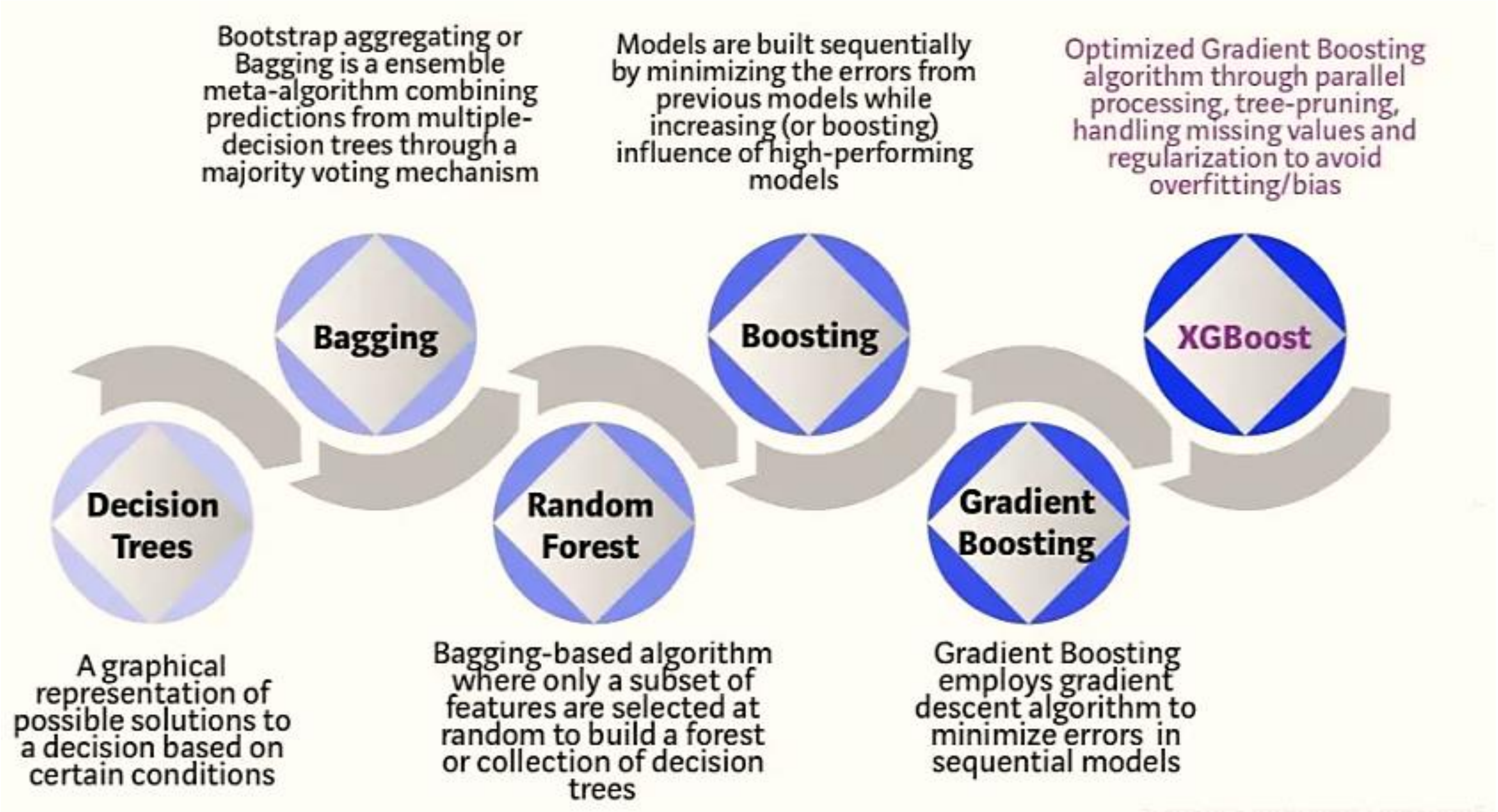


- Boosting consists of using the "original" training data and iteratively creating multiple models by using a weak learner. Each new model tries to "fix" the errors which previous models make.
- In boosting, each tree is grown using information from previous tree.



Evolution of XGBoost



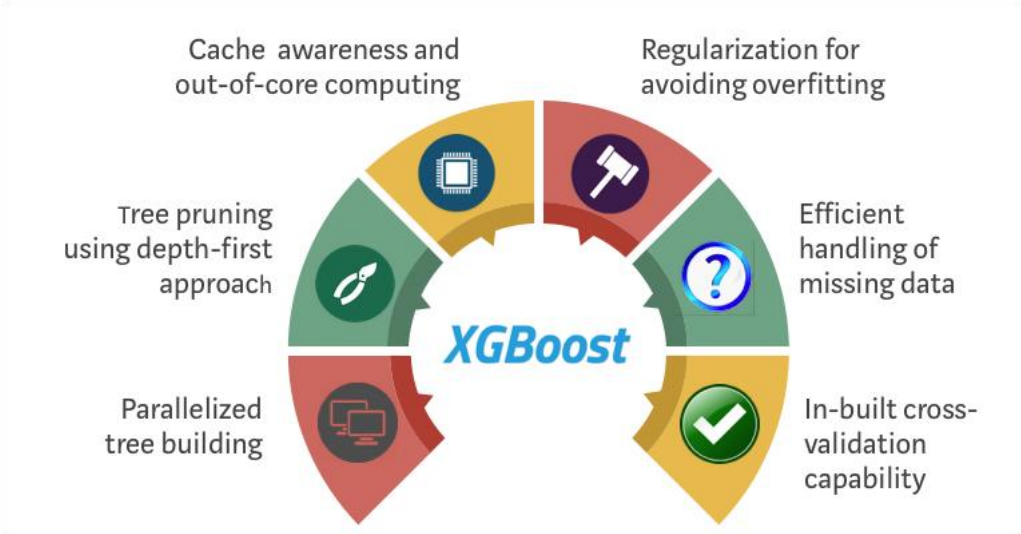


Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

XGBoost: eXtreme Gradient Boosting

- XGBoost is an open-source gradient boosting library developed by Tianqi Chen (2014) focused on developing efficient and scalable machine learning algorithms.
- 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.
- XGBoost includes several other features that can improve model performance, such as handling missing values, automatic feature selection, and model ensembling.



LightGBM (Light Gradient Boosted Machine)

- LightGBM is an open-source gradient boosting library developed by Microsoft (2016) that is fast and efficient, making it suitable for large-scale learning tasks.
- LightGBM can handle categorical features, but requires one-hot encoding, ordinal encoding or other preprocessing
- LightGBM includes several other features that can improve model performance, such as handling missing values, automatic feature selection, and model ensembling.

LightGBM

Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

CatBoost (Category Boosting)

- CatBoost is an open-source gradient boosting library developed by Yandex (2017) that is specifically designed to handle categorical data.
- CatBoost can handle categorical features directly, without the need for one-hot encoding or other preprocessing.
- CatBoost includes several other features that can improve model performance, such as handling missing values, automatic feature selection, and model ensembling.

XGBoost vs LightGBM vs CatBoost

	XGBoost	LightGBM	CatBoost
Developer	Tianqi Chen (2014)	Microsoft (2016)	Yandex (2017)
Base Model	Decision Trees	Decision Trees	Decision Trees
Tree growing algorithm	Depth-wise tree growth Leaf-wise is also available	Leaf-wise tree growth	Symmetric tree growth
Parallel training	Single GPU	Multiple GPUs	Multiple GPUs
Handling categorical features	Encoding required (one-hot, ordinal, target, label, …)	Automated encoding using categorical feature binning	No encoding required
Splitting method	Pre-sorted and histogram based	GOSS (Gradient based one-side sampling)	Greedy method

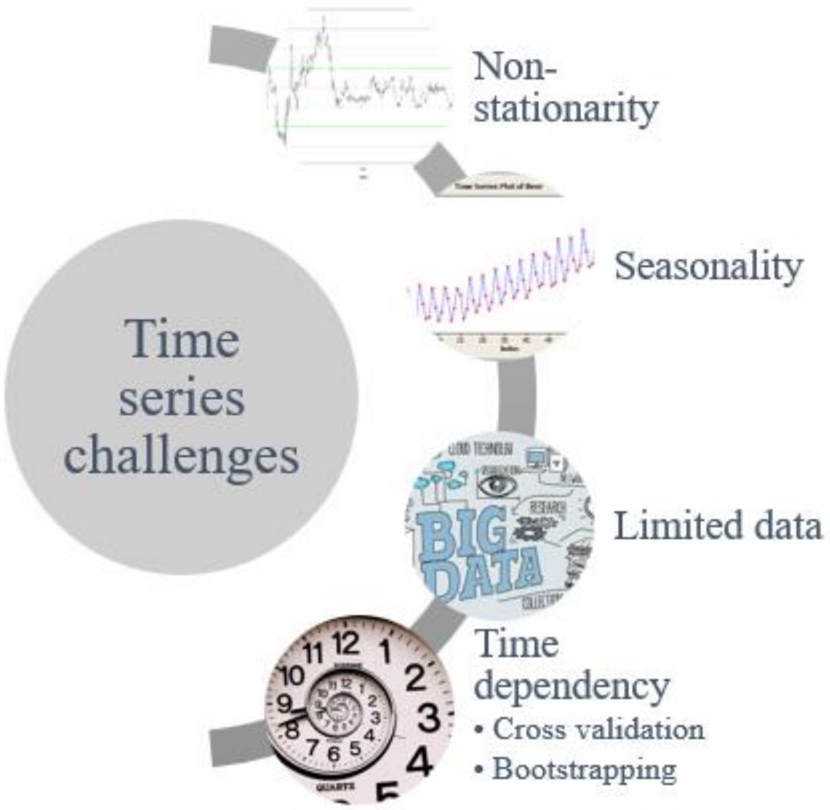
dmlc XGBoost LightGBM CatBoost

Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

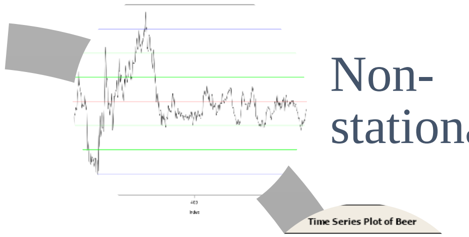
Module 10 – Part III

Challenges in Time Series Machine Learning

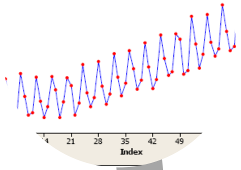


Challenges in Time Series Machine Learning

Time series challenges



Non-stationarity



Seasonality

Limited data

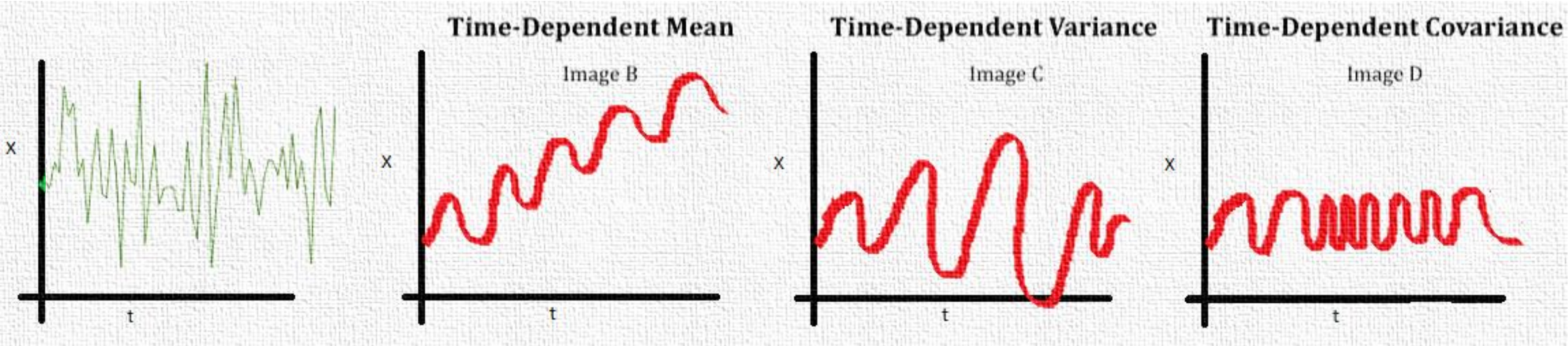
Time dependency
- Cross validation
- Bootstrapping

JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

Stationarity

- Stationary vs Non-Stationary Data. What makes a data set Stationary?
- In a stationary timeseries, the statistical properties do not depend on the time



Stationary	Non-Stationary	Non-Stationary	Non-Stationary

- Data with trend and seasonality are NOT stationary!

Time Series Cross Validation

- With time series data, we cannot shuffle the data! TS data is not IID.
- We also need to avoid data leakage!

- The main time series CV methods are:
  1) Purged K-Fold CV
  2) Walk forward rolling / expanding window
  3) Combinatorial purged CV

Purged K-Fold CV

- Leakage takes place when the training set contains information that also appears in the testing set.
- Leakage will enhance the model performance
- Solution: Purging and Embargoing

- Purged K-Fold CV: Adding purging and embargoing whenever we produce a train/test split in K-Fold CV.

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

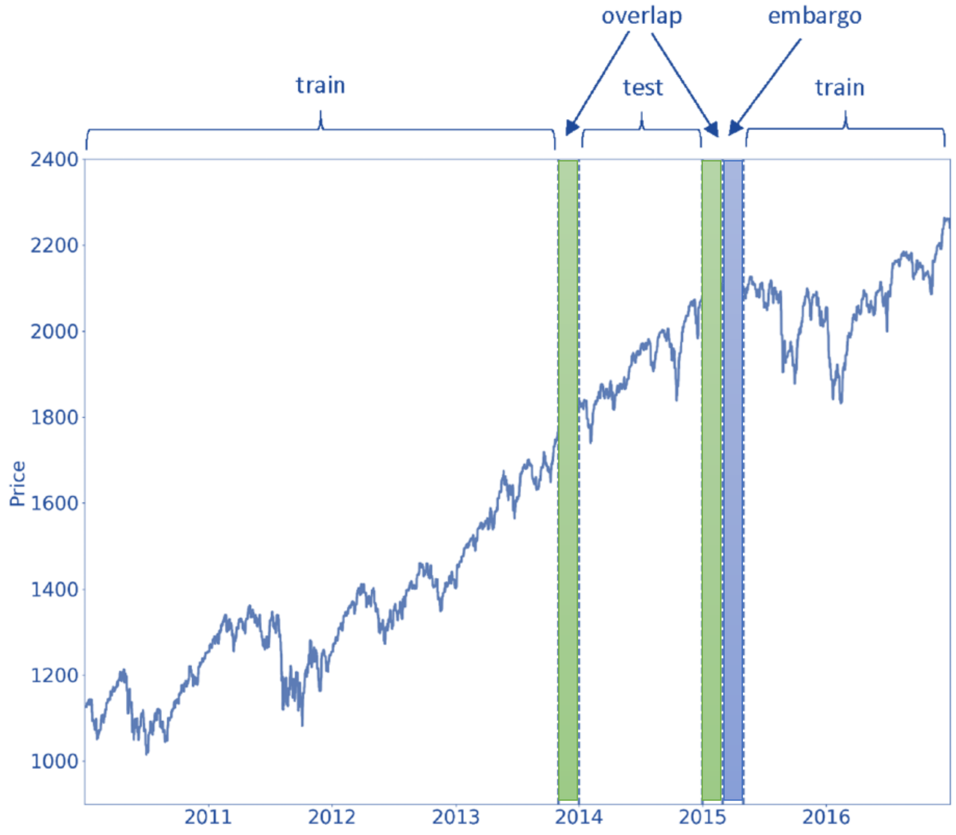
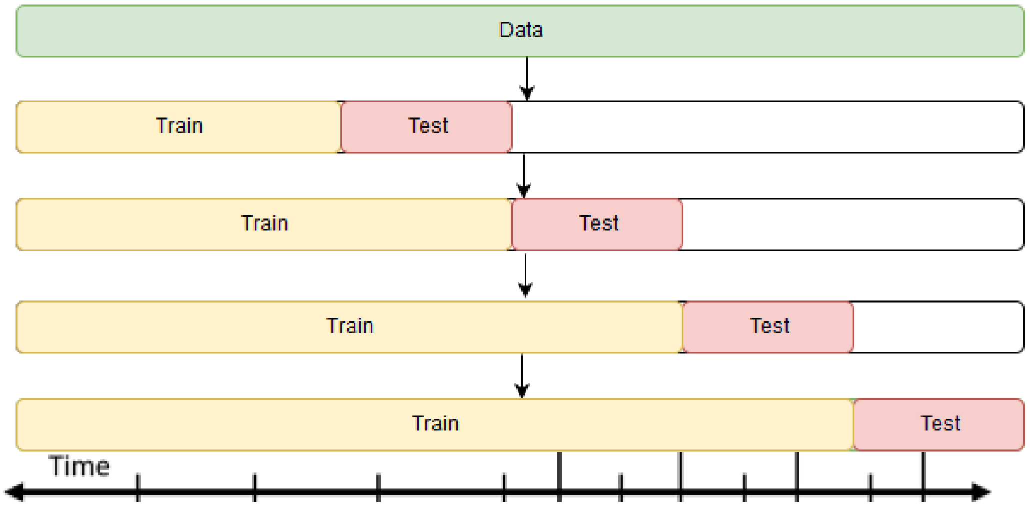


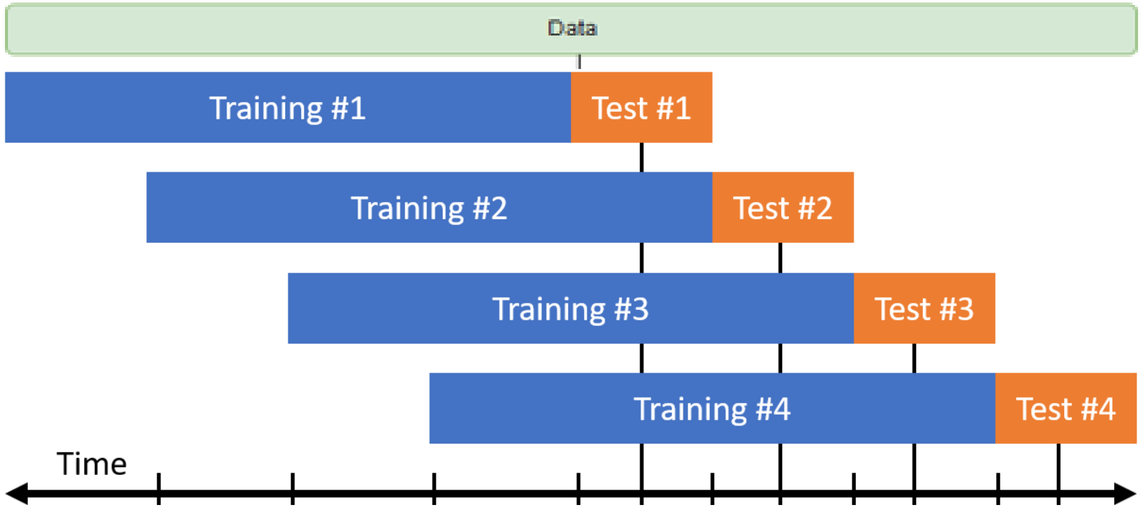
FIGURE 7.3 Embargo of post-test train observations

Walk Forward Cross Validation

Walk forward cross validation
Expanding windows

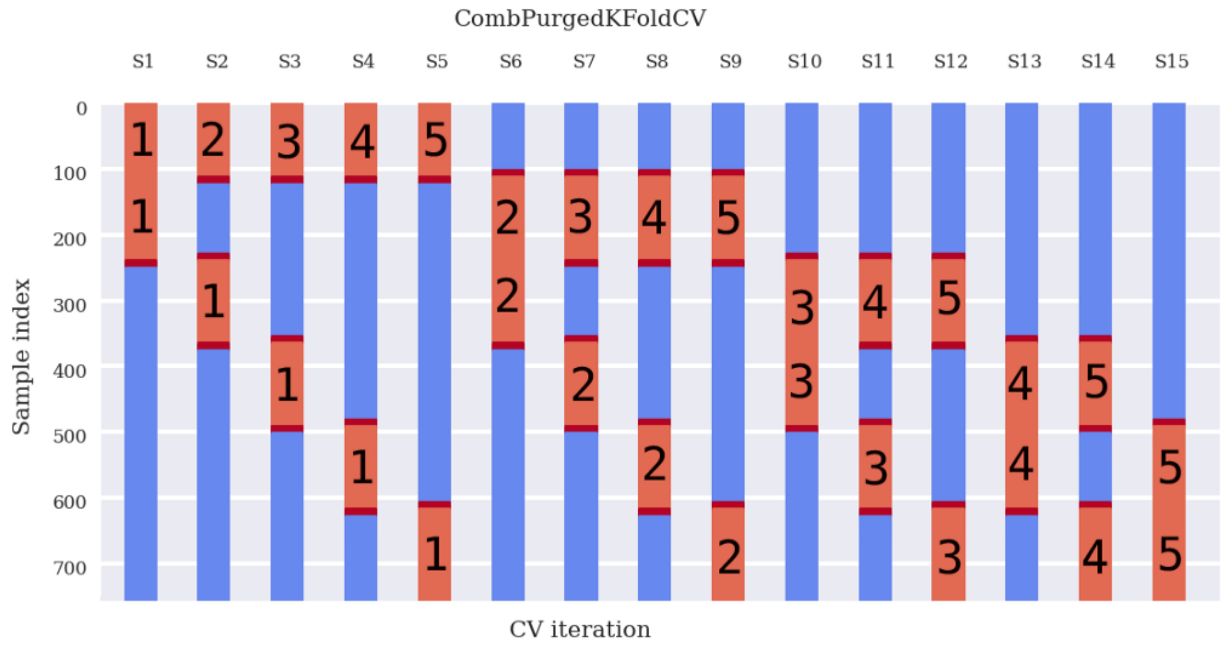


Walk forward cross validation
Rolling windows



Combinatorial Purged Cross Validation (CPCV)

- The goal is to generate multiple unique back-test path that span the entire data set.
- In each path, we can look at the model's OOS performance for the entire time period.



Time Series Bootstrapping

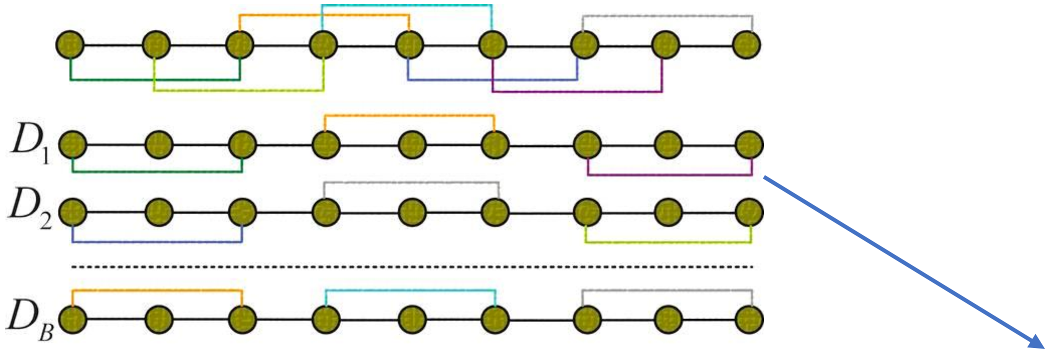
- IID bootstrapping (random sample with replacement) does not work for time series data with temporal dependency.
- Time series Bootstrapping methods:
  - Parametric (based on models with iid residuals and resampling from residuals. Example: ARIMA bootstrap)
  - Non-parametric block bootstrap (data is directly resampled. Assumption: blocks can be samples so that they are approximately iid)
    - Moving Block Bootstrap (MBB)
    - Circular Block Bootstrap (CBB)
    - Stationary Bootstrap (SB)

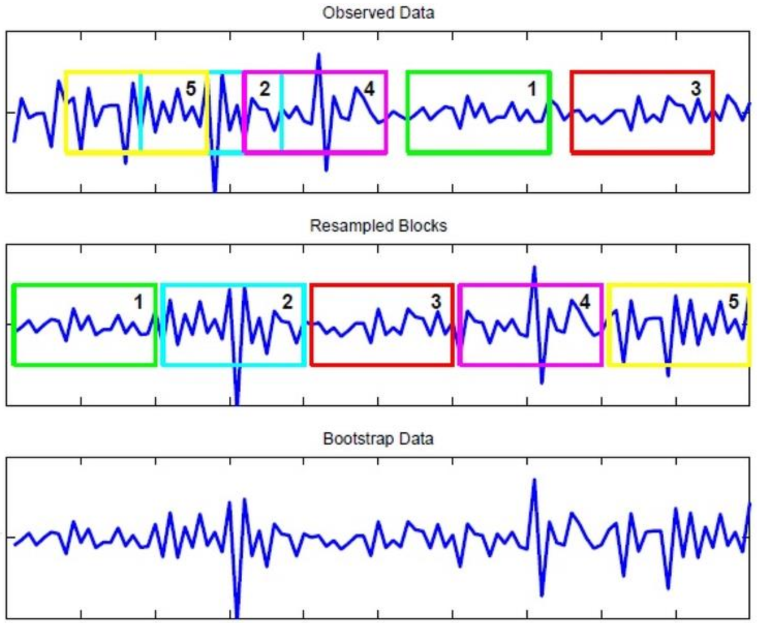
Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Moving Block Bootstrap (MBB)

- Moving Block Bootstrap, samples overlapping fixed size blocks of m consecutive observations.
- Blocks starts at indices 1, … , T-m+1

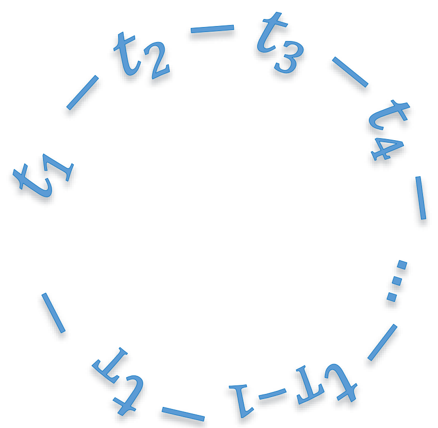




Circular Block Bootstrap (CBB)

- CBB is a simple extension of MBB which assumes the data live on a circle so that $y_{T+1} = y_1$, $y_{T+2} = y_2$, etc.

- CBB has better finite sample properties since all data points get sampled with equal probability.



Stationary Bootstrap (SB)

- In SB, the block size is no longer fixed.
- Chooses an average block size of m rather than an exact block size.
- Popularity of SB stems from difficulty in determining optimal m
- Once applied to stationary data, the resampled pseudo time series by SB are stationary. This is not the case for MBB and CBB.