PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

Lecture outline:

- Decision Tree Review
- Ensemble Methods
- Bootstrap
- Training and Tuning Bagging Models
- Tree Bagging
- Random Forest
- Ensemble Tree Shapley Values
- Ensemble Tree Methods Hands-on

Motivation

Motivation for ensemble tree, tree bagging and random forest

- builds on flexible and easy to understand decision trees
- extend to much more powerful predictive machine learning methods, with ensemble learning

Solatary black spruce tree in Hinton, Alberta, Canada, image from https://hikebiketravel.com/6-fun-things-to-do-in-hinton-alberta-in-winter.

Black spruce forest near Hinton, Alberta, east of Japser National Park, Canada, image from https://en.wikivoyage.org/wiki/Hinton.

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

Lecture outline:

- **Decision Tree Review**

Decision Tree

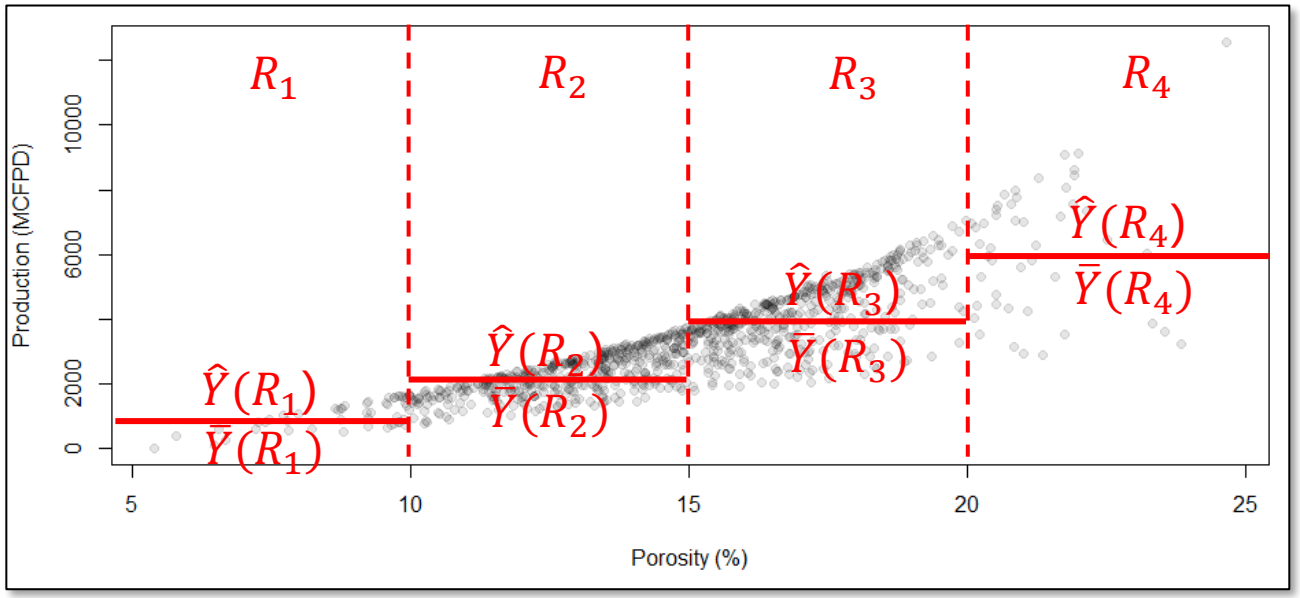
Predictor feature space segmentation-based prediction

The fundamental idea is to divide the predictor space, $X_1, \ldots, X_m$, into J mutually exclusive, exhaustive regions

- **mutually exclusive** – any combination of predictors only belongs to a single region, R_j
- **exhaustive** – all combinations of predictors belong a region, R_j, regions cover entire feature space (range of the variables being considered)

The same prediction in each region, mean of training data in region, $\hat{Y}(R_j) = \bar{Y}(R_j)$ or $\hat{Y}(R_j) = \arg\max(Y(R_j))$

For example, predict production, $\hat{Y}$, from continuous porosity, X_1,



4 region decision tree with data and predictions by region, $R_j, j = 1, \ldots, J$.

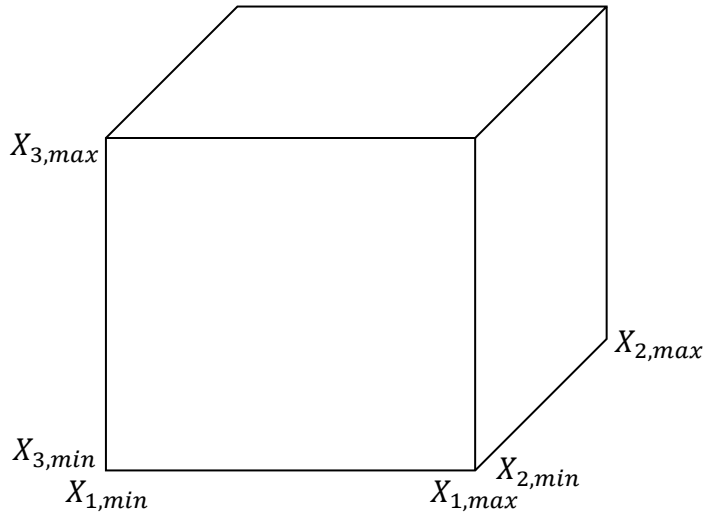
Decision Tree

Predictor Feature Space

The space that includes all possible estimation problems, i.e., the combination of all possible predictor feature values, $x_1, x_2, \ldots, x_m$.

Typically this is defined by the range of possible values, $x_\alpha \in [X_{\alpha,min}, X_{\alpha,max}]$, resulting in,

- line segment – 1 predictor feature
- rectangle – 2 predictor features
- rectangular cuboid - 3 predictor features
- hyperrectangle - >3 predictor features



Schmatic of predictor feature space.

Decision Tree

How do we construct regions, $R_1, R_2, \ldots, R_J$, for our predictions?

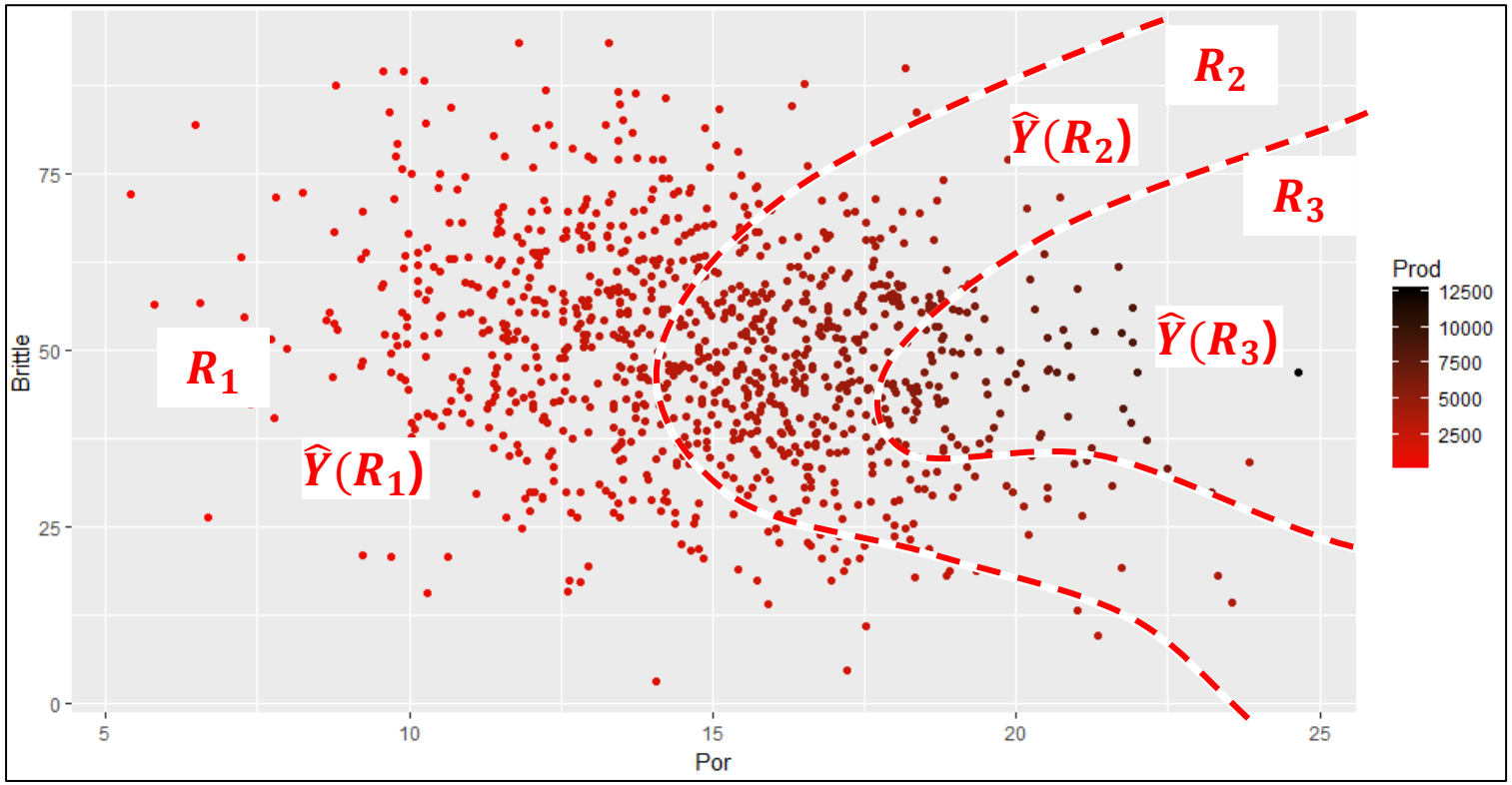
- Our regions could be any shape!
- Consider this 2 predictor feature and 1 response feature problem.

Consider this prediction of unconventional well production (MCFPD) from porosity (%) and brittleness (%)

Efficient Regions!

But,

- complicated
- difficult to calculate
- many parameters



Predict production from porosity and brittleness with 3 regions with data and predictions by region, $R_j, j = 1, \ldots, J$.

Decision Tree

How do we construct regions, $R_1, R_2, \ldots, R_J$, for our predictions?

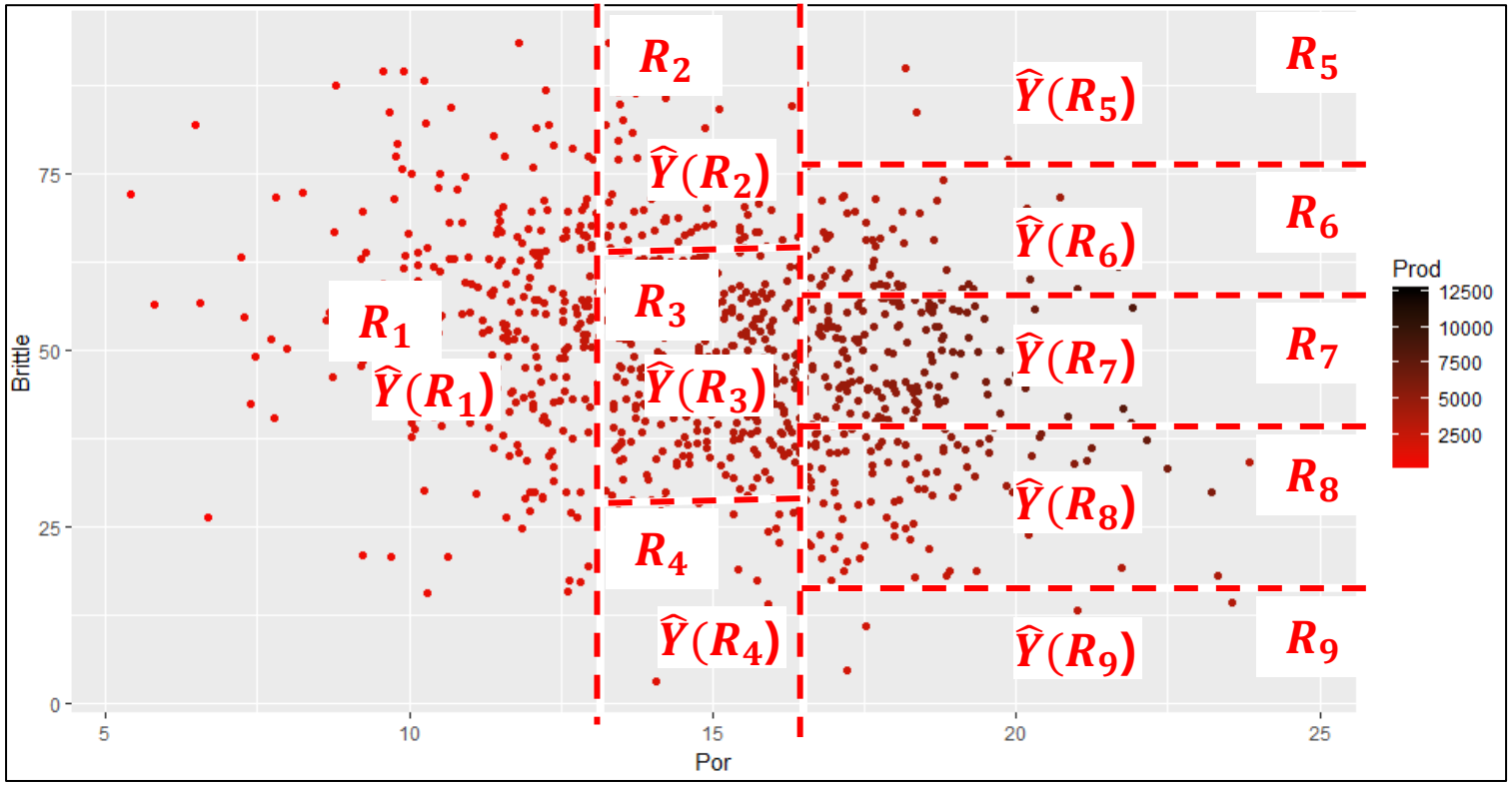
- They could be any shape!
- We decide to use high-dimensional cuboid → simple interpretation / rules

Hierarchical segmentation over the features – somewhat **flexible, compact model!**

Simple Regions!

And,

- easy to calculate
- few parameters
- represented as a tree model



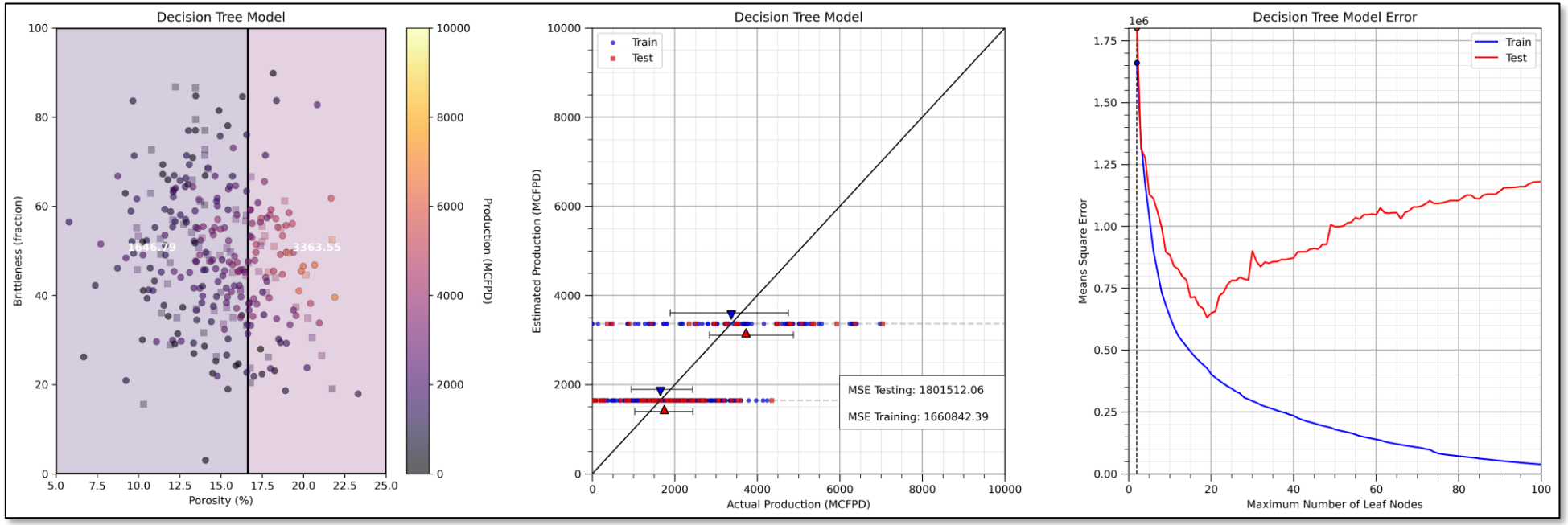
Predict production from porosity and brittleness with 9 cuboid regions with data and predictions by region, $R_j, j = 1, \ldots, J$.

Decision Tree Training

How do we construct regions, $R_1, R_2, \ldots, R_J$, for our predictions?

Recursive, binary splitting

- **Greedy** - at each step, the method selects the choice that minimizes RSS. There is no attempt to look ahead, jointly optimize over multiple choices
- **Top-down** - at the beginning all data belong to a single region, top of the tree, greedy selection of the single best split over any feature that best reduces the RSS



First best split, file is Interactive_Decision_Tree.ipynb, also see Decision Tree chapter of Applied Machine Learning e-book.

Decision Tree Loss

The decision tree loss function

Regression Residual Sum of Squares:

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

where $\hat{y}_{R_j}$ is the estimate and y_i is the training data value.

$$\hat{y} = \frac{1}{|R_j|} \sum_{X_i \in R_j} y_i$$

$|R_j|$ is the number of training data in j region and y_i is a training data, determined by predictor feature values X_i in R_j.

Classification Weighted Average Gini Impurity:

$$Gini_{total} = \sum_{j=1}^{J} \frac{N_j}{N} Gini(j)$$

where N_j number of training data in region j and N is the number of training data.

$$Gini(j) = 1 - \sum_{c=1}^{C} p_{j,c}^2$$

where C number of cateogries, and $p_{j,c}$ is the proportion of training data in region j of category c.

Example

	c=1	c=2	c=3
N_j	4	3	3
p_i	0.4	0.3	0.3
p_i^2	0.16	0.09	0.09

$$\sum p_i^2 = 0.34$$

$$Gini(j) = 1 - 0.34 = 0.66$$

Decision Tree Hyperparameters

Decision Tree Hyperparameters

- **Number of regions** – very easy to understand, you know what the model will be

Other Hyperparameters,

- **Minimum reduction in RSS** – could stop early, e.g., a low reduction in RSS split could lead to a subsequent split with a larger reduction in RSS

- **Minimum number of training data in each region** – related to the concept of accuracy of the region mean prediction, i.e., we need at least n data for a reliable mean

- **Maximum number of levels** – forces symmetric trees, similar number of splits to get to each region

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

Lecture outline:

- **Ensemble Methods**

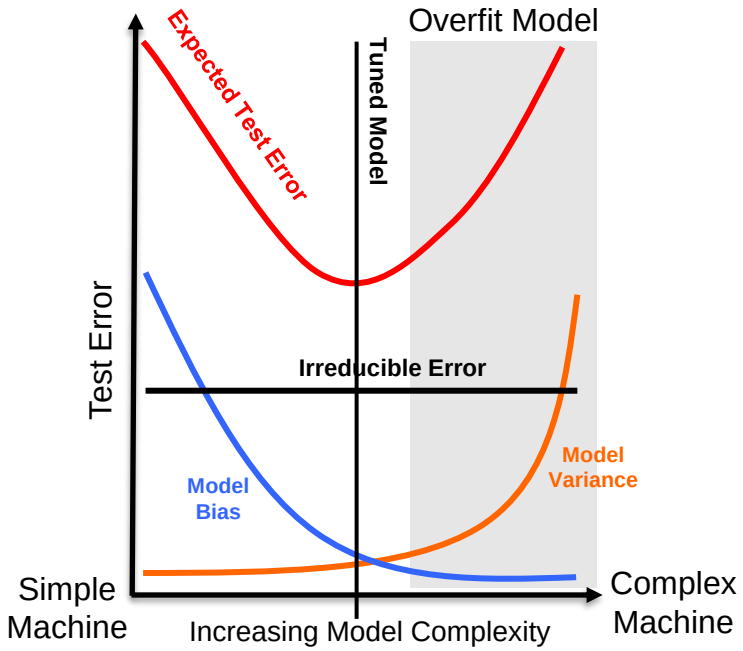
Ensemble Prediction Method

What is the Testing Accuracy of Our Predictive Machine Learning Models?

$$\mathbf{E}\left[\left(y_0 - \hat{f}(x_1^0, \dots, x_m^0)\right)^2\right] = \underbrace{\left(E\left[\hat{f}(x_1^0, \dots, x_m^0)\right] - f(x_1^0, \dots, x_m^0)\right)^2}_{\text{Model Bias}^2} + \underbrace{E\left[\left(\hat{f}(x_1^0, \dots, x_m^0) - E\left[\hat{f}(x_1^0, \dots, x_m^0)\right]\right)^2\right]}_{\text{Model Variance}} + \underbrace{\sigma_e^2}_{\text{Irreducible Error}}$$

- **Model Variance** is the error in the model predictions due to sensitivity to the data (what if we used different training data?)
- **Model Bias** is error in the model predictions due to using an approximate model / model is too simple
- **Irreducible Error** is error in the model predictions due to missing features and limited samples can't be fixed with modeling / entire feature space is not sampled

Model variance limits the complexity and text accuracy of our models!



Model variance and bias trade-off.

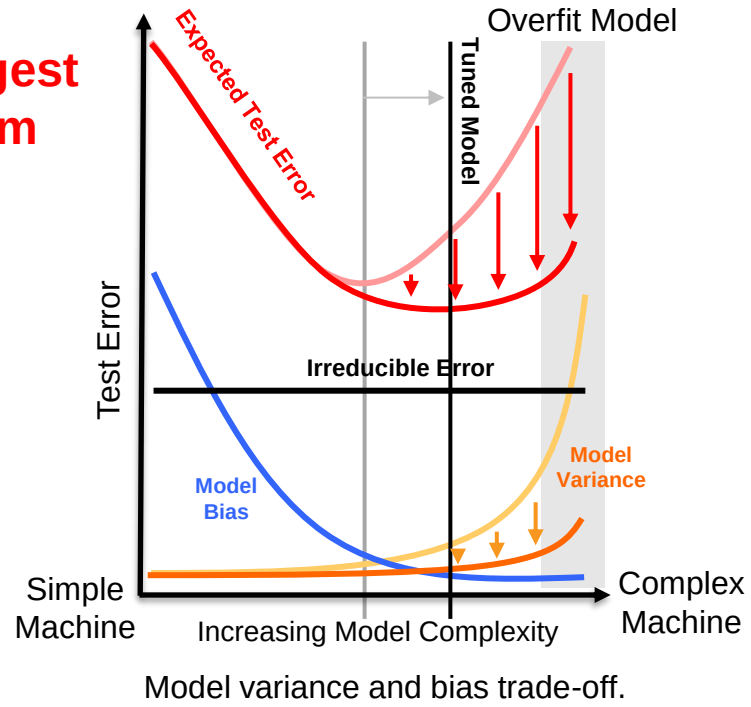
Ensemble Prediction Method

How Can we Improve the Testing Accuracy of Our Predictive Machine Learning Models?

$$\mathbf{E}\left[\left(y_0 - \hat{f}(x_1^0, \dots, x_m^0)\right)^2\right] = \underbrace{\left(E\left[\hat{f}(x_1^0, \dots, x_m^0)\right] - f(x_1^0, \dots, x_m^0)\right)^2}_{\text{Model Bias}^2} + \underbrace{E\left[\left(\hat{f}(x_1^0, \dots, x_m^0) - E\left[\hat{f}(x_1^0, \dots, x_m^0)\right]\right)^2\right]}_{\text{Model Variance}} + \underbrace{\sigma_e^2}_{\text{Irreducible Error}}$$

- **Model Variance** is the error in the model predictions due to sensitivity to the data (what if we used different training data?)

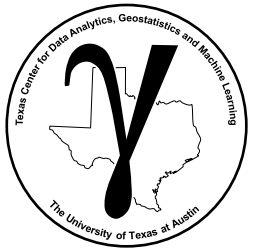
The Biggest Problem



Model variance and bias trade-off.

How Can We Reduce Model Variance?

- So, we can use more complicated and more accurate models!
- By standard error in the average, we observe the reduction in variance by averaging! → $SE_{E\{y\}} = \sigma_{\bar{y}} = \frac{\sigma_y}{\sqrt{n}}$



Ensemble Prediction Method

We Can Reduce Model Variance by Calculating Many Estimates and Averaging them Together!

We will need to make B estimates,

$$\hat{y}^b = \hat{f}^b(X_1, \ldots, X_m), \qquad b = 1, \ldots, B$$

and then our ultimate estimate will be the average (regression) or plurality (classification) of our estimates,

Regression Ensemble Estimate

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^b$$

Classification Ensemble Estimate

$$\hat{y} = arg\ max(\hat{y}^b)$$

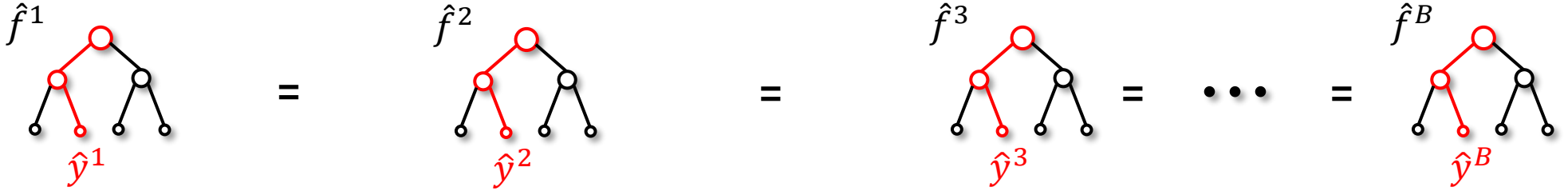
Ensemble Prediction Method

Calculate Multiple Estimates for Our Prediction Problem

This requires multiple prediction models, to make B predictions, $\hat{y}^b = \hat{f}^b(X_1, \ldots, X_m), b = 1, \ldots, B$

$\hat{y}^1 = \hat{f}^1(x_1, \ldots, x_m)$ $\hat{y}^2 = \hat{f}^2(x_1, \ldots, x_m)$ $\hat{y}^3 = \hat{f}^3(x_1, \ldots, x_m)$ $\hat{y}^B = \hat{f}^B(x_1, \ldots, x_m)$

But we only have access to a single dataset, $Y, X_1, \ldots, X_m$; therefore, every model will be the same!



$$\hat{y} = \hat{f}^1(x_1, \ldots, x_m) = \hat{f}^2\ (x_1, \ldots, x_m) = \hat{f}^3\ (x_1, \ldots, x_m) = \ \ldots . = \hat{f}^B\ (x_1, \ldots, x_m)$$

Our models are generally deterministic, train with the same data and hyperparameters and we get the same estimate.

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

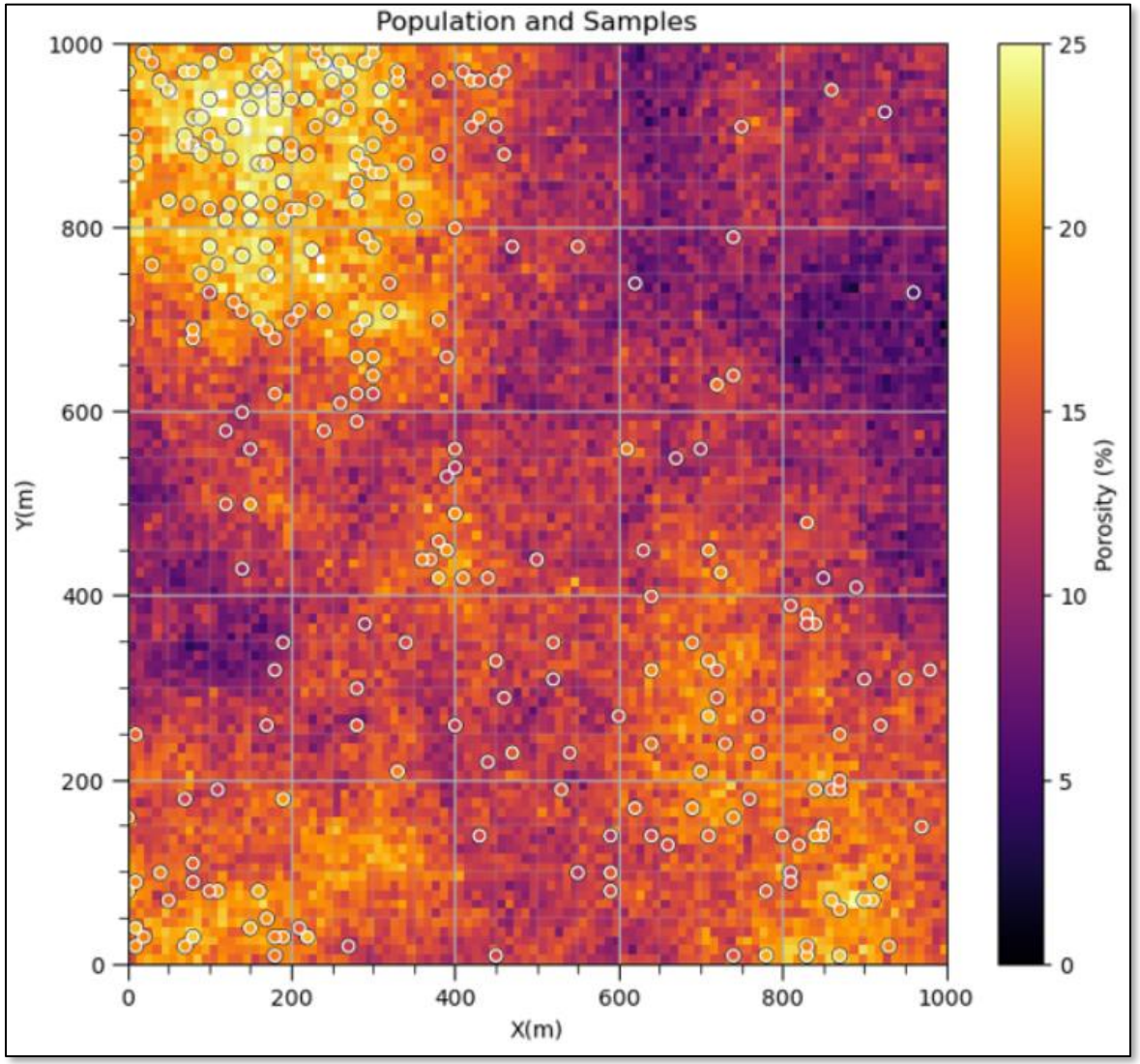
Lecture outline:

- **Bootstrap**

Bootstrap

Uncertainty in the Sample Statistics

- One source of uncertainty is the paucity of data.
- Do these 200 or so wells provide a precise (and accurate estimate) of the mean? standard deviation? skew? P13?
- What is the impact of uncertainty in the mean porosity e.g. 20%+/-2%?



Samples and population, from Bootstrap chapter of Applied Geostatistics in Python e-book.

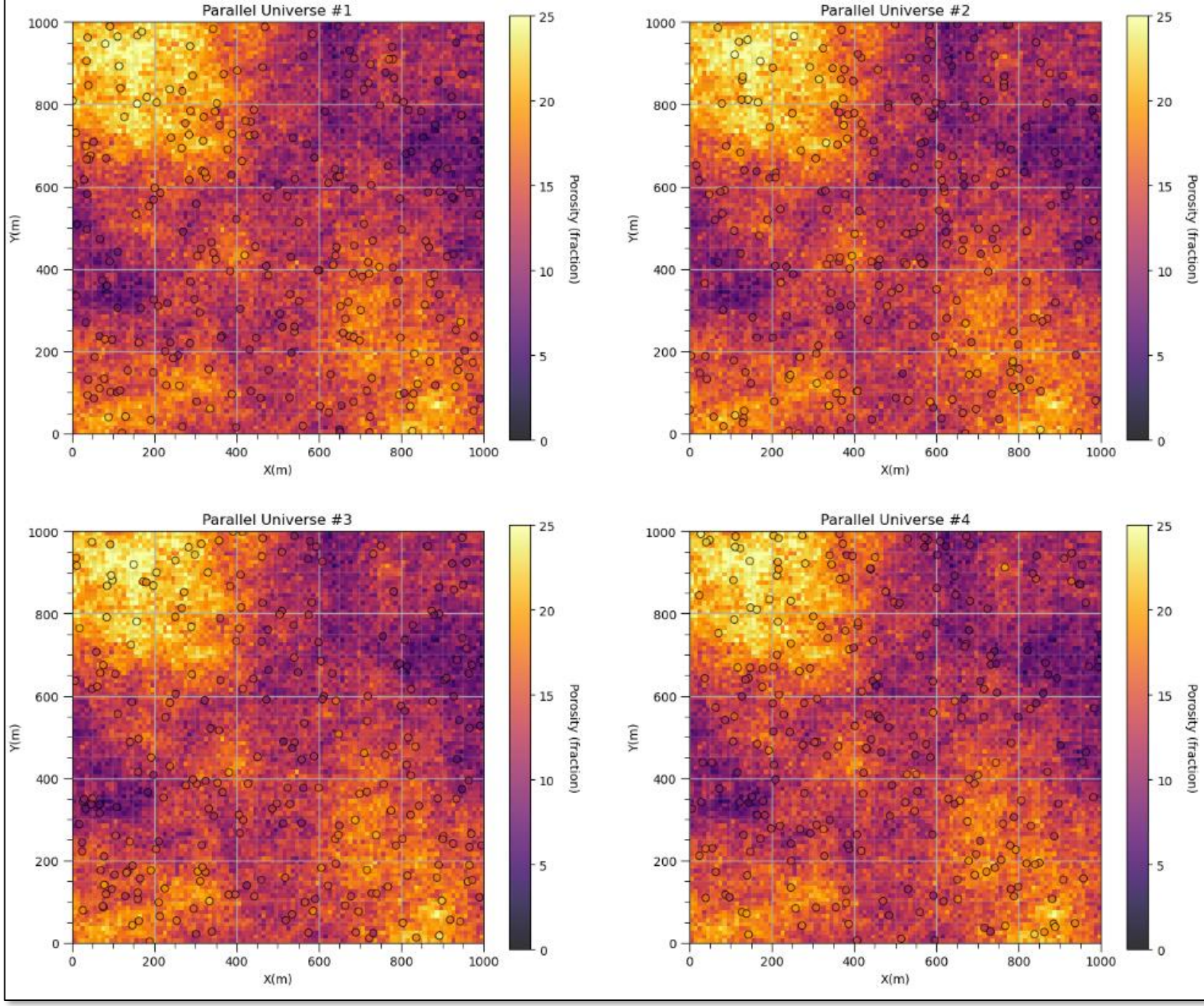
Bootstrap

Uncertainty Due to Data Paucity

What if we had ‘L’ different datasets? L parallel universes where we collected n samples from the inaccessible truth (the population).

We only exist in 1 universe.

- this is not possible.



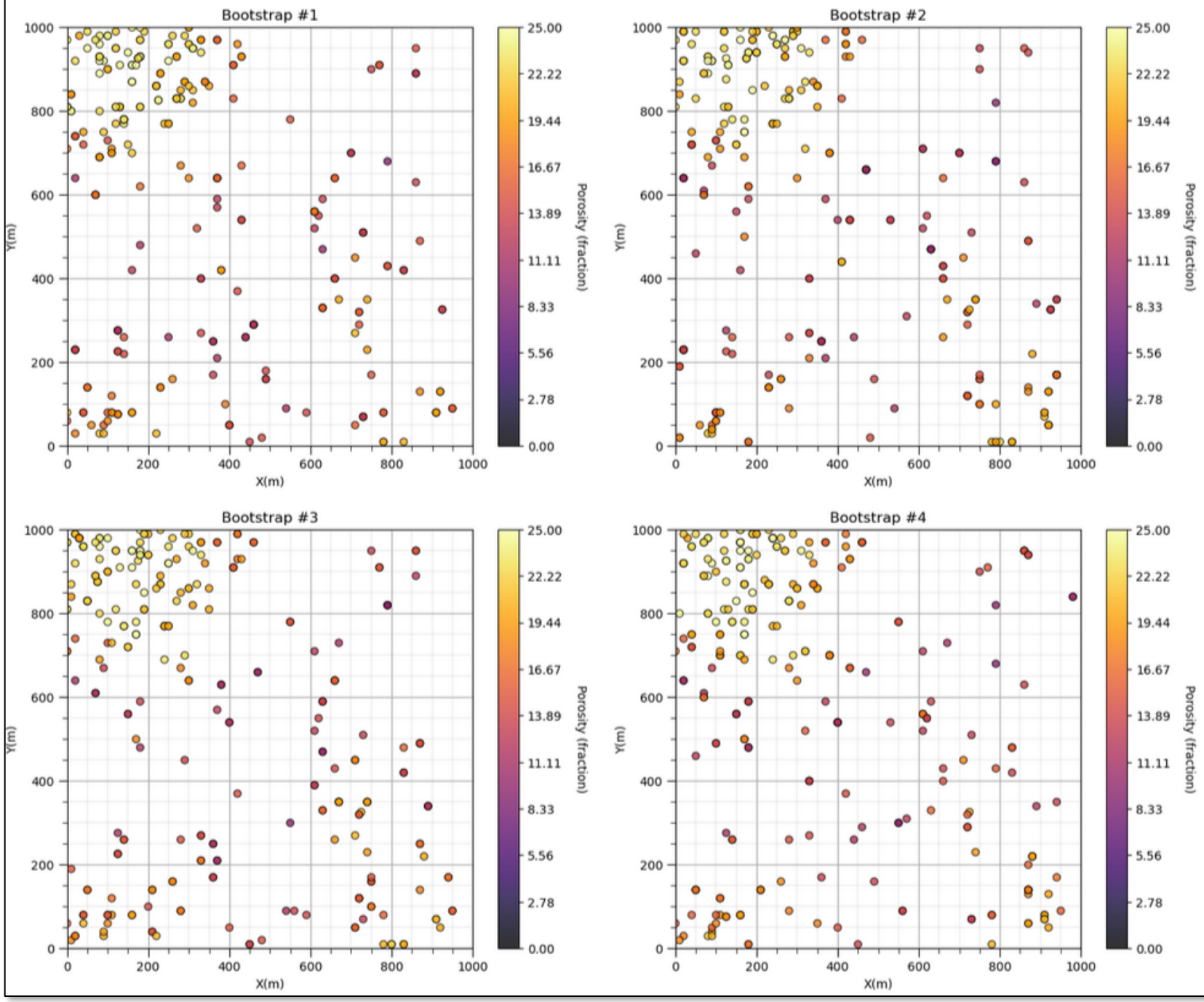
Multiple dataset realizations from the truth population, from Bootstrap chapter of Applied Geostatistics in Python e-book.

Bootstrap

Uncertainty Due to Data Paucity

Instead we sample n times from the dataset with replacement

- bootstrap realizations of the data
- vary by due to some samples being left out and others sampled multiple times.



Multiple dataset bootstrap realizations, from Bootstrap chapter of Applied Geostatistics in Python e-book.

Bootstrap Definition

Bootstrap
- method to assess the uncertainty in a sample statistic by repeated random sampling with replacement
- simulating the sampling process to acquire dataset realizations

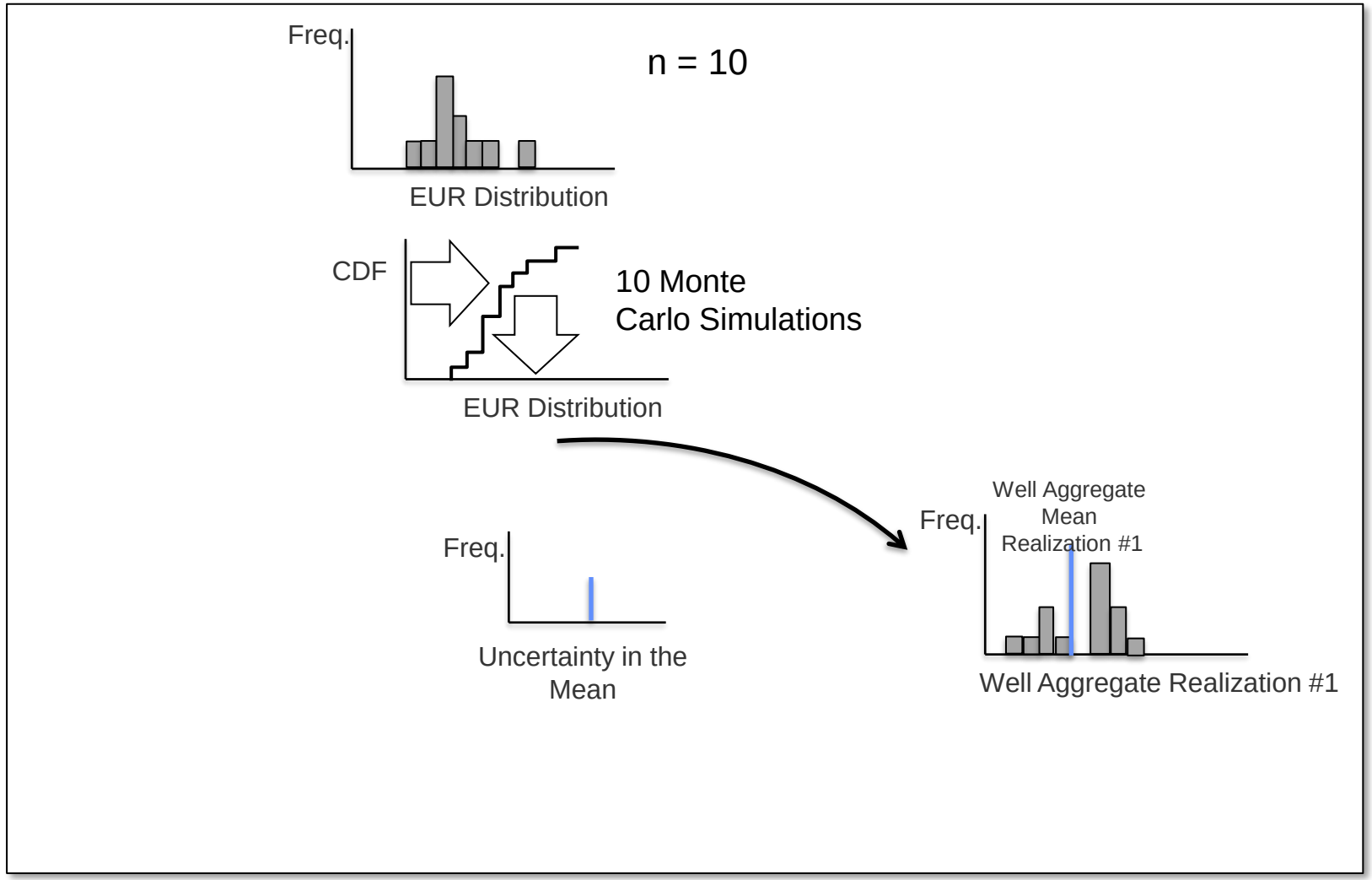
Assumptions
- sufficient, representative sampling

Limitations
- assumes the samples are representative
- assumes stationarity
- only accounts for uncertainty due to too few samples, e.g., no uncertainty due to changes away from data
- does not account for area of interest
- assumes the samples are independent
- does not account for other local information sources

No spatial context

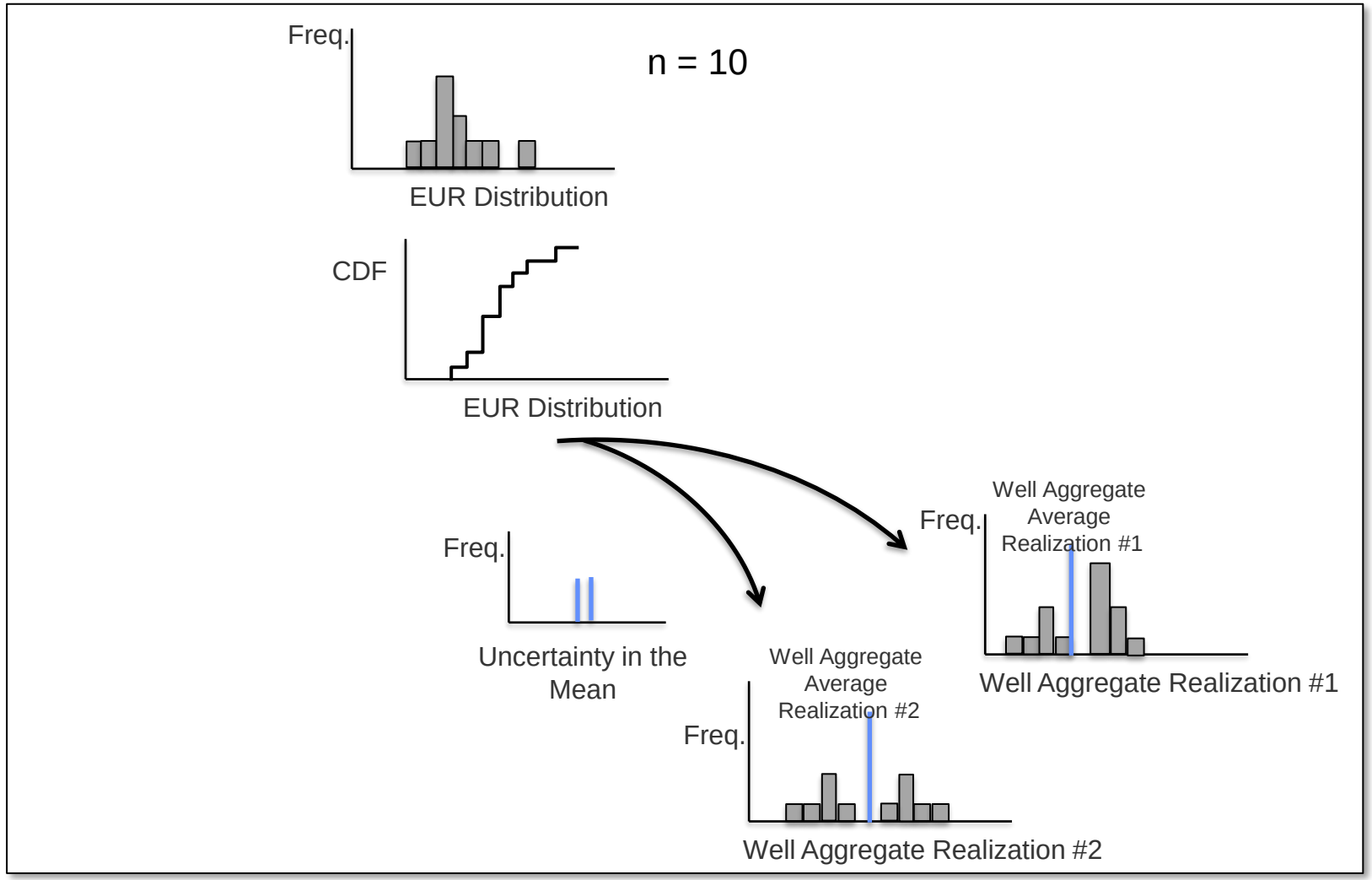
Bootstrap

Bootstrap for Uncertainty in EUR Mean



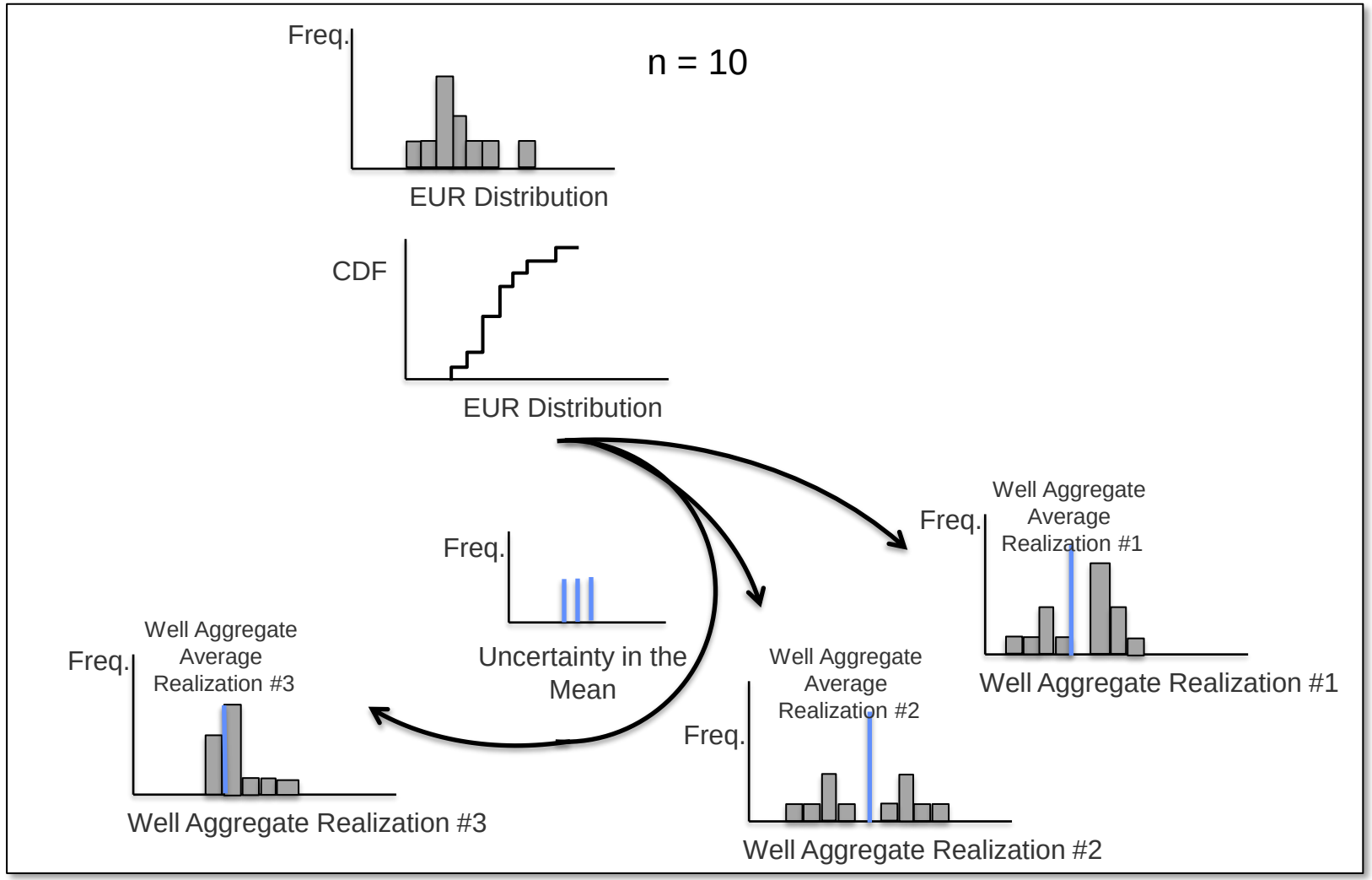
Bootstrap

Bootstrap for Uncertainty in EUR Mean



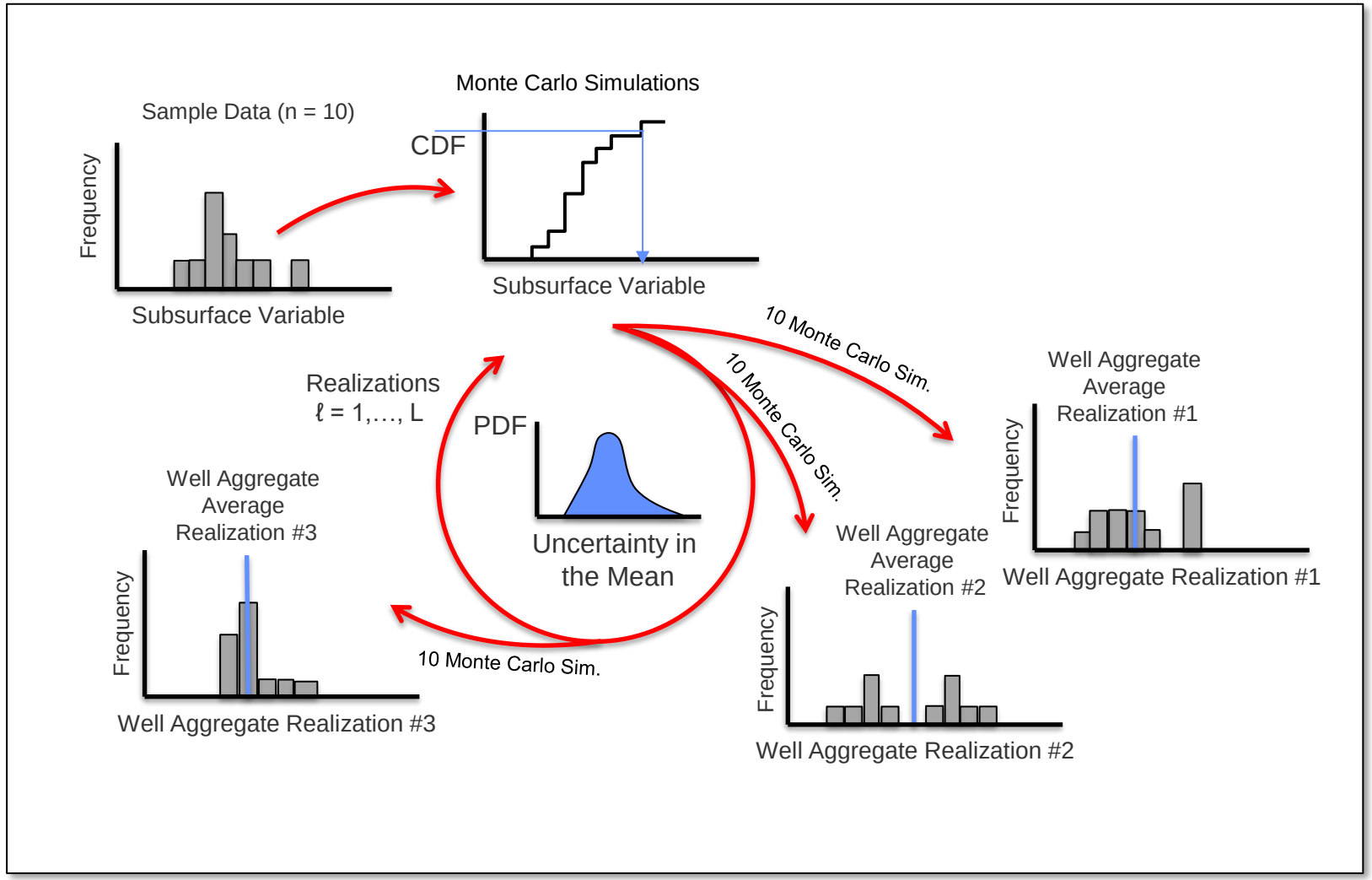
Bootstrap

Bootstrap for Uncertainty in EUR Mean



Bootstrap

Bootstrap for Uncertainty in EUR Mean



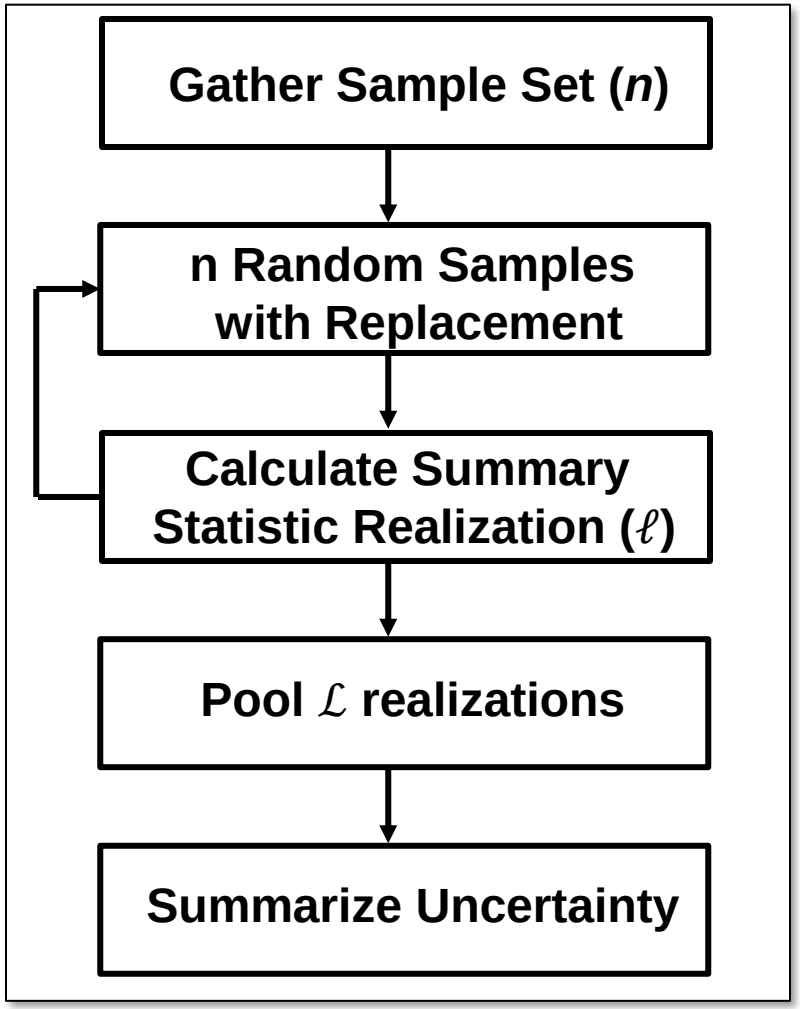
Bootstrap

Bootstrap Approach developed by Efron (1982)

- Statistical resampling procedure to calculate uncertainty in a calculated statistic from the data itself.
- For uncertainty in the mean solution is standard error:

$$\sigma_{\bar{x}}^2 = \frac{\sigma_s^2}{n}$$

- Extremely powerful. Could get uncertainty in any statistic! e.g., P13, skew etc.
- Would not be possible without bootstrap.
- Advanced forms account for spatial information and strategy (game theory).



Bootstrap workflow for uncertaint in statistic.

Bootstrap

You now know about one of the most powerful statistical tools!

- Caveats:
  - assumes the sample set is representative
  - unbiased and covers the full range
  - assumes all samples are independent if not consider Journel's spatial bootstrap (1993).
- You can even do bootstrap in Excel.

Boostrap Demonstration

Michael Pyrcz, the University of Texas at Austin, @GeostatsGuy

Method: Bootstrap uncertainty in distribution statistics by resampling with replacement from the initial sample dataset.

Details: Resample with replacement n times to build a realization. Repeat for L=1,000 realizations. Calculate various sum

Observations: Uncertainty will decrease as number of samples (n) increases. Bootstrap assumes no spatial continuity bet

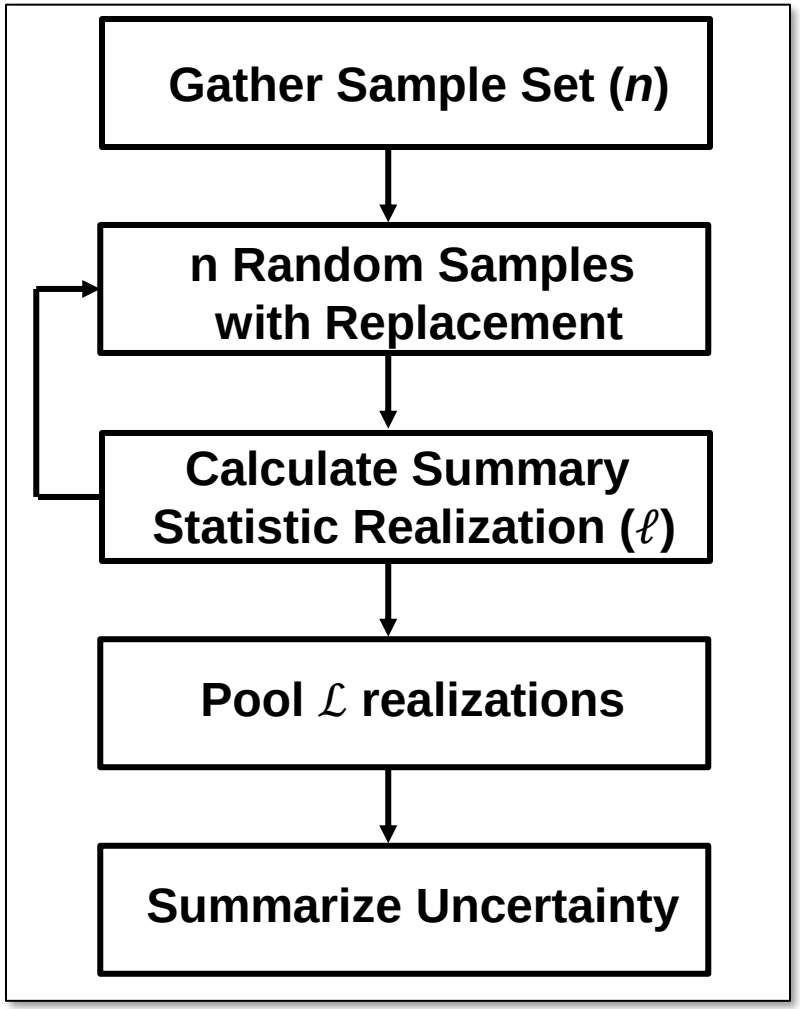
Sample Data Set

Index	P-value	Norm[0,1]
1	0.586	0.218
2	0.741	0.647
19	0.851	1.040
20	0.219	-0.776

Samples	Real1	Real2	Real3	Real4	Real5	Real6	Real7	Real8	Real9	Real10	Real11	Real12	Real13	Real14	Real15	Real16	Real17
1	-0.642	0.784	0.218	-0.023	-0.776	1.040	1.967	-0.025	-0.025	-0.570	1.040	-1.155	1.973	-1.652	-0.230	0.890	0.218
2	0.218	1.040	-1.652	0.890	-0.025	0.784	-1.155	-0.709	-1.155	0.167	-1.652	-0.023	-0.023	-0.776	0.890	-1.155	-0.025
19	0.890	-1.155	0.473	0.647	0.167	0.508	0.218	0.890	-0.776	0.473	0.473	-0.023	1.040	0.784	0.473	-0.709	-1.652
20	0.473	-0.025	0.218	0.890	-0.230	0.647	-0.025	0.647	1.967	0.167	0.708	0.708	1.040	0.218	-1.155	-1.155	-0.709

Statisti	Real1	Real2	Real3	Real4	Real5	Real6	Real7	Real8	Real9	Real10	Real11	Real12	Real13	Real14	Real15	Real16	Real17
Mean	0.30	0.45	0.37	0.16	0.25	0.10	0.03	0.13	-0.11	0.10	0.10	0.11	0.30	-0.12	-0.07	0.06	0.05
St.Dev.	0.99	0.95	0.71	0.96	0.87	0.93	0.78	0.84	1.09	0.52	0.82	0.83	0.95	0.87	0.87	0.85	1.06
Min	-1.65	-1.16	-1.65	-1.65	-1.65	-1.65	-1.16	-1.16	-1.65	-0.78	-1.65	-1.65	-1.65	-1.65	-1.65	-1.16	-1.65
Max	1.97	1.97	1.97	1.97	1.97	1.04	1.97	1.97	1.97	1.04	1.04	1.04	1.97	1.04	1.97	1.97	1.97
P10	-1.20	-0.72	-0.26	-0.81	-0.66	-1.20	-0.81	-0.72	-1.20	-0.64	-1.16	-1.20	-0.81	-1.65	-1.16	-1.16	-1.65
P50	0.49	0.58	0.47	0.07	0.32	0.51	0.10	-0.02	-0.13	0.17	0.35	0.32	0.58	0.22	-0.02	0.22	0.10
P90	1.13	1.97	0.90	1.00	1.00	1.04	0.73	1.04	1.97	0.65	0.90	0.90	1.13	0.78	0.79	0.90	1.13

Bootstrap in Excel with random and DataBase functions, Bootstrap_Demo.xlsx.

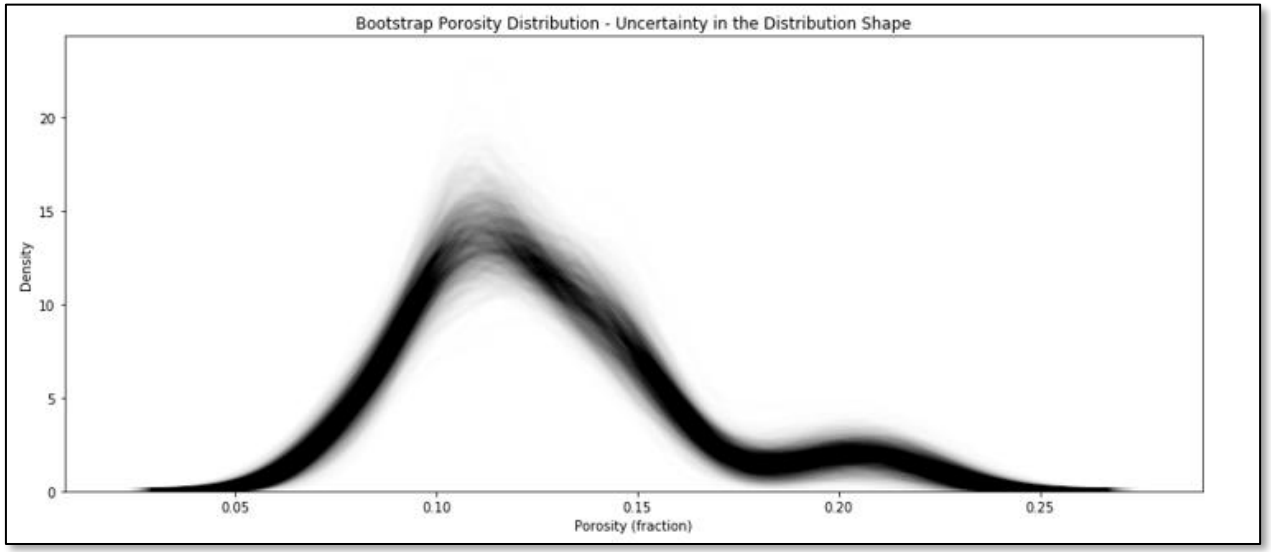


Bootstrap workflow for uncertaint in statistic.

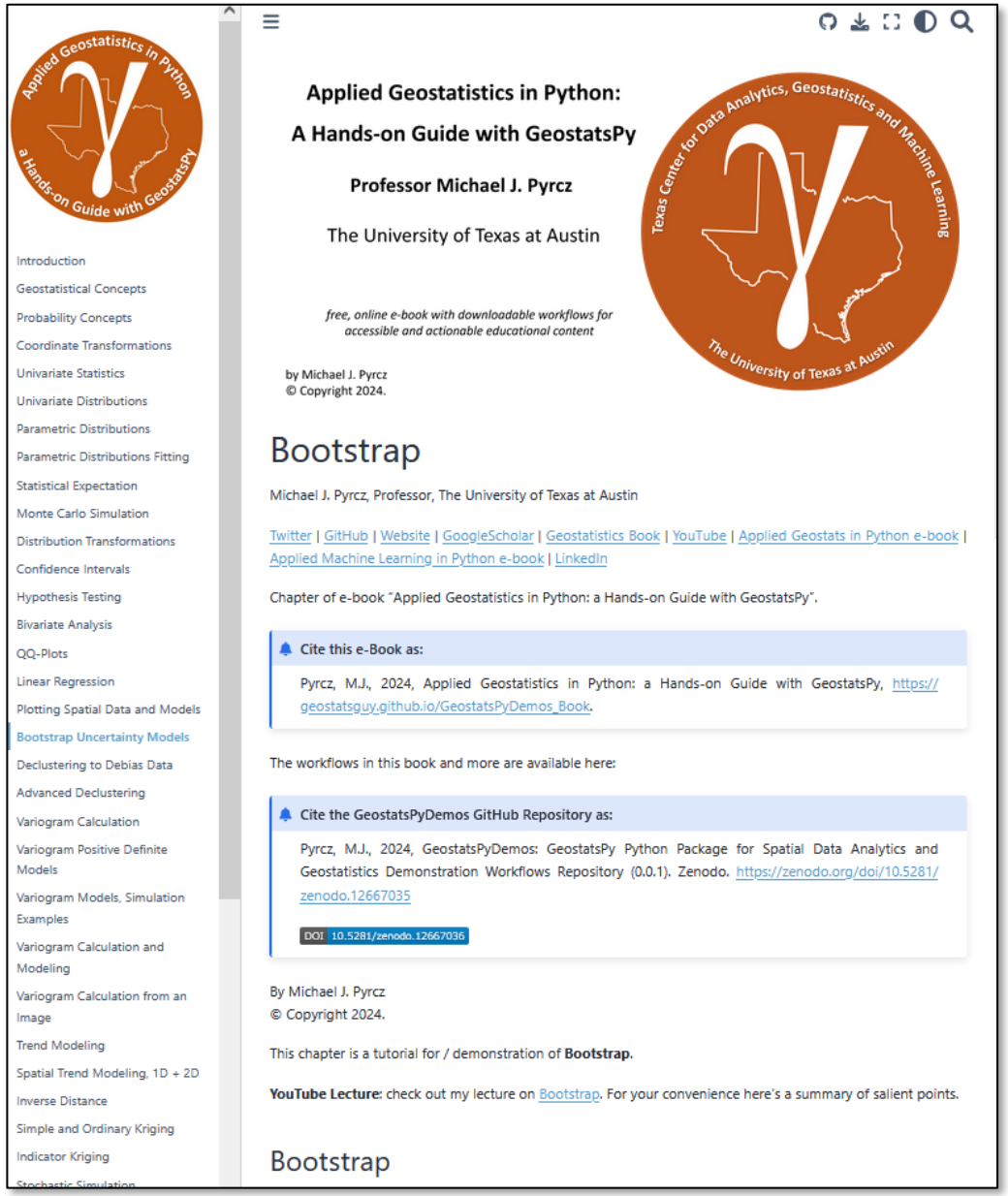
Bootstrap Demonstration in Python

Demonstration workflow for boostrap for uncertainty in statistics and models,

- a variety of example statistics



Bootstrap for uncertainty in statistics, file is SubsurfaceDataAnalytics_bootstrap.ipynb.



Bootstrap Uncertainty Model chapter of Applied Geostatistics in Python.

More on Bootstrap

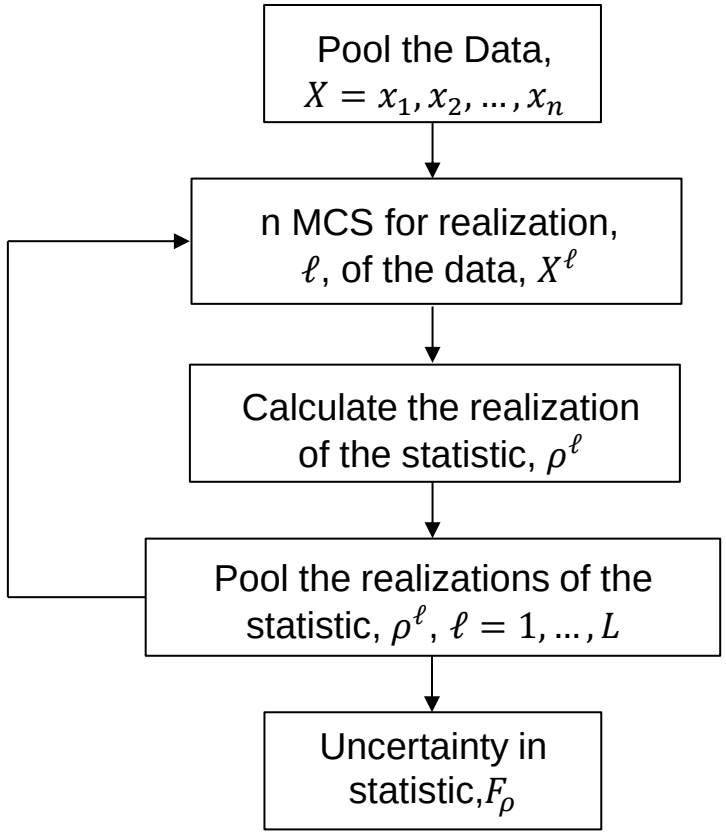
Let me reinforce, the bootstrap approach may be applied to calculate uncertainty in any statistic, from new realizations of the data.

- We can even bootstrap ML models, known as bagging. More later.

Notation,

- Note, $P_k(0.5)^\ell$ is the 50th percentile of permeability, k, bootstrap data realization, ℓ.

- and ${s_1^2}^\ell$ is the sample variance of the 1st bootstrap dataset's ℓ realization.



ℓ, realization of the data distribution(s)

50th Percentile $P_k(50)^\ell$

Dykstra Parsons $dp^\ell = \frac{P_k(50)^\ell - P_k(16)^\ell}{P_k(50)^\ell}$

Welch's t-statistic

2 paired distributions $\hat{t}^\ell = \frac{{\bar{x}_1}^\ell - {\bar{x}_2}^\ell}{\sqrt{\left(\frac{{s_1^2}^\ell}{n_1} + \frac{{s_2^2}^\ell}{n_2}\right)}}$

$P_k(50)^\ell$, dp^ℓ, $\hat{t}^\ell$ realization of the statistic

Bootstrap for Ensemble Methods

Machine Learning Bagging

1. Apply statistical bootstrap to obtain multiple realizations of the data,

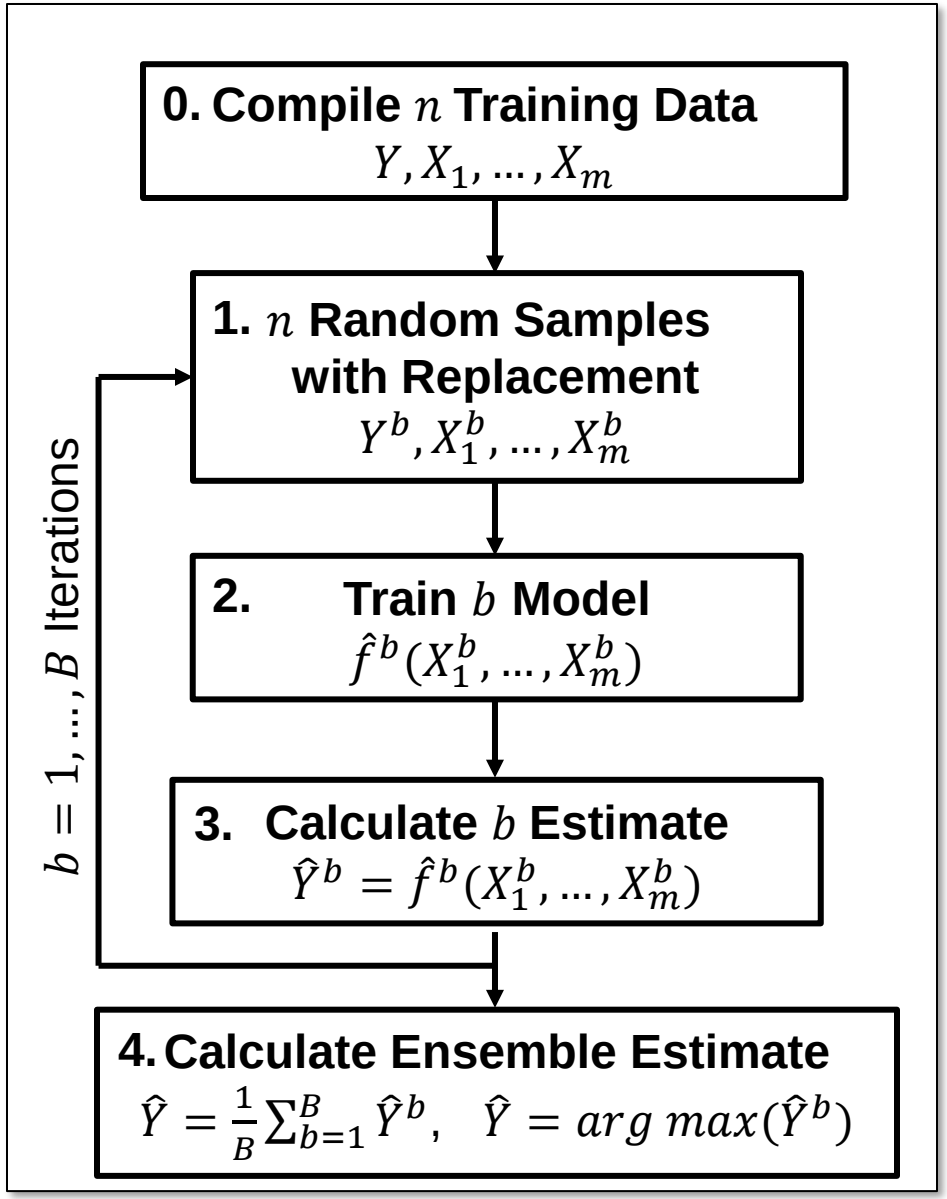
$$Y^b, X_1^b, \ldots, X_m^b, b = 1, \ldots, B$$

2. Train a prediction model (estimator) for each data realization,

$$\hat{f}^b(X_1^b, \ldots, X_m^b)$$

3. Calculate a prediction with each estimator,

$$\hat{Y}^b = \hat{f}^b(X_1^b, \ldots, X_m^b)$$



The bagging machine learning workflow.

Bootstrap for Ensemble Methods

Machine Learning Bagging

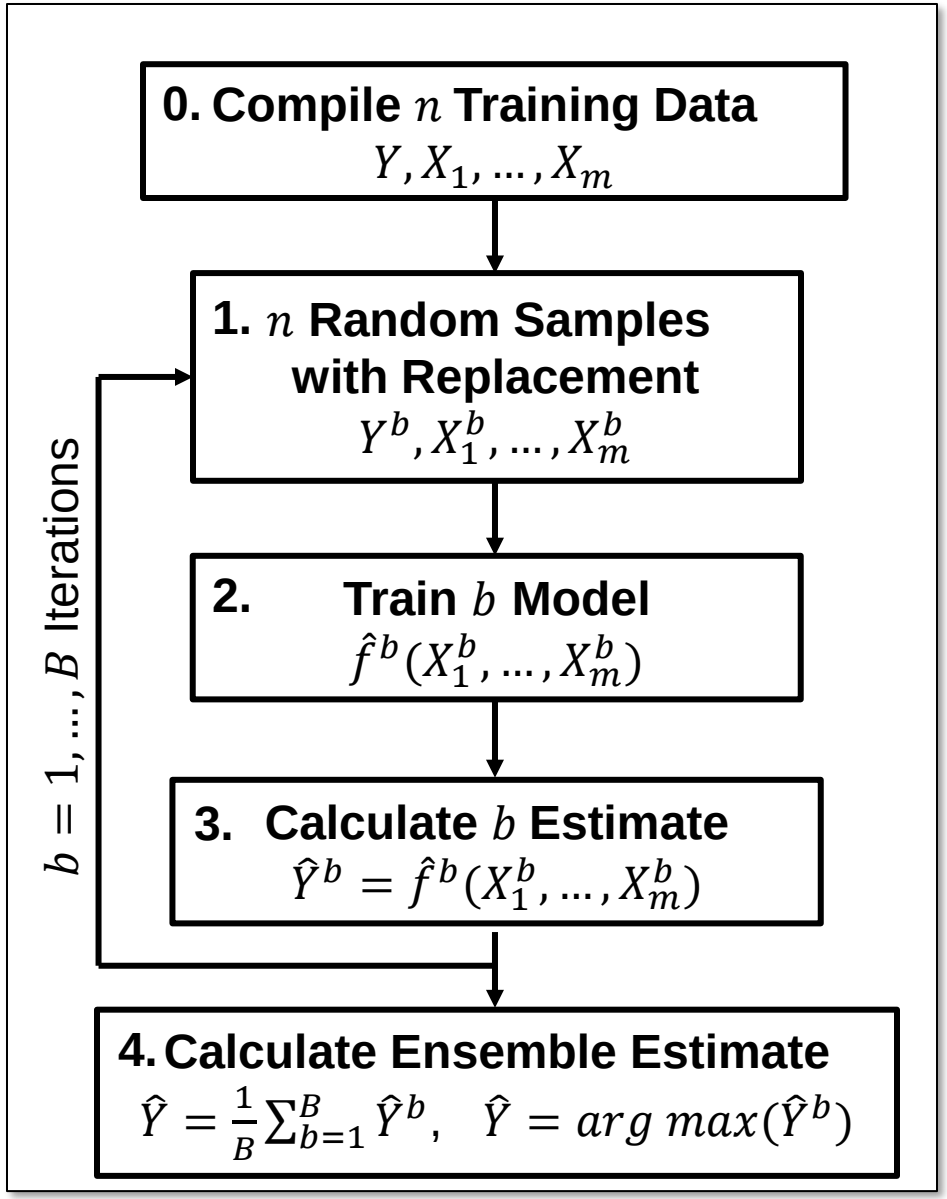
4. Aggregate the ensemble of B predictions over the estimators:

- Regression – aggregate the ensemble predictions with the average,

$$\hat{Y} = \frac{1}{B}\sum_{b=1}^{B}\hat{Y}^b$$

- Classification – aggregate the ensemble predictions with majority-rule,

$$\hat{Y} = arg\ max(\hat{Y}^b)$$

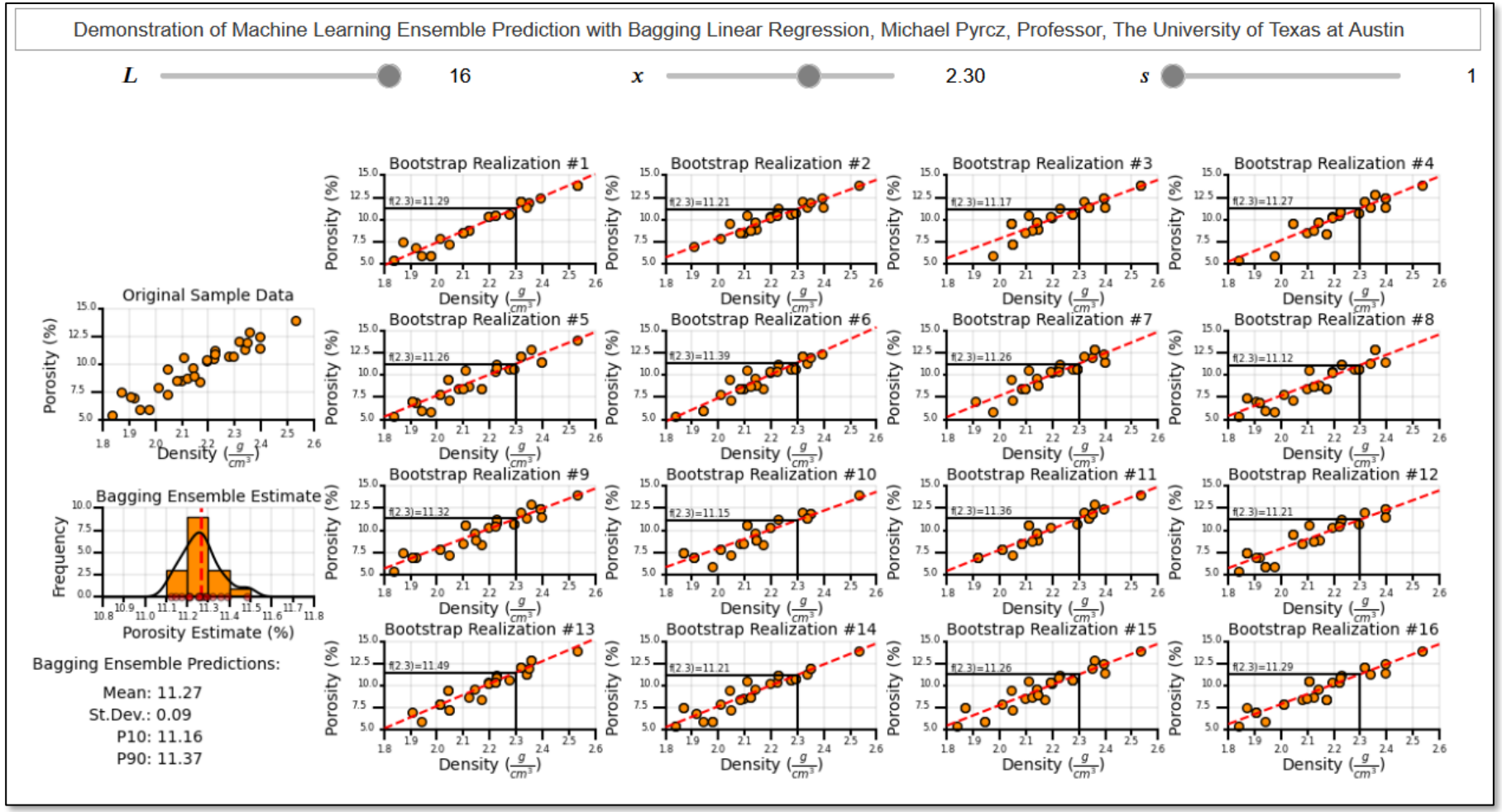


The bagging machine learning workflow.

Bootstrap for Ensemble Methods

Machine Learning Bagging

- you can bootstrap any statistic and,
- you can apply bagging to any machine learning model
- here's linear regression bagging



Interactive machine learning bagging with linear regression, 16 data bootstrap, model and prediction realizations aggregated by averaging, file is Interactive_Bootstrap_Bagging.ipynb .

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

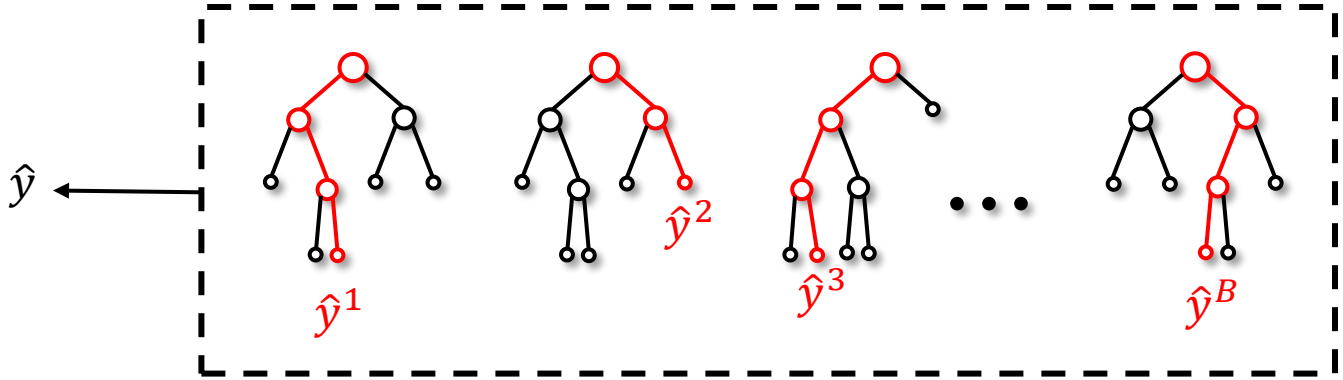
Lecture outline:

- **Training and Tuning Bagging Models**

Bagging Ensemble Models

What is the Bagging Regression Model?

Multiple models each trained on different bootstrap data realizations, all with the same hyperparameter(s).



The bagging model, we train individually, but we tune the ensemble.

The bagging prediction, $\hat{y}$, the aggregate of the individual estimators, is the output of this model.

the model

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B}\left[\hat{f}^1 + \hat{f}^2 + \hat{f}^3 + \cdots + \hat{f}^B\right]$$

the model

$$\hat{y} = \underset{max}{arg}\left[\hat{f}^1, \hat{f}^2, \hat{f}^3, \cdots, \hat{f}^B\right]$$

Bagging regression predictions by averaging or plurality of multiple prediction models.

Bagging Ensemble Models

Training the Estimators within the Bagging Ensemble

Each model is trained with their respective bootstrapped data realization,

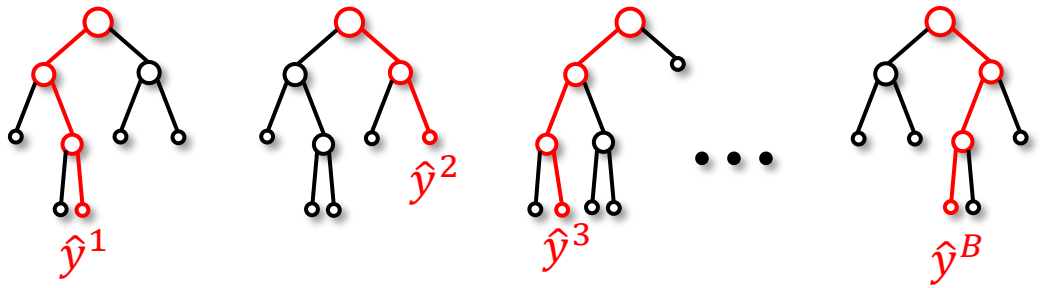
- **during training each model minimizes error of the individual estimator** with the bootstrapped data realization.

$$MSE = \sum_{i=1}^{n} \left(\hat{y}_i^b - y_i \right)^2$$

- **each estimator is trained separately**, but they all share the same hyperparameters.

This provides the flexibility to build the best possible model to fit each bootstrap dataset.

Estimators in the ensemble of models are trained individually, but share the same hyperparameter(s).



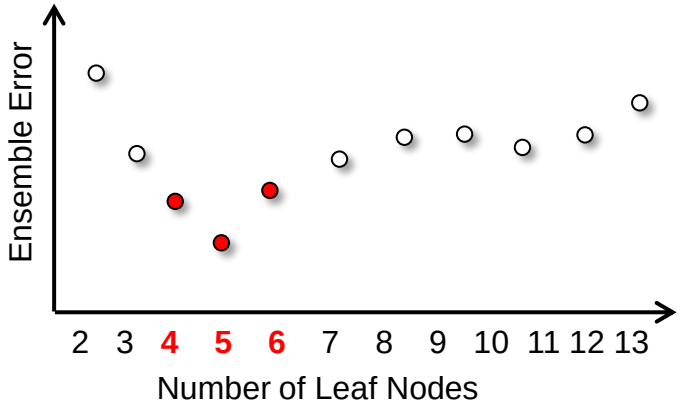
Bagging Ensemble Models

Tuning Bagged Models

We tune our bagged model with the error of the bagging estimate from aggregating the ensemble of estimates, e.g.,

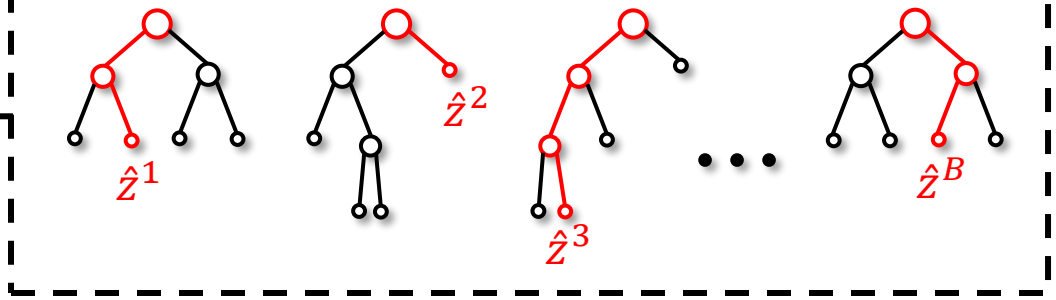
$$MSE = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2, \quad \hat{y}_i = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_i^b$$

- We do not consider the error of individual model estimators, $\hat{y}_i^b - y_i$, within the ensemble for model tuning.



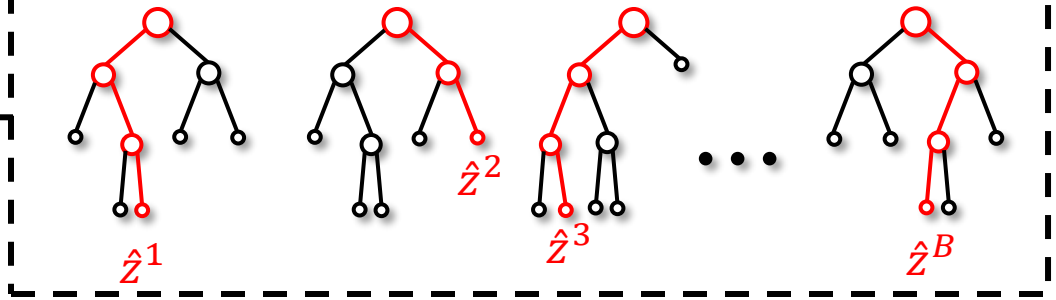
Hyperparameter – Number of Leaf Nodes = 4

$(\hat{y}_i - y_i)^2$



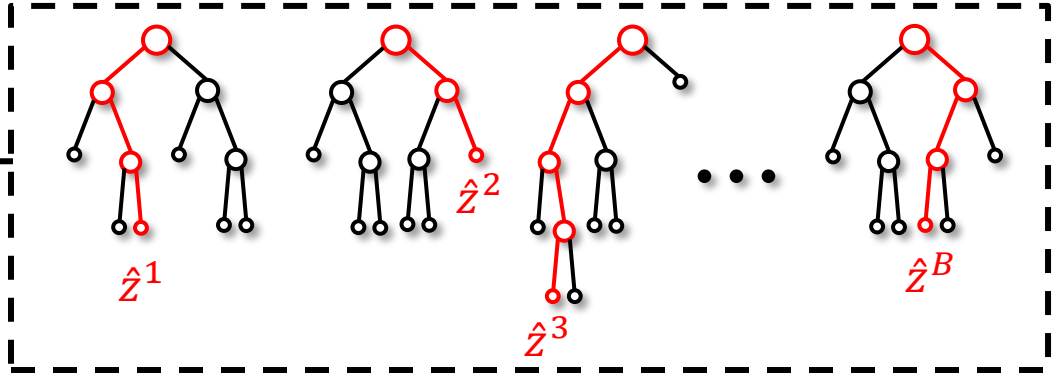
Hyperparameter – Number of Leaf Nodes = 5

$(\hat{y}_i - y_i)^2$



Hyperparameter – Number of Leaf Nodes = 6

$(\hat{y}_i - y_i)^2$

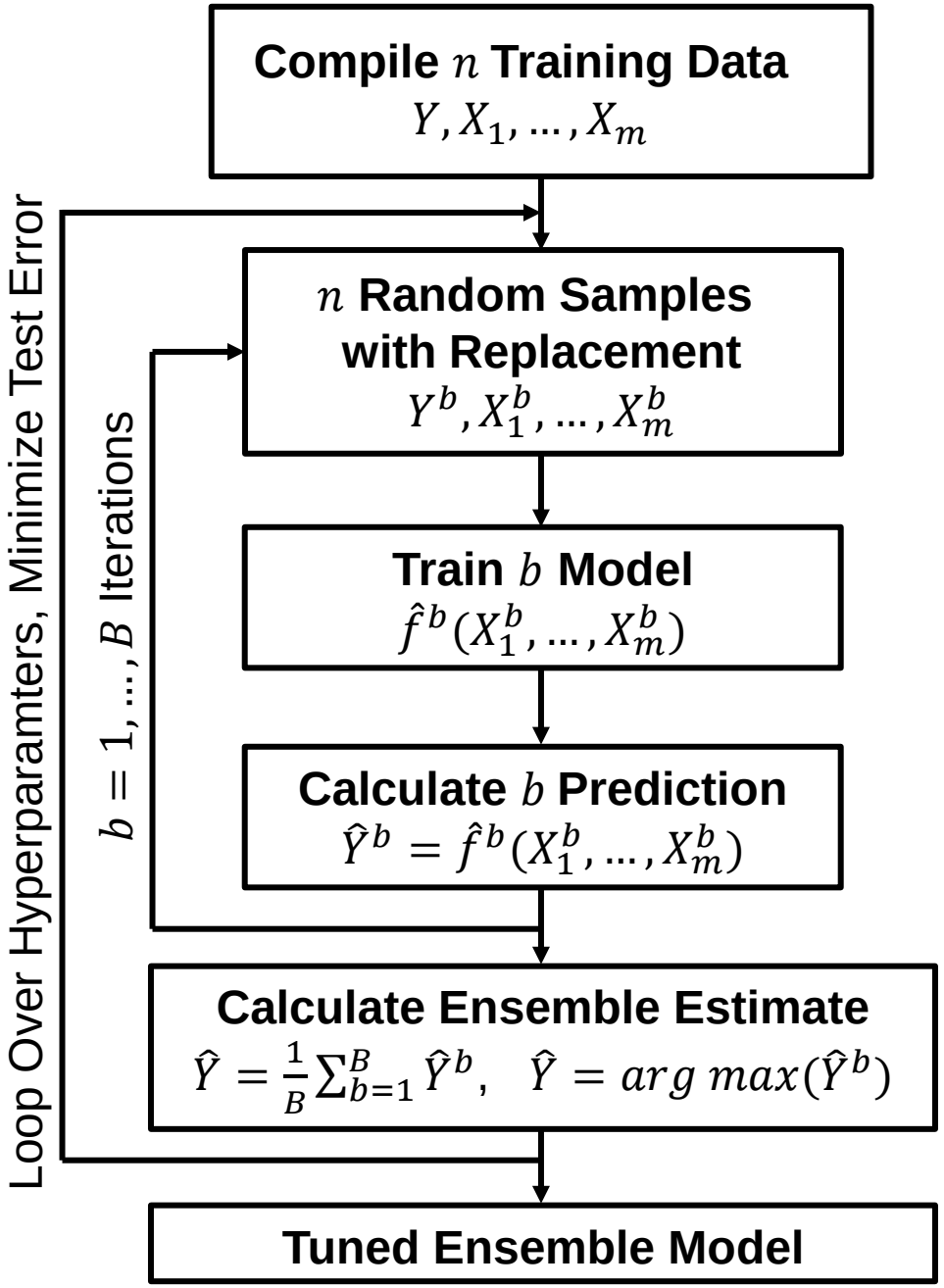


Bagging Ensemble Models

Tuning Bagged Models

For clarity, let's add the tunning to our previous training bagging models workflow

- we loop over hyperparameters
- minimize the test error of the ensemble estimates



The workflow for tuning a bagging model.

Out-of-Bag Cross Validation Hyperparameter Tuning

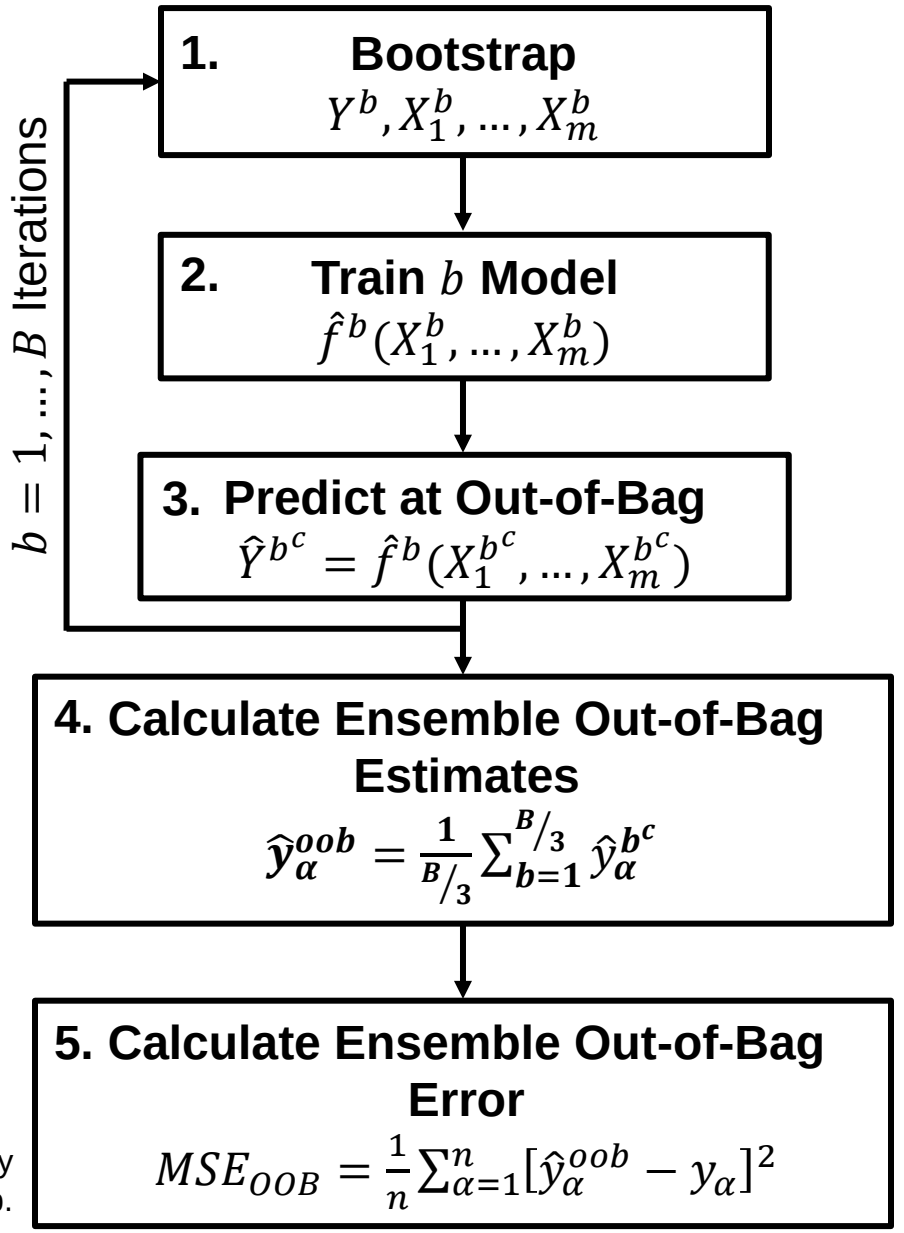
No Need for Train and Test Split

In expectation $\frac{1}{3}$ of the training data is left out of each bootstrap data realization, b^c; therefore, cross validation is built in.

1. Sample with replacement $\frac{2}{3}$ of the training data (in expectation), $Y^b, X_1^b, \ldots, X_m^b$

2. Train an estimator with the $\frac{2}{3}$ of training data (in expectation)

3. Predict at the out-of-bag samples, $X_1^{b^c}, \ldots, X_m^{b^c}$, $\frac{1}{3}$ of the training data (in expectation)



Out-of-bag error calculation workflow, to apply add to the hyperparameter tuning loop.

Out-of-Bag Cross Validation Hyperparameter Tuning

No Need for Train and Test Split

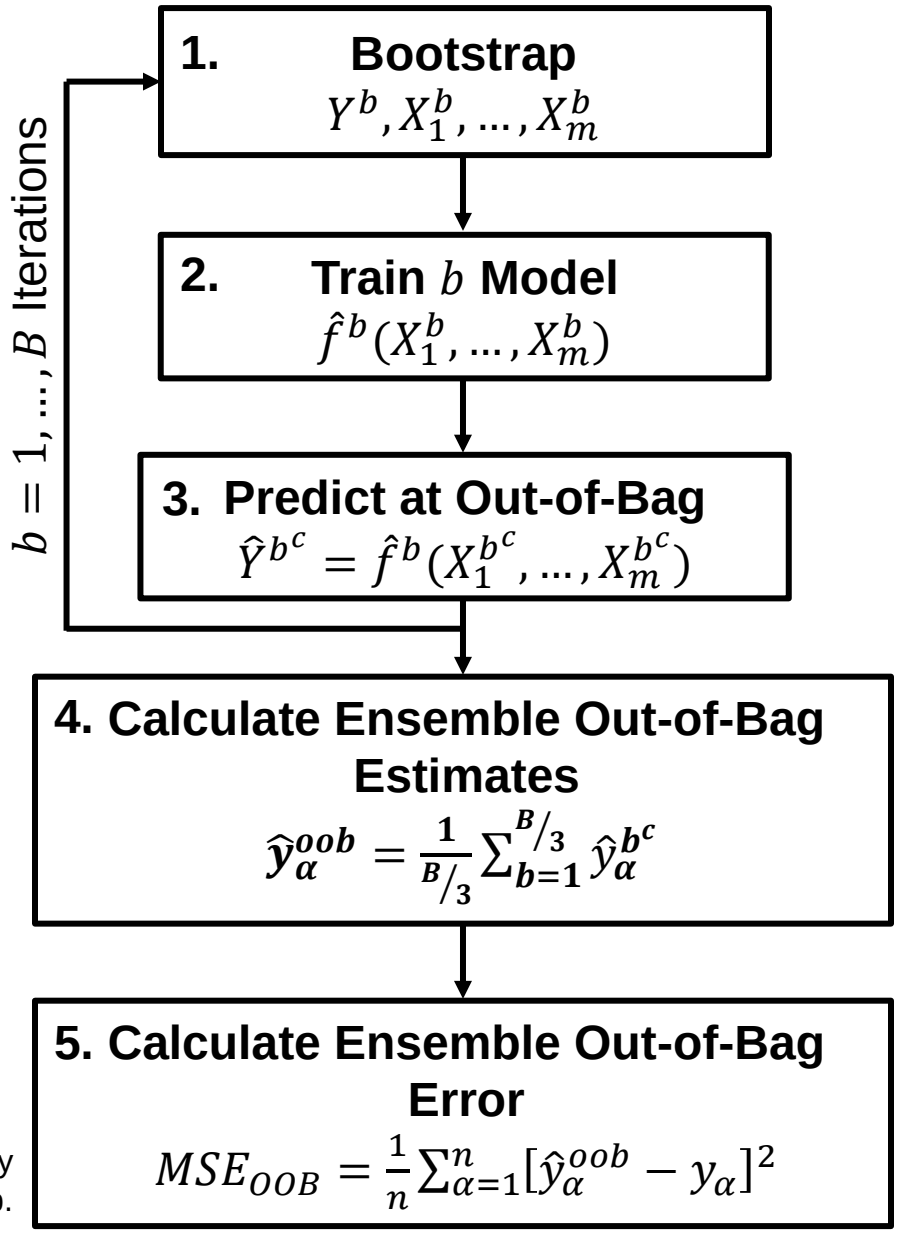
In expectation $\frac{1}{3}$ of the training data is left out of each bootstrap data realization, b^c; therefore, cross validation is built in.

4. Pool the $B/3$ predictions (in expectation) for each sample data from all the B models and make an out-of-bag prediction

$$\hat{y}_\alpha^{oob} = \frac{1}{B/3} \sum_{b=1}^{B/3} \hat{y}_\alpha^{b^c}$$

5. Calculate the out-of-bag error to access model performance.

$$MSE_{OOB} = \frac{1}{n} \sum_{\alpha=1}^{n} [\hat{y}_\alpha^{oob} - y_\alpha]^2$$

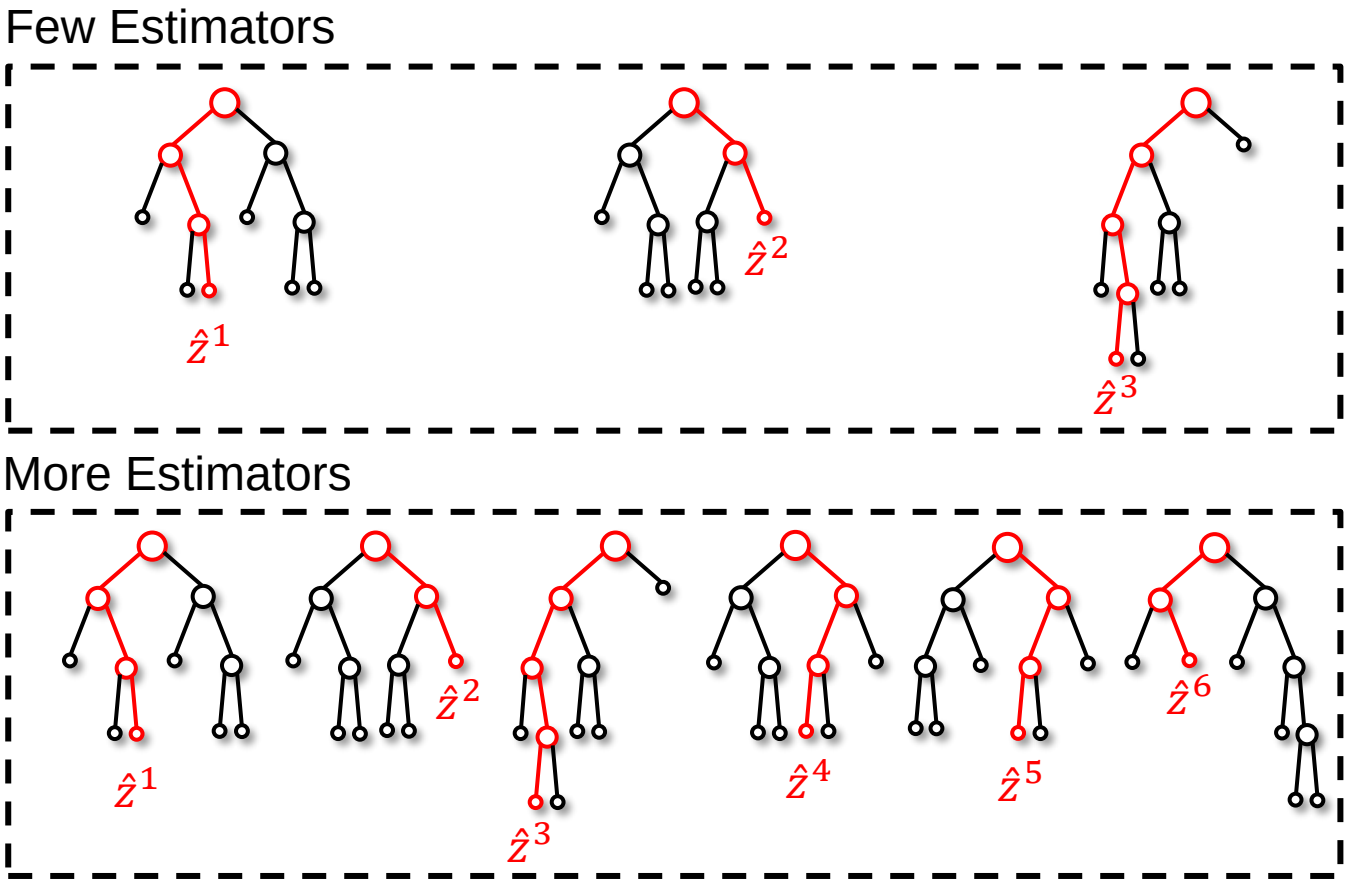


Out-of-bag error calculation workflow, to apply add to the hyperparameter tuning loop.

Number of Estimators

Number of Estimators is an Important Hyperparameter for Bagging Models

- **More estimators** – improve generalization up to a point, increasing the number of trees generally improves performance and reduces variance, as predictions are averaged across more models.
- **Diminishing returns** - beyond a point, adding more estimators gives little or no improvement and only increases computational cost.
- **Improved Stability** - more trees reduce the likelihood of overfitting to random noise in the training set.

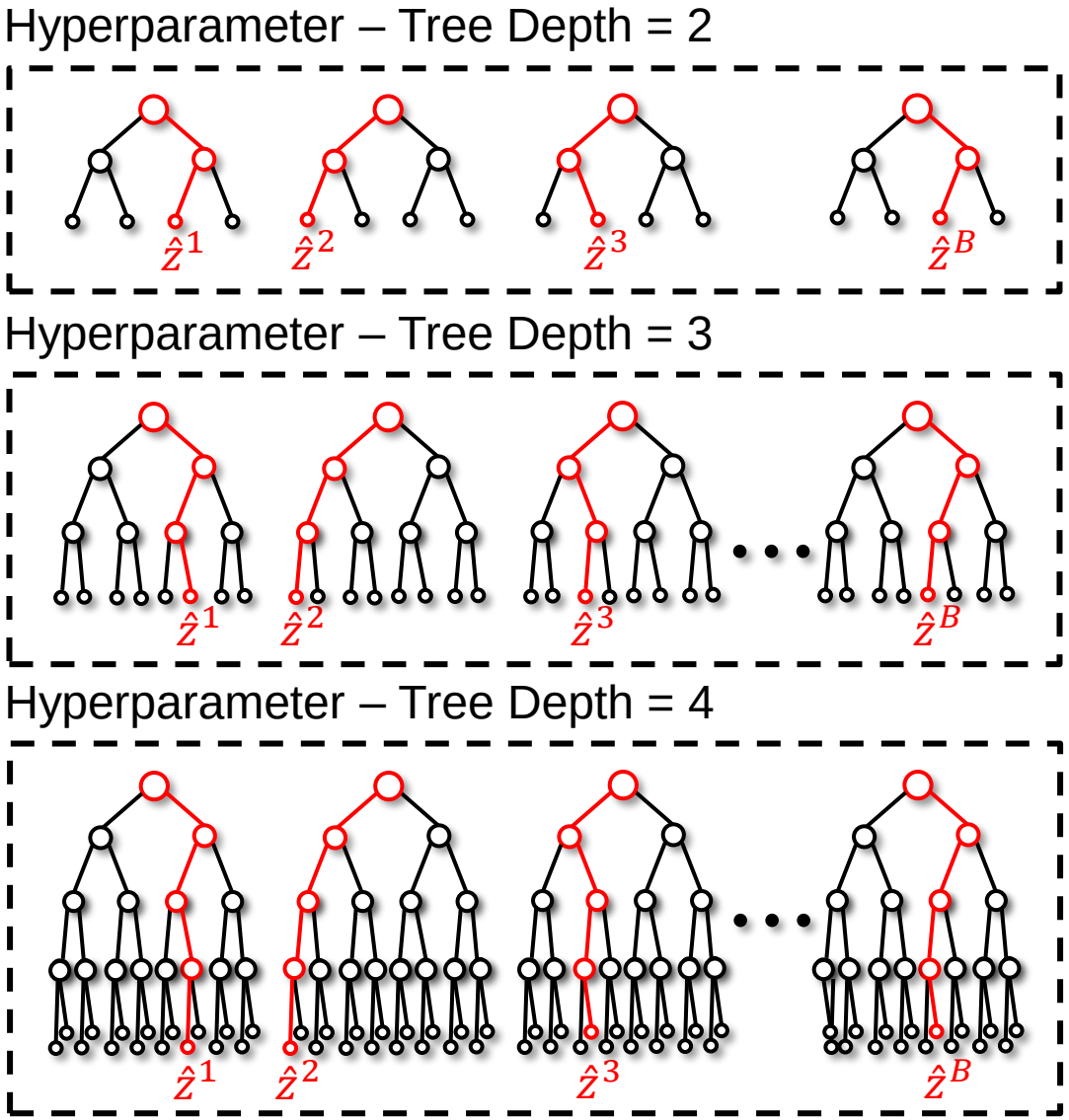


Number of estimators, few (upper) and more (lower), within the ensemble model.

Tree Complexity

Estimator Complexity Hyperparameters for Bagging Models

- **More complicated models** – bagging reduces model variance, so we often train more complicated models for the ensemble.
- **Too simple models** – may not see any improvement from bagging as model variance is not an issue.
- **Feature interactions** – more complicated models capture more of the interactions between features, e.g., tree bagging models with tree depth d can capture d-way feature interactions



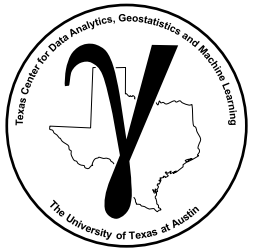
Ensembles with different tree depth hyperarmeters, 2 (upper), 3 (middle) and 4 (lower).

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

Lecture outline:

- **Tree Bagging**



Tree Bagging

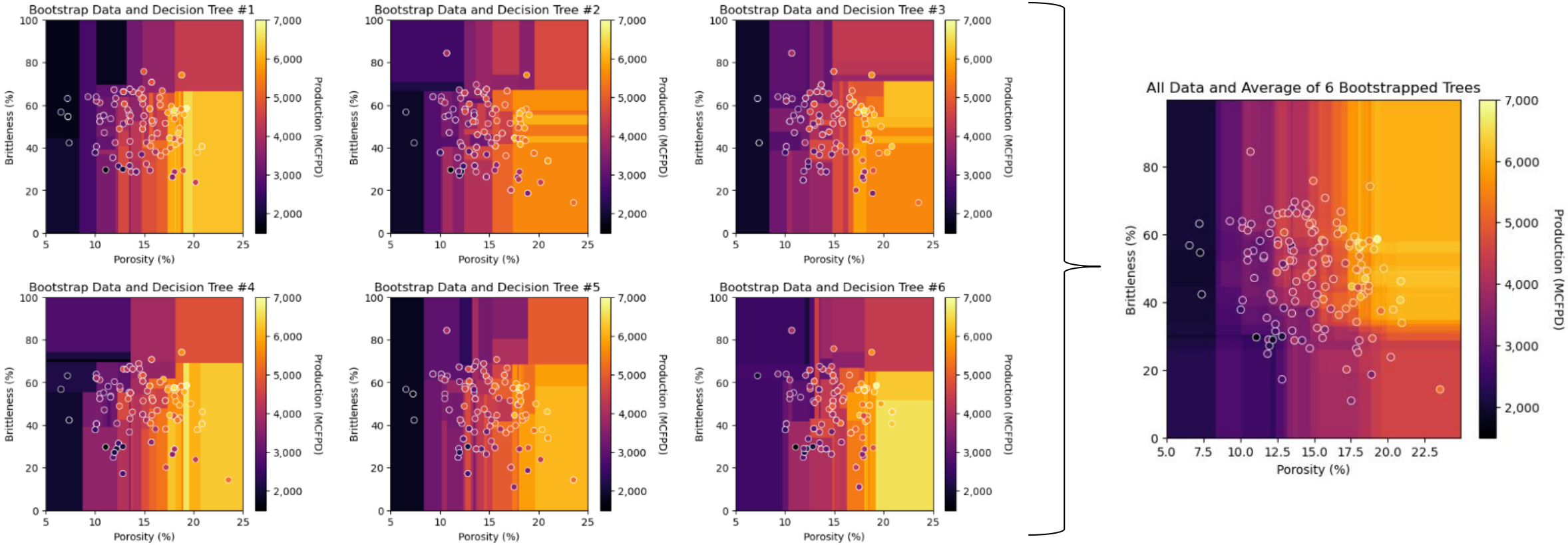
Build an ensemble of decision trees with multiple, bootstrap realizations of the data.

Comments:

- The ensemble approach **reduces model variance**
- **Hyperparameter tune over the entire ensemble model.** All trees in the ensemble have the same hyperparameters.
- **Number of estimators is an additional, important hyperparameter** in addition to tree complexity.
- In expectation, $1/3$ of the data is not used for each tree, this provides the opportunity to have access to out-of-bag samples for cross validation, so **we can build our model and cross validate with all the data at once, no train and test split.**
- **Overgrown trees will often outperform simpler trees** due to reducation in model variance.
- Spoiler Alert - we want the trees to be decorrelated, diverse to maximize the reduction in model variance, **this leads to random forest.**

Tree Bagging Example

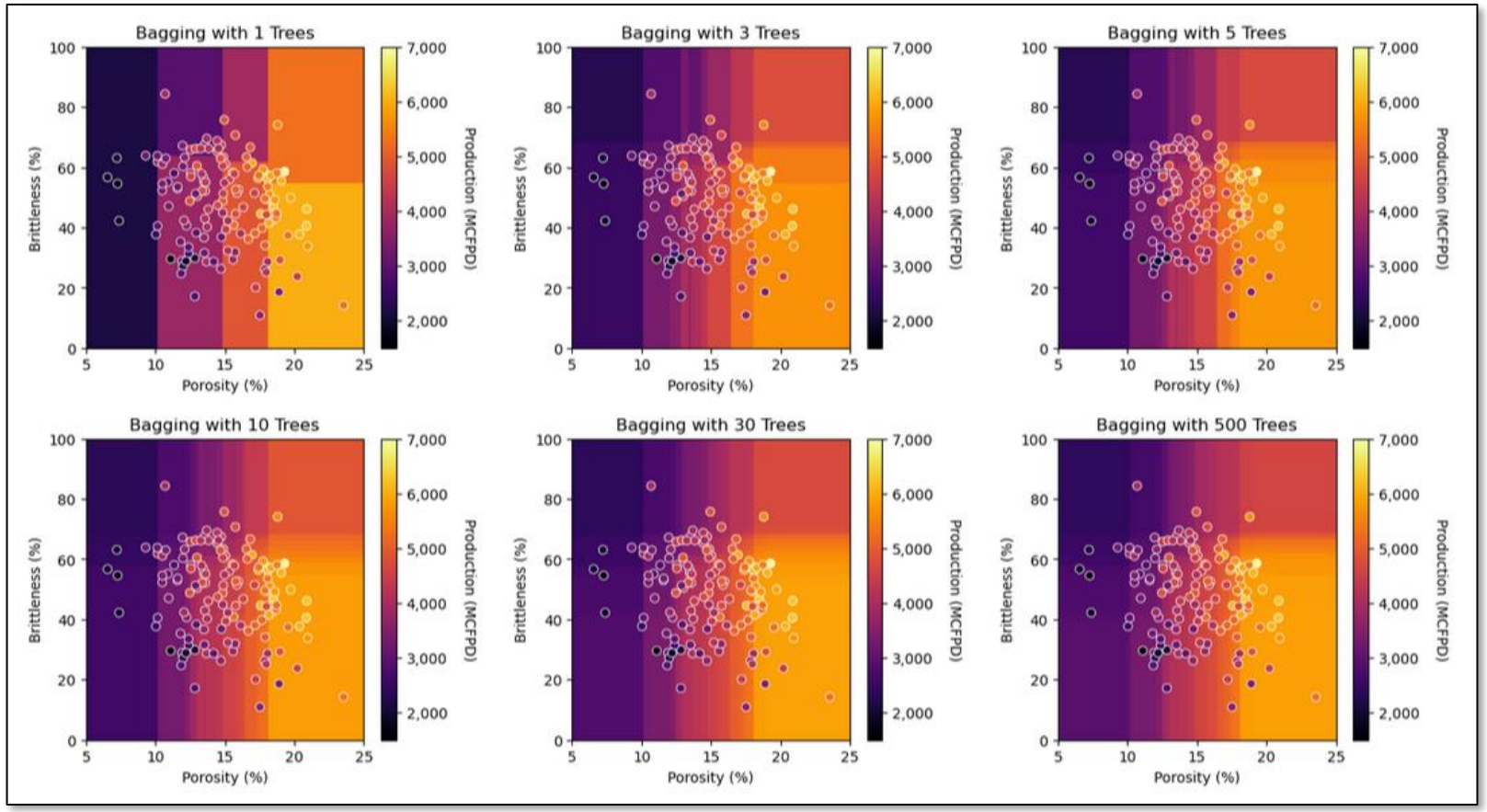
Build an ensemble of decision trees with multiple, bootstrap realizations of the data and average the predictions from all models.



6 bootstrapped, complicated decision trees (left) and the bagging model, average of all 6 models (right), from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

Tree Bagging Example

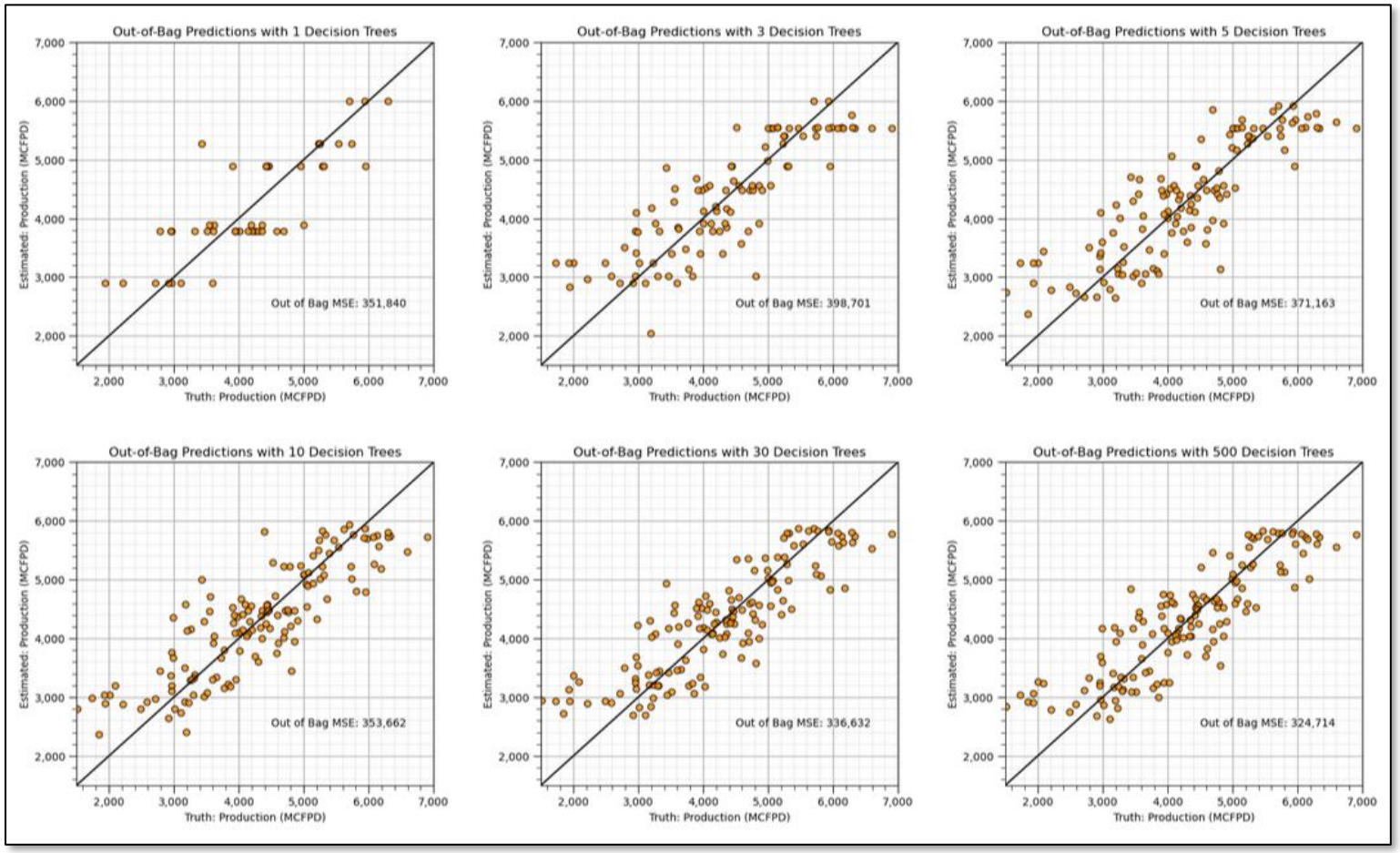
Observe the impact on the prediction model with the addition of more trees – transition from a discontinuous to continuous prediction model!



6 tree bagging prediction models and all training data with increasing number of trees,
from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

Tree Bagging Example

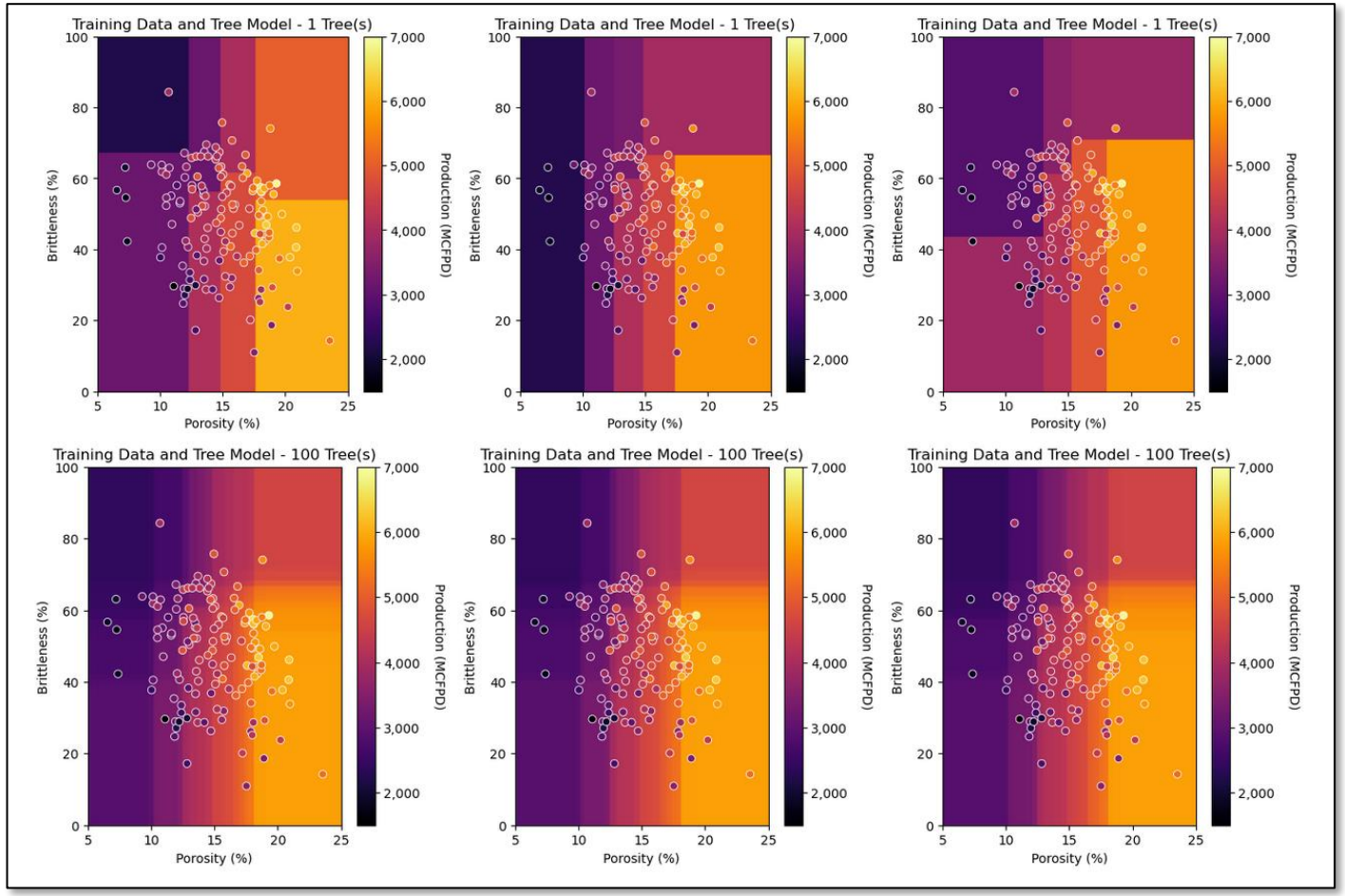
Observe the improved testing accuracy in cross validation with increasing number of trees.



Cross validation with 6 tree bagging prediction models with increasing number of trees,
from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

Tree Bagging Example

Observe the reduction in model variance with increasing number of trees.



3 models with 1 and 100 trees to demonstrate the reduction in model variance with increased ensemble aggregation, from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

Lecture outline:

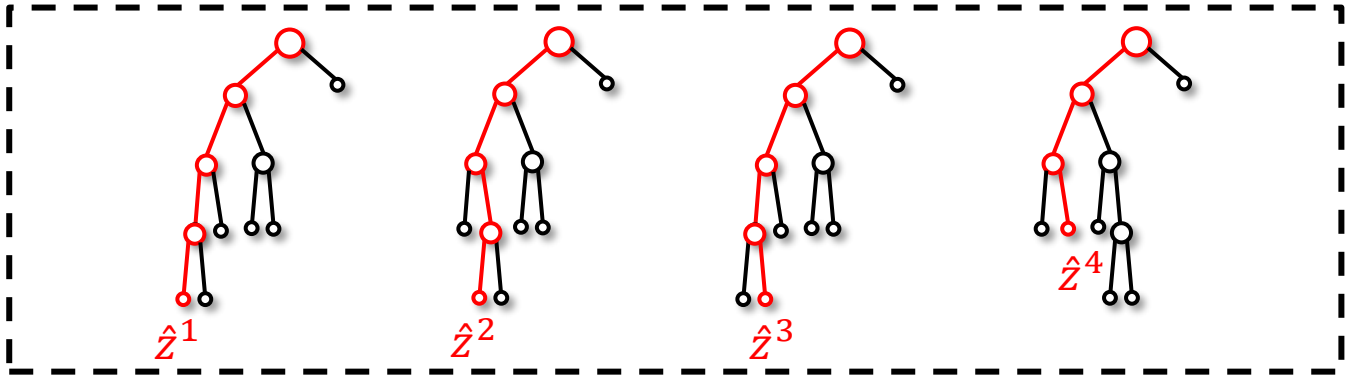
- **Random Forest**

Random Forest

A limitation with tree bagging is that the individual trees may be highly correlated

This occurs when there is a dominant predictor feature as it will always be applied to the top split(s)

- the result is all the trees in the ensemble are very similar (i.e., correlated)



Highly correlated trees in a tree bagging ensemble model.

With highly correlated trees, there is significantly less reduction in model variance with the ensemble

- consider, standard error in the mean assumes the samples n are independent!

$$\sigma_{\bar{x}}^2 = \frac{\sigma_s^2}{n}$$

- correlation between samples reduces the n to a n effective, as correlation $\uparrow$, n effective $\downarrow$.

Random Forest

Random forest is tree bagging and for each split only a subset p of the m available predictors are candidates for splits (selected at random).

$$p \ll m$$

- This forces each tree in the ensemble to evolve in dissimilar manner,

Common defaults for p for classification,

$$p = \sqrt{m} \text{ or } log_2(p)$$

and for regression,

$$p = \frac{m}{3}$$

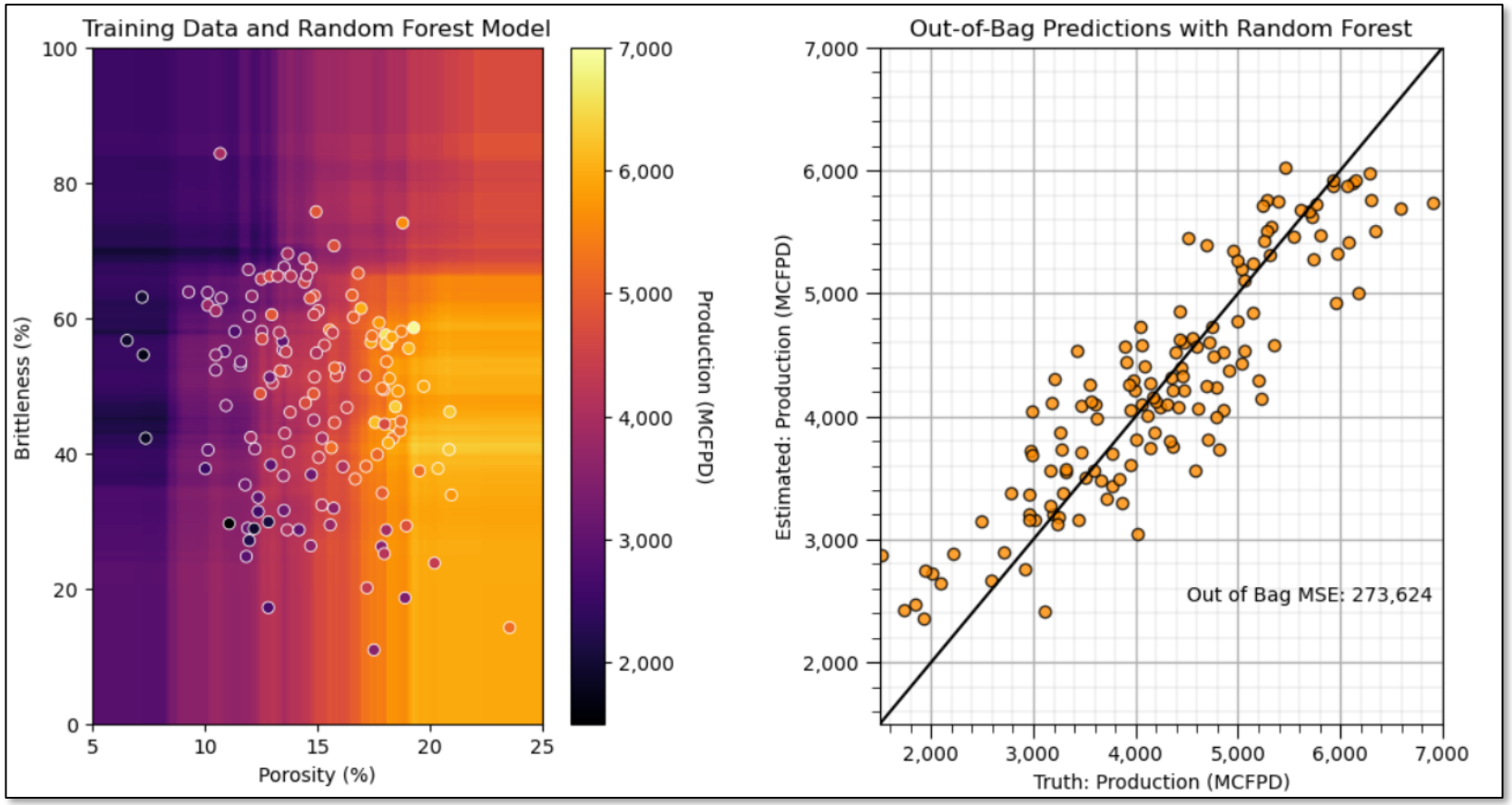
Lower p less correlation, better generalization, higher p more correlation, may overfit.

- note, too low p will underfit with high model bias

Random Forest Example

Example random forest model for the previous prediction problem

- 300 trees, trained to a maximum depth of 7, $p = 1$, 1 predictor feature randomly selected for each split

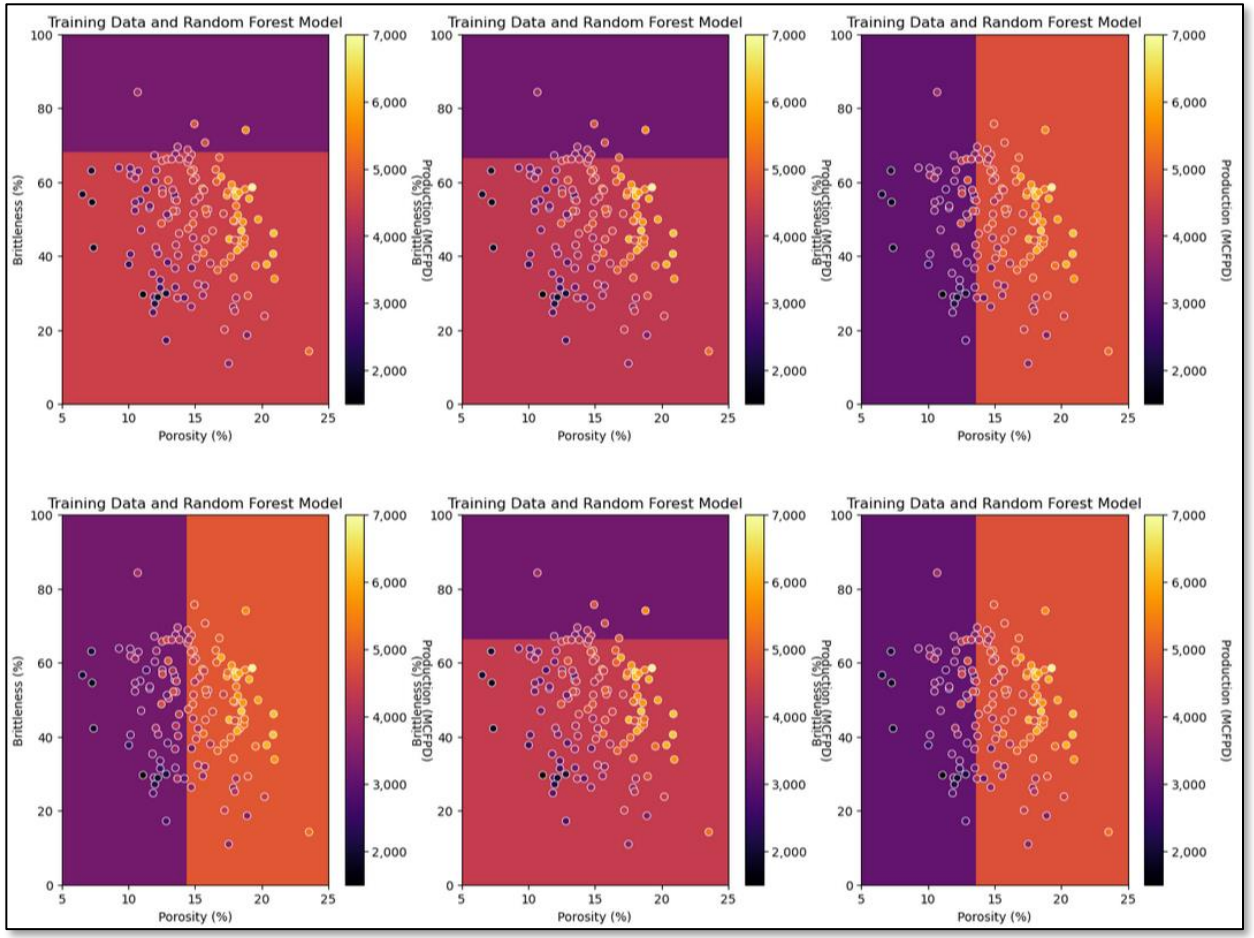


Random forest prediction model and model cross validation,
from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

Random Forest Example

Are the trees diverse? Let's freeze 6 random forest trees at the first split.

Porosity is the dominant feature, but the model selected brittleness ½ cases for the first split.



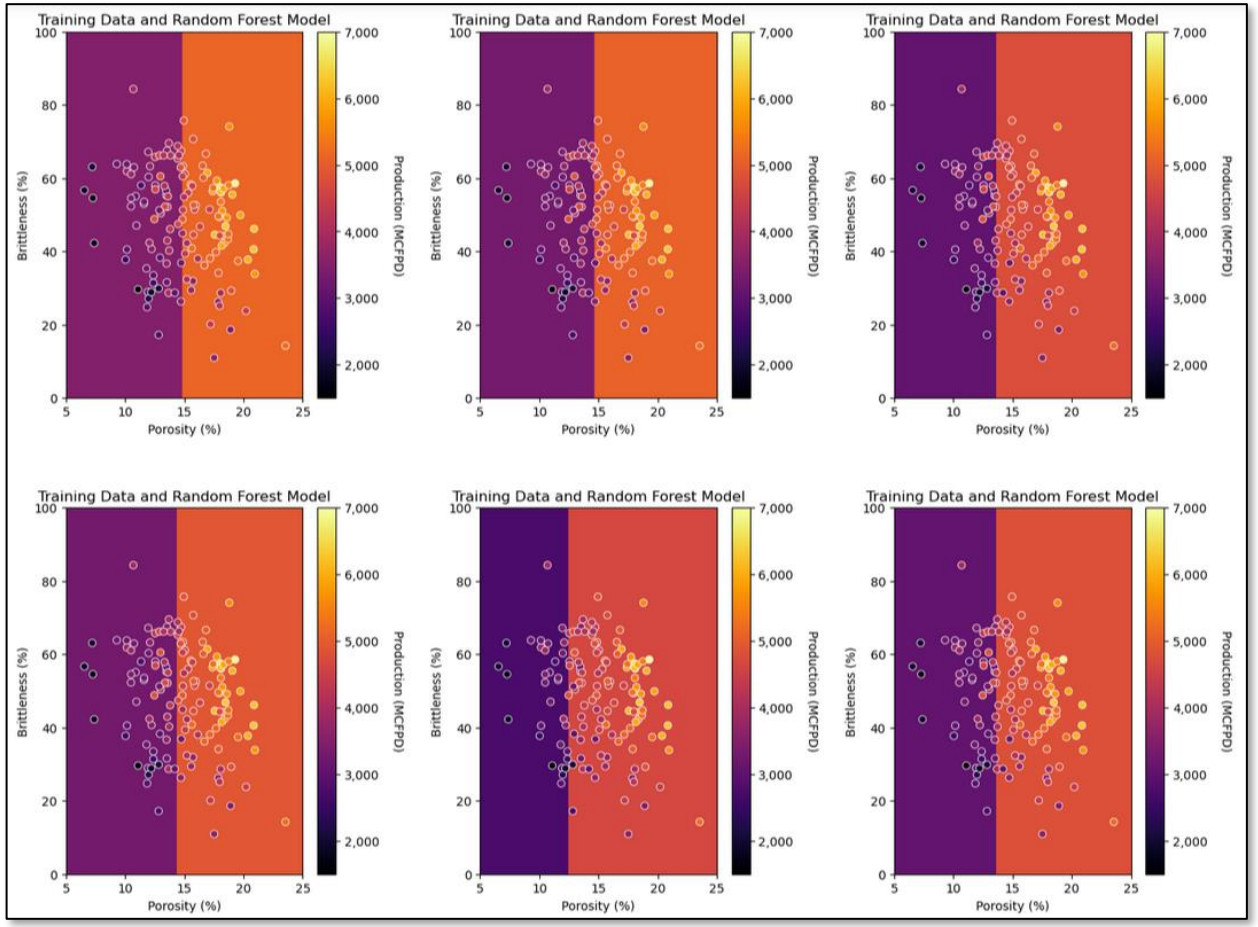
6 tree from a random forest frozen at the first split (p = 1),
from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

Random Forest Example

Compare to tree bagging, just set $p = m$ to get tree bagging from random forest!

Porosity is the dominant feature, and for all cases the first split is porosity.

- There is very little tree diversity on the first split!
- Likely highly correlated trees.

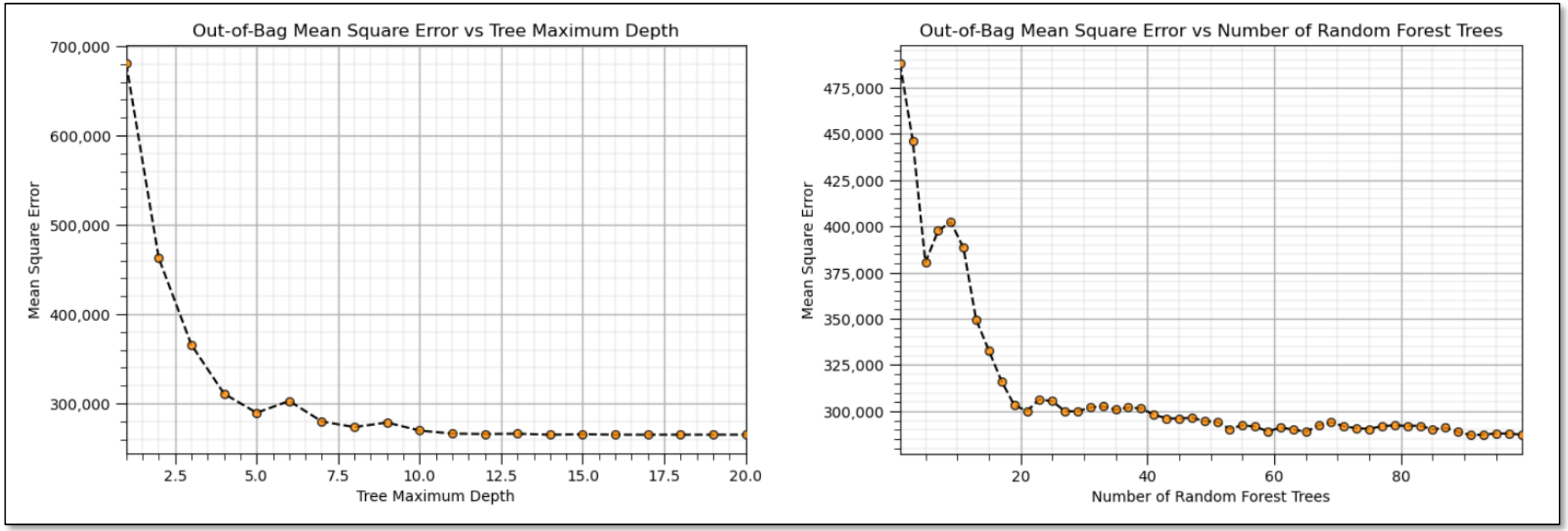


6 tree from tree bagging frozen at the first split (p = 1),
from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

Out-of-Bag Cross Validation Hyperparameter Tuning

Loop over multiple hyperparameters and calculate the out-of-bag prediction performance

- note that our random forest model is robust and resistant to overfit, the out-of-bag performance evaluation is approximately monotonically increasing



Out-of-bag MSE vs. number of random forest trees (left) and maximum tree depth (right), from Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

PGE 383 Subsurface Machine Learning

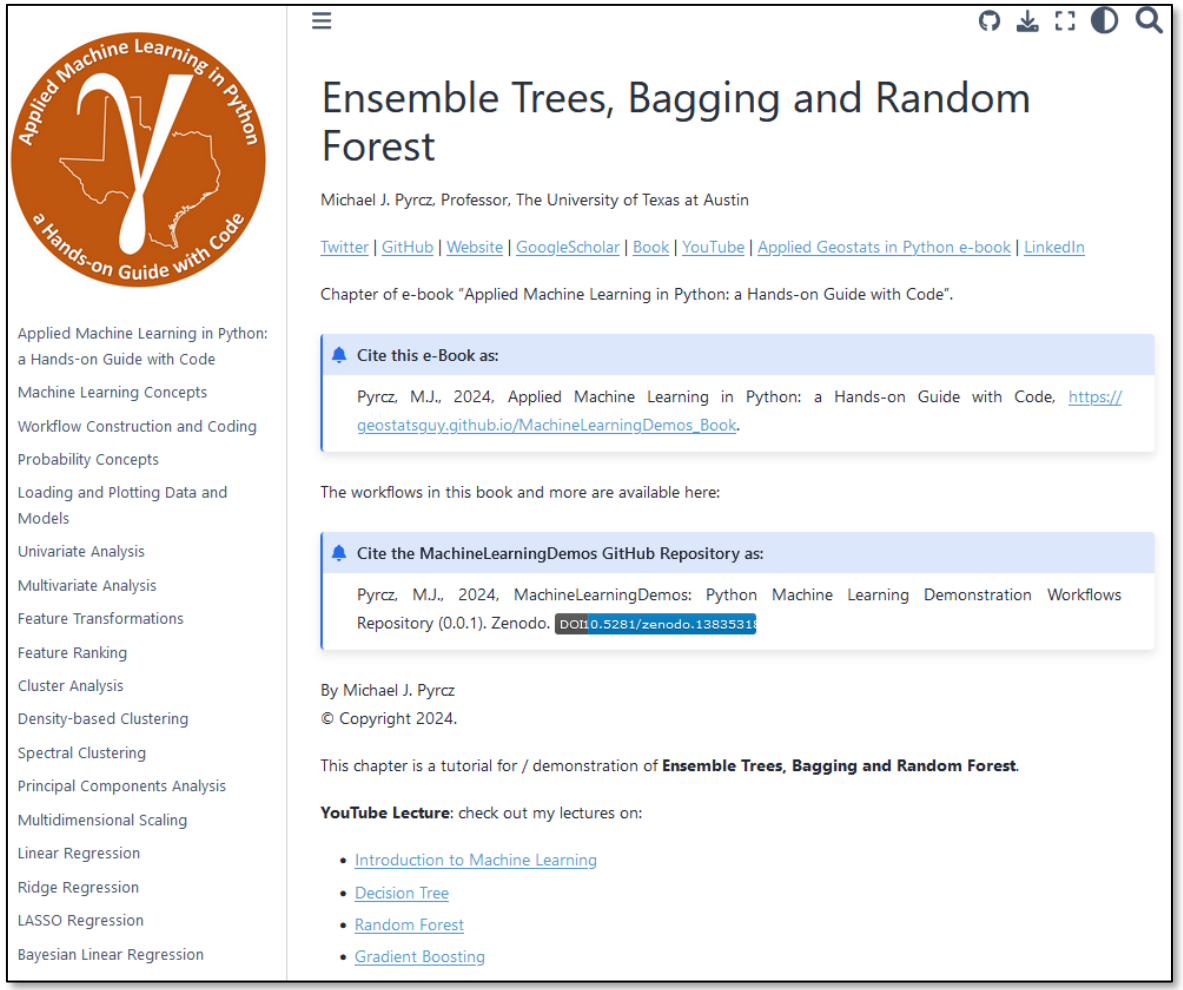
Lecture 15: Ensemble Tree

Lecture outline:

- **Ensemble Tree Methods Hands-on**

Ensemble Tree Demonstration in Python

Demonstration of ensemble tree with a well-documented workflow.



Bagging Tree and Random Forest chapter of Applied Machine Learning in Python e-book.

PGE 383 Subsurface Machine Learning

Lecture 15: Ensemble Tree

Lecture outline:

- Decision Tree Review
- Ensemble Methods
- Bootstrap
- Training and Tuning Bagging Models
- Tree Bagging
- Random Forest
- Ensemble Tree Shapley Values
- Ensemble Tree Methods Hands-on