PGE 383 Subsurface Machine Learning

Lecture 10c: Tuning Hyperparameters

Lecture outline:

- **Training and Testing**
- **Model Goodness Metrics**
- **Cross Validation Workflows**

Motivation

Let’s formalize concepts and define terms for hyperparameter tuning.

1. How do we judge a model as good?

- what is a good match with the training data as part of a loss function?
- for tuning model complexity?
- for checking our trained and tuned models in the real world?
- and working with continuous, categorical and image data?

2. How do we split train and test data?

- proportion to withhold
- hold out vs. k-fold cross validation

PGE 383 Subsurface Machine Learning

Lecture 10c: Tuning Hyperparameters

Lecture outline:

- Training and Testing Data Splits

Model Parameters

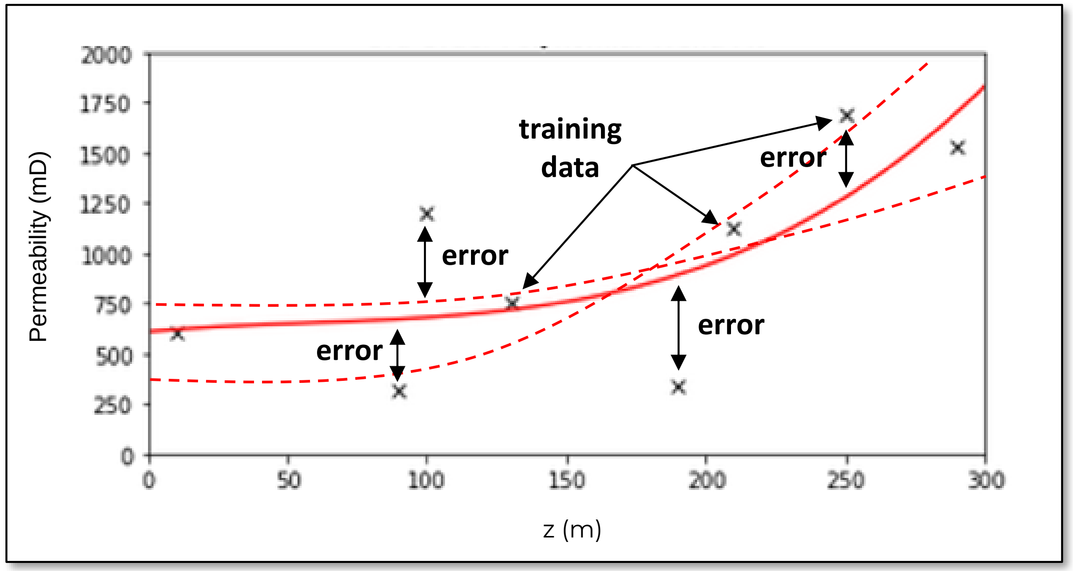
Model Parameters

- Fit during training phase to minimize error at the training data
- For this 3rd order polynomial:

$$y = b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

Parameters:

b_3, b_2, b_1 and b_0



Setting model parameters to minimize the error relative to training data.

Model HyperParameters

Model Hyperparameters

- Constrain the model complexity.
- Select hyperparameters that maximize accuracy with the testing data.

- For a polynomial model:

⋮

7th Order: $y = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$

⋮

5th Order: $y = b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$

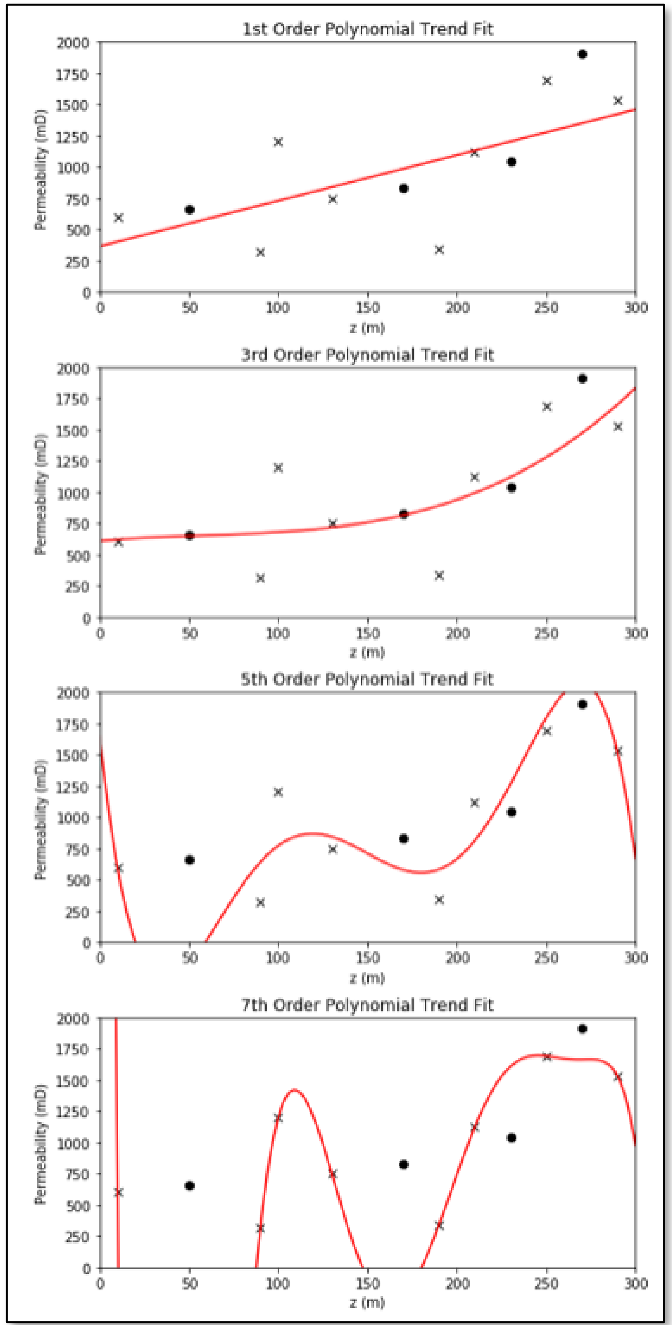
⋮

3rd Order: $y = b_3x^3 + b_2x^2 + b_1x + b_0$

2nd Order: $y = b_2x^2 + b_1x + b_0$

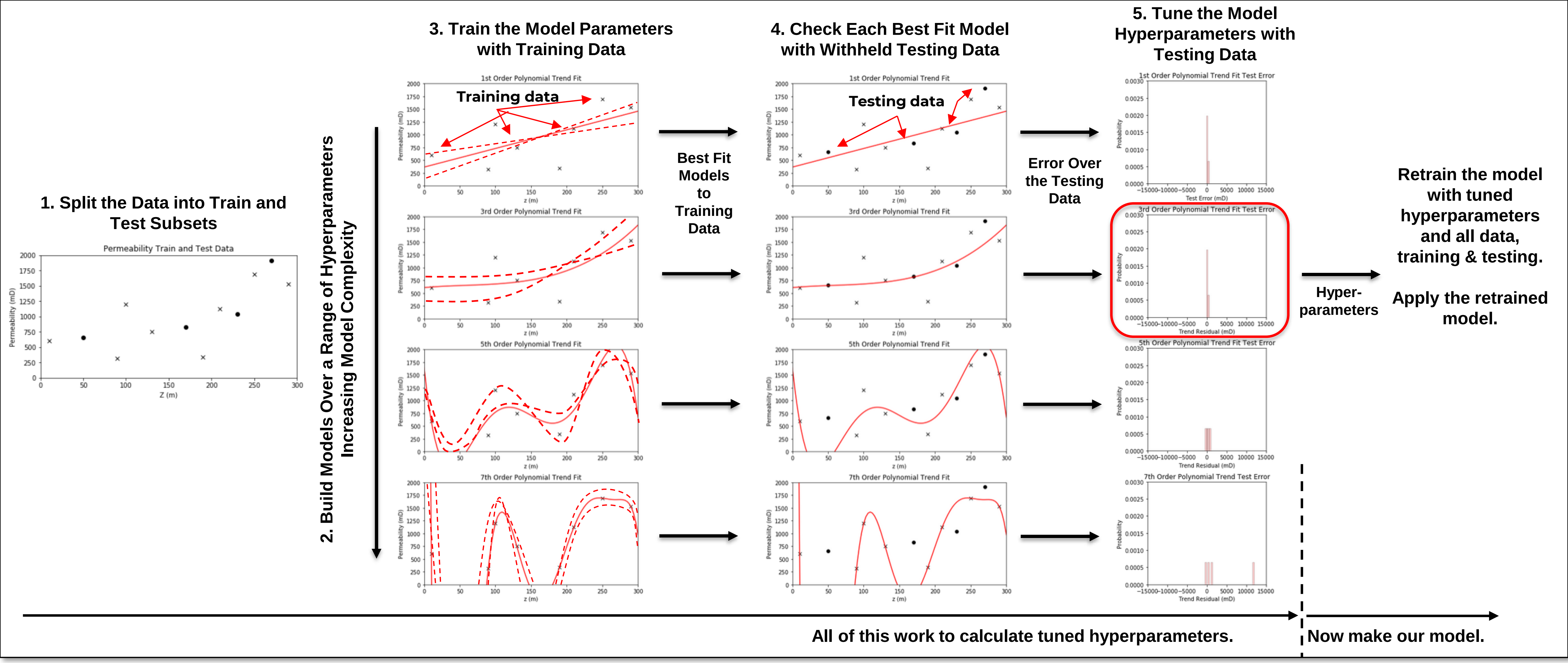
1st Order: $y = b_1x + b_0$

Increasing Complexity



Varying the model complexity, model hyperparameter, to maximize fit with testing data.

Predictive Model Workflow

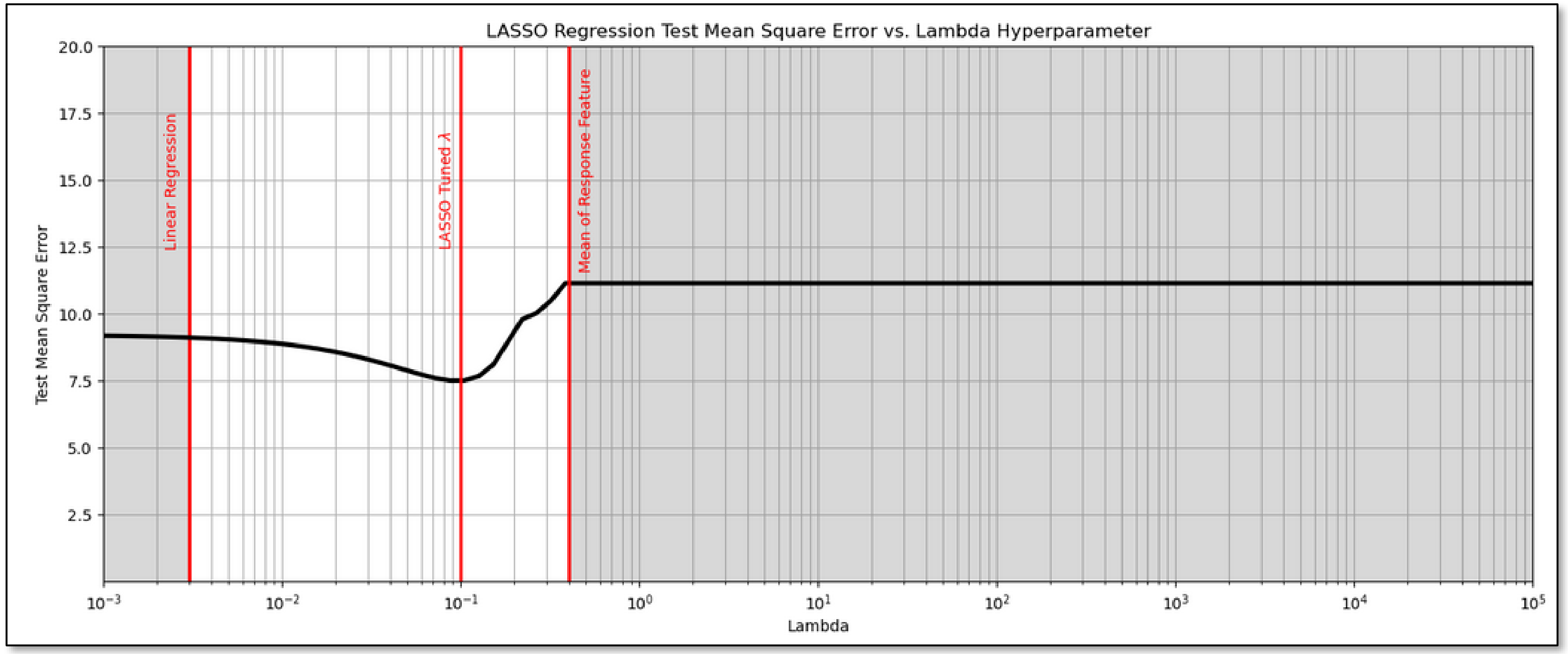

1. Split the Data into Train and Test Subsets
Permeability Train and Test Data
2. Build Models Over a Range of Hyperparameters Increasing Model Complexity
3. Train the Model Parameters with Training Data
Training data
1st Order Polynomial Trend Fit
3rd Order Polynomial Trend Fit
5th Order Polynomial Trend Fit
7th Order Polynomial Trend Fit
Best Fit Models to Training Data
4. Check Each Best Fit Model with Withheld Testing Data
Testing data
Error Over the Testing Data
5. Tune the Model Hyperparameters with Testing Data
1st Order Polynomial Trend Fit Test Error
3rd Order Polynomial Trend Fit Test Error
5th Order Polynomial Trend Fit Test Error
7th Order Polynomial Trend Test Error
Hyper-parameters
Retrain the model with tuned hyperparameters and all data, training & testing.
Apply the retrained model.
All of this work to calculate tuned hyperparameters.
Now make our model.

Machine learning model building workflow to avoid overfit.

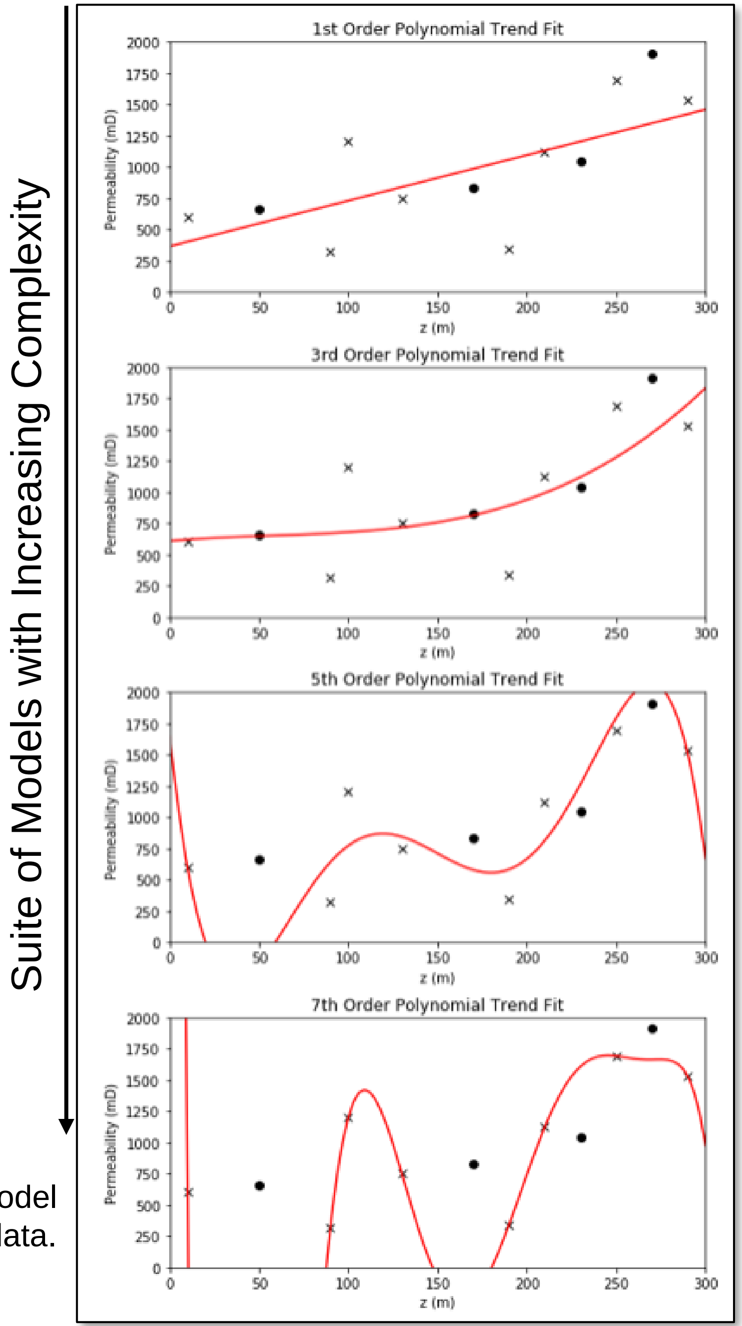
Hyperparameter Tuning

What do we have?

- A suite of models of variable level of complexity, and a summary measure of model accuracy for withheld testing data, e.g., an metric / error norm, like mean square error for each model etc.



Test mean square error vs. LASSO hyperparameter, LASSO chapter of Applied Machine Learning in Python.



Varying the model complexity, model hyperparameter, to maximize fit with testing data.

Training and Testing Data

Model Parameter Training and Model Hyperparameter Tuning

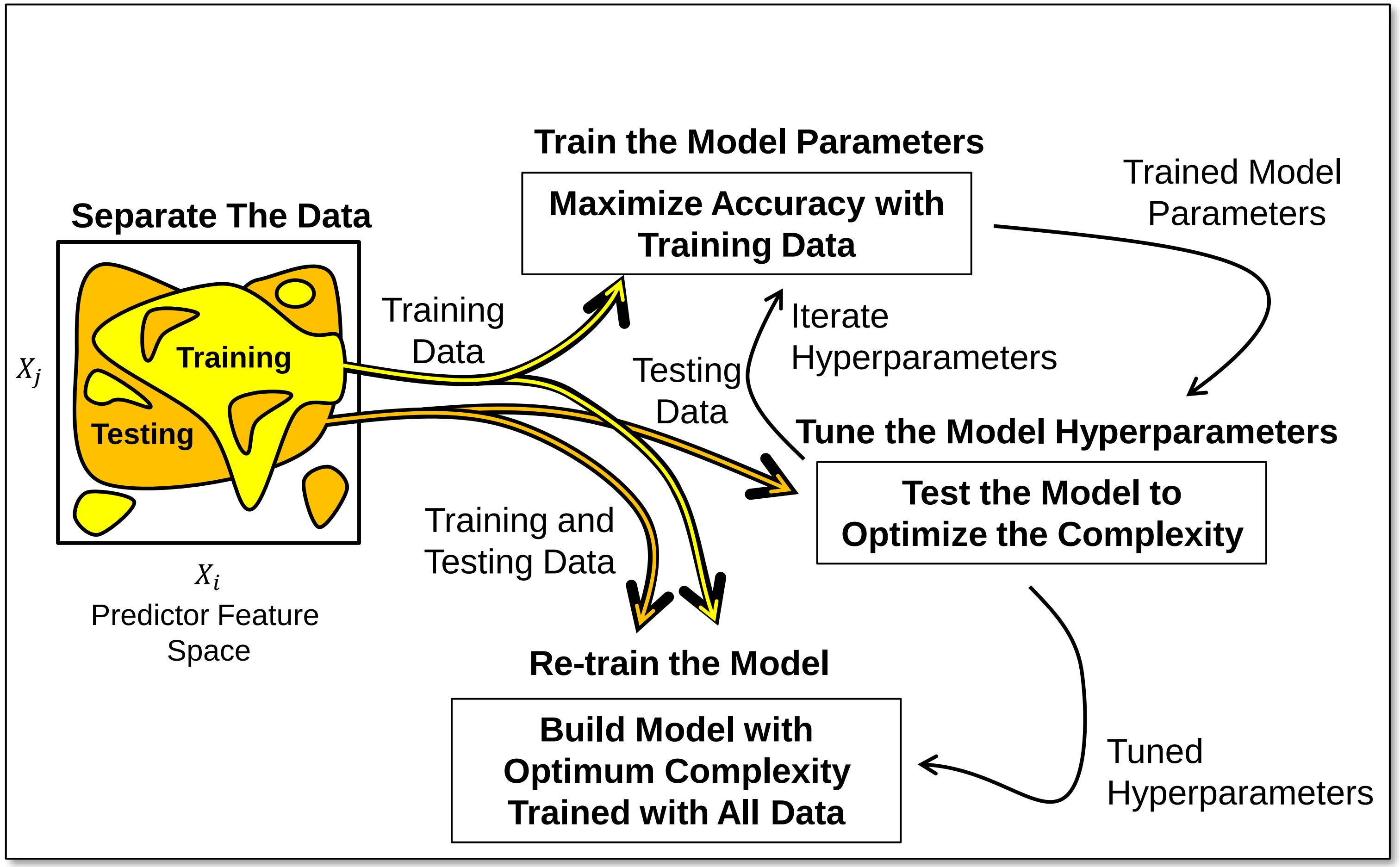
Establish subsets of the data for testing of the model

Training data

- trains model parameters
- trains the final model for real world use

Testing data

- withheld from training model parameters to avoid model overfit
- tunes model hyperparameters
- trains the final model for real world use



Separate The Data
Training
Testing
X_j
X_i
Predictor Feature Space
Training Data
Testing Data
Training and Testing Data
Train the Model Parameters
Maximize Accuracy with Training Data
Trained Model Parameters
Iterate Hyperparameters
Tune the Model Hyperparameters
Test the Model to Optimize the Complexity
Re-train the Model
Build Model with Optimum Complexity Trained with All Data
Tuned Hyperparameters

Schematic of the use of training and testing data for model parameter training and model hyperparameter tuning workflow to avoid overfit.

Training and Testing Split

Fair Testing in Spatial / Temporal Settings

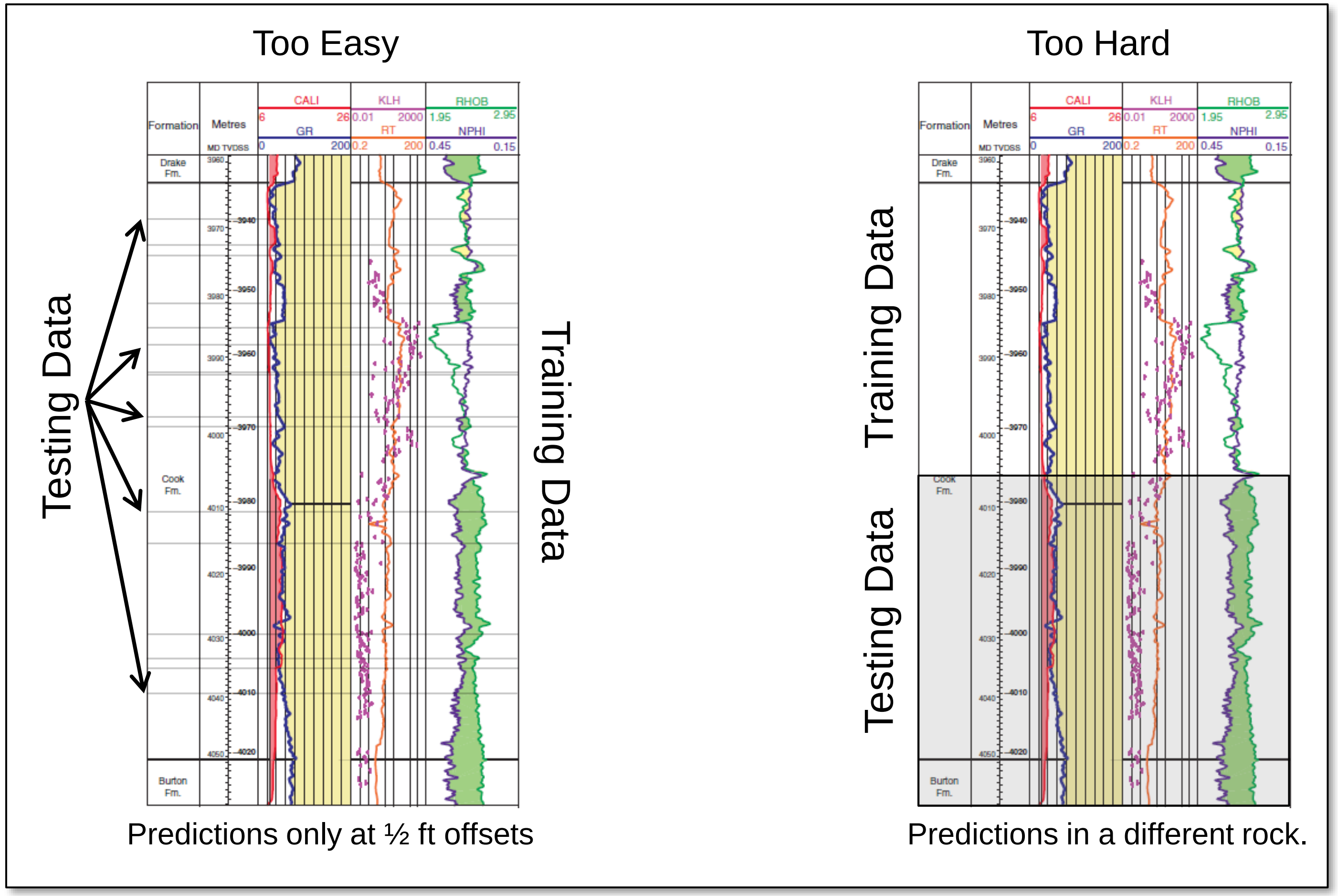
The train and test data split should be fair

The prediction difficulty (interpolation, extrapolation) should be similar to the planned real-world use of the prediction model.

Dress rehearsal of actual model use

See Dr. Julian Salazar's work on fair train and test splits (Salazar et al., 2022).

UT PGE PHD



Testing data split by random selection (left), and by unique population (right).

Training and Testing Split

Fair Testing in Spatial / Temporal Settings

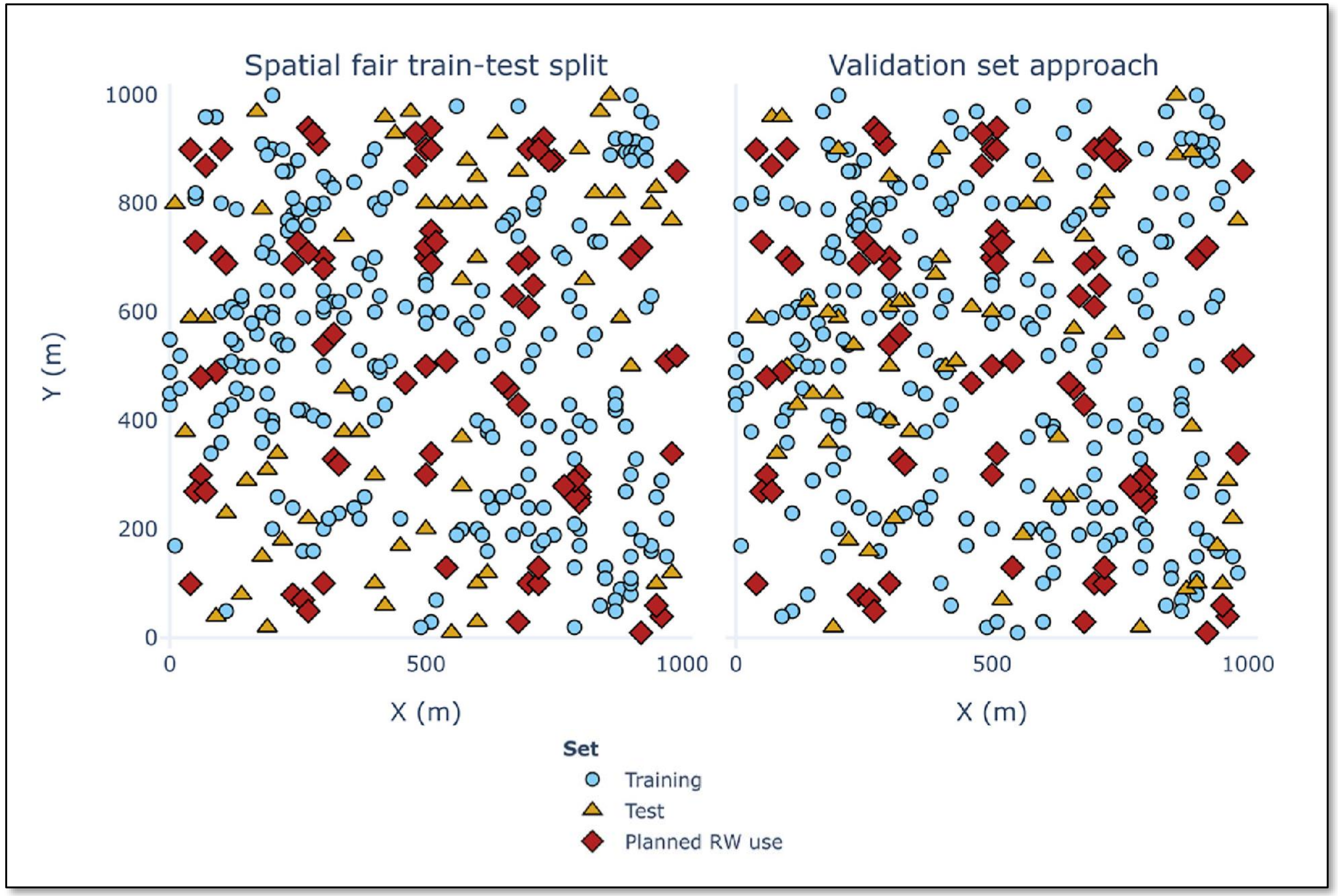
The train and test data split should be fair

The prediction difficulty (interpolation, extrapolation) should be similar to the planned real-world use of the prediction model.

Dress rehearsal of actual model use

See Dr. Julian Salazar's work on fair train and test splits (Salazar et al., 2022).

UT PGE PHD



Fair train and test split to match difficulty of planned real-world use of the model (left), and random split (right).

Training and Testing Split

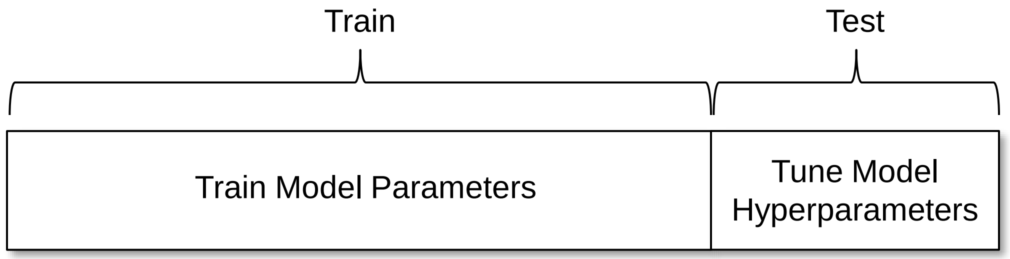
How Much Data Should be Withheld for Testing?

The proportion in testing is recommended by various sources from 30% - 15% of the total dataset.

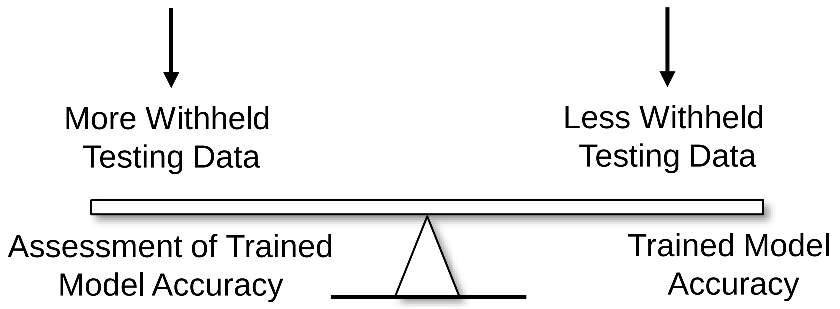
- Data withheld for testing reduces the data available for training; therefore, reduces the accuracy of the model.
- Data withheld for testing improves the accuracy of the assessment of the model performance.

Various authors have experimented on a variety of training and testing ratios and have recommended splits for their applications:

- The optimum ratio of training and testing split depends on problem setting
- Could consider the difficulty in model parameter training (e.g., the number of model parameters) and the difficulty in model hyperparameter tuning (e.g., number of hyperparameters, range of reponse feature outcomes).



The train and test data splits approach.


More Withheld Testing Data
Less Withheld Testing Data
Assessment of Trained Model Accuracy
Trained Model Accuracy

Alternative Testing Workflow

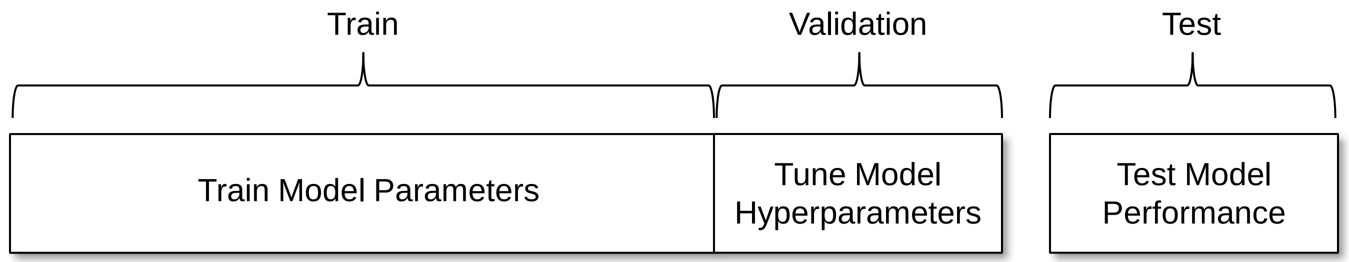
Training, Validation and Testing

There is a more complete workflow commonly applied. Note: to avoid confusion in our class we will use the train and test approach only.

- **Train with training data**. Models sees and learns from this data to train the model parameters.

- **Validate with validation data**. Unbiased evaluation of model fit to tune the model hyperparameters (testing data in train and test workflow).

- **Test with testing data**. Data withheld until the model is complete to provide a final evaluation. Commonly applied to compare multiple competing models. This data had no role in building the model.



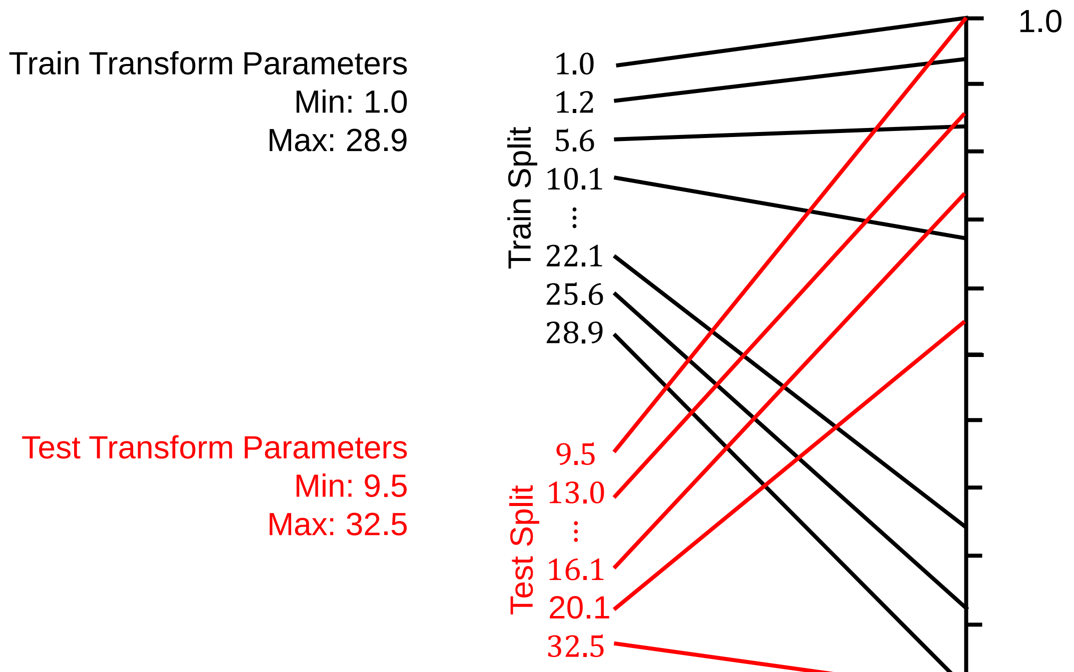
The train, validate and test data splits approach.

Data Transformations and Train and Test Split

Do Not Transform Train and Test Independently

We will likely not honor the relative magnitudes and perhaps even the rank orders.

- inconsistent, unstable mapping of any new dataset to the application of your workflow!



Min-max normalization of train and test independently.

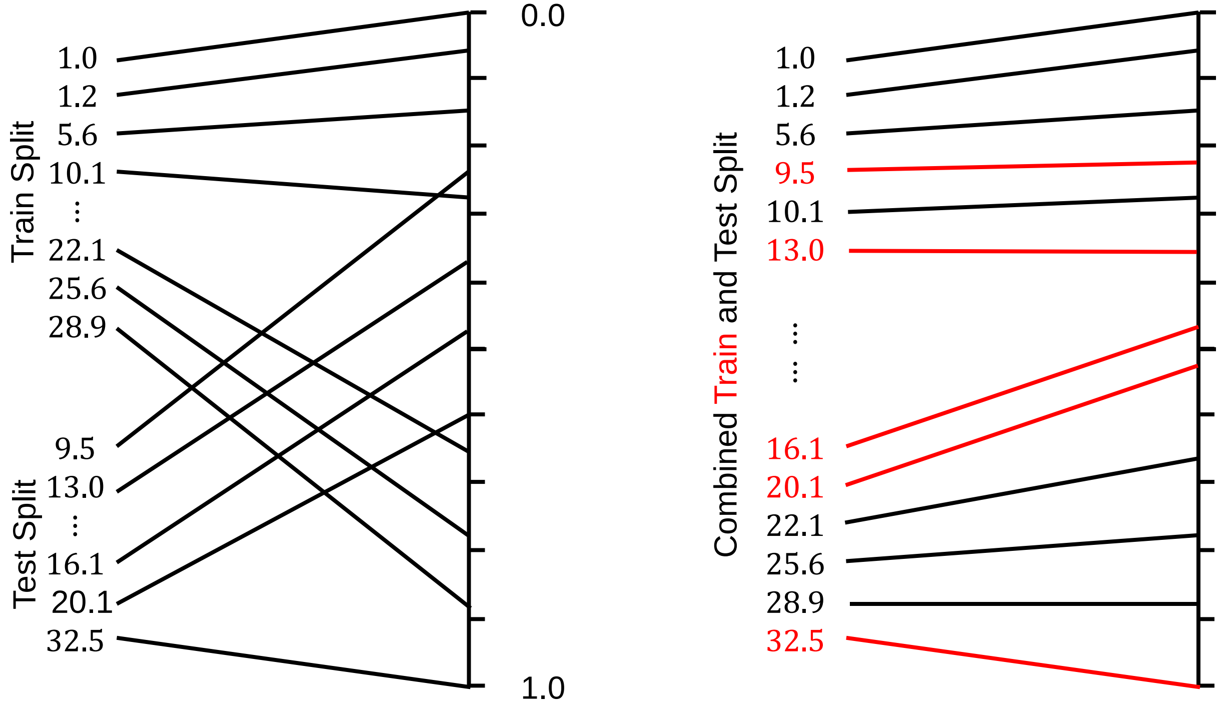
Data Transformations and Train and Test Split

Do Not Transform Train and Test Together (yes, that is in our workflows now! To be corrected)

Consistent transformation, honors min and max, but potential information leakage, testing data improves the inference of statistics for the transform.

- fails the "dress rehearsal" / "simulate real-world application" criteria for train and test split.

Train and Test Transform Parameters
Min: 1.0
Max: 32.5



Min-max normalization of train and test combined, plotted separated (left) and shuffled together to demonstrate rank preserving (right).

Data Transformations and Train and Test Split

Transform Train Split and the Apply Same Transform to the Test Split (use train parameters)

Avoid potential information leakage, test data helping inform the predictions, through transformation parameters.

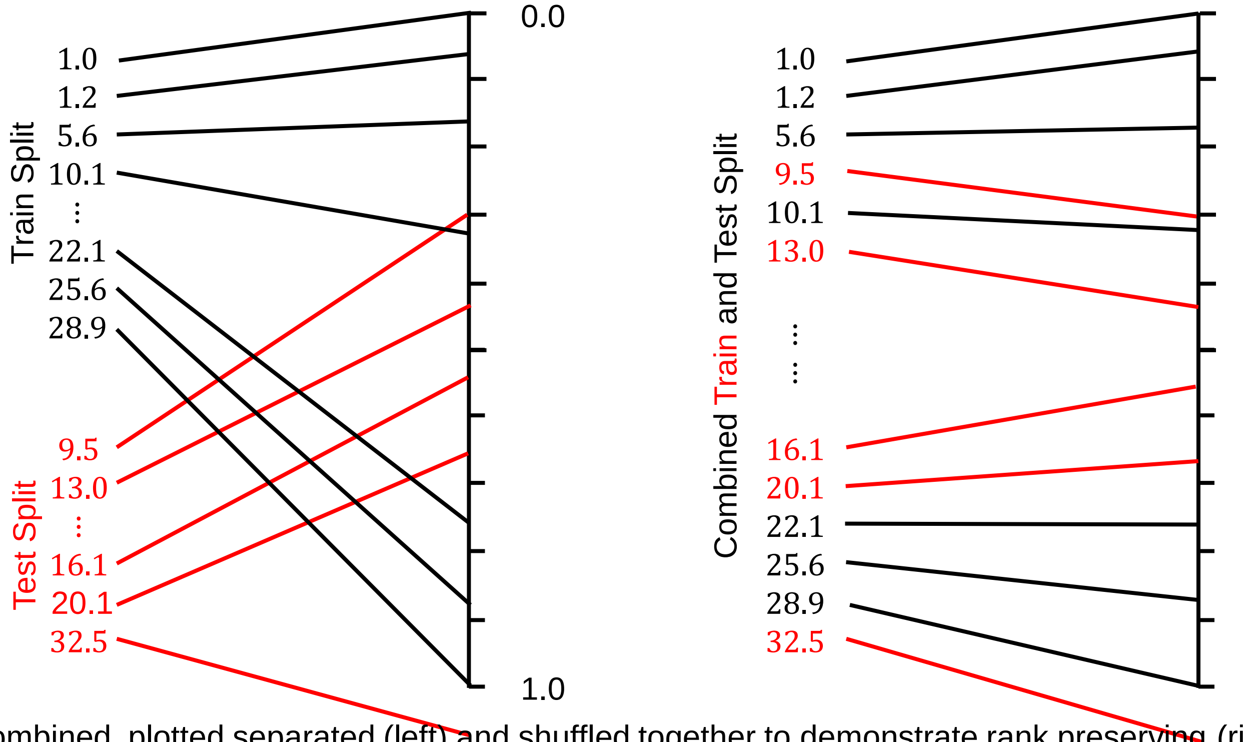
Train Transform Parameters
Min: 1.0
Max: 28.9

Comments:
Distort the real-world inference pipeline – should be a "Dress rehearsal" – model is only trained on train data and test data has now impact on model training.

Information leakage – gives the model training indirect access to the test data distribution

Hides Train, Test and Future Use Distribution drift – hiding potential extrapolation.

If train, test, and future use distributions are the same, e.g., the system is bounded, there is no issue.



Min-max normalization of train and test combined, plotted separated (left) and shuffled together to demonstrate rank preserving (right).

Data Transformations and Train and Test Split

How Do We Do This?

For hold-out cross validation this is easy.

1. instantiate transform
2. fit transform on train split
3. perform same transform on test split

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
```

For k-fold cross validation we have 2 choices:

1. manual loops over the folds $\longrightarrow$

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)
scores = []

for train_idx, val_idx in kf.split(X):
    X_train, X_val = X[train_idx], X[val_idx]
    y_train, y_val = y[train_idx], y[val_idx]

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_val_scaled = scaler.transform(X_val)

    model = LogisticRegression()
    model.fit(X_train_scaled, y_train)

    preds = model.predict(X_val_scaled)
    scores.append(accuracy_score(y_val, preds))
```

2. pipelines

```
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LogisticRegression())
])

cv = KFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(pipe, X, y, cv=cv)
print(scores.mean())
```

PGE 383 Subsurface Machine Learning

Lecture 10c: Tuning Hyperparameters

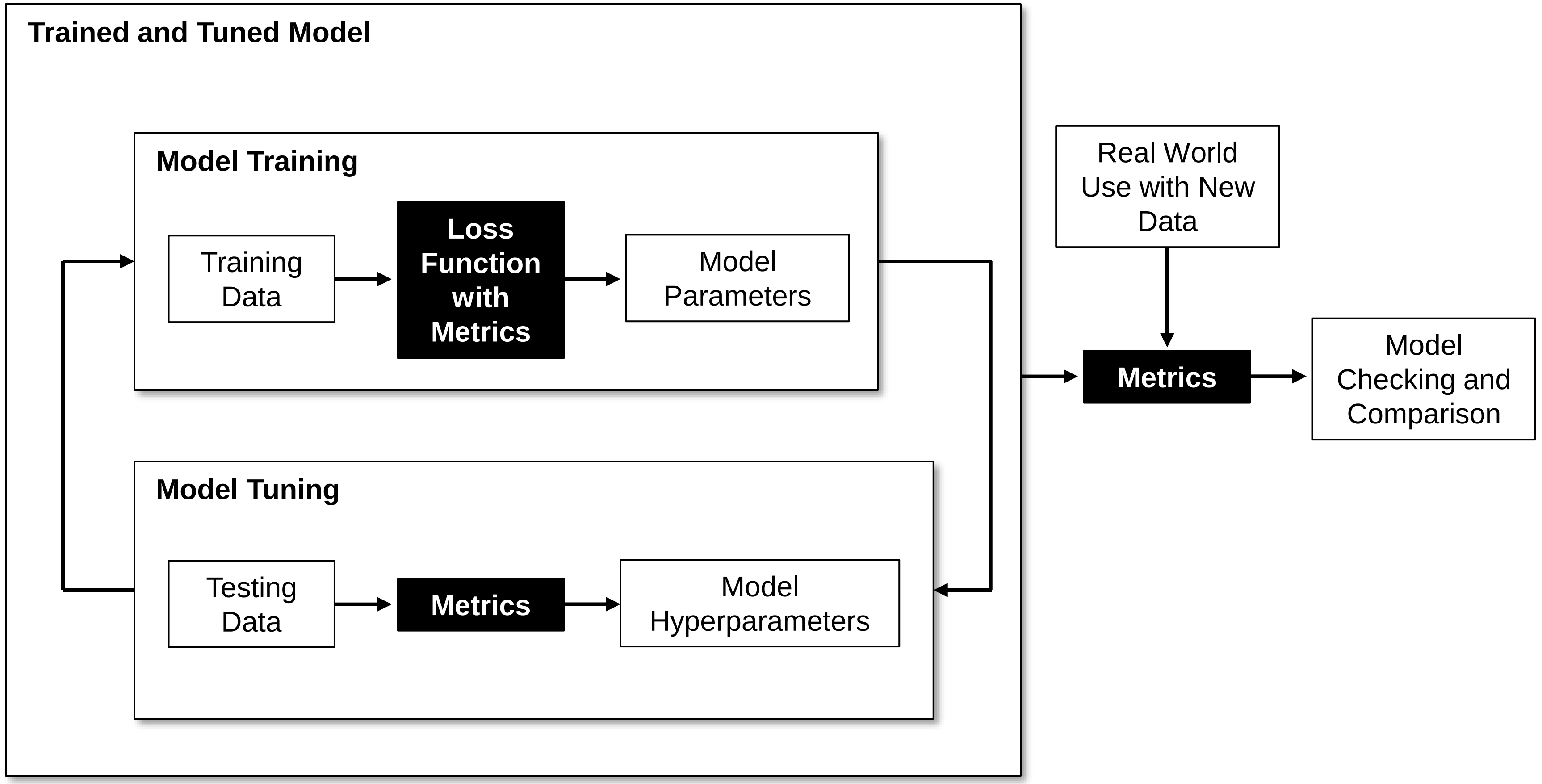
Lecture outline:

- **Model Goodness Metrics**

Application of Model Metrics

Model Metrics

These are used in loss functions (training), metrics (tuning) and for general model checking!



Various applications for model metrics in machine learning modeling workflows.

Model Metrics

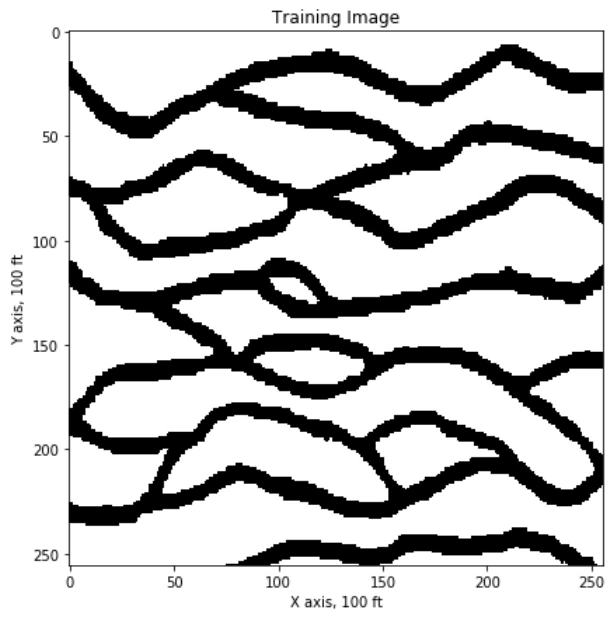
Model Metrics

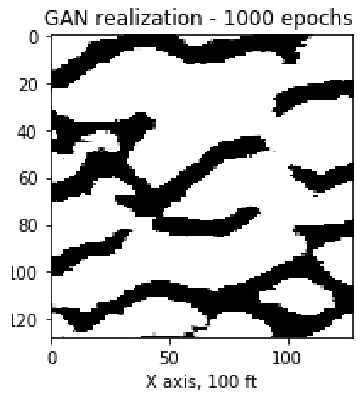
Choice of model metric depends primarily on the context of the prediction problem,

- classification vs. regression
- individual estimates vs. entire subsets in space (images) or time (signals)
- estimation vs. uncertainty

with additional considerations previously discussed,

- L^1 vs L^2 norms
- consistency with model assumptions, r^2, only for linear models





Training image and GenAI image from a Generative Adversarial Network, SubsurfaceDataAnalytics_GenerativeAdversarialNetwork.ipynb

Regression Model Metrics

Regression Model Metrics

Mean Square Error (MSE)

- L^2 norm – sensitive to large errors

$$Test\ MSE = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_i - \hat{y}_i)^2 = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (\Delta y_i)^2$$

Mean Absolute Error (MAE)

- L^1 norm – less sensitive to large errors

$$Test\ MAE = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} |y_i - \hat{y}_i| = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} |\Delta y_i|$$

Regression Model Metrics

Regression Model Metrics

Variance Explained – note, **we only use this for linear models**

- Proportion of variance of the response feature captured by the model
- Takes advantage of the additivity of variance

Total Variance = Variance Explained + Variance Not Explained

$$\boldsymbol{\sigma^2_{explained}} = \frac{\mathbf{1}}{n_{test}} \sum_{\boldsymbol{i=1}}^{n_{test}} (\boldsymbol{\hat{y}_i} - \boldsymbol{\bar{y}})^{\mathbf{2}} \qquad \boldsymbol{\sigma^2_{not\ explained}} = \frac{\mathbf{1}}{n_{test}} \sum_{\boldsymbol{i=1}}^{n_{test}} (\boldsymbol{y_i} - \boldsymbol{\hat{y}_i})^{\mathbf{2}}$$

$$r^2 = \frac{\sigma^2_{explained}}{\sigma^2_{explained} + \sigma^2_{not\ explained}} = \frac{\sigma^2_{explained}}{\sigma^2_{total}}$$

Issues

- for linear regression (1 predictor feature), recall $r^2 = \left(\rho_{X,y}\right)^2$, like correlation coefficients, linearity assumption & sensitive to outliers.
- e.g., recall impact of adding outliers and 2 populations on correlation (Simpson's Paradox)!

Regression Model Metrics

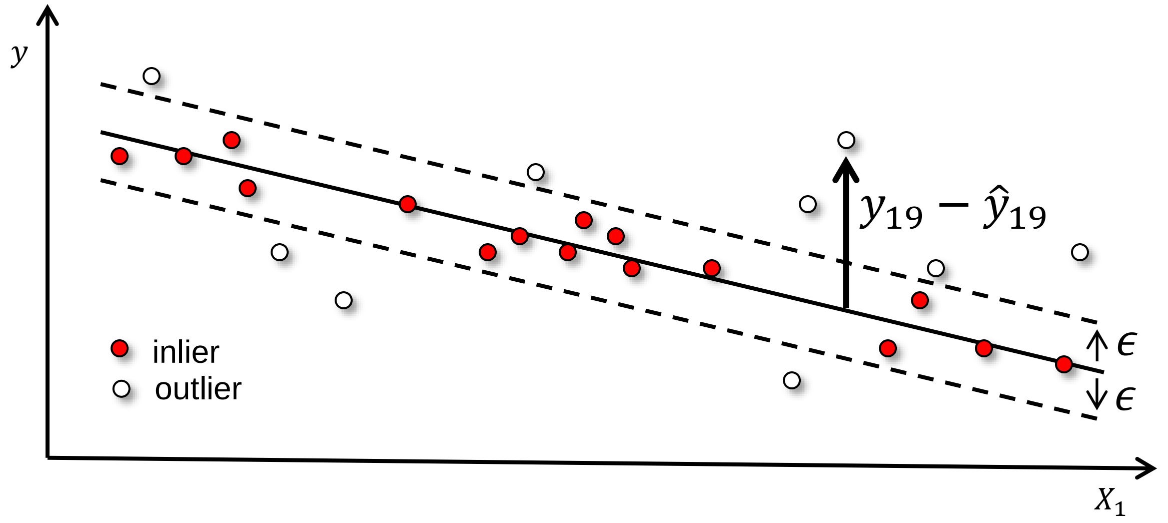
Regression Model Metrics

Inlier Ratio, proportion of testing data within a margin, ϵ, of the model, $\hat{y}_i$.

$$IR = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} I(y_i, \hat{y}_i)$$

Given the indicator transform:

$$I(y_i, \hat{y}_i) = \begin{cases} 1, & \text{if } |y_i - \hat{y}_i| \leq \epsilon \\ 0, & \text{otherwise} \end{cases}$$



Testing data, model with margin, ϵ, and outliers (white) and inliers (red) identified, 16 inliers out of 25 data samples, $IR = 0.64$.

Issues: what proportion of data is the model close enough? What is the best margin size?

Classification Testing Metrics

Confusion Matrix

- Matrix with the categorical truth vs. predicted, $K \; x \; K$ matrix where K is the categorical feature cardinality.
- Visualize and diagnose all the combinations of correct and misclassification with the classification model.

$$\text{Truth}\quad\begin{array}{c|c|c|c|}
\cline{2-4}
C_{k=1} & \mathbf{15} & \mathbf{15} & \mathbf{9} \\
\cline{2-4}
C_{k=2} & \mathbf{5} & \mathbf{22} & \mathbf{2} \\
\cline{2-4}
C_{k=3} & \mathbf{7} & \mathbf{15} & \mathbf{4} \\
\cline{2-4}
\multicolumn{1}{c}{} & \multicolumn{1}{c}{\hat{C}_{k=1}} & \multicolumn{1}{c}{\hat{C}_{k=2}} & \multicolumn{1}{c}{\hat{C}_{k=3}}
\end{array}$$

Predicted

Model predicts category 3, $\boldsymbol{\hat{C}_{k=3}}$,
but the true values is category 1, $\boldsymbol{C_{k=1}}$.

- Perfect accuracy is number of each class in the training data on the diagnonal

$$\text{Truth}\quad\begin{array}{c|c|c|c|}
\cline{2-4}
C_{k=1} & \boldsymbol{n_1} & \mathbf{0} & \mathbf{0} \\
\cline{2-4}
C_{k=2} & \mathbf{0} & \boldsymbol{n_2} & \mathbf{0} \\
\cline{2-4}
C_{k=3} & \mathbf{0} & \mathbf{0} & \boldsymbol{n_3} \\
\cline{2-4}
\multicolumn{1}{c}{} & \multicolumn{1}{c}{\hat{C}_{k=1}} & \multicolumn{1}{c}{\hat{C}_{k=2}} & \multicolumn{1}{c}{\hat{C}_{k=3}}
\end{array}$$

Predicted

n for each category
on the diagonal

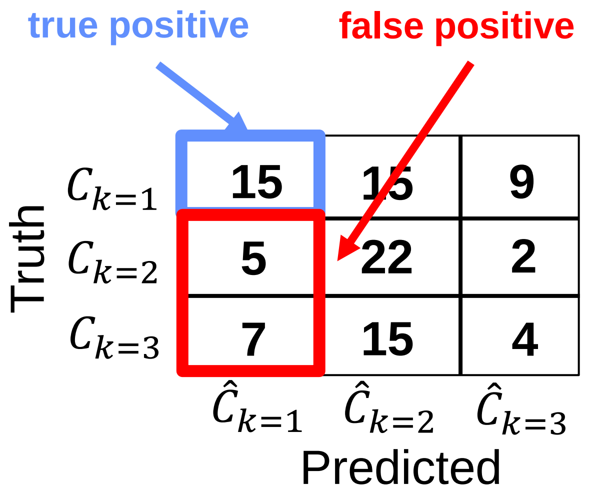
Classification Testing Metrics

Precision – for category k, the ratio of true positive over all positives

$$precision_k = \frac{n_{k\ true\ positive}}{n_{k\ true\ positive} + n_{k\ false\ positive}} = \frac{true\ positive}{all\ positives}$$

$precision_k = Probability(k\ is\ happenning \mid model\ says\ k\ is\ happenning)$, does not account for false negatives.

$\boldsymbol{k = 1}$

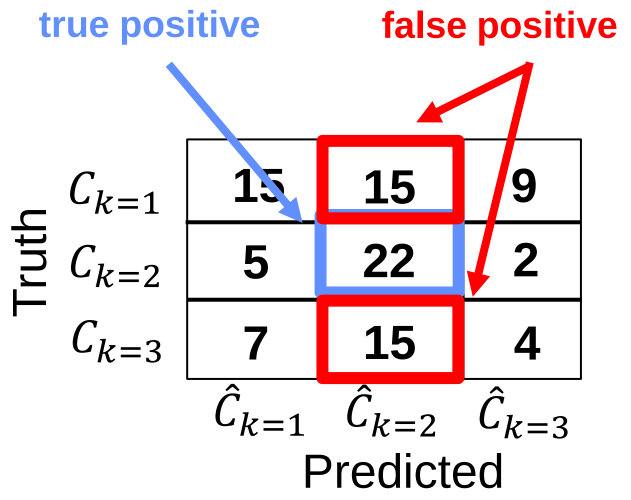


	$\hat{C}_{k=1}$	$\hat{C}_{k=2}$	$\hat{C}_{k=3}$
$C_{k=1}$	15	15	9
$C_{k=2}$	5	22	2
$C_{k=3}$	7	15	4

$precision_1$

$$\frac{15}{15 + (5+7)} = \frac{15}{27} = 0.56$$

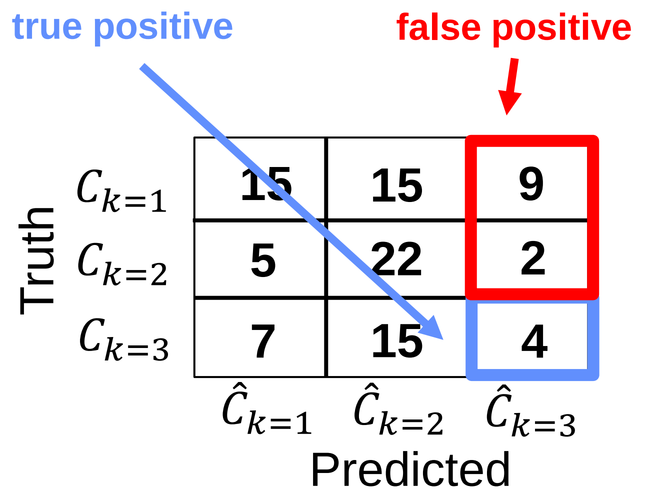
$\boldsymbol{k = 2}$



	$\hat{C}_{k=1}$	$\hat{C}_{k=2}$	$\hat{C}_{k=3}$
$C_{k=1}$	15	15	9
$C_{k=2}$	5	22	2
$C_{k=3}$	7	15	4

$$\frac{22}{22 + (15+15)} = \frac{22}{52} = 0.42$$

$\boldsymbol{k = 3}$



	$\hat{C}_{k=1}$	$\hat{C}_{k=2}$	$\hat{C}_{k=3}$
$C_{k=1}$	15	15	9
$C_{k=2}$	5	22	2
$C_{k=3}$	7	15	4

$precision_3$

$$\frac{4}{4 + (2+9)} = \frac{4}{15} = 0.27$$

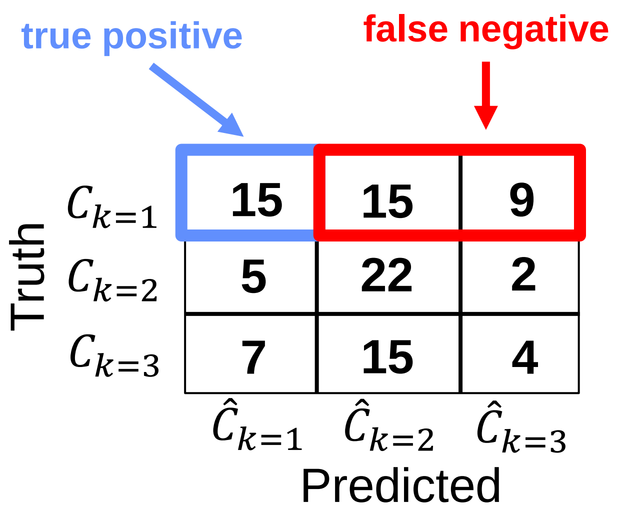
Classification Testing Metrics

Recall (called sensitivity in medical) – for group k, the ratio of true positive over all cases of k.

$$Recall_k = \frac{n_{k\ true\ positive}}{n_k = n_{k\ true\ positive} + n_{k\ false\ negative}}$$

How many of group k did we catch? Does not account for false positives.

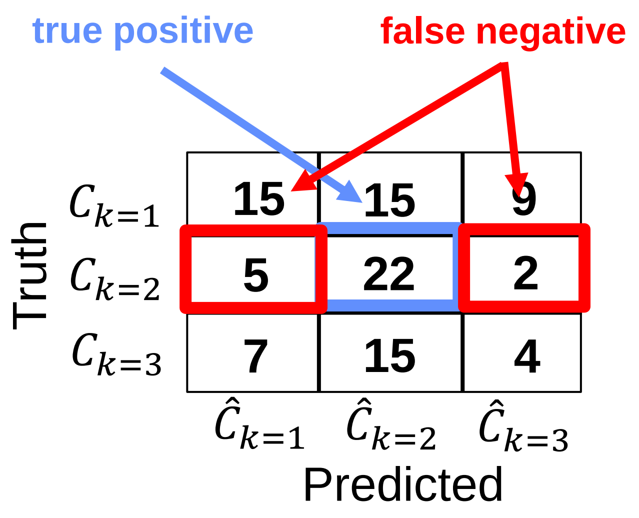
$\boldsymbol{k = 1}$



$$\begin{array}{c|ccc} & \hat{C}_{k=1} & \hat{C}_{k=2} & \hat{C}_{k=3} \\ \hline C_{k=1} & 15 & 15 & 9 \\ C_{k=2} & 5 & 22 & 2 \\ C_{k=3} & 7 & 15 & 4 \end{array}$$

$$\frac{15}{15+(15+2)} = \frac{15}{32} = 0.47$$

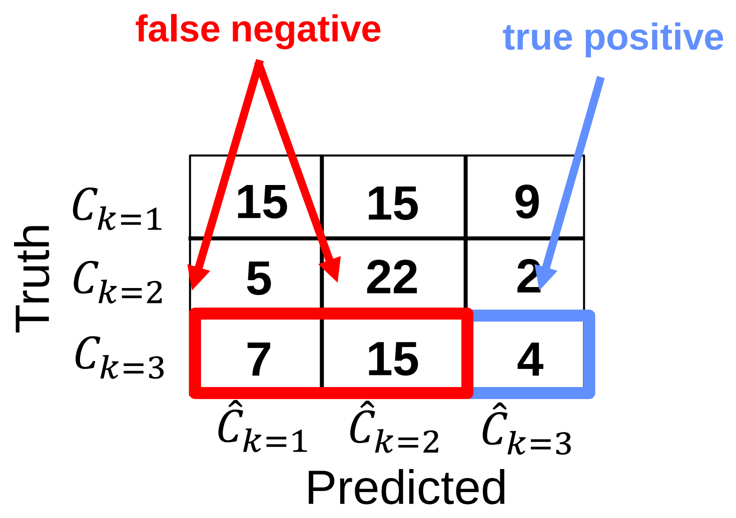
$\boldsymbol{k = 2}$



$$\begin{array}{c|ccc} & \hat{C}_{k=1} & \hat{C}_{k=2} & \hat{C}_{k=3} \\ \hline C_{k=1} & 15 & 15 & 9 \\ C_{k=2} & 5 & 22 & 2 \\ C_{k=3} & 7 & 15 & 4 \end{array}$$

$$\frac{22}{22+(5+9)} = \frac{22}{36} = 0.61$$

$\boldsymbol{k = 3}$



$$\begin{array}{c|ccc} & \hat{C}_{k=1} & \hat{C}_{k=2} & \hat{C}_{k=3} \\ \hline C_{k=1} & 15 & 15 & 9 \\ C_{k=2} & 5 & 22 & 2 \\ C_{k=3} & 7 & 15 & 4 \end{array}$$

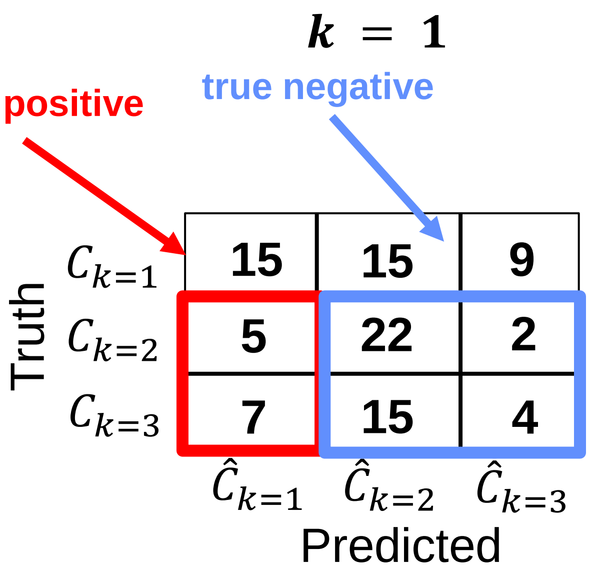
$$\frac{4}{4+(7+15)} = \frac{4}{26} = 0.15$$

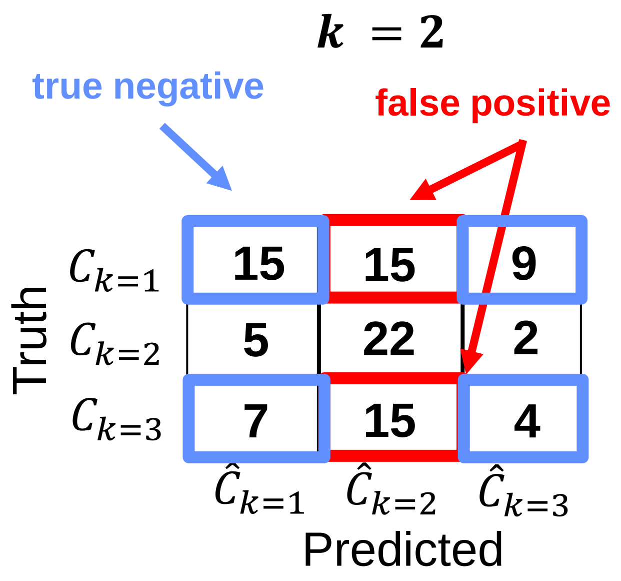
Classification Testing Metrics

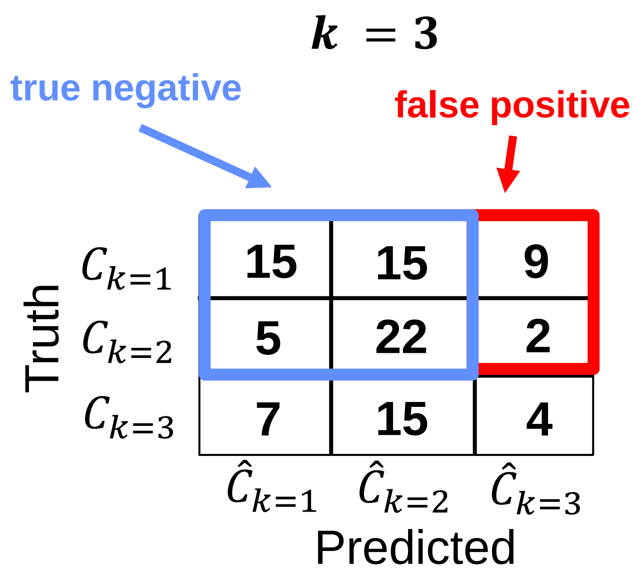
Specificity (true negative rate)

$$Specificity_k = \frac{n_{k\ true\ negative}}{\underbrace{n_{k\ true\ negative} + n_{k\ flase\ positive}}_{n_{\neq k}}}$$

How many of not group k did we catch? Does not account for true positives!







$$\frac{22+9+15+4}{(22+9+15+4)+(5+7)} = \frac{50}{62} = 0.81$$

$$\frac{15+2+7+4}{(15+2+7+4)+(15+15)} = \frac{28}{58} = 0.48$$

$$\frac{15+15+5+22}{(15+15+5+22)+(2+9)} = \frac{57}{68} = 0.84$$

Classification Testing Metrics

Precision and Recall measure 2 components of categorical accuracy. Let's combine them into one measure.

	Precision	Recall	f1-score
$k = 1$	0.56	0.47	0.51
$k = 2$	0.42	0.61	0.49
$k = 3$	0.27	0.15	0.19

$$f1 - score_k = \frac{2}{\frac{1}{Precision_k} + \frac{1}{Recall_k}}$$

$f1 - score$ is the Harmonic mean of precision and recall for k.

- Harmonic mean is sensitive the to lowest score, good performance in one score cannot average out / make up for bad performance in the other!

Classification Testing Metrics

Classification Testing Metrics Example

Another example from the naïve Bayes workflow.

- $$Precision_{k=Low} = \frac{n_{k\ true\ positive}}{n_{k\ true\ positive} + n_{k\ false\ positive}} = \frac{26}{26+1} = 0.96$$

- $$Recall_{k=Low} = \frac{n_{k\ true\ positive}}{n_k} = \frac{26}{26+2} = 0.93$$

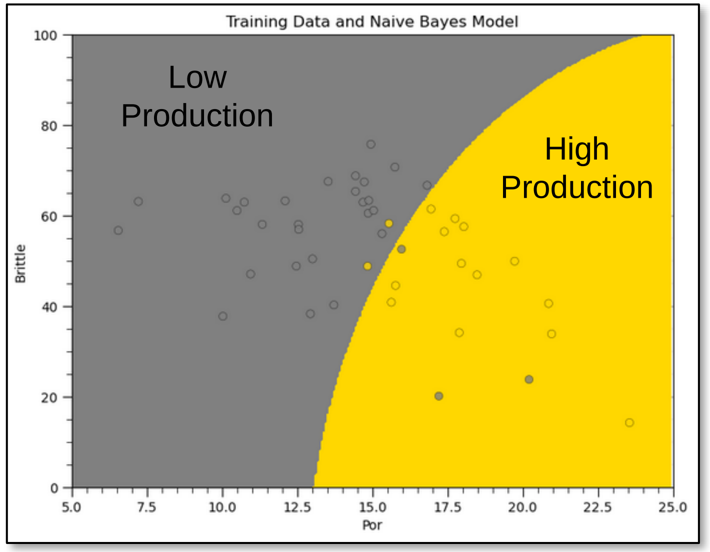
- $$f1 - score_{k=Low} = \frac{2}{\frac{1}{Precision_{k=Low}} + \frac{1}{Recall_{k=Low}}} = \frac{2}{\frac{1}{0.96} + \frac{1}{0.93}} = 0.95$$

true positive
k=low

$$\text{Truth}\quad \begin{array}{lcc} & & \\ \text{Low} & [[\boxed{26} & 2] \\ \text{High} & [\ \boxed{1} & 21]] \\ & \text{Low} & \text{High} \end{array}$$

Predicted

false positive
k=low



Naive Bayes classification model, from the naïve Byes chapter of Applied Machine Learning in Python e-book

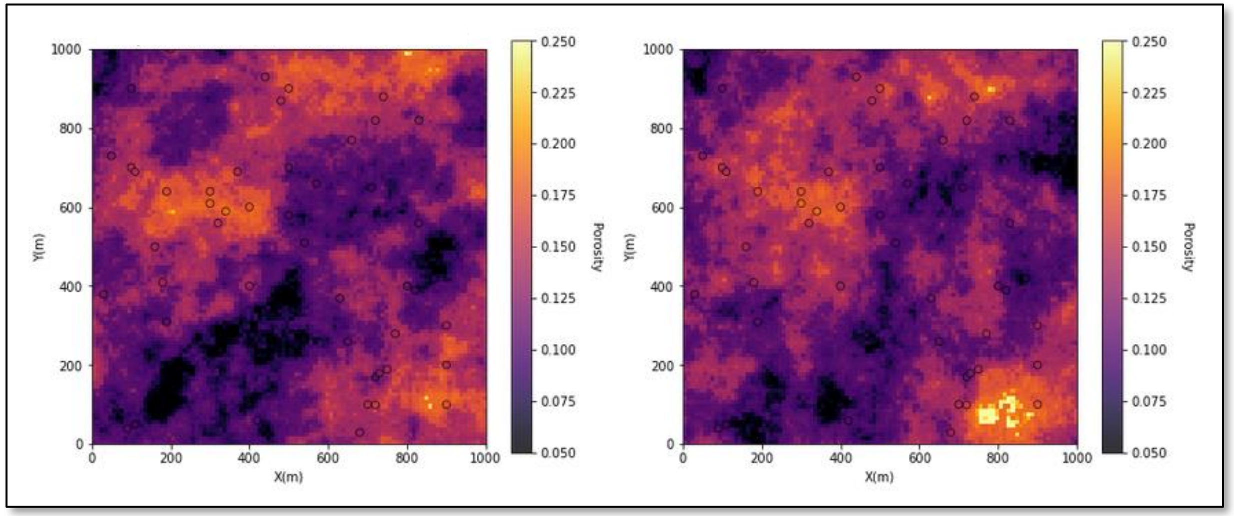
```
             precision    recall  f1-score   support

       Low        0.96      0.93      0.95        28
      High        0.91      0.95      0.93        22

avg / total       0.94      0.94      0.94        50
```

Image Model Metrics

What if we want to compare images?



Training image (left) and generated image (right).

Sometimes our model predictions are entire images, we may want to compare our generated images to original training images.

- We will cover methods such as auto encoders that project images to lower dimensional representations and generative adversarial networks that make new images.

Image Model Metrics

Pixel-by-pixel comparisons

MSE pixel-by-pixel

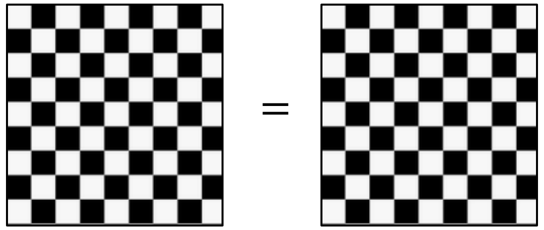
$$Test\ MSE = \frac{1}{n_x \cdot n_y} \sum_{iy=1}^{n_y} \sum_{ix=1}^{n_x} \left(y(\mathbf{u}_{ix,iy}) - \hat{y}(\mathbf{u}_{ix,iy}) \right)^2$$

Correlation pixel-by-pixel

$$\rho_{y,\hat{y}} = \frac{1}{n_x \cdot n_y \cdot \sigma_x \cdot \sigma_x} \sum_{iy=1}^{n_y} \sum_{ix=1}^{n_x} \left(y\left(\mathbf{u}_{ix,iy}\right) - \bar{y} \right) \cdot \left(\hat{y}\left(\mathbf{u}_{ix,iy}\right) - \bar{\hat{y}} \right)$$

- The pixel-by-pixel methods are very sensitive to local exactness.

Perfectly Same



Perfectly Different

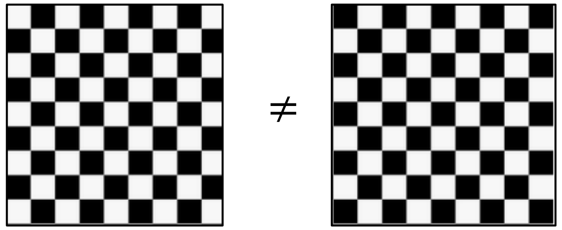


Image Model Metrics

Generalized image comparison with Structural Similarity Index Metric (SSIM)

Based on combination of 3 image comparisons:

1. Luminance (l) – similar average intensity
2. Contrast (c) – similar variance in intensity
3. Structures (s) – correlation between collocated pixels

$$l(a,b) = \frac{2\mu_a\mu_b + c_1}{\mu_a + \mu_a + c_1}$$

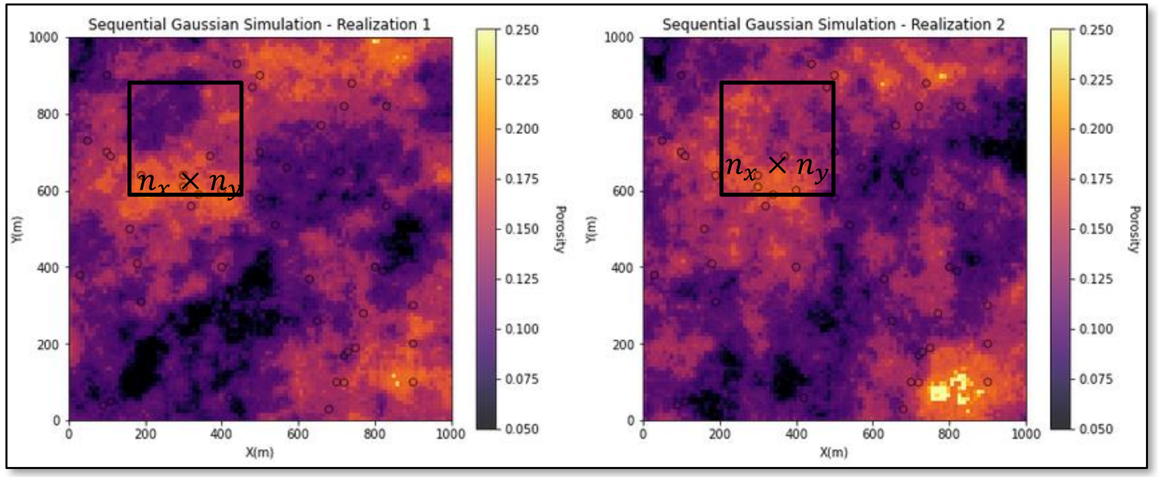
$$c(a,b) = \frac{2\sigma_a\sigma_b + c_2}{\sigma_a^2 + \sigma_b^2 + c_2}$$

$$s(a,b) = \frac{2\sigma_{a,b} + c_3}{\sigma_a\sigma_b + c_3}$$

$$\boldsymbol{SSIM(a,b)} = I(a,b)^{\boldsymbol{\alpha}} \cdot c(a,b)^{\boldsymbol{\beta}} \cdot s(a,b)^{\boldsymbol{\gamma}}$$

where α, β, γ are weights.

The calculation is aggregated our over multiple windows of the images:



Images for comparison.

Probability Distribution-based Metrics

What if we have calculated a probability with our model?

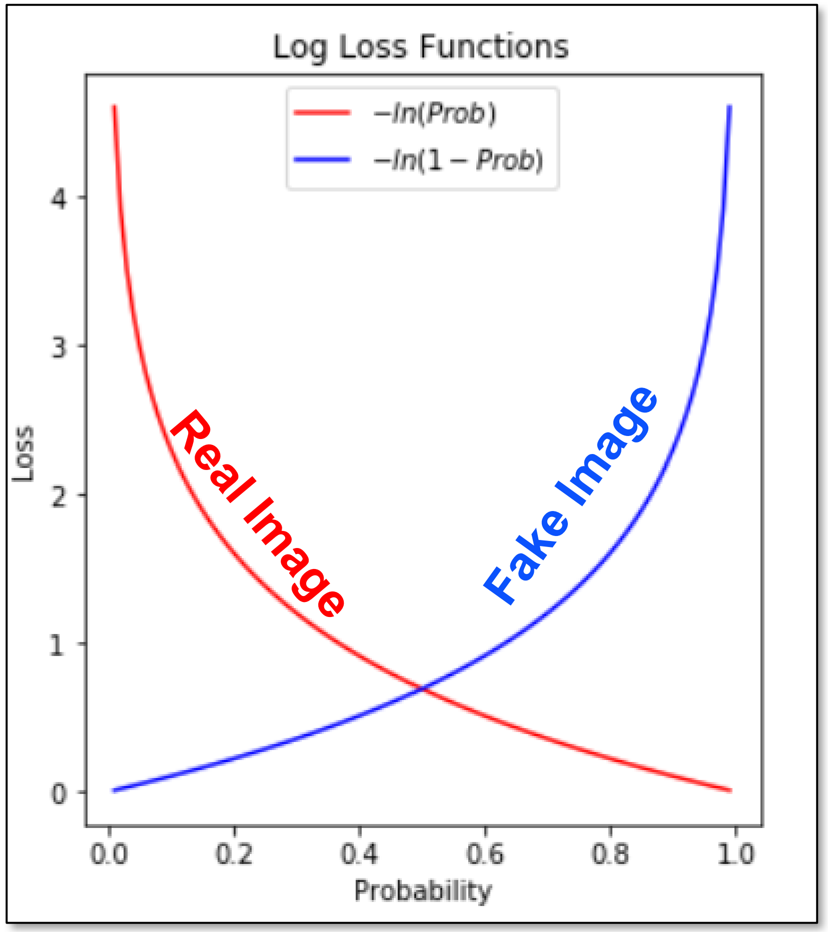
For example, in generative adversarial models, the discriminator has a goal to predict a probability of real image accurate.

- Probability = 0.0 if image is fake
- Probability = 1.0 if image is real

We use negative log-loss to convert from probability to loss for each case.

- By minimizing this we are actually minimizing the cross-entropy or Kullback-Leibler Divergence (more later).

- One hint for now, Kullback-Leibler Divergence (relative entropy) is statistical distance between distributions



Negative log loss functions to map probability to loss.

PGE 383 Subsurface Machine Learning

Lecture 10c: Tuning Hyperparameters

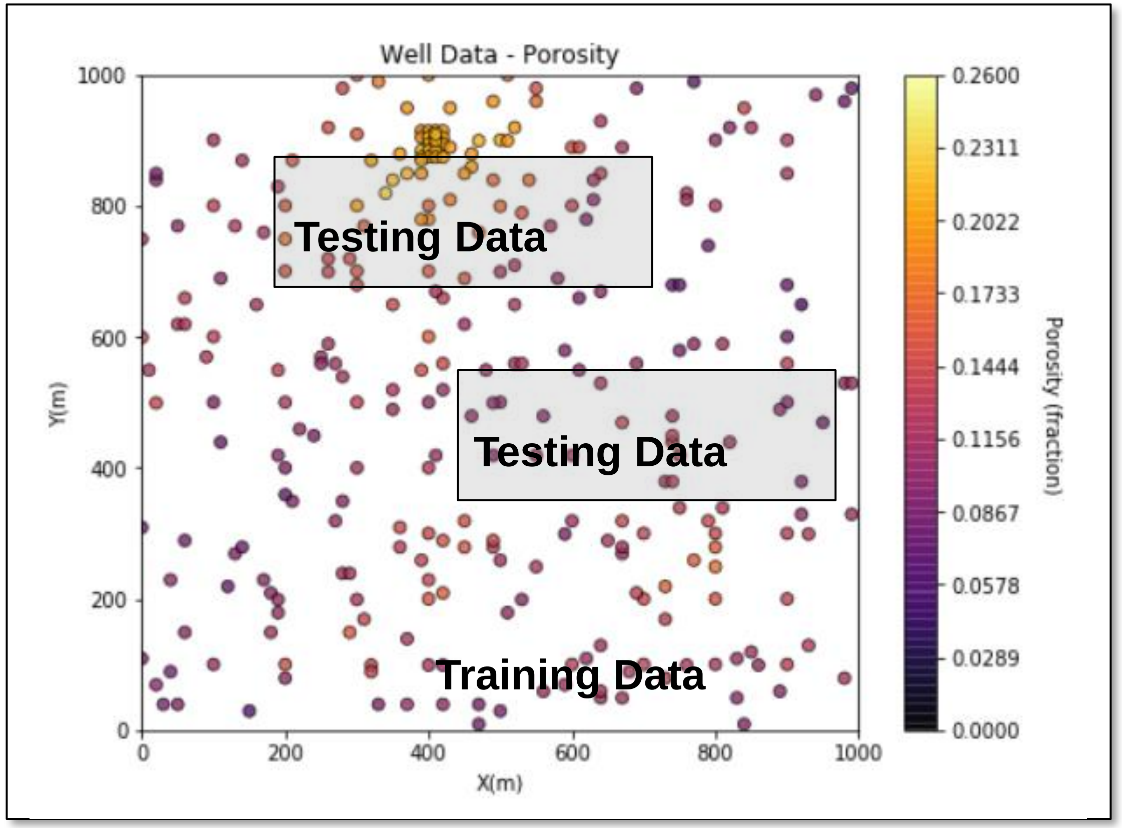
Lecture outline:

- **Cross Validation Workflows**

Model Cross Validation Workflows

Cross Validation – Hold Out Method - The Standard Train – Test Workflow

- Withhold subset of the data during model training
- Then testing the trained model with withheld subset dataset
- Sensitive to the selection of testing, must make sure cross validation is fair
- Training data set (used for training), testing data set (withheld for testing)



Predictions in a distinctly different range of reservoir values.

Model Cross Validation Workflows

Leave-one-out Cross Validation (LOO CV)

- An exhaustive cross validation method.
- Combines training and testing into one step

Loop over all data, withhold that data

- Train on $n - 1$ data and test on the withheld single data
- Calculate model goodness metric
- Aggregate accuracy norm (single error) over all data, n
- Typically, too easy of an estimation problem
- K-fold is a more general and robust approach, this method assumes $\mathrm{K} = n$.

Model Cross Validation Workflows

K-fold Cross Validation (K-fold CV)

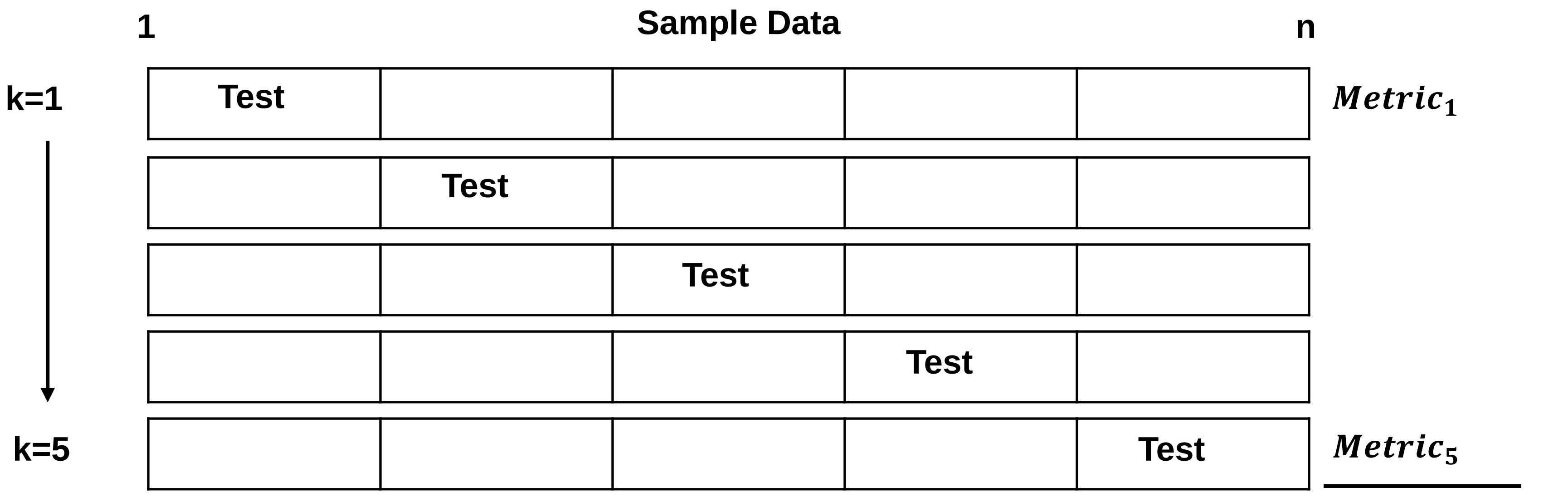
A exhaustive cross validation approach (all data are tested), but it samples a limited set of prediction problems

- Select K, integer number of folds, impacts the size of testing set! Break data set into K subsets, equal size n/k

Loop over k subsets:

- use data outside the k subset to predict inside the k subset
- calculate the model goodness metric

Aggregate accuracy norms over all subsets, k



Cycle test over k folds, all other data are train.

$Metric\ Aggregation$

Model Cross Validation Workflows

Leave-p-out Cross Validation (LpO-CV)

- An exhaustive cross validation approach.
- For any p extract all possible p sized testing data sets.
- Train and calculate the accuracy norm with the withheld testing data
- Summarize over all possible p sized testing data sets.
- The possible number of datasets is $\binom{n}{p}$, n choose p. This is the binomial coefficient.

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}, \text{ where } n > p$$

- Aggregate accuracy norm over all combinations.

Model Cross Validation Workflows Limitations

Commonly Used Cross Validation Methods in the Python scikit-learn package,

sklearn.model_selection.train_test_split(X,y,train_size,random_state) – hold out method

- random selection for training and testing
- specify train_size or test size, fraction or number
- use random_state for repeatability
- stratify will enforce whole sample statistics in the train and test sample subsets

sklearn.cross_val_score(model_object,X,y,cv,scoring) – k-fold cross validation

- wrapper that performs k-fold cross validation with k random subsets
- k is the cv parameter
- scoring allows assignment of a custom model accuracy metric
- you must aggregate the k testing metrics

Model Cross Validation Workflows Limitations

Methods in the Python scikit-learn package,

sklearn.cross_validate(model_object,X,y,cv,scoring) – k-fold validation with more detailed results

- wrapper that performs k-fold cross validation with k random subsets
- k is the cv parameter
- scoring allows assignment of a custom model in training accuracy metric
- scoring parameter may be a list of model accuracy metrics
- outputs include fit and scoring times, estimates and all scores in a dictionary object

Model Cross Validation Workflows Limitations

Issues with Cross Validation

Peeking, information leakage – some information is transmitted from the withheld data into the model, some model decision(s) use all the data. Pipelines and wrappers help with this.

Fair Train and Test Split – many practitioners use random selection for train and test split (we use it, it is built into scikit-learn) and this may be too easy of a prediction problem

Black Swans / Stationarity – the model cannot be tested for data events not available in the data

This is also known as the ‘**No Free Lunch Theorem**’ in machine learning

‘even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience’ - Hume (1739–1740)

Model Cross Validation Workflows Limitations

Issues with Cross Validation

Subsurface Validation – it is not possible to validate open earth systems – Oreskes et al., 1994. Here's the abstract from their paper:

'Verification and validation of numerical models of natural systems is impossible. This is because natural systems are never closed and because model results are always nonunique. Models can be confirmed by the demonstration of agreement between observation and prediction, but confirmation is inherently partial. Complete confirmation is logically precluded by the fallacy of affirming the consequent and by incomplete access to natural phenomena. Models can only be evaluated in relative terms, and their predictive value is always open to question. The primary value of models is heuristic.'

Abstract from https://science.sciencemag.org/content/263/5147/641

Model Cross Validation Workflows Limitations

Issues with Cross Validation

‘All models are wrong, but some are useful’ – George Box

Parsimony – since all models are wrong, an economical description of the system. Occam’s Razor

Worrying Selectively – since all models are wrong, figure out what is most importantly wrong.

‘Be humble, the earth will surprise you!’ – Michael Pyrcz

Concepts from https://en.wikipedia.org/wiki/All_models_are_wrong

PGE 383 Subsurface Machine Learning

Lecture 10c: Tuning Hyperparameters

Lecture outline:

- **Training and Testing**
- **Model Goodness Metrics**
- **Cross Validation Workflows**