Cross Validation

STOR 320-002 Introduction to Data Science

Fall 2024 - Mo Liu

Review from last lecture

Questions:

- In PCA, when the number of PC gets larger, does the model become more flexible or less flexible?

- In LASSO or Ridge regression, when λ gets lager, does the model become more flexible or less flexible?

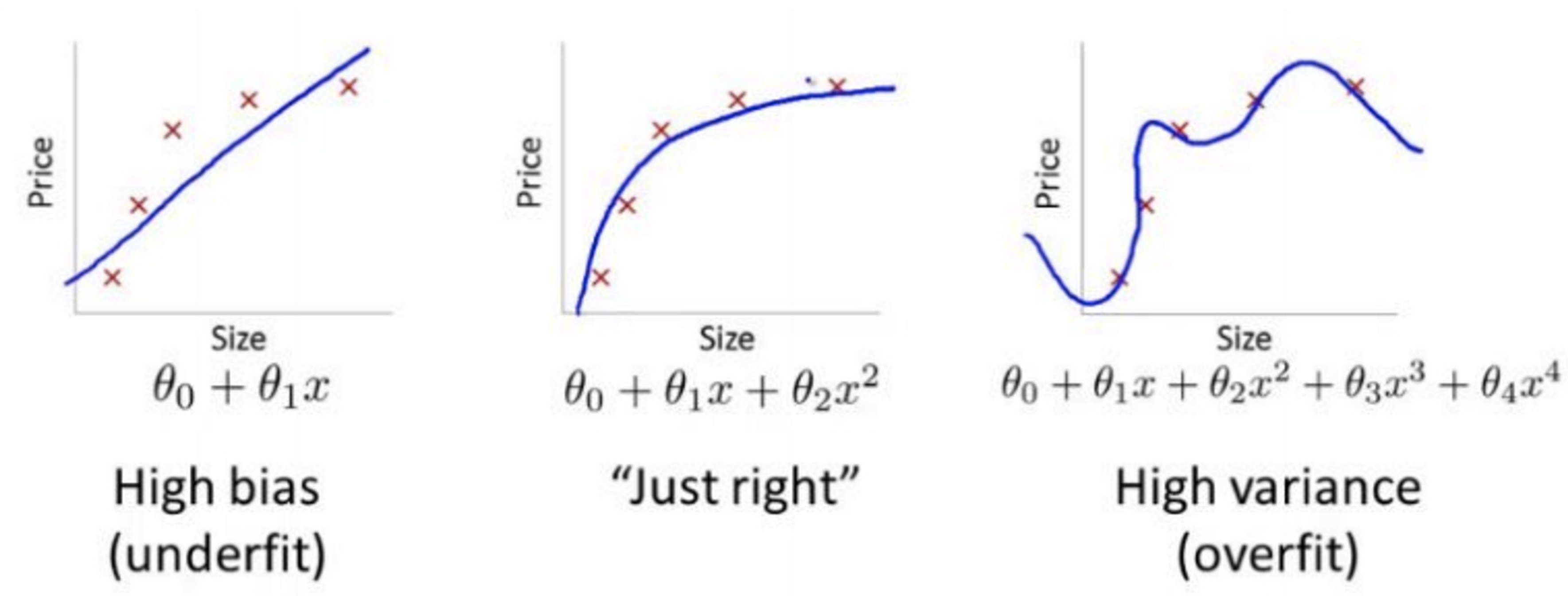
Hyperparameters of models

- PCA: Number of PC
- LASSO or Ridge: scale of regularization parameter

Overfitting

- Overfitting occurs when the estimated model fits the noise in the training data

- All statistical learning methods are at risk for overfitting



Example of LASSO

- What is the R-squared (R2) and out-of-sample R-squared (OSR2) when λ equals the following values?

  - 10^{-6}
  - 10^{-5}
  - 10^{-4}
  - 10^{-3}
  - 10^{-2}
  - 10^{-1}

+ Overfitting

- Tell-tale sign of overfitting is when a model performs very well on the training set in a way that will not translate to good performance on new data
- Overfitted models are often more complicated and do not necessarily accord with intuition/judgment
- Overfitted models often capture idiosyncrasies or errors in the data instead of capturing actual relationships that will hold for new data

+ Overfitting

- Overfitting is possible due to differences between how models are fitted and how they are used
  - Model fitting seeks to maximize accuracy of the model on a known set of training data
  - Models are used to make predictions on new, previously unseen data
- Care must be taken to make sure that the model we estimate does not suffer from overfitting
- How to find the best value of hyperparameters?

Grid Search for λ

- Generate a candidate set of λ value for LASSO: exponential of {-5, -4, -3, -2, - 1, 0}

- Evaluate the performance at each candidate λ value
  - How to evaluate the performance for λ?
  - Can we directly look at OSR2?

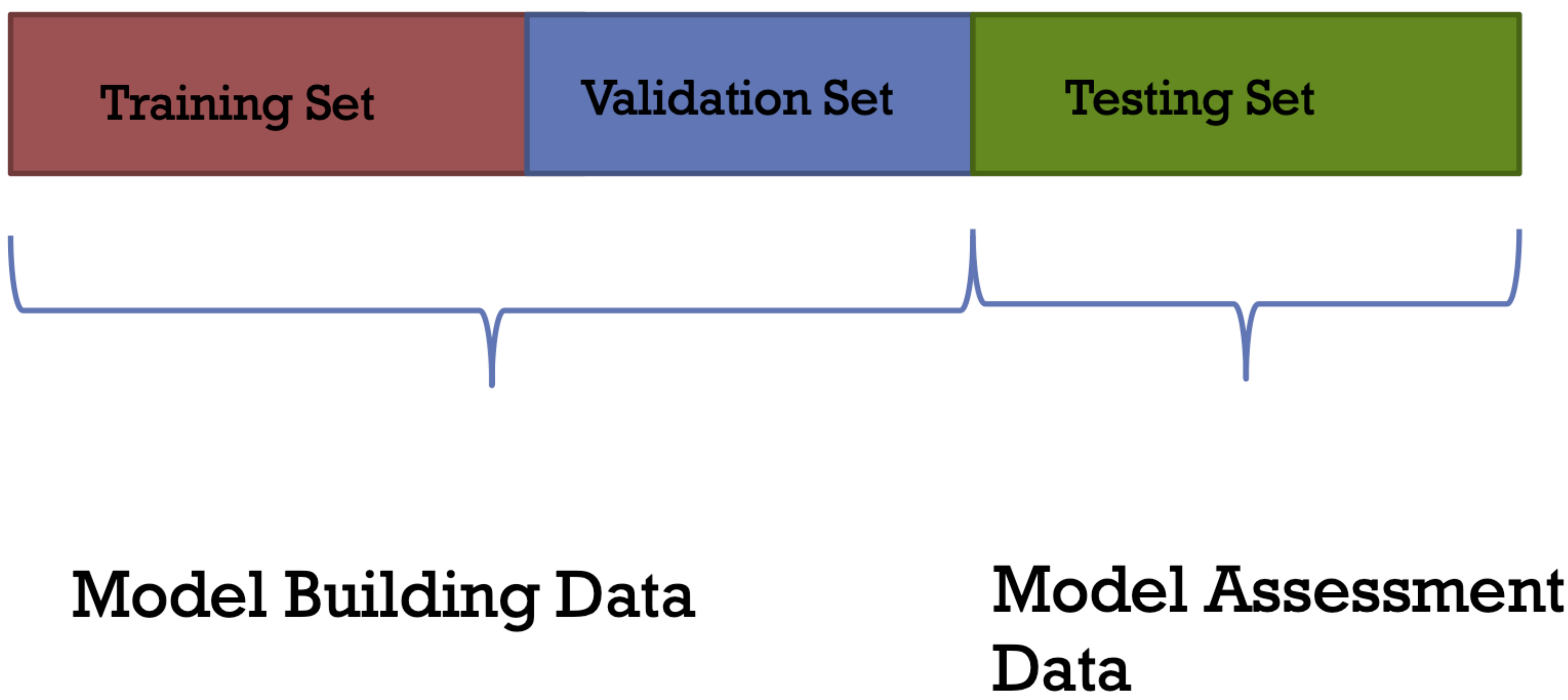
Training, Test, and Validation Sets

- Recall our usual split into training and test set:
  - Training set is used to build the model
  - Test set is used to asses its performance
- Question: can we estimate test set aka out of sample performance during the training phase *without* touching the test set data?
- Answer: yes! The simplest way to is to create a "virtual" test set called the validation set

Validation Set Approach

- Step 1: Split into 3 sets – training, validation, and test
- Step 2: Build a sequence of models using the training data (for example, using different λ values in LASSO)
- Step 3: Evaluate each model's performance (e.g. R2, accuracy, TPR, FPR, …) on the validation set
- Step 4: Pick the best model and use the test set to estimate future real-world performance of the model
- (By the way, are you "allowed" to iterate steps 2 and 3?)

+ Training, Test, and Validation Sets



+ Validation Set Approach

- You've probably already sort of used the validation set approach
- For example, if you've looked at the two or more OSR2 values (however that was test data)
- Best practice is to keep the test set in a vault, although it's okay to look at the performance of a handful of models
- This becomes dangerous when we want to check many different models

Advantages and Disadvantages of the Validation Set Approach

- Advantages:
  - Conceptually simple
  - Easy to implement
- Disadvantages
  - Estimates tend to be sensitive to the results of the random number generator
  - "Waste of data" – especially troublesome if n is small

Leave-One-Out Cross-Validation (LOOCV)

- As an example, suppose that we want to estimate the error rate (1 minus the accuracy) of a CART classification model

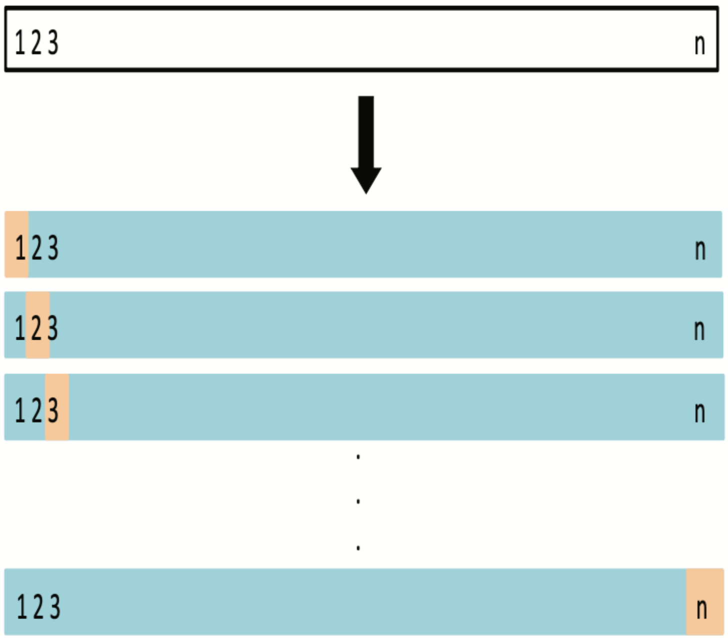
- Idea:
  - Take the training set of size *n* and remove observation *i*
  - Build the model on the remaining *n – 1* observations
  - Check if we made an accurate prediction $\hat{y}_i$ on the *i*th example that we held out:

$$\mathrm{Err}_i = \begin{cases} 1 & \hat{y}_i \neq y_i \\ 0 & \hat{y}_i = y_i \end{cases}$$

+ Leave-One-Out Cross-Validation (LOOCV)

- LOOCV says to repeat the removal, training process and error calculation for **all** *n* data samples and use their average to estimate the out of sample performance:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Err}_i$$



Leave-One-Out Cross-Validation (LOOCV)

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{Err}_i$$

- Repeat the process of calculating $\mathrm{CV}_{(n)}$ for each potential model
- Compare the results as before and pick the best model

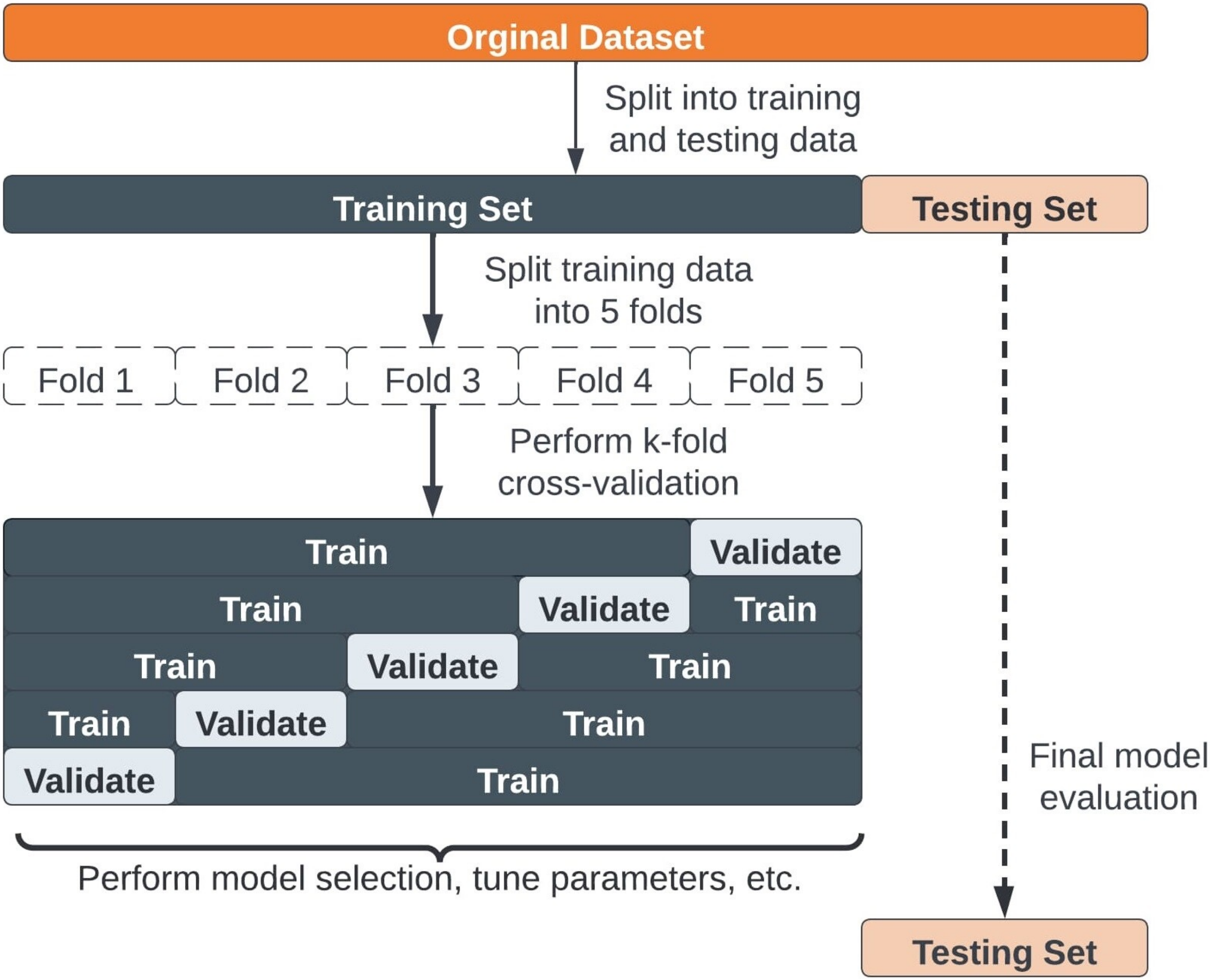
+ Advantages and Disadvantages of LOOCV

- Advantages:
  - The result is entirely deterministic given the training data (no random number generation)
  - Estimates tends to be more accurate "on average" (less bias)

- Disadvantages
  - Computationally intensive: requires training *n* * *(number of candidate models)* different models
  - Results within each $CV_{(n)}$ calculation are correlated – averaging correlated numbers may not reduce variance

+ A Compromise: k-fold Cross-Validation

- Step 1: Divide the training data into k different groups (e.g., k = 5 or 10)
- Step 2:
  - Remove group 1 and train on the remaining k-1 groups
  - Compute the average error on the held out group 1, Err_1
  - Repeat for group 2, group 3, …, group k
- Step 3:
  - Average the results: $$\text{CV}_{(k)} = \frac{1}{k}\sum_{j=1}^{k} \text{Err}_j$$
- Repeat steps 2-3 for each candidate model


Orginal Dataset
Split into training and testing data
Training Set
Testing Set
Split training data into 5 folds
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Perform k-fold cross-validation
Train
Validate
Train
Validate
Train
Train
Validate
Train
Train
Validate
Train
Validate
Train
Perform model selection, tune parameters, etc.
Final model evaluation
Testing Set

k-fold Cross-Validation

- Final step: After selecting your parameters via k-fold cross validation, retrain on the entire dataset using those parameters
- Some more details:
  - $k = 5$ or 10 are commonly used and tend to be good choices
  - For regression, use the MSE (aka RSS) or R2 as the performance metric
- When $k = 1$, it becomes LOOCV

Which methods should we use?

- Validation set approach is better when:
  - The sample size is small
  - The model is time-consuming to build
  - For example, deep learning model

- K-fold cross validation is better when:
  - The sample size is large
  - The model is easy to build

k-fold Cross-Validation in Python

- from sklearn.model_selection import GridSearchCV
- alpha_grid = {'alpha': np.logspace(-8, -1, num=50, base=10)}
- lasso_cv = GridSearchCV(model, param_grid = alpha_grid, scoring='neg_mean_squared_error', cv=10, verbose=1)
- lasso_cv.fit(X_train_lasso, y_train)

GridSearchCV, scoring

- GridSearchCV(model, param_grid =, scoring=, cv=10, verbose=1)

Regression	
'explained_variance'	metrics.explained_variance_score
'neg_max_error'	metrics.max_error
'neg_mean_absolute_error'	metrics.mean_absolute_error
'neg_mean_squared_error'	metrics.mean_squared_error
'neg_root_mean_squared_error'	metrics.root_mean_squared_error
'neg_mean_squared_log_error'	metrics.mean_squared_log_error
'neg_root_mean_squared_log_error'	metrics.root_mean_squared_log_error
'neg_median_absolute_error'	metrics.median_absolute_error
'r2'	metrics.r2_score
'neg_mean_poisson_deviance'	metrics.mean_poisson_deviance
'neg_mean_gamma_deviance'	metrics.mean_gamma_deviance
'neg_mean_absolute_percentage_error'	metrics.mean_absolute_percentage_error
'd2_absolute_error_score'	metrics.d2_absolute_error_score

GridSearchCV, cv

- GridSearchCV(model, param_grid =, scoring='neg_mean_squared_error', cv=10, verbose=1)

cv : *int, cross-validation generator or an iterable, default=None*

Determines the cross-validation splitting strategy. Possible inputs for cv are:

- None, to use the default 5-fold cross validation,
- integer, to specify the number of folds in a `(Stratified)KFold`,
- CV splitter,
- An iterable yielding (train, test) splits as arrays of indices.

For integer/None inputs, if the estimator is a classifier and `y` is either binary or multiclass, `StratifiedKFold` is used. In all other cases, `KFold` is used. These splitters are instantiated with `shuffle=False` so the splits will be the same across calls.

Refer User Guide for the various cross-validation strategies that can be used here.

Randomly split k-fold sets

- By default, there is no shuffling in Kfold

`sklearn.model_selection`.KFold

class `sklearn.model_selection.`**`KFold`**(*n_splits=5*, ***, *shuffle=False*, *random_state=None*) [source]

K-Folds cross-validator

Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default).

Each fold is then used once as a validation while the k - 1 remaining folds form the training set.

Read more in the User Guide.

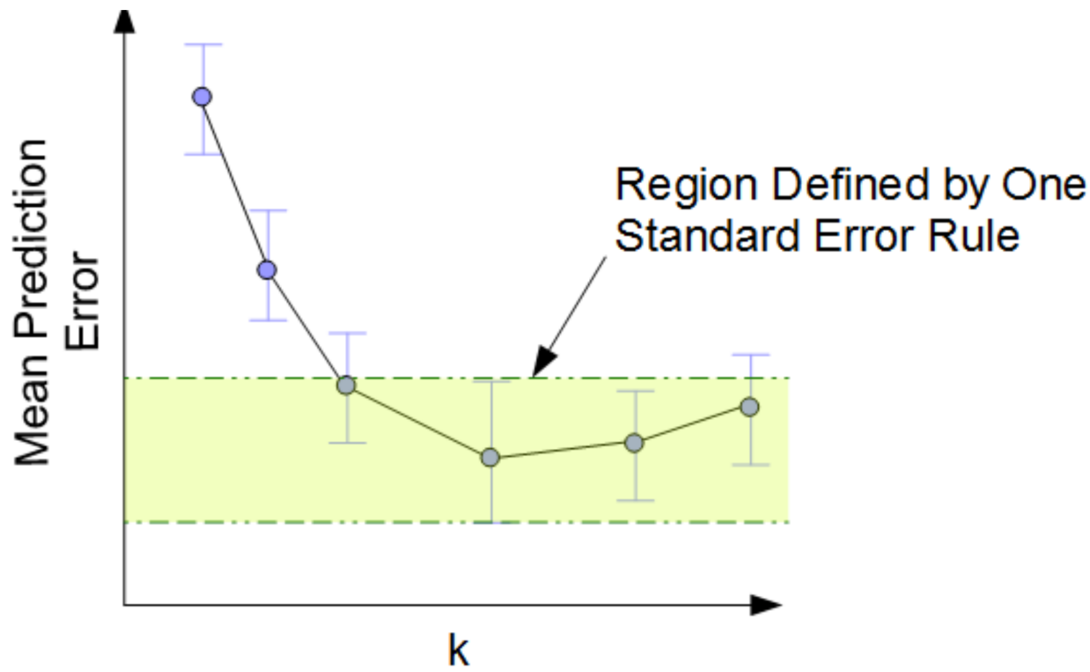
- What if the training set is sorted in some way, for example, by time?

- from sklearn.model_selection import GridSearchCV
- from sklearn.model_selection import KFold

- alpha_grid = {'alpha': np.logspace(-8, -1, num=15, base=10)}
- cv = KFold(n_splits = 10, random_state = 1, shuffle = True)
- lasso_cv = GridSearchCV(lasso, param_grid = alpha_grid, scoring='neg_mean_squared_error', cv=cv, verbose=1)
- lasso_cv.fit(X_train_lasso, y_train)

Selecting Parameters via Cross-Validation

- One Standard Error Rule
  - Find model with minimum error
  - Select the simplest model whose mean falls within 1 standard deviation of the minimum



Custom loss function

- MAE: mean absolute error
- What if we only concern about the case when the prediction error is larger than $2,000?
- In other words, we want to minimize the number of predictions whose error is larger than $2,000?

Acknowledgement

- The figure of 5-fold cross validation is taken from https://www.aptech.com/blog/understanding-cross-validation/