University of Pittsburgh

ECE 2195: Special Topics – Computers Machine Learning

Cross Validation – Model Selection

Mai Abdelhakim, PhD
ECE Department
Swanson School of Engineering
University of Pittsburgh
maia@pitt.edu

Many Options

- How to decide which machine learning algorithm to use?
- Which features to use?
    - Add more features
    - Reduce number of features
    - What degree of polynomial or interaction term
        - Include x^2, x_1x_2 …
- How to find best tuning parameters (such as tuning parameter for regularization)

Model and Parameter Selection

- Do we use the training set accuracy to select the model?
  - NO
- Do we use the test set accuracy to select the model?
  - Example: Suppose we want to find the best regularization parameter to use in ridge regression, we try tuning parameter alpha =0.1, 1, 100, 1000
  - Can we do the following?

```
train test split
for C in [ 0.1, 1, 100, 1000]
   fit the model with x_train, y_train
   find accuracy with x_test, y_test
```

    Can we use C that got the highest accuracy on test data?

Model and Parameter Selection

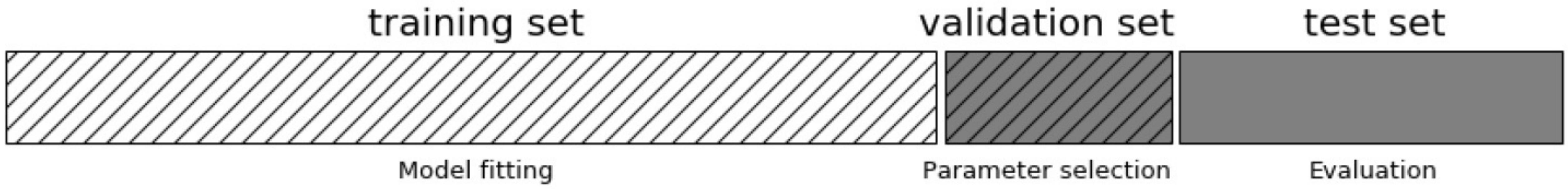
- Do we use the training set accuracy to select the model?
  - NO
- Do we use the test set accuracy to select the model?
  - Example: Suppose we want to find the best regularization parameter to use in ridge regression, we try tuning parameter alpha =0.1, 1, 100, 1000
  - Can we do the following?

```
train test split
for C in [ 0.1, 1, 100, 1000]
   fit the model with x_train, y_train
   find accuracy with x_test, y_test
Can we use C that got the highest accuracy on test data?
```

WRONG: test data is only for evaluation... should not be used to select the model

Solution: 3 Splits of Data Instead of 2

- If we used the test set accuracy to tune a model, then it cannot be used to access the model accuracy
  - Would provide unfair evaluation!
- Solution: Have 3 splits of the data
  - One set for **training**,
  - One set for model selection (called **validation** set or **cross-validation set**)
  - One set for **testing**



To Get Three Splits in Python

- Use train test split twice:

 # First split into two, one test and another one which includes both (train and validation)

X_trainval, **X_test**, **Y_trainval**, **Y_test** = train_test_split(

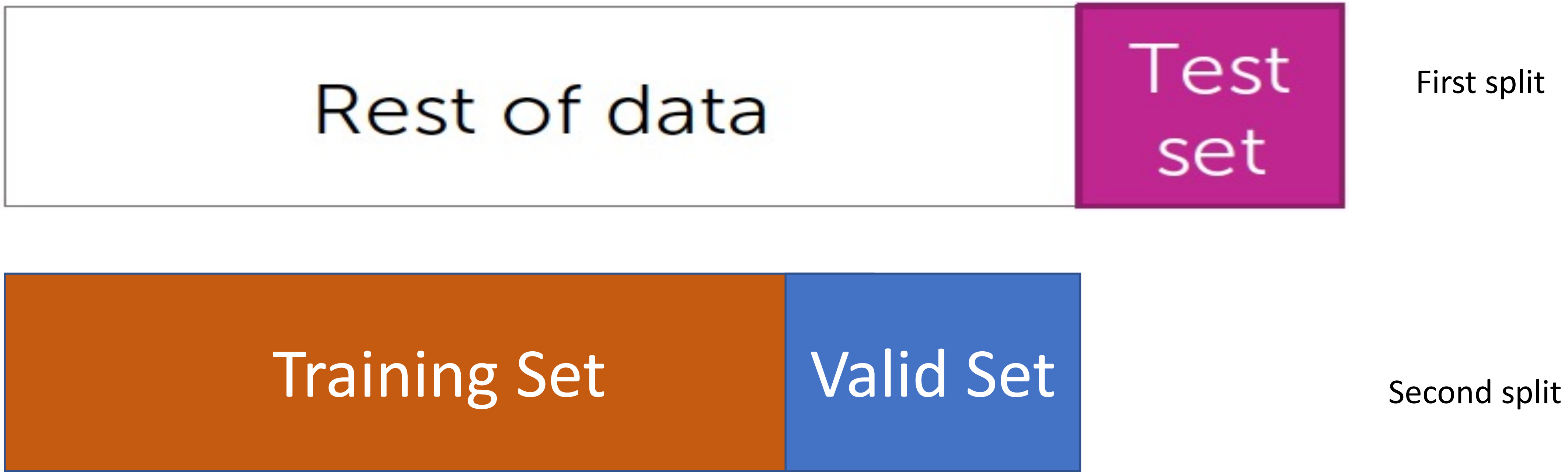
iris.data, iris.target, random_state=0)

 # split train and validation set into training set and validation set

X_train, X_valid, Y_train, Y_valid = train_test_split(

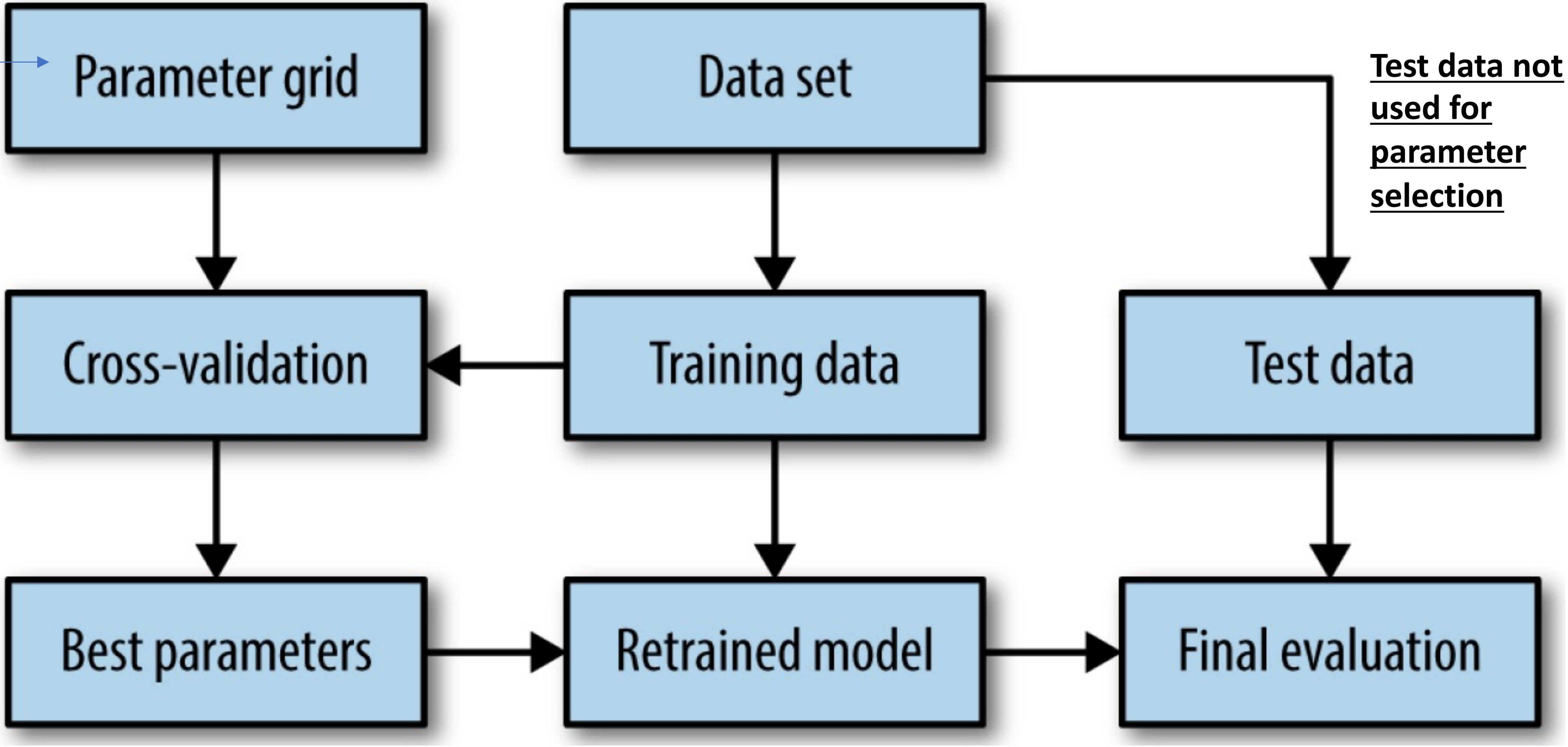
X_trainval, **y_trainval**, random_state=0)

Search with Cross Validation



Parameter Selection : Typical Workflow

Values for parameters we want to check



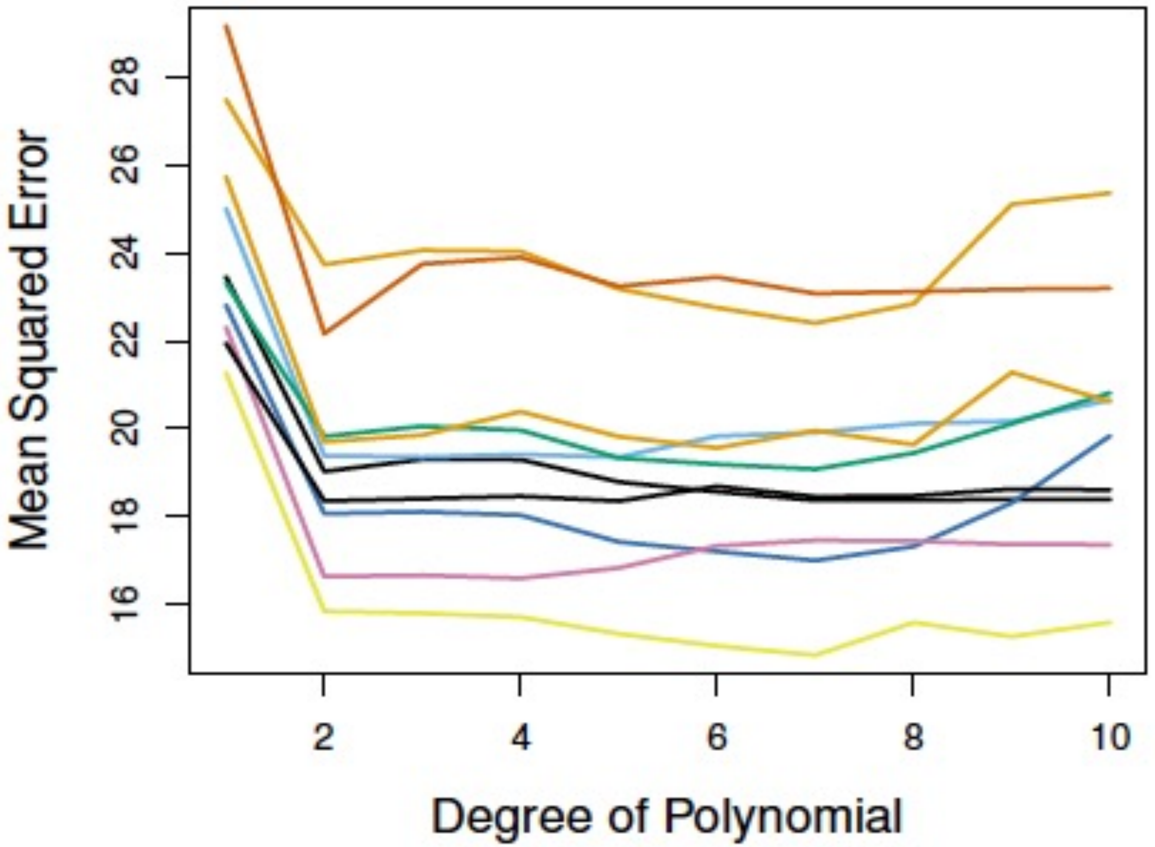
<u>Test data not used for parameter selection</u>

Lots of variability depending on the split

- There could be a lot of variability in the performance depending on the split of the data

Example for Illustration: MSE of Auto Data

- Assume we want to know what degree of polynomial we should use in linear regression model for the auto data set (miles per gallon vs horsepower)
- Each random split can result in a different performance
  - Try using different random state for splitting and get the accuracy!

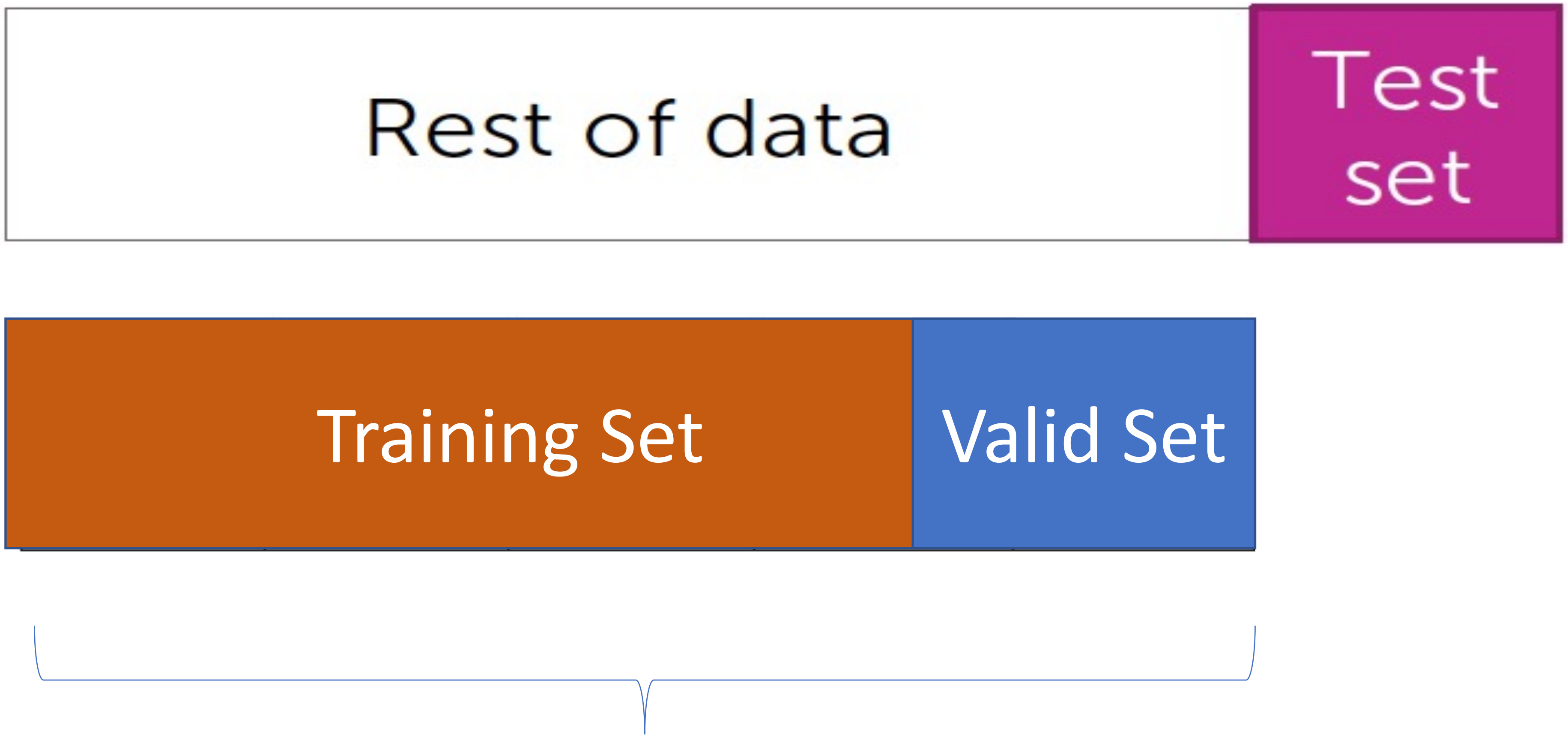


Each curve is obtained using a different random split of the data.. Then get MSE on validation set

Solution: K-fold Cross Validation

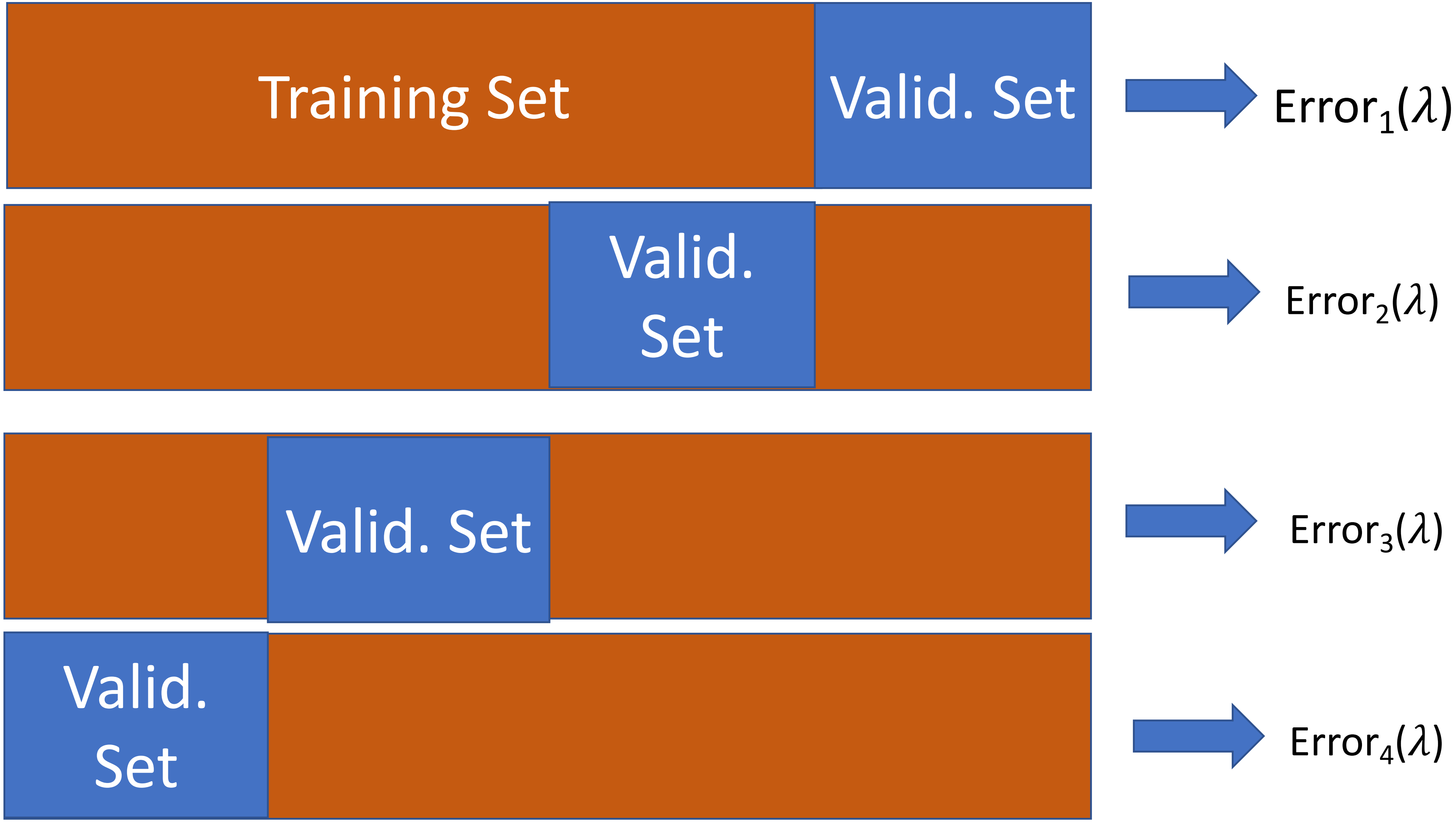
- More stable method of evaluating performance of machine learning algorithms
- K-fold Cross validation: repeatedly split the data into different train and validation sets
  - Get parameters based on best average accuracy over all the splits
- In 4-fold cross validation, we have 4 different splits of the data

Search with Cross Validation



Use K-fold cross validation to get more stable evaluation on the validation set then choose parameters accordingly

Example: Get Tuning Parameter λ for Regularization, with 4-fold Cross Validation



Cross Validation for Model Selection – Finding Regularization Parameter

- Repeat for different parameters/models
  - In each get the accuracy of the k-fold
- Choose parameter (e.g. λ) that **minimize average error** over the K folds:
  - Average error $= \frac{1}{K} \sum_{j=1}^{K} \text{Error}_j\ (\lambda)$

Cross-Validation in Python

from **sklearn.model_selection** import **cross_val_score**
from **sklearn.linear_model** import **Ridge**
RegModel = Ridge(alpha=c)
scores = **cross_val_score**(RegModel, X_trainval, Y_trainval, cv=5) ➔ Get vector of scores for the different splits (the function does the splits internally)

Model we want to evaluate | features | True labels | Number of folds

Example: for k fold cross validation the output can be : [0.95, 0.90, 0.93, 0.89, 0.91]

scores.mean() ➔ get the average score of all splits, R_squared is default

Cross Validation for Accurate Model Evaluation

- If we don’t want to do parameter selection, but want to get more accurate measure of performance, we can use k-fold cross validation on train and test sets.

- In some cases, the most difficult observations are in training set only and test set has all observations that can be easily predicted
    - Very good performance on test data

- You may be unlucky when all hard-to-predict examples are only in test set
    - Poor performance, as model may not be well-trained

K-Fold Cross-Validation on Train and Testing for More Stable Result of a Model Evaluation

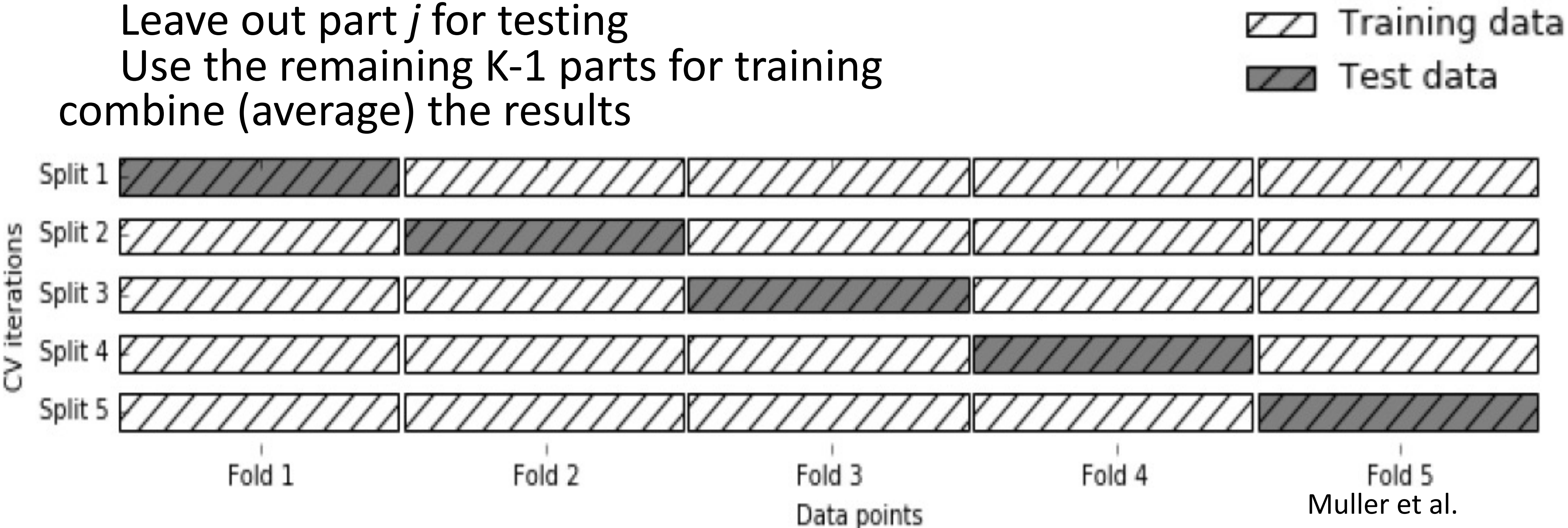
- Remember: here we are not selecting model, but we are finding more stable measure for accuracy .. So we have two sets (train and test)
- Same steps as before: Randomly divide data into K parks (1,2,3..K), then

  For j =1 ,2 …K

      Leave out part *j* for testing

      Use the remaining K-1 parts for training

  combine (average) the results



Muller et al.

Overall Performance

- In a regression setting, if MSE_j for fold j (in split j), then overall accuracy is

$$MSE = \frac{1}{K}\sum_{j=1}^{K} MSE_j$$

- Same concept is applied in classification
  - Accuracy of 3-fold CV is: [0.95, 0.90, 1], what is the overall accuracy?

Leave One out Cross Validation

- Leave one out cross validation is a special case of K-fold CV when each fold has one sample for test
  - Each time you test the performance using a single observation

- **Python:**

from **sklearn.model_selection** import **LeaveOneOut**
loo = LeaveOneOut()
scores = cross_val_score(Model, Features, Target, **cv=loo**)

n points, training has n-1 points and test has one observation point