

# INFSCI 2915: Machine Learning

## Cross Validation – Model Selection

Mai Abdelhakim

School of Computing and Information

610 IS Building

Spring 2018

# Many Options

- How to decide which machine learning algorithm to use?
- Which features to use?
  - Add more features
  - Use less number of features
  - What degree of polynomial or interaction term
    - Include  $x^2, x_1x_2 \dots$
- How to find best tuning parameters (such as tuning parameter for regularization)
- Reading:
  - Chapter 5 textbook,
  - Chapter 5, Introduction to Machine learning with Python

# 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 logistic regression, we try  $C=0.1, 1, 100, 1000$
  - 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?

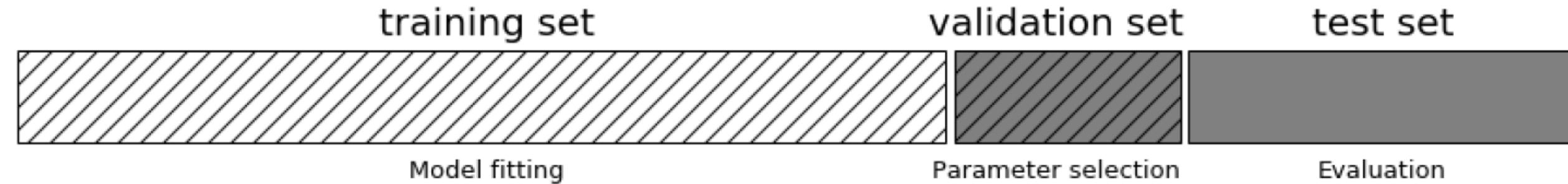
# Model and Parameter Selection

- 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 logistic regression, we try  $C=0.1, 1, 100, 1000$
  - We can say:  
~~train test split (or k-fold)~~  
~~for  $C \in [0.1, 1, 100, 1000]$~~   
~~fit the model with  $x_{\text{train}}, y_{\text{train}}$~~   
~~find accuracy with  $x_{\text{test}}, y_{\text{test}}$~~   
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 we cannot use it to access the model accuracy as this will give unfair results!
- 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:

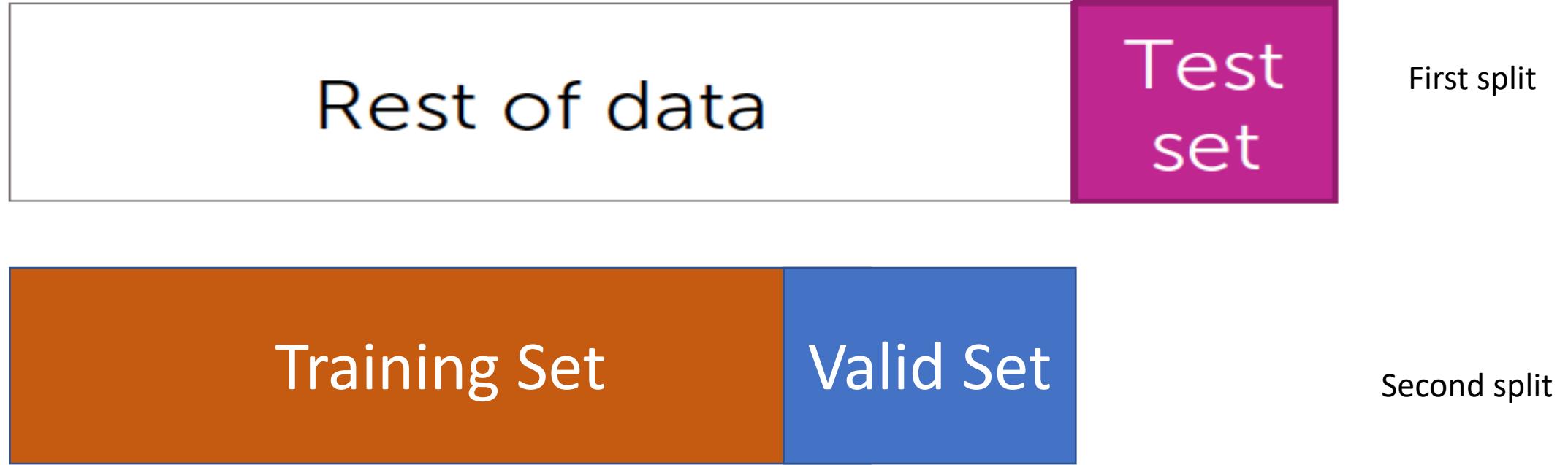
# First split into two, one test and another one which includes both (train and validation)

```
X_trainval, X_test, Y_trainval, Y_test = train_test_split(  
iris.data, iris.target, random_state=0)
```

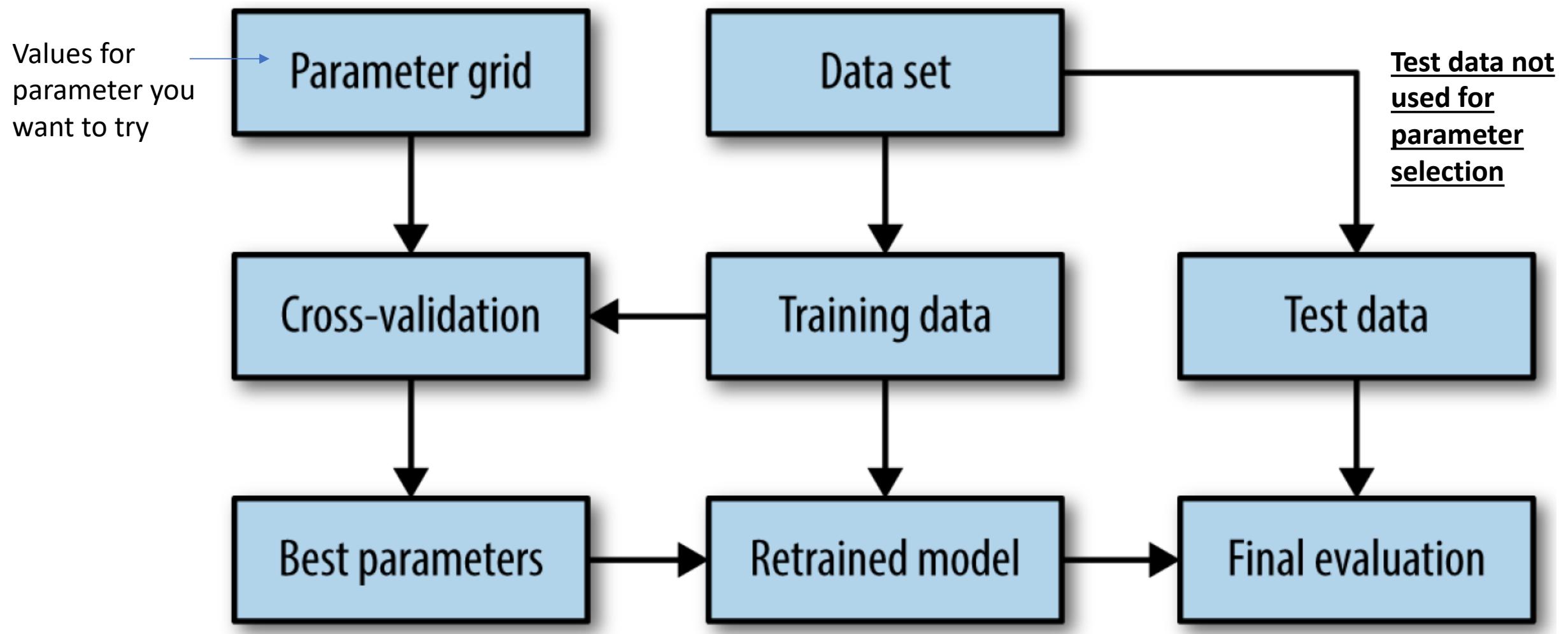
# split train and validation set into training set and validation set

```
X_train, X_valid, Y_train, Y_valid = train_test_split(  
X_trainval, y_trainval, random_state=0)
```

# Search with Cross Validation



# Parameter Selection



```
# Split data into test and trainval set
X_trainval, X_test, Y_trainval, Y_test = train_test_split(
    X_features, Y_targetClass, random_state=0)
# split train+validation set into training and validation sets
X_train, X_valid, Y_train, Y_valid = train_test_split(
    X_trainval, Y_trainval, random_state=0)

best_score = 0 # initialize the best_score to zero which will then be updated

for c in [0.001, 1, 1000]: #iterate over the values we need to try for the parameter
    # for each value of the parameter, train the model
    FittedLogRegModel= LogisticRegression(C=c).fit(X_train,Y_train)

    # evaluate the logistic regression on the validation set
    score = FittedLogRegModel.score(X_valid, Y_valid)

    # if we got a better score, store the score and parameter associated with it
    if score > best_score:
        best_score = score
        best_parameters = c

# after this loop we got the best value of c (stored in best_parameter)

# build a model on the combined training and validation (to get a larger training data for the final model)
# and evaluate it on the test set
SelectedLogRegModel = LogisticRegression(C=best_parameters).fit(X_trainval, Y_trainval)
test_score = SelectedLogRegModel.score(X_test, Y_test)
print("Best score on validation set is:", best_score)
print("Best parameter for regularization (C) is: ", best_parameters)
print("Test set score with best C parameter is", test_score)
```

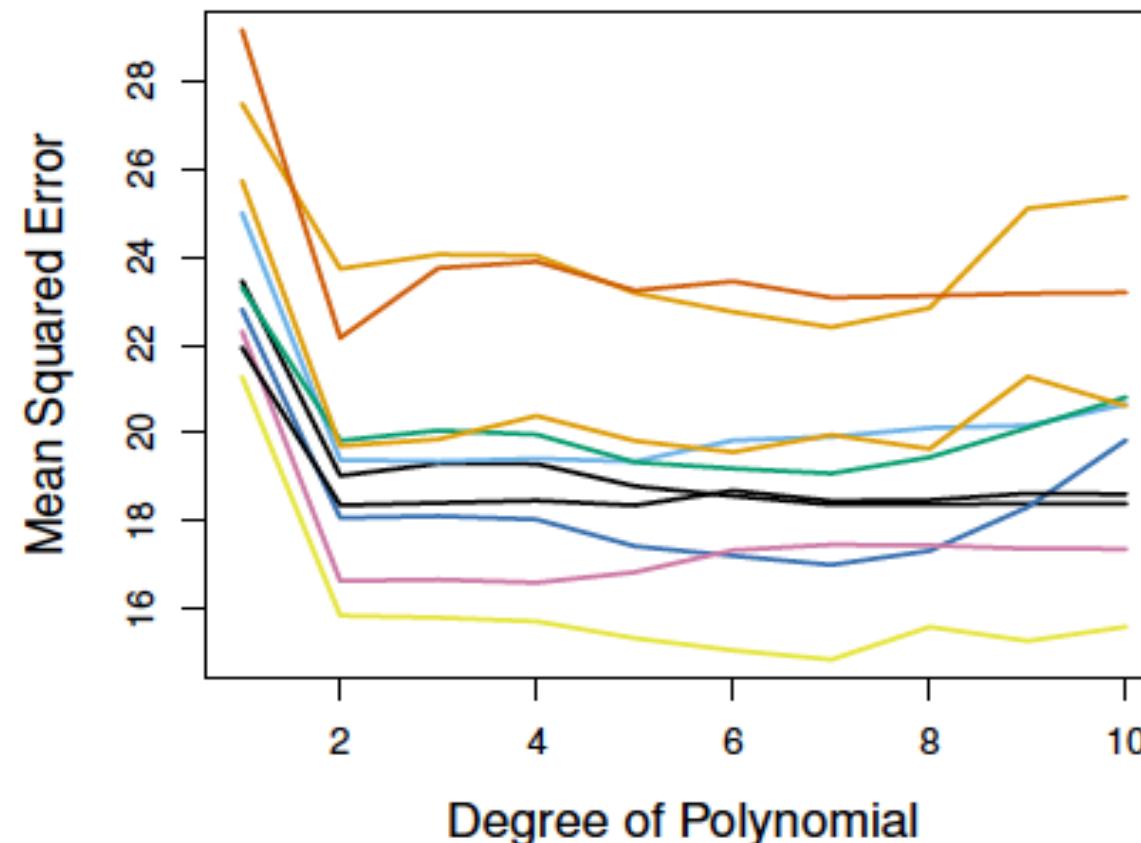
Code Example: Try C (regularization)= [0.001, 1, 1000] in logistic regression and find value that results in best performance

# Problem

- There could be a lot of variability in the performance depending on the split of the data
- We need a more stable approach for model evaluation

# 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!

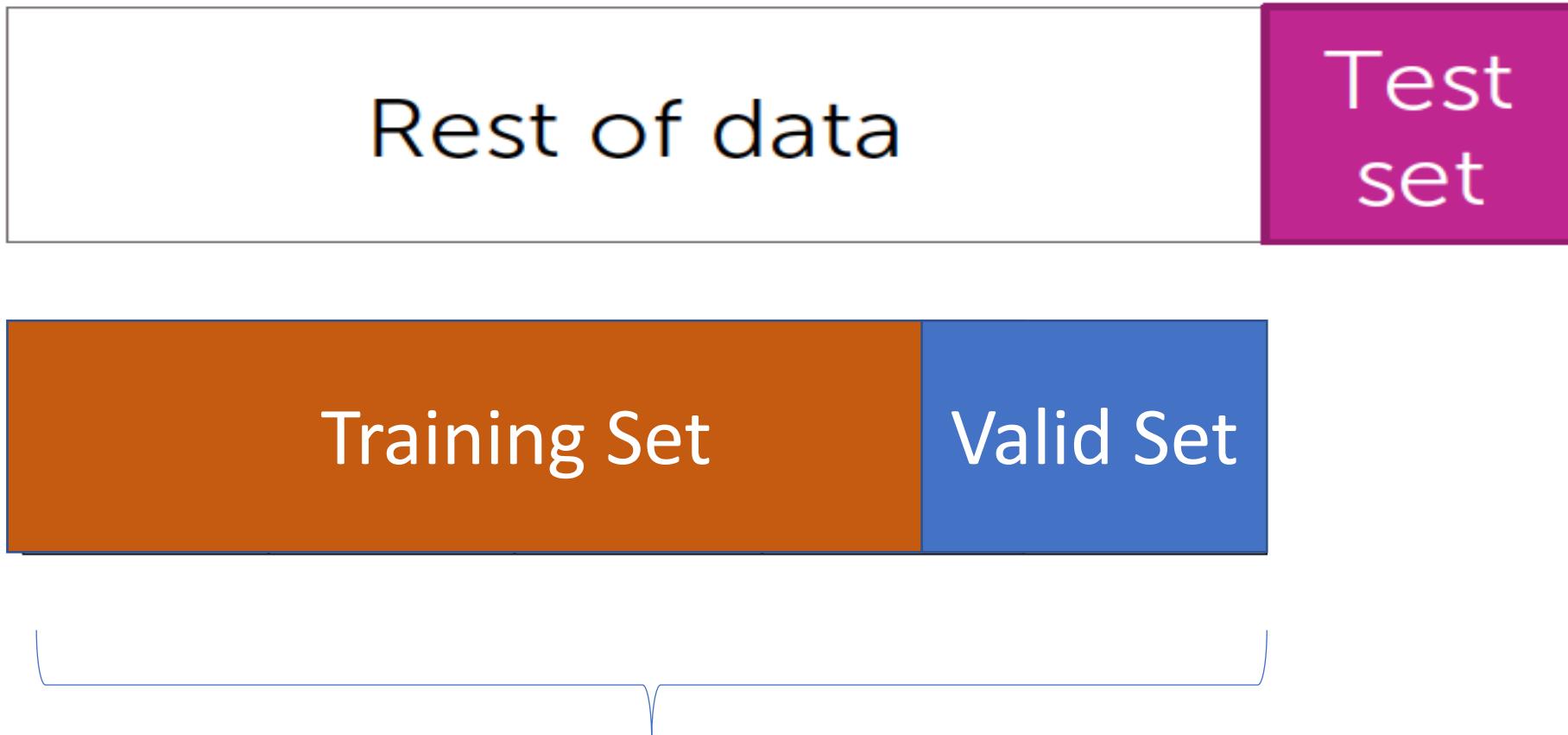


Each curve is obtained using a different random split of the data.. Then get MSE on validation set

# Solution: K-fold Cross Validation

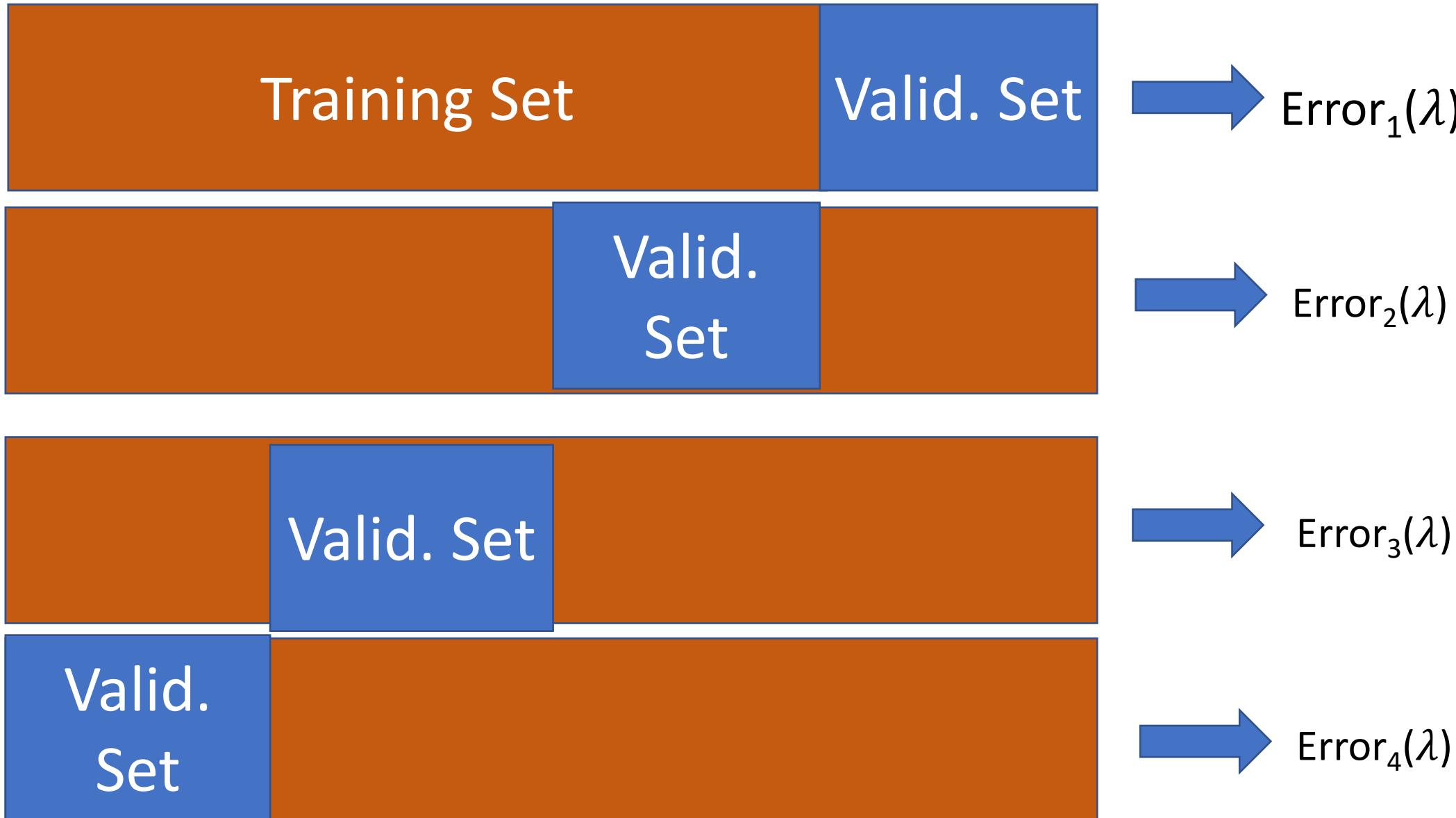
- More stable method of evaluating performance of machine learning algorithms
- Cross validation repeatedly split the data into different train and validation sets
  - Refitting a model of interest and evaluate each time
  - Get average accuracy
- In 4-fold cross validation, we have 4 different splits 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 $\lambda$ for Regularization, with 4-fold Cross Validation



# Cross Validation for Model Selection – Finding Regularization Parameter

- Get  $\lambda$  that **minimize average error** over the K folds:
  - Average error =  $\frac{1}{K} \sum_{j=1}^K \text{Error}_j (\lambda)$
- **Note:** for classification problems, the proportion between classes are the same in each fold as they are in the whole dataset (stratified K-fold cross validation)
  - For example, if data set has 80% class A, and 20% class B, then each portion draws 80% from class A and 20% from class B
  - This is the default implementation in Python

# Cross-Validation in Python

```
from sklearn.model_selection import cross_val_score  
from sklearn.linear_model import LogisticRegression  
logreg = LogisticRegression()  
scores = cross_val_score(logreg, X_trainval, Y_trainval, cv=5) ➔
```

Model we want to evaluate      features      True labels      Number of folds



Get vector of scores for the different splits (the function does the splits internally)

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

```
best_score = 0 # initialize the best_score to zero which will then be updated

kfolds=5 # set the number of folds

for c in [0.001, 1, 1000]: #iterate over the values we need to try for the parameter
    # for each value of c,
    # train the model
    logRegModel = LogisticRegression(C=c)
    # perform cross-validation
    scores = cross_val_score(logRegModel, X_trainval, Y_trainval, cv=kfolds)

    # compute mean cross-validation accuracy
    score = np.mean(scores)

    # if we got a better score, store the score and parameters
    if score > best_score:
        best_score = score
        best_parameters = c

# rebuild a model on the combined training and validation set
SelectedLogRegModel = LogisticRegression(C=best_parameters).fit(X_trainval, Y_trainval)

test_score = SelectedLogRegModel.score(X_test, Y_test)
print("Best score on validation set is:", best_score)
print("Best parameter for regularization (C) is: ", best_parameters)
print("Test set score with best C parameter is", test_score)
```

Code Example: Try C =[0.001, 1, 1000] in logistic regression and find value that results in best performance using 5 fold CV

# 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 training 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

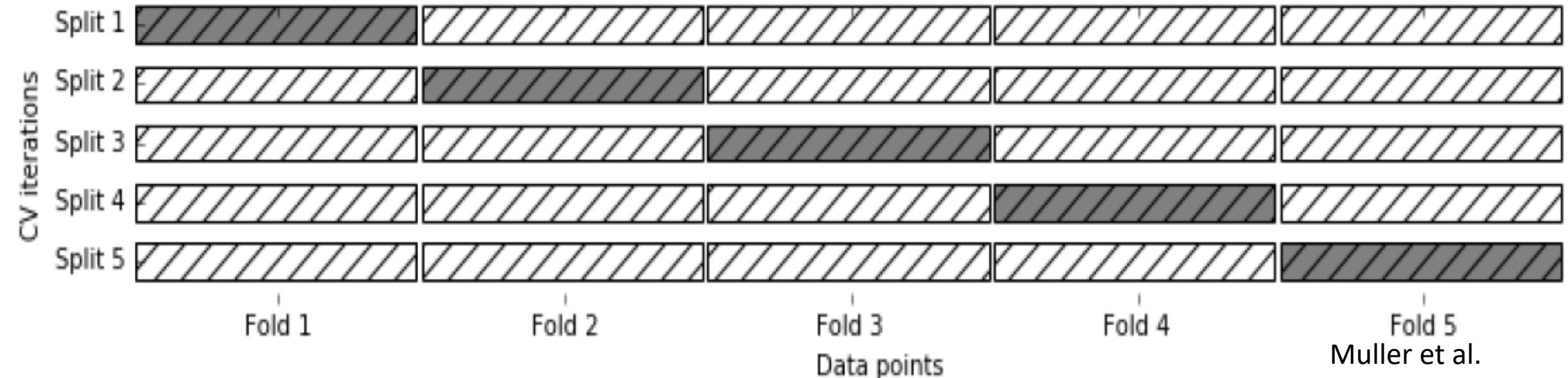
# K-Fold Cross-Validation on Train and Testing for More Stable Result of a Model Evaluation

- Remember: here we are not selecting model, but we are finding more stable measure for accuracy .. So we have two splits only (train and test)
- Same steps as before: Randomly divide data into K parts (1,2,3..K), then  
For  $j = 1, 2 \dots K$

Leave out part  $j$  for testing

Use the remaining  $K-1$  parts for training  
combine (average) the results

 Training data  
 Test data



# 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 is one sample
  - Each time you test the performance using a single observation

- Python:

```
from sklearn.model_selection import LeaveOneOut  
loo = LeaveOneOut()  
scores = cross_val_score(logreg, iris.data, iris.target, cv=loo)
```

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

