COSC 3337 : Data Science I

N. Rizk

College of Natural and Applied Sciences

Department of Computer Science

University of Houston

CROSS-VALIDATION AND MODEL SELECTION

How to check if a model fit is good?

- The R^2 statistic has become the almost universally standard measure for model fit in linear models.
- What is R^2?

$$R^2 = 1 - \frac{\sum(y_i - f_i)^2}{\sum(y_i - \bar{y})^2}$$

← Model error

← Variance in the dependent variable

- It is the ratio of error in a model over the total variance in the dependent variable.
- Hence the lower the error, the higher the R^2 value.

How to check if a model fit is good?

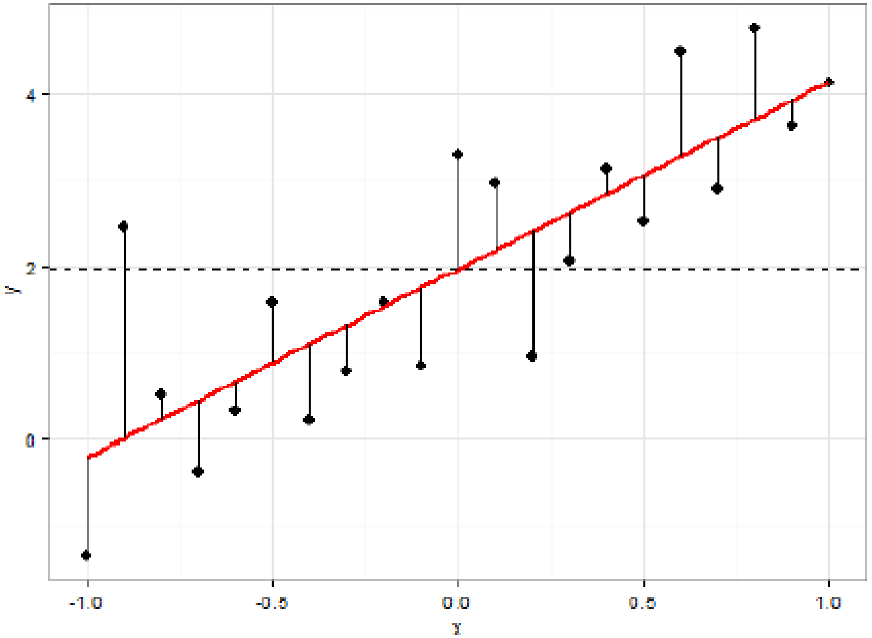
$$\sum(y_i - f_i)^2 = 18.568$$

$$\sum(y_i - \bar{y})^2 = 55.001$$

$$R^2 = 1 - \frac{18.568}{55.001}$$

$$R^2 = 0.6624$$

A decent model fit!



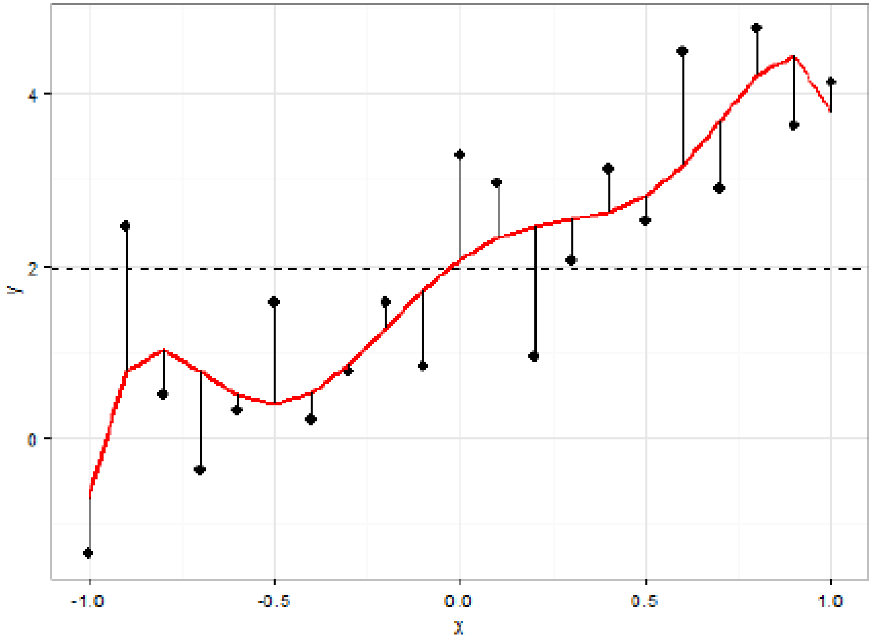
How to check if a model fit is good?

$$\sum(y_i - f_i)^2 = 15.276$$

$$\sum(y_i - \bar{y})^2 = 55.001$$

$$R^2 = 1 - \frac{15.276}{55.001}$$

$$R^2 = 0.72$$



Is this a better model?

No, overfitting!

OVERFITTING

- Modeling techniques tend to overfit the data.
- Multiple regression:

✓*Every* time you add a variable to the regression, the model's R^2 goes up.

✓Naïve interpretation: *every* additional predictive variable helps to explain yet more of the target's variance. But that can't be true!

✓Left to its own devices, Multiple Regression will fit *too many* patterns.

✓A reason why modeling requires subject-matter expertise.

OVERFITTING

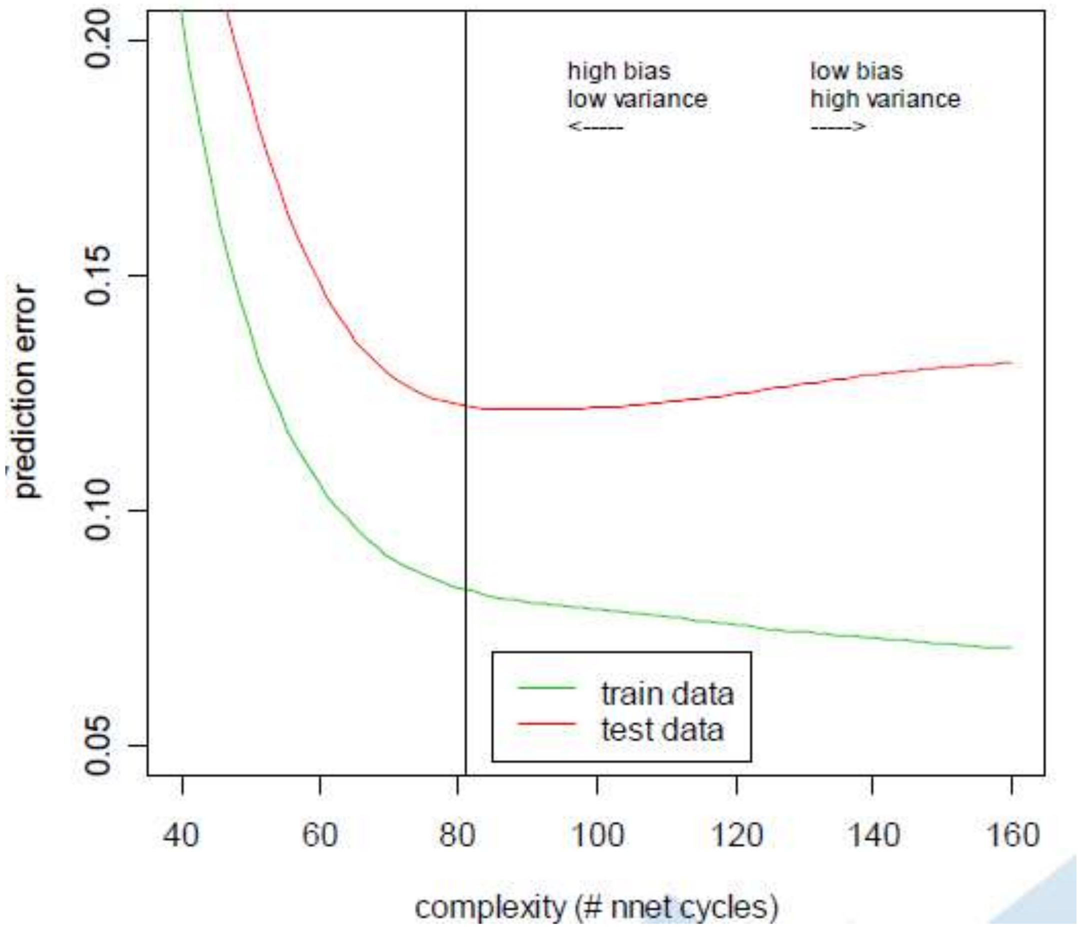
- Error on the dataset used to *fit* the model can be misleading

› Doesn’t predict future performance.

- Too much complexity can diminish model’s accuracy on future data.

› Sometimes called the Bias-Variance Tradeoff.

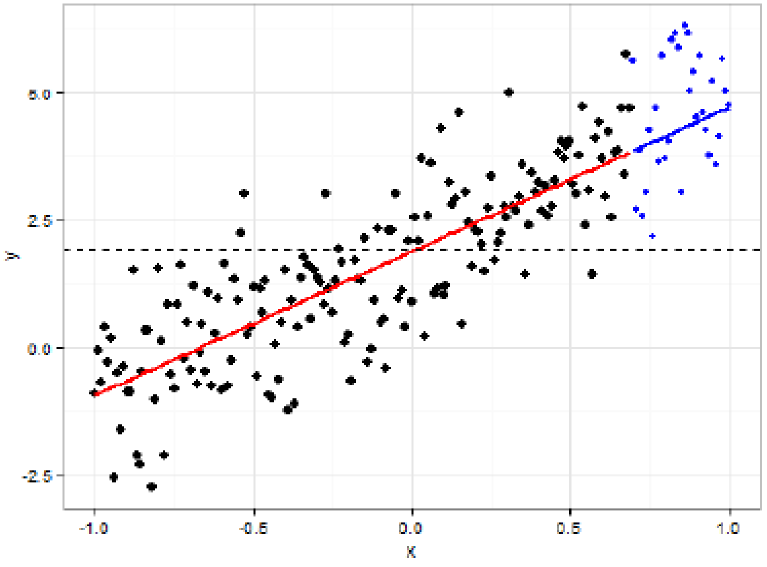
Training vs Test Error

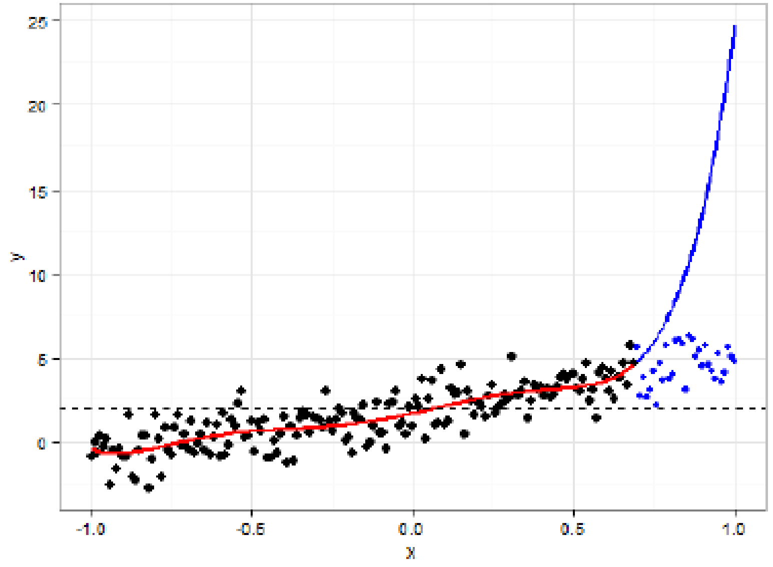


OVERFITTING

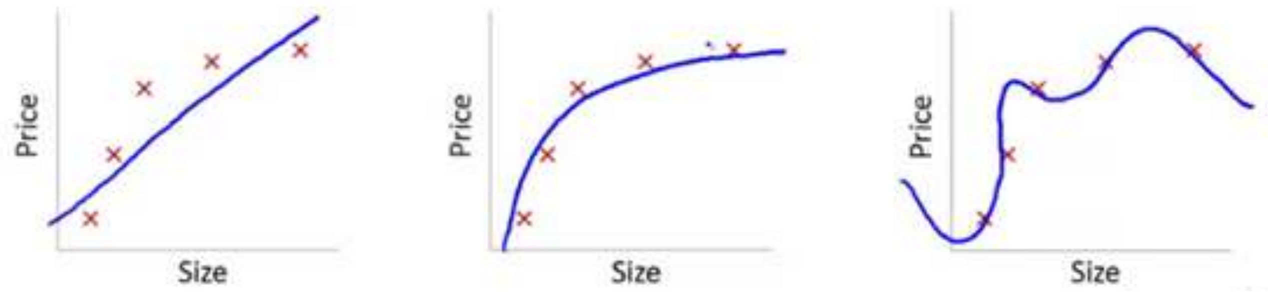
- What are the consequences of overfitting?

›“*Overfitted models will have high R^2 values, but will perform poorly in predicting out-of-sample cases*”





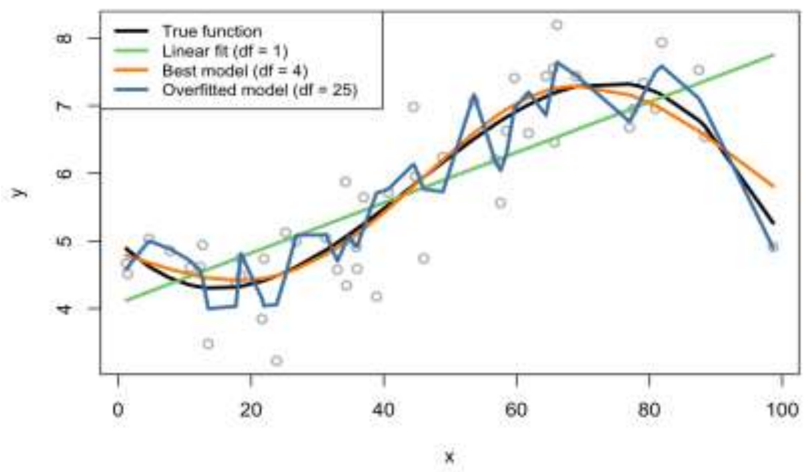
Why do models lose stability?



High error from training data points
➔underfitting

Low training error and generalization of the relationship

Zero training error
➔Overfitting



Thus, the need for validation techniques

WHY WE NEED CROSS-VALIDATION?

- R^2, also known as coefficient of determination, is a popular measure of quality of fit in regression. However, it does not offer any significant insights into how well our regression model can predict future values.
- When an MLR equation is to be used for prediction purposes, it is useful to obtain empirical evidence as to its generalizability, or its capacity to make accurate predictions for new samples of data. This process is sometimes referred to as “validating” the regression equation.

Why *cross-validation is needed ?*

- One way to address this issue is to literally obtain a new sample of observations. That is, after the MLR equation is developed from the original sample, the investigator conducts a new study, replicating the original one as closely as possible, and uses the new data to assess the predictive validity of the MLR equation.
- This procedure is usually viewed as impractical because of the requirement to conduct a new study to obtain validation data, as well as the difficulty in truly replicating the original study.
- An alternative, more practical procedure is *cross-validation*.

CROSS-VALIDATION

- In cross-validation the original sample is split into two parts. One part is called the training (or *derivation)* sample, and the other part is called the *validation (or validation + testing)* sample.

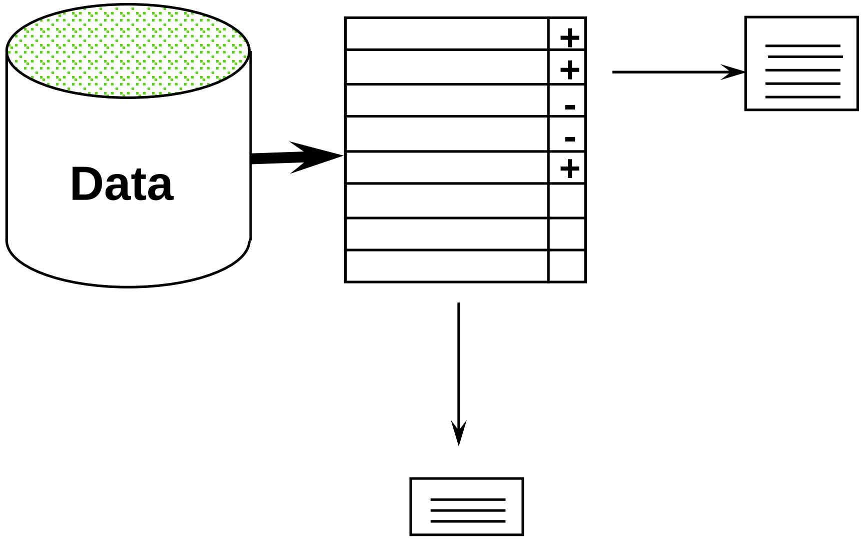
1) What portion of the sample should be in each part?

If sample size is very large, it is often best to split the sample in half. For smaller samples, it is more conventional to split the sample such that 2/3 of the observations are in the derivation sample and 1/3 are in the validation sample.

Classification Step 1:
Split data into train and test sets

THE PAST

Results Known



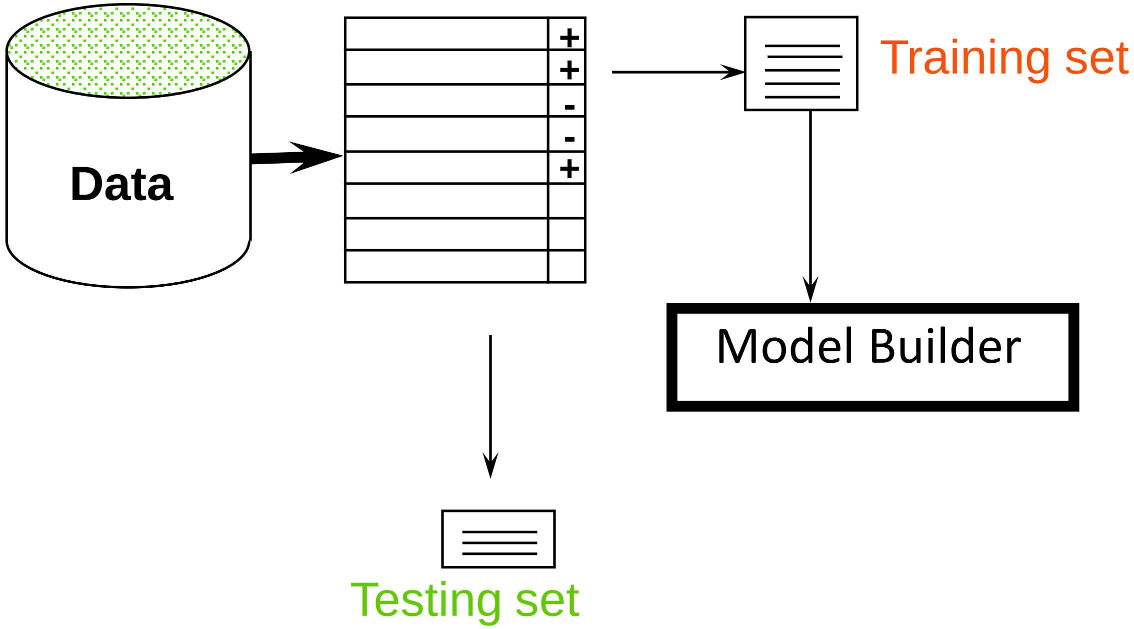
Training set

Testing set

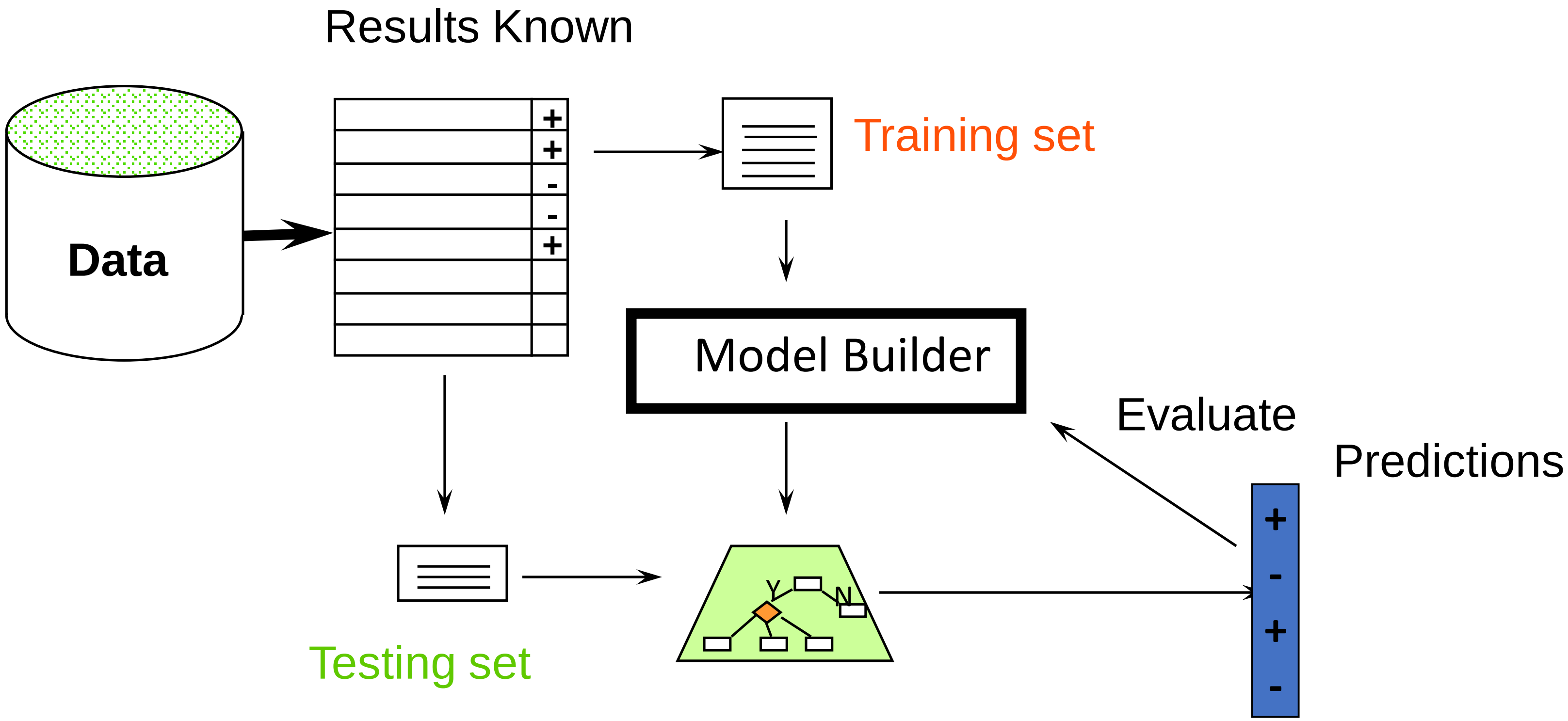
Classification Step 2: Build a model on a training set

THE PAST

Results Known



Classification Step 3:
Evaluate on test set (Re-train?)



CROSS-VALIDATION

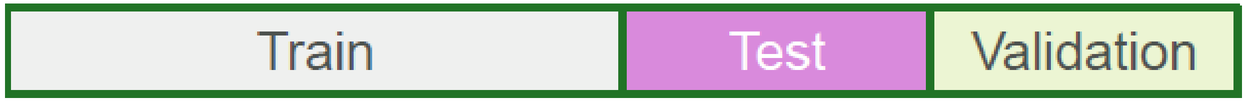
2) How should the sample be split?

The most common approach is to divide the sample randomly, thus theoretically eliminating any systematic differences. One alternative is to define matched pairs of subjects in the original sample and to assign one member of each pair to the derivation sample and the other to the validation sample.

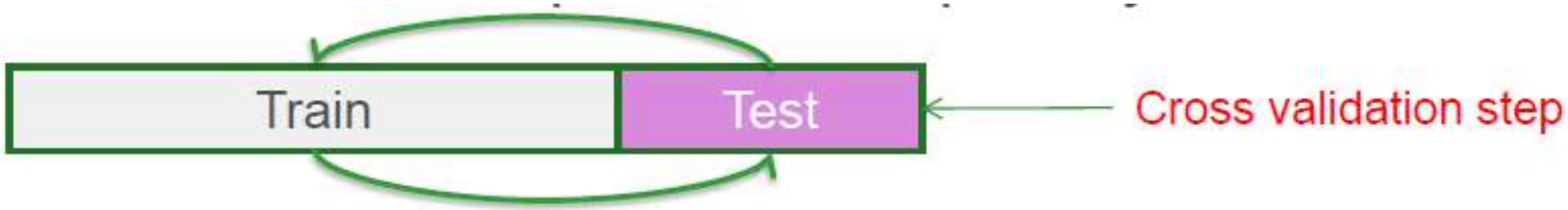
- Modeling of the data uses one part only. The model selected for this part is then used to predict the values in the other part of the data. A valid model should show good predictive accuracy.
- One thing that R-squared offers no protection against is overfitting. On the other hand, cross validation, by allowing us to have cases in our testing set that are different from the cases in our training set, inherently offers protection against overfitting.

CROSS VALIDATION – THE IDEAL PROCEDURE

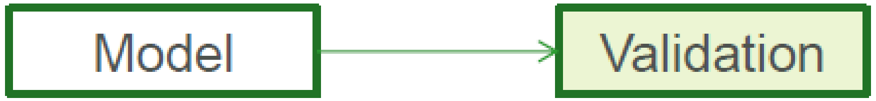
1.Divide data into three sets, training, validation and test sets



2.Find the optimal model on the training set, and use the test set to check its predictive capability



3.See how well the model can predict the test set



4.The validation error gives an unbiased estimate of the predictive power of a model

```
train, validation = train_test_split(data, test_size=0.50,
random_state = 5)
```

TRAINING/TEST DATA SPLIT

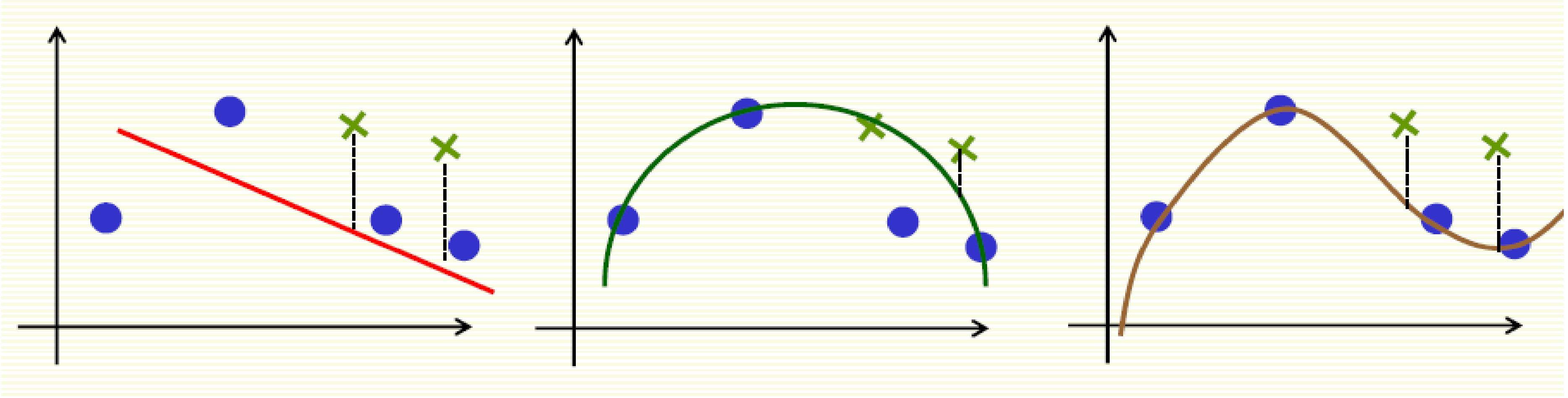
Talked about splitting data in training/test sets

- training data is used to fit parameters
- test data is used to assess how classifier generalizes to new data

What if classifier has "non-tunable" parameters?

- a parameter is "non-tunable" if tuning (or training) it on the training data leads to overfitting

TRAINING/TEST DATA SPLIT



What about test error? Seems appropriate

- degree 2 is the best model according to the test error

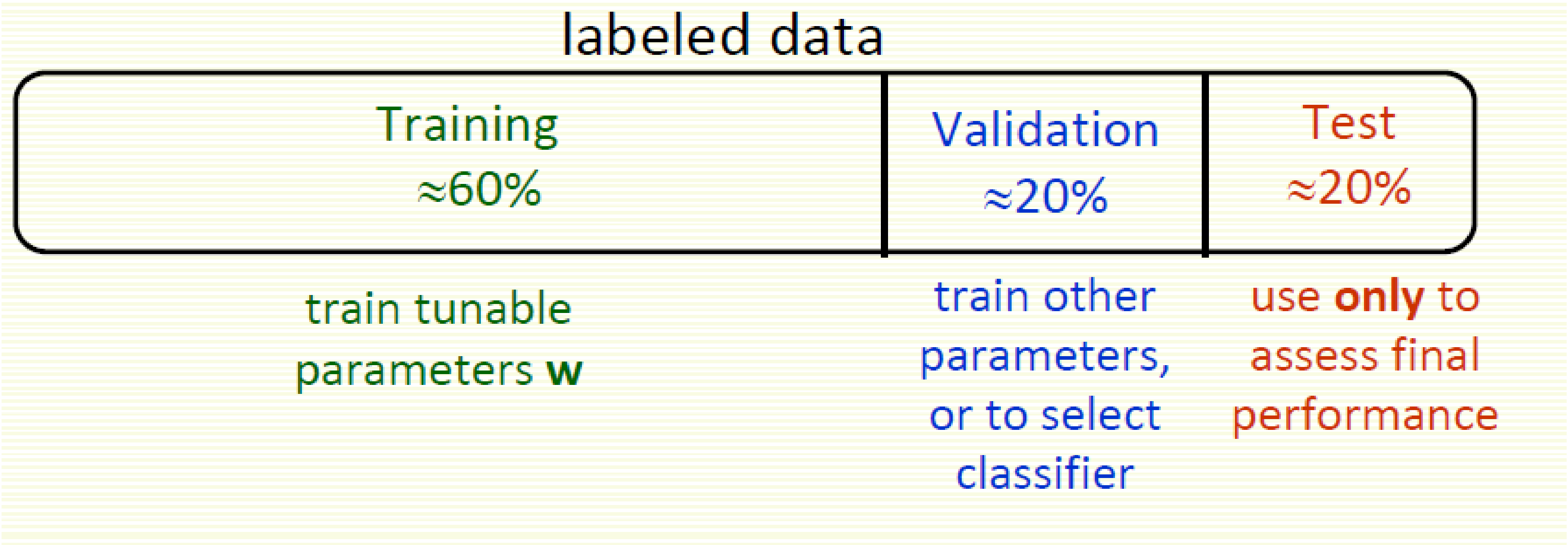
Except what do we report as the test error now?

- Test error should be computed on data that was **not used for training at all**
- Here used “test” data for training, i.e. choosing model

VALIDATION DATA

Same question when choosing among several classifiers

- our polynomial degree example can be looked at as choosing among 3 classifiers (degree 1, 2, or 3)
- Solution: split the labeled data into three parts



labeled data

Training ≈60%

Validation ≈20%

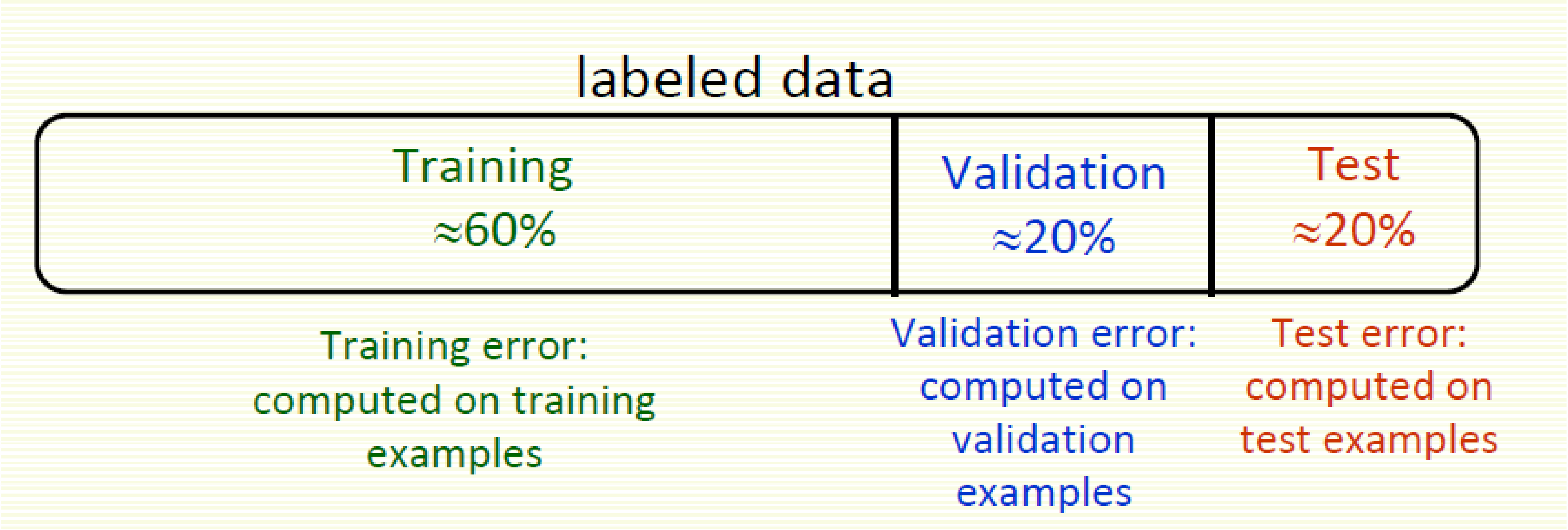
Test ≈20%

train tunable parameters **w**

train other parameters, or to select classifier

use **only** to assess final performance

TRAINING/ VALIDATION



labeled data

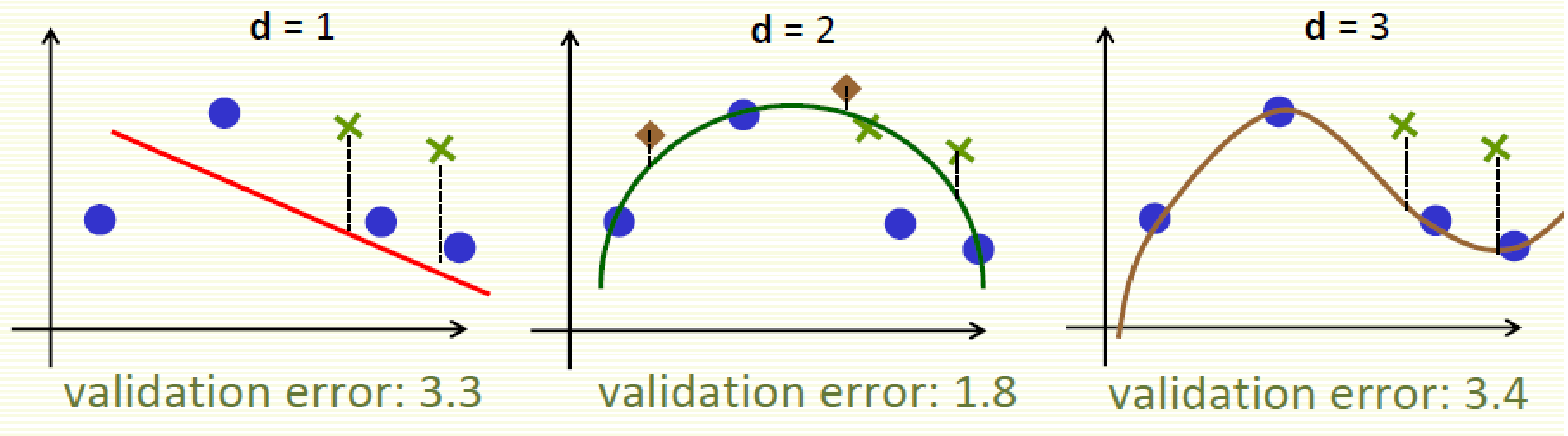
Training ≈60%	Validation ≈20%	Test ≈20%

Training error: computed on training examples

Validation error: computed on validation examples

Test error: computed on test examples

Training/Validation/Test Data

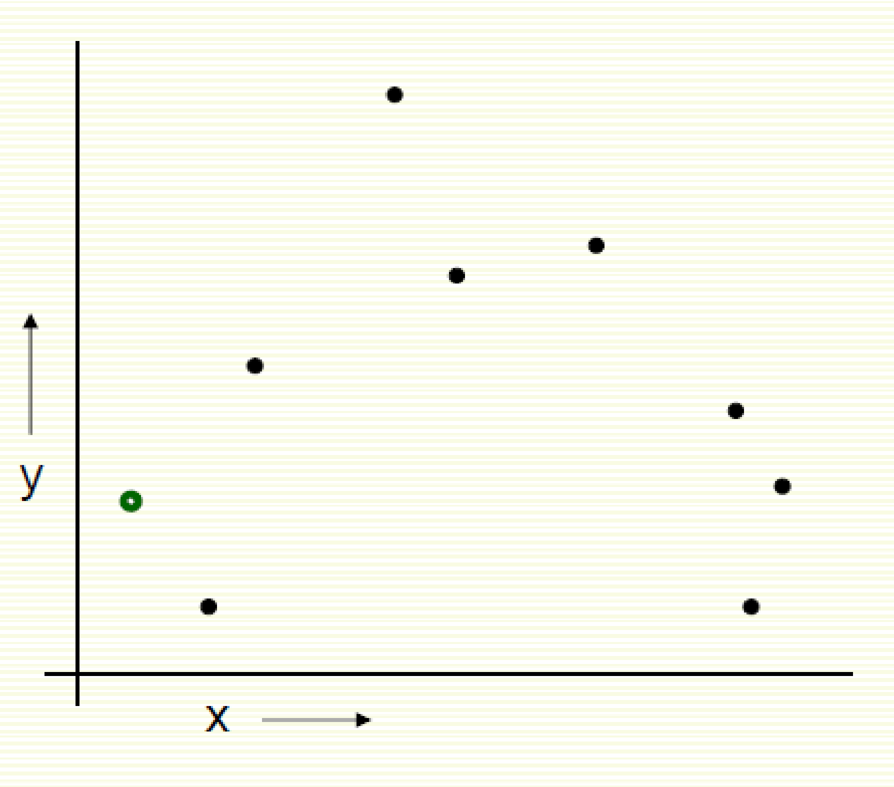


- Training Data
- Validation Data

  d = 2 is chosen
- Test Data

  1.3 test error computed for **d** = 2

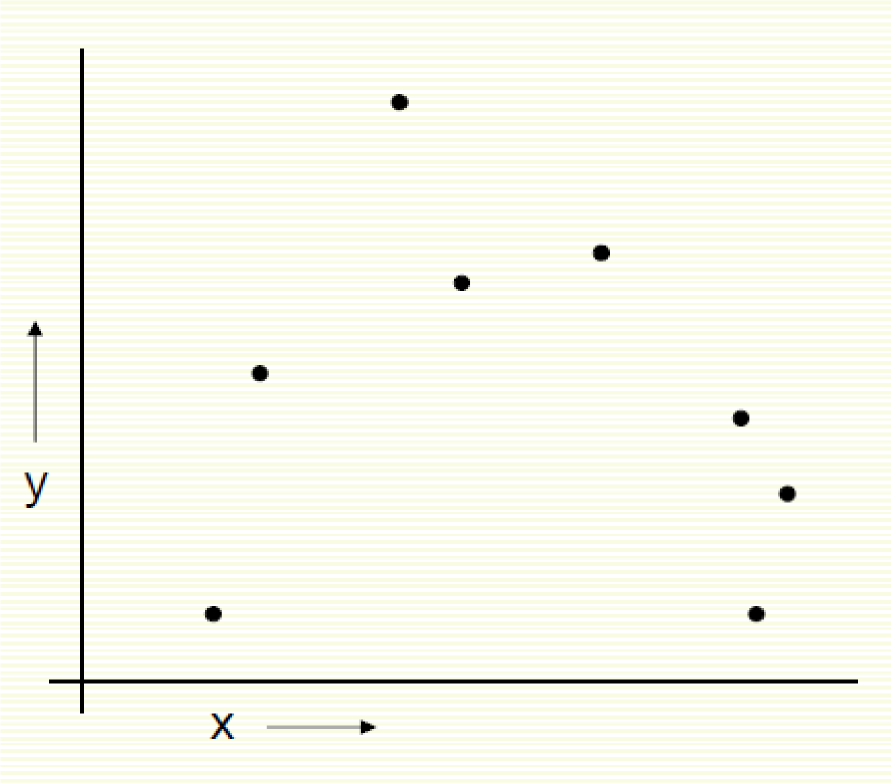
LOOCV (Leave-one-out Cross Validation)



- For k=1 to R

1. Let (**x**^{k},**y**^{k}) be the **k** example

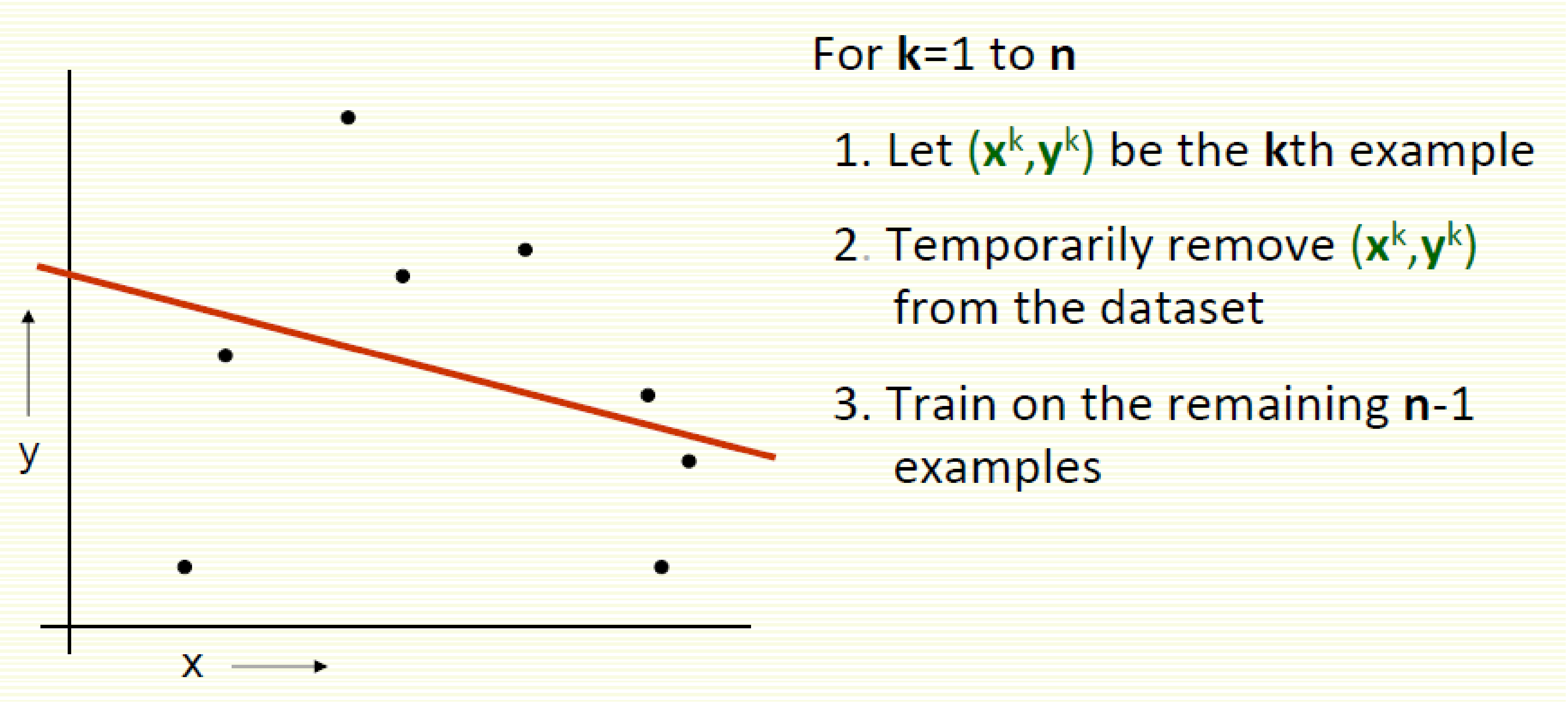
LOOCV (Leave-one-out Cross Validation)



For **k**=1 to **n**

1. Let ($\mathbf{x}^k$,$\mathbf{y}^k$) be the **k**th example
2. Temporarily remove ($\mathbf{x}^k$,$\mathbf{y}^k$) from the dataset

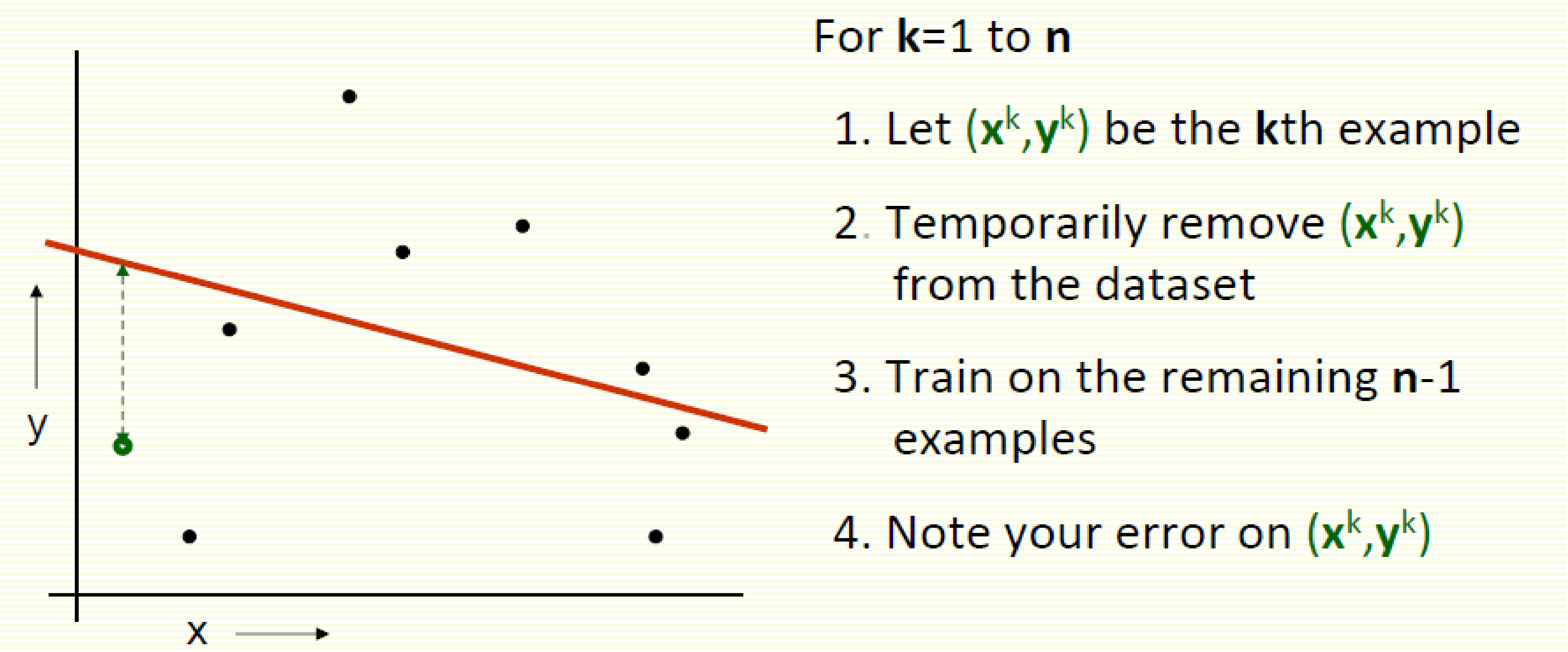
LOOCV (Leave-one-out Cross Validation)



For **k**=1 to **n**

1. Let $(\mathbf{x}^k, \mathbf{y}^k)$ be the **k**th example
2. Temporarily remove $(\mathbf{x}^k, \mathbf{y}^k)$ from the dataset
3. Train on the remaining **n**-1 examples

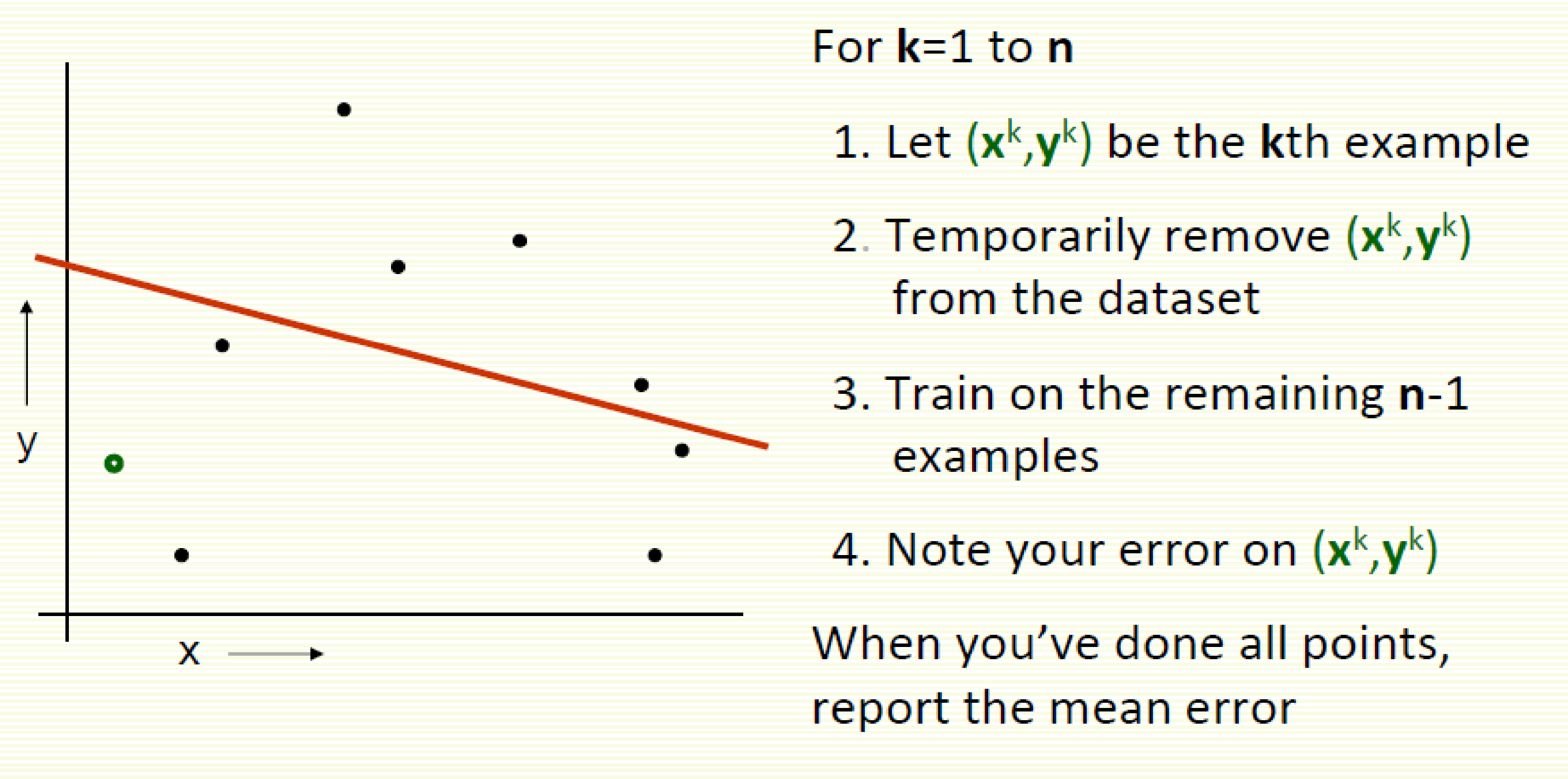
LOOCV (Leave-one-out Cross Validation)



For **k**=1 to **n**

1. Let $(\mathbf{x}^k,\mathbf{y}^k)$ be the **k**th example
2. Temporarily remove $(\mathbf{x}^k,\mathbf{y}^k)$ from the dataset
3. Train on the remaining **n**-1 examples
4. Note your error on $(\mathbf{x}^k,\mathbf{y}^k)$

LOOCV (Leave-one-out Cross Validation)

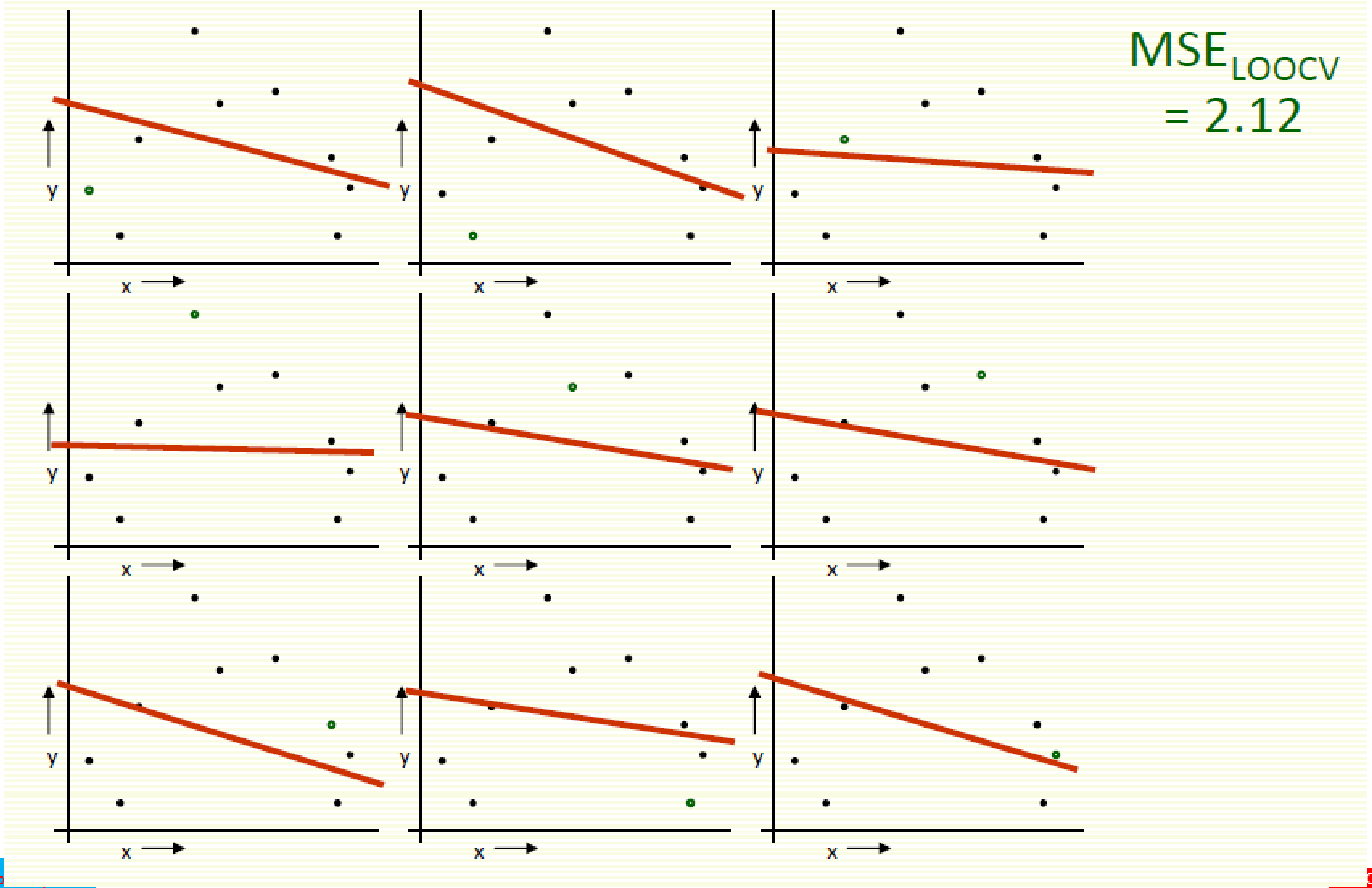


For **k**=1 to **n**

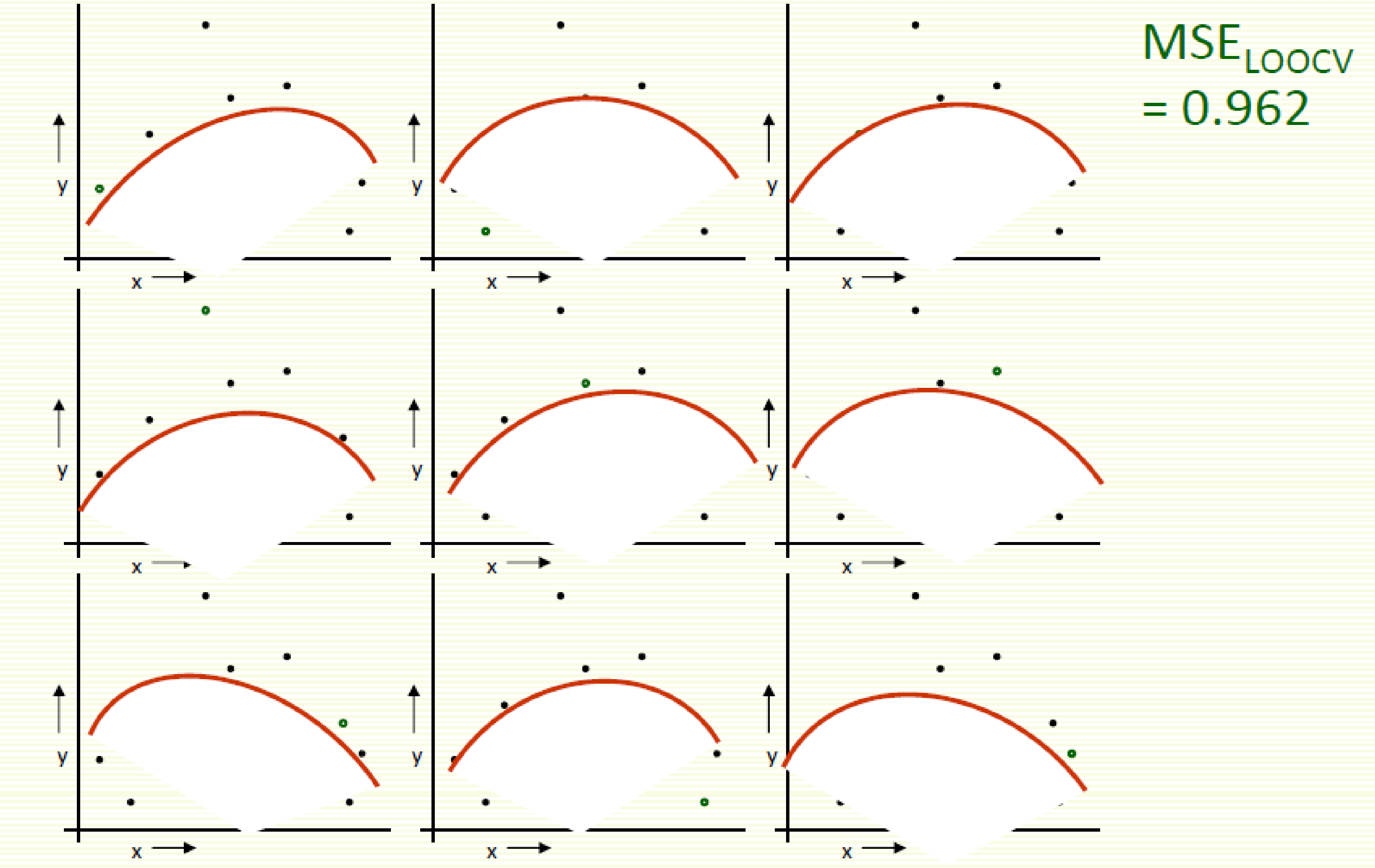
1. Let $(\mathbf{x}^k,\mathbf{y}^k)$ be the **k**th example
2. Temporarily remove $(\mathbf{x}^k,\mathbf{y}^k)$ from the dataset
3. Train on the remaining **n**-1 examples
4. Note your error on $(\mathbf{x}^k,\mathbf{y}^k)$

When you've done all points, report the mean error

LOOCV (Leave-one-out Cross Validation)

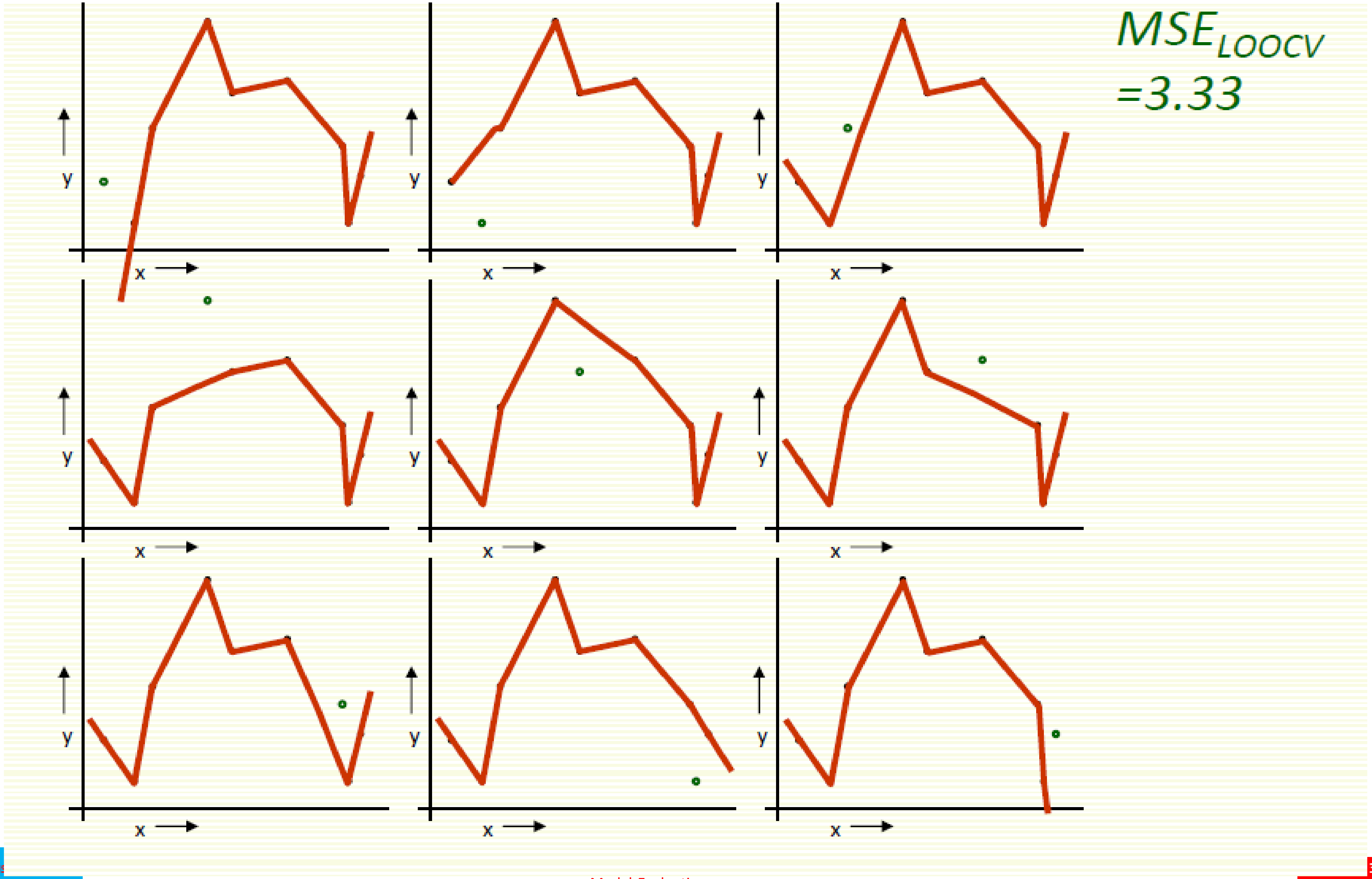


LOOCV for Quadratic Regression



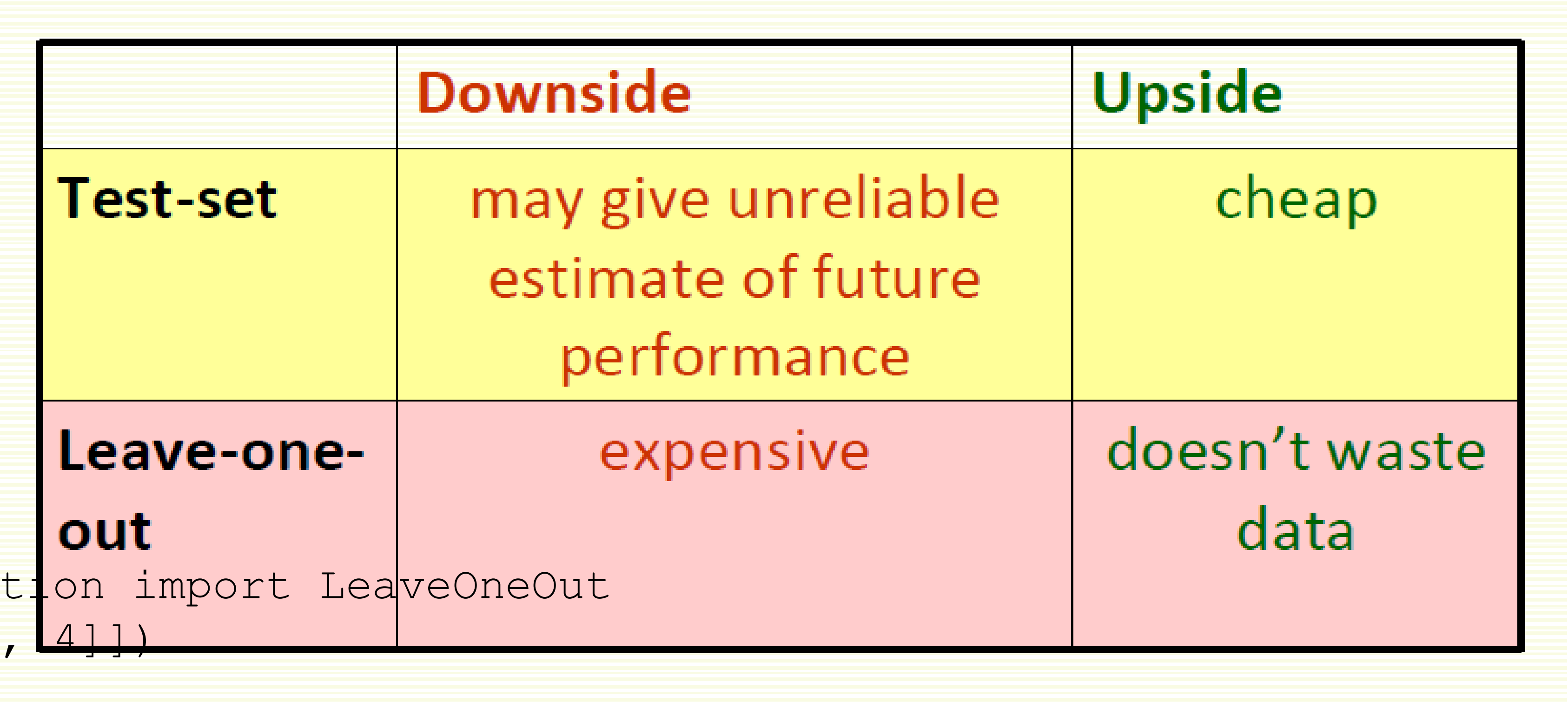
MSE_{LOOCV}
$= 0.962$

LOOCV for Join The Dots



MSE_{LOOCV}
=3.33

Which kind of Cross Validation?



	Downside	Upside
Test-set	may give unreliable estimate of future performance	cheap
Leave-one-out	expensive	doesn’t waste data

```
from sklearn.model_selection import LeaveOneOut
X = np.array([[1, 2], [3, 4]])
y = np.array([1, 2])
loo = LeaveOneOut()
loo.get_n_splits(X)

for train_index, test_index in loo.split(X):
        print("train:", train_index, "validation:", test_index)
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
```

K-FOLD CROSS VALIDATION

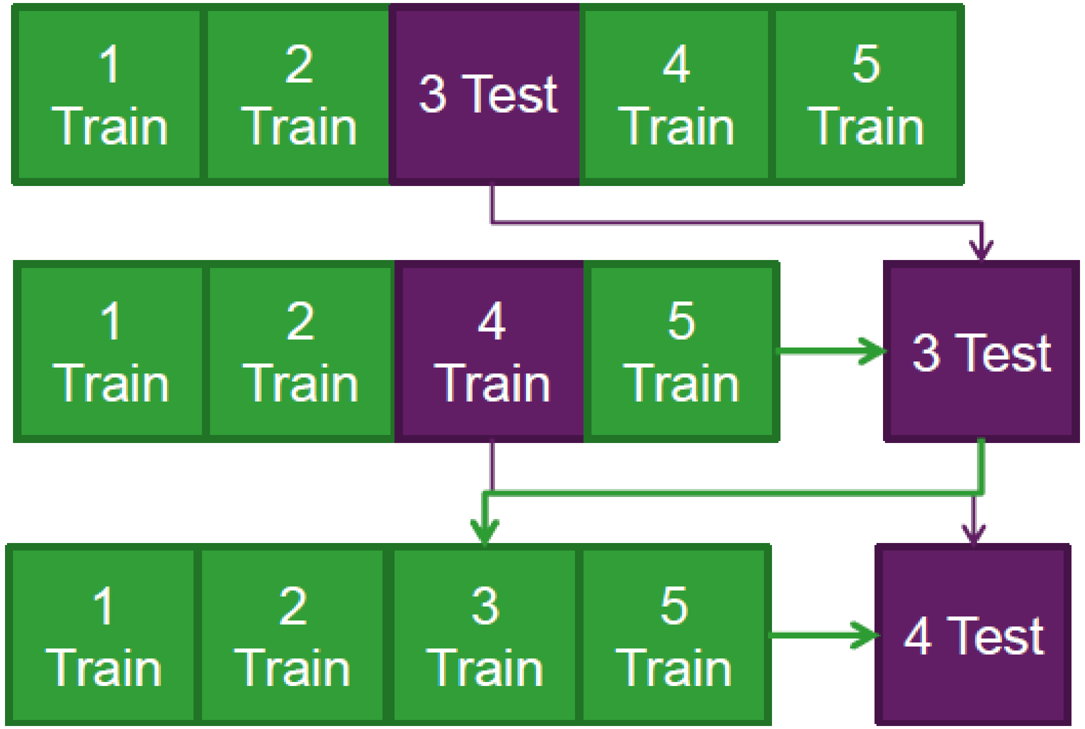
›Since data are often scarce, there might not be enough to set aside for a validation sample

›To work around this issue k-fold CV works as follows:

1. Split the sample into k subsets of equal size
2. For each fold, estimate a model on all the subsets except one
3. Use the left out subset to test the model, by calculating a CV metric of choice
4. Average the CV metric across subsets to get the CV error

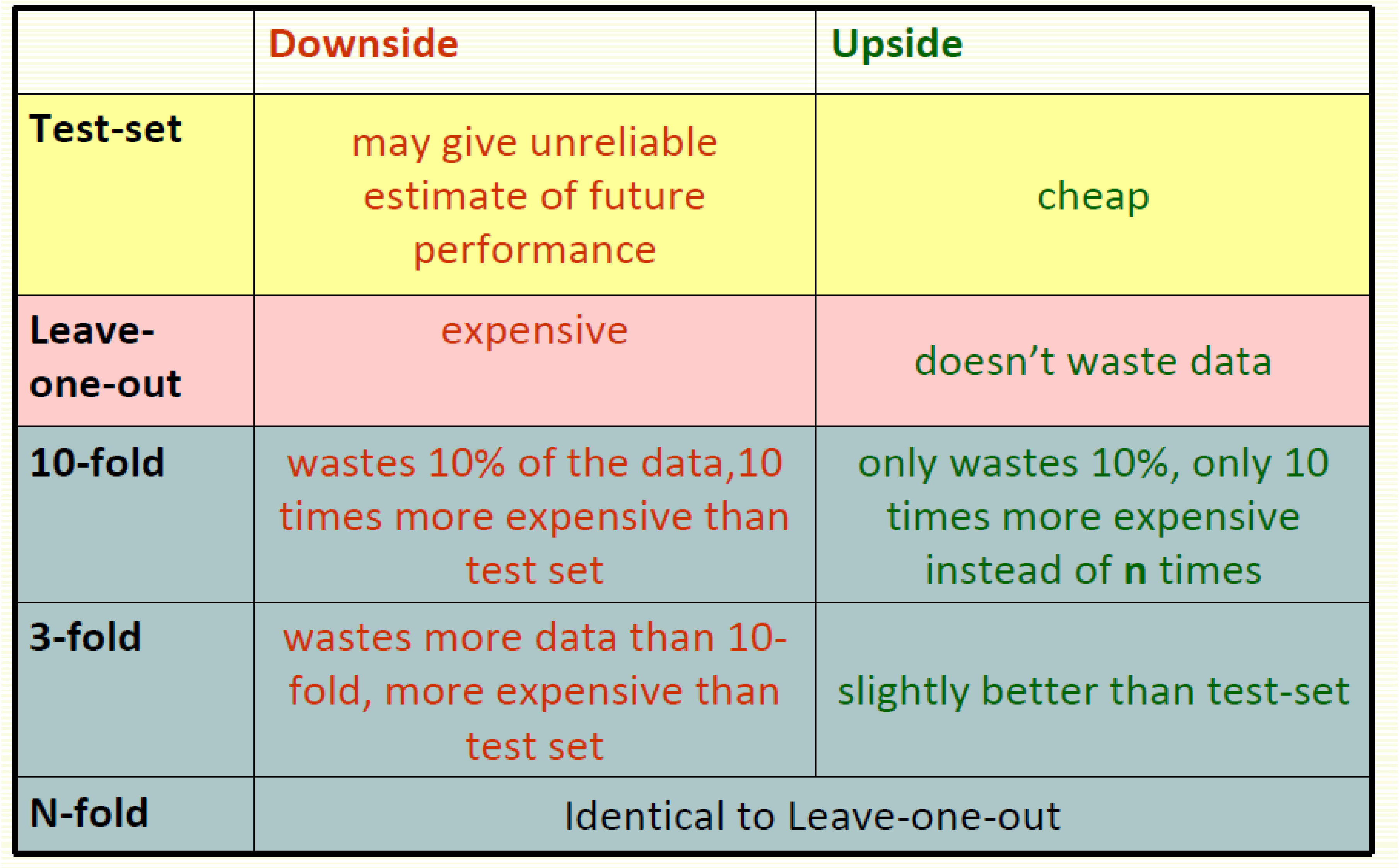
›This has the advantage of using all data for estimating the model, however finding a good value for *k* can be tricky

K-fold Cross Validation Example



1. Split the data into 5 samples
2. Fit a model to the training samples and use the test sample to calculate a CV metric.
3. Repeat the process for the next sample, until all samples have been used to either train or test the model

Which kind of Cross Validation?



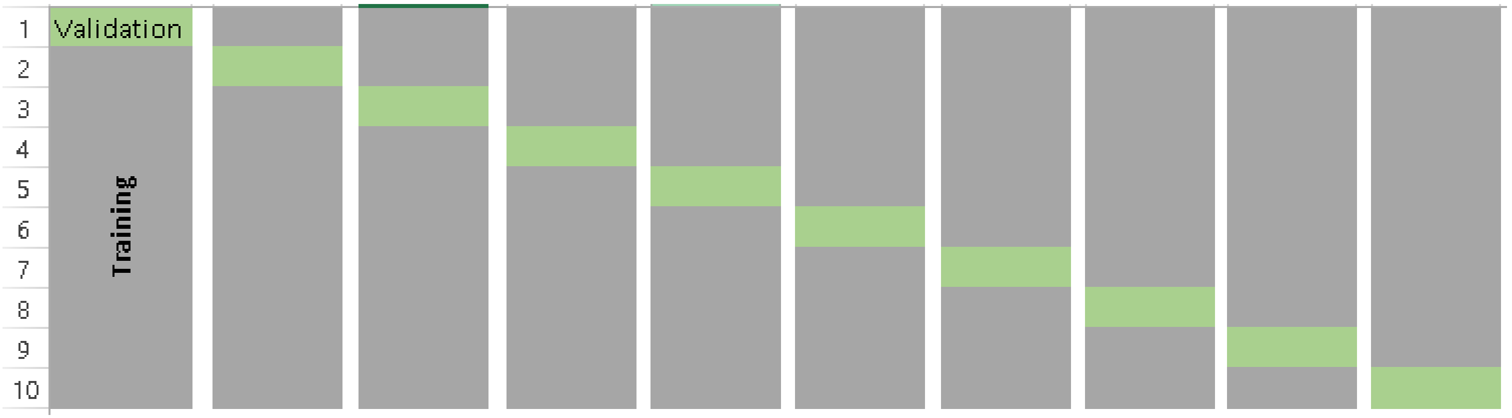
	Downside	Upside
Test-set	may give unreliable estimate of future performance	cheap
Leave-one-out	expensive	doesn't waste data
10-fold	wastes 10% of the data,10 times more expensive than test set	only wastes 10%, only 10 times more expensive instead of **n** times
3-fold	wastes more data than 10-fold, more expensive than test set	slightly better than test-set
N-fold	Identical to Leave-one-out	

Improve cross-validation

- Even better: *repeated* cross-validation

Example:

10-fold cross-validation is repeated 10 times and results are averaged (reduce the variance)



10 fold in Pyhton

```
from sklearn.model_selection import KFold
kf = RepeatedKFold(n_splits=5, n_repeats=10, random_state=None)

for train_index, test_index in kf.split(X):
      print("Train:", train_index, "Validation:",test_index)
      X_train, X_test = X[train_index], X[test_index]
      y_train, y_test = y[train_index], y[test_index]
```

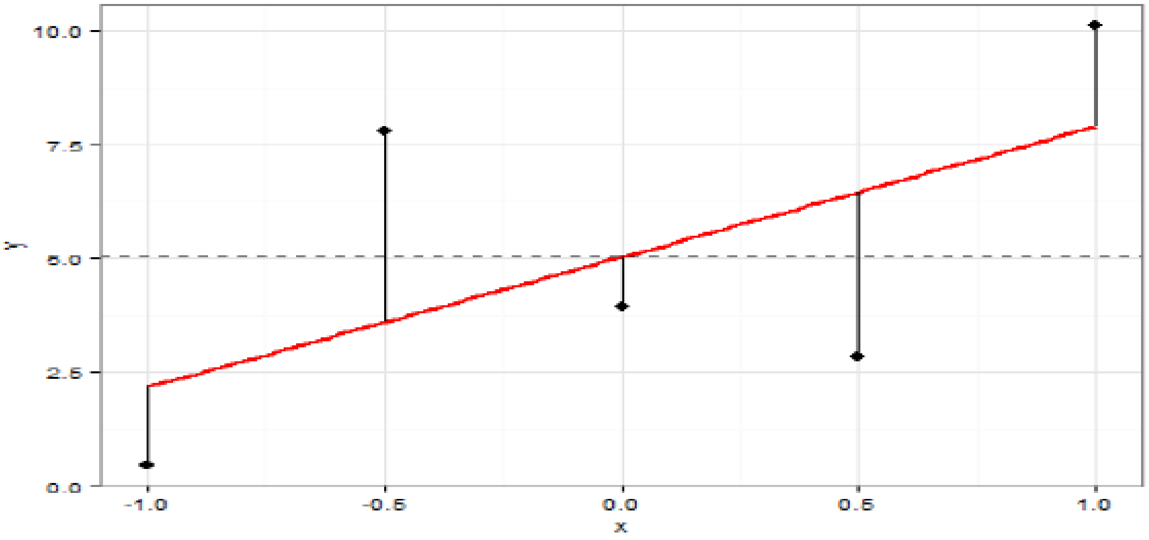
Cross Validation - Metrics

- How do we determine if one model is predicting better than another model?

The basic relation:

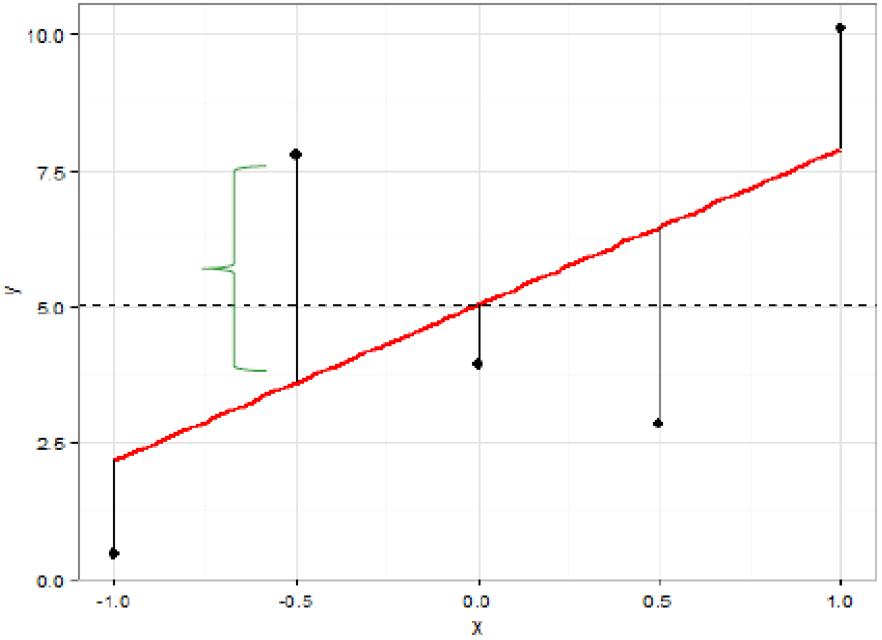
› $Error_i = y_i - f_i$

The difference between observed (*y*) and predicted value (*f*), when applying the model to unseen data

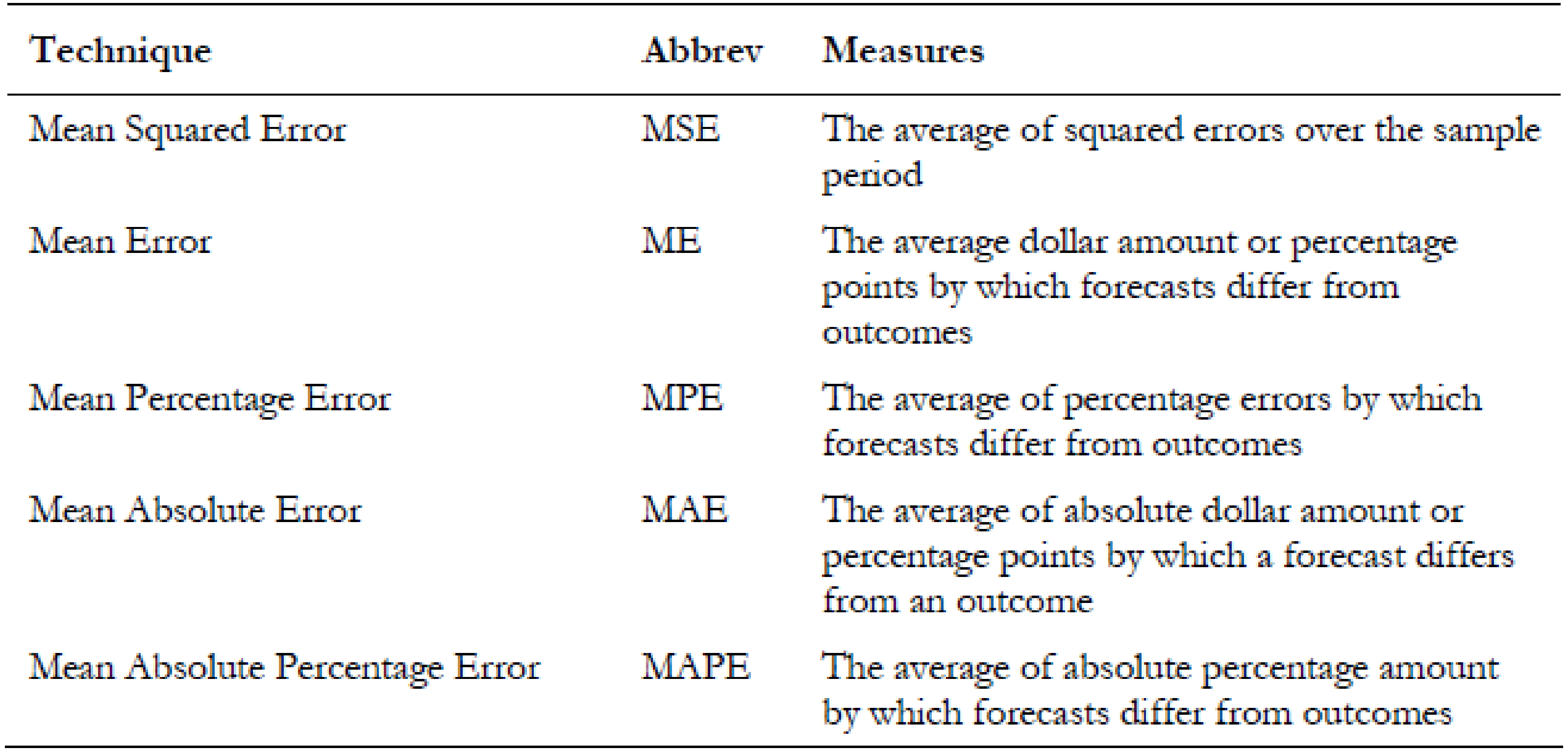


Cross Validation Metrics

- Mean Squared Error (MSE)
  - $1/n\sum(y_i - f_i)^2$
  - 7.96
- Root Mean Squared Error (RMSE)
  - $\sqrt{1/n\sum(y_i - f_i)^2}$
  - 2.82
- Mean Absolute Percentage Error (MAPE)
  - $(1/n\sum|\frac{y_i - f_i}{y_i}|)$*100
  - 120%

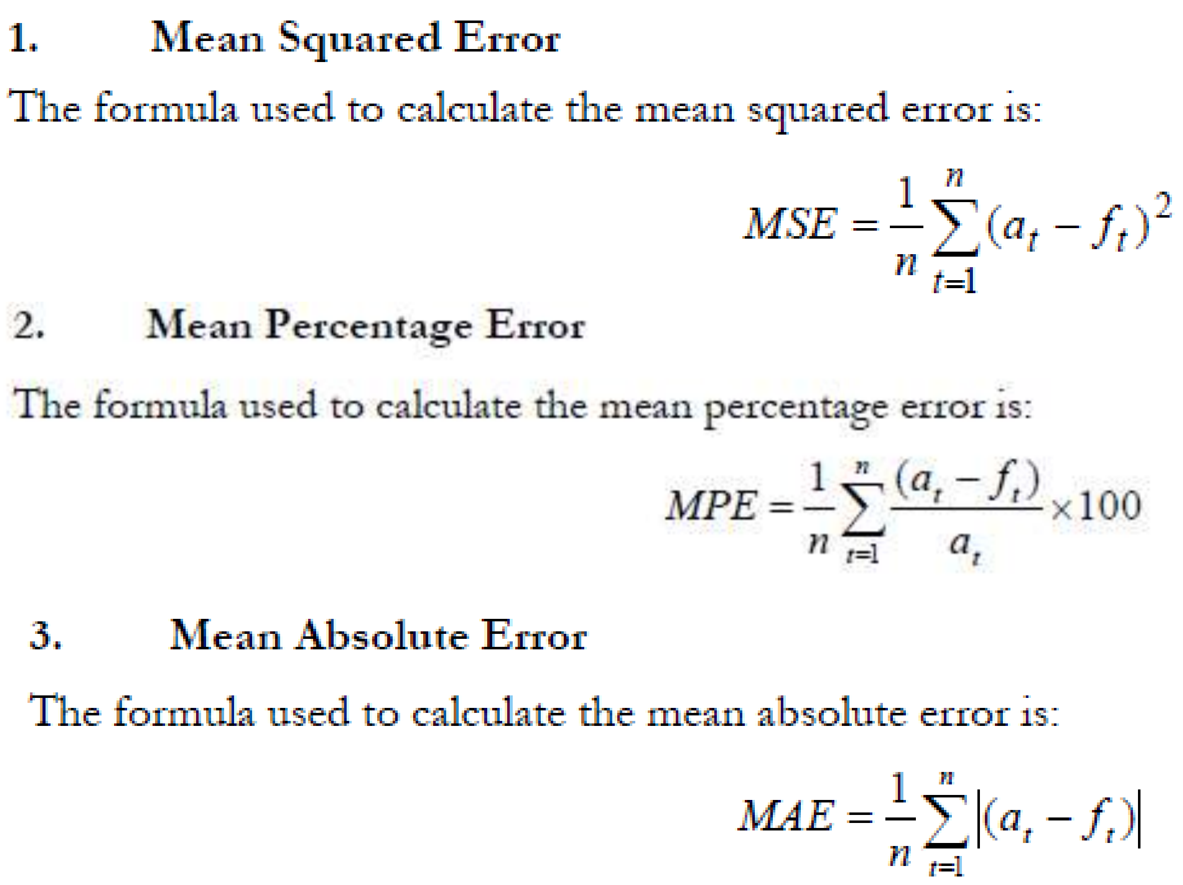


MEASURING THE MODEL ACCURACY



Technique	Abbrev	Measures
Mean Squared Error	MSE	The average of squared errors over the sample period
Mean Error	ME	The average dollar amount or percentage points by which forecasts differ from outcomes
Mean Percentage Error	MPE	The average of percentage errors by which forecasts differ from outcomes
Mean Absolute Error	MAE	The average of absolute dollar amount or percentage points by which a forecast differs from an outcome
Mean Absolute Percentage Error	MAPE	The average of absolute percentage amount by which forecasts differ from outcomes

MEASURING THE MODEL ACCURACY



1. **Mean Squared Error**

The formula used to calculate the mean squared error is:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(a_t - f_t)^2$$

2. **Mean Percentage Error**

The formula used to calculate the mean percentage error is:

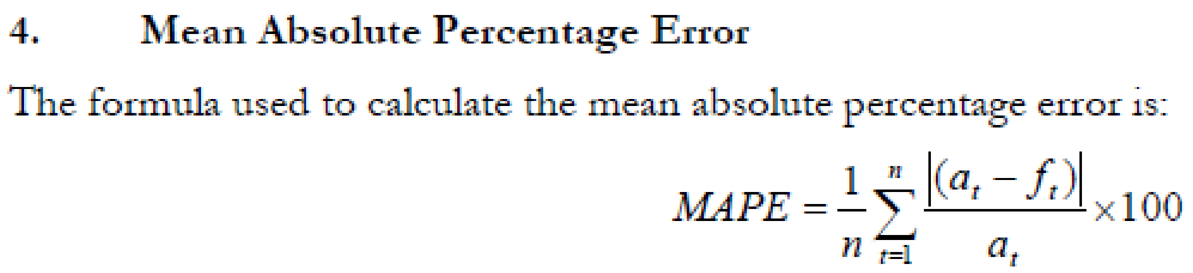
$$MPE = \frac{1}{n}\sum_{t=1}^{n}\frac{(a_t - f_t)}{a_t}\times 100$$

3. **Mean Absolute Error**

The formula used to calculate the mean absolute error is:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\left|(a_t - f_t)\right|$$

MEASURING THE MODEL ACCURACY



4. **Mean Absolute Percentage Error**

The formula used to calculate the mean absolute percentage error is:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|(a_t - f_t)|}{a_t}\times 100$$

Best Practice for Reporting Model Fit

1. Use Cross Validation to find the best model
2. Report the RMSE and MAPE statistics from the cross validation procedure
3. Report the R Squared from the model as you normally would.

The added cross-validation information will allow one to evaluate not how much variance can be explained by the model, but also the predictive accuracy of the model. Good models should have a high predictive AND explanatory power!

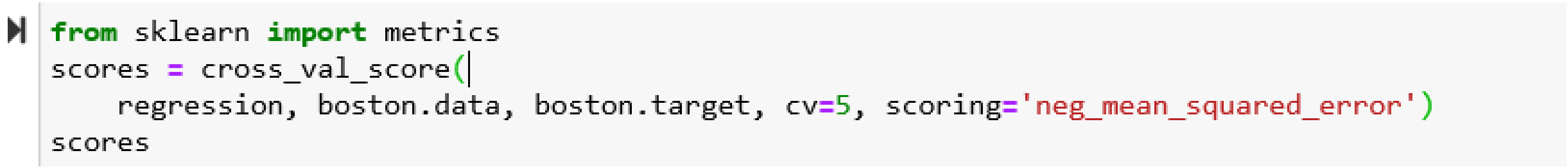
EXAMPLE

The following procedure is followed for each of the k "folds":

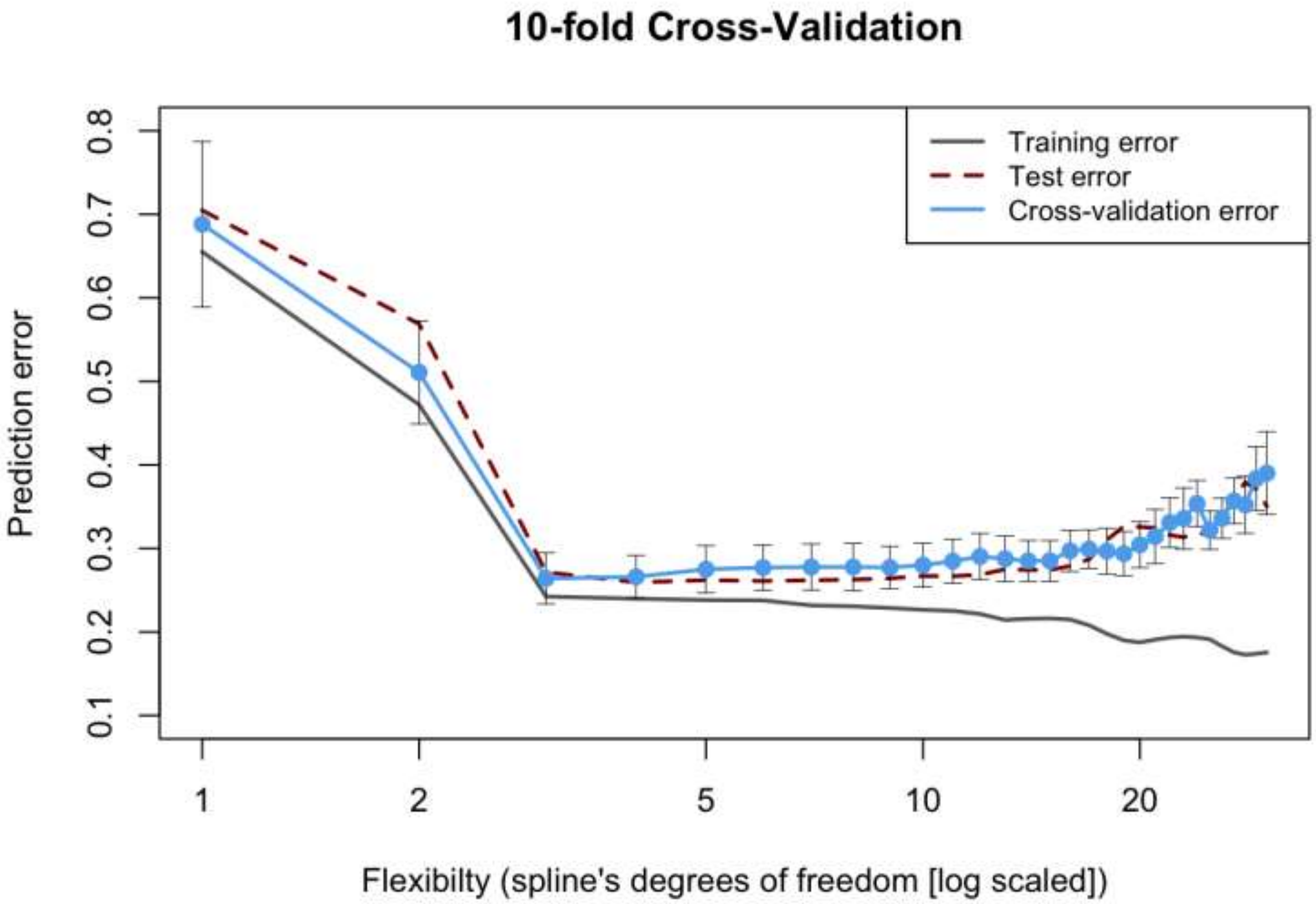
•A model is trained using k-1 of the folds as training data;

•the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

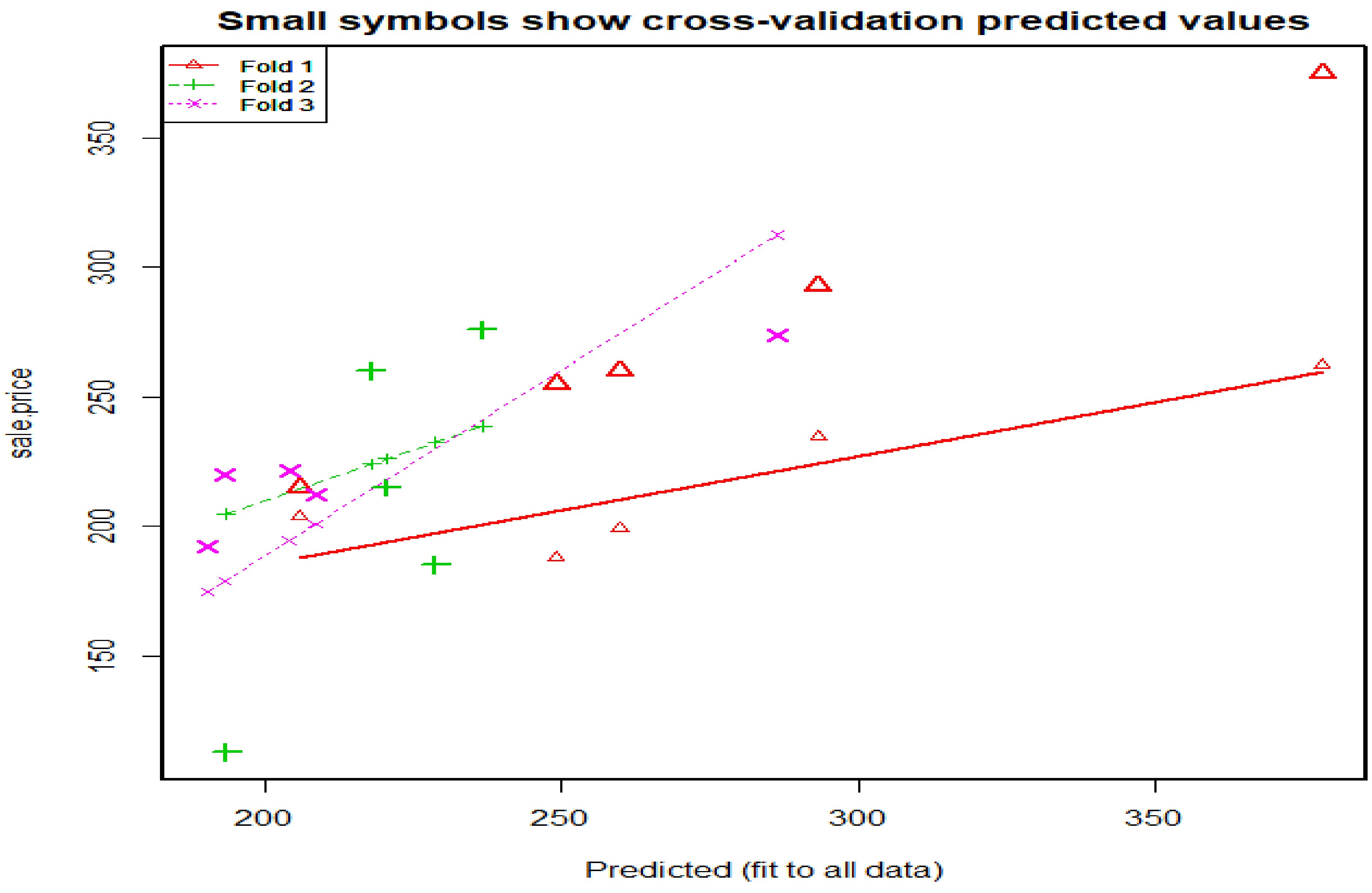
The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.



```
from sklearn import metrics
scores = cross_val_score(
    regression, boston.data, boston.target, cv=5, scoring='neg_mean_squared_error')
scores
```

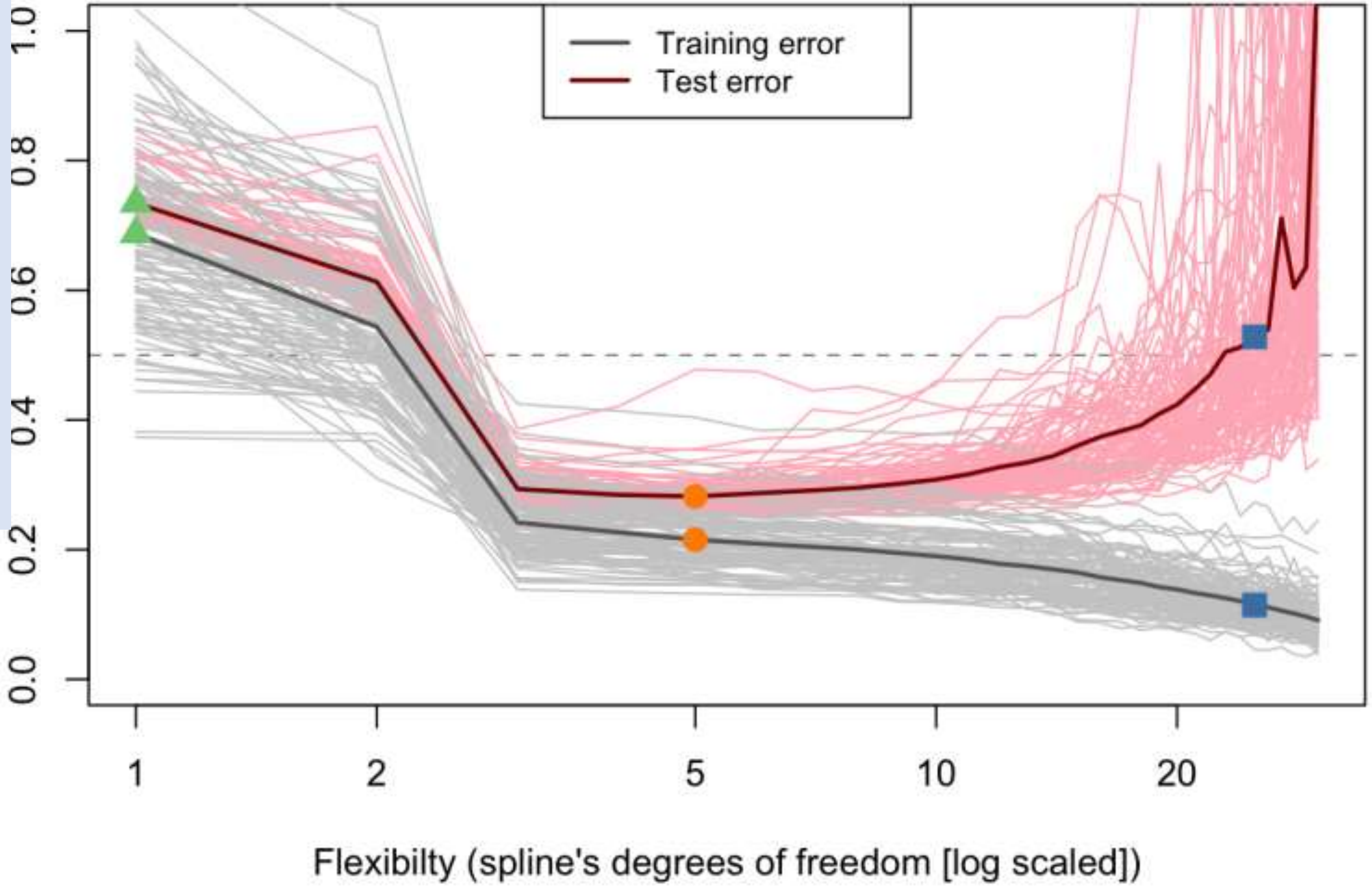


Often a “one-standard error” rule is used with cross-validation, according to which one should choose the most parsimonious model whose error is no more than one standard error above the error of the best model. In the example above, the best model (that for which the CV error is minimized) uses 3 degrees of freedom, which also satisfies the requirement of the one-standard error rule.


Small symbols show cross-validation predicted values
Fold 1
Fold 2
Fold 3
sale.price
150
200
250
300
350
200
250
300
350
Predicted (fit to all data)

For each training sample and fitted model, we compute the corresponding test error using a large test sample generated from the same population.

These are represented in the following plot together with their averages, which are shown using thicker lines. The solid points represent the three models illustrated in the previous diagram.



One can see that the training errors decrease monotonically as the model gets more complicated (and less smooth). On the other side, even if the test error initially decreases, from a certain flexibility level on it starts increasing again. The change point occurs in correspondence of the orange model, that is, the model that provides a good compromise between bias and variance. The reason why the test error starts increasing for degrees of freedom larger than 3 or 4 is the so called **overfitting** problem.