Underfitting, Overfitting and Cross-Validation

Prof. Mingkui Tan

SCUT Machine Intelligence Laboratory (SMIL)

Contents

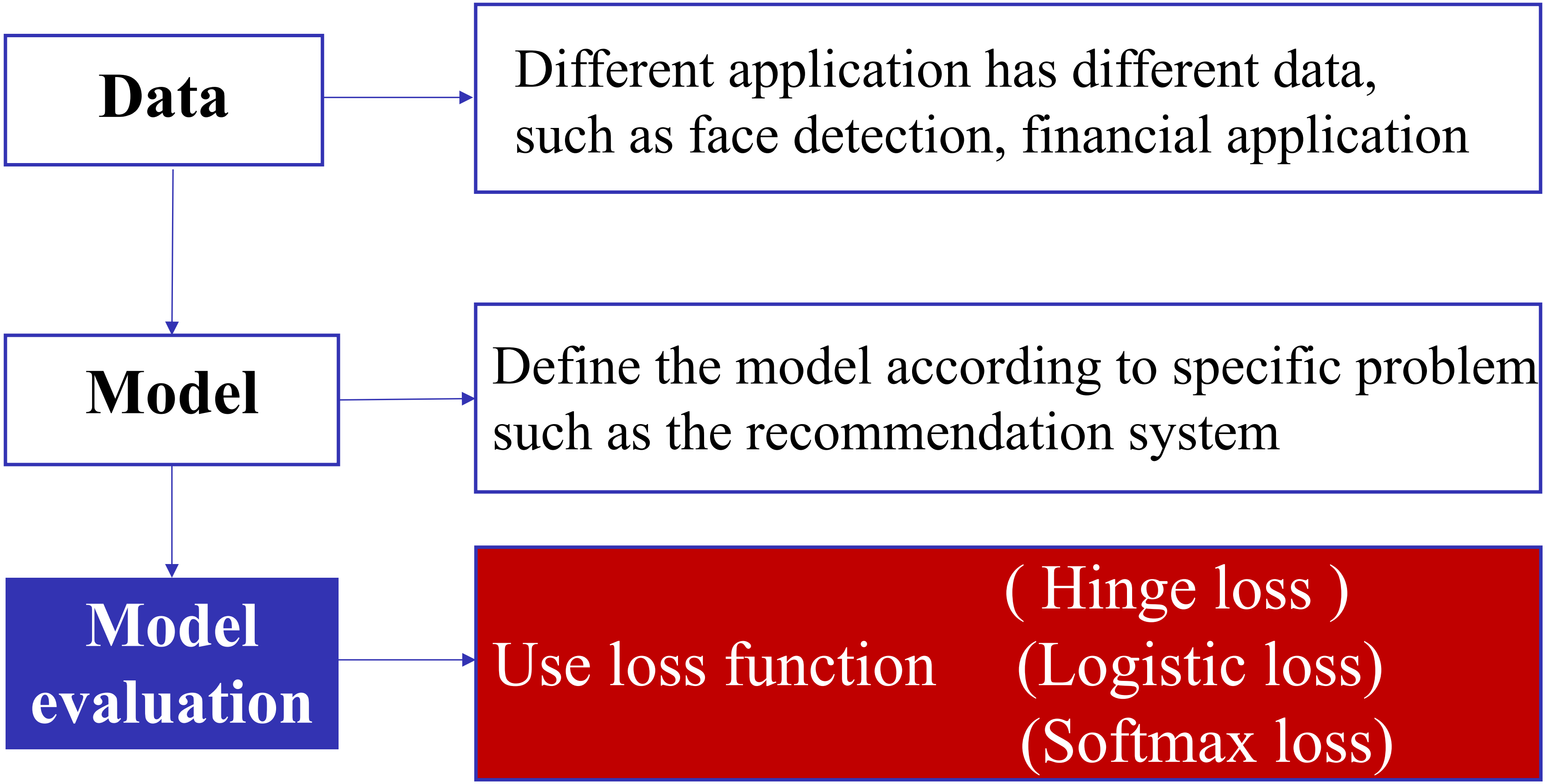
1 Training, Testing and Validation Split

2 Underfitting and Overfitting

3 Bias-Variance Trade-off

3 Cross-Validation

Main Elements of Machine Learning



Data

Different application has different data, such as face detection, financial application

Model

Define the model according to specific problem such as the recommendation system

Model evaluation

Use loss function (Hinge loss) (Logistic loss) (Softmax loss)

Given a data set *D*, how to evaluate the performance of a learned model?

Data Notation

Data:

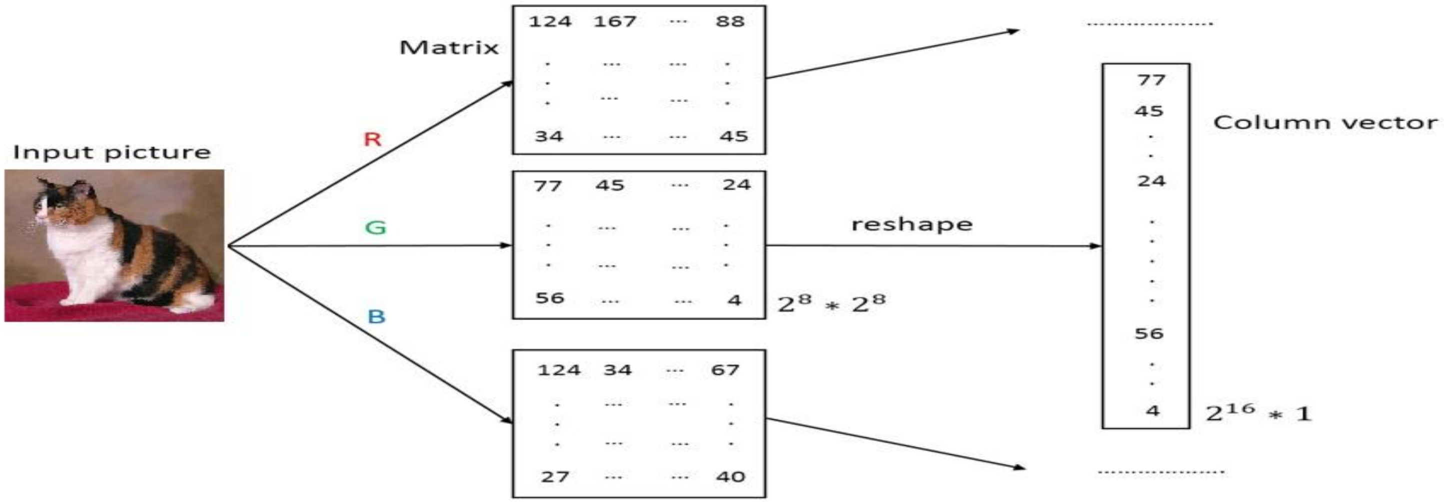
$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

$\mathbf{x}$ is the input, which is usually presented as a column vector

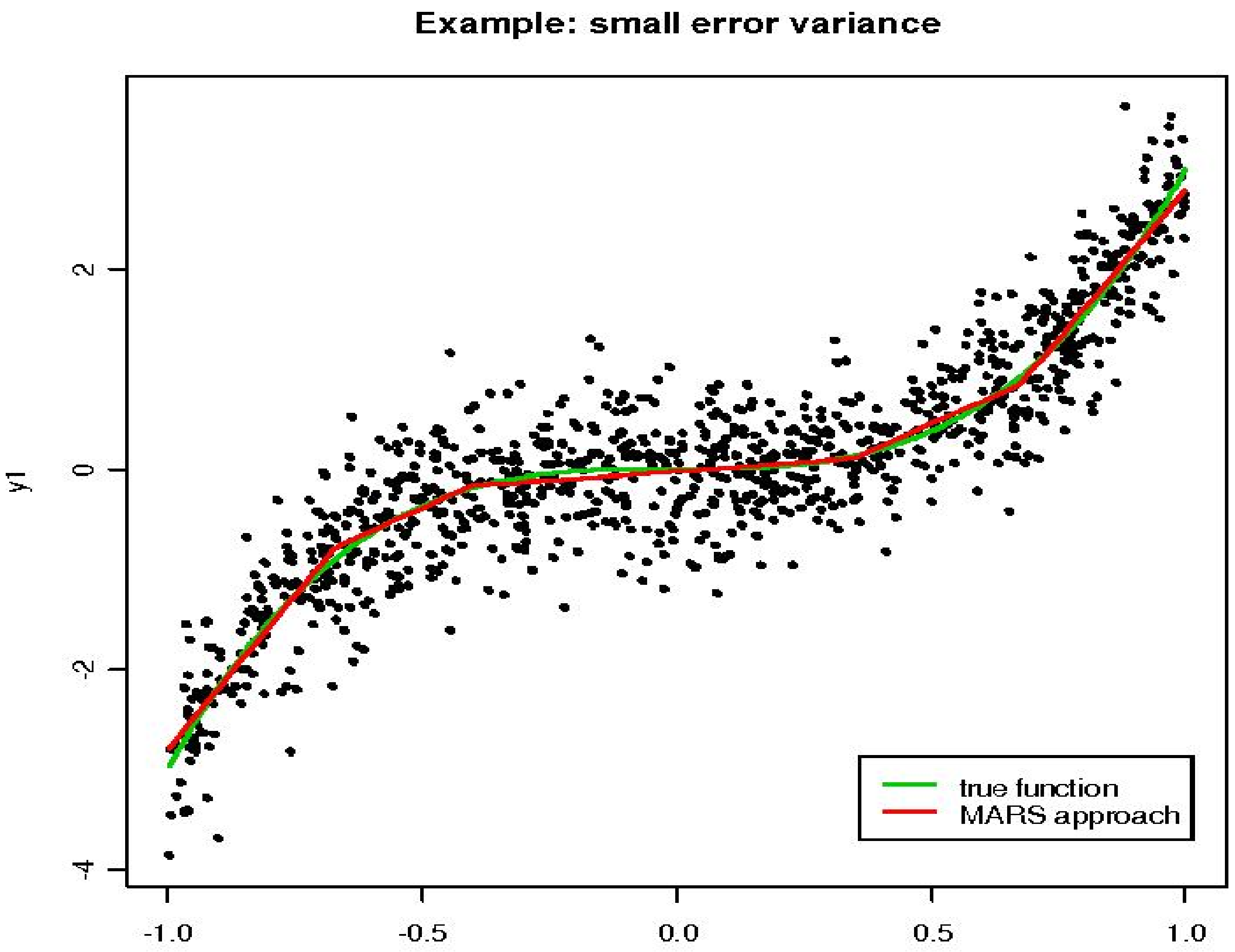
y is the output, for example, a person's name

n is the number of samples

For example, $\mathbf{x}$ can be a picture stored as a matrix:



Regression Example



Regression

Loss:

- Absolute value loss:

$$l(\hat{y}_i, y_i\ ) = |\hat{y}_i - y_i|$$

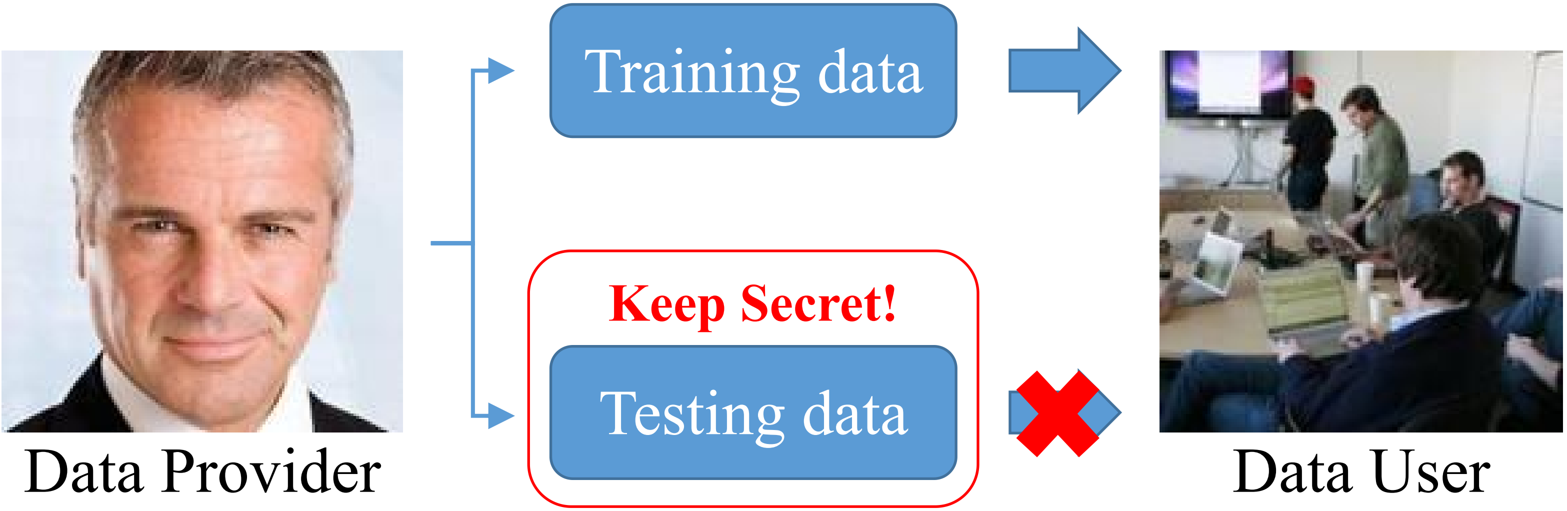
- Least squares loss:

$$l(\hat{y}_i, y_i) = \frac{1}{2}(\hat{y}_i - y_i)^2$$

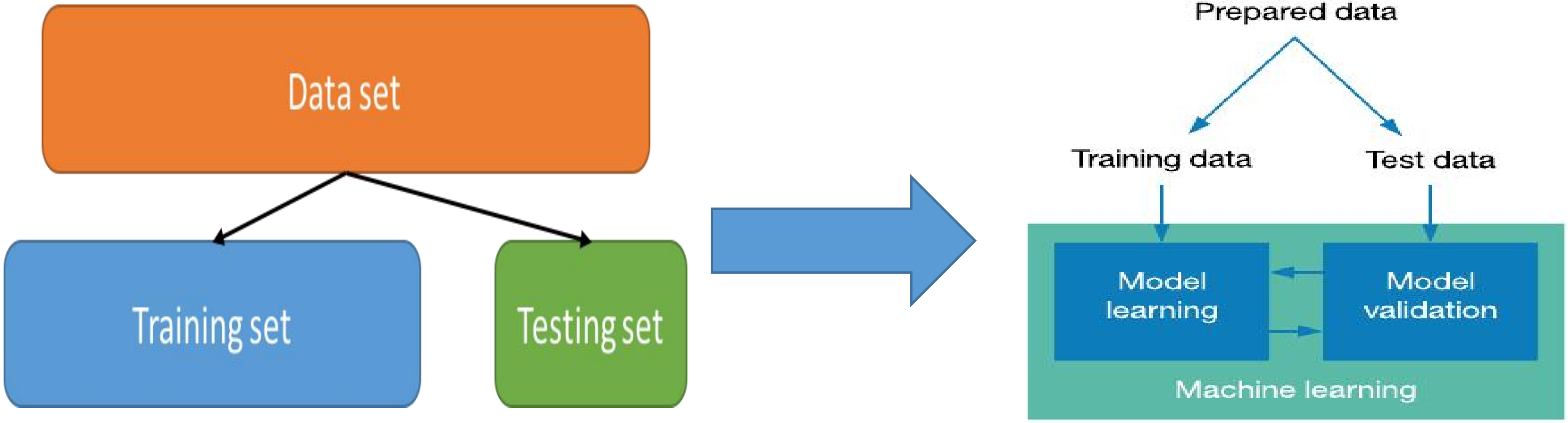
Given a data set *D*, how to evaluate the performance of a learned model?

Training-Testing-Validation Data Split

- Simple idea: Split data into two sets
  - Training set: Data used to fit the model
  - Testing set: Data used to evaluate the model

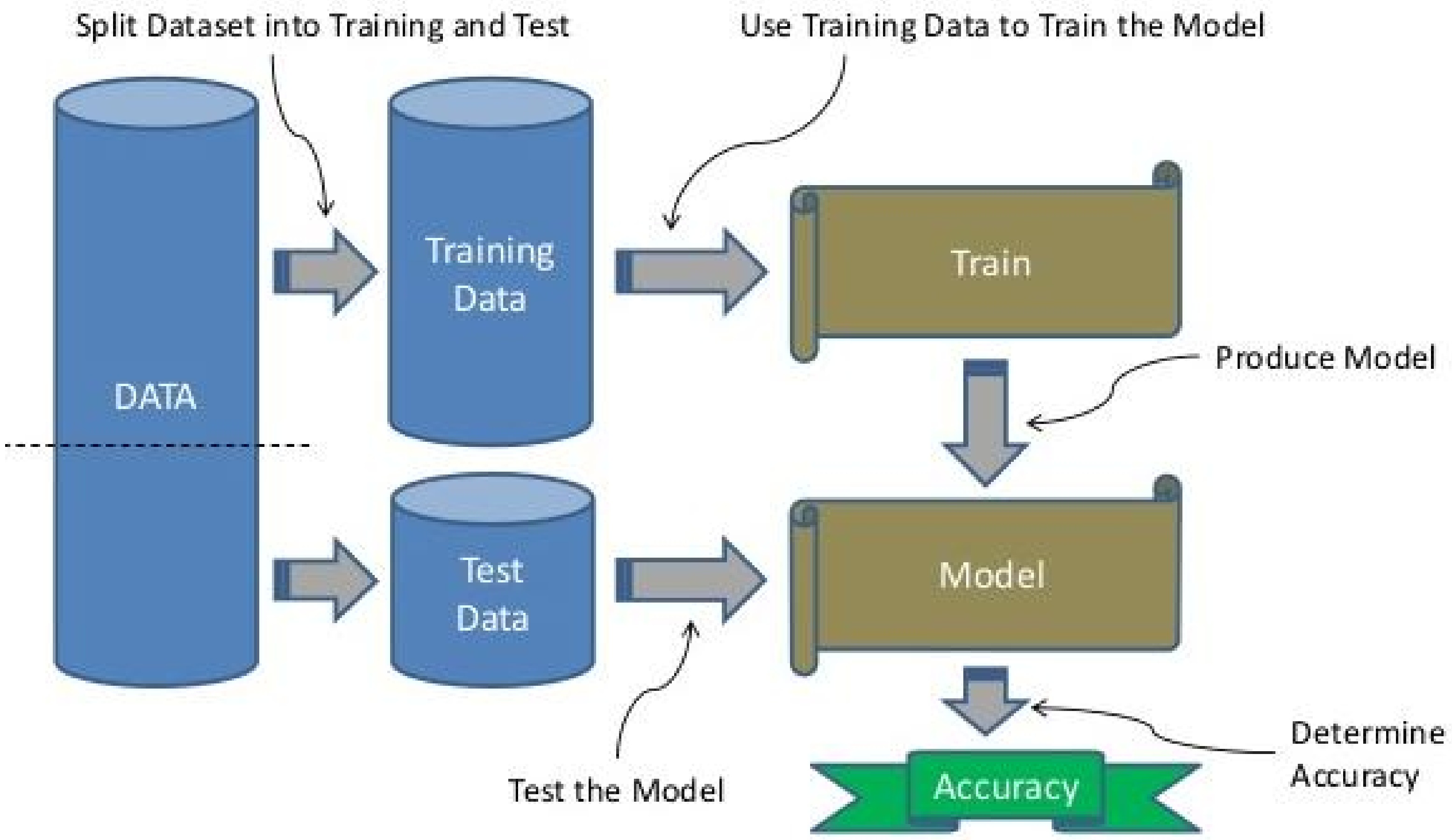


Data Split

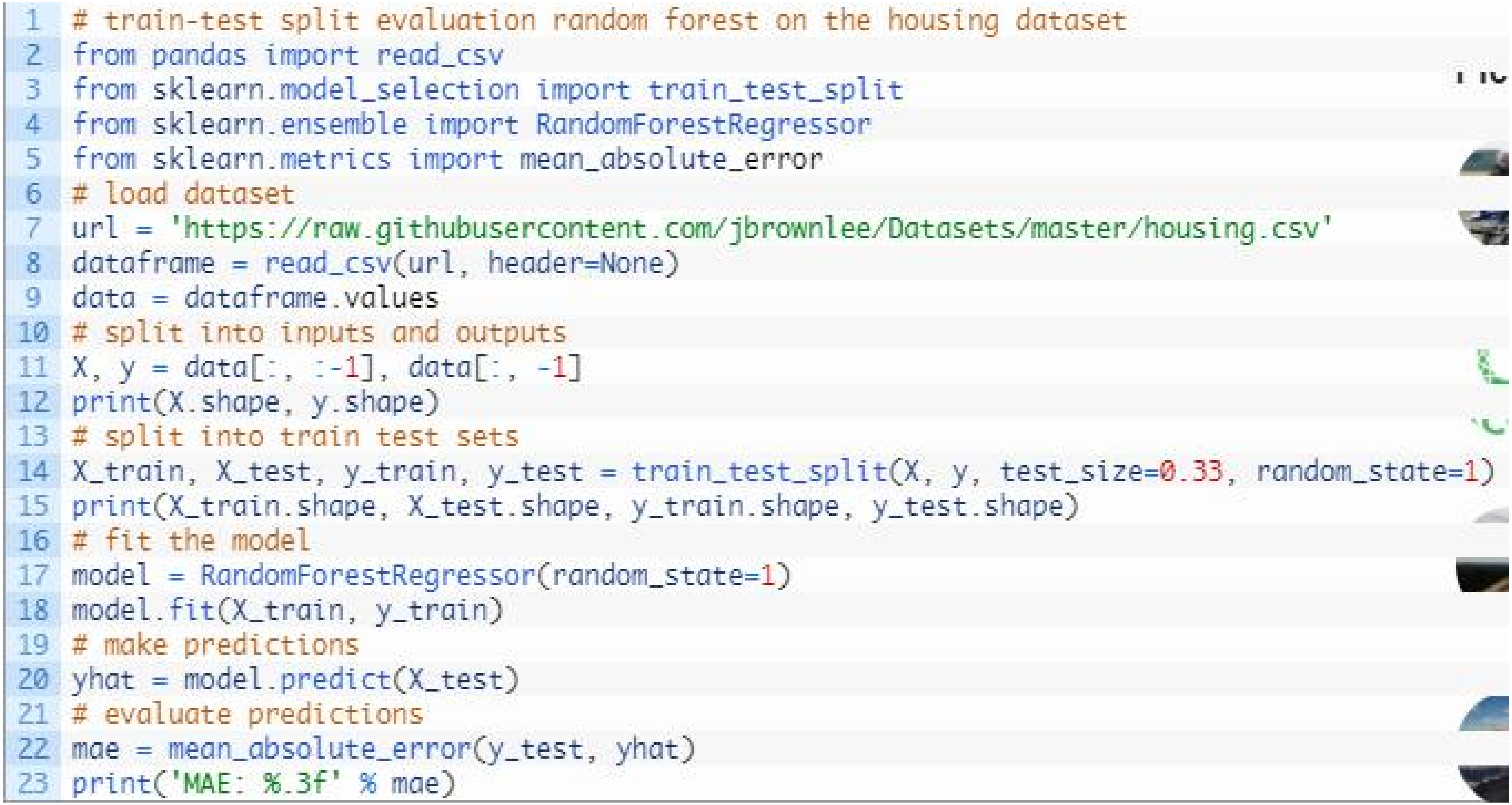


- Training set
  - $\approx 70\%$ of data, $D_{train} = \{x^{(i)}, y^{(i)}\}$, total n_{train} examples
- Testing set
  - $\approx 30\%$ of data, $D_{test} = \{x_{test}^{(i)}, y_{test}^{(i)}\}$, total n_{test} examples
- Choose examples randomly for training/testing split

Data Split for Training and Testing



Train-test split example

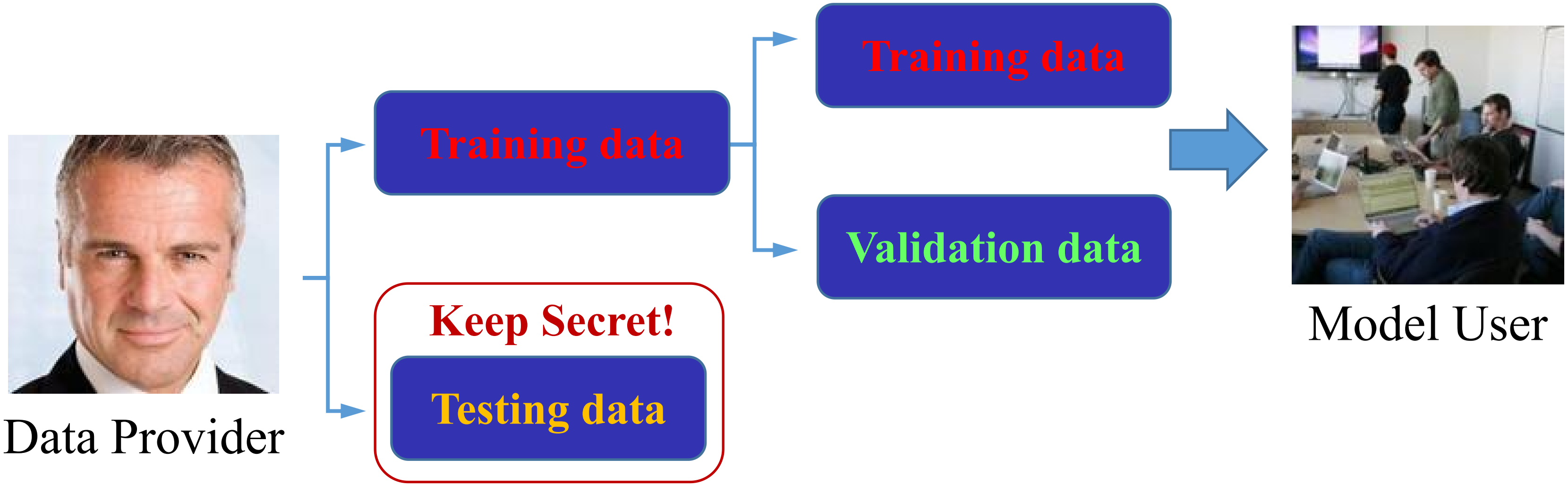


```python
# train-test split evaluation random forest on the housing dataset
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
# load dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
dataframe = read_csv(url, header=None)
data = dataframe.values
# split into inputs and outputs
X, y = data[:, :-1], data[:, -1]
print(X.shape, y.shape)
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
# fit the model
model = RandomForestRegressor(random_state=1)
model.fit(X_train, y_train)
# make predictions
yhat = model.predict(X_test)
# evaluate predictions
mae = mean_absolute_error(y_test, yhat)
print('MAE: %.3f' % mae)
```

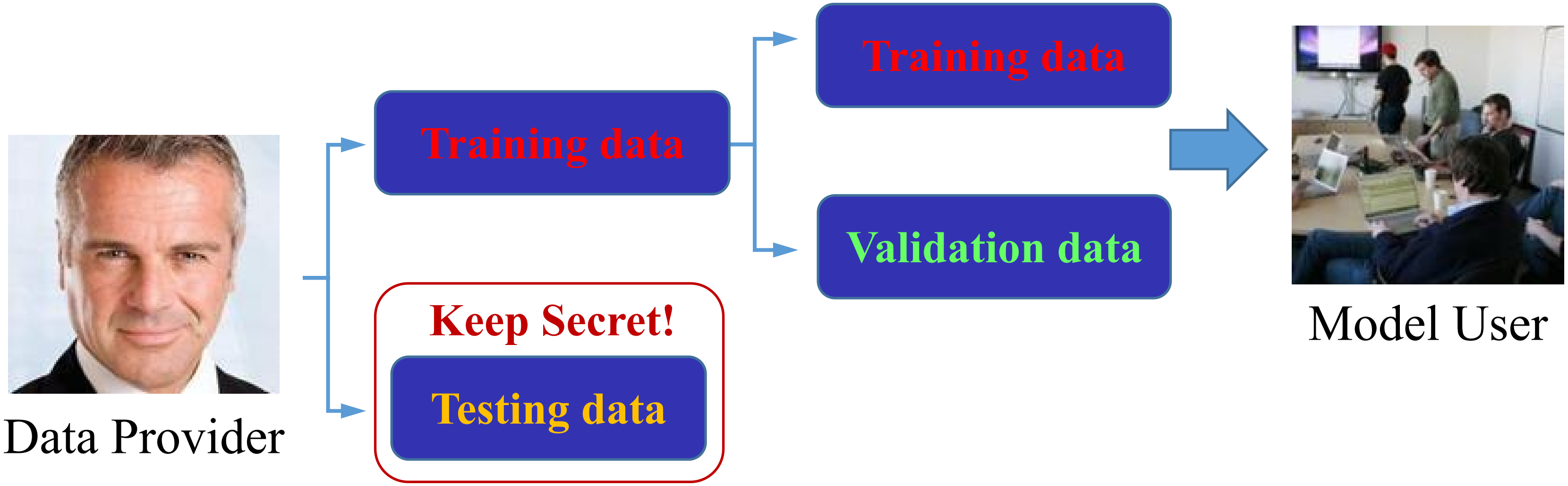
Source available: **https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/**

Training-Testing-Validation Data Split

- **Split data into three sets**
  - We still have training and testing sets
  - But additionally, we have a validation set to test the performance of our model depending on the parameter



Why We Need Validation Set?



Data Provider

Training data

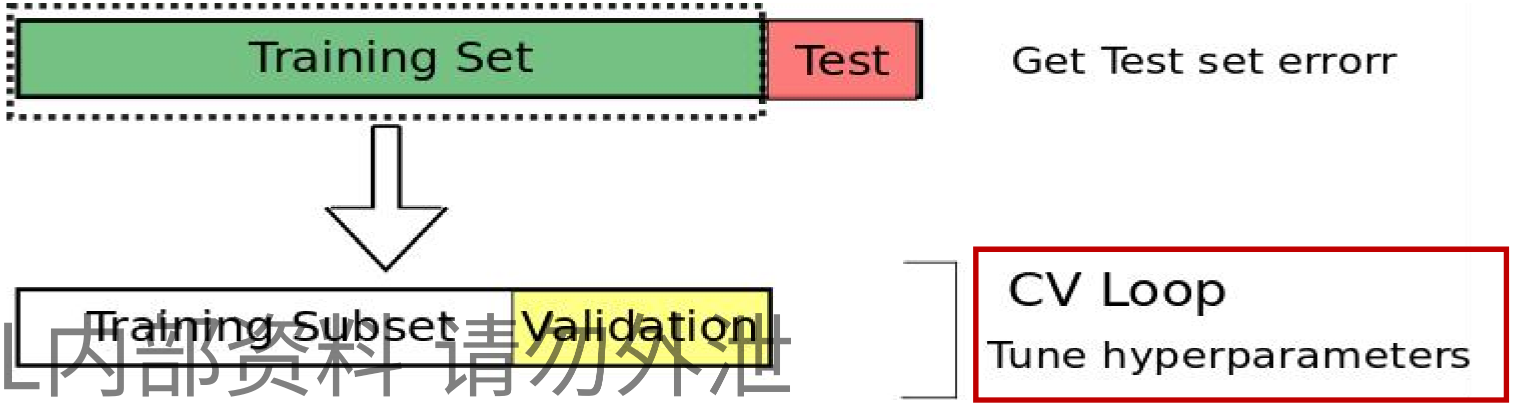
Keep Secret!

Testing data

Training data

Validation data

Model User



Training Set

Test

Get Test set errorr

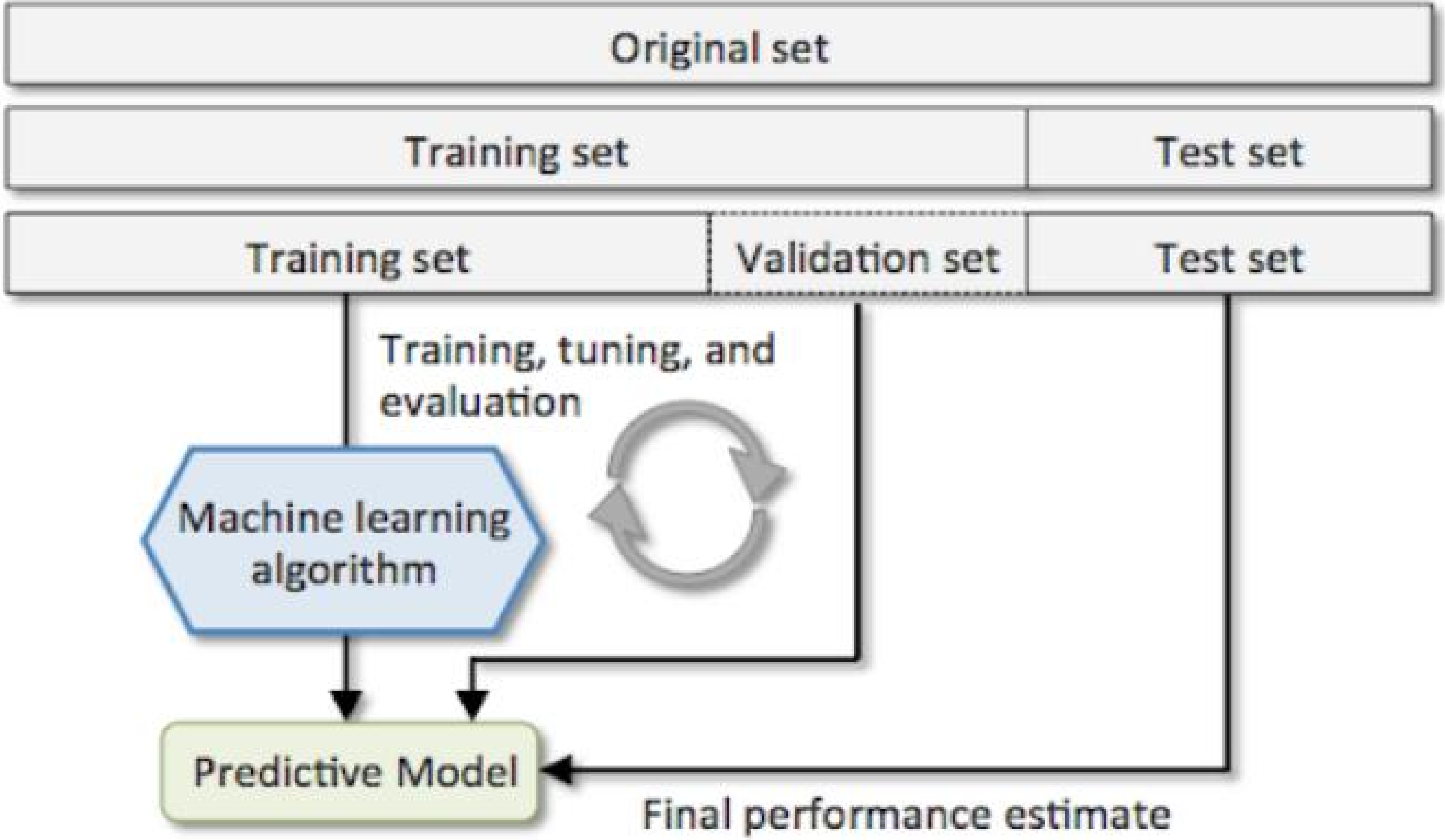
Training Subset

Validation

CV Loop

Tune hyperparameters

Why We Need Validation Set?


Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

Why We Need Validation Set?

- Business Reasons
  - Need to choose the best model
  - Measure accuracy/power of the selected model
  - Better to measure ROI of the modeling project
- Statistical Reasons
  - Model building techniques are inherently designed to minimize “loss” or “bias”
  - To an extent, a model will always fit “noise” as well as “signal”
  - If you just fit a bunch of models on a given dataset and choose the “best” one, it will likely be overly “optimistic”

Contents

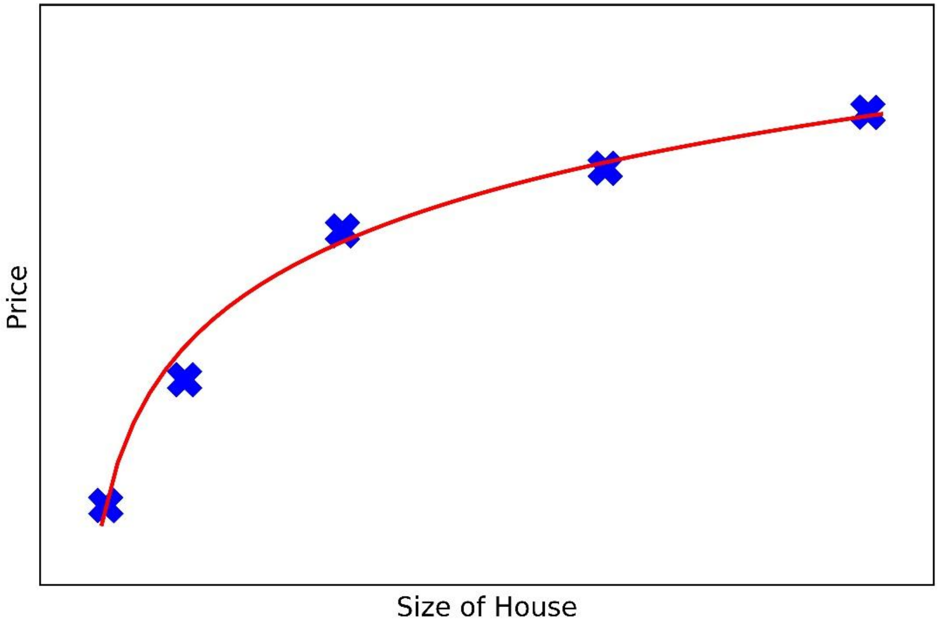
1 Underfitting and Overfitting

2 Bias-Variance Trade-off

3 Cross-Validation

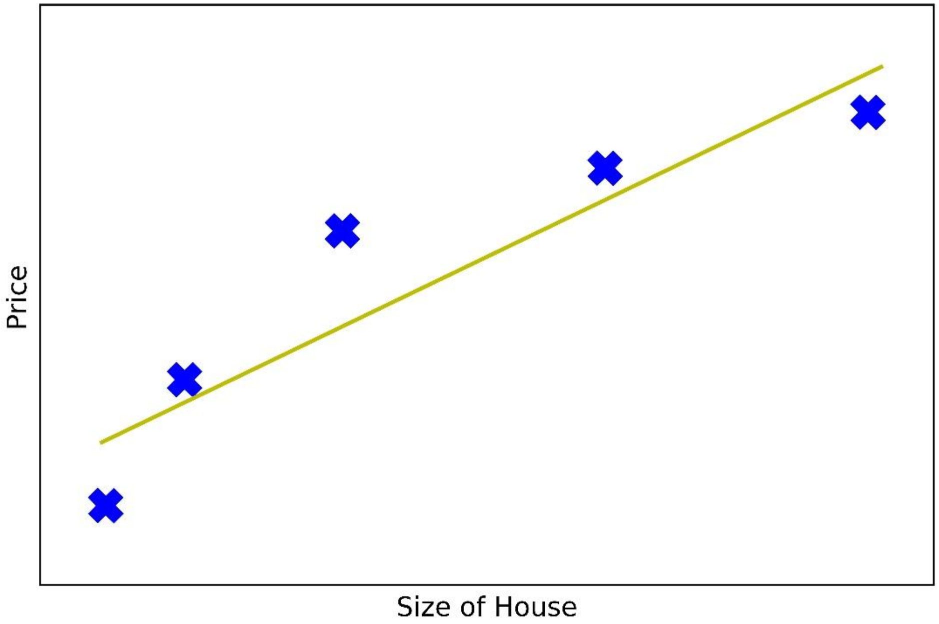
General Fitting Scheme

- Model is fitting and can capture the underlying trend of the data



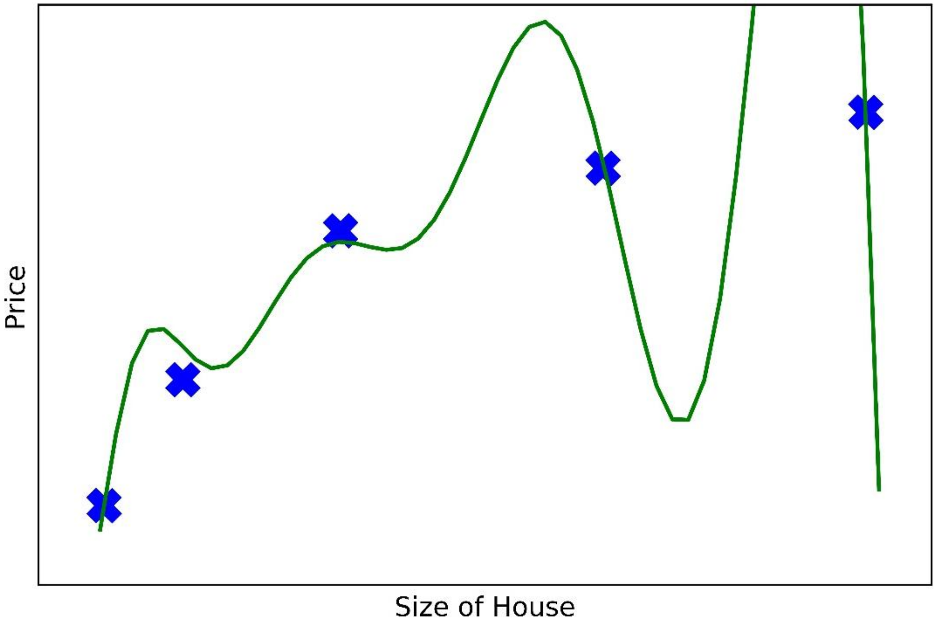
Underfitting

- Model cannot capture the underlying trend of the data

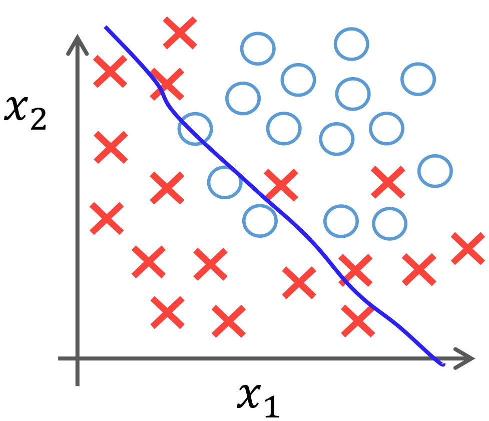


Overfitting

- The model is too complex to capture the true trend
- The model seeks to fit the noise or outlier of the data

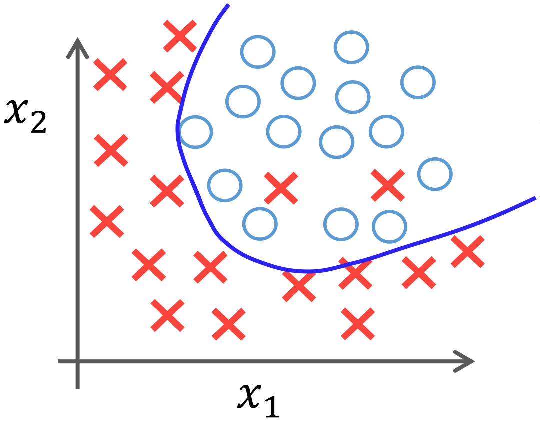


Underfitting vs Overfitting

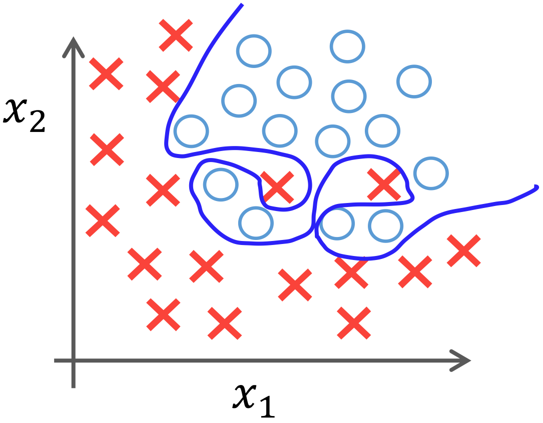


$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(g = sigmoid function)



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\ + \theta_3 x_1^2 + \theta_4 x_2^2 \\ + \theta_5 x_1 x_2)$$



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \\ + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 \\ + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \cdots$$

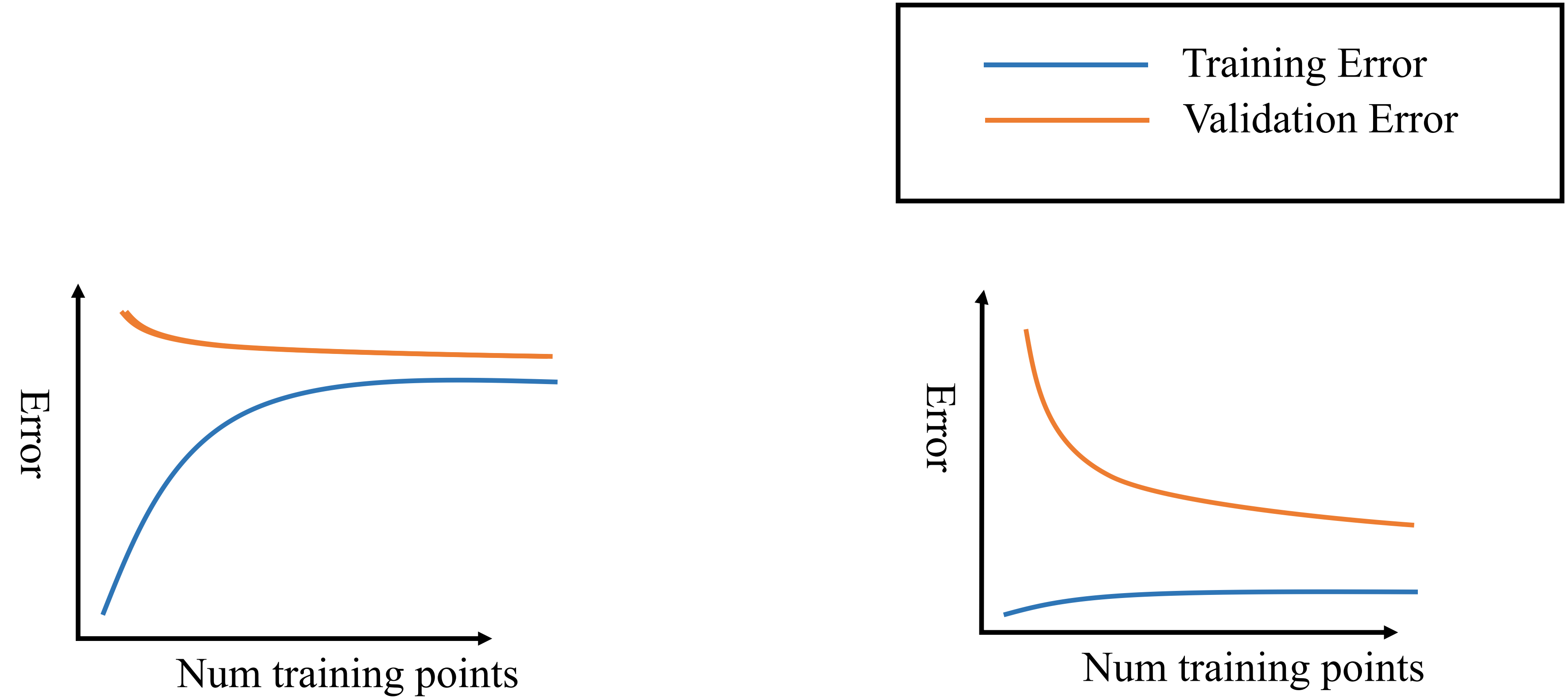
- Comprehend from Taylor expansion
    - The more terms, the more complex, the more power

$$f(x) = f(x_0) + \nabla f(x_0)^{\mathrm{T}}(x - x_0) + \frac{1}{2!}(x - x_0)^{\mathrm{T}} \nabla^2 f(x_0)(x - x_0) + \ldots$$

Signs of Underfitting and Overfitting

- How to judge underfitting or overfitting?
  - Underfitting: If the training set’s error is relatively large and the generalization error is large
    - Need to increase capacity (complexity of models)
  - Overfitting: If the training set’s error is relatively small and the generalization error is large
    - Need to decrease capacity (complexity of models)
    - Or increase training set

Signs of Underfitting and Overfitting



High Bias
(Underfitting)

High Variance
(Overfitting)

Contents

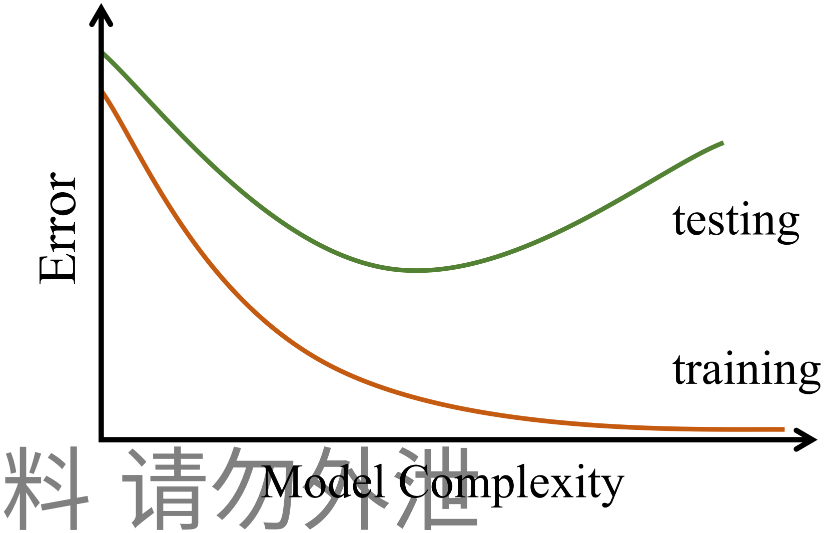
1 Underfitting and Overfitting

2 Bias-Variance Trade-off

3 Cross-Validation

Bias-Variance Trade-off

- Complex model
  - Too complex can diminish the model's accuracy on future data (called the Bias-Variance Trade-off)
  - Low Bias
    - Model fits well on the training data
  - High Variance
    - Model is more likely to make a wrong prediction



Contents

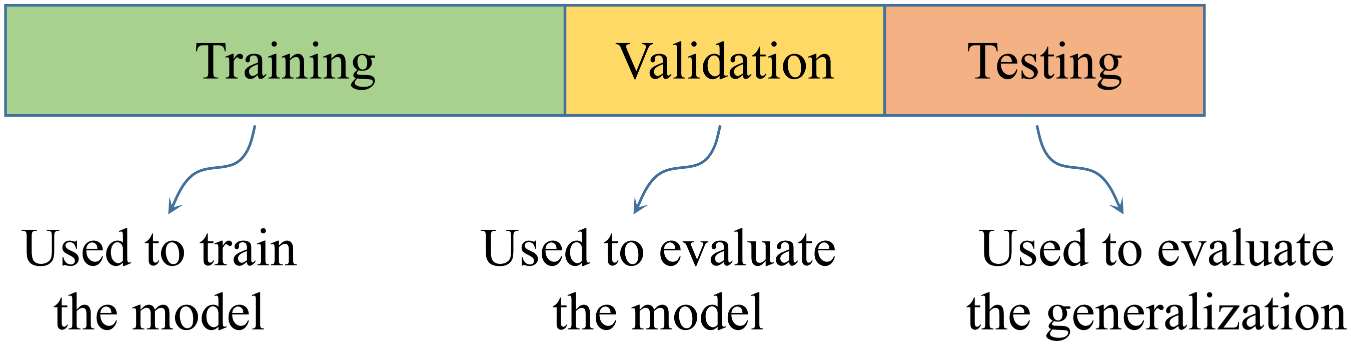
1 Underfitting and Overfitting

2 Bias-Variance Trade-off

3 Cross-Validation

Tuning Learning Parameter

- Use validation set for tuning hyper-parameters
- Use testing set only for final evaluation



Training

Validation

Testing

Used to train the model

Used to evaluate the model

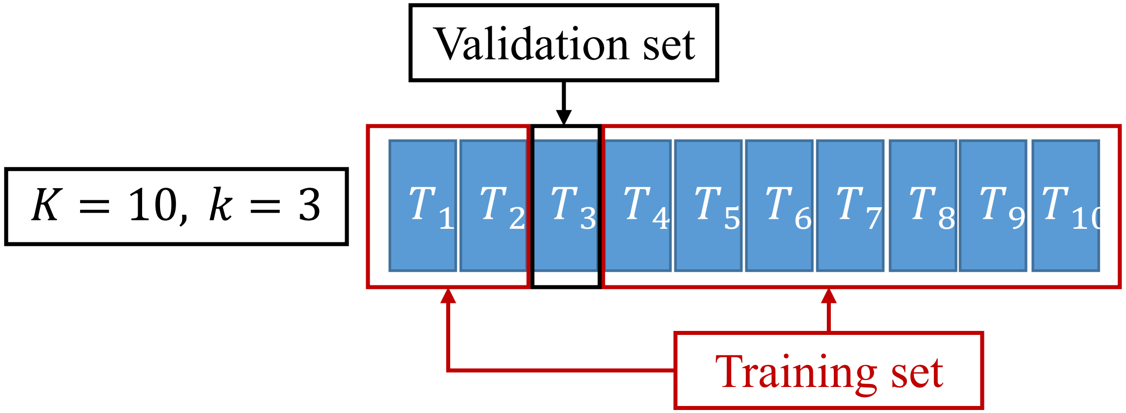
Used to evaluate the generalization

K-Fold Cross-validation

- If we want to reduce variability in the data
  - We can use multiple rounds of cross-validation using different partitions
  - And then, average the result overall rounds
- Given a data *S* sampled from the population *D*

K-Fold Cross-validation

- Split data S into K equal disjoint subsets $(T_1, \cdots, T_K)$
- Perform the following steps for $k = 1, \cdots, K$
  - Use $R_k = S - T_k$ as the training set
  - Build classifier C_k using R_k
  - Use T_k as the validation set, compute error $Err_k = error(C_k, T_k)$
- Let $Err^{ave} = \frac{1}{K}\sum_{k=1}^{K} Err_k$
  - This is the averaged error rate


Validation set
$K = 10,\ k = 3$
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10}
Training set

Validation for Evaluation

- Training error

  $$J_{train}(\theta) = \frac{1}{2m}\sum cost\left(x^{(i)}, y^{(i)}\right)$$

- Validation error

  $$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum cost\left(x_{cv}^{(i)}, y_{cv}^{(i)}\right)$$

- For model selection
  - Obtain $\theta^{(1)}, \cdots, \theta^{(d)}$ and select the best (lowest) $J_{cv}\left(\theta^{(i)}\right)$
- Testing error

  $$J_{test}(\theta) = \frac{1}{2m_{test}}\sum cost\left(x_{test}^{(i)}, y_{test}^{(i)}\right)$$

- For final evaluation
  - Evaluate generalization error $J_{test}\left(\theta^{(i)}\right)$ on the testing set

Tuning Regularzation Parameter λ

- Suppose we are fitting a model with high-order polynomial

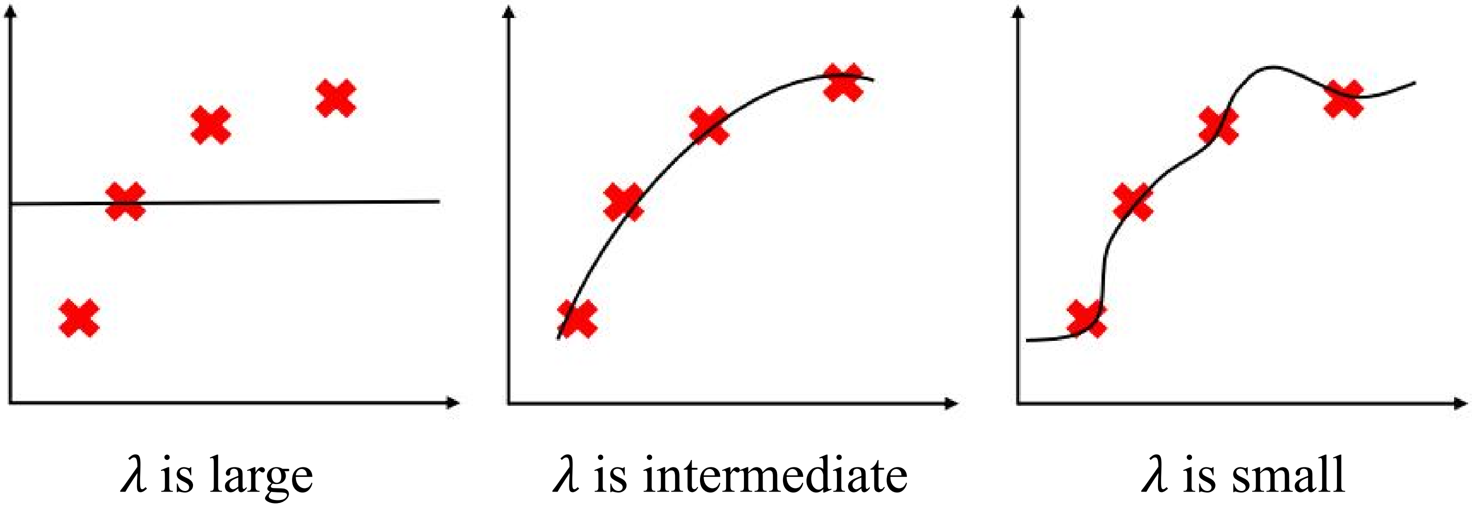
$$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_n x^n$$

- To prevent overfitting, we use regularization

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} cost(h_\theta(x_i), y_i) + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Tuning Regularzation Parameter λ

- If λ is too large, all θ are penalized and $\theta_1 \approx \theta_2 \approx \cdots \approx 0$, $h_\theta(x) \approx \theta_0$
- If λ is intermediate, the model fits well
- If λ is too small, the model fits too well, i.e. overfitting



Tuning Regularzation Parameter λ

- Choose good λ on the validation set
  - Choose a range of possible values for $\lambda(0.02, \cdots, 0.24)$
  - That gives us 12 models with different parameters � to check
  - For each λ_i:
    - Learn θ_i
    - Calculate $J_{cv}(\theta_i)$
    - Take λ_i with lowest $J_{cv}(\theta_i)$
  - Finally, we report the test error as $J_{test}(\theta_i)$

Tuning Learning Parameter

- Choosing λ with K-Fold Cross-validation
  - Split your data into training, validation, and testing set
  - For every possible value λ, estimate the error rate
  - Select λ with least average error rage Err^{ave}
  - Final evaluation of the testing set

Thank You