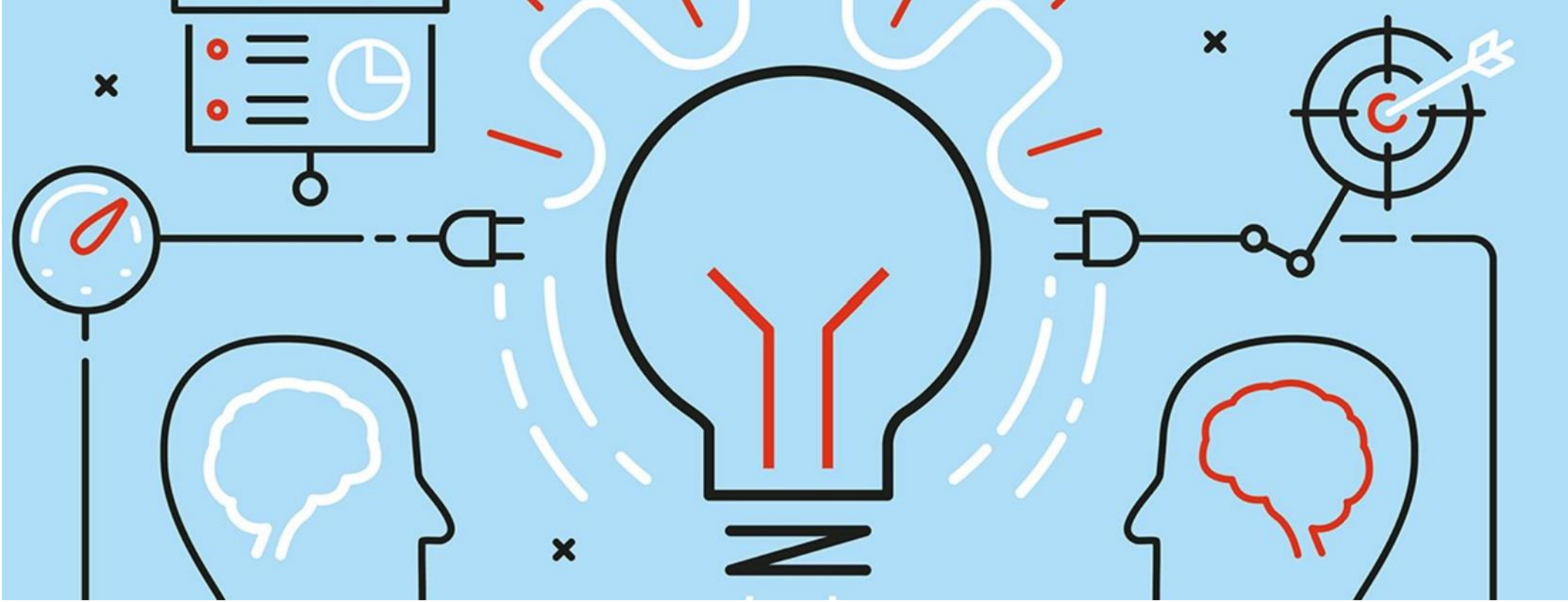


Supervised Machine Learning

(The Data School)

- AI Xin
- ai_xin@np.edu.sg
- School of InfoComm Technology
- Ngee Ann Polytechnic

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Learning Objectives

Upon completion of this course, you should be able to:

- Understand supervised machine learning models
- Select, Build, Train and Evaluate the Models
- Tuning model hyperparameters to achieve the best performance
- Apply the models to solve the real-life problems

Topics

1. Introduction (recap on Linear & Logistic Regression)
2. K-Nearest Neighbors
3. Decision Tree Model
4. Support Vector Machine
5. Ensemble Learning and Random Forest

1. Introduction to Supervised Machine Learning

Machine Learning

What is Machine Learning?

- Machine Learning is the science (and art) of programming computers so they can learn from data.

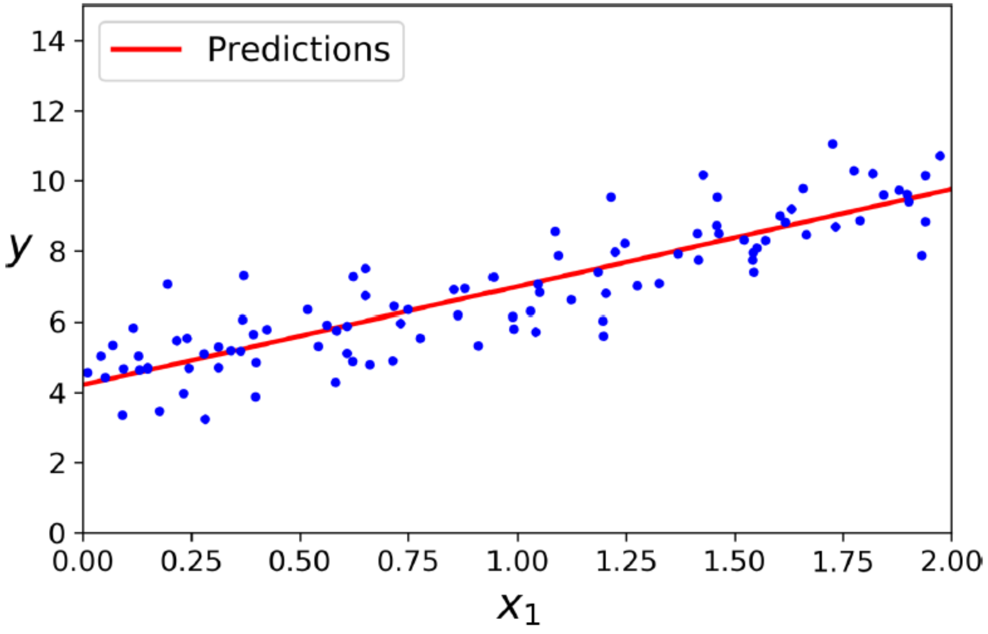
- A slightly more general definition:

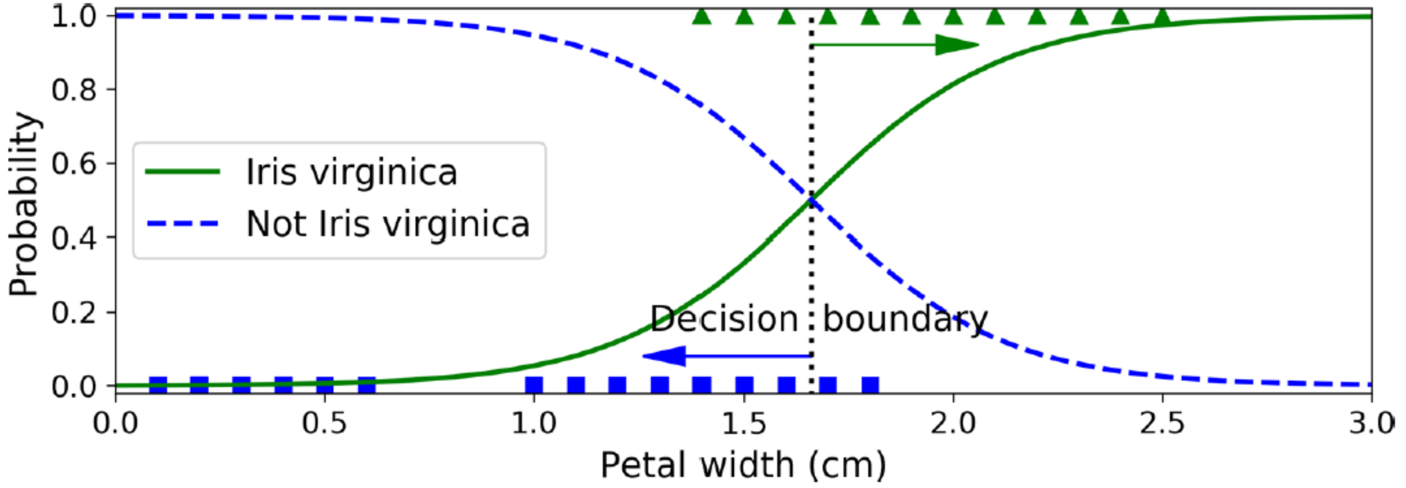
  [Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

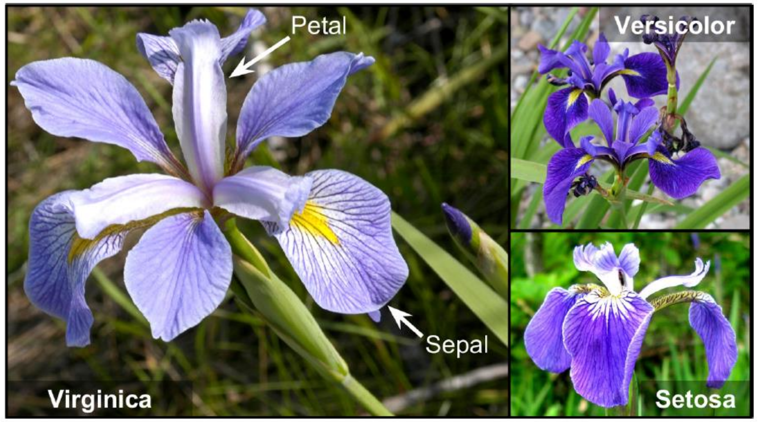
  —Arthur Samuel, 1959

Logistic Regression

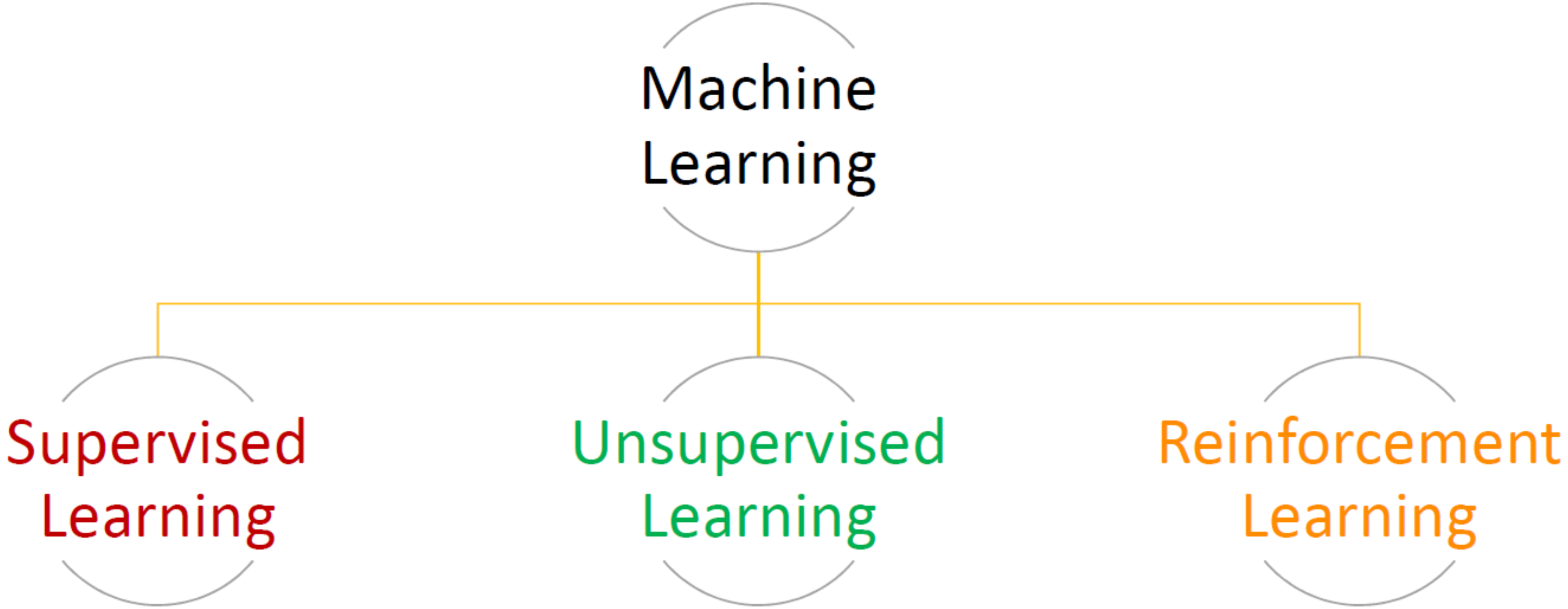
Linear Regression





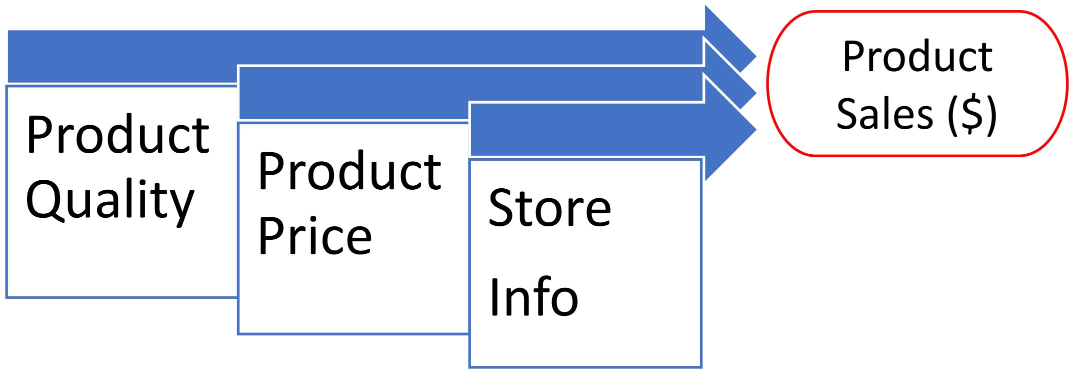


Types of Machine Learning



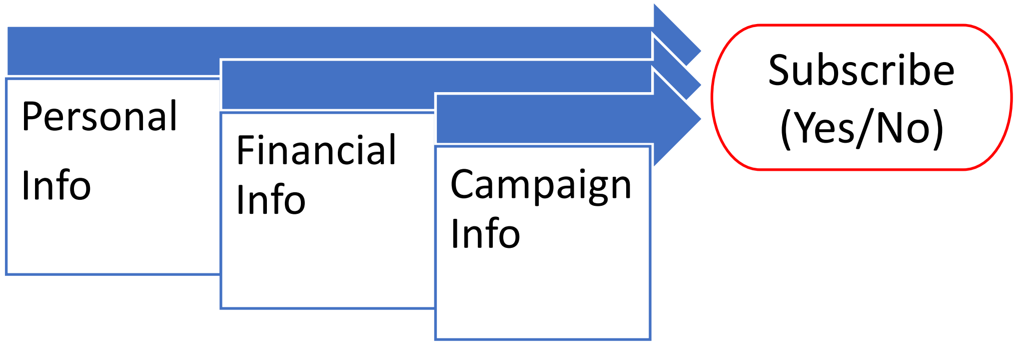
Supervised ML

Regression



Supermarket Sales Forecast

Classification



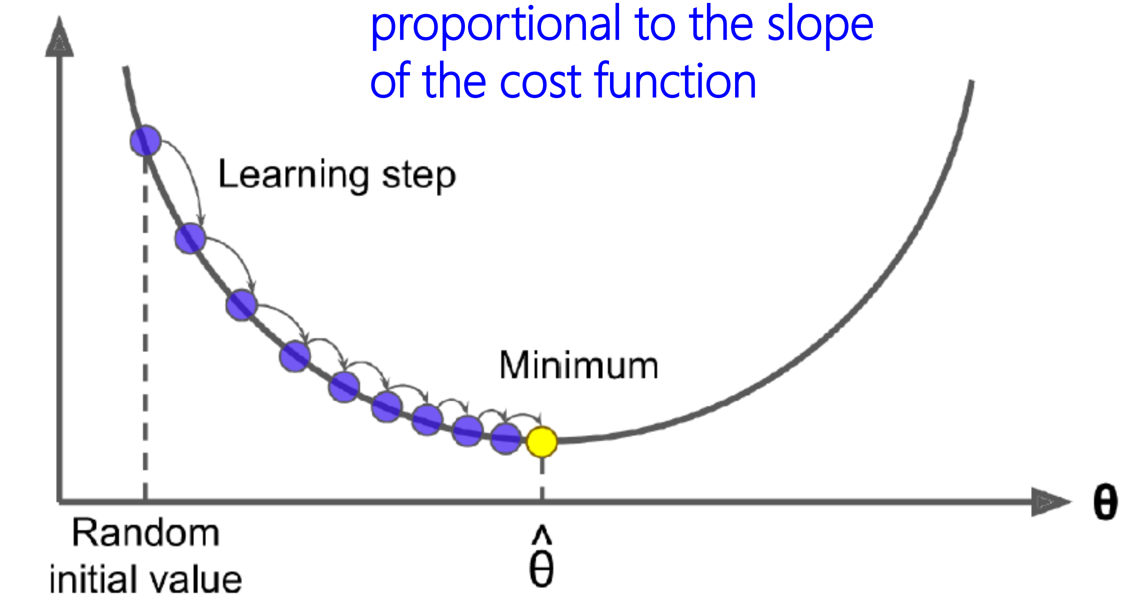
Bank Marketing Campaign

Challenges of ML

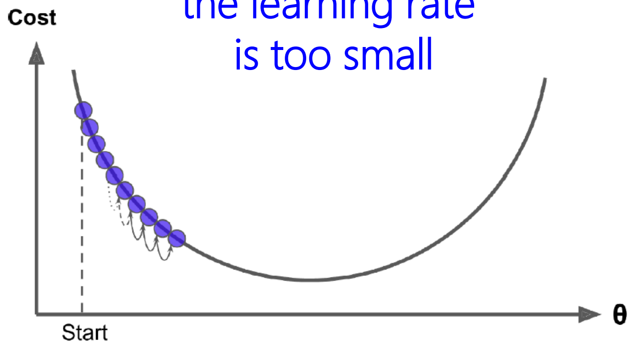
- Insufficient Quantity of Training Data
- Nonrepresentative Training Data
- Poor-Quality Data
- Irrelevant Features
- Overfitting the Training Data
- Underfitting the Training Data

Gradient Descent

the learning step size is proportional to the slope of the cost function

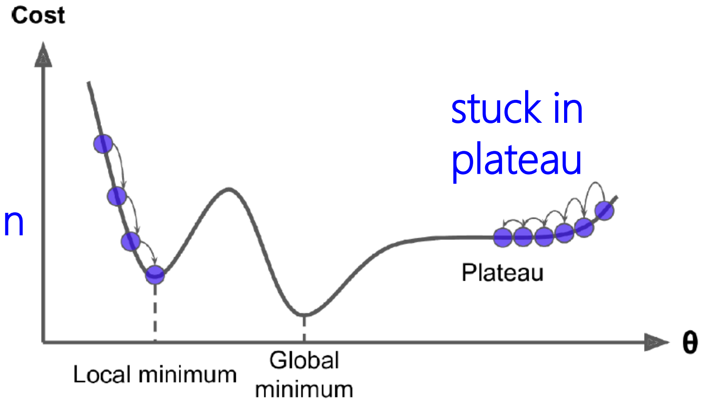


the learning rate is too small

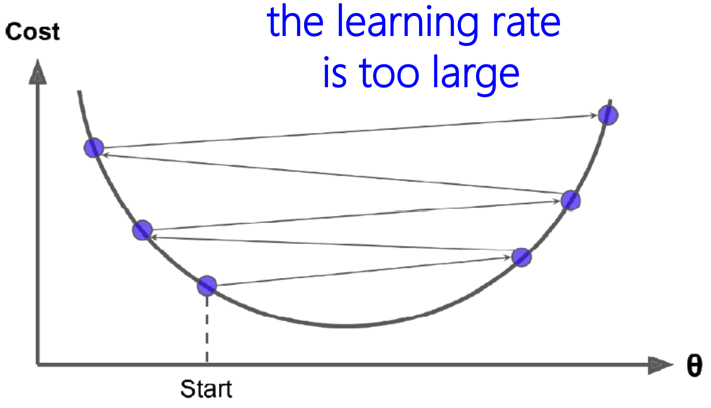


stuck in local min

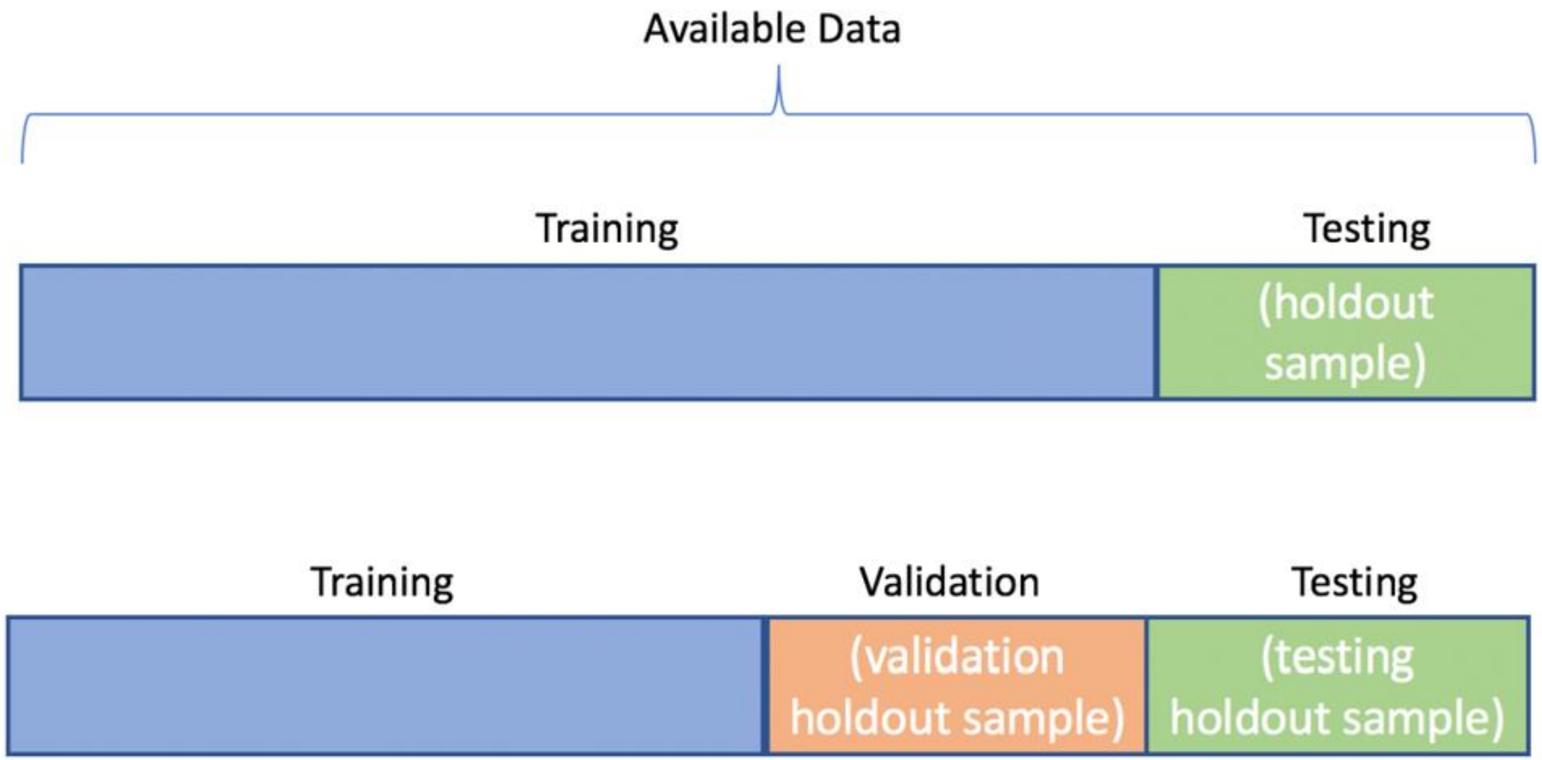
stuck in plateau



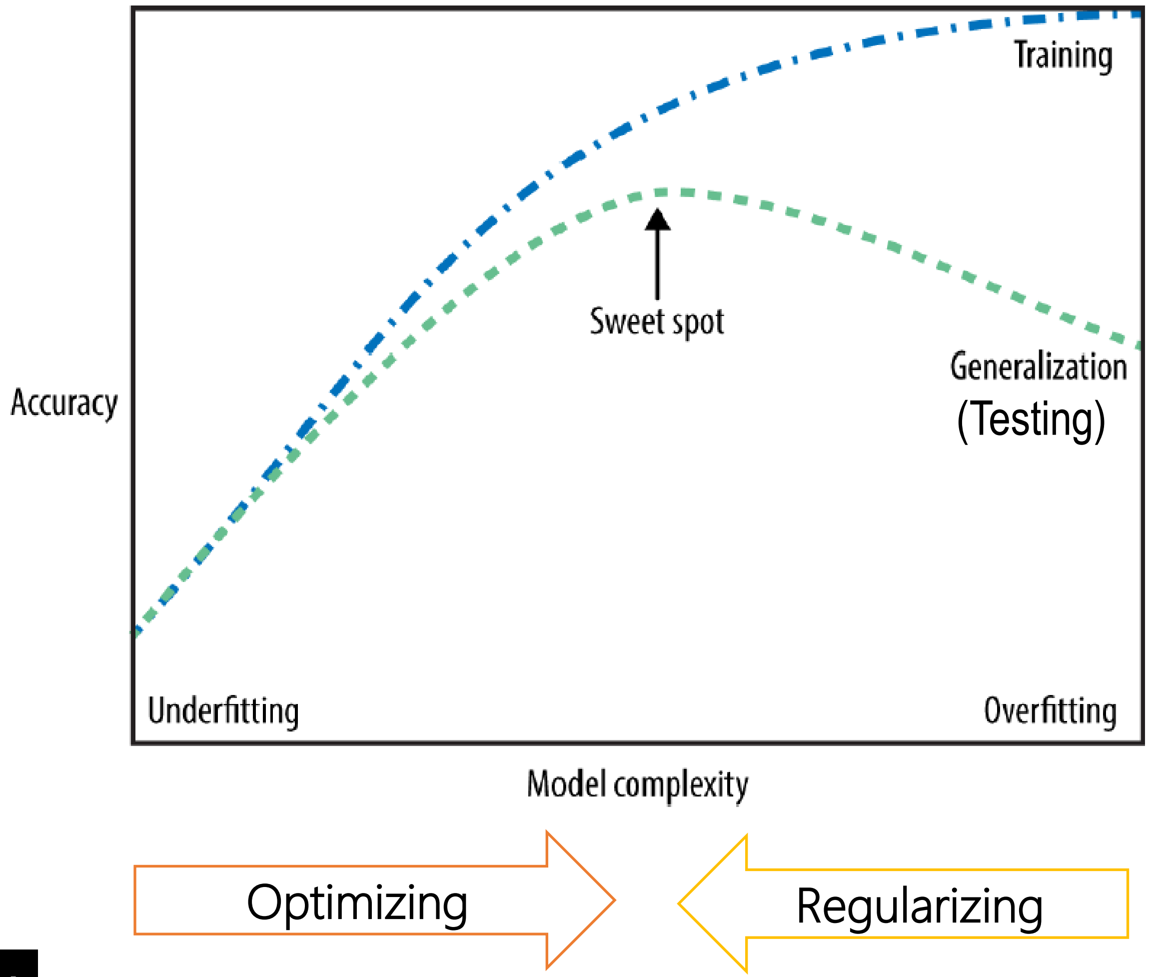
the learning rate is too large



Testing and Validating



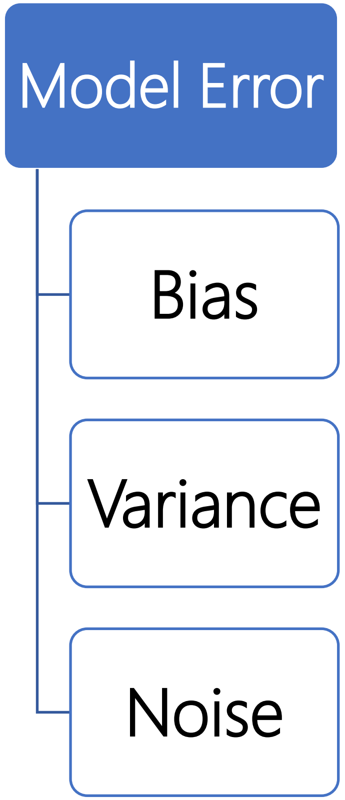
Generalization, Overfitting and Underfitting

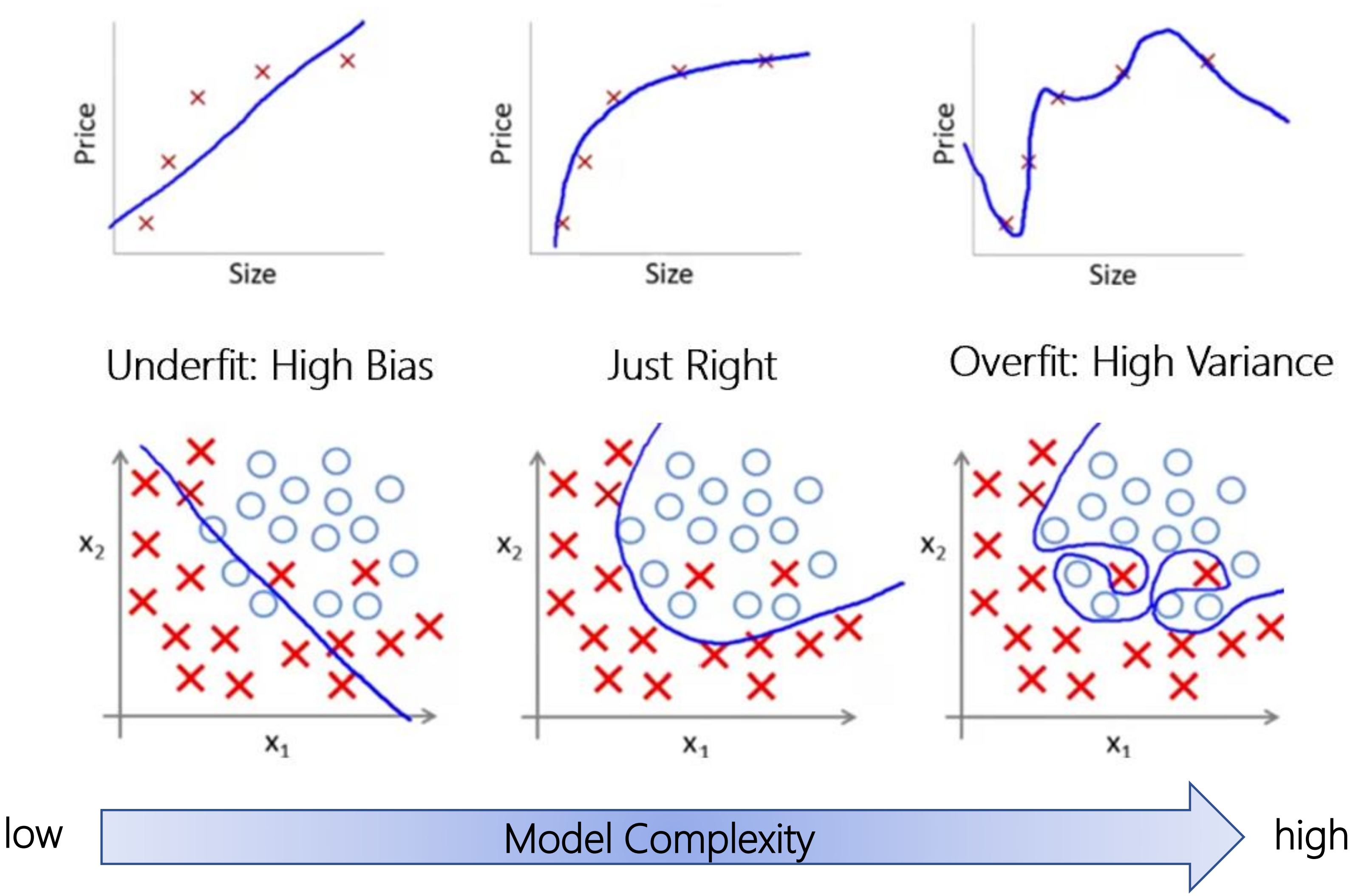


Balancing Optimization and Generalization

Tradeoff of Model Complexity against Training and Testing accuracy

The Bias-Variance Tradeoff

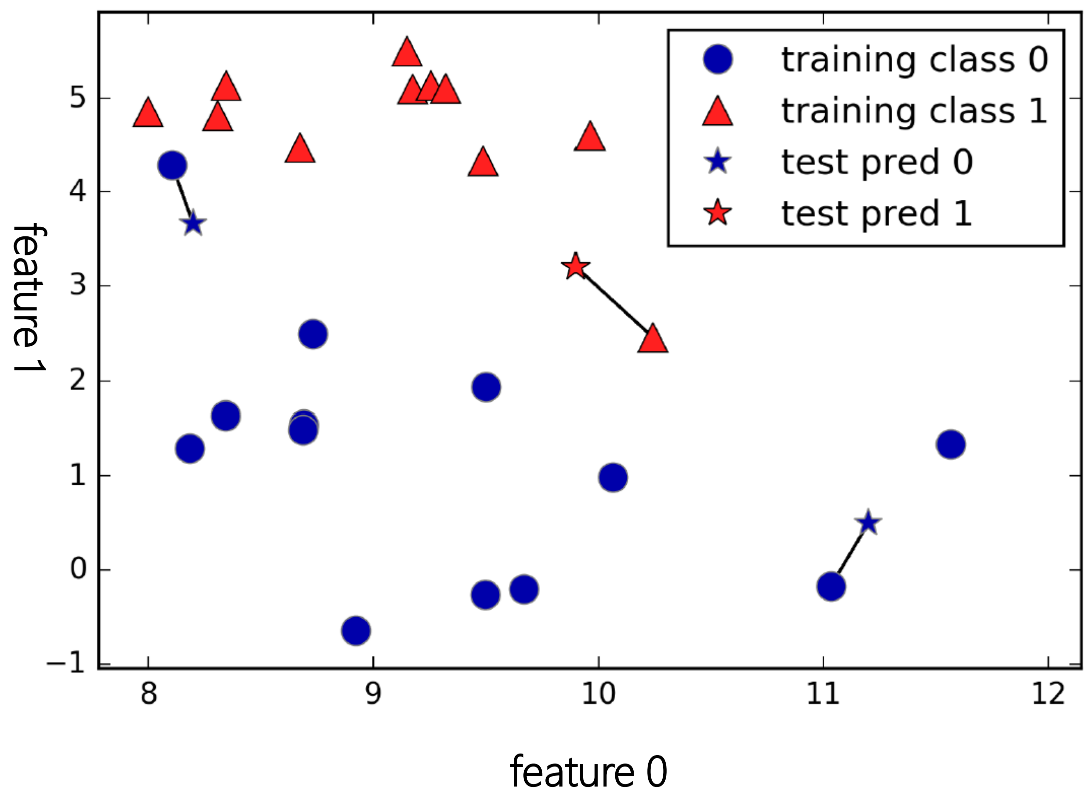




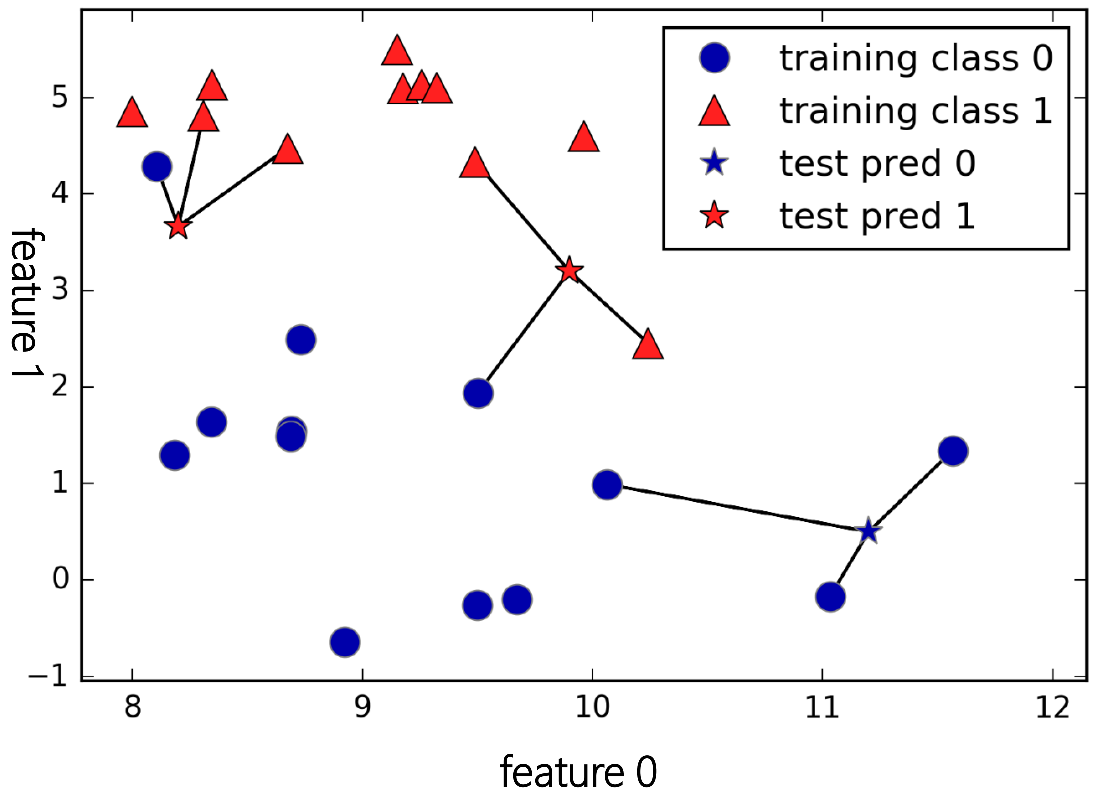
2. K-Nearest Neighbors

Classification

one-nearest-neighbor model

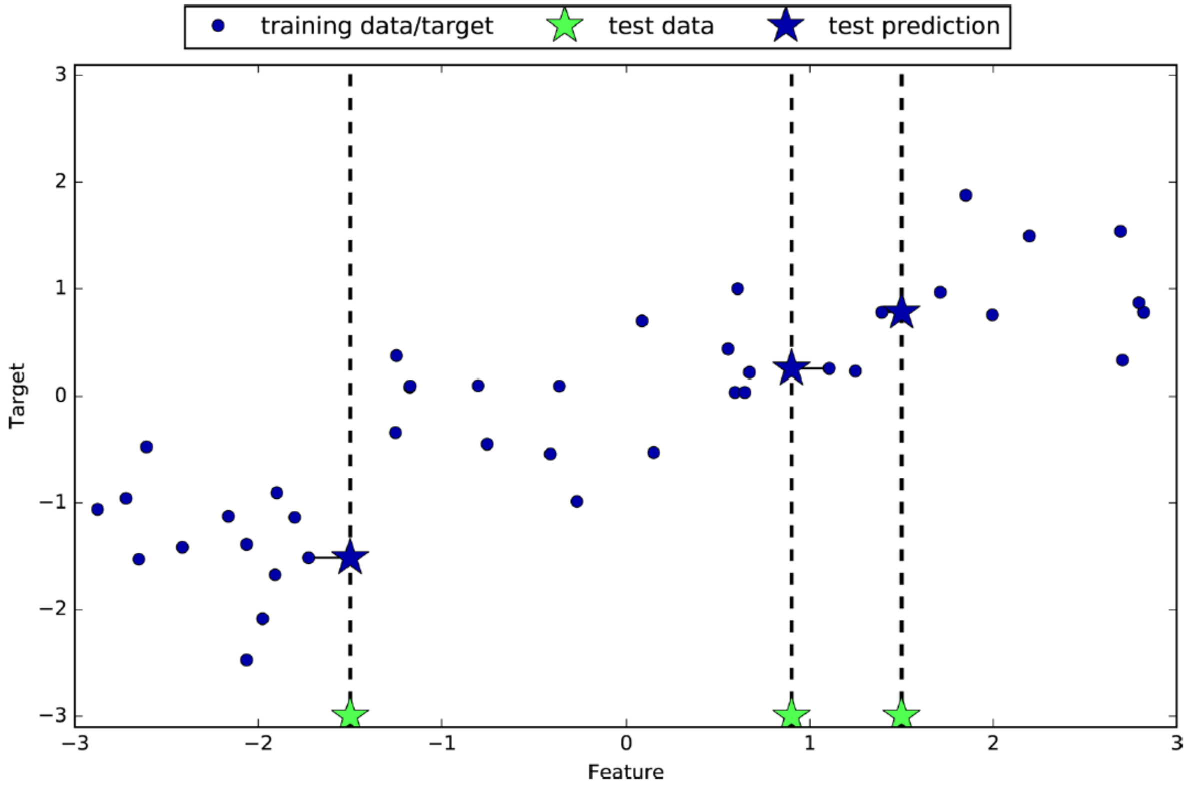


three-nearest-neighbor model

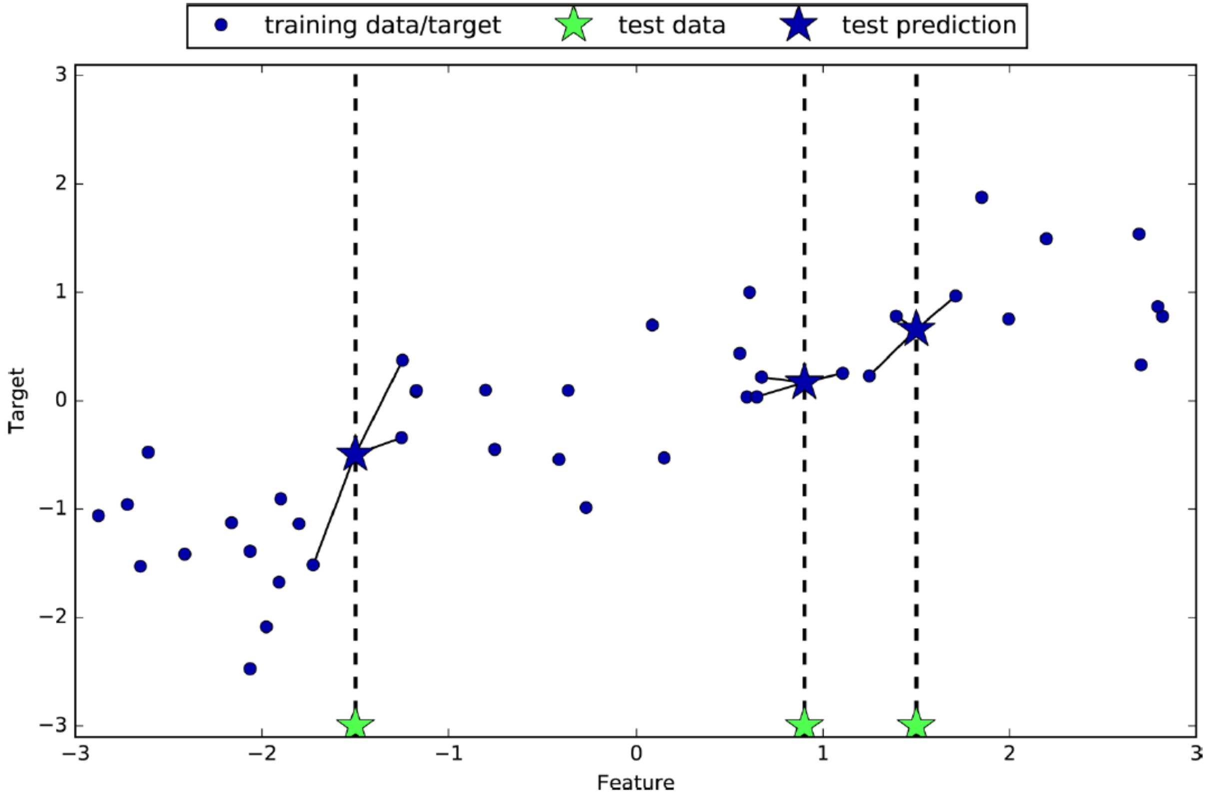


Regression

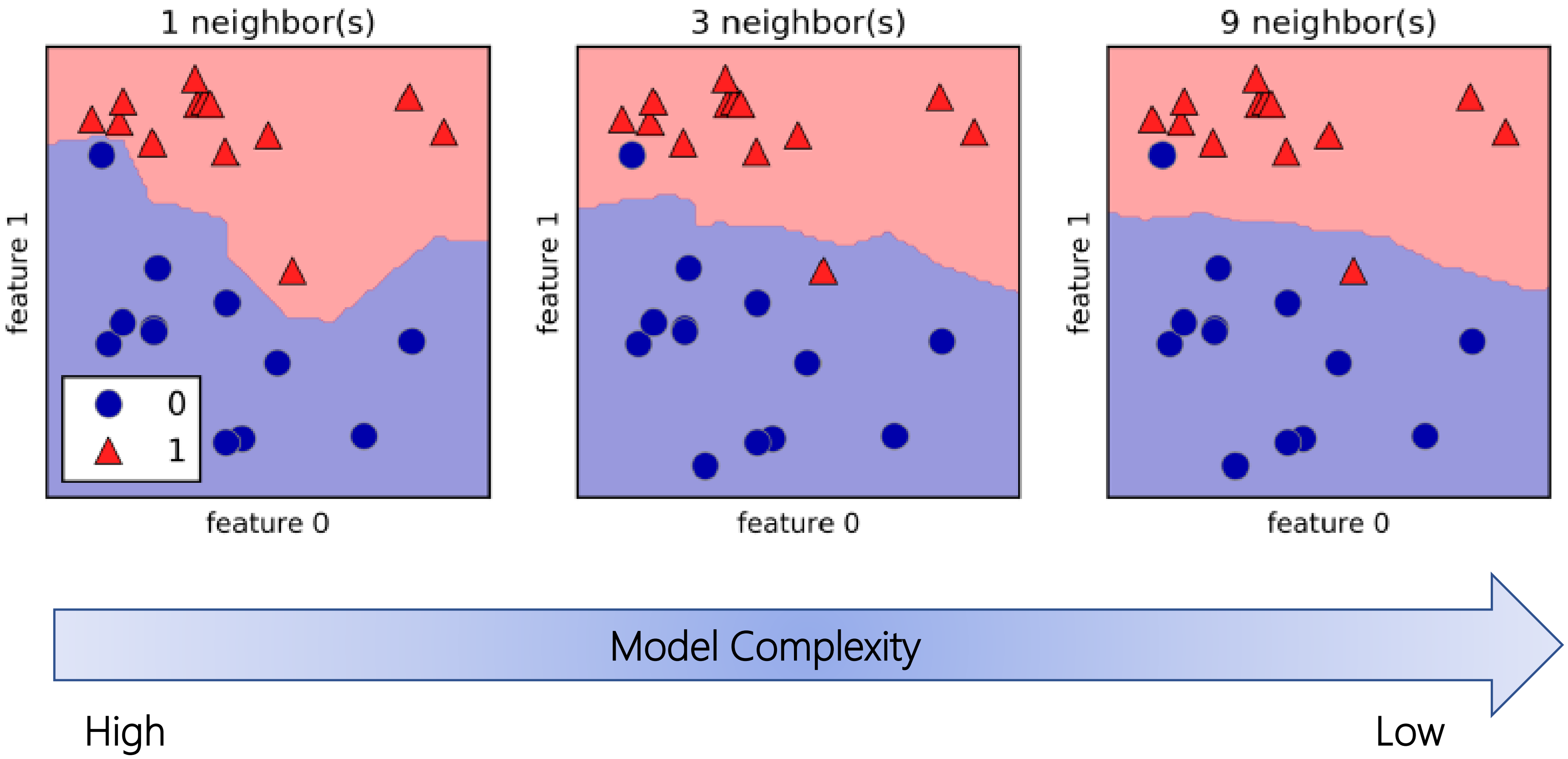
one-nearest-neighbor model



three-nearest-neighbor model

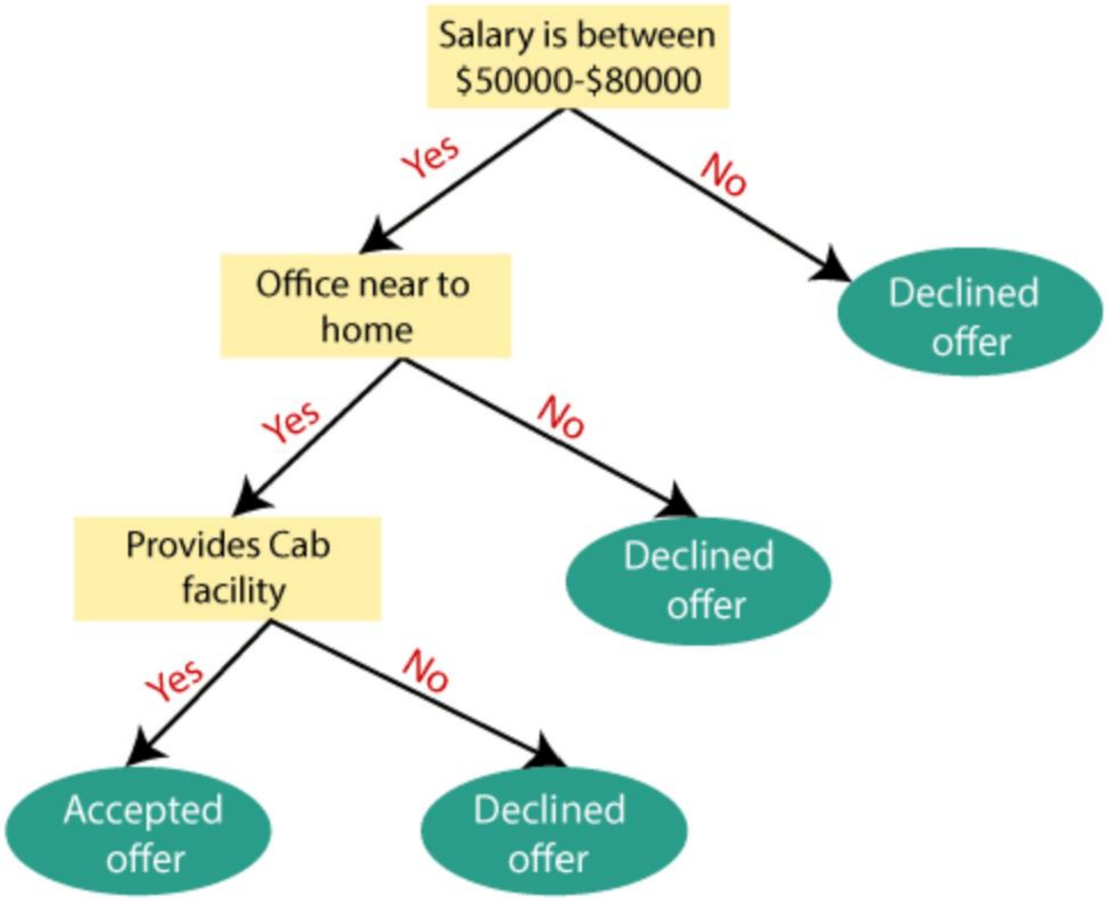


Decision boundaries created by the nearest neighbors model for different values of n_neighbors



3. Decision Tree Model

Decision Tree Model



- Intuitive and easy to interpret
- Require very little data preparation
- Don’t require feature scaling
- Easily deployed in rule-based system
- Build-in variable selection

Image source: https://www.mygreatlearning.com/blog/decision-tree-algorithm/

CART Algorithm

- CART: Classification and Regression Tree

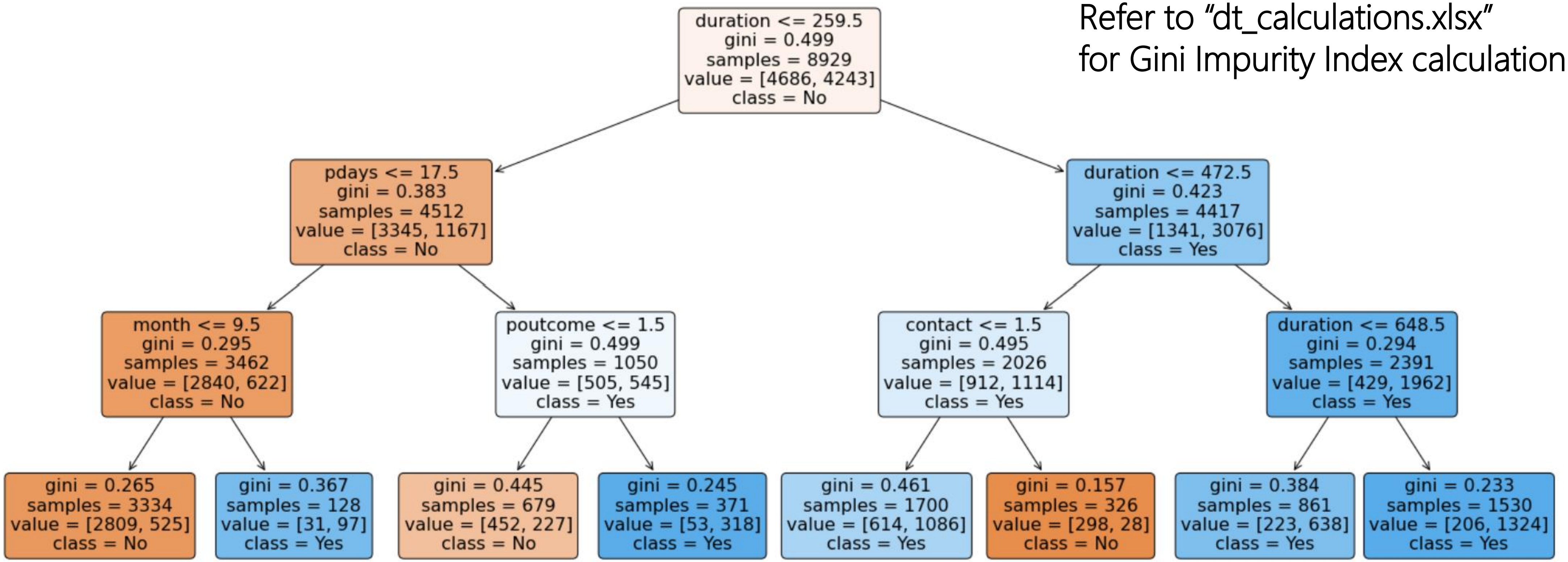
- Split the data into two subgroups to make the decision nodes as pure as possible

- How to measure the purity of a node?
  - Classification task:
    - e.g. Gini Impurity Index
    - the lower the Gini, the purer the node

  - Regression task:
    - e.g. Mean Squared Error
    - the lower the MSE, the purer the node

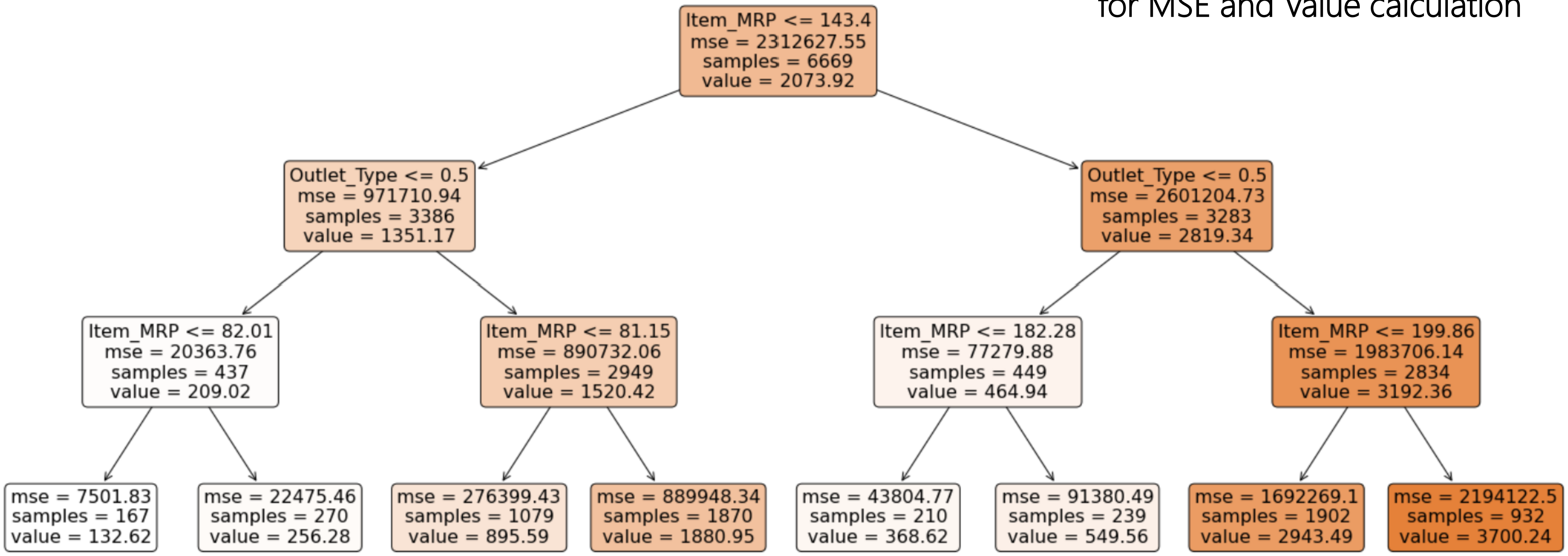
Classification

Refer to "dt_calculations.xlsx"
for Gini Impurity Index calculation



Regression

Refer to "dt_calculations.xlsx" for MSE and Value calculation



The Steps for CART

1. For every input feature
   - identify all possible binary split points
   - choose the best split point with the highest reduction in impurity/error

2. Rank the best splits and choose the feature that has the highest reduction in impurity/error

3. Divide the data into subgroups defined by the split

4. Continue the splitting process until:
   - All the nodes are 100% pure/error free or
   - Stopping condition is met
     1. Max tree depth
     2. Min samples at leaf node
     3. Min samples a node must have before it can split
     4. Others

4. Support Vector Machine

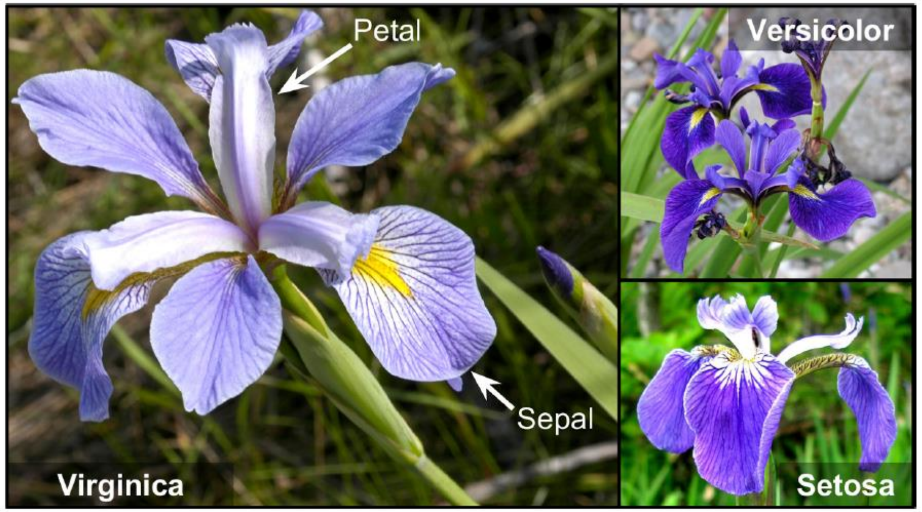
SVM Model

- One of the most popular machine learning model, powerful and versatile

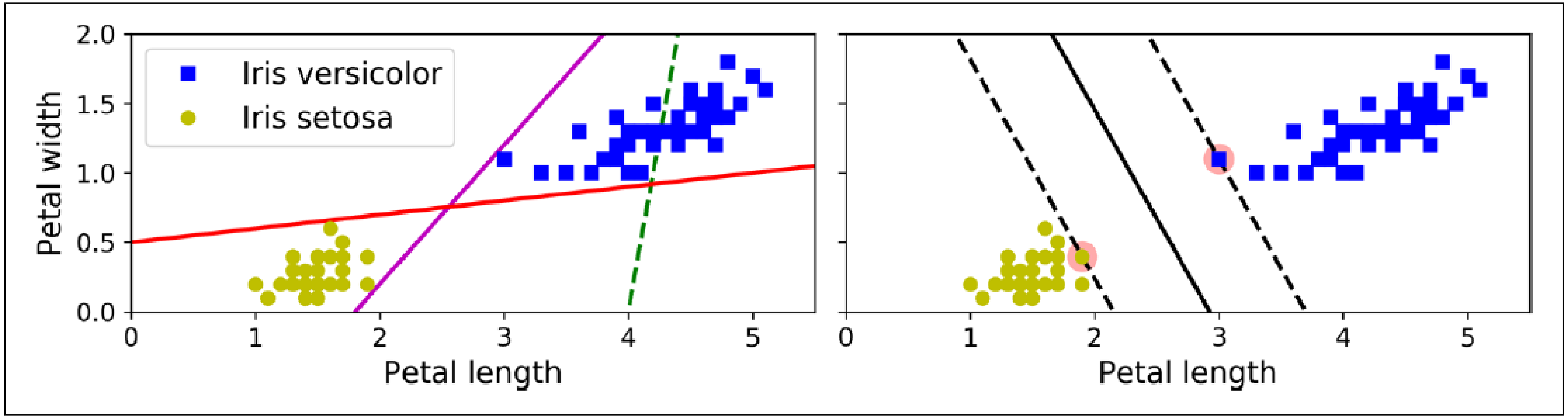
- Capable of performing linear/non-linear, classification/regression tasks but particularly suited for classification tasks

  - SVM Classification:
    - Linear SVM Model
    - Kernel Trick
    - Nonlinear SVM Model

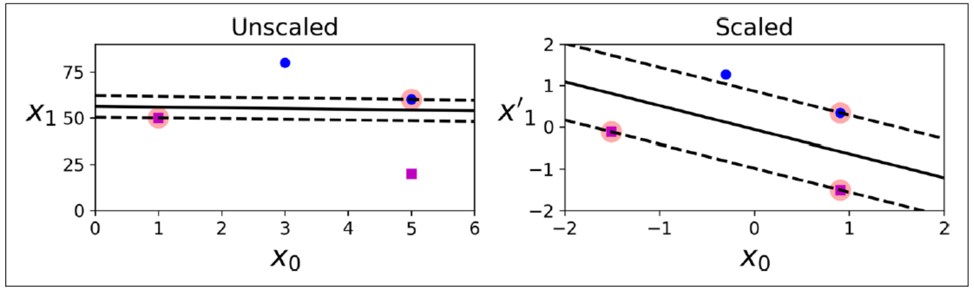
  - SVM Regression



Linear SVM Classification

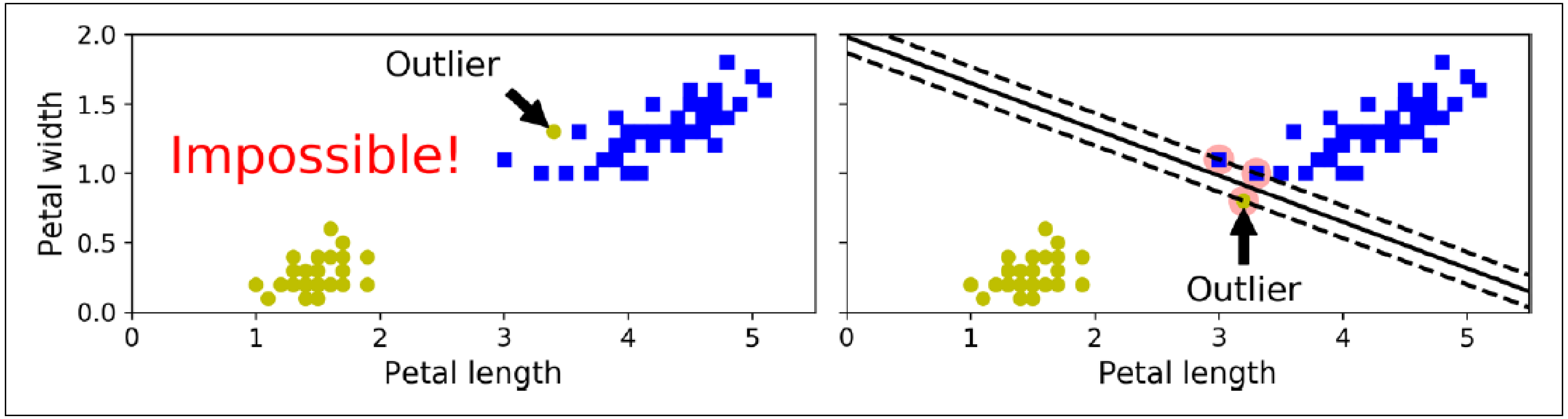


- Fitting the widest possible street between the two classes (large margin classification)
- All instances must be off the street and on the right side (i.e. No Margin Violations)??



Sensitive to Feature Scales

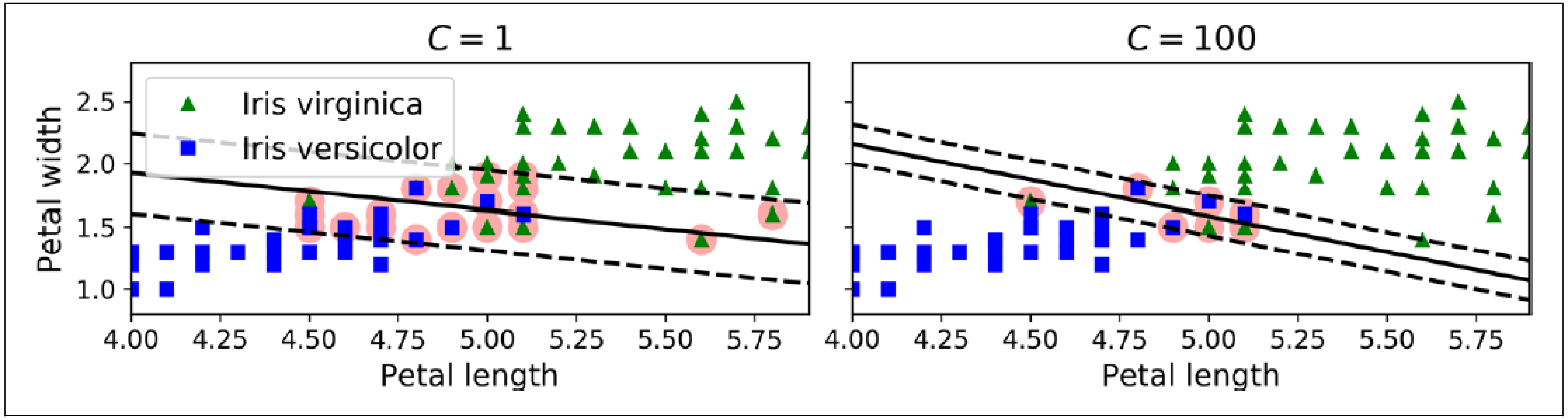
Use C to control the margin violations



Hard Margin Classification

- no margin violations
- sensitive to outliers

Soft Margin Classification



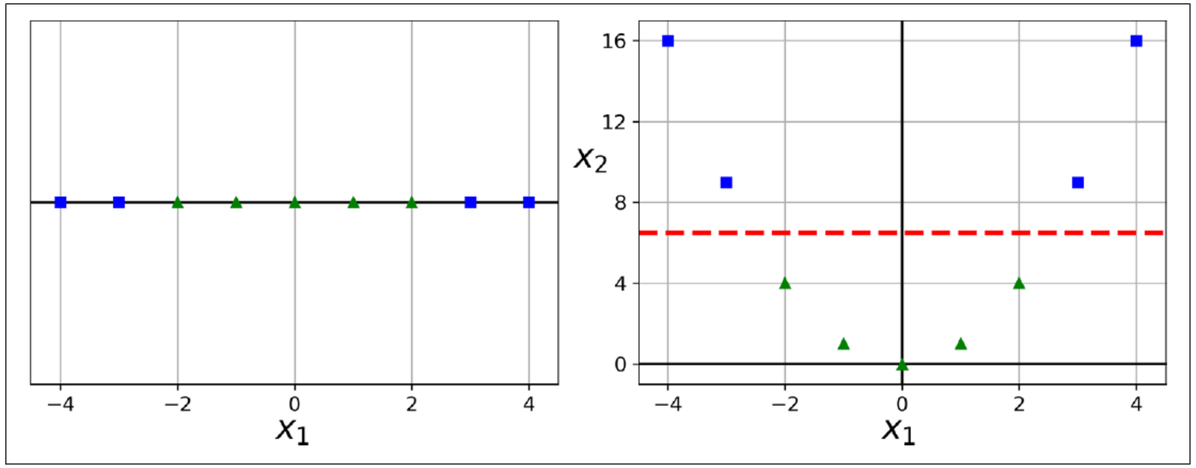
Large Margin Violations (wide street)

Fewer Margin Violations (narrow street)

Kernel Tricks

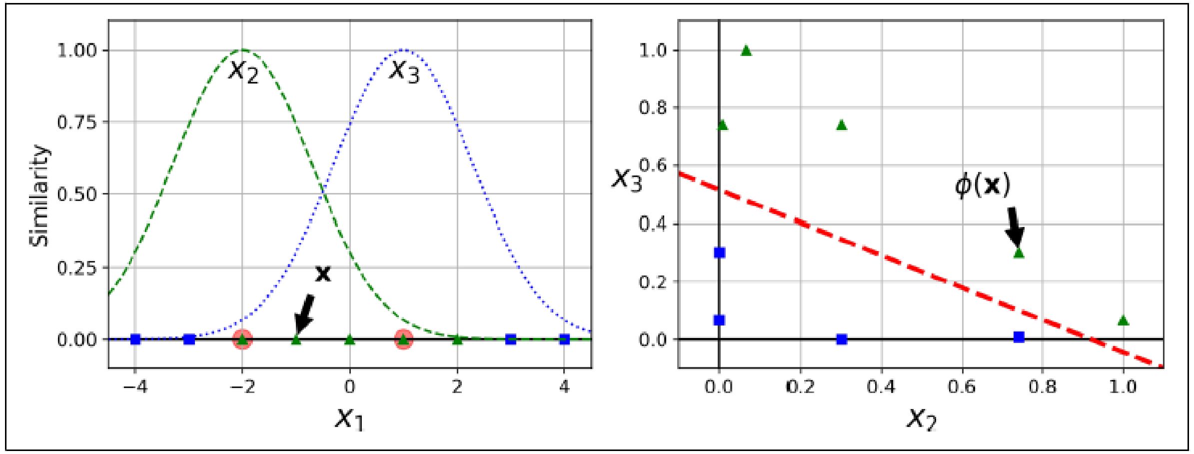
Feature Transformation using Polynomial Function

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$



Feature Transformation using Gaussian Radial Basis Function (RBF)

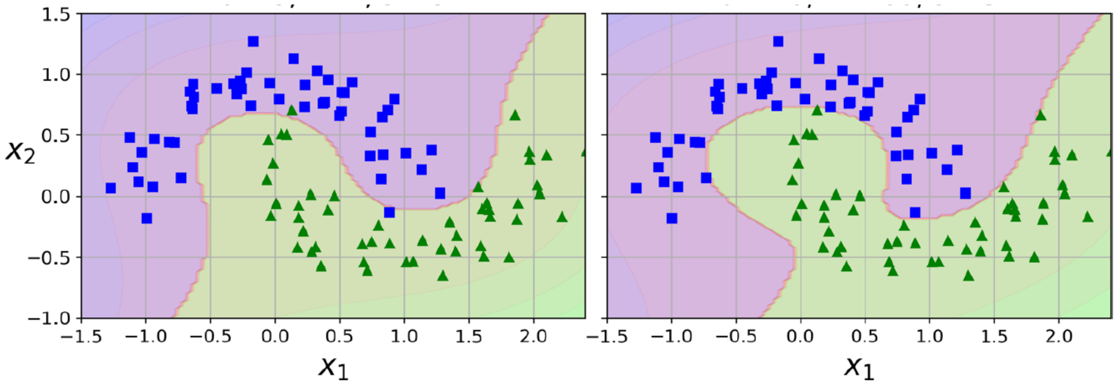
$$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \| \mathbf{x} - \ell \|^2\right)$$



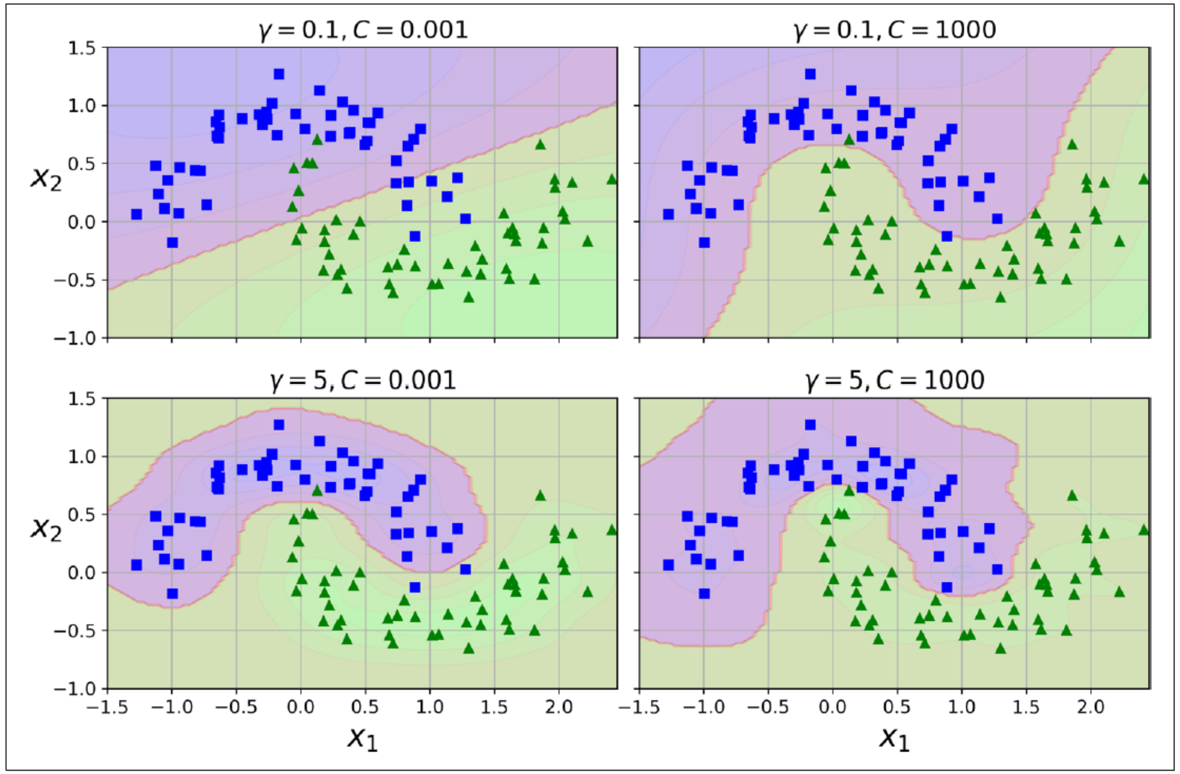
Nonlinear SVM Classification

Polynomial Kernel

d = 3 d = 10



RBF Kernel



Tuning SVM Models

Choose from different kernels

Always Try Linear Kernel First
- esp. Large Dataset
- Simple and Fast

Also Try RBF Kernel
- Small-Medium Dataset
- Works well in most cases

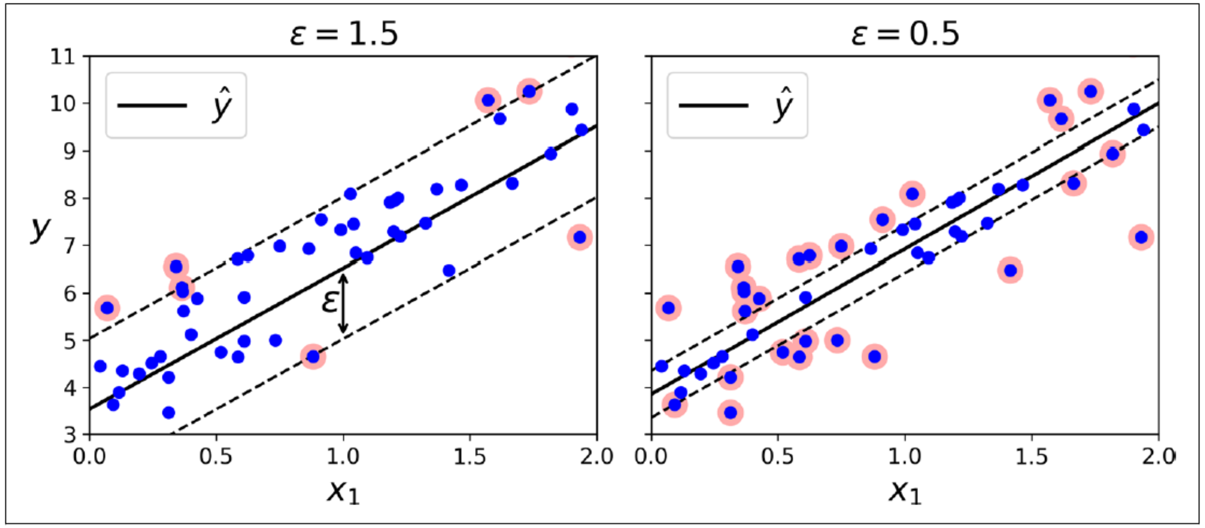
Explore Other Kernels
- Spare time
- Spare computation power

Tuning Hyperparameters (Grid Search)

- First do a very coarse search and then a finer search
- Having a good sense of what each hyperparameter does helps on searching in the right direction

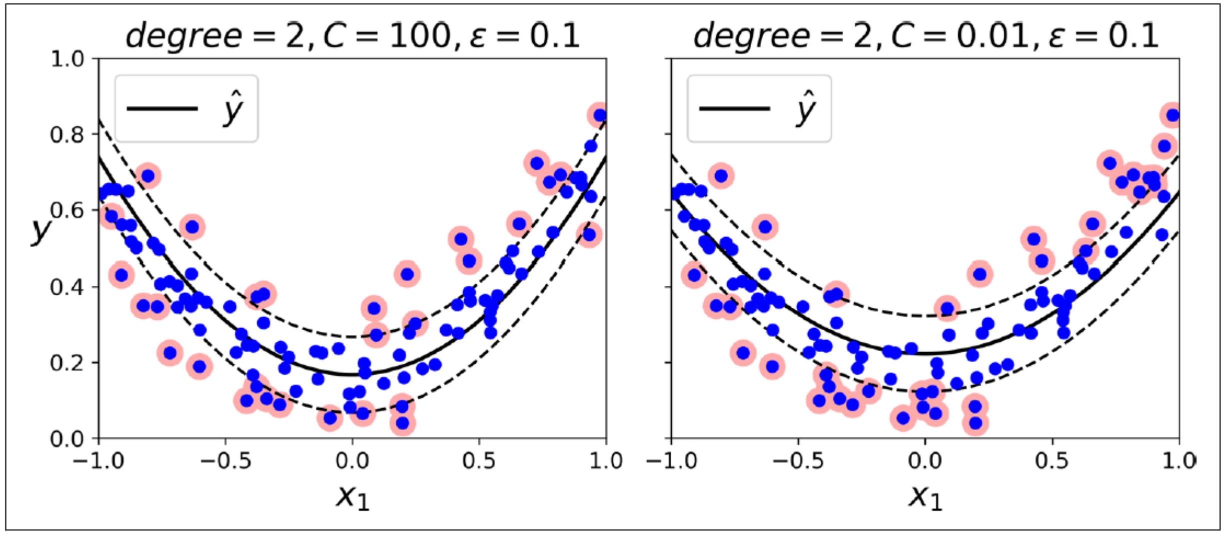
Underfitting (Low d, C, γ) → Overfitting (High d, C, γ)

SVM Regression



Fit as many instances as possible on the street while limiting margin violations (i.e. instances off the street)

The width of a street is controlled by hyperparameter ϵ



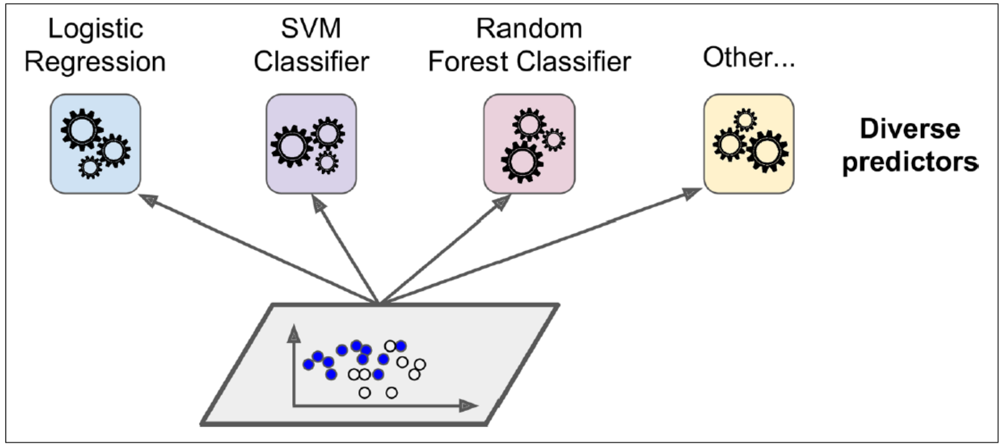
5. Ensemble Learning and Random Forest

What is Ensemble?

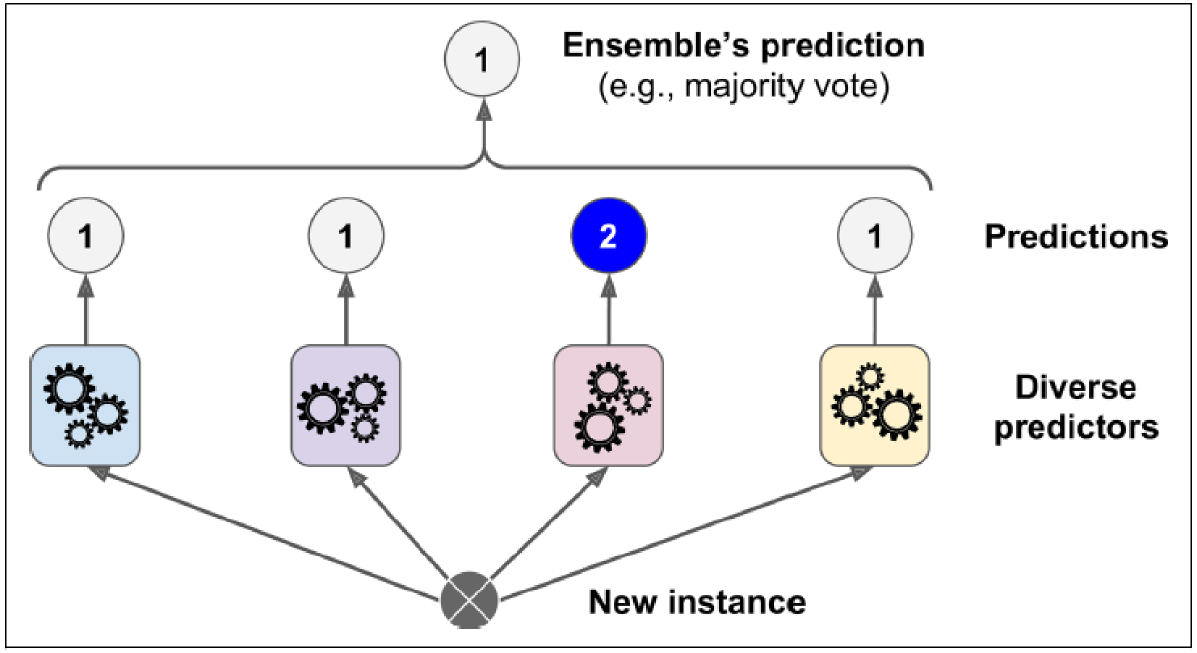
- Wisdom of Crowd
  - Aggregate the predictions of a group of predictors, you will get better predictions than with the best individual predictor
- Voting Classifier
- Bagging (Bootstrap Aggregating) Classifier & Regressor
- Random Forest Classifier & Regressor
  - an ensemble of Decision Trees via bagging method
- Boosting Classifier & Regressor

Voting Classifier

Training Diverse Classifiers



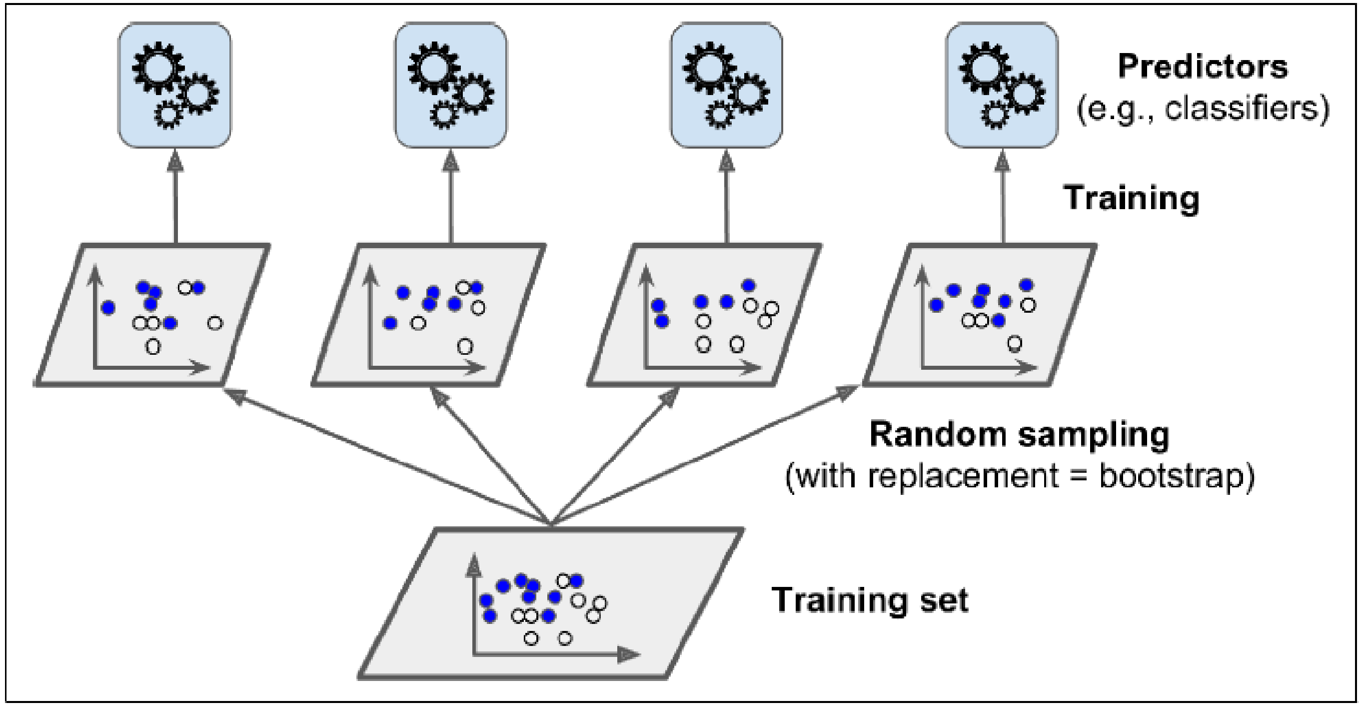
Hard Voting Classifier Predictions



<u>Soft Voting Classifier</u>
predict the class with the highest class probability, averaged over all the individual classifiers

Bagging: Bootstrap Aggregating

Training several predictors on different random samples of the training set

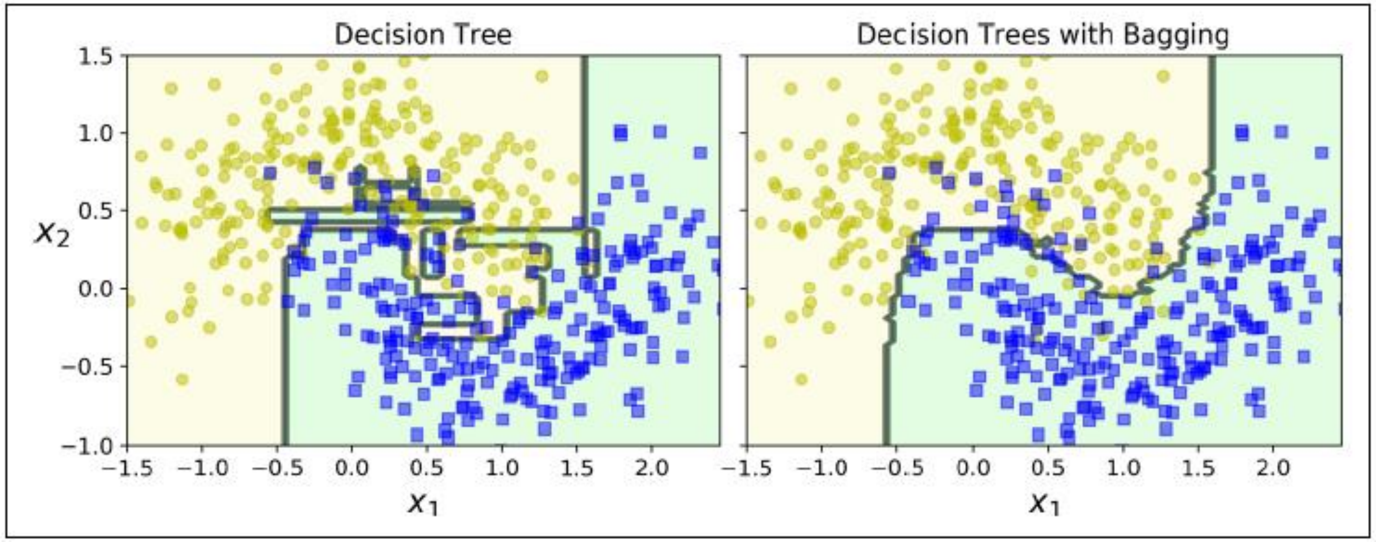


Bootstrap Sampling

Sample ID	Bootstrap Sample 1	Bootstrap Sample 2	Bootstrap Sample 3	Bootstrap Sample 4
1	8	1	7	3
2	7	6	7	5
3	4	4	5	7
4	7	2	8	9
5	4	3	8	1
6	2	3	1	2
7	6	6	3	3
8	10	3	9	5
9	10	9	1	9
10	9	10	1	1

Random Forest

- An ensemble of Decision Trees, generally trained via the bagging method
- Typically with max_samples set to the size of the training set
- Introduces extra randomness while growing the trees
  - searches for the best feature among a random subset of features
- Result in great tree diversity
- A more general model



Feature Importance of Random Forest

Bank Marketing Campaign

feature	importance
duration	0.395053
pdays	0.087093
balance	0.074528
month	0.074498
age	0.065932
poutcome	0.057947
day	0.056703
housing	0.041895
contact	0.032497
previous	0.027731
job	0.027496
campaign	0.019101
education	0.018552
marital	0.011539
loan	0.009000
default	0.000435
deposit	NaN

Supermarket Sales Forecast

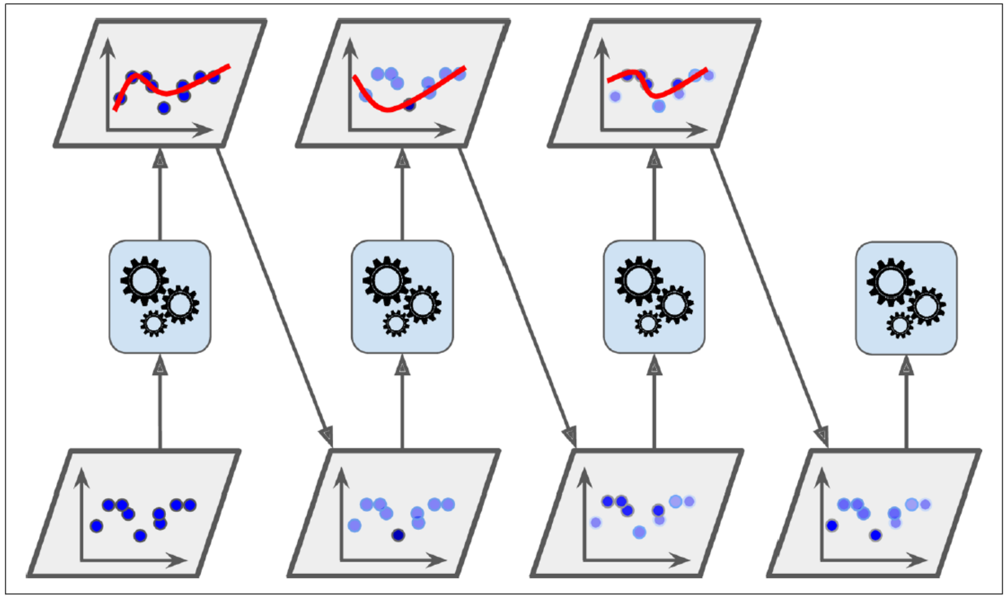
feature	importance
Item_MRP	0.566254
Outlet_Type	0.378453
Outlet_Establishment_Year	0.038922
Item_Visibility	0.008752
Item_Type	0.003000
Item_Weight	0.002129
Item_Fat_Content	0.001075
Outlet_Identifier	0.000807
Outlet_Size	0.000477
Outlet_Location_Type	0.000132
Item_Outlet_Sales	NaN

Boosting

- Combine several weak learners into a strong learner
- General idea is to train predictors sequentially, each trying to correct its predecessor
- AdaBoost and Gradient Boosting
    - XGBoost (Extreme Gradient Boosting): Extremely fast, scalable and portable
- If overfitting to the training set
    - Reduce the number of estimators/predictors
    - more strongly regularizing the base estimator

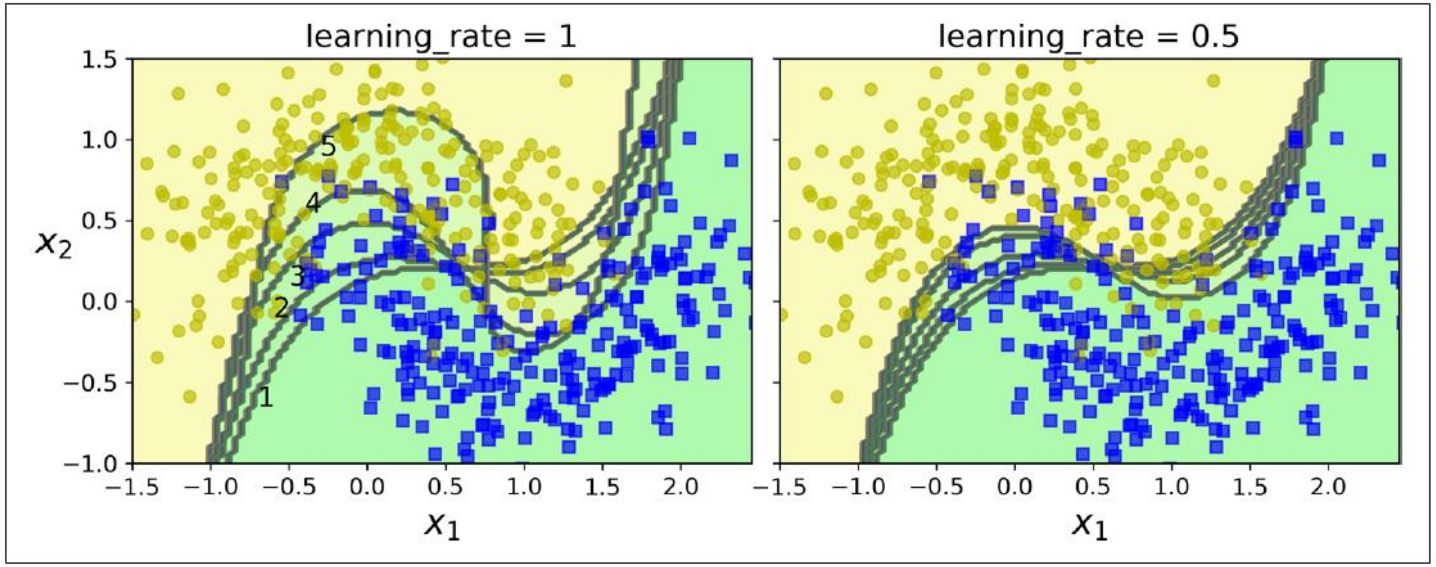
AdaBoost

- Train a base classifier
- Use it to make predication on training set
- Increases the relative weight of misclassified training instances
- Then train a 2nd classifier, using the updated weights, and again makes predictions, updates the training instance weights, and so on



- After all predictors are trained, the ensemble makes predictions like bagging, except that predictors have different weights depending on their overall accuracy on the weighted training set.

In AdaBoost, we use learning_rate to control how fast/slow the misclassified instance weights are boosted….



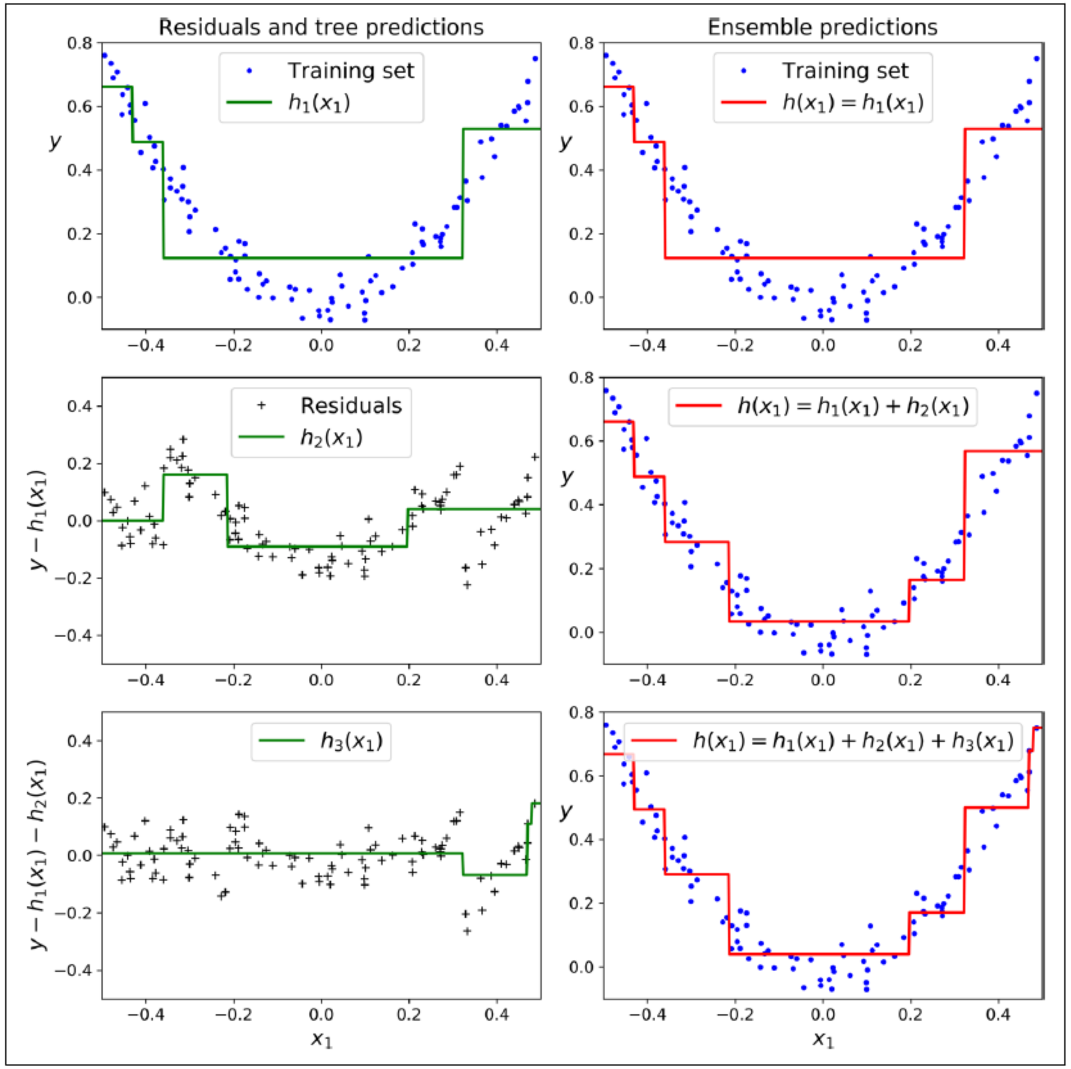
Decision boundaries of consecutive predictors in AdaBoost, with different learning_rate

Gradient Boosting

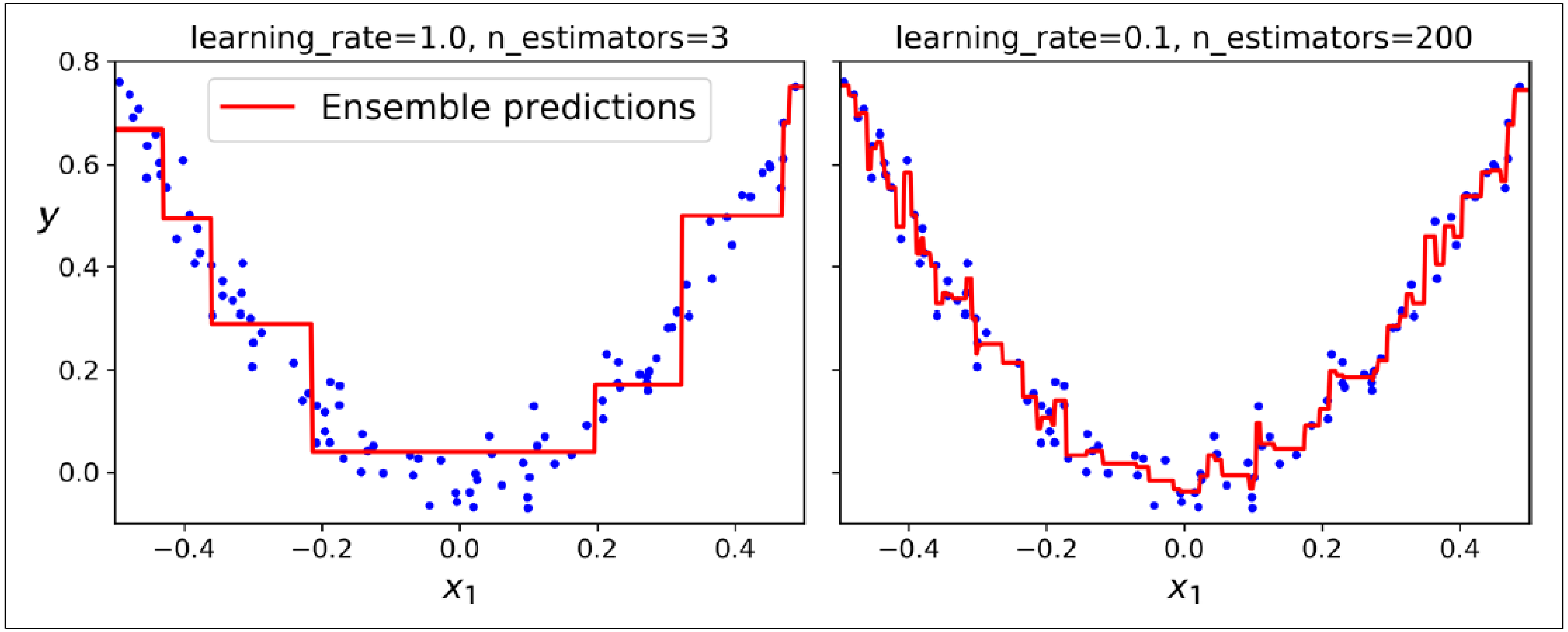
1st predictor is trained normally

2nd predictor is trained on 1st predictor's residuals

3rd predictor is trained on 2nd predictor's residuals



In Gradient Boosting, the learning_rate hyperparameter scales the contribution of each tree. If you set it to a low value, you will need more trees in the ensemble to fit the training set, but the predictions will usually generalize better.



Not Enough Trees

Too Many Trees

Summary

1. Linear Regression and Logistic Regression
2. K-Nearest Neighbors
3. Decision Tree Model
4. Support Vector Machine
5. Ensemble Model (Random Forest)

Day 3: Online Learning

Session 1: End-to-End ML Project

1. Look at the big picture
2. Get data and Explore the data
3. Prepare the Data for ML models
4. Select, Train and Fine-Tune the Models
5. Launch, Monitor and Maintain Your System

Session 2: Classification

1. Training Classifiers
   - Binary Classifier
   - Multiclass Classifier
2. Performance Measures
   - Confusion Matrix
   - Precision and Recall
   - ROC Curve
3. Error Analysis

THE END

References

Aurélien Géron (2019). *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd edition.

Andreas C. Müller & Sarah Guido (2019). *Introduction to Machine Learning with Python*.

Machine Learning (Coursera Course) https://www.coursera.org/learn/machine-learning

Kaggle https://www.kaggle.com/

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY