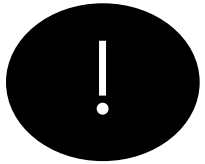
LEGAL NOTICE



1. All functionality, descriptions, ideas, drawings, features, specifications and other information provided in this document are (i) Samsung's proprietary and confidential information, (ii) subject to the non-disclosure agreement regarding "Project Samsung Innovation Campus (e.g SIC)", or "Samsung Innovation Campus (e.g SIC)" or (iii) provided for discussion purpose only, and (iv) shall not be disclosed by the recipient to any third party.
2. Samsung reserves the right to make changes to this document, at any time, without obligation on Samsung to provide notification of such change.
3. Nothing in this document grants to the recipient any license or right in or to information or materials provided in this document, or any intellectual property therein. SAMSUNG PROPRIETARY AND CONFIDENTIAL
4. THIS DOCUMENT AND ANY INFORMATION CONTAINED HEREIN ARE PROVIDED ON "AS IS"

SAMSUNG

Samsung Innovation Campus

Artificial Intelligence Course

Together for Tomorrow!
Enabling People
Education for Future Generations

Module 5 – Supervised Learning

Artificial Intelligence Course

Chapter Description

Chapter objectives

- ✓ Be able to introduce machine learning-based data analysis according to the business objective, strategy, and policy and manage the overall process.
- ✓ Be able to select and apply a machine learning algorithm that is the most suitable to the given problem and perform hyperparameter tuning.
- ✓ Be able to design, maintain, and optimize a machine learning workflow for AI modeling by using structured and unstructured data.

Chapter contents

- ✓ Unit 5a. Application of Supervised Learning Model for Classification
- ✓ Unit 5b. Decision Tree
- ✓ Unit 5c. Naïve Bayes Algorithm
- ✓ Unit 5d. KNN Algorithm
- ✓ Unit 5e. SVM Algorithm
- ✓ Unit 5f. Ensemble Algorithms

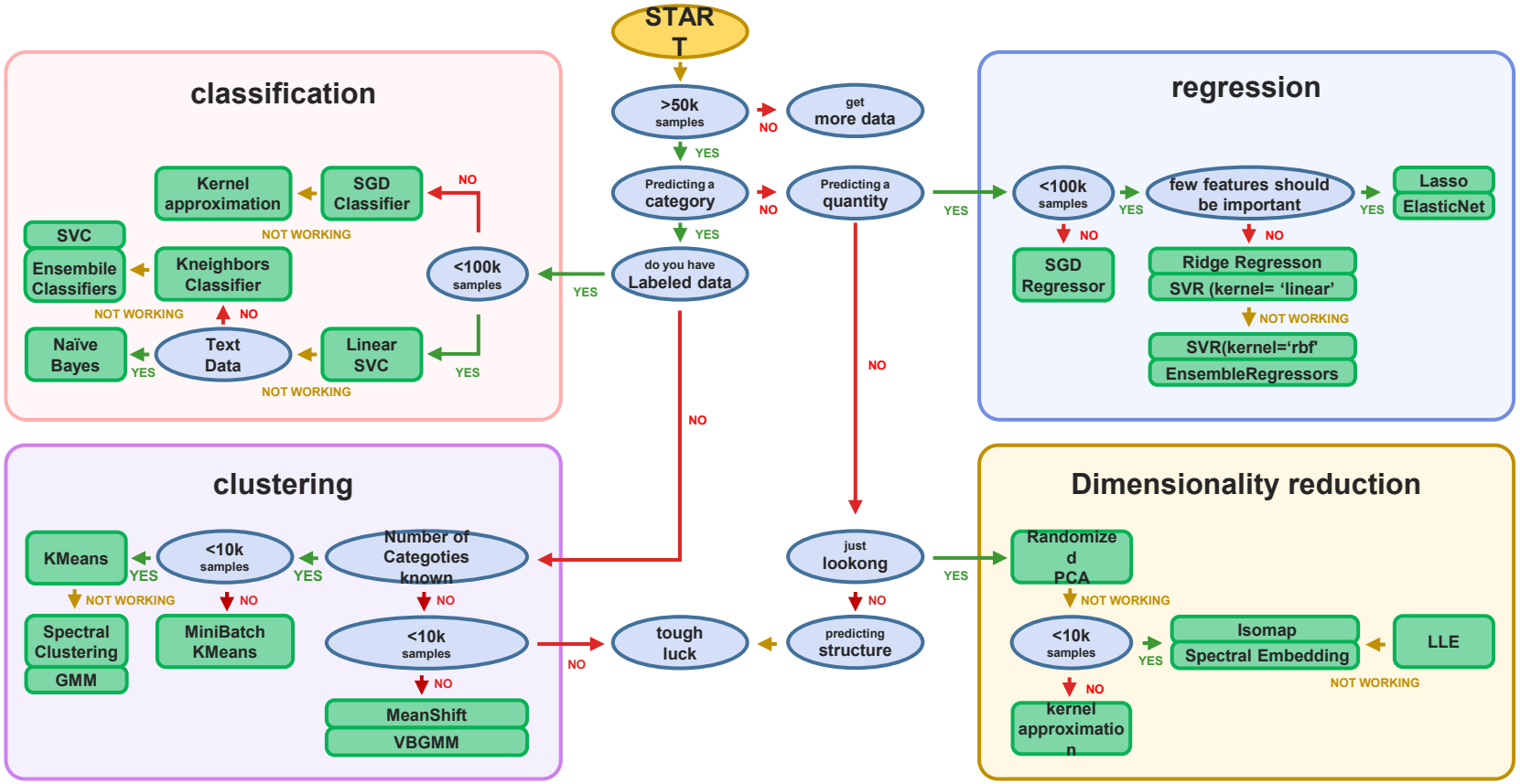
Unit 5a.

Application of Supervised Learning Model for Classification

- 5a.1. Training and Testing in Machine Learning
- 5a.2. Logistic Regression Basics
- 5a.3. Logistic Regression Performance Metrics

Training and Testing in Machine Learning

Select a machine learning algorithm suitable for the data

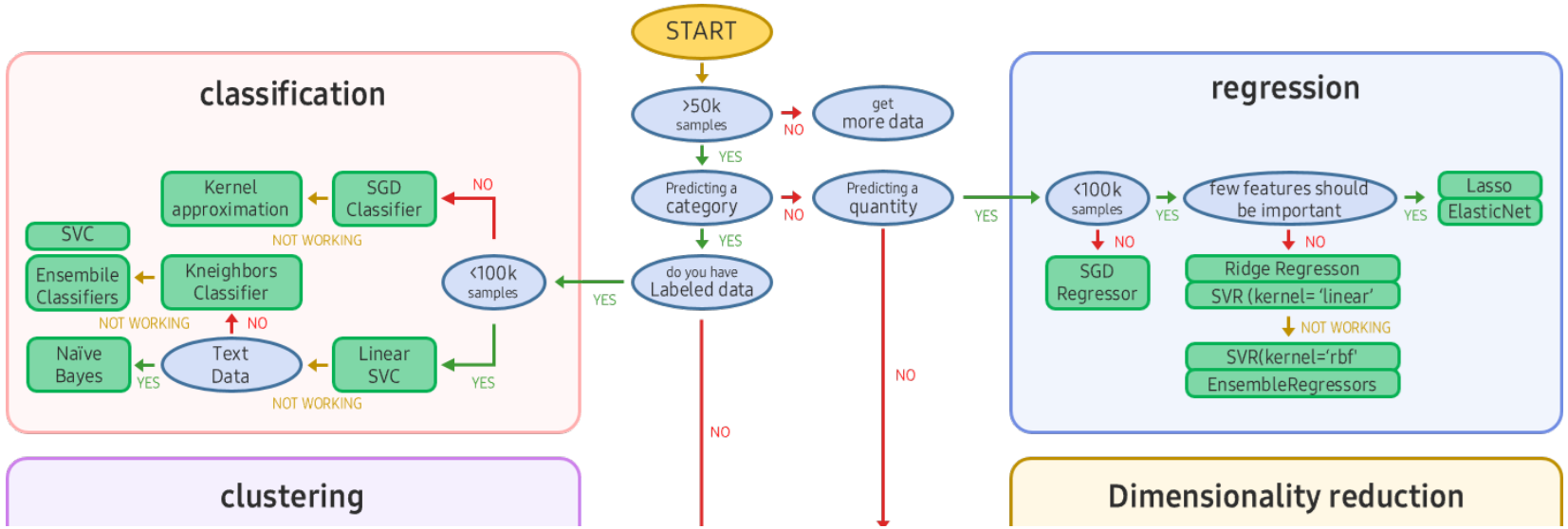


5a.1. Training and Testing in Machine Learning

Select a machine learning algorithm suitable for the data

- Supervised learning is machine learning tasks that try to predict or classify the values of objective variables (or response variables) in unseen data as the training data set has the objective variables (or response variables) to be predicted (Y).
- It is called supervised learning because the algorithm is told to predict a certain target.
- Supervised learning is mainly classified into classification and regression problems depending on the types of objective variables (or response variables).
- In other words, if the objective variables (or response variables) are discrete or nominal variables such as ‘male/female’, ‘spam mail/ham mail’, ‘positive/neutral/negative’, ‘Seoul/Busan/Gyeonggi/Gangwon,’ classification method is applied. On the other hand, if the objective variables (or response variables) are numerical type such as 0~10, -500~500, -3.5~3.5, -∞~∞, regression is applied.
- When the given data does not have classification marks or objective variables (or response variables) and if it’s not trying to predict target value, unsupervised learning is applied.
- Unsupervised learning is classified into clustering, association, dimension reduction, and others depending on the analysis purpose and methods, and further details will be explained in other chapters.

Select a machine learning algorithm suitable for the data



- As shown in the figure above, if there are correct answers and it is for classification purposes, or has a lot of data, neural network can be applied. If not, it is possible to use decision tree or SVC algorithm.
- Also, if the data to be classified is in text, Naïve Bayes algorithm can be used.
- If it's not text data, KNN method is used, and it is also possible to use ensemble method with a better performance.

Machine learning for classification

- Classification is used if the objective variables (or response variables) can be classified into a certain type of categories such as discrete or nominal type. It is the most commonly and frequently found in the machine learning-based data analysis.
- Machine learning algorithms for classification can be applied to extensive daily and business problems.

  Ex Frequently used examples

  (1) Classifying spam mails

  (2) Prediction of corporate bankruptcy

  (3) Prediction of customer loss

  (4) Classifying customers' credit rating

  (5) Prediction of occurrence of a certain disease (e.g., cancer, heart disease, etc.)

  (6) Prediction of customer reaction to a specific marketing event

  (7) Prediction of customer's purchase

- The examples provided above are extremely few of the real-life applications, and there are infinite fields of the business that classification type machine learning is applied. This type of machine learning has been continuously applied with new R&D and business methods.

Types of machine learning algorithm for classification

- There are many different kinds of classification type machine learning algorithms, and some of them can be used for regression problems. The following table provides descriptions of frequently used classification methods.

Type	Concept	Note
K-Nearest Neighbor	A classification method that uses the majority rule on the closest k objective variable values (or response variables) based on the distance between data coordination other than a certain data.	Lazy Learning
Naïve Bayes	A method based on the Bayes' theorem that makes classification towards the higher probability by expressing the conditional probability of objective variables (or response variables) as multiplication of the prior probability and likelihood function. All of the observed values are assumed to statistically independently occur from other observed values. (Referred to as a Naïve model since the assumption is given without confidence.)	Probability model (Bayes' theorem based conditional probability)
Logistic Regression	A method to estimate the probability of objective variables through the maximum likelihood estimation by assuming that the probability of the objective variable value being in a certain category is in the logistic function shape when the explanatory value is given.	Probability model (maximum likelihood estimation)
Decision Tree	A method to create classification rules by splitting the branches towards lower impurity or entropy in the order of variables that are most associated with objective variables.	Divide & Conquer

Types of machine learning algorithm for classification

- There are many different kinds of classification type machine learning algorithms, and some of them can be used for regression problems. The following table provides descriptions of frequently used classification methods.

Type	Concept	Note
Artificial Neural Network	A method inspired by the human neuron network. Comprised of the input node, hidden node, and output node, this analysis method is used to solve complicated classification or black box value prediction problems.	Black box test
Support Vector Machine	A method to classify certain data by finding a plane that maximizes the margin between data in different categories.	Linear and non-linear (Kernel trick)
Random Forest	An ensemble method to decide the final classification result by making various decision trees based on given data and aggregating the predicted results of each decision tree through voting.	Ensemble model

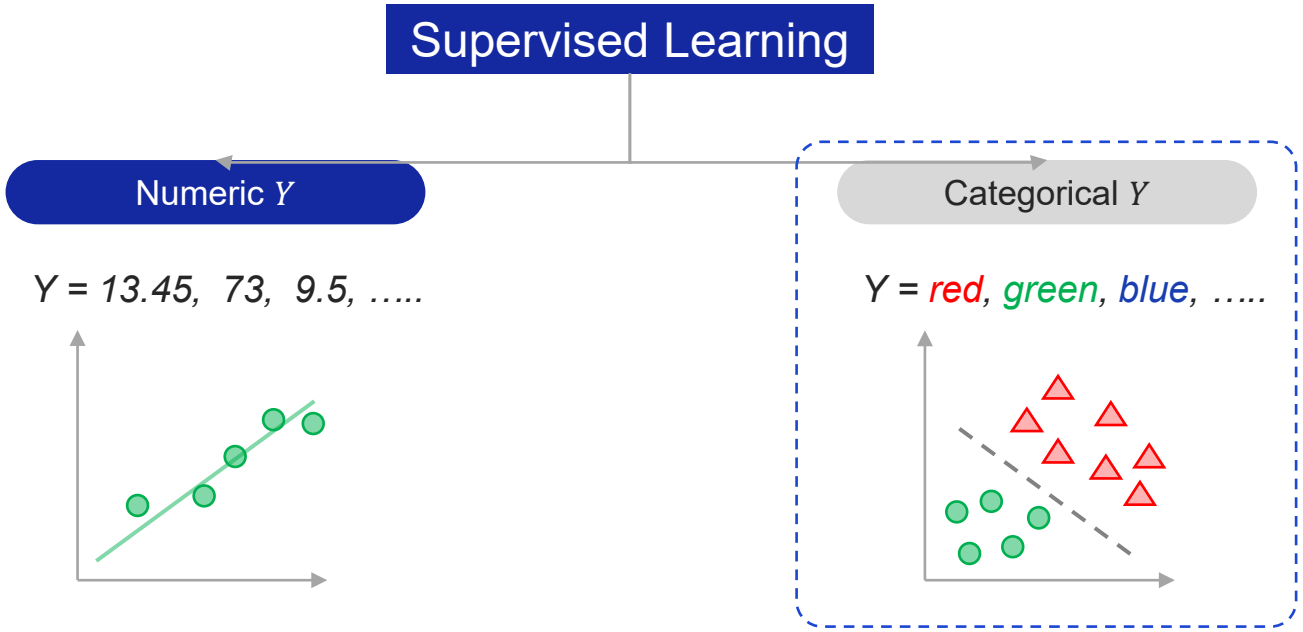
Unit 5a.

Application of Supervised Learning Model for Classification

- 5a.1. Training and Testing in Machine Learning
- 5a.2. Logistic Regression Basics
- 5a.3. Logistic Regression Performance Metrics

Logistic Regression Basics

- What is logistic regression analysis?
  - In contrast to general linear regression analysis is used for numerical prediction, logistic regression analysis is used to classify the category of objective variables (y) to be predicted. In other words, what is being predicted is not y value which is an objective variable, but it is P(Y=i), which is the probability of objective variable y to become a certain category (i).



- About logistic regression

vc

- There is one or more explanatory variables: $X_1, X_2, \ldots, X_k$
- There is one response variable: Y
- The response variable Y is binary where possible values are $\{0,1\}$, $\{False, True\}$, $\{No, Yes\}$, etc.
- One of the most basic classification algorithms.

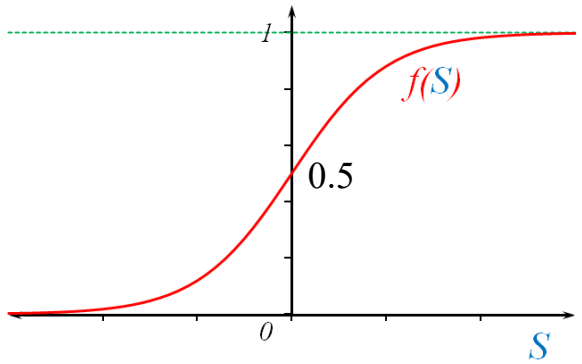
- Pros
  - Simple and relatively easy to implement.
  - Source of intuitive insights.
  - Fast training.
- Cons
  - Not among the most accurate classification algorithms.
  - Assumes that the explanatory variables are independent without multicollinearity.

- About logistic regression
  - The linear combinations of variables X_i is the so-called “Logit” denoted here as S

$$S = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

  - The conditional probability of Y being equal to 1 is denoted as $P(Y = 1|\{x_i\})$.
  - “Sigmoid” or “Logistic” function connects the probability with the logit

$$f(S) = \frac{e^S}{1 + e^S}$$



- About logistic regression
  - If p = $probability\ of\ Y$=1, we can define

$$Odds = \frac{p}{1-p}$$

$$Logit\ = Log(Odds) = Log\left(\frac{p}{1-p}\right)$$

  - The logistic function is the inverse of the logit (and vice versa)

$$S = Log\left(\frac{p}{1-p}\right) \quad \Leftrightarrow \quad p = \frac{e^{S}}{1+e^{S}}$$

- About logistic regression
  - For logistic regression, the categories of objective variable (Y) to be predicted are 0 and 1 (binomial logistic regression model), and the possibility of the objective variable Y category becoming 1 is expressed as p(Y=1) = P(Y). Start from expressing the description as regression equation as follows, and the left side of the equation is commonly referred to as 'odds.'
  - The 'odds' signify dividend rate, which categorizes into success rate/failure rate. If the success rate is high, it becomes 1, and if it is low, it becomes 0. So, in other words it is success rate. Adding a log to odds becomes logit, and logistic is the function.

$$\frac{P(Y)}{1-P(Y)} = \exp(\beta_0 + \beta_1 X)$$

  - The left side of the equation is ratio of probability and the right side is exponential function which has (0, ∞) range. In order to give range with (−∞, ∞) values on both sides, add log on both sides of the equation.

$$\log(\frac{P(Y)}{1-P(Y)}) = \beta_0 + \beta_1 X$$

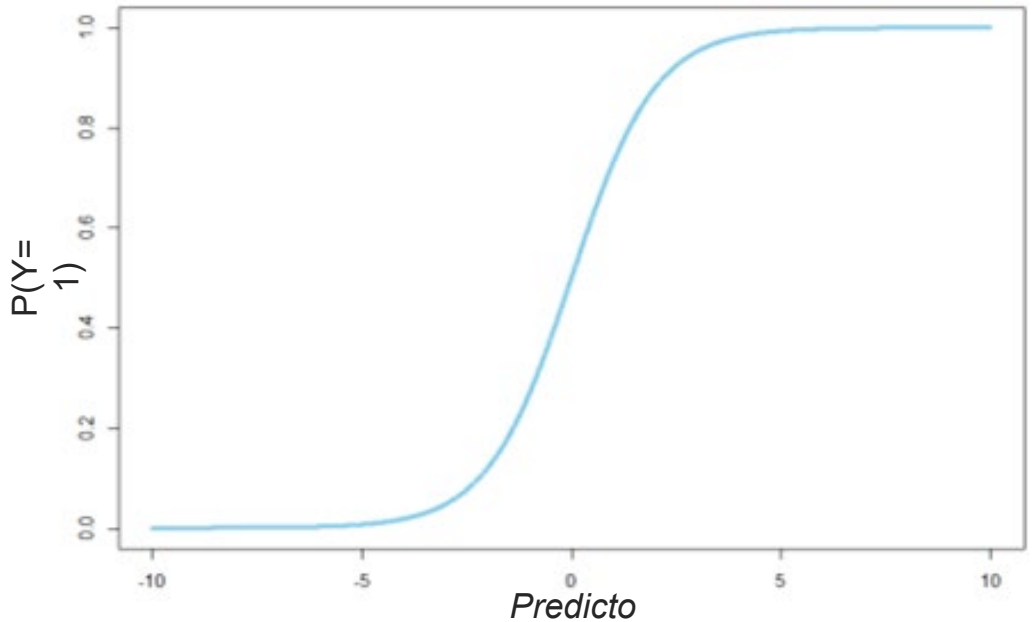
  - When looking at the equation in detail, the $\beta\_0+\beta\_1$ on the right side is a linear model which has a range (−∞, ∞), and the left side also has a range (−∞, ∞). The log(P(Y))/(1-P(Y)) on the left side of the equation is called logit function.
  - Or, add 'exp' to both of sides of the equation and arrange it regarding P(Y) to get the following equation.

- About logistic regression

$$P(Y=1) = \frac{e^{(\beta_0 + \beta_1 X)}}{1 + e^{(\beta_0 + \beta_1 X)}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

  - Summing up, logistic regression analysis is an algorithm that establishes a model by using the logistic function from the above equation and predict parameters β_0, β_1 from the probability P(Y=1) = P(Y) where objective variable Y having categorical value 1 from the train data. For predicting parameters β_0, β_1, maximum likelihood estimation is generally used. This is an analytical method that changes the formula and since direct calculation is difficult, estimation is done by giving a certain initial value and performing repeated calculation to adjust the value as numerical calculation.

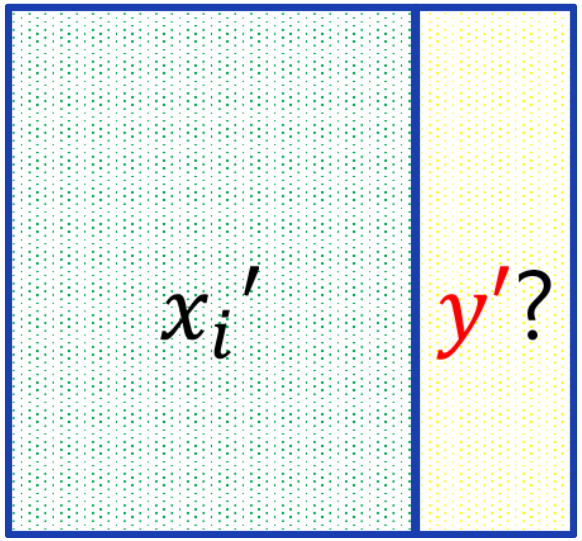
- About logistic regression
  - The following figure shows a graph of logistic function. As shown in the graph, the range of X-axis is $-\infty$ to ∞. Y-axis is the probability of objective variable Y occurrence, and it has a range between 0 and 1. So, the function P(Y=1) shows an S-shaped curve from 0 to 1 as the x value is increased.



Logistic regression graph

- Logistic regression training and prediction (testing)

2) Prediction step: when a new set of $\{x_i'\}$ is given, calculate the value of y' which was unknown.

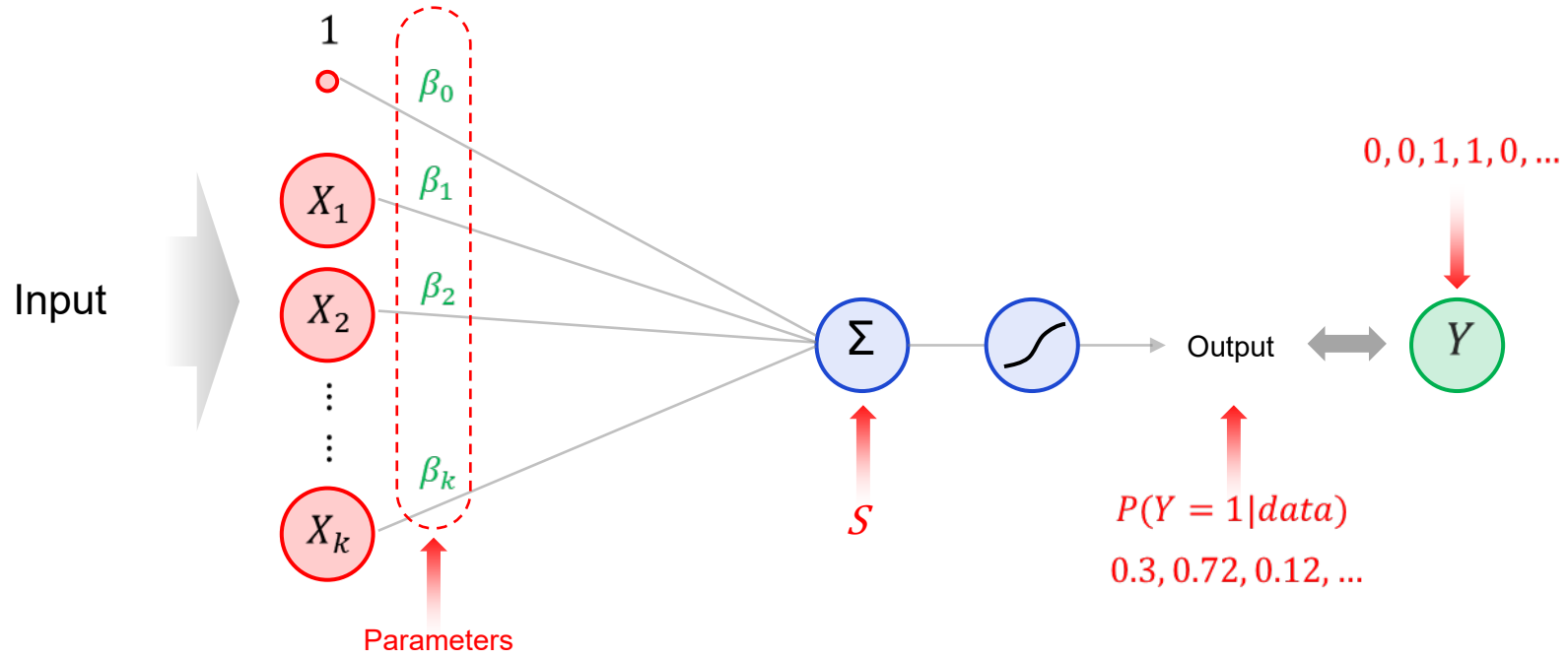


Compare the conditional probability with a cutoff.

$$P(Y' = 1|\{x_i'\}) > \text{Cutoff ?}$$

$$\Rightarrow \quad \hat{y} = 1 \quad \text{or} \quad 0$$

- Logistic regression training and prediction (testing)



- Training by gradient descent algorithm
  - We can get the parameter set $\{\beta_i\}$ by minimizing a target function $L(\boldsymbol{\beta})$.

    We get to minimize the difference between the real Y and the predicted $\hat{Y}$.
  - We can think of $L(\boldsymbol{\beta})$ as a "loss" or "cost".
  - $L(\boldsymbol{\beta})$ is a function of the parameter set $\{\beta_i\}$ which can be represented by a vector $\boldsymbol{\beta}$.

- Training by gradient descent algorithm
  - $L(\boldsymbol{\beta})$ is the **negative** of logarithmic likelihood defined as following

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} Log\left(1 + e^{-y_i \boldsymbol{\beta}^t \boldsymbol{x}_i}\right)$$

  - In the above relation, $\boldsymbol{x_i}$ and y_i represent values given by the training dataset.
  - Here, we assumed the conversion from $y_i = \{0,1\}$ to $y_i = \{-1,+1\}$.

- Training by gradient descent algorithm
  - $L(\boldsymbol{\beta})$ is the **negative** of logarithmic likelihood defined as following

$$L(\boldsymbol{\beta}) = \sum_{i=1}^{n} Log\left(1 + e^{-y_i \boldsymbol{\beta}^t \boldsymbol{x}_i}\right)$$

$$\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,k} \end{bmatrix}, \quad \boldsymbol{x}_i^t = [1, x_{i,1}, x_{i,2}, \cdots, x_{i,k}]$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \quad \boldsymbol{\beta}^t = [\beta_0, \beta_1, \cdots, \beta_k]$$

- Training by gradient descent algorithm
  - The gradient of $L(\boldsymbol{\beta})$ is denoted as $\nabla L(\boldsymbol{\beta})$ which is also a function of the parameter set $\{\beta_i\}$ or $\boldsymbol{\beta}$.
  - The expression for the gradient $\nabla L(\boldsymbol{\beta})$ is:

$$\nabla L(\boldsymbol{\beta}) = -\sum_{i=1}^{N} \frac{y_i \boldsymbol{x}_i e^{-\boldsymbol{\beta}^t \boldsymbol{x}_i}}{1 + e^{-\boldsymbol{\beta}^t \boldsymbol{x}_i}}$$

  - The growth of $L(\boldsymbol{\beta})$ is steepest along the direction of $\nabla L(\boldsymbol{\beta})$.

    $\Leftrightarrow$ The descent of $L(\boldsymbol{\beta})$ is steepest along the direction of $-\nabla L(\boldsymbol{\beta})$.

- Training by gradient descent algorithm
  - The expression for the gradient $\nabla L(\boldsymbol{\beta})$ is:

$$\nabla L(\boldsymbol{\beta}) = -\sum_{i=1}^{N} \frac{y_i \boldsymbol{x}_i e^{-\boldsymbol{\beta}^t \boldsymbol{x}_i}}{1 + e^{-\boldsymbol{\beta}^t \boldsymbol{x}_i}}$$

  - The gradient $\nabla L(\boldsymbol{\beta})$ is obtained by calculating the partial derivatives of $L(\boldsymbol{\beta})$.

$$\nabla L(\boldsymbol{\beta}) = \begin{bmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \vdots \\ \frac{\partial L}{\partial \beta_K} \end{bmatrix}$$

- Training by gradient descent algorithm
  - $L(\boldsymbol{\beta})$ is minimized iteratively in small "steps" pushing $\boldsymbol{\beta}$ along the direction $-\nabla L(\boldsymbol{\beta})$

    1) $\boldsymbol{\beta}$ is randomly initialized.

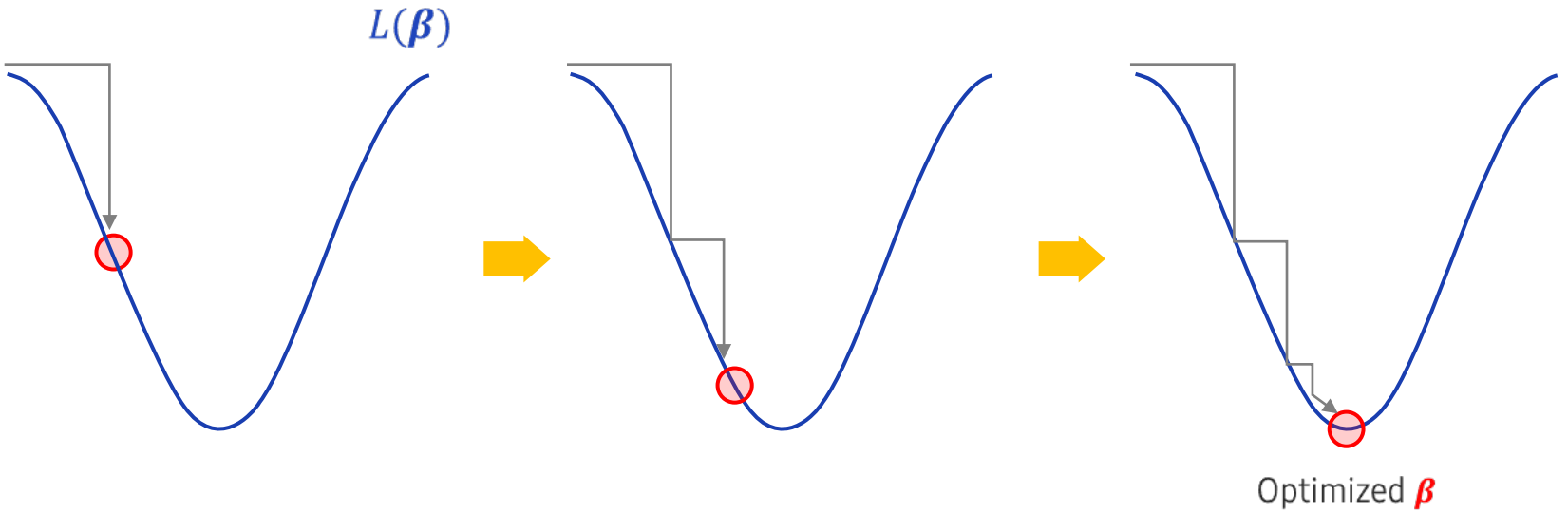
    2) Calculate the gradient $\nabla L(\boldsymbol{\beta})$.

    3) Update $\boldsymbol{\beta}$ by one step: $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \nabla L(\boldsymbol{\beta})$.

    Convergence speed is controlled by the "Learning rate" η.

    4) Repeat from the step 2) for a fix number of times (epochs).

- Training by gradient descent algorithm
  - $L(\boldsymbol{\beta})$ is minimized iteratively in small “steps” pushing $\boldsymbol{\beta}$ along the direction $-\nabla L(\boldsymbol{\beta})$



- Training by gradient descent algorithm
  - The gradient function in Python

In [1]:
```
def sigmoid(x):
    s=1.0/(1.0 + np.exp(-x))
    return s
def gradient(X,Y,beta):
    z = np.dot(X.beta.T)*Y
    ds = -Y*(1-sigmoid(z))*X
    return ds.sum(axis=0)
```

- Training by gradient descent algorithm
  - The gradient function in Python

In [2]:
```
def train(self, input_X, input_Y, n_epochs):
    ones_column = np.ones((input_X.shape[0],1))
    X = np.concatenate((ones_column, input_X), axis=1)
    Y = (2*input_Y - 1).reshape(-1,1)
    for n in range(n_epochs):
        self.beta = self.beta - self.rate*gradient(X,Y,self.beta)
    return self.beta
```

Unit 5a.

Application of Supervised Learning Model for Classification

- 5a.1. Training and Testing in Machine Learning
- 5a.2. Logistic Regression Basics
- 5a.3. Logistic Regression Performance Metrics

Logistic Regression Performance Metrics

- Confusion matrix
  - In the classification machine learning method, the most commonly used method for results evaluation of analysis model is the metric calculation including classification accuracy by using confusion matrix.
  - The confusion matrix refers to the matrix which makes a crosstable of predicted classification category from the analysis model and actual classified category of data.

• Confusion matrix

Predicted categorical value

Actual categorical value	Y	N
Y	**O (TP: True Positive)**	**X (FN: False Negative)**
N	**X (FP: False Positive)**	**O (TN: True Negative)**

- Confusion matrix
  - Confusion matrix can be made with a 2X2 cross table along with 3X3 and higher crosstables. However, for convenience in explanation, only 2X2 confusion matrix will be used in this chapter.
  - In the confusion matrix of the previous slide, the diagonally placed two O are cases when the predicted and actual categorical values are the same, which show that the classification machine learning predicted the results properly.
  - On the other hand, other parts are cases when the predicted and actual categorical values are not the same, meaning that the machine learning model made incorrect predictions.
  - The categories that the analysis is mostly interested in are positive categories, and the others are called negative categories. Depending on the accuracy of prediction (true or false) regarding positive and negative categories, accurate classification of interested categories is called TP (True Positive), while the accurate classification of uninterested categories is called TN (True Negative).
  - Inaccurate classification of uninterested categories into interested categories is called FP (False Positive), and inaccurate classification of interested categories into uninterested categories is called FN (False Negative).
  - There are various kinds of metrics based on different combinations of TP, TN, FP, FN of the confusion matrix for analysis result evaluation of classification machine learning methods.

- Metric
  - Major metrics that can be calculated from the confusion matrix provided above include accuracy, error rate = 1-accuracy, sensitivity (also referred to as recall, hit ratio, TP rate, etc.), specificity, FP rate, precision, and others. Among them, accuracy, sensitivity, and precision are the most frequently used metrics.
  - Also, there are F-Measure (or F1-Score) that combines sensitivity and precision, and Kappa Statistics where the predicted and actual values of the analysis model are exactly the same. The calculation formulas and definitions of various kinds of metrics are provided in the following table <5-1>.

5a.3. Logistic Regression Performance Metrics

• Metric

Metric	Calculation formula	Definition
accuracy	(TP+TN) / (TP+TN+FP+FN)	Ratio of accurate prediction of actual classification category (Ratio of TP and TN from the entire prediction)
error rate	(FP+FN) / (TP+TN+FP+FN)	Ratio of inaccurate prediction of actual classification category (Identical to 1-accuracy)
sensitivity = TP Rate	(TP) / (TP+FN)	Ratio of accurate prediction to 'positive' from actual 'positive' categories (True Positive – also referred to as Recall, Hit Ratio, TP Rate)
specificity	(TN) / (TN+FP)	Ratio of accurate prediction to 'negative' from actual 'negative' categories (True Negative)
FP Rate	(FP) / (TN+FP)	Ratio of inaccurate prediction to 'positive' from actual 'negative' categories = 1-specificity
precision	(TP) / (TP+FP)	Ratio of actual 'True Positive' from the ratio of predicted 'positive.'
F-Measure (F1-Score)		Ranged between 0~1, it is the harmonic mean between precision and sensitivity (recall). If both of the precision and sensitivity are high, f-Measure also tends to have a larger value.
Kappa Statistic		The value after eliminating coincidental agreement between predicted and actual values of the model. (Ranged between 0~1, and when the value is closer to 1, the predicted and actual values of the model coincide accurately, and the values do not coincide when the value gets closer to 0.)

- Metric
  - Among the metrics from the previous slide, sensitivity signifies how well the actual ‘positive’ category is predicted ‘positive,’ and the precision is the index that shows the ratio of actual ‘positive’ from the predicted ‘positive’ categories, so they are metrics that directly explain how well the classification machine learning analysis model classifies interested categorical values of objective variables.
  - Thus, sensitivity and precision are the most frequently used and most significant metrics for classification machine learning results in real life.

- Confusion matrix

Ex

	Actual 0	Actual 1
Predicted 0	120	5
Predicted 1	15	20

- Confusion matrix is a contingency table that counts the frequencies of the actual vs predicted.

- Confusion matrix: Accuracy

Ex

	Actual 0	Actual 1
Predicted 0	120	5
Predicted 1	15	20

$$\text{Accuracy} = \frac{Number\ of\ correct\ predictions}{Total\ number\ predictions}$$

- Accuracy is the ratio between the diagonal sum and the total sum.

- Confusion matrix: Sensitivity

Ex

	Actual 0	Actual 1
Predicted 0	120	5
Predicted 1	15	20

$$\text{Sensitivity} = \frac{Number\ of\ correctly\ predicted\ 1s}{Total\ number\ of\ 1s}$$

- Confusion matrix: Specificity

Ex

	Actual 0	Actual 1
Predicted 0	120	5
Predicted 1	15	20

$$\text{Specificity} = \frac{Number\ of\ correctly\ predicted\ 0s}{Total\ number\ of\ 0s}$$

- Confusion matrix: Precision

Ex

	Actual 0	Actual 1
Predicted 0	120	5
Predicted 1	15	20

$$\text{Precision} = \frac{Number\ of\ correctly\ predicted\ 1s}{Total\ number\ of\ 1s}$$

- Note
  - Accuracy alone is not sufficient for testing.

    Ex If frauds constitute only 1% of all the transactions, a fraud detection system (FDS) that predicts as non-fraud all of the transactions, the accuracy would be quite high at 99%.

    However, such FDS would be useless because it misses out that 1% that really matters.

  - We should also consider metrics other than just the accuracy.

- Terminology

$$\text{Accuracy} = \frac{Number\ of\ correct\ predictions}{Total\ number\ predictions}$$

$$\text{Sensitivity} = \frac{Number\ of\ correctly\ predicted\ 1s}{Total\ number\ of\ 1s}$$

$$\text{Specificity} = \frac{Number\ of\ correctly\ predicted\ 0s}{Total\ number\ of\ 0s}$$

$$\text{Precision} = \frac{Number\ of\ correctly\ predicted\ 1s}{Total\ number\ of\ 1s}$$

$$\text{Cohen's kappa } \kappa = \frac{Accuracy - p_e}{1 - p_e}$$

where p_e is the probability of correct prediction by chance.

5a.3. Logistic Regression Performance Metrics

- Terminology

True Positive Rate = Sensitivity

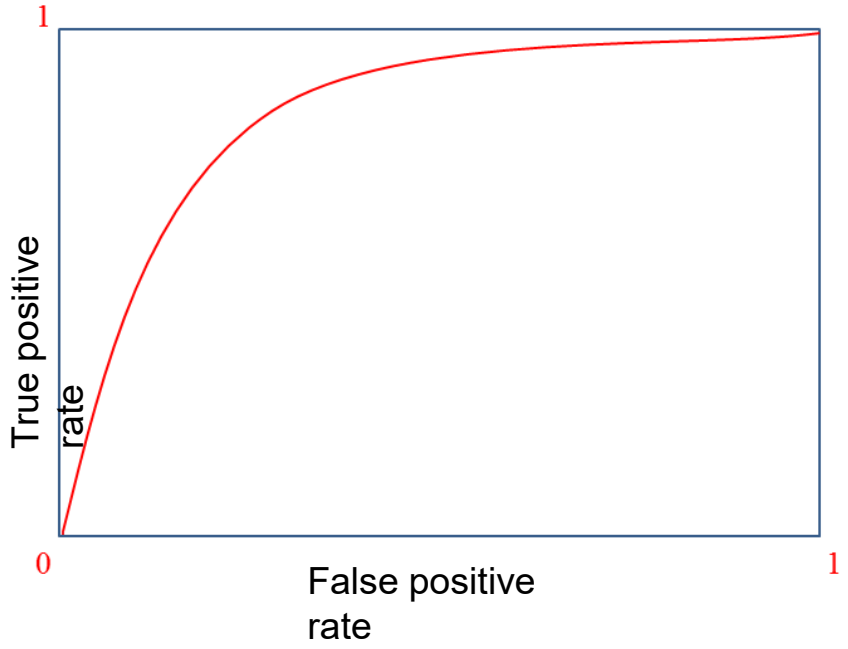
True Negative Rate = Specificity

$$\text{False Positive Rate} = \frac{Number\ of\ actual\ 0s\ that\ are\ incorrectly\ predicted\ as\ 1}{Total\ number\ of\ 0s} = 1\text{- Specificity}$$

$$\text{False Negative Rate} = \frac{Number\ of\ actual\ 1s\ that\ are\ incorrectly\ predicted\ as\ 0}{Total\ number\ of\ 1s} = 1\text{- Sensitivity}$$

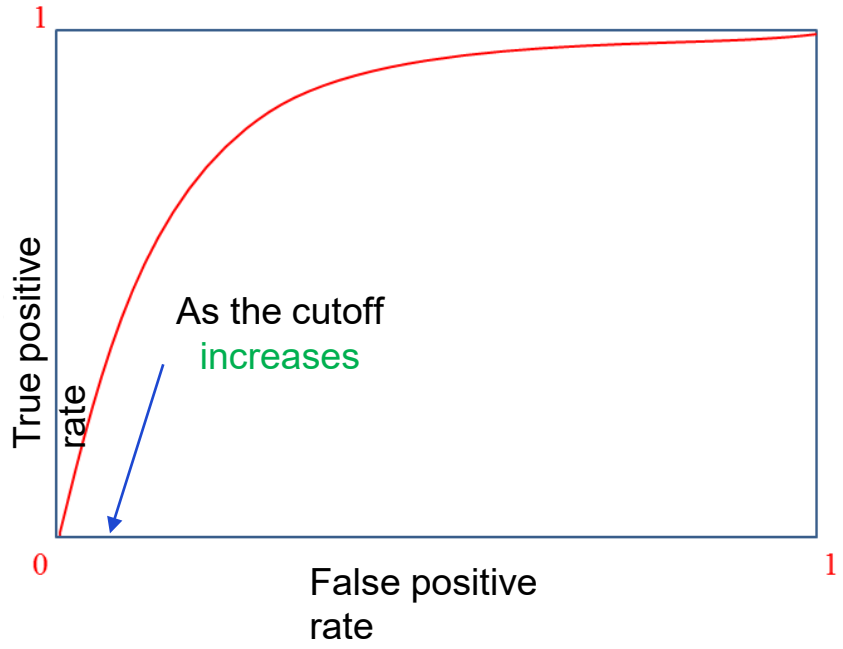
Positive Predicted Value = Precision

- ROC curve



- ROC curve is a parametric plot with respect to the *cutoff* probability.

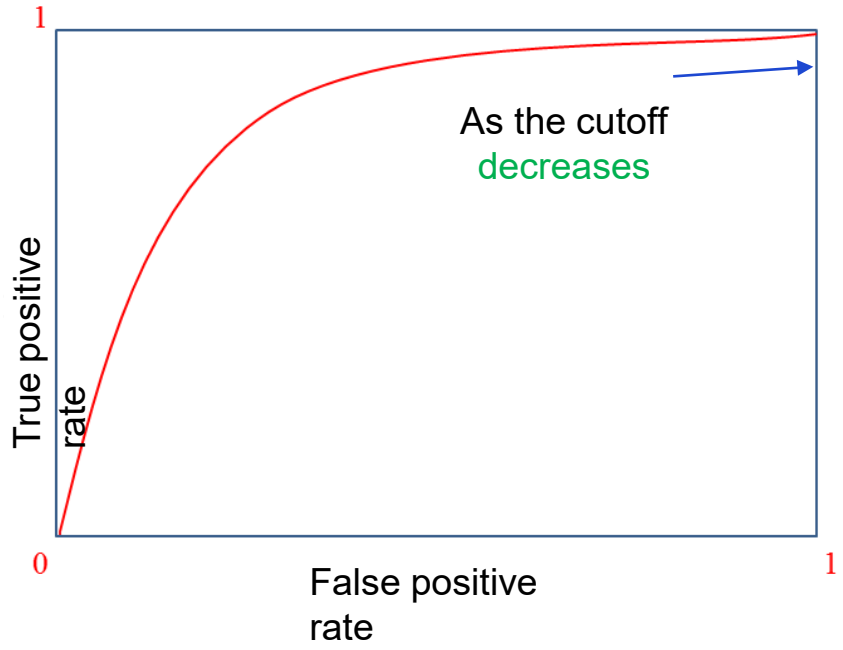
• ROC curve



As the cutoff increases (closer to 1)	
Performance metric	Increase/ Decrease
True Positive Rate (Sensitivity)	⇓
Specificity	⇑
False Positive Rate (1-Specificity)	⇓
Precision	⇑

- ROC curve is a parametric plot with respect to the *cutoff* probability.

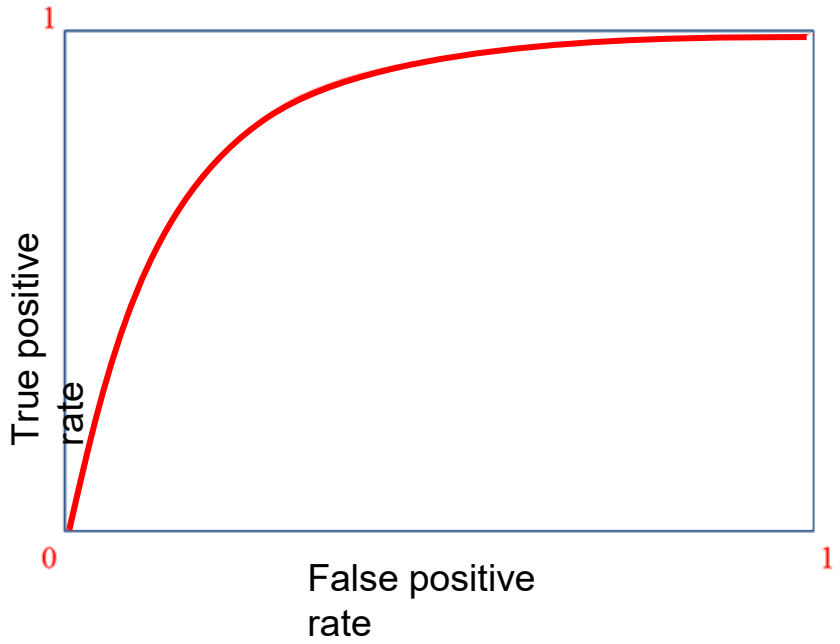
- ROC curve



As the cutoff decreases (closer to 0)	
Performance metric	Increase/ Decrease
True Positive Rate (Sensitivity)	⇑
Specificity	⇓
False Positive Rate (1-Specificity)	⇑
Precision	⇓

- ROC curve is a parametric plot with respect to the *cutoff* probability.

- ROC curve



- AUC stands for Area Under the Curve.
- AUC closer to 1 means good overall performance.

- Interpretation using Bayes' theorem

Ex There are 100 observations out of which in 6 cases the actual response is 1.

In the rest of 94 cases, the actual response is 0. It is known that for a given logistic regression model, the sensitivity = 0.92 and the specificity = 0.90.

For a new observation (only explanatory variables), this model predicts 1 as the response.

What is the probability of this prediction being correct?

a) We have $P(1) = 0.06$, $P(0)=0.94$

$P(Predicted\ 1|Actual\ 1) = 0.92$ ⇦ "Sensitivity"

$P(Predicted\ 0 \mid Actual\ 0)=0.90$ ⇦ "Specificity"

We can also derive $P(Predicted\ 1 \mid Actual\ 0)=1-P(Predicted\ 0 \mid Actual\ 0)=0.1$

- Interpretation using Bayes' theorem

Ex There are 100 observations out of which in 6 cases the actual response is 1.

In the rest of 94 cases, the actual response is 0. It is known that for a given logistic regression model, the sensitivity = 0.92 and the specificity = 0.90.

For a new observation (only explanatory variables), this model predicts 1 as the response.

What is the probability of this prediction being correct?

b) The answer we are seeking is given by

$$P(Actual\ 1|Predicted\ 1) = \frac{P(Predicted\ 1|Actual\ 1)P(1)}{P(Predicted\ 1|Actual\ 1)P(1) + P(Predicted\ 1|Actual\ 0)P(0)}$$

$$= \frac{0.92 \times 0.06}{0.92 \times 0.06 + 0.1 \times 0.94} \cong 0.37$$

Coding Exercise #0501

Follow practice steps on 'ex_0501.ipynb' file.

Unit 5b.

Decision Tree

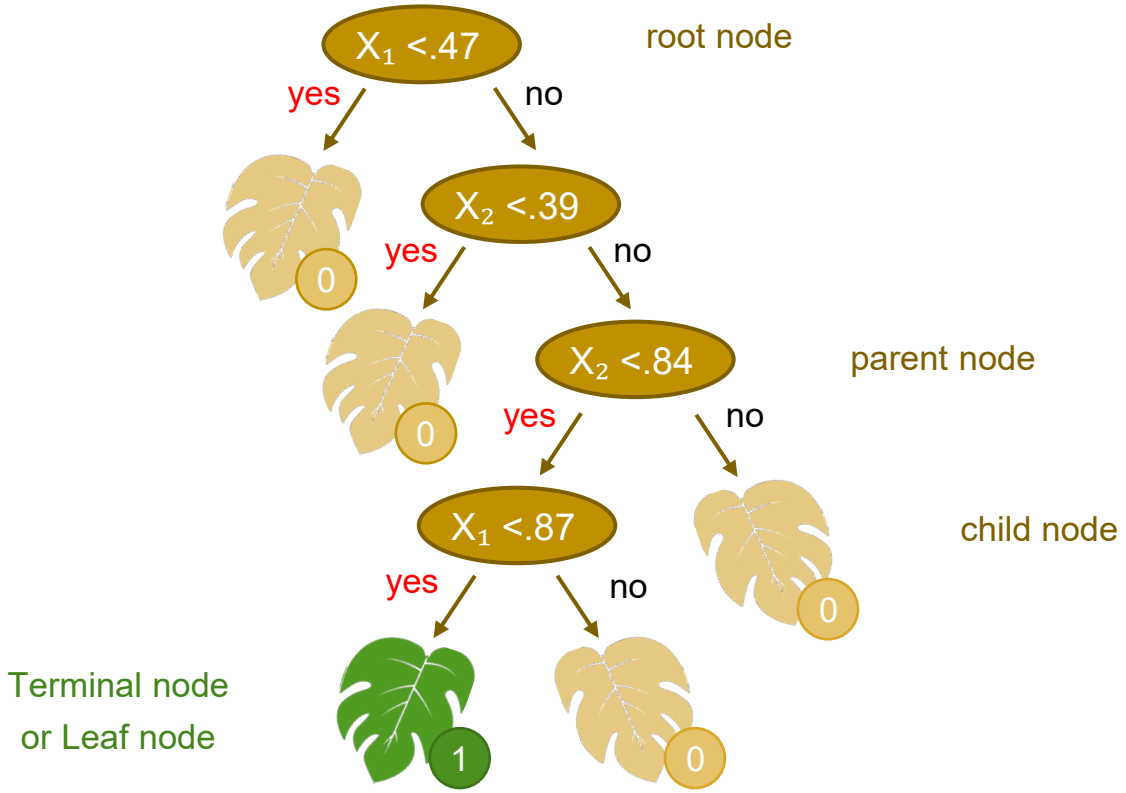
- 5b.1. Tree Algorithm

Overview of Decision Tree

- Definition
  - A classification model that analyzes data that were collected from the past and expresses the patterns found (characteristics of each category) as combination of features.
- Purpose
  - Classification of unseen data and prediction of categorical values
  - Extraction of generalized knowledge in tree structure from the data
- Classification depending on the type of objective variables
  - Categorical variable: Classification Tree
  - Continuous variable: Regression Tree

Composition

- Node, Branch, Depth



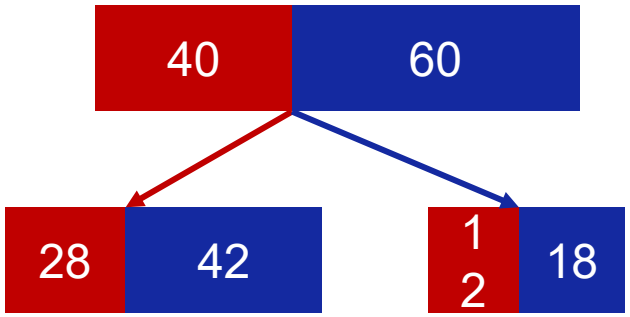
How to construct a decision tree model

Construction of a decision tree

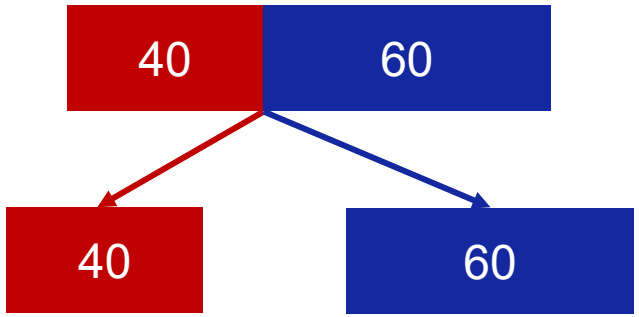
Construction of a decision tree	Making a decision tree by designating appropriate split criterion and stopping rule according to the purpose and data structure of analysis
Branching	Removing branches that have a high risk of error rate or inappropriate rules
Validity evaluation	Evaluation of the decision tree through cross validation using the gain chart, risk chart, or test data.
Interpretation and prediction	Interpretation of the decision tree and setting a prediction model

Split criterion of the decision tree

- Analysis process of a decision tree
  - Repetitive splitting: Repetitive splitting of the dimensional space of independent variables using training data
  - Branching: Branching with evaluation data
- Split criterion
  - Create the classification tree so that the purity of the child node is greater than the that of parent node.



No Improvement

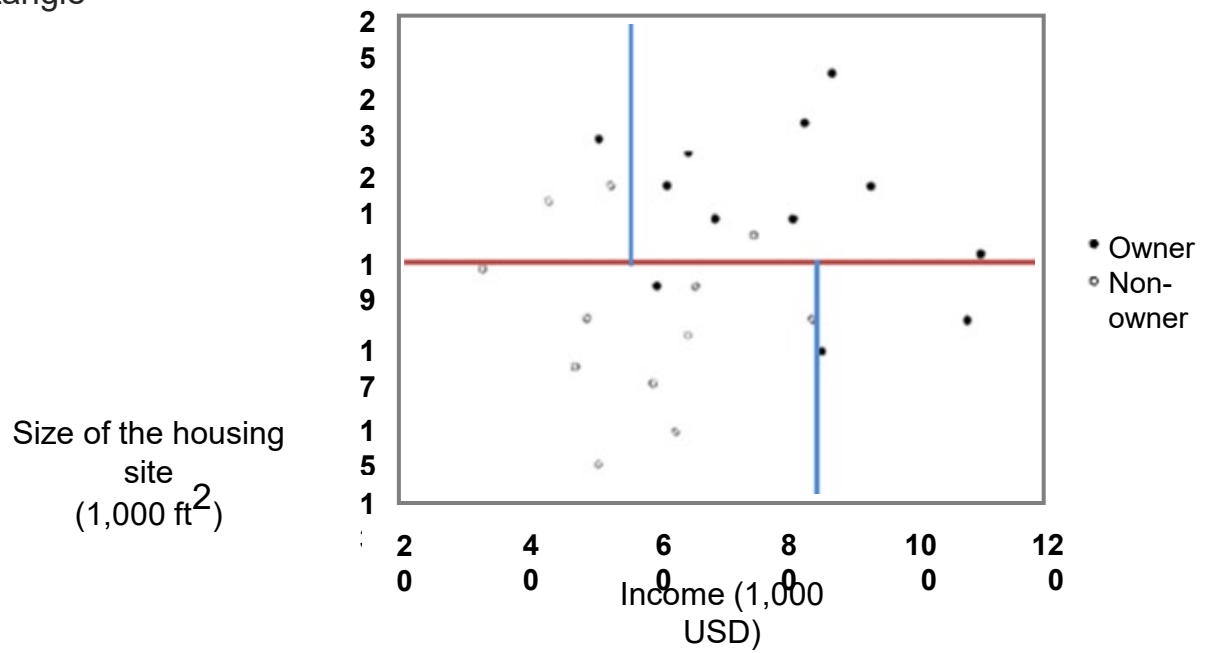


Perfect Split

Repetitive splitting process

Purpose

- Divide the entire space into rectangles and make each of it as pure or homogeneous as possible.
- Definition of ‘pure’:
  - Divide the area into pure, or homogenous, rectangular spaces as much as possible
  - All variables in the final rectangle belong to the same group.



Repetitive splitting process

1) Select xi which is one of the variables and the xi value (si as split criterion) is designated to split the p-dimension space into two.
2) xi → {xi<=si} ∪ {xi>si}
3) Select a variable again and split in the same way.
4) Repeat the process until it reaches desired purity.

Split criterion

- Discrete objective variables
  - Chi squared statistic – p value: Creates child nodes with predictor variable with the lowest P value and the optimal partitioning
  - Gini index: Selects child nodes with predictor variable that reduces the Gini index and the optimal partitioning
  - Entropy measure: Creates child nodes with predictor variable with the lowest entropy measure and the optimal partitioning

Split criterion

Continuous objective variables

- F statistic in ANOVA: Creates child nodes with predictor variable with the lowest P value and the optimal partitioning
- Variance reduction: Creates child nodes with the optimal partitioning that maximizes variance reduction

Selection of algorithms and classification variables

	Discrete objective variables	Continuous objective variables
CHAID (multi space partitioning)	Chi squared statistic	ANOVA F statistic
CART (binary space partitioning)	Gini index	Variance reduction
C4.5	Entropy measure	

Impurity measure

Gini index

- Selects child nodes with predictor variable that reduces the Gini index and the optimal partitioning
- If the T data set is split into k categories and the category performance ratios are p1 , •••, pk, it is expressed as the following equation.

$$Gini(T) = 1 - \sum_{l=1}^{k} p_l^2$$

high impurity(diversity), low purity

GI = 1 –(3/8)²–(3/8)²–(1/8)²–(3/8)²= .69

low impurity(diversity), high purity

GI = 1 –(6/7)²–(1/7)²= .24

Entropy measure

- In thermodynamics, entropy measures degree of disorder.
- Creates child nodes with predictor variable with the lowest entropy measure and the optimal partitioning.
- If the T data set is split into k categories and the category performance ratios are p1 , •••, pk, it is expressed as the following equation.

$$Entropy(T) = -\sum_{l=1}^{k} p_l \log_2 p_l$$

Ex If 4 categories consist of ratios of 0.25, 0.25, 0.25, 0.25 (T0)

$$Entropy(T_0) = -(0.25 \log_2 0.25) * 4 = 1.39$$

Ex If 4 categories consists of ratios of 0.5, 0.25, 0.25, 0 (T1)

$$Entropy(T_1) = -(0.5 \log_2 0.5 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25) = 1.04$$

Stopping criteria

- A rule to designate the current node as terminal node without further splitting
  - Designates the depth of decision tree
  - Designates the minimum number of records in the terminal node

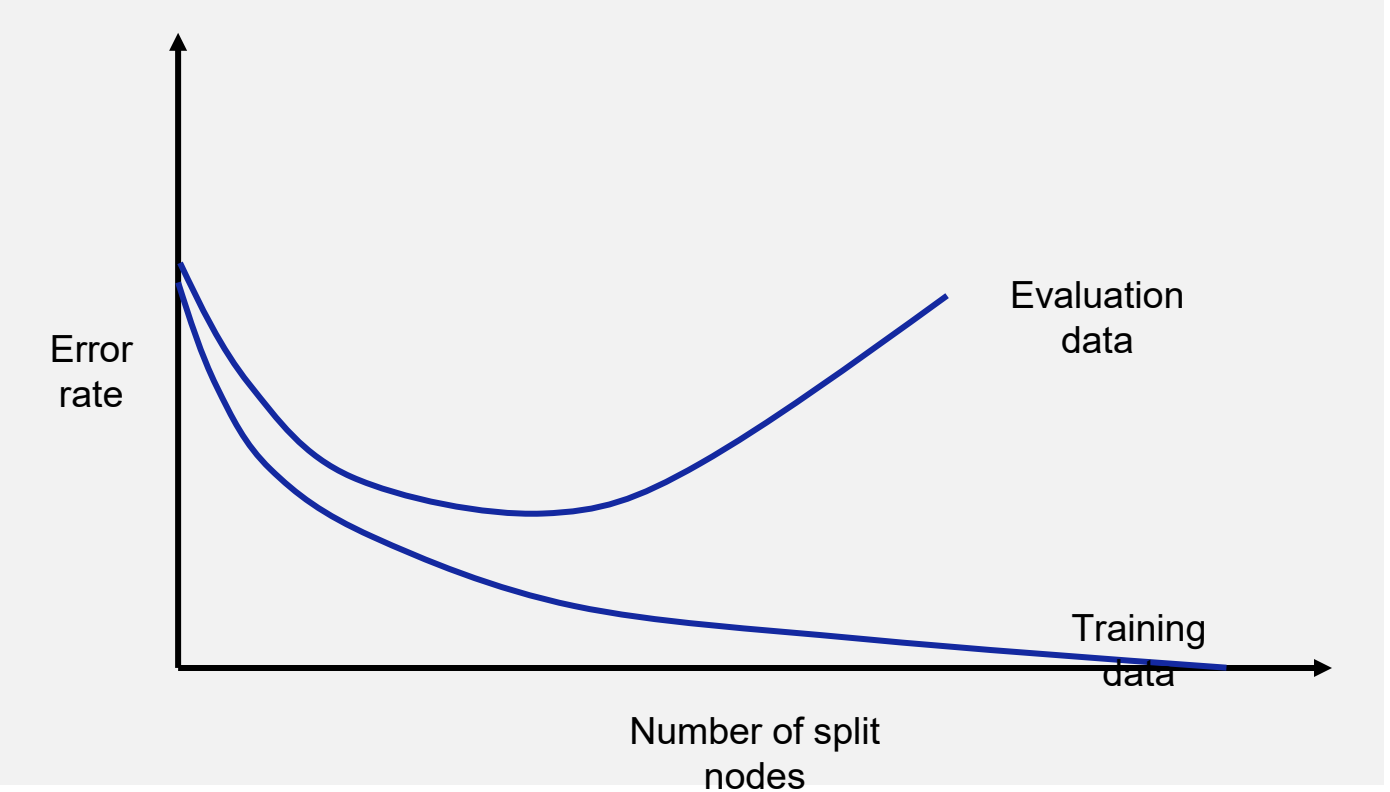
Branching criteria

- Application of test data
  - Application of the test data to the constructed model
  - Reviewing the predictive value of the constructed model through test data
  - Removing the branches that have a high risk of error rate or inappropriate rule of inference

- By an expert
  - An expert reviews the validity of rules suggested in the constructed model
  - Removing rules without validity

Overfitting problem

Overfitting problem graph



Pros

- Creation of understandable rules (can be expressed with SQL)
- Useful in classification prediction
- Able to work with both continuous and discrete variables
- Shows a more relatively significant variable

Cons

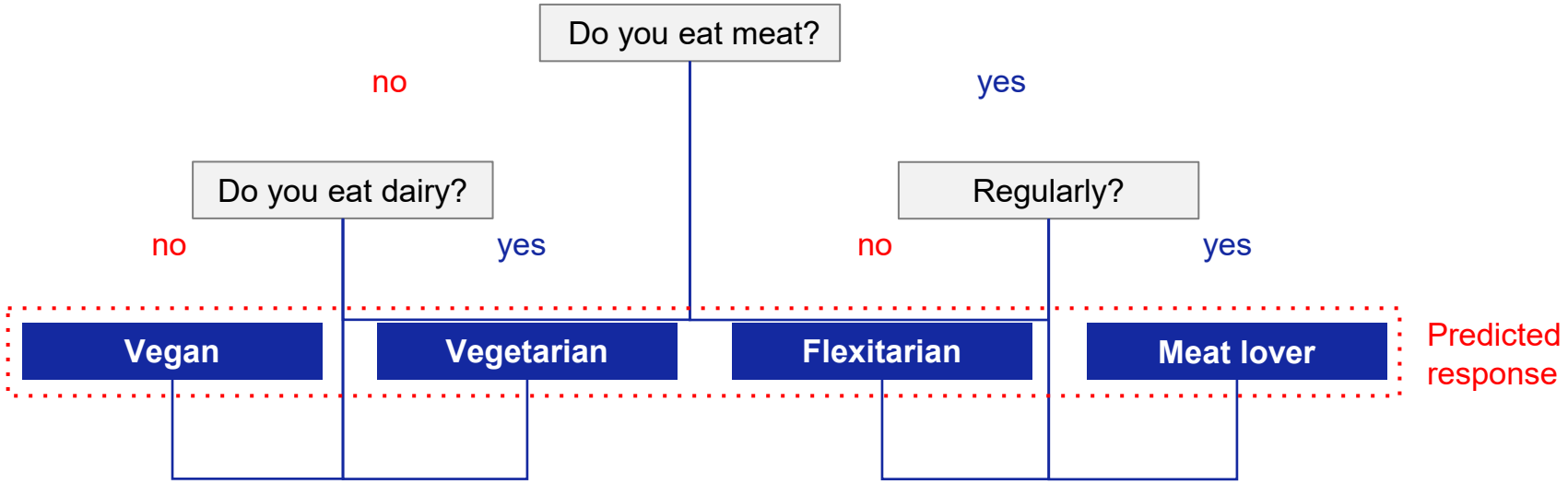
- Not suitable to predict continuous variable values
- Unable to perform time series analysis
- Non-stable

Tree Algorithm

- Pros
  - Intuitive and easy to understand.
  - No assumptions about the variables.
  - No need to scale or normalize data.
  - Not that sensitive to the outliers.
- Cons
  - Not that powerful in the most basic form.
  - Prone to overfitting. Thus, “pruning” is often required.

Classification Tree

Ex



- Training step creates an inverted tree structure as above.
- Conditions are evaluated at the nodes and then branch out.
- Each leaf node corresponds to a **region** in the configurational space.

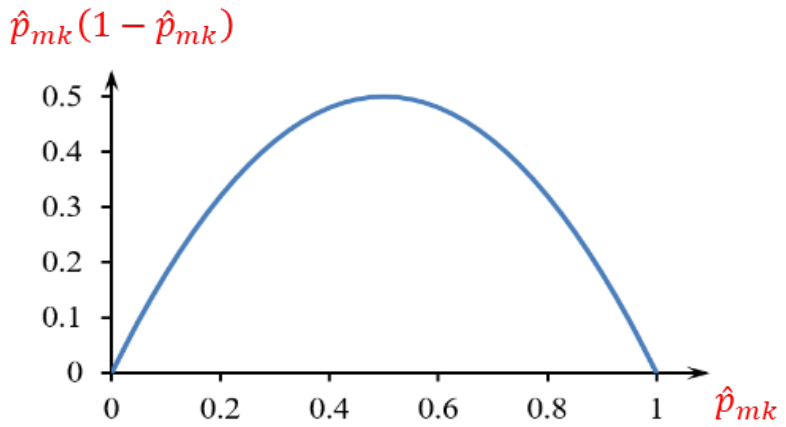
Classification Tree

- The tree structure is trained by minimizing the Gini impurity (or entropy).

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} {\hat{p}_{mk}}^2$$

or

$$Entropy_m = -\sum_{k=1}^{K} \hat{p}_{mk} Log(\hat{p}_{mk})$$



- G_m is the Gini impurity in the leaf node m.
- $Entropy_m$ is the entropy in the leaf node m.

} The smaller, the better.

- Here, $\hat{p}_{mk}$ is the proportion of the class k in the leaf node m.
- K is the total number of possible classes.
- The class with the largest proportion is the prediction at that leaf node.

Classification Tree: procedure

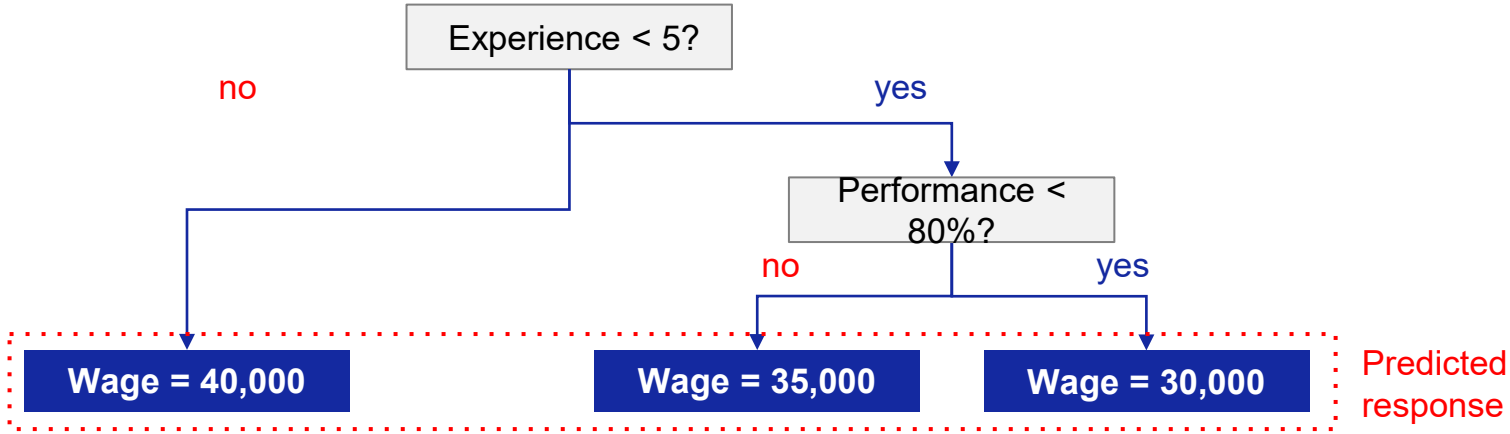
a) Make a basic tree.

b) Prune branches that do dot provide better performance.
Pruning can be done during the cross-validation step.

c) Predict with the optimized tree.

Regression Tree

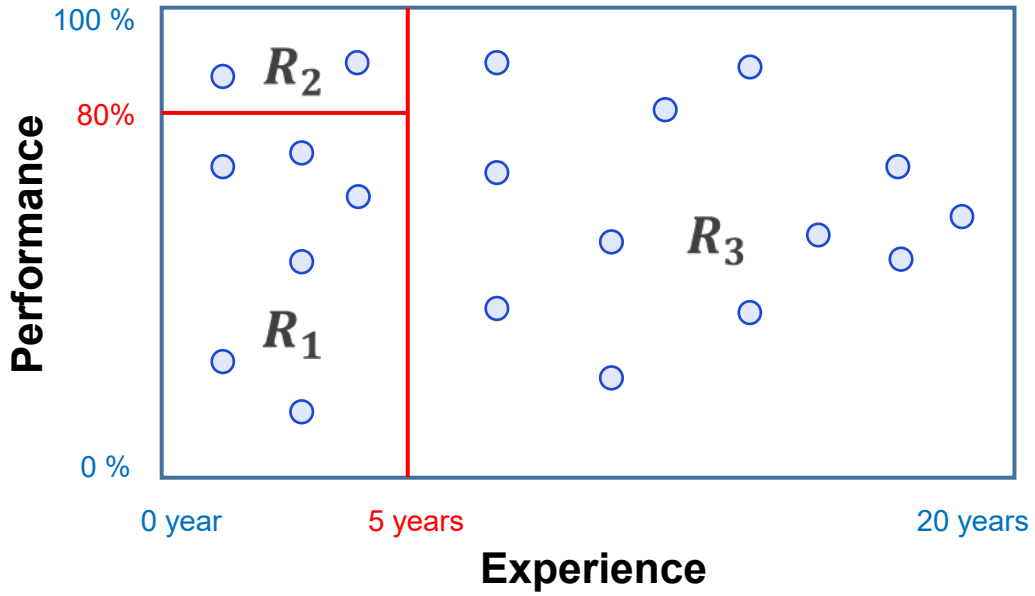
Ex



- Predicts numeric values rather than categories or classes.
- Conditions are evaluated at the nodes and then branch out.
- Each leaf node corresponds to a region in the configurational space.

Regression Tree

Ex



- Each leaf node corresponds to a region in the configurational space.

Regression Tree

- The configurational space is split into regions: {$R\_1, R\_2, \cdots, R\_J$ }
- The tree structure is trained by minimizing the RSS (residual sum of squares):

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$$

  - J is the total number of leaf nodes.
  - The predicted value in the j-th region $\hat{y}_{R_j}$ is given by the average in that region.
  - For the observations belonging to the region R_j, the same predicted response $\hat{y}_{R_j}$ is assigned analogous to the classification Tree.

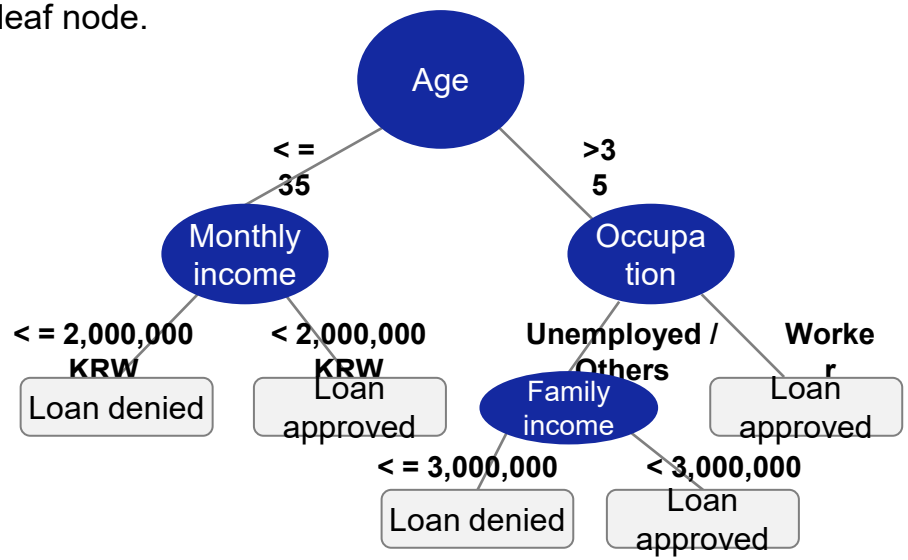
Scikit-Learn DecisionTreeClassifier/Regressor Hyperparameters:

Hyperparameter	Explanation
max_depth	The maximum depth of a tree.
min_samples_leaf	The minimum number of sample points required to be at a leaf node.
min_samples_split	The minimum number of sample points required to split an internal node.
max_features	The number of features to consider when looking for the best split.
max_leaf_nodes	The maximum number of leaf nodes in the tree.

- Need to be tuned for optimized performance.
- More information can be found at:
  Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
  Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

Decision Tree

- Decision tree refers to a modeling method that branches independent variables (explanatory variables or features) that affect classification or prediction of objective variables as a shape of tree from root to leaf nodes according to their reference values.
- In the decision tree, each node is split in the form of if-then depending on the characteristics or reference values of explanatory variables, so when following the tree structure it is possible to easily understand how the attribute value of data is classified into which category.
- The following figure provided below is a typical form of decision tree. From the example, ‘age’ is the root, and it can be inferred that ‘age’ is the most significant variable when deciding loan approval.
- The squared shape node at the end of each branch is the leaf node.



Split Criteria of Decision Tree

- Decision tree gives a question to each node and separates data by branching according to the response.
- In order to evaluate how well the data is separated, specific criterion is required. In general, impurity is the evaluation criterion, and the impurity becomes higher as various classifications are mixed in the node, and it is the lowest when there's only one classification in the node.
- Thus, lower impurity of each node after node splitting signifies the decision tree is well classified.
- Representative impurity measures are Gini impurity and entropy, etc.

$$\text{Gini } Coff. = 1 - \sum_{i=1}^{K} p_i^2 \text{ , } Entropy\ Coff. = -\sum_{i=1}^{K} p_i \log_2 p_i$$

- The impurity index is 0 when $p_i = 0$ or $p_i = 1$, and it gets the largest when $p_i = \frac{1}{2}$, thus making a parabola. In other words, the impurity index is the lowest when there's a certain classification in the node or the node is completely free from any classification. In contrast, the impurity becomes the largest when there are many classifications found in the same node.

Coding Exercise #0502

Follow practice steps on 'ex_0502.ipynb' file.

Unit 5c.

Naïve Bayes Algorithm

- 5c.1. Naïve Bayes Algorithm

Naïve Bayes Algorithm

- About Naïve Bayes algorithm
  - It is a straightforward application of the Bayes' theorem.

- Pros
  - Intuitive and simple.
  - Not that sensitive to the noise and outliers.
  - Fast.

- Cons
  - Assumes that the features are independent which may not be strictly true.
  - Not among the best performing algorithms.

- About Naïve Bayes algorithm
  - Bayes' theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

  - Now we take A=$Class$ and B=$Data$, then:

$$P_{post}(Class) = P(Class|Data) = \frac{P(Data|Class)P_{prior}(Class)}{P(Data)}$$

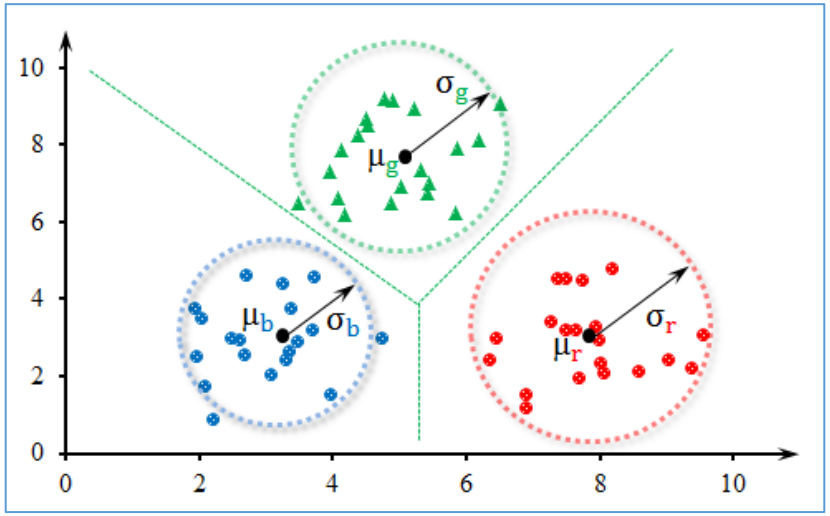
  - We are more interested in comparing relative probabilities.

$$P_{post}(Class) \propto P(Data|Class)P_{prior}(Class)$$

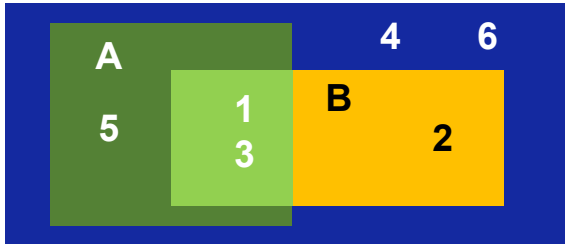
- About Naïve Bayes algorithm
  - We can approximate $P(Data \mid Class)$ by a Gaussian (normal distribution).

$$P_{post}(Class) \propto \frac{1}{\sqrt{2\pi}\sigma_j} exp\left(-\frac{1}{2{\sigma_j}^2}(x-\mu_j)^2\right) P_{prior}(Class)$$

where the parameters μ_j and σ_j are “learned” from the training data.



- Random variable
  - The variable whose values is unknown until the outcome
  - Independent events
    - If the probability of simultaneous occurring of two cases is identical to the multiplication of the probabilities of each event to occur, then the two events are independent to each other.
    - $P(A \cap B) = P(A)P(B)$
  - Dice

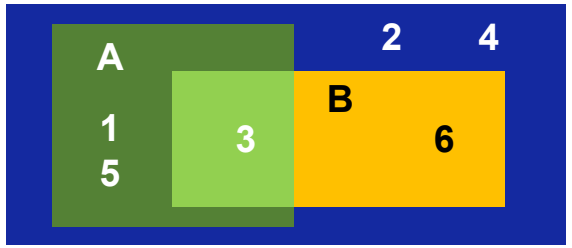


Conditional probability

- Events of B (1,2,3) when the A events (1,3,5) occur
- 0.6666667

		Total	Event A	Event B	Probability A	Probability B
Event A	Odd number	1	1	1	0.5	0.5
Event B	Less than 3	2		2		
		3	3	3		
		4				
		5	5			
		6				

- Random variable
  - The variable whose values is unknown until the outcome
  - Independent events
    - Not affected
  - Dice

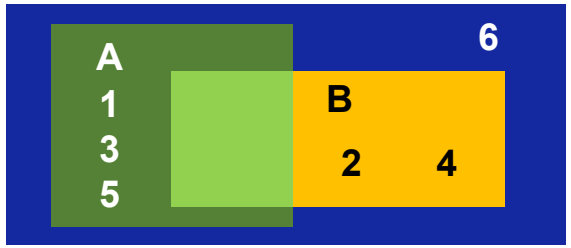


		Total	Event A	Event B	Probability A	Probability B
Event A	Odd number	1	1		0.5	0.333
Event B	Multiple of 3	2				
		3	3	3		
		4				
		5	5	6		
		6				

Conditional probability

- Events of B (3, 6) when the A events (1,3,5) occur
- 0.333333333

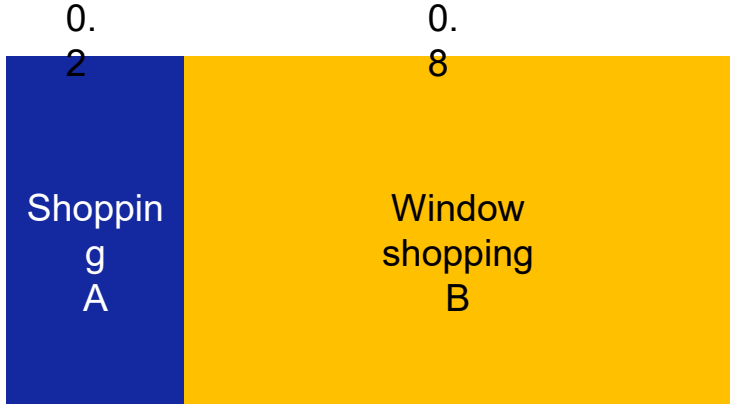
- Random variable
  - The variable whose values is unknown until the outcome
  - Exclusive events
    - Intersection is the null set
  - Dice



- Introduction to Bayesian statistics

1) Differentiate groups of people for ‘shopping’ and ‘window shopping’ through Bayesian interpretation.

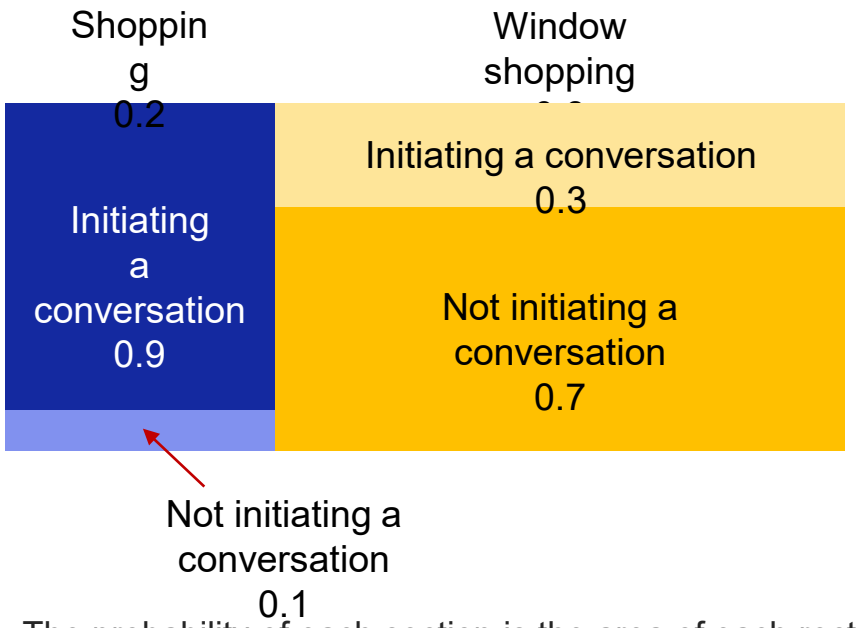
- Setting prior probability for different types
- Prior probability: Probability before getting certain information



- Introduction to Bayesian statistics

2) Set the conditional probability of 'initiating a conversation' for each type.

	Probability of initiating a conversation	Probability of not initiating a conversation	Total
Shopping	0.9	0.1	1
Window shopping	0.3	0.7	1

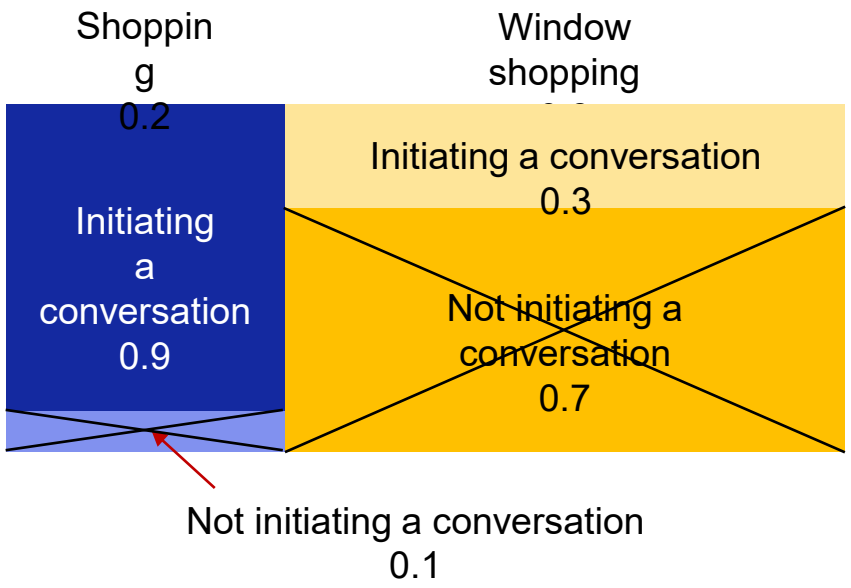


- The probability of each section is the area of each rectangle. (0.18, 0.02, 0.24, 0.56)

- Introduction to Bayesian statistics

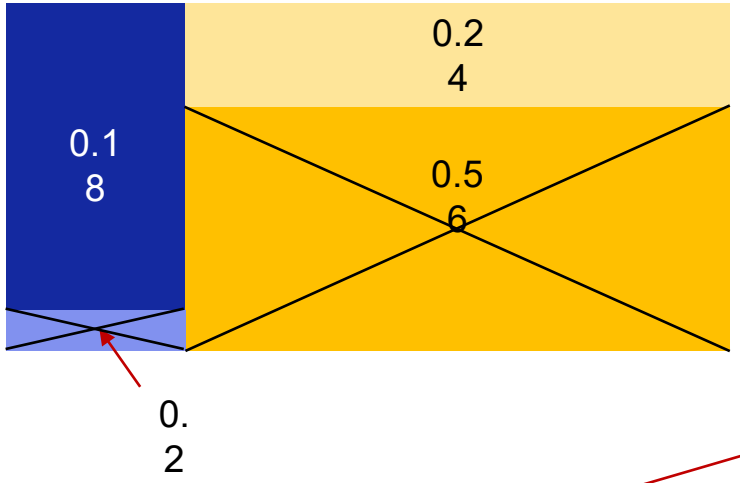
3) Remove ‘impossible event’ from the observations.

- The customer initiated a conversation. (additional information)



- Introduction to Bayesian statistics

4) Calculate Bayesian probability of the shopping group.



	Shopping	Window shopping
Normalization	3/7	4/7
	43%	

Result
Initiating a conversation → Cause
A certain group

- The result is 0.18:0.24, which is 3:4.
- So, can be estimated that the probability of the customer who initiated a conversation being in the shopping group is 3/7 (43%), and it is either called Bayesian statistics or posterior probability.
- Chooses if a person who initiated a conversation is in the shopping or window shopping group based on probability.

- Introduction to Bayesian statistics
  - The probability of a customer who initiated a conversation makes a purchase is now different.
  - Bayesian updating for customer type

Final summary	Prior probability of being a shopping group	0.2 ↘
	Observing to initiate a conversation	
	Posterior probability of being a shopping group	3/7 → 0.428571

  - The customer wouldn't be in the shopping group for sure, but the probability is double.
  - **Bayesian estimation** is '**performing Bayesian updating to the posterior probability based on the behavioral observation of prior probability.**'
  - Such estimation method is called Bayesian statistics.
  - **Bayes' theorem**
    - **Obtaining posterior probability based on prior probability**

Coding Exercise #0503

Follow practice steps on 'ex_0503.ipynb' file.

Unit 5d.

KNN Algorithm

- 5d.1. KNN Algorithm

KNN Algorithm

- About KNN (K Nearest Neighbors)
  - One of the simplest algorithms.
  - Prediction is based on the k nearest neighboring points.
  - There are classification and regression variants.
    - Classification: prediction decided by the majority class of the nearest neighbors.
    - Regression: prediction given by average of the nearest neighbors.

- About KNN (K Nearest Neighbors)
  - K-nearest neighbors is a method for making classification and prediction for classification of a data set where the category of the objective variable is unknown by designating the most similar category of the surrounding data set.
  - For KNN, specific criteria are required for how to measure the 'analogy' between the certain data set and surrounding data set and how many data sets will be used for final classification of the objective variable category.

1) Analogy measurement

- There are many ways to measure the analogy between data. In general, analogy calculation is done by inversing the squared Euclidean distance between two points or using Pearson correlation coefficient. If the points are discrete variables, use Jaccard coefficient.

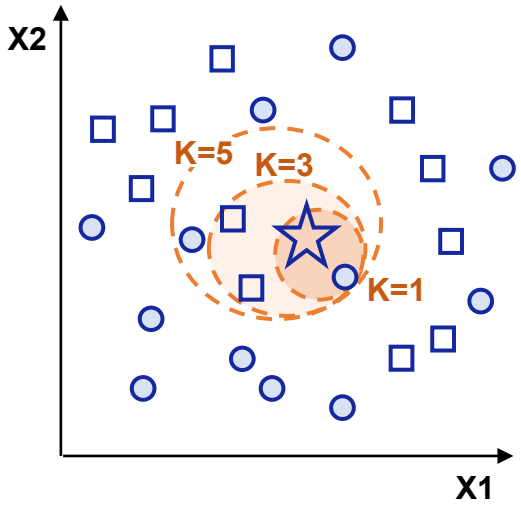
2) Objective variable classification criteria

- K in the KNN refers to number of surrounding data points that will be used for classifying objective variables of the specific data point after measuring the analogy between the specific data point and other surrounding data sets.

Ex Take movie recommendation as an example, where it is the recommendation of a movie that is similar to the customer's favorite movie and the customer did not watch yet. If so, when considering only one movie that is the most similar to the customer's favorite, it is '1-nearest neighbor,' and when considering three similar movies, it would be '3-nearest neighbor.'

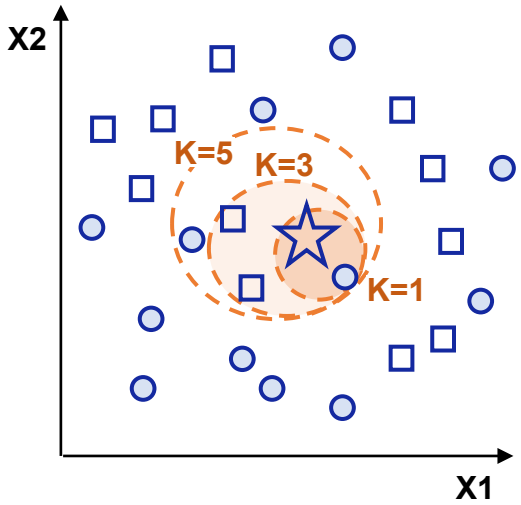
- Likewise, K-nearest neighbor is a method to determine a new category according to the principle of majority rule by finding k surrounding data points similar to the specific data point and considering the classification category of the specific data point.

• About KNN (K Nearest Neighbors)



- The figure above is a conceptual expression of the change in objective variables depending on K value setting in the K-nearest neighbor method. The '☆' point is the data value that needs to be classified, and the points shaped in a square and circle are other data points present around.
- When setting K=1, since the closest data point to the star is a circle, the objective variable of '☆' will be classified as '○.' On the other hand, if K=3, three points that are closest to '☆' will be considered, which include two squares and one circle. So, in this case, '☆' will be classified as '□'. Also, if K=5, because there are more circles around '☆,' it will be again classified as '○'.

• About KNN (K Nearest Neighbors)

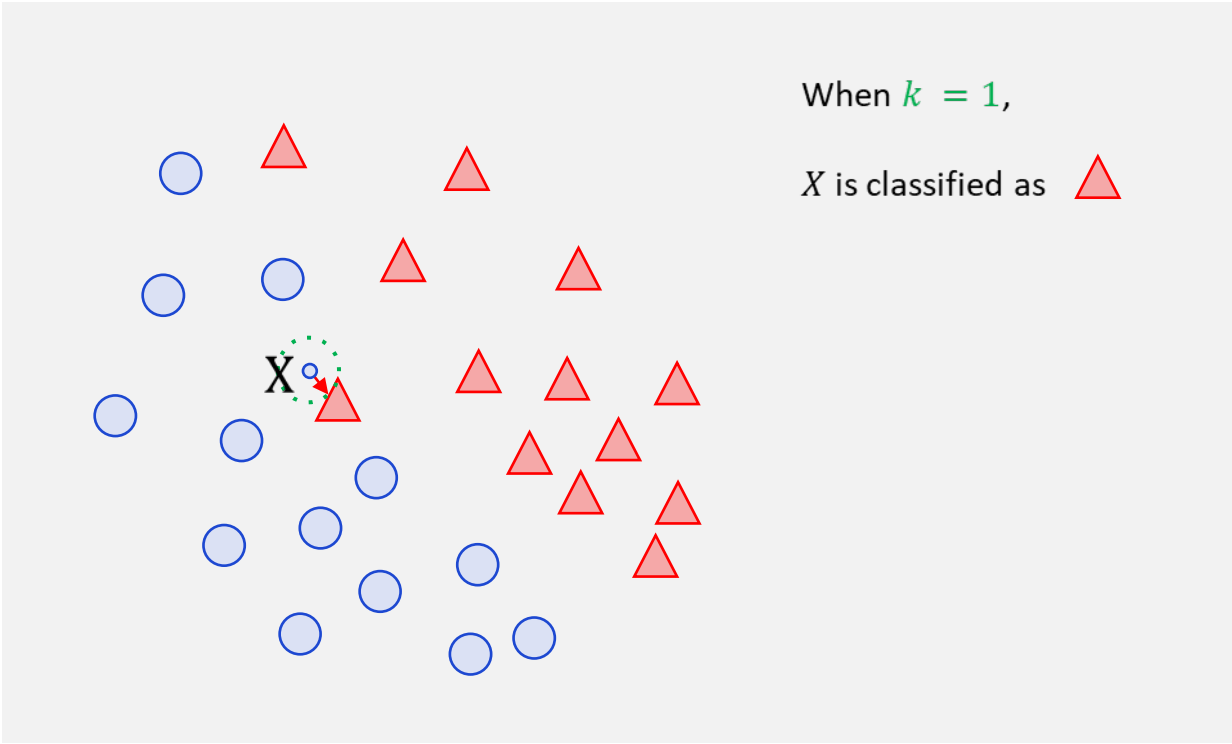


- So, in K-nearest neighbor, different K values significantly change the predicted result of the objective variable category. This is why setting an appropriate K value is important in the K-nearest neighbor method.
- However, there are no clear theoretical or statistical standards for an appropriate K value. Most of the time, different K values are set for repeated tests, and the final K value is set once it shows the optimal classification performance. In general, a random initial K value is designated between K=3 and K=9, which is then used to test the classification performance by using training and evaluation data for selecting the optimal K value.

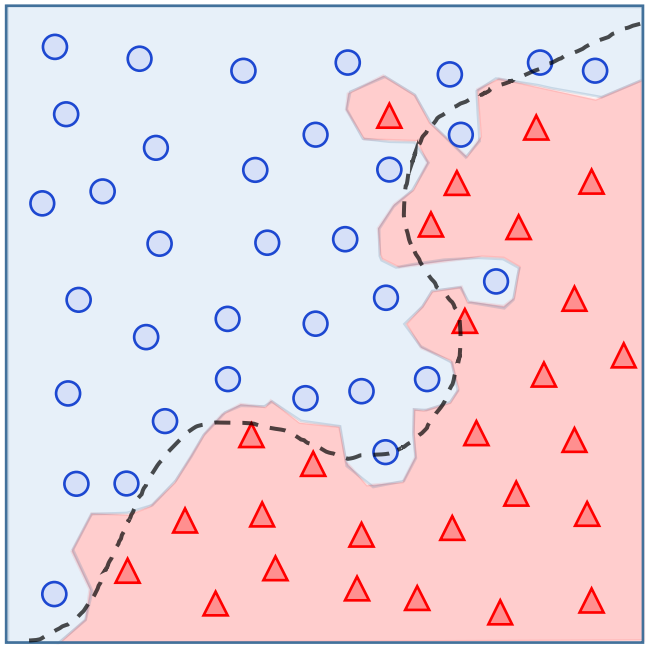
- Pros
  - Simple and intuitive.
  - No model parameters to calculate. So, there is no training step.

- Cons
  - Since there is no “model”, little insight can be extracted.
  - No model parameters that store the learned pattern. The training dataset is required for prediction.
  - Prediction is not efficient ⇨ “Lazy algorithm”.

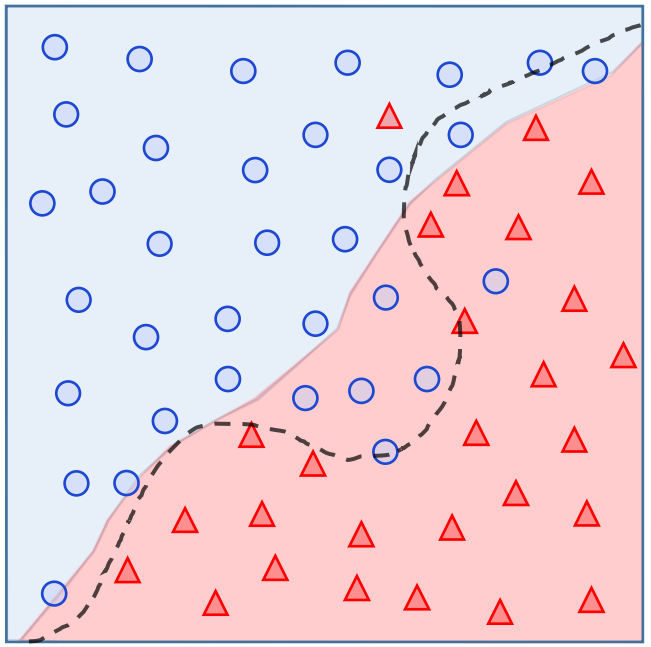
- KNN classification algorithm



- KNN classification algorithm
  - When k is too small, overfitting may happen.



  - When k is too small, overfitting may happen.



Coding Exercise #0504

Follow practice steps on 'ex_0504.ipynb' file.

Unit 5e.

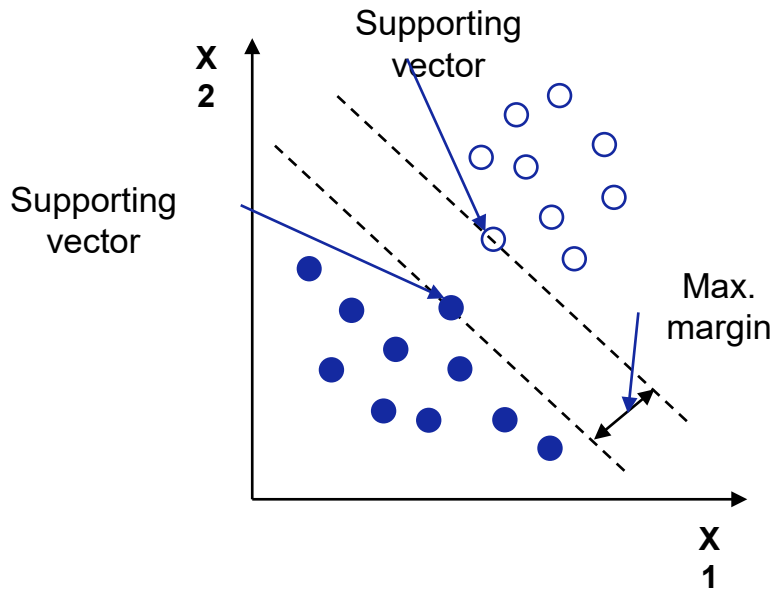
SVM Algorithm

- 5e.1. SVM Algorithm

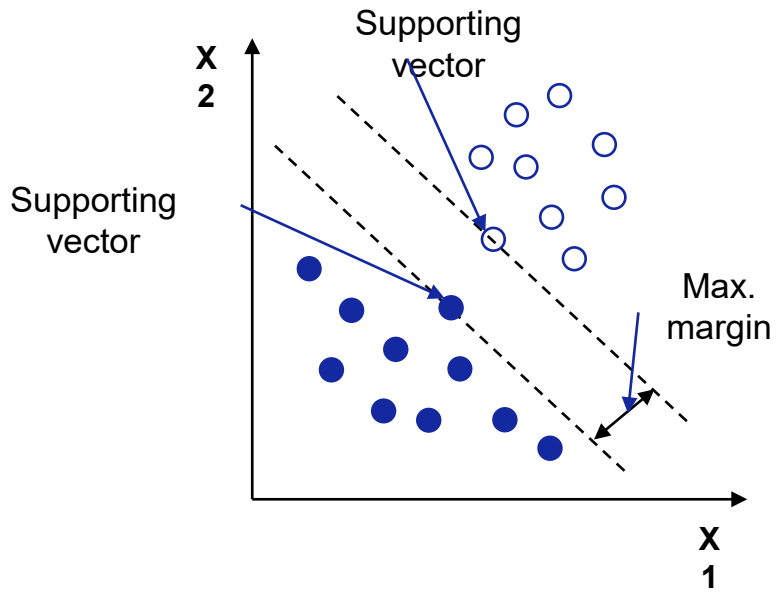
SVM Algorithm

- About SVM (Support Vector Machine):
  - Enhanced classification accuracy by maximizing the margin.
  - Effective non-linear classification boundary by the “kernel” transformation.

- About SVM (Support Vector Machine)
  - Support vector is a model that classifies data by finding the line (or hyperplane) where the distance (margin) between data in different categories becomes maximum.
  - As shown in the figure below, the support vector machine model finds the hyperplane that split the data in two different categories with maximum margin to classify data.
  - There would be many lines or planes that split data in two different categories, but points that go over the classification boundary can occur in evaluation data or unknown future data as the points near the boundary slightly change.



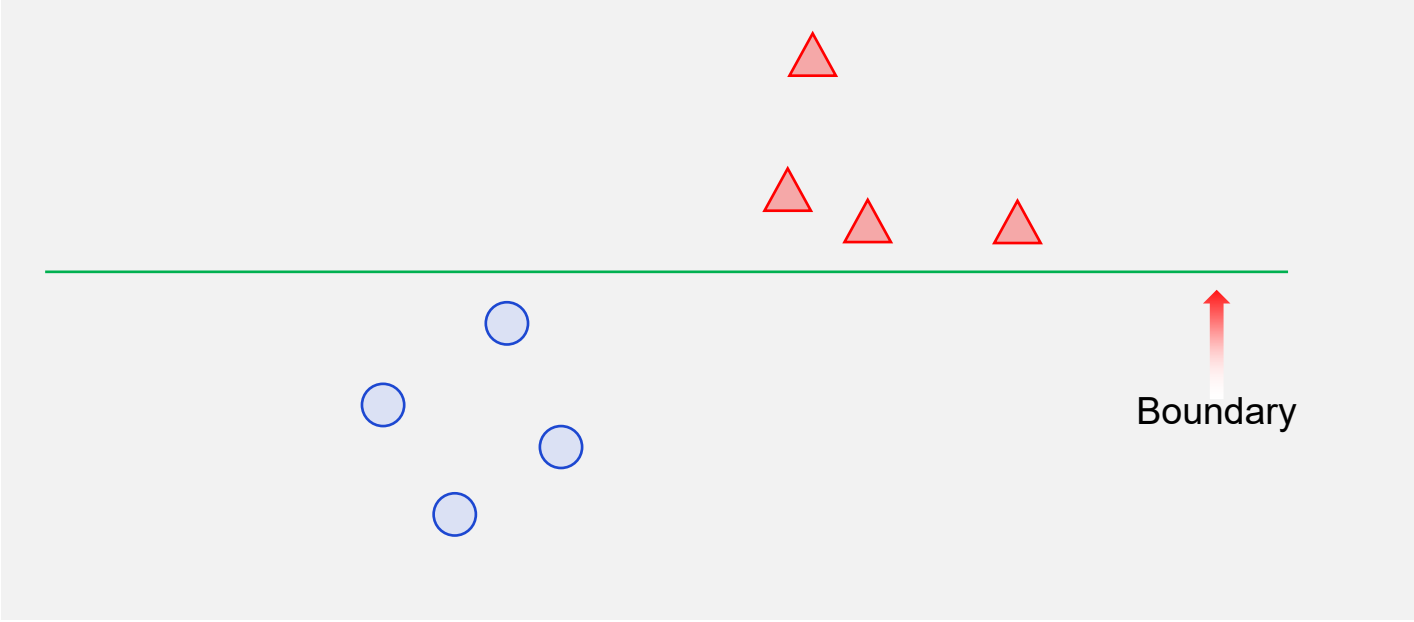
- About SVM (Support Vector Machine)
  - To minimize such possibility, a line (or hyperplane) should be found that makes the maximum margin between different data categories. In other words, the aim is to try to find a hyperplane that can induce maximum differentiation for the best possible generalization to classify and predict future data, not the current training data.
  - The point in the category closest to the boundary line is called a support vector, and each classification should have at least one support vector.



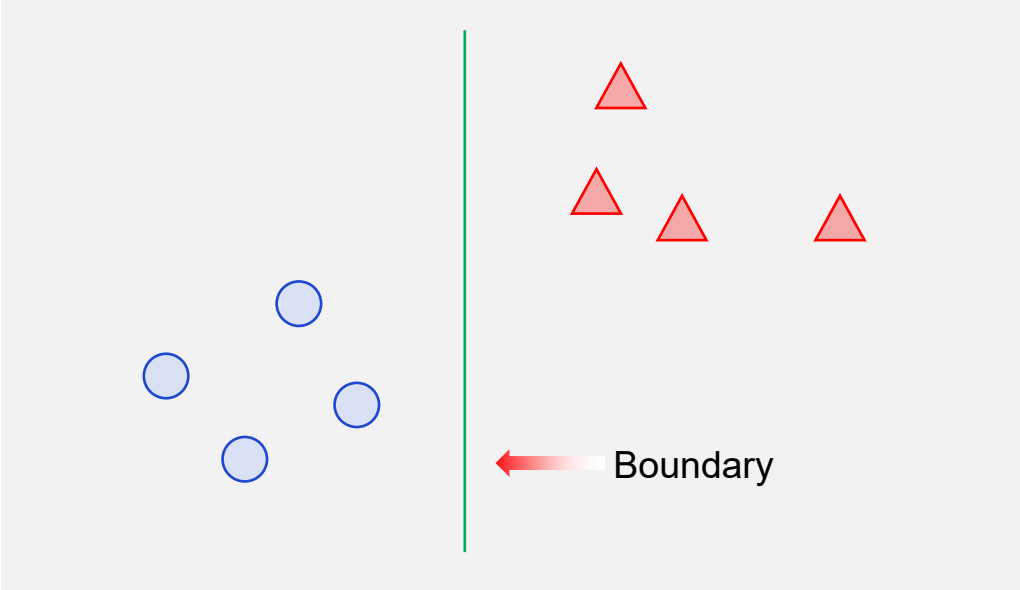
- About SVM (Support Vector Machine)
  - However, it is not always possible to classify all data linearly. Sometimes, data classification should be done in a curve or more complex non-linear classification plane.
  - If so, use the Kernel trick method for mapping the given data into a higher dimension and find a hyperplane that can classify the data in the converted dimension.
  - With the Kernel trick method, instead of converting the data to a higher dimension, it is possible to result in a non-linear classification without making data into a higher dimension by using the Kernel function that transforms into a similar value when performing internal calculation between vectors in higher dimension.
  - The major Kernel functions include polynomial Kernel, Gaussian Kernel, sigmoid kernel, etc.

- Pros
  - Not very sensitive to the outliers.
  - Performance is good.
- Cons
  - Training is relatively slow. Performs poorly for large data.
  - The kernel and the hyperparameter set should be carefully optimized.
  - Not much insight can be gained.

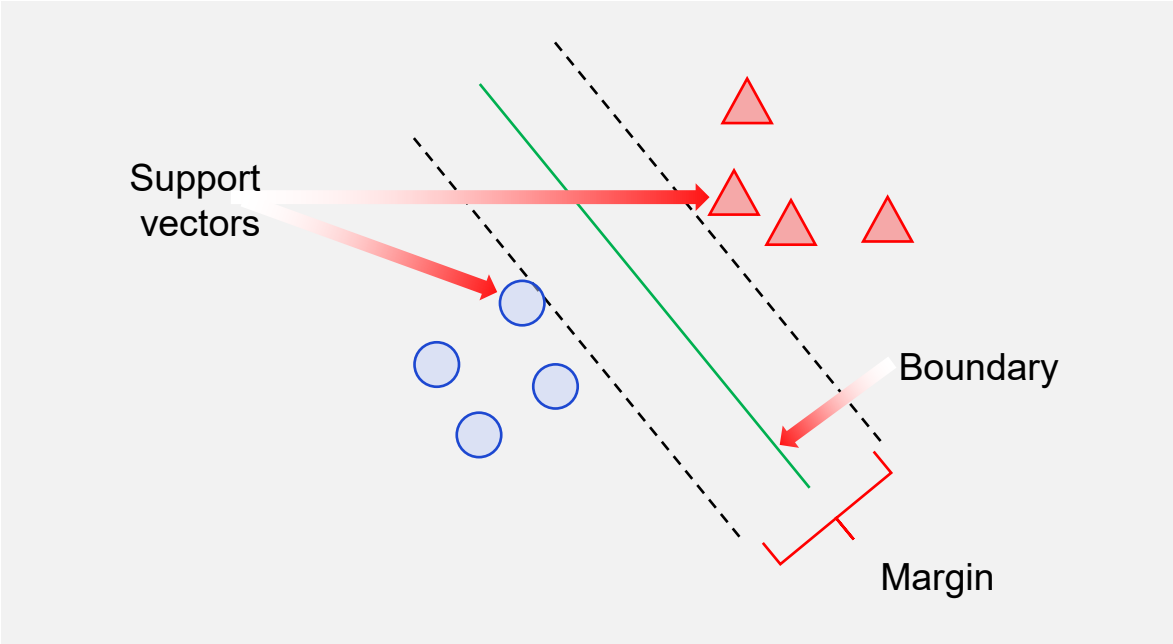
- SVM classification algorithm



- SVM classification algorithm



- SVM classification algorithm



- Hyperplane
  - For a k dimensional configurational space, the hyperplane has the dimension $k-1$.

    Ex For a two dimensional space, a hyperplane is a bisecting line that can be parametrized as

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

    The two dimensional space is subdivided into two:

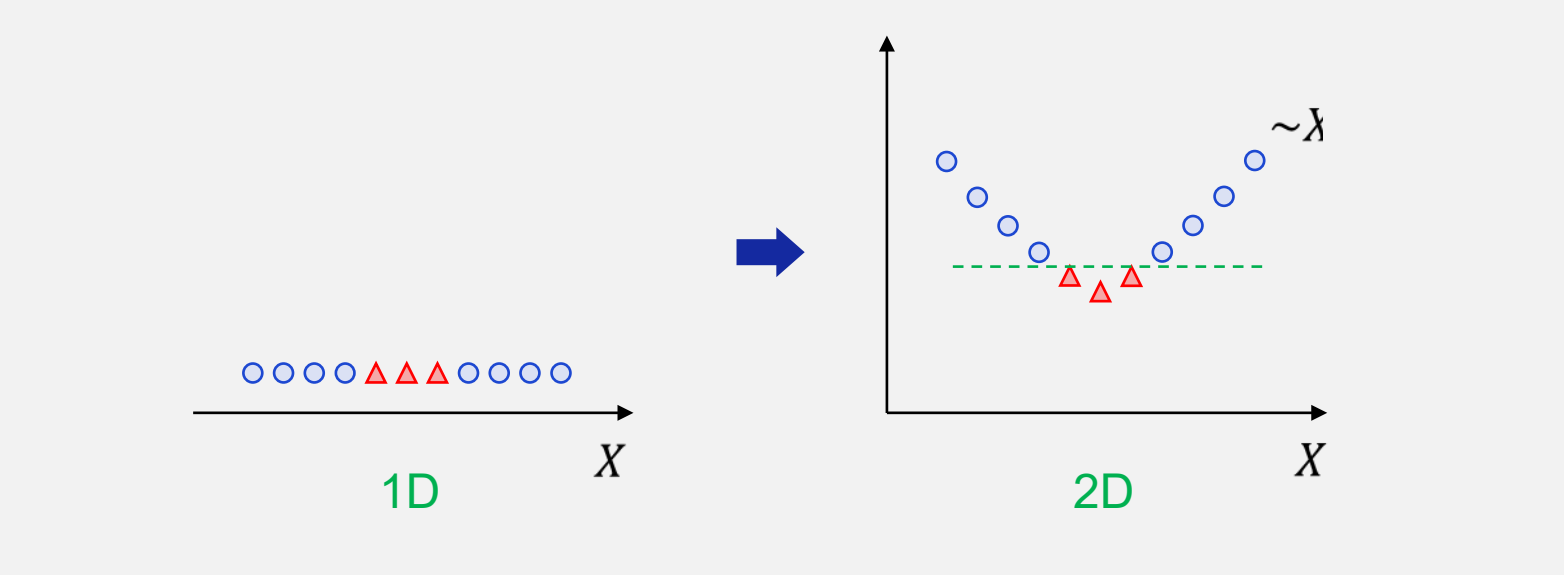
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$$
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 < 0$$

    An observation would belong to either one of them ⇨ binary classification!

- Margin optimization: (for the binary y)
  - The boundary margin is maximized.
  - If a clear margin is not possible to establish, error is allowed within a limit.
  - Target is to maximize M by optimizing the parameters $\beta_0, \beta_1, \ldots, \beta_k$ subject to the constraints:
    - **Constraint 1:** $\sum_{j=1}^{k} {\beta_j}^2 = 1$
    - **Constraint 2:** $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik}) \geq M(1 - \varepsilon_i), \quad i = 1,2,\ldots,n$
      y_i is either -1 or +1.
    - **Constraint 3:** $\varepsilon_i \geq 0$ and $\sum_{i=1}^{k} \varepsilon_i \leq C$
      ⇦ C is a hyperparameter related to the miss-classification errors.

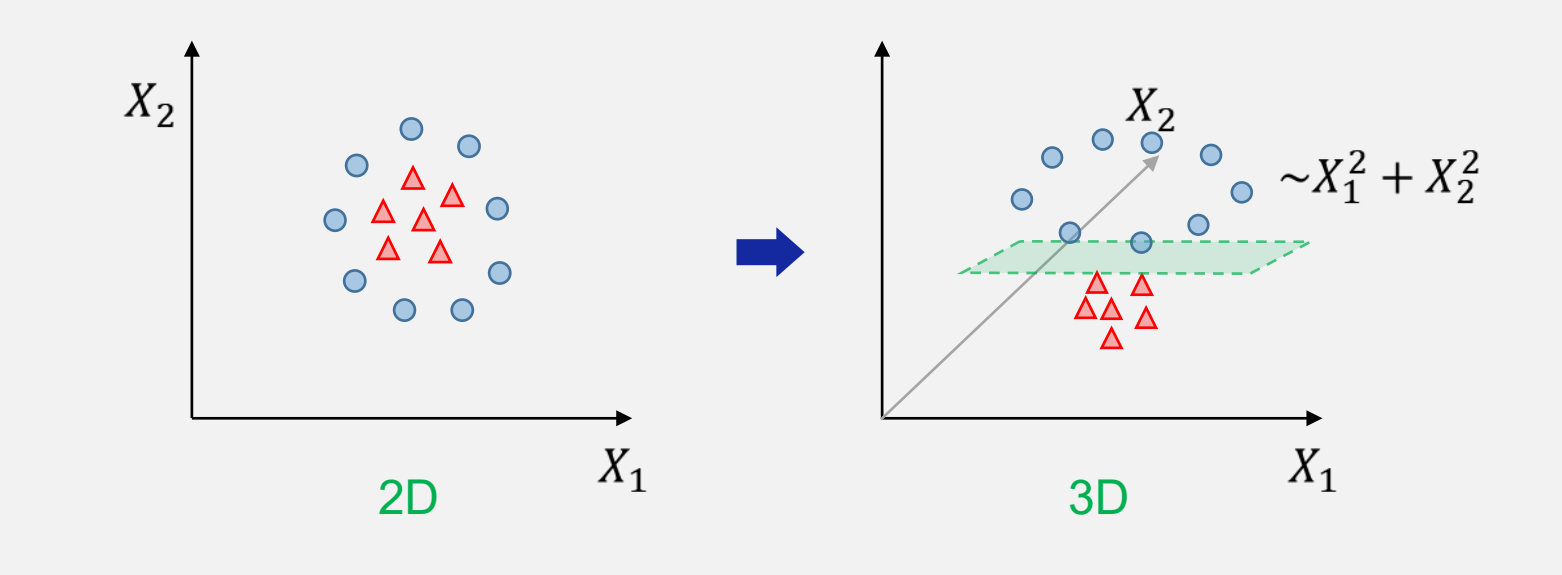
- Kernel
  - Mapping to a higher dimension using the “kernel” functions.
  - Kernel functions introduce an effective non-linear classification boundary.

Ex Polynomial kernel.



- Kernel
  - Mapping to a higher dimension using the “kernel” functions.
  - Kernel functions introduce an effective non-linear classification boundary.

Ex Polynomial kernel.



- Kernel
  - Effective mapping to a higher dimension by giving the inner product of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.
    - Linear: $K(\mathrm{x}_1, \mathrm{x}_2) = \mathrm{x}_1^{\mathrm{t}}\mathrm{x}_2$
    - Polynomial: $K(\mathrm{x}_1, \mathrm{x}_2) = (\mathrm{x}_1^{\mathrm{t}}\mathrm{x}_2 + \mathrm{c})^{\mathrm{d}}$ where $\mathrm{c} > 0$
    - Sigmoid: $K(\mathrm{x}_1, \mathrm{x}_2) = tanh(\mathrm{a}(\mathrm{x}_1^{\mathrm{t}}\mathrm{x}_2) + \mathrm{b})$ where $\mathrm{a}, \mathrm{b} > 0$
    - Radial function basis (rbf): $K(\mathrm{x}_1, \mathrm{x}_2) = exp(-\gamma\|\mathrm{x}_1 - \mathrm{x}_2\|^2)$ where $\gamma > 0$

Coding Exercise #0505

Follow practice steps on 'ex_0505.ipynb' file.

Unit 5f.

Ensemble Algorithms

- 5f.1. The concept of Ensemble Algorithm and Voting
- 5f.2. Bagging & Random Forest
- 5f.3. Boosting

Ensemble algorithms

- About ensemble algorithms:
  - Strong predictive model based on the weaker learners.
  - **Voting type**:
    - A collection of basic learners that “vote”.
    - An ensemble of different kinds of learners. **Ex** Combine Tree, KNN, SVM, etc.
  - **Bagging type**:
    - A collection of independent weak learners that “vote”.
    - An ensemble of the same kind of weak learners. **Ex** Random Forest.
  - **Boosting type**:
    - A series of weak learners that adaptatively learn and predict. **Ex** AdaBoost, GBM, XGBoost, etc.
    - The series is grown by adding new learners multiplied by the “boosting weights”.

5f.1. The concept of Ensemble Algorithm and Voting

- About ensemble algorithms:
  - Making a more suitable decision making by appropriately combining multiple opinions obtained by different experts
  - Majority Voting – Analysis by using many different classification models with one identical train set
  - Bagging (different train set)
    - bootstrap + aggregating
    - Bootstrap + aggregating
    - Bagging (random sampling with replacement (random bootstrap))

      Ex Random Forest
  - Boosting
    - Retain 50% of incorrectly classified data or use weight for sample selection.

- bootstrap
  - Estimation of unknown statistics
  - Easy and effective estimation method with unknown distribution of the model parameter sample
  - Process of recalculating the statistics and model for each sample through additional sampling with replacement from the current sample
    - No assumption required such that the parameters or sample statistics should be in normal distribution
  - bootstrap sample – Aggregation of observed data (sampling with replacement obtained from the record value and dependent variable)

Contents to be learned in the future

- Resampling – Involves permutation under sampling without replacement
- Bootstrap aggregation – Making a result from aggregating predicted values obtained from different bootstrap samples.

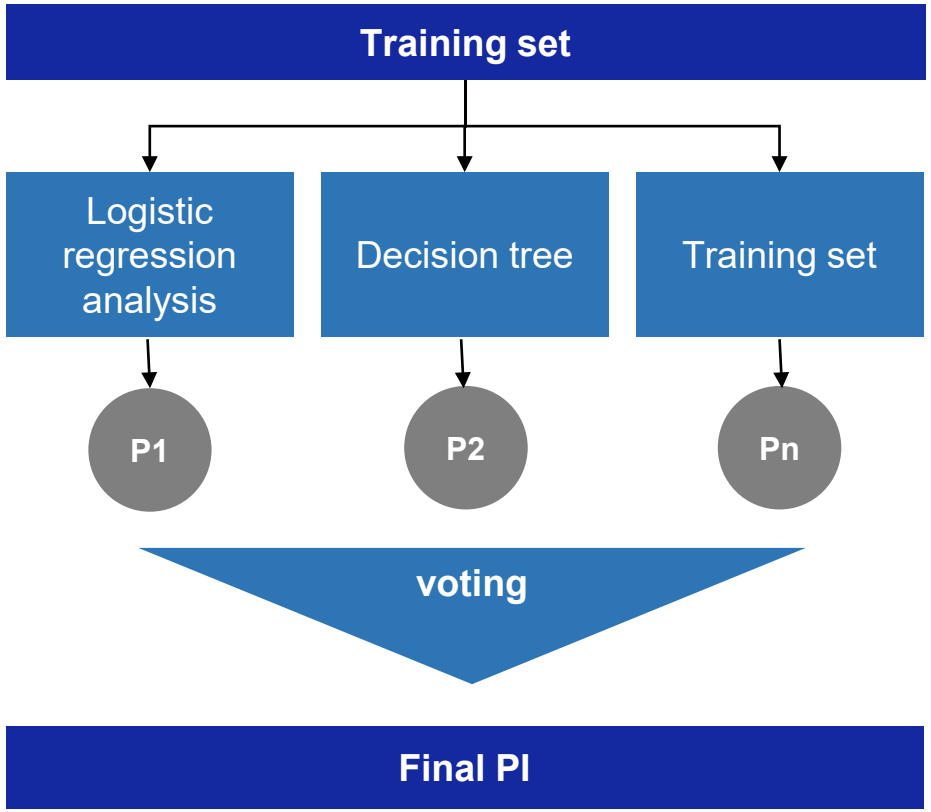
- What is Ensemble Learning?
  - According to Wikipedia, the term ‘ensemble’ is defined as follows.
    - In statistical mechanics, ensemble of a system refers to the collection of the equivalent systems.
  - In other words, it is an assemble of similar groups.
  - Instead of expecting performance results from a single model, the purpose of ensemble learning is to draw a better result by using collective intelligence of different models such as averaging out many different single models or making a decision based on majority vote, etc.
  - There are many different ensemble methods to use collective intelligence.
    - Voting –Drawing results through voting
    - Bagging – Bootstrap aggregating (duplicated creation of various samples)
    - Boosting – Weighting by supplementing previous errors
    - Stacking –A meta model based on different models
  - There could be more other different methods since the ensemble learning applies a certain technique/methodology. However, the four methods listed above are representative ensemble techniques which are provided in the sklearn library.

Voting Ensemble

- Voting ensemble
  - Can be applied to classification and regression.
  - The learners that form an ensemble should be of different kinds for a good performance.
  - Two voting methods for the classifier:
    - Hard: predicted class label is given by the majority rule voting.
    - Soft: predicted class label is given by the **argmax** of the sum of the predicted probabilities.

- Voting
  - As the word itself, voting makes a decision through making votes. Voting is similar with bagging that it uses a voting method, but they are highly differentiated from each other as follows.
    - Voting: Combines different algorithm models.
    - Bagging: Uses different sample combinations within the same algorithm.
  - Voting selects final results through making final voting on results deduced by different algorithms.
  - Voting is classified into hard vote and soft vote.
    - Hard vote: Decides the final value of the result through voting.
    - Soft vote: Draws the final value by adding all of the probability values of getting the final result and then calculating every probability of the final result.

• Voting



- Use a single training set.
- Use different classification models.
- Predictive value
- If the predictive values of each analysis model are different from voting, choose the result with the most values.
  - hard voting
  - soft voting

□ Predict the highest class by averaging out the prediction of individual classifier.

- Voting
  - Hard Vote
    - Taking classification as an example, if the predictive values for classification are 1,0,0,1,1, since 1 has three votes and 0 has 2 votes, 1 becomes the final predictive value in hard voting method.
  - Soft Vote
    - Soft vote method calculates the average value of each probability and then determines the one with the highest probability.
    - If the probability of getting class 0 is (0.4, 0.9, 0.9, 0.4, 0.4) and the probability of getting class 1 is (0.6, 0.1, 0.1, 0.6, 0.6), the final probability of getting class 0 is (0.4+0.9+0.9+0.4+0.4) / 5 = 0.44, and the final probability of getting class 1 is (0.6+0.1+0.1+0.6+0.6) / 5 = 0.4. So, the selected final value is different from the result of the hard vote above.
  - In general, using the soft vote method is considered more reasonable than the hard vote method in competitions, because the soft vote method provides a much better actual performance result.

- Voting ensemble in Scikit-Learn:
  - To import the voting ensemble as class:

```
from sklearn.ensemble import VotingClassifier          # For classification
from sklearn.ensemble import VotingRegressor           # For regression
```

  - To instantiate an object that implements voting ensemble:

Ex
```
myKNN = KNeighborsClassifier(n_neighbors = 3)
myLL = LogisticRegression()
myVotingEnsemble=VotingClassifier(estimators=[('lr',myLL),('knn',myKNN)],voting='hard')
```

- Voting ensemble in Scikit-Learn:
  - We can train and predict just like any other estimator:

    Ex myVotingEnsemble.fit(X_train, Y_train)

    myVotingEnsemble.predict(X_test)

- Scikit-Learn VotingClassifier/Regressor Hyperparameters:

Hyperparameter	Explanation
estimators	The list of basic learner objects.
voting	Either ‘soft’ or ‘hard’ (for classifier only).

  - More information can be found at:

    Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

    Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html

Unit 5f.

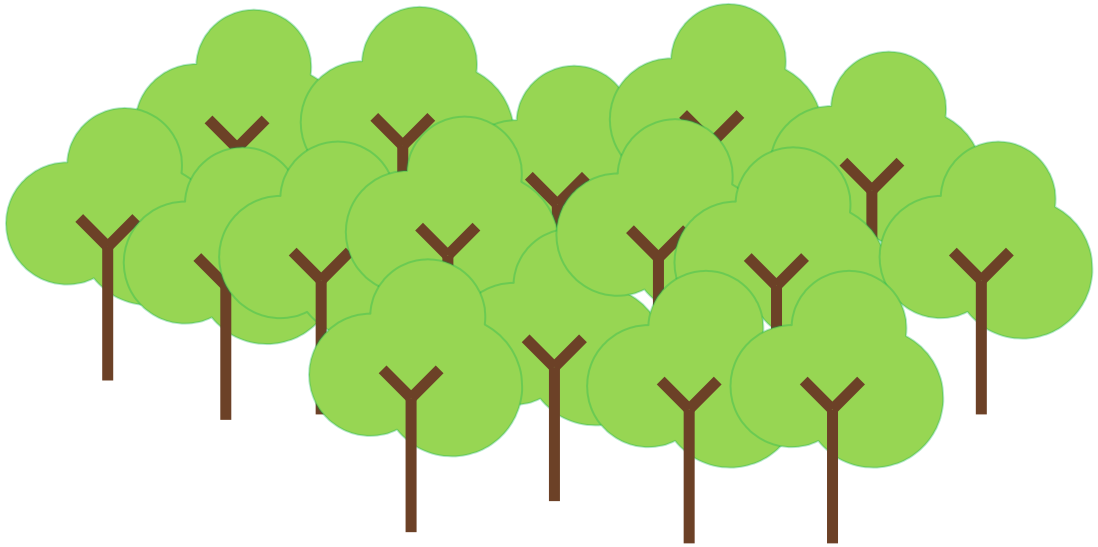
Ensemble Algorithms

- 5f.1. The concept of Ensemble Algorithm and Voting
- 5f.2. Bagging & Random Forest
- 5f.3. Boosting

Bagging Ensemble: Random Forest

- About Random Forest
  - An ensemble algorithm based on trees.
  - Can be applied to classification and regression.
- Pros
  - Powerful.
  - Few assumptions.
  - Little or no concern about the overfitting problem.
- Cons
  - Training is time consuming.

- Random Forest algorithm:



  - Randomly chosen trees form a forest.
  - Prediction is decided by the majority vote.

- Random Forest algorithm:

1) Make trees with the randomly selected variables and observations.

2) Keep only those that have the lowest Gini impurity (or entropy).

3) Repeat from the step 1) for a given number of times.

4) Using the trees gathered during the training step, we can make predictions by majority vote.

- Scikit-Learn RandomForestClassifier/Regressor Hyperparameters:

Hyperparameter	Explanation
n_estimators	The number of trees in the forest.
max_depth	The maximum depth of a tree.
min_samples_leaf	The minimum number of sample points required to be at a leaf node.
min_samples_split	The minimum number of sample points required to split an internal node.
max_features	The number of features to consider when looking for the best split.

  - Except for “n_estimators” the rest are analogous to those of DecisionTreeClassifier/Regressor.
  - More information can be found at:

    Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

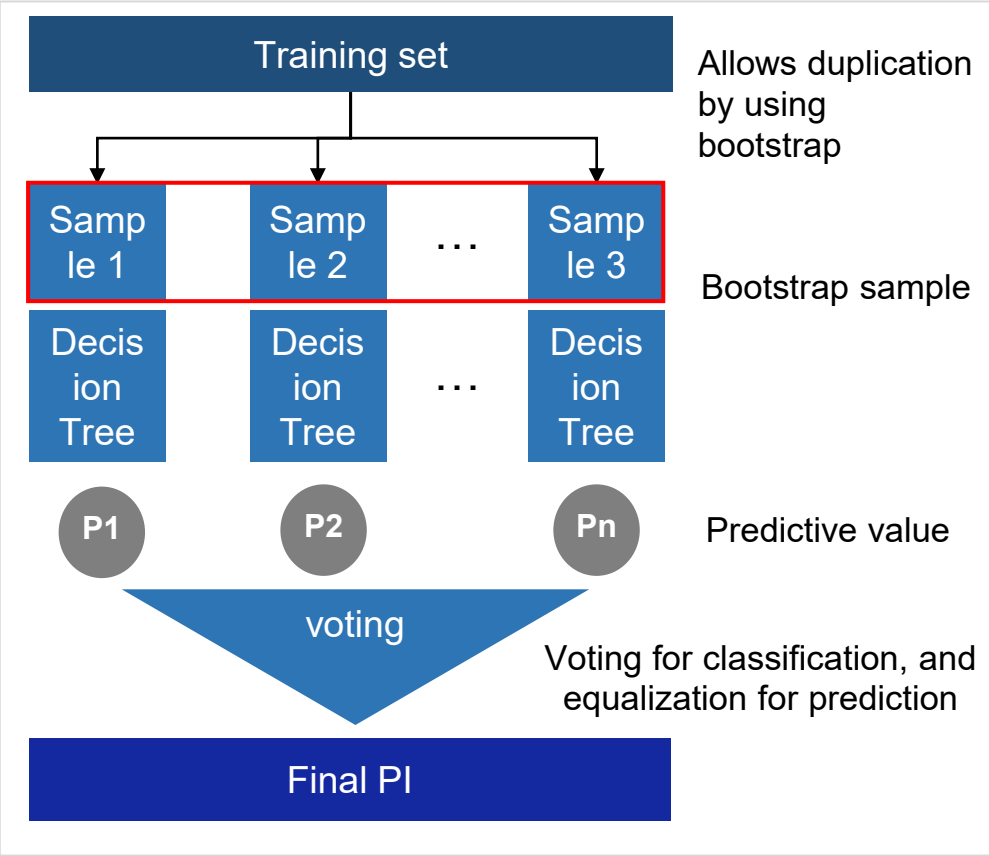
    Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

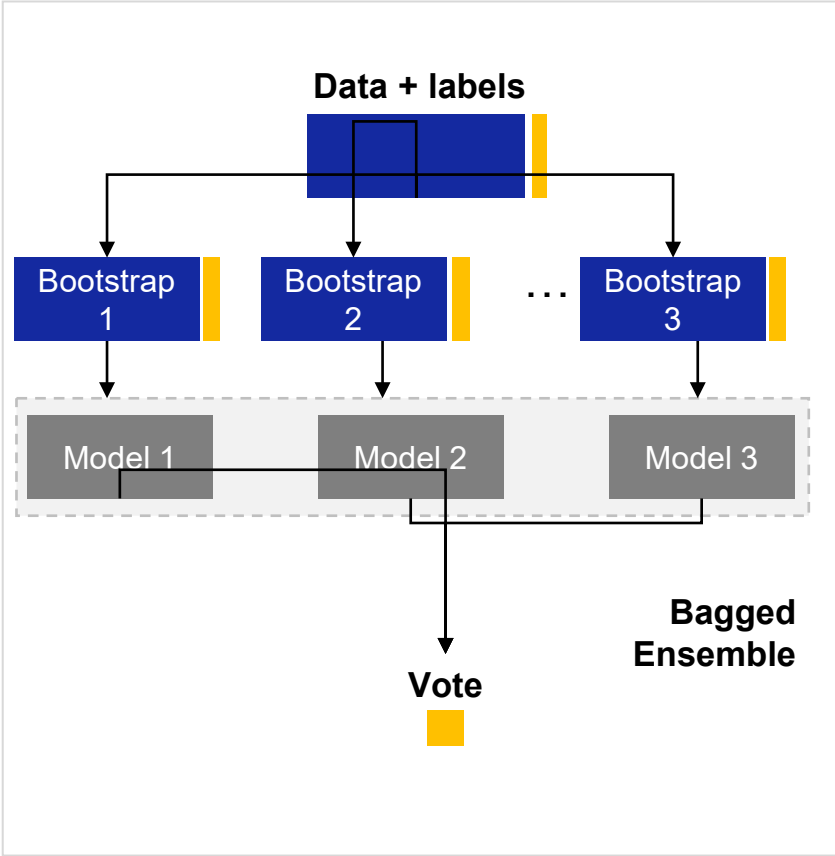
- Bagging
  - Bagging-based ensemble method

    Ex Random Forest algorithm
    - Easy to use since it is well constructed in the Sklearn library.
    - Relatively quick performance speed
    - High performance

  - Basically, ensemble method raises the performance level and because it is easy to use, this method has been widely used. The bagging-based ensemble method is commonly found in the high ranked solutions in Kaggle.

• Bagging


Training set
Allows duplication by using bootstrap
Samp le 1
Samp le 2
…
Samp le 3
Bootstrap sample
Decis ion Tree
Decis ion Tree
…
Decis ion Tree
P1
P2
Pn
Predictive value
voting
Voting for classification, and equalization for prediction
Final PI


Data + labels
Bootstrap 1
Bootstrap 2
…
Bootstrap 3
Model 1
Model 2
Model 3
Bagged Ensemble
Vote

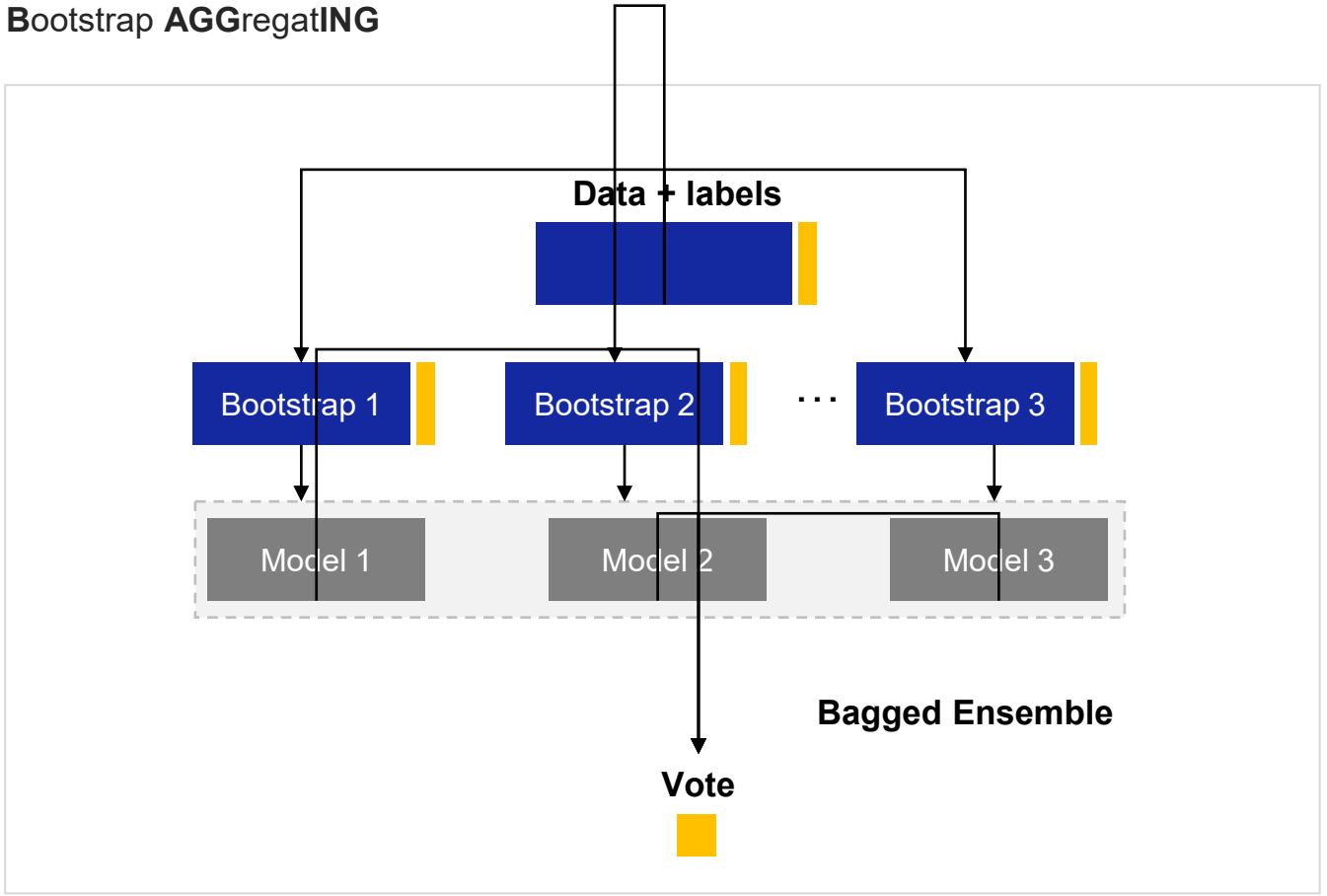
- Bagging
  - Bagging is an abbreviation of Bootstrap Aggregating.
  - Bootstrap = Sample
  - Aggregating = Adding up
  - Bootstrap refers to a method that allows overlapping of different data sets for sampling and splitting.

- Bagging

Ex Random Forest is a typical bagging method algorithm.

- It creates multiple decision trees and performs sampling of different data sets, while allowing overlapped data sets.
- If the data set consists of [1, 2, 3, 4, 5],
  - Group 1 = [1, 2, 3]
  - Group 2 = [1, 3, 4]
  - Group 3 = [2, 3, 5]
- This is the bootstrap method, and in the classification problem, voting is done to each tree that is trained with different sampling for the final prediction result.
- In regression problems, the average of each obtained value is calculated.

- Bagging
  - “**Bagging**” : **B**ootstrap **AGG**regat**ING**



- Differences Between Bagging and Voting
  - The greatest difference between the bagging and voting methods is whether to use multiple single algorithms or apply various algorithms to the same sample data set.
  - In general, the bragging method is to train a single algorithm with different sampling data sets and perform voting. It has relatively better usability than the voting method because it uses a single algorithm. Thus, what is important is the hyperparameter of the single algorithm.
  - Taking the Random Forest algorithm as an example again, it is suitable to obtain the baseline score as it requires simple hyperparameter setting such as how may trees to use (n_estimators), maximum depth (max_depth), minimum number of samples for splitting (min_samples_leaf), etc.

- Advantages of the Bagging Ensemble
  - When using the bagging method, it can reduce variance compared to making a prediction with a single model. There are three major train errors of a model, which are variance, noise, and bias. (Of course, more significant factors include overfitting/underfitting, and many different kinds of preprocessing issues, but assume that they are already being set.)
  - The ensemble method reduces variance, thus enhancing the performance of the final result.

- sklearn.ensemble.BaggingClassifer/BaggingRegressor
  - The sklearn library package provides the wrapper class called BaggingClassifier/BaggingRegressor.
  - When designating the base algorithm to the bese_estimator parameter, the BaggingClassifier/BaggingRegressor performs bagging ensemble.

Unit 8.

Ensemble Algorithms

- 5f.1. The concept of Ensemble Algorithm and Voting
- 5f.2. Bagging & Random Forest
- 5f.3. Boosting

Boosting Ensemble: AdaBoost

- About AdaBoost
  - A sequence of weak learners such as trees.
  - A weighted ensemble of weak learners.
    - One set of weights that increase the importance of the better performing learners.
    - One set of weights that increase the importance of the wrongly classified observations.
  - Similar pros and cons as the Random Forest.

- AdaBoost classification algorithm
  - Let's suppose n observations for the training step: $\boldsymbol{x}_i$ and y_i.
    We also suppose that $y_i \in \{-1,+1\}$. (binary y)
  - We will make a series of weak learners $G_m(\boldsymbol{x})$ with $m = 1,\ldots,M$.
  - The ensemble classifier is made up by a linear combination of these weak learners:

$$G_{ensemble}(\boldsymbol{x}) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(\boldsymbol{x})\right)$$

where α_m are the "boosting weights" that need to be calculated.

  - Heavier weight is given to a better performing learner.

- AdaBoost classification algorithm

1) For the first step (m=1), equal weight is assigned to the observations: $w_i^{(1)}=\frac{1}{n}$

2) For the boost sequence m=1,…, M:

a) Train the learner $G_m(\boldsymbol{x})$ using observations weighted by $w_i^{(m)}$.

b) Calculate the error ratio: $$\varepsilon_m = \frac{\sum_{i=1}^{n} w_i^{(m)} I(y_i \neq G_m(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i^{(m)}}$$

⇨ $I(y_i \neq G_m(\boldsymbol{x}_i))$ gives 1 for an incorrect prediction, else 0.

⇨ $0 \leq \varepsilon_m \leq 1$

- AdaBoost classification algorithm

3) For the boost sequence m=1,…, M:

c) Calculate the boosting weight: α_m $\alpha_m = \frac{1}{2} log\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$

⇨ As $\varepsilon_m \to 0$, α_m is a large positive number. The learner is given more importance!

⇨ As $\varepsilon_m \cong 0.5$, $\alpha_m \cong 0$.

⇨ As $\varepsilon_m \to 1$, α_m is a large negative number.

d) For the next step, the weights of the **wrongly** predicted observation are rescaled by a factor e^{a}_{m}.

This can be compactly expressed a: $w_i^{(m+1)} = w_i^{(m)} \times e^{\alpha_m \cdot I\left(y_i \neq G_m(x_i)\right)}$ where $i = 1, \ldots, n$

⇨ In the next sequence step, the wrongly predicted observations receive heavier weight.

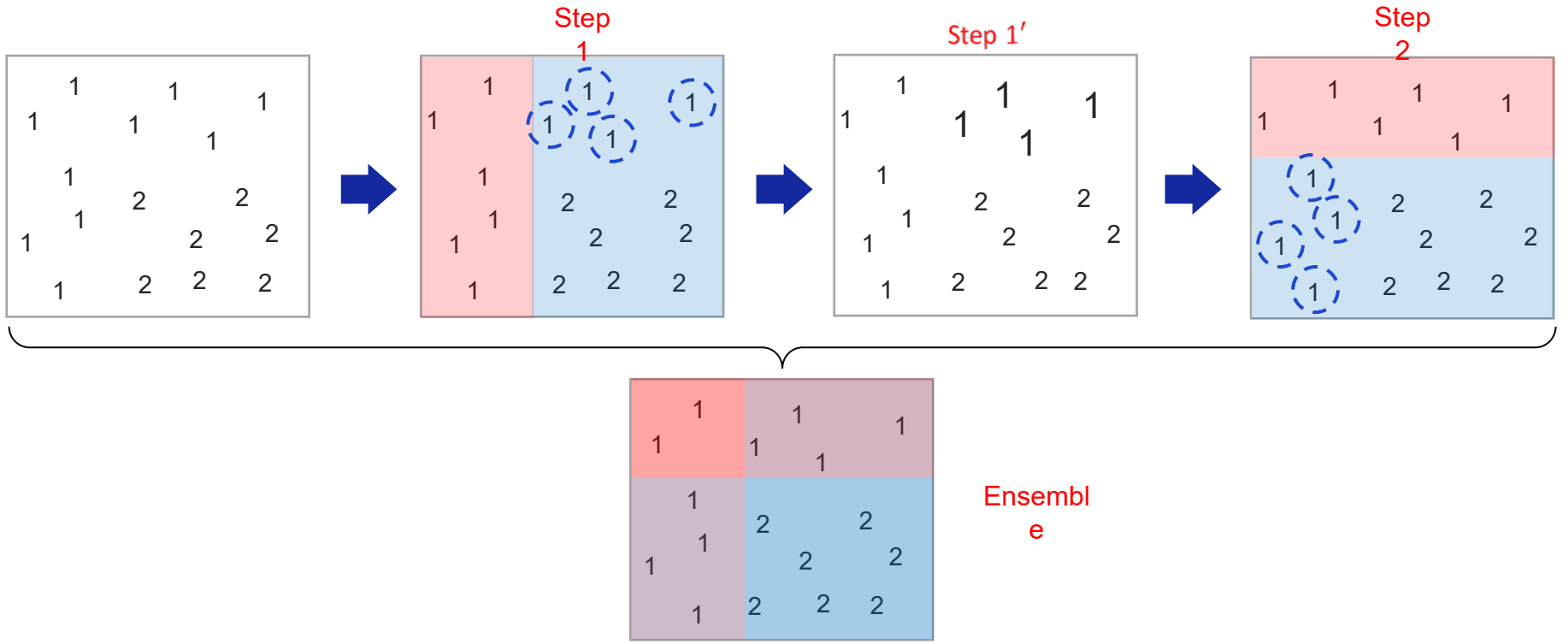
- AdaBoost classification algorithm:

4) The ensemble classifier is made up by a linear combination of weak learners:

$$G_{ensemble}(\boldsymbol{x}) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(\boldsymbol{x})\right)$$

5) For a new testing condition $\boldsymbol{x}'$, we can predict y' by $G\_ensemble\ (\boldsymbol{x}')$.

- AdaBoost classification algorithm:



- Scikit-Learn AdaBoostClassifier/Regressor Hyperparameters:

Hyperparameter	Explanation
base_estimator	The base estimator with which the boosted ensemble is built.
n_estimators	The maximum number of estimators at which boosting is terminated.
learning_rate	The rate by which the contribution of each learner is shrunken.
algorithm	Either ‘SAMME’ or ‘SAMME.R’.

- “base_estimator” is by default None which means DecisionTreeClassifier(max_depth=1).
- More information can be found at:

  Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

  Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html

Boosting Ensemble: GBM & XGBoost

- Gradient Boosting Machine (GBM)
  - GBM is also made up by a sequence of weak learners similar to AdaBoost.
  - The disagreement between the predicted $\hat{y}$ and the true y can be represented by a "loss" function.
  - In the sequence of weak learners, the learner at step m, $G_m(\boldsymbol{x})$ can be obtained by moving the previous step learner $G_{m-1}(\boldsymbol{x})$ towards the direction that decreases the loss as defined above.
  - These updating movements are restricted to a set of base learners.

- Scikit-Learn GradientBoostingClassifier/Regressor Hyperparameters

Hyperparameter	Explanation
loss	The loss function.
n_estimators	The number of boosting steps (weak learners).
learning_rate	The contribution of each weak learner.
subsample	The fraction of data that will be used by individual weak learner.

- Need to be tuned for optimized performance.
- More information can be found at:

  Classifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

  Regressor: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

- Extreme Gradient Boosting (XGBoost)
  - Improves upon GBM in the execution speed.
  - More resistant to the overfitting than GBM.
  - Not included in the Scikit-Learn library. Requires installation of the “xgboost” library.

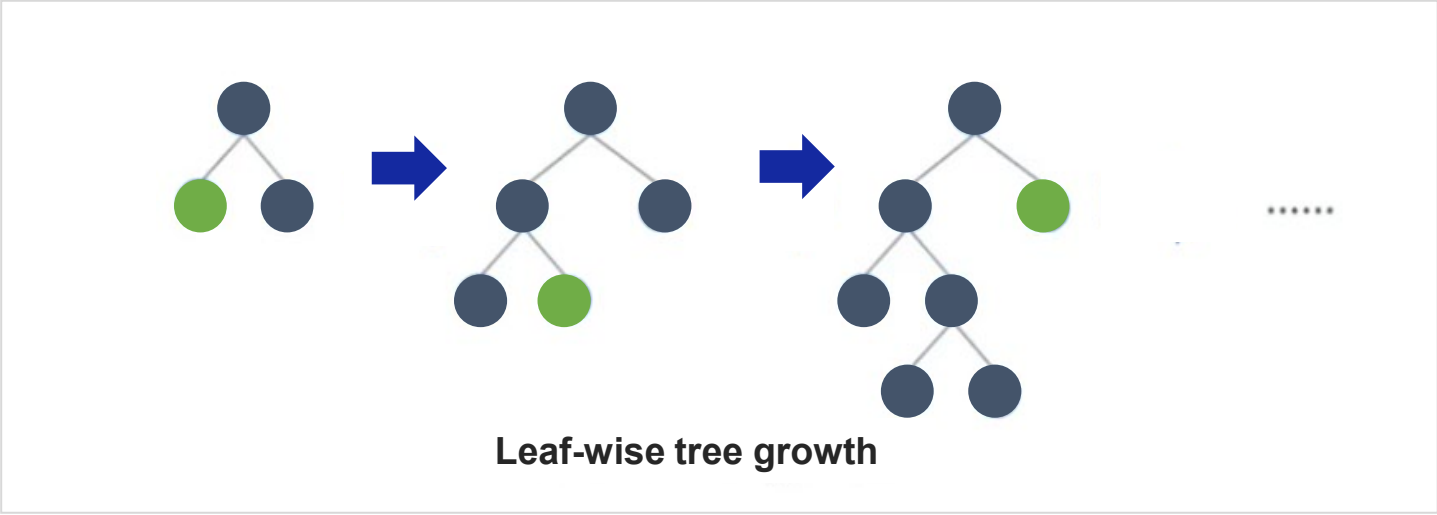
- XGBClassifier/Regressor Hyperparameters

Hyperparameter	Explanation
booster	gbtree or gblinear.
n_estimators	The number of boosting steps (weak learners).
learning_rate	The contribution of each weak learner.
subsample	The fraction of data that will be used by individual weak learner.

- Need to be tuned for optimized performance.
- More information can be found at: https://xgboost.readthedocs.io/en/latest/python/index.htm

- XGBoost
  - XGBoost is a library that implemented the gradient boosting algorithm so that it can be used in the distributed system.
  - It supports both regression and classification problems. This popularly used algorithm features good performance and resource efficiency.
  - Gradient boost is a representative algorithm that is realized by using the boosting method, and the XGBoost is the library that is realized by using this algorithm to support parallel training.
  - It has been recently used a lot due to its high performance and computer resource application rate, and it has become more popular as it was frequently used by top rankers of Kaggle.

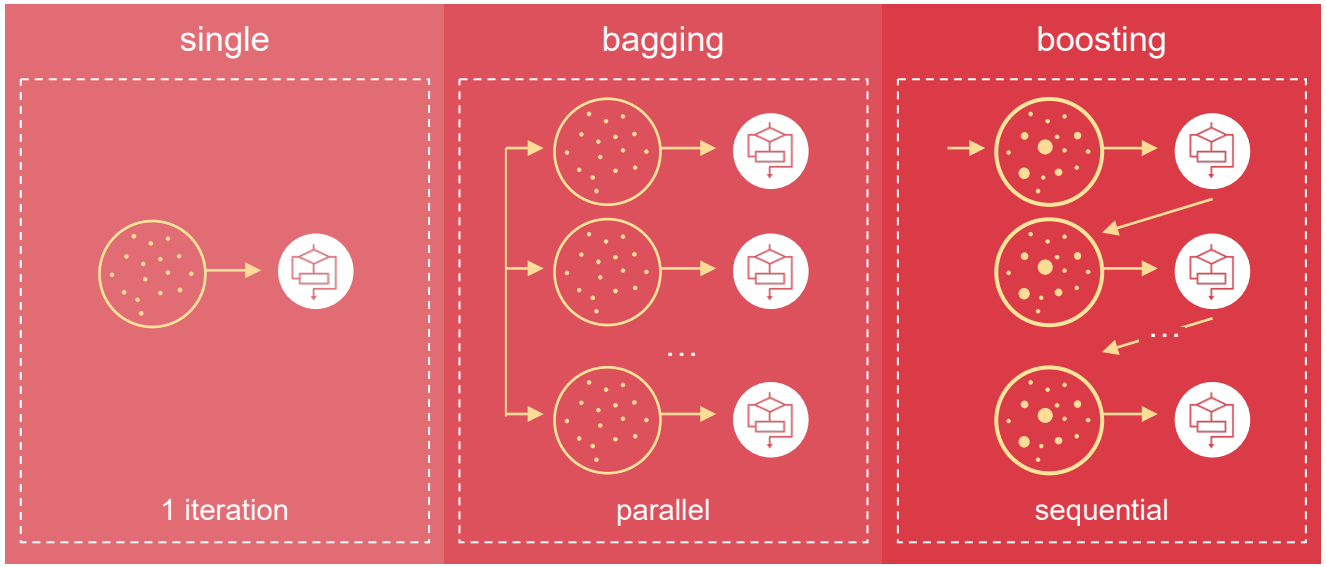
- Light GBM
  - While the tree is vertically expanded with Light GBM, other algorithms expand the tree horizontally. In other words, while the Light GBM is leaf-wise, other algorithms are level-wise.
  - For expansion, the leaf with max. delta loss is selected. When expanding the same leaf, the leaf-wise algorithm can reduce more loss than the level-wise algorithm.
  - The following diagram shows how a boosting algorithm is realized that is different from the Light GBM.



Light GBM operation method

- Boosting
  - The boosting algorithm is also ensemble learning. After learning weak learning machines in order, it supplements errors by adding weight to inaccurately predicted data from the previous learning.
  - The difference from other ensemble methods is that it performs sequential learning and that it supplements errors by adding weight. However, disadvantages include parallel processing is difficult since its sequential property, and because of it it takes longer learning time compared to other ensembles.

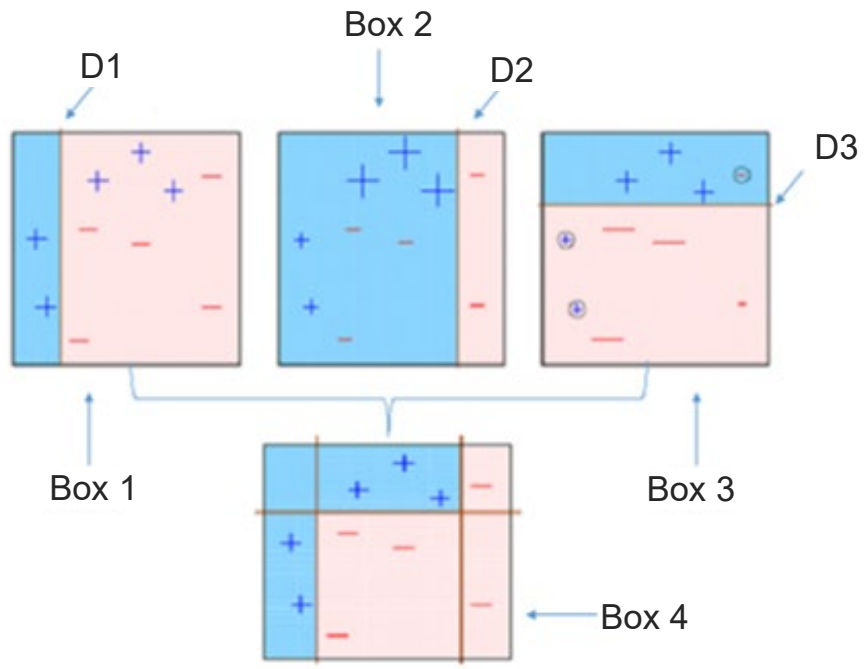
Single estimator, Bagging, Boosting



- Learning a series of predictors by supplementing previous models
  - AdaBoost
    - Train the first distributor (e.g., decision tree) in the training set and make prediction.
    - The weight of the train sample with inaccurately classified algorithm is relatively increased.
    - The second distributor uses updated weight to be trained at the training set and makes prediction again.
    - The weight is updated again, and the same process is repeated.
  - Gradient boosting
    - Similar to AdaBoost, gradient boosting sequentially adds predictors to ensemble to correct the previous errors.
    - However, instead of modifying the sample weight repeatedly as AdaBoost, it trains a new predictor to residual error created by the previous predictor.
    - For regression problems– gradient boosted regression tree, GBRT

- AdaBoost (Adaptive Boosting)
  - As explained previously, the AdaBoost can be briefly explained with making sequential learning of weak learning machines and supplementing errors while applying weight to inaccurately predicted data.
  - AdaBoost (Adaptive Boosting) is one of the most representative boosting algorithms, and as shown in its name, it is an adaptive boosting method.

- Principle of the AdaBoost



- Box 1 is the result of weak learning machine classified as D1 sector. However, because there are data sets expressed in + distributed in the red sector, the error rate is quite high.
- In Box 2, the D2 line is moved to the right to supplement error rate from Box 1. Here, the data sets expressed in – are distributed in the blue sector for better performance, but it is not satisfactory yet.
- In Box 3, the D3 line is horizontally drown on the top. However, the data set expressed in – is incorrectly classified.
- By training the previous Box 1, 2, and 3, Box 4 finds the most ideal combination. Compared to the previous three individual learning machines, it shows much better performance.

- Principle of the AdaBoost
  - How is the weight applied for combination?

    Ex Suppose that the following weight will be applied to the performance of Box 1~3.

    Performance of Box 1: weight = 0.2

    - Performance of Box 2: weight = 0.5
    - Performance of Box 3: weight = 0.6

    It can be expressed as the following formula.

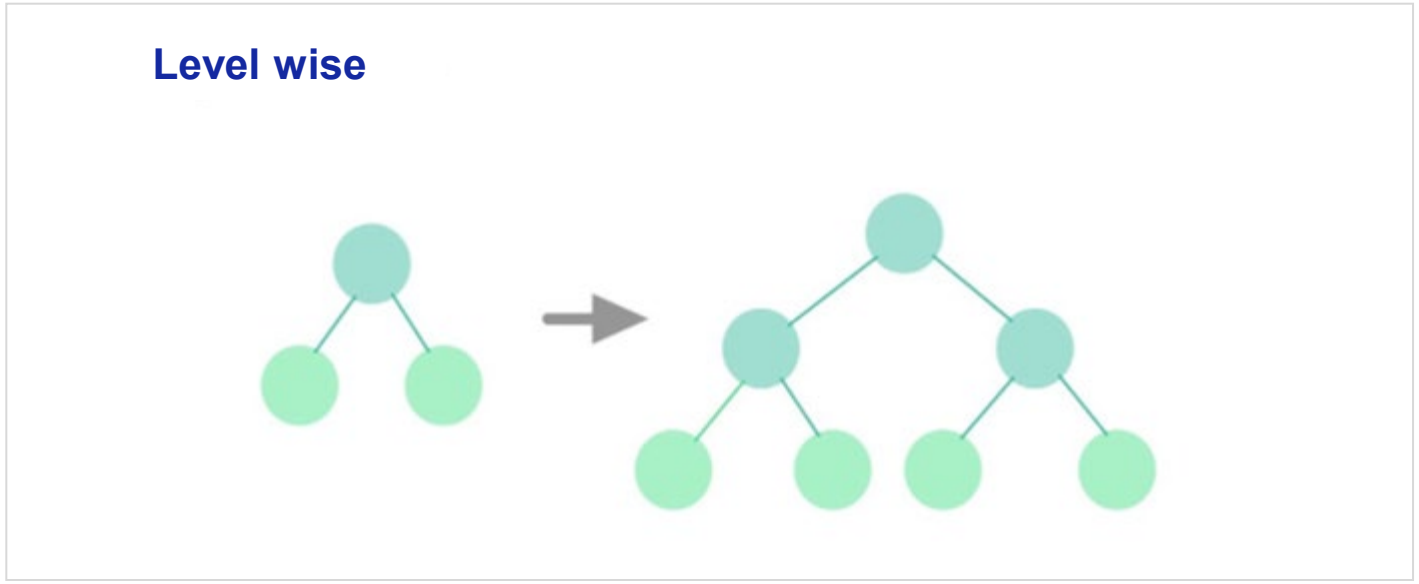
$$0.2 * \text{Box } 1 + 0.5 + \text{Box } 2 + 0.6 * \text{Box } 3 = \text{Box } 4$$

- Gradient Descent
  - The key to the boosting method is to supplement errors from the previous learning.
  - The AdaBoosting and Gradient Descent methods are slightly different in how to supplement errors.
  - Gradient descent uses differentiation to minimize the difference between the predicted value and actual data.
    - Weight
    - Input_data = feature data (input data)
    - Bias
    - Y_actual = actual data value
    - Y_predict = predicted value
    - Loss = error rate
  - Y_predict = weight*input_data+bias
    - The predictive value can be obtained from the above formula, and calculating the difference with actual data will result in the total error rate.
  - Loss = Y_predict – Y_actual
    - (There are many different functional formulas to define error rate including root mean square error, mean absolute error, etc., but the above definition is provided for convenience.)
    - The purpose of gradient descent is to find the weight that makes the loss closest to 0.

- Gradient Boosting Machine (GBM)
  - The boosting algorithm in gradient descent is called gradient boosting machine, which is abbreviated as GBM. It is provided in the sklearn package, and is applicable to both classification and regression problems.
    - GradientBoostingClassifier
    - GradientBoostingRegressor
  - Extremely easy to apply in actual problems.
  - XGBoost and LightGBM are two major ML packages that are mostly used in Kaggle.
  - XGBoost and LightGBM are not provided in the existing sklearn package, but the sklearn wrapper class has allowed easy application for fit& predict similar to the ML class of the existing sklearn package.

- Gradient Boosting Machine (GBM)
  - Together with the XGBoost, LightGBM is the most highlighted boosting algorithm. Although XGBoost has an excellent performance, but it takes too long learning period.
  - Advantages of the LightGBM
    - Short learning time
    - Relatively small memory use
    - Automatic conversion and optimal splitting of categorical features

- Characteristics of the LightGBM
  - LightGBM uses the leaf wise method. The existing tree-based algorithm uses a level wise method, which is characterized by splitting while maintaining a balanced tree as much as possible, thus the tree depth would be minimum. A disadvantage of level wise method is that it takes some time to make a balanced tree.
  - However, the leaf wise method of LightGBM does not balance the tree, but it continuously splits the leaf node that has the max. data loss. This way, the tree depth becomes greater and an asymmetrical tree is created. The repetition of leaf node with the max. data loss minimizes predicted error loss than the splitting of a balanced tree.



- Hyperparameter tuning methods
  - The increased Num_leaves value raises accuracy, but it increases the tree depth and makes the model complex, thus increasing the probability of overfitting.
  - Min_data_in_leaf is a significant parameter for overfitting. It depends on the Num_leaves and the size of training data, but in general, it prevents high value of tree depth when setting a higher value.
  - Max_depth constraints the size of depth. Improves overfitting when combined with two parameters above.

- XGBoost

Parameter	Default value	Description
num_iterations	100	Designates the number of tree for repetitive work. Overfitting occurs if the value is too high.
learning_rate	0.1	The learning rate that is updated when repeatedly conducting boosting steps. The value is designated between 0~1.
max_depth	1	Identical to the max_depth of tree-based algorithms. When entering a value smaller than 0, no restriction is applied to tree depth.
min_data_in_leaf	20	Identical to the min_samples_leaf of decision tree. Used as a parameter to control overfitting.
num_leaves	3	Signifies the max. number of leaf for one tree.
boosting	GBDT	
bagging_fraction	1.0	Designates the ratio for data sampling. Used to control overfitting.
feature_fraction	1.0	Ratio of the random feature for individual tree learning
Lambda_l1	0.0	The value for L1 regulation
Lambda_l2	0.0	The value for L2 regulation

Coding Exercise #0506

Follow practice steps on 'ex_0506.ipynb' file

Coding Exercise #0507

Follow practice steps on 'ex_0507.ipynb' file

Coding Exercise #0508

Follow practice steps on 'ex_0508.ipynb' file

SAMSUNG

Together for Tomorrow!

Enabling People

Education for Future Generations

End of Document