

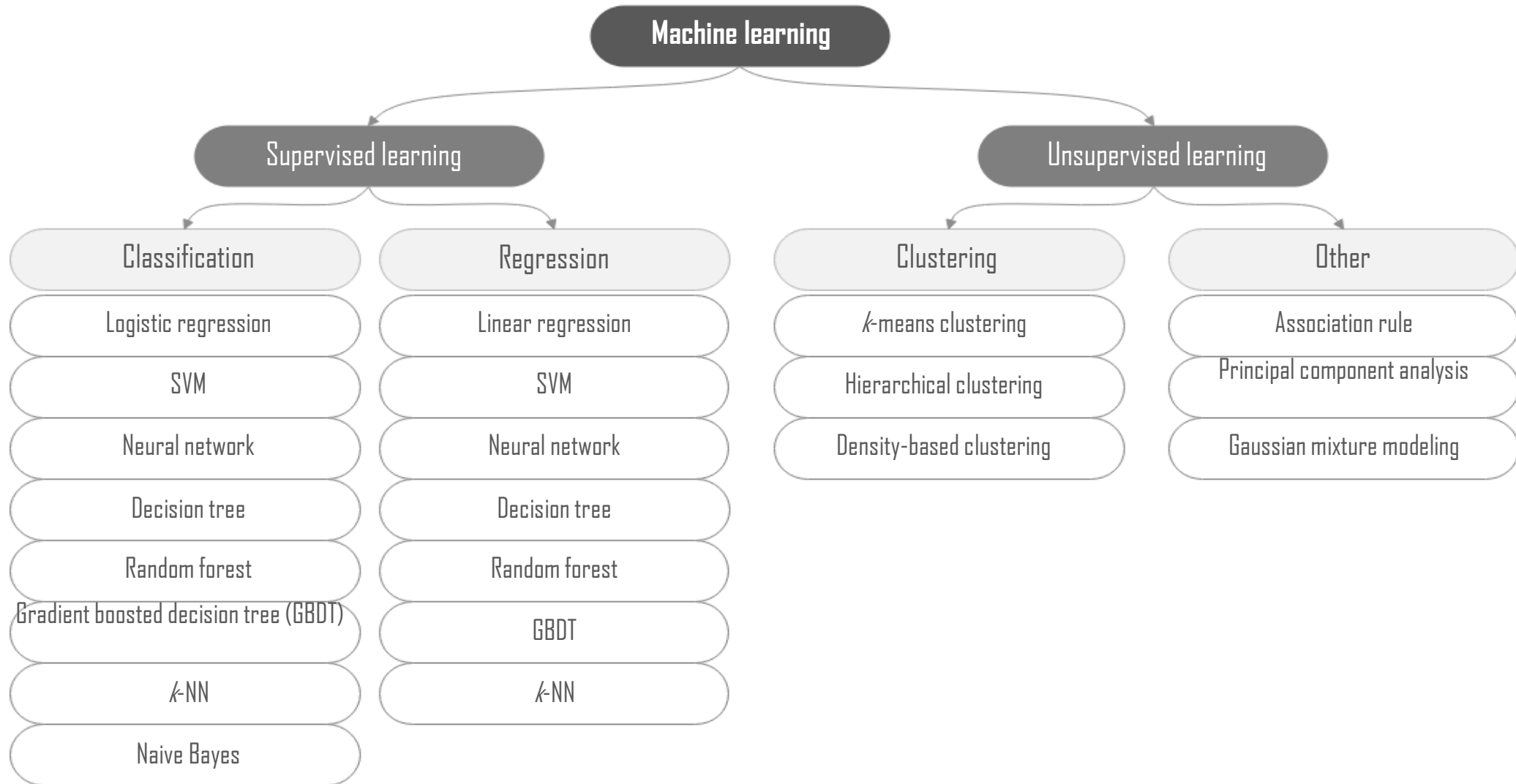
# HCIA-AI Huawei

Eng.Eman Elgallad

## Lecture 4- Supervised Learning

- Supervised Learning
- Regression
  - Linear Regression
- Classification
  - Logistic Regression
  - KNN
  - SVM
  - Decision Tree
  - Ensemble Learning
  - Naïve Bayes

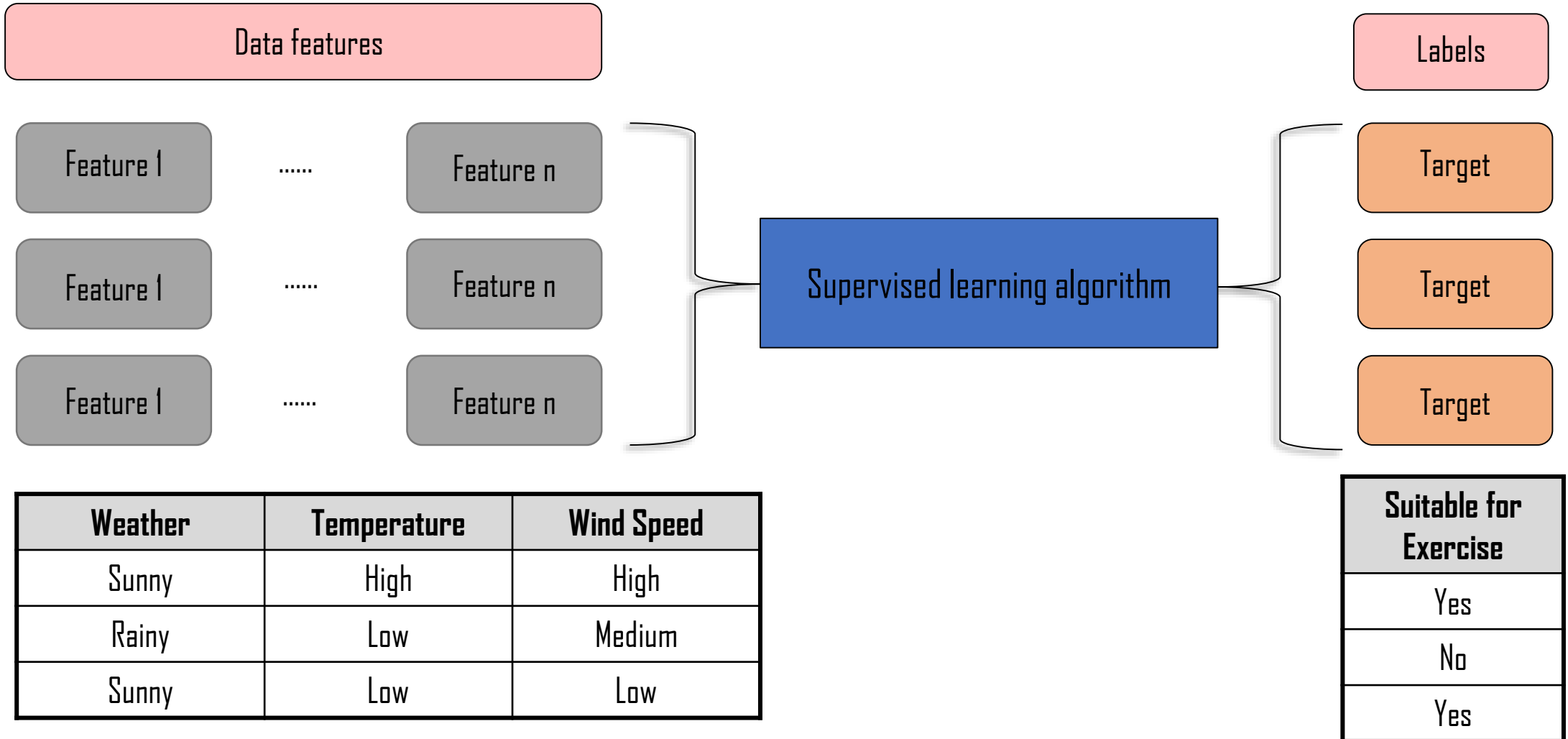
# Machine Learning Algorithm Overview



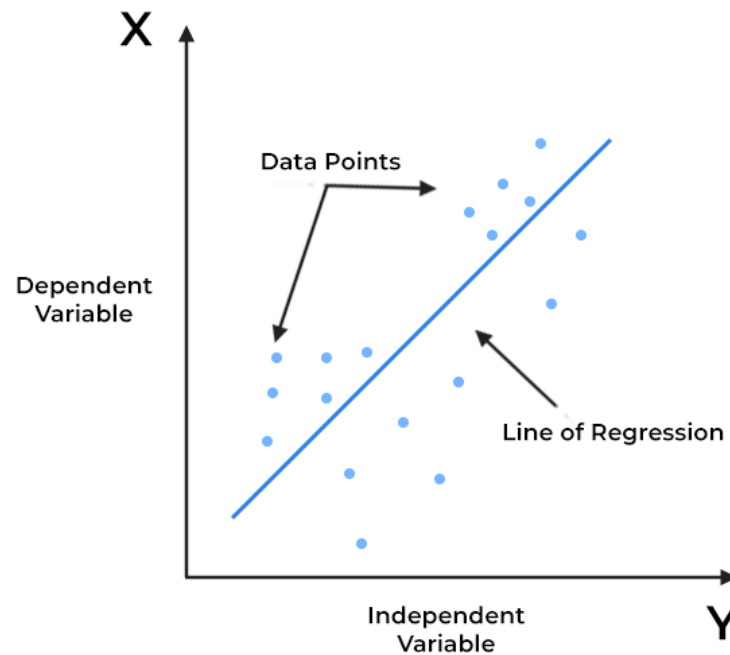
# Types of Machine Learning

- **Supervised learning:** The program takes a known set of samples and trains an optimal model to generate predictions. Then, the trained model maps all inputs to outputs and performs simple judgment on the outputs. In this way, unknown data is classified.
- **Unsupervised learning:** The program builds a model based on unlabeled input data. For example, a clustering model groups objects based on similarities. Unsupervised learning algorithms model the highly similar samples, calculate the similarity between new and existing samples, and classify new samples by similarity.
- **Semi-supervised learning:** The program trains a model through a combination of a small amount of labeled data and a large amount of unlabeled data.
- **Reinforcement learning:** The learning systems learn behavior from the environment to maximize the value of reward (reinforcement) signal function. Reinforcement learning differs from supervised learning of connectionism in that, instead of telling the system the correct action, the environment provides scalar reinforcement signals to evaluate its actions.
- Machine learning evolution is producing new machine learning types, for example, self-supervised learning, contrastive learning, generative learning.

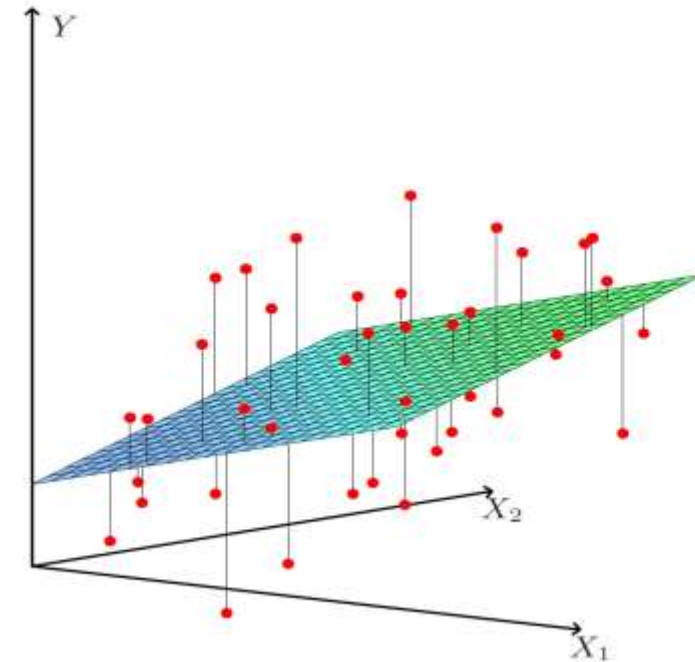
# Supervised Learning



- Linear regression uses the regression analysis of mathematical statistics to determine the quantitative relationship between two or more variables.
- X-axis = Independent variable
- Y-axis = Output / dependent variable
- Line of regression = Best fit line for a model



Simple linear regression



Multiple linear regression

# Linear Regression

| Ram Capacity | Cost |
|--------------|------|
| 2 GB         | \$12 |
| 4 GB         | \$16 |
| 8 GB         | \$28 |
| 16 GB        | \$62 |



Mathematically these slant lines follow the following equation,

$$Y = m * X + b$$

Where X = dependent variable (target)

Y = independent variable

m = slope of the line (slope is defined as the 'rise' over the 'run')

# Linear Regression

$$h_w(x) = w^T x + b$$

ML experts have a different notation to the above slope-line equation,

$y$  = output variable. Variable  $y$  represents the continuous value that the model tries to predict.

$x$  = input variable. In machine learning,  $x$  is the feature, while it is termed the independent variable in statistics. Variable  $x$  represents the input information provided to the model at any given time.

$b$  =  $y$ -axis intercept (or the bias term).

$W$  = the regression coefficient or scale factor. In classical statistics,  $w$  is the equivalent of the slope of the best-fit straight line of the linear regression model = weights (in general).

$$y = w^T x + b + \varepsilon$$

- The error  $\varepsilon$  is affected by many independent factors. Linear regression assumes that the error  $\varepsilon$  follows normal distribution.

- Mean absolute error (MAE). An MAE value closer to 0 indicates the model fits the training data better.

$$MAE = \frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$

- Mean squared error (MSE).

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

- The value range of  $R^2$  is  $[0,1]$ . A larger value indicates that the model fits the training data better.  $TSS$  indicates the difference between samples, and  $RSS$  indicates the difference between the predicted values and sample values.

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{\sum_{i=1}^m (y_i - \bar{y}_i)^2}$$

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

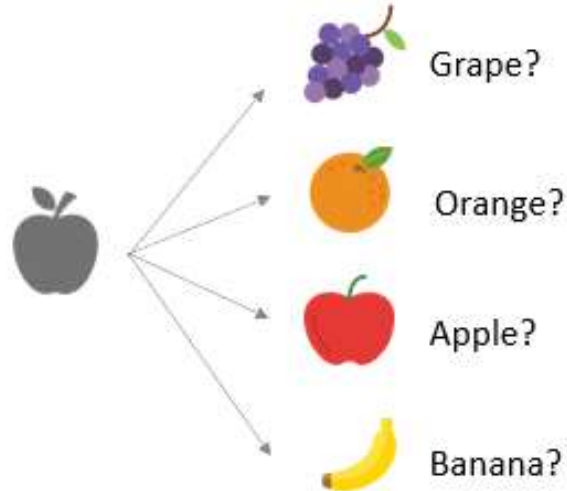
There are four main classification tasks in Machine learning:

- binary
- multi-class
- multi-label
- imbalanced classifications.

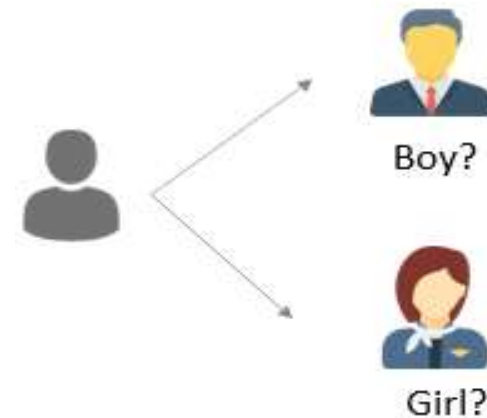
In a **binary classification** task, the goal is to classify the input data into two mutually exclusive categories. For example: true and false; positive and negative; 0 and 1; spam and not spam, etc.

The **multi-class classification**, on the other hand, has at least two mutually exclusive class labels, where the goal is to predict to which class a given input example belongs to.

Multi-class classification problem

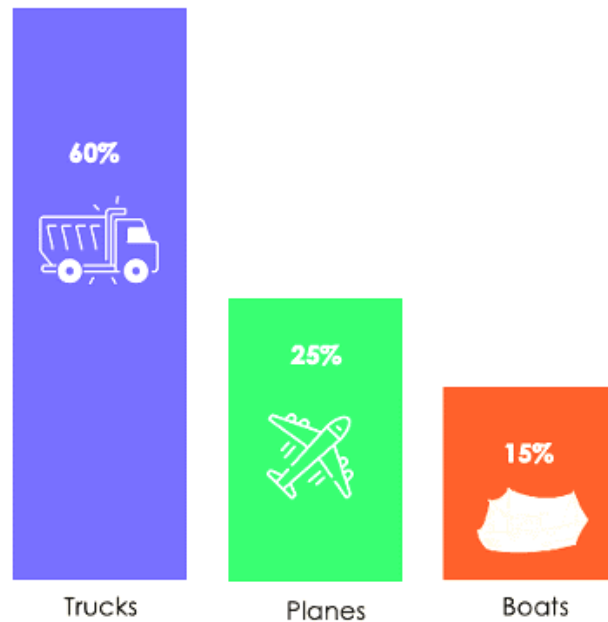


Binary classification problem



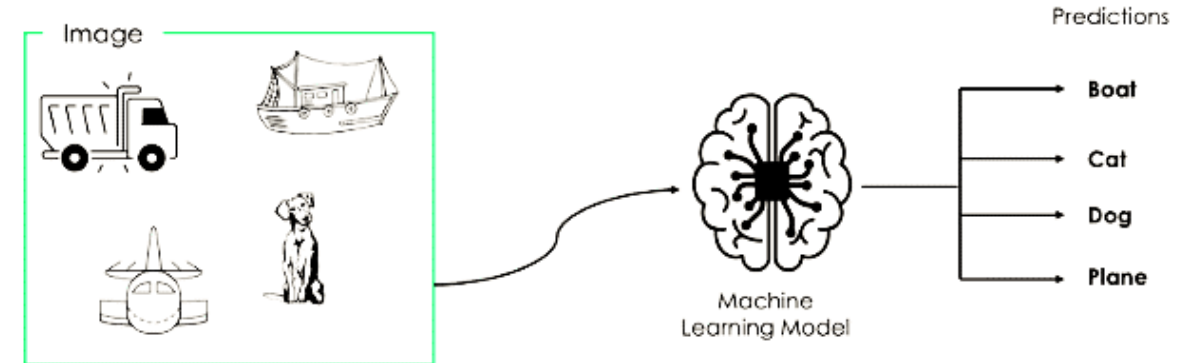
- **Imbalanced Classification**

the number of examples is unevenly distributed in each class. Let's consider the following 3-class classification scenario where the training data contains: 60% of trucks, 25% of planes, and 15% of boats.



- **Multi-Label Classification**

In multi-label classification tasks, we try to predict 0 or more classes for each input example. In this case, there is no mutual exclusion because the input example can have more than one label.



Most of the binary classification algorithms can be also used for multi-class classification. These algorithms include but are not limited to:

Logistic Regression.

Random Forest

Naive Bayes.

K-Nearest Neighbors.

Gradient Boosting .

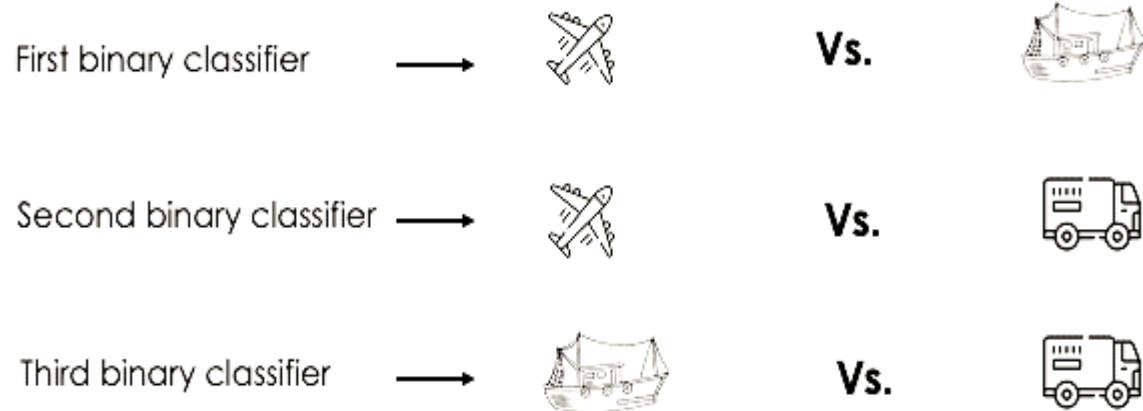
SVM.

*SVM and Logistic Regression do not support multi-class classification by default?*

*→ That's correct. However, we can apply binary transformation approaches such as one-versus-one and one-versus-all to adapt native binary classification algorithms for multi-class classification tasks.*

**One-versus-one:** this strategy trains as many classifiers as there are pairs of labels. If we have a 3-class classification, we will have three pairs of labels, thus three classifiers, as shown below.

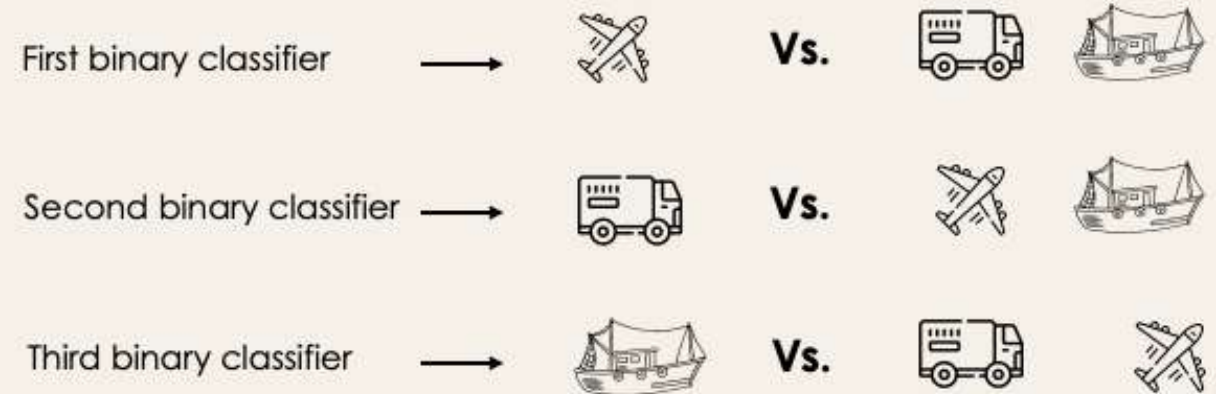
### One-vs-one



**One-versus-rest:** at this stage, we start by considering each label as an independent label and consider the rest combined as only one label. With 3-classes, we will have three classifiers.

In general, for  $N$  labels, we will have  $N$  binary classifiers.

### One-vs-rest



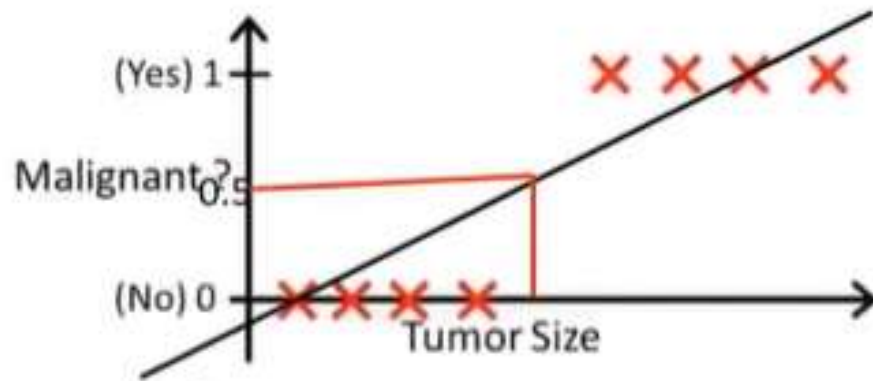
The logistic regression model is based on the logistic function, which transforms any continuous value into a value between 0 and 1.

This allows the model to output the probability of belonging to a certain class. In other words, logistic regression is used to predict the probability of an event occurring, such as whether a customer will buy a product or not, based on a set of independent variables such as age, gender, income, etc.

The logistic regression model estimates the probability of the dependent variable (i.e., the event occurring) based on the independent variables. The model then assigns the observation to the class with the highest probability.

**For example**, if the probability of a customer buying a product is estimated to be 0.8, the model would predict that the customer is likely to buy the product.

Why regression ? Because its equation based on regression rule



Threshold classifier output  $h_{\theta}(x)$  at 0.5:

If  $h_{\theta}(x) \geq 0.5$ , predict “y = 1”

If  $h_{\theta}(x) < 0.5$ , predict “y = 0”

But according to this graph we would have two problems if we worked with Linear Regression:

- The output will be continuous values
- The MSE will be so big between points of each class

Logistic regression works by estimating the probability of a binary dependent variable (y) based on one or more independent variables (x1, x2, ..., xn).

The logistic regression model uses the logistic function, also known as the sigmoid function, to map any continuous value to a value between 0 and 1. The sigmoid function has an S-shaped curve, and it is given by the equation:

$$p = 1 / (1 + e^{(-z)})$$

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

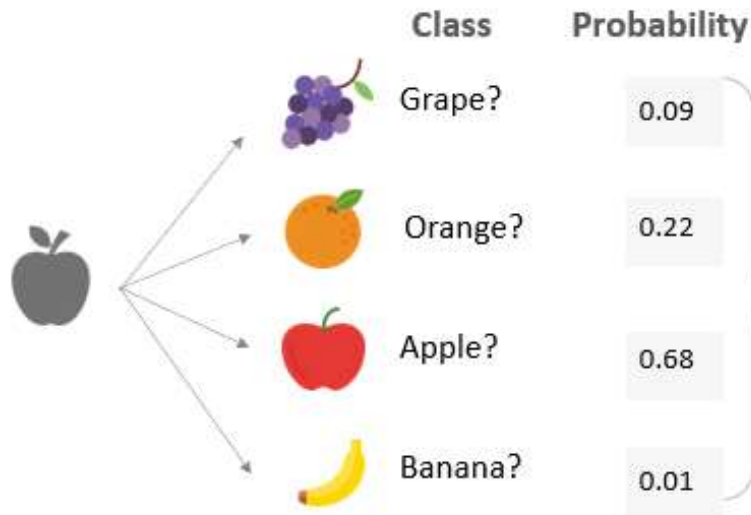
Where:

- p is the probability of the dependent variable taking the value 1 (i.e., success or positive outcome)
- e is the mathematical constant approximately equal to 2.71828
- z is the linear combination of the independent variables, and their coefficients ( $\beta_0, \beta_1, \dots, \beta_n$ ), which are estimated by the model.  $z = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n$ .

SoftMax regression: SoftMax regression, also known as multiclass logistic regression, is a generalization of logistic regression that can handle multiple classes directly.

It assigns a probability value to each class in a multi-class classification problem. The sum of all the probabilities is 1.

■ SoftMax may present the generated classes as follows:



- **Sum of all probabilities:**
  - $0.09 + 0.22 + 0.68 + 0.01 = 1$
- The most possible object in the image: **Apple**

$$p(y = k | x; w) = \frac{e^{w_k^T x}}{\sum_{l=1}^K e^{w_l^T x}}, k = 1, 2, \dots, K$$

# Logistic Regression:

## Strengths of Logistic Regression:

- LR is a simple and easy-to-implement model that does not require extensive computational resources.
- LR is a linear model, which makes it interpretable and allows for easy identification of important predictors.
- LR can handle both continuous and categorical predictors.
- LR can produce probability estimates of the target class, which can be useful for decision-making and ranking.
- LR can handle missing data by excluding observations with missing values from the analysis.

## Weaknesses of Logistic Regression:

- LR assumes that the relationship between the predictors and the target variable is linear. This may not always be the case, and non-linear relationships may be missed.
- LR assumes that the predictors are not highly correlated with each other.
- LR may not perform well when the classes are imbalanced or when the decision boundary is non-linear.
- LR may not capture complex interactions between predictors or higher order relationships

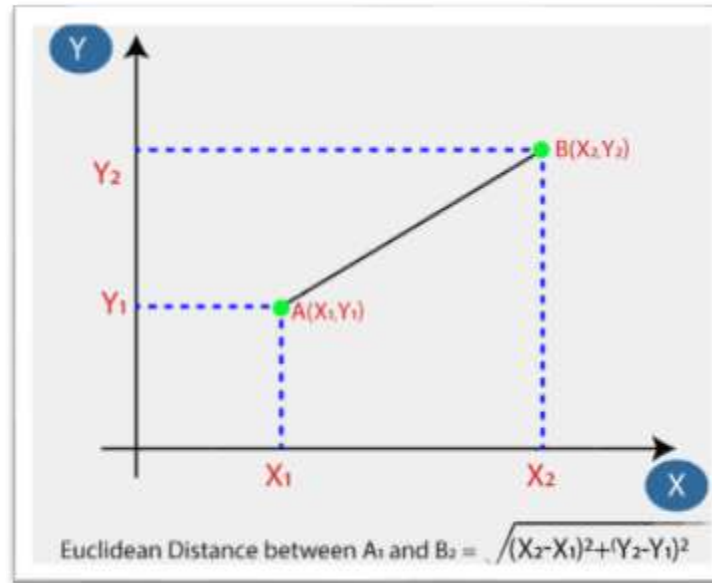
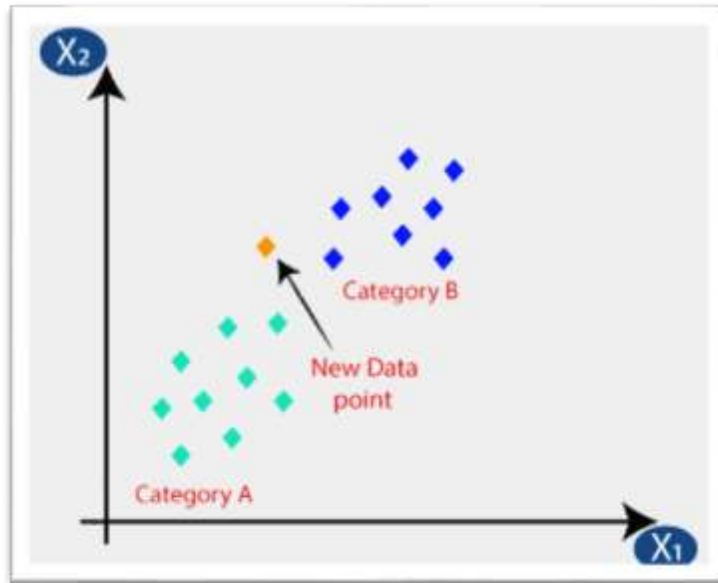
K-Nearest Neighbors is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

It is a non-parametric and lazy learning algorithm, meaning that it does not make any assumptions about the distribution of the data and does not require training prior to making predictions.

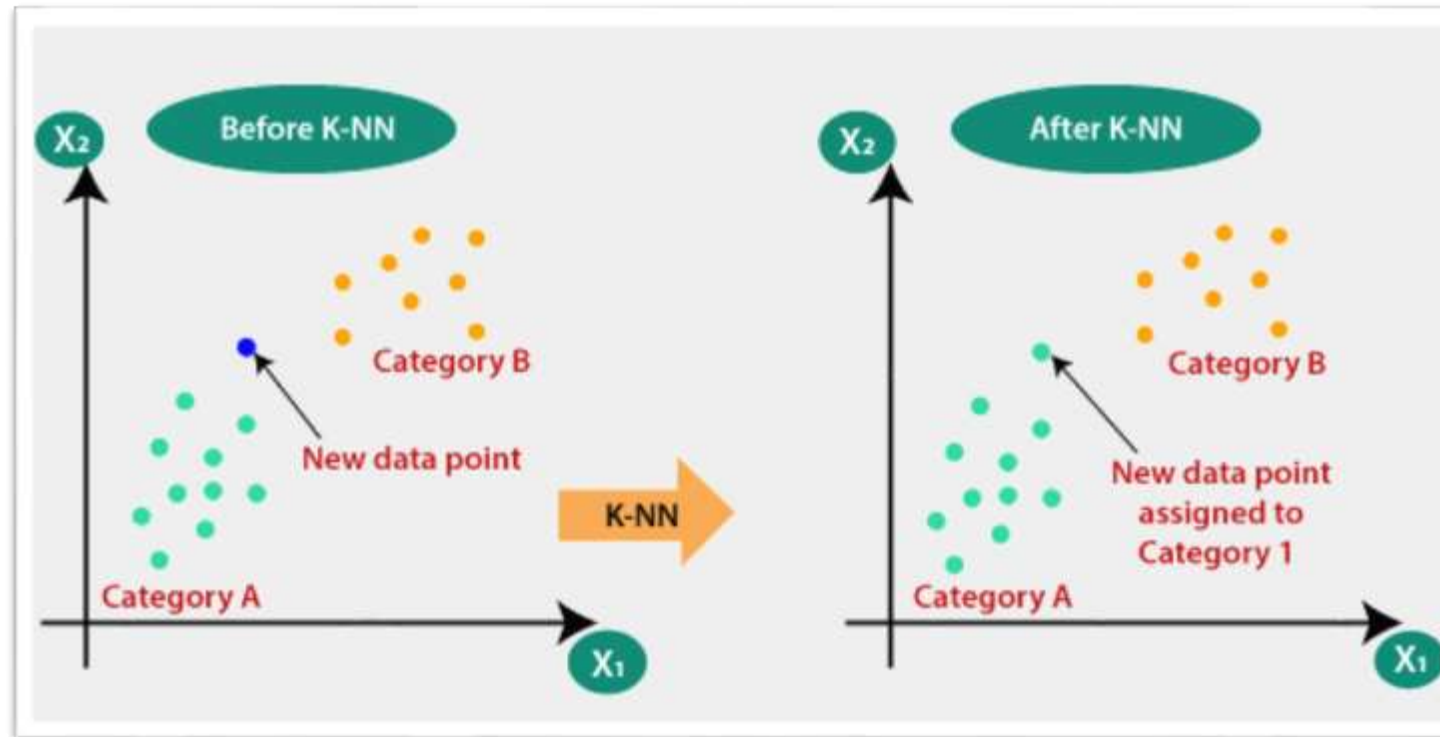
The KNN algorithm works in the following steps:

- Calculate the distance between the new data point and all points in the training set.
- Sort the distances in ascending order and select the K nearest neighbors.
- Assign a class label or numerical value to the new data point based on the most common class label or the average of the values of the K nearest neighbors.

## How does KNN work?



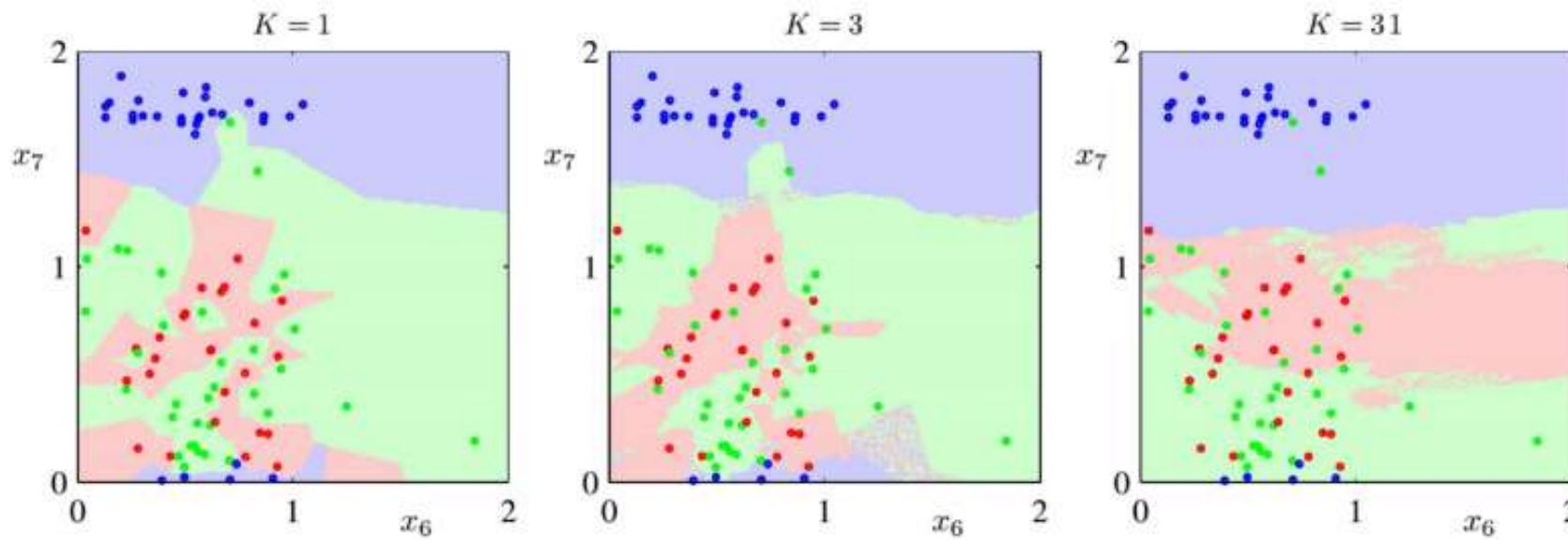
Output:



- Minkowski distance: The formula for Minkowski distance is in generalized form and we can manipulate it to get different distance metrics.
- The p value in the formula can be manipulated to give us different distances like:
  - p = 1, when p is set to 1 we get Manhattan distance
  - p = 2, when p is set to 2 we get Euclidean distance
- Euclidean distance
  - $d = \sqrt{(x_{22} - x_{11})^2 + (y_{22} - y_{11})^2}$
- Manhattan distance
  - sum for i to N sum || X1i – X2i ||

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

## KNN Overfitting and Underfitting.



Overfitting

Underfitting

The value of  $k$  in the KNN algorithm is related to the error rate of the model. A small value of  $k$  could lead to overfitting as well as a big value of  $k$  can lead to underfitting.

# Strengths and Weaknesses of KNN

## Strengths:

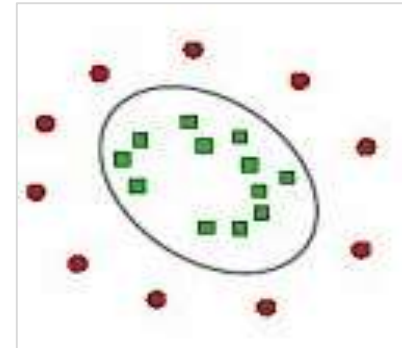
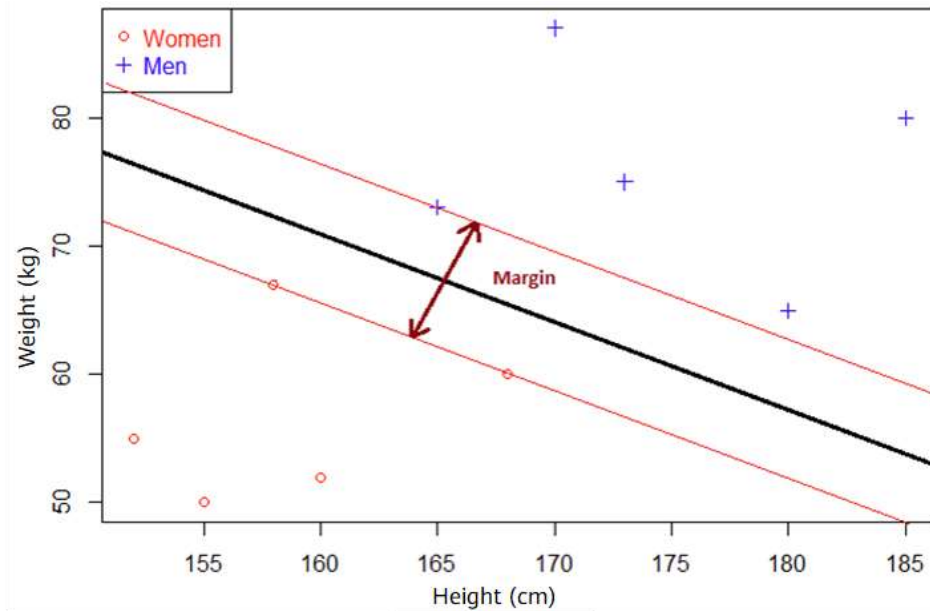
- Simple and easy to understand.
- Can work well on small datasets with low dimensions.
- Can handle both classification and regression tasks.

## Weaknesses:

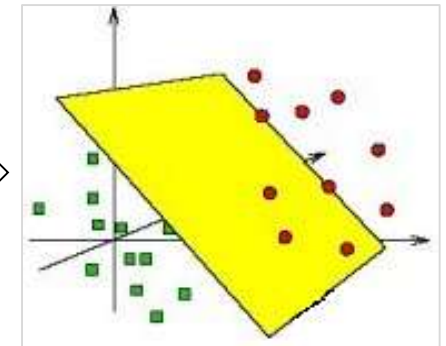
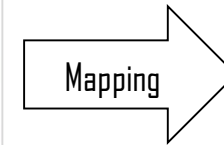
- Can be computationally expensive on large datasets or high-dimensional data.
- The choice of the hyperparameter K can have a significant impact on performance.
- Requires a lot of memory to store the training set.

# Support Vector Machine

- Support vector machines (SVMs) are binary classification models.
- Their basic model is the linear classifier that maximizes the width of the gap between the two categories in the feature space. SVMs also have a kernel trick, which makes it a non-linear classifier.



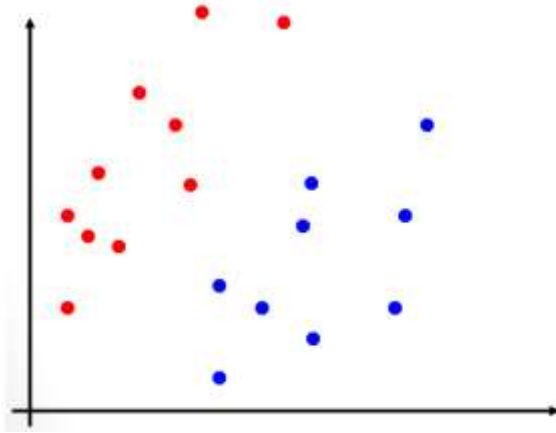
Difficult to split in a low-dimensional space.



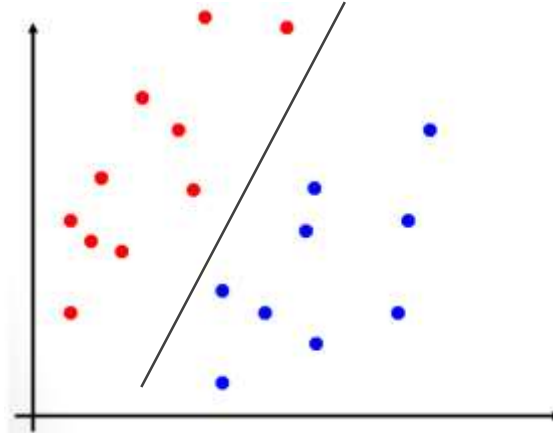
Easy to split in a high-dimensional space.

# Linear SVM (1)

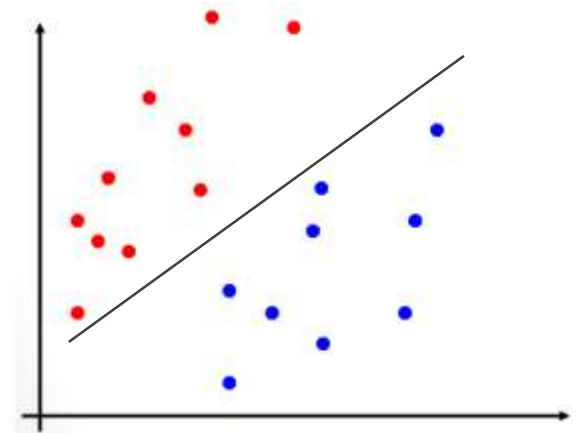
- How can we divide the red and blue data points with just one line?



Two-dimensional data set with two sample categories



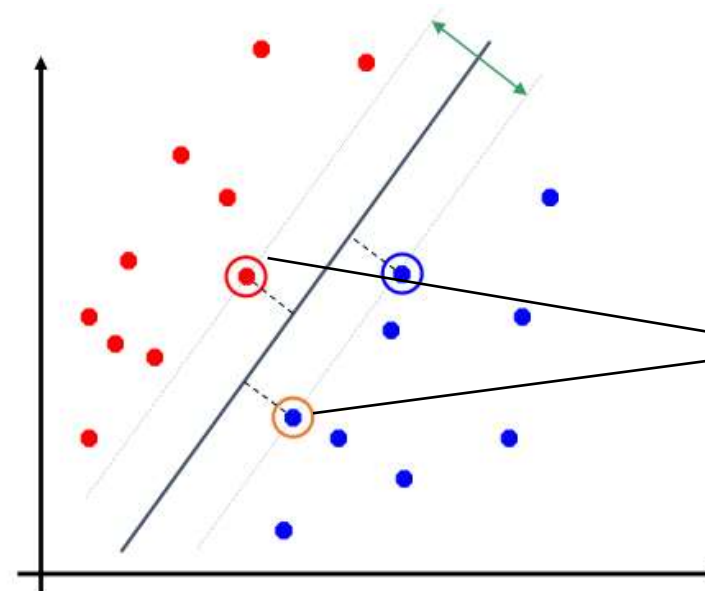
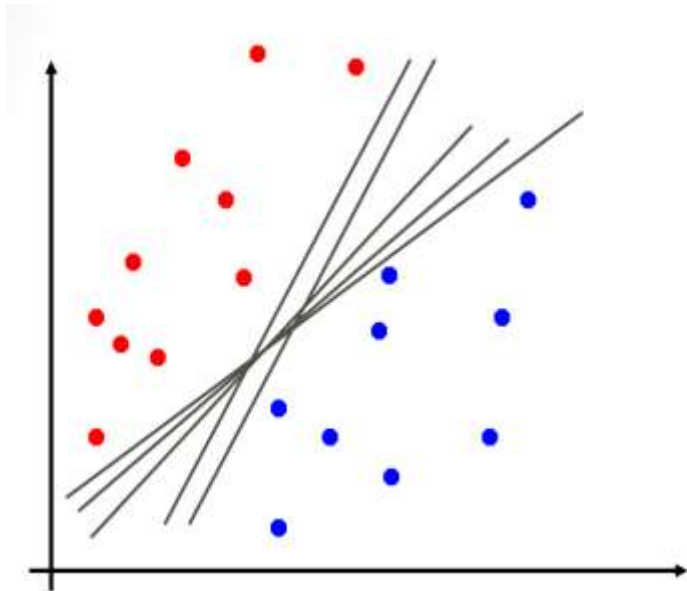
or



Both the division methods on the left and right can divide data. But which is correct?

# Linear SVM (2)

- We can use different straight lines to divide data into different categories. SVMs find a straight line and keep the most nearby points as **far** from the line as possible. This gives the model a strong generalization capability. These most nearby points are called **support vectors**.
- In the two-dimensional space, a straight line is used for division; in the high-dimensional space, a **hyperplane** is used for division.



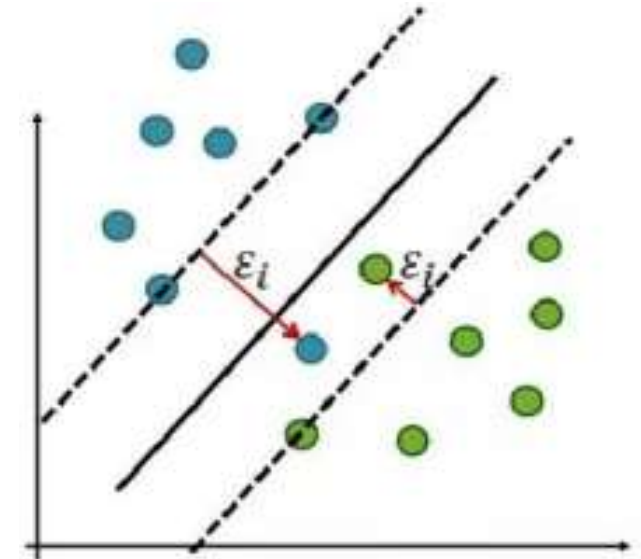
Maximize the distance  
from each support vector  
to the line

# Linear SVM (3)

- How SVM be controlled to deal with outliers ?

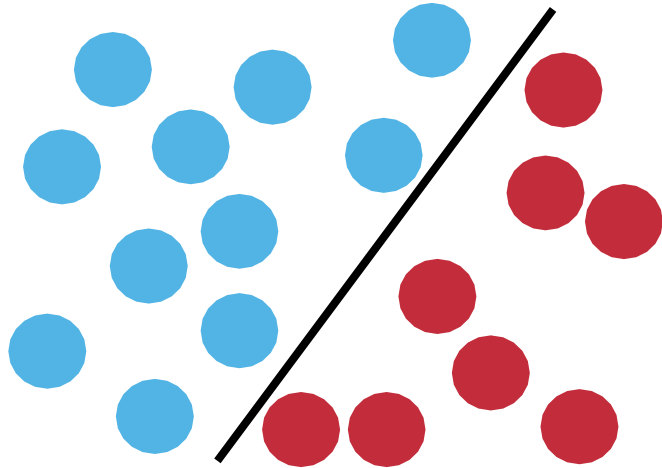
As shown with the data given that there are some outliers that would make the linear separation has some sort of errors.

In this case the SVM wouldn't work with their hard maximum margin classifier concept, so a **soft margin** was developed called soft margin classifier (AKA Support vector classifier) which allow for some misclassification and attempted to maximize margin or in other words make a balance between the error and the margin. This method is the realistic method used as it is hard to find data that is completely linearly separable with no error at all (hard classifiers)

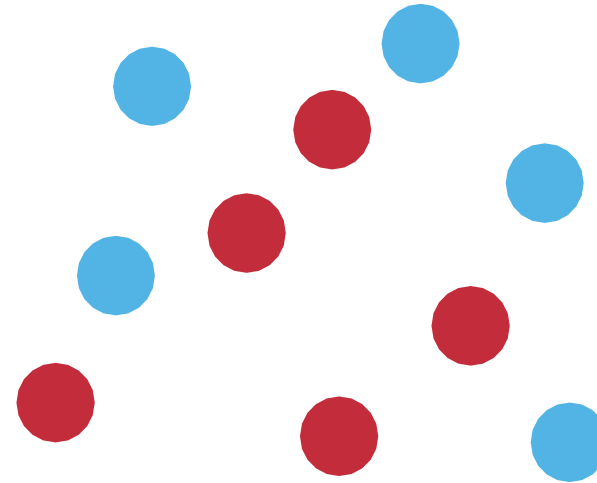


# Non-linear SVM (I)

- How can we divide a linear inseparable data set?



Linear SVM works well on a linear separable data set.

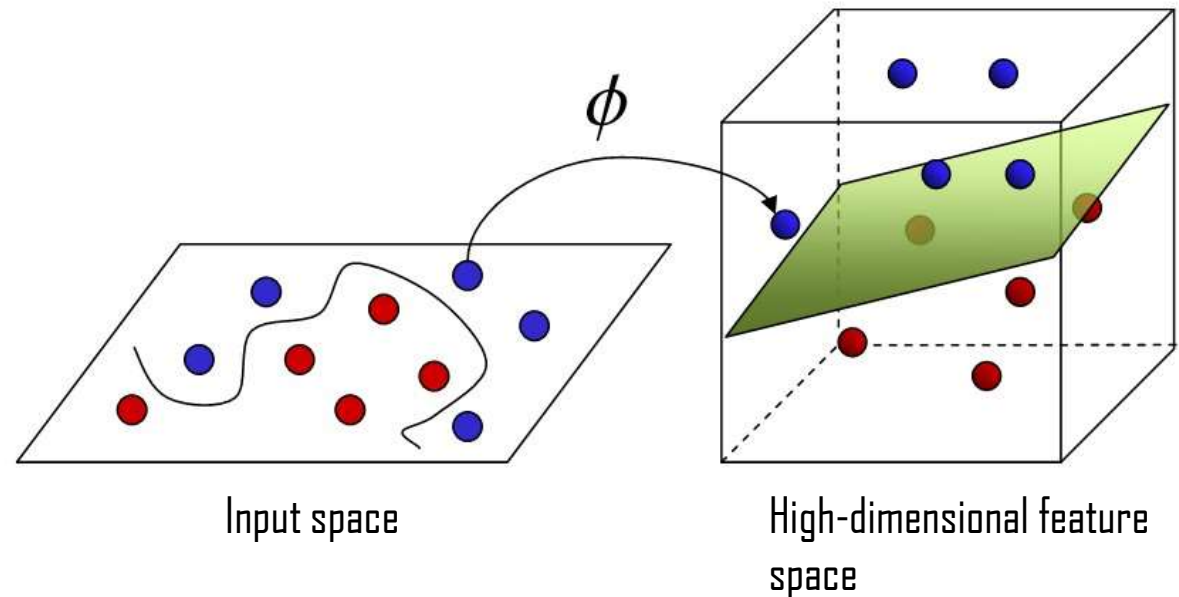
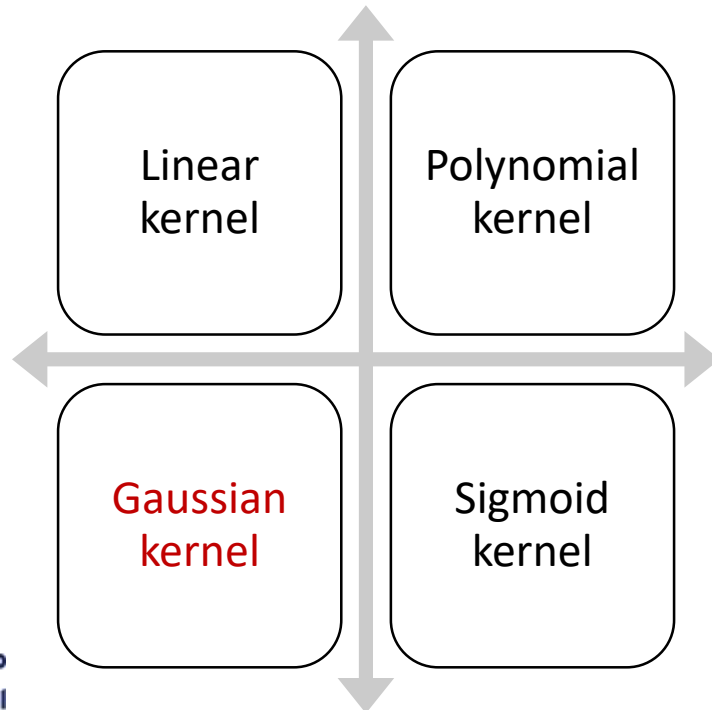


A non-linear data set cannot be divided using a straight line.

# Non-linear SVM (2)

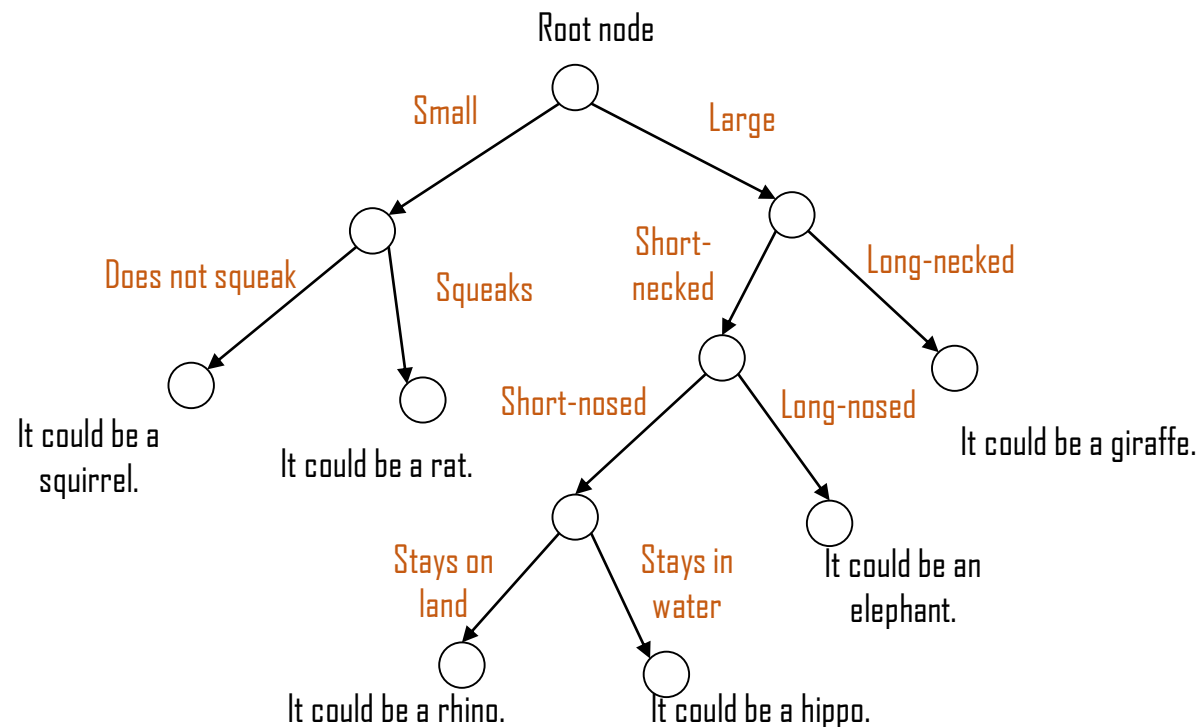
- Kernel functions can be used to create non-linear SVMs.
- Kernel functions allow algorithms to fit a maximum-margin hyperplane in a transformed high-dimensional feature space.

Common kernel functions

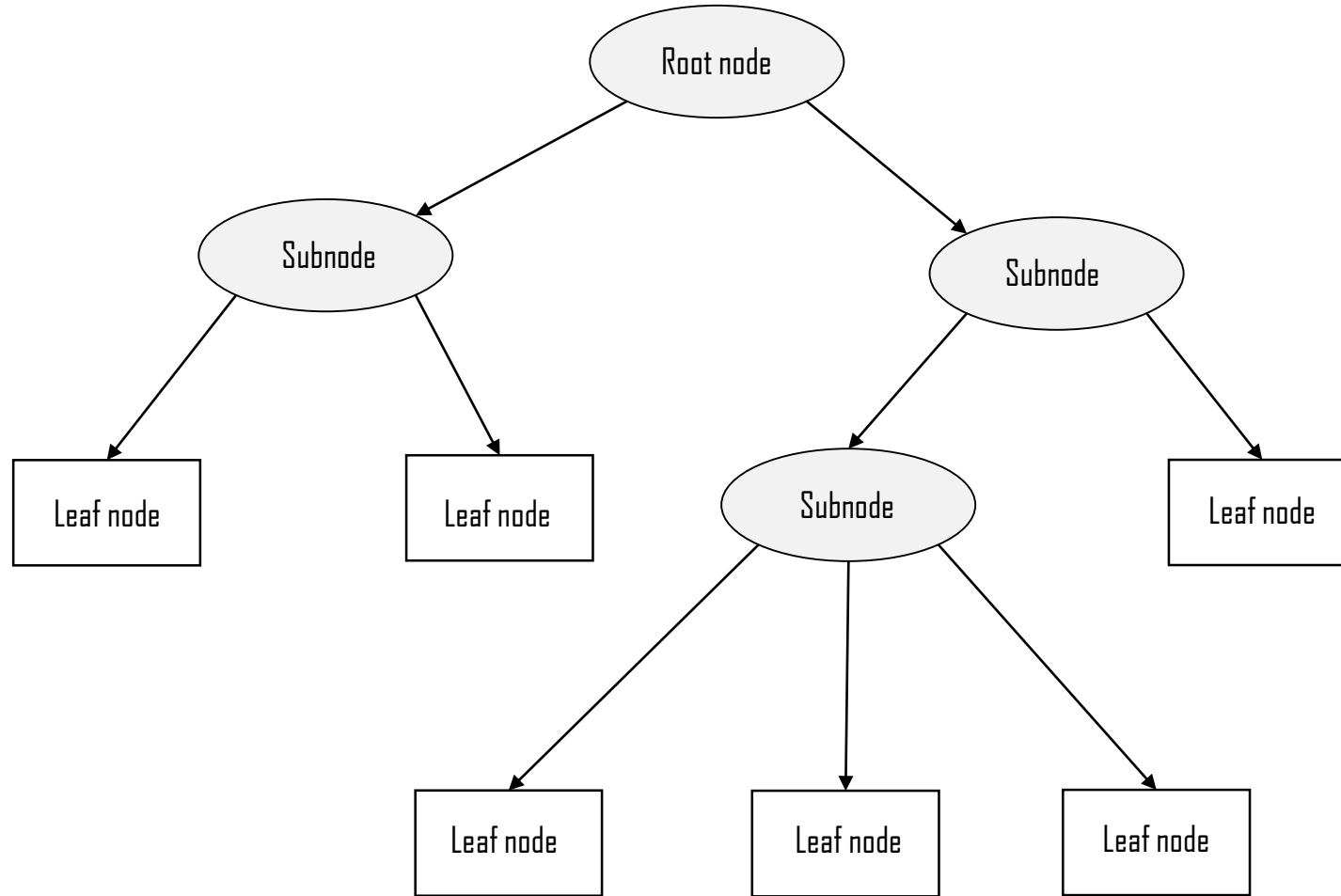


# Decision Tree

- Each non-leaf node of the decision tree denotes a test on an attribute; each branch represents the output of a test; and each leaf (or terminal) node holds a class label. The algorithm starts at the root node (topmost node in the tree), tests the selected attributes on the intermediate (internal) nodes, and generates branches according to the output of the tests. Then, it saves the class labels on the leaf nodes as the decision results.



# Structure of a Decision Tree



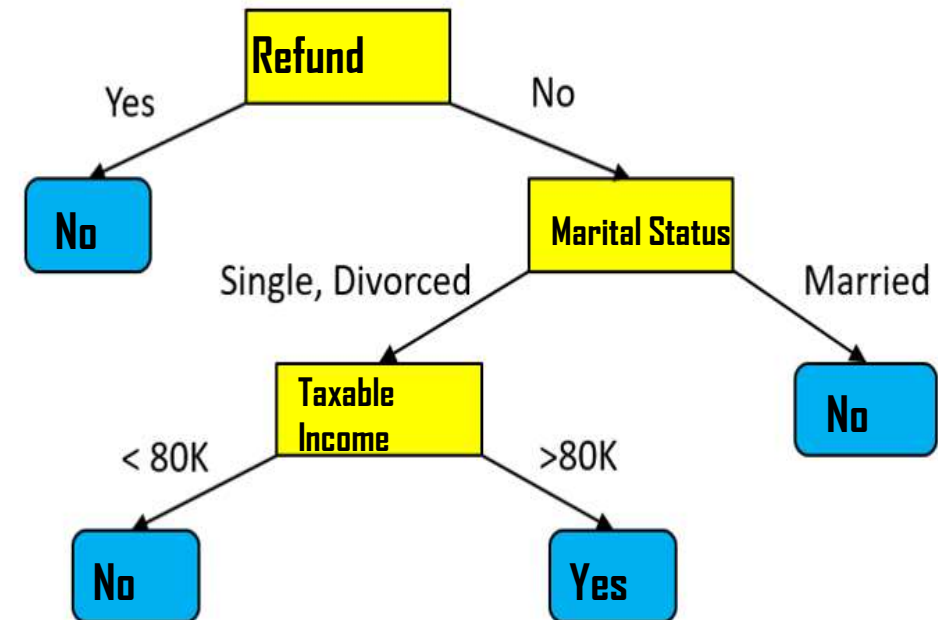
# Decision Tree Construction Process

- **Feature selection:** Select one of the features of the training data as the split standard of the current node. (Different standards distinguish different decision tree algorithms.)
- **Decision tree generation:** Generate sub nodes from top down based on the selected feature and stop until the dataset can no longer be split.
- **Pruning:** The decision tree may easily become overfitting unless necessary pruning is performed to reduce the tree size and optimize its node structure. Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances.

# Decision Tree Example

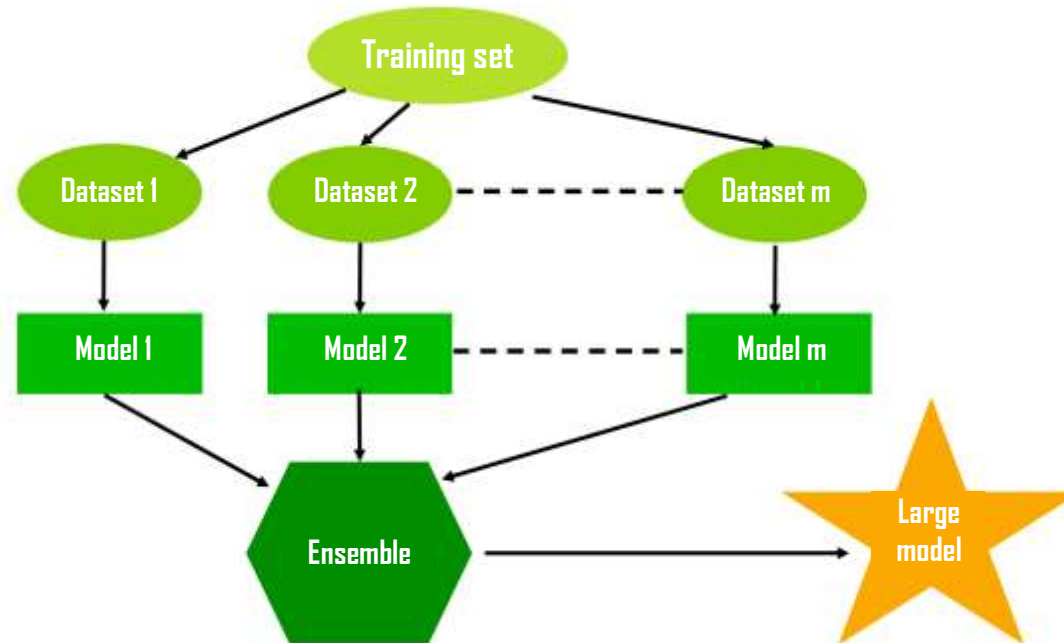
- The following figure shows a decision tree for a classification problem. The classification result is affected by three attributes: refund, marital status, and taxable income.

| TID | Refund | Marital Status | Taxable Income | Cheat |
|-----|--------|----------------|----------------|-------|
| 1   | Yes    | Single         | 125K           | No    |
| 2   | No     | Married        | 100K           | No    |
| 3   | No     | Single         | 70K            | No    |
| 4   | Yes    | Married        | 120K           | No    |
| 5   | No     | Divorced       | 95K            | Yes   |
| 6   | No     | Married        | 60K            | No    |
| 7   | Yes    | Divorced       | 220K           | No    |
| 8   | No     | Single         | 85K            | Yes   |
| 9   | No     | Married        | 75K            | No    |
| 10  | No     | Single         | 90K            | Yes   |



# Ensemble Learning

- Ensemble learning is a machine learning paradigm in which multiple learners are trained and combined to resolve an issue. When multiple learners are used, the generalization capability of the ensemble can be much stronger than that of a single learner.
- For example, If you ask thousands of people at random a complex question and then summarize their answers, the summarized answer is more accurate than an expert's answer in most cases. This is the **wisdom of the crowd**.

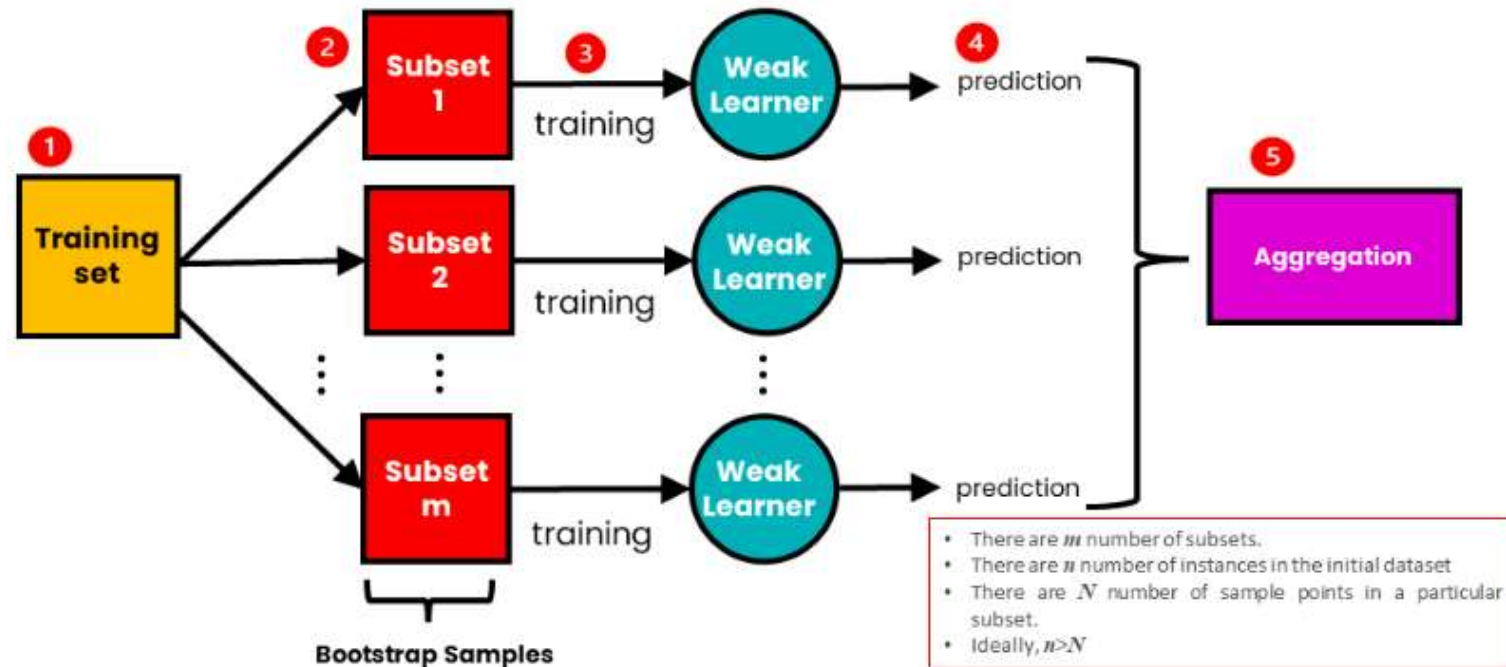


# Types of Ensemble Learning-Bagging

We use bagging for combining weak learners of high variance. Bagging aims to produce a model with lower variance than the individual weak models. These weak learners are homogenous, meaning they are of the same type. Bagging is also known as Bootstrap aggregating. It consists of two steps: bootstrapping and aggregation.

## Bootstrapping

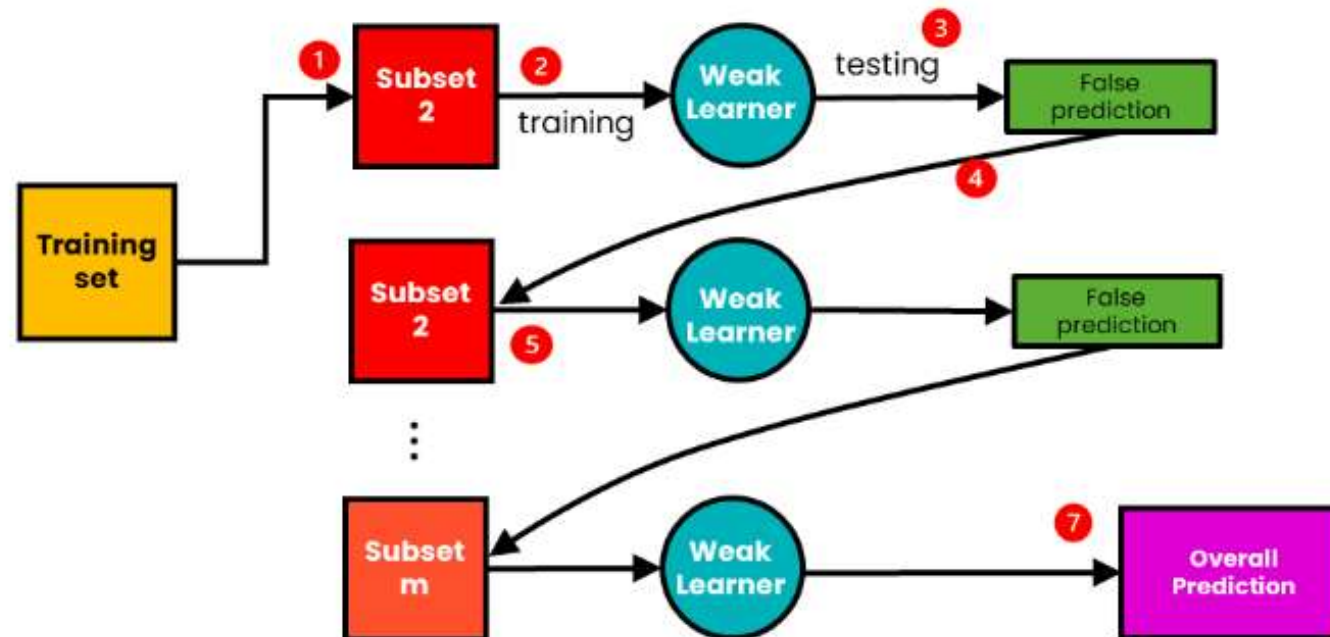
Involves resampling subsets of data with replacement from an initial dataset. In other words, subsets of data are taken from the initial dataset. These subsets of data are called **bootstrapped datasets** or, simply, bootstraps. Resampled 'with replacement' means an individual data point can be sampled multiple times. Each bootstrap dataset is used to train a weak learner.



# Types of Ensemble Learning-Boosting

We use boosting for combining weak learners with high bias. Boosting aims to produce a model with a lower bias than that of the individual models. Like in bagging, the weak learners are homogeneous.

**Boosting** involves sequentially training weak learners. Here, each subsequent learner improves the errors of previous learners in the sequence. A sample of data is first taken from the initial dataset. This sample is used to train the first model, and the model makes its prediction. The samples can either be correctly or incorrectly predicted. The samples that are wrongly predicted are reused for training the next model. In this way, subsequent models can improve on the errors of previous models



# Types of Ensemble Learning-Stacking

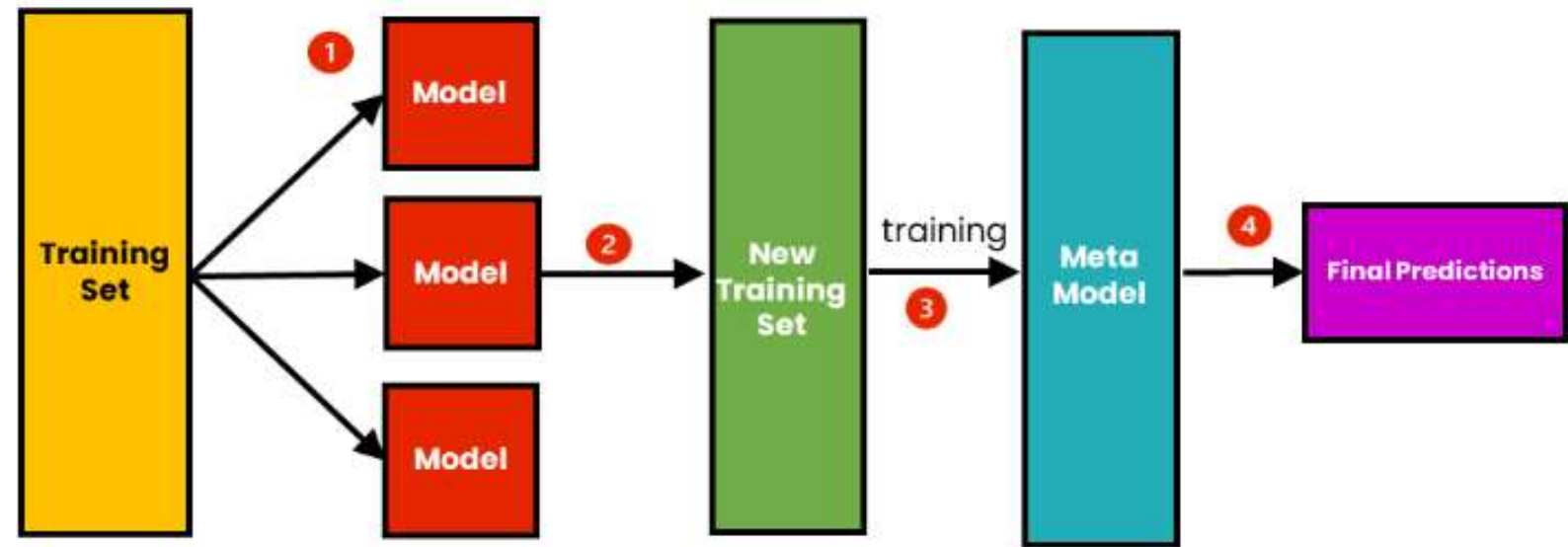
We use stacking to improve the prediction accuracy of strong learners. Stacking aims to create a single robust model from multiple heterogeneous strong learners.

**Stacking differs from bagging and boosting in that:**

It combines strong learners

It combines heterogeneous models

It consists of creating a Metamodel.



A metamodel is a model created using a new dataset.

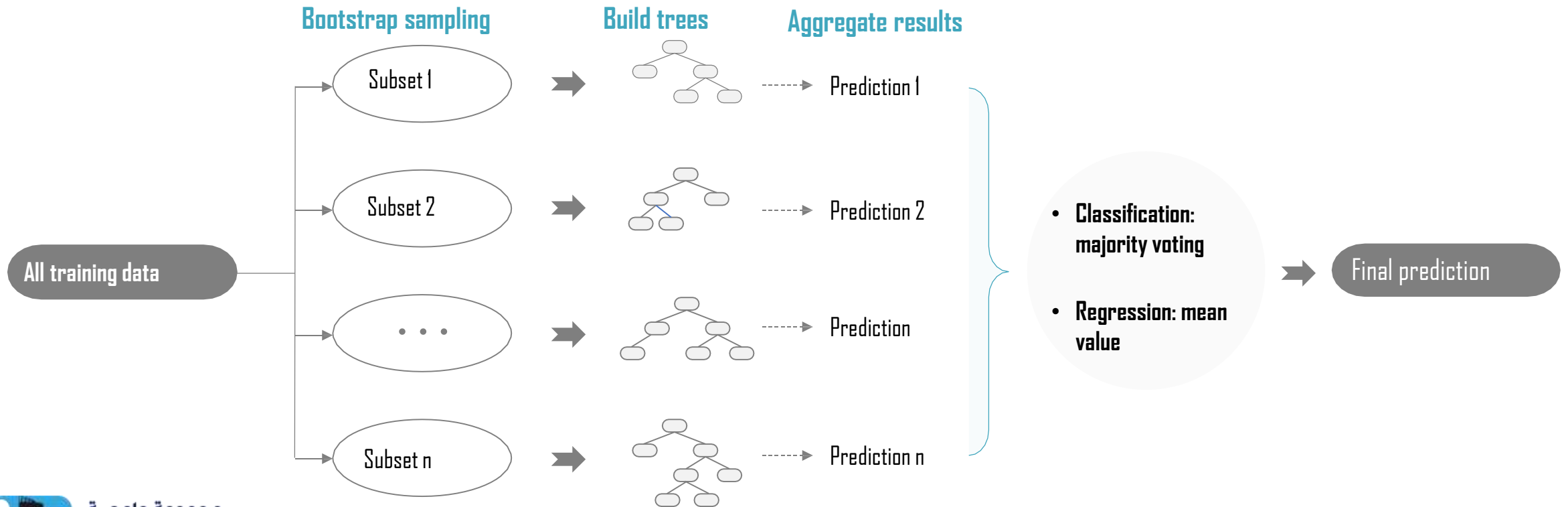
Individual heterogeneous models are trained using an initial dataset. These models make predictions and form a single new dataset using those predictions. This new data set is used to train the metamodel, which makes the final prediction. The prediction is combined using weighted averaging.

# When to use Bagging vs Boosting vs Stacking?

|                       | Bagging                  | Boosting           | Stacking           |
|-----------------------|--------------------------|--------------------|--------------------|
| Purpose               | Reduce Variance          | Reduce Bias        | Improve Accuracy   |
| Base Learner Types    | Homogeneous              | Homogeneous        | Heterogeneous      |
| Base Learner Training | Parallel                 | Sequential         | Meta Model         |
| Aggregation           | Max Voting,<br>Averaging | Weighted Averaging | Weighted Averaging |

# Ensemble Learning - Random Forest

- Random forest = Bagging + Classification and regression tree (CART)
- Random forest builds multiple decision trees and aggregates their results to make prediction more accurate and stable.
  - The random forest algorithm can be used for classification and regression problems.



- **Naive Bayes** classifiers are a family of simple "probabilistic classifiers" based on **Bayes' theorem** with **strong independence assumptions between the features**. For a given sample feature  $X$ , the probability that the sample belongs to category  $H$  is:

$$P(C_k | X_1, \dots, X_n) = \frac{P(X_1, \dots, X_n | C_k) P(C_k)}{P(X_1, \dots, X_n)}$$

- $X_1, X_2, \dots, X_n$  are data features, which are usually described by ~~their~~ measurement values of the attribute set.
  - For example, the attribute of the color feature may be red, yellow, and blue.
- $C_k$  indicates that the data belongs to a specific class  $C$ .
- $P(C_k | X_1, X_2, \dots, X_n)$  is the posterior probability, or the posterior probability of  $H$  under condition  $C_k$ .
- $P(C_k)$  is the prior probability independent of  $X_1, X_2, \dots, X_n$ .
- $P(X_1, X_2, \dots, X_n)$  is the prior probability of  $X$ .

- Feature independent hypothesis example:
  - If a fruit is red, round, and about 10 cm in diameter, it can be considered an apple.
  - A Naive Bayes classifier believes that each of these features independently contributes to the probability of the fruit being an apple, regardless of any possible correlation between color, roundness, and diameter features.



# Performance Evaluation of Classification

- Terms:

- $P$ : positive, indicating the number of real positive cases in the data.
- $N$ : negative, indicating the number of real negative cases in the data.
- $TP$ : true positive, indicating the number of positive cases that are correctly classified.
- $TN$ : true negative, indicating the number of negative cases that are correctly classified.
- $FP$ : false positive, indicating the number of positive cases that are incorrectly classified.
- $FN$ : false negative, indicating the number of negative cases that are incorrectly classified.

| Predicted \ Actual | Yes  | No   | Total   |
|--------------------|------|------|---------|
| Yes                | $TP$ | $FN$ | $P$     |
| No                 | $FP$ | $TN$ | $N$     |
| Total              | $P'$ | $N'$ | $P + N$ |

Confusion matrix

- The confusion matrix is an  $m \times m$  table at minimum. The entry  $CM_{i,j}$  in the first  $m$  rows and  $m$  columns indicates the number of cases that belong to class  $i$  but are labeled as  $j$ .
  - For classifiers with high accuracy, most of the cases should be represented by entries on the diagonal of the confusion matrix from  $CM_{1,1}$  to  $CM_{m,m}$ , while other entries are 0 or close to 0. That is,  $FP$  and  $FN$  are close to 0.

# Performance Evaluation of Classification (2)

| Measurement                                                  | Formula                                                                                                              |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
| Accuracy, recognition rate                                   | $\frac{TP + TN}{P + N}$                                                                                              |
| Error rate, misclassification rate                           | $\frac{FP + FN}{P + N}$                                                                                              |
| True positive rate, sensitivity, recall                      | $\frac{TP}{P}$                                                                                                       |
| True negative rate, specificity                              | $\frac{TN}{N}$                                                                                                       |
| Precision                                                    | $\frac{TP}{TP + FP}$                                                                                                 |
| $F_1$ value, harmonic mean of precision and recall           | $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$                            |
| $F_\beta$ value, where $\beta$ is a non-negative real number | $\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$ |

# Performance Evaluation of Classification Example

- In this example, an ML model was trained to identify an image of a cat. To evaluate the model's performance, 200 images were used, of which 170 of them were cats.
- The model reported that 160 images were cats.

$$\text{Precision: } P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5\%$$

$$\text{Recall: } R = \frac{TP}{P} = \frac{140}{170} = 82.4\%$$

$$\text{Accuracy: } ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 75\%$$

| Predicted \ Actual | <i>yes</i> | <i>no</i> | Total |
|--------------------|------------|-----------|-------|
| <i>yes</i>         | 140        | 30        | 170   |
| <i>no</i>          | 20         | 10        | 30    |
| Total              | 160        | 40        | 200   |

