Supervised Machine Learning Techniques with R

Dr. Olga Korosteleva, CSULB

SoCal R Users Group Workshop

May 25, 2024

Outline

A Big Picture

- We consider three types of settings: continuous response variable (**regression**), 0-1 response variable (**binary classification**), and 0-1-2, etc. response variable (**multinomial classification**).
- **Methods involving randomness**: Decision Tree (reg, bi, multi), Random Forest (reg, bi, multi), and Naïve Bayes (bi, multi).
- **Methods involving math optimization**: Gradient Boosting (reg, bi, multi), Support Vector Machine (reg, bi, multi), k-Nearest Neighbor (reg, bi, multi), and Artificial Neural Network (reg, bi, multi).

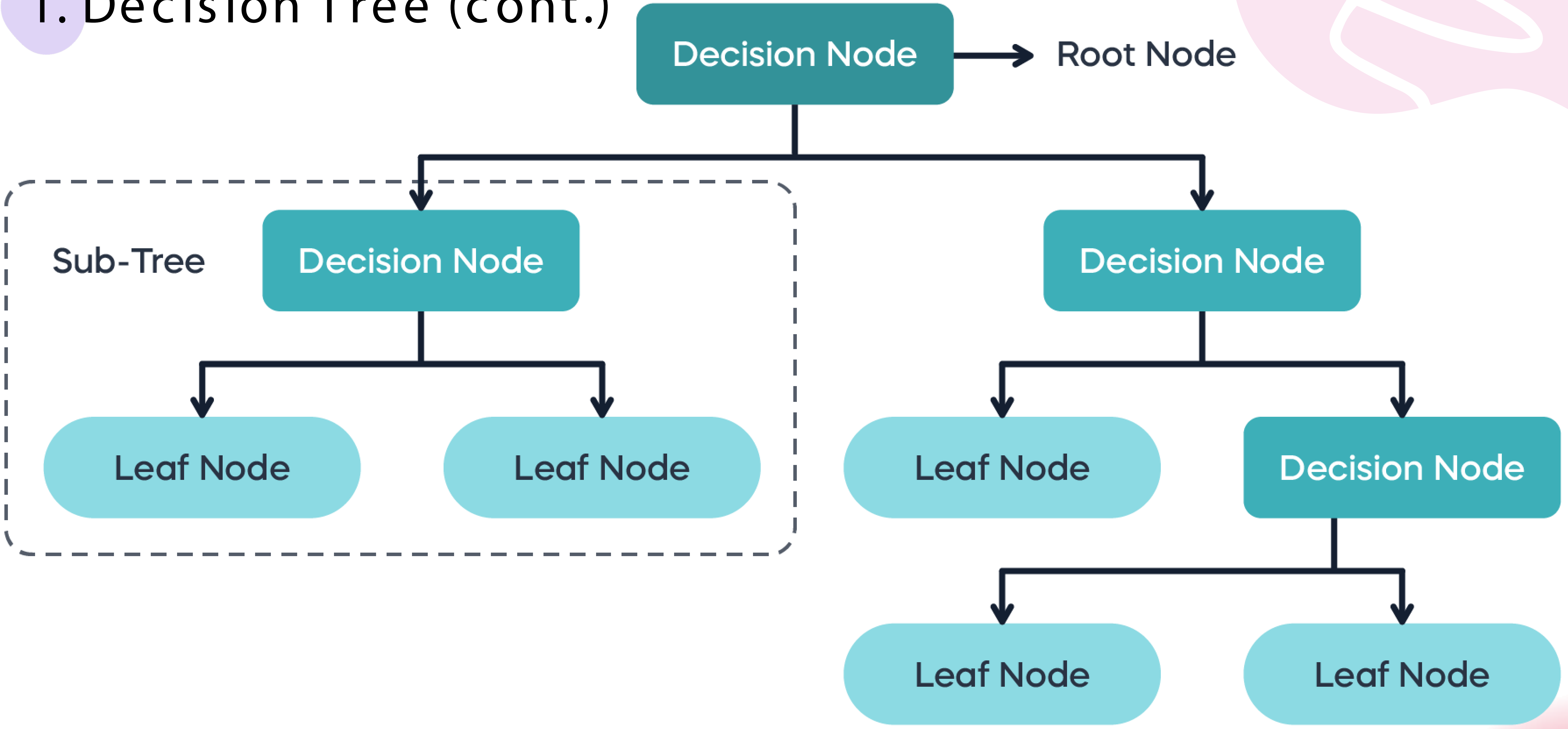
A Big Picture

- **Methods for time-series data:**
  - **(involves randomness):** change-point detection (continuous variable), and anomaly detection (continuous variable)
- **Method for texts:** Natural Language Processing

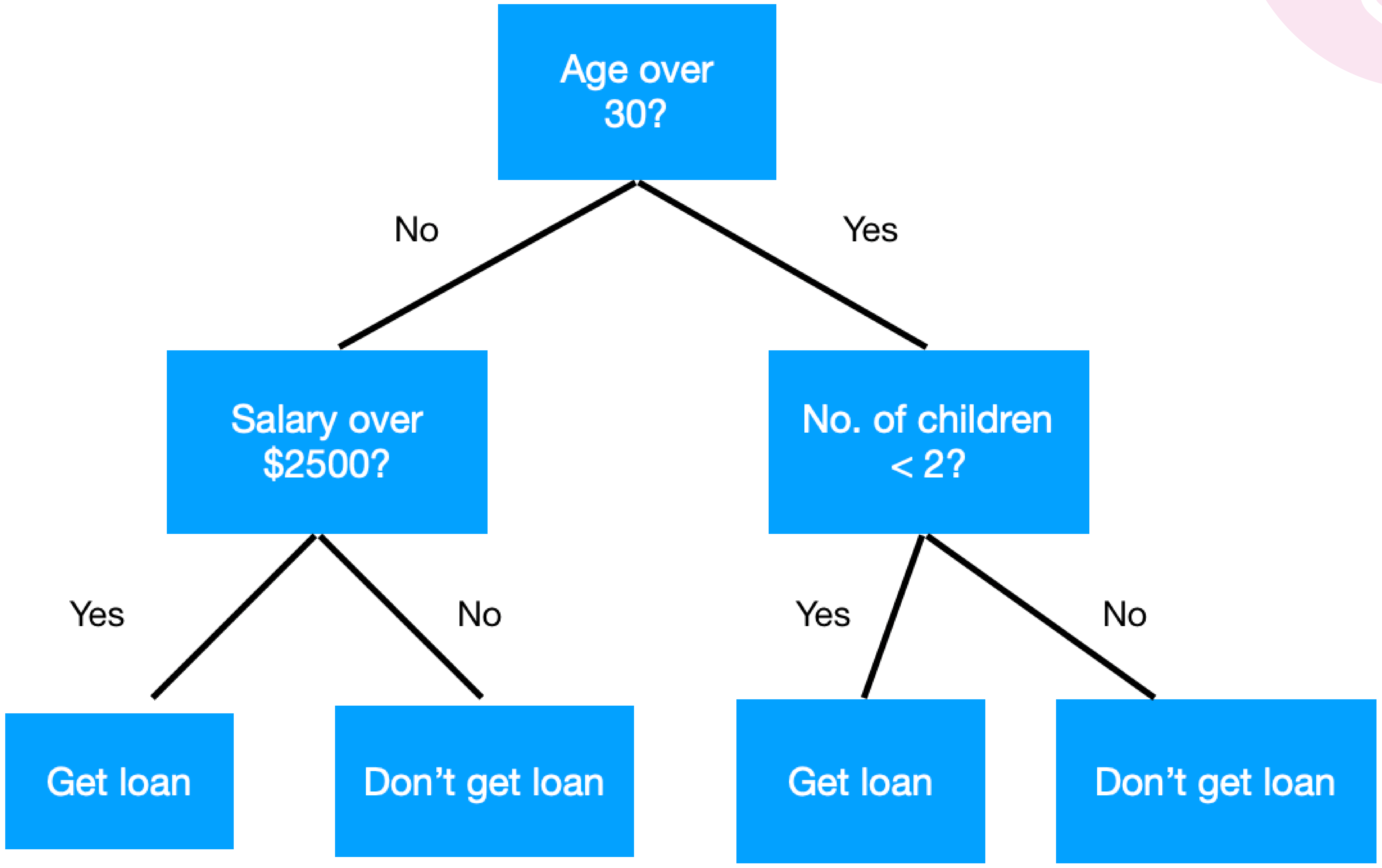
1. Decision Tree

A **decision tree** is a type of flowchart that shows a clear pathway to a decision. It starts at a single point (or **root node**) which then branches (or **splits**) in two or more directions. Each branch offers different possible outcomes until a final outcome is achieved. When plotted as a graph, it resembles a tree. The nodes at the end of the decision path are called **leaf nodes** (or **terminal nodes**). Between the root node and the leaf nodes, there are **internal nodes** (or **decision nodes**).

1. Decision Tree (cont.)



1. Decision Tree (cont.)



1. Decision Tree (cont.)

- **Decision trees** work for regression, binomial classification, and multinomial classification.
- The **response variable** is called the **target variable** and **predictors** are termed **features**.
- A decision tree **partitions** the population into **homogeneous non-overlapping sets**, based on the **most significant features**.
- If a **full decision tree** is too complex, **pruning** is applied, reducing the number of splits and terminal nodes.

1. Decision Tree (cont.)

- When fitting a model, it is customary to split the original data set randomly into a **training set** (70% or 80% of the data rows) and a **testing set** (remaining 30% or 20% of the rows).
- Then the model is developed on the **training set** and is used to **predict** values for the **testing set**.
- The **accuracy of prediction** is computed as a measure of the **goodness-of-fit** of the model.

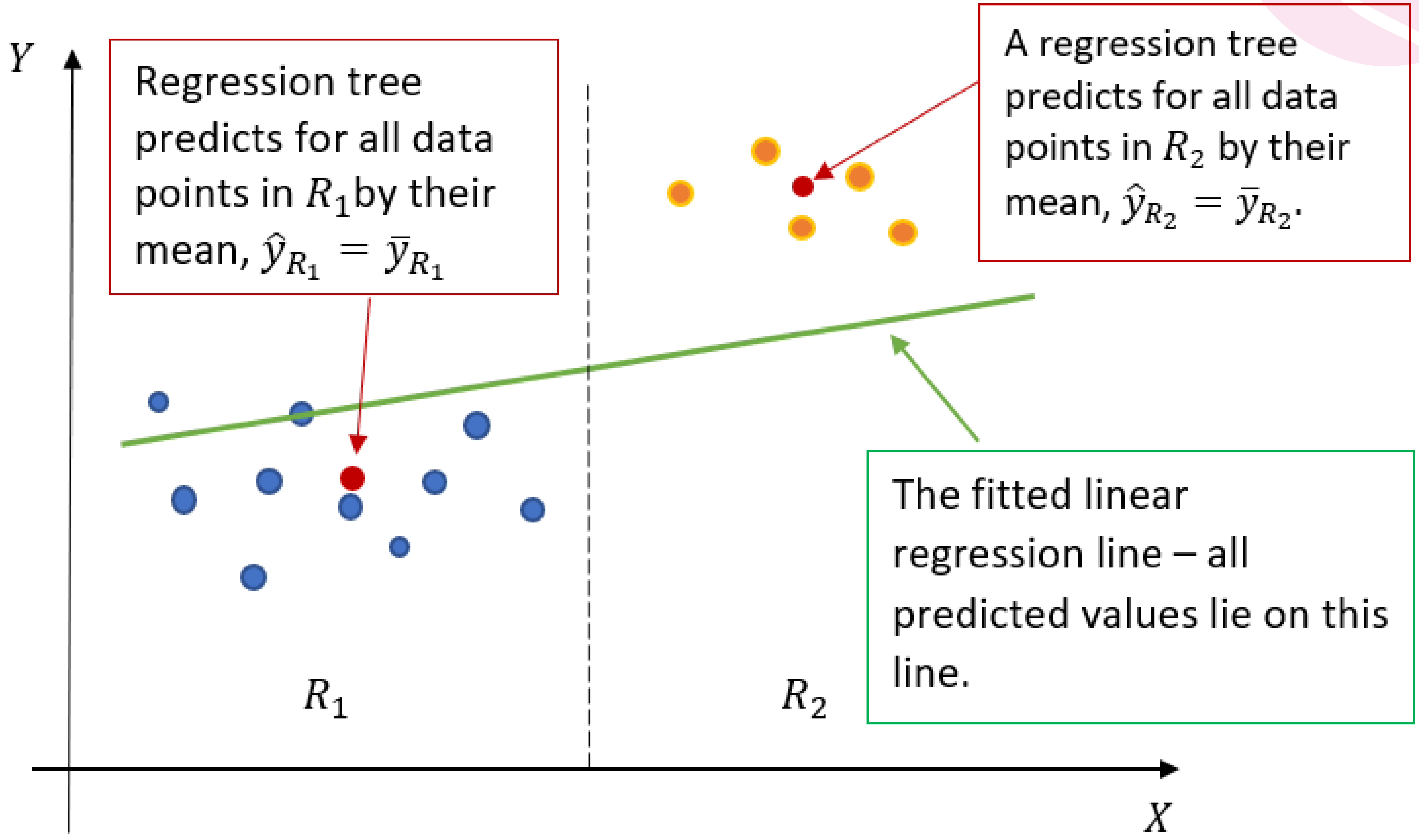
1. Decision Tree (cont.)

- For the **regression problem** (continuous target variable), the **prediction accuracy** is the proportion of predicted values that lie within 10%, 15%, and 20% of the actual values.
- For the **classification problem (bi or milti)**, prediction is given as the **probability** of each class, so the class with the **highest predicted probability** is assumed to be the **predicted class**. The **accuracy** in this case is the **proportion of classes** predicted **correctly**.

1. Decision Tree (cont.)

- For **all decision trees** (regression or classification), the splitting procedure is based on **minimizing** some **quantity**.
- Each split in a **regression tree** minimizes the **residual sum of squares** $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- It can be shown that the fitted value $\hat{y}_i$ is the **sample mean** of all observations on the same side of the split.

1. Decision Tree (cont.)



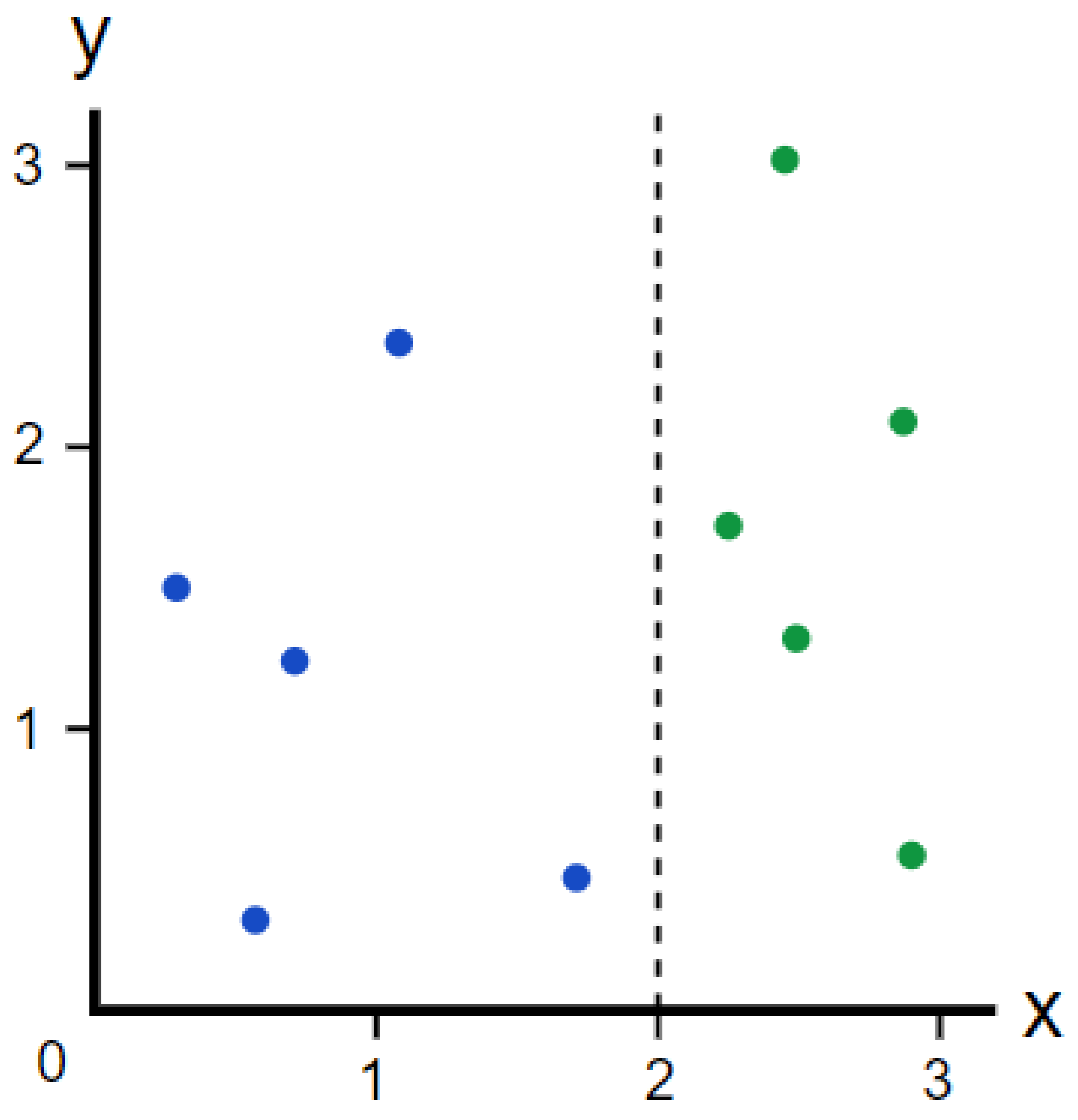
Y
Regression tree predicts for all data points in R_1 by their mean, $\hat{y}_{R_1} = \bar{y}_{R_1}$
A regression tree predicts for all data points in R_2 by their mean, $\hat{y}_{R_2} = \bar{y}_{R_2}$.
The fitted linear regression line – all predicted values lie on this line.
R_1
R_2
X

1. Decision Tree (cont.)

- Each split in a **classification tree** minimizes the **Gini impurity index** $G = \sum_{k=1}^{K} p_k(1 - p_k)$ where p_k is the proportion of observations in the *k*th class.

Example (*K*=2). Suppose we observe 10 data points, 5 blue and 5 green.

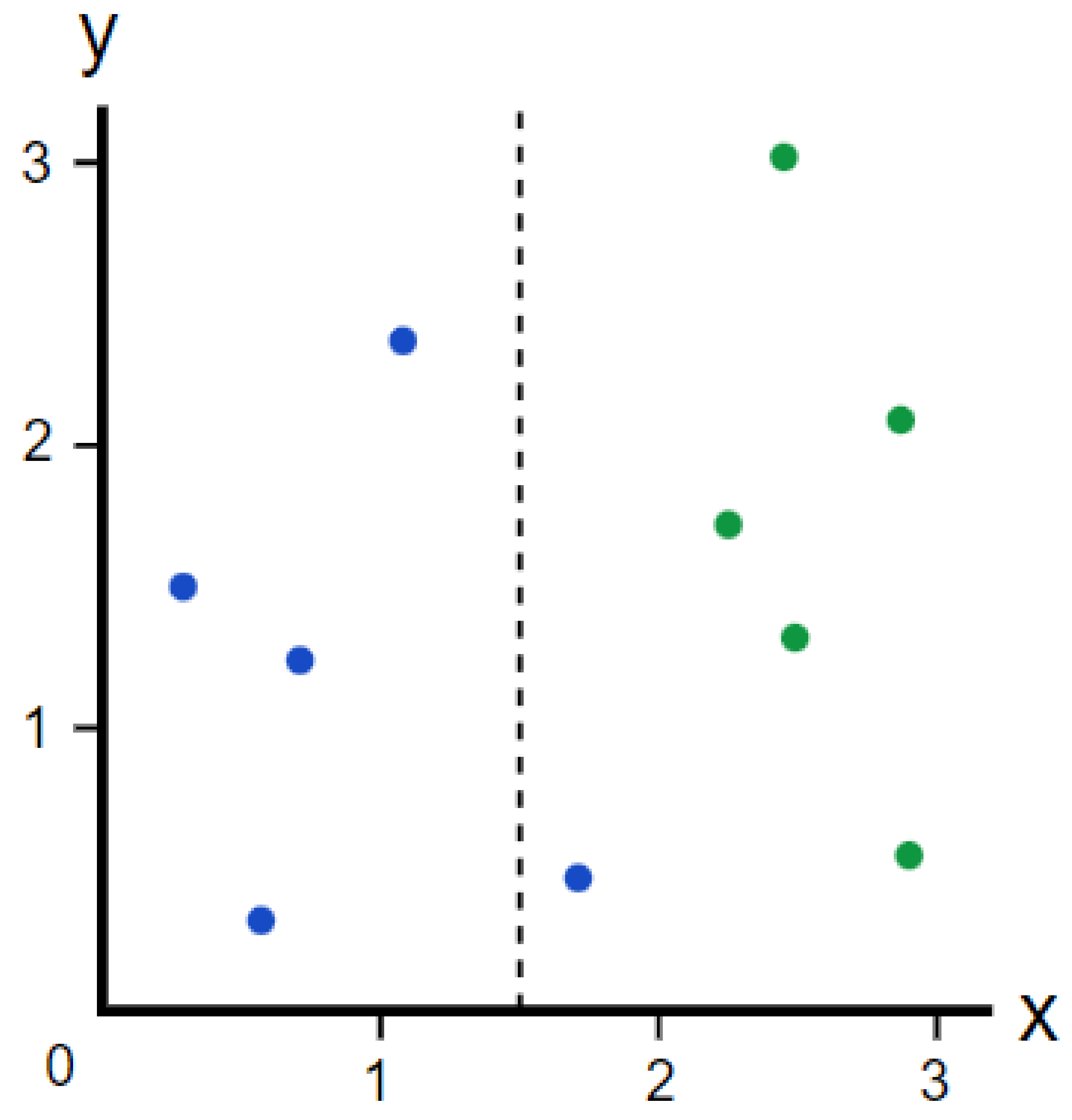
1. Decision Tree (cont.)



A Perfect Split

Consider a "perfect" vertical split at $X = 2$. 100% of the blue dots are classified correctly as blue, and 100% of the green dots are classified correctly as green, so the **Gini impurity index for the region on the left** is $G_{left} = P(blue)(1 - P(blue)) + P(green)(1 - P(green)) = (1)(1-1) + (0)(1-0) = 0$, and the one of the right is $G_{right} = 0$. Thus, a "perfect" split turned the data into two regions with zero impurity.

1. Decision Tree (cont.)



An Imperfect Split

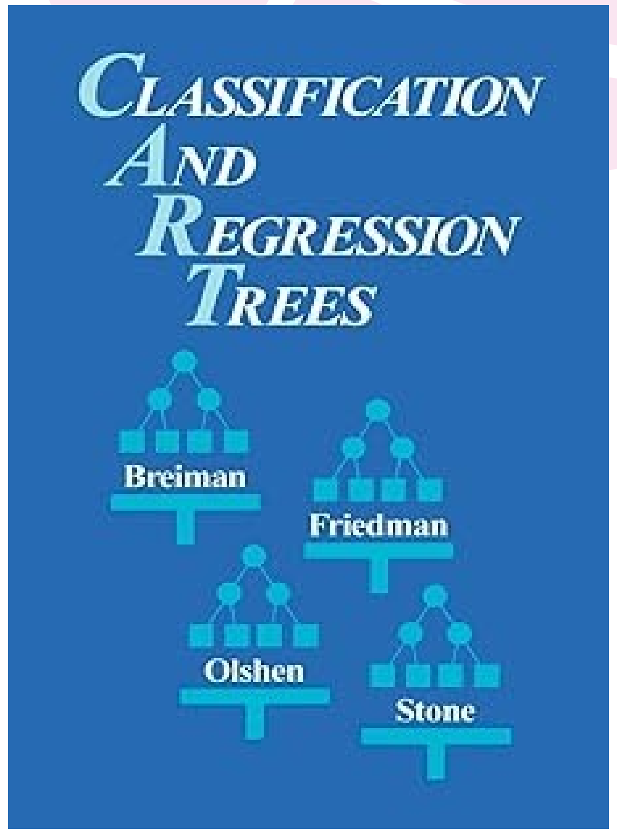
Consider now an "imperfect" split at $X = 1.5$ The region on the left has only blue data points, so we know that G_{left} =0. The region on the right has 1 blue and 5 green data points, and hence, $G_{right} = P(blue)(1 - P(blue)) + P(green)(1 - P(green)) = \left(\frac{1}{6}\right)\left(1 - \frac{1}{6}\right) + \left(\frac{5}{6}\right)\left(1 - \frac{5}{6}\right) = \frac{10}{36} = 0.2778.$

The **quality of a split** is determined by **weighting** the impurity of **each region** by how many data points it has. Since the left region has 4 data points and the right region has 6, we get $G = (0.4)(0) + (0.6)(0.2778) = 0.1667 > 0$, so the "perfect" split is preferred.

1. Decision Tree (cont.)

Historical Note.

- The **decision tree algorithm** was introduced in the book “Classification and Regression Trees” by Breiman, L., Friedman, J., Olshen, R., and C. Stone, Chapman and Hall, Wadsworth, New York, 1984.



- The **Gini impurity index** is named after an Italian statistician **Corrado Gini (1884-1965)** who proposed the idea in his 1912 book published in Italian under the name of “Variabilità e Mutabilità” (“Variability and Mutability”).

1. Decision Tree – Coding Examples

- Regression Tree – Housing Data
- Binary Classification Tree – Pneumonia Data
- Multinomial Classification Tree – Movie Data

2. Random Forest

- Decision trees are prone to **bias** and **overfitting**.
- A more practical approach is to construct **multiple decision trees** and **combine** the results into a **single output**. This approach is termed the **ensemble method**.
- The most well-known ensemble method is **bagging** (**bootstrap** + **aggregation**). This method was introduced in 1996 by Leo Breiman (1928-2005).

2. Random Forest (cont.)

- In the **bagging** method, data points in the **training set** are sampled **with replacement** (producing a **bootstrap sample**).
- Decision trees are generated **independently** for each **bootstrap sample**.
- The results are then **aggregated**. For **regression** trees, an **average of predicted values** is computed. For **classification** trees, the **majority of the predictions** define the predicted class.
- The **bagging method** reduces **variance** and, as a result, yields **more accurate** predictions than **individual** decision trees.

2. Random Forest (cont.)

- The **random forest algorithm** is an extension of the bagging method as it utilizes both **bagging** and **feature randomness.**
- **Feature randomness** generates a **random subset of features.** This ensures a **low correlation** among **generated decision trees**.
- The results are then **aggregated** into a **single prediction**.
- **The random forest algorithm** has **three main hyperparameters:** the number of nodes, the number of trees, and the number of variables sampled.

2. Random Forest (cont.)

- Random forest makes it easy to evaluate the **feature importance**, which is the **contribution of each splitting feature** to the model.
- To measure the feature importance, the **loss reduction** method is used. It measures how much the model's **accuracy decreases** when a given feature is **excluded**.
- For random forest **regression**, the **loss function** is the **mean squared error** $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.
- For random forest **classification**, the **loss function** is the **reduction in the Gini impurity index**.

2. Random Forest – Coding Examples

- Random Forest Regression – Housing Data
- Random Forest Binary Classification – Pneumonia Data
- Random Forest Multinomial Classification – Movie Data

3. Naïve Bayes

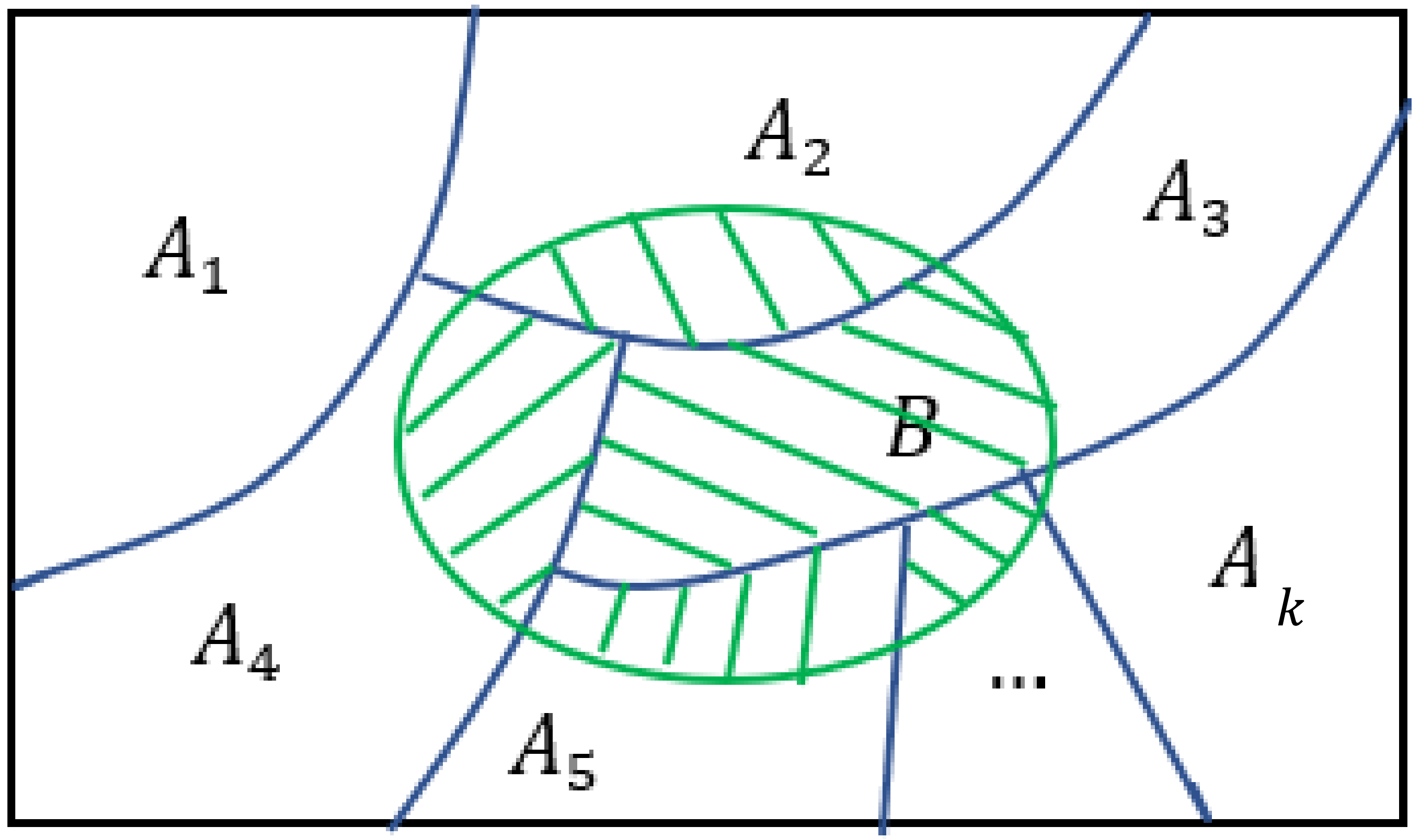
- **Naïve Bayes classification** is a method used for binary and multinomial classifications but not a regression.
- The method utilizes **the Bayes' formula**.

Historical Note.
Reverend Thomas Bayes (c. 1701 – 1761) was an English statistician, philosopher, and Presbyterian minister.

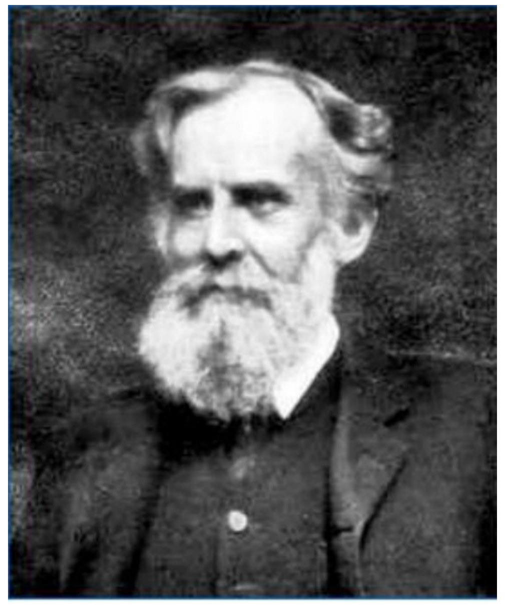
3. Naïve Bayes (cont.)

- Suppose events $A_1, \dots, A_k$ **partition** the set of all possible outcomes of a random phenomenon, and don't overlap.
- Suppose an event B has happened.
- Knowing the **prior probabilities** $P(A_1), \dots, P(A_k)$ and the **conditional probabilities** $P(B|A_1), \dots, P(B|A_k)$, we can update our knowledge about the events $A_1, \dots, A_k$ by computing the **posterior probabilities** via the **Bayes' formula**.

3. Naïve Bayes (cont.)



Historical Note. John Venn (1834 –1923) was a British logician and philosopher. He introduced the **Venn diagram** in 1880.



Bayes' Formula. For any fixed $i, i = 1, \dots, k$, the conditional probability of A_i given B is computed as

$$P(A_i|B) = \frac{P(A_i \; and \; B)}{P(B)} = \frac{P(B|A_i)P(A_i)}{P(B)} = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{k} P(B|A_j)P(A_j)}.$$

3. Naïve Bayes (cont.)

Back to Naïve Bayes Classification.

- Suppose there are k predictors $\mathbf{X} = (X_1, \ldots, X_k)$, which can be categorical, or continuous. Let Y denote the response variable. By the Bayes' formula,

$$P(Y|\mathbf{X}) = \frac{P(\mathbf{X}|Y)P(Y)}{P(\mathbf{X})}.$$

- Next, we assume that the predictors are **conditionally independent**, given Y, and write

$$P(Y|\mathbf{X}) = \frac{P(Y)\prod_{i=1}^{k} P(X_i|Y)}{P(\mathbf{X})}.$$

This assumption is rather **naive**, hence the **name** of the technique.

3. Naïve Bayes (cont.)

- In **classification problem** (binary or multinomial), we compute the conditional (posterior) probability $P(Y|\mathbf{X})$ for each class, and predict the class with the **highest probability**.
- Since the denominator $P(\mathbf{X})$ is present in each expression, it can be ignored. That is, $P(Y|\mathbf{X})$ is **proportional** to $P(Y)\prod_{i=1}^{k} P(X_i|Y)$ up to a **multiplicative constant**.
- So, to apply the Naïve Bayes Classification method, we need to know how to estimate the **prior probability** $P(Y = y)$ and the **conditional probabilities** $P(X_i = x|Y = y), i = 1, \dots, k,$ for each **response class** y.

3. Naïve Bayes (cont.)

- To estimate the **prior probability** $P(Y = y)$ of each class y, we compute the **fraction of observations in each class** in the **training set**.
- For **categorical predictors**, we compute **the empirical conditional probabilities** $P(X_i = x|Y = y)$ as the **fraction of observations in the class** $Y = y$ in the **training set** for which $X_i = x$.
- For **continuous predictors**, we assume that the **underlying distribution** is **normal (Gaussian)** with the **estimated mean** of $\bar{x}$ and **estimated variance** $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$.

3. Naïve Bayes (cont.)

Characteristics of Naive Bayes Classifiers.

- Robust to **outliers** because they **average out** when computing posterior probabilities.
- Handles **missing values** by ignoring the missing data points in calculations.
- Robust to **irrelevant predictors** since $P(X_i|Y)$is almost **uniformly distributed** and factors out in comparisons of posterior probabilities.
- **Correlated** predictors can **degrade the performance** of the technique. The **conditional independence** assumption is the key.

3. Naïve Bayes – Example

- Suppose the training data are as given in the table below.

ID	Home Owner	Marital Status	Annual Income ($K)	Defaulted Borrower
1	yes	single	125	no
2	no	married	100	no
3	no	single	70	no
4	yes	married	120	no
5	no	divorced	95	yes
6	no	married	60	no
7	yes	divorced	220	no
8	no	single	85	yes
9	no	married	75	no
10	no	single	90	yes

3. Naïve Bayes – Example (cont.)

ID	Home Owner	Marital Status	Annual Income ($K)	Defaulted Borrower
1	yes	single	125	no
2	no	married	100	no
3	no	single	70	no
4	yes	married	120	no
5	no	divorced	95	yes
6	no	married	60	no
7	yes	divorced	220	no
8	no	single	85	yes
9	no	married	75	no
10	no	single	90	yes

- The **prior probabilities** are

  $P(\text{default} = \text{"no"}) = 7/10 = 0.7$ and
  $P(\text{default} = \text{"yes"}) = 3/10 = 0.3.$

- The **conditional probabilities** are:

$P(\text{homeowner} = \text{"yes"} \,|\text{default} = \text{"no"}) = 3/7, P(\text{homeowner} = \text{"no"}|\text{default} = \text{"no"}) = 4/7, P(\text{homeowner} = \text{"yes"}|\text{default} = \text{"yes"}) = 0,$ $P(\text{homeowner} = \text{"no"}|\text{default} = \text{"yes"}) = 1, P(\text{marital} = \text{"single"}|\text{default} = \text{"no"}) = 2/7, P(\text{marital} = \text{"married"}|\text{default} = \text{"no"}) = 4/7, P(\text{marital} = \text{"divorced"}|\text{default} = \text{"no"}) = 1/7, P(\text{marital} = \text{"single"}|\text{default} = \text{"yes"}) = 2/3, P(\text{marital} = \text{"married"}|\text{default} = \text{"yes"}) = 0, P(\text{marital} = \text{"divorced"}|\text{default} = \text{"yes"}) = 1/3.$

3. Naïve Bayes – Example (cont.)

- For $\mathbf{default} = \text{"}\mathbf{no}\text{"}$, the **prior density** for annual income is **normal** with **mean** = (125 + 100 + 70 + 120 + 60 + 220 + 75)/7 = 110, and **variance** = 2975, and for $\mathbf{default} = \text{"}\mathbf{yes}\text{"}$, it is **normal** with **mean** = (95 + 85 + 90)/3 = 90, and **variance** = 25.
- Suppose we would like to **predict** the **default status** for a person who is not a homeowner, who is single, and whose annual income is \$120K. We write $P(\mathbf{X}|\text{default} = \text{"no"}) = P(\text{homeowner} = \text{"no"}|\text{default} = \text{"no"})$

$$P(\text{marital} = \text{"single"}|\text{default} = \text{"no"})P(\text{income} = \$120\text{K}|\text{default} = \text{"no"})$$

$$=(4/7)(2/7)\frac{1}{\sqrt{(2\pi)(2975)}}\exp\left(-\frac{(120-110)^2}{(2)(2975)}\right) = 0.001215,$$

3. Naïve Bayes – Example (cont.)

and $P(\mathbf{X}|\text{default} = \text{"yes"}) = P(\text{homeowner} = \text{"no"}|\text{default} = \text{"yes"})$

$$P(\text{marital} = \text{"single"}|\text{default} = \text{"yes"})P(\text{income} = \$120\text{K}|\text{default} = \text{"yes"})$$

$$=(1)(2/3)\frac{1}{\sqrt{(2\pi)(25)}}\exp\left(-\frac{(120-90)^2}{(2)(25)}\right) = 8.1 \cdot 10^{-10}.$$

- Hence, $P(\text{default} = \text{"no"}|\mathbf{X}) = P(\text{default} = \text{"no"})P(\mathbf{X}|\text{default} = \text{"no"})/P(\mathbf{X})$

$$= \frac{(0.7)(0.001215)}{P(\mathbf{X})} = \frac{0.000851}{P(\mathbf{X})},$$

and $P(\text{default} = \text{"yes"}|\mathbf{X}) = P(\text{default} = \text{"yes"})P(\mathbf{X}|\text{default} = \text{"yes"})/P(\mathbf{X})$

$$= \frac{(0.3)(8.1 \cdot 10^{-10})}{P(\mathbf{X})} = \frac{2.43 \cdot 10^{-10}}{P(\mathbf{X})}.$$

- We can see that $P(\text{default} = \text{"no"}|\mathbf{X}) > P(\text{default} = \text{"yes"}|\mathbf{X})$ and so we **predict no default** for this person.

3. Naïve Bayes – Coding Examples

- Naïve Bayes Binary Classification – Pneumonia Data
- Naïve Bayes Multinomial Classification – Movie Data

4. Gradient Boosting

4.1. Boosting Method

- First we consider a method known as **boosting**. Models are generated sequentially and iteratively, meaning that it is necessary to have information from iteration i before conducting iteration $i + 1$.
- The **boosting method** was introduced by Michael Kearns and Leslie Valiant in 1989.

Dr. Michael Kearns, UPenn

Dr. Leslie Valiant, Harvard University

4.1. Boosting Method (cont.)

- The question was whether it was possible to combine weak models (**weak learners**) to produce a single strong model (a **strong learner**).
- **Weak learners** are only slightly better than chance at predicting a response. A **strong learner** is well-correlated with the response.
- The idea of the **boosting method** is to **build iteratively** weak machine learning models on a **continually updated response variable** in the **training data set** and then add them together to produce a final **strong learning model**.

4.1. Boosting Method (cont.)

- In the **relation** $y = f(x)$, we need to **estimate** the **function** f.
- A **boosting algorithm** proceeds as follows:

1. The **initial estimator** is set to **zero**, $\hat{f}(x) = 0$, and the **residuals** are set to current responses $r = y$ for all elements in the training set.

2. The number of **boosting trees** B is specified and then the loop over $b = 1, \dots, B$ is run:

Step 1. A **weak-learning tree** $\hat{f}_b$ with k splits is grown on the training data (x, r).

Step 2. Estimator $\hat{f}$ is updated as $\hat{f}_{new}(x) = \hat{f}_{old}(x) + \lambda \hat{f}_b(x)$ for some scale parameter lambda $\lambda, 0 < \lambda < 1$, called the **shrinkage rate** (or **learning rate**).

4.1. Boosting Method (cont.)

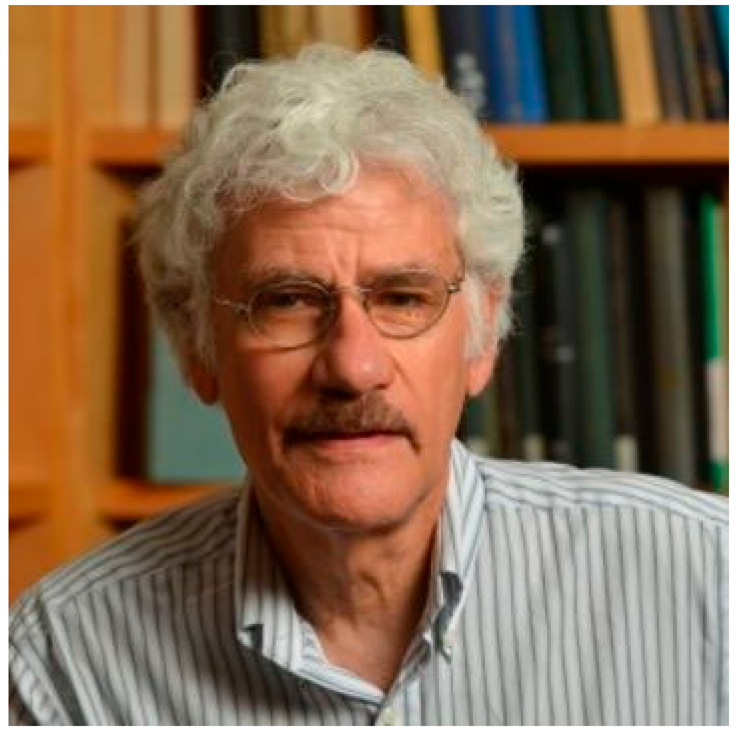
Step 3. Residuals are updated as $r_{new} = r_{old} - \lambda \hat{f}_b(x)$ and used as the response variable for the next iteration.

3. The **final boosted model** is computed as the **sum** of individual **weak learners**, $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}_b(x)$.

- Notice that each subsequent tree is fitted to the **residuals** of the **previous** iteration. Hence, each subsequent iteration is slowly improving the overall performance of the model.
- In the **boosting algorithm**, there are **three hyperparameters**: the number of boosted trees B, the number of splits k, and the shrinkage rate λ.

4.2. Gradient Boosting Method

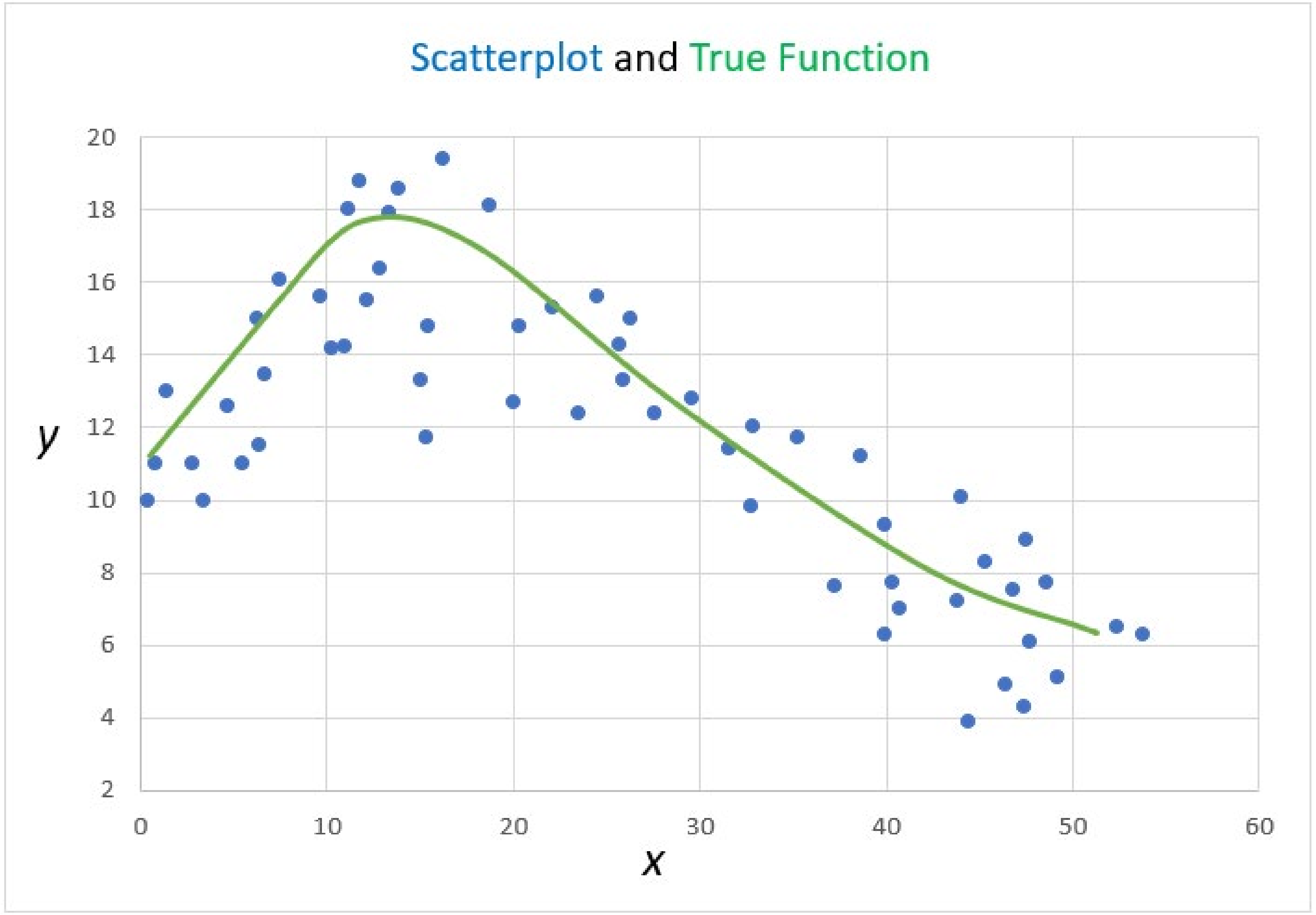
- The **gradient boosting method** combines the method of **gradient descent** (or **steepest descent**) and the **boosting algorithm**.
- It was first introduced by Jerome Friedman in 1999.



Dr. Jerome Friedman, Professor Emeritus, Stanford University

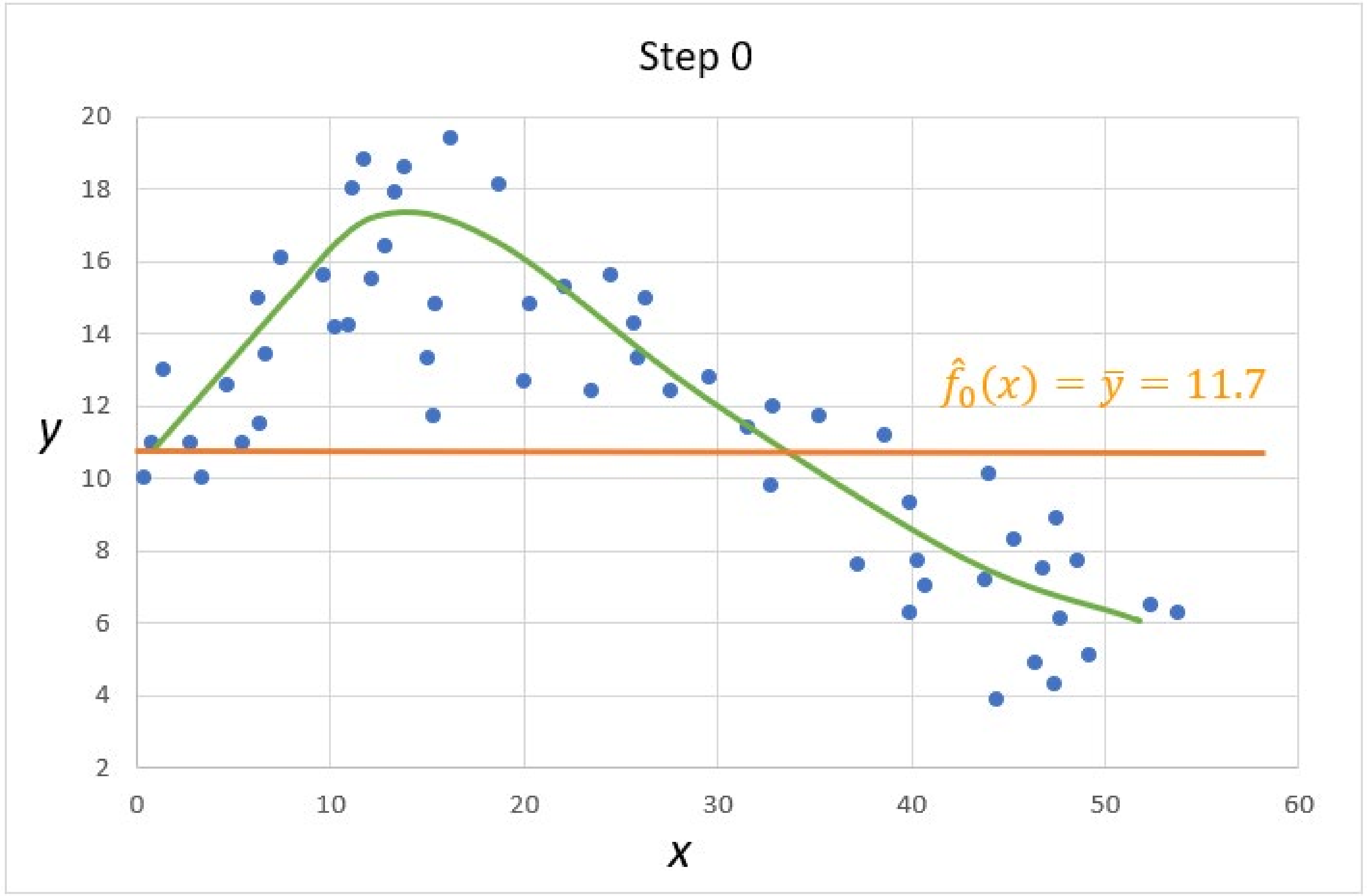
4.2. Gradient Boosting Method - Example

Here we consider a **simple example** to explain how **gradient boosting** works. Suppose y depends on x through a non-linear relationship $y = f(x)$ depicted in the scatterplot below.



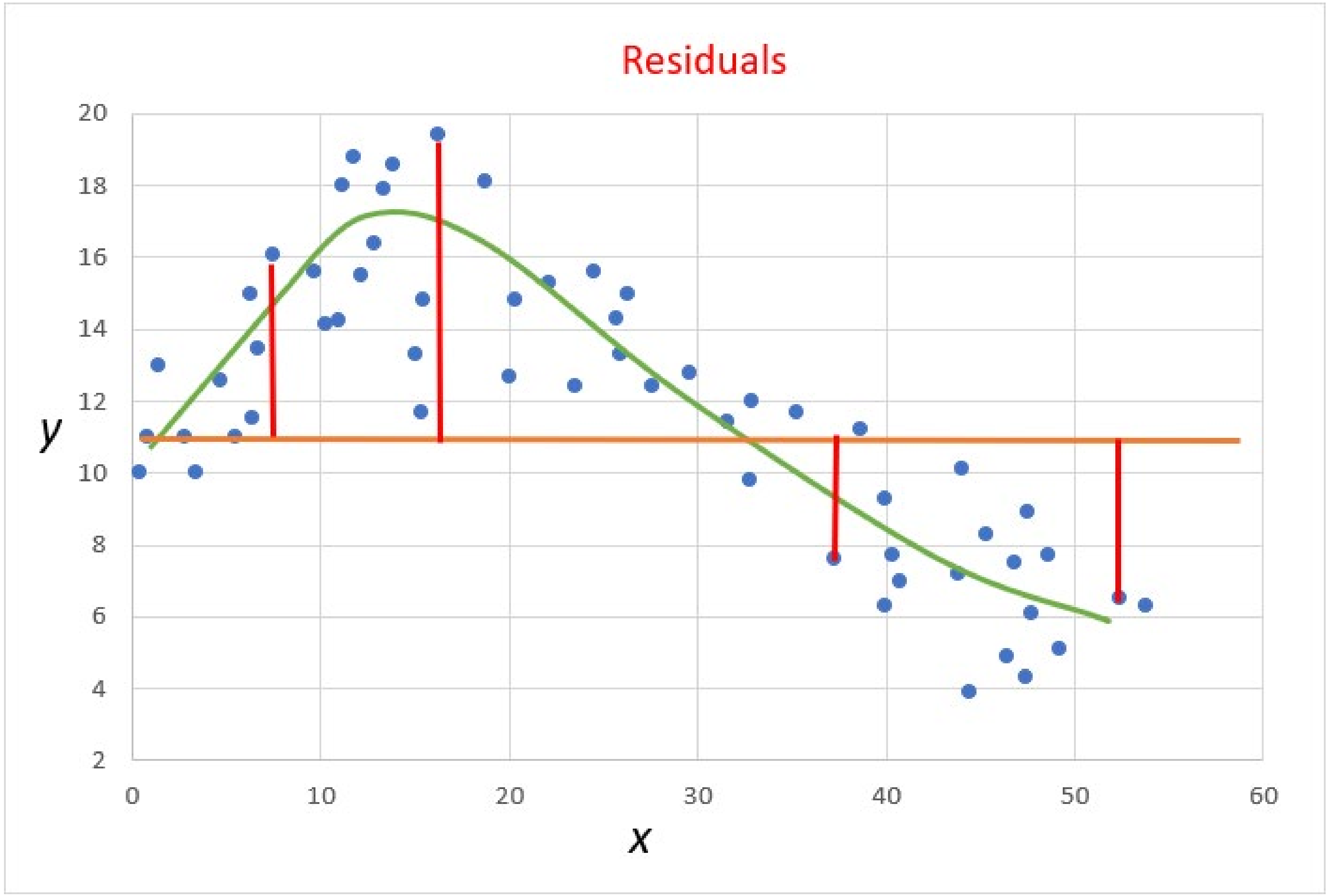
4.2. Gradient Boosting Method – Example (cont.)

We **initially predict** the response y by the sample mean $\bar{y}$, that is, we let $\hat{f}_0(x) = \bar{y}$.



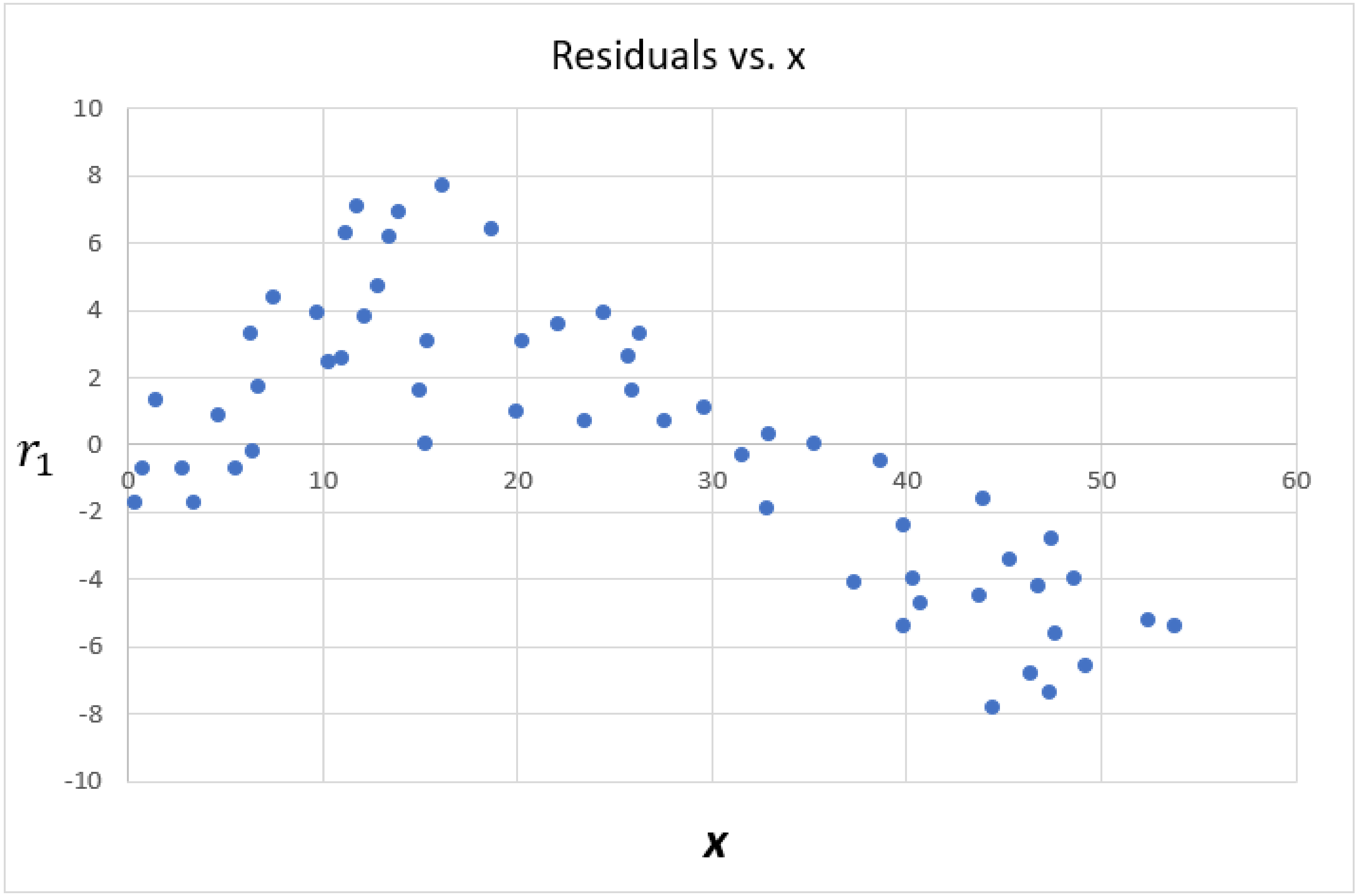
4.2. Gradient Boosting Method – Example (cont.)

To improve our prediction, we will focus on the **residuals** (i.e., the **vertical distances** between the observed y's and the prediction $\bar{y}$). The **residuals** $r_1 = y - \bar{y}$ are shown as the **vertical red lines** in the figure below.



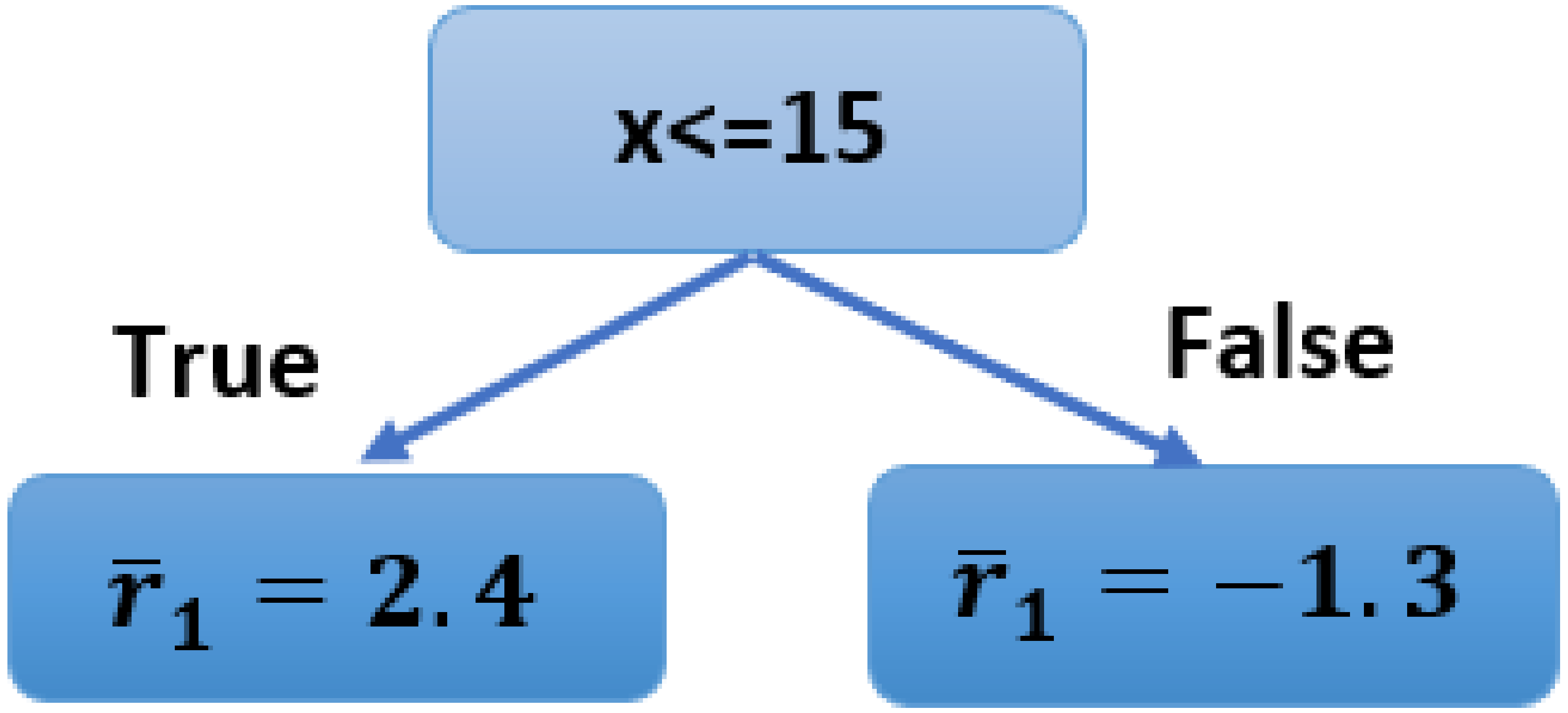
4.2. Gradient Boosting Method – Example (cont.)

Next, we plot the residuals against x.



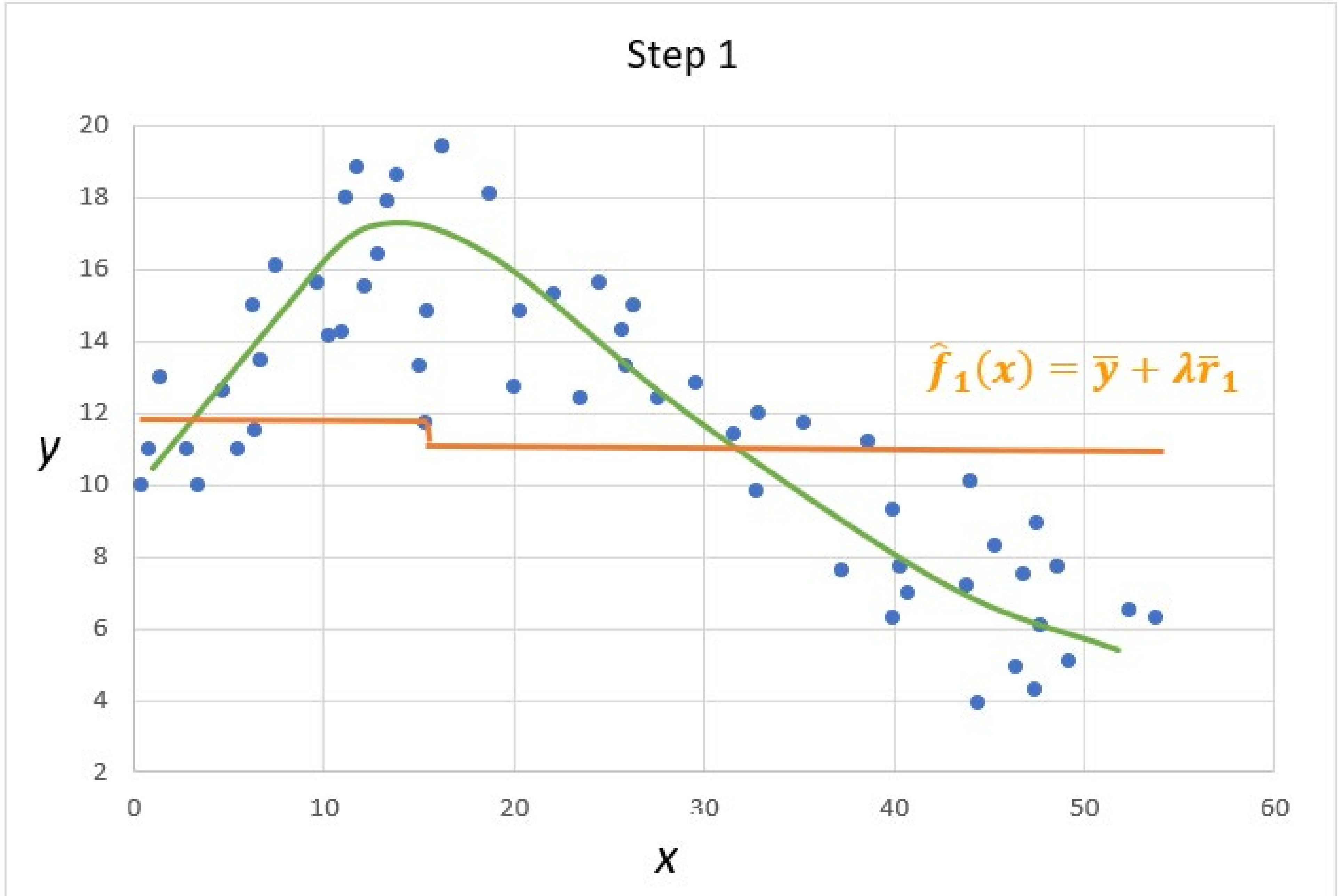
4.2. Gradient Boosting Method – Example (cont.)

In the next step, we use the residuals r_1 as the **target variable** and x as the **feature** and produce a **decision stump** shown in the picture.



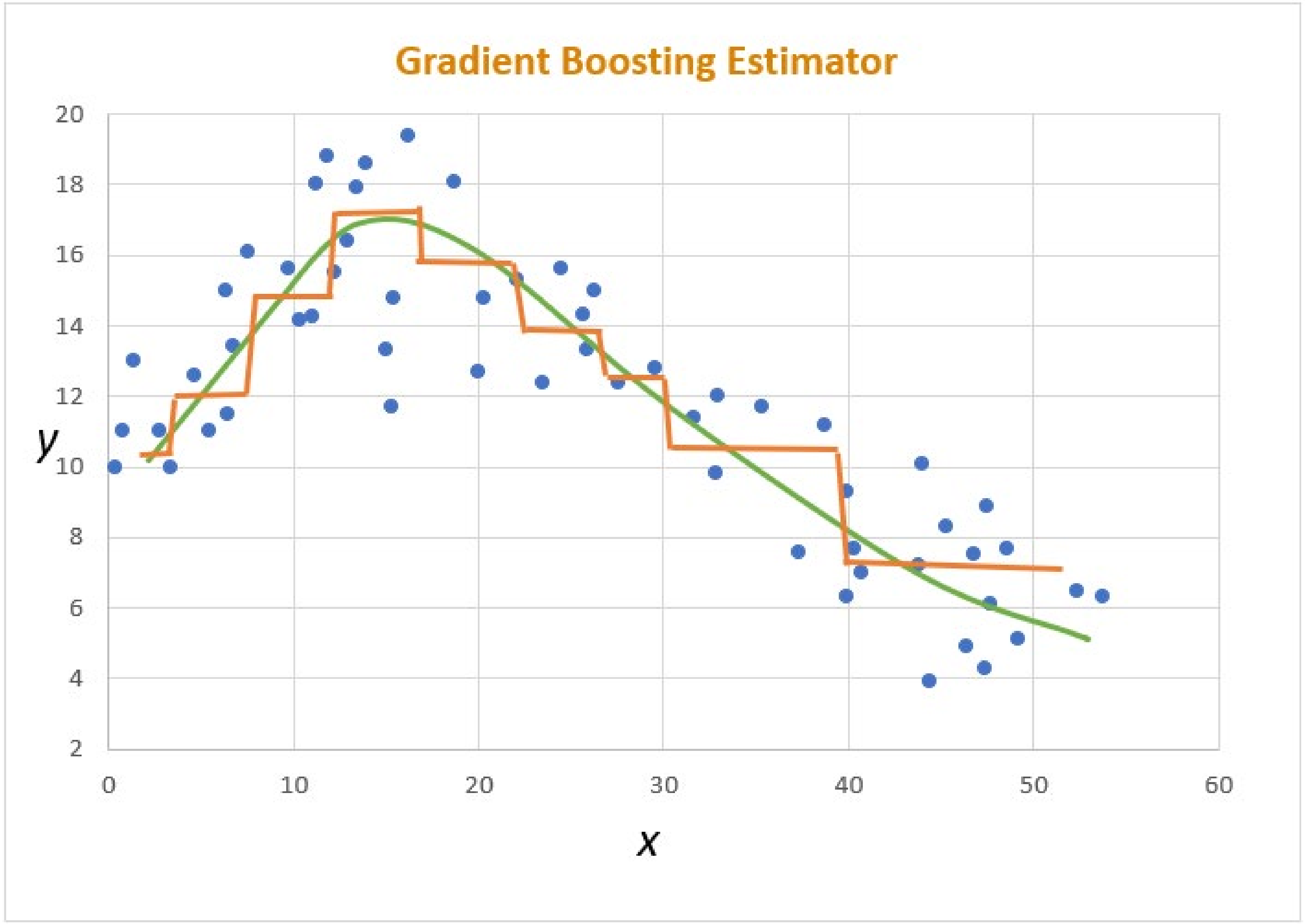
4.2. Gradient Boosting Method – Example (cont.)

Taking the learning rate $\lambda = 0.2$, we update the estimator of f as follows: for $x \leq 15, \hat{f}_1(x) = \hat{f}_0(x) + \lambda\,\bar{r}_1 = \bar{y} + \lambda\,\bar{r}_1 = 11.7 + (0.2)(2.4) = 11.8$, and for $x > 15$, $\hat{f}_1(x) = 11.7 + (0.2)(-1.3) = 11.0$.



4.2. Gradient Boosting Method – Example (cont.)

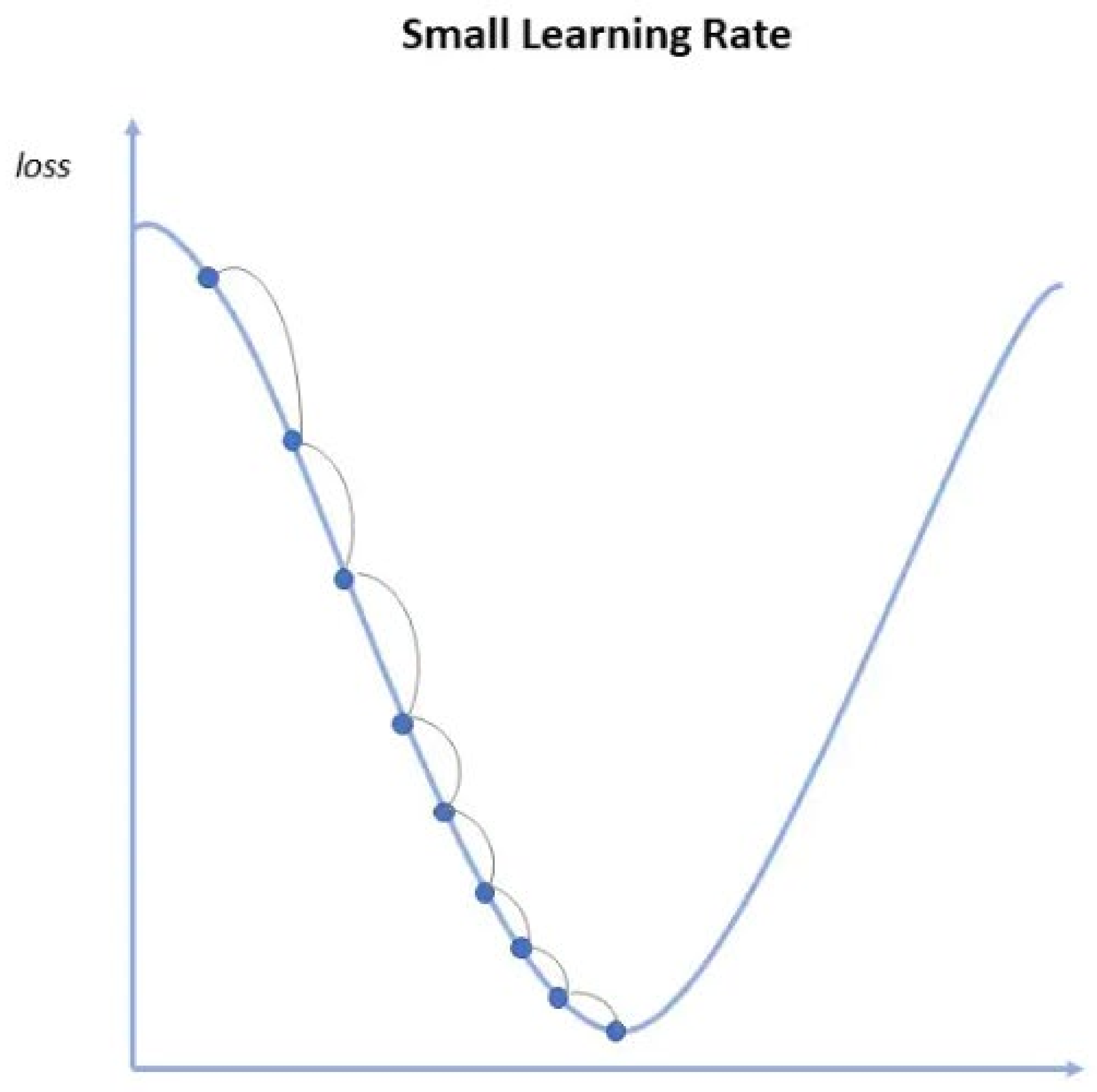
- In the next step, we update the residuals to $r_2 = y - \hat{f}_1(x)$ and build a regression tree, which will give us another split and another pair of estimates $\bar{r}_2$. We then update the fitted function $\hat{f}_2(x) = \hat{f}_1(x) + \lambda\, \bar{r}_2$.
- We iterate these steps until the model prediction stops improving.

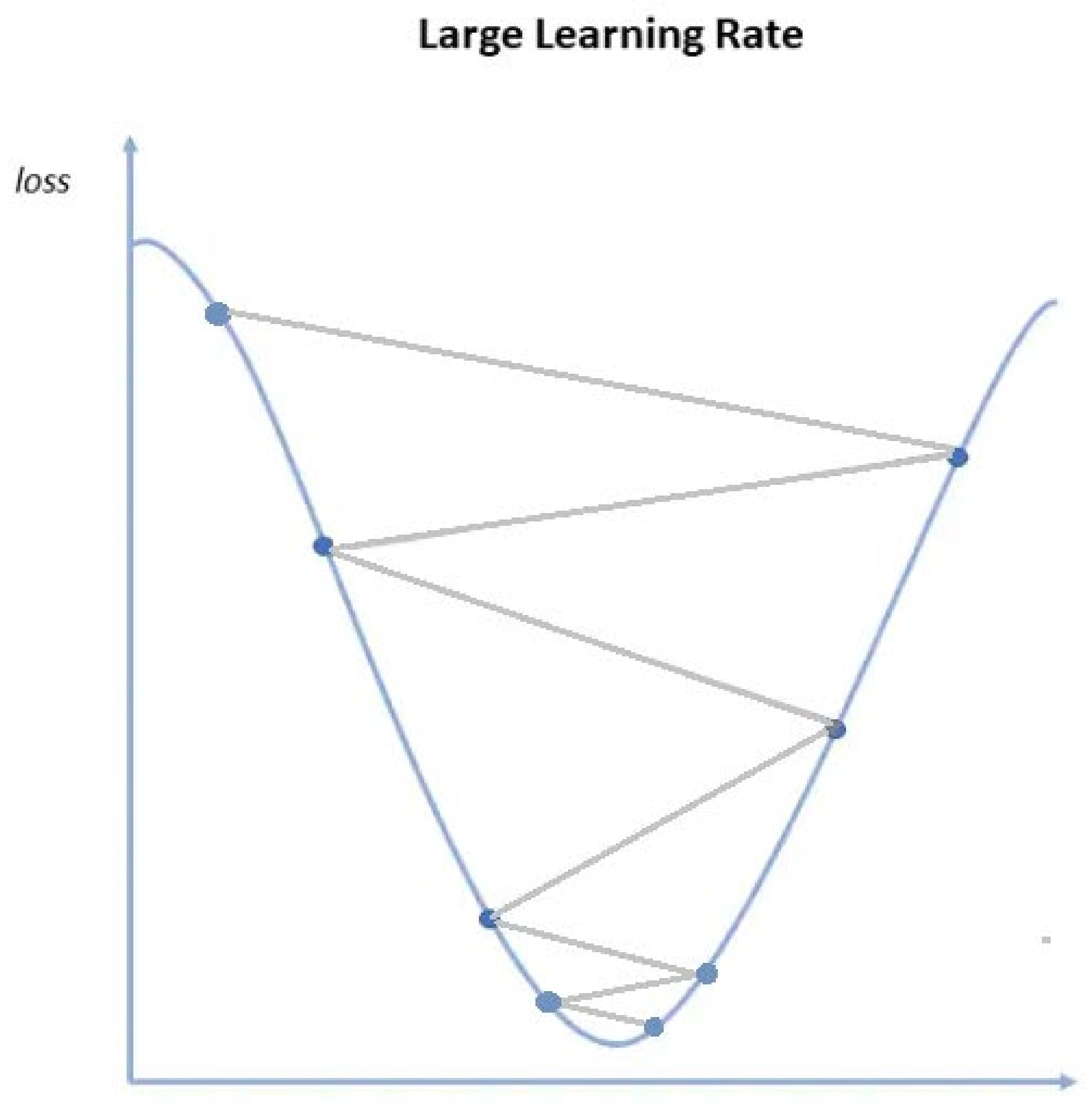


4.2. Gradient Boosting Method (cont.)

- The **method of gradient boosting** is closely related to the method of **gradient descent** (or **steepest descent**), which is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function.
- The idea is to take repeated steps in the opposite direction of the **gradient of the function** at the current point because this is the **direction** of the **steepest descent**.
- If the **learning rate** (**length of step**) is small, we would descend **slowly** along **one slope**.
- However, we can choose to take **large learning steps**. The **convergence** is still **guaranteed** but it will take more time and computations to reach the minimum.

4.2. Gradient Boosting Method (cont.)





4.2. Gradient Boosting Method – Coding Examples

- Gradient Boosting Regression – Housing Data
- Gradient Boosting Binary Classification – Pneumonia Data
- Gradient Boosting Multinomial Classification – Movie Data

5. Support Vector Machine

- The **Support Vector Machine (SVM)** algorithm is a machine learning tool for regression and classification.

Historical Note.
An SVM method was first proposed by Vladimir Vapnik in his book "The Nature of Statistical Learning Theory", Springer-Verlag, New York, 1995.

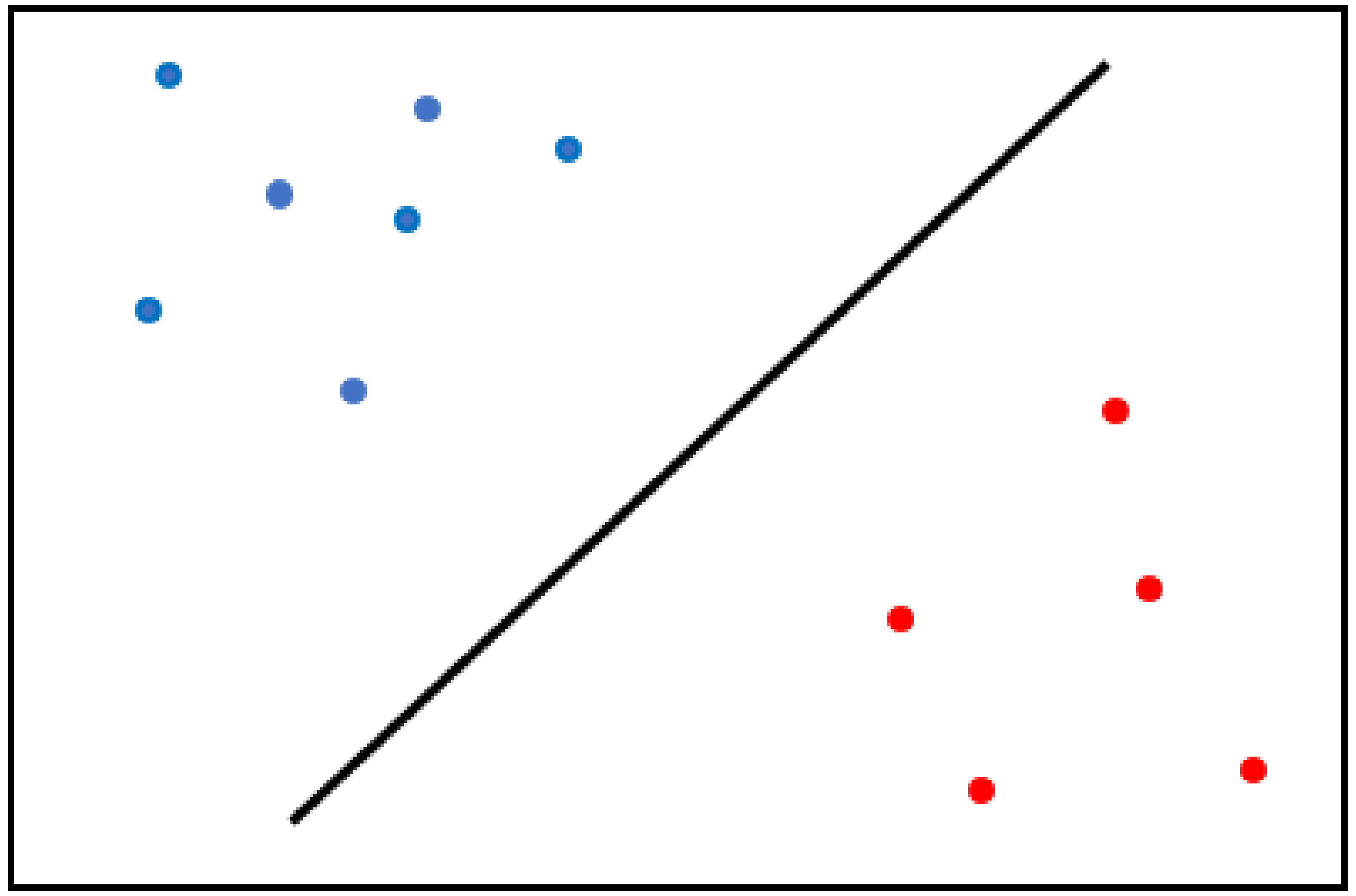
5. Support Vector Machine (cont.)

- The goal of an SVM is to find a **function** $f(x_1, \ldots, x_k)$ that deviates from the observed response y by a value **not greater than** a **pre-specified** amount for each **training point**, and at the same time is **as flat as possible**.

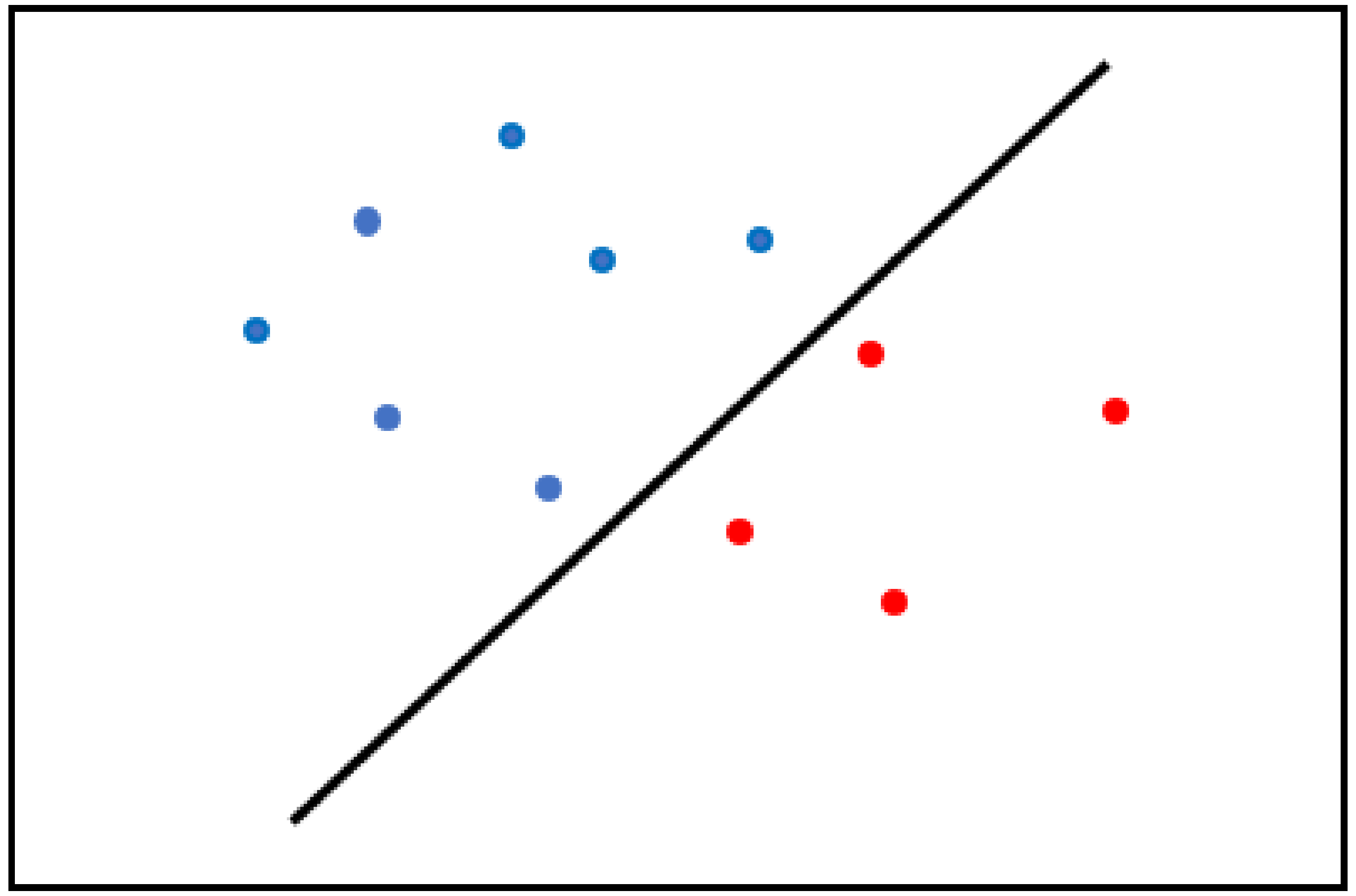
  <Fast forward 5 pages of algebra, involving minimization of Lagrangian, subject to certain constraints. The Karush-Kuhn-Tucker (KKT) complementarity conditions are used as optimization constraints.>

- An **ideal** situation is when two groups of data points can be **separated** by a **straight line**. Moreover, if the groups are **far away** from each other, then it is **easy to do**, but what if these groups of points lie **very close** to each other?

5. Support Vector Machine (cont.)

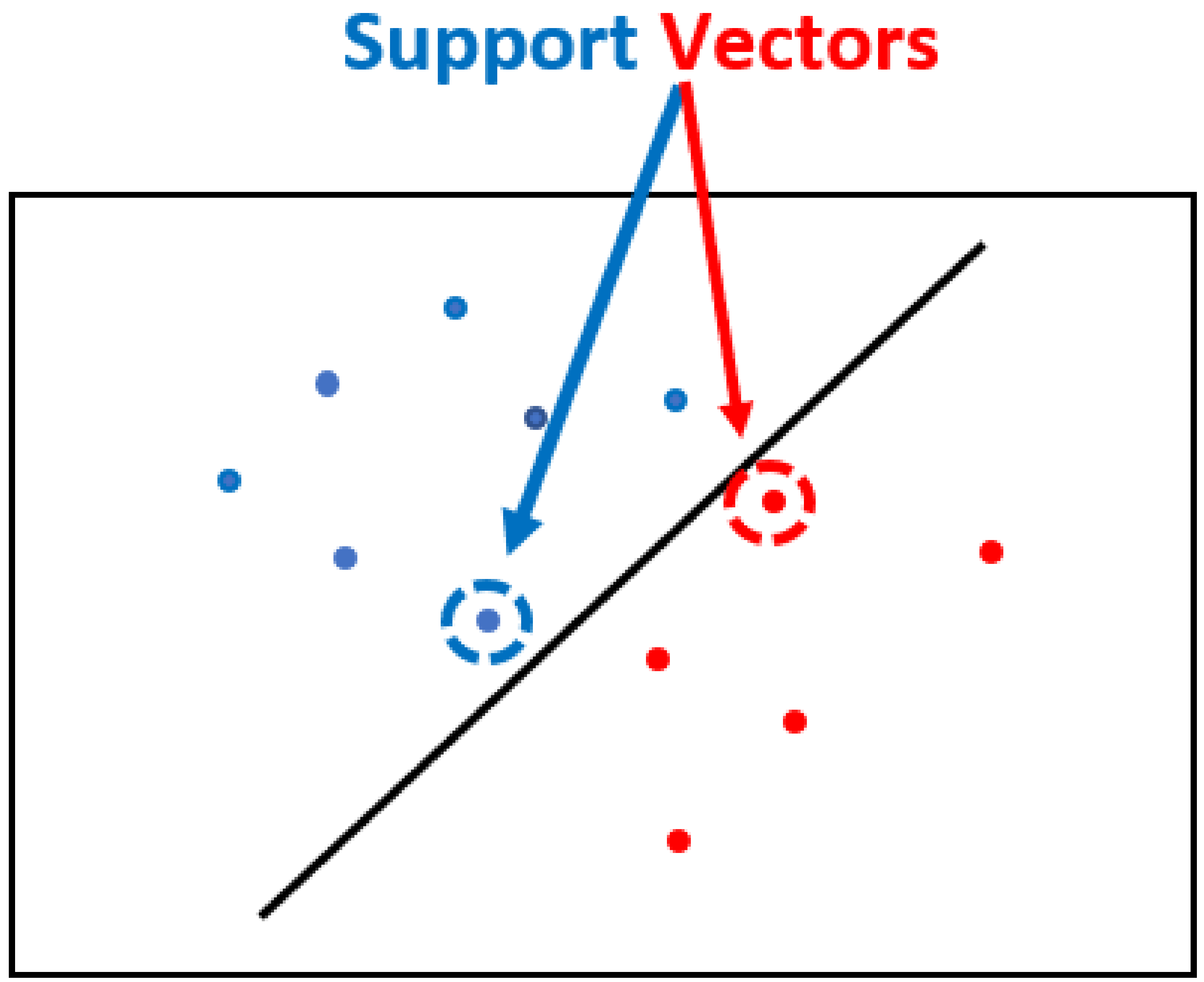


Groups of points are far away from each other – easy to separate with a linear function.



Groups of points are close together – harder to separate with a linear function.

5. Support Vector Machine (cont.)



Support Vectors are data points that lie the closest to the separating line. Hence, the **name** of the technique.

5. Support Vector Machine (cont.)

- For some data sets, classes cannot be separated by a **linear hyperplane** of the form $\sum_{i=1}^{k} w_i x_i = a$ where $w_1, \dots, w_k$ and a are some coefficients.
- In those situations, a **kernel trick technique** is utilized. A **kernel-based separating hyperplane** may be chosen from the following three:

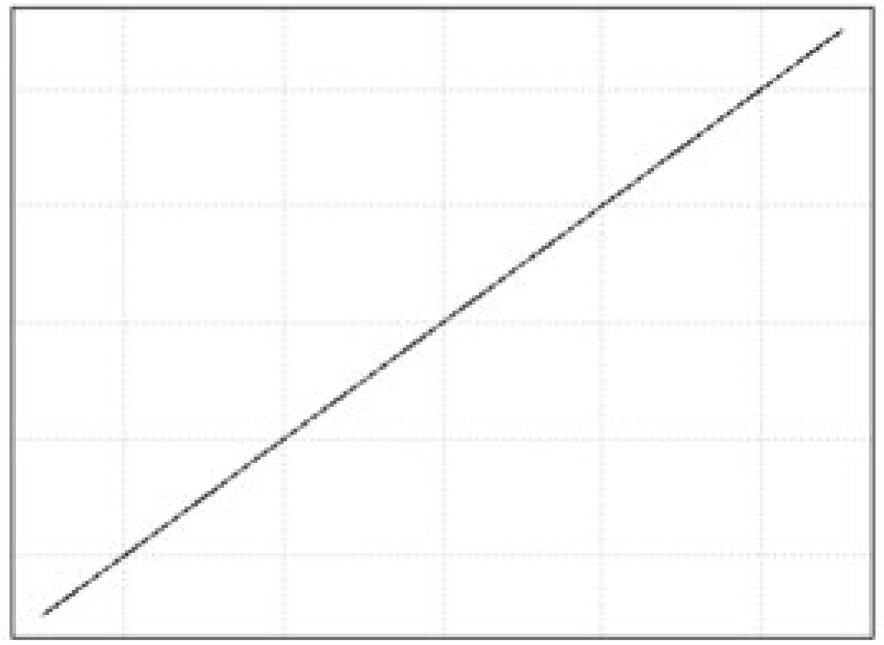
  - **Polynomial:** $\sum_{i=1}^{k} w_i x_i^d = a$ where $d > 1$,
  - **Radial Basis Function (RBF) (or Gaussian):** $\exp\left(-\sum_{i=1}^{k} w_i x_i^2\right) = a$,

  or

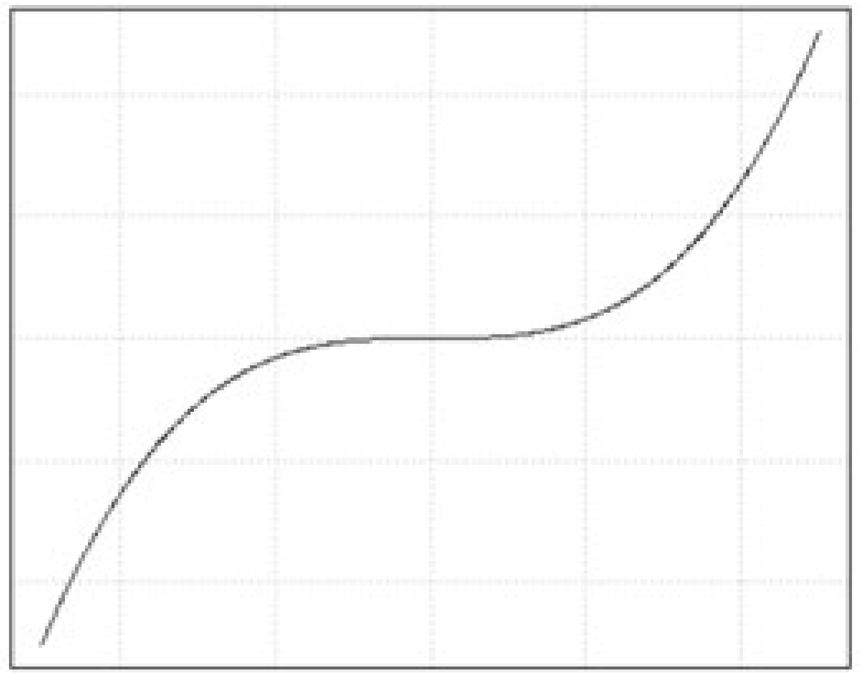
  - **Hyperbolic Tangent (or Sigmoid):** $\tanh\left(\sum_{i=1}^{k} w_i x_i^2\right) = a$.
  - $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}, -\infty < x < \infty$.

5. Support Vector Machine (cont.)

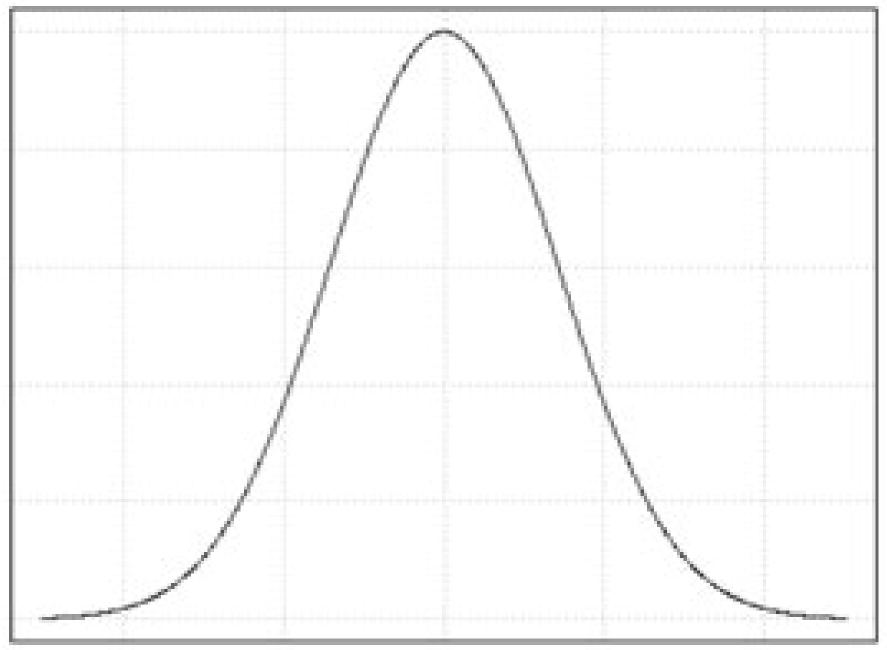
Linear Kernel



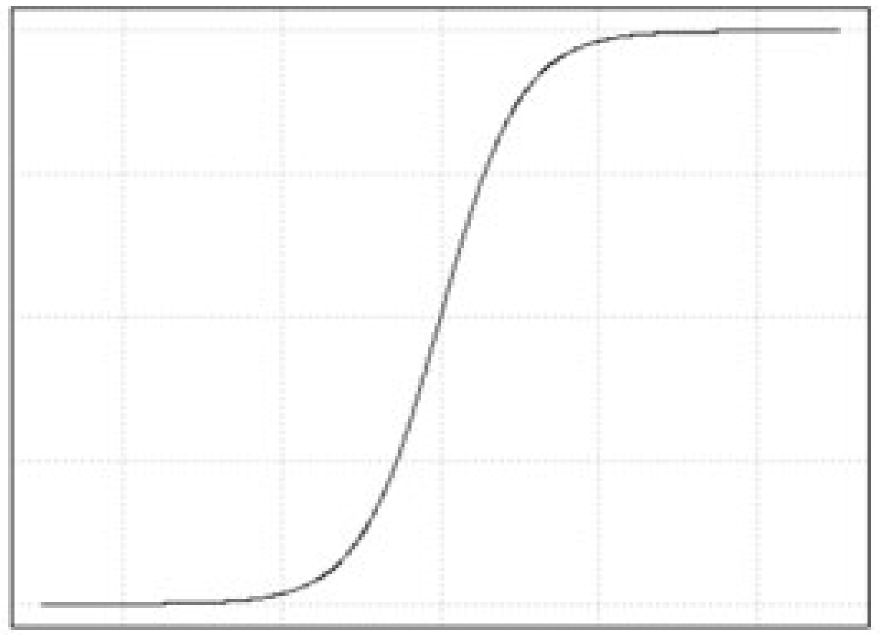
Polynomial Kernel



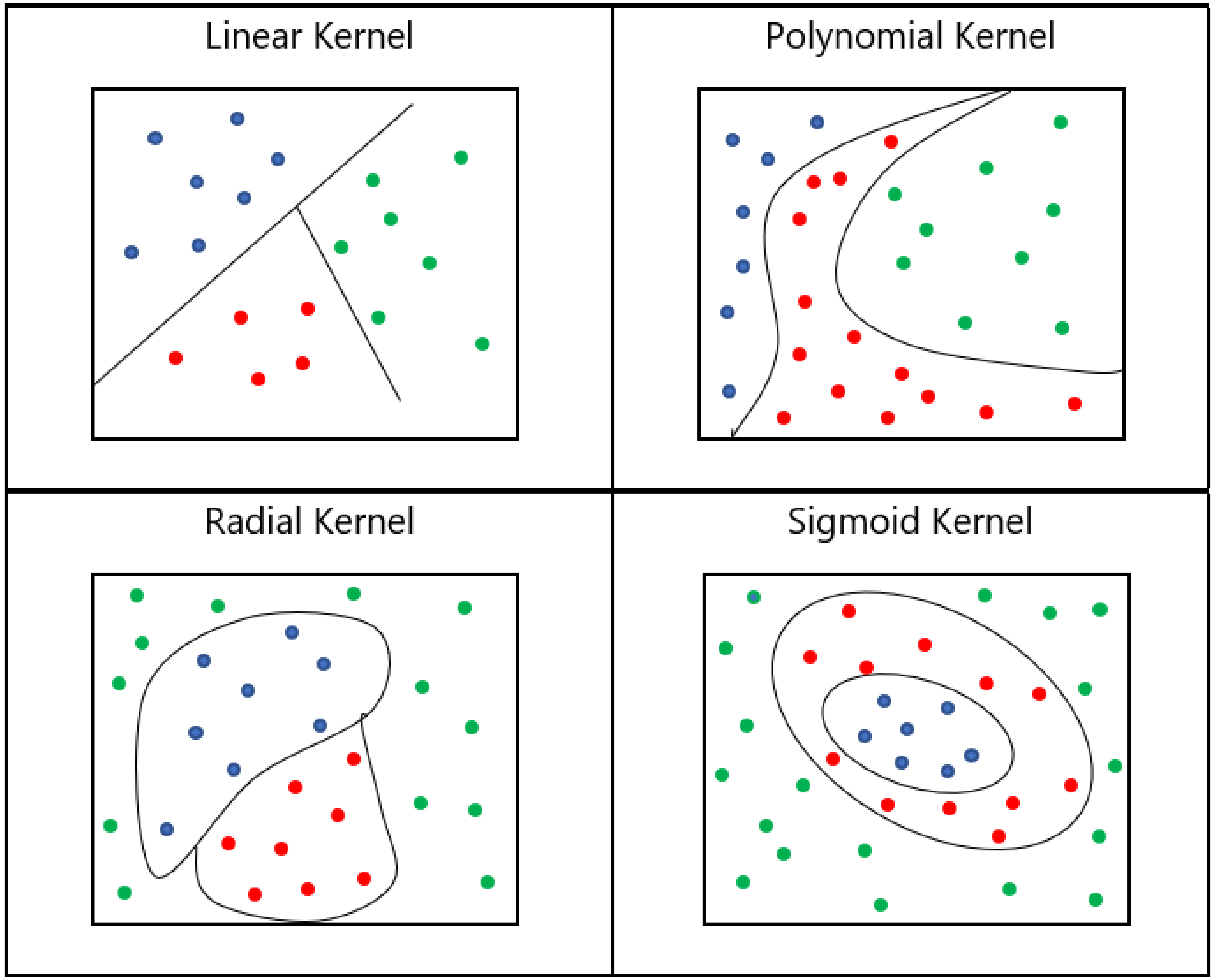
Radial Kernel



Sigmoid Kernel



5. Support Vector Machine (cont.)



Note. Some points will be misclassified as they fall in the wrong group.

5. Support Vector Machine – Coding Examples

- Support Vector Machine Regression – Housing Data
- Support Vector Machine Binary Classification – Pneumonia Data
- Support Vector Machine Multinomial Classification – Movie Data

6. k-Nearest Neighbor

- "A man is known for the company he keeps." - a proverb.
- In the $\boldsymbol{k}$ **Nearest-Neighbor** (**kNN**) **algorithm**, the space is divided into classes with k **nearest neighbors** in each **class**.
- The **Euclidean distance** is used to measure the distance between neighbors.
- The **Euclidean distance** between two d-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_d)$ and **y**$= (y_1, \ldots, y_d)$ in our regular **Euclidean geometry** is defined as

$$distance(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}.$$

6. k-Nearest Neighbor (cont.)

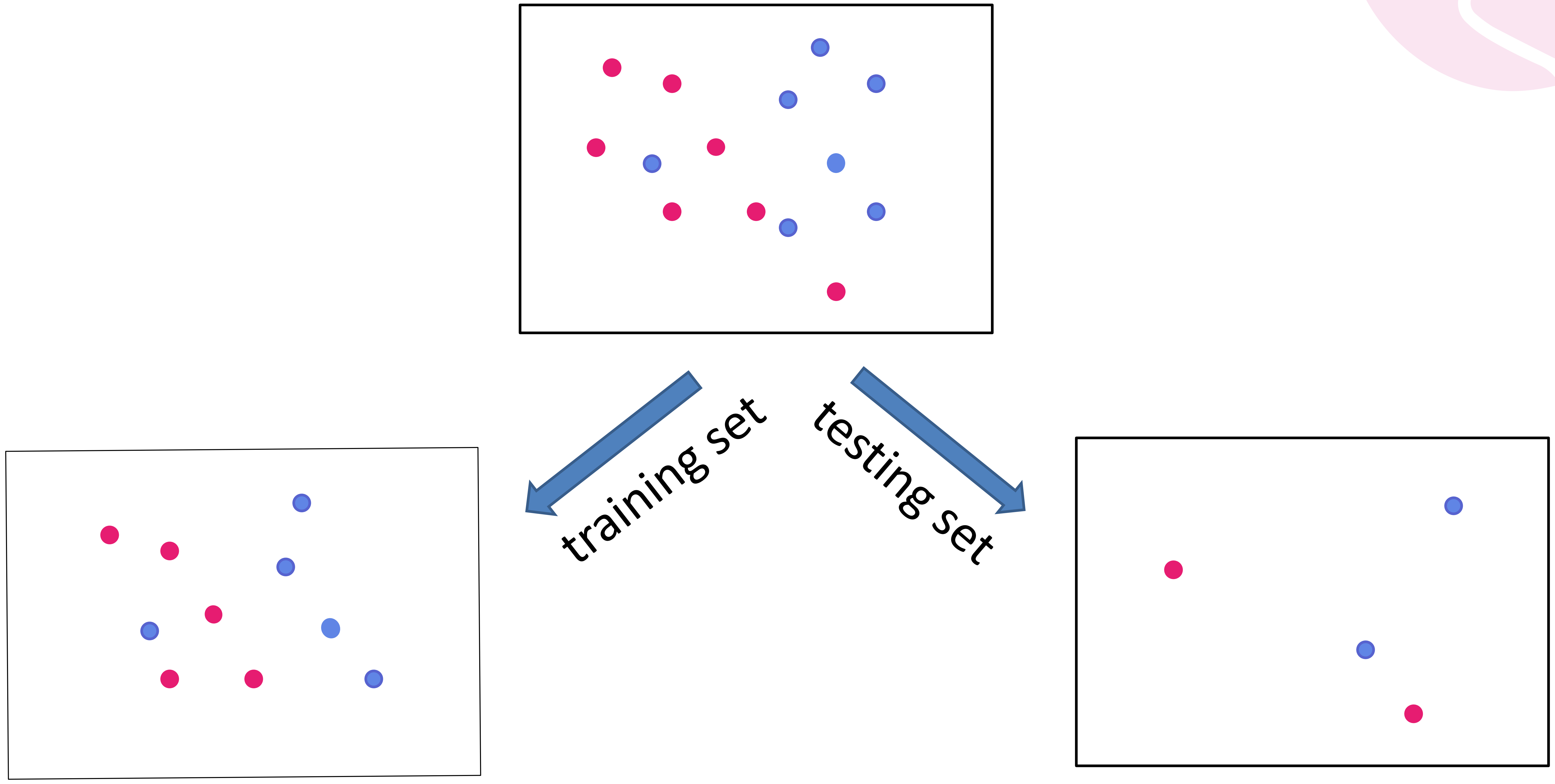
Historical Note.

- **The kNN algorithm** was first described in “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”, by Evelyn Fix and Joseph Hodges, Technical Report 4, USAF School of Aviation Medicine, Randolph Field, TX, 1951.
- Later Thomas Cover and Peter Hart gave the **statistical foundation** for this method in “Nearest neighbor pattern classification”. IEEE Transactions on Information Theory, vol. IT-13, no. 1, pp. 21–27, 1967.

6. k-Nearest Neighbor (cont.)

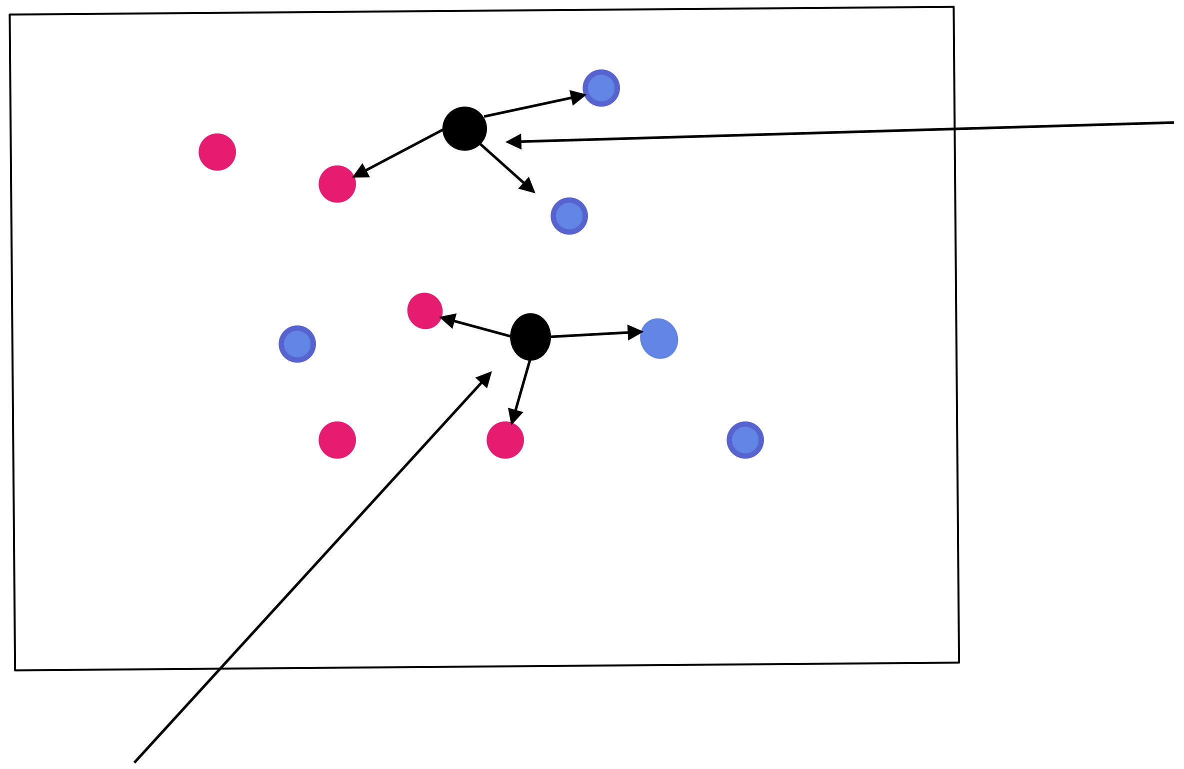
- For **regression**, the **mean value** of the class is used for prediction.
- In **classification**, the class with the **majority** of observations is assigned as the predicted class.
- For classification, it is wise to consider an **odd number of neighbors** to avoid **ties**.

6. k-Nearest Neighbor - Example



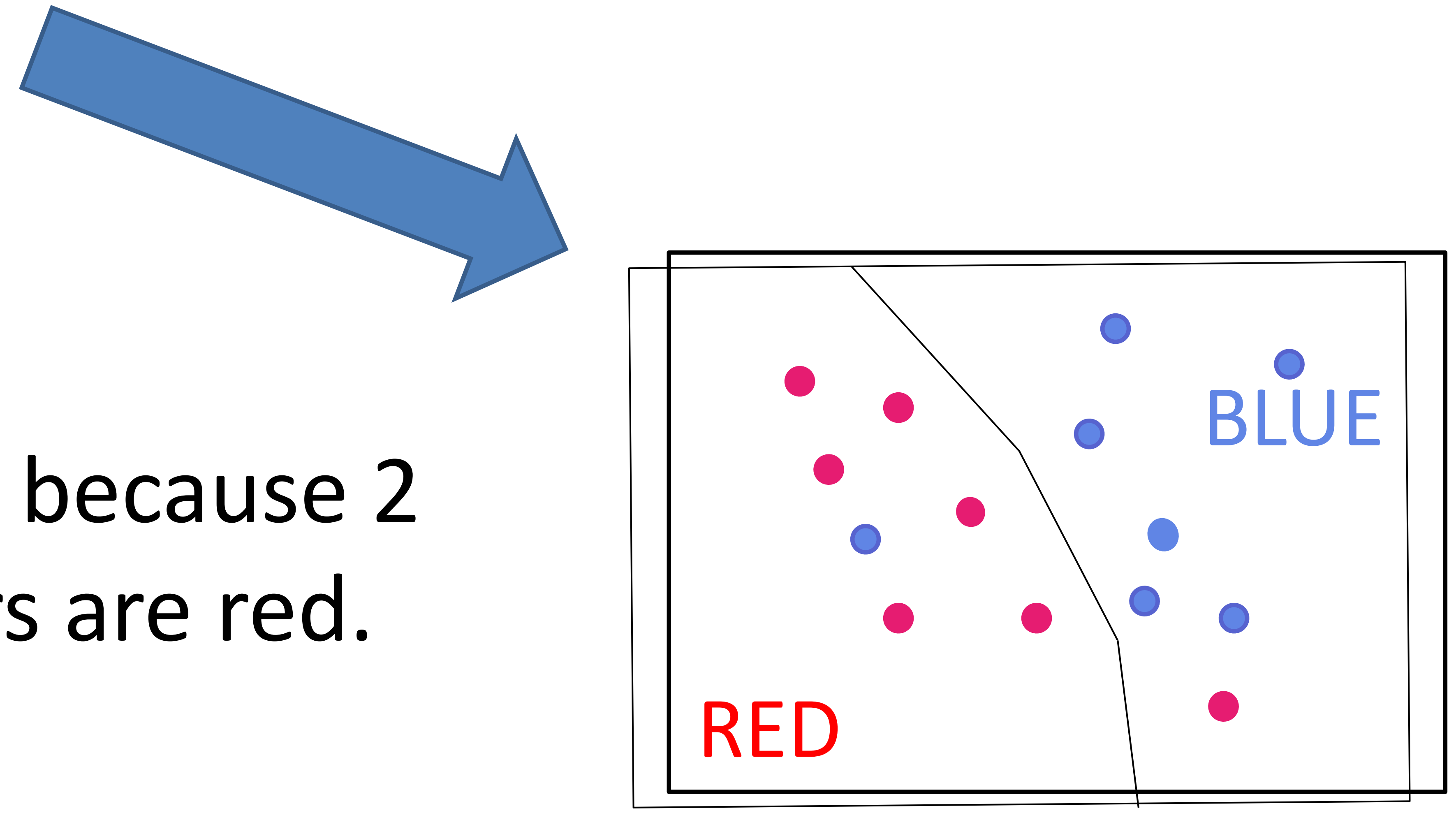
6. k-Nearest Neighbor (cont.)

training set, $k = 3$



This point is classified as blue because 2 out of its 3 nearest neighbors are blue.

This point is classified as red because 2 out of its 3 nearest neighbors are red.



- In the testing set, one red point will be misclassified as blue.

5. Support Vector Machine – Coding Examples

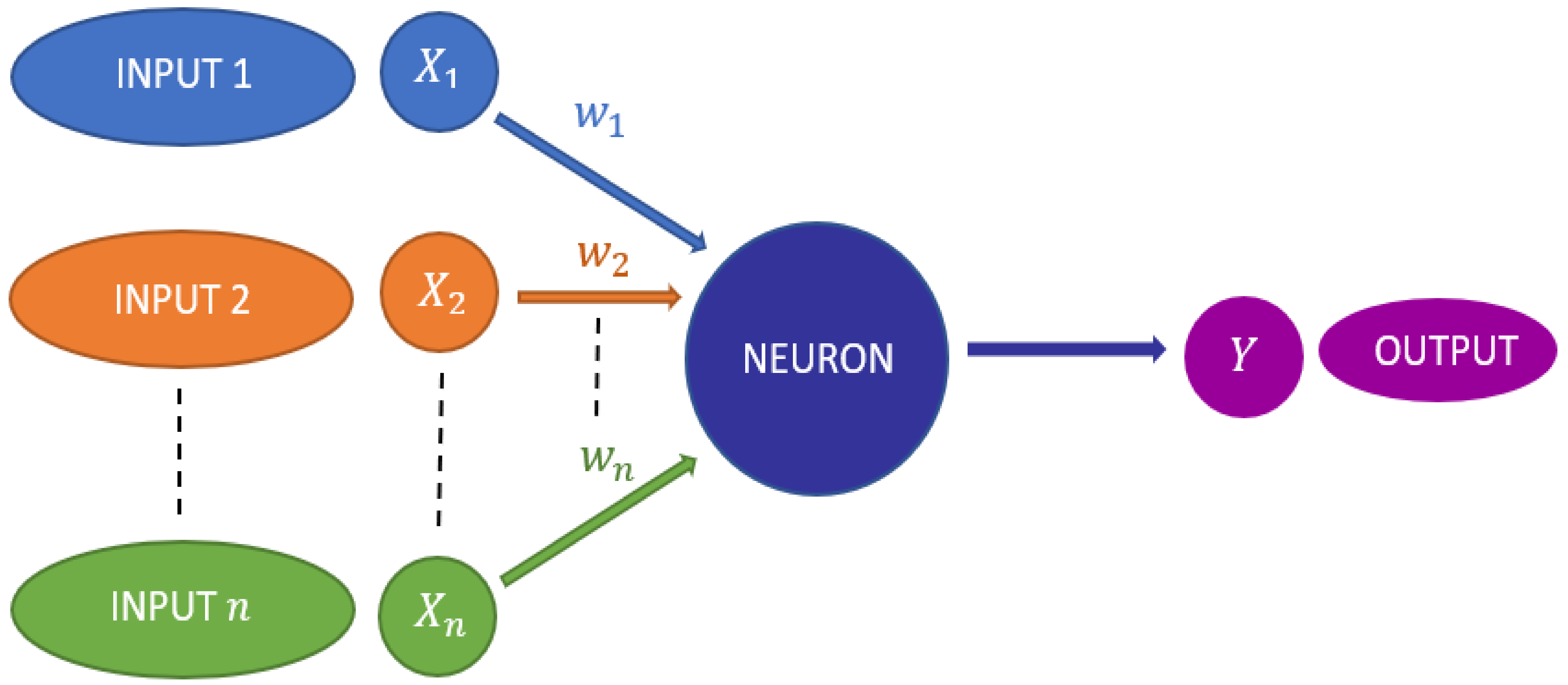
- k-Nearest Neighbor Regression – Housing Data
- k-Nearest Neighbor Binary Classification – Pneumonia Data
- k-Nearest Neighbor Multinomial Classification – Movie Data

7. Artificial Neural Network

- An **Artificial Neural Network** (ANN) attempts to mimic the **network of neurons** that makes up a **human brain** so that computers will have the option to understand things and make decisions in a **human-like manner**.
- An ANN consists of an **input layer**, **hidden layers of nodes** (or **neurons**, or **perceptrons**), and an **output layer**.
- The **input layer** receives the **raw input**, which is then processed by **multiple hidden layers**, and the **output layer** produces the prediction.
- Every **neuron** in the **hidden layers** is a **linear combination** of the input variables with some **weights**.

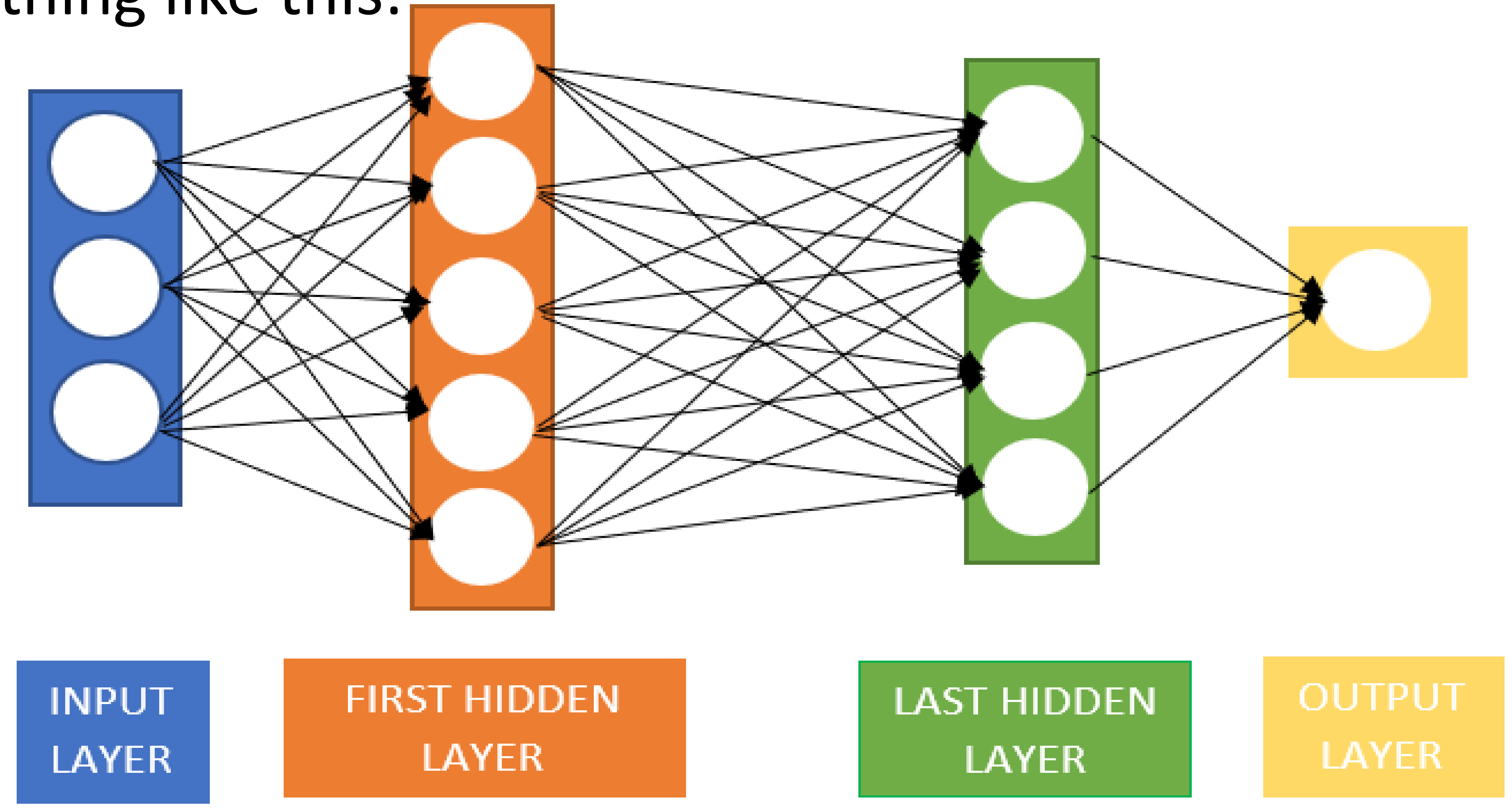
7. Artificial Neural Network (cont.)

- An ANN with **one hidden layer** and a **single neuron** looks something like this:



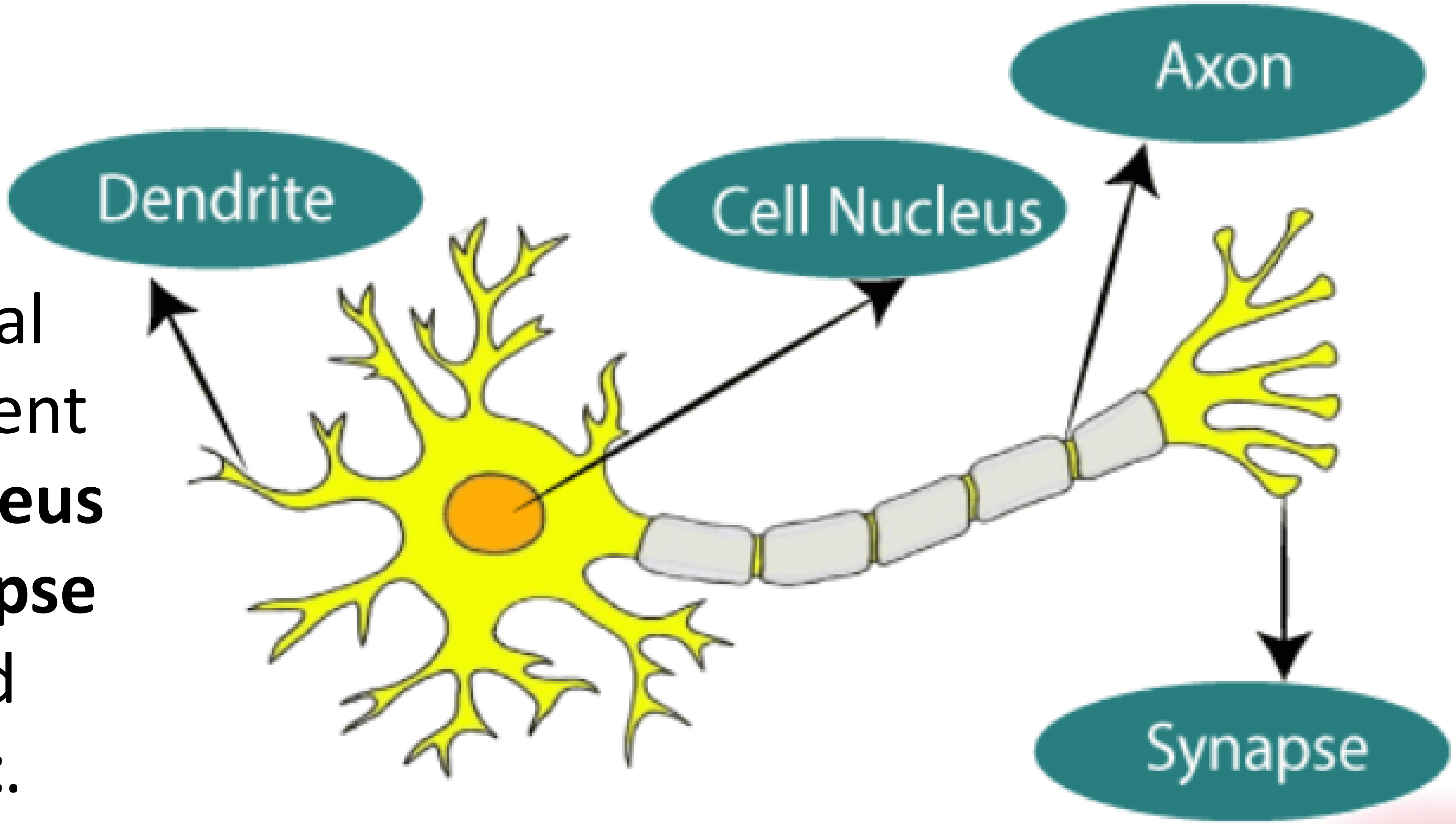
7. Artificial Neural Network (cont.)

- More generally, an ANN with **multiple hidden layers** would look something like this:



7. Artificial Neural Network (cont.)

- A typical diagram of a **biological neural network** in the brain looks like this:



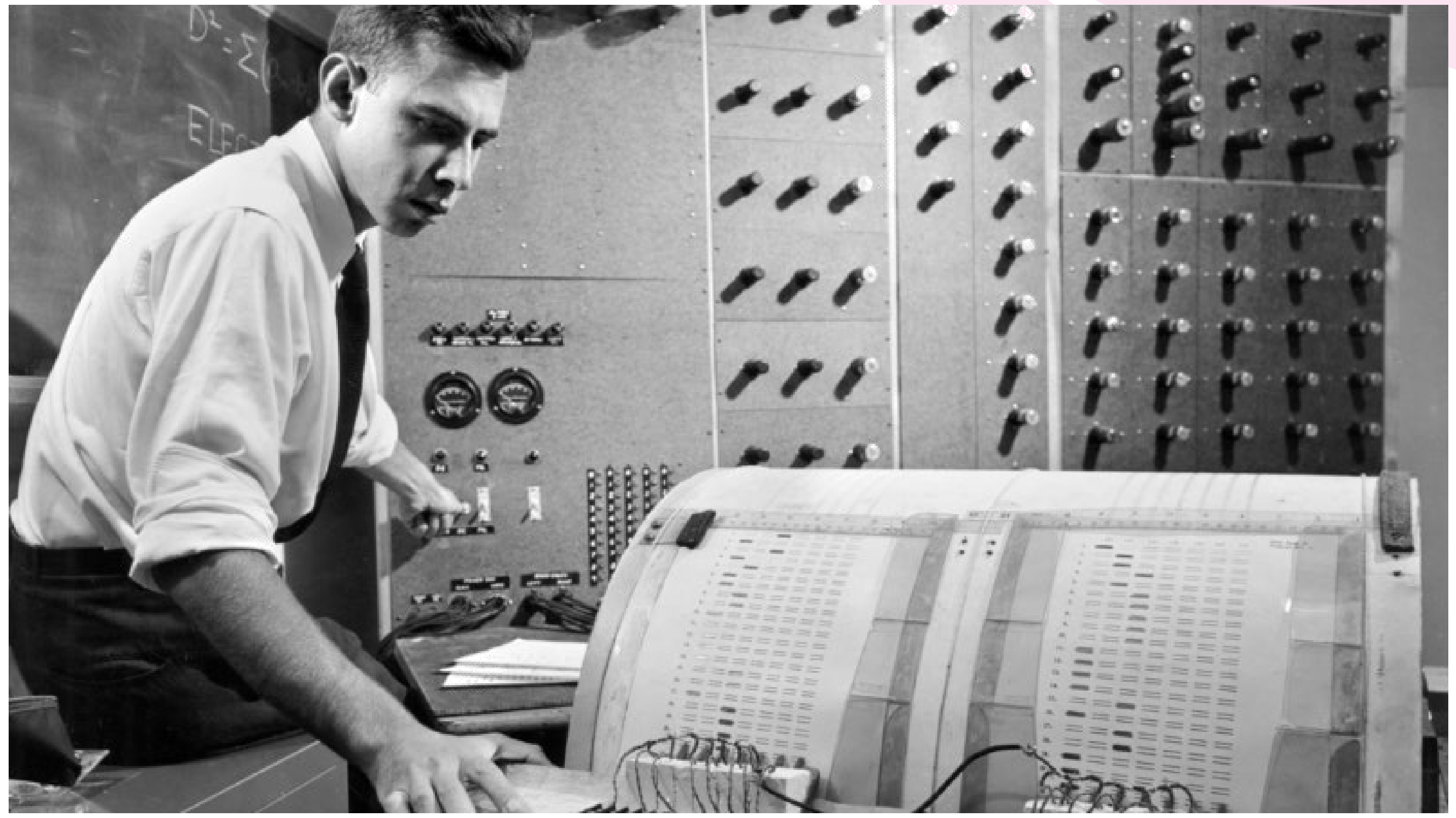
- **Dendrites** from biological neural networks represent **inputs** in ANN, **cell nucleus** represents **nodes**, **synapse** represents **weights**, and **axon** represents **output**.

7. Artificial Neural Network (cont.)

CORNELL CHRONICLE

Historical Note.

The **oldest type** of neural network, known as **Perceptron**, was introduced by Frank Rosenblatt (1928-1971) in 1958.



Division of Rare and Manuscript Collections

Frank Rosenblatt '50, Ph.D. '56, works on the "perceptron" – what he described as the first machine "capable of having an original idea."

Professor's perceptron paved the way for AI – 60 years too soon

Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review,* 65(6), 386–408.

7. Artificial Neural Network (cont.)

Theoretical Foundation.

- In an ANN, the **input** goes through a **series of transformations** using the **hidden layers**, which finally result in the **output** expressed as a **linear combination** of **weighted input features** with a **bias term** included.
- After producing the **output**, an **error** (or **loss**) is calculated and a **correction** is sent back to the network. This process is known as **back** (or **backward) propagation**.

Historical Note.

- An ANN with **back propagation** was introduced in Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). “Learning representations by back-propagating errors”. *Nature*, 323(6088), 533-536.

7. Artificial Neural Network (cont.)

- An ANN starts with a set of **initial weights** and then gradually modifies the weights during the **training cycle** to settle down to a **set of weights** capable of realizing the **input-output mapping** with a **minimum error**.
- Denote by $\mathbf{x}_i = (x_{i1}, \dots, x_{ik})', i = 1, \dots, n$, the set of vectors of **input variables**, and let $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n)'$ be the **output vector**. Also, suppose there is one hidden layer with m **neurons** $h_1, \dots, h_m$.
- The **response** of the **hidden layer** for the *i*th individual is the vector $\mathbf{h}_i = (h_{i1}, \dots, h_{im})'$.

7. Artificial Neural Network (cont.)

- An ANN produces outputs governed by the relations:

$$\mathbf{h}_i = f(\mathbf{W}_h \mathbf{x}_i + \mathbf{b}_i) \quad \text{and} \quad \hat{y}_i = f(\mathbf{W}_i^* \mathbf{h}_i + b_i^*),$$

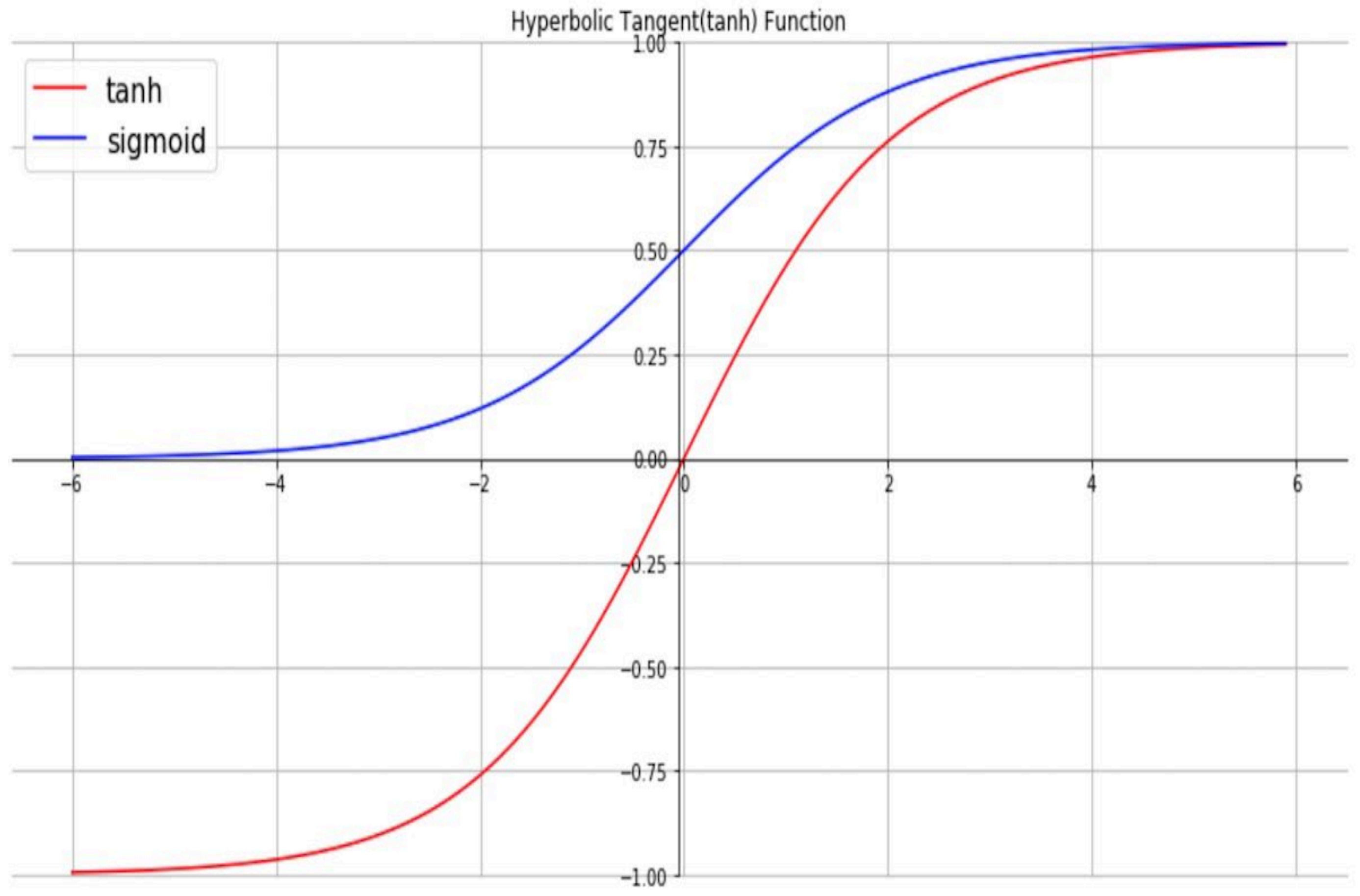
  where f is the **activation function**, $\mathbf{W}_h = \begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mk} \end{bmatrix}$ is the **hidden layer weight matrix**, $\mathbf{W}_i^* = (w_{i1}^*, \dots, w_{im}^*)$ is the vector of **output weights** for the *i*th individual, $\mathbf{b}_i = (b_{i1}, \dots, b_{im})'$ is the **hidden layer bias** vector for the *i*th individual, and b_i^* is the **output layer bias** for the *i*th individual.

7. Artificial Neural Network (cont.)

- The **activation functions** that can be employed in R are:
  - **logistic** $f(x) = \frac{\exp(x)}{1+\exp(x)}\ , -\infty < x < \infty,$

and

  - **hyperbolic tangent** (or **tanh**) $f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}, -\infty < x < \infty.$



7. Artificial Neural Network (cont.)

- The **method of steepest descent** is used to update the weights**.** They are updated according to the recursive relation

$$w_{ij}^{*}(new) = w_{ij}^{*}(old) - \lambda \frac{\partial L}{\partial w_{ij}^{*}(old)}, j = 1, \dots, m,$$

where λ represents the **learning rate**, and L is the **loss function**.

- The same algorithm applies to the weights in the hidden layers $\mathbf{W}_h$.

7. Artificial Neural Network (cont.)

- For an **ANN regression**, the **loss function** is the **mean squared error**

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{W}_i^*\mathbf{h}_i + b_i^*)\right)^2.$$

- For an **ANN classifiers**, the **loss function** is the **binary cross-entropy** or **multi-class cross-entropy** defined as

$$E = -\sum_{k=1}^{K} p_k \ln(p_k)$$

where p_k is the **proportion of observations** in the *k*th class.

7. Artificial Neural Network – Coding Examples

- ANN Regression – Housing Data
- ANN Binary Classification – Pneumonia Data
- ANN Multinomial Classification – Movie Data

8. Change-point Detection

- Consider **a time series data set** consisting of n **normally distributed observations**. And suppose that the first segment of k observations has a $N(\mu_1, \sigma^2)$ distribution, whereas the remaining segment of $n-k$ observations has a $N(\mu_2, \sigma^2)$ distribution where $\mu_1 \neq \mu_2$. That is, a **change in mean** occurs at some **unknown step** k.
- The **change-point detection** is a collection of methods to identify the value of k.
- The change-point detection problem is also applicable to finding k when the mean doesn't change but the **variance** does, or when **both mean and variance** change.

8. Change-point Detection (cont.)

- The **change-point detection** method also extends to the case of **several segments** with different means, variances, or both.
- There are many well-developed methods to identify the point(s) of change. We will present the **theory** for **the most basic approach**.
- The **method of binary segmentation** is often used to detect the change points. First, **one change point** is detected in the **complete set of observations**, then the **series is split** around **this change point**, and the algorithm is applied to the two resulting segments. The process continues until a pre-specified number of splits is detected.

8. Change-point Detection (cont.)

- To identify the value of k where the change occurs, the method of **maximum likelihood estimation** is employed.
- We assume that $y_1, \dots, y_k \sim N(\mu_1, \sigma^2)$ and $y_{k+1}, \dots, y_n \sim N(\mu_2, \sigma^2)$.
- The maximum likelihood estimators of $\mu_1, \mu_2,$ and σ^2 *are*

$$\hat{\mu}_1 = \bar{y}_1 = \frac{1}{k}\sum_{i=1}^{k} y_i\,, \qquad \hat{\mu}_2 = \bar{y}_2 = \frac{1}{n-k}\sum_{i=k+1}^{n} y_i\,,$$

$$\text{and} \quad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(y_i - \frac{\sum_{j=1}^{n} y_j}{n}\right)^2 .$$

8. Change-point Detection (cont.)

- The **likelihood function** for these data has the form
$$L\left(\hat{\mu}_1, \hat{\mu}_2, \sigma^2 \middle| y_1, \ldots, y_n\right)$$
$$= (2\pi\hat{\sigma}^2)^{-n/2} \exp\left\{-\frac{\sum_{i=1}^{k}(y_i - \bar{y}_1)^2 \ + \ \sum_{i=k+1}^{n}(y_i - \bar{y}_2)^2}{2\hat{\sigma}^2}\right\}.$$
- The value of k that **maximizes** the likelihood function is the **optimal** one. In practice, though, k is chosen to **minimize** the **Akaike Information Criterion** $AIC = -2\ln L + 2\,p$ where p is the number of parameters that have to be estimated from the data (in our example, we estimated $\hat{\mu}_1, \hat{\mu}_2$, and, $\hat{\sigma}^2$, so $p = 3$).
- The term $2p$ is called a **penalty term** as it penalizes for introducing **too many parameters** into the model.

8. Change-point Detection (cont.)

Historical Note.

- The pioneering work in the field of **change-point detection** is the book “Detection of Abrupt Changes - Theory and Application” by **Michèle Basseville** and **Igor Nikiforov**, Prentice-Hall, Inc., 1993.

- The **AIC criterion** was introduced by **Hirotugu Akaike** (1927 –2009) in 1974 in his article “A new look at the statistical model identification”. *IEEE Transactions on Automatic Control*, 19 (6): 716 - 723.

8. Change-point Detection – Coding Example

We apply the change-point detection methodology to daily closing Tesla stock prices between 6/30/2010 and 5/9/2024, a total of 3,489 observations. These historical data were downloaded from **https://yahoofinance.com.**

9. Anomaly Detection

- **Anomalies** of a **time series data** can be defined as **outliers** of the **remainders** once the **linear trend** and **seasonal periodicity** are taken into account.
- The **outliers** are determined as observations lying below $Q_1 - 3 \cdot IQR$ or above $Q_3 + 3 \cdot IQR$, where Q_1 is the **first quartile** (**25th percentile**), Q_3 is the **third quartile** (**75th percentile**), and the **interquartile range** is $IQR = Q_3 - Q_1$.
- More generally, in R, outliers are defined as values that lie $\frac{0.15}{\alpha} \cdot IQR$ distance away from the **quartiles**. The default value of $\alpha = 0.05$, thus resulting in the multiplicative constant of 3.

9. Anomaly Detection – Coding Example

We apply the anomaly detection methodology to daily closing Tesla stock prices between 6/30/2010 and 5/9/2024.

10. Natural Language Processing

Thank you