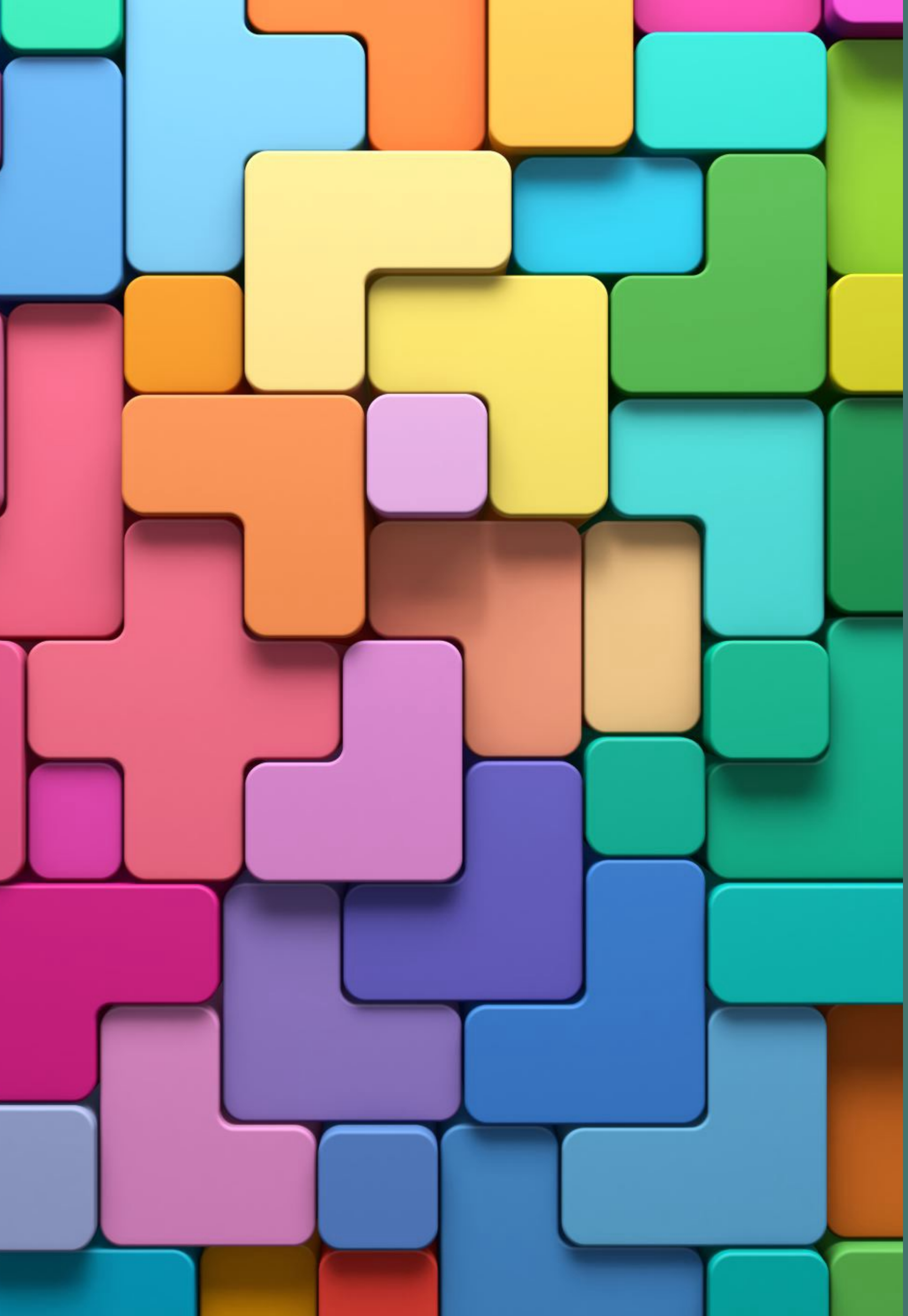


# WEEK 11 LECTURE

## *CLASSIFICATION*

## OBJECTIVES

- Logistic Regression
- Poisson Regression
- Naive Bayes Classifier
- K-Nearest Neighbors
- Decision Tree
- Support Vector Machine

## LOGISTIC REGRESSION

- **Pros:**
  - It is a very simple and efficient algorithm.
  - The model has low variance.
  - It provides probability score for observations.
- **Cons:**
  - It’s not good at handling a large number of categorical features.
  - It assumes that the data is free of missing values and predictors are independent of each other.

## POISSON REGRESSION

- Poisson regression is often used for modeling count data. It is used to answer the questions such as what factors can predict the frequency of an event.
- The model for Poisson regression model is
$$\log y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$
- The estimated regression equation is
$$\log \hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$$
- Requirement: The count (dependent) variable's variance is not greater than its mean.
- Coefficient interpretation: When $x_i$ is increased by 1 (other variables are held constant), $y$ is multiplied by $e^{b_i}$.

# POISSON REGRESSION (CONT.)

- How to determine the estimated coefficients $b_i$:

1) Consider the likelihood function

$$L(\beta_0, \beta_1, \dots, \beta_p) = \prod_{i=1}^{n} \frac{e^{-\lambda(x_i)} \lambda(x_i)^{y_i}}{y_i!}$$

where $\lambda(x_i) = e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}}$.

2) Maximize the likelihood function $L(\beta_0, \beta_1, \dots, \beta_p)$.

## POISSON REGRESSION (CONT.)

- **Pros:**
  - It does not predict negative values. It has a minimum value of 0. Thus, it is useful for distribution in which the mean or the most typical value is close to 0.
  - The model is useful for skewed data with long tail. So, it is appropriate for counts of rare events.
  - In general, for counts of events, it is a better model compared to the normal model.
- **Cons:**
  - The primary problem is over-dispersion.

## NAIVE BAYES CLASSIFIER

- Naive Bayes algorithm is a classification technique based on the Bayes' theorem. It assumes the independence of independent variables.
- Naive Bayes classifier is popular, especially in text classification, spam filtering, and recommendation systems.
- Suppose that we wish to classify an observation into one of $K$ classes, $K \geq 2$. Let $\pi_k$ denotes the overall or prior probability that a randomly chosen observation comes from the $k^{\text{th}}$ class. Let $f_k(X) = P(X|Y = k)$ denote the density function of $X$ for an observation that comes from the $k^{\text{th}}$ class. Then Bayes' theorem states that

$$p_k(x) = P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{K} \pi_i f_i(x)}$$

# NAIVE BAYES CLASSIFIER (CONT.)

- Assuming that within the $k^{\text{th}}$ class, the $p$ predictors are independent. This means that for $k = 1,2, \dots, K$,
$$f_k(x) = f_{k1}(x_1) \cdot f_{k2}(x_2) \cdot \dots \cdot f_{kp}(x_p)$$
where $f_{kj}$ is the density function of the $j^{\text{th}}$ predictor among observations in the $k^{\text{th}}$ class. The posterior probability is then given by
$$p_k(x) = P(Y = k|X = x) = \frac{\pi_k \cdot f_{k1}(x_1) \cdot f_{k2}(x_2) \cdot \dots \cdot f_{kp}(x_p)}{\sum_{i=1}^{K} [\pi_i \cdot f_{k1}(x_1) \cdot f_{k2}(x_2) \cdot \dots \cdot f_{kp}(x_p)]}$$
for $k = 1, 2, \dots, K$.
- Estimate $\pi_1, \dots, \pi_K$ as the proportion of training observations belonging to the $k^{\text{th}}$ class, for $k = 1, 2, \dots, K$.

## NAIVE BAYES CLASSIFIER (CONT.)

- To estimate the density functions $f_{kj}$ using training data $x_{1j}, \dots, x_{nj}$, consider a few options.
  - If $X_j$, $j = 1, \dots, p$, are quantitative, we can assume that $X_j | Y = k \sim N(\mu_{jk}, \sigma_{jk}^2)$, i.e. within each class, the $j$th predictor is normally distributed.
  - If $X_j$, $j = 1, \dots, p$, are quantitative, we can also make a histogram for the observations of the $j$th predictor within each class. Then we can estimate $k_{kj}(x_j)$ as the fraction of the training observations in the $k$th class that belong to the same histogram bin as $x_j$. Alternatively, we can use a kernel density estimator, which is essentially a smoothed version of a histogram.

## NAIVE BAYES CLASSIFIER (CONT.)

- If $X_j$, $j = 1, \dots, p$, are qualitative, then we can simply count the proportion of training observations for the $j$th predictor corresponding to each class. For instance, suppose that $X_j \in \{1, 2, 3\}$, and we have 100 observations in the $k$th class. Suppose that the $j$th predictor takes on values of 1, 2, and 3 in 32, 55, and 13 of those observations, respectively. Then we can estimate $f_{kj}$ as

$$\hat{f}_{kj}(x_j) = \begin{cases} 0.32 & if\ x_j = 1 \\ 0.55 & if\ x_j = 2 \\ 0.13 & if\ x_j = 3 \end{cases}$$

## NAIVE BAYES CLASSIFIER (CONT.)

- Pros:
  - This algorithm is easy and works very fast.
  - It can also be used to solve multi-class prediction problems as it’s quite useful with them.
  - This classifier performs better than other models with less training data if the assumption of independence of features holds.
- Cons:
  - It assumes that all the features are independent. In real life, this is hard to hold true.
  - It is also known as a bad estimator, so the probability outputs are not to be taken too seriously.

## NAIVE BAYES CLASSIFIER (CONT.)

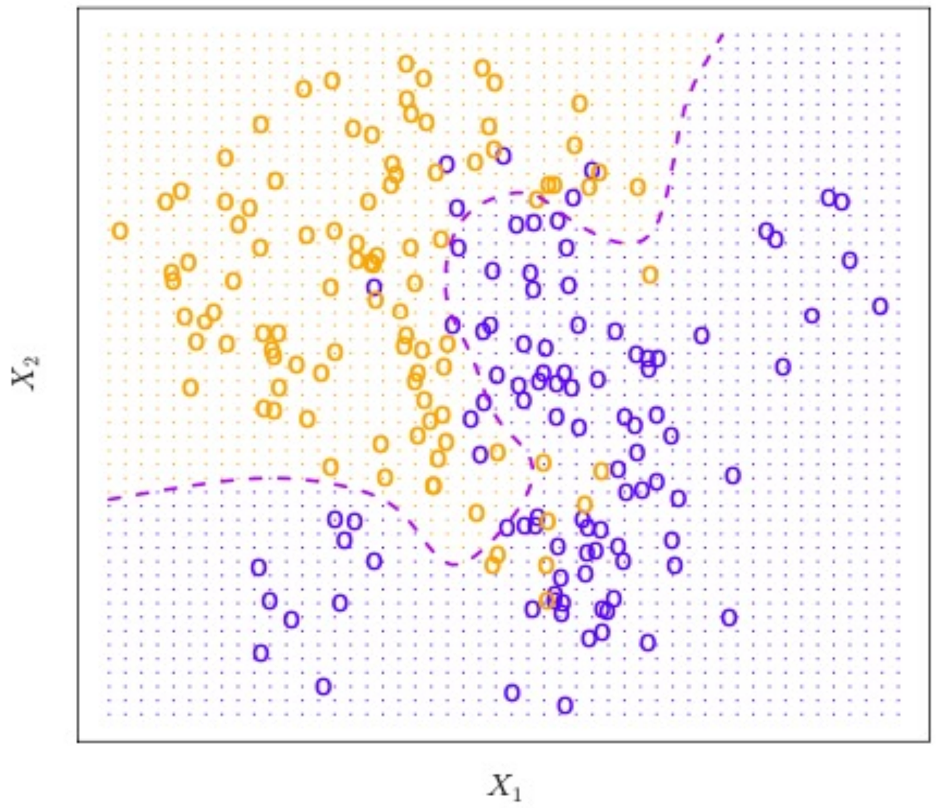


**FIGURE 2.13.** *A simulated data set consisting of* 100 *observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

## K-NEAREST NEIGHBORS (KNN)

- KNN attempts to estimate the conditional distribution of $Y$ given $X$, and then classify a given observation to the class with highest estimated probability. Given a positive integer $K$ and a test observation $x_0$, the KNN classifier first identifies the $K$ points in the training data that are closest to $x_0$, represented by the index set $\mathcal{N}_0$. It then estimates

  $$P(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

  Finally, KNN classifies the test observation $x_0$ to the class with the largest probability from the above probabilities.

# K-NEAREST NEIGHBORS (CONT.)

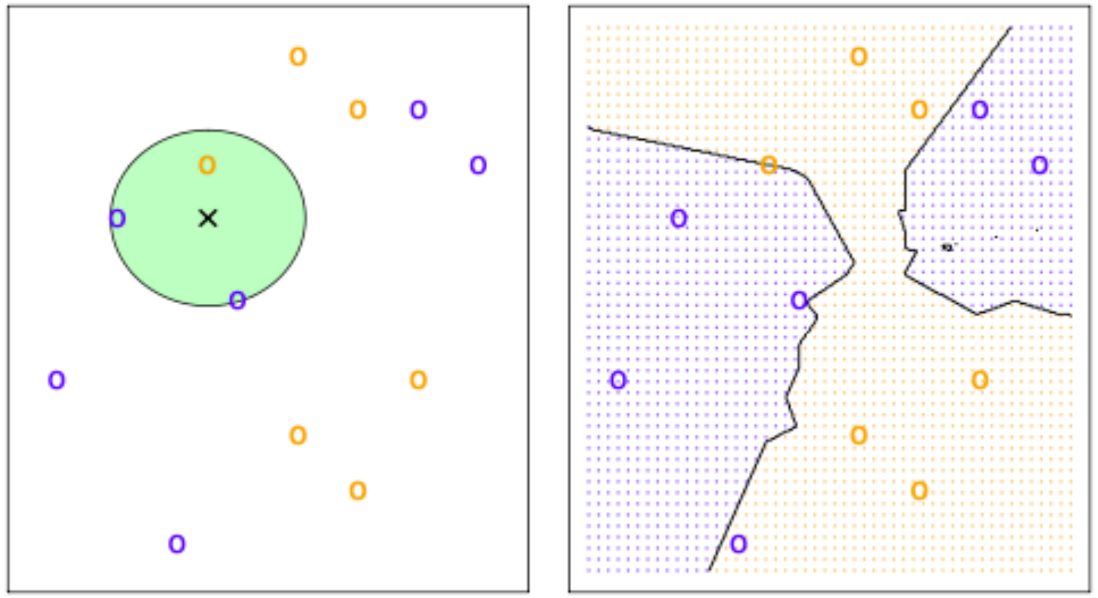


**FIGURE 2.14.** *The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.* Right: *The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

## K-NEAREST NEIGHBORS (CONT.)

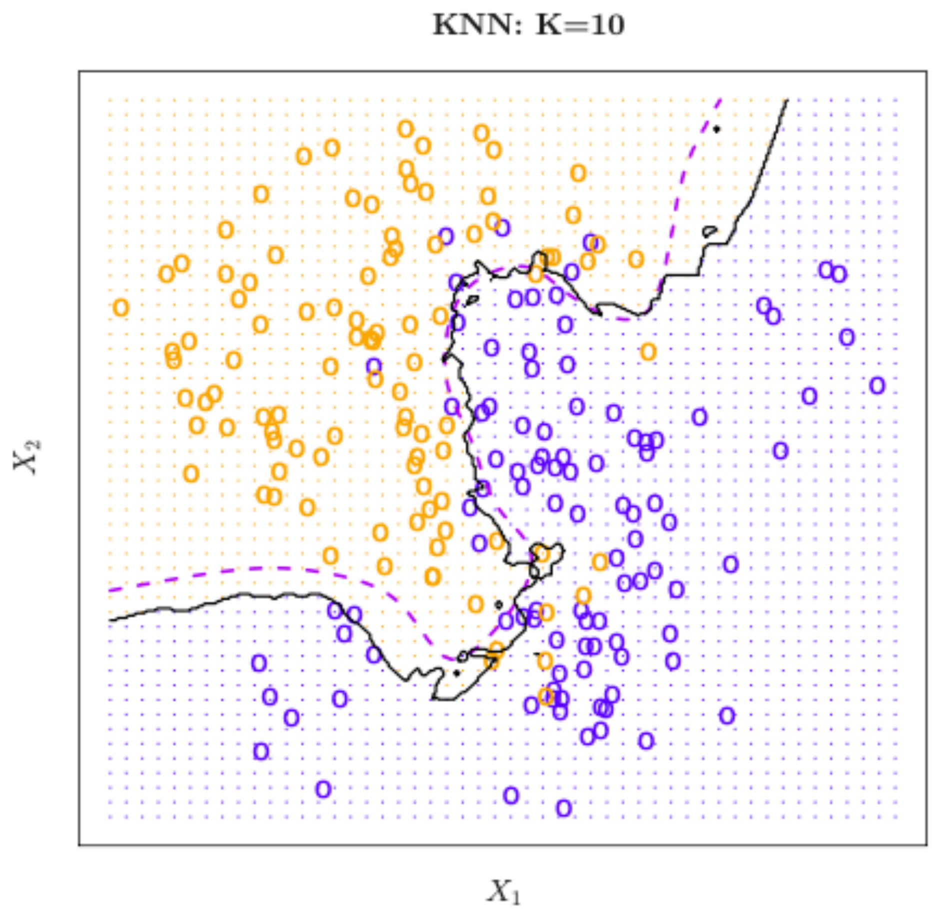


**FIGURE 2.15.** *The black curve indicates the KNN decision boundary on the data from Figure 2.13, using* $K = 10$*. The Bayes decision boundary is shown as a purple dashed line. The KNN and Bayes decision boundaries are very similar.*

## K-NEAREST NEIGHBORS (CONT.)

- **Pros:**
  - It is a simple and easy to implement.
  - There is no need to make additional assumptions.
  - It can be used for classification, regression and search.
- **Cons:**
  - It gets slow as the number of data points and/or predictors/features increase.
  - It assumes that the data is free of missing values and predictors are independent of each other.

## DECISION TREE

- Decision trees can be applied to both regression and classification problems.
- Roughly speaking, there are two steps in building a regression tree.
  1) Divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ – into $J$ distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
  2) For every observation that falls into the region $R_j$, we make the same prediction, which is simply the mean of the response values for the training observations in $R_j$.
- The construction of the regions $R_1, R_2, \ldots, R_J$ is beyond the scope of this course.

## DECISION TREE (CONT.)

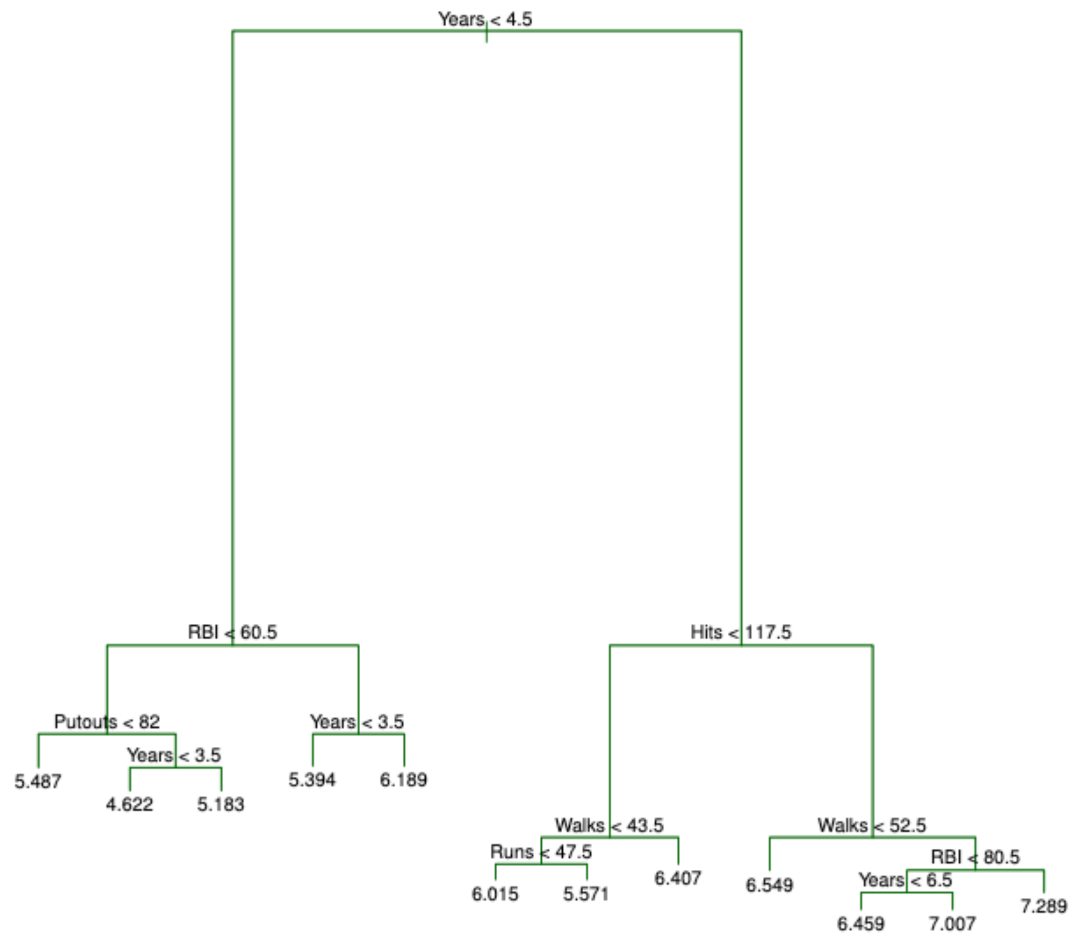
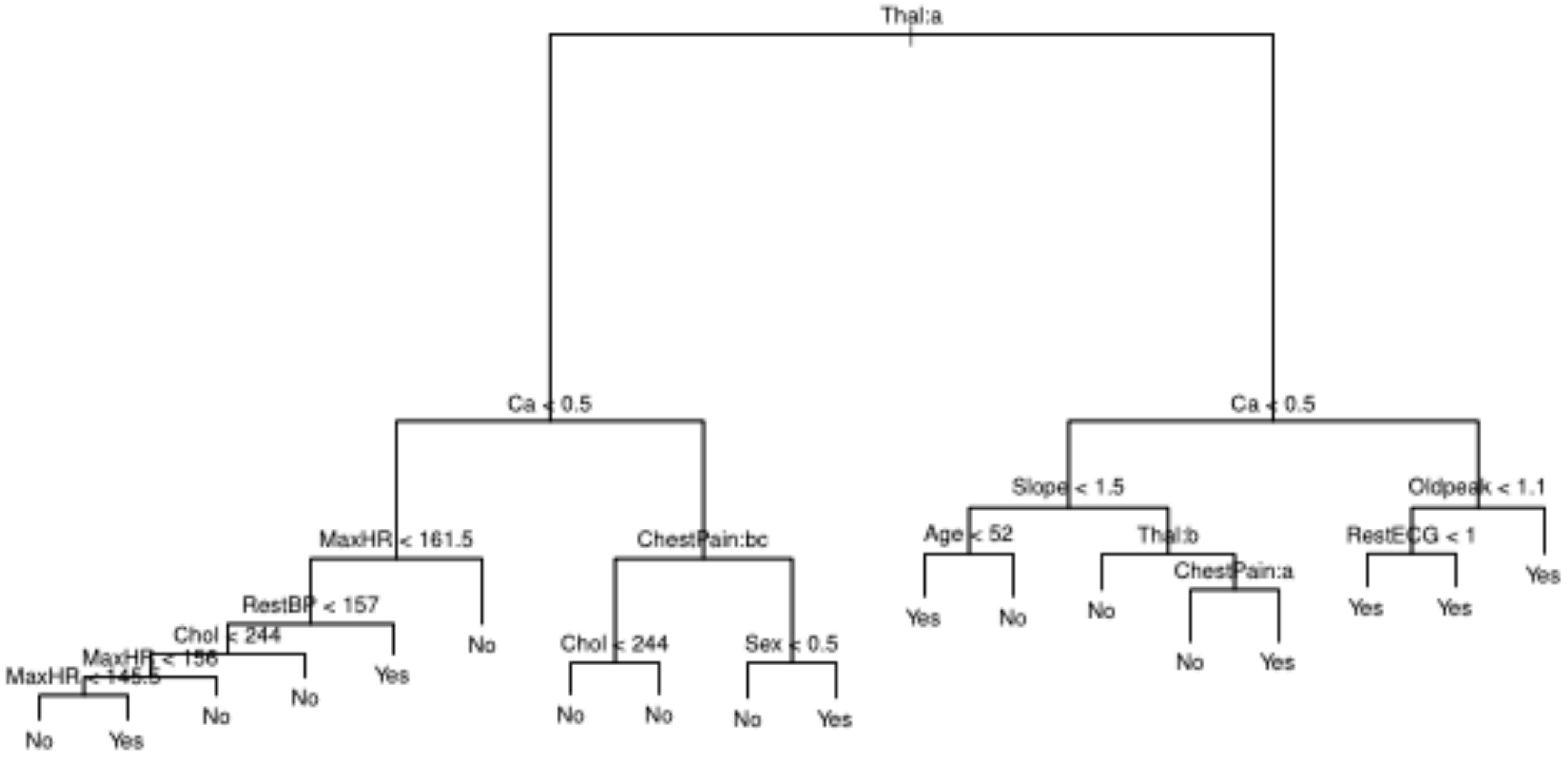


**FIGURE 8.4.** *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

## DECISION TREE (CONT.)

- A classification tree is very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one. There are two main steps:
  1) Divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ – into $J$ distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
  2) For every observation that falls into the region $R_j$, we predict that each observation belongs to the most commonly occurring class of training observations in the region $R_j$.

# DECISION TREE (CONT.)



# SUPPORT VECTOR CLASSIFIER

- Support vector classifier finds a hyperplane in an $N$-dimensional space($N$ – the number of features) that distinctly classifies the data points. SVM can be used for both regression and classification tasks, but it is widely used in classification.
- The hyperplane is chosen to correctly separate most of the training observations into the two classes, but it may misclassify a few observations. It is the solution to the optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,\,M}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,$$

## SUPPORT VECTOR CLASSIFIER (CONT.)

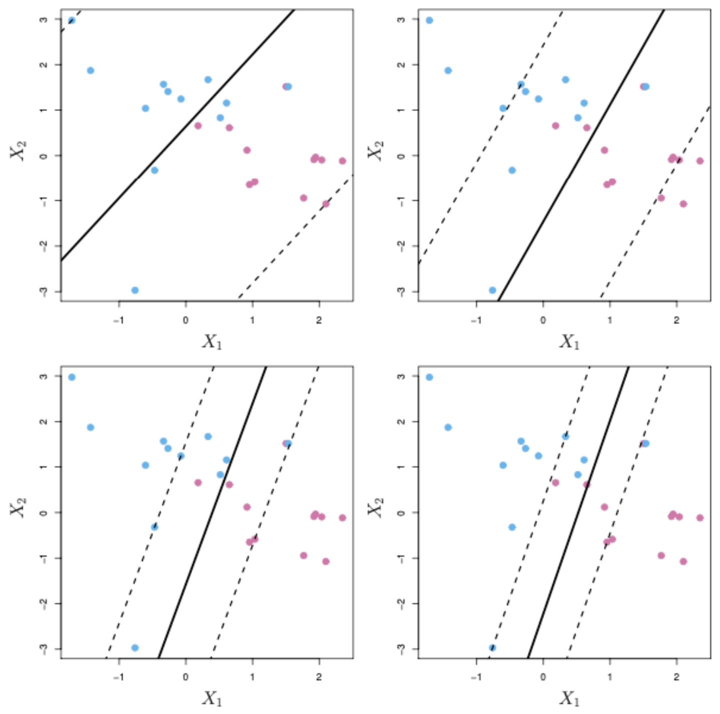


**FIGURE 9.7.** *A support vector classifier was fit using four different values of the tuning parameter $C$ in (9.12)–(9.15). The largest value of $C$ was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When $C$ is large, then there is a high tolerance for observations being on the wrong side of the margin, and so the margin will be large. As $C$ decreases, the tolerance for observations being on the wrong side of the margin decreases, and the margin narrows.*

# SUPPORT VECTOR CLASSIFIER (CONT.)

- The support vector classifier is a natural approach for classification in the two-class setting, if the boundary between the two classes is linear. However, in practice we are sometimes faced with non-linear class boundaries.

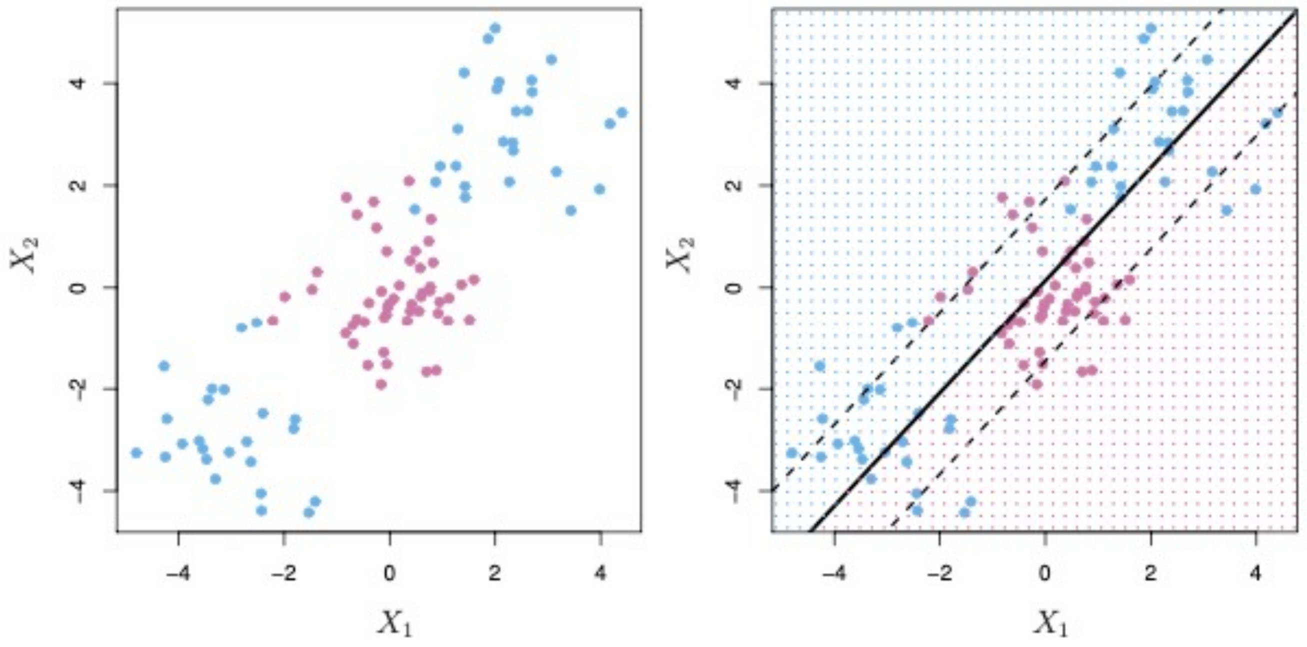
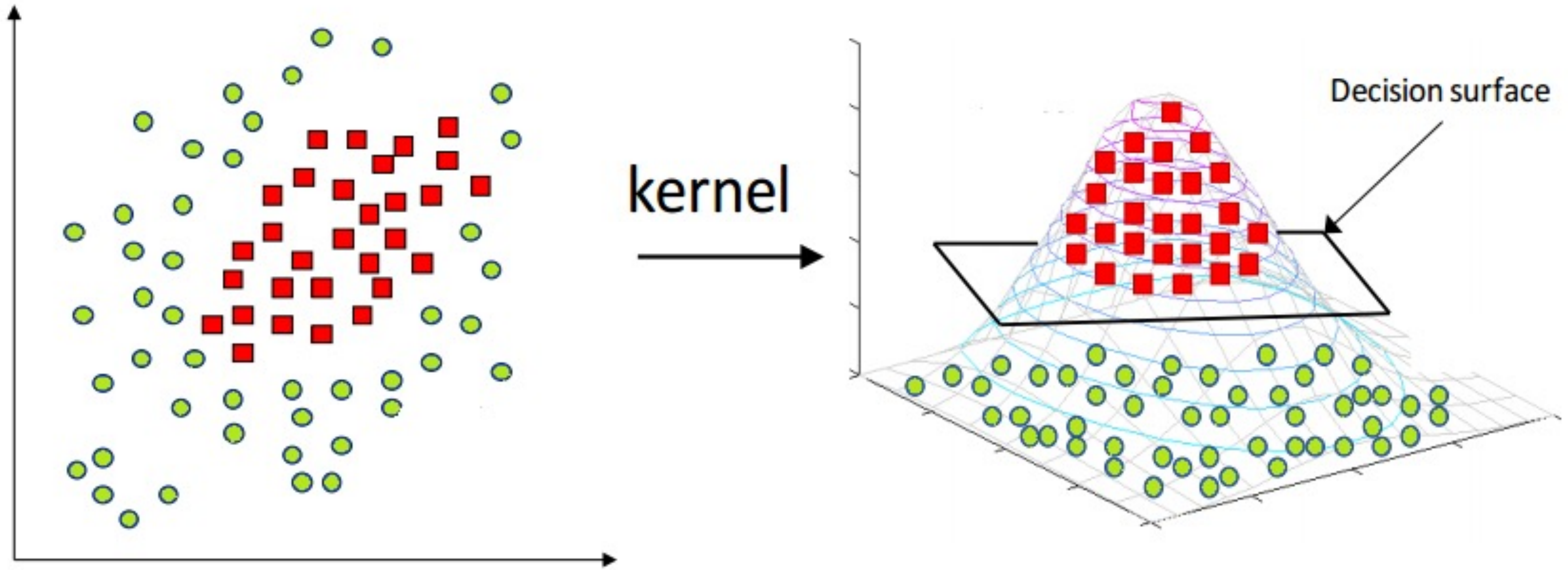


**FIGURE 9.8.** Left: *The observations fall into two classes, with a non-linear boundary between them.* Right: *The support vector classifier seeks a linear boundary, and consequently performs very poorly.*

## SUPPORT VECTOR MACHINE (SVM)



Source: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

# SUPPORT VECTOR MACHINE (CONT.)

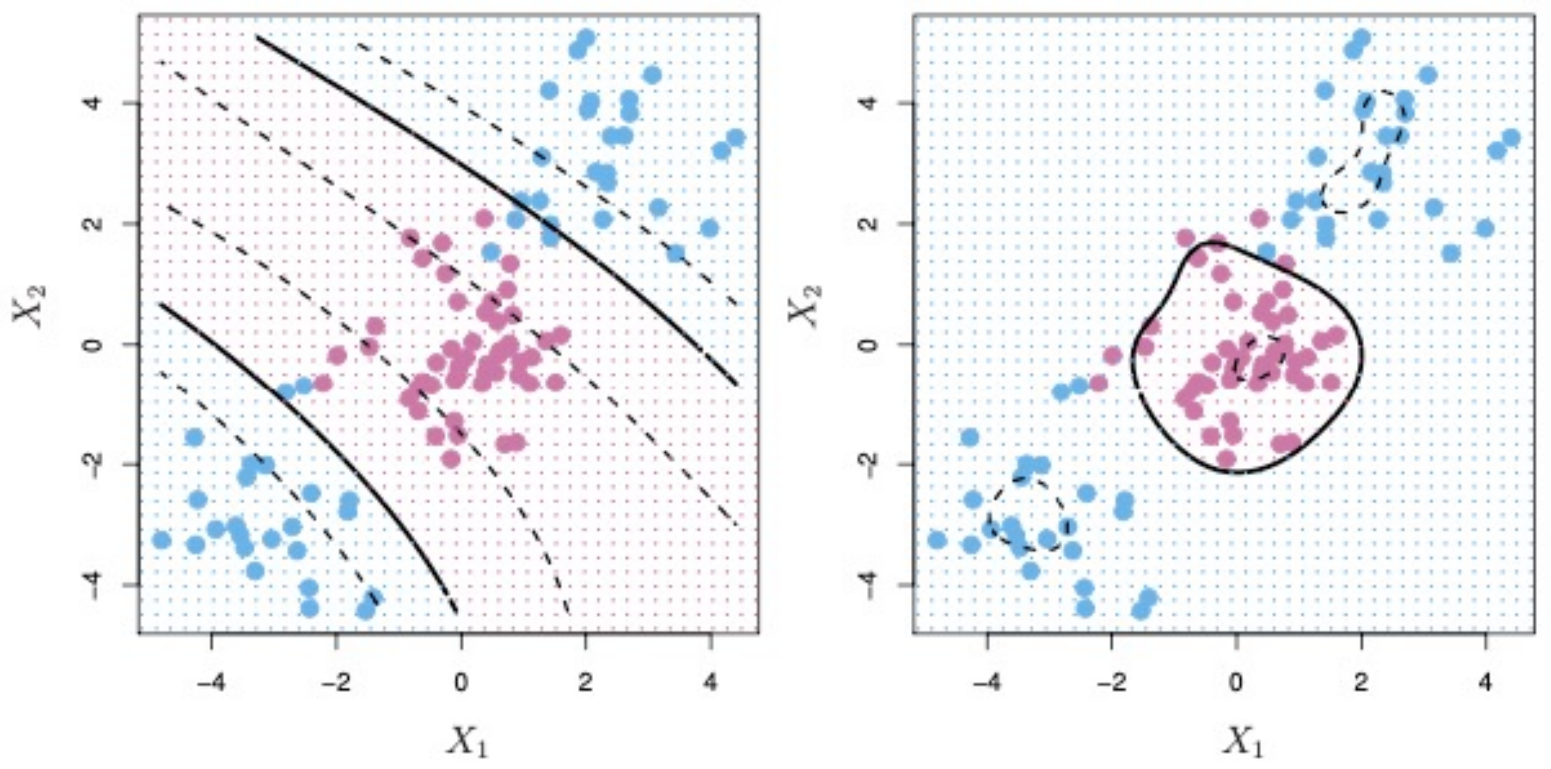


**FIGURE 9.9.** Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

## SUPPORT VECTOR MACHINE (CONT.)

- **Pros:**
  - It work well when there is a clear margin of separation between classes.
  - It is more effective in high-dimensional spaces.
- **Cons:**
  - It is not good for large data sets.
  - It is not good when classes are overlapping.