Explainable Machine Learning

Interpretable Models 2

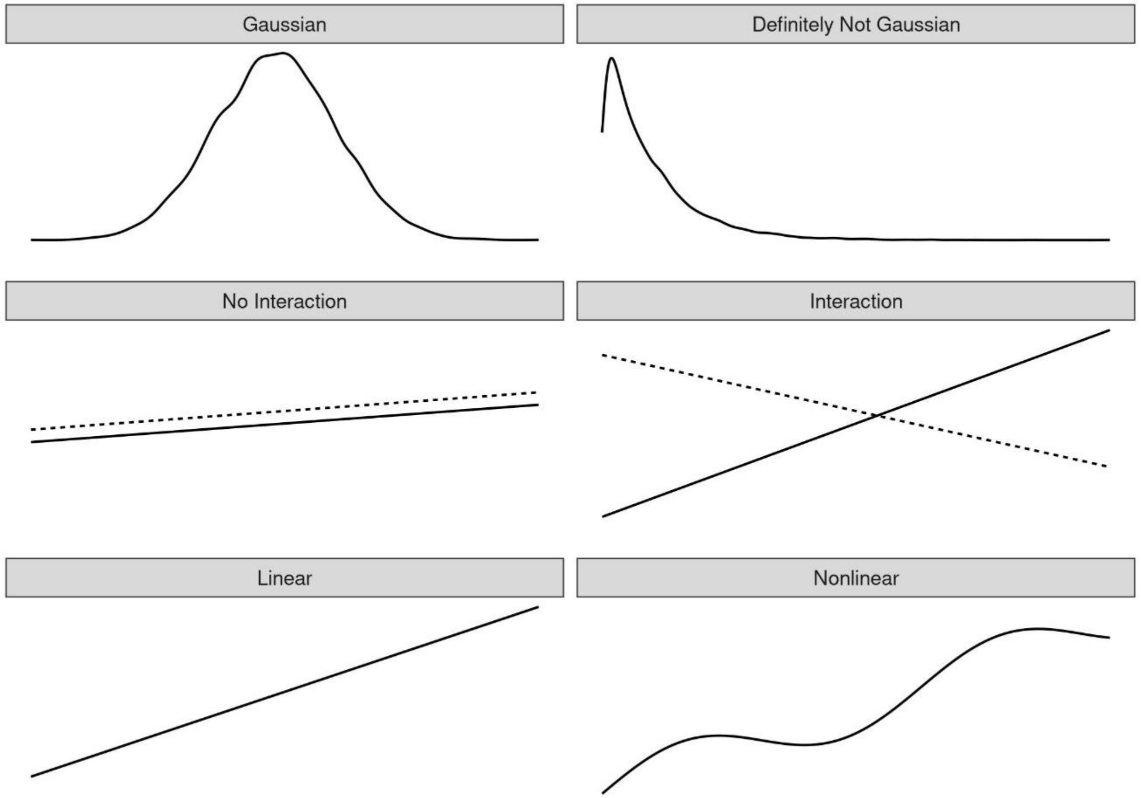
Shim Jaewoong

jaewoong@seoultech.ac.kr

Interpretable Models: **GLM, GAM and more**

Weakness of Linear Regression

- Many assumptions for linear regression are often violated in reality
  1. The outcome given the features might have a **non-Gaussian** distribution
  2. **Interaction** between the features might exist
  3. The outcome might be **nonlinear**.



→ Generalized Linear Models (GLM)

→ Adding interactions manually

→ Generalized Additive Models (GAM)
or transformation of features

Generalized Linear Models (GLM)

- The outcome can be….
  - a category (cancer vs. healthy)
  - a count (number of children)
  - the time to the occurrence of an event (time to failure of a machine)
  - a very skewed outcome with a few very high values (household income)

The core concept of any GLM is:
Keep the weighted sum of the features, but allow non-Gaussian outcome distributions and **connect the expected mean of this distribution and the weighted sum through a possibly nonlinear function**.

Generalized Linear Models (GLM)

- Logistic regression is an example of GLM!

  - **Probabilistic Interpretation of Logistic Regression**
    - Assume $y \sim Bernoulli(\hat{y})$, $\hat{y} = \sigma(\mathbf{w}^T \boldsymbol{x})$

$$p(y = 1|x; \mathbf{w}) = \sigma(\mathbf{w}^T \boldsymbol{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \boldsymbol{x})}$$

$$p(y = 0|x; \mathbf{w}) = 1 - \sigma(\mathbf{w}^T \boldsymbol{x}) = \frac{\exp(-\mathbf{w}^T \boldsymbol{x})}{1 + \exp(-\mathbf{w}^T \boldsymbol{x})}$$

▼

$$p(y|x; \mathbf{w}) = \underbrace{\left(\sigma(\mathbf{w}^T \boldsymbol{x})\right)^y \left(1 - \sigma(\mathbf{w}^T \boldsymbol{x})\right)^{1-y}}_{\text{p.f. of } Bernoulli(\hat{y})}$$

  - **Maximum Likelihood Estimation** (with respect to $\hat{y}$)

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \overbrace{\prod_{(\boldsymbol{x}_i, y_i) \in D} p(y_i|\boldsymbol{x}_i; \mathbf{w})}^{\text{likelihood}} = \underset{\mathbf{w}}{\operatorname{argmax}} \overbrace{\sum_{(\boldsymbol{x}_i, y_i) \in D} \log p(y_i|\boldsymbol{x}_i; \mathbf{w})}^{\text{log-likelihood}}$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \boxed{\sum_{(\boldsymbol{x}_i, y_i) \in D} [y_i \log \sigma(\mathbf{w}^T \boldsymbol{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \boldsymbol{x}_i))]}$$

▶negative of binary cross entropy

Generalized Linear Models (GLM)

Three components: $$g(E_Y(y|x)) = \beta_0 + \beta_1 x_1 + \ldots \beta_p x_p$$

link function

probability distribution
(from the exponential family)

linear predictor

	Linear regression	**Logistic regression**	**…**
Distribution	Gaussian	Bernoulli	…
Link function	Identity function	Logit function (inverse of sigmoid function)	…

Generalized Linear Models (GLM)

- the outcome (y) : a count of something (e.g. number of children living in a household)
  - GLM with the Poisson distribution
  - Natural logarithm as a link function

$$ln(E_Y(y|x)) = x^T\beta$$

- the outcome (y) : yes/no
  - GLM with the Bernoulli distribution
  - Logit function as a link function

$$x^T\beta = ln\left(\frac{E_Y(y|x)}{1 - E_Y(y|x)}\right) = ln\left(\frac{P(y=1|x)}{1 - P(y=1|x)}\right)$$

Can be transformed to logistic regression formula

- the outcome (y) : always positive (e.g. time between two events)
  - GLM with the exponential distribution
  - Negative inverse as a link function

Generalized Linear Models (GLM)

- Example

the distribution of the target variable, the number of coffees on a given day

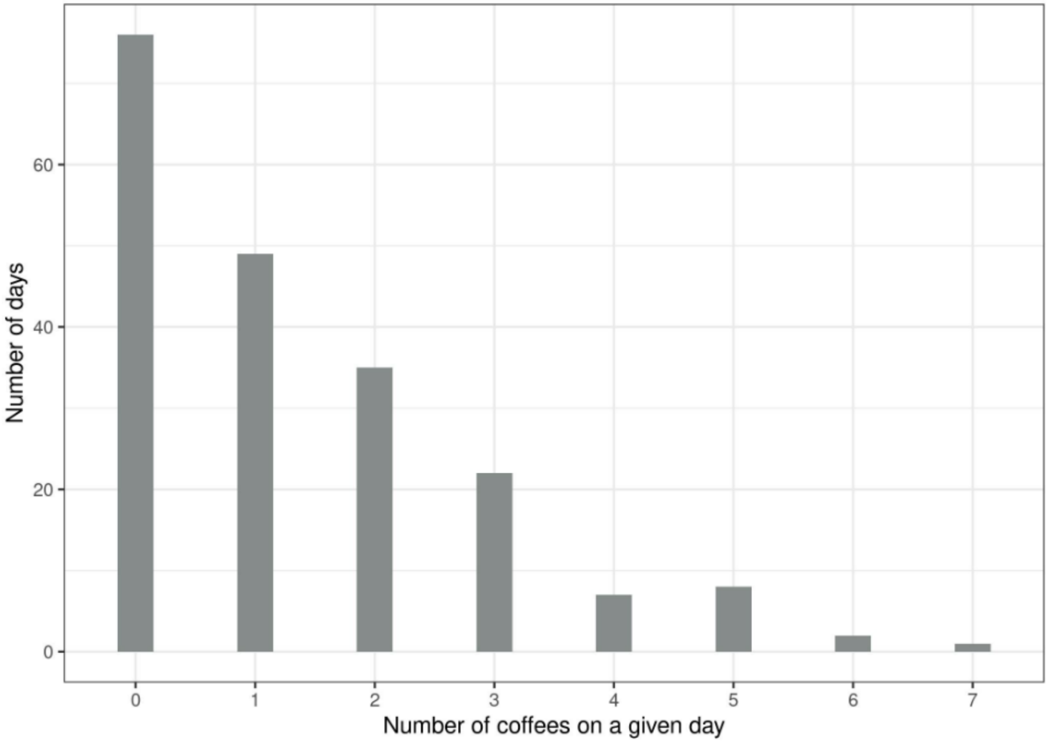
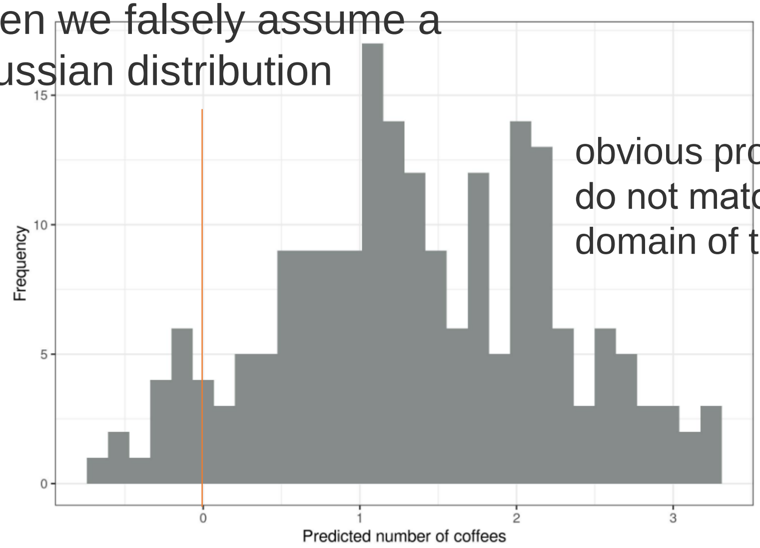


FIGURE 5.9: Simulated distribution of number of daily coffees for 200 days.

When we falsely assume a Gaussian distribution



obvious problem: the predictions do not match the "allowed" domain of the true outcome

FIGURE 5.10: Predicted number of coffees dependent on stress, sleep and work. The linear model predicts negative values.

The fitted Poisson GLM leads to

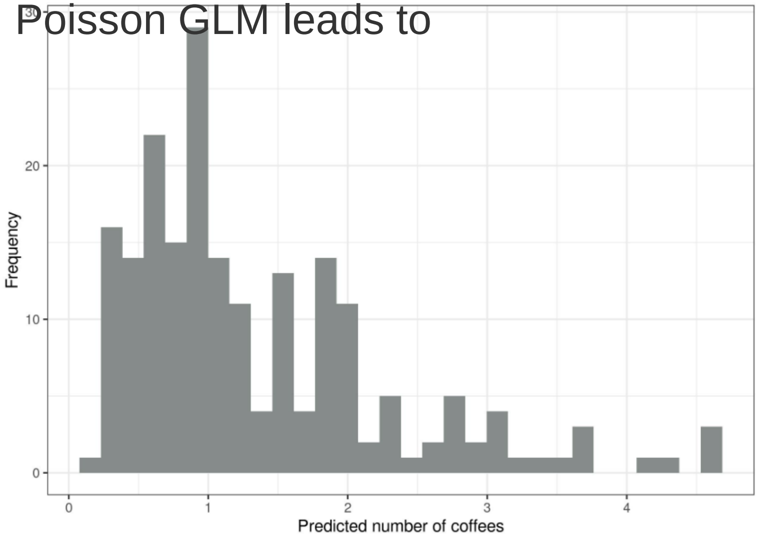


FIGURE 5.11: Predicted number of coffees dependent on stress, sleep and work. The GLM with Poisson assumption and log link is an appropriate model for this dataset.

Generalized Linear Models (GLM)

- Interpretation
  - GLM with Poisson distribution and log link

$$ln(E(\text{coffee}|\text{str}, \text{slp}, \text{wrk})) = \beta_0 + \beta_{\text{str}} x_{\text{str}} + \beta_{\text{slp}} x_{\text{slp}} + \beta_{\text{wrk}} x_{\text{wrk}}$$

$$E(\text{coffee}|\text{str}, \text{slp}, \text{wrk}) = exp(\beta_0 + \beta_{\text{str}} x_{\text{str}} + \beta_{\text{slp}} x_{\text{slp}} + \beta_{\text{wrk}} x_{\text{wrk}})$$

TABLE 5.3: Weights in the Poisson model

	weight	exp(weight) [2.5%, 97.5%]
(Intercept)	-0.16	0.85 [0.54, 1.32]
stress	0.12	1.12 [1.07, 1.18]
sleep	-0.15	0.86 [0.82, 0.90]
workYES	0.80	2.23 [1.72, 2.93]

- Increasing the stress level by one point multiplies the expected number of coffees by the factor 1.12
- The predicted number of coffees on a work day is on average 2.23 times the number of coffees on a day off

Interactions

- The linear regression model assumes that the effect of one feature is the same regardless of the values of the other features (= no interactions)
  - To predict the number of bicycles rented, there may be an interaction between temperature and whether it is a working day or not.

- ***Interaction Features*** **– products between original features**
  - $(x_1, x_2) \rightarrow (x_1 x_2)$
  - $(x_1, x_2, x_3) \rightarrow (x_1 x_2, x_2 x_3, x_1 x_3, x_1 x_2 x_3)$
  - ...

- **Example: Bivariate Quadratic Regression with Interaction**

$$\hat{y} = w_{00} + w_{10} x_1 + w_{01} x_2 + w_{20} x_1^2 + w_{11} x_1 x_2 + w_{02} x_2^2$$

Interactions

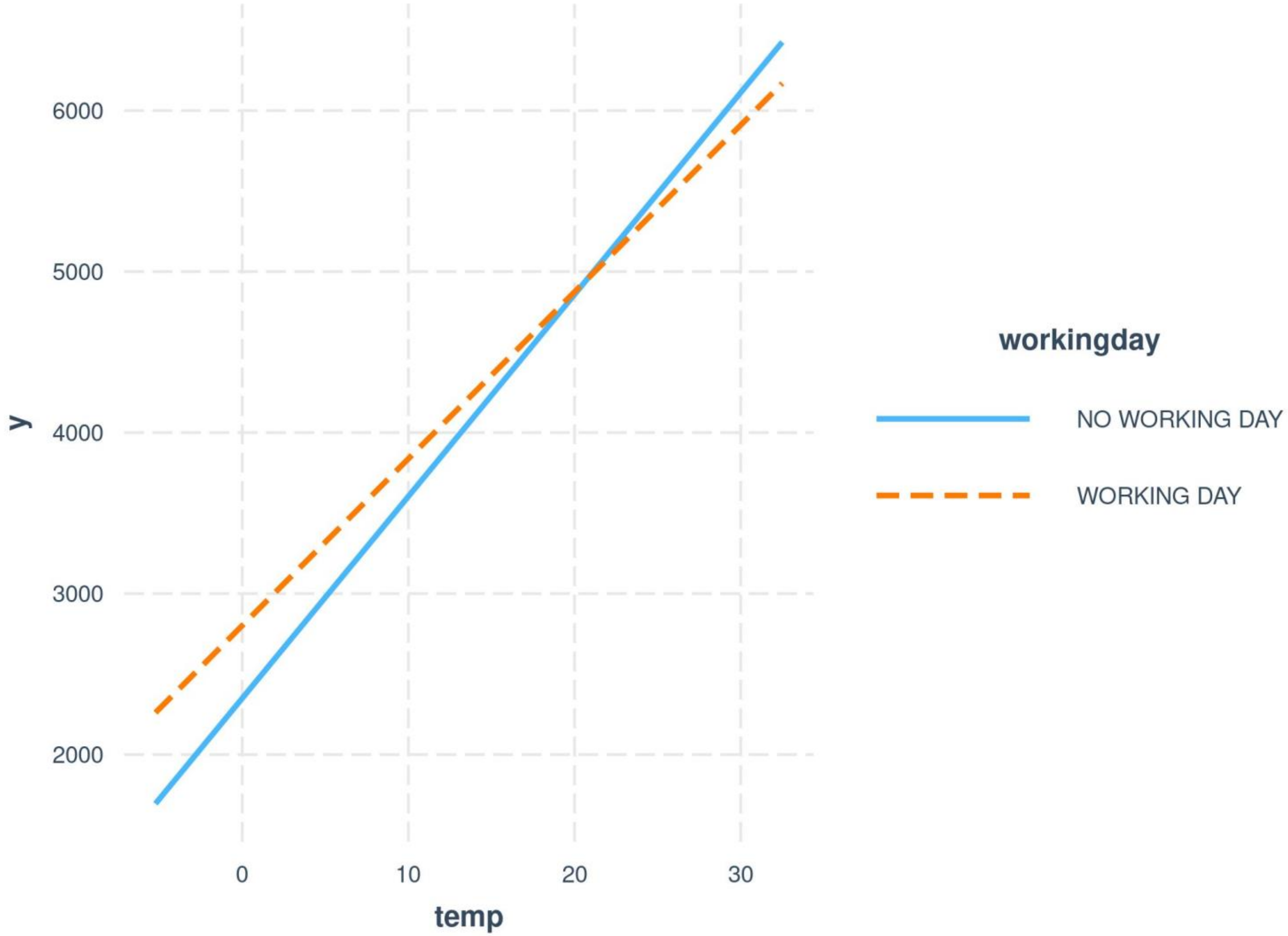
- Example
  - Bike rental prediction

	Weight	Std. Error	2.5%	97.5%
(Intercept)	2185.8	250.2	1694.6	2677.1
seasonSPRING	893.8	121.8	654.7	1132.9
seasonSUMMER	137.1	161.0	-179.0	453.2
seasonFALL	426.5	110.3	209.9	643.2
holidayHOLIDAY	-674.4	202.5	-1071.9	-276.9
workingdayWORKING DAY	451.9	141.7	173.7	730.1
weathersitMISTY	-382.1	87.2	-553.3	-211.0
weathersitRAIN/…	-1898.2	222.7	-2335.4	-1461.0
temp	125.4	8.9	108.0	142.9
hum	-17.5	3.2	-23.7	-11.3
windspeed	-42.1	6.9	-55.5	-28.6
days_since_2011	4.9	0.2	4.6	5.3
workingdayWORKING DAY:temp	-21.8	8.1	-37.7	-5.9

Does the temperature have a negative effect given it is a working day?

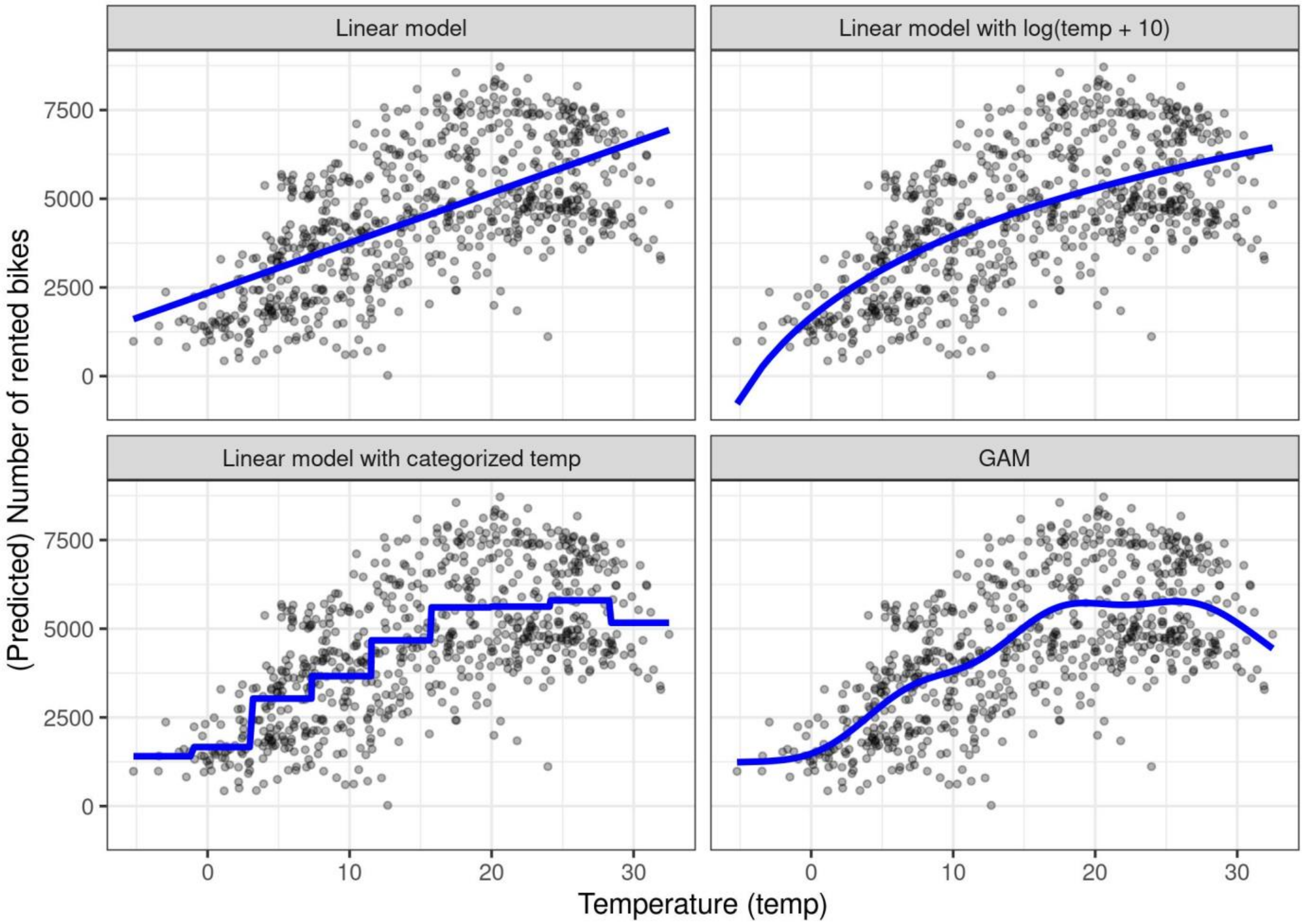
Interactions

- Example
  - Bike rental prediction



Generalized Additive Models (GAM)

- **The world is not linear**
  - **How to model nonlinear relationship?**
    - Simple transformation of the feature (e.g. logarithm)
    - Categorization of the feature
    - Generalized Additive Models (GAMs)



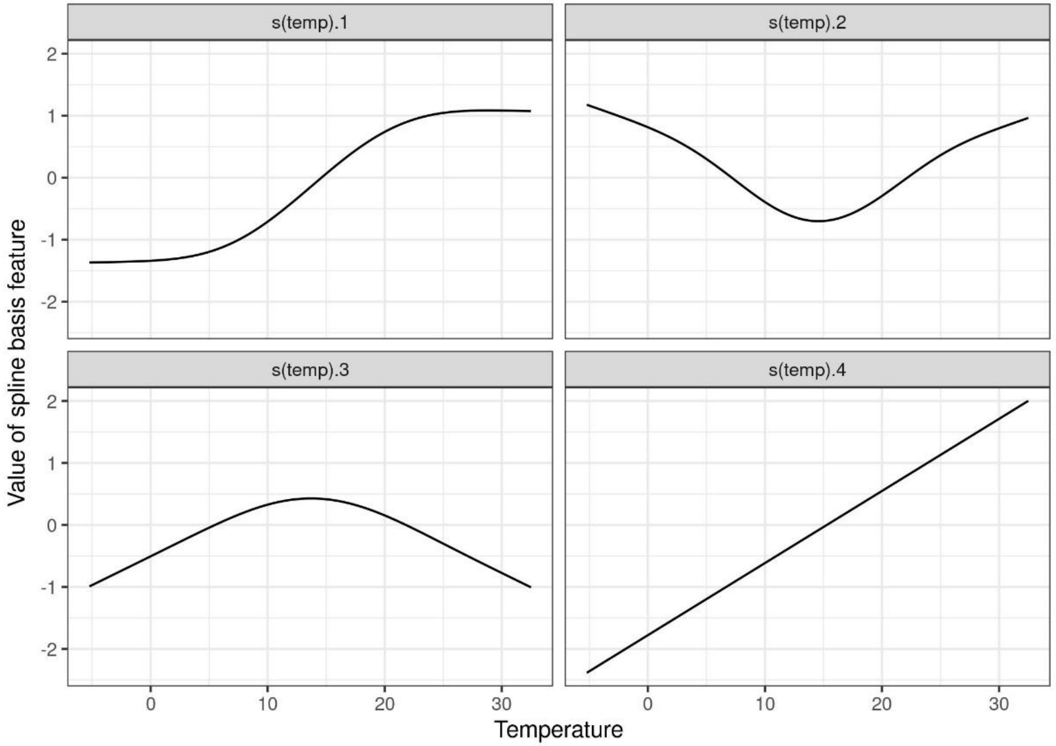
Generalized Additive Models (GAM)

- GAM
  - Allows for flexible nonlinearities in several variables, but retains the additive structure of linear models.

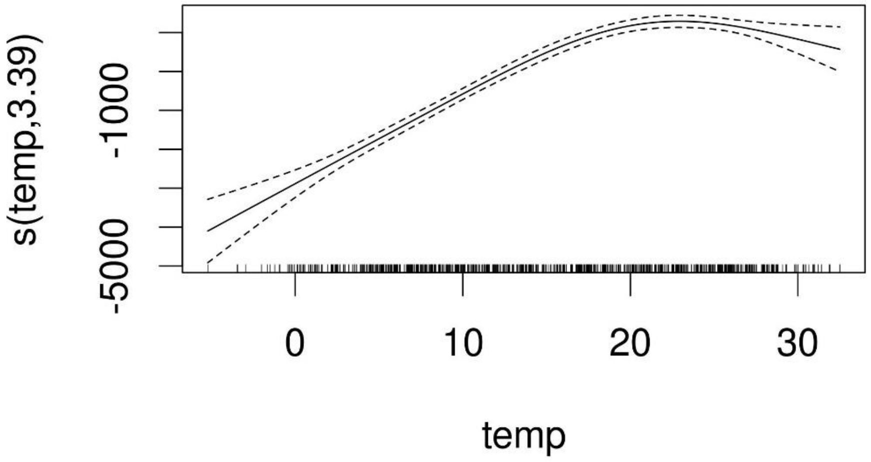
$$g(E_Y(y|x)) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p)$$

Smooth function **(spline)** : constructed from simpler basis functions

Basis functions



	weight
(Intercept)	4504.35
s(temp).1	989.34
s(temp).2	740.08
s(temp).3	2309.84
s(temp).4	558.27



Generalized Additive Models (GAM)

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon$$

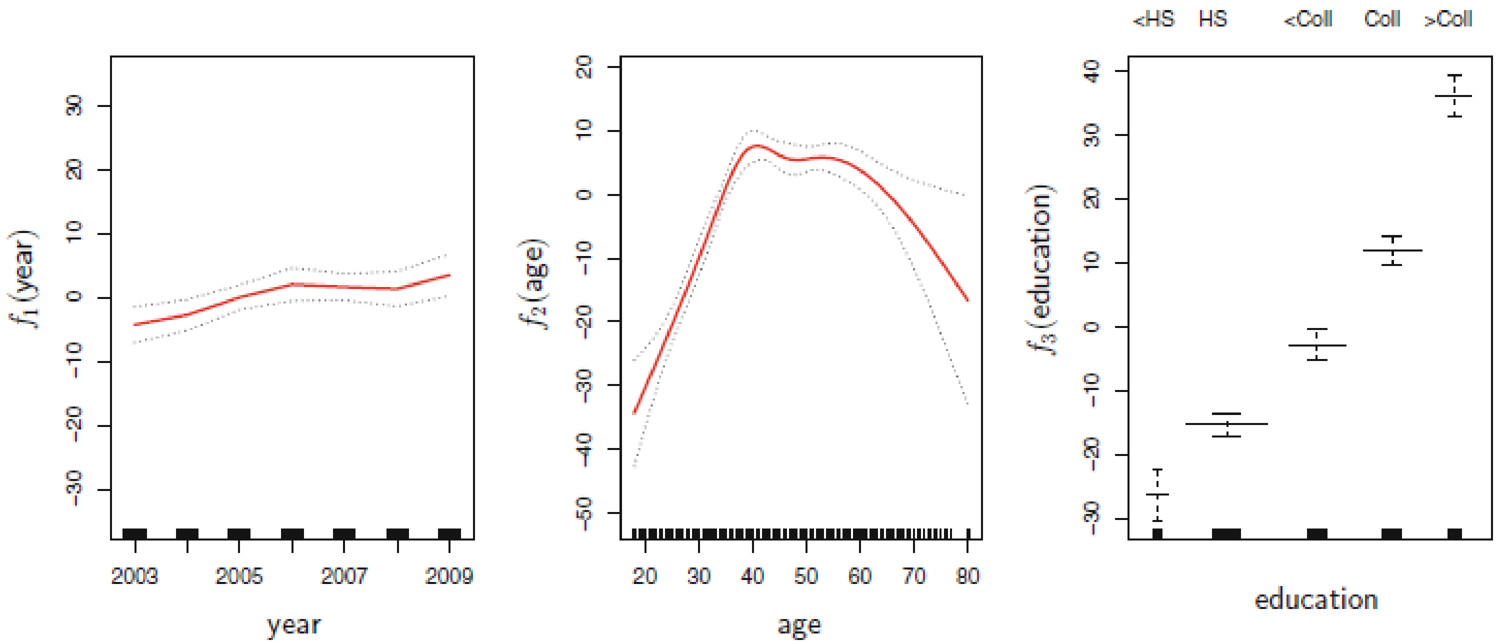


FIGURE 7.11. *For the* `Wage` *data, plots of the relationship between each feature and the response,* `wage`*, in the fitted model (7.16). Each plot displays the fitted function and pointwise standard errors. The first two functions are natural splines in* `year` *and* `age`*, with four and five degrees of freedom, respectively. The third function is a step function, fit to the qualitative variable* `education`*.*

- Holding constant age and education, wages increase with year.

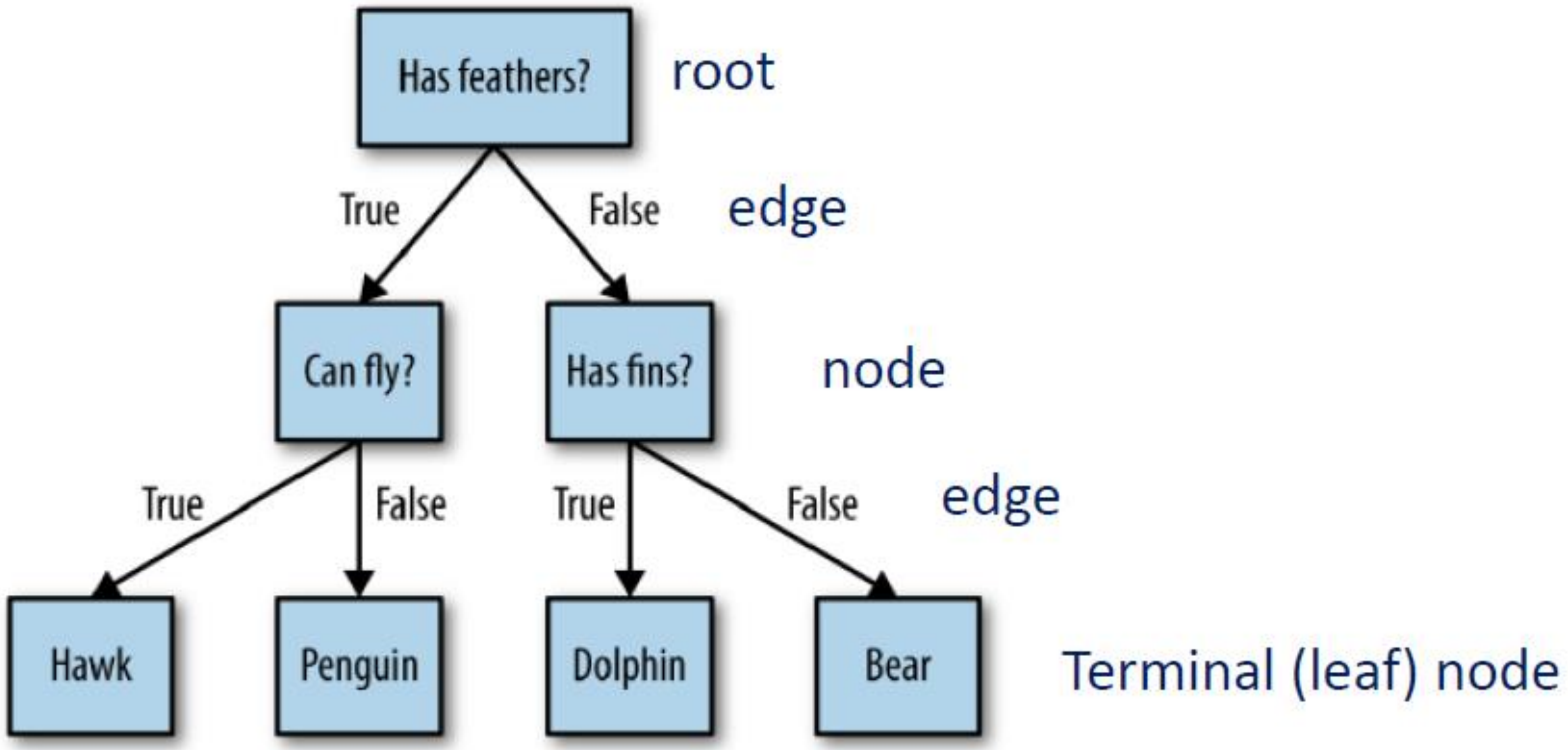
- Holding constant year and education, wages initially increase then decrease with age.

- Holding constant year and age, wages increase with education level.

Interpretable Models:
Decision Tree

Decision Trees

- A decision tree is a hierarchy of if/else questions leading to a decision.
  - Each node either represents a question or a terminal node (also called a leaf) that contains the answer.
  - The edges connect the answers to a question with the next question you would ask.



Decision Trees

- **Decision Tree Learning**
  - There are finitely many different decision trees.
  - **Optimal algorithm** for the construction of a tree is to simply generate all possible trees and choose the best one.
  - The obvious disadvantage of this algorithm is its unacceptably high computation time, as soon as the number of features becomes somewhat larger.
  - Thus, we use **heuristic algorithms** with greedy strategy.

* Because <u>greedy strategy</u> is used for construction of the tree, the trees are in general suboptimal.

Decision Trees

- Given a (training) dataset $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)\}$ such that $\boldsymbol{x}_i = (x_{i1}, \dots, x_{id}) \in \mathbb{R}^d$ is the *i*-th input vector of *d* features and y_i is the corresponding target label.

- **General Procedure –** Repeatedly split a node into two parts so as to minimize the impurity of outcome within the new parts.
  - The training dataset D constitutes the root node
  - Repeat the following process
    - Try all possible splits in all nodes and features to find the **"best split"**
    - Split the node

Decision Trees

- **How to determine the best split (for classification)**
  - Nodes with purer class distribution are preferred

C0: 5 C1: 5	C0: 9 C1: 1
High degree of impurity	Low degree of impurity

  - **Measures of node impurity**

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

$$Error(t) = 1 - \max_i P(i \mid t)$$

Impurity Measures: Gini Index

- **Gini Index at a node t:**

$$\mathrm{GINI}(t) = 1 - \sum_j [p(j|t)]^2$$

(NOTE: $p(j|t)$ is the fraction of class j at node t).

- Maximum $(1 - 1/n_c)$ when objects are equally distributed among all classes, implying least interesting information
- Minimum (0) when all objects belong to one class, implying most interesting information
- For 2-class problem $(p, 1-p)$: GINI $= 1 - p^2 - (1-p)2 = 2p(1-p)$

C1	**0**
C2	**6**
Gini=0.000	

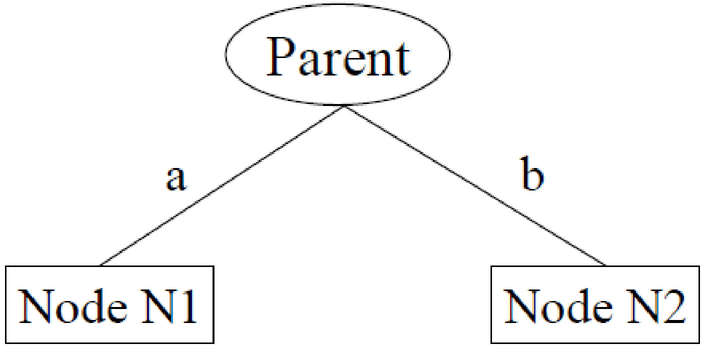
C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Impurity Measures: Gini Index

- **How to determine the best split (for classification)**
  1. **Compute impurity measure (P) before splitting**
  2. **Compute impurity measure (M) after splitting**
     - Compute impurity measure of each child node
     - M is the weighted impurity of children
  3. **Choose the feature that produces the highest gain**
     - **Gain = P - M**



	Parent
C1	**7**
C2	**5**
Gini = 0.486	

Gini(N1)
$= 1 - (5/6)^2 - (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 - (2/6)^2 - (4/6)^2$
$= 0.444$

	N1	**N2**
C1	**5**	**2**
C2	**1**	**4**
Gini=0.361		

Weighted Gini of N1 N2
$= 6/12 * 0.278 +$
$6/12 * 0.444$
$= 0.361$

Gain = 0.486 – 0.361 = 0.125

Impurity Measures: Gini Index

- **Computing Gini Index: Categorical Attributes**
  - For each distinct value (or partition), gather counts for each class in the dataset
  - Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

Impurity Measures: Gini Index

- **Computing Gini Index: Continuous Attributes**
  - Use binary decisions based on one value
  - Several choices for the splitting value
    - Number of possible splitting values = Number of distinct values
  - Each splitting value has a count matrix associated with it
    - Class counts in each of the partitions, $A < v$ and $A \geq v$
  - Simple method to choose best v
    - For each v, scan the database to gather count matrix and compute its Gini index.
    - Computationally Inefficient! Repetition of work.

ID	Home Owner	Marital Status	Annual Income	Defaulted
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

Annual Income ?

	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

Impurity Measures: Gini Index

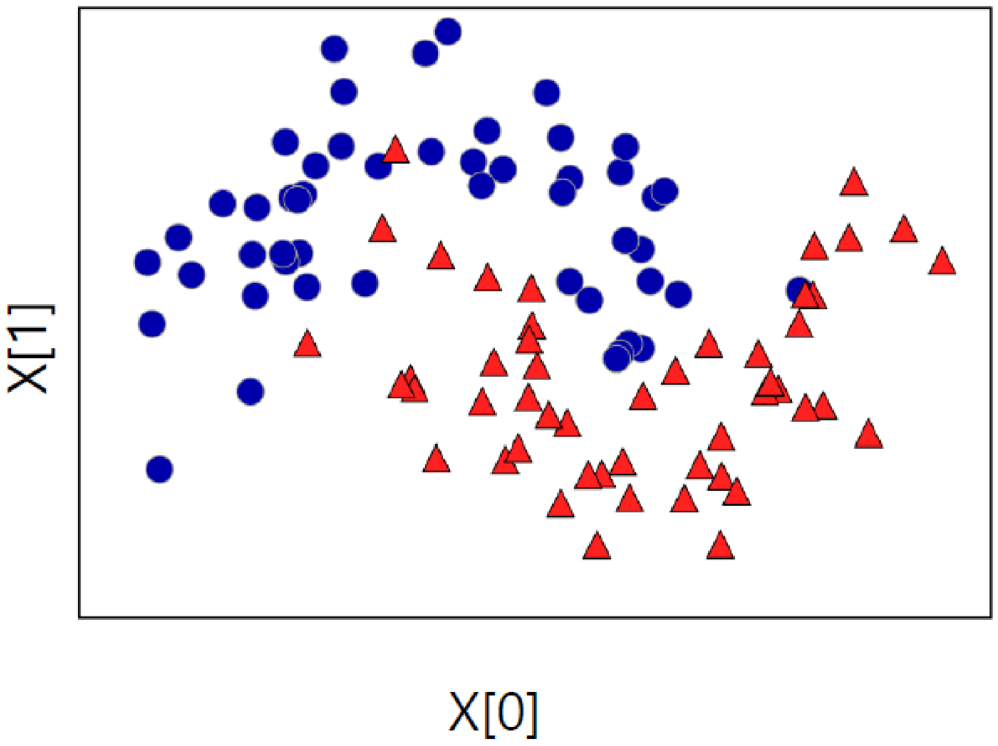
- **Computing Gini Index: Continuous Attributes**
  - For each attribute,
    - Sort the attribute on values
    - Linearly scan these values, each time updating the count matrix and computing Gini index
    - Choose the split position that has the least Gini index

	Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
		Annual Income									
Sorted Values →		60	70	75	85	90	95	100	120	125	220

Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

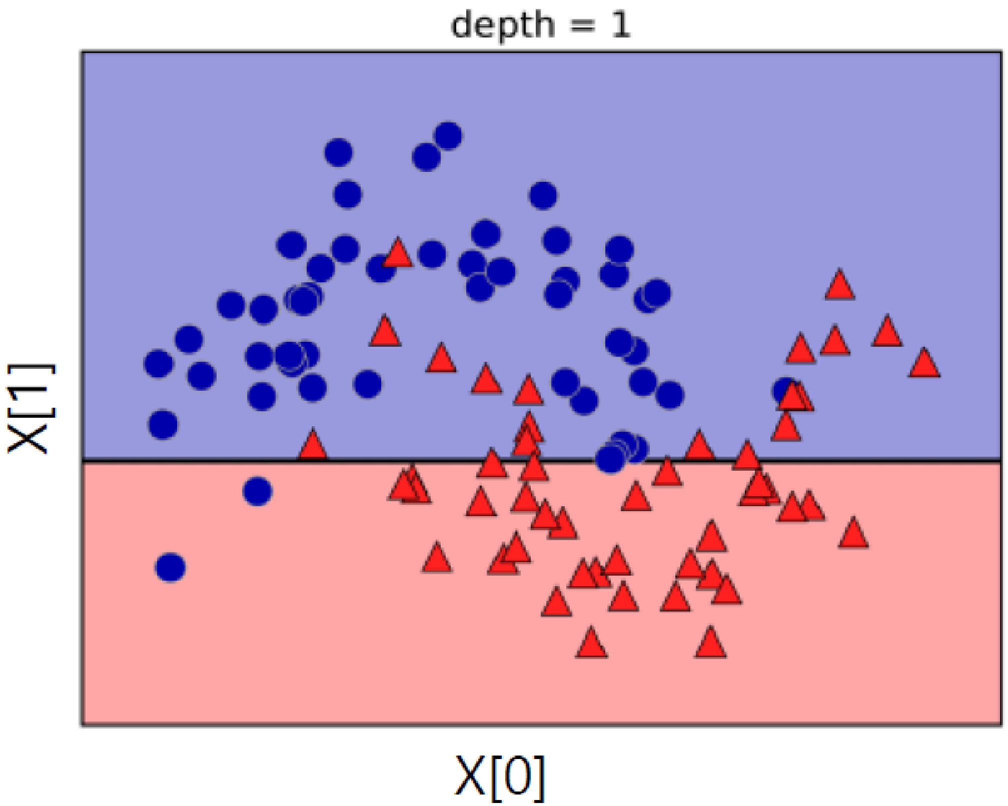
Decision Trees

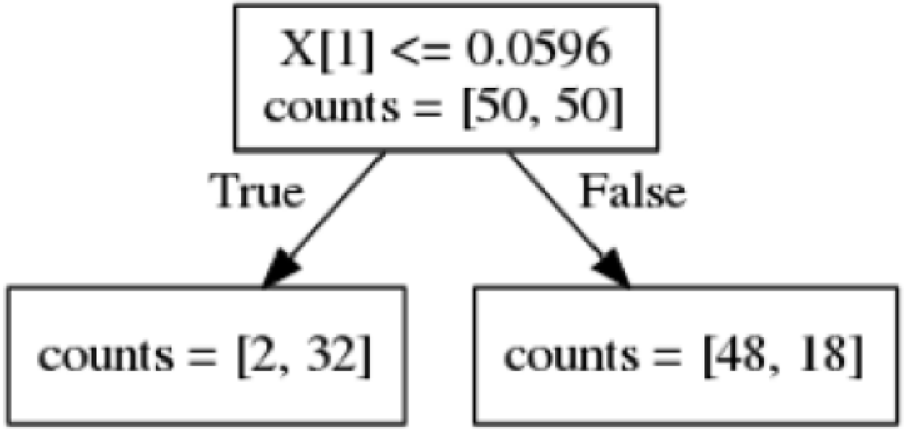
- **Example of building a decision tree (for classification)**
  - The *two-moons* dataset consists of two half-moon shapes, with each class consisting of 75 data points.



Decision Trees

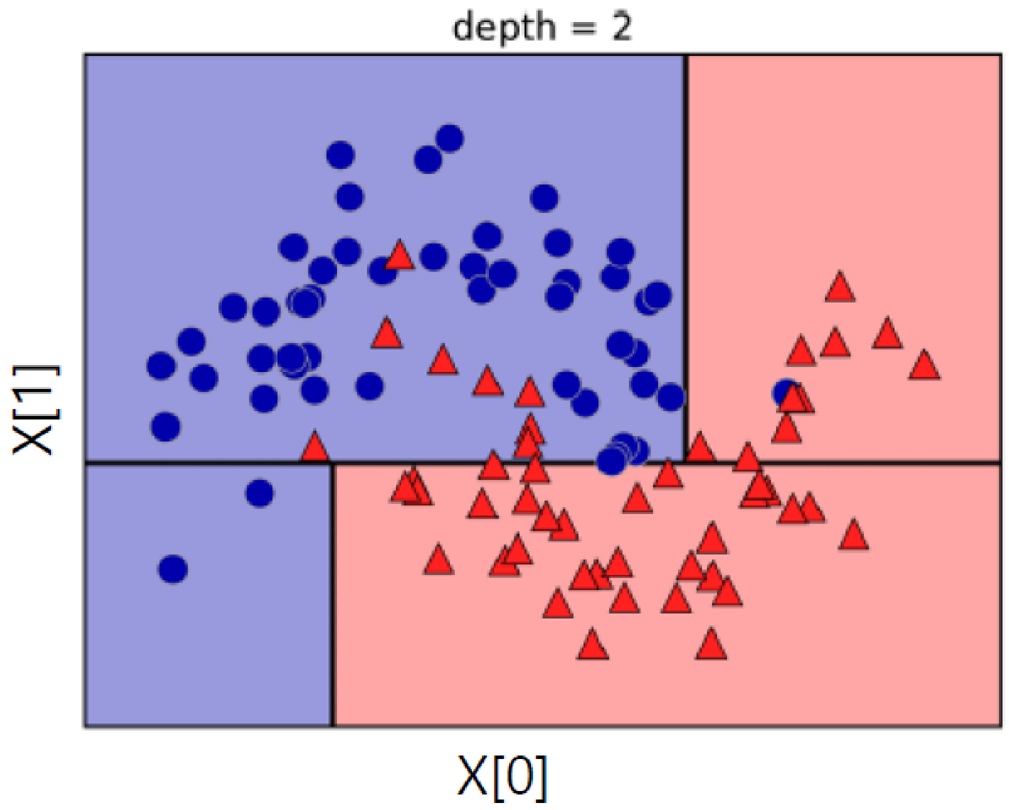
- **Example of building a decision tree (for classification)**
  - The algorithm searches over all possible tests and finds the one that is most informative about the target.

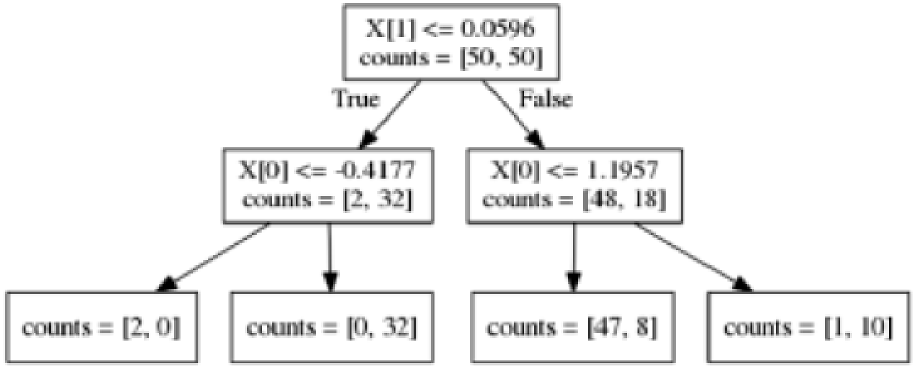




Decision Trees

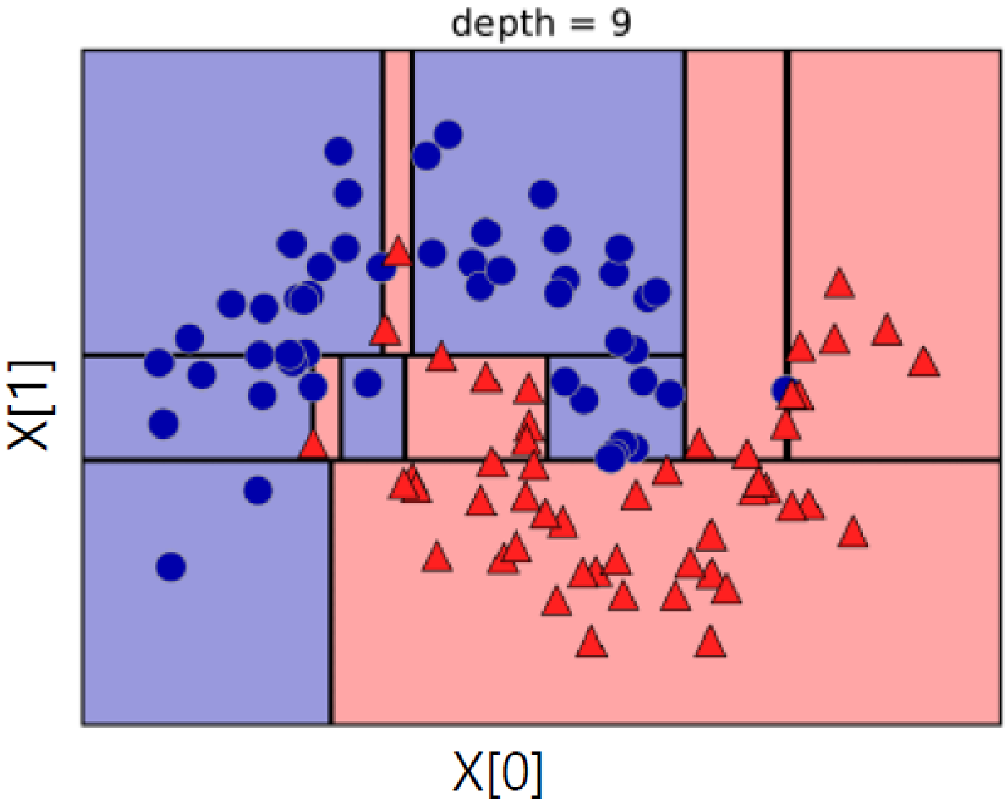
- **Example of building a decision tree (for classification)**
  - We can build a more accurate model by repeating the process of looking for the most informative next split for the left and the right region.

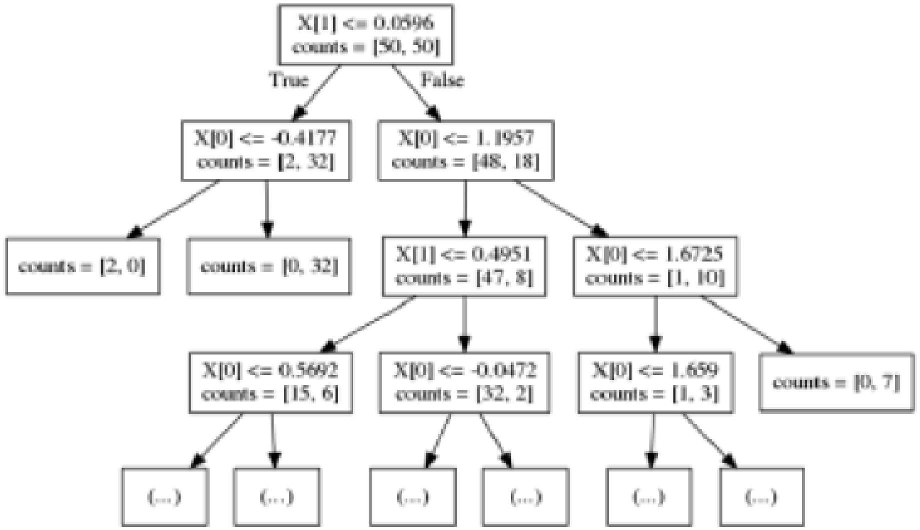




Decision Trees

- **Example of building a decision tree (for classification)**
  - The recursive partitioning of the data is repeated until each region in the partition (each leaf in the tree) become homogeneous.





Decision Trees

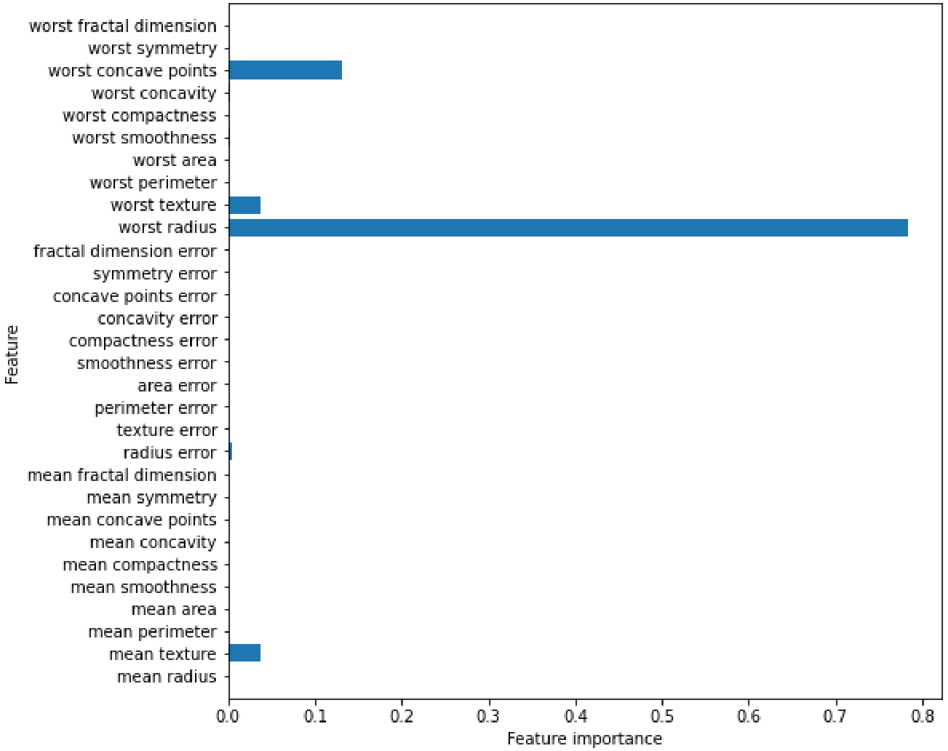
- **To make a prediction for a new data point, we traverse the tree to find the leaf the data point falls into.**

- **For Classification**
  - The output for the data point is the majority class of the training points in this leaf.

- **For Regression**
  - The output for the data point is the mean target of the training points in this leaf.

Feature Importance in Decision Trees

- **feature importance summarizes the workings of a tree by rating how important each feature is for the decision the tree makes.**
  - The importance of a feature is computed as the (normalized) **total reduction of the criterion** brought by that feature.
  - It is a number between 0 and 1 for each feature, where 0 means "not used at all" and 1 means "perfectly predicts the target."



Decision Trees

- **Strengths**
  - Decision trees work well when you have a mix of continuous and categorical features.
  - The algorithms are completely invariant to scaling of the data. (no data scaling is needed)
  - Feature selection & reduction is automatic.
  - It is robust to noise.
  - The resulting model can easily be visualized and understood.

- **Weaknesses**
  - Even with the use of pre-pruning, they tend to overfit and provide poor generalization performance.

    → the ensemble methods are usually used in place of a single decision tree.
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the optimal tree.
  - quite **unstable**.
    - A few changes in the training dataset can create a completely different tree.
  - **lack of smoothness.**
    - Slight changes in the input feature can have a big impact on the predicted outcome, which is usually not desirable.

Random Forest

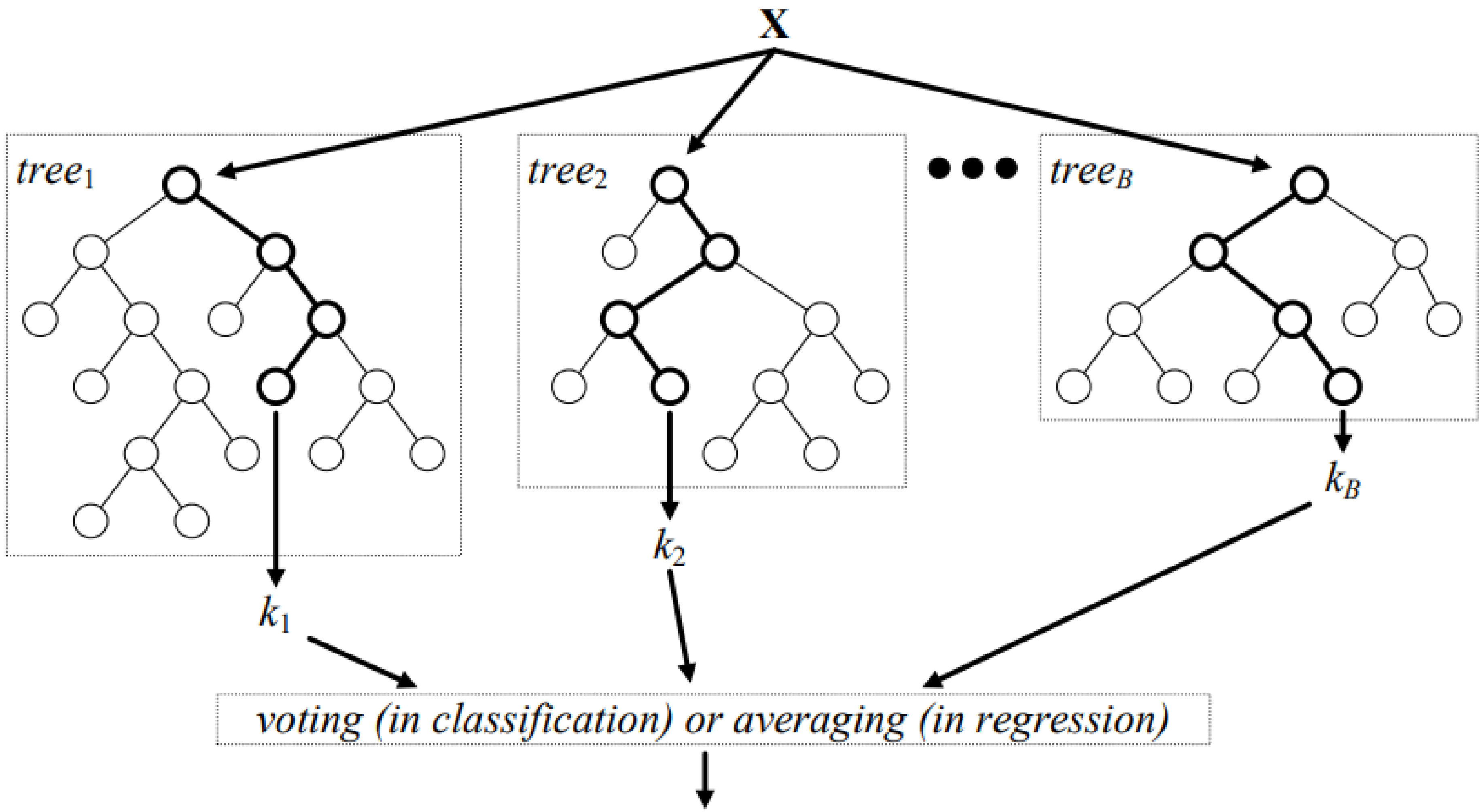
- A specialized **bagging** for decision tree algorithms

- An ensemble of decision trees, where each tree is slightly different from the others by **injecting additional randomness into the tree building**
  - Each tree might do a relatively good job of predicting, but will likely overfit on part of the data in different ways.
  - If we build many trees, we can reduce the amount of overfitting by averaging their results while retaining the predictive power of the trees.
  - Random forests get their name from injecting randomness into the tree building to ensure each tree is different.

Random Forest

- Two ways in which the trees in a random forest are randomized
  - **Bagging**
    - bootstrap: It leads to each decision tree in the random forest being built on a slightly different dataset.
      - → From a list ['a', 'b', 'c', 'd'], possible examples of bootstrap samples are ['b', 'd', 'd', 'c'] and ['d', 'a', 'd', 'a'].
  - **Randomly chosen features in each split test (Randomized tree)**
    - in each node, the algorithm randomly selects a subset of the features, and it looks for the best possible test involving one of these features
      - → each node in a tree can make a decision using a different subset of the features.

Random Forest



Random Forest

- To make a prediction for a new data point, we first make a prediction for every tree in the forest.

- For Classification,
  - Each tree makes a “soft” prediction, providing a probability for each possible output label.
  - The probabilities predicted by all the trees are averaged.
  - The output for the data point is the class with the highest average probability.

- For Regression
  - The output for the data point is the mean prediction of the trees in the forest.

Feature Importance in Random forest

- Similarly to the decision tree, the random forest provides feature importances
  - Computed by aggregating the feature importances over the trees in the forest.
  - Typically, the feature importances provided by the random forest are more reliable than the ones provided by a single tree.

Interpretable Models:
Decision Rules

Decision Rules

- Classify records by using a collection of “if…then…” rules

- Rule: IF Condition → THEN Y
  - where
    - Condition is a conjunction of tests on attributes
    - y is the class label
  - Examples of classification rules:
    - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) → Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) → Evade=No

Decision Rules

Training Data

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

Model:
Rule-Based Classifier

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Characteristics of Rules

- **Coverage** of a rule:
  – Fraction of records that satisfy the antecedent of a rule
- **Accuracy** of a rule:
  – Fraction of records that satisfy the antecedent that also satisfy the consequent of a rule

Coverage-Accuracy Tradeoff

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Class**
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**

(Status=Single) $\rightarrow$ No

Coverage = 40%, Accuracy = 50%

Characteristics of Rules

- **Mutually exclusive** rule set
  - The rules are independent of each other
  - Every object is covered by at most one rule

- A rule set is not mutually exclusive
  - An object may trigger more than one rule
  - Solution?
    - Unordered rule set – use voting schemes
    - Ordered rule set – use highest rank rule
      - The rules are rank-ordered according to their priority
      - An ordered rule set is known as a decision list
      - An object is assigned to the class label of the highest ranked rule it has triggered

Characteristics of Rules

- **Exhaustive** rule set
  - The rules account for every possible combination of attribute values
  - Each object is covered by at least one rule

- A rule set is not exhaustive
  - An object may not trigger any rules
  - Solution?
    - Use a default class

Building Classification Rules

- Direct Method:
  - Extract rules directly from data
  - Examples: RIPPER

- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - Examples: C4.5rules

OneR (Learning Rules from a Single Feature)

1. Discretizing all features

location	size	pets	value
good	small	yes	high
good	big	no	high
good	big	no	high
bad	medium	no	medium
good	medium	only cats	medium
good	small	only cats	medium
bad	medium	yes	medium
bad	small	yes	low
bad	medium	yes	low
bad	small	no	low

2. Create a cross table

	value=low	value=medium	value=high
location=bad	3	2	0
location=good	0	2	3

	value=low	value=medium	value=high
size=big	0	0	2
size=medium	1	3	0
size=small	2	1	1

	value=low	value=medium	value=high
pets=no	1	1	2
pets=only cats	0	2	0
pets=yes	2	1	1

3. Create a rule which predicts the most frequent class
4. Calculate the total error
5. Select the feature with the smallest total error

IF `size=small` THEN `value=low`
IF `size=medium` THEN `value=medium`
IF `size=big` THEN `value=high`

A OneR model is a decision tree with only one split.

RIPPER (Sequential covering)

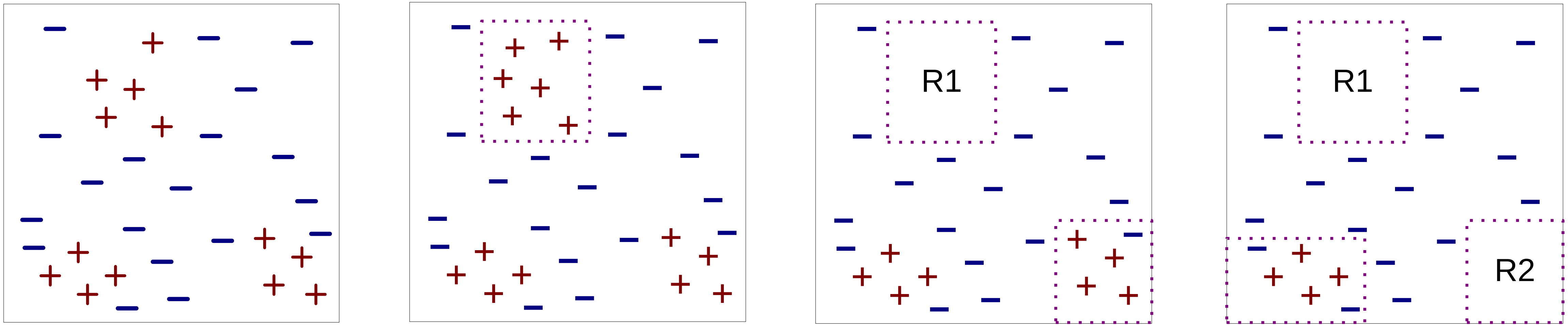
- **RIPPER – Repeated Incremental Pruning to Produce Error Reduction**

- **For 2-class problem,**
  - Choose one of the classes as positive class, and the other as negative class
  - Learn the rule set for positive class
  - Negative class will be default class

- **For multi-class problem,**
  - Order the classes according to increasing class prevalence (or decreasing importance)
  - Learn the rule set for the first class as positive class (treat the rest as negative class)
  - Repeat with the next class as positive class

RIPPER (Sequential covering)

- Learning a rule set by sequential covering algorithm
  - Start from an empty rule set
  - Repeat the following until stopping criterion is met
    - Find the best rule that covers the current set of positive objects
    - Add the rule to the rule set
    - Eliminate objects covered by the rule
  - Example:



(i) Original Data (ii) Step 1 (iii) Step 2 (iv) Step 3

Advantages & Disadvantages

- Advantages
  - IF-THEN rules are **easy to interpret**.
  - The **prediction with IF-THEN rules is fast**
  - IF-THEN rules usually generate sparse models, which means that not many features are included. They **select only the relevant features** for the model.

- Disadvantages
  - focuses on classification and almost **completely neglects regression.**
  - numeric features must be categorized if you want to use them.

Interpretable Models: **RuleFit**

Friedman, Jerome H, and Bogdan E Popescu. “Predictive learning via rule ensembles.” The Annals of Applied Statistics. JSTOR, 916–54. (2008)

RuleFit

- Simple linear relationship + feature intractions
- RuleFit learns a **sparse linear model** with the original features and also a number of **new features(interactions)** that are decision rules.
- RuleFit automatically generates these features from decision trees (any tree ensemble algorithm can be used. E.g. gradient boosting, random forest).

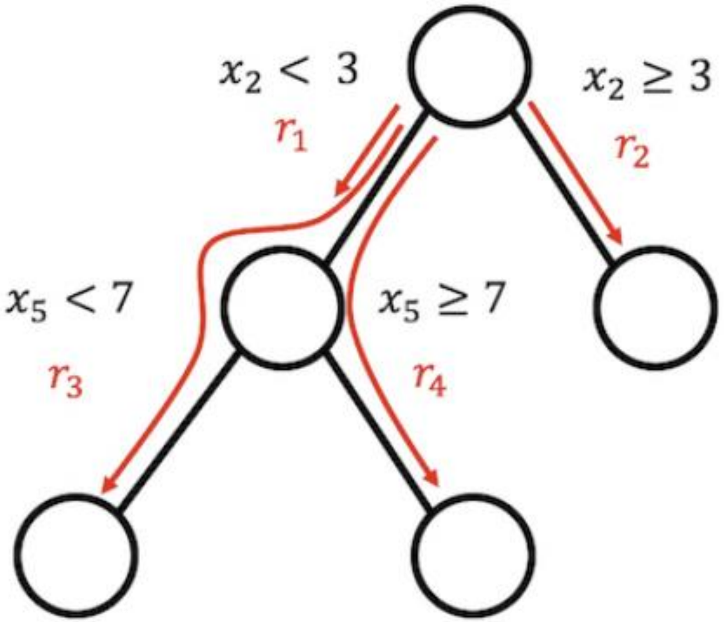


FIGURE 5.21: 4 rules can be generated from a tree with 3 terminal nodes.

RuleFit Algorithm

- RuleFit consists of two components:

1. Create **Rules** from decision trees. (*Rule generation*)
2. **Fit** a linear model with the original features and the new rules as input. (*Sparse linear model*)

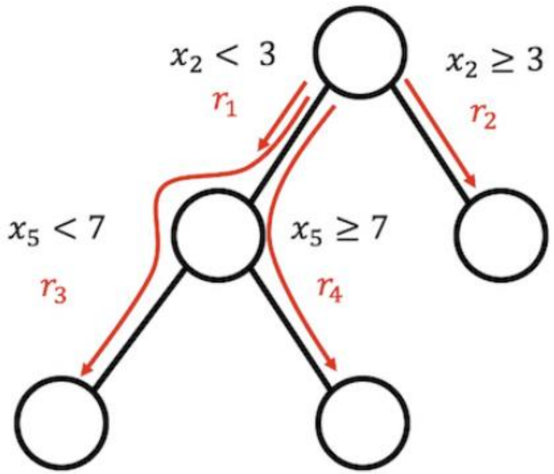
RuleFit Algorithm

- Rule generation
  - The rules are constructed by decomposing decision trees

any **tree ensemble algorithm** can be used to generate the trees for RuleFit. (random forest, AdaBoost)
A tree ensemble can be described with this general formula:

$$\hat{f}(x) = a_0 + \sum_{m=1}^{M} a_m \hat{f}_m(X)$$

RuleFit **extracts all possible rules from a tree**, not only from the leaf nodes



discard the predicted value in each node and only keep the conditions

the number of rules created from an ensemble of M trees with t_m terminal nodes each is:

$$K = \sum_{m=1}^{M} 2(t_m - 1)$$

** learn trees with random depth so that many diverse rules with different lengths are generated*

RuleFit Algorithm

- Sparse linear model
  - Every rule and every original feature becomes a feature in the linear model and gets a weight estimate.

1. **Winsorize the original features**

$$l_j^*(x_j) = min(\delta_j^+, max(\delta_j^-, x_j))$$

δ quantiles of the data distribution, As a rule of thumb, you can choose δ = 0.025

2. **Normalize**

$$l_j(x_j) = 0.4 \cdot l_j^*(x_j)/std(l_j^*(x_j))$$

The 0.4 is the average standard deviation of rules with a uniform support distribution of $s_k \sim U(0,1)$.

   - so that they have the same prior importance as a typical decision rule

3. **Train a sparse linear model (Lasso)**

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{k=1}^{K} \hat{\alpha}_k r_k(x) + \sum_{j=1}^{p} \hat{\beta}_j l_j(x_j)$$

$$(\{\hat{\alpha}\}_1^K, \{\hat{\beta}\}_0^p) = argmin_{\{\hat{\alpha}\}_1^K, \{\hat{\beta}\}_0^p} \sum_{i=1}^{n} L(y^{(i)}, f(x^{(i)})) + \lambda \cdot \left(\sum_{k=1}^{K} |\alpha_k| + \sum_{j=1}^{p} |\beta_j| \right)$$

RuleFit

- Interpretation & Example
  - The interpretation is the **<u>same as for “normal” linear models</u>**. The only difference is that the model has **<u>new features derived from decision rules</u>**. Decision rules are binary features: A value of 1 means that all conditions of the rule are met, otherwise the value is 0.
  - Example : predict the number of rented bicycles on a given day

five of the rules that were generated by RuleFit

Description	Weight	Importance
days_since_2011 > 111 & weathersit in (“GOOD”, “MISTY”)	795	303
37.25 <= hum <= 90	-20	278
temp > 13 & days_since_2011 > 554	676	239
4 <= windspeed <= 24	-41	204
days_since_2011 > 428 & temp > 5	356	174

- If days_since_2011 > 111 & weathersit in (“GOOD”, “MISTY”), then the predicted number of bikes increases by 795, when all other feature values remain fixed.
- In total, 278 such rules were created from the original 8 features. Quite a lot! But thanks to Lasso, only 59 of the 278 have a weight different from 0.

Advantages & Disadvantages

- RuleFit automatically adds **feature interactions** to linear models.
  - We don't need to manually add interaction terms.
- RuleFit can handle both classification and regression tasks.
- To avoid too large rule(for interpretability), set the maximum depth of the trees small.

- Interpretable?
  - The interpretability degrades with increasing number of features in the model.
  - since it is a linear model, the weight interpretation is still unintuitive.

If the trained model has following rules… *"temp > 10"*

"temp > 15 & weather='GOOD'"

…

The interpretation of the estimated weight:
"Assuming all other features remain fixed, the predicted number of bikes increases by β2 when the weather is good and temperature above 15 degrees."

Interpretable Models: **Others**

Naïve Bayes Classifier

- Bayes Theorem
  - A probabilistic framework for solving classification problems
  - Conditional Probability:

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)}$$

$$P(X \mid Y) = \frac{P(X,Y)}{P(Y)}$$

  - Bayes Theorem:

$$P(Y \mid X) = \frac{P(X \mid Y)P(Y)}{P(X)}$$

Naïve Bayes Classifier

- Example
  - Given:
    - A doctor knows that meningitis causes stiff neck 50% of the time → $P(S|M)$
    - Prior probability of any patient having meningitis is 1/50,000 → $P(M)$
    - Prior probability of any patient having stiff neck is 1/20 → $P(S)$
  - If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Naïve Bayes Classifier

- Consider each attribute and class label as random variables
- Given a record with attributes (X1, X2,…, Xd), the goal is to predict class Y
  - Specifically, we want to find the value of Y that maximizes P(Y| X1, X2,…, Xd)
- Can we estimate P(Y| X1, X2,…, Xd) directly from data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

Naïve Bayes Classifier

- Approach:
  - compute posterior probability $P(Y \mid X_1, X_2, \ldots, X_d)$ using the Bayes theorem

$$\boxed{P(Y \mid X_1 X_2 \ldots X_n) = \frac{P(X_1 X_2 \ldots X_d \mid Y) P(Y)}{P(X_1 X_2 \ldots X_d)}}$$

  - *Maximum a-posteriori* : Choose Y that maximizes
    $P(Y \mid X_1, X_2, \ldots, X_d)$

  - Equivalent to choosing value of Y that maximizes
    $P(X_1, X_2, \ldots, X_d \mid Y)\ P(Y)$

- How to estimate $P(X_1, X_2, \ldots, X_d \mid Y)$?

Naïve Bayes Classifier

- Conditional Independence
  - X and Y are conditionally independent given Z if $P(X, Y|Z) = P(X|Z)\ P(Y|Z)$
  - Example: Arm length and reading skills
    - Young child has shorter arm length and limited reading skills, compared to adults
    - If age is fixed, no apparent relationship between arm length and reading skills
    - Arm length and reading skills are conditionally independent given age
  - Assume conditional independence,
  - then $P(X_1, X_2, \ldots, X_d\,|Y) = P(X_1|Y)\,P(X_2|Y) \ldots P(X_d|Y)$

Naïve Bayes Classifier

- Assume conditional independence among attributes X_i when class is given:
  - $P(X_1, X_2, \ldots, X_d | Y) = P(X_1 | Y) P(X_2 | Y) \ldots P(X_d | Y)$
  - In training phase, we can estimate $P(X_i | Y)$ for all X_i and Y combinations from the training data
  - In the test phase, new point is classified to Y if $P(Y)P(X_1|Y)P(X_2|Y) \ldots P(X_d|Y)$ is maximal.

Interpretability :
clear for each feature how much it contributes towards a certain class prediction

Naïve Bayes Classifier

- **Estimating Probabilities $P(X_j|Y)$ from Data**
  - For categorical attributes:
    - $P(X_j = a | Y = c) = P(X_j = a, Y = c)/P(Y = c)$

$$= \frac{\text{number of training objects whose class label } Y \text{ is } c \text{ and attribute value for } X_j \text{ is } a}{\text{number of training objects whose class label } Y \text{ is } c}$$

  - For continuous attributes:
    - Discretization – Partition the range into bins
      - Replace continuous value with bin value
      - Attribute changed from continuous to categorical (ordinal)
    - Probability density estimation
      - Assume attribute follows a probability distribution (e.g., a normal distribution)
      - Use data to estimate the parameters of the distribution (e.g., mean and variance)
      - Once the probability distribution is estimated, use it to estimate the conditional probability $P(X_j|Y)$

Naïve Bayes Classifier

- Example: Tax-Evasion Detection
  - Given a query object

$$\mathbf{X} = (\text{Refund} = \text{No}, \text{Marital Status} = \text{Divorced}, \text{Income} = 120\text{K})$$

  - Can we estimate $P(\text{Evade} = \text{Yes}|\mathbf{X})$ and $P(\text{Evade} = \text{No}|\mathbf{X})$ from the training dataset?

Training Dataset

Tid	Refund	Marital Status	Taxable Income	Evade
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

Conditional Independence

- P(**X** | Yes) =
  P(Refund = No | Yes) x
  P(MS = Divorced | Yes) x
  P(Income = 120K | Yes)

- P(**X** | No) =
  P(Refund = No | No) x
  P(MS = Divorced | No) x
  P(Income = 120K | No)

Naïve Bayes Classifier

- Example: Tax-Evasion Detection
  - Given a query object

$$\mathbf{X} = (\text{Refund} = \text{No}, \text{Marital Status} = \text{Divorced}, \text{Income} = 120\text{K})$$

For Refund:

P(Refund = Yes | Yes) = 0
P(Refund = No | Yes) = 1

For Marital Status:

P(Marital Status = Single | No) = 2/7
P(Marital Status = Divorced | No) = 1/7
P(Marital Status = Married | No) = 4/7
P(Marital Status = Single | Yes) = 2/3
P(Marital Status = Divorced | Yes) = 1/3
P(Marital Status = Married | Yes) = 0

For Taxable Income:

If class = No: sample mean = 110
sample variance = 2975
If class = Yes: sample mean = 90
sample variance = 25
Assume Normal Distribution

- P(**X** | No) = P(Refund=No | No)
  * P(Marital Status = Divorced | No)
  * P(Income=120K | No)
  $= 4/7 \times 1/7 \times 0.0072 = 0.0006$

- P(**X** | Yes) = P(Refund=No | Yes)
  * P(Marital Status = Divorced | Yes)
  * P(Income=120K | Yes)
  $= 1 \times 1/3 \times 1.2 \times 10^{-9} = 4 \times 10^{-10}$

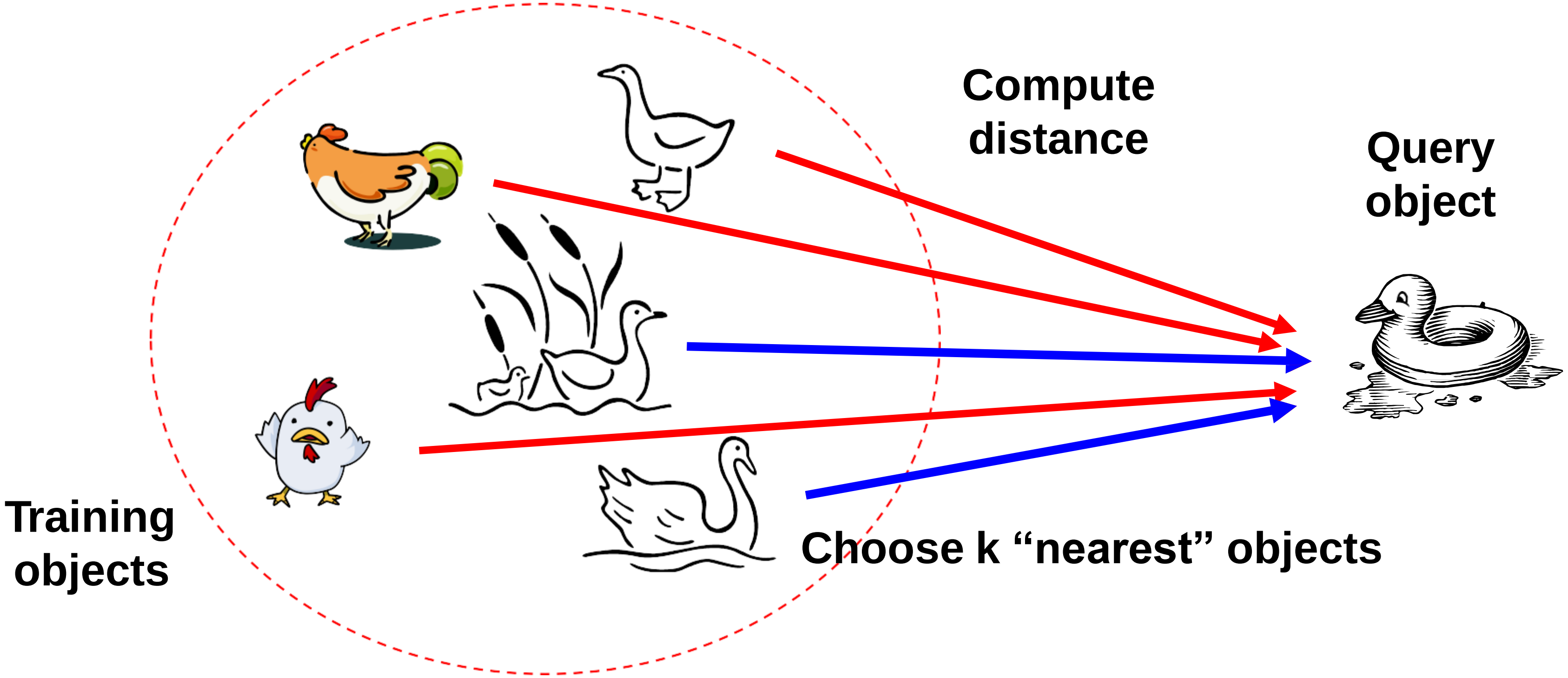
P(**X**|No)P(No) > P(**X**|Yes)P(Yes)

→ P(No|**X**) > P(Yes|**X**)

→ Class = No

K-Nearest Neighbors

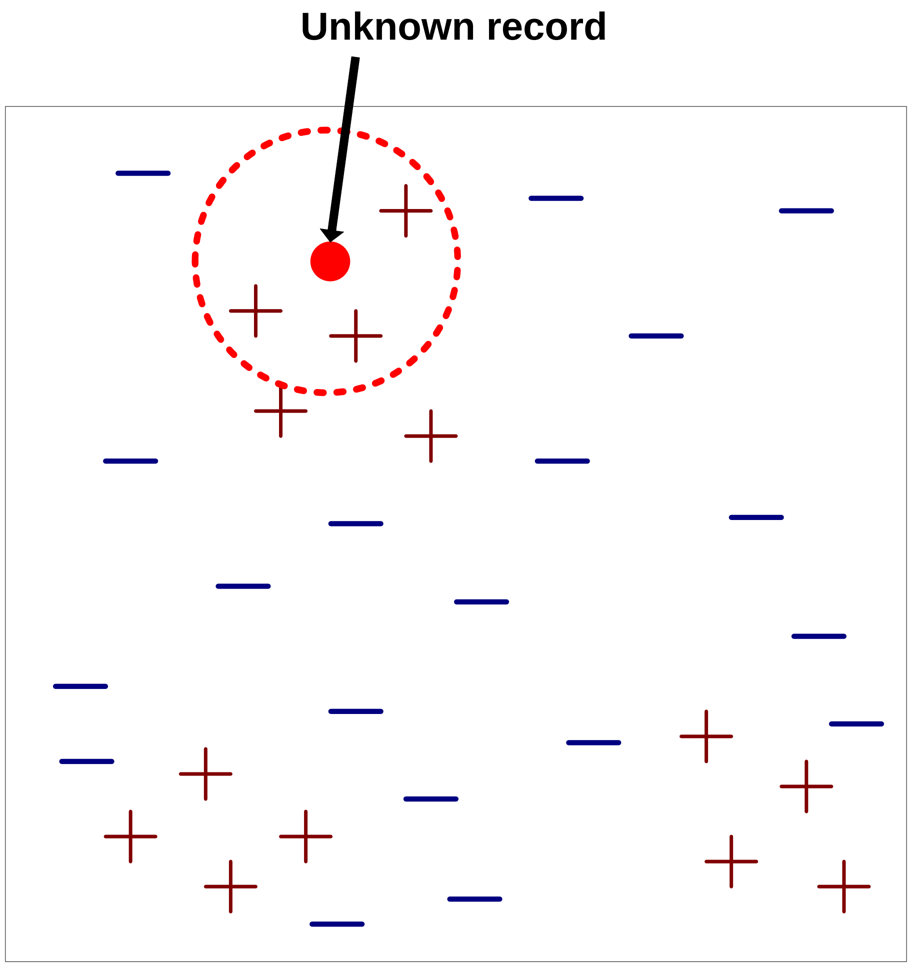
- Basic idea: If it walks like a duck, quacks like a duck, then it's probably a duck



K-Nearest Neighbors

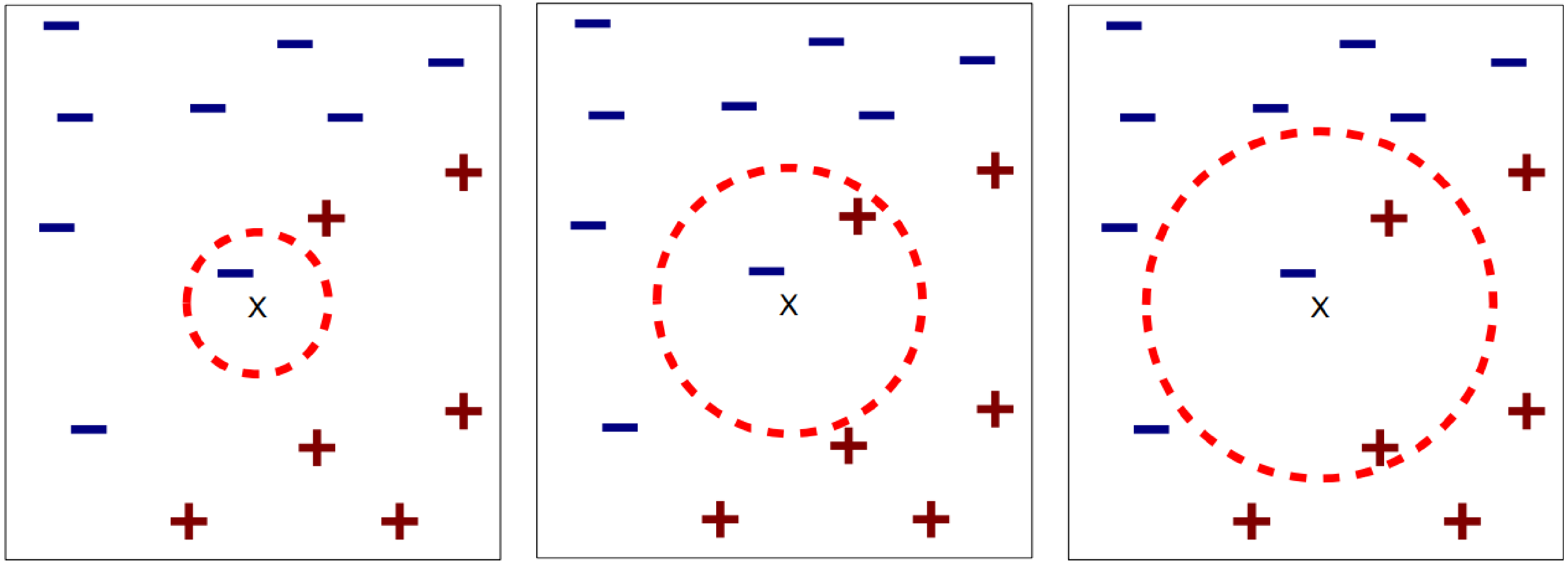
- **Training Phase: Prepare the following**
  - **Training data**, the set of labeled objects
  - **Distance measure** to compute distance between objects
  - **k**, the number of nearest neighbors to retrieve
  - **Weighting scheme** to aggregate the class labels of nearest neighbors

- **Test Phase: Classify a query object**
  - Compute distance to training objects
  - Identify k nearest neighbors
  - Use the class labels of nearest neighbors to determine the class label of the query object



K-Nearest Neighbors

- Choosing the number of nearest neighbors k
  - Smaller k → capture local structure in data (but also noise)
  - Larger k → provide more smoothing, less noise, but may miss local structure



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-Nearest Neighbors

- Advantages:
  - Training is not required (do not build a model explicitly)
  - Can produce arbitrarily shaped decision boundaries
  - Decisions are easy to understand

- Disadvantages:
  - Classifying objects are relatively expensive
  - Selection of right distance measure is essential but difficult
  - Superfluous or redundant attributes can create problems

    : Distance between neighbors could be dominated by these attributes

K-Nearest Neighbors

- instance-based learning algorithm.
  - → no interpretability on a modular level.

- To explain a prediction, you can always retrieve the k neighbors that were used for the prediction.
  - → If an instance consists of hundreds or thousands of features, then it is not interpretable, I would argue. But if you have few features or a way to reduce your instance to the most important features, presenting the k-nearest neighbors can give you good explanations.