INTRODUCTION
TO
MACHINE LEARNING

MODULE 2

How machines learn - Data storage, Abstraction, Generalisation, Evaluation Machine learning in practice - Types of machine learning algorithms.

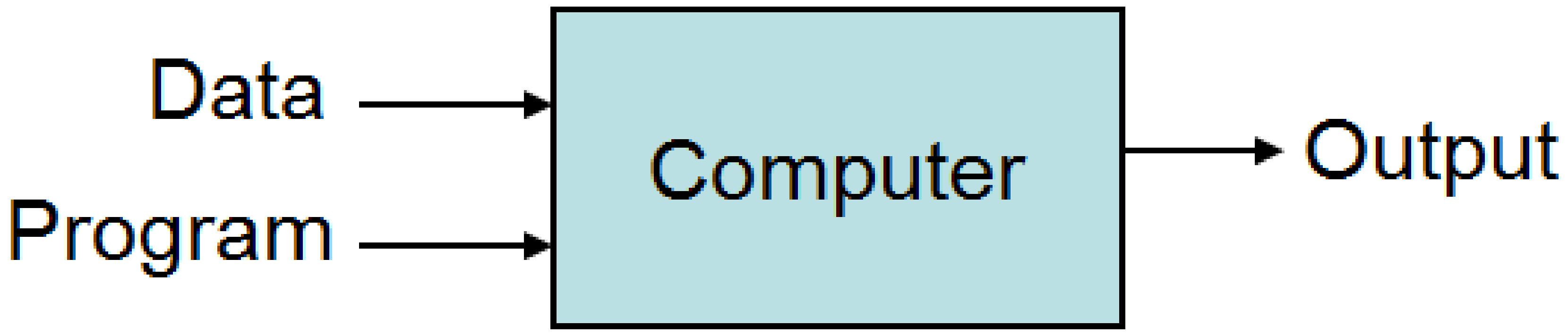
Lazy learning: Classification using K-Nearest Neighbour algorithm Measuring similarity with distance, Choice of k, Preparing data for use with k-NN.

Probabilistic learning: Understanding Naive Bayes - Conditional probability and Bayes theorem, Naive Bayes algorithm for classification, The Laplace estimator, Using numeric features with Naive Bayes.

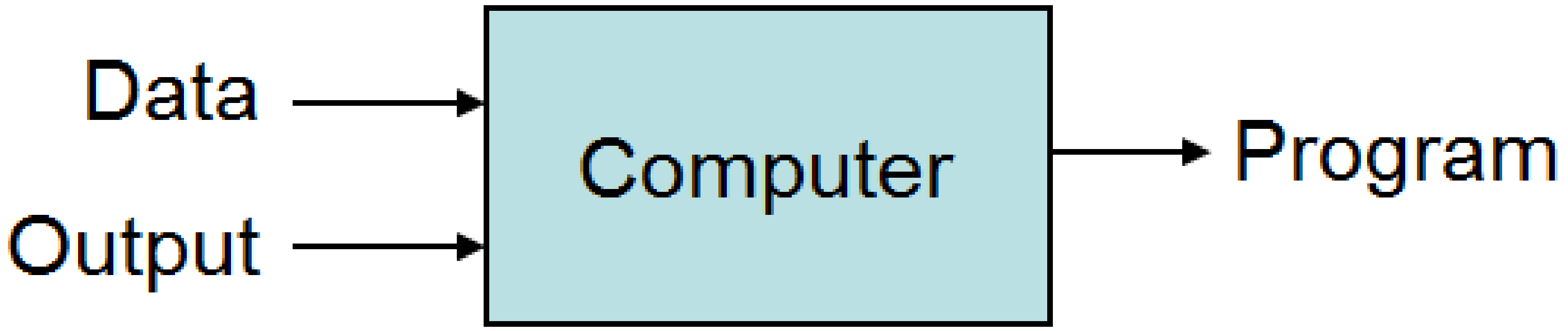
INTRODUCTION TO MACHINE LEARNING

- The field of machine learning provides a set of algorithms that transform data into actionable knowledge.
- The field of study interested in the development of computer algorithms to transform data into intelligent action is known as **machine learning**.
- A closely related sibling of machine learning, data mining, is concerned with the generation of novel insights from large databases.
- Although there is some disagreement over how widely machine learning and data mining overlap, a potential point of distinction is that machine learning focuses on teaching computers how to use data to solve a problem, while data mining focuses on teaching computers to identify patterns that humans then use to solve a problem.

Traditional Programming



Machine Learning



HOW MACHINES LEARN

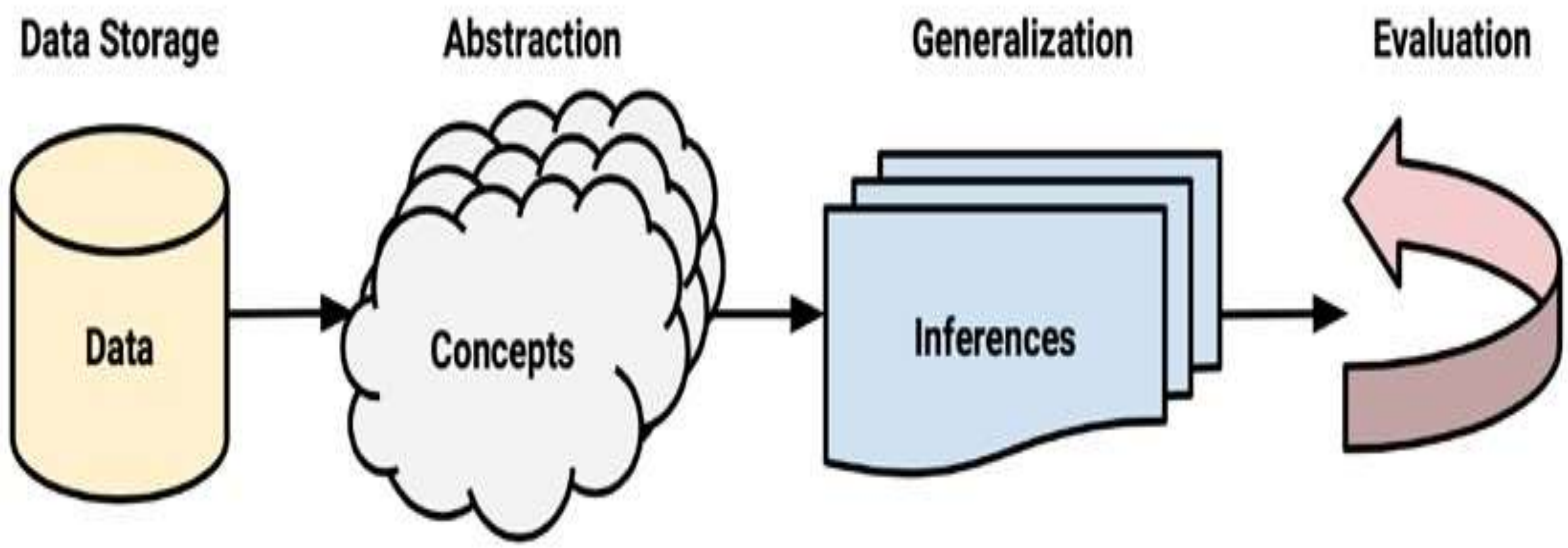
- A formal definition of machine learning proposed by computer scientist Tom M Mitchell states that
- **“A machine learns whenever it is able to utilize its ‘an experience’ such that its performance improves on similar experiences in the future”.**
- Human brains are naturally capable of learning from birth, but the conditions necessary for computers to learn must be made explicit.
- Regardless of whether the learner is a human or machine the basic learning process is similar.

It can be divided into four interrelated Components -

- Data storage
- Abstraction
- Generalization
- Evaluation

1. Data storage -
    - Utilizes observation memory, and recall to provide a factual basis for further reasoning.
2. Abstraction -
    - Involves the translation of stored data into broader representations and concepts.
3. Generalization -
    - Uses abstracted data to create knowledge and inferences that drive action in new contexts.
4. Evaluation -
    - Provides a feedback mechanism to measure the utility of learned knowledge and inform potential improvements.

STEPS IN THE LEARNING PROCESS:



1. DATA STORAGE

- All learning must begin with data.
- Humans and computers utilize data storage as a foundation for more advanced reasoning.
- In a human being this consists of a brain that uses electrochemical signals in a network of biological cells to store and process observations for short and long term future recall.
- Computers uses hard disk drives, flash memory and random access memory (RAM) in combination with a central processing unit (CPU).

2. ABSTRACTION

- This work of assigning meaning to stored data occurs during the abstraction process, in which raw data comes to have a more abstract meaning.
- **Knowledge representation**, the formation of logical structures that assist in turning raw sensory information into a meaningful insight.
- During a machine's process of knowledge representation, the computer summarizes stored raw data using **a model**, an explicit description of the patterns within the data.

There are many different types of models. You may be already familiar with some. Examples include:

- Mathematical equations
- Relational diagrams such as trees and graphs
- Logical if/else rules
- Groupings of data known as clusters

2. ABSTRACTION (CONTINUED)

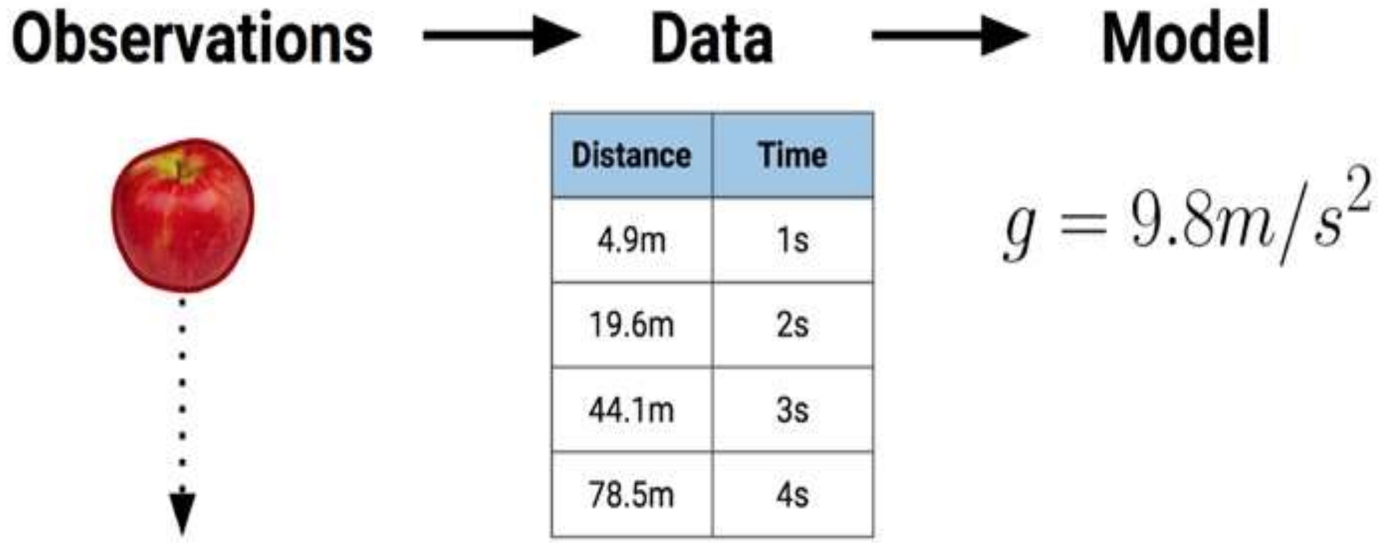
TRAINING

- The process of fitting a model to a dataset is known as training.
- When the model has been trained, the data is transformed into an abstract form that summarizes the original information.

2. ABSTRACTION (CONTINUED)

Eg: -discovery of gravity.

- By fitting equations to observational data, Sir Isaac Newton inferred the concept of gravity. But the force we now know as gravity was always present. It was not recognized until Newton recognized it as an abstract concept that relates some data to others.
  - i.e, by becoming the ‘g’ term in a model that explains observations of falling objects.



Distance	Time
4.9m	1s
19.6m	2s
44.1m	3s
78.5m	4s

$$g = 9.8m/s^2$$

3. GENERALIZATION

- The term generalization describes the process of turning abstracted knowledge into a form that can be utilized for future action, on tasks that are similar, but not identical.
- Process is a bit difficult to describe.
- It has been imagined as a search through the entire set of models (inferences) that could be abstracted during training.
- In other words, if you can imagine a hypothetical set containing every possible theory that could be established from the data, generalization involves the reduction of this set into a manageable number of important findings.

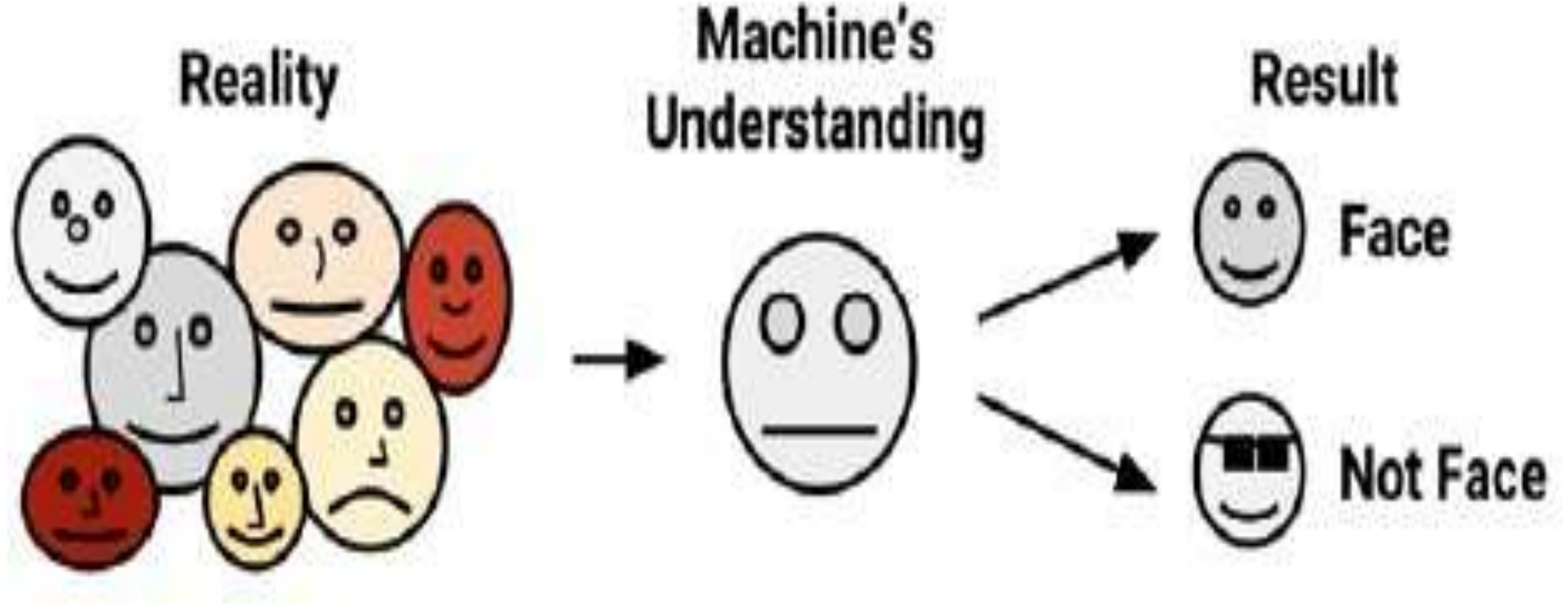
3. GENERALIZATION (CONTINUED)

- In generalization, the learner is tasked with limiting the patterns it discovers, to only those that will be most relevant to its future tasks.
- Generally it is not feasible to reduce the number of patterns by examining them one by one and ranking them by future utility.
- So, machine learning algorithms generally employ shortcuts that reduce the search space more quickly.
- The algorithm will uses **heuristics technique** designed for solving a problem more quickly.

3. GENERALIZATION (CONTINUED)

- Heuristics are routinely used by human beings to quickly generalize experience to new scenarios.
- If you have ever utilized your gut instinct to make a snap decision prior to fully evaluating your circumstances, you were intuitively using mental heuristics.
- The incredible human ability to make quick decisions often relies not on computer-like logic, but rather on heuristics guided by emotions.
- Sometimes, this can result in illogical conclusions.

3. GENERALIZATION (CONTINUED)



4. EVALUATION

- The final step in the generalization process is to evaluate or measure the learner's success in spite of its biases and use this information to inform additional training if needed.
- Generally, evaluation occurs after a model has been trained on an initial training dataset.
- Then, the model is evaluated on a new test dataset in order to judge how well its characterization of the training data generalizes to new, unseen data.
- It's worth noting that it is exceedingly rare for a model to perfectly generalize to every unforeseen case.

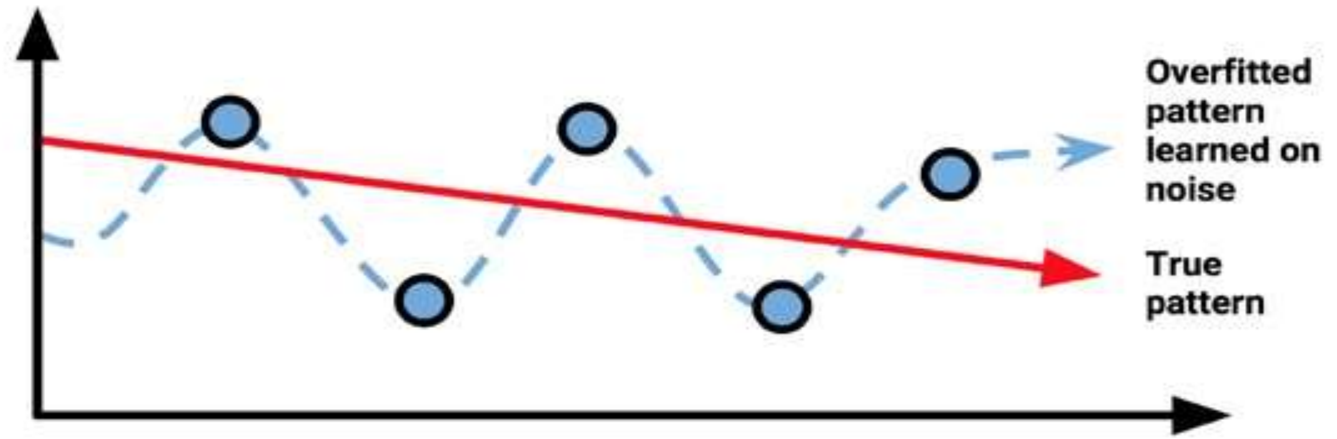
4. EVALUATION (CONTINUED)

In parts, models fail to perfectly generalize due to the problem of noise, a term that describes unexplained or unexplainable variations in data. Noisy data is caused by seemingly random events, such as :

- Measurement error due to imprecise sensors that sometimes add or subtract a bit from the readings.
- Issues with human subjects, such as survey respondents reporting random answers to survey questions, in order to finish more quickly.
- Data quality problems, including missing, null, truncated, incorrectly coded, or corrupted values.
- Phenomena that are so complex or so little understood that they impact the data in ways that appear to be unsystematic.

4. EVALUATION (CONTINUED)

- Trying to model noise is the basis of a problem called **overfitting**.
- Because most noisy data is unexplainable by definition, attempting to explain the noise will result in erroneous conclusions that do not generalize well to new cases.
- Efforts to explain the noise will also typically result in more complex models that will miss the true pattern that the learner tries to identify.
- A model that seems to perform well during training, but does poorly during evaluation, is said to be overfitted to the training dataset, as it does not generalize well to the test dataset



MACHINE LEARNING IN PRACTICE

To apply the learning process to real-world tasks, we'll use a five-step process. Regardless of the task at hand, any machine learning algorithm can be deployed by following these steps:

- Data collection
- Data exploration and preparation
- Model training
- Model evaluation
- Model improvement

1. DATA COLLECTION

- The data collection step involves gathering the learning material an algorithm will use to generate actionable knowledge.
- In most cases, the data will need to be combined into a single source like a text file, spreadsheet, or database.

2. DATA EXPLORATION AND PREPARATION

- The quality of any machine learning project is based largely on the quality of its input data.
- Thus, it is important to learn more about the data and its nuances during a practice called data exploration.
- Additional work is required to prepare the data for the learning process.
- This involves fixing or cleaning so-called "messy" data, eliminating unnecessary data, and recoding the data to conform to the learner's expected inputs.

3. MODEL TRAINING

- By the time the data has been prepared for analysis, you are likely to have a sense of what you are capable of learning from the data.
- The specific machine learning task chosen will inform the selection of an appropriate algorithm, and the algorithm will represent the data in the form of a model.

4. MODEL EVALUATION

- Because each machine learning model results in a biased solution to the learning problem, it is important to evaluate how well the algorithm learns from its experience.
- Depending on the type of model used, you might be able to evaluate the accuracy of the model using a test dataset or you may need to develop measures of performance specific to the intended application.

5. MODEL IMPROVEMENT

- If better performance is needed, it becomes necessary to utilize more advanced strategies to augment the performance of the model.
- Sometimes, it may be necessary to switch to a different type of model altogether.
- You may need to supplement your data with additional data or perform additional preparatory work as in step two of this process.

TYPES OF INPUT DATA

UNIT OF OBSERVATION

- It is used to describe the smallest entity with measured properties of interest for a study.
- Commonly, the unit of observation is in the form of persons, objects or things, transactions, time points, geographic regions, or measurements.
- Sometimes, units of observation are combined to form units such as person-years, which denote cases where the same person is tracked over multiple years; each person-year comprises of a person's data for one year.

TYPES OF INPUT DATA (CONTINUED)

UNIT OF ANALYSIS

- It is the smallest unit from which the inference is made.
- Although it is often the case, the observed and analyzed units are not always the same.
- For example, data observed from people might be used to analyze trends across countries.

TYPES OF INPUT DATA (CONTINUED)

DATASETS

That store the units of observation and their properties can be imagined as collections of data consisting of:

- **Examples:** instances of the unit of observation for which properties have been recorded.
- **Features:** recorded properties or attributes of examples that may be useful for learning.

TYPES OF INPUT DATA (CONTINUED)

E.g to build a learning algorithm to identify spam e mail

- Unit of observation – e-mail messages
- Examples - specific messages
- Features - consist of the words used in the messages

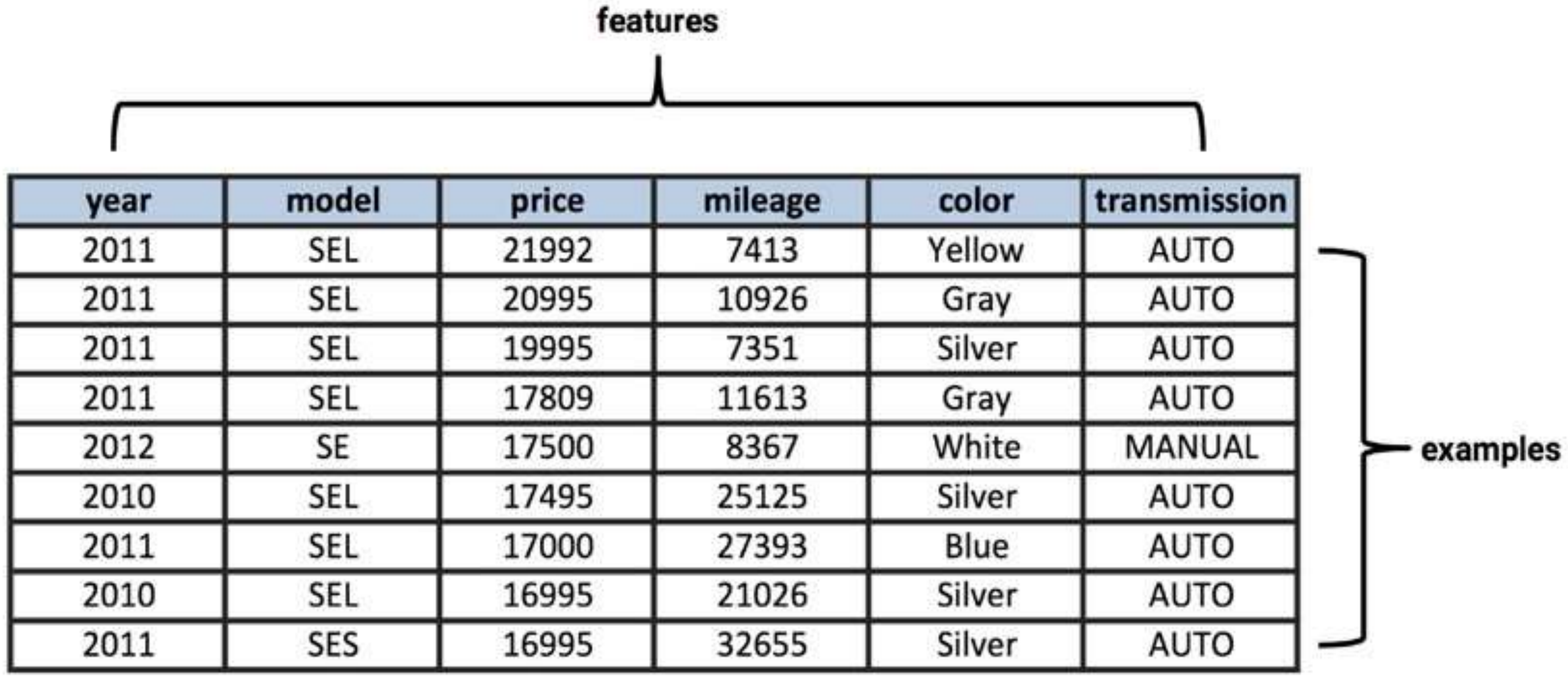
TYPES OF INPUT DATA (CONTINUED)

Examples and features

- Do not have to be collected in any specific form.
- Commonly gathered in **matrix format** which means that each example has exactly the same features.

TYPES OF INPUT DATA (CONTINUED)

- Each row - example
- Each column - feature
- Eg : examples of automobiles


features
examples

year	model	price	mileage	color	transmission
2011	SEL	21992	7413	Yellow	AUTO
2011	SEL	20995	10926	Gray	AUTO
2011	SEL	19995	7351	Silver	AUTO
2011	SEL	17809	11613	Gray	AUTO
2012	SE	17500	8367	White	MANUAL
2010	SEL	17495	25125	Silver	AUTO
2011	SEL	17000	27393	Blue	AUTO
2010	SEL	16995	21026	Silver	AUTO
2011	SES	16995	32655	Silver	AUTO

VARIOUS FORM OF “FEATURES”

- Numeric
- Categorical / nominal
- Ordinal

1. NUMERIC FEATURE

The feature which represents a characteristic measured in numbers

- Number of black pixels
- Noise ratios, length of sounds, relative power
- Frequency of specific terms
- Height, weight, etc..

2. CATEGORICAL/NOMINAL FEATURE

A feature which is an attribute that consists of a set of categories.

Allows you to assign categories. E.g

- ❑ Gender
- ❑ Colour of a ball

3. ORDINAL FEATURE

- A special case of categorical variables is called ordinal variable.
- A nominal variable with categories falling in an ordered list.
- An ordinal variable is a categorical variable for which the possible values are ordered.
- E.g clothing sizes small medium, and large
- E.g a measurement of customer satisfaction on a scale from “not at all happy” to “very happy”.
- E.g : ordinal variable - Educational qualification – SSLC, Plus Two, degree, PG

- It is important to consider “**what the features represent**”, as the **“type”** and **“number of features”** in your dataset which will assist in determining an appropriate machine learning algorithm for your task.

TYPES OF MACHINE LEARNING ALGORITHMS

Machine learning algorithms are divided into categories according to their purpose

- Predictive model
- Descriptive model

Learning algorithms

- Supervised learning
- Unsupervised learning

PREDICTIVE MODEL

- Used for tasks that involve the prediction of one value using other values in the dataset.
- The learning algorithm attempts to discover and model the relationship between the target feature (the feature being predicted) and the other features.
- E.g. model is used to predict whether an email is spam or “ham”
- E.g. supervised learning algorithms are used.

DESCRIPTIVE MODEL

- No single feature is more important than any other
- No target to learn

•Process of training a descriptive model is called unsupervised learning

Descriptive modeling task

- Pattern discovery (market basket analysis)
- Clustering (segmentation analysis)

If user buys A e.g. bread) then machine should automatically give him a suggestion to buy B(e.g. Jam)

DESCRIPTIVE MODEL

- **Pattern discovery -** used to identify useful associations within data.
- **Clustering -** dividing a dataset into homogeneous groups.

SUPERVISED LEARNING

- Process of training a predictive model is known as supervised learning.
- Supervised learning algorithm attempts to optimize a function (model) to find the combination of feature values that result in the target output.
- Supervised learning is where you have input variables (X) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output

$Y = f(X)$

SUPERVISED LEARNING (CONTINUED)

- The often used supervised machine learning task of predicting which category an example belongs to is known as **classification**.
- Classifier

E.g. we could predict whether

- An e mail message is spam
- A person has cancer
- A football team will win or lose
- An applicant will default on a loan

UNSUPERVISED LEARNING

•Have input data (X) and no corresponding output variables.

The goal for unsupervised learning is

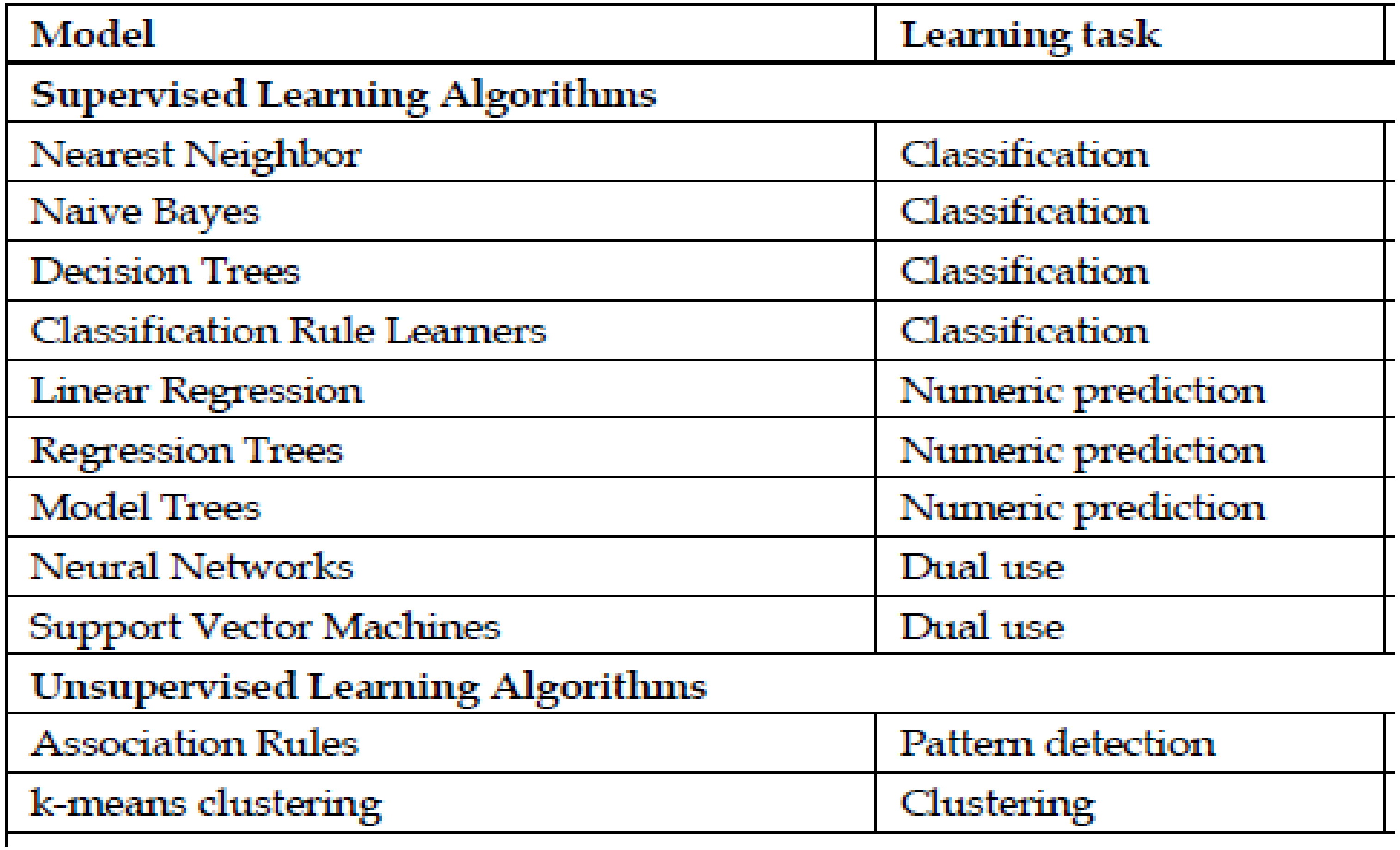
- To "model the underlying structure or distribution in the data" in order to learn more about the data

Clustering

Association rules

- K means for clustering problems
- Apriori algorithm for association rule learning problems

GENERAL TYPES OF MACHINE LEARNING ALGORITHMS

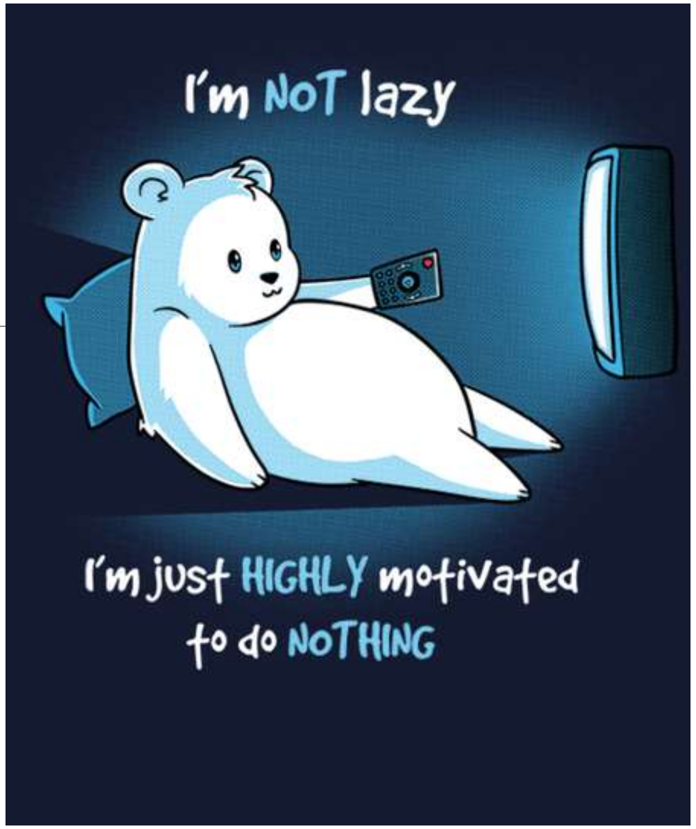


Model	Learning task
Supervised Learning Algorithms	
Nearest Neighbor	Classification
Naive Bayes	Classification
Decision Trees	Classification
Classification Rule Learners	Classification
Linear Regression	Numeric prediction
Regression Trees	Numeric prediction
Model Trees	Numeric prediction
Neural Networks	Dual use
Support Vector Machines	Dual use
Unsupervised Learning Algorithms	
Association Rules	Pattern detection
k-means clustering	Clustering

LAZY LEARNING

Lazy Learning

Lazy learning methods simply store the data and generalizing beyond these data is postponed until an explicit request is made.



Understanding nearest neighbor classification

- In a single sentence, nearest neighbor classifiers are defined by their characteristic of classifying unlabeled examples by assigning them the class of similar labeled examples.
- Despite the simplicity of this idea, nearest neighbor methods are extremely powerful.

They have been used successfully for:

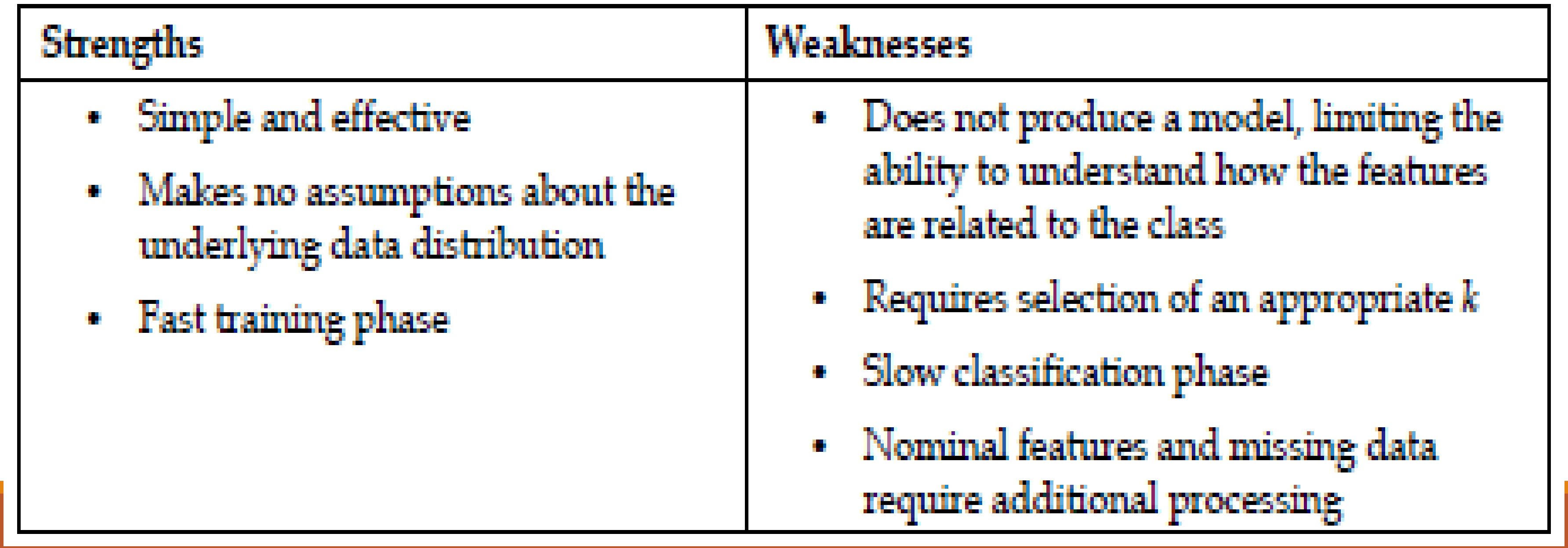
- Computer vision applications, including optical character recognition and facial recognition in both still images and video
- Predicting whether a person will enjoy a movie or music recommendation
- Identifying patterns in genetic data, perhaps to use them in detecting specific proteins or diseases

- In general, nearest neighbor classifiers are well-suited for classification tasks, where relationships among the features and the target classes are numerous, complicated, or extremely difficult to understand, yet the items of similar class type tend to be fairly homogeneous.
- Another way of putting it would be to say that if a concept is difficult to define, but you know it when you see it, then nearest neighbors might be appropriate.
- On the other hand, if the data is noisy and thus no clear distinction exists among the groups, the nearest neighbor algorithms may struggle to identify the class boundaries.

The k-NN algorithm

– The nearest neighbors approach to classification is exemplified by the k-nearest neighbors algorithm (k-NN).

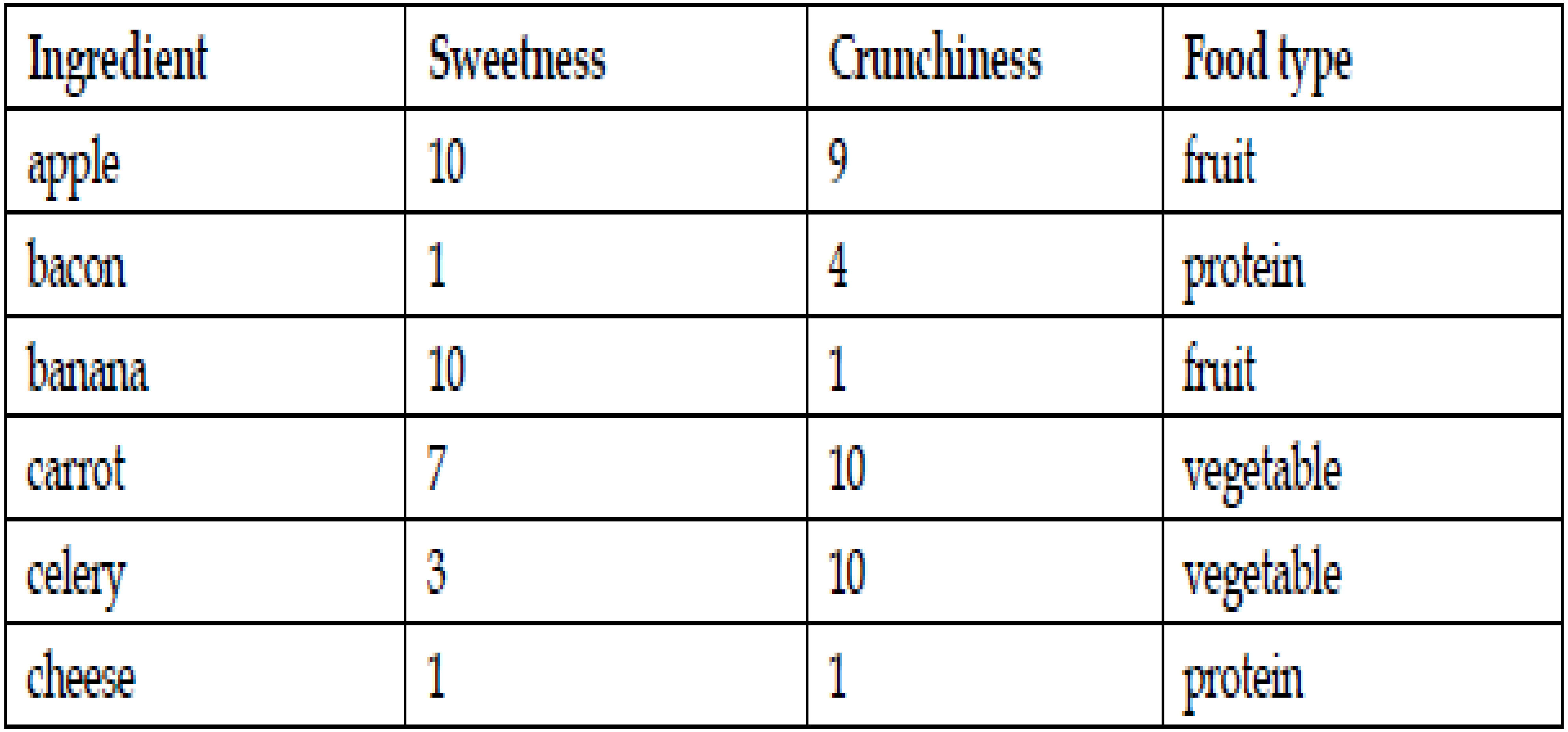
– Although this is perhaps one of the simplest machine learning algorithms, it is still used widely.



Strengths	Weaknesses
• Simple and effective • Makes no assumptions about the underlying data distribution • Fast training phase	• Does not produce a model, limiting the ability to understand how the features are related to the class • Requires selection of an appropriate *k* • Slow classification phase • Nominal features and missing data require additional processing

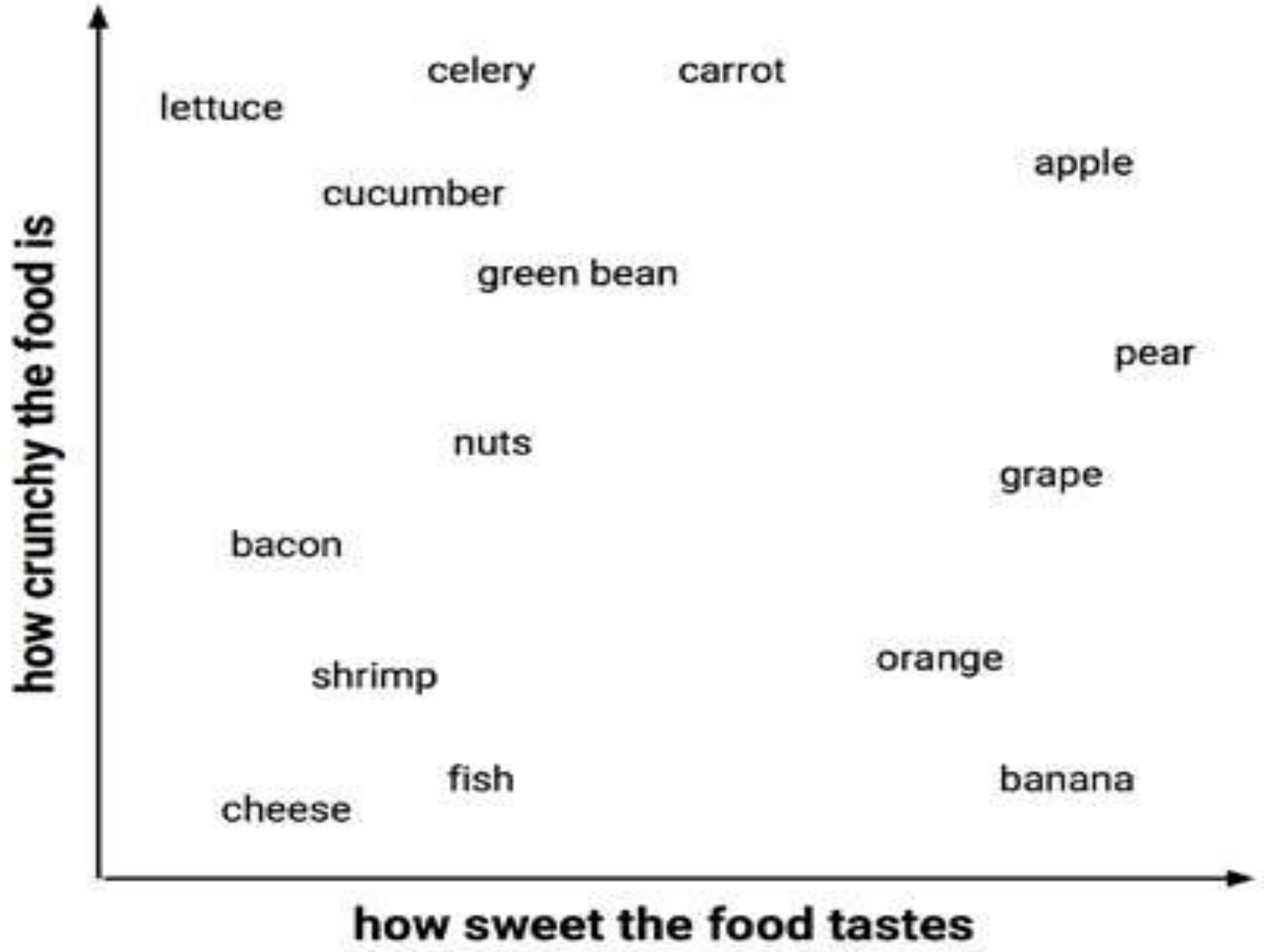
- The k-NN algorithm gets its name from the fact that it uses information about an example's k-nearest neighbors to classify unlabeled examples.
- The letter k is a variable term implying that any number of nearest neighbors could be used.
- After choosing k, the algorithm requires a training dataset made up of examples that have been classified into several categories, as labeled by a nominal variable.
- Then, for each unlabeled record in the test dataset, k-NN identifies k records in the training data that are the "nearest" in similarity.
- The unlabeled test instance is assigned the class of the majority of the k nearest neighbors.

- To illustrate this process, let's revisit the blind tasting experience.
- Suppose that prior to eating the mystery meal we had created a dataset in which we recorded our impressions of a number of ingredients we tasted previously.
- To keep things simple, we rated only two features of each ingredient.
- The first is a measure from 1 to 10 of how crunchy the ingredient is and the second is a 1 to 10 score of how sweet the ingredient tastes.
- We then labeled each ingredient as one of the three types of food: fruits, vegetables, or proteins.
- The first few rows of such a dataset might be structured as follows:

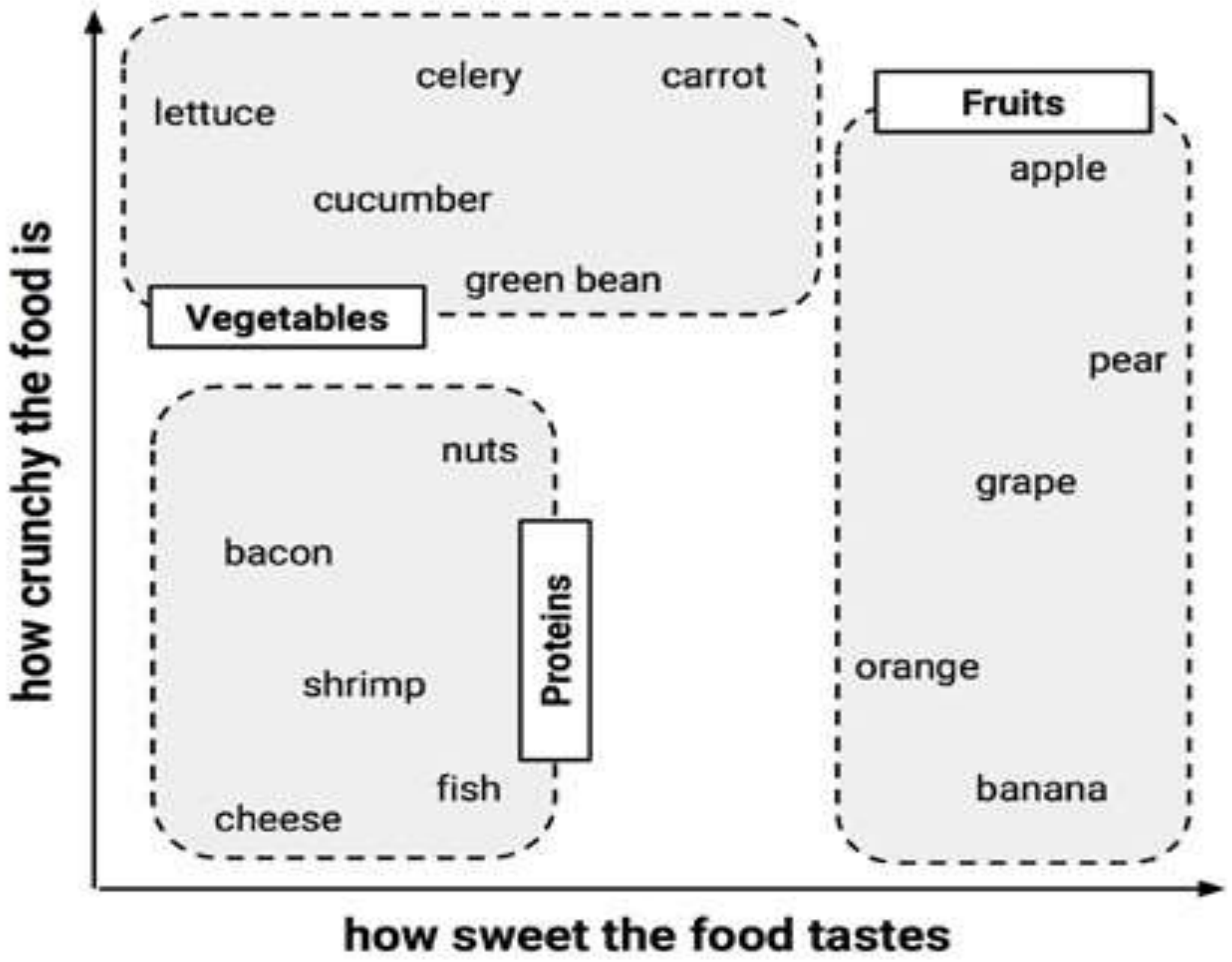


Ingredient	Sweetness	Crunchiness	Food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

- The k-NN algorithm treats the features as coordinates in a multidimensional feature space.
- As our dataset includes only two features, the feature space is two-dimensional.
- We can plot two-dimensional data on a scatter plot, with the x dimension indicating the ingredient's sweetness and the y dimension, the crunchiness.
- After adding a few more ingredients to the taste dataset, the scatter plot might look similar to this:

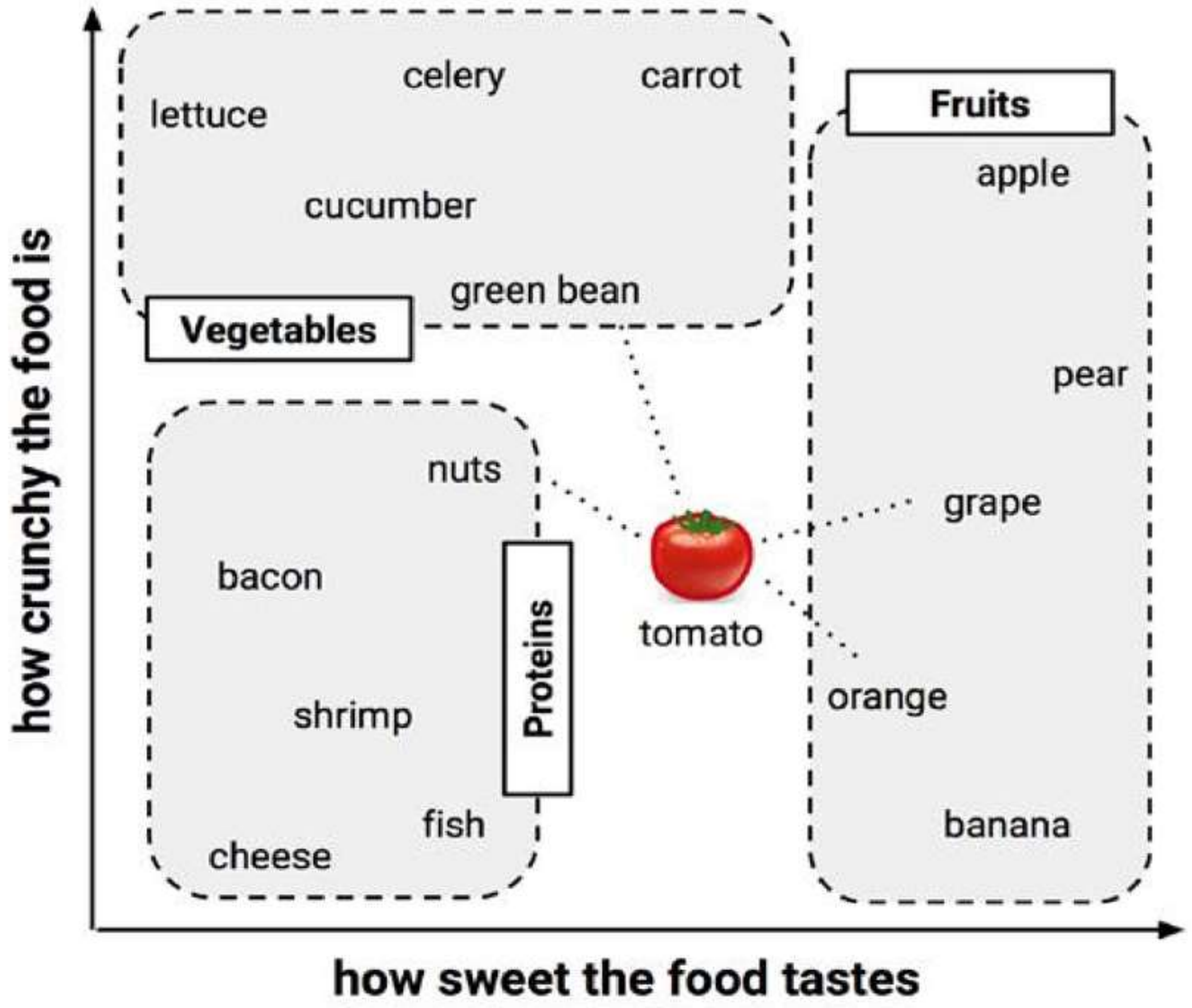

celery
carrot
lettuce
apple
cucumber
green bean
pear
nuts
grape
bacon
orange
shrimp
fish
banana
cheese
how crunchy the food is
how sweet the food tastes

- Did you notice the pattern?
- Similar types of food tend to be grouped closely together.
- As illustrated in the next diagram, vegetables tend to be crunchy but not sweet, fruits tend to be sweet and either crunchy or not crunchy, while proteins tend to be neither crunchy nor sweet:



Vegetables
lettuce
celery
carrot
cucumber
green bean
Fruits
apple
pear
grape
orange
banana
Proteins
nuts
bacon
shrimp
fish
cheese
how crunchy the food is
how sweet the food tastes

- Suppose that after constructing this dataset, we decide to use it to settle the age-old question: is tomato a fruit or vegetable?
- We can use the nearest neighbor approach to determin which class is a better fit, as shown in the following diagram



Measuring similarity with distance

- Locating the tomato's nearest neighbors requires a **distance function**, or a formula that measures the similarity between the two instances.
- There are many different ways to calculate distance.
- Traditionally, the k-NN algorithm uses **Euclidean distance**, which is the distance one would measure if it were possible to use a ruler to connect two points, illustrated in the previous figure by the dotted lines connecting the tomato to its neighbors.

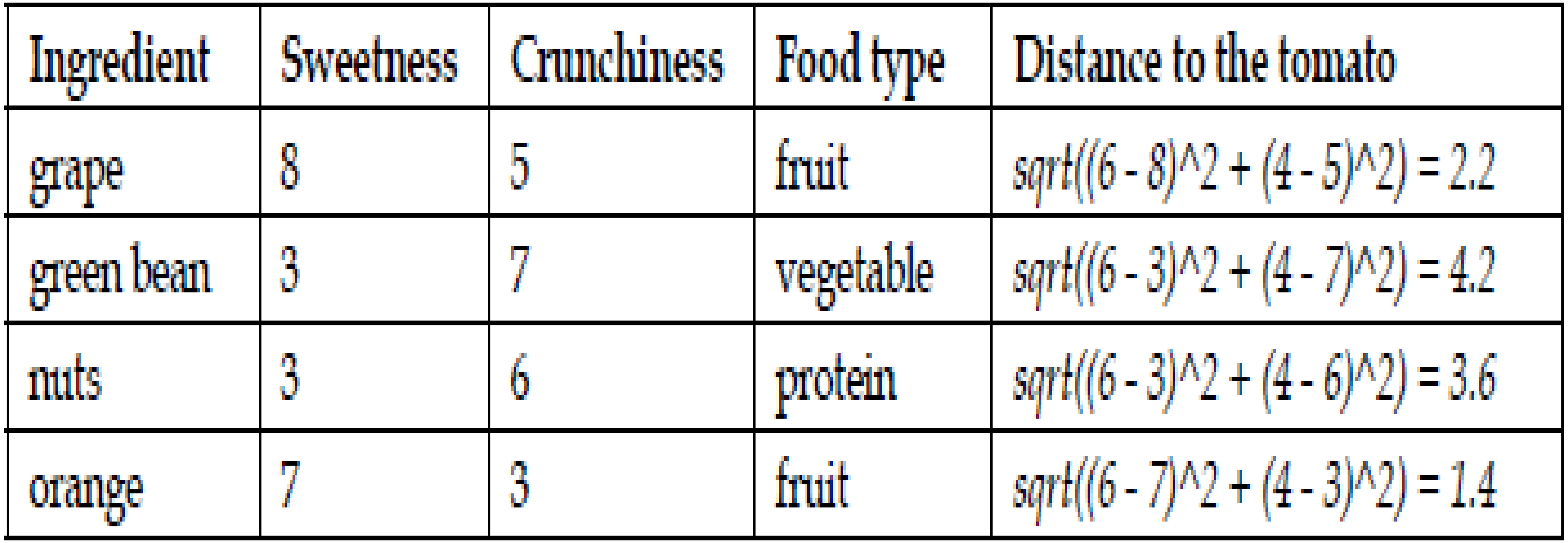
- Euclidean distance is specified by the following formula, where p and q are the examples to be compared, each having n features.
- The term p1 refers to the value of the first feature of example p, while q1 refers to the value of the first feature of example q:

$$\text{dist}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2}$$

- The distance formula involves comparing the values of each feature.
- For example, to calculate the distance between the tomato (sweetness = 6, crunchiness = 4), and the green bean (sweetness = 3, crunchiness = 7), we can use the formula as follows:

$$\mathrm{dist}(\text{tomato, green bean}) = \sqrt{(6-3)^2 + (4-7)^2} = 4.2$$

– In a similar vein, we can calculate the distance between the tomato and several of its closest neighbors as follows:



Ingredient	Sweetness	Crunchiness	Food type	Distance to the tomato
grape	8	5	fruit	*sqrt*((6 - 8)^2 + (4 - 5)^2) = 2.2
green bean	3	7	vegetable	*sqrt*((6 - 3)^2 + (4 - 7)^2) = 4.2
nuts	3	6	protein	*sqrt*((6 - 3)^2 + (4 - 6)^2) = 3.6
orange	7	3	fruit	*sqrt*((6 - 7)^2 + (4 - 3)^2) = 1.4

- To classify the tomato as a vegetable, protein, or fruit, we'll begin by assigning the tomato, the food type of its single nearest neighbor.
- This is called 1-NN classification because k = 1.
- The orange is the nearest neighbor to the tomato, with a distance of 1.4.
- As orange is a fruit, the 1-NN algorithm would classify tomato as a fruit.
- If we use the k-NN algorithm with k = 3 instead, it performs a vote among the three nearest neighbors: orange, grape, and nuts.
- Since the majority class among these neighbors is fruit (two of the three votes), the tomato again is classified as a fruit.

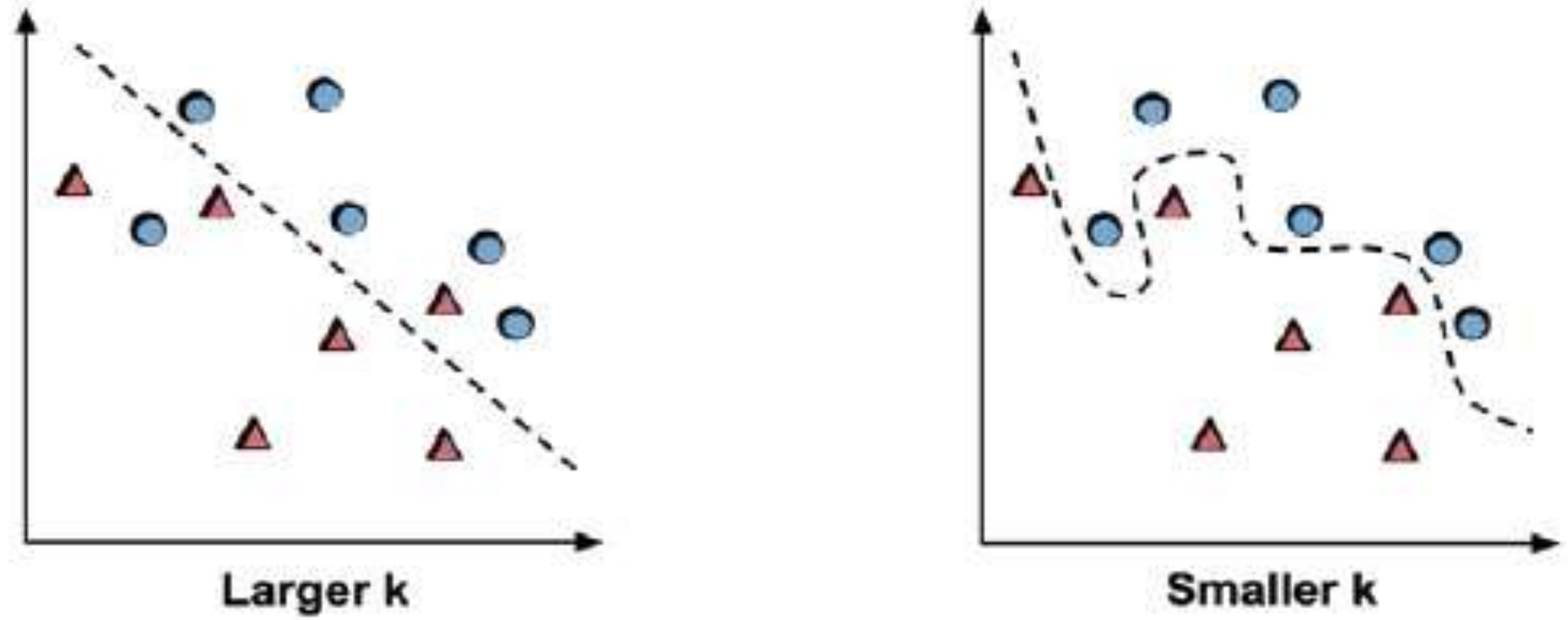
Choosing an appropriate k

- The decision of how many neighbors to use for k-NN determines how well the model will generalize to future data.
- The balance between **overfitting and underfitting** the training data is a problem known as **bias-variance tradeoff**.
- **Overfitting** – occurs when a statistical model or machine learning algorithm captures the noise of the data.
- **Underfitting** – refers to a model that can neither model the training data nor generalize to new data.

- Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner so that it runs the risk of ignoring small, but important patterns.
- Suppose we took the extreme stance of setting a very large k, as large as the total number of observations in the training data.
- With every training instance represented in the final vote, the most common class always has a majority of the voters.
- The model would consequently always predict the majority class, regardless of the nearest neighbors.

- On the opposite extreme, using a single nearest neighbor allows the noisy data or outliers to unduly influence the classification of examples.
- For example, suppose some of the training examples were accidentally mislabeled.
- Any unlabeled example that happens to be nearest to the incorrectly labeled neighbor will be predicted to have the incorrect class.
- Obviously, the best k value is somewhere between these two extremes.

- The following figure illustrates, more generally, how the decision boundary (depicted by a dashed line) is affected by larger or smaller k values.
- Smaller values allow more complex decision boundaries that more carefully fit the training data.



- In practice, choosing k depends on the difficulty of the concept to be learned, and the number of records in the training data.
- One common practice is to begin with k equal to the square root of the number of training examples.
- In the food classifier we developed previously, we might set k = 4 because there were 15 example ingredients in the training data and the square root of 15 is 3.87.

- However, such rules may not always result in the single best k.
- An alternative approach is to test several k values on a variety of test datasets and choose the one that delivers the best classification performance.

Preparing data for use with k-NN

There are several methods for scaling.

– min-max normalization.

– z-score standardization.

– dummy coding.

1. Min-max Normalization

- The traditional method of rescaling features for k-NN is min-max normalization.
- This process transforms a feature such that all of its values fall in a range between 0 and 1.
- The formula for normalizing a feature is as follows:

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

2. Z-score Standardization

- The resulting value is called a z-score.
- The z-scores fall in an unbound range of negative and positive numbers.
- Unlike the normalized values, they have no predefined minimum and maximum.
- The following formula subtracts the mean value of feature X, and divides the outcome by the standard deviation of X:

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$

3. Dummy Coding

- The Euclidean distance formula is not defined for nominal data.
- Therefore, to calculate the distance between nominal features, we need to convert them into a numeric format.
- A typical solution utilizes dummy coding, where a value of 1 indicates one category, and 0, the other.
- For instance, dummy coding for a gender variable could be constructed as:

$$\text{male} = \begin{cases} 1 & \text{if x = male} \\ 0 & \text{otherwise} \end{cases}$$

3. Dummy Coding (Continued)

- Notice how the dummy coding of the two-category (binary) gender variable results in a single new feature named male.
- There is no need to construct a separate feature for female; since the two sexes are mutually exclusive, knowing one or the other is enough.

3. Dummy Coding (Continued)

– An n-category nominal feature can be dummy coded by creating the binary indicator variables for (n - 1) levels of the feature.

– For example, the dummy coding for a three-category temperature variable (for example, hot, medium, or cold) could be set up as (3 - 1) = 2 features, as shown here:

$$\text{hot} = \begin{cases} 1 & \text{if x = hot} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{medium} = \begin{cases} 1 & \text{if x = medium} \\ 0 & \text{otherwise} \end{cases}$$

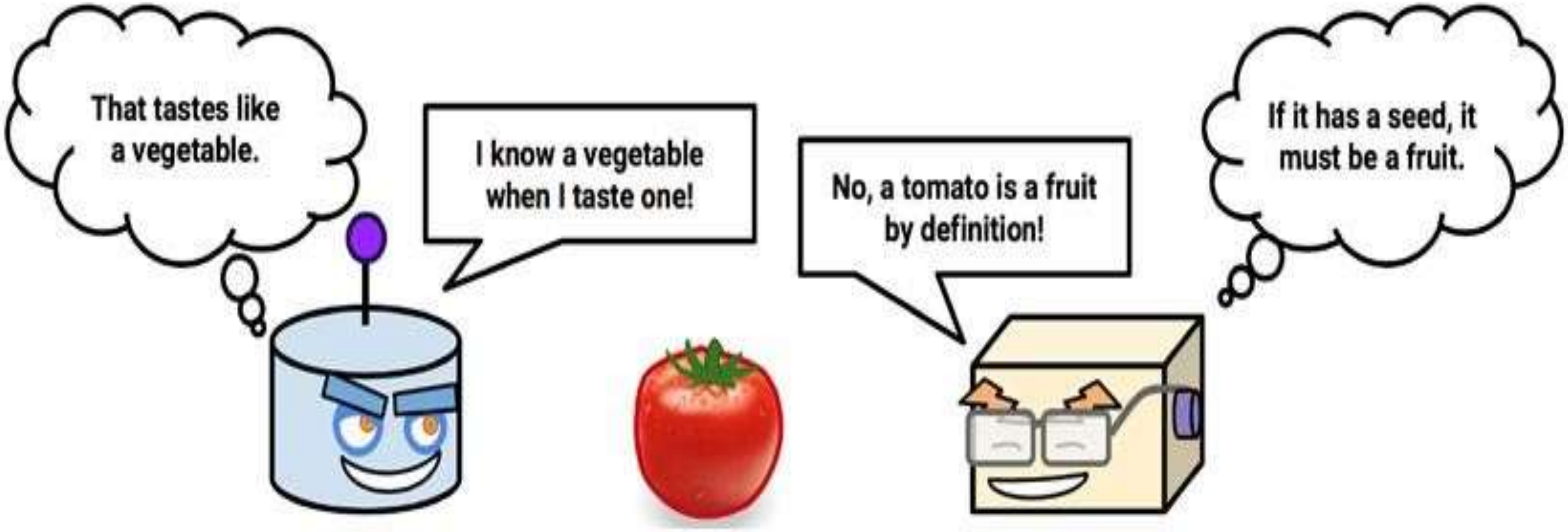
3. Dummy Coding (Continued)

- Knowing that hot and medium are both 0 is enough to know that the temperature is cold.
- We, therefore, do not need a third feature for the cold category.
- A convenient aspect of dummy coding is that the distance between dummy coded features is always one or zero, and thus, the values fall on the same scale as min-max normalized numeric data.
- No additional transformation is necessary.

Why is the k-NN algorithm lazy?

- Classification algorithms based on the nearest neighbor methods are considered lazy learning algorithms because, technically speaking, no abstraction occurs.
- The abstraction and generalization processes are skipped altogether, and this undermines the definition of learning.
- The downside is that the process of making predictions tends to be relatively slow in comparison to training.
- Due to the heavy reliance on the training instances rather than an abstracted model, lazy learning is also known as **instance-based learning or rote learning**.

- As instance-based learners do not build a model, the method is said to be in a class of non-parametric learning methods—no parameters are learned about the data.
- Without generating theories about the underlying data, non- parametric methods limit our ability to understand how the classifier is using the data.
- On the other hand, this allows the learner to find natural patterns rather than trying to fit the data into a preconceived and potentially biased functional form.


That tastes like a vegetable.
I know a vegetable when I taste one!
No, a tomato is a fruit by definition!
If it has a seed, it must be a fruit.

Probabilistic Learning

Probabilistic Learning – Classification Using Naive Bayes

- When a meteorologist provides a weather forecast, precipitation is typically described with terms such as "70 percent chance of rain.“
- They use data on past events to extrapolate future events.
- A 70 percent chance of rain implies that in 7 out of the 10 past cases with similar conditions, precipitation occurred somewhere in the area.

Probabilistic Classifier

- A classifier that is able to predict, given a sample input, a probability distribution over a set of classes, (rather than only outputting the most likely class that the sample should belong to).

Understanding Naive Bayes

- The technique descended from the work of the 18th century mathematician **Thomas Bayes**, who developed foundational principles to describe the probability of events, and how probabilities should be revised in the light of additional information.
- These principles formed the foundation for what are now known as Bayesian methods.

Understanding Naive Bayes (Continued)

- A probability is a number between 0 and 1 (that is, between 0 percent and 100 percent), which captures the chance that an event will occur in the light of the available evidence.
- The lower the probability, the less likely the event is to occur.
- A probability of 0 indicates that the event will definitely not occur, while a probability of 1 indicates that the event will occur with 100 percent certainty.

Understanding Naive Bayes (Continued)

- Classifiers based on Bayesian methods utilize training data to calculate an observed probability of each outcome based on the evidence provided by feature values.
- When the classifier is later applied to unlabeled data, it uses the observed probabilities to predict the most likely class for the new features.
- It's a simple idea, but it results in a method that often has results on par with more sophisticated algorithms.

Understanding Naive Bayes (Continued)

In fact, Bayesian classifiers have been used for:

– Text classification, such as junk e-mail (spam) filtering

– Intrusion or anomaly detection in computer networks

– Diagnosing medical conditions given a set of observed symptoms

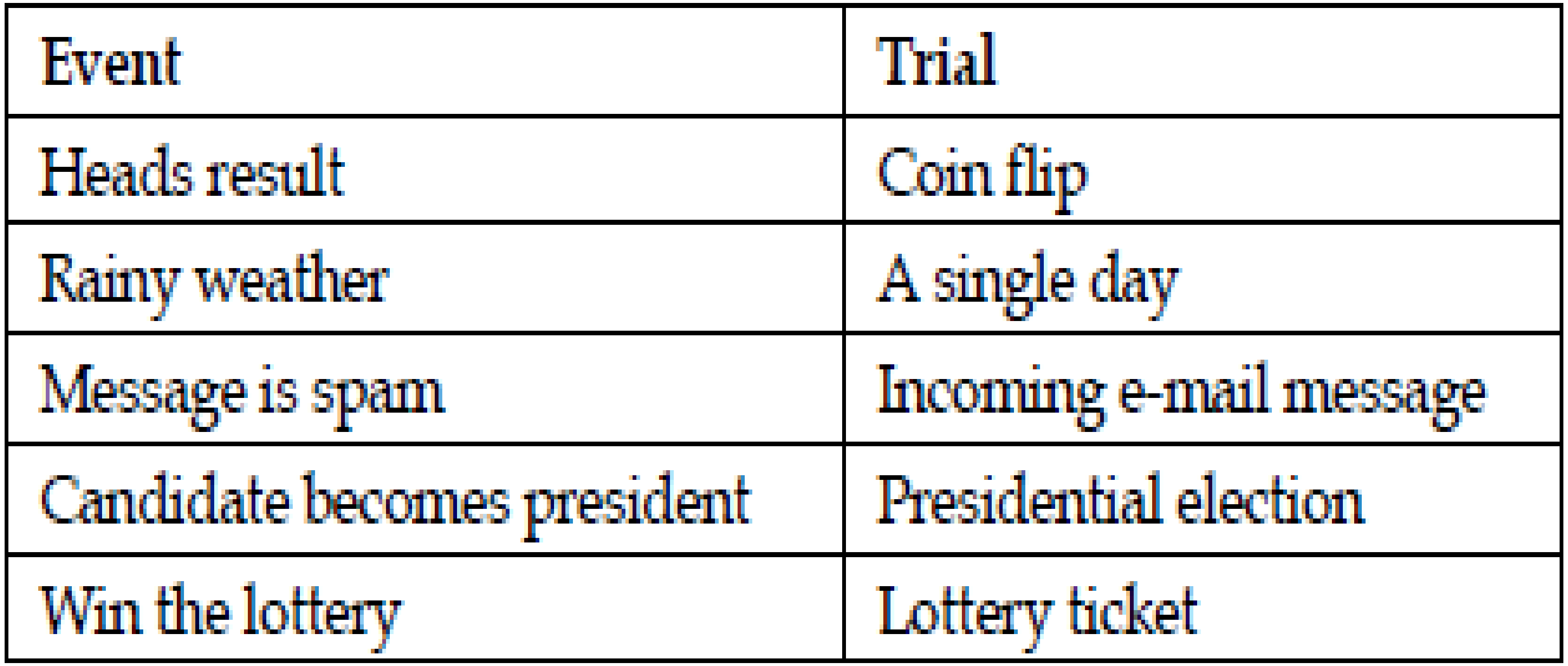
Understanding Naive Bayes (Continued)

- Bayesian classifiers are best applied to problems in which the information from numerous attributes should be considered simultaneously in order to estimate the overall probability of an outcome.
- While many machine learning algorithms ignore features that have weak effects, Bayesian methods utilize all the available evidence to subtly change the predictions.
- If large number of features have relatively minor effects, taken together, their combined impact could be quite large.

Basic concepts of Bayesian methods

- Bayesian probability theory is rooted in the idea that the estimated likelihood of **an event**, or a potential outcome, should be based on the evidence at hand across **multiple trials**, or opportunities for the event to occur.
- Bayesian methods provide insights into how the probability of these events can be estimated from the observed data.
- The following table illustrates events and trials for several real- world outcomes:

Basic concepts of Bayesian methods (Continued)



Event	Trial
Heads result	Coin flip
Rainy weather	A single day
Message is spam	Incoming e-mail message
Candidate becomes president	Presidential election
Win the lottery	Lottery ticket

Understanding probability

- The probability of an event is estimated from the observed data by dividing the number of trials in which the event occurred by the total number of trials.
- For instance, if it rained 3 out of 10 days with similar conditions as today, the probability of rain today can be estimated as 3 / 10 = 0.30 or 30 percent.
- Similarly, if 10 out of 50 prior email messages were spam, then the probability of any incoming message being spam can be estimated as 10 / 50 = 0.20 or 20 percent.
- To denote these probabilities, we use notation in the form P(A), which signifies the probability of event A. For example, P(rain) = 0.30 and P(spam) = 0.20.

Understanding probability (Continued)

- The probability of all the possible outcomes of a trial must always sum to 1, because a trial always results in some outcome happening.
- Thus, if the trial has two outcomes that cannot occur simultaneously, such as rainy versus sunny or spam versus ham (nonspam), then knowing the probability of either outcome reveals the probability of the other.
- For example, given the value P(spam) = 0.20, we can calculate P(ham) = 1 – 0.20 = 0.80.
- This concludes that spam and ham are **mutually exclusive and exhaustive events**, which implies that they cannot occur at the same time and are the only possible outcomes.

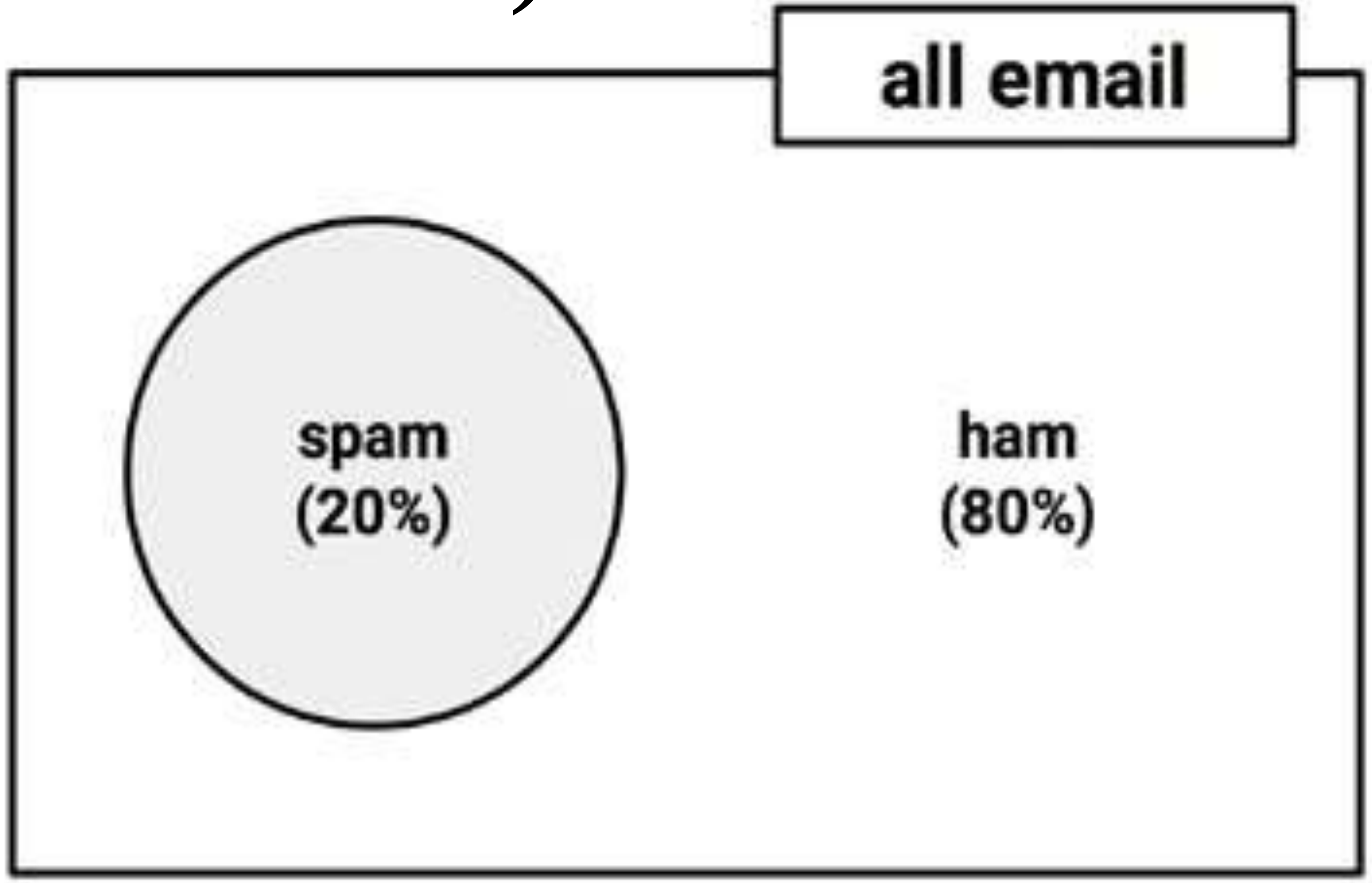
Understanding probability (Continued)

- Because an event cannot simultaneously happen and not happen, an event is always mutually exclusive and exhaustive with its **complement**, or the event comprising of the outcomes in which the event of interest does not happen.
- The complement of event A is typically denoted A^c or A'.
- Additionally, the shorthand notation P(¬A) can used to denote the probability of event A not occurring, as in P(¬spam) = 0.80. This notation is equivalent to $P(A^c)$.

Understanding probability (Continued)

- To illustrate events and their complements, it is often helpful to imagine a two-dimensional space that is partitioned into probabilities for each event.
- In the following diagram, the rectangle represents the possible outcomes for an e-mail message.
- The circle represents the 20 percent probability that the message is spam.
- The remaining 80 percent represents the complement P(¬spam) or the messages that are not spam:

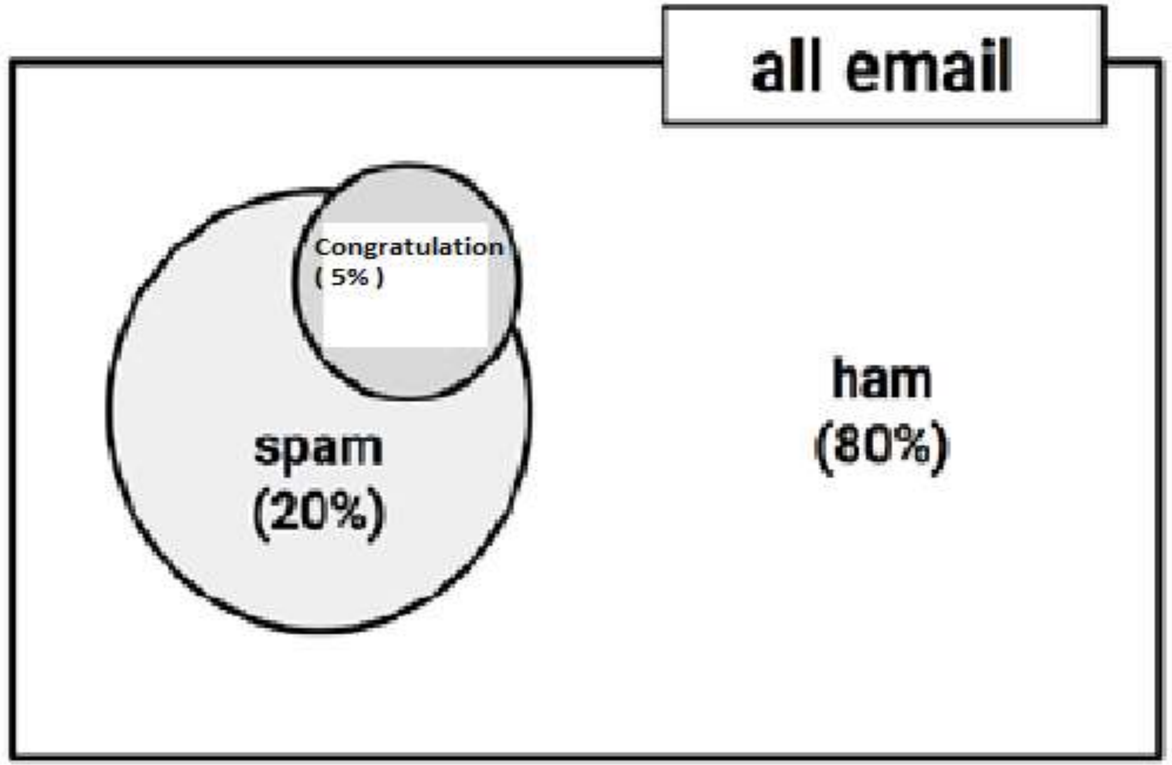
Understanding probability (Continued)



Understanding joint probability

- Often, we are interested in monitoring several nonmutually exclusive events for the same trial.
- If certain events occur with the event of interest, we may be able to use them to make predictions.
- Consider, for instance, a second event based on the outcome that an e-mail message contains the word “Congratulations”.
- In most cases, this word is likely to appear only in a spam message; its presence in an incoming e-mail is therefore a very strong piece of evidence that the message is spam.
- The preceding diagram, updated for this second event, might appear as shown in the following diagram:

Understanding joint probability (Continued)



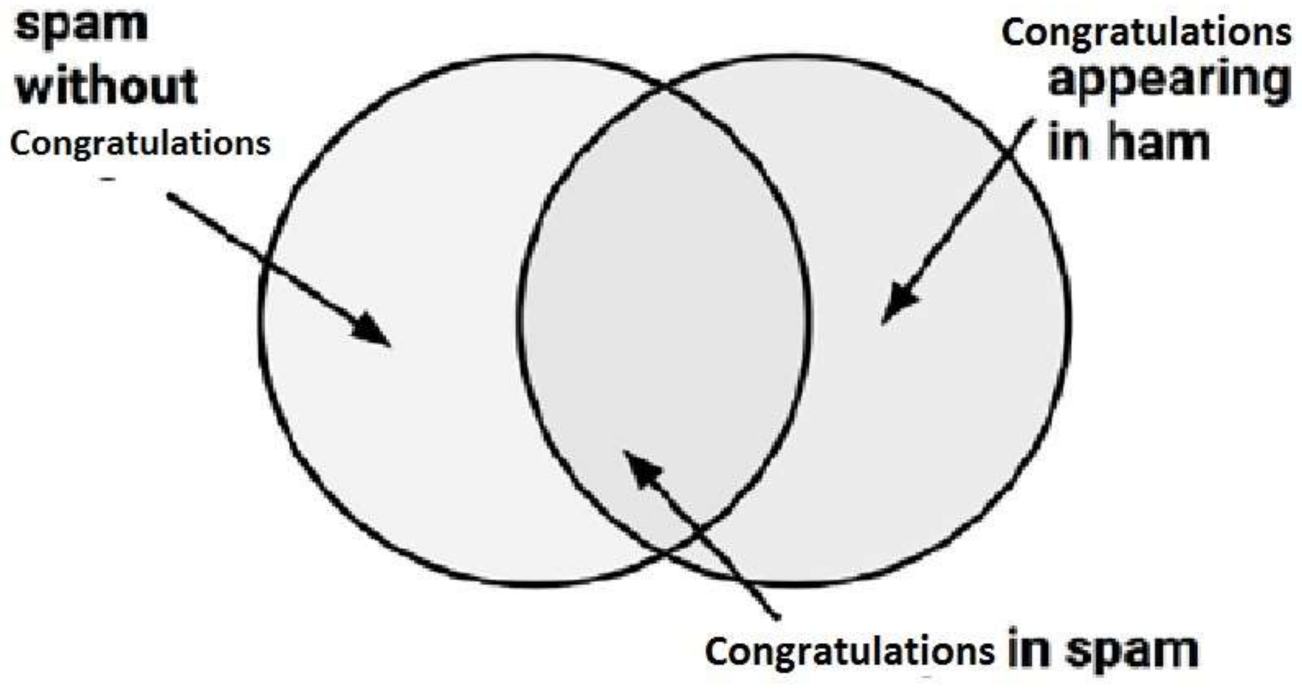
Understanding joint probability (Continued)

- Notice in the diagram that the Congratulations circle does not completely fill the spam circle, nor is it completely contained by the spam circle.
- This implies that not all spam messages contain the word Congratulations and not every e-mail with the word Congratulations is spam.

Understanding joint probability (Continued)

- To zoom in for a closer look at the overlap between the spam and Congratulations circles, we'll employ a visualization known as a **Venn diagram**.
- First used in the late 19th century by John Venn, the diagram uses circles to illustrate the overlap between sets of items.
- In most Venn diagrams, the size of the circles and the degree of the overlap is not meaningful.
- Instead, it is used as a reminder to allocate probability to all possible combinations of events:

Understanding joint probability (Continued)


spam without Congratulations
Congratulations appearing in ham
Congratulations in spam

Understanding joint probability (Continued)

- We know that 20 percent of all messages were spam (the left circle) and 5 percent of all messages contained the word Congratulations (the right circle).
- We would like to quantify the degree of overlap between these two proportions.
- In other words, we hope to estimate the probability that both P(spam) and P(Congratulations) occur, which can be written as P(spam ∩ Congratulations).
- The upside down 'U' symbol signifies the intersection of the two events; the notation A ∩ B refers to the event in which both A and B occur.

Understanding joint probability (Continued)

- Calculating P(spam ∩ Congratulations) depends on the **joint probability** of the two events or how the probability of one event is related to the probability of the other.
- If the two events are totally unrelated, they are called **independent events**.
- This is not to say that independent events cannot occur at the same time; event independence simply implies that knowing the outcome of one event does not provide any information about the outcome of the other.
- For instance, the outcome of a heads result on a coin flip is independent from whether the weather is rainy or sunny on any given day.

Understanding joint probability (Continued)

- If all events were independent, it would be impossible to predict one event by observing another.
- In other words, **<u>dependent events are the basis of predictive modeling.</u>**
- Just as the presence of clouds is predictive of a rainy day, the appearance of the word Congratulations is predictive of a spam e- mail.

Understanding joint probability (Continued)

- Calculating the probability of dependent events is a bit more complex than for independent events.
- If P(spam) and P(Congratulations) were independent, we could easily calculate P(spam ∩ Congratulations), the probability of both events happening at the same time.
- Because 20 percent of all the messages are spam, and 5 percent of all the e- mails contain the word Congratulations, we could assume that 1 percent of all messages are spam with the term Congratulations.
- This is because 0.05 * 0.20 = 0.01. More generally, for independent events A and B, the probability of both happening can be expressed as P(A ∩ B) = P(A) * P(B).

Understanding joint probability (Continued)

- This said, we know that P(spam) and P(Congratulations) are likely to be highly dependent, which means that this calculation is incorrect.
- To obtain a reasonable estimate, we need to use a more careful formulation of the relationship between these two events, which is based on advanced Bayesian methods.

Computing conditional probability with Bayes' theorem

- The relationships between dependent events can be described using Bayes' theorem, as shown in the following formula.
- This formulation provides a way of thinking about how to revise an estimate of the probability of one event in light of the evidence provided by another event:

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

Computing conditional probability with Bayes’ theorem (Continued)

- The notation P(A|B) is read as the probability of event A, given that event B occurred.
- This is known as **conditional probability**, since the probability of A is dependent (that is, conditional) on what happened with event B.
- Bayes' theorem tells us that our estimate of P(A|B) should be based on P(A ∩ B), a measure of how often A and B are observed to occur together, and P(B), a measure of how often B is observed to occur in general.

Computing conditional probability with Bayes' theorem (Continued)

- Bayes' theorem states that the best estimate of P(A|B) is the proportion of trials in which A occurred with B out of all the trials in which B occurred.
- In plain language, this tells us that if we know event B occurred, the probability of event A is higher the more often that A and B occur together each time B is observed.
- In a way, this adjusts P(A ∩ B) for the probability of B occurring; if B is extremely rare, P(B) and P(A ∩ B) will always be small; however, if A and B almost always happen together, P(A|B) will be high regardless of the probability of B.

Computing conditional probability with Bayes' theorem (Continued)

- By definition, P(A ∩ B) = P(A|B) * P(B), a fact that can be easily derived by applying a bit of algebra to the previous formula.
- Rearranging this formula once more with the knowledge that P(A ∩ B) = P(B ∩ A) results in the conclusion that P(A ∩ B) = P(B|A) * P(A), which we can then use in the following formulation of Bayes' theorem:

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

Computing conditional probability with Bayes' theorem (Continued)

- Without knowledge of an incoming message's content, the best estimate of its spam status would be P(spam), the probability that any prior message was spam, which we calculated previously to be 20 percent.
- This estimate is known as the **prior probability**.
- The prior probability of an event is the probability of the event computed before the collection of new data.

Computing conditional probability with Bayes' theorem (Continued)

- Suppose that you obtained additional evidence by looking more carefully at the set of previously received messages to examine the frequency that the term Congratulations appeared.
- The probability that the word Congratulations was used in previous spam messages, or P(Congratulations | spam), is called the **likelihood**.
- The probability that Congratulations appeared in any message at all, or P(Congratulations), is known as the **marginal likelihood**.

Computing conditional probability with Bayes’ theorem (Continued)

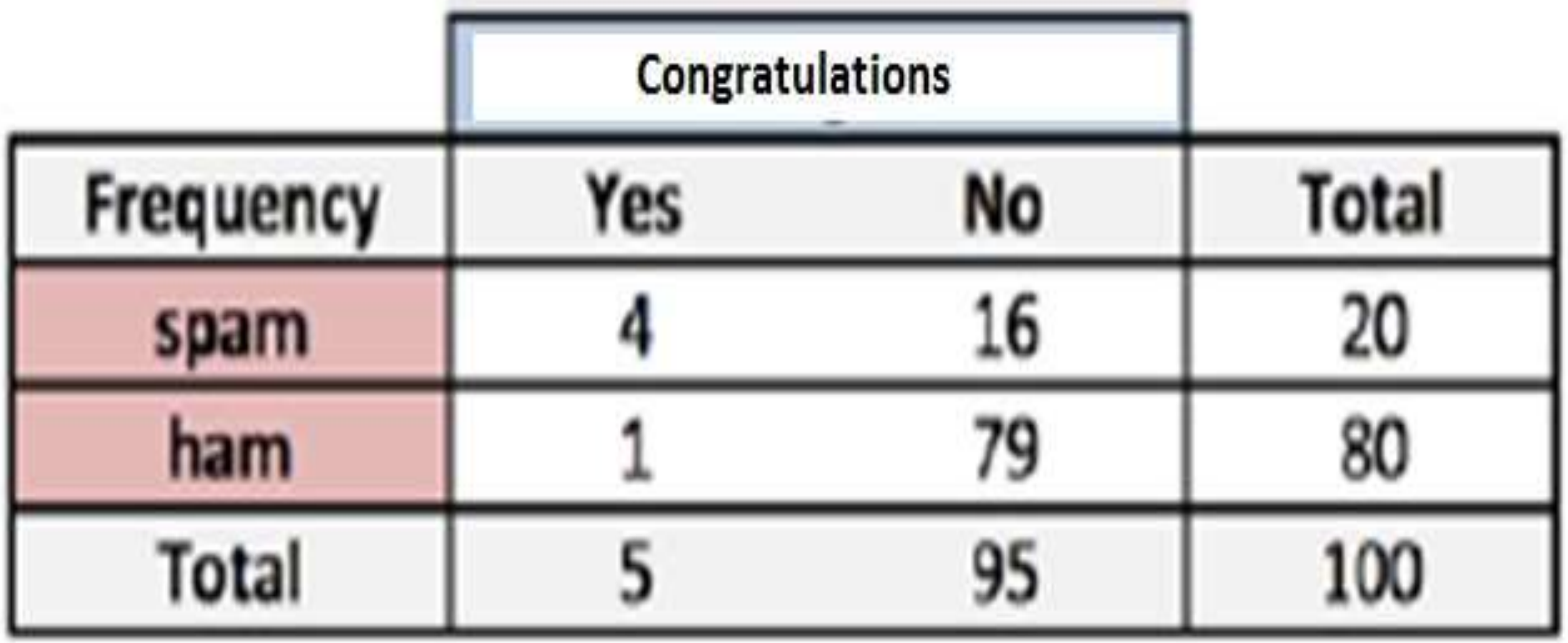
- By applying Bayes' theorem to this evidence, we can compute a posterior probability that measures how likely the message is to be spam.
- If the posterior probability is greater than 50 percent, the message is more likely to be spam than ham and it should perhaps be filtered.
- The following formula shows how Bayes’ theorem is applied to the evidence provided by the previous e-mail messages:

Computing conditional probability with Bayes’ theorem (Continued)

$$\underbrace{P(\text{spam}|\text{Congrats})}_{\text{posterior probability}} = \frac{\overbrace{P(\text{Congrats}|\text{spam})}^{\text{likelihood}}\,\overbrace{P(\text{spam})}^{\text{prior probability}}}{\underbrace{P(\text{Congrats})}_{\text{marginal likelihood}}}$$

Computing conditional probability with Bayes' theorem (Continued)

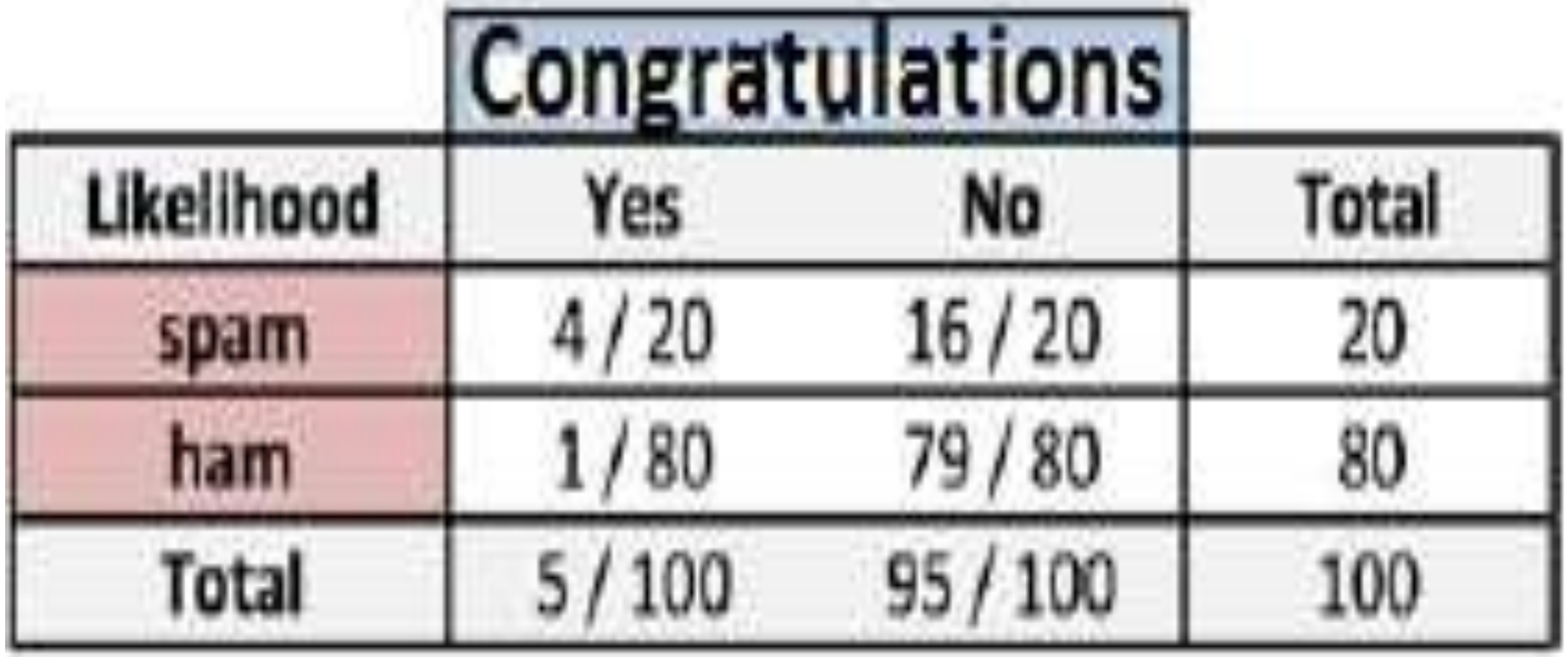
– Frequency Table - Number of times "congratulations" appeared in spam and ham messages.



	Congratulations		
Frequency	Yes	No	Total
spam	4	16	20
ham	1	79	80
Total	5	95	100

Computing conditional probability with Bayes' theorem (Continued)

– Likelihood Table



	Congratulations		
Likelihood	Yes	No	Total
spam	4 / 20	16 / 20	20
ham	1 / 80	79 / 80	80
Total	5 / 100	95 / 100	100

Computing conditional probability with Bayes' theorem (Continued)

– The likelihood table reveals that P(Congratulations=Yes | spam) = 4/20 = 0.20, indicating that the probability is 20 percent that a message contains the term Congratulations, given that the message is spam.

– Additionally, since P(A ∩ B) = P(B|A) * P(A), we can calculate P(spam ∩ Congratulations) as P(Congratulations | spam) * P(spam) = (4/20) * (20/100) = 0.04.

– The same result can be found in the frequency table, which notes that 4 out of the 100 messages were spam with the term Congratulations.

– Either way, this is four times greater than the previous estimate of 0.01 we calculated as P(A ∩ B) = P(A) * P(B) under the false assumption of independence.

– This, of course, illustrates the importance of Bayes' theorem while calculating joint probability.

Computing conditional probability with Bayes' theorem (Continued)

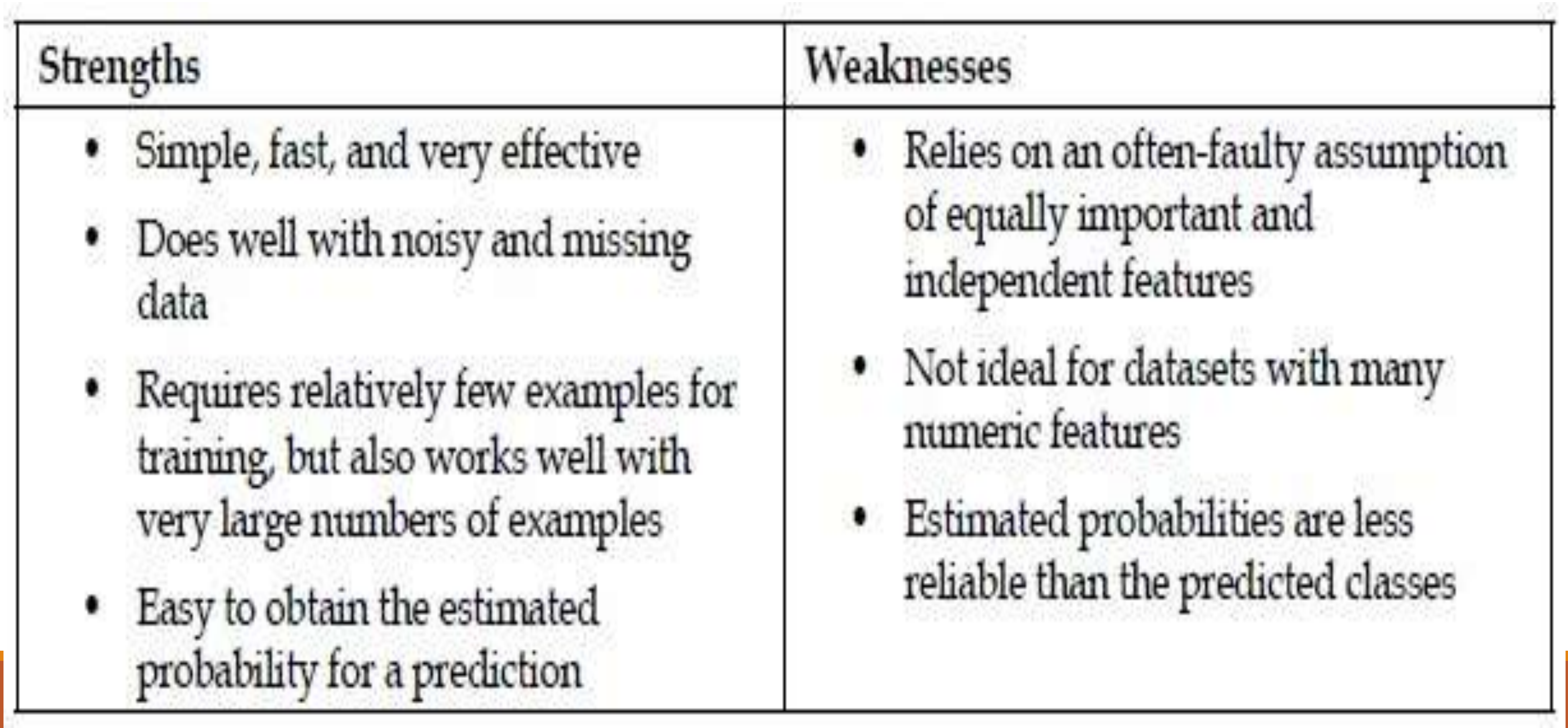
- To compute the posterior probability, P(spam |Congratulations), we simply take P(Congratulations| spam) * P(spam) / P(Congratulations) or (4/20) * (20/100) / (5/100) = 0.80.
- Therefore, the probability is 80 percent that a message is spam, given that it contains the word Congratulations.
- In light of this result, any message containing this term should probably be filtered.

The Naive Bayes algorithm

- The Naive Bayes algorithm describes a simple method to apply Bayes' theorem to classification problems.
- Although it is not the only machine learning method that utilizes

  Bayesian methods, it is the most common one.
- This is particularly true for text classification, where it has become the de facto standard.
- The strengths and weaknesses of this algorithm are as follows:

The Naive Bayes algorithm (Continued)



Strengths	Weaknesses
• Simple, fast, and very effective • Does well with noisy and missing data • Requires relatively few examples for training, but also works well with very large numbers of examples • Easy to obtain the estimated probability for a prediction	• Relies on an often-faulty assumption of equally important and independent features • Not ideal for datasets with many numeric features • Estimated probabilities are less reliable than the predicted classes

The Naive Bayes algorithm (Continued)

- The Naive Bayes algorithm is named as such because it makes

  some "naive“ assumptions about the data.
- In particular, Naive Bayes assumes that all of the features in the

  dataset are equally important and independent.
- These assumptions are rarely true in most real-world applications.

The Naive Bayes algorithm (Continued)

- For example, if you were attempting to identify spam by monitoring e- mail messages, it is almost certainly true that some features will be more important than others.
- For example, the e-mail sender may be a more important indicator of spam than the message text.
- Additionally, the words in the message body are not independent from one another, since the appearance of some words is a very good indication that other words are also likely to appear.
- A message with the word Congratulations will probably also contain the words “earn” or “subscribe”.

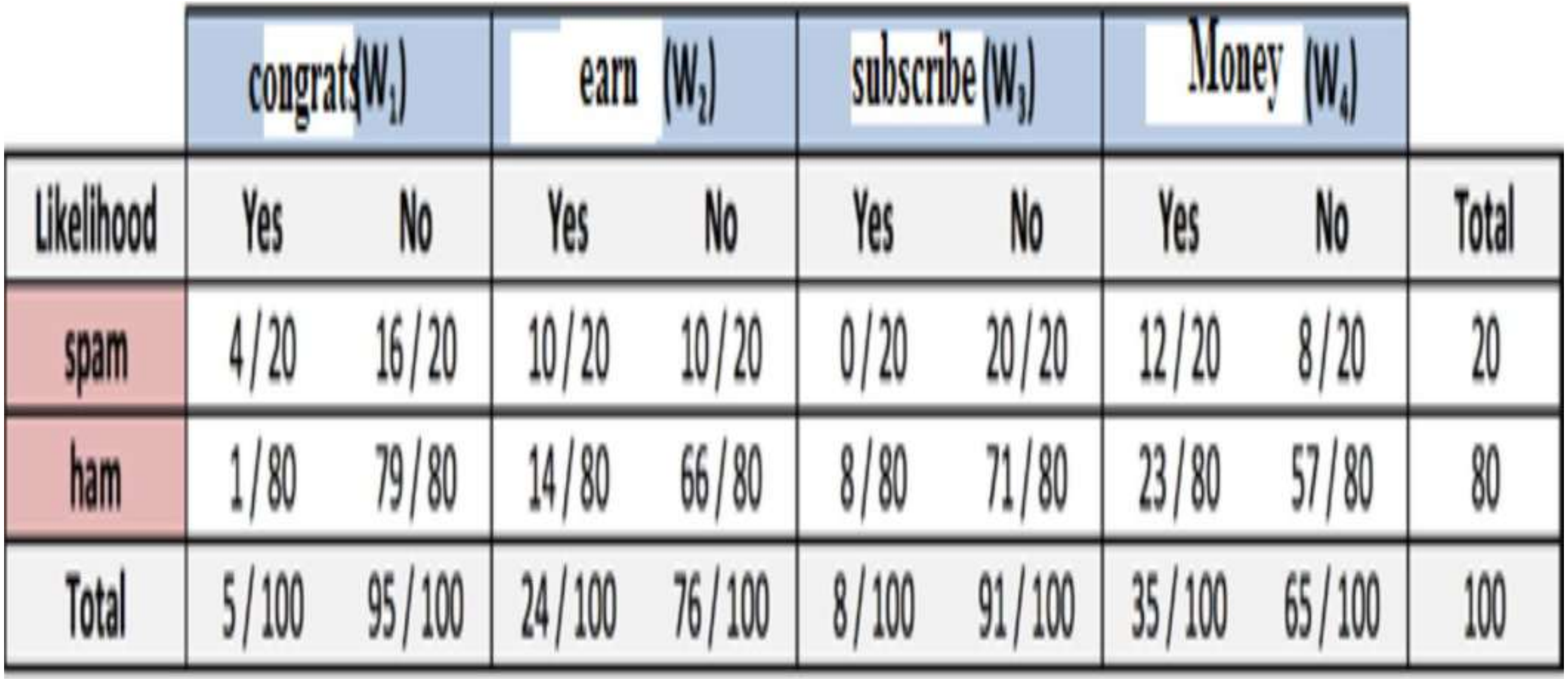
The Naive Bayes algorithm (Continued)

- However, in most cases when these assumptions are violated, Naive Bayes still performs fairly well.
- This is true even in extreme circumstances where strong dependencies are found among the features.
- Due to the algorithm's versatility and accuracy across many types of conditions, Naive Bayes is often a strong first candidate for classification learning tasks.

Classification with Naive Bayes

- Let's extend our spam filter by adding a few additional terms to be monitored in addition to the term Congratulations: Earn, Subscribe, and Money.
- The Naive Bayes learner is trained by constructing a likelihood table for the appearance of these four words (labeled W1, W2, W3, and W4), as shown in the following diagram for 100 e-mails:

Classification with Naive Bayes (Continued)



	congrats(W_1)		earn (W_2)		subscribe(W_3)		Money (W_4)		
Likelihood	Yes	No	Yes	No	Yes	No	Yes	No	Total
spam	4/20	16/20	10/20	10/20	0/20	20/20	12/20	8/20	20
ham	1/80	79/80	14/80	66/80	8/80	71/80	23/80	57/80	80
Total	5/100	95/100	24/100	76/100	8/100	91/100	35/100	65/100	100

Classification with Naive Bayes (Continued)

- Suppose that a message contains the terms congratulations and Money, but does not contain either earn or subscribe.
- Using Bayes' theorem, we can define the problem as shown in the following formula.
- It captures the probability that a message is spam, given that Congratulations = Yes, Earn = No, Subscribe = No, and Money = Yes

$$P(\text{spam} \mid W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4 \mid \text{spam}) P(\text{spam})}{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)}$$

Classification with Naive Bayes (Continued)

- For a number of reasons, this formula is computationally difficult to solve.
- As additional features are added, tremendous amounts of memory are needed to store probabilities for all of the possible intersecting events; imagine the complexity of a Venn diagram for the events for four words, let alone for hundreds or more.

Classification with Naive Bayes (Continued)

- The work becomes much easier if we can exploit the fact that Naive Bayes assumes independence among events.
- Specifically, it assumes class-conditional independence, which means that events are independent so long as they are conditioned on the same class value.
- Assuming conditional independence allows us to simplify the formula using the probability rule for independent events, which states that $P(A \cap B) = P(A) * P(B)$.

Classification with Naive Bayes (Continued)

- Because the denominator does not depend on the class (spam or ham), it is treated as a constant value and can be ignored for the time being.
- This means that the conditional probability of spam can be expressed as:

$$P(\text{spam}|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \propto P(W_1|\text{spam})P(\neg W_2|\text{spam})P(\neg W_3|\text{spam})P(W_4|\text{spam})P(\text{spam})$$

Classification with Naive Bayes (Continued)

- And the probability that the message is ham can be expressed as:

$$P(\text{ham} \mid W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \propto P(W_1 \mid \text{ham}) P(\neg W_2 \mid \text{ham}) P(\neg W_3 \mid \text{ham}) P(W_4 \mid \text{ham}) P(\text{ham})$$

- Note that the equals symbol has been replaced by the proportional-to symbol (similar to a sideways, open- ended '8') to indicate the fact that the denominator has been omitted.

Classification with Naive Bayes (Continued)

- Using the values in the likelihood table, we can start filling numbers in these equations. The overall likelihood of spam is then:
- (4/20) * (10/20) * (20/20) * (12/20) * (20/100) = 0.012
- While the likelihood of ham is:
- (1/80) * (66/80) * (71/80) * (23/80) * (80/100) = 0.002
- Because 0.012/0.002 = 6, we can say that this message is six times more likely to be spam than ham.
- However, to convert these numbers into probabilities, we need to perform one last step to reintroduce the denominator that had been excluded.

Classification with Naive Bayes (Continued)

- Essentially, we must rescale the likelihood of each outcome by dividing it by the total likelihood across all possible outcomes.
- In this way, the probability of spam is equal to the likelihood that the message is spam divided by the likelihood that the message is either spam or ham:
- 0.012/(0.012 + 0.002) = 0.857

Classification with Naive Bayes (Continued)

- Similarly, the probability of ham is equal to the likelihood that the message is ham divided by the likelihood that the message is either spam or ham:
- 0.002/(0.012 + 0.002) = 0.143
- Given the pattern of words found in this message, we expect that the message is spam with 85.7 percent probability and ham with 14.3 percent probability.
- Because these are mutually exclusive and exhaustive events, the probabilities sum to 1.

Classification with Naive Bayes (Continued)

- The Naive Bayes classification algorithm we used in the preceding example can be summarized by the following formula.
- The probability of level L for class C, given the evidence provided by features F_1 through F_n, is equal to the product of the probabilities of each piece of evidence conditioned on the class level, the prior probability of the class level, and a scaling factor 1/Z, which converts the likelihood values into probabilities:

$$P(C_L|F_1, \ldots, F_n) = \frac{1}{Z} p(C_L) \prod_{i=1}^{n} p(F_i|C_L)$$

The Laplace estimator

- Before we employ Naive Bayes in more complex problems, there are some nuances to consider.
- Suppose that we received another message, this time containing all four terms: Congratulations, Earn, Subscribe, Money.
- Using the Naive Bayes algorithm as before, we can compute the likelihood of spam as:
- (4/20) * (10/20) * (0/20) * (12/20) * (20/100) = 0
- The likelihood of ham is: (1/80) * (14/80) * (8/80) * (23/80) * (80/100) = 0.00005
- Therefore, the probability of spam is: 0/(0 + 0.00005) = 0

The Laplace estimator (Continued)

– The probability of ham is:
– 0.00005/(0 + 0. 0.00005) = 1
– These results suggest that the message is spam with 0 percent probability and ham with 100 percent probability.
– P(spam | subscribe) = 0%.

The Laplace estimator (Continued)

- Because probabilities in the Naive Bayes formula are multiplied in a chain, this 0 percent value causes the posterior probability of spam to be zero, giving the word Subscribe the ability to effectively nullify and overrule all of the other evidence.
- Even if the e-mail was otherwise overwhelmingly expected to be spam, the absence of the word Subscribe in spam will always veto the other evidence and result in the probability of spam being zero.
- A solution to this problem involves using something called the **Laplace estimator**, which is named after the French mathematician Pierre- Simon Laplace.

The Laplace estimator (Continued)

- The Laplace estimator essentially **adds a small number to each of the counts in the frequency table,** **which ensures that each feature has a nonzero probability of occurring with each class**.
- Typically, the Laplace estimator is set to 1, which ensures that each class-feature combination is found in the data at least once.

The Laplace estimator (Continued)

- Using a Laplace value of 1, we add one to each numerator in the likelihood function.
- The total number of 1 values must also be added to each conditional probability denominator.
- The likelihood of spam is therefore:
- (5/24) * (11/24) * (1/24) * (13/24) * (20/100) = 0.0004
- The likelihood of ham is:
- (2/84) * (15/84) * (9/84) * (24/84) * (80/100) = 0.0001

The Laplace estimator (Continued)

– This means that the probability of spam is 80 percent, and the probability of ham is 20 percent, which is a more acceptable result than the one obtained when the term Subscribe alone determined the result.

Using numeric features with Naive Bayes

- Because Naive Bayes uses frequency tables to learn the data, each feature must be categorical in order to create the combinations of class and feature values comprising of the matrix.
- Since numeric features do not have categories of values, the preceding algorithm does not work directly with numeric data.
- There are, however, ways that this can be addressed.

Using numeric features with Naive Bayes (Continued)

- One easy and effective solution is to **discretize** numeric features, which simply means that the numbers are put into categories known as **bins**.
- For this reason, **discretization** is also sometimes called **binning**.
- This method is ideal when there are large amounts of training data, a common condition while working with Naive Bayes.

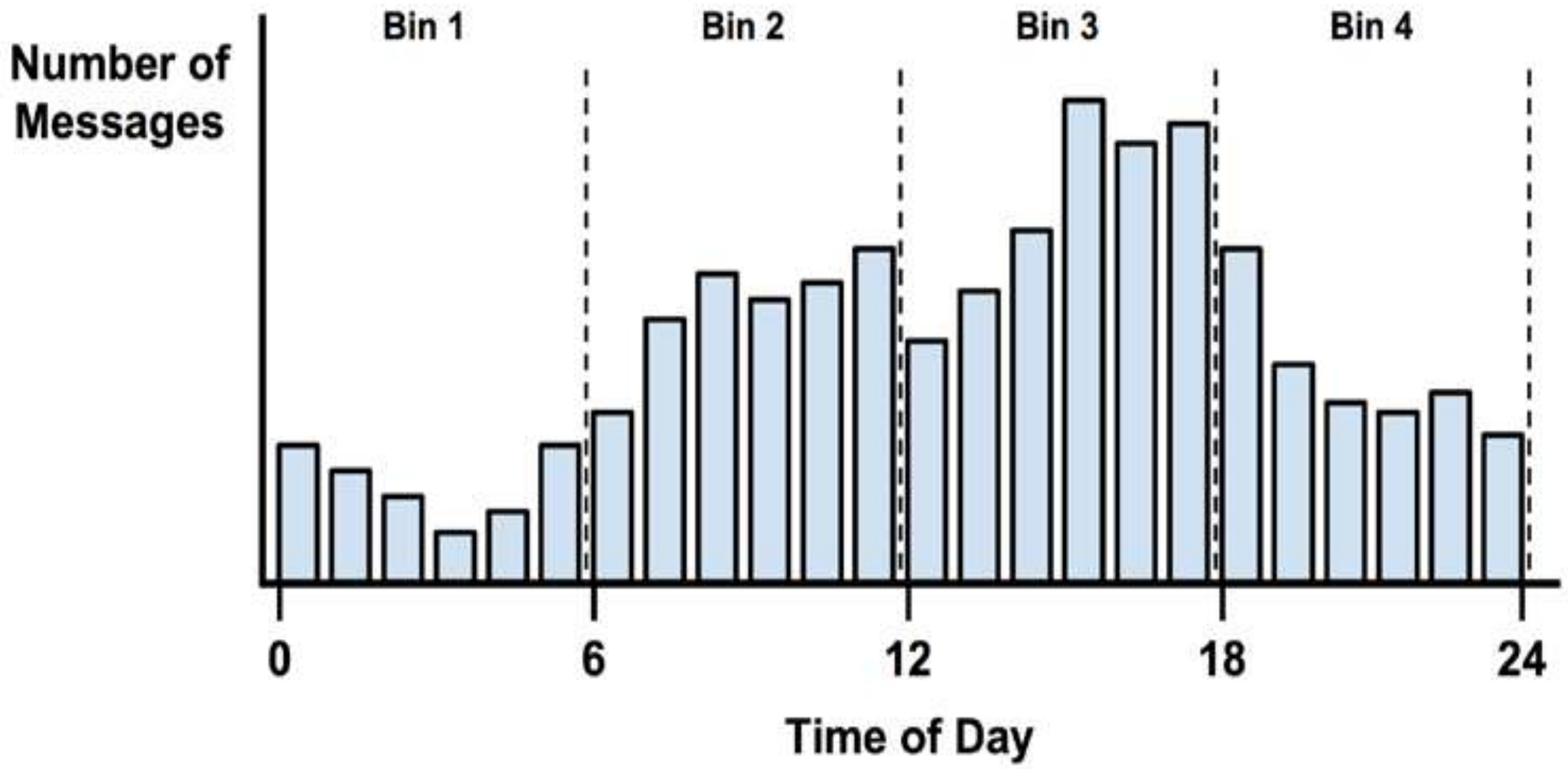
Using numeric features with Naive Bayes (Continued)

- There are several different ways to discretize a numeric feature.
- Perhaps the most common is to explore the data for natural categories or cut points in the distribution of data.
- For example,
  - Suppose that you added a feature to the spam dataset that recorded the time of night or day the e-mail was sent, from 0 to 24 hours past midnight.

Using numeric features with Naive Bayes (Continued)

- Depicted using a histogram, the time data might look something like the following diagram.
- In the early hours of the morning, the message frequency is low.
- The activity picks up during business hours and tapers off in the evening.
- This seems to create four natural bins of activity, as partitioned by the dashed lines indicating places where the numeric data are divided into levels of a new nominal feature, which could then be used with Naive Bayes:

Using numeric features with Naïve Bayes (Continued)



Using numeric features with Naive Bayes (Continued)

- Keep in mind that the choice of four bins was somewhat arbitrary based on the natural distribution of data and a hunch about how the proportion of spam might change throughout the day.
- We might expect that spammers operate in the late hours of the night or they may operate during the day, when people are likely to check their e-mail.
- This said, to capture these trends, we could have just as easily used three bins or twelve.

Using numeric features with Naive Bayes (Continued)

- One thing to keep in mind is that discretizing a numeric feature always results in a reduction of information as the feature's original granularity is reduced to a smaller number of categories.
- It is important to strike a balance here.
- Too few bins can result in important trends being obscured.
- Too many bins can result in small counts in the Naive Bayes frequency table, which can increase the algorithm’s sensitivity to noisy data.