



MODULE 2.2 – LAZY LEARNING

**PREPARED BY,
SHELLY SHIJU GEORGE
ASSISTANT PROFESSOR**

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.

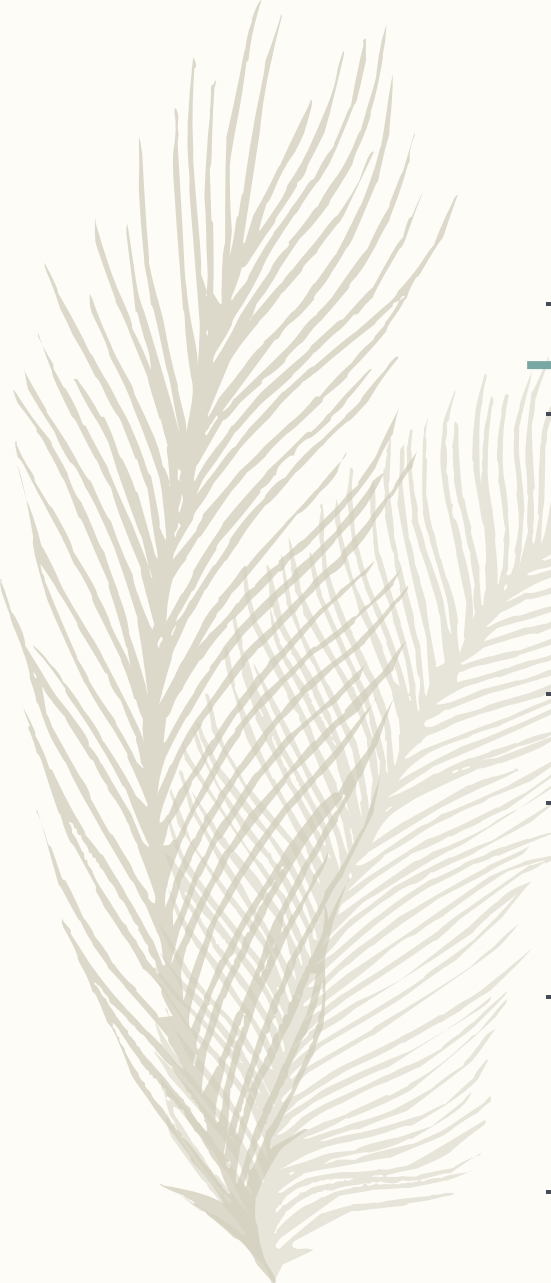


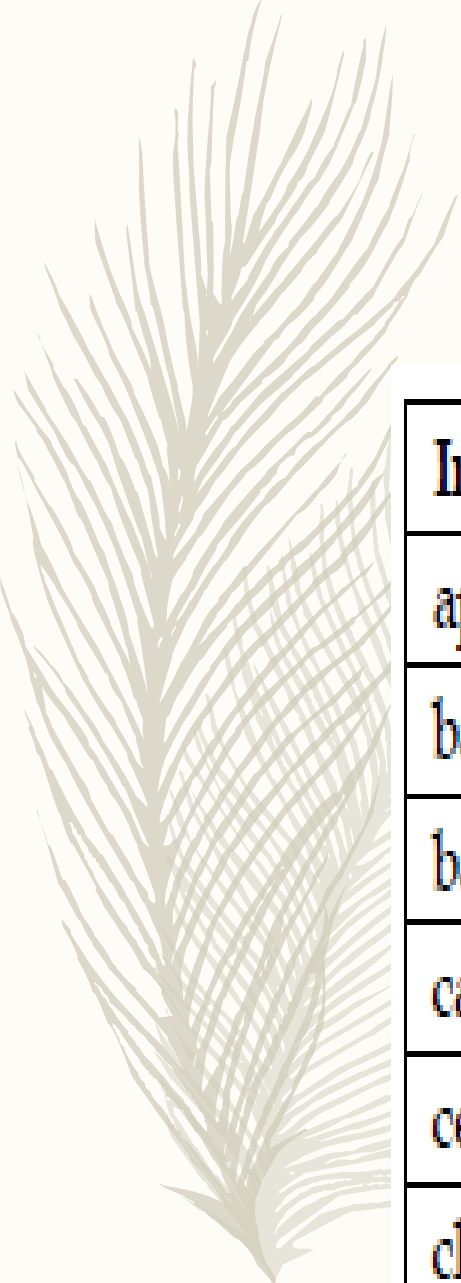
The k-NN algorithm

-
- 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
<ul style="list-style-type: none">• Simple and effective• Makes no assumptions about the underlying data distribution• Fast training phase	<ul style="list-style-type: none">• 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:

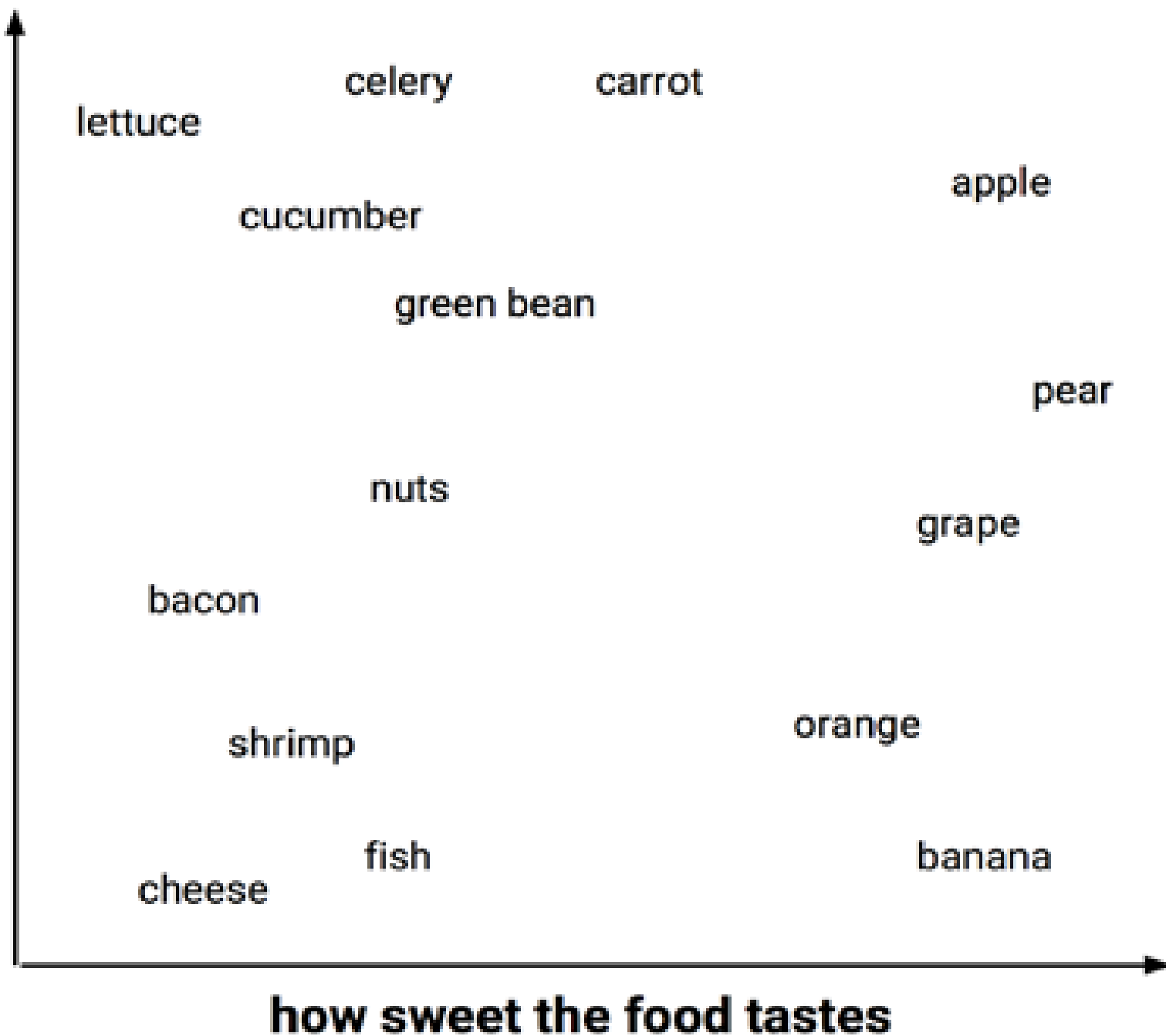


Ingredient	Sweetness	Crunchiness	Food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

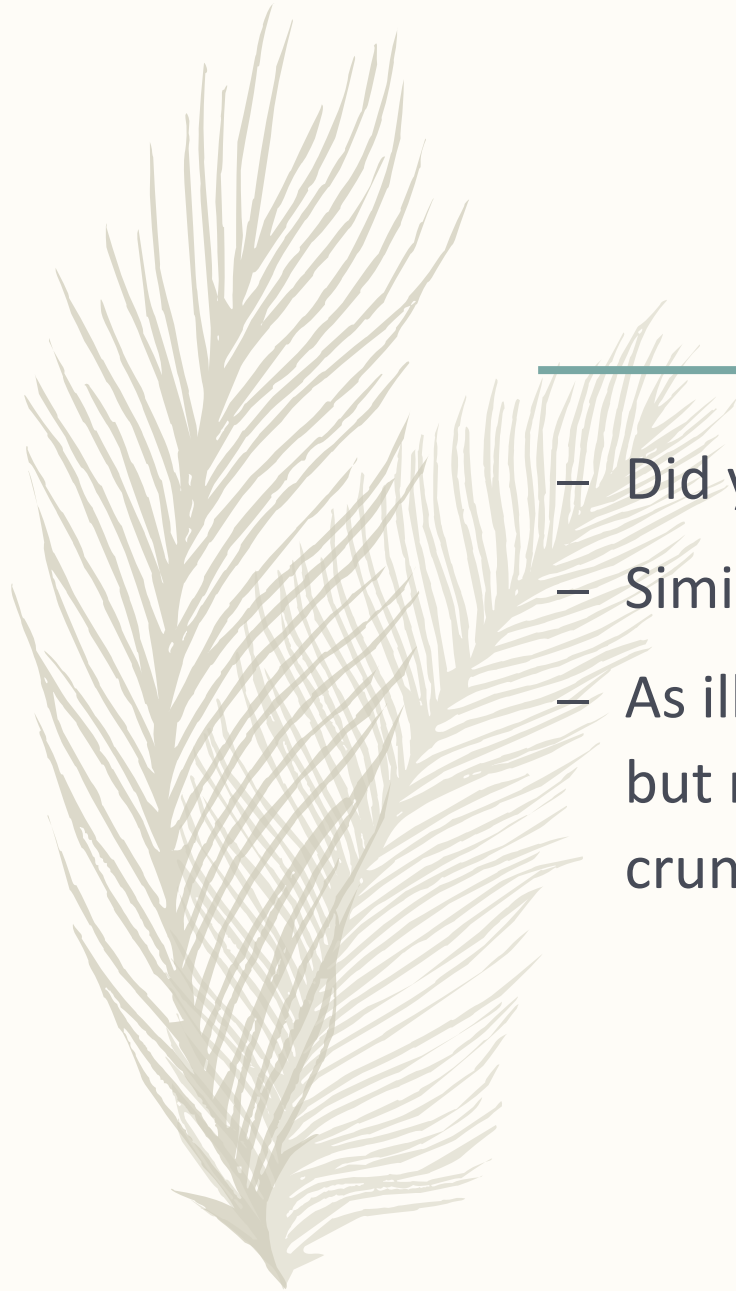
- 
-
- The k-NN algorithm treats the features as coordinates in a multidimensional feature space.
 - As our dataset includes only two features, the feature space is two-dimensional.
 - We can plot two-dimensional data on a scatter plot, with the x dimension indicating the ingredient's sweetness and the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



how crunchy the food is



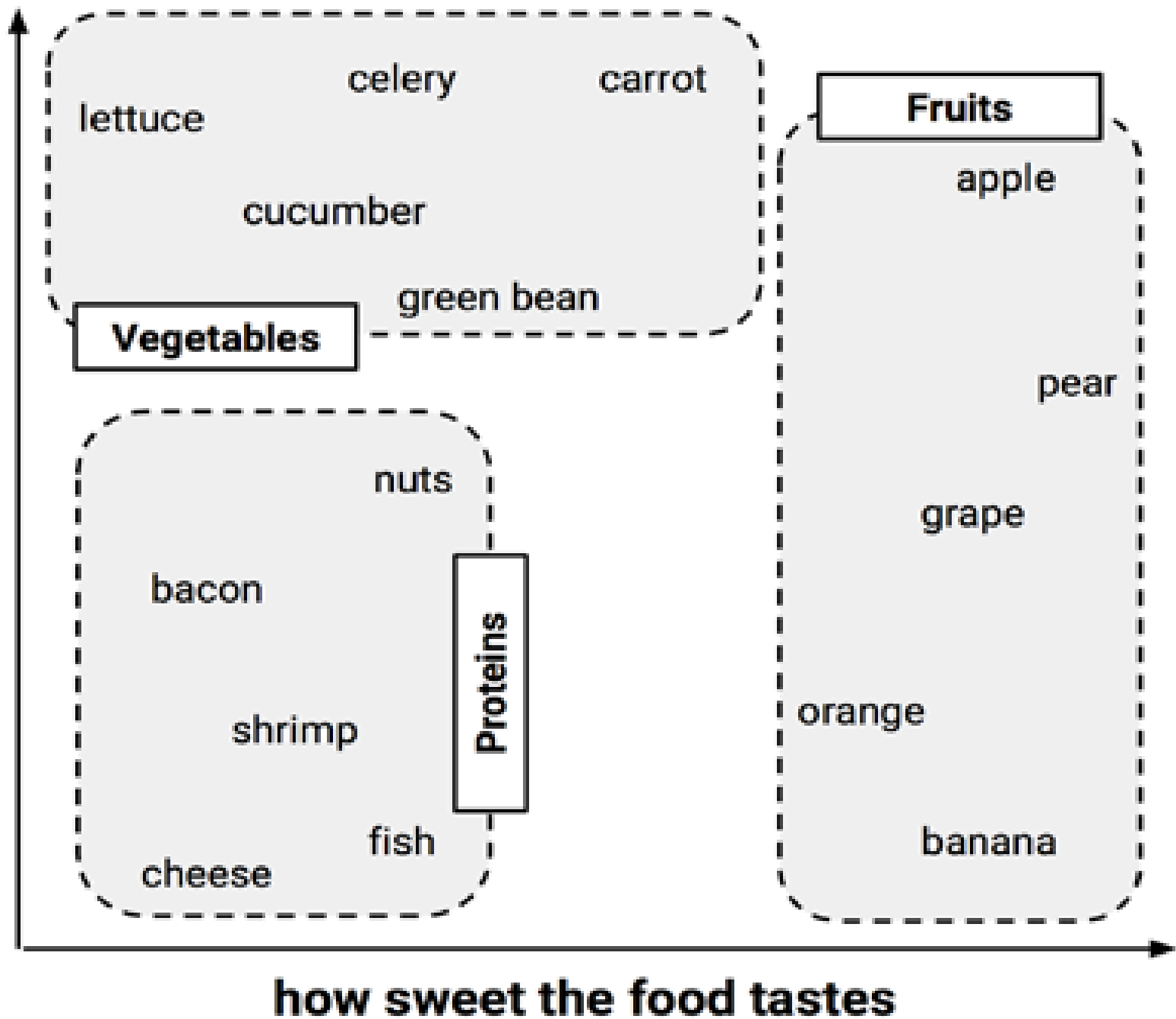
how sweet the food tastes




-
- 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:



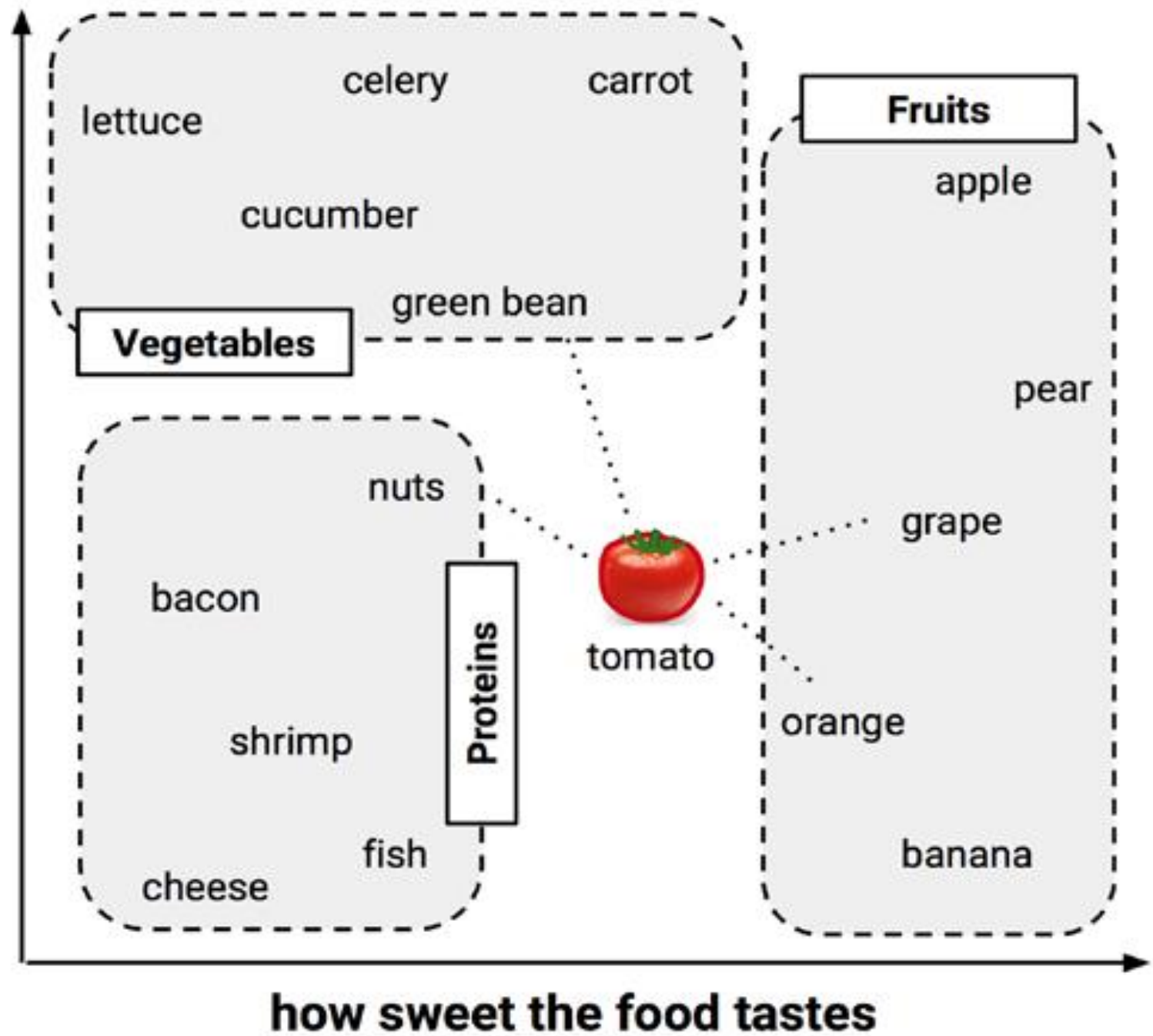
how crunchy the food is




- 
-
- 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 determine which class is a better fit, as shown in the following diagram:



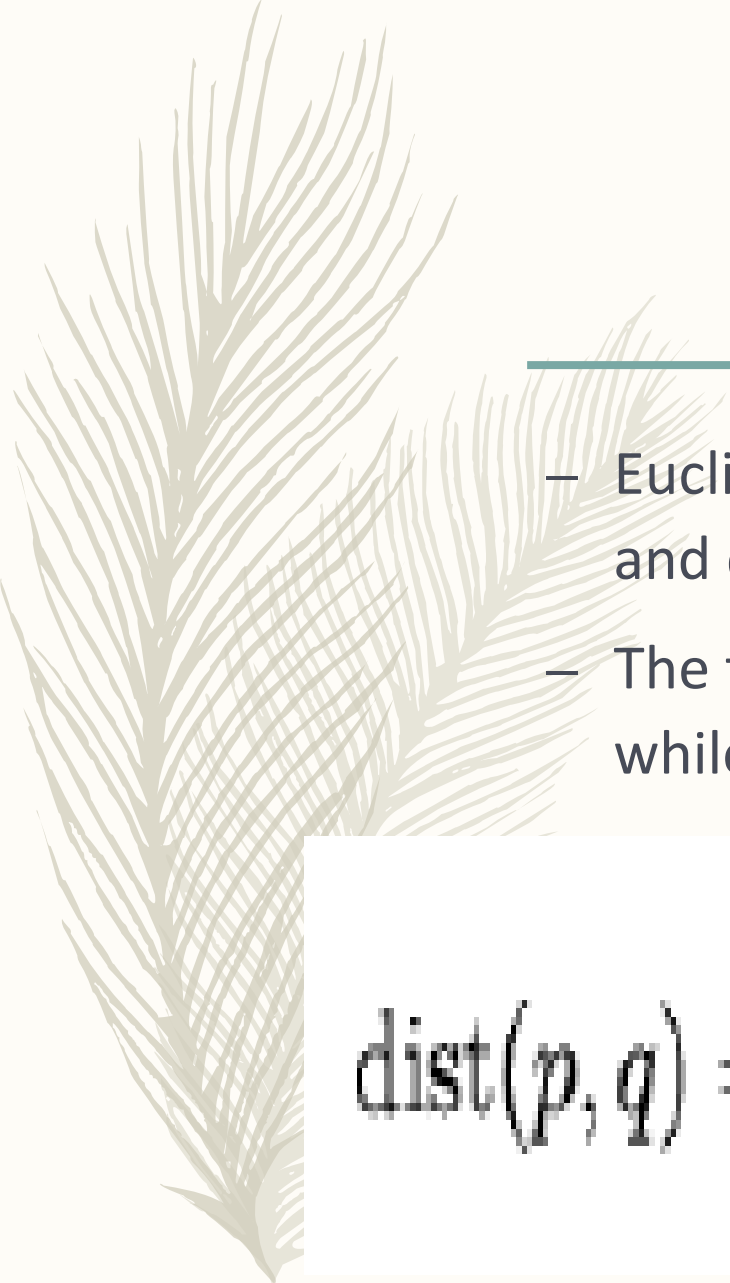
how crunchy the food is



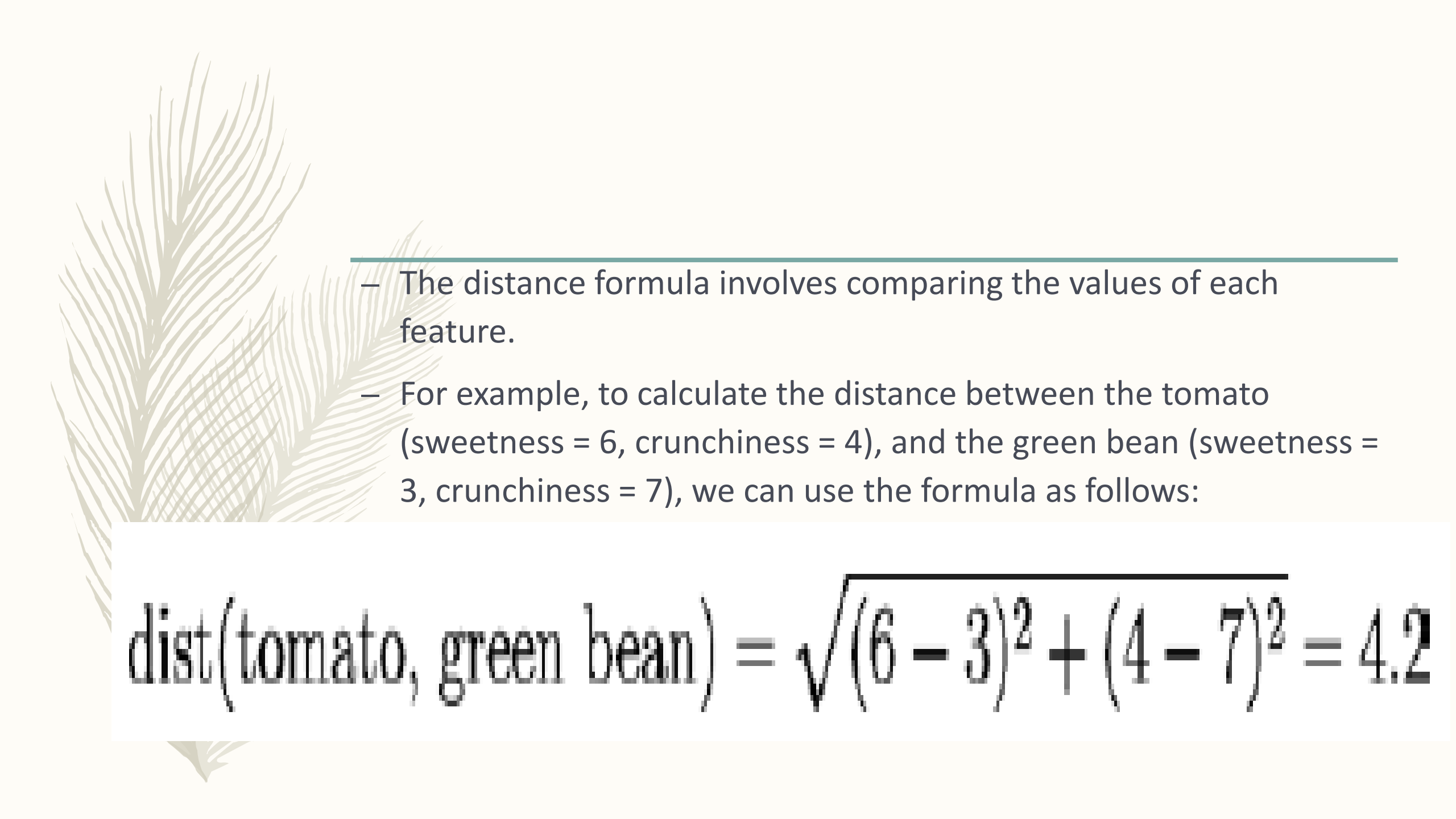


Measuring similarity with distance


- 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:


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

- 
-
- 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:

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

- 
-
- 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	$\text{sqrt}((6 - 8)^2 + (4 - 5)^2) = 2.2$
green bean	3	7	vegetable	$\text{sqrt}((6 - 3)^2 + (4 - 7)^2) = 4.2$
nuts	3	6	protein	$\text{sqrt}((6 - 3)^2 + (4 - 6)^2) = 3.6$
orange	7	3	fruit	$\text{sqrt}((6 - 7)^2 + (4 - 3)^2) = 1.4$

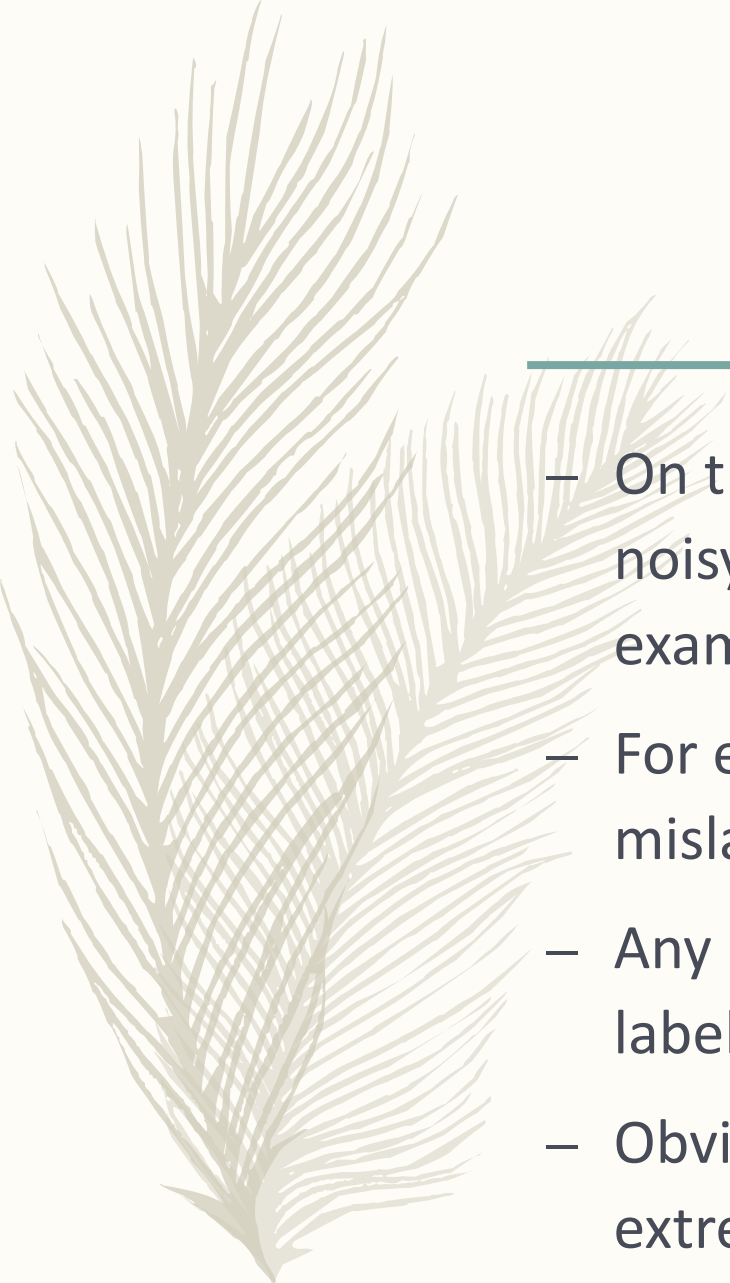
- 
-
- To classify the tomato as a vegetable, protein, or fruit, we'll begin by assigning the tomato, the food type of its single nearest neighbor.
 - This is called 1-NN classification because $k = 1$.
 - The orange is the nearest neighbor to the tomato, with a distance of 1.4.
 - As orange is a fruit, 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.



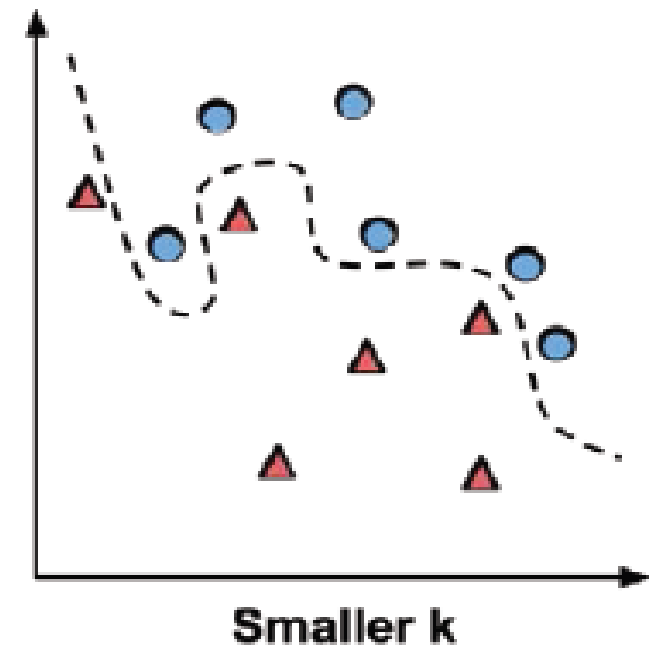
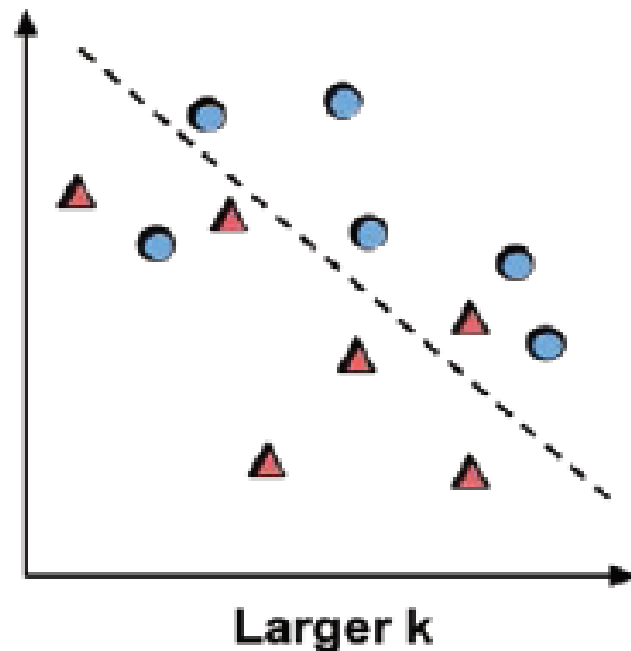
Choosing an appropriate k

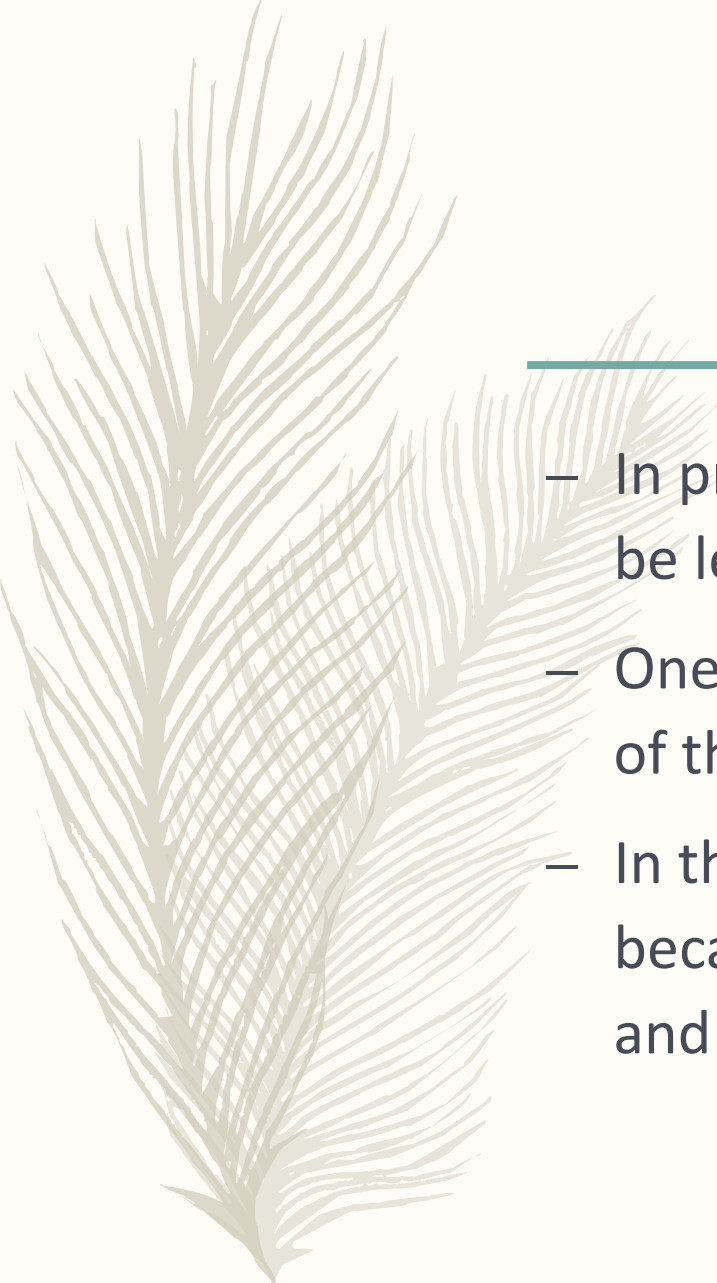
- The decision of how many neighbors to use for k -NN determines how well the model will generalize to future data.
- The balance between overfitting and underfitting the training data is a problem known as bias-variance tradeoff.
- Overfitting – occurs when a statistical model or machine learning algorithm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

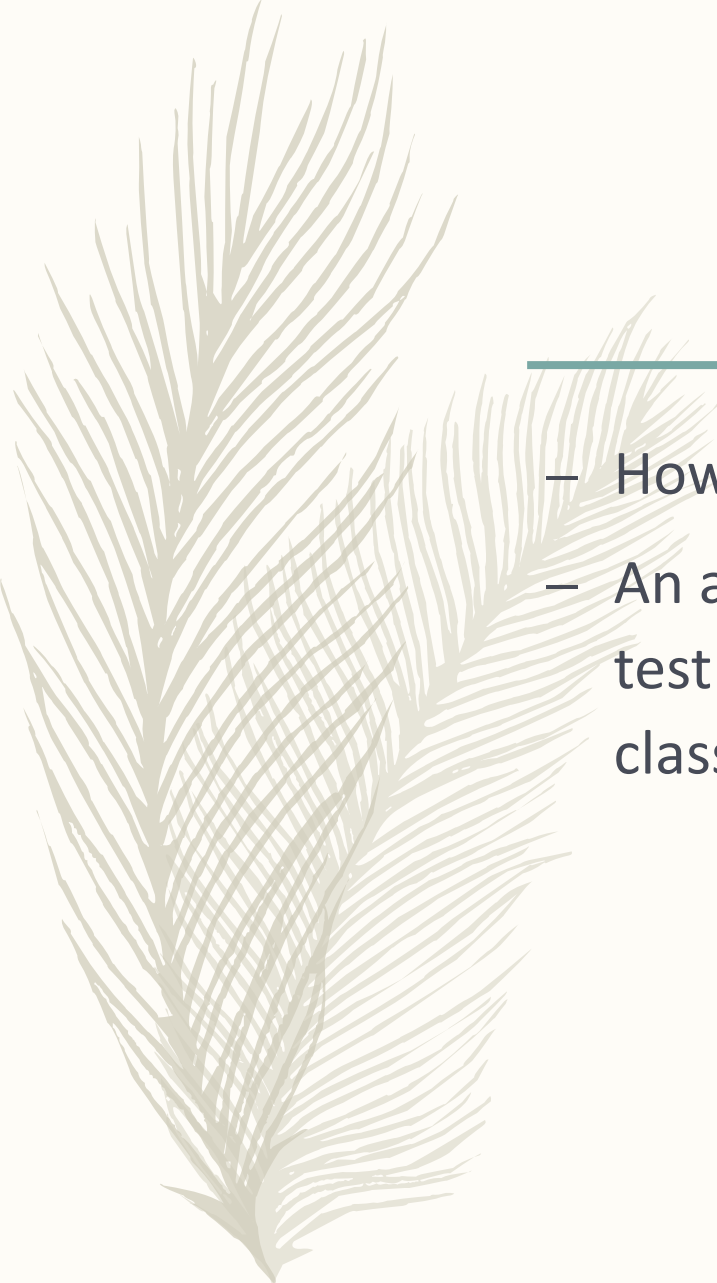
- 
-
- Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner so that it runs the risk of ignoring small, but important patterns.
 - Suppose we took the extreme stance of setting a very large k , as large as the total number of observations in the training data.
 - 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:

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



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:

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




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.

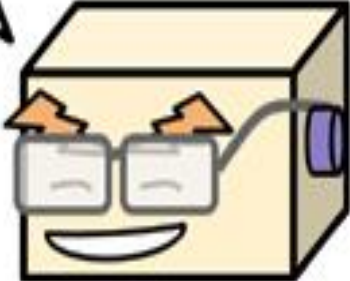


That tastes like
a vegetable.

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.