Lecture 5: K Nearest Neighbor

Naïve Bayes: Recap

- The **Naive Bayes** algorithm describes a simple method to apply Bayes' theorem to classification problems.
  - The Naive Bayes algorithm is named as such because it makes some "naive" assumptions about the data.
    - Naive Bayes assumes that all of the features in the dataset are equally important and independent. (strong assumption)
    - However, in most cases when these assumptions are violated, Naive Bayes still performs fairly well.
- Naive assumptions + Bayes’ theorem

Naïve Bayes: Recap

- $$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$ Bayes' theorem

- $$= \frac{\left(\prod_{i=1}^{n} P(A_i \mid C)\right) P(C)}{P(A_1 A_2 \ldots A_n)}$$ conditional independence assumption

- $$\propto \left(\prod_{i=1}^{n} P(A_i \mid C)\right) P(C)$$ The final prediction depends on $P(A_i|C)$ and P(C)

$$\frac{P(A_i, C)}{P(C)}$$

Naïve Bayes: Recap

WheelType	Auction	IsBadBuy
Alloy	OTHER	Yes
Special	ADESA	No
Alloy	MANHEIM	No
unkwnWheel	OTHER	No
unkwnWheel	OTHER	Yes

- IsBadBuy = Yes (40%; 2 instances)

WheelType:	Alloy	1
	Special	0
	unkwnWheel	1
Auction:	ADESA	0
	MANHEIM	0
	OTHER	2

- IsBadBuy = No (60%; 3 instances)

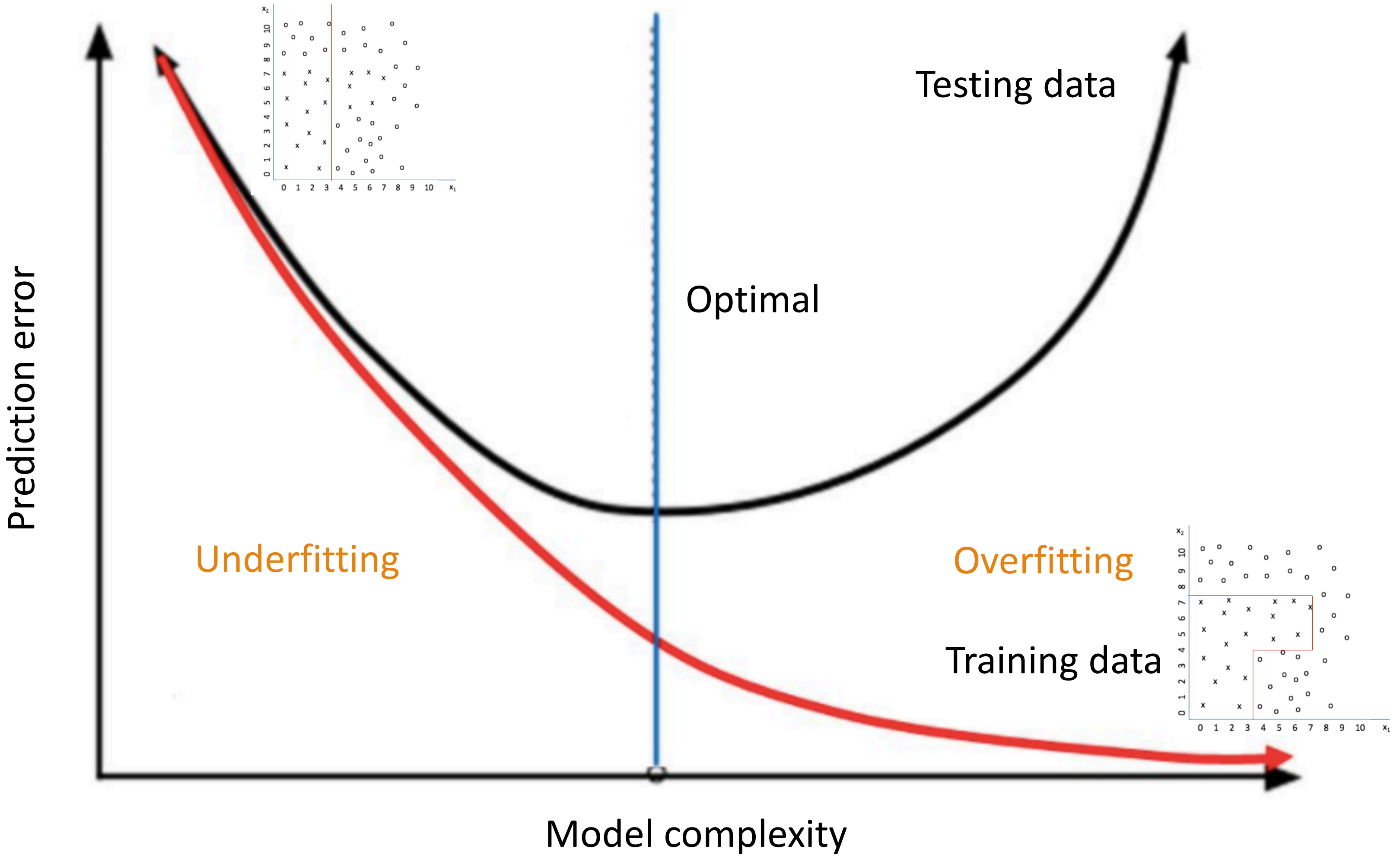
WheelType:	Alloy	1
	Special	1
	unkwnWheel	1
Auction:	ADESA	1
	MANHEIM	1
	OTHER	1

Prediction for (WheelType=unkwnWheel, Auction=OTHER)

$$\left(\prod_{i=1}^{n} P(A_i \mid C)\right) P(C)$$

- P(IsBadBuy = Yes|WheelType=unkwnWheel, Auction=OTHER) ∝ P(WheelType=unkwnWheel| IsBadBuy = Yes) *P(Auction=OTHER| IsBadBuy = Yes)*P(C) = 0.5 * 1 * 0.4 = 0.2
- P(IsBadBuy = No|WheelType=unkwnWheel, Auction=OTHER) ∝ P(WheelType=unkwnWheel| IsBadBuy = No) *P(Auction=OTHER| IsBadBuy = No)*P(C) = 0.333 * 0.333 * 0.6 = 0.0665

Generalization and Overfitting: Recap



Model Comparison: Recap

- Comparing Decision Tree and Naïve Bayes
  - Decision Tree with max depth = 2

ACC	PRECISION1	PRECISION2	TPR1	TPR2	F11	F12
89.63005	89.72765	86.93878	99.47489	23.48401	94.35019	36.97917

ACC	PRECISION1	PRECISION2	TPR1	TPR2	F11	F12
89.59653	89.59697	89.58333	99.61700	22.16495	94.34168	35.53719

target \ pred	No	Yes
No	2601	10
Yes	302	86

  - Naïve Bayes with Laplace smooth = 1

ACC	PRECISION1	PRECISION2	TPR1	TPR2	F11	F12
87.03042	90.55364	49.91763	95.01149	33.40684	92.72902	40.02642

ACC	PRECISION1	PRECISION2	TPR1	TPR2	F11	F12
86.39547	89.66511	45.49550	95.36576	26.03093	92.42762	33.11475

target \ pred	No	Yes
No	2490	121
Yes	287	101

Compare the performances on testing set

Model Comparison: Recap

- Overall model performance comparison
  - Accuracy
- Compare performances on each class
  - Precision: confidence/effectiveness of predictions
  - Recall: ability of identifying instances belonging to a class

F-measure: single metrics combines precision and recall and measures the overall performance on each class

Decision Tree Model

ACC	PRECISION1	PRECISION2	TPR1	TPR2	F11	F12
89.59653	89.59697	89.58333	99.61700	22.16495	94.34168	35.53719

pred		
target	No	Yes
No	2601	10
Yes	302	86

TP/(TP+FP) TN/(TN+FN) TP/(TP+FN) TN/(TN+FP)

Naïve Bayes Model

ACC	PRECISION1	PRECISION2	TPR1	TPR2	F11	F12
86.39547	89.66511	45.49550	95.36576	26.03093	92.42762	33.11475

pred		
target	No	Yes
No	2490	121
Yes	287	101

Overview

- Understanding K Nearest Neighbor
  - The K-NN algorithm
    - Measuring similarity with distance
    - Choosing an appropriate K
    - Preparing data for use with K-NN
- Cross Validation

Intuition of Nearest Neighbor Classification

- Dark Dining
  - Customers are served in a completely darkened restaurant by waiters who move carefully around memorized routes using only their sense of touch and sound.
  - The basic concept is that the removal of vision enhances the other senses, like taste and smell, and food will be experienced in new ways.

Intuition of Nearest Neighbor Classification

- Dark Dining
  - Determine the type of food: fruits, vegetables, or proteins.
    - Experience

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

Intuition of Nearest Neighbor Classification

- Birds of a feather flock together
  - Things that are alike are likely to have properties that are alike.
  - Machine learning uses this principle to classify data by placing it in the same category as similar or "nearest" neighbors.
- Nearest neighbor classifiers: classify unlabeled examples/instances by assigning them the class of similar labeled examples/instances.

K Nearest Neighbor Algorithm

- 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.
  - A training dataset made up of examples that have been classified into several categories.
  - 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.

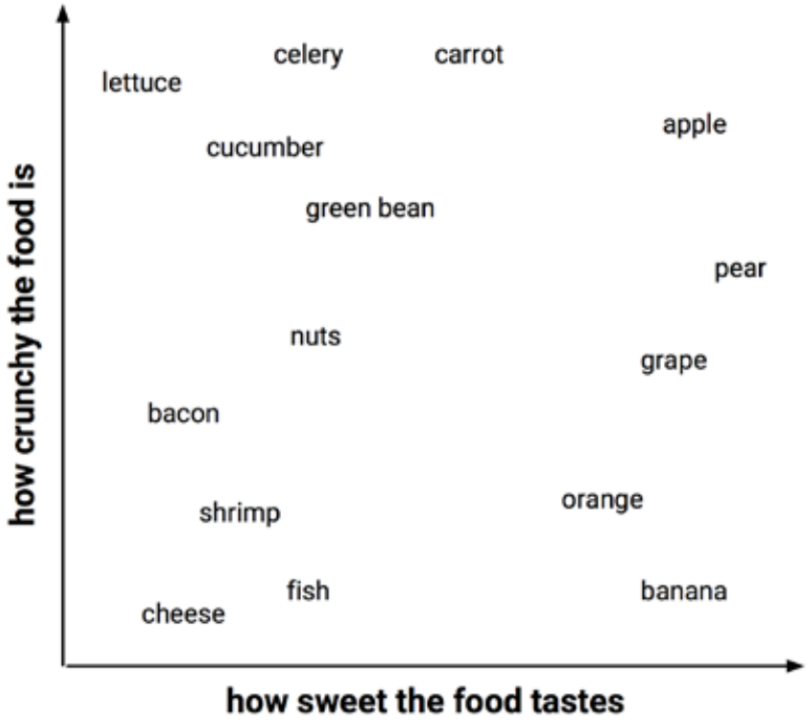
Intuition of Nearest Neighbor Classification

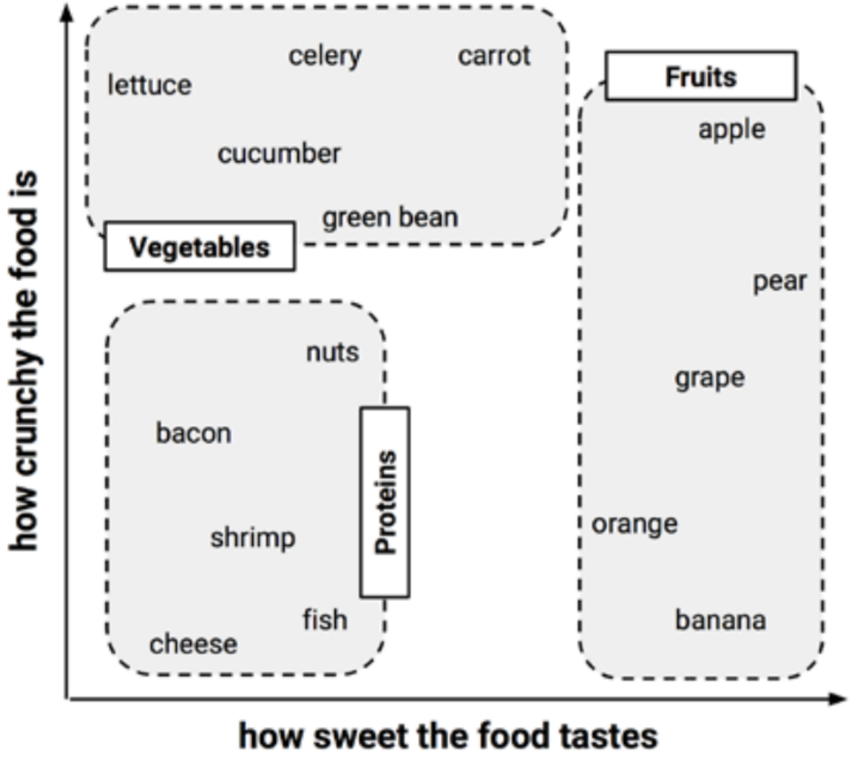
- Dark Dining
  - Suppose that prior to eating the mystery meal we had created a dataset in which we recorded our impressions of a number of ingredients we tasted previously.
  - To keep things simple, we rated only two features of each ingredient.
    - The first is a measure from 1 to 10 of how **crunchy** the ingredient is.
    - The second is a 1 to 10 score of how **sweet** the ingredient tastes.
  - We labeled each ingredient as one of the three types of food: fruit, vegetable, or protein.

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

K Nearest Neighbor Algorithm

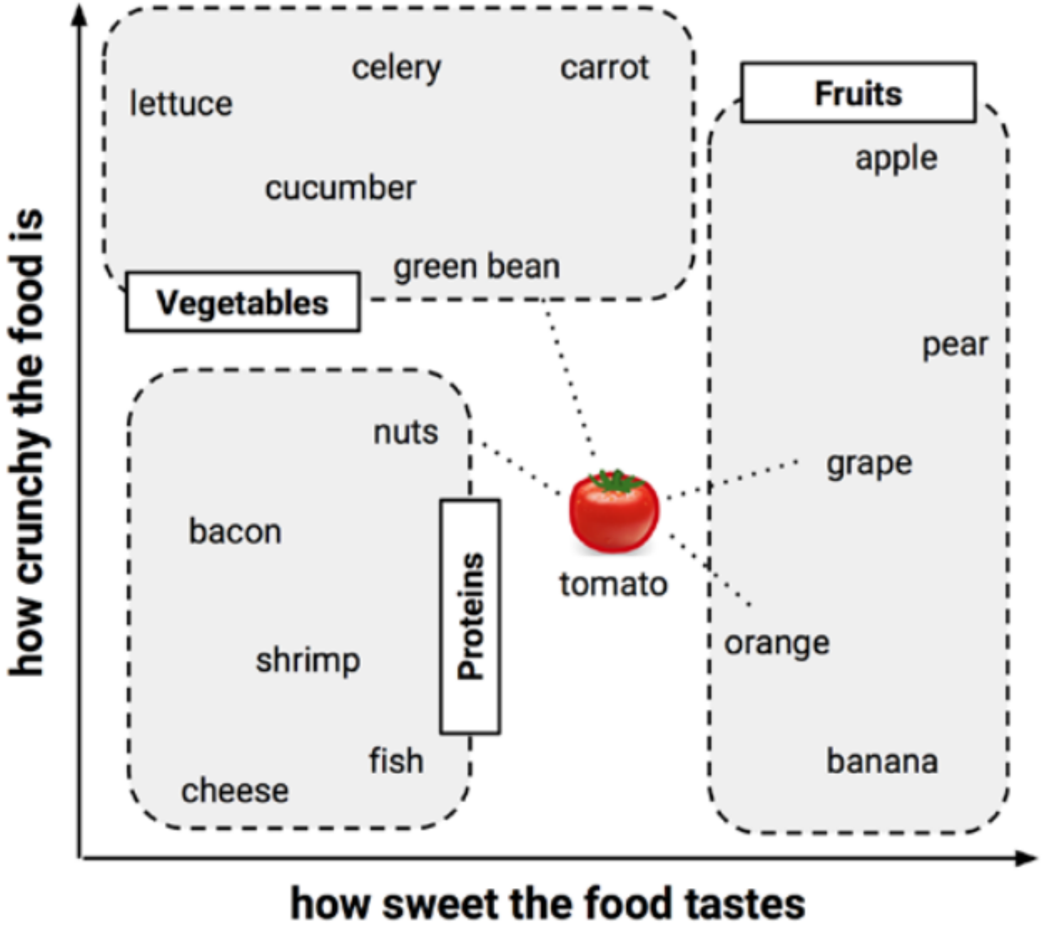
- Similar types of food tend to be grouped closely together





K Nearest Neighbor Algorithm

- Suppose you are given a tomato: protein, fruit or vegetable?
- We can use the nearest neighbor approach to determine which class is a better fit.



Measuring Similarity with Distance

- We use **distance function**, or a formula that measures the similarity between the two instances.
  - An instance (e.g., a customer) has a list of variables
    - e.g., attributes of a customer such as age, spending, gender etc.
  - To measure similarity between two instances, we measure similarity between these instances' attribute values based on a distance function.
    - Input: attribute values of two instances
    - Output: distance

Measuring Similarity with Distance

- Distance measure (how similar or dissimilar two instances are) satisfies these conditions:
  - Non-negative
  - Distance between the same instances = 0
  - Symmetric, distance (A,B) = distance (B,A)
  - The distance between two instances, A & B, is no longer than the sum of the distance from A to another object C and the distance from C to B

Measuring Similarity with Distance

- There are many different ways to calculate distance. The most commonly used measures are
  - Manhattan distance
  - Euclidean distance

Measuring Similarity with Distance

- For two instances X and Y with n variables, Manhattan distance is defined as:

$$d(X,Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

where $x_1 \ldots x_n$ are values of variables of instance X

and $y_1 \ldots y_n$ are values of variables of instance Y

Measuring Similarity with Distance

- E.g., Manhattan distance (Tom, Jack)=|32-35|+|5400-6000|=603

NAME	AGE	SPENDING($)
SUE	21	2300
CARL	27	2600
TOM	32	5400
JACK	35	6000
DAN	44	6200
JILL	50	7000

Measuring Similarity with Distance

- For two instances X and Y with *n* variables, Euclidean distance is defined as:

$$d(X,Y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2}$$

where $x_1 \ldots x_n$ are values of variables of instance X

and $y_1 \ldots y_n$ are values of variables of instance Y

Measuring Similarity with Distance

- E.g., Euclidean distance (Tom, Jack)=

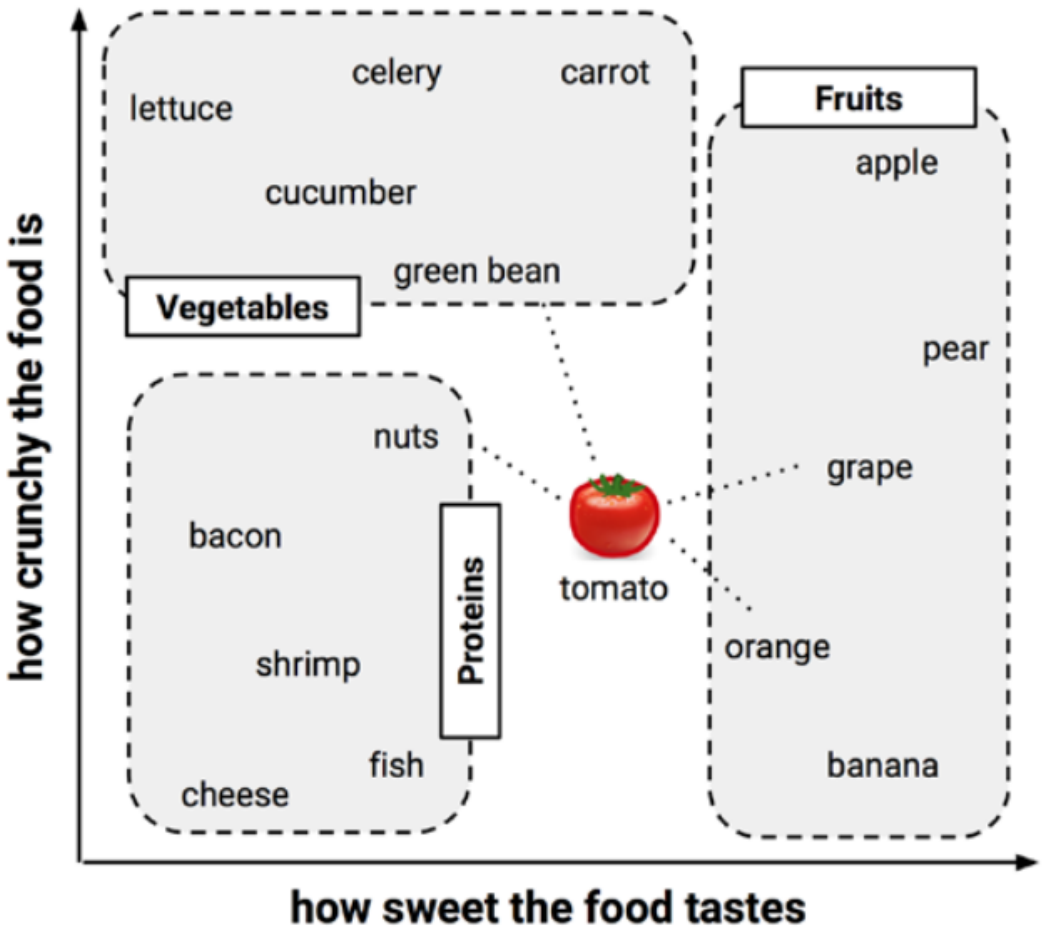
$$\sqrt{(32-35)^2+(5400-6000)^2} = 60.075$$

NAME	AGE	SPENDING($)
SUE	21	2300
CARL	27	2600
TOM	32	5400
JACK	35	6000
DAN	44	6200
JILL	50	7000

The distance is dominated by the second variable

Measuring Similarity with Distance

- Traditionally, the k-NN algorithm uses Euclidean distance.
  - For example, to calculate the distance between tomato *(sweetness = 6, crunchiness = 4)* and several of its closest neighbors:



Measuring Similarity with Distance

- Traditionally, the k-NN algorithm uses Euclidean distance.
  - For example, to calculate the distance between tomato *(sweetness = 6, crunchiness = 4)* and several of its closest neighbors:

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*

Choosing an Appropriate k

- 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.

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*

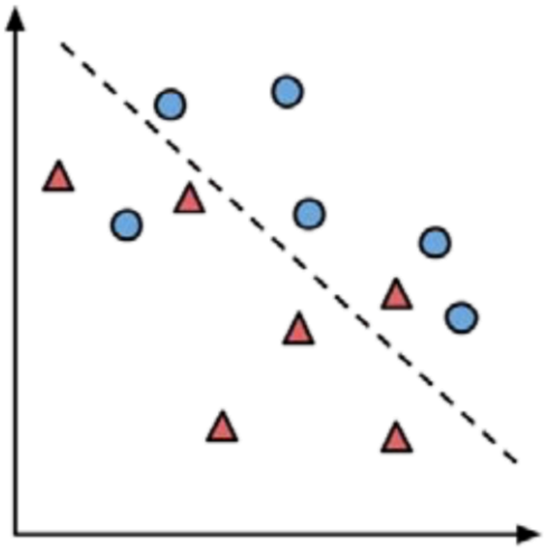
Choosing an Appropriate k

- The decision of how many neighbors to use for k-NN determines how well the model will generalize to future data.
  - 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. UNDERFIT K = total number of instances in the training data
  - On the opposite extreme, using a single nearest neighbor allows the noisy data or outliers to unduly influence the classification of examples. OVERFIT K = 1

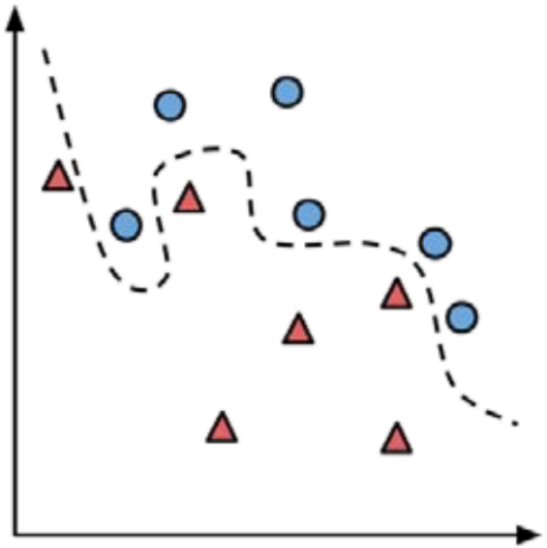
Most common category

Choosing an Appropriate k

- 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.



Larger k



Smaller k

Choosing an Appropriate k

- 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.
  - A less common, but interesting solution to this problem is to choose a larger *k*, but apply a **weighted voting** process in which the vote of the closer neighbors is considered more authoritative than the vote of the far away neighbors.

Preparing Data for Use with K-NN

- Features are typically transformed to a **standard range** prior to applying the k-NN algorithm.
  - The rationale for this step is that the distance formula is highly dependent on how features are measured.
  - If certain features have a much larger range of values than the others, the distance measurements will be strongly dominated by the features with larger ranges.

Preparing Data for Use with K-NN

- Dark Dining
  - Add an additional feature to describe a food's spiciness, which is measured using the Scoville scale.
    - Scoville scale: standardized measure of spice heat, range from zero to over a million.

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$$

$$\text{where } x_1 \ldots x_n \text{ are values of variables of object X}$$

$$\text{and } y_1 \ldots y_n \text{ are values of variables of object Y}$$

    - Difference between sweet and non-sweet or crunchy and non-crunchy foods is at most 10.
    - Difference between spicy and non-spicy foods can be over a million

Preparing Data for Use with K-NN

- Features are typically transformed to a **standard range** prior to applying the k-NN algorithm.
  - Each feature contributes relatively equally to the distance formula
  - **min-max normalization**

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

  - **z-score standardization**

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

Preparing Data for Use with K-NN

- E.g., Euclidean distance (Tom, Jack)=

$$\sqrt{(32-35)^2+(5400-6000)^2} = 60.075$$

NAME	AGE	SPENDING($)
SUE	21	2300
CARL	27	2600
TOM	32	5400
JACK	35	6000
DAN	44	6200
JILL	50	7000

The distance is dominated by the second variable

Preparing Data for Use with K-NN

NAME	AGE	SPENDING($)
SUE	21	2300
CARL	27	2600
TOM	32	5400
JACK	35	6000
DAN	44	6200
JILL	50	7000

NAME	AGE	SPENDING($)
SUE	0	0
CARL	0.207	0.064
TOM	0.379	0.660
JACK	0.483	0.787
DAN	0.793	0.830
JILL	1	1

- Euclidean distance (Tom, Jack)=

$$\sqrt{(0.379-0.483)^2+(0.660-0.787)^2} = 0.164$$

Preparing Data for Use with K-NN

- The Euclidean distance is not defined for categorical variables.
  - Convert categorical variables into numeric format – **dummy coding**
    - **Create dummy variables to replace the original categorical variable.**
    - Dummy coding for a binary variable: gender

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

    - Dummy coding for n-category variable: temperature variable (for example, hot, medium, or cold)

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

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

We need n-1 dummy variables

Preparing Data for Use with K-NN

NAME	AGE	SPENDING($)	GENDER
SUE	21	2300	F
CARL	27	2600	M
TOM	32	5400	M
JACK	35	6000	M
DAN	44	6200	M
JILL	50	7000	F

NAME	AGE	SPENDING($)	GENDER
SUE	0	0	1
CARL	0.207	0.064	0
TOM	0.379	0.660	0
JACK	0.483	0.787	0
DAN	0.793	0.830	0
JILL	1	1	1

K Nearest Neighbor Evaluation

Auction	Color	IsBadBuy	MMRCurrentAu	Size	TopThreeAm	VehBCost	VehicleAge	VehOdo	WarrantyCos	WheelType
ADESA	WHITE	No	2871	LARGE TRUC	FORD	5300	8	75419	869	Alloy
ADESA	GOLD	Yes	1840	VAN	FORD	3600	8	82944	2322	Alloy
ADESA	RED	No	8931	SMALL SUV	CHRYSLER	7500	4	57338	588	Alloy
ADESA	GOLD	No	8320	CROSSOVER	FORD	8500	5	55909	1169	Alloy
ADESA	GREY	No	11520	LARGE TRUC	FORD	10100	5	86702	853	Alloy
ADESA	SILVER	No	2659	COMPACT	GM	4100	7	73810	1455	Covers
ADESA	RED	No	4645	VAN	FORD	5600	5	85003	1633	Covers
ADESA	SILVER	No	4352	LARGE	GM	5900	5	88991	2152	Covers
ADESA	SILVER	No	5142	MEDIUM	GM	6600	5	80077	1373	Alloy
ADESA	MAROON	No	9983	MEDIUM	OTHER	7500	3	71952	1272	Alloy
ADESA	WHITE	No	4165	MEDIUM	OTHER	6200	4	23881	462	Covers
ADESA	GOLD	No	2422	VAN	GM	5100	9	83238	5392	Alloy
ADESA	SILVER	No	6603	MEDIUM	OTHER	7300	3	68165	728	Covers
ADESA	GREEN	No	6149	LARGE	FORD	6600	5	93346	1774	Alloy
ADESA	SILVER	Yes	6057	MEDIUM	CHRYSLER	6400	3	73963	1389	Covers
ADESA	SILVER	No	8113	SPECIALTY	CHRYSLER	10400	5	64839	1215	Alloy
ADESA	RED	No	6702	MEDIUM	GM	7100	4	63151	923	Covers
ADESA	MAROON	No	3320	MEDIUM	GM	4700	7	92782	1209	Alloy
ADESA	GREY	No	7708	SPECIALTY	CHRYSLER	9400	5	72592	1389	Alloy
ADESA	WHITE	No	2700	MEDIUM	GM	3900	8	88667	2712	Alloy
ADESA	RED	No	7860	MEDIUM	CHRYSLER	7500	2	50644	754	Covers
ADESA	SILVER	No	7785	LARGE	GM	8300	3	58384	1500	Alloy
ADESA	BLUE	No	8091	LARGE SUV	FORD	9500	6	80906	1113	Alloy
ADESA	WHITE	No	6793	SMALL SUV	OTHER	7935	5	59801	754	Alloy
ADESA	WHITE	No	6741	MEDIUM SU	FORD	9335	6	77178	1740	unkwnWheel
ADESA	GREY	No	3895	SMALL SUV	FORD	7100	8	79030	1220	unkwnWheel
ADESA	SILVER	Yes	6554	MEDIUM	OTHER	6700	4	61315	728	Alloy
ADESA	SILVER	No	2988	MEDIUM	GM	4700	9	92792	2651	Alloy
ADESA	GREY	No	5396	SPORTS	FORD	6600	6	82271	853	Alloy

70% training data

30% testing data

K Nearest Neighbor Evaluation

- Train K Nearest Neighbor on **training data (70%)**

Auction	Color	IsBadBuy	MMRCurrentAu	Size	TopThreeAm	VehBCost	VehicleAge	VehOdo	WarrantyCos	WheelType
ADESA	WHITE	No	2871	LARGE TRUC	FORD	5300	8	75419	869	Alloy
ADESA	GOLD	Yes	1840	VAN	FORD	3600	8	82944	2322	Alloy
ADESA	RED	No	8931	SMALL SUV	CHRYSLER	7500	4	57338	588	Alloy
ADESA	GOLD	No	8320	CROSSOVER	FORD	8500	5	55909	1169	Alloy
ADESA	GREY	No	11520	LARGE TRUC	FORD	10100	5	86702	853	Alloy
ADESA	SILVER	No	2659	COMPACT	GM	4100	7	73810	1455	Covers
ADESA	RED	No	4645	VAN	FORD	5600	5	85003	1633	Covers
ADESA	SILVER	No	4352	LARGE	GM	5900	5	88991	2152	Covers
ADESA	SILVER	No	5142	MEDIUM	GM	6600	5	80077	1373	Alloy
ADESA	MAROON	No	9983	MEDIUM	OTHER	7500	3	71952	1272	Alloy
ADESA	WHITE	No	4165	MEDIUM	OTHER	6200	4	23881	462	Covers
ADESA	GOLD	No	2422	VAN	GM	5100	9	83238	5392	Alloy
ADESA	SILVER	No	6603	MEDIUM	OTHER	7300	3	68165	728	Covers
ADESA	GREEN	No	6149	LARGE	FORD	6600	5	93346	1774	Alloy
ADESA	SILVER	Yes	6057	MEDIUM	CHRYSLER	6400	3	73963	1389	Covers
ADESA	SILVER	No	8113	SPECIALTY	CHRYSLER	10400	5	64839	1215	Alloy
ADESA	RED	No	6702	MEDIUM	GM	7100	4	63151	923	Covers
ADESA	MAROON	No	3320	MEDIUM	GM	4700	7	92782	1209	Alloy
ADESA	GREY	No	7708	SPECIALTY	CHRYSLER	9400	5	72592	1389	Alloy
ADESA	WHITE	No	2700	MEDIUM	GM	3900	8	88667	2712	Alloy
ADESA	RED	No	7860	MEDIUM	CHRYSLER	7500	2	50644	754	Covers
ADESA	SILVER	No	7785	LARGE	GM	8300	3	58384	1500	Alloy
ADESA	BLUE	No	8091	LARGE SUV	FORD	9500	6	80906	1113	Alloy
ADESA	WHITE	No	6793	SMALL SUV	OTHER	7935	5	59801	754	Alloy
ADESA	WHITE	No	6741	MEDIUM SU	FORD	9335	6	77178	1740	unkwnWheel
ADESA	GREY	No	3895	SMALL SUV	FORD	7100	8	79030	1220	unkwnWheel
ADESA	SILVER	Yes	6554	MEDIUM	OTHER	6700	4	61315	728	Alloy
ADESA	SILVER	No	2988	MEDIUM	GM	4700	9	92792	2651	Alloy
ADESA	GREY	No	5396	SPORTS	FORD	6600	6	82271	853	Alloy

K-NN Stores the training data

K Nearest Neighbor Evaluation

- Make predictions on **testing data (30%)** and **training data (70%)**
  - Search through its similarities with training instances to find *k* nearest neighbors in the training set.
  - Estimate its target variable value based on *k* nearest neighbors' known target values.
    - Binary Classification – e.g. majority class of *k* nearest neighbors

K Nearest Neighbor Evaluation

Auction	Color	IsBadBuy	MMRCurrentAu	Size	TopThreeAm	VehBCost	VehicleAge	VehOdo	WarrantyCos	WheelType
ADESA	RED		7860	MEDIUM	CHRYSLER	7500	2	50644	754	Covers

Auction	Color	IsBadBuy	MMRCurrentAu	Size	TopThreeAm	VehBCost	VehicleAge	VehOdo	WarrantyCos	WheelType
ADESA	WHITE	No	2871	LARGE TRUC	FORD	5300	8	75419	869	Alloy
ADESA	GOLD	Yes	1840	VAN	FORD	3600	8	82944	2322	Alloy
ADESA	RED	No	8931	SMALL SUV	CHRYSLER	7500	4	57338	588	Alloy
ADESA	GOLD	No	8320	CROSSOVER	FORD	8500	5	55909	1169	Alloy
ADESA	GREY	No	11520	LARGE TRUC	FORD	10100	5	86702	853	Alloy
ADESA	SILVER	No	2659	COMPACT	GM	4100	7	73810	1455	Covers
ADESA	RED	No	4645	VAN	FORD	5600	5	85003	1633	Covers
ADESA	SILVER	No	4352	LARGE	GM	5900	5	88991	2152	Covers
ADESA	SILVER	No	5142	MEDIUM	GM	6600	5	80077	1373	Alloy
ADESA	MAROON	No	9983	MEDIUM	OTHER	7500	3	71952	1272	Alloy
ADESA	WHITE	No	4165	MEDIUM	OTHER	6200	4	23881	462	Covers
ADESA	GOLD	No	2422	VAN	GM	5100	9	83238	5392	Alloy
ADESA	SILVER	No	6603	MEDIUM	OTHER	7300	3	68165	728	Covers
ADESA	GREEN	No	6149	LARGE	FORD	6600	5	93346	1774	Alloy
ADESA	SILVER	Yes	6057	MEDIUM	CHRYSLER	6400	3	73963	1389	Covers
ADESA	SILVER	No	8113	SPECIALTY	CHRYSLER	10400	5	64839	1215	Alloy
ADESA	RED	No	6702	MEDIUM	GM	7100	4	63151	923	Covers
ADESA	MAROON	No	3320	MEDIUM	GM	4700	7	92782	1209	Alloy
ADESA	GREY	No	7708	SPECIALTY	CHRYSLER	9400	5	72592	1389	Alloy
ADESA	WHITE	No	2700	MEDIUM	GM	3900	8	88667	2712	Alloy
ADESA	RED	No	7860	MEDIUM	CHRYSLER	7500	2	50644	754	Covers
ADESA	SILVER	No	7785	LARGE	GM	8300	3	58384	1500	Alloy
ADESA	BLUE	No	8091	LARGE SUV	FORD	9500	6	80906	1113	Alloy
ADESA	WHITE	No	6793	SMALL SUV	OTHER	7935	5	59801	754	Alloy
ADESA	WHITE	No	6741	MEDIUM SU	FORD	9335	6	77178	1740	unkwnWheel
ADESA	GREY	No	3895	SMALL SUV	FORD	7100	8	79030	1220	unkwnWheel
ADESA	SILVER	Yes	6554	MEDIUM	OTHER	6700	4	61315	728	Alloy
ADESA	SILVER	No	2988	MEDIUM	GM	4700	9	92792	2651	Alloy
ADESA	GREY	No	5396	SPORTS	FORD	6600	6	82271	853	Alloy

Calculate similarities

find *k* nearest neighbors in the training set

Make prediction: majority class of *k* nearest neighbors

K Nearest Neighbor

- Training phase:
  - Import and store training data in the “memory”
  - No actual “model” is built
  - Fast but memory intensive
- Prediction phase:
  - The main work and time consuming
- Common characterization
  - Instanced-based learning algorithms
  - Memory-based learning algorithms
  - Lazy learners

K Nearest Neighbor

- Issues with K Nearest Neighbor
  - Dimensionality and Domain knowledge
    - There is a problem with having too many attributes, or many that are irrelevant to the similarity judgment.
      - Since all of the attributes (dimensions) contribute to the distance calculation, instance similarity can be confused and misled by the presence of too many irrelevant attributes.

K Nearest Neighbor

- Issues with K Nearest Neighbor
  - Computational Efficiency
    - The main computational cost of a nearest neighbor method occurs in the prediction step, when the training instances must be queried to find nearest neighbors of a new instance, which can be very time consuming.

K Nearest Neighbor

- Issues with K Nearest Neighbor
  - Does not produce a model, limiting the ability to understand how the features are related to the class
  - Require selection of an appropriate *k*

Cross Validation

- The repeated holdout is the basis of a technique known as **k-fold cross-validation** (or **k-fold CV**), which has become the industry standard for estimating model performance.
  - k-fold CV randomly divides the data into *k* **folds**.
  - The most common convention is to use **10-fold cross-validation** (10-fold CV)
  - It generally results in a less biased model compare to other methods: it ensures that every observation from the original dataset has the chance of appearing in training and test set.

Cross Validation

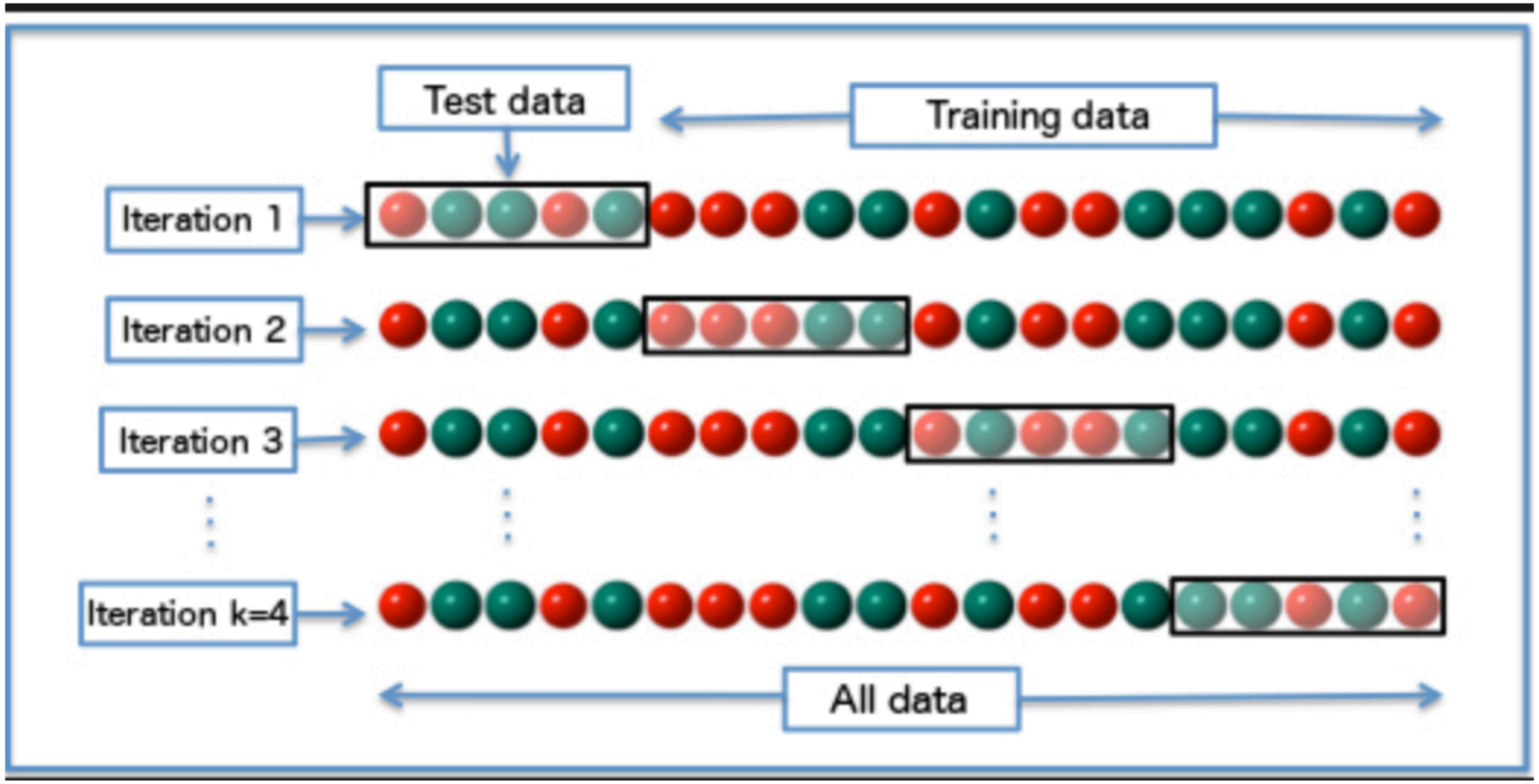


Image Sourced From Wikipedia

Cross Validation

- Cross validation steps
  - Split the entire data randomly into *k* folds.
  - Then fit the model using the *k*-1 folds and validate/test the model using the *k*th fold. Store the evaluation results.
  - Repeat this process until every fold serve as the test set. Then take the average of your recorded evaluation results.