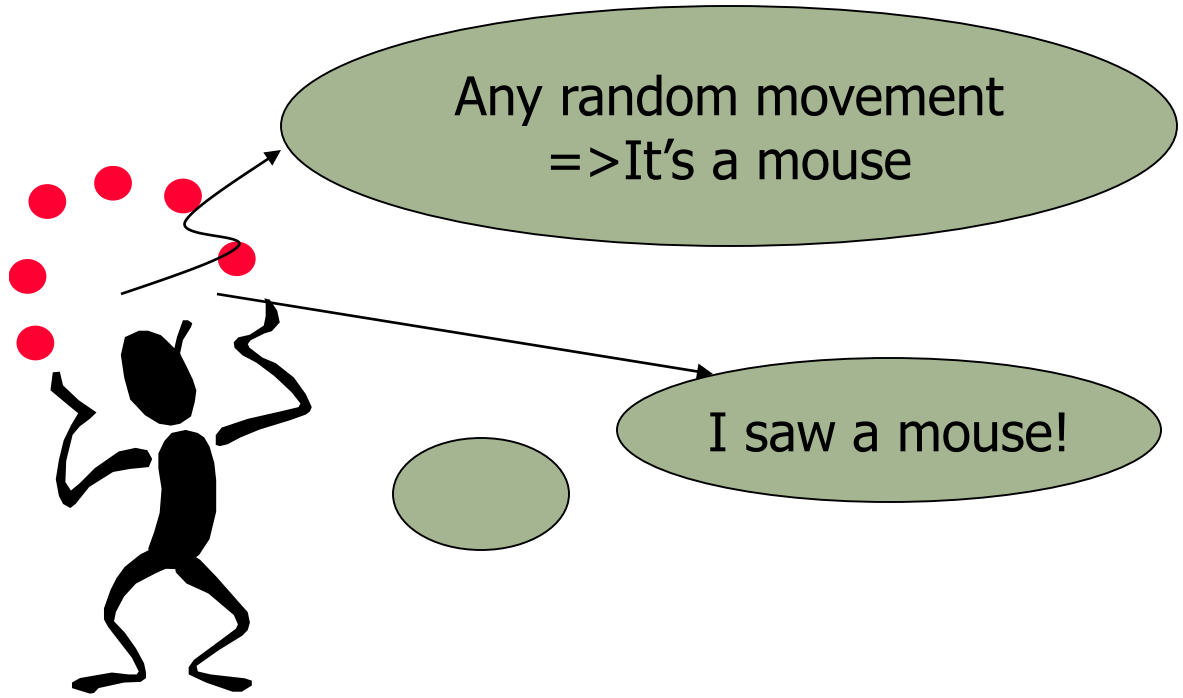
Lazy Learning

Module 2

K-Nearest Neighbor Learning

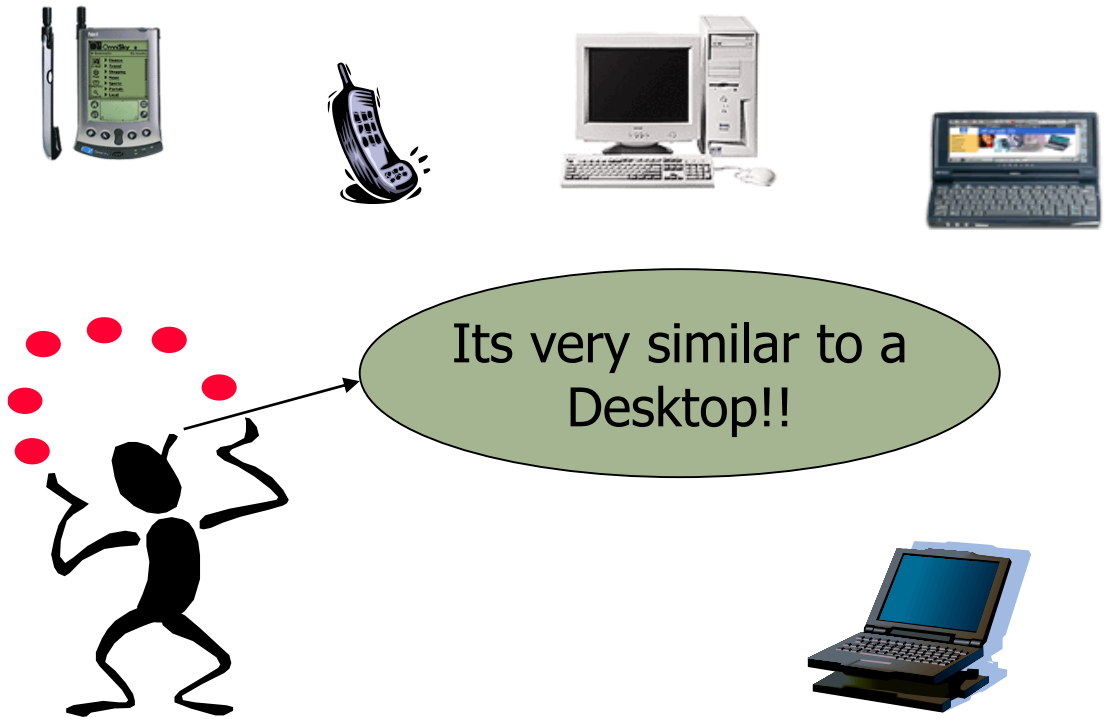
Different Learning Methods

- Eager Learning



Instance-based Learning

- Lazy Learning



Different Learning Methods

- Eager Learning
  - Explicit description of target function on the whole training set
- Instance-based Learning
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - Referred to as “Lazy” learning

Instance-Based Methods (Lazy Learners)

- Lazy learner waits until the last minute before doing any model construction in order to classify a given test tuple.
- That is, when given a training tuple, a lazy learner simply stores it and waits until it is given a test tuple.
- Only when it sees the test tuple, does it perform generalization in order to classify the tuple based on its similarity to the stored training tuples.
- Unlike eager learning methods, lazy learners do less work when a training tuple is presented and more work when making a classification or prediction.
- They are also referred to as instance based learners.

Instance-based (Lazy) Learning

Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified.

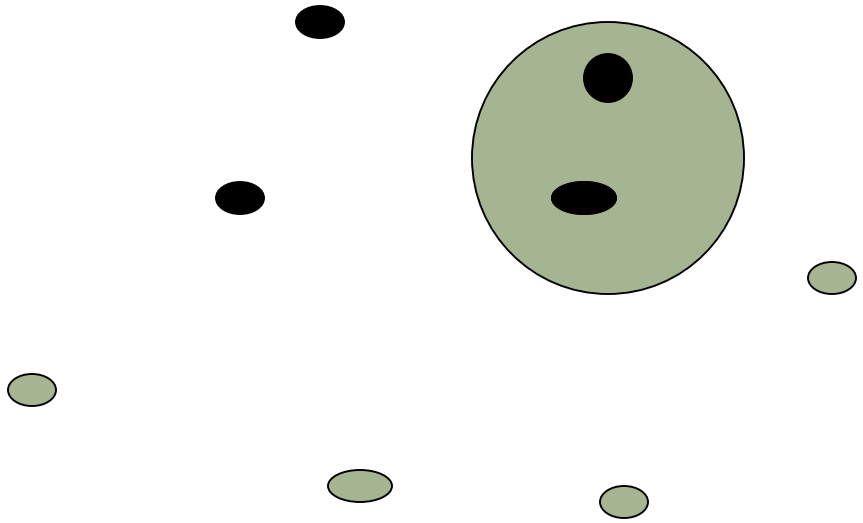
Typical approaches

- *k*-nearest neighbor approach
  - Instances represented as points in a Euclidean space.
- Locally weighted regression
  - Constructs local approximation
- Case-based reasoning
  - Uses symbolic representations and knowledge-based inference

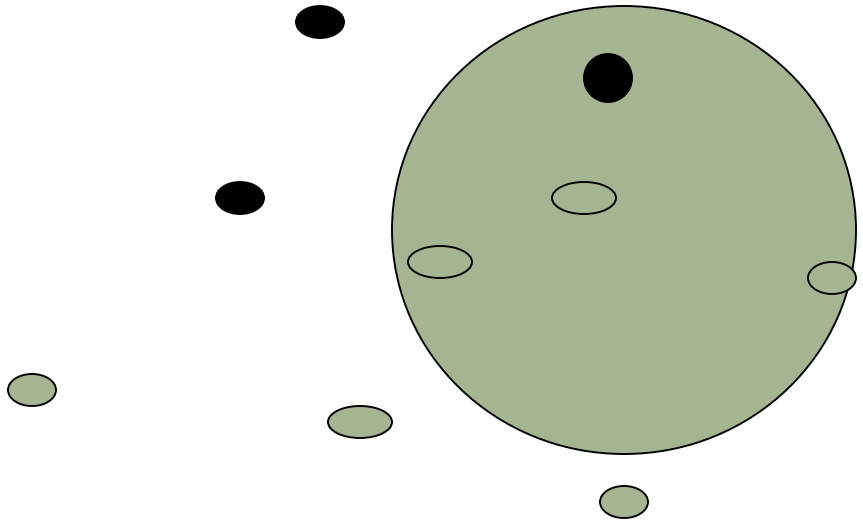
The *k*-Nearest Neighbor Algorithm

- Nearest-neighbor classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it.
- The training tuples are described by 'n' attributes.
- Each tuple represents a point in an n-dimensional space.
- In this way, all of the training tuples are stored in an n-dimensional pattern space.
- When given an unknown tuple, a **k-nearest-neighbor classifier searches the pattern space for the k training** tuples that are closest to the unknown tuple. These k training tuples are the k "nearest neighbors" of the unknown tuple.

1-Nearest Neighbor



3-Nearest Neighbor



K-Nearest Neighbor…

- "Closeness" is defined in terms of a distance metric, such as Euclidean distance.

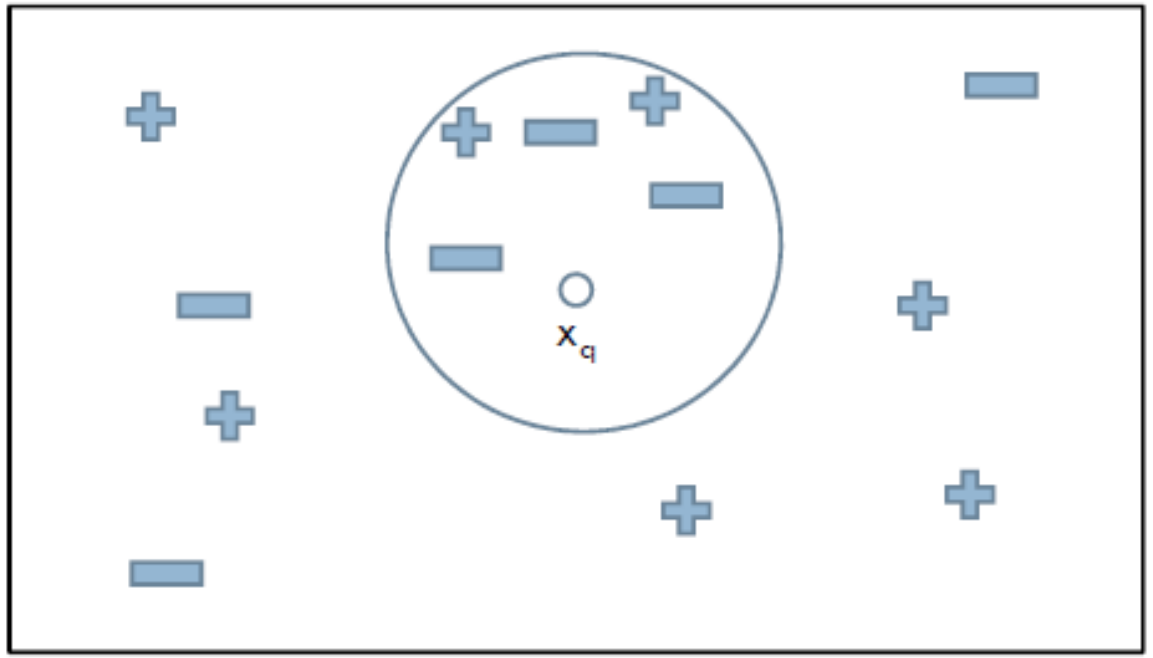
- The Euclidean distance between two points or tuples, say, $X_1 = (x_{11}, x_{12}, \ldots, x_{1n})$ and $X_2 = (x_{21}, x_{22}, \ldots, x_{2n})$, is

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}$$

K-Nearest Neighbor…

- **How determine a good value for 'k', the no. of neighburs?**
  - Starting with *k = 1, we use a test set to estimate the error rate* of the classifier. This process can be repeated each time by incrementing *k to allow for* one more neighbor. The *k value that gives the minimum error rate may be selected.*

K-Nearest Neighbor – By Example



If K = **5**, then in this case query instance x_q will be classified as negative since three of its nearest neighbors are classified as negative.

The letter k is a variable term implying that any number of nearest neighbors could be used.

The *k*-Nearest Neighbor Algorithm- Features

- All instances correspond to points in the n-D space
  - Classification is delayed till a new instance arrives
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real- valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to x_q.

K-Nearest Neighbor

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

KNN- Procedure

Step1- Determine the parameter, K. (Let K= 3)

Step2- Calculate the distance between the query Instance and training samples.

Step3- Sort the distance and determine nearest neighbours based on 'k'

Step4- Gather the class label of nearest neighbours

Step5- Use majority of the category of nearest neighbours as the prediction value of the query instance.

K-Nearest Neighbor – By Example

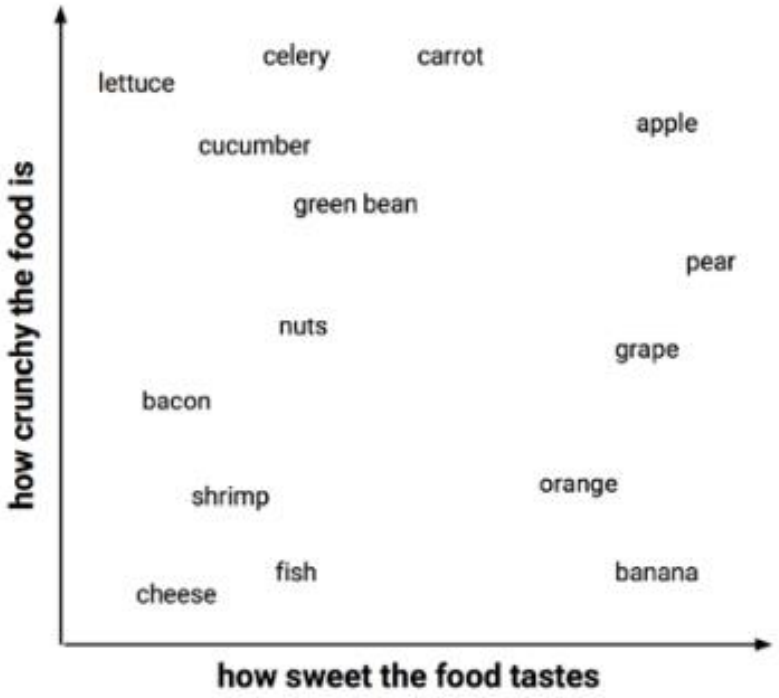
To illustrate this process, consider the following dataset:

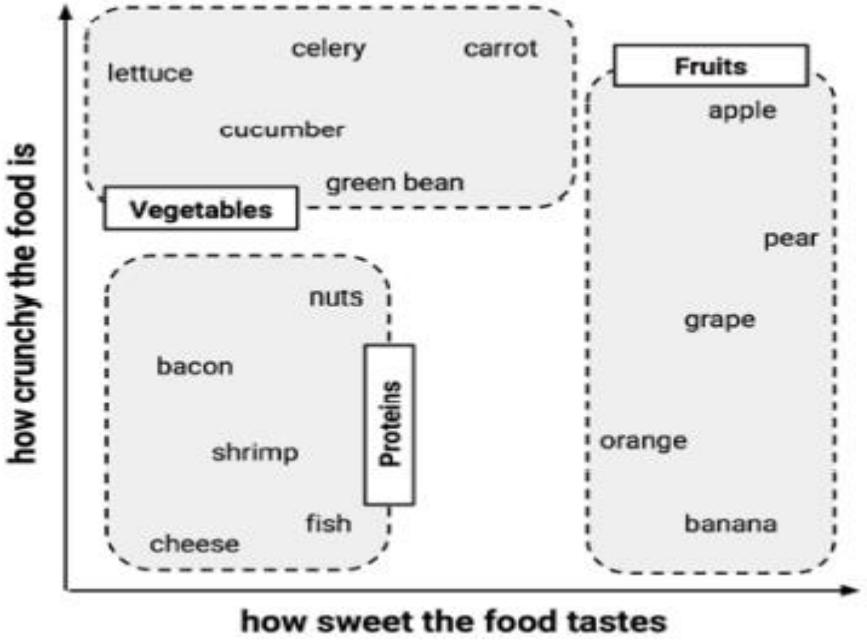
Ingredient	Sweetness	Crunchiness	Food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

Here we consider only two features of each ingredient- **Sweetness** & **Crunchiness**. The first is a 1 to 10 score of how sweet the ingredient tastes and the second is a measure from 1 to 10 of how crunchy the ingredient is . We then labeled each ingredient as one of the three types of food: **fruits**, **vegetables**,or **proteins**.

K-Nearest Neighbor – By Example….

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:

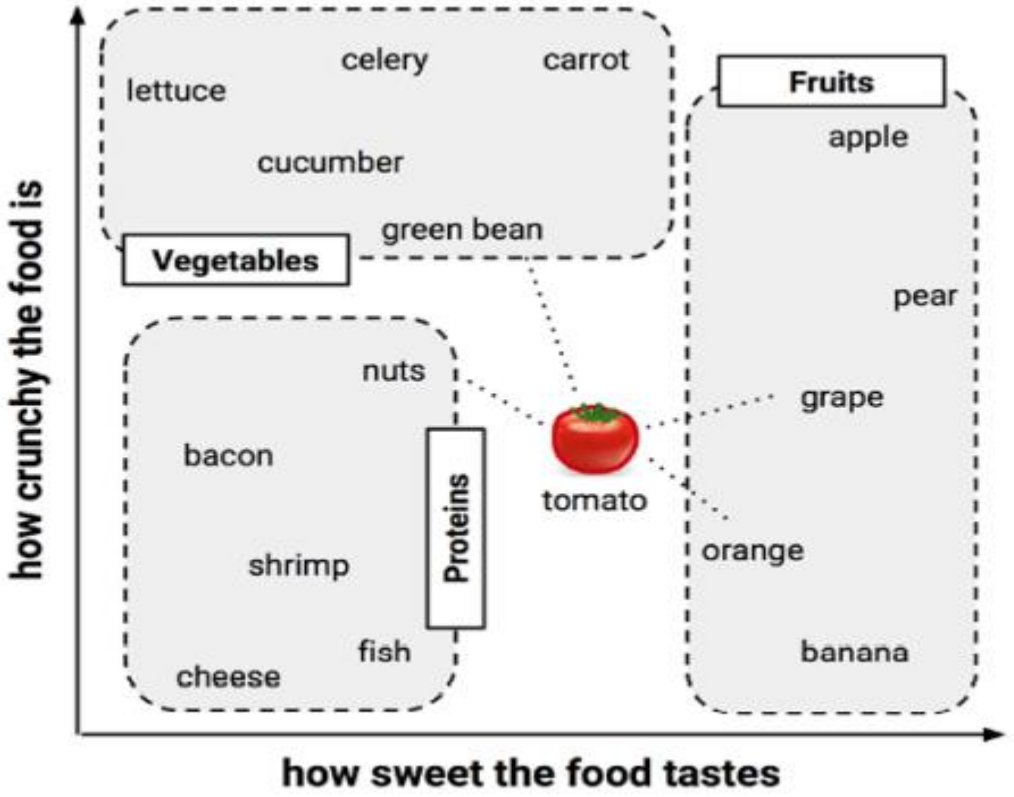




Similar types of food tend to be grouped closely together. As in the second diagram, vegetables tend to be crunchy but not sweet, fruits tend to be sweet and either crunchy or not crunchy, while proteins tend to be neither crunchy nor sweet

K-Nearest Neighbor – By Example….

Now the 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



K-Nearest Neighbor – By Example….

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.

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:

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

K-Nearest Neighbor – By Example….

Measuring similarity with distance

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

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:

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

In a similar way, 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*

K-Nearest Neighbor – By Example….

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*

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.**

Exercises

1. Based on the survey conducted in an institution the students are classified based on the 2 attributes- Academic excellence and other achievements. Consider the data set given. Find the classification of a student with value of X is 5 and Y is 7 based on the data of trained samples using KNN algorithm. Choose k = 3.

X [Academic Excellence]	Y [Activities]	Z [Classification]
8	6	Outstanding
5	6	Good
7	3	Good
6	9	Outstanding

Exercises

2. We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

X1 (Acid Durability)	X2 Strength	Y Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with X1 = 3 and X2 = 7. Without another expensive survey, can we guess what the classification of this new tissue is?

Exercises

3.

radius_mean	texture_mean	diagnosis
11.42	20.38	1
11.36	17.57	0
18.05	16.15	1
15.53	33.56	1
12.47	17.31	0

Use KNN to determine the diagnosis value of instance with radius_mean 12.3 and texture_mean 22.05