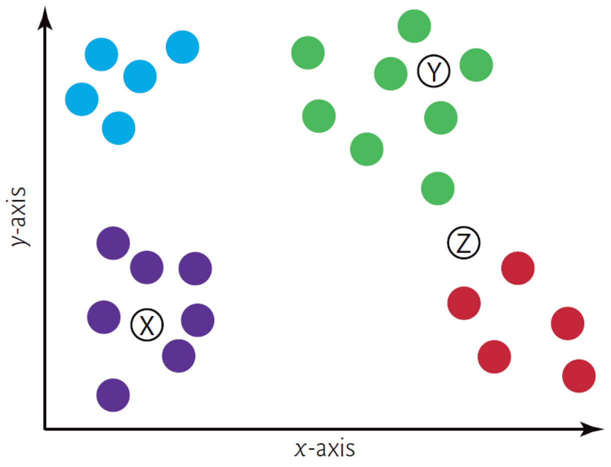
k-Nearest Neighbors

Classification is a **supervised machine learning**, which attempts to predict the distinct class to which a sample belongs. For example, if you have images of dogs and images of cats, you can classify each image as a "dog" or a "cat". This is a **binary classification problem** because there are two classes ("dog" or "cat").

The **k-nearest neighbors (k-NN)** algorithm is one of the **simplest** machine-learning classification algorithms. This algorithm attempts to predict a test sample's class by looking at the *k* training samples that are **nearest** (in distance) to the **test sample**. For example, consider the following diagram in which the **blue**, **purple**, **green** and **red** dots represent **four classes**.



We want to predict the classes to which the new samples **X**, **Y** and **Z** belong. Let's assume we'd like to make these predictions using each sample's **three nearest neighbors** ($k = 3$).

- The **X**'s three nearest neighbors are all **purple dots**, so we'd predict that X's class is purple.
- The **Y**'s three nearest neighbors are all **green dots**, so we'd predict that Y's class is green.
- For **Z**, the choice is **not as clear**, because it appears between the green and red dots.

  Of the three nearest neighbors, **one is green** and **two are red**. In the *k*-NN algorithm, **the class with the most "votes" wins**. So, we'd predict that **Z**'s class is **red**.

  Picking an **odd *k* value** in the kNN algorithm avoids ties by **ensuring there's never an equal number of votes**.

There are many distance metrics, but the **Euclidean distance** is the most commonly used metric. The formula for computing the Euclidean distance between two data points is

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Design a class named **BinaryKNN** that has the following data attributes:

- **neighbors** (a list of 2d data points)
- **k** (the number of the nearest neighbors)

The BinaryKNN class should have an **_ _init_ _** method that accepts the number of the nearest neighbors (k) as an argument. The value should be assigned to the object's **k** data attribute. It should also assign an empty list to the **neighbors** data attribute.

The class should also have the following methods:

- **fit**

  The fit method takes a list of 2d data points (*x*, *y*), and stores it in the **neighbors** list.
- **predict**

  The predict method calculates the distance between a given data point and all the neighbors, then determines the *k* nearest neighbors, and finally returns the class (0 or 1).
- **nearest_neighbors**

  The nearest_neighbors method should return the value of *k*.

Using the following dataset, classify the two instances (3, 7) and (6, 5) with $k = 3$.

x	*y*	Class
7	7	0
7	4	0
3	4	1
1	4	1

Optional Enhancement: create a module named **classifier** and put your code inside the module.