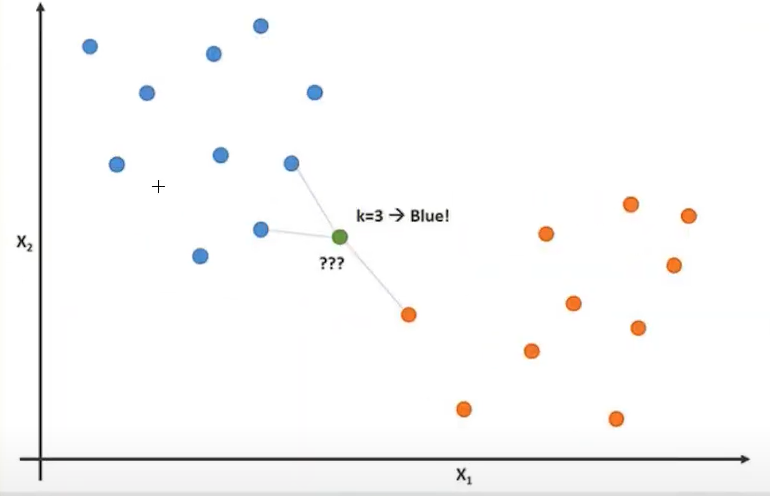
**K Nearest Neighbors and implementation on Iris dataset**

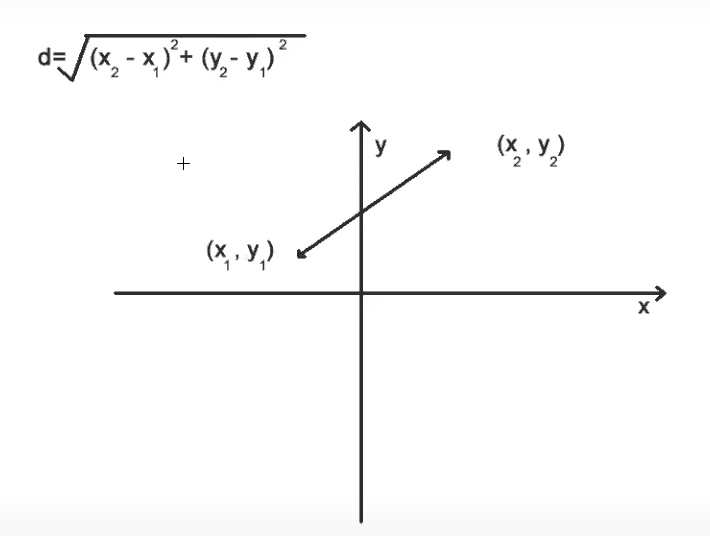
**1.kNN and how it works**

Let’s consider an example with two classes and two-dimensional feature vector.



We have here two classes blue one and orange one and two-feature vectors with two-dimensions (x1 and x2). Also, there are some training samples and for **each new sample** (green one) that we want to classify. Then, we want to calculate the distance of green sample to each of the training samples and then we look at the nearest neighbors. (In this case we will look at the three nearest neighbors (k=3) and then we’ll choose or predict the label, based on the most common class labels (so we have on picture above two blue classes and one orange class).

In order to calculate the distances, we used the Euclidean distance. In 2D example:



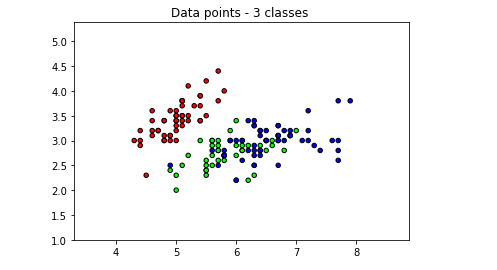
In ED in 2D case of 2 points is defined as the square root over and we have for each feature vector 0-component, square difference so we have x2-x1 + y2-y1 squared.

**2. Visualization of DATASET:**

Let’s look at our dataset and figure out what is X and y. We will use Iris dataset from scikit-learn module.

We will get some train samples and test samples and the associated training labels and test labels.

Graph with 3 classes (red, green and blue).



**3. Let’s to CODE:**

We have to create a class KNN and then methods that we want to implement:

First, we have to implement **Init method** with self and k=3 (which is a number of nearest neighbors). Inside the init we want to store the k (self.k = k).

Next, we have to implement the conventions of other machine learning libraries (ex. Scikit-learn library).

Another implementation is the **fit method** that will fit the training samples and training labels.

And last, we want to implement also **predict method**. Here, we want to predict new samples.

In our Fit method, kNN algorithm does not involve training steps, basically we want to store our training samples.

In our predict method, we will get multiple samples using ( X ). We will write helper method = so we want to do this for each of the samples. We will make list of the predicted labels = []. (list comprehension) and then we want to call this self.\_predict with one sample(x)for all of our samples in our tests samples . And then convert the list to NumPy array.

We have to create a **helper method \_\_predict** with only one sample (x).

The \_predict method needs to be computed. We will calculate all of distances and look at the nearest neighbors and the labels of the nearest neighbors and then we ‘ll choose our most common class label (1).

Then in our **test file** we will create classifier and k=3 neighbors and then to try 5 neighbors to experiment. We will fit X\_train and y\_train and predict the test samples. And then we will calculate the accuracy ( how many of our predictions are correctly classified ) using accuracy method.