K Nearest Neighbours

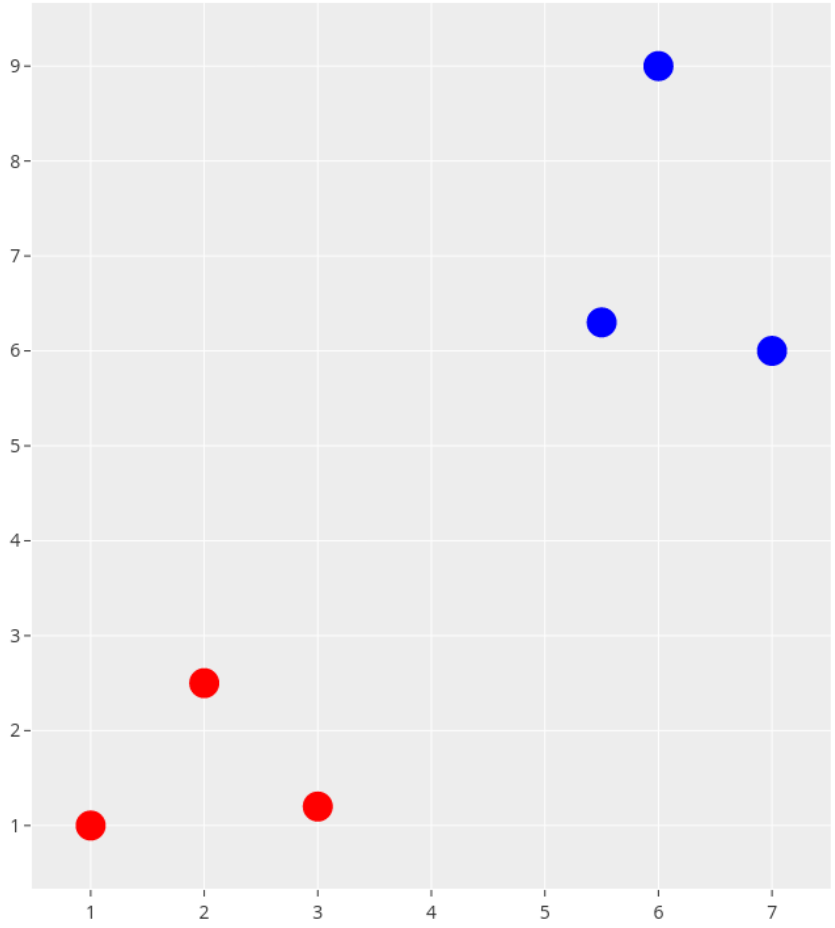
Non-parametric models – Lazy Learning

Lazy Learning is a method by which all the datapoints in the dataset are stored, so that when a query is to be made, the dataset can be retrieved and used to make predictions based on the input query.

E.g. k-NN, Lazy Naïve Bayes

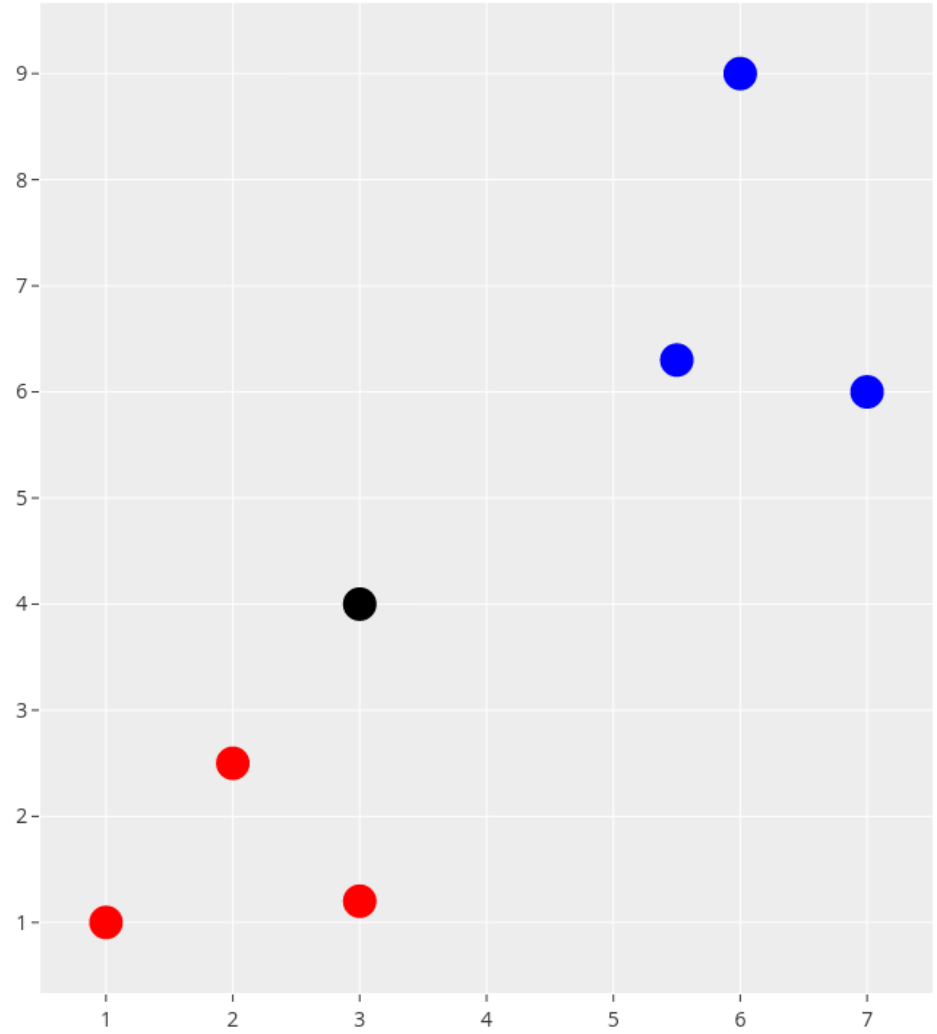
Assume the following toy dataset

X	Y	Category
1	1	Red
2	2.5	Red
3	1.2	Red
5.5	6.3	Blue
6	9	Blue
7	6	Blue



Now let's add a new point

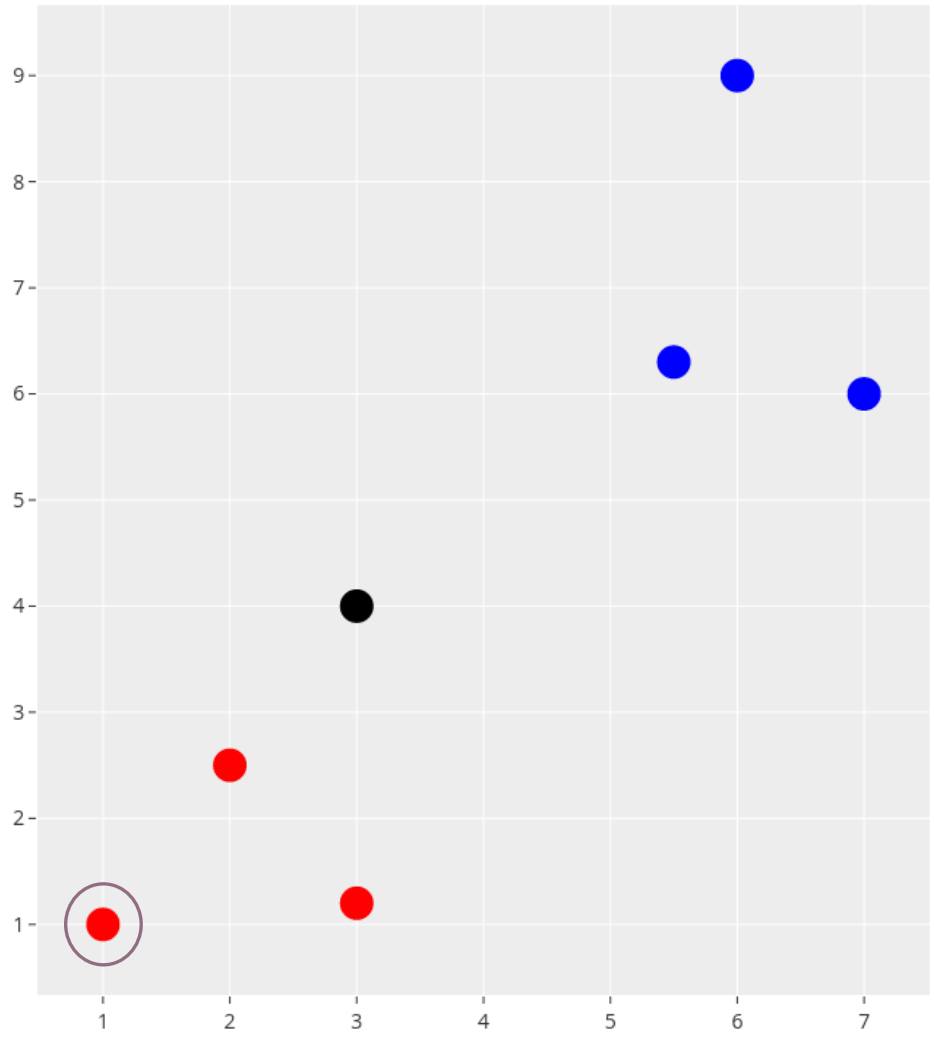
Let the new point be k = [3, 4] (colored in black)



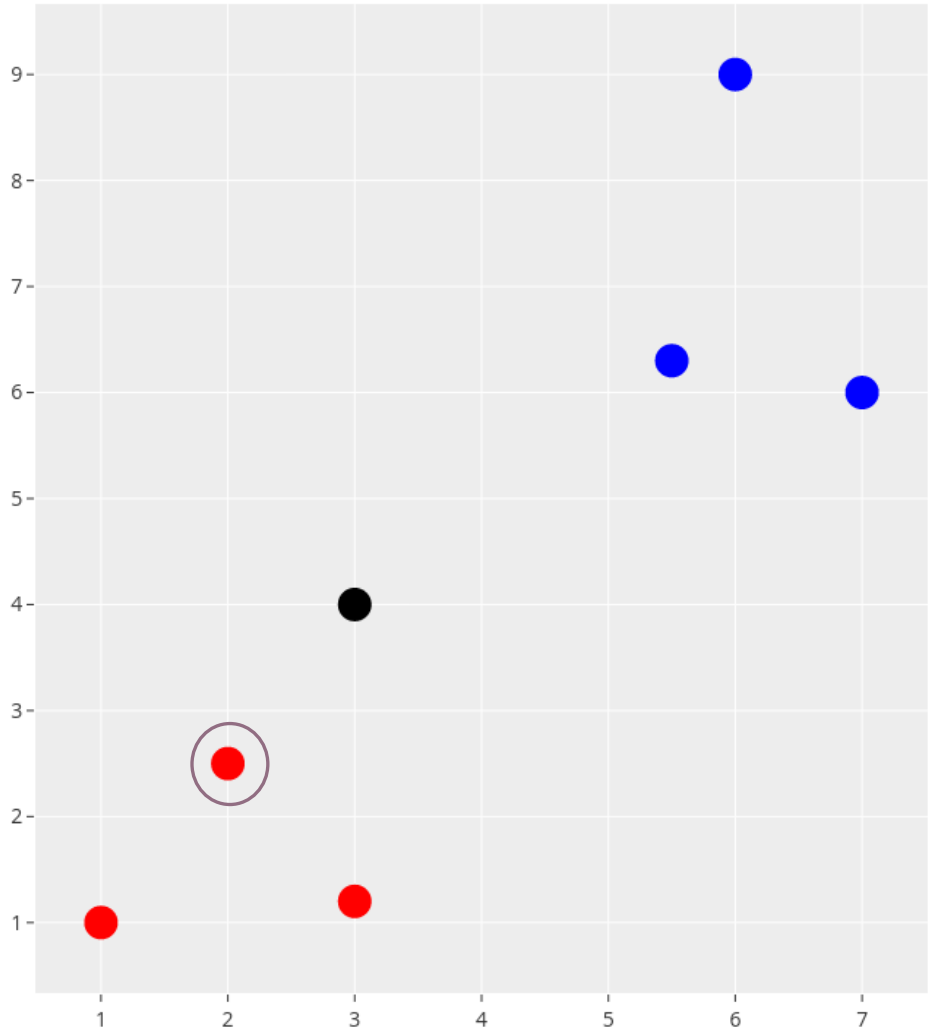
The Euclidean Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

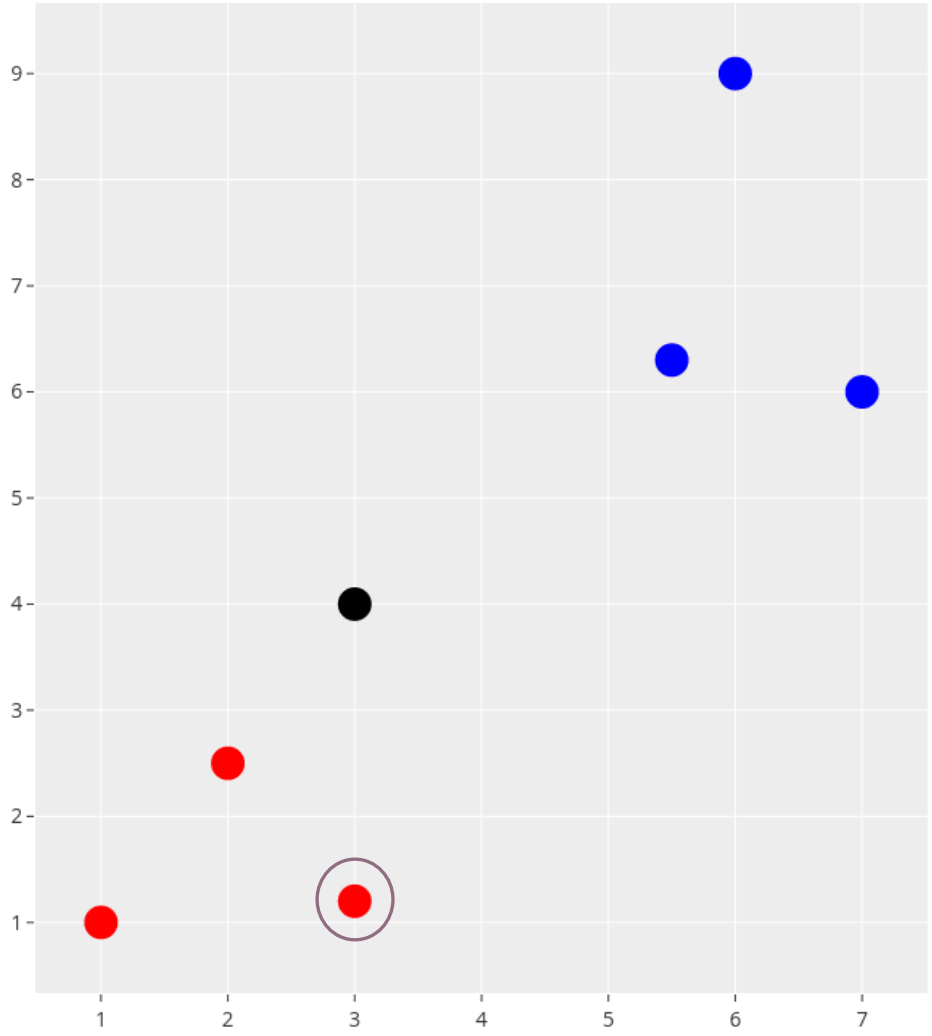
X	Y	Category	Distance between point and k
1	1	Red	$\text{Sqrt}((3 - 1)^2 + (4 - 1)^2) = 3.6$
2	2.5	Red	
3	1.2	Red	
5.5	6.3	Blue	
6	9	Blue	
7	6	Blue	
3	4	?	



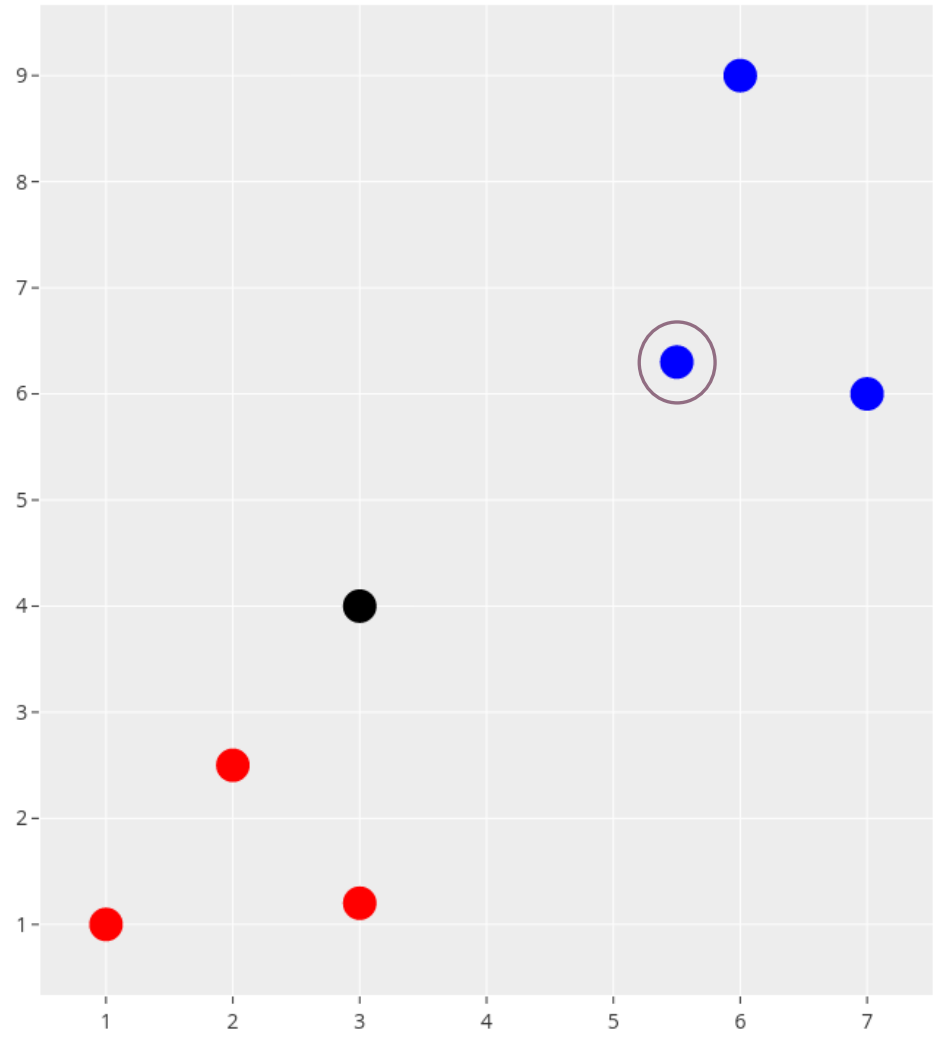
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	$\text{Sqrt}((3 - 2)^2 + (4 - 2.5)^2) = 1.8$
3	1.2	Red	
5.5	6.3	Blue	
6	9	Blue	
7	6	Blue	
3	4	?	



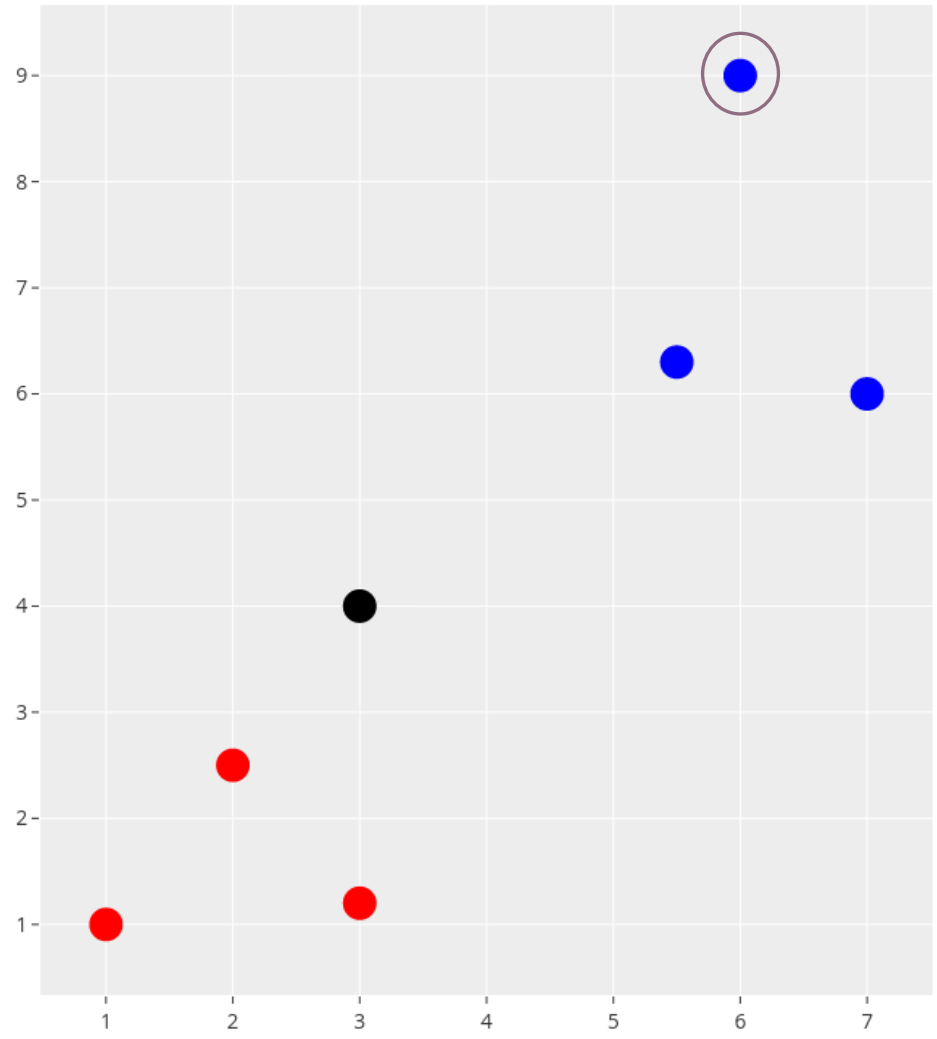
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	1.8
3	1.2	Red	$\text{Sqrt}((3 - 3)^2 + (4 - 1.2)^2) = 2.8$
5.5	6.3	Blue	
6	9	Blue	
7	6	Blue	
3	4	?	



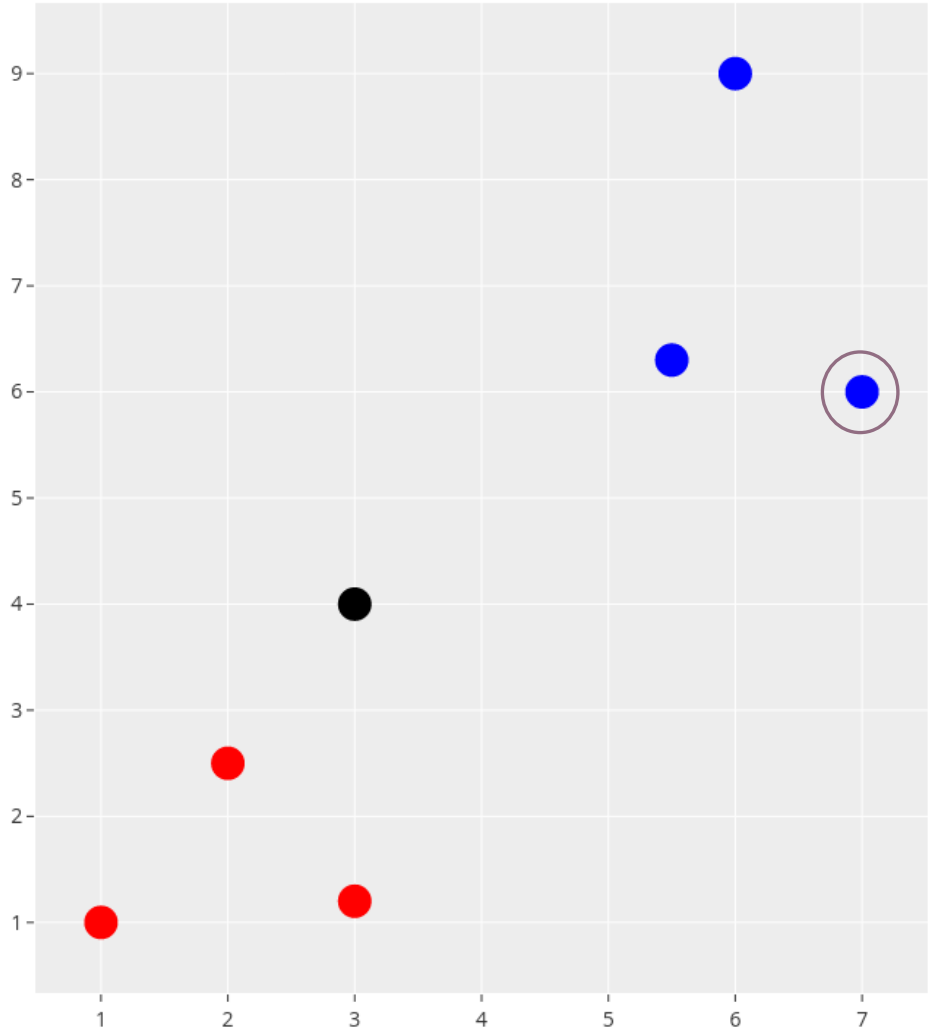
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	1.8
3	1.2	Red	2.8
5.5	6.3	Blue	$\text{Sqrt}((3 - 5.5)^2 + (4 - 6.3)^2) = 3.4$
6	9	Blue	
7	6	Blue	
3	4	?	



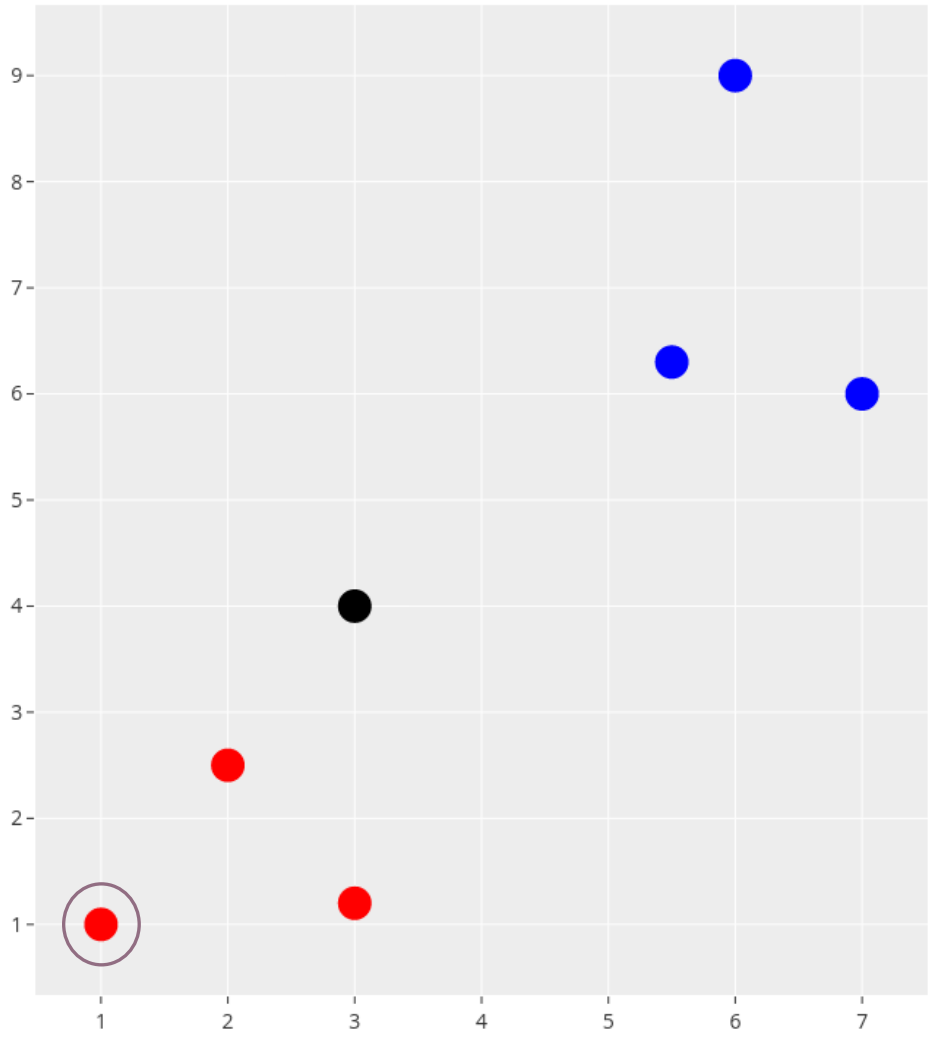
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	1.8
3	1.2	Red	2.8
5.5	6.3	Blue	3.4
6	9	Blue	$\text{Sqrt}((3 - 6)^2 + (4 - 9)^2) = 5.8$
7	6	Blue	
3	4	?	



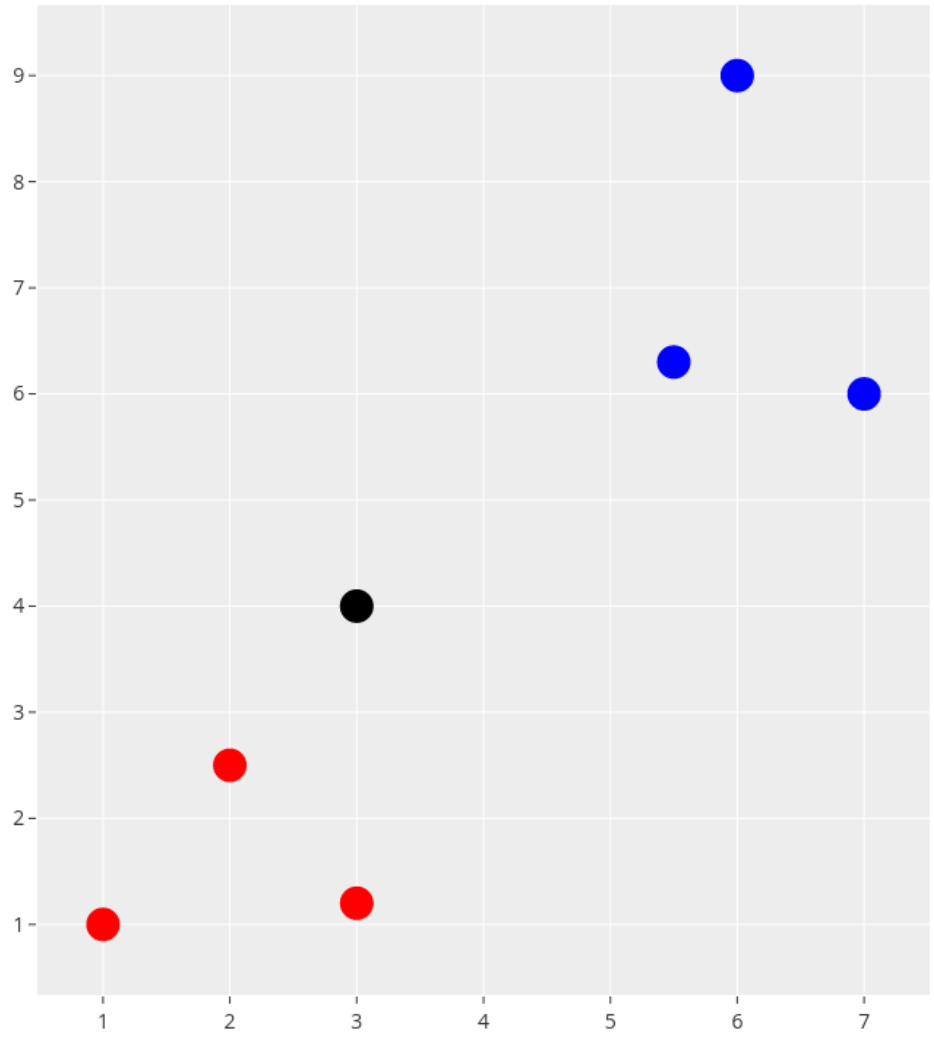
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	1.8
3	1.2	Red	2.8
5.5	6.3	Blue	3.4
6	9	Blue	5.8
7	6	Blue	$\text{Sqrt}((3 - 7)^2 + (4 - 6)^2) = 4.5$
3	4	?	



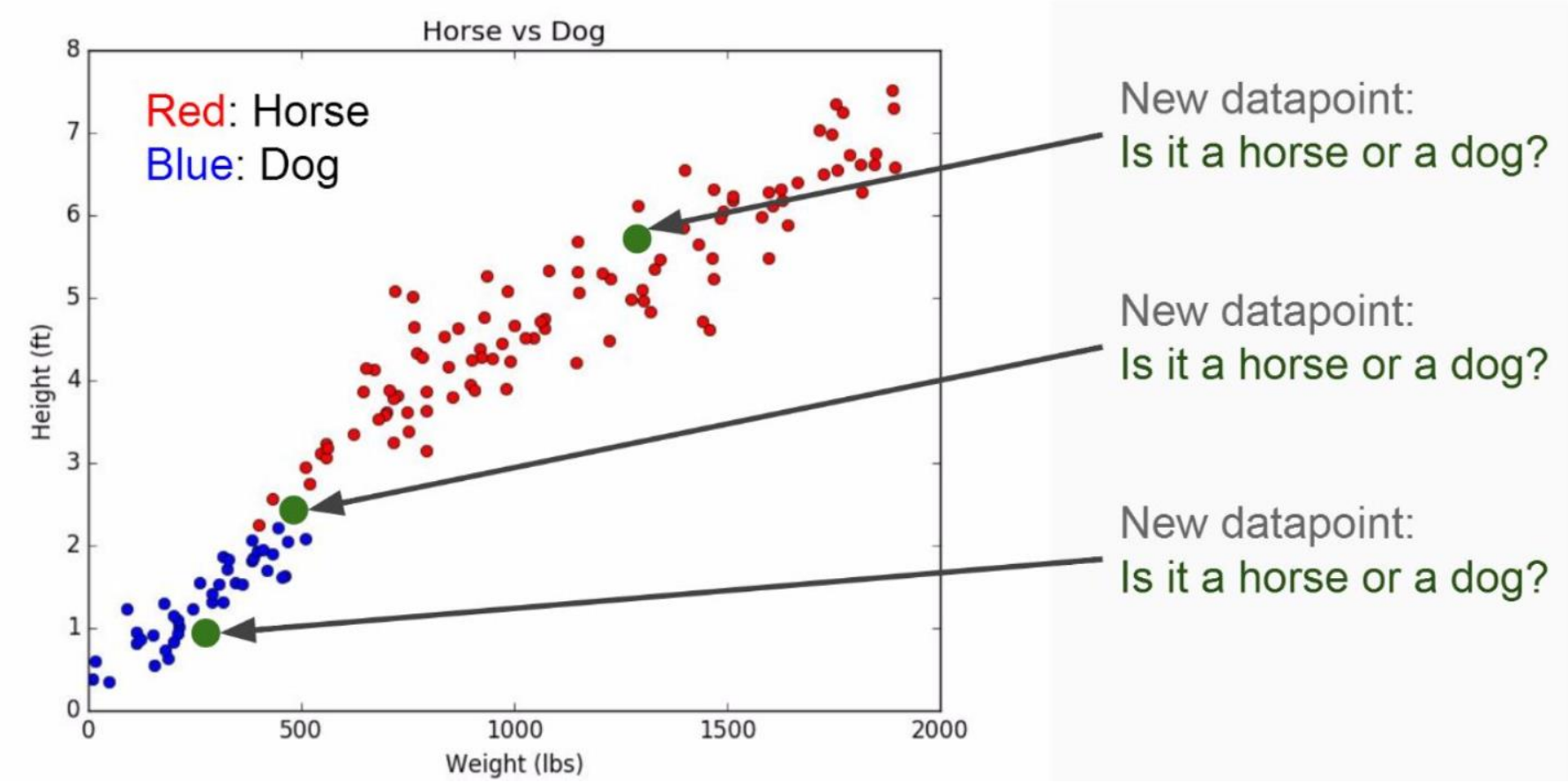
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	1.8
3	1.2	Red	2.8
5.5	6.3	Blue	3.4
6	9	Blue	5.8
7	6	Blue	4.5
3	4	?	



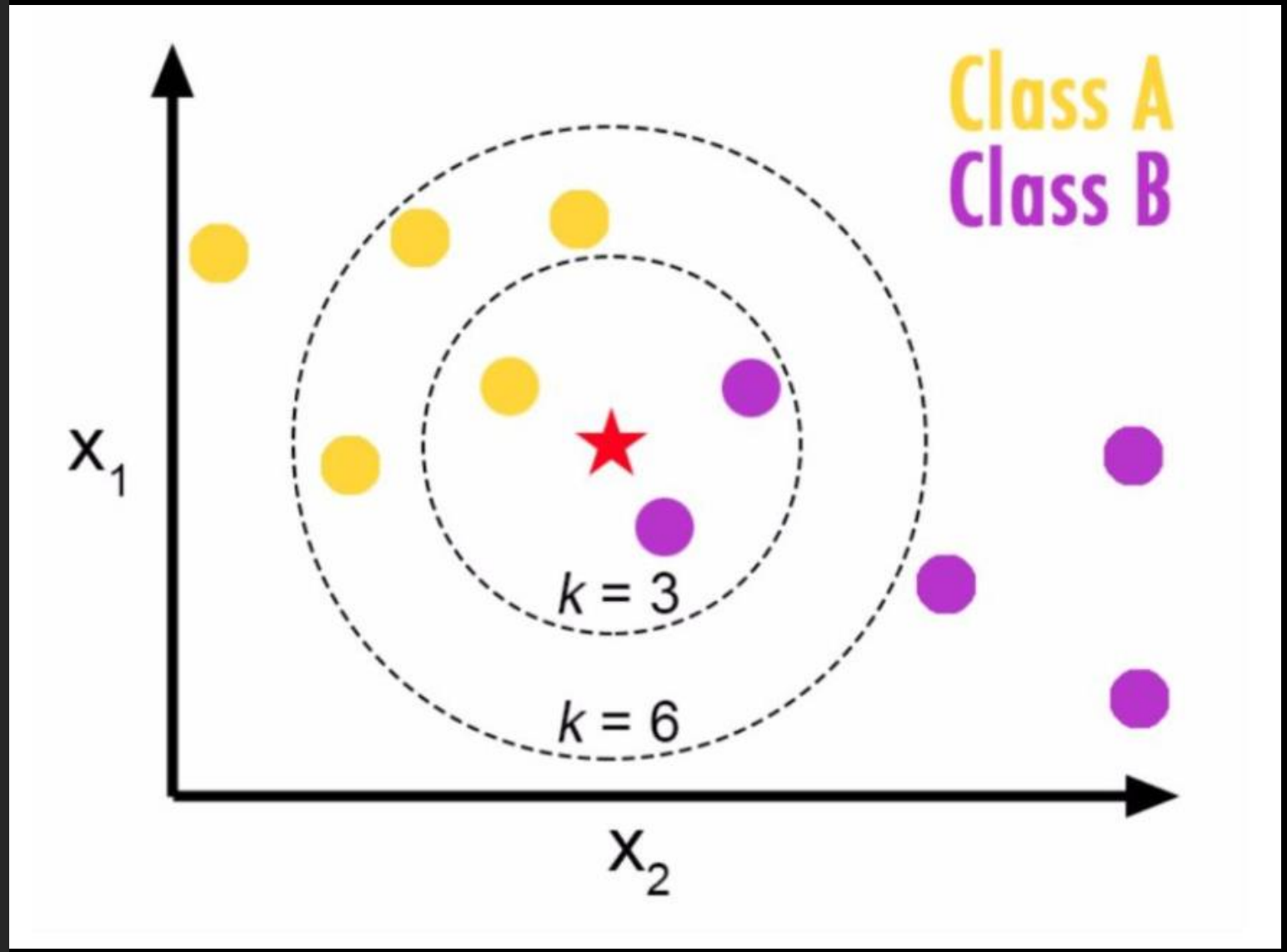
X	Y	Category	Distance between point and k
1	1	Red	3.6
2	2.5	Red	1.8
3	1.2	Red	2.8
5.5	6.3	Blue	3.4
6	9	Blue	5.8
7	6	Blue	4.5
3	4	?	



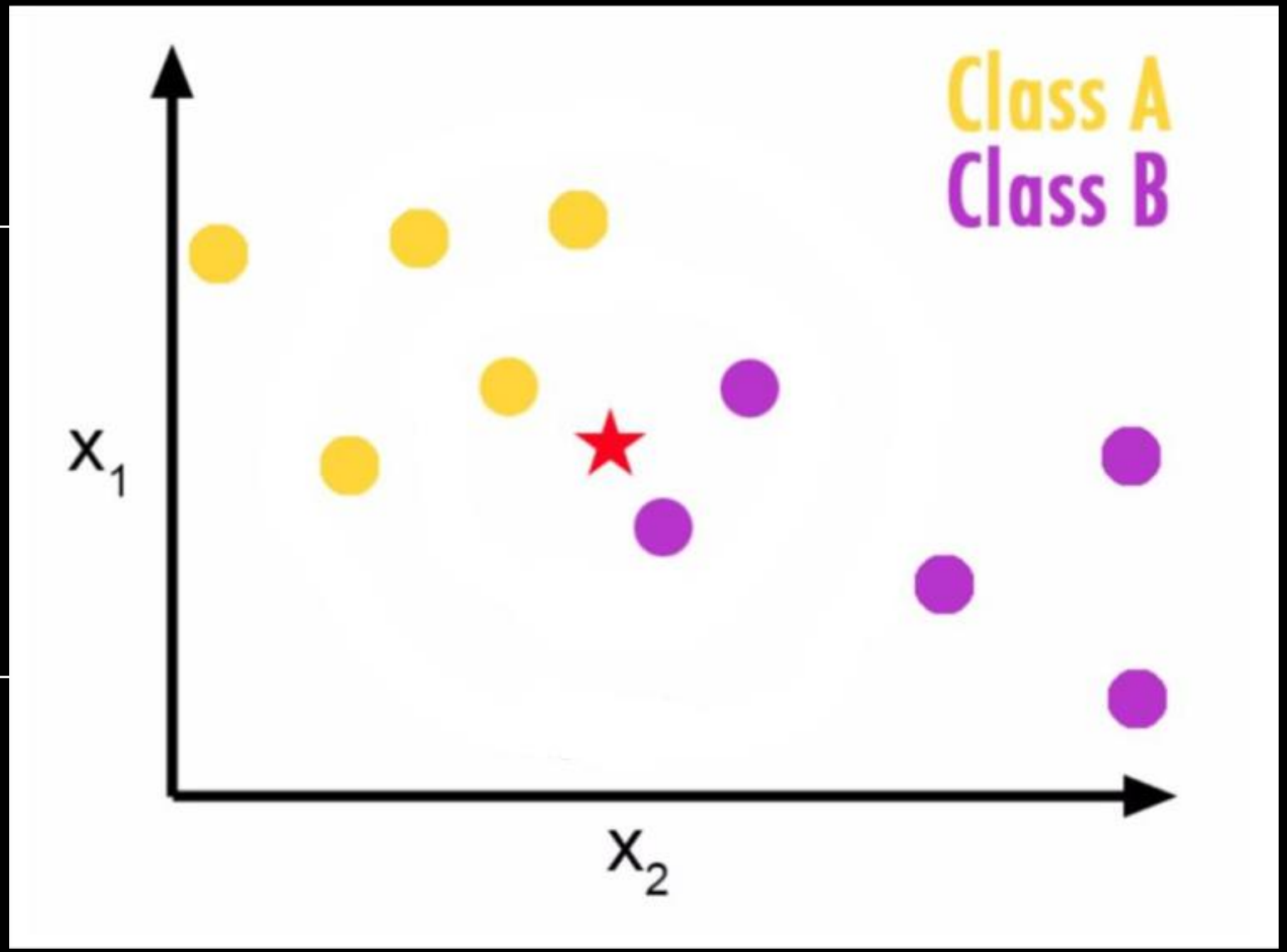
A Real example?



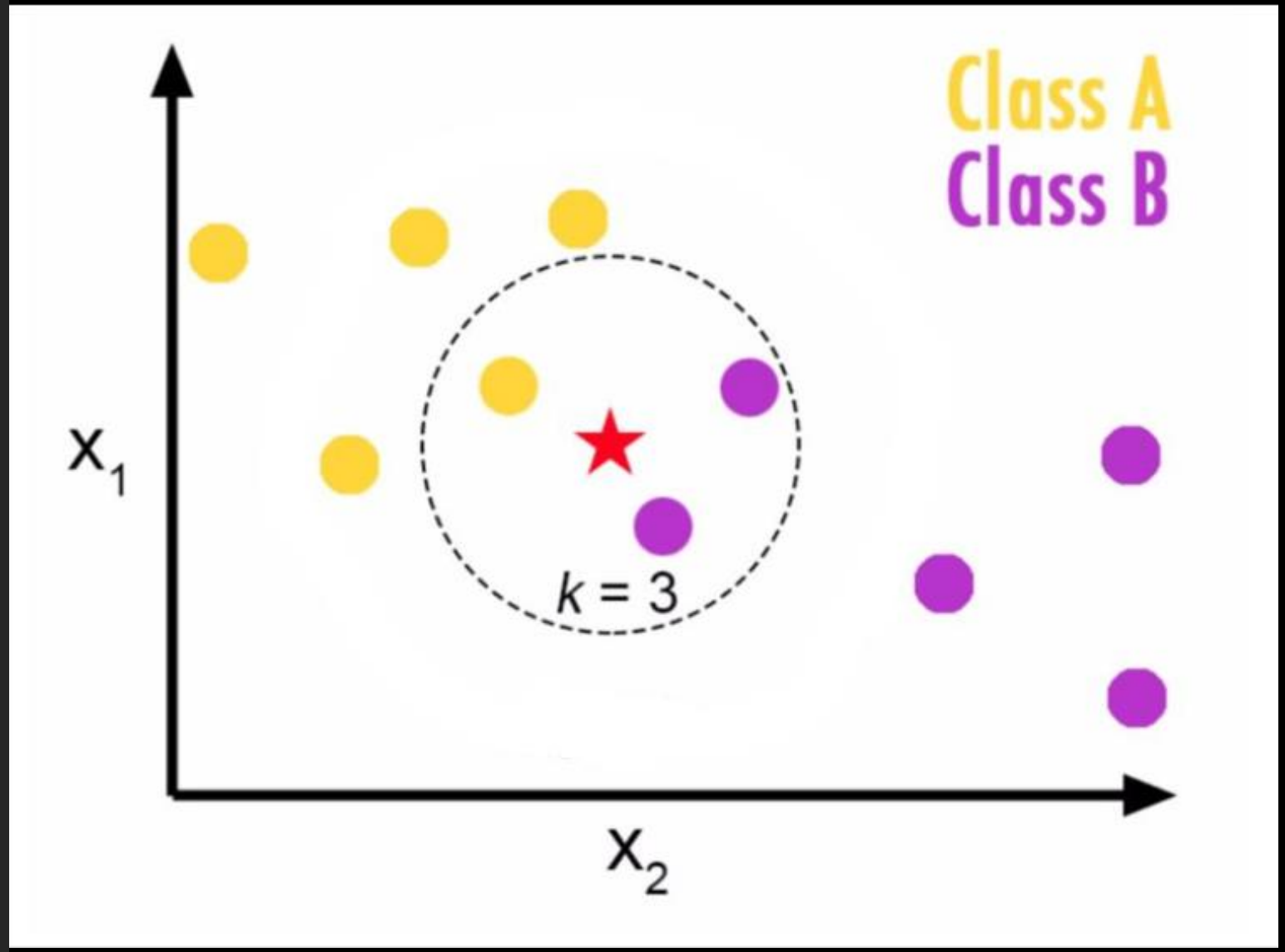
k-NN – Majority Vote



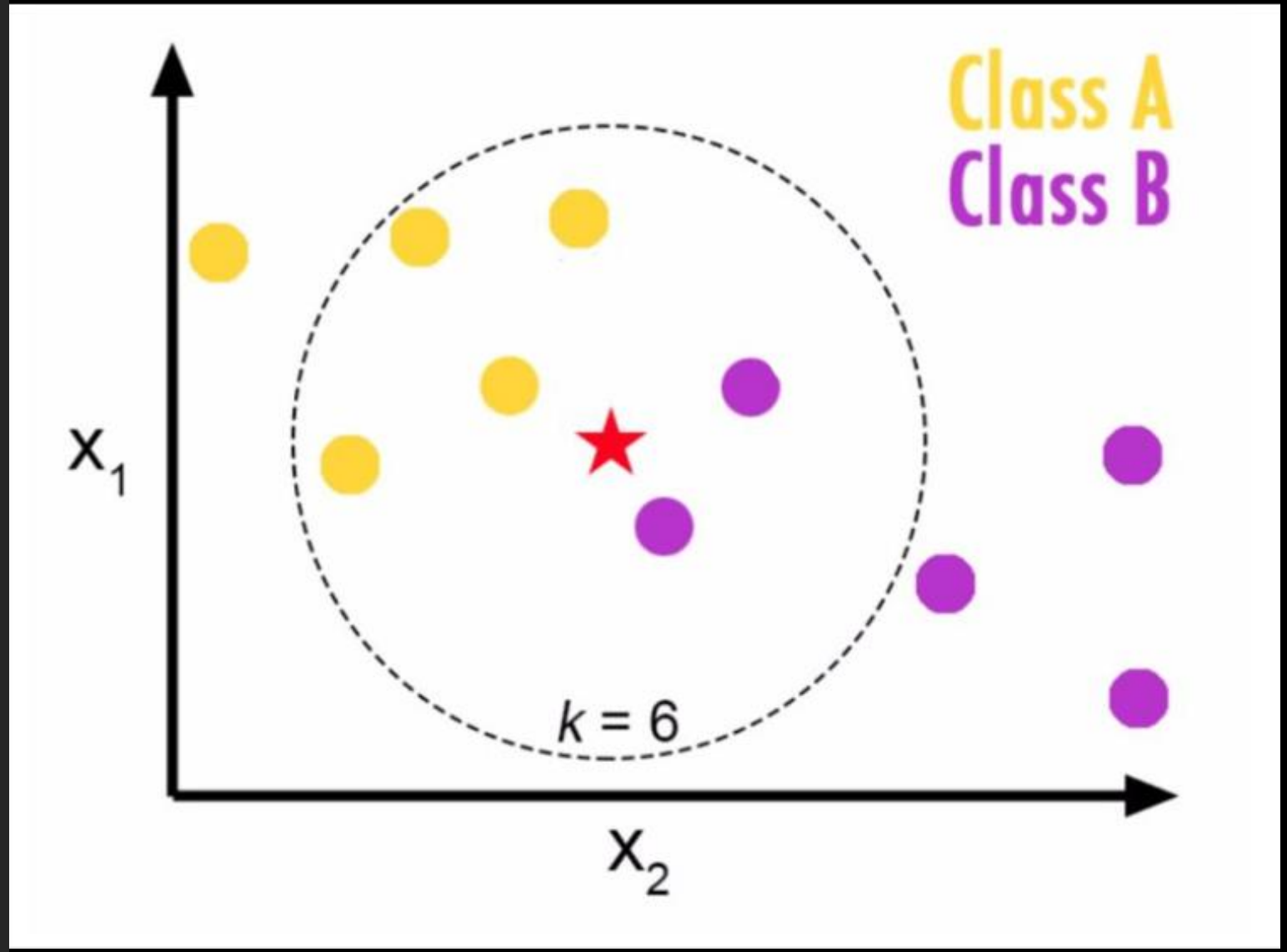
k-NN



k-NN

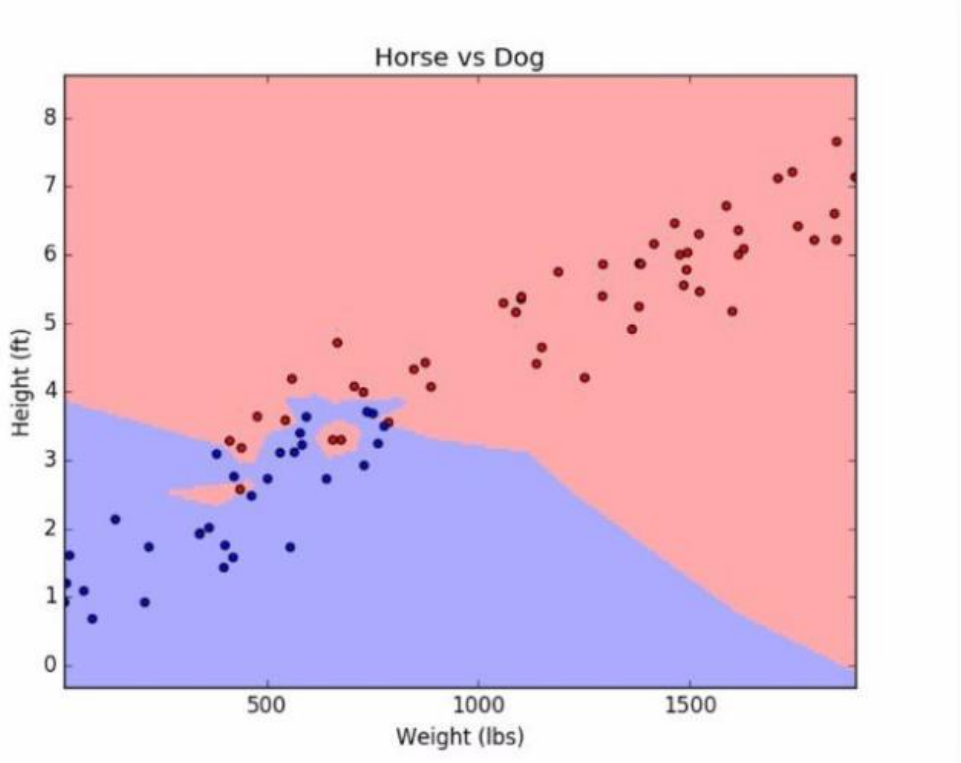


k-NN

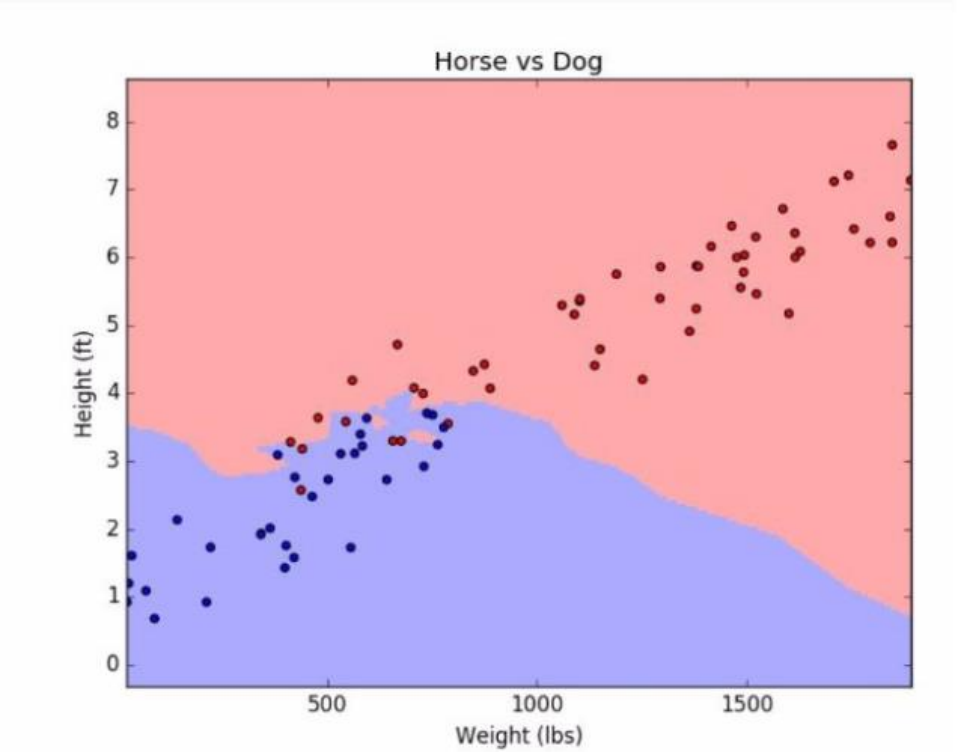


k-NN – How K changes the output

K = 1

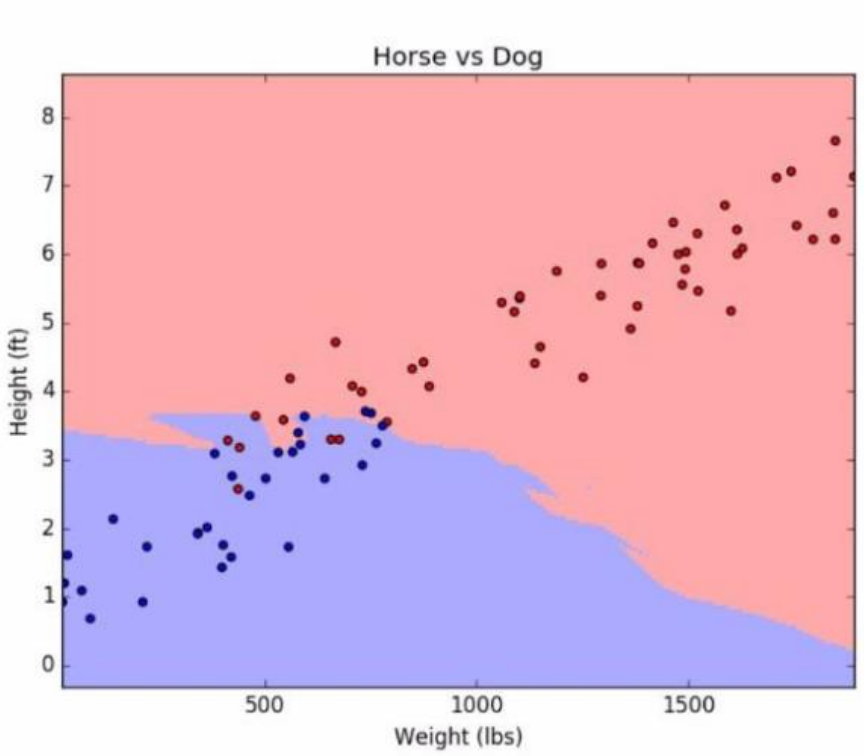


K = 5

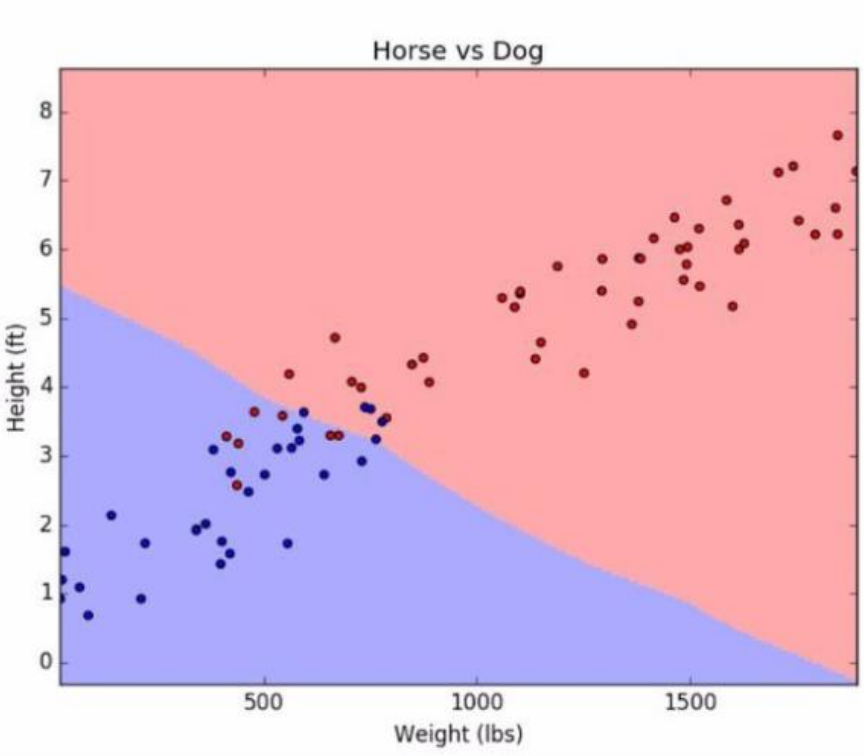


k-NN – How K changes the output

K = 10



K = 50



Representing data as vectors

Assume the following dataset:

Sepal Length	Sepal Width	Flower species
5.1	3.5	Iris-setosa
4.9	3	Iris-setosa
4.7	3.2	Iris-setosa
4.6	3.1	Iris-setosa
7	3.2	Iris-versicolor
6.4	3.2	Iris-versicolor
6.9	3.1	Iris-versicolor
5.5	2.3	Iris-versicolor

$$\begin{bmatrix} 5.1 \\ 3.5 \end{bmatrix} \quad \begin{bmatrix} 4.9 \\ 3.0 \end{bmatrix} \quad \begin{bmatrix} 4.7 \\ 3.2 \end{bmatrix} \quad \begin{bmatrix} 4.6 \\ 3.2 \end{bmatrix}$$

$$\begin{bmatrix} 7.0 \\ 3.2 \end{bmatrix} \quad \begin{bmatrix} 6.4 \\ 3.2 \end{bmatrix} \quad \begin{bmatrix} 6.9 \\ 3.1 \end{bmatrix} \quad \begin{bmatrix} 5.5 \\ 2.3 \end{bmatrix}$$

Each element of a vector represents a feature of the data point. For eg, in the above vector, the first element in the vector represents Sepal Length and the second element represents Sepal Width.

Representing data as vectors

Now that we have understood the vector representation:

$$\begin{bmatrix} 5.1 \\ 3.5 \end{bmatrix} \quad \begin{bmatrix} 4.9 \\ 3.0 \end{bmatrix} \quad \begin{bmatrix} 4.7 \\ 3.2 \end{bmatrix} \quad \begin{bmatrix} 4.6 \\ 3.2 \end{bmatrix}$$

$$\begin{bmatrix} 7.0 \\ 3.2 \end{bmatrix} \quad \begin{bmatrix} 6.4 \\ 3.2 \end{bmatrix} \quad \begin{bmatrix} 6.9 \\ 3.1 \end{bmatrix} \quad \begin{bmatrix} 5.5 \\ 2.3 \end{bmatrix}$$

To compute the different between two datapoints, we can simply use the vector difference and norm. For eg:

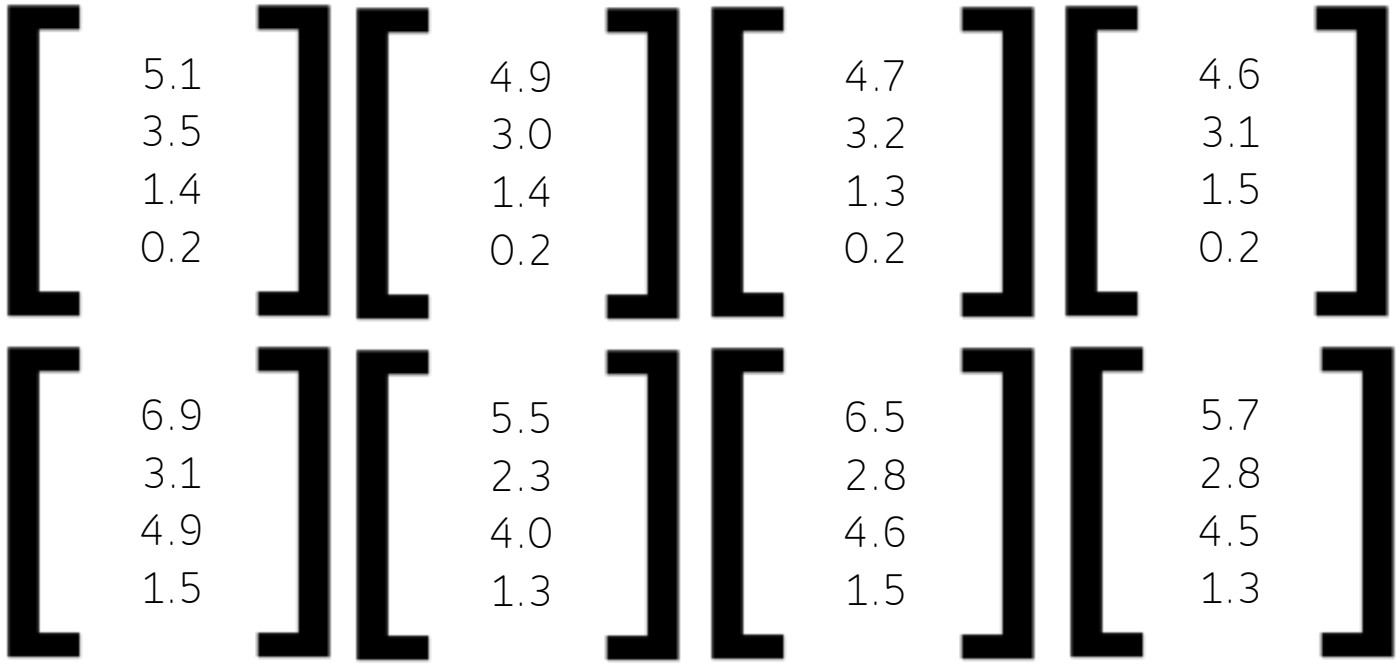
$$\begin{bmatrix} 5.1 \\ 3.5 \end{bmatrix} - \begin{bmatrix} 4.9 \\ 3.0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix}$$

$$\text{L2 Norm} = (0.2)^2 + (0.5)^2 = 0.29$$

Why is this notation important?

A lot of times, each data point is represented by more than 2 or 3 features.
The vector notation allows us to represent data. Let's take the Iris dataset itself.
The Iris dataset provides 4 features: Sepal Length, Sepal Width, Petal Length, Petal Width

Sepal Length	Sepal Width	Petal Length	Petal Width	Flower species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor



Why is this notation important?

Sepal Length	Sepal Width	Petal Length	Petal Width	Flower species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor

$$\begin{bmatrix} 5.1 \\ 3.5 \\ 1.4 \\ 0.2 \end{bmatrix} - \begin{bmatrix} 4.9 \\ 3.0 \\ 1.4 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.5 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{L2 Norm} = (0.2)^2 + (0.5)^2 + 0^2 + 0^2 = 0.29$$

Validation sets and Hyper-parameters

What is the best k value for the k-NN model?

Which distance is the best distance metric for k-NN?

How do we approximate the model's performance on real world data, i.e, performance of the model on data that isn't present in the dataset?

These questions form the basis of most real-time problems faced by ML Engineers.

Hyperparameters

The hypermeters are the knobs that we, the Machine Learning Engineers, have to use to tune the accuracy of the model.

However, we aren't completely on our own. There are some heuristics that we can use to understand better which knobs to use.

But to first choose the right hyperparameter, right value for k, we must find a way to evaluate the model on data that isn't used to make new queries.

Train-Validation/Dev sets

It is impossible to exactly get the kind of data that we'll be feeding the Machine Learning model or querying the model.

A better way to approximate what the real time data might look like is to first shuffle the dataset we have and then randomly split a percentage of dataset that you use for querying/training and set aside the rest of the dataset for validation purposes.

So, what now?

We first split the dataset like this.

Then, we first develop the model based on the train set.

The Dev/Validation set also has the input and their correct ground truth values. So, we take the inputs of the Dev set, get the predictions from the model, compare them with the ground truth values for the corresponding input and predicted value and check how many times our model got the output right.

Let's look at an example for how to calculate the accuracy of the model on the Dev set.

We first split the dataset like this.

Input Vector	Predicted class	Ground Truth Class
[5.7, 4.4]	Iris-Setosa	Iris-Setosa
[5.1, 3.8]	Iris-Versicolor	Iris-Setosa
[6.1, 2.8]	Iris-Setosa	Iris-Setosa
[4.9, 3.1]	Iris-Versicolor	Iris-Setosa
[6.7, 3.1]	Iris-Versicolor	Iris-Versicolor
[5.8, 2.6]	Iris-Setosa	Iris-Veriscolor
[4.9, 2.4]	Iris-Versicolor	Iris-Versicolor
[6.7, 3.0]	Iris-Versicolor	Iris-Versicolor

Total datapoints: 8
Correctly Predicted: 6
Predicted wrongly: 2
Accuracy = (6/8) * 100 = 75%

Hyperparameter Tuning for k-NN

1. Set a value for k = 1
2. Set the training data as query data for the k-NN Model
3. Get the test accuracy by using the validation/dev set data's inputs and comparing the predicted outputs with the real outputs
4. Note down the test accuracy for the given k
5. Increment the value of k by 1
6. Go back to step 2 until we check until k = n (n is decided by the data scientist)
7. Take the value of k that gave the model with the highest test accuracy.

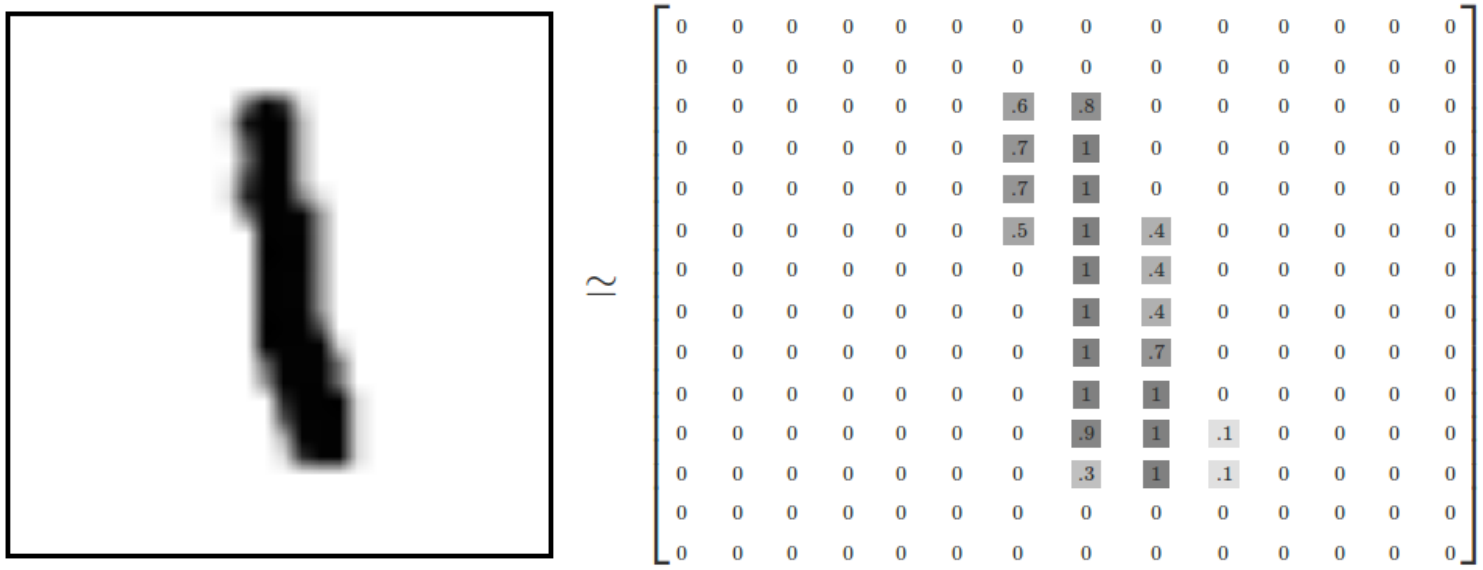
Image Classification with K-NN

The MNIST Dataset

The data

Each image contains a digit and is tagged properly.

Each image is a matrix



Each image is a data point with 768 dimensions

Comparing two images