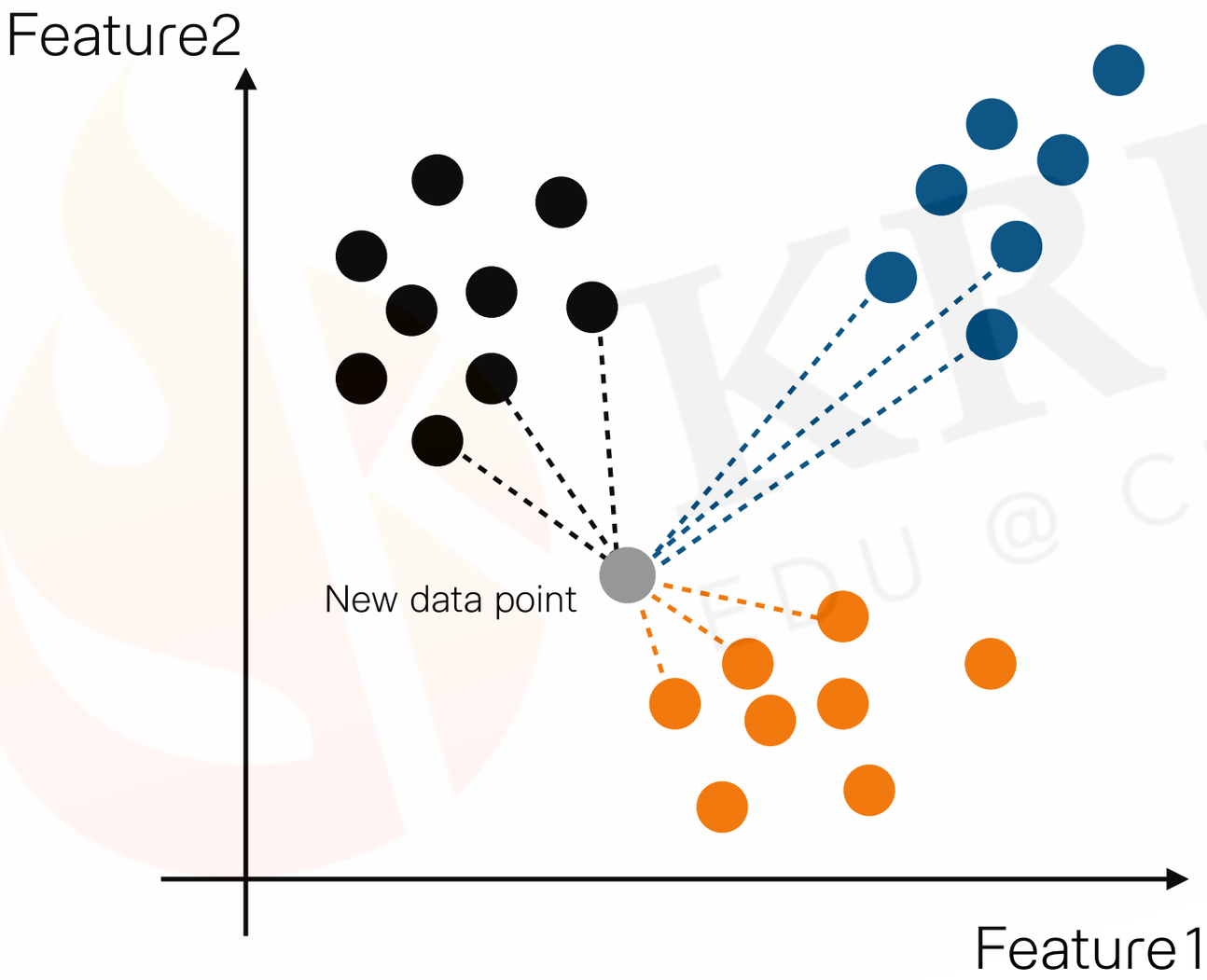
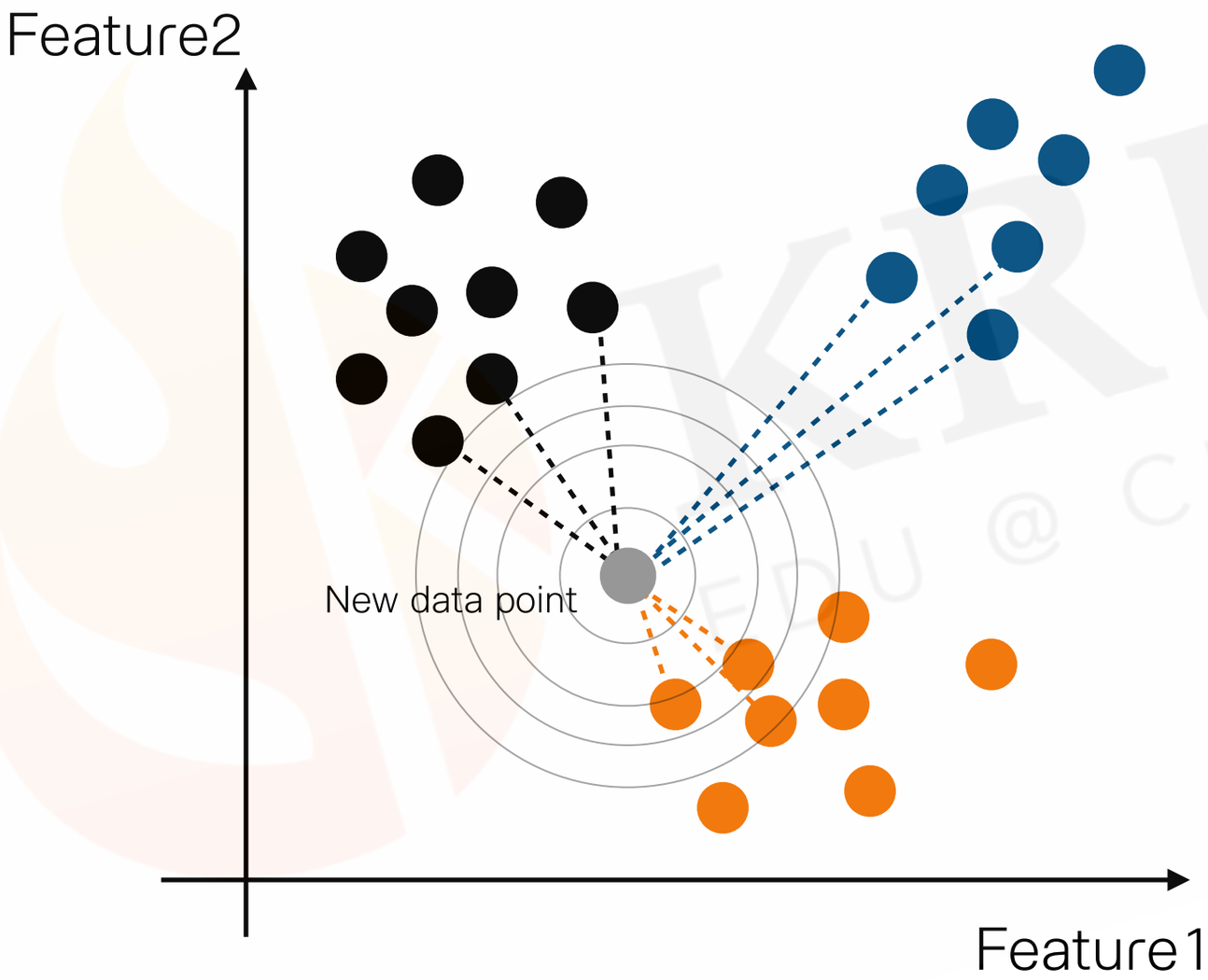
K-NN: Basic Concept

k nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. This algorithms classifies unlabeled data points into well defined groups.



K-NN: Basic Concept

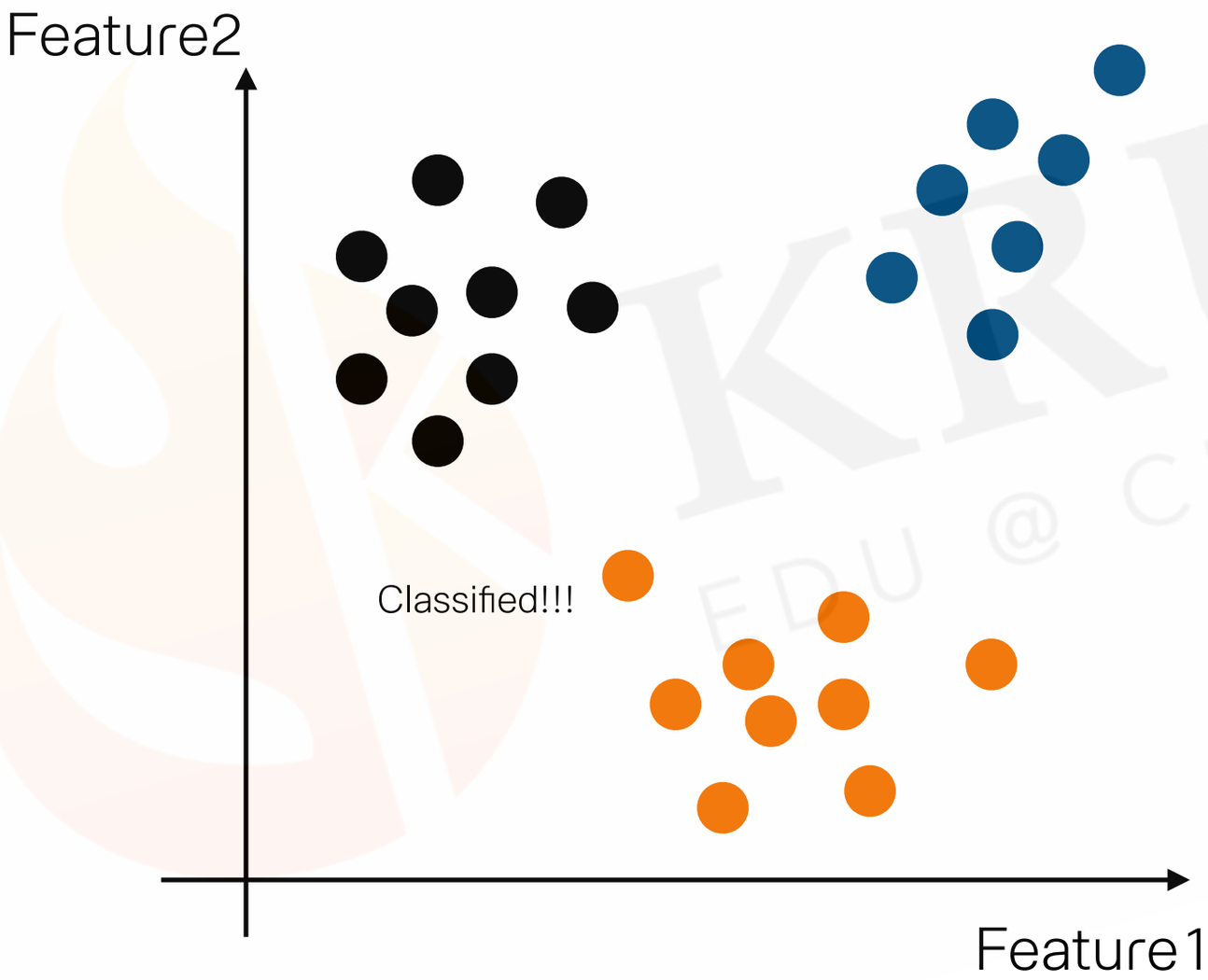
- Choose the number K of neighbours
- Take the K nearest neighbours of the new data point, according to the Euclidean distance (or something else)
- Among these K neighbours, count the number of data points in each category.
- Assign the new data point to the category where you counted the most neighbours.



Number of nearest neighbourhood	
●	0
●	0
●	3

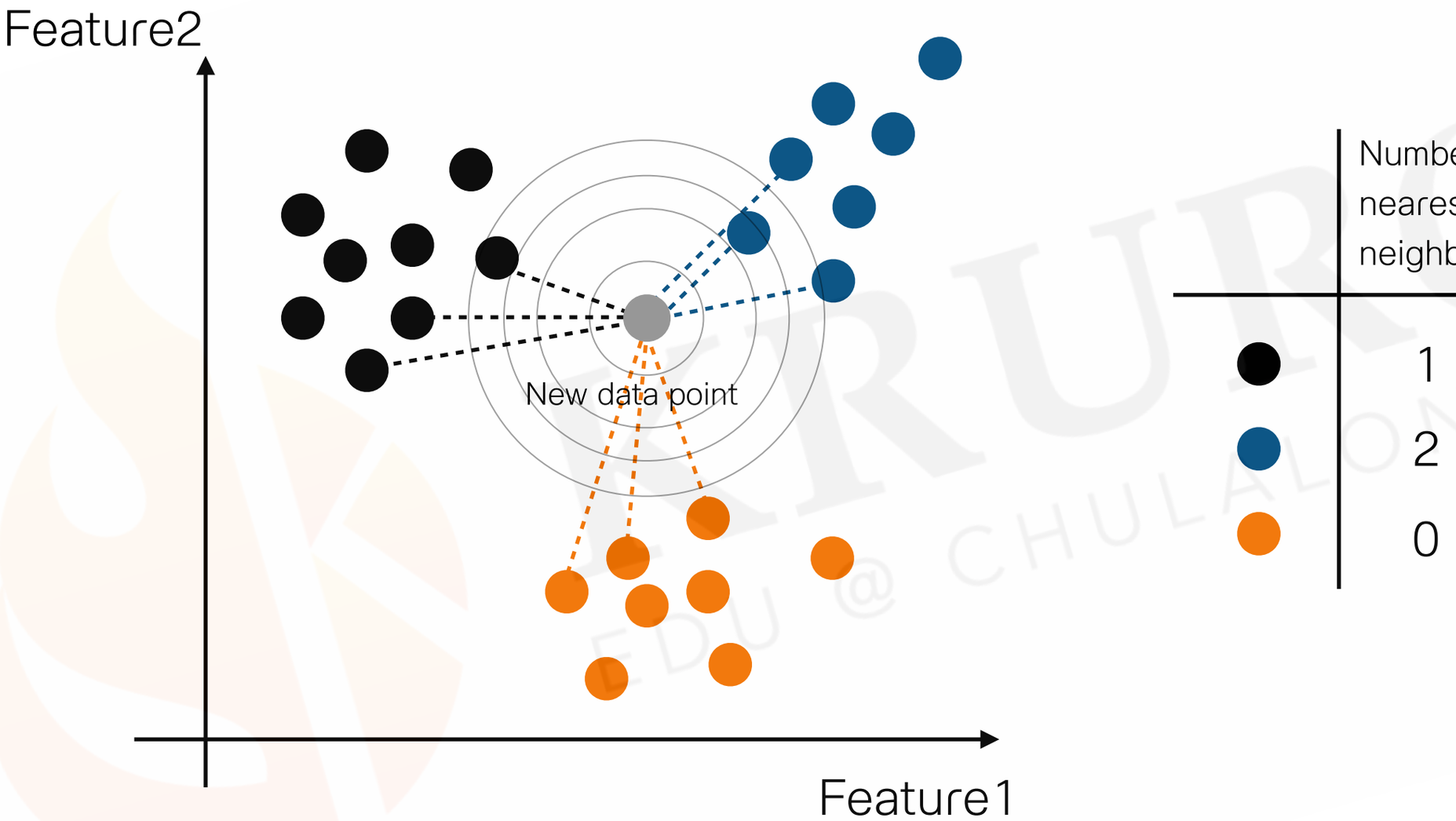
K-NN: Basic Concept

- Choose the number K of neighbours
- Take the K nearest neighbours of the new data point, according to the Euclidean distance (or something else)
- Among these K neighbours, count the number of data points in each category.
- Assign the new data point to the category where you counted the most neighbours.

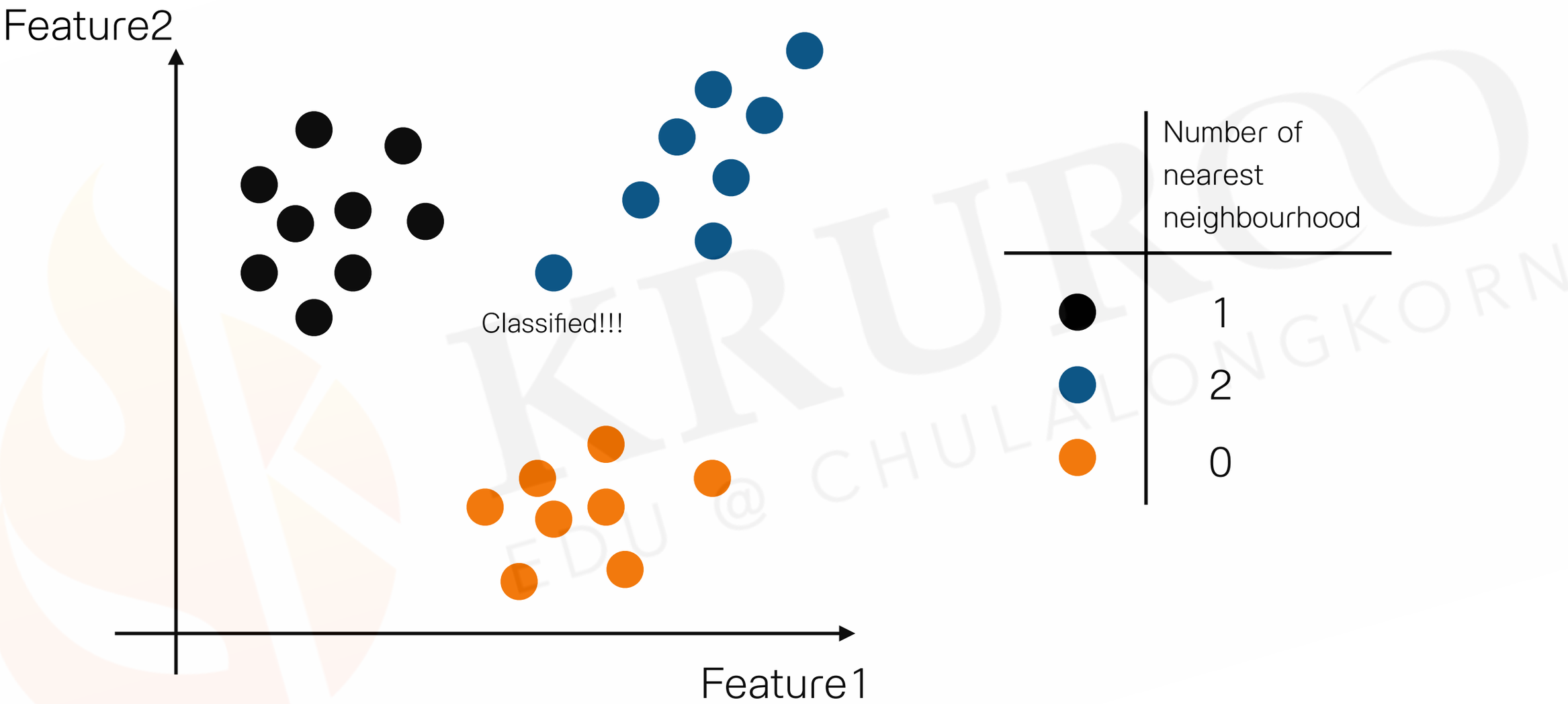


Number of nearest neighbourhood	
●	0
●	0
●	3

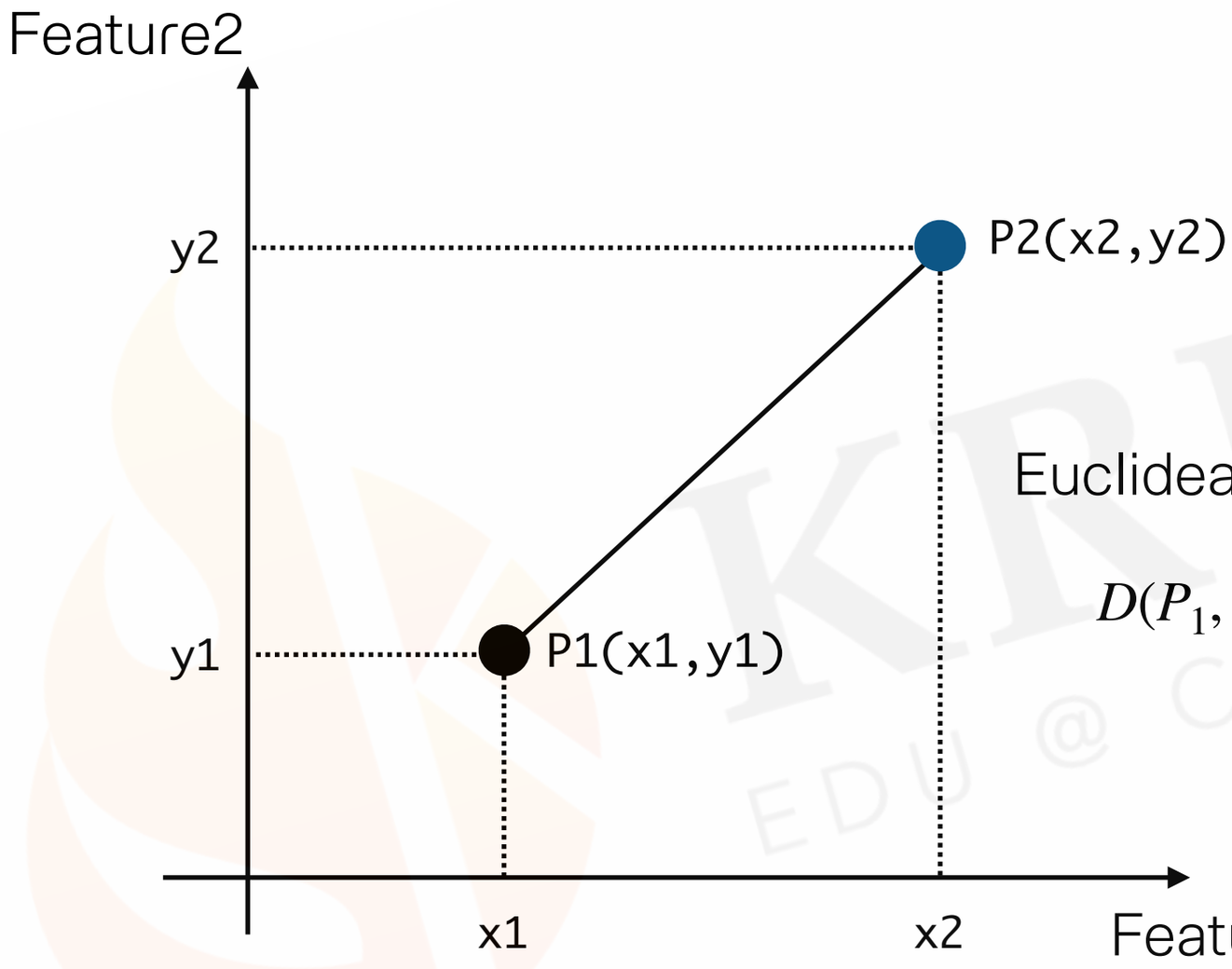
K-NN: Basic Concept



K-NN: Basic Concept



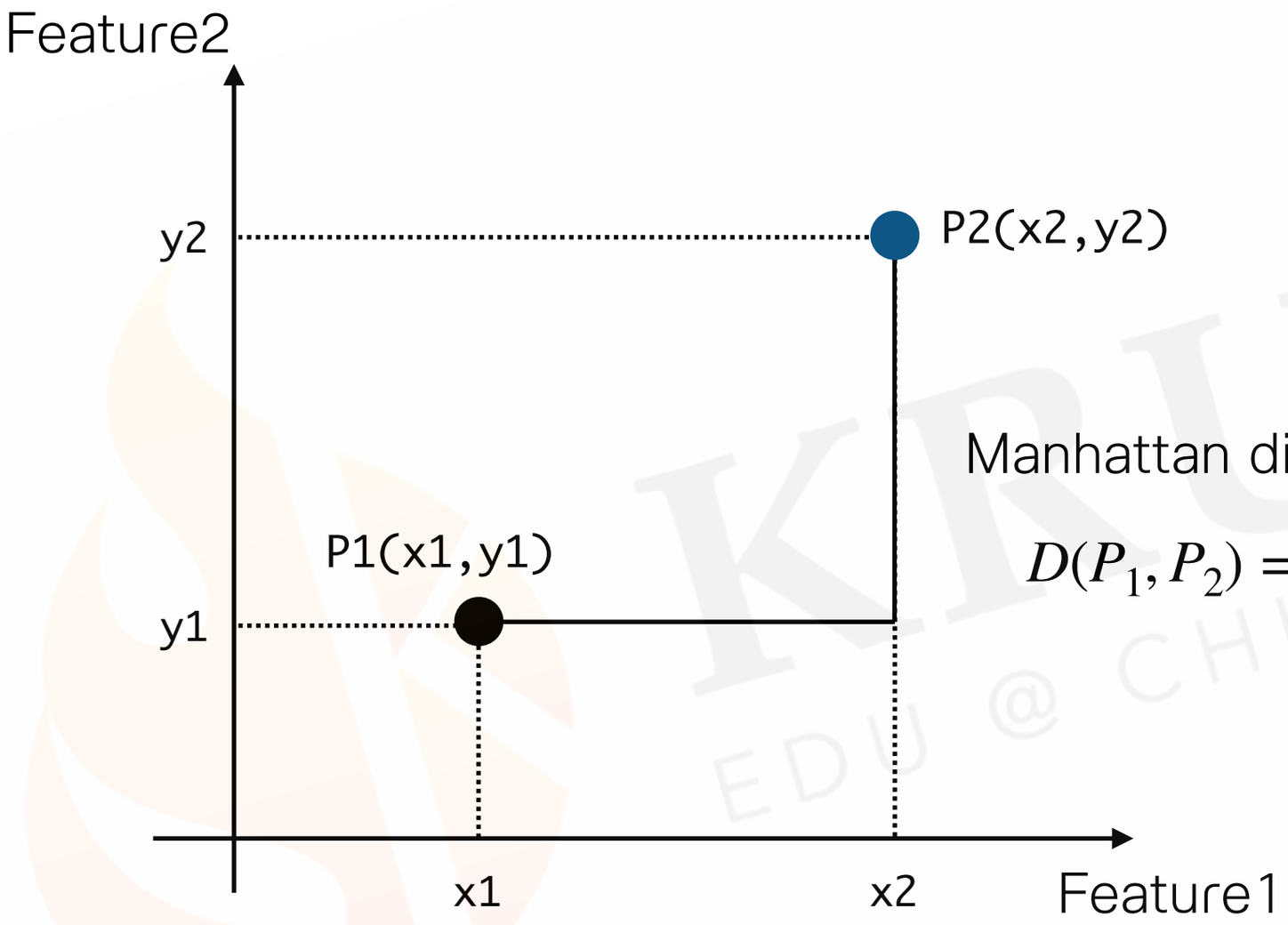
2D Euclidean distance



Euclidean distance between point P1 and P2 is

$$D(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

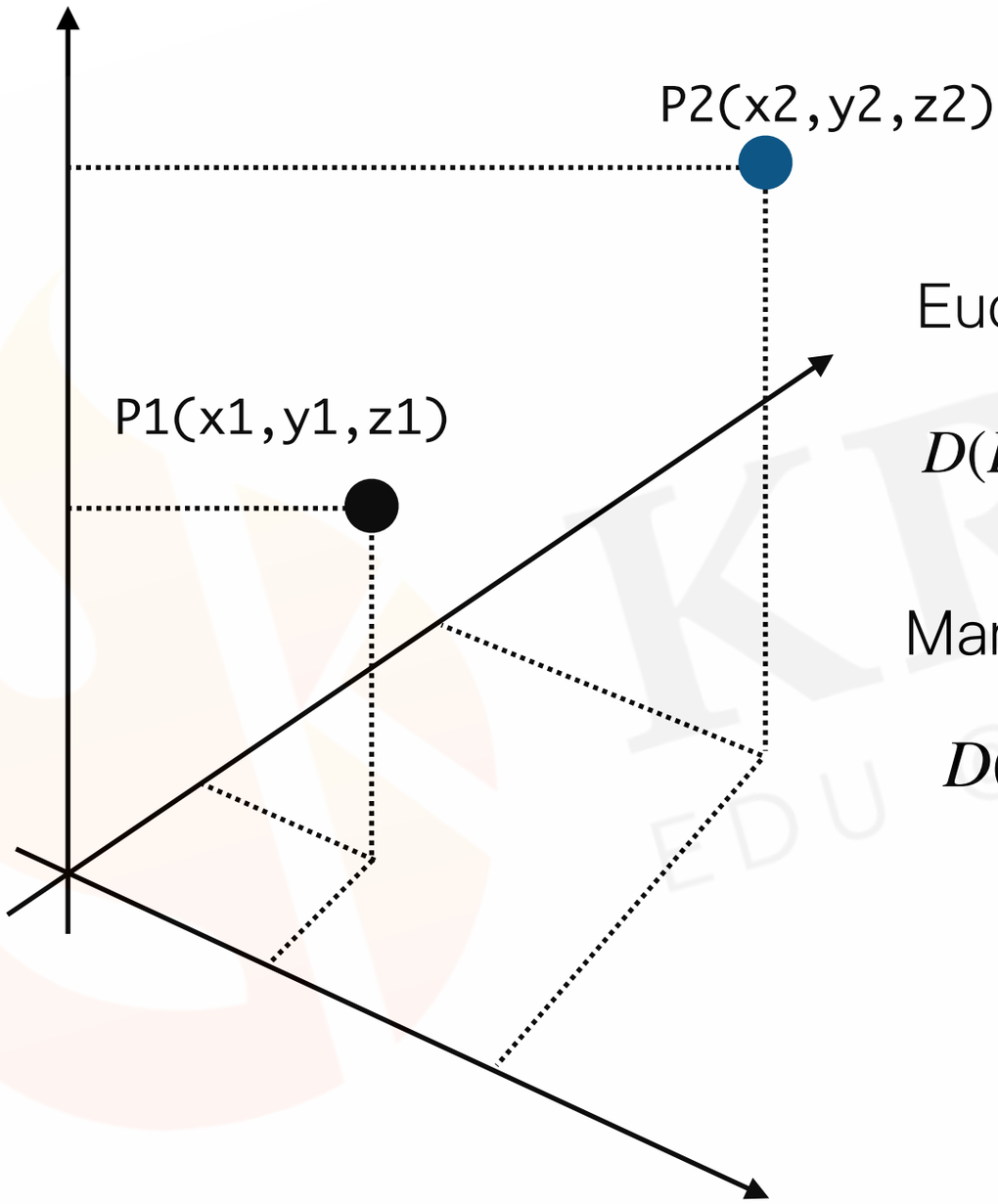
Manhattan distance



Manhattan distance between point P1 and P2 is

$$D(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2|$$

3D distance



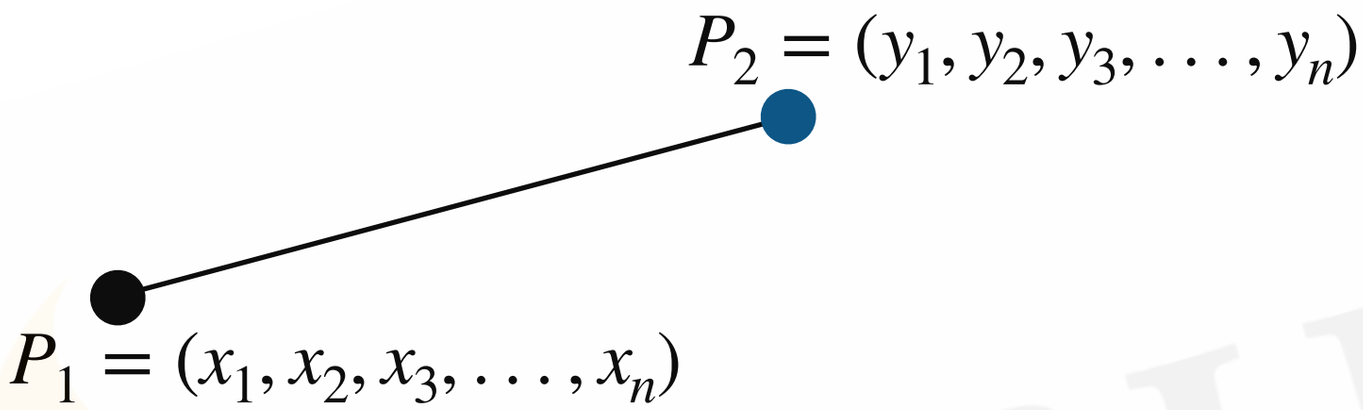
Euclidean distance between point P1 and P2 is

$$D(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Manhattan distance between point P1 and P2 is

$$D(P_1, P_2) = |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|$$

Multidimensional Euclidean distance



Euclidean distance between point P1 and P2 is

$$D(P_1, P_2) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \ldots + (x_n - y_n)^2}$$

Manhattan distance between point P1 and P2 is

$$D(P_1, P_2) = \sum_{i=1}^{n} |x_i - y_i|$$

K-Nearest Neighbourhood

```
> install.packages("class")
> library(class)
> pred<-knn(train.dat,test.dat,train.label)
> confusionMatrix(predicted,reference)
```

```
> library(mlbench)
> data(Sonar)
```

Sonar, Mines Vs. Rocks

This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network [1]. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp. The label associated with each record contains the letter "R" if the object is a rock and "M" if it is a mine (metal cylinder). The numbers in the labels are in increasing order of aspect angle, but they do not encode the angle directly.

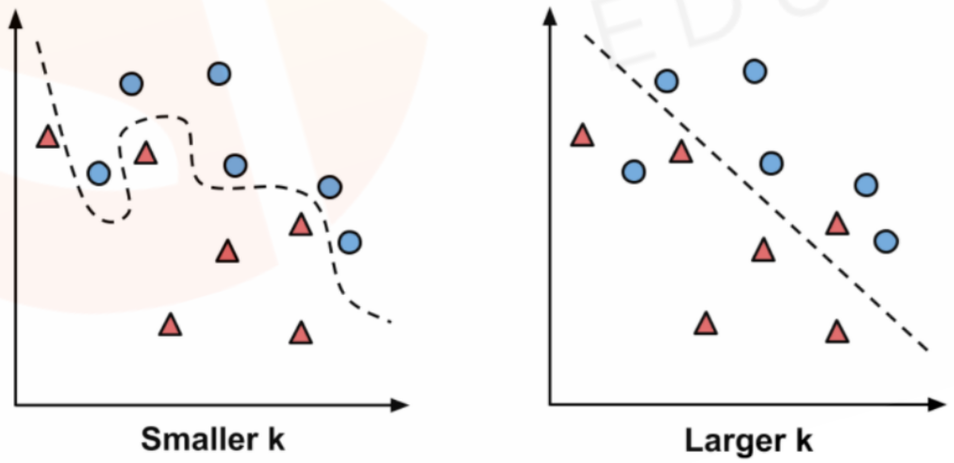
https://www.rdocumentation.org/packages/mlbench/versions/2.1-1/topics/Sonar

Improve the model performance:

Hyperparameter K

K is a hyperparameter of KNN that specifies **the number of neighbours** to consider when making the classification.

- **K=1** means that the only single nearest, most similar, neighbour was used to classify the new (unlabeled) data point.
- **K=5** means that five nearest, most, most similar, neighbours was used to classify the new (unlabeled) data point.
- **K=20** means that twenty nearest, most, most similar, neighbours was used to classify the new (unlabeled) data point.



Preprocessing data for K-NN

- K-NN assumes numeric data - many distance functions assume that your data are in numeric format and it is difficult to define the distance between categories.

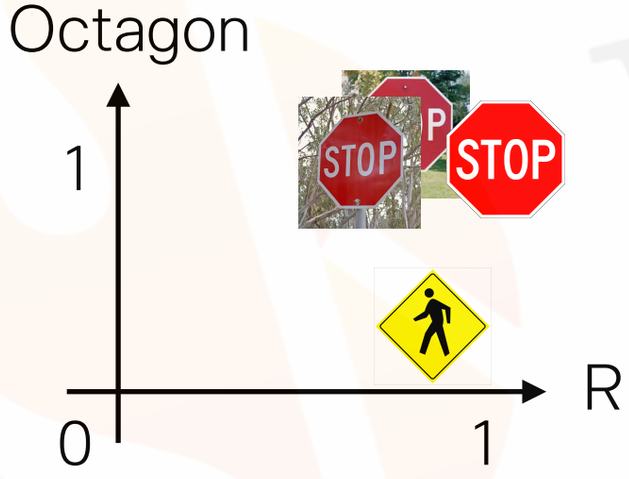
	Rectangle	Diamond	Octagon	R	G	B
	0	0	1	255	4	0
	0	1	0	254	244	0

Preprocessing data for K-NN

- It is important to be aware that when calculating distance, each features should be measured with the same scales.
- Such a different scale allows the features with a wider range to have more influence over the distance measuring.

Preprocessing data for K-NN

- It is important to be aware that when calculating distance, each features should be measured with the same scales.
- Such a different scale allows the features with a wider range to have more influence over the distance measuring.



Standardised scale: $Z = \frac{x - \bar{x}}{SD}$

Normalized scale: $N = \frac{x - min(x)}{max(x) - min(x)}$

The second hyper-parameter (optional)

Minkowski distance: $D(P_1, P_2) = \left(\sum_{i=1}^{n} |x_i - y_i|^q \right)^{1/q}$

```
> control<-trainControl(method="repeatedcv",number=10,repeats=5,
                        savePredictions="all")
> grid<-expand.grid(kmax = 1:10, # allows to test a range of k values
                    distance = 1:5, # allows to test a range of distance values
                    kernel = 'rectangular')
> fit<-train(Class~.,data=train.dat,
   method="kknn",
   trControl=control,
   tuneGrid=grid,
   preProcess=c("range"))
```

The third hyper-parameter (optional)

Weighted k-Nearest-Neighbor Techniques and Ordinal Classification

Klaus Hechenbichler
hechen@stat.uni-muenchen.de
Institut für Statistik, Ludwig-Maximilians-Universität München,
Akademiestraße 1, 80799 München, Germany

Klaus Schliep
k.p.schliep@massey.ac.nz
Allan Wilson Centre for Molecular Ecology and Evolution, Massey University,
Private Bag 11222, Palmerston North, New Zealand

13th October 2004

Abstract

In the field of statistical discrimination k-nearest neighbor classification is a well-known, easy and successful method. In this paper we present an extended version of this technique, where the distances of the nearest neighbors can be taken into account. In this sense there is a close connection to LOESS, a local regression technique. In addition we show possibilities to use nearest neighbor for classification in the case of an ordinal class structure. Empirical studies show the advantages of the new techniques.

https://epub.ub.uni-muenchen.de/1769/1/paper_399.pdf

The third hyper-parameter (optional)

Weighted k-Nearest-Neighbor classification (wkNN)

1. Let $L = \{(y_i, x_i), i = 1, \ldots, n_L\}$ be a learning set of observations x_i with given class membership y_i and let x be a new observation, whose class label y has to be predicted.

2. Find the $k + 1$ nearest neighbors to x according to a distance function $d(x, x_i)$.

3. The $(k + 1)$th neighbor is used for standardization of the k smallest distances via

$$D_{(i)} = D(x, x_{(i)}) = \frac{d(x, x_{(i)})}{d(x, x_{(k+1)})} \quad .$$

4. Transform the normalized distances $D_{(i)}$ with any kernel function $K(.)$ into weights $w_{(i)} = K(D_{(i)})$.

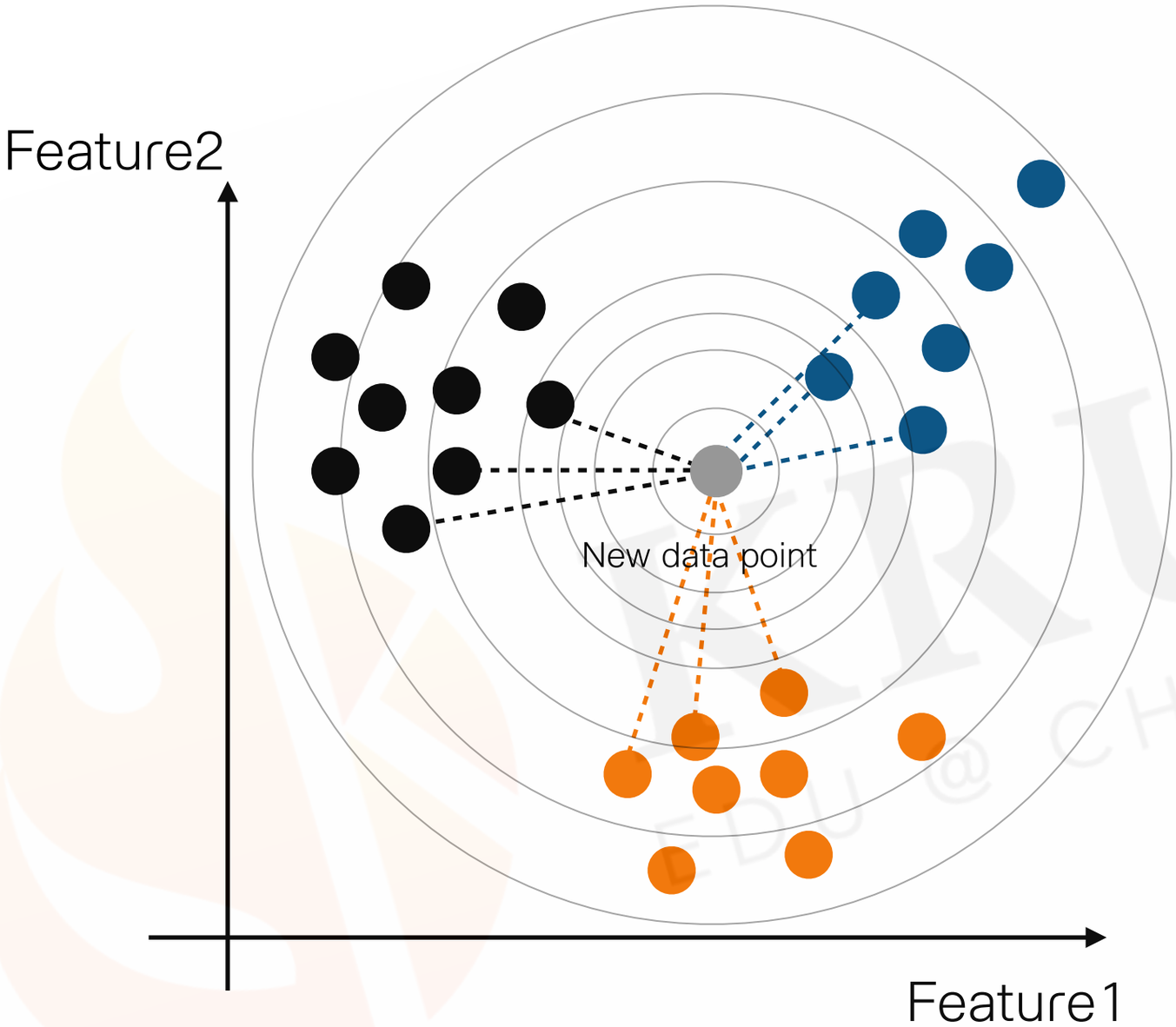
5. As prediction for the class membership y of observation x choose the class, which shows a weighted majority of the k nearest neighbors

$$\hat{y} = \max_r \left(\sum_{i=1}^{k} w_{(i)} I(y_{(i)} = r) \right) \quad .$$

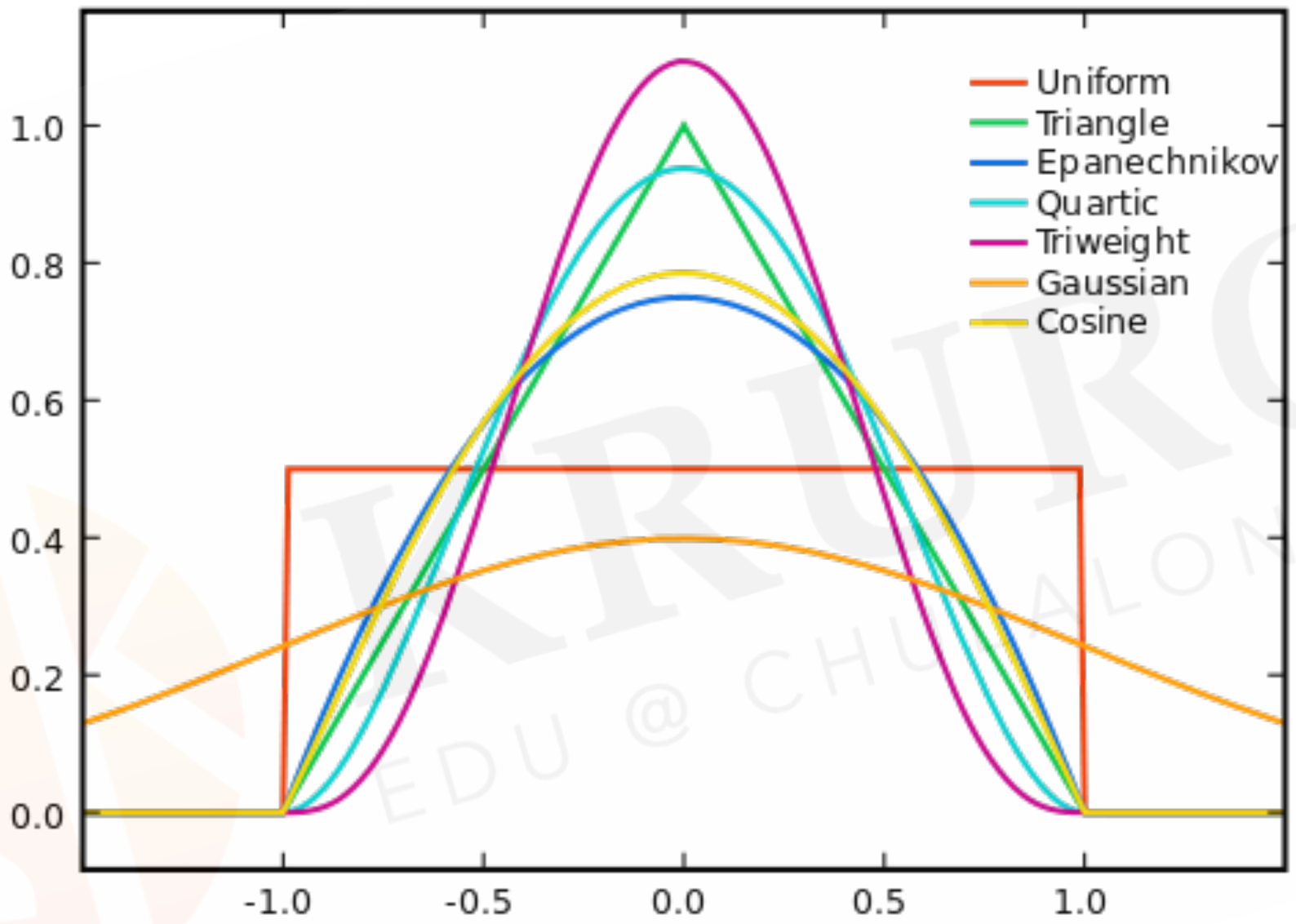
---

https://epub.ub.uni-muenchen.de/1769/1/paper_399.pdf

Weighted K-NN

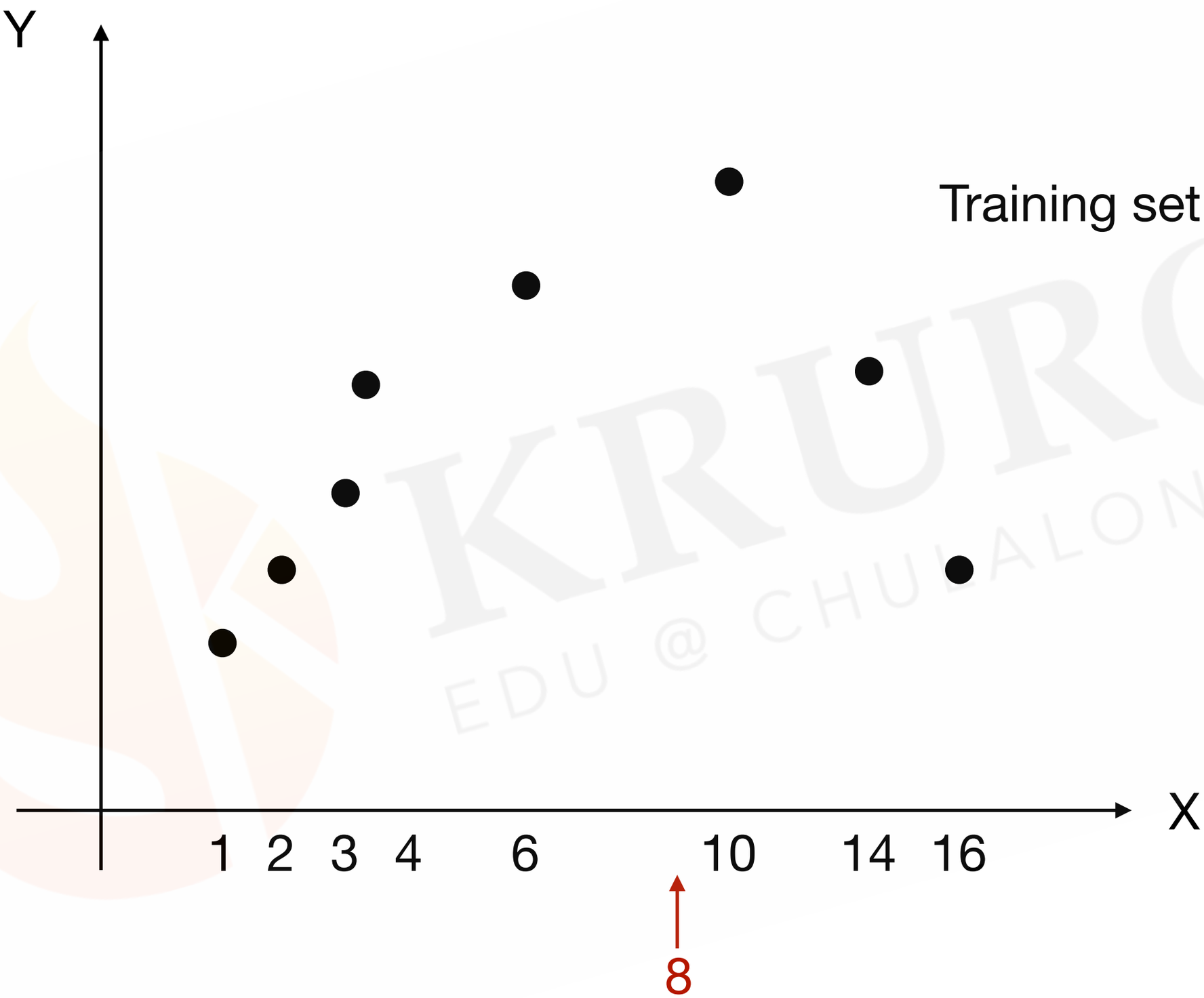


Kernel

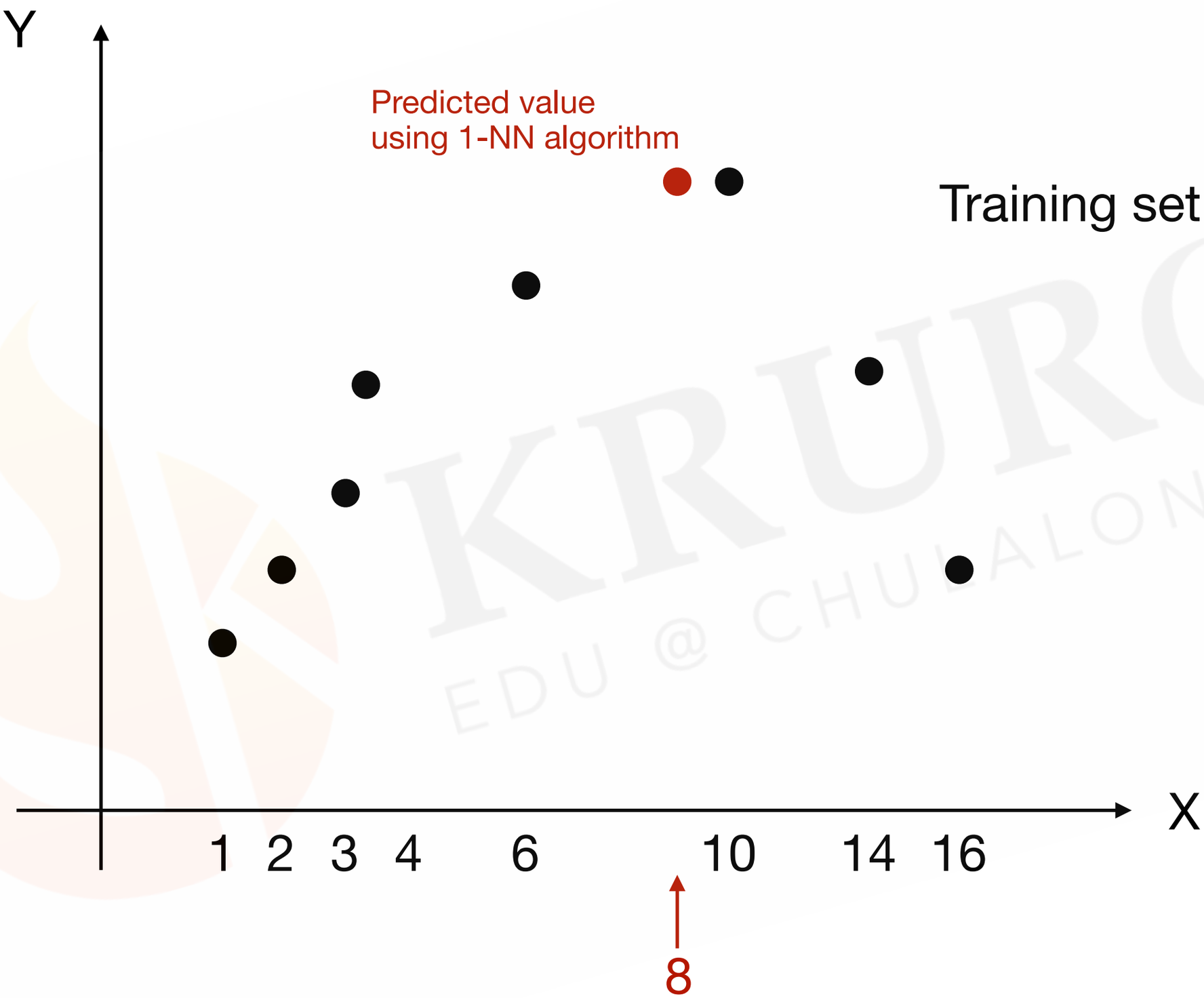


https://en.wikipedia.org/wiki/Kernel_(statistics)

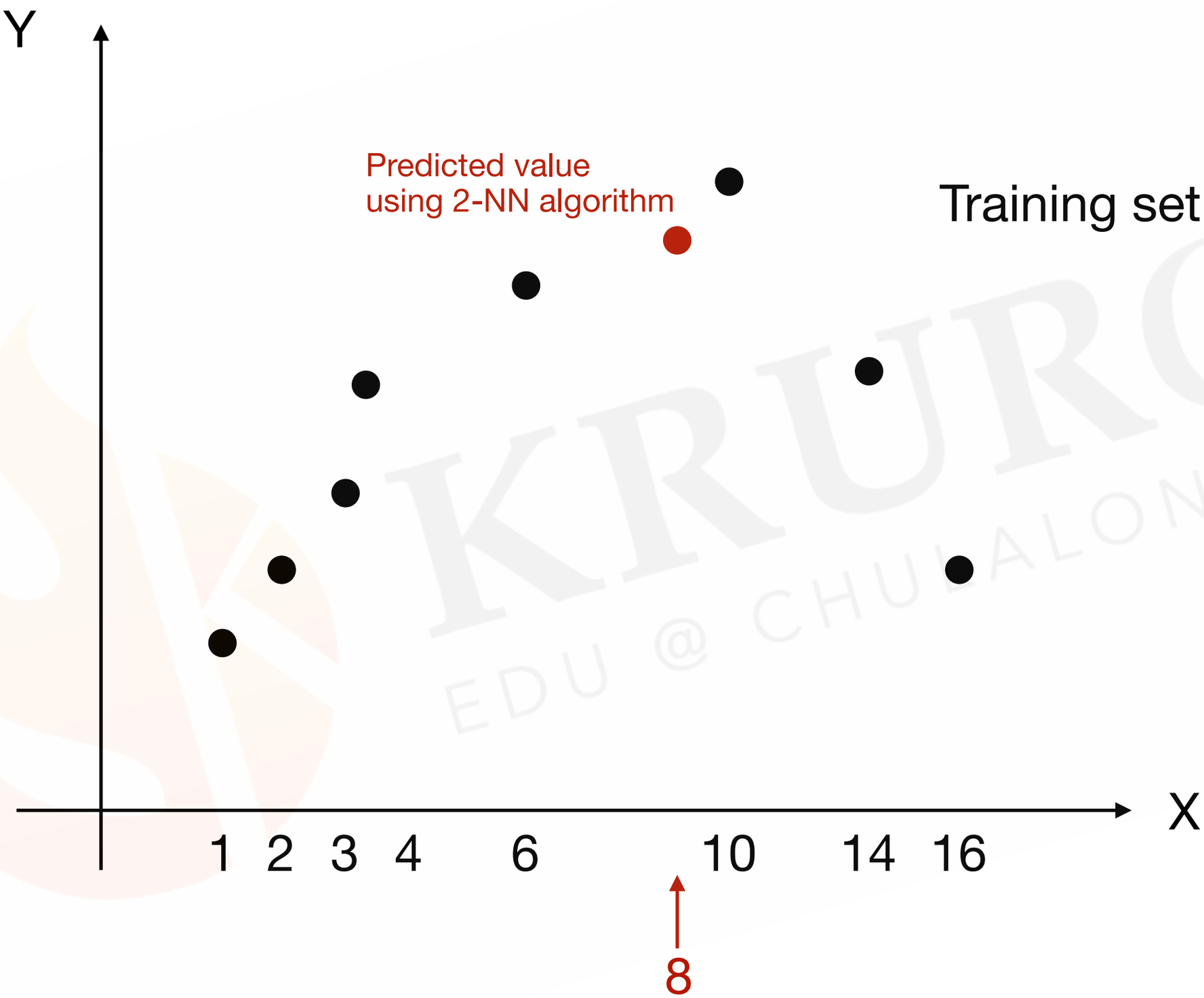
K-NN regression



K-NN regression



K-NN regression



K-NN regression