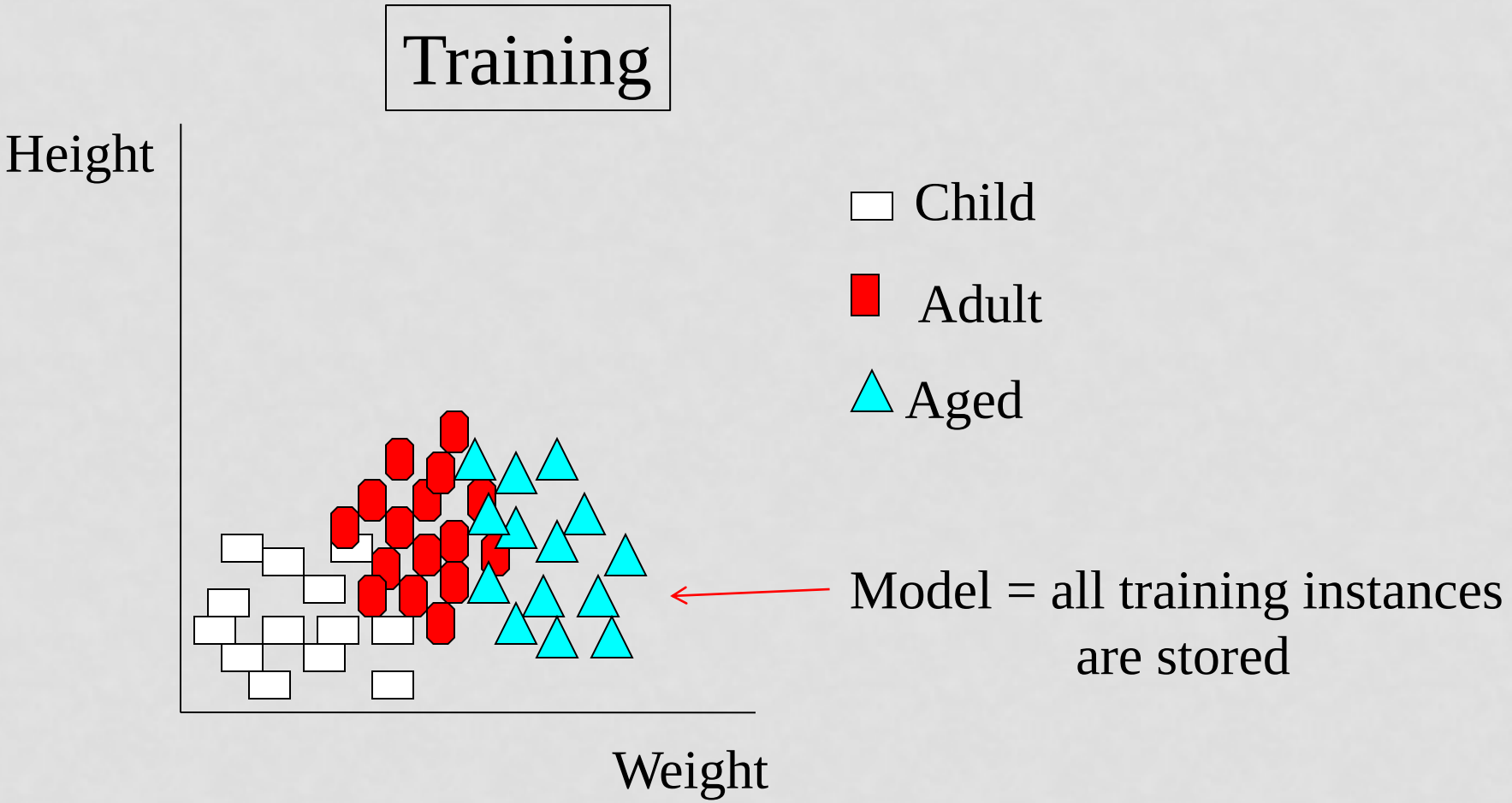
KNN: K-NEAREST NEIGHBOURS

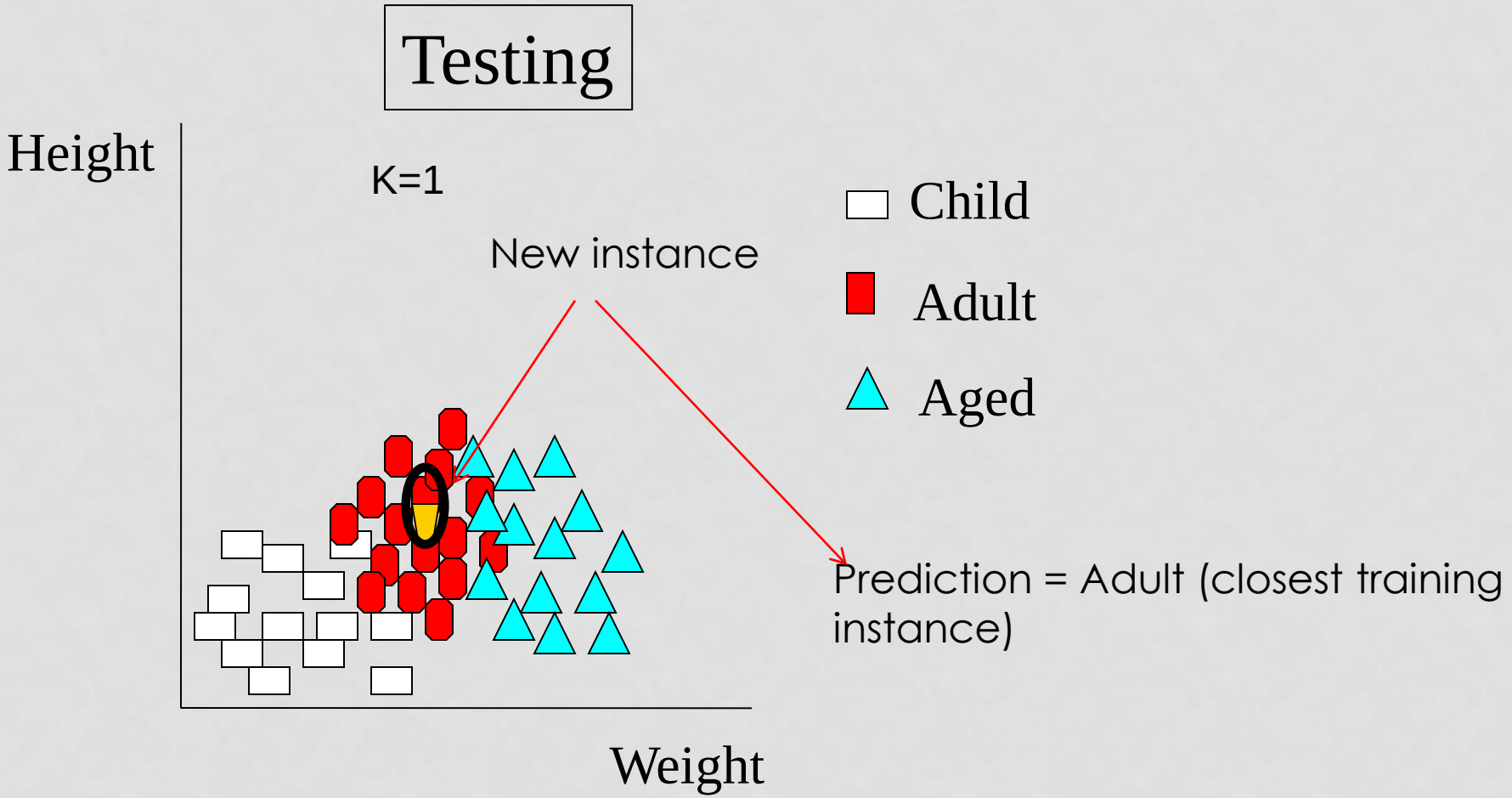
CLASSIFICATION PROBLEMS

- Let us suppose that we have a database of people
- For each person we know two features: weight and height
- And we intend to classify new persons into three classes: child, adult, aged

K-NEAREST NEIGHBORS (KNN)

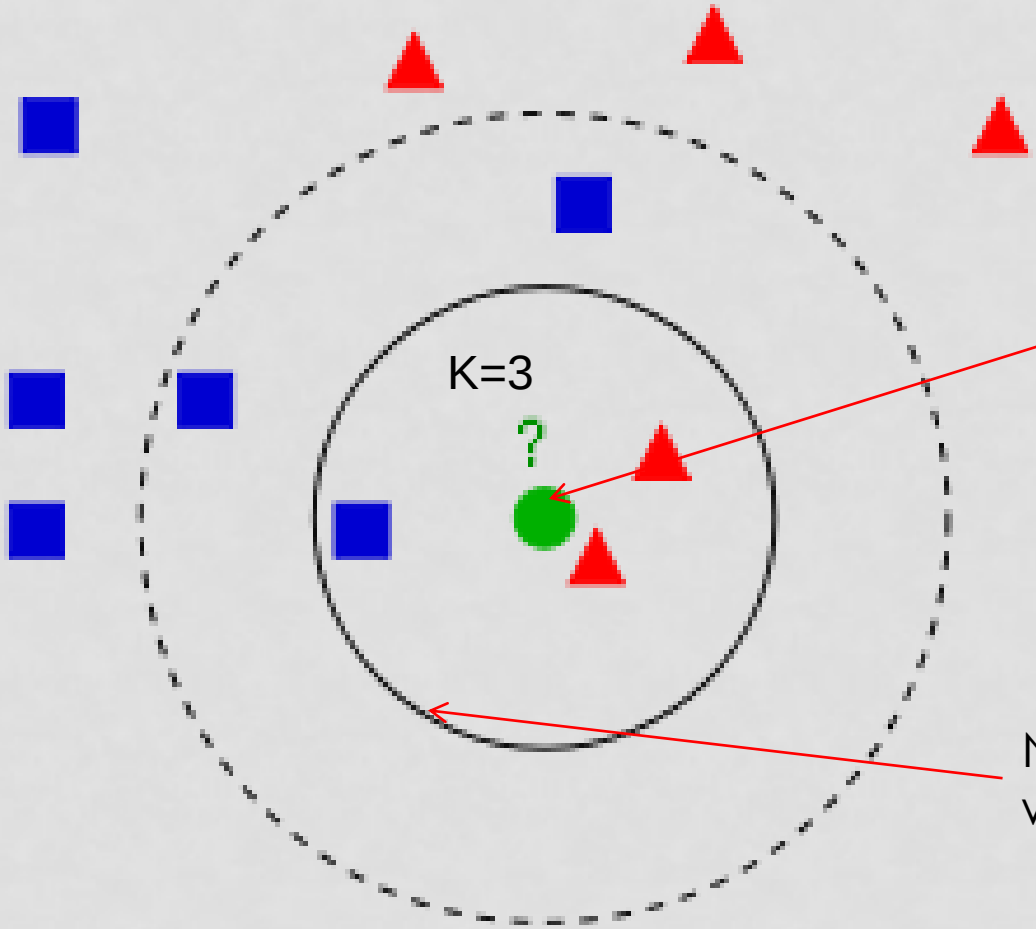


K-NEAREST NEIGHBORS (KNN)



K-NEAREST NEIGHBORS (KNN)

K> 2 => classify new instances as the majority class of the k-nearest neighbours

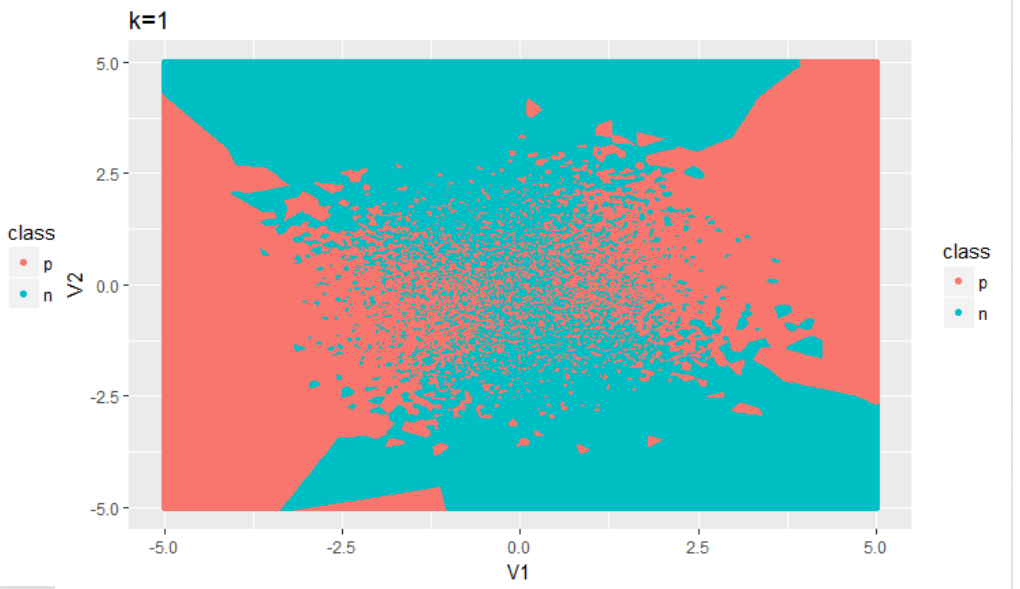


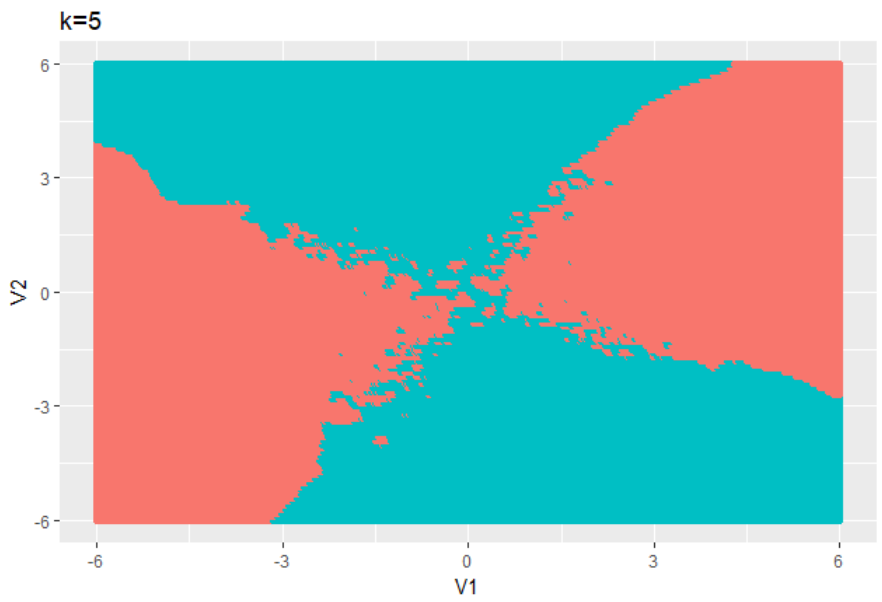
Test instance

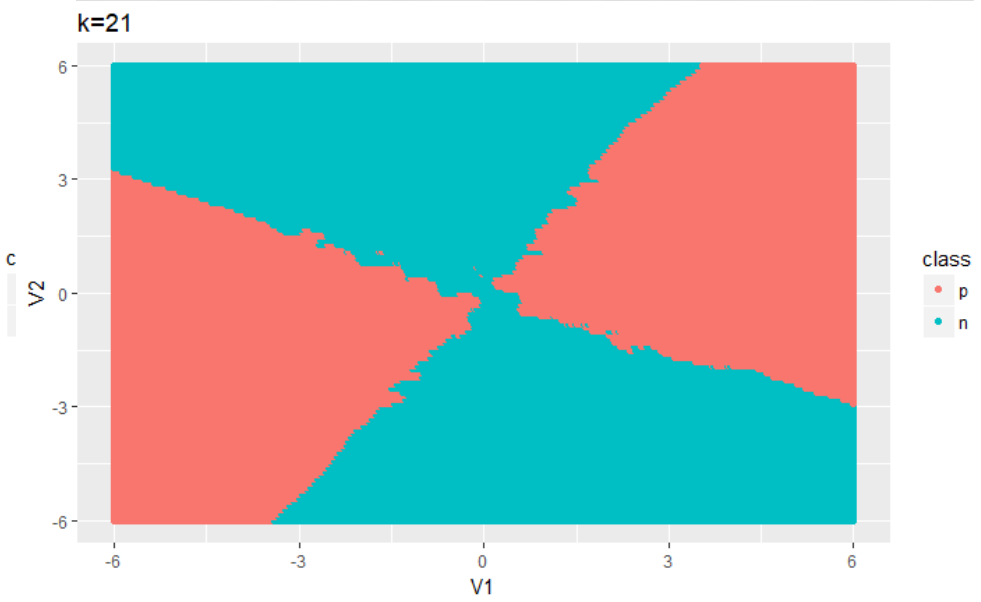
Triangles = 2
Squares = 1
Prediction (majority class in the neighbourhood) = **Triangle**

Neighbourhood with k=3


4
0
-4
V2
-6
-3
0
3
6
V1

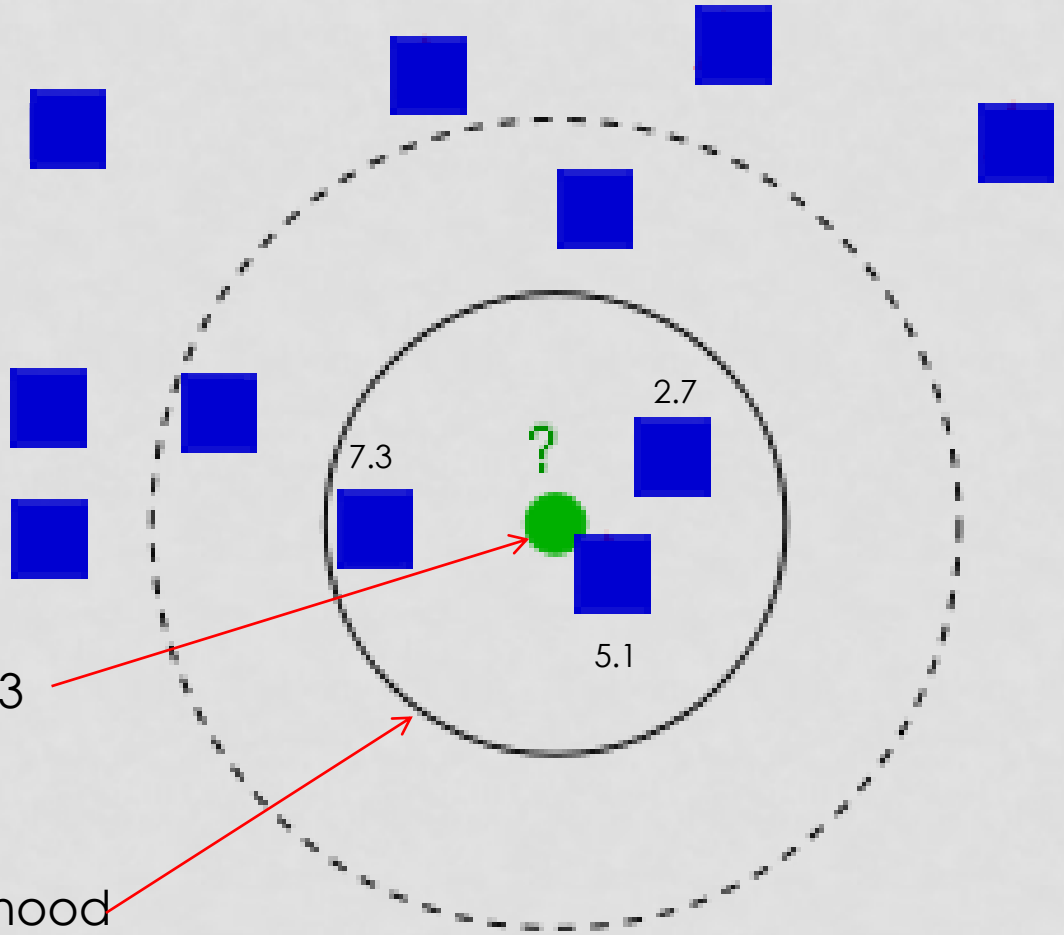

k=1
5.0
2.5
0.0
-2.5
-5.0
V2
-5.0
-2.5
0.0
2.5
5.0
V1
class
p
n


k=5
6
3
0
-3
-6
V2
-6
-3
0
3
6
V1


k=21
6
3
0
-3
-6
V2
-6
-3
0
3
6
V1
class
p
n

KNN FOR REGRESSION

It can be easily extended for **regression** by computing the average of the k-nearest neighbours



Prediction = (7.3+2.7+5.1)/3

Neighbourhood with k=3

Minkowsky distance

Typically, the Euclidean distance is used:

$$L_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{i=n} (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

n attributes

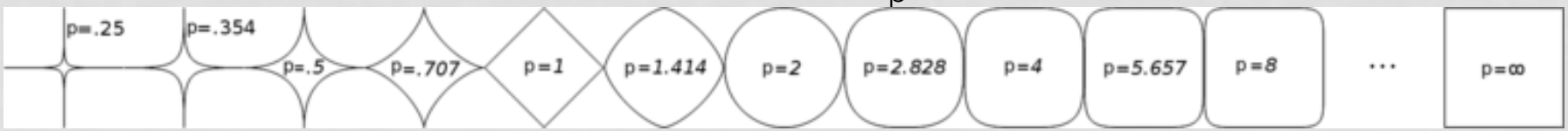
$\mathbf{x} = (x_1, x_2, \ldots, x_n)$
$\mathbf{y} = (y_1, y_2, \ldots, y_n)$

But in some problems, the Minkowsky distance, for some p, might work better,

$$L_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{i=n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

$$L_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max\{|x_1 - y_1|, |x_2 - y_2|, \ldots, |x_n - y_n|\}$$

Set of points for which $L_p(\mathbf{x},\mathbf{y}) = 1$



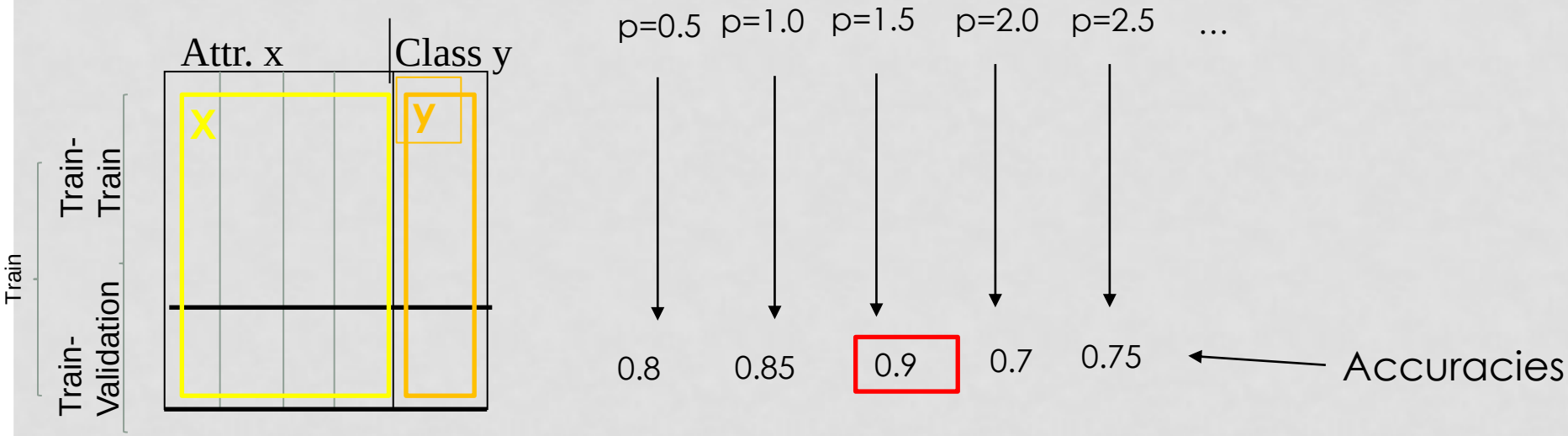
L_1 L_2 L_∞

HYPER-PARAMETERS OF KNN

- While K (the number of neighbors) is the main hyper-parameter of KNN
- In some cases, the exponent *p* of the Minkowsky distance might be a relevant hyper-parameter as well.
- In R, the Minkowsky distance is not available in KNN libraries FNN nor class
- In scikit-learn, it is one of the hyper-parameters, with default value p = 2 (that is, the Euclidean distance)

class `sklearn.neighbors.`**`KNeighborsClassifier`**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

HYPER-PARAMETER TUNING WITH A VALIDATION PARTITION



- KNN is tried with different values of p,
- The accuracy of each one is computed on the train-validation partition
- The best p (=1.5) is identified
- Then, we can use p=1.5 for making predictions on new data