Classification

- **K-Nearest Neighbor**

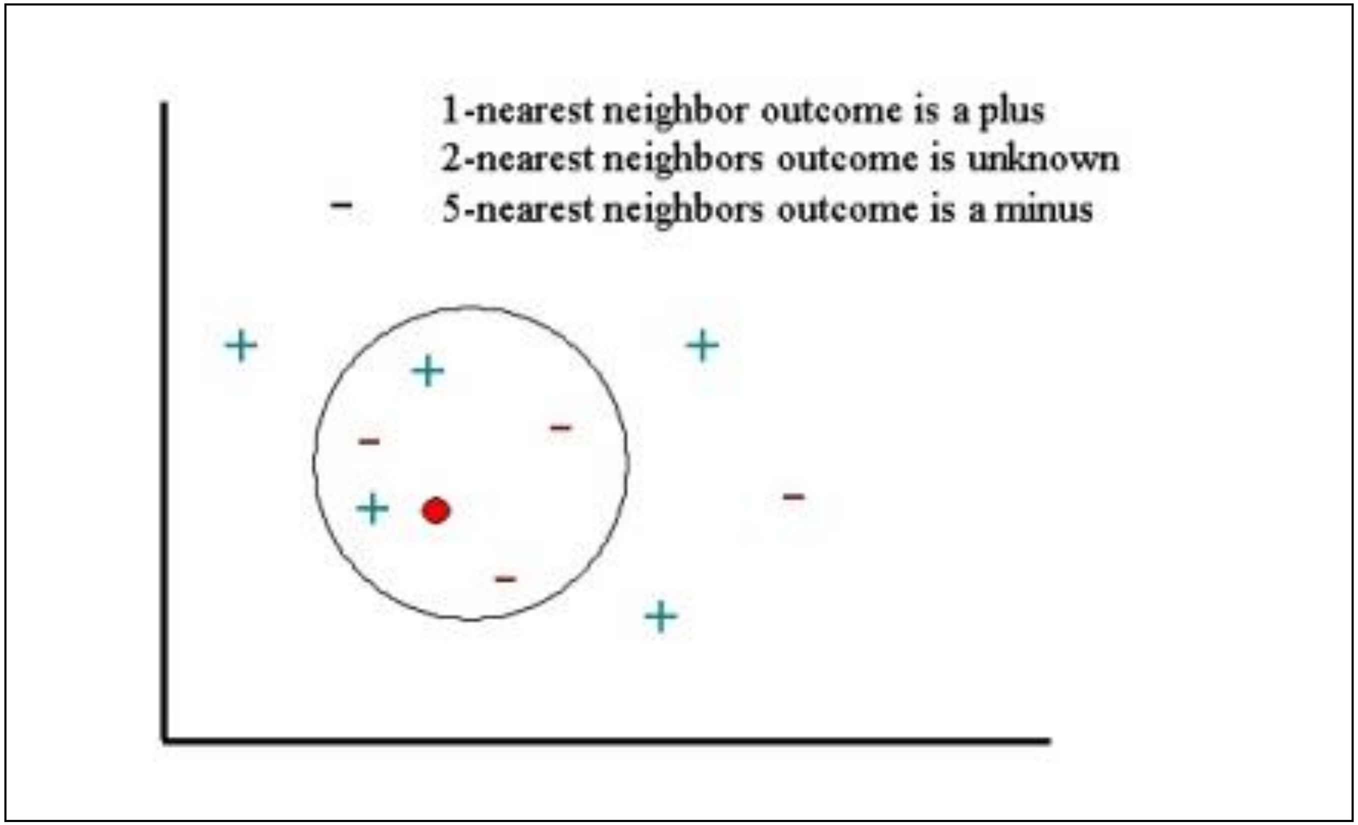
K-Nearest Neighbor

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

K-Nearest Neighbor...Example

To demonstrate a *k*-nearest neighbor analysis, let's consider the task of classifying a new object (query point) among a number of known examples. This is shown in the figure below, which depicts the examples (instances) with the plus and minus signs and the query point with a red circle. Our task is to estimate (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, we want to know whether the query point can be classified as a plus or a minus sign.

K-Nearest Neighbor...Example



K-Nearest Neighbor...Distance measures

The data structure we have is like (x, y), where y is class labels (as we have classification task, it is continuous variable in case of regression). The features x's can either be numeric (Age, income, family size etc.) or non-numeric (Gender, designation etc.).

According to nature of feature (x), we use different distance measures to find nearness of x's.

K-Nearest Neighbor...Distance measures

For numeric features

Euclidean distance:

$$D(x - x_{new}) = \sqrt{\sum_{i=1}^{k} (x_i - x_{i,new})}$$

Manhattan distance:

$$D(x - x_{new}) = \sum_{i=1}^{k} | x_i - x_{i,new} |$$

Minkowski distance:

$$D(x - x_{new}) = \left(\sum_{i=1}^{k} \left(| x_i - x_{i,new} | \right)^{p} \right)^{1/p}$$

K-Nearest Neighbor...Distance measures

For non-numeric (categorical) features

Here we recode non-numeric features to numeric. For example, Male=0, Female=1 or Income level Low=0, Medium=1, High=2.

Hamming distance:

$$D(x - x_{new}) = \sum_{i=1}^{k} | x_i - x_{i,new} |$$

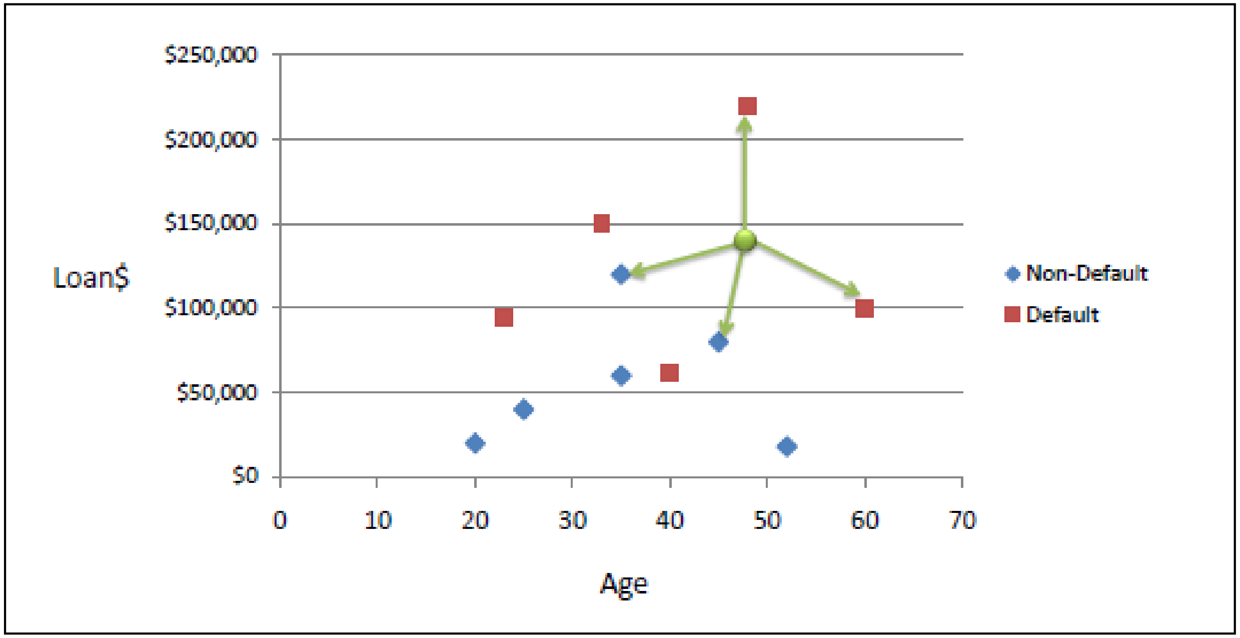
K-Nearest Neighbor...K?

- Divide the data as training and testing data.
- Choose k=1, and calculate test error.
- Accuracy = 1- test error.
- Add k by unity and again calculate Accuracy.
- Accuracy will start falling.
- Select the value of k, when Accuracy start increasing again.

- The value for k is generally chosen as the square root of the number of observations.

K-Nearest Neighbor...Example

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



K-Nearest Neighbor...Example

We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01

>> Default=Y

K-Nearest Neighbor...Example

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	?		

Euclidean Distance

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

K-Nearest Neighbor...Example

Using the standardized distance on the same training set, the unknown case returned a different neighbor which is not a good sign of robustness.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

K-Nearest Neighbor

R/Python session on knn

Thank You