Data Mining Laboratory

Ewa Figielska

Classification

- **Classification** - the most common data mining task.
- For example:
  - *Banking:* whether a particular credit card transaction is fraudulent;
  - *Medicine:* diagnosing whether a particular disease is present;
  - *Law:* determining whether a will was written by the actual person deceased or fraudulently by someone else;

- In classification there is a target **categorical** variable, which is partitioned into predetermined classes or categories, e.g. the target income bracket can be partitioned into high income, middle income, and low income.
- Each record in the set of records (database) contains information on the target variable as well as a set of input (predictor) variables.

- Consider the excerpt from a data set:

Subject	Age	Gender	Occupation	Income Bracket
001	47	F	Software engineer	High
002	28	M	Marketing consultant	Middle
003	35	M	Unemployed	Low
…				

The example of a classification task: to **classify** the income bracket of persons not currently in the database, based on the other characteristics associated with that person, such as age, gender, and occupation.

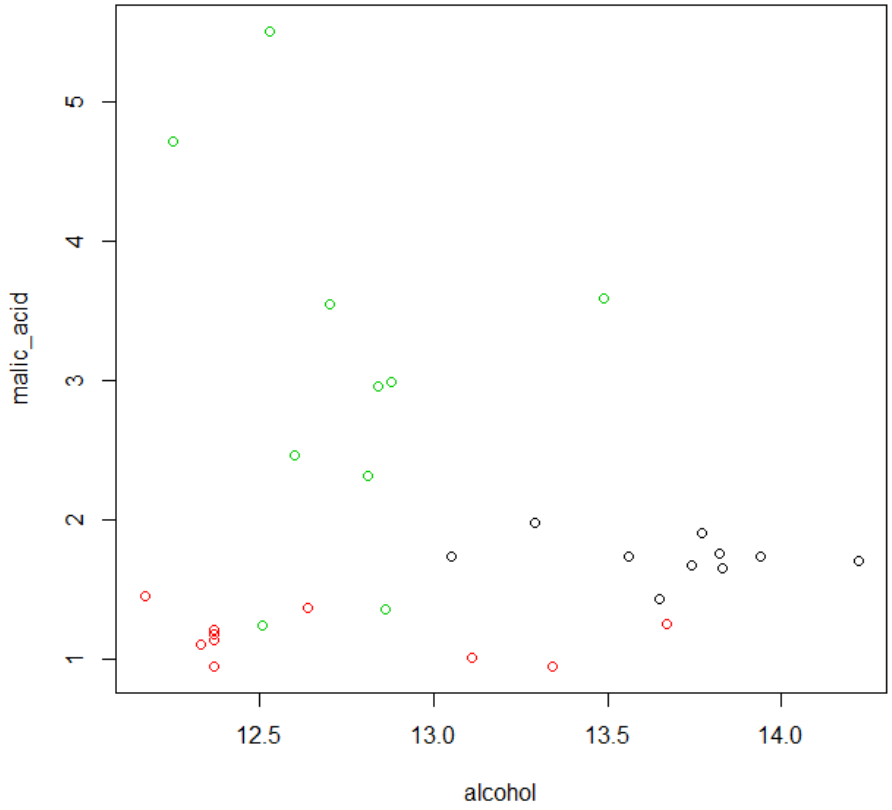
k-nearest neighbor algorithm

- k-nearest neighbor (knn) algorithm – can be used for classification, estimation and prediction.
- knn is an example of **instance-based learning**, in which the training data set is stored, so that a classification for a new unclassified record may be found simply by comparing it to the k most similar records in the training set.

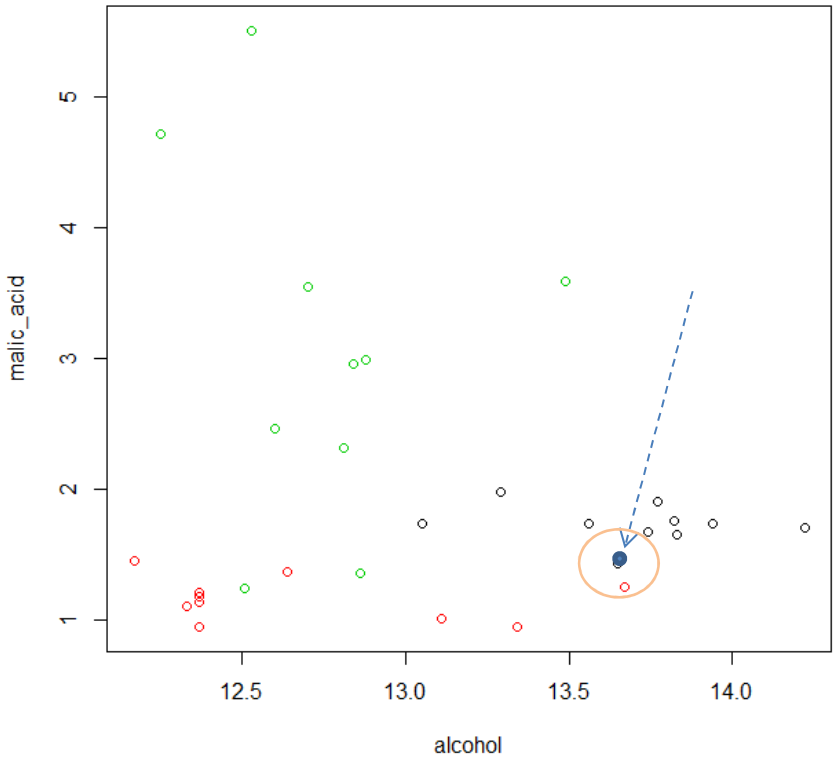
Example:
File “mwine.csv”

We are interested in classifying the type of wine based on certain wine characteristics, such as the amount of alcohol and malic acid.

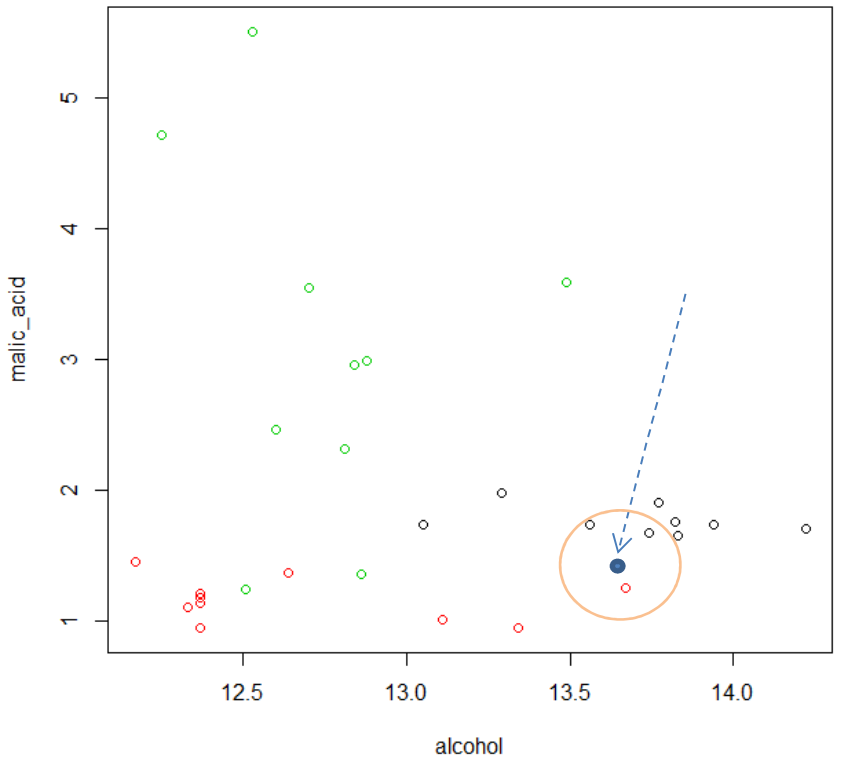
In the scatter plot, black points indicate type t1, red points indicate type t2, and green points indicate type t3.



Example



- Suppose that the indicated record represents new data.
- If $k = 1$, the new record will be classified as a red (t2) one as the closest single neighbor is red.



- If $k = 3$, the new record will be classified as a black one (t1) since there are two black and one red points among the closest three neigbors of the new point.

Problems

- Issues involved in building a classifier using the *k*-nearest neighbor algorithm include:
  - How many neighbors should we consider (what is *k*)?
  - How do we measure distance?
  - Should all points be weighted equally, or should some points have more influence than others?
  - How do we combine the information from more than one observation?

Twofold cross validation - example

1. #install.packages **("class")**
2. **library(class)** #for the knn algorithm
3. **d<-read.csv(file="mwine.csv",header=TRUE,sep=',') #d**ata are taken from "mwine.csv" file (it is a fragment of "wine" file from UCI); data contain two input attributes : alcohol and malic_acid (in columns 2 and 3) and a class attribute, wine_type, (in column 1).
4. **d<-d[sample(nrow(d**)),] #data are randomly shuffled. It is necessary, as the original data can be ordered in some specific way.

5. # creating test and training sets
6. **test_indices<-c(1:10)** #determining the indices of records to be included in the test set
7. **d.test <- d[test_indices,2:3]** #test set contains data from columns 2 and 3 of the first 10 records.
8. **d.test.class<- d[test_indices,1]** #d.test.class contains data from the class column of the first 10 records.
9. **d.train <- d[-test_indices, 2:3]** #training set contains records which are not indicated as the test set. d.train does not contain class values.
10. **d.train.class <- d[-test_indices, 1]** #d.train.class contain class values for the training set.

11. #k-nearest neighbor algorithm
12. **knn.result <- knn(d.train, d.test, d.train.class, k = 1**) # knn(train, test, cl, k=1, l=0, prob= FALSE, use.all = TRUE), where **train** is a matrix of training set cases, **test** is a matrix of test set cases, **cl** is a factor of true classifications of training set, **k** is a number of neighbours considered.

13. **t<-table(d.test.class,knn.result**) #creating a **confusion matrix**
14. **err_rate<-mean(knn.result!=d.test.class)** #calculating the **error rate**
15. #printing the results
16. **cat("\n\nerror rate = ",err_rate,"\n")** #concatenates strings and prints
17. **print(t)**

Evaluating the results

- The obtained results – classification

```
> knn.result
 [1] t2 t2 t3 t3 t1 t3 t1 t2 t3 t1
Levels: t1 t2 t3
```

Test set with true classification

	wine type	alcohol	malic acid
16	t2	12.17	1.45
25	t3	12.51	1.24
26	t3	12.60	2.46
22	t3	12.88	2.99
3	t1	13.83	1.65
30	t3	12.84	2.96
14	t2	13.67	1.25
11	t2	12.37	0.94
13	t2	12.64	1.36
5	t1	13.77	1.90
29	t3	13.49	3.59
1	t1	13.94	1.73
10	t1	13.65	1.43
6	t1	13.74	1.67
23	t3	12.81	2.31
12	t2	12.33	1.10
17	t2	12.37	1.21
19	t2	12.37	1.17
15	t2	12.37	1.13
2	t1	13.05	1.73
7	t1	13.56	1.73
20	t2	13.34	0.94
18	t2	13.11	1.01
4	t1	13.82	1.75
8	t1	14.22	1.70
27	t3	12.25	4.72
24	t3	12.70	3.55
21	t3	12.86	1.35
9	t1	13.29	1.97
28	t3	12.53	5.51

- **Confusion matrix** - a matrix of the correct and incorrect classifications made by the algorithm. The columns represent the predicted classifications, and the rows represent the actual (true) classifications.

	knn.result		
d.test.class	t1	t2	t3
t1	2	0	0
t2	1	2	1
t3	0	1	3

- $\textbf{error rate} = \frac{\text{The number of misclassified records}}{\text{The number of records in the test set}} = 0.3$

(poor classifier)

k-fold cross validation - example

```
1.  library(class)
2.  d<-read.csv(file="iris.csv",header=TRUE,sep=',') # the iris data set is used
3.  d<-d[sample(nrow(d)),] #data are randomly shuffled

4.  #Creating the equally size folds, the number of folds is 10
5.  nbreaks<-10
6.  folds <- cut(seq(1,nrow(d)),breaks=nbreaks,labels=FALSE) #function cut() divides a given range into
    intervals; the leftmost interval corresponds to level one, the next leftmost to level two and so on.

7.  av_err_rate<-0   #setting the initial value for average error rate
8.  for(i in seq(1:nbreaks)){
9.     test_indices <- which(folds==i,arr.ind=TRUE) #function which() segments the data by fold
10.    d.test <- d[test_indices,1:4]        #without class, columns 1:4 contain the input variables (petal sizes)
11.    d.test.class<- d[test_indices,5]    #only class,  column 5 contains the target variable
12.    d.train <- d[-test_indices, 1:4]
13.    d.train.class <- d[-test_indices, 5]

14.    knn.result <- knn(d.train, d.test, d.train.class, k = 1) #knn algorithm

15.    t<-table(d.test.class,knn.result) #confusion matrix
16.    err_rate<-mean(knn.result!=d.test.class) #error rate
17.    av_err_rate = av_err_rate + err_rate   #summing the error rates
18.    cat("\n\nFold = ",i,"\t error rate = ",err_rate,"\n")  #printing the results for an iteration
19.    print(t)
20.  }
21.  av_err_rate = av_err_rate/nbreaks   #calculating the average error rate
```

Example of the results

```
Fold =  7        error rate =  0
            knn.result
d.test.class setosa versicolor virginica
  setosa          7          0         0
  versicolor      0          4         0
  virginica       0          0         4


Fold =  8        error rate =  0.06666667
            knn.result
d.test.class setosa versicolor virginica
  setosa          4          0         0
  versicolor      0          3         0
  virginica       0          1         7


Fold =  9        error rate =  0
            knn.result
d.test.class setosa versicolor virginica
  setosa          4          0         0
  versicolor      0          6         0
  virginica       0          0         5


Fold =  10       error rate =  0
            knn.result
d.test.class setosa versicolor virginica
  setosa          5          0         0
  versicolor      0          4         0
  virginica       0          0         6

>   av_err_rate = av_err_rate/nbreaks

>   av_err_rate
[1] 0.04
```

Dictionary

- target variable – zmienna celu
- input/predictor variable – zmienna opisująca
- knn – algorytm k-najbliższych sąsiadów
- cross validation – walidacja krzyżowa
- confusion matrix – macierz pomyłek
- error rate – poziom błędu