Introduction to machine learning in computational biology

by Davide Chicco

davide.chicco@gmail.com www.DavideChicco.it

Medical Biophysics Tech Talks

2017-10-13

Princess Margaret Cancer Centre
University Health Network
UHN

UNIVERSITY OF TORONTO

Outline

Session 1 – Information and theory

- 1a - Introduction to machine learning
  - what is machine learning?
  - dataset arrangement
  - supervised/unsupervised learning
  - overfitting
- 1b - Overview of machine learning programming languages and platforms (Torch, Python Theano, R)

Session 2 – Practice

- 2a - Introduction to k-nearest neighbors (k-NN)
- 2b - Exercise in R. Usage of k-NN for binary classification of cancer-related data

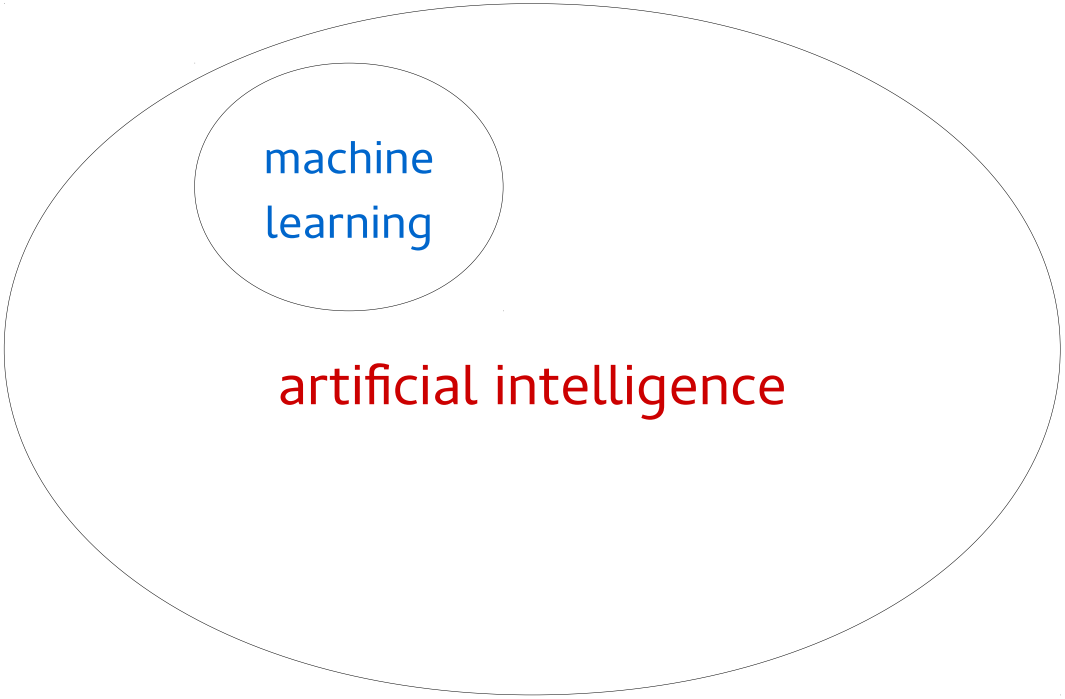
Outline

Session 1 – Information and theory

- 1a - Introduction to machine learning
  - ➢ what is machine learning?
  - ➢ dataset arrangement
  - ➢ supervised/unsupervised learning
  - ➢ overfitting
- 1b - Overview of machine learning programming languages and platforms (Torch, Python Theano, R)

What is machine learning?

What is machine learning?
(computational intelligence)
(data mining)
(pattern recognition)

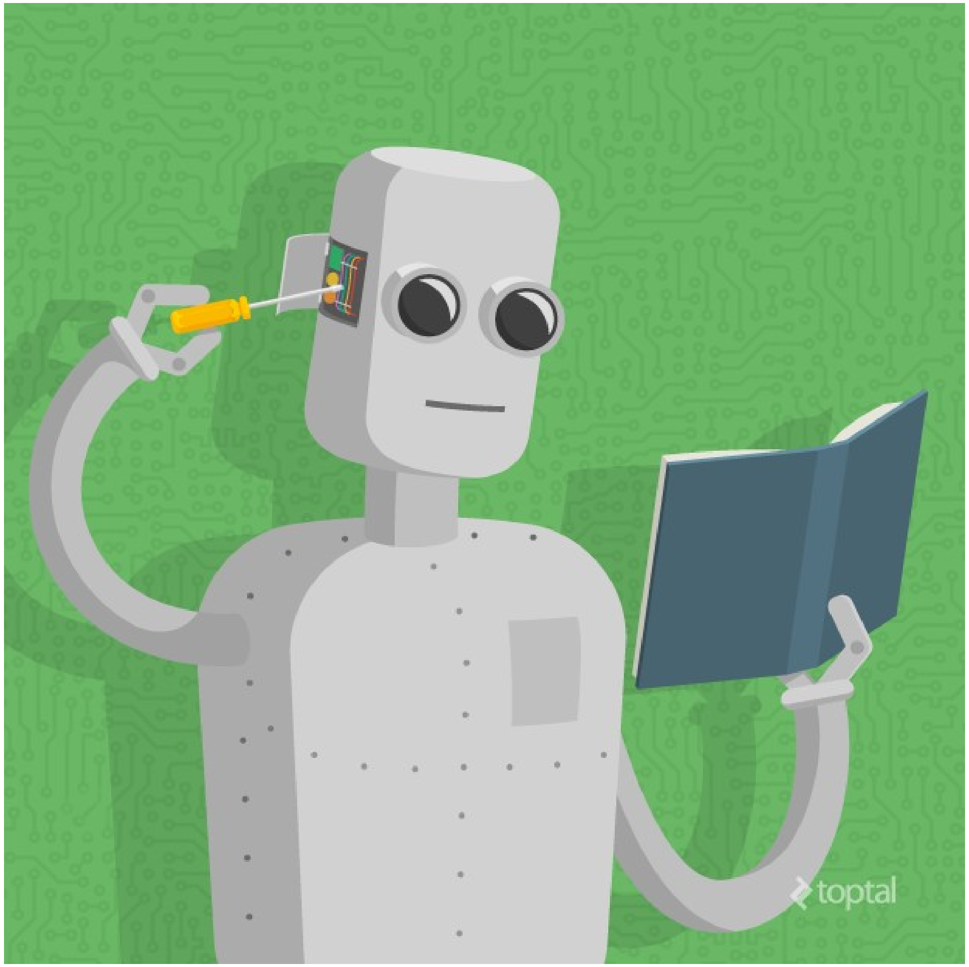


What is machine learning?

What is machine learning?
(computational intelligence)
(data mining)
(pattern recognition)

“[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.”

(Arthur Samuel, 1954)



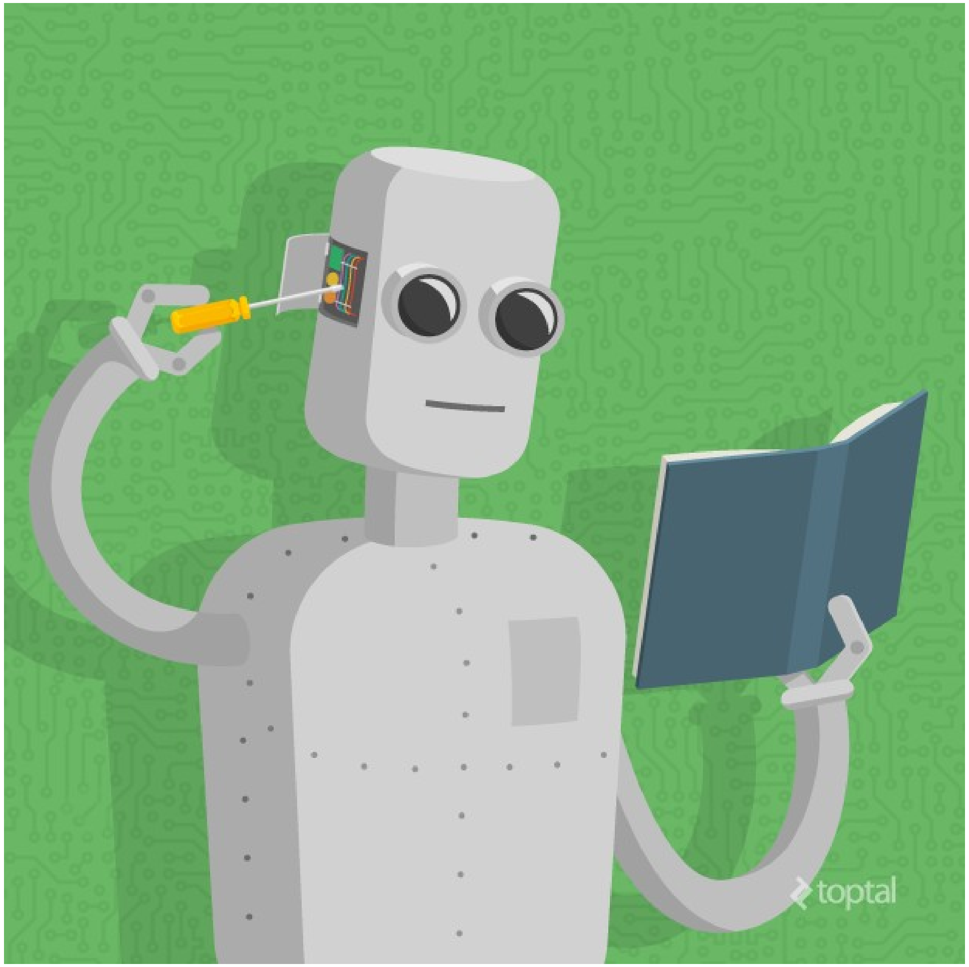
(c) Image from Toptal.com

What is machine learning?

What is machine learning?
(computational intelligence)
(data mining)
(pattern recognition)

”a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.“

(Tom Mitchell, 1997)



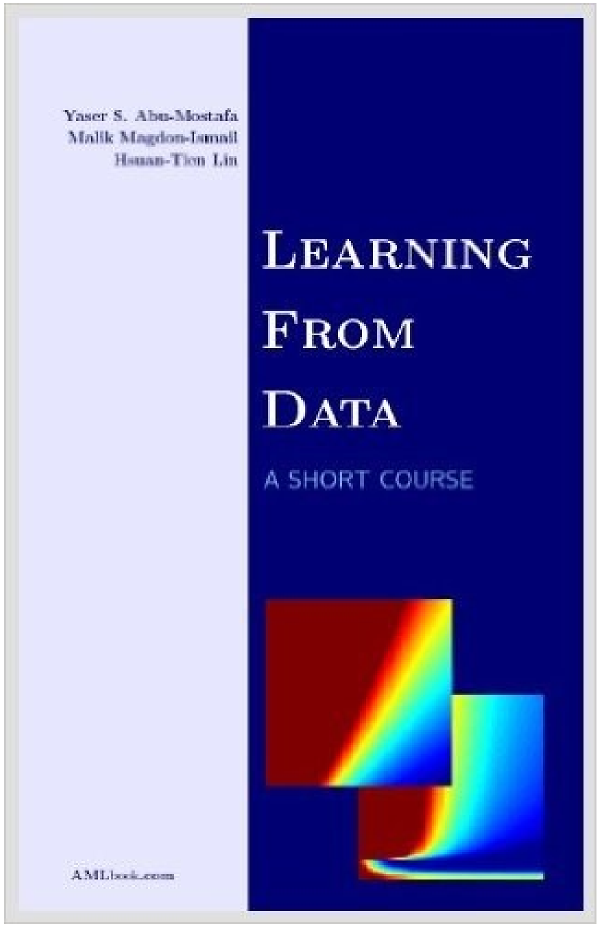
(c) Image from Toptal.com

What is machine learning?

What is machine learning?
(computational intelligence)
(data mining)
(pattern recognition)

"Machine learning [is] the technology that enables computational systems to adaptively improve their performance with experience accumulated from the observed data"

(Yaser Abu-Mostafa, 2012)



Learning from data

Machine learning example: series of number

1 2 4 8 16 32 ...

what is the next number?

Learning from data

Machine learning example: series of number

1 2 4 8 16 32 ...

what is the next number?

64

Learning from data

Machine learning example: series of number

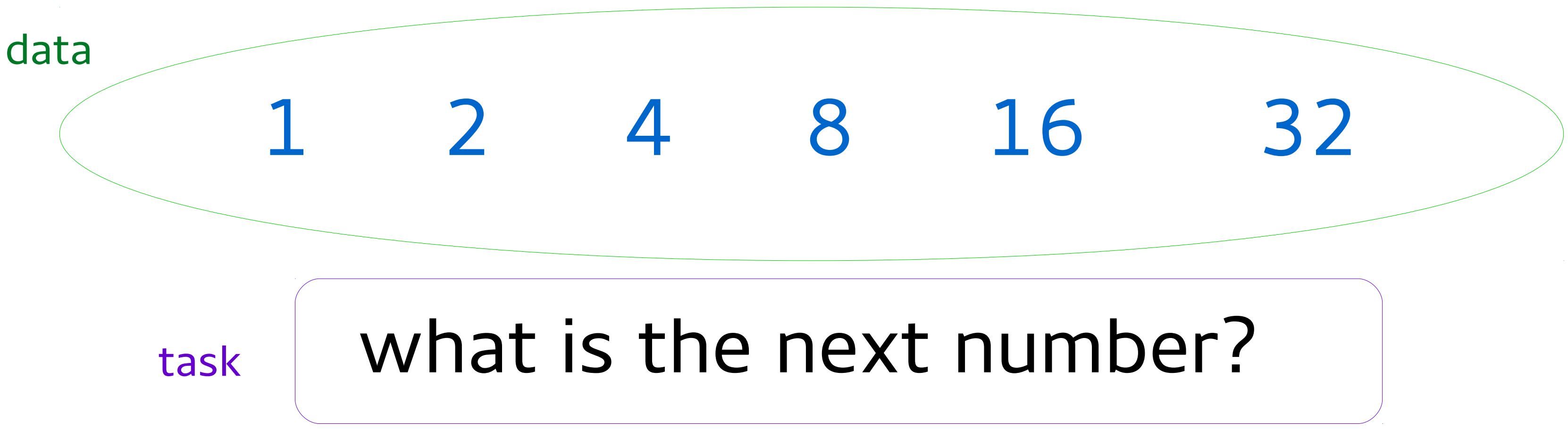
data

1 2 4 8 16 32

what is the next number?

Learning from data

Machine learning example: series of number



data

1 2 4 8 16 32

task

what is the next number?

Learning from data

Machine learning example: series of number

data

1 2 4 8 16 32

task

what is the next number?

prediction 64

Machine learning algorithm

Hyper-parameters

- These parameters express “higher-level” properties of the model such as its complexity or how fast it should learn. Hyperparameters are usually fixed before the actual training process begins.

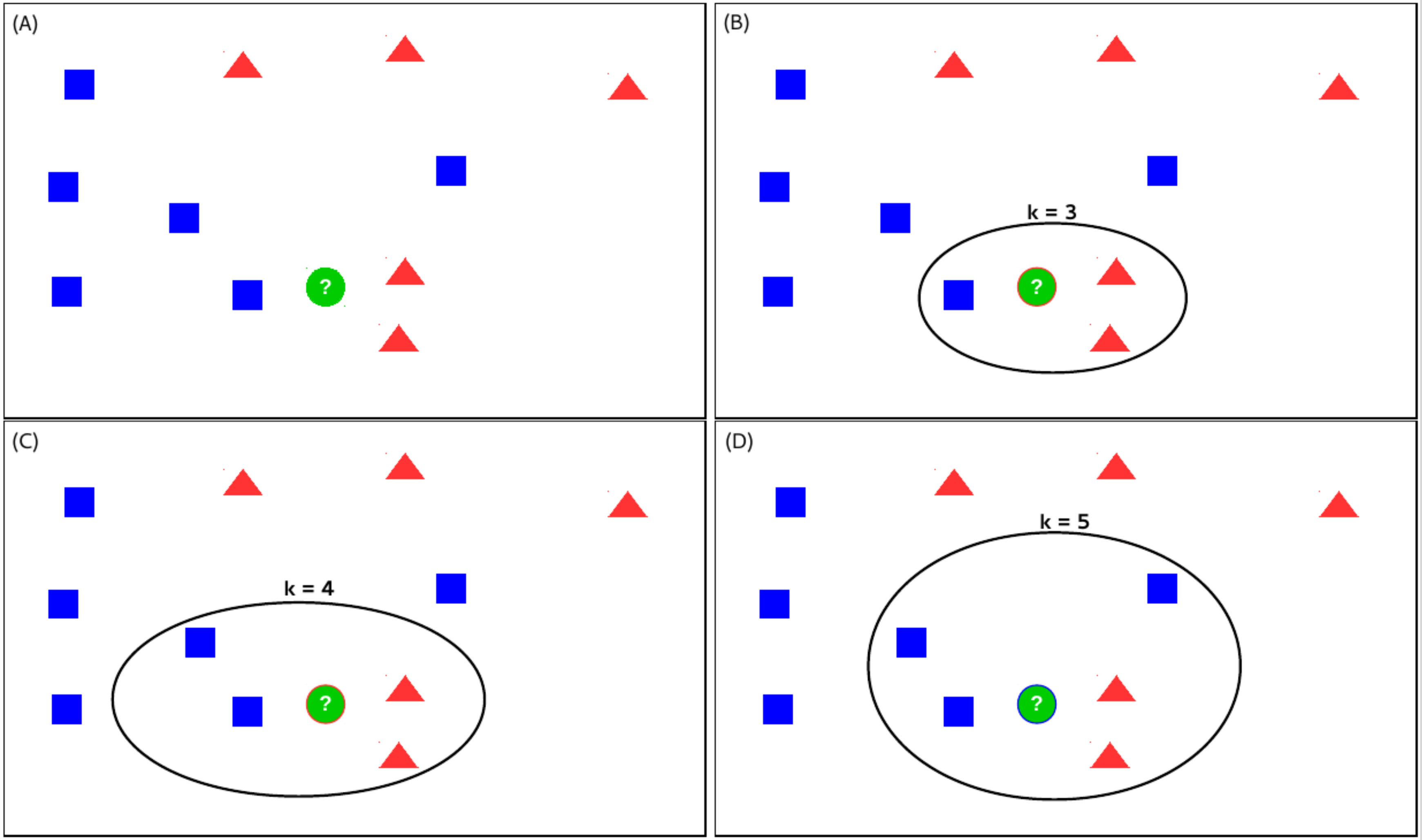
- Their values can strongly influence the performance and the results of the machine learning algorithm application

- Examples:
  - Number k of clusters in *k*-nearest neighbors

Machine learning algorithm

Hyper-parameters

- Number k of clusters in k-nearest neighbors



Machine learning algorithm

Hyper-parameters

- These parameters express “higher-level” properties of the model such as its complexity or how fast it should learn. Hyper-parameters are usually fixed before the actual training process begins.
- Their values can strongly influence the performance and the results of the machine learning algorithm application
- Examples: number of clusters in k-nearest neighbors

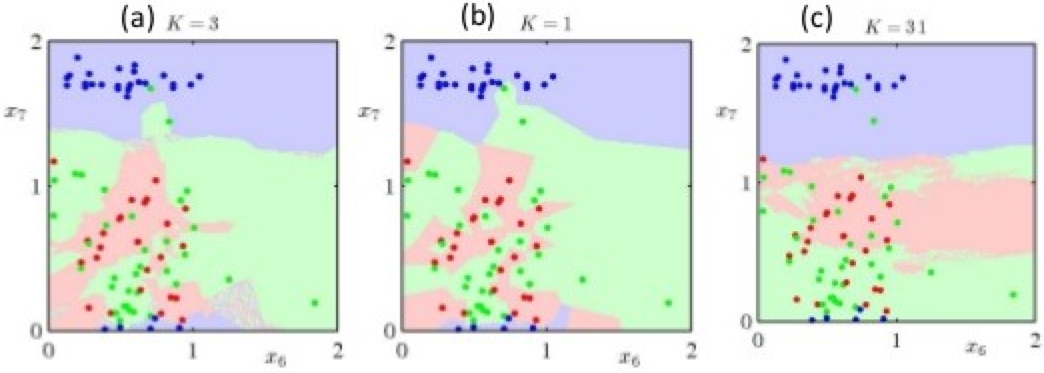


Fig. 9 K-Nearest Neighbor Classifiers (K=3, 1, 31)   Bishop [3]

Machine learning algorithm

Hyper-parameters

- Finding the best values for the hyper-parameters is a key point in machine learning

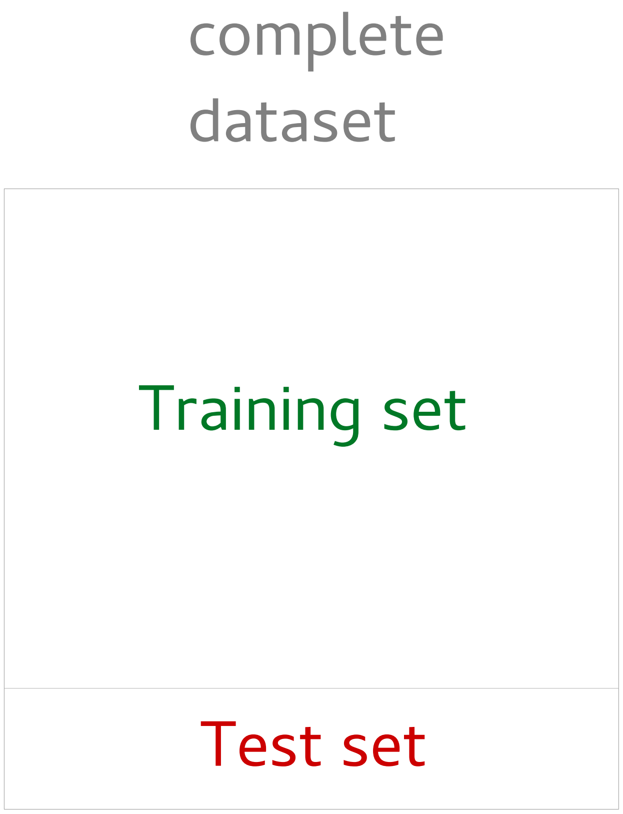
- Usually, the the best practice is a grid search on all the possible values (or most of them), on an independent subset

Dataset arrangement

If you have a machine learning algorithm already optimized (where there are no hyper-parameters to tune), you have to split the dataset into 2 subsets:

1 - a training set, used **only** to train the algorithm
(usually the 80% of the available dataset)

2 - a test set, used **only** to test the algorithm
(usually the 20% of the available dataset)



complete
dataset

Training set

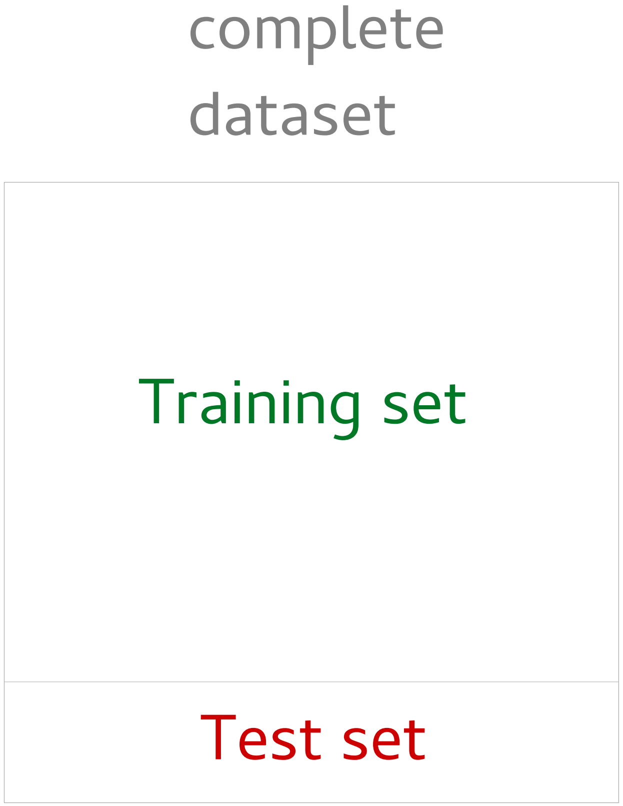
Test set

Dataset arrangement

If you have a machine learning algorithm already optimized (where there are no hyper-parameters to tune), you have to split the dataset into 2 subsets:

1 - a training set, used **only** to train the algorithm
(usually the 80% of the available dataset)

2 - a test set, used **only** to test the algorithm
(usually the 20% of the available dataset)



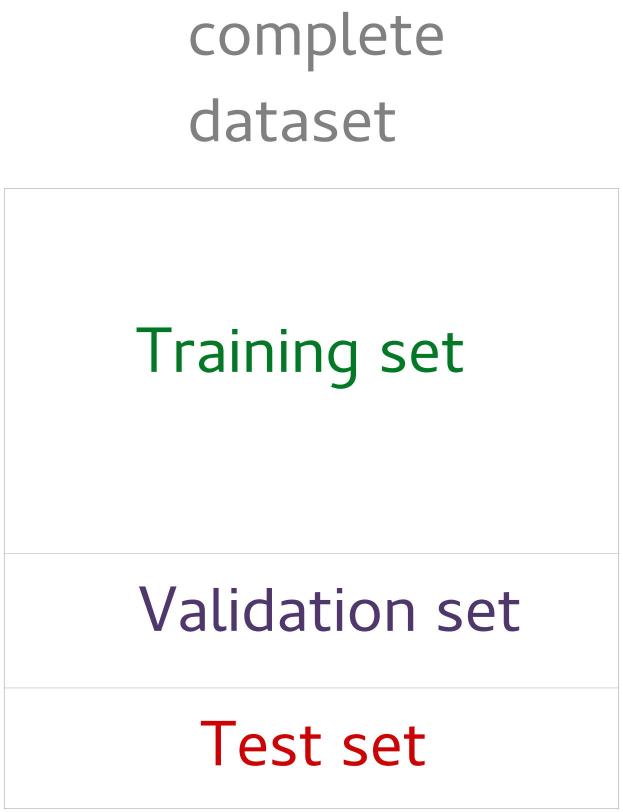
Dataset arrangement

But if you have a machine learning algorithm to optimize (where you have to select the best hyper-parameters), you have to split the dataset into:

1 - a training set, used **only** to train the algorithm (usually the 60% of the available dataset)

2 - a validation set, used **only** to evaluate the trained algorithm model and its hyper-parameters (usually the 20% of the available dataset)

3 - a test set, used **only** to test the algorithm (usually the 20% of the available dataset)



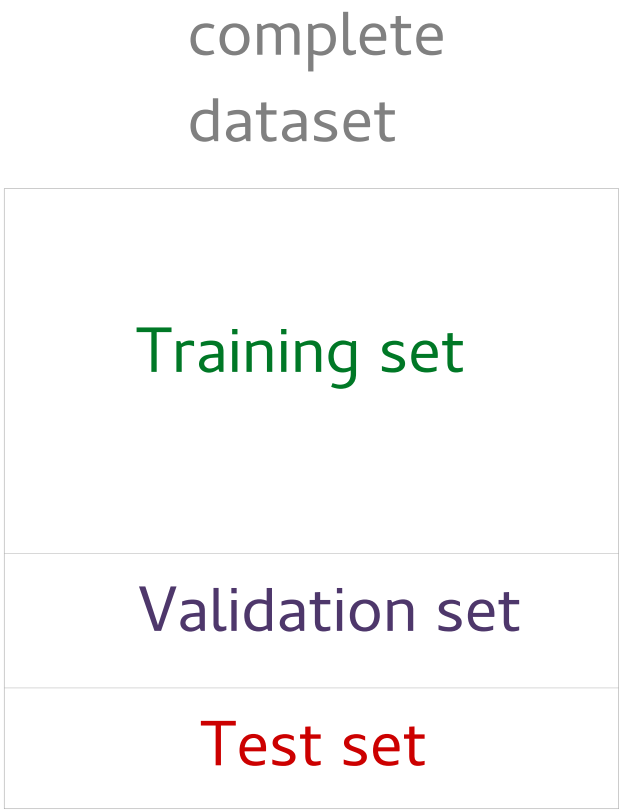
Dataset arrangement

But if you have a machine learning algorithm to optimize (where you have to select the best hyper-parameters), you have to split the dataset into:

1 - a training set, used **only** to train the algorithm (usually the 60% of the available dataset)

2 - a validation set, used **only** to evaluate the trained algorithm model and its hyper-parameters (usually the 20% of the available dataset)

3 - a test set, used **only** to test the algorithm (usually the 20% of the available dataset)



complete dataset

Training set

Validation set

Test set

Dataset arrangement

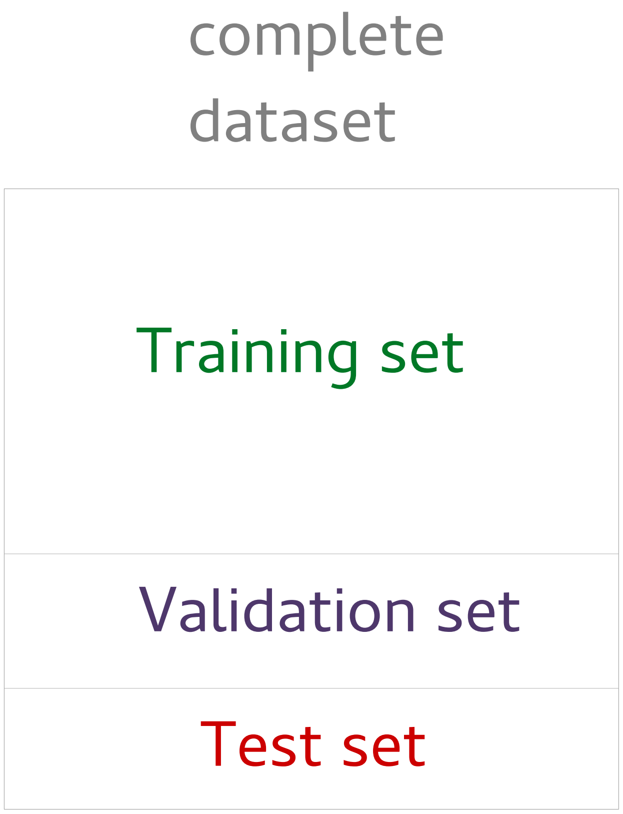
Example, suppose you have an artificial neural network and you have to decide its hyper-parameters (what number of hidden layers and hidden units)

1 – choose a new configuration of hyper-parameters, then train on the training set

2 – after training, evaluate your model by applying it to the validation set

3 – if the evaluation on the validation set led to sufficient accuracy (e.g. MCC >= 0.5), apply the trained model to the test set

– else: go back to point 1



complete dataset

Training set

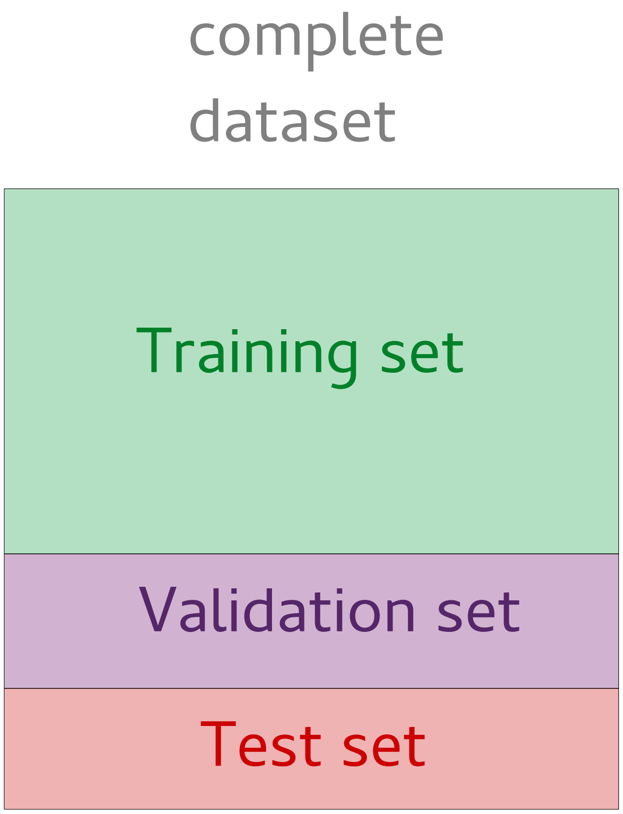
Validation set

Test set

Dataset arrangement

IMPORTANT: THESE SUBSETS MUST ALWAYS BE INDEPENDENT!!!

SO NO INTERSECTIONS!!! 🚫



A data intersection between these subsets will completely invalidate and corrupt your procedure

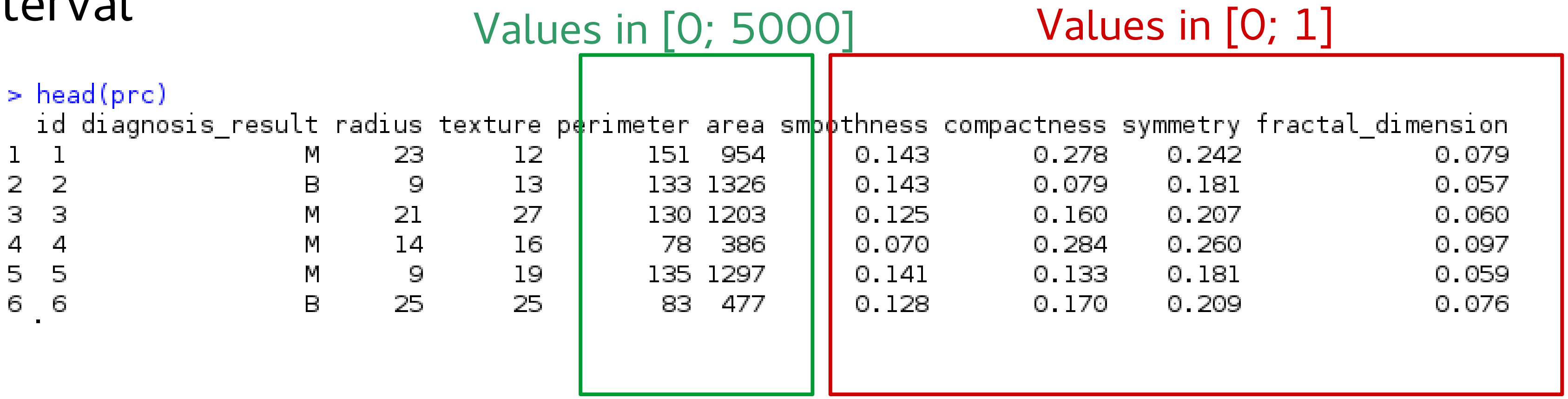
Data engineering is often the key!

- Often the success of a machine learning algorithm is not the algorithm, but the data engineering (or feature engineering)

- Often gathering data, integrating them, cleaning them and pre-processing them might be the key for success

- Why? It's fundamental to add knowledge and expertise about the domain, and to prepare a dataset "ready" to solve a specific problem

- Often, for example, it's necessary to normalize the data into the [0, 1] interval



```
> head(prc)
  id diagnosis_result radius texture perimeter area smoothness compactness symmetry fractal_dimension
1  1                M     23      12       151  954      0.143       0.278    0.242             0.079
2  2                B      9      13       133 1326      0.143       0.079    0.181             0.057
3  3                M     21      27       130 1203      0.125       0.160    0.207             0.060
4  4                M     14      16        78  386      0.070       0.284    0.260             0.097
5  5                M      9      19       135 1297      0.141       0.133    0.181             0.059
6  6                B     25      25        83  477      0.128       0.170    0.209             0.076
```

Machine learning problems

Supervised learning

- we have training data with labels of the correct answers
- use training data to prepare the algorithm

Unsupervised learning

- no training data labels
- what to learn: interesting associations in the data
- often there is no single correct answer

Reinforcement learning

- continuous interaction from the environment

Machine learning problems

Supervised learning

- we have training data with labels of the correct answers
- use training data to prepare the algorithm

Unsupervised learning

- no training data labels
- what to learn: interesting associations in the data
- often there is no single correct answer

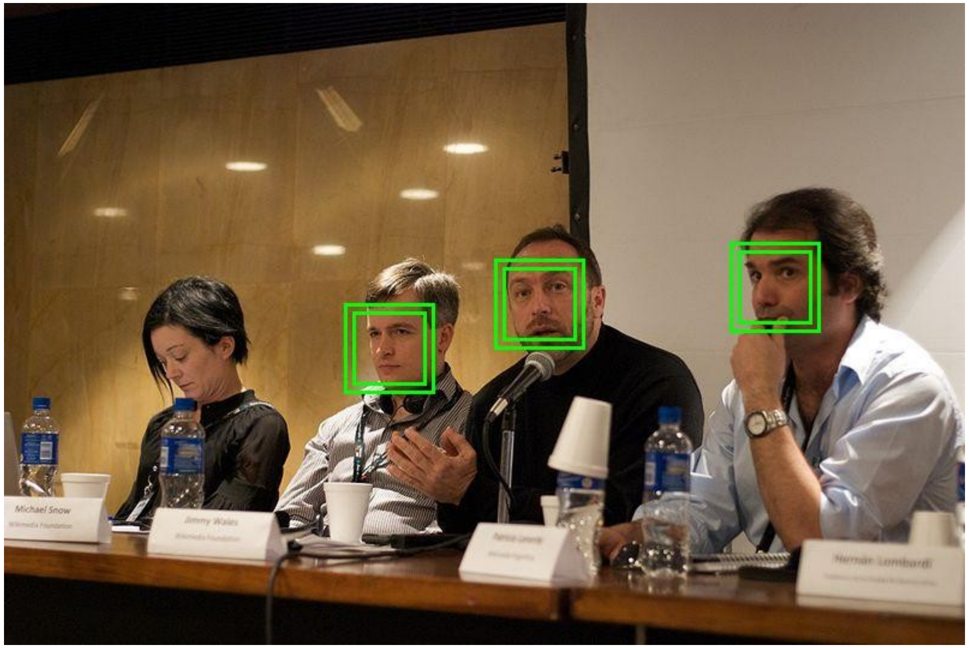
Reinforcement learning

- continuous interaction from the environment

Machine learning problems

Supervised learning

- we have training data with labels of the correct answers
- use training data to prepare the algorithm



Example: face detection (Image from CreativeCommons.org)

Machine learning problems

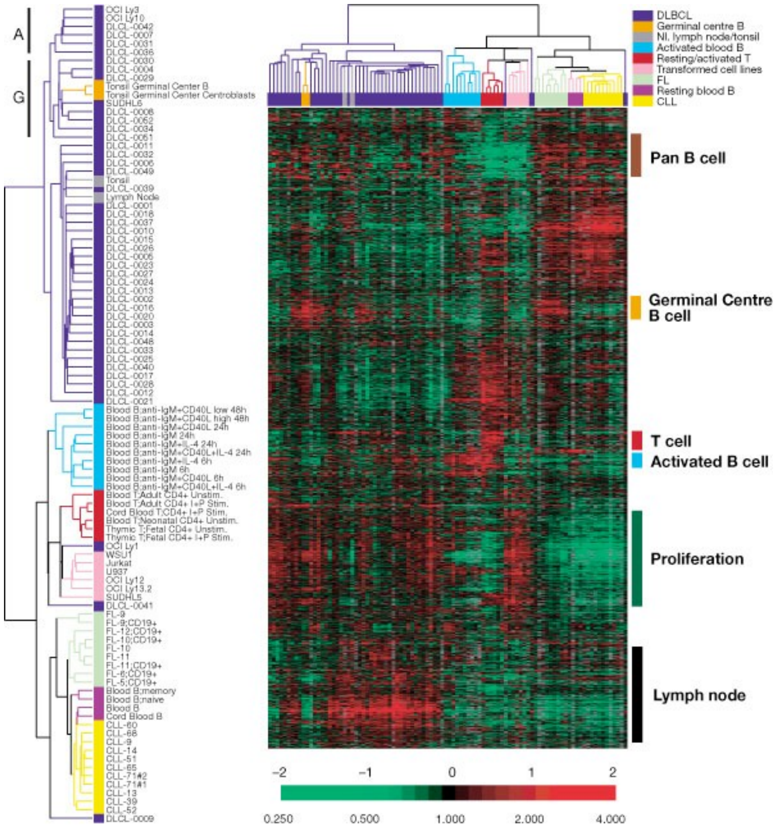
Unsupervised learning

- no training data labels
- what to learn: interesting associations in the data
- often there is no single correct answer

Example: gene expression data clustering

Activity levels of gene expression measured in lymphoma patients

Cluster analysis determined three different subtypes (where only two were known before), having different clinical outcomes



Machine learning problems

Reinforcement learning

- continuous interaction from the environment

Example: stock exchange data (Image from CreativeCommons.org)

Machine learning problems

WE FOCUS ON THIS TODAY

Supervised learning

- we have training data with labels of the correct answers
- use training data to prepare the algorithm

Unsupervised learning

- no training data labels
- what to learn: interesting associations in the data
- often there is no single correct answer

Reinforcement learning

- continuous interaction from the environment

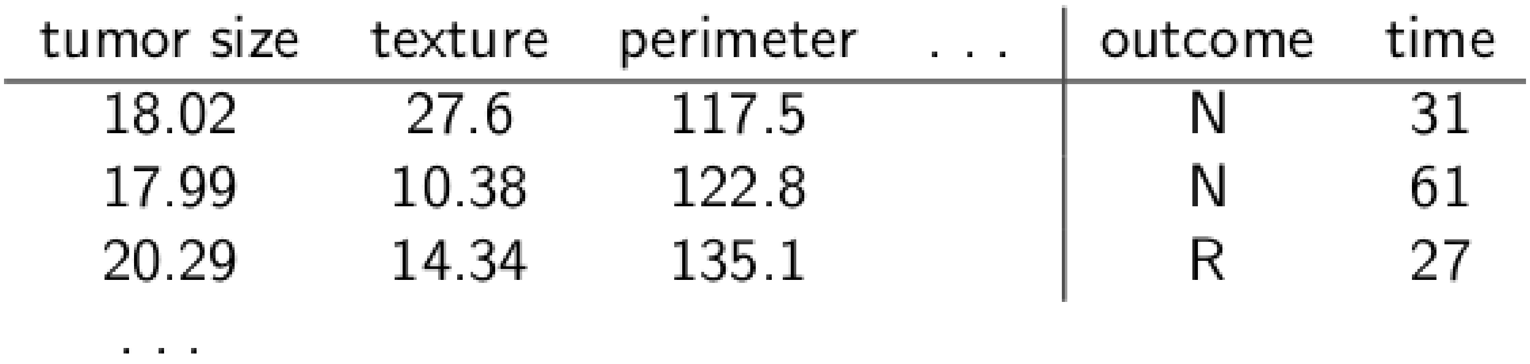
Dictionary

Example: tumor dataset

- Cell samples were taken from tumors in breast cancer patients before surgery, and imaged
  - Tumors were excised
  - Patients were followed to determine whether or not the cancer recurred, and how long until recurrence or disease free

- 32 real-valued variables per tumor.
- 2 variables that can be predicted:
  – Outcome (R=recurrence, N=non-recurrence)
  – Time (until recurrence, for R, time healthy, for N)

Dictionary



tumor size	texture	perimeter	. . .	outcome	time
18.02	27.6	117.5		N	31
17.99	10.38	122.8		N	61
20.29	14.34	135.1		R	27
. . .					

Example: tumor dataset

- Columns are called input variables or features or attributes
- The outcome and time (which we are trying to predict) are called output variables or targets
- A row in the table is called training example or instance
- The whole table is called dataset
- The problem of predicting the recurrence is called (binary) classification
- The problem of predicting the time is called regression

Problem definition

Supervised learning problem

- Let X denote the space of input values
- Let Y denote the space of output values
- Given a data set $D \subset X \times Y$, find a function: $h : X \rightarrow Y$
- such that h(x) is a good predictor for the value of y

- h is called a hypothesis

- Problems are categorized by the type of output domain
  – If $Y = R$ (real numbers set), this problem is called regression
  – If Y is a categorical variable (i.e., part of a finite discrete set), the problem is called classification
  – In general, Y could be a lot more complex (graph, tree, etc), which is called structured prediction

Steps solving a supervised learning problem

Steps for solution:

1) Decide what the input-output pairs are (e.g. input: gene expression data of patients; output: healthy/unhealthy patients)

2) Decide how to encode inputs and outputs. This defines the input space X, and the output space Y. (e.g. input: table of real values; output: vector of 0/1 values, which correspond to false/true)

3) Choose a class of hypotheses/representations H (e.g. the patients' status can be inferred through a method which clusters the input gene expression data)

Steps solving a supervised learning problem

Steps for solution:

4) Choose an error function (cost function) to define the best hypothesis (e.g. MSE: mean square error: ||predictedValue(i) – valueTarget(i)||ˆ2)

5) Choose an algorithm for searching efficiently through the space of hypotheses (linked to (3): e.g. choose k-nearest neighbors, or k-means)

Overfitting

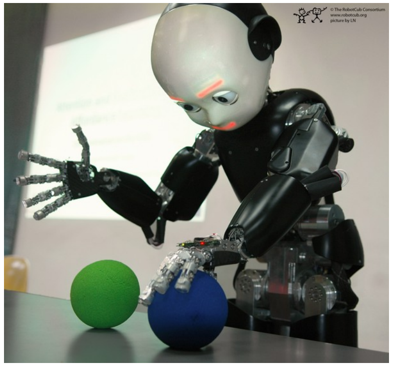
Very important problem

- Overfitting happens when an algorithm adapts "too much" to the training set, and so then performs very badly in the validation set and in the test set

- The algorithm gets somehow "hallucinated" by the training set

- E.g. suppose you train a robot to recognize plants, and then it "thinks" that everything is a plant

Training: "This is a plant"


© The RobotCub Consortium
www.robotcub.org
picture by LN

Testing: "This is a plant" WRONG

Overfitting

Very important problem

- An algorithm is well trained if it minimizes the error during training and if it is able to generalize well in the validation set and test set

- Some (not definitive) solutions:

    - Held out approach (as we already said, divide dataset into 3 independent subsets: training set, validation set, test set)

    - Regularization (penalization in the loss function for complex models) [we won't see this here]

    - More data

    - Cross-validation

Overfitting

Very important problem

- An algorithm is well trained if it minimizes the error during training and if it is able to generalize well in the validation set and test set

- Some (not definitive) solutions:
  - Held out approach (as we already said, divide dataset into 3 independent subsets: training set, validation set, test set)
  - Regularization (penalization in the loss function for complex models) [we won't see this here]
  - More data
  - Cross-validation

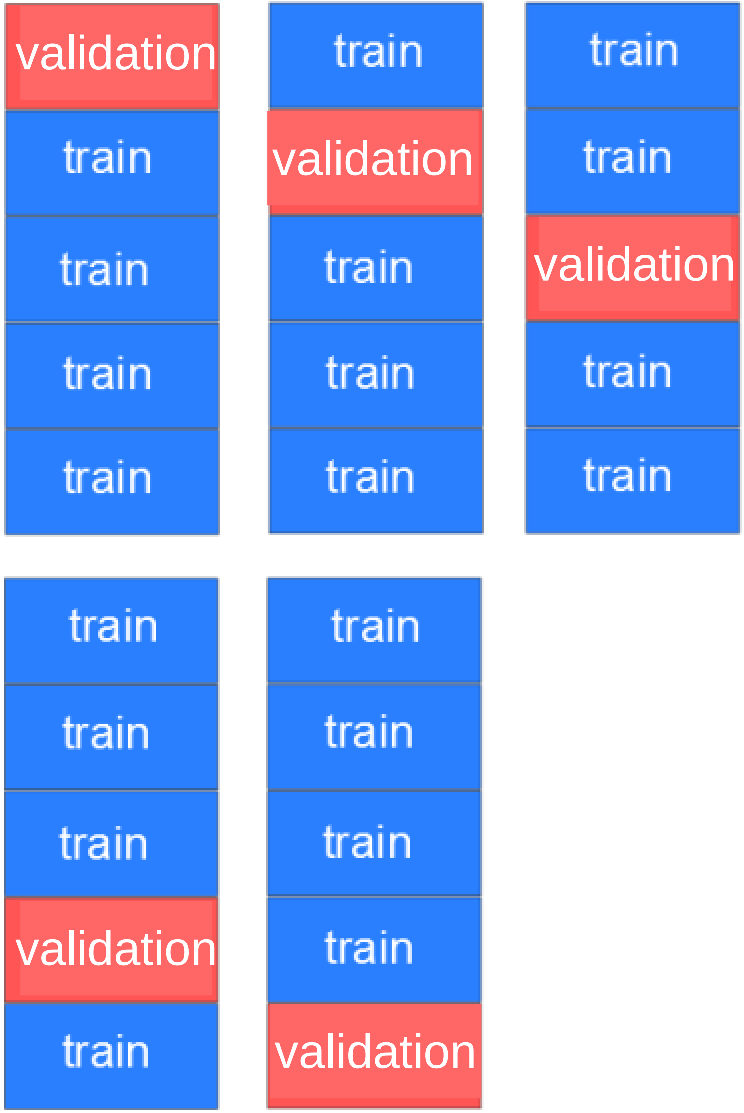
THESE METHODS CAN HELP CONTRASTING OVERITTING

BUT THE CANNOT COMPLETELY SOLVE THE PROBLEM!

Overfitting

Cross validation

- Choose a number of folds (usually k=5)
- Divide the dataset (training set and validation set, excluding the test set) into k folds
- For each ith fold (i=1,...,5):
    - choose the ith fold as validation set
    - choose all the other folds as training set
    - train the model on the training set and evaluate it on the validation set
- Output: all the predictions made for each element of the dataset



validation
train
train
train
train

train
validation
train
train
train

train
train
validation
train
train

train
train
train
validation
train

train
train
train
train
validation

Which machine learning algorithm to choose?

Thumb-rule

Start with a simple algorithm!

If it works, great! You'll have all the parameters and features easily under control.

If it does NOT work, good anyway. You'll have a weak classifier to make comparison with other algorithms.

Which machine learning programming languages?

Go with open source, open access, open science tools

- R (pro: easy to use, especially for beginners; cons: slow, and not suitable for big data)

- Torch (pro: fast, libraries for deep learning; cons: complicated for beginners)

- Python scikit-learn (pro: fast, libraries for many algorithms; cons: complicated for beginners)

Avoid proprietary software (e.g. MATLAB)!

- you or your institution has to pay a license; if you write pieces of code in that language, and then you have to change job, or collaborate with someone who does not have the license, you will not be able to use your code again!

Outline

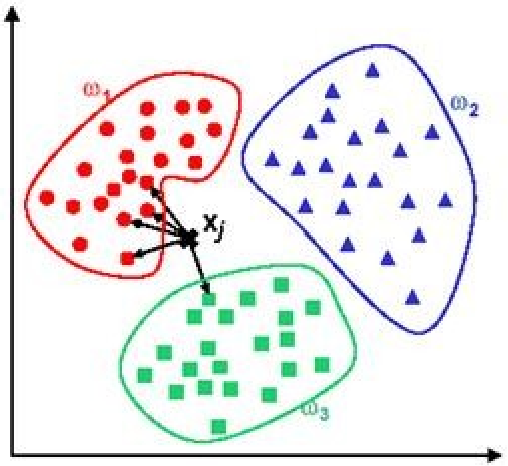
Session 2 – Practice

- 2a - Introduction to k-nearest neighbors (k-NN)
- 2b - Exercise in R. Usage of k-NN for binary classification of cancer-related data

k-nearest neighbors algorithm

k-NN

- 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 segregates unlabeled data points into well defined groups



(c) Byclb.com

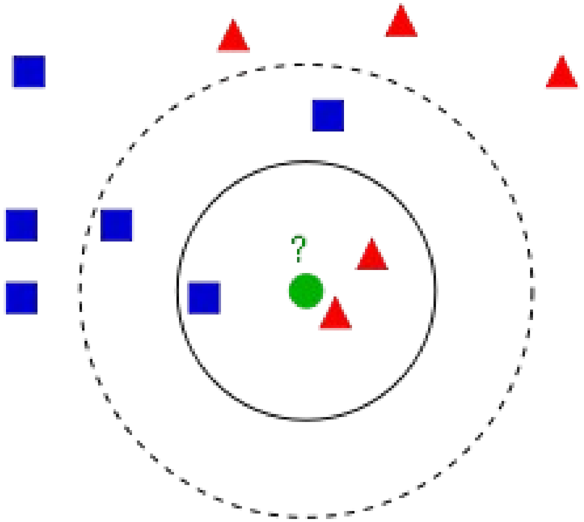
k-nearest neighbors algorithm

k-NN

- k: hyper-parameter that represents the number of neighbors to consider

- The selection of k will determine how well the data can be utilized to generalize the results of the kNN algorithm. A large k value has benefits which include reducing the variance due to the noisy data; the side effect being developing a bias due to which the learner tends to ignore the smaller patterns which may have useful insights.

(c) Analytics Vidhya



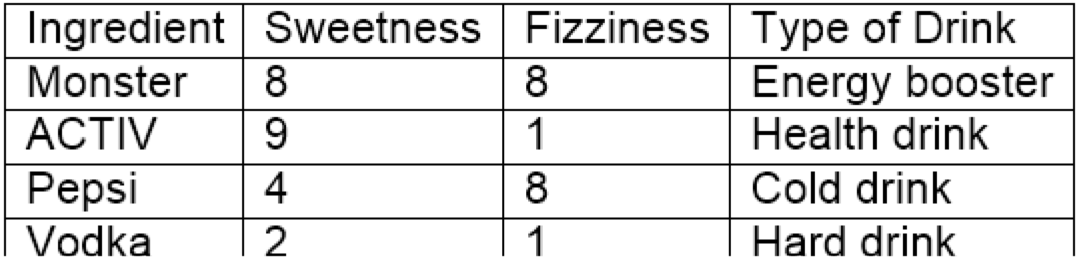
Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

(c) Wikipedia

k-nearest neighbors algorithm

k-NN example

- Let's consider 10 'drinking items' which are rated on two parameters on a scale of 1 to 10. The two parameters are "sweetness" and "fizziness". This is more of a perception based rating and so may vary between individuals. I would be considering my ratings (which might differ) to take this illustration ahead. The ratings of few items look somewhat as:

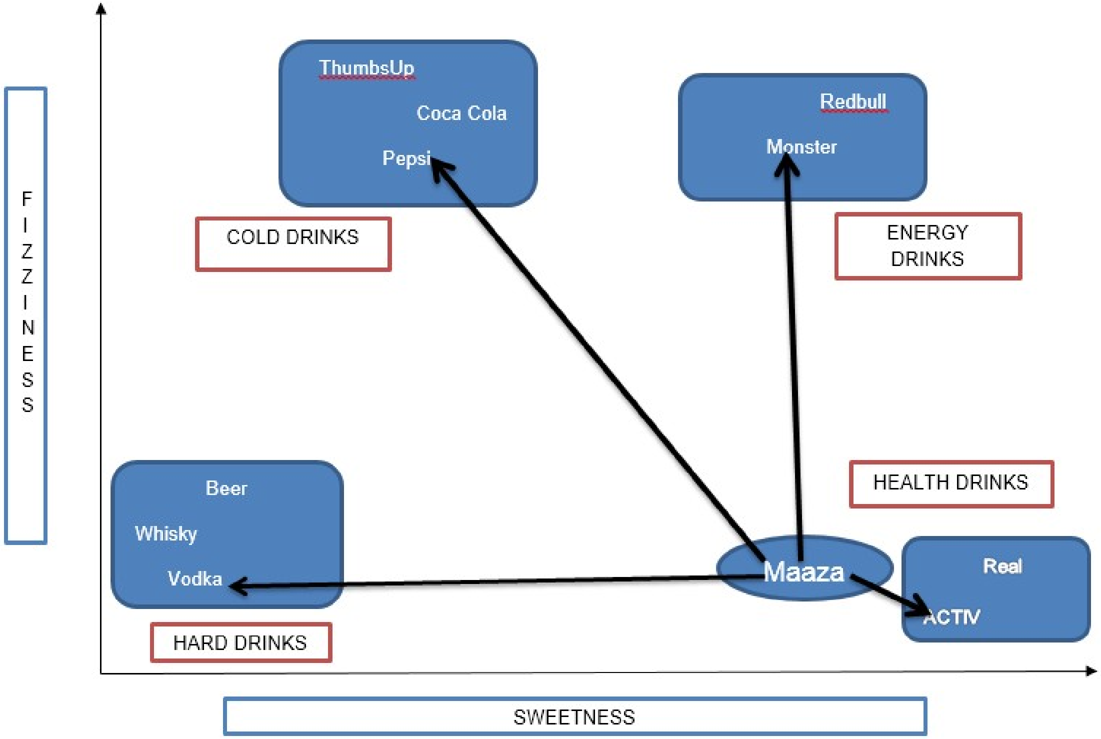


Ingredient	Sweetness	Fizziness	Type of Drink
Monster	8	8	Energy booster
ACTIV	9	1	Health drink
Pepsi	4	8	Cold drink
Vodka	2	1	Hard drink
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…

k-nearest neighbors algorithm

k-NN example

- From the above figure, it is clear we have bucketed the 10 items into 4 groups namely, 'COLD DRINKS', 'ENERGY DRINKS', 'HEALTH DRINKS' and 'HARD DRINKS'. The question here is, to which group would 'Maaza' fall into? This will be determined by calculating distance.

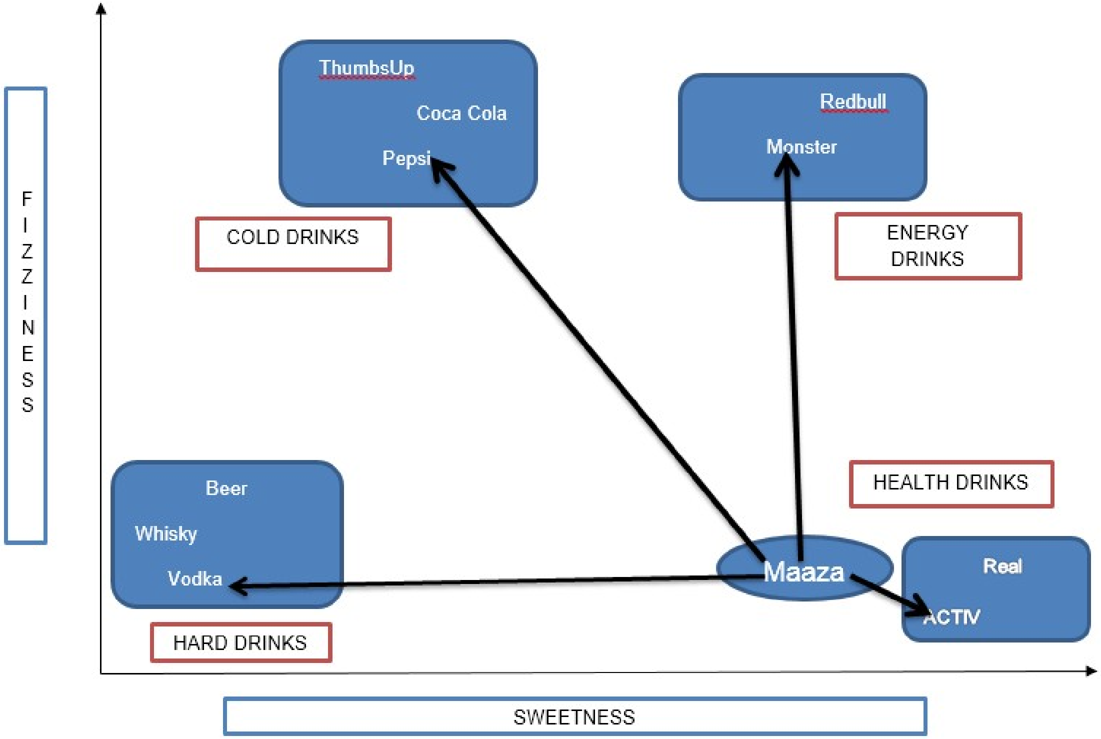


(c) Analytics Vidhya

k-nearest neighbors algorithm

k-NN example

- "Sweetness" determines the perception of the sugar content in the items. "Fizziness" ascertains the presence of bubbles in the drink due to the carbon dioxide content in the drink. Again, all these ratings used are based on personal perception and are strictly relative.

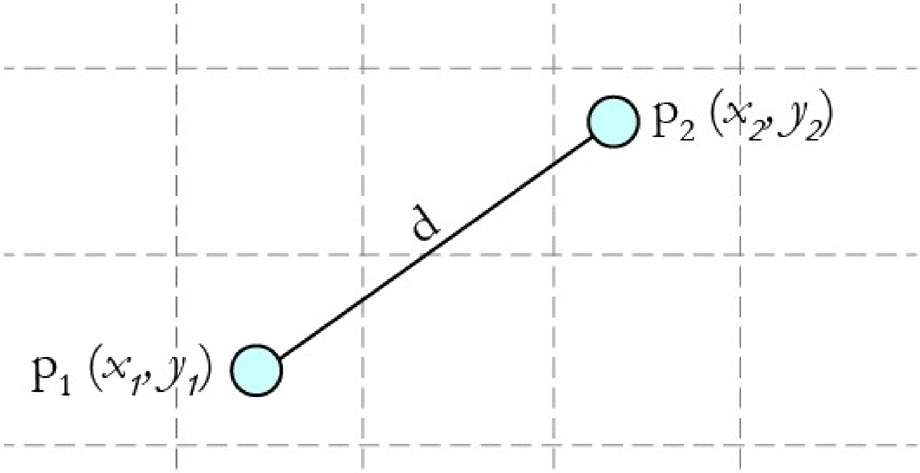


(c) Analytics Vidhya

k-nearest neighbors algorithm

k-NN example: calculating distance

- Now, calculating distance between ‘Maaza’ and its nearest neighbors (‘ACTIV’, ‘Vodka’, ‘Pepsi’ and ‘Monster’) requires the usage of a distance formula, the most popular being Euclidean distance formula i.e. the shortest distance between the 2 points which may be obtained using a ruler.

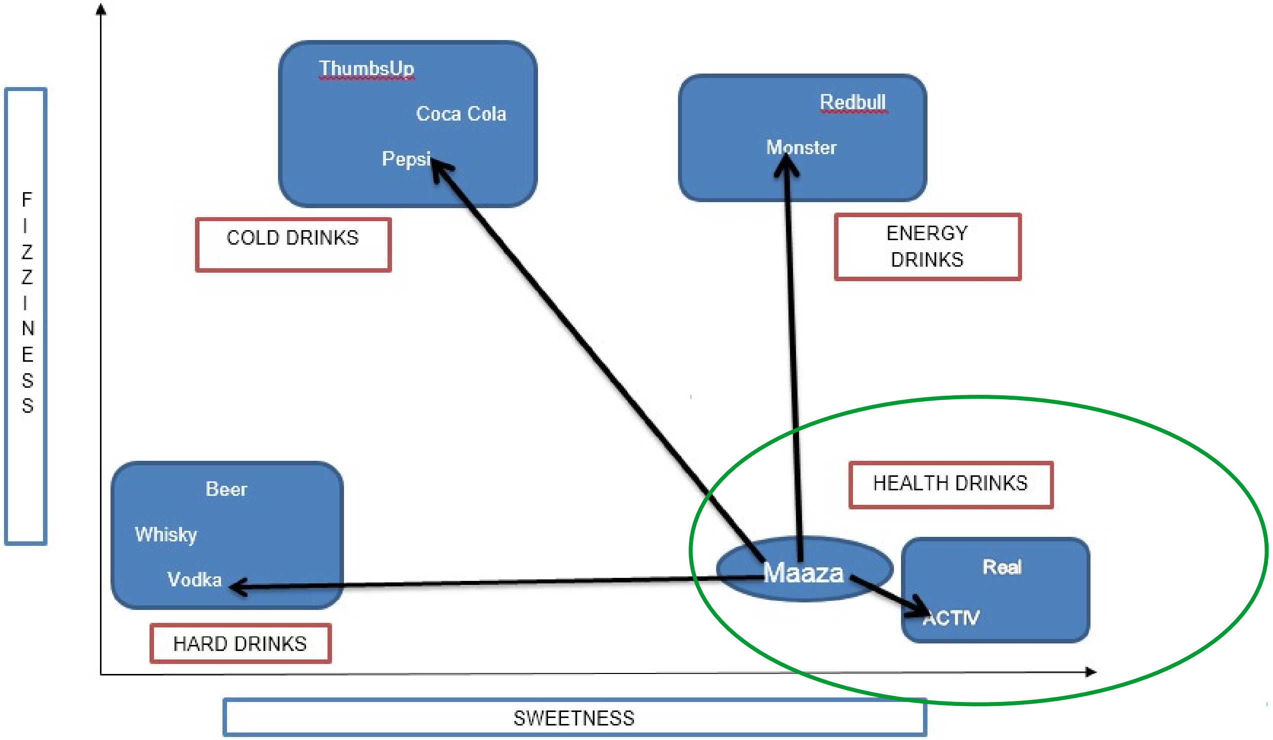


$$\text{Euclidean distance (d)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

k-nearest neighbors algorithm

k-NN example: calculating distance

- If *k*=1, the algorithm will consider only 1 nearest neighbor. The nearest neighbor to Maaza is ACTIV, so the algorithm will assign Maaza to the HEALTHY DRINKS cluster

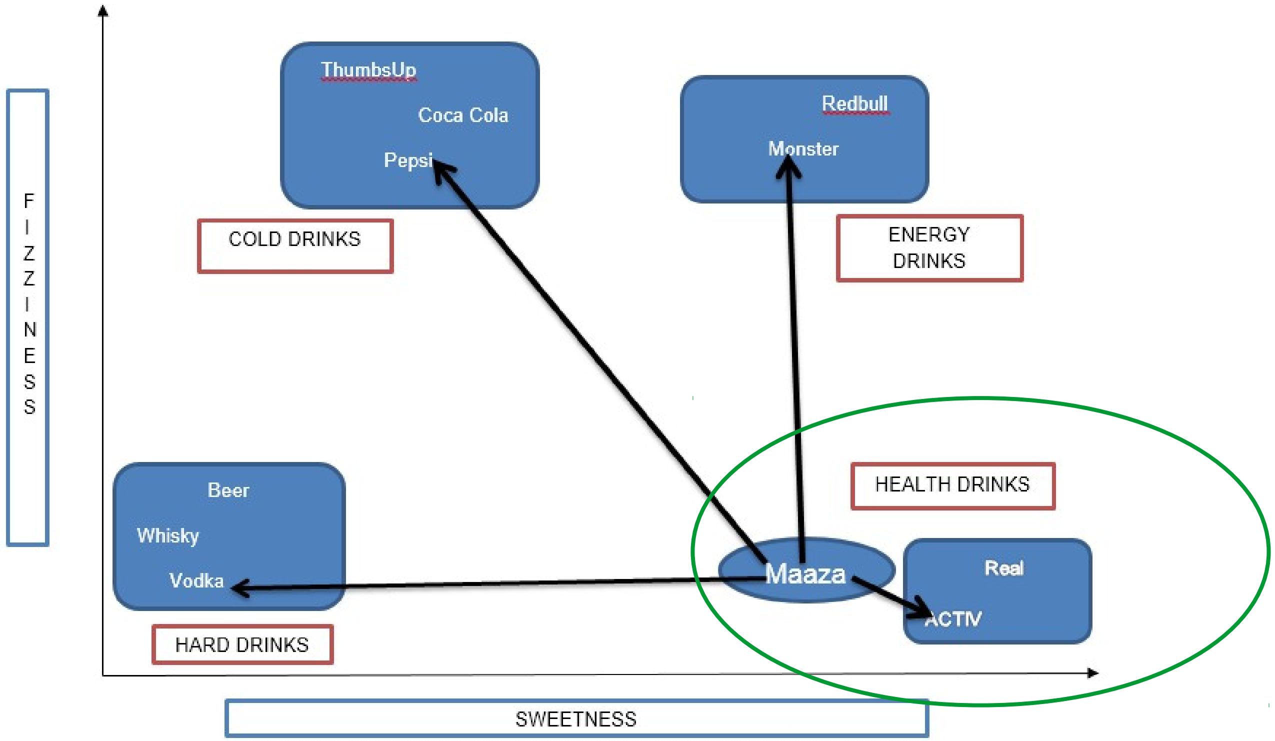


(c) Analytics Vidhya

k-nearest neighbors algorithm

k-NN example: calculating distance

- If *k*=2, the algorithm will consider 2 nearest neighbors. The nearest neighbors to Maaza are ACTIV and Real, so the algorithm will assign Maaza to the HEALTHY DRINKS cluster, again

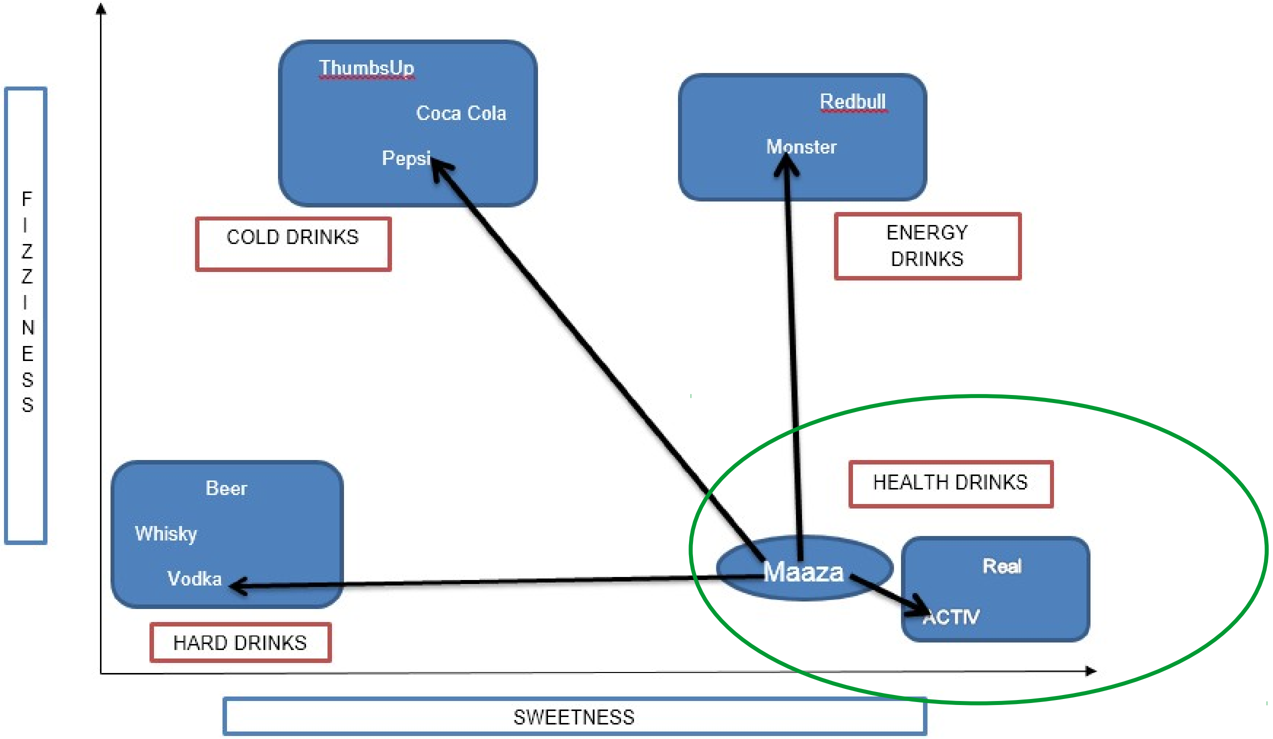


(c) Analytics Vidhya

k-nearest neighbors algorithm

k-NN example: calculating distance

- If *k*=3, the algorithm will consider 3 nearest neighbors. The nearest neighbors to Maaza are ACTIV, Real (HEALTHY DRINKS), and Monster (ENERGY DRINKS). The algorithm will assign Maaza to the HEALTHY DRINKS cluster, again (most frequent class among the 3 neighbors)



(c) Analytics Vidhya

k-nearest neighbors algorithm

Practical session with R

We are going to apply the *k*-nearest neighbors algorithm to classify cancer data

Machine learning finds extensive usage in pharmaceutical industry especially in detection of oncogenic (cancer cells) growth. R finds application in machine learning to build models to predict the abnormal growth of cells thereby helping in detection of cancer and benefiting the health system.

Let's see the process of building this model using kNN algorithm in R Programming.

k-nearest neighbors algorithm

Practical session with R – 1, data collection

We will use a data set of 100 patients (created solely for the purpose of practice) to implement the *k*-nn algorithm and thereby interpreting results .The data set has been prepared keeping in mind the results which are generally obtained from DRE exam.

The data set consists of 100 observations and 10 variables (out of which 8 numeric variables and one categorical variable and is ID) which are as follows: Radius, Texture, Perimeter, Area, Smoothness, Compactness, Symmetry, Fractal dimension

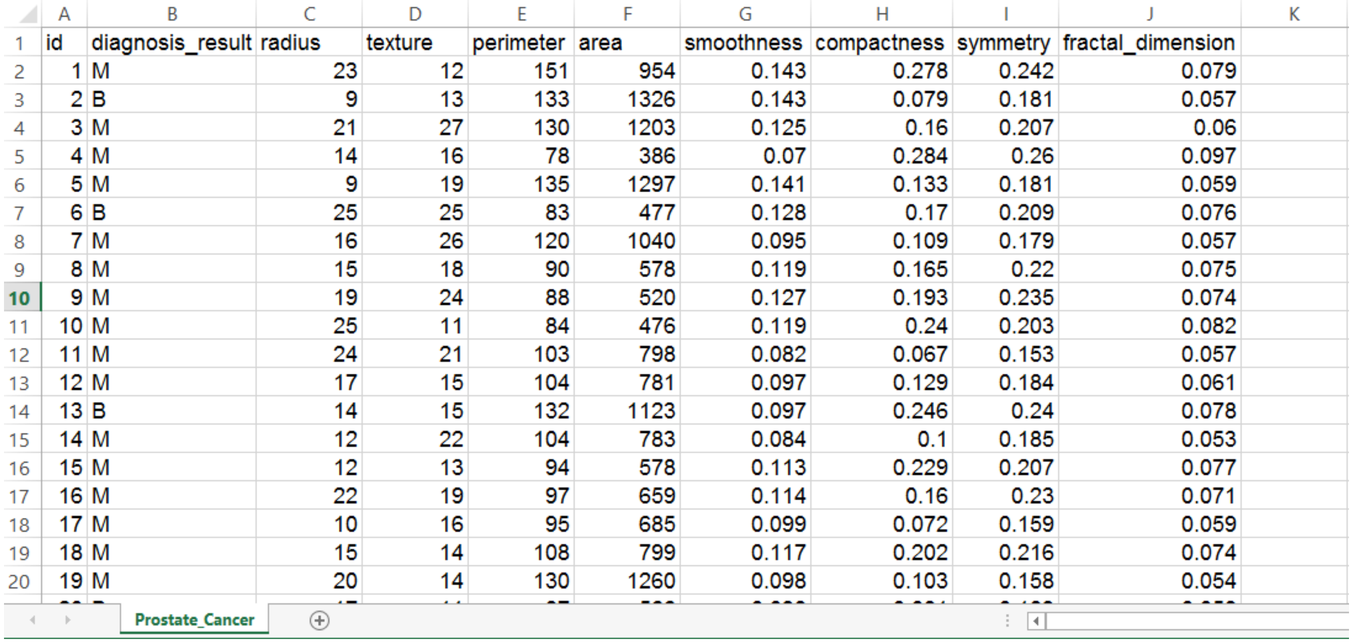
The goal is to classify each instance into Benign or Malignant

The dataset file can be downloaded at: www.bit.ly/prostate_cancer_DRE

k-nearest neighbors algorithm

Practical session with R – 1, data collection

Here's how this data table looks like:

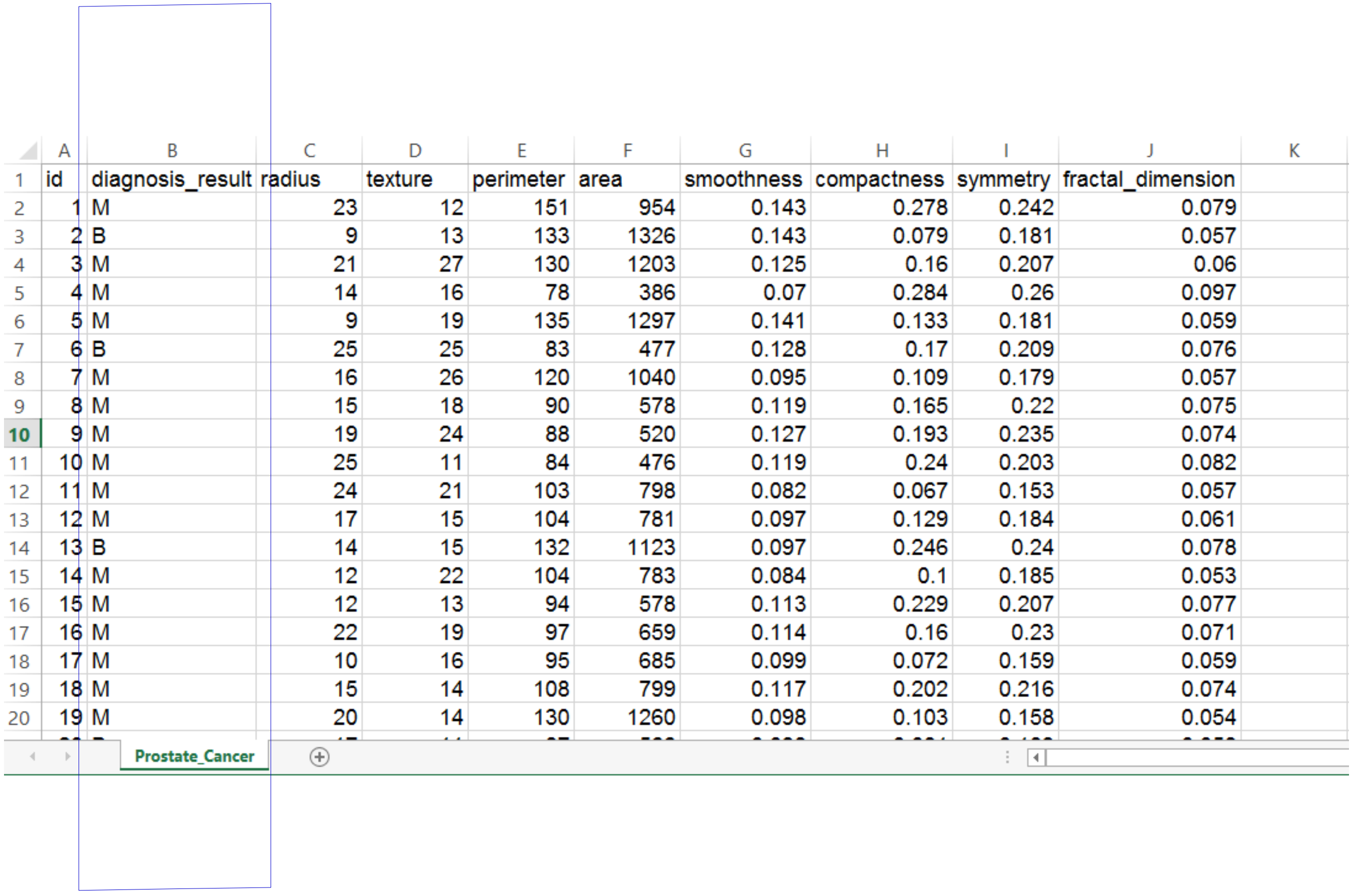


id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
1	M	23	12	151	954	0.143	0.278	0.242	0.079
2	B	9	13	133	1326	0.143	0.079	0.181	0.057
3	M	21	27	130	1203	0.125	0.16	0.207	0.06
4	M	14	16	78	386	0.07	0.284	0.26	0.097
5	M	9	19	135	1297	0.141	0.133	0.181	0.059
6	B	25	25	83	477	0.128	0.17	0.209	0.076
7	M	16	26	120	1040	0.095	0.109	0.179	0.057
8	M	15	18	90	578	0.119	0.165	0.22	0.075
9	M	19	24	88	520	0.127	0.193	0.235	0.074
10	M	25	11	84	476	0.119	0.24	0.203	0.082
11	M	24	21	103	798	0.082	0.067	0.153	0.057
12	M	17	15	104	781	0.097	0.129	0.184	0.061
13	B	14	15	132	1123	0.097	0.246	0.24	0.078
14	M	12	22	104	783	0.084	0.1	0.185	0.053
15	M	12	13	94	578	0.113	0.229	0.207	0.077
16	M	22	19	97	659	0.114	0.16	0.23	0.071
17	M	10	16	95	685	0.099	0.072	0.159	0.059
18	M	15	14	108	799	0.117	0.202	0.216	0.074
19	M	20	14	130	1260	0.098	0.103	0.158	0.054

k-nearest neighbors algorithm

Practical session with R – 1, data collection

Here's how this data table looks like:



id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
1	M	23	12	151	954	0.143	0.278	0.242	0.079
2	B	9	13	133	1326	0.143	0.079	0.181	0.057
3	M	21	27	130	1203	0.125	0.16	0.207	0.06
4	M	14	16	78	386	0.07	0.284	0.26	0.097
5	M	9	19	135	1297	0.141	0.133	0.181	0.059
6	B	25	25	83	477	0.128	0.17	0.209	0.076
7	M	16	26	120	1040	0.095	0.109	0.179	0.057
8	M	15	18	90	578	0.119	0.165	0.22	0.075
9	M	19	24	88	520	0.127	0.193	0.235	0.074
10	M	25	11	84	476	0.119	0.24	0.203	0.082
11	M	24	21	103	798	0.082	0.067	0.153	0.057
12	M	17	15	104	781	0.097	0.129	0.184	0.061
13	B	14	15	132	1123	0.097	0.246	0.24	0.078
14	M	12	22	104	783	0.084	0.1	0.185	0.053
15	M	12	13	94	578	0.113	0.229	0.207	0.077
16	M	22	19	97	659	0.114	0.16	0.23	0.071
17	M	10	16	95	685	0.099	0.072	0.159	0.059
18	M	15	14	108	799	0.117	0.202	0.216	0.074
19	M	20	14	130	1260	0.098	0.103	0.158	0.054

target column

k-nearest neighbors algorithm

Practical session with R – 2, preparing the data

We have to read the dataset file

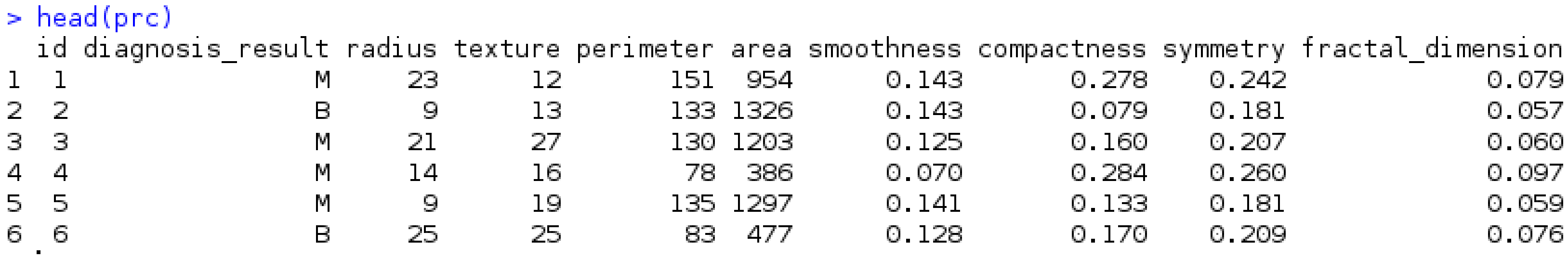
Suppose we have the data file in the data folder:
PATH_TO_DATA/prostate_cancer_DRE_exam_set.csv

```
prc_data <- read.csv("PATH_TO_DATA/prostate_cancer_DRE_exam_set.csv",
stringsAsFactors = FALSE)        # read.csv() imports the required data set and saves it to the prc
data frame. stringsAsFactors = FALSE: helps to convert every character vector to a factor wherever it
makes sense.

str(prc_data)        # We use this command to see whether the data is structured or not.
```

k-nearest neighbors algorithm

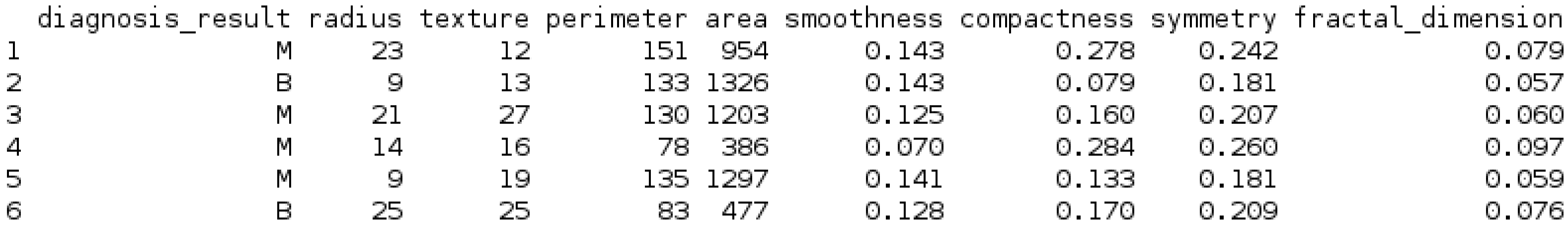
Practical session with R – 2, preparing the data

`head(prc_data)` # to take a look to the first lines of the table



```
> head(prc)
  id diagnosis_result radius texture perimeter area smoothness compactness symmetry fractal_dimension
1  1                M     23      12       151  954      0.143       0.278    0.242             0.079
2  2                B      9      13       133 1326      0.143       0.079    0.181             0.057
3  3                M     21      27       130 1203      0.125       0.160    0.207             0.060
4  4                M     14      16        78  386      0.070       0.284    0.260             0.097
5  5                M      9      19       135 1297      0.141       0.133    0.181             0.059
6  6                B     25      25        83  477      0.128       0.170    0.209             0.076
```

`prc_data <- prc_data[-1]` # removes the first variable(id) from the data set.



```
  diagnosis_result radius texture perimeter area smoothness compactness symmetry fractal_dimension
1                M     23      12       151  954      0.143       0.278    0.242             0.079
2                B      9      13       133 1326      0.143       0.079    0.181             0.057
3                M     21      27       130 1203      0.125       0.160    0.207             0.060
4                M     14      16        78  386      0.070       0.284    0.260             0.097
5                M      9      19       135 1297      0.141       0.133    0.181             0.059
6                B     25      25        83  477      0.128       0.170    0.209             0.076
```

k-nearest neighbors algorithm

Practical session with R – 2, preparing the data

```
prc_data <- prc_data[sample(nrow(prc_data)),]        # we shuffle the rows, to remove any possible rank-related patterns of data
```

```
table(prc_data$diagnosis_result)        # it helps us to get the numbers of patients
```

```
 B      M
38     62
```

k-nearest neighbors algorithm

Practical session with R – 2, normalizing numeric data

This normalization is of paramount importance since the scale used for the values for each variable might be different. The best practice is to normalize the data and transform all the values to a common scale.

```
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x))) }
```

The first variable in our data set (after removal of id) is ‘diagnosis_result’ which is not numeric in nature. So, we start from 2nd variable. The function lapply() applies normalize() to each feature in the data frame. The final result is stored to prc_n data frame using as.data.frame() function

```
prc_data_norm <- as.data.frame(lapply(prc_data[2:9], normalize))
```

Let's check the normalization:

```
summary(prc_data_norm$radius)
```

k-nearest neighbors algorithm

Practical session with R – 2, training set and test set

To simplify this exercise, we heuristically fix k=10, so we do not run any optimization of this hyper-parameter. Because of this decision, we won't split the dataset into 3 subsets (training set, validation set, and test set) as usually we would do, but we will split only into training set, and test set.

We train our k-NN algorithm on training set and test it on test set. For this, we would divide the data set into 2 portions in the ratio of 80% / 20% (assumed) for the training and test data set respectively. You may use a different ratio altogether depending on the problem requirement.

```
training_set_size <- 80
dataset_size <- dim(prc_data_norm)[1]

prc_data_train <- prc_data_norm[1:training_set_size,]
prc_data_test <- prc_data_norm[(training_set_size+1):dataset_size,]
```

Our target variable is ‘diagnosis_result’ which we have not included in our training and test data sets.

```
prc_data_train_labels <- prc_data[1:training_set_size, 1]
prc_data_test_labels <- prc_data[(training_set_size+1):dataset_size, 1]
```

This code takes the diagnosis factor in column 1 of the prc data frame and on turn creates prc_data_train_labels and prc_data_test_labels data frame.

k-nearest neighbors algorithm

Practical session with R – 3, traning the model

Let's now train our model on the training set, and test it on the test data, through the knn() function.

The knn () function needs to be used to train a model for which we need to install a package 'class'. The knn() function identifies the k-nearest neighbors using Euclidean distance where k is a user-specified number.

```
library(class)

K <- 10

prc_data_test_pred <- knn(train = prc_data_train, test = prc_data_test, cl =
prc_data_train_labels, k=K)
```

IMPORTANT: to choose the best value for the hyper-parameter k, we should use an optimization procedure (training on training data; evaluate the model on the validation data; select the model which led to the top performance in the validation data, and apply it to the test data). Here, for simplicity, we select k=10, that is the square root of the number of observations k = sqrt(100) = 10

prc_data_test_pred contains the targets predicted by k-NN for the test set

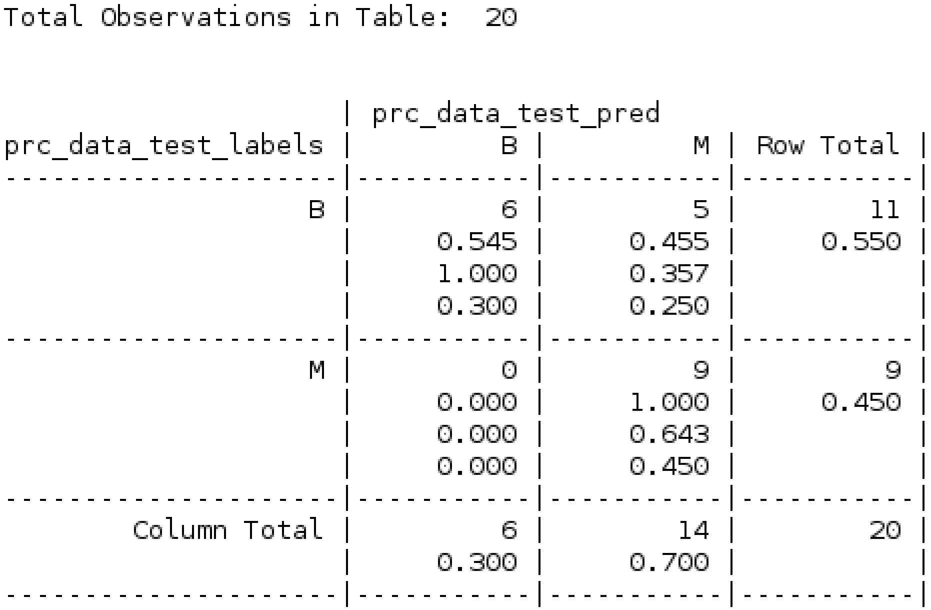
k-nearest neighbors algorithm

Practical session with R – 4, evaluate the model performance

We have built the model but we also need to check the accuracy of the predicted values in prc_data_test_pred as to whether they match up with the known values in prc_data_test_labels. To ensure this, we need to use the CrossTable() function available in the package 'gmodels'.

```
library("gmodels")

CrossTable(x=prc_data_test_labels, y=prc_data_test_pred, prop.chisq=FALSE)
```

The output will be something like this:



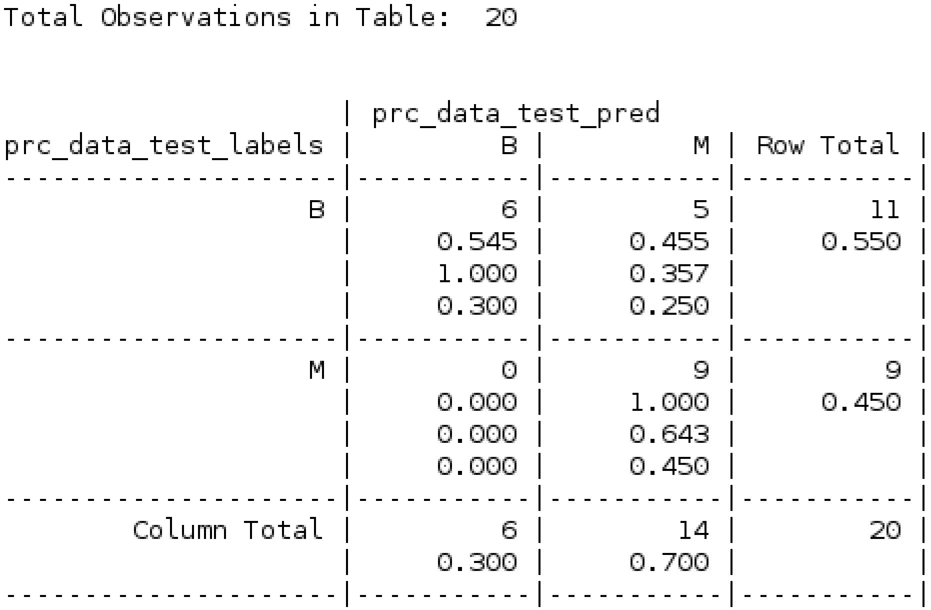
```
Total Observations in Table:  20


                     | prc_data_test_pred
prc_data_test_labels |         B |         M | Row Total |
---------------------|-----------|-----------|-----------|
                   B |         6 |         5 |        11 |
                     |     0.545 |     0.455 |     0.550 |
                     |     1.000 |     0.357 |           |
                     |     0.300 |     0.250 |           |
---------------------|-----------|-----------|-----------|
                   M |         0 |         9 |         9 |
                     |     0.000 |     1.000 |     0.450 |
                     |     0.000 |     0.643 |           |
                     |     0.000 |     0.450 |           |
---------------------|-----------|-----------|-----------|
        Column Total |         6 |        14 |        20 |
                     |     0.300 |     0.700 |           |
---------------------|-----------|-----------|-----------|
```

k-nearest neighbors algorithm

Practical session with R – 4, evaluate the model performance

The test data consisted of 35 observations. Out of which 5 cases have been accurately predicted (TN->True Negatives) as Benign (B) in nature which constitutes 14.3%. Also, 16 out of 35 observations were accurately predicted (TP-> True Positives) as Malignant (M) in nature which constitutes 45.7%. Thus a total of 16 out of 35 predictions where TP i.e, True Positive in nature.

There were no cases of False Negatives (FN) meaning no cases were recorded which actually are malignant in nature but got predicted as benign. The FN's if any poses a potential threat for the same reason and the main focus to increase the accuracy of the model is to reduce FN's.



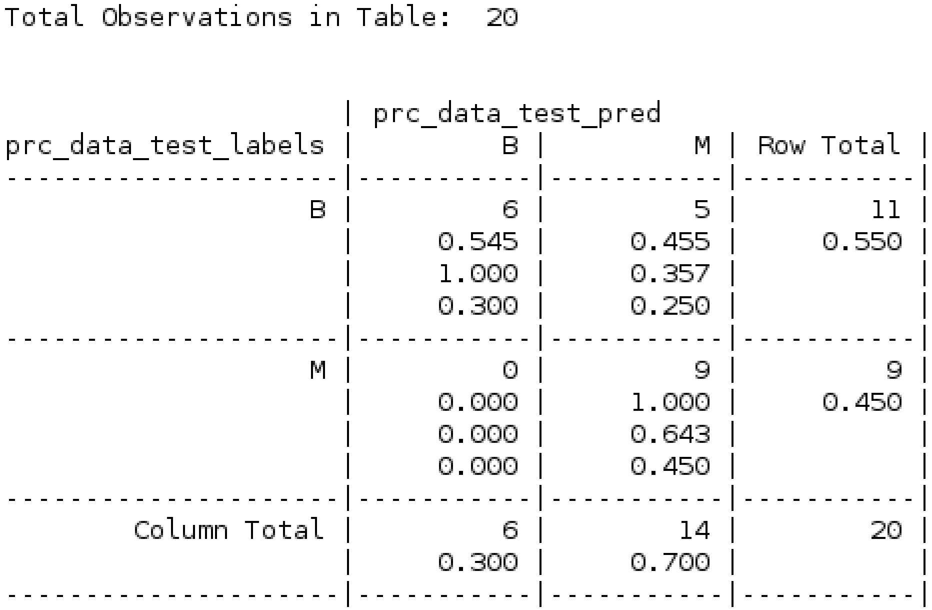
```
Total Observations in Table:  20


                     | prc_data_test_pred
prc_data_test_labels |         B |         M | Row Total |
---------------------|-----------|-----------|-----------|
                   B |         6 |         5 |        11 |
                     |     0.545 |     0.455 |     0.550 |
                     |     1.000 |     0.357 |           |
                     |     0.300 |     0.250 |           |
---------------------|-----------|-----------|-----------|
                   M |         0 |         9 |         9 |
                     |     0.000 |     1.000 |     0.450 |
                     |     0.000 |     0.643 |           |
                     |     0.000 |     0.450 |           |
---------------------|-----------|-----------|-----------|
        Column Total |         6 |        14 |        20 |
                     |     0.300 |     0.700 |           |
---------------------|-----------|-----------|-----------|
```

k-nearest neighbors algorithm

Practical session with R – 4, evaluate the model performance

There were 14 cases of False Positives (FP) meaning 14 cases were actually benign in nature but got predicted as malignant.

The total accuracy of the model is 60 %((TN+TP)/35) which shows that there may be chances to improve the model performance



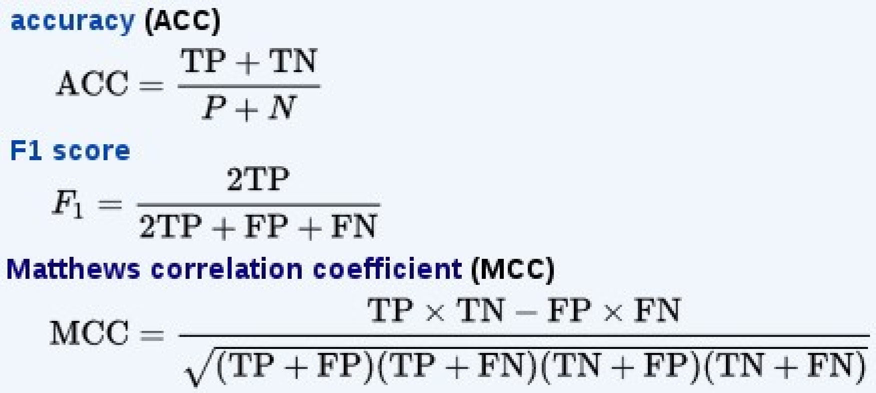
```
Total Observations in Table:  20


                     | prc_data_test_pred
prc_data_test_labels |         B |         M | Row Total |
---------------------|-----------|-----------|-----------|
                   B |         6 |         5 |        11 |
                     |     0.545 |     0.455 |     0.550 |
                     |     1.000 |     0.357 |           |
                     |     0.300 |     0.250 |           |
---------------------|-----------|-----------|-----------|
                   M |         0 |         9 |         9 |
                     |     0.000 |     1.000 |     0.450 |
                     |     0.000 |     0.643 |           |
                     |     0.000 |     0.450 |           |
---------------------|-----------|-----------|-----------|
        Column Total |         6 |        14 |        20 |
                     |     0.300 |     0.700 |           |
---------------------|-----------|-----------|-----------|
```

k-nearest neighbors algorithm

Practical session with R – 4, evaluate the model performance

When the prediction set is made of real value and there's not a fixed likelihood threshold to compute the confusion matrix, the best evaluation score to use is the PRECISION-RECALL Area Under the Curve (PR-AUC). This curve considers all the possible likelihood thresholds.

In our case, both the input dataset and the predictions are binary, so we can consider the following scores.



accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N}$$

F1 score

$$F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Matthews correlation coefficient (MCC)

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

(c) Wikipedia

The best (most useful and effective) score to use for a classification problem is the Matthews correlation coefficient (MCC), because it is based upon the size of the 4 confusion matrix categories.

k-nearest neighbors algorithm

Practical session with R – 4, evaluate the model performance

We can compute the MCC score with this R function developed on Kaggle.com

```
  # Compute the Matthews correlation coefficient (MCC) score
  # Jeff Hebert 9/1/2016
  # Geoffrey Anderson 10/14/2016
mcc <- function (actual, predicted)
{
  TP <- sum(actual == 1 & predicted == 1)
  TN <- sum(actual == 0 & predicted == 0)
  FP <- sum(actual == 0 & predicted == 1)
  FN <- sum(actual == 1 & predicted == 0)

  sum1 <- TP+FP; sum2 <-TP+FN ; sum3 <-TN+FP ; sum4 <- TN+FN;
  denom <- as.double(sum1)*sum2*sum3*sum4
  if (any(sum1==0, sum2==0, sum3==0, sum4==0)) {
    denom <- 1
  }
  mcc <- ((TP*TN)-(FP*FN)) / sqrt(denom)
  return(mcc)
}
```

The results can be between -1 (worst prediction) and +1 (perfect prediction).

k-nearest neighbors algorithm

Practical session with R – 4, evaluate the model performance

In our example, we have first to transform the "B" and "M" labels to 0s and 1s:

```
prc_data_test_labels_binary_TEMP <- replace(prc_data_test_labels,
prc_data_test_labels=="M", 1)
prc_data_test_labels_binary <- replace(prc_data_test_labels_binary_TEMP,
prc_data_test_labels=="B", 0)
prc_data_test_labels_binary <- as.numeric (prc_data_test_labels_binary)
prc_data_test_labels_binary


prc_data_test_pred_AS_CHAR <- as.character(prc_data_test_pred)

prc_data_test_pred_binary_TEMP <- replace(prc_data_test_pred_AS_CHAR,
prc_data_test_pred_AS_CHAR=="M", 1)

prc_data_test_pred_binary <- replace(prc_data_test_pred_binary_TEMP,
prc_data_test_pred_AS_CHAR=="B", 0)

prc_data_test_pred_binary <- as.numeric (prc_data_test_pred_binary)
prc_data_test_pred_binary


mcc(prc_data_test_labels_binary, prc_data_test_pred_binary)
```

The result is +0.59 (-1 <= MCC <= +1)

k-nearest neighbors algorithm

Practical session with R – 5, implement optimization

Exercise for the audience: optimize the value of k

Steps:

- split the input dataset into training set, validation set, and test set
- try different values of k (how?)
- choose the best model (how?)

References

Books, papers, courses

Books:
- C. Bishop, "Pattern recognition and machine learning", Springer, 2006
- P. Baldi, "Machine learning: the bioinformatics approach", MIT Press, 2001

Papers:
- P. Domingos, "A few useful things to learn about machine learning", Communications of ACM, 2012

Videocourses:
- Andrew Ng, "Machine learning", Coursera https://www.coursera.org/learn/machine-learning

The end

You can find slides and exercise code on my website:

www.DavideChicco.it

For any question, doubt, or possible **collaborations**, please contact me at:

davide.chicco@gmail.com