Module 3 - Data Science and Machine Learning.

3.2 Supervised Learning.

Author:

By Prof. Alberto Fernández Hilario

Full Professor at the University of Granada, Andalusian Interuniversity Institute on Data Science and Computational Intelligence (DasCI)

Brief Instructions

Reminder: Introduction to NoteBook

The *Jupyter* (Python) notebook is an approach that combines blocks of text (like this one) along with blocks or cells of code. The great advantage of this type of cells is their interactivity, as they can be executed to check the results directly on them.

Very important: the order of the instructions (code blocks) is fundamental, so each cell of this notebook must be executed sequentially. In case of omitting any of them, the program may throw an error (an exit block will be displayed with a red message in English), so you should start from the beginning in case of doubt. To make this step easier, you can go to the "Run Environment" menu and click on "Run previous".

Go for it!

Click on the "play" button on the left side of each code cell. Lines starting with a hashtag (#) are comments and do not affect the execution of the programme.

You can also click on each cell and do "*ctrl+enter*" (*cmd+enter* on Mac).

When the first of the blocks is executed, the following message will appear:

"Warning: This notebook was not created by Google.

The creator of this notebook is \@go.ugr.es. It may request access to your data stored in Google or read data and credentials from other sessions. Please review the source code before running this notebook. If you have any questions, please contact the creator of this workbook by sending an email to \<author>@go.ugr.es. "

Don't worry, you will have to trust the contents of the Notebook and click on "*Run anyway*". All the code runs on an external compute server and will not affect your computer at all. No information or credentials will be requested, so you will be able to continue the course safely.

Each time you run a block, you will see the output just below it. The information is usually always the last instruction, along with any `print()` (print command) in the code.

INDEX

In this *NoteBook*:

1. We will revisit the concept of supervised learning and its fundamentals.
2. We will discuss the main characteristics that influence supervised learning.
3. The concept of Trustworthy Artificial Intelligence and its influence on decision-making in data collection, model adaptation, and result auditing will be introduced.
4. We will explain the importance of the validation of learning models.
5. We will present different validation alternatives.

Contents:

1. WHAT IS SUPERVISED LEARNING?

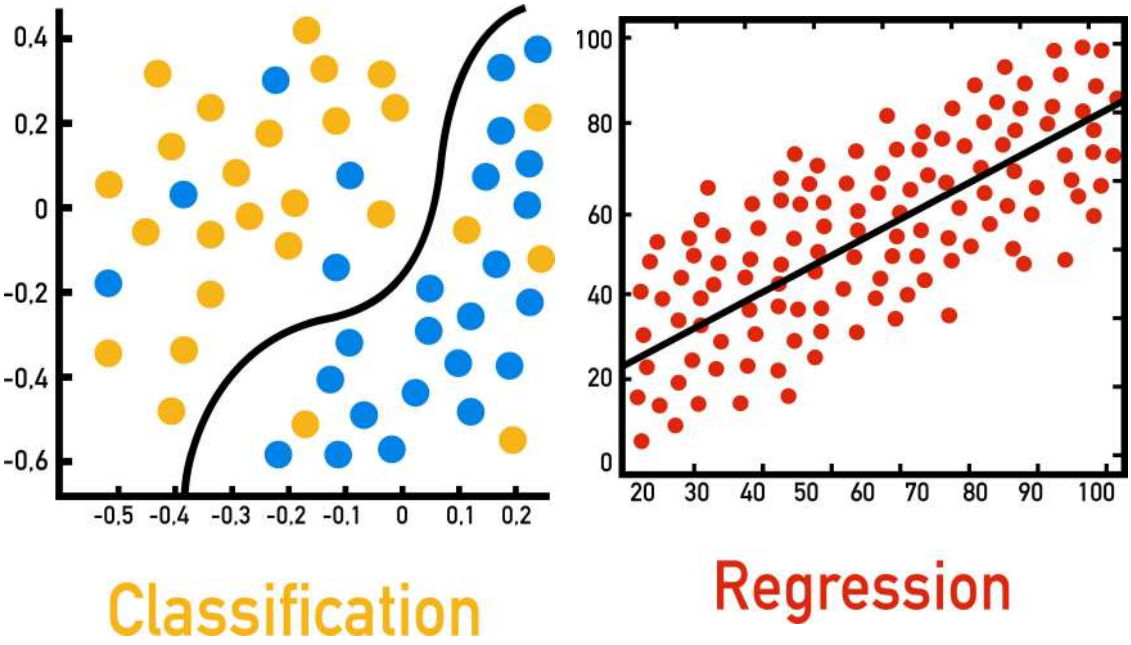
In previous modules in this course, we highlighted some interesting examples of the use of machine learning in the field of bioinformatics. Specifically, in Capsule 2 of Module 1 (*Bioinformatics. applications in biosciences and biohealth*), we listed drug development and discovery, microbiology, biomedical text mining, and precision or personalized medicine, among others. More specifically, machine learning can be directly used in these fields as a diagnostic tool to determine the category to which a patient belongs (e.g., healthy versus diseased). Another valid option would be to use machine learning to optimize the specific dosage of a drug for a specific group of patients, in this case by calculating a numerical value.

In both the examples provided above, the goal is to predict the value of an output variable such as a type of patient or the dose of a given medicine. When humans perform this task, they do so based on knowledge gained from their experiences with other similar cases (in these examples, patients and drugs). In the case of machine learning, this experience is gathered from instances in a dataset, as already described in the first capsule of this module.

Both the examples of the diagnosis of patients and the estimation of a drug dosage are types of "*supervised learning*". According to the type of output variable of the given

problem, the two fundamental tasks within supervised learning are **classification** and **regression**.

Classification refers to when the objective is to determine the category of an instance within a fixed set of values (the diagnosis of patients). On the other hand, and broadly speaking, regression seeks to create a mathematical interpolation function for a real variable (the drug dose estimate). The input variables that define the problem or case study are used in both cases; machine learning algorithms usually look for strong correlations between input and output variables to build a high-quality model.



In both cases, the input variables that define the problem or case study will be used. In particular, Machine Learning algorithms usually look for high correlations between input and output variables in order to build a high quality model.

2. DATA CHARACTERISTICS THAT INFLUENCE LEARNING

Within supervised (classification and regression) tasks, the learning algorithm tries to find the best fit for the data during the training phase. To this end, input variables describing each problem instance are used.

In this sense, it is important to discuss whether it is possible to determine the amount of data that is optimal for correct learning, as well as the ideal relationship between the number of instances and the number of variables that represent them. Another very relevant issue is to understand the details associated with the input variables themselves. Finally, we must emphasize the need for quality data and the avoidance of data bias.

2.1 Discussion of the number of instances

There is no general valid answer regarding the amount of information required for supervised learning tasks for any case study or problem.

From a statistical point of view, the larger the sample (the data set), the more representative it is. Thus, the more information available, the more different cases can be covered, and the better the model will fit the actual case under study.

It is indeed possible to extract knowledge from very little data, even from as little as 50 to 100 instances. However, it is highly likely that the model generated will be too specific and therefore, will not be useful for further applications. The number of instances required for correct learning will depend on the difficulty of the problem to be solved, although a rule of thumb is that it should be at least 10 times the number of parameters used to set up the algorithm.

Finally, the question remains of whether there is a specific ratio between the number of instances and variables that is ideal for correct learning. Unfortunately, there is no exact answer to this question, although we can reuse the generic rule from above and so in this case, the number of instances must be at least 10 times the number of input variables used.

2.2 Discussion of input variables

As indicated above, machine learning algorithms tend to look for strong correlations between input and output variables.

Most of them prefer numerical input variables (real values such as gene expression levels or the pH of a product), although nominal variables (categories such as color or gender) can also be used.

The only important thing is to be careful not to include variables that may distort the knowledge you want to extract. For example, there might be a strong relationship between a person's age and the survival time for a particular disease, but our aim is to find this association reflected directly in the genetic data values. These are known as "**confounding**" data variables and can cause spurious associations that must be identified and avoided, especially when working in the area of bioinformatics.

Another relevant issue with respect to variables is "the curse of dimensionality'". The term dimensionality refers to the cardinality (the number of elements) of a set of variables. In bioinformatic-type problems, the number of variables used for the study is often in the thousands or even tens of thousands. This makes the learning task exceedingly difficult because it reduces the algorithm's ability to find an optimal correlation between the inputs and outputs.

Thus, we must put considerable effort into the data preparation phase, selecting only the most important variables for the study in hand. Indeed, models built on subsets of high-quality predictor variables are often superior to those generated using all the original variables.

3. TRUSTWORTHY ARTIFICIAL INTELLIGENCE: ENSURING ETHICS IN MODELS

In a world driven by Artificial Intelligence (AI) and Data Science, reliability and ethics are paramount. Remember that when we talk about AI, we are referring to systems that use Machine Learning, as these terms are closely interconnected today.

In this section, we will explore the increasing importance of ensuring transparency, fairness, and security in the use of AI, especially in biological contexts.

3.1 Explainable Artificial Intelligence and Model Transparency Ensuring that AI systems and/or Machine Learning models are transparent and their decisions are explainable is essential. In biology, imagine a model that helps diagnose diseases based on genomic data. It's crucial that biologists can understand how a diagnosis is reached. Model transparency and explainability ensure that AI decisions are reliable and ethical.

To ensure the above, some of the most common strategies are as follows:

1. **Model Interpretability**: This involves choosing Machine Learning models that are naturally easier to understand. In Module 5, we will discuss the properties that determine when a model tends to be more interpretable, which is especially due to the ability to trace its decision-making logic.

2. **Data and Model Visualization**: Visual representations can help users better understand how a model works. Graphics, diagrams, and visual representations of data and modeled results can make information more accessible.

3. **Explainability Techniques**: Specific methods have been developed to explain more complex models, i.e., those that are not interpretable per se. One approach is to use feature importance plots, which show which variables are most influential in model predictions.

4. **Documentation and Metadata**: Maintaining detailed records and documentation of models and data is essential for transparency. This includes information on how data was collected, how the model was trained, and how decisions were made.

3.2 Addressing Bias and Fairness in Data

In the exciting world of Data Science, there is a significant concern that goes beyond mathematics and modeling techniques: bias. Data bias can lead to unfair, inequitable, and ethically problematic outcomes. In this context, bias refers to the presence of prejudices in the data that can negatively affect certain groups or outcomes. For example, if a Machine Learning model used in biodiversity conservation is primarily trained on data from one geographic region, it may overlook issues in other areas.

A fundamental concept to grasp is that **algorithms themselves are not inherently biased**. Instead, it's the bias residing in the data that can introduce inequitable results. This means that to create fair and ethical Artificial Intelligence and Machine Learning systems, we must pay attention to both the data collection and preparation process and the design and training of our models.

A key aspect in understanding data bias is the notion of **protected variables**, which are data features such as gender, race, or age, that are subject to potential discrimination. Identifying these protected variables is crucial for evaluating potential sources of bias.

A common approach is to assess bias based on groups identified by these protected variables. It is quite common today to conduct a model audit to thoroughly examine a

model's behavior in relation to so-called "bias metrics."

Let's imagine that we are developing a Machine Learning model to predict an individual's risk of developing a heart disease based on their genetic data and health history. In this context, two key variables could be related to bias: the gender and age of the patients. Gender is a binary variable (male or female), and age can be grouped into different categories (e.g., young, adult, and elderly). In this case, we could focus on the so-called "disparate positive prediction rate metric," which refers to how many patients from each group (gender and age) are labeled as "at risk" by the model. For example, we could calculate the positive prediction rate for men and women separately and for each age group. If we observe significant differences in these rates, we might be dealing with bias in the model.

3.3 Security, Robustness, and Regulatory Compliance in Artificial Intelligence

In the field of Data Science, it is essential to understand and apply concepts related to security, robustness, and regulation in Machine Learning algorithms. These concepts become critical to ensure that technological advancements are not only effective but also safe and ethical.

1. **Algorithm Robustness:** Robustness refers to the ability of a Machine Learning algorithm to maintain its performance even in unusual situations or when dealing with noisy data. In biology, this translates to the models' ability to analyze genetic data where there may be sequencing errors or unexpected variability.

2. **Security in Biological Data:** Given the sensitivity of biological data, it is crucial to ensure its security. This involves protecting patients' privacy and ensuring that genetic or medical data is not used inappropriately. For example, when sharing patient data for research, it is essential to ensure anonymization and the protection of personal information. Currently, the so-called **Federated Learning** is a paradigm that allows training models without the need to share sensitive data from different sources.

3. **AI Act and Regulations:** The AI Act is a significant regulation in the European Union that seeks to regulate AI to ensure its ethical and safe use. This means that any researcher or developer must comply with specific regulations to ensure safety and ethics in projects containing an AI component. In particular, limits on the development of applications are defined based on their risk to people. Not all AI projects in biology present the same level of risk. Some, like medical diagnosis, can have a high impact on people's health. Others, like pattern recognition in genetic sequences, may have moderate risk. Understanding and categorizing these risk levels is crucial for the responsible application of AI.

4. COMPLEMENTARY CASE STUDY: LEARNING ABOUT BREAST CANCER THROUGH IMAGES

Throughout this course, we will work on a learning problem related to **skin melanoma.** However, in this present section, for the sake of simplicity, we are going to perform some initial tests using a relatively simple dataset, known as *breast cancer*.

The input variables for this dataset were calculated from a digitized image of a fine needle aspirate of a breast mass. They describe the characteristics of the cell nuclei represented in the image in three-dimensional space.

This is a small dataset extensively described in the scientific literature, and many machine learning tools are available for it (e.g., *Scikit-Learn*).*

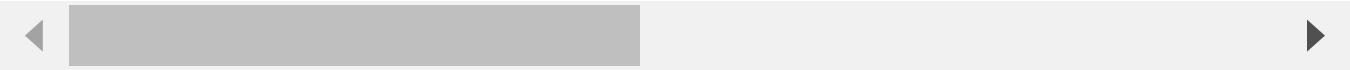
To use this dataset, first we must view the code that allows the data to be stored in a straightforward way as Python variables. Take a good look at the structure of the code block, which we will describe in more detail below.

In [1]:
```
import pandas as pd
#Scikit-learn contains its own data repository
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()
#we store the input (features/variables) and output (class) in two variables X and
X, y = data.data, data.target
#Transform variables a from numpy type to DataFrame type (for convenience of use)
X = pd.DataFrame(X,columns = data.feature_names)
y = pd.DataFrame(y,columns=["label"])

#Observe the first five samples
X.head()
```

Out[1]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 30 columns



he following information can be extracted from this code:

- `X` and `y` are two `dataframe`-type data structures that are stored in `Pandas` (*a library widely used in Python*). A `dataframe` is understood as an n-dimensional vector, which generally makes it much more convenient to work with this type of data structure than with the one used in the `NumPy` library.
- `X` contains the input data matrix, a one-dimensional vector containing the label of each sample in `X`. For simplicity, in this case example, we decided to solve a classification problem (a categorical-type output variable).

- In supervised-type learning, `X` and `y` will be used to build and evaluate the machine learning model.

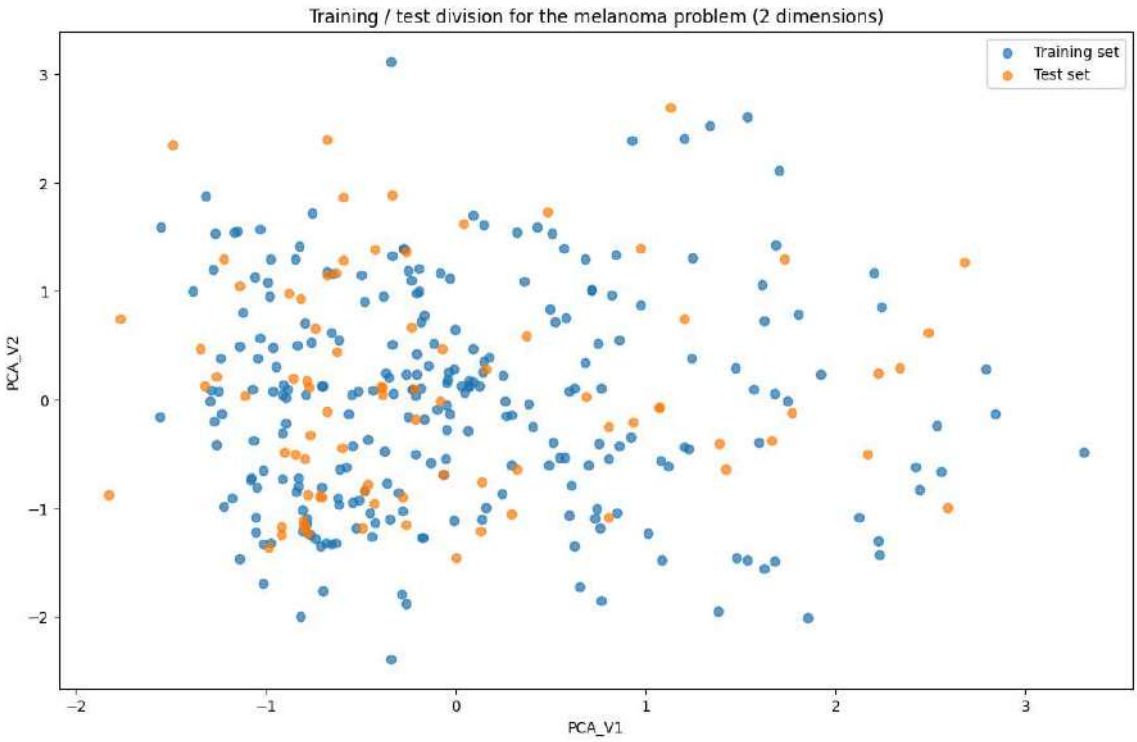
5. THE NEED FOR MACHINE LEARNING MODEL VALIDATION

As indicated above, the supervised learning training procedure seeks to create a model that perfectly fits the input data. To check how good this fit is, the model-building algorithm uses what is known as a '*performance metric*'. This will determine the error produced by the model in the process of being generated when making a prediction based on the training set instances.

Once the learning has completed, it is vital to perform a model validation process to determine the quality of the model when run on new, real data. To do this, a so-called 'test set' is used, which should contain new instances that were not used during the training phase.

According to the above, there are two very distinct phases in supervised learning, namely, **training** (or just "*train*") and **validation** (better kn own as "*test*"). We must insist that two totally independent data sets are used for these phases because, if a test example is used during training, the quality of the model generated will be overestimated.

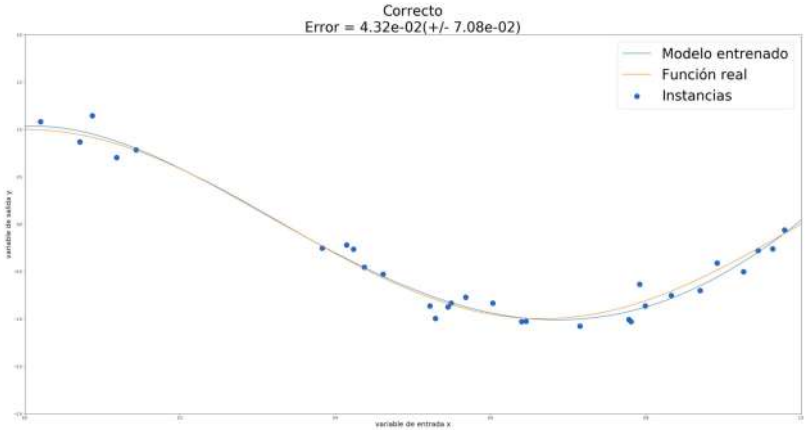
The following image shows how the input dataset is divided into two separate groups, where each example is marked with a different color according to whether it belongs to a "training" (blue) or "test" (orange) set.



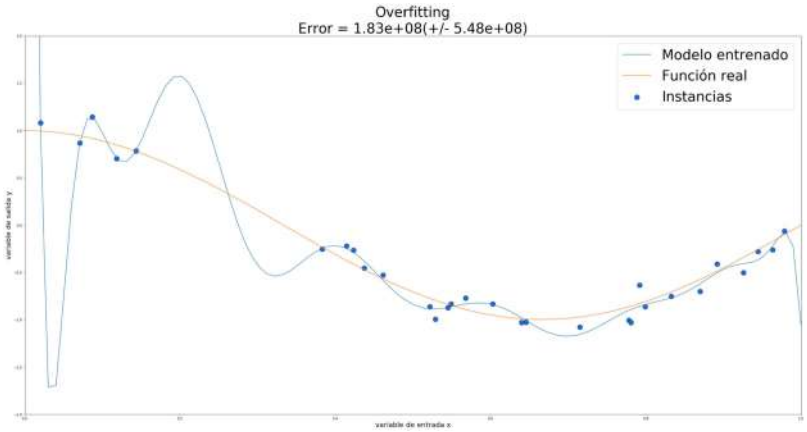
5.1 Validation case studies

A performance metric is calculated for each set based on the prediction results obtained through the training and testing. At this point, several outcome scenarios are possible:

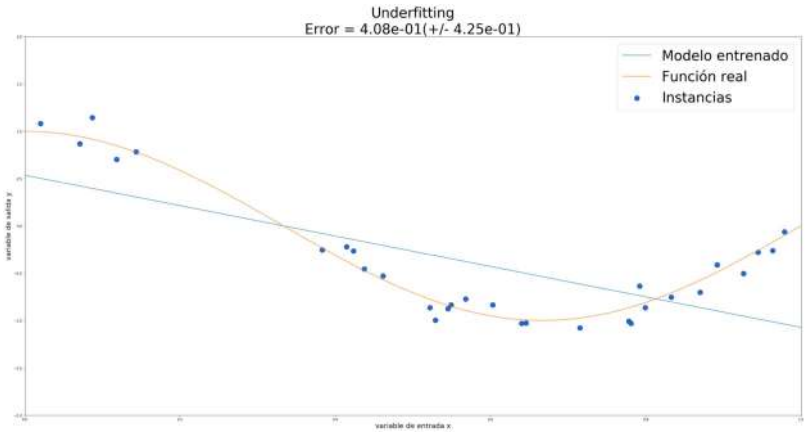
- A *high-quality* performance metric is obtained for both the training and test sets meaning that the model is fully effective and the process is complete.



- A *high-quality* result is achieved for the training, but a *very-low quality* result for the testing. This means that the model has fallen into *over-fitting*. In other words, the model we have built is so specific to the training data that it is unable to generalize well to different (test) data.



- Both the training and test sets obtain *low-quality* scores (*under-fitting*). This indicates that the model is not good and so data preprocessing must be applied, or the learning must be repeated with proper parameter tuning. Alternatively, the system must simply be fed with more data



5.2 Types of validation techniques

There are three different methodologies for splitting our original data set into training and test sets, as described below:

- The simplest is "**hold-out**", where the original data is split into two, equal disjointed sets. For example, 60% of initial instances are used for training and the remaining 40% are used for testing.
- Because of its statistical validity, the most widely used method is **k-fold cross-validation**, where the original set is split into *k* disjointed parts. For example, the instances are separated into 5 groups without replacement.
- There is also the possibility of performing exhaustive experimentation with the '**leave one out validation**' (LOOV technique). In this case the whole set is used for training, except for one example which is used for testing.

In the following, these main techniques will be explained in more detail, paying special attention to the most recommended of them: k-partition cross-validation. The rationale for prioritizing this methodology is that all instances of the data set are used in the different test partitions, which promotes greater statistical rigor and thus, validity of the results obtained.

5.3 Hold-out validation

As introduced above, the **hold-out** technique creates two simple sets, i.e., a training and a test file, by partitioning a given percentage of the data. The percentage selected for each subset is left to the user, although typical values are above 60, 75, or 80% of the set for training, using the rest for testing.

The advantage of the hold-out method is that it is remarkably simple and efficient to perform. However, this method is not recommended because the final quality of the model will depend to a large extent on how the data were split. In other words, by separating the instances in a completely random way, it is possible that the ones selected for testing were the most difficult to identify.

Therefore, the hold-out technique is usually used when initial tests are carried out in order to discover the basic behavior of the different models (machine learning algorithms) applied to the problem under study.

Here we present the code used to create the hold-out data partitions, along with all the necessary instructions required to understand its specific functions. The main parameter used in this case is the `ratio` to define the partition between the *training* and *test* sets, respectively. Some additional details about the source code are noted below:

- The Scikit-learn library will be used as a support means throughout this course and so all functionality and modules from it will be imported through the first lines of the code.
- Data partitioning is performed randomly by default and so, what is known as a 'seed' must be used. This is a fixed value that allows us to always generate the same results each time the code is executed. This seed is indicated in the `random_state` parameter.
- `print()` will always be used as the last instruction in order to display a result that will allow the entire process to be interpreted.

In [2]:
```
#Loads the code needed to do hold-out
from sklearn.model_selection import train_test_split

#Parameters used:
rd = 42 #because it is a non-deterministic process, a seed is set.
ratio = 0.2 #Most important parameter: what is the ratio used for training and test

#Next, the division is done. Important: the input and output (X and y) are divided.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ratio,random_st

#Print the indices of train and test with respect to the original set.
print("%s %s" % (X_train.index, X_test.index))
```

```
Int64Index([ 68, 181,  63, 248,  60,  15, 290, 137, 155, 517,
            ...
            330, 214, 466, 121,  20,  71, 106, 270, 435, 102],
           dtype='int64', length=455) Int64Index([204,  70, 131, 431, 540, 567, 36
9,  29,  81, 477,
            ...
            549, 530, 163, 503, 148, 486,  75, 249, 238, 265],
           dtype='int64', length=114)
```

The following code snippets show the size of the sets that will have just been created

In [3]:
```
#First train (X & y)
X_train.shape, y_train.shape
```

Out[3]:
```
((455, 30), (455, 1))
```

In [4]:
```
#Then test (X & y)
X_test.shape, y_test.shape
```

Out[4]:
```
((114, 30), (114, 1))
```

Based on the results shown above, by performing an 80:20 split (applying a `ratio = 0.2` for the test), 455 instances were used in the training ($569 \cdot 0.8$) and 114 for the test ($569 \cdot 0.2$).

To correctly perform the validation, it is extremely important that both sets are disjointed, i.e., that no test instance was seen during the training. This fact is checked in the following code block which 'asks' if any data ("instance") of the training set (`X_train`) is contained in the test set (`X_test`). The result should always be `False` for all the rows.

In [5]:
```
X_train.isin(X_test) #Instruction that checks if any test element repeats in train

#The first column to be observed is the original index, which is unordered.
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
68	False	False	False	False	False	False	False	False	False
181	False	False	False	False	False	False	False	False	False
63	False	False	False	False	False	False	False	False	False
248	False	False	False	False	False	False	False	False	False
60	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
71	False	False	False	False	False	False	False	False	False
106	False	False	False	False	False	False	False	False	False
270	False	False	False	False	False	False	False	False	False
435	False	False	False	False	False	False	False	False	False
102	False	False	False	False	False	False	False	False	False

455 rows × 30 columns

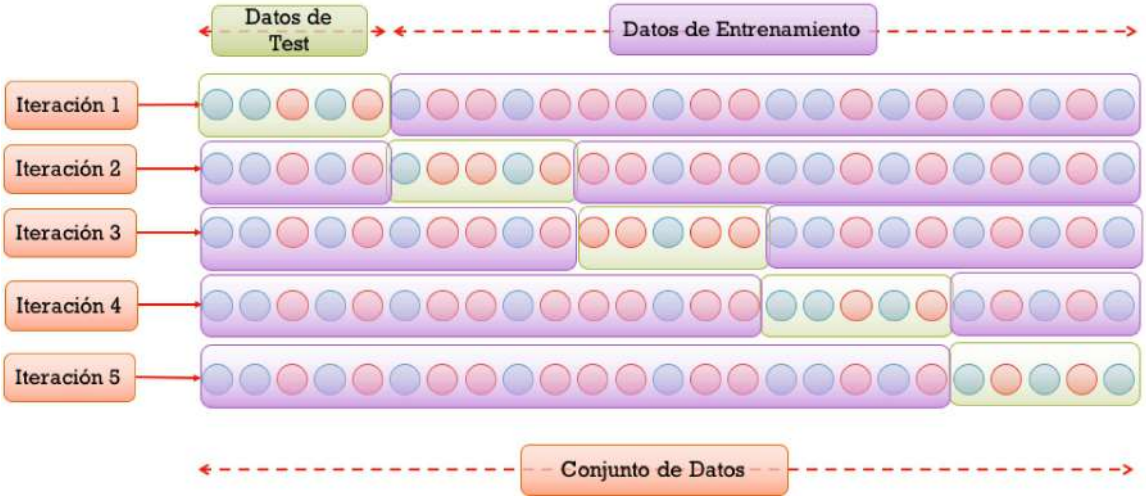
5.4 K-fold Cross-validation

This approach is the most widely used and is recommended when performing supervised learning. The main reason for this is its statistical rigor, as well as the use of the whole dataset for training and testing, by using an iterative procedure (i.e., a procedure repeated a certain number of times).

In particular, when using k-fold cross-validation, the data is divided into k disjointed subsets, called 'folds' or 'partitions'. As indicated above, the objective is to validate the models with different combinations of these partitions.

Specifically, the instances belonging to each of the k-folds will be stored in a different test set; while the union of the instances found in the remaining k−1 folds will be used to build each of the training sets.

As an example, if a cross-validation is performed with 5 folds (`k = 5`), the data would be distributed as shown in the following figure, taking 1/5 for testing and the remaining 4/5 for training for each fold.



The following shows how this data splitting can be done using Python, again taking advantage of the functionality of the scikit-learn library.

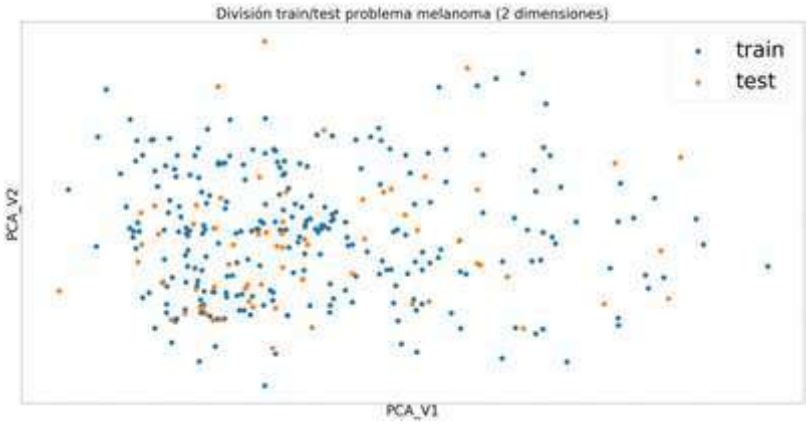
In [6]:
```
from sklearn.model_selection import KFold #Load the necessary functions

#Parameters
rd = 42
partitions = 5

#First difference with hold-out, partition indexes are created a priori
kf = KFold(n_splits=partitions,shuffle=True,random_state=rd)

#Next, each subdivision is performed.
#Note that it is necessary to make it iterative (for loop) for each partition.
for train, test in kf.split(X,y):
  print("%s %s" % (train, test))
  #Sets are created iteratively (iloc is a "search" function)
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
```

```
[  0   1   3   4   5   7   8  12  13  14  15  16  17  18  19  20  21  22
  23  24  25  26  27  28  31  32  33  34  35  36  37  38  40  41  42  43
  44  45  46  47  48  49  50  51  52  53  54  56  57  58  59  60  61  62
  63  64  65  66  67  68  69  71  74  80  85  87  88  89  91  92  93  94
  95  96  97  98  99 100 102 103 105 106 107 108 111 112 113 114 115 116
 117 119 120 121 122 123 124 125 126 127 128 129 130 133 134 135 136 137
 138 139 141 142 143 146 147 149 150 151 152 154 155 156 157 159 160 161
 162 164 166 168 169 170 171 172 173 174 175 176 178 179 180 181 183 184
 185 186 187 189 190 191 192 193 194 195 197 198 199 200 201 202 205 206
 207 209 210 212 213 214 215 216 217 218 219 220 221 223 224 225 226 227
 229 230 231 232 233 234 236 237 239 240 241 242 243 244 245 246 247 248
 251 252 253 254 256 258 259 260 261 262 263 266 267 268 269 270 272 273
 276 277 278 279 280 282 283 285 286 287 288 289 290 291 292 293 294 295
 296 297 298 299 300 301 302 303 304 306 307 308 309 310 311 312 313 314
 315 316 317 318 319 321 323 324 325 326 327 328 330 335 336 337 338 339
 340 341 342 343 344 345 346 347 348 349 350 351 352 354 355 356 357 358
 359 360 361 363 364 365 366 367 368 370 371 372 373 374 375 376 377 378
 379 381 383 385 386 387 388 389 390 391 392 396 397 398 399 400 401 402
 403 404 405 406 407 409 410 411 412 413 414 415 416 417 418 419 420 421
 423 426 427 428 429 430 432 433 434 435 436 437 438 439 440 442 443 444
 445 446 447 448 449 450 451 452 453 454 455 456 458 459 460 461 463 465
 466 467 469 470 471 472 473 474 475 476 478 479 480 481 483 484 485 487
 488 489 490 491 492 493 494 495 496 497 498 499 501 502 504 505 506 507
 508 509 510 512 513 514 515 516 517 518 519 521 522 523 524 525 529 533
 534 536 537 539 541 542 543 544 545 546 547 548 550 552 553 554 558 559
 560 562 563 566 568] [  2   6   9  10  11  29  30  39  55  70  72  73  75  76  77
78  79  81
  82  83  84  86  90 101 104 109 110 118 131 132 140 144 145 148 153 158
 163 165 167 177 182 188 196 203 204 208 211 222 228 235 238 249 250 255
 257 264 265 271 274 275 281 284 305 320 322 329 331 332 333 334 353 362
 369 380 382 384 393 394 395 408 422 424 425 431 441 457 462 464 468 477
 482 486 500 503 511 520 526 527 528 530 531 532 535 538 540 549 551 555
 556 557 561 564 565 567]
[  1   2   3   4   5   6   8   9  10  11  12  13  14  16  20  21  23  26
  27  28  29  30  32  34  35  36  37  38  39  40  41  43  44  45  47  48
  50  51  52  53  55  58  59  61  62  64  65  67  70  71  72  73  74  75
  76  77  78  79  80  81  82  83  84  85  86  87  90  91  92  94  95  96
  97  98  99 100 101 102 103 104 105 106 107 109 110 111 112 115 116 118
 119 120 121 122 123 125 127 128 129 130 131 132 133 134 135 136 138 139
 140 142 143 144 145 146 147 148 150 151 152 153 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 175 177 178 179 180 182 183 186
 187 188 189 190 191 193 194 196 197 198 200 201 202 203 204 205 206 207
 208 211 212 213 214 215 216 217 219 220 221 222 223 224 225 226 228 229
 230 232 233 235 236 237 238 239 240 241 242 243 246 249 250 251 252 253
 254 255 256 257 258 259 260 262 263 264 265 266 267 269 270 271 273 274
 275 276 278 279 281 282 283 284 285 286 288 291 292 293 294 295 296 297
 299 300 302 303 305 306 307 308 309 312 313 314 315 316 317 318 320 321
 322 323 324 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
 342 343 344 345 347 348 349 350 351 352 353 354 356 357 358 359 360 361
 362 363 365 366 367 368 369 370 371 372 373 375 376 377 378 379 380 382
 383 384 385 386 387 388 389 391 392 393 394 395 397 400 401 403 405 406
 408 409 412 413 415 416 417 418 419 420 422 423 424 425 427 429 430 431
 432 433 435 436 437 438 439 440 441 443 444 445 447 448 450 451 452 454
 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 471 472 473
 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491
 492 493 496 499 500 502 503 504 505 506 508 509 510 511 513 514 515 516
 518 519 520 521 522 524 525 526 527 528 529 530 531 532 533 534 535 536
 537 538 540 543 544 546 548 549 551 552 554 555 556 557 558 559 560 561
 563 564 565 566 567] [  0   7  15  17  18  19  22  24  25  31  33  42  46  49  54
56  57  60
  63  66  68  69  88  89  93 108 113 114 117 124 126 137 141 149 154 155
 172 173 174 176 181 184 185 192 195 199 209 210 218 227 231 234 244 245
 247 248 261 268 272 277 280 287 289 290 298 301 304 310 311 319 325 341
 346 355 364 374 381 390 396 398 399 402 404 407 410 411 414 421 426 428
```

```
 434 442 446 449 453 470 494 495 497 498 501 507 512 517 523 539 541 542
 545 547 550 553 562 568]
[  0   1   2   4   6   7   8   9  10  11  12  13  14  15  17  18  19  20
  21  22  24  25  27  28  29  30  31  32  33  34  35  39  40  41  42  43
  44  46  47  49  51  52  53  54  55  56  57  58  60  61  62  63  64  65
  66  68  69  70  71  72  73  75  76  77  78  79  80  81  82  83  84  85
  86  87  88  89  90  91  93  95  98  99 100 101 102 104 105 106 107 108
 109 110 112 113 114 115 117 118 120 121 124 125 126 127 128 129 130 131
 132 133 134 135 136 137 138 140 141 142 144 145 148 149 151 153 154 155
 156 158 159 160 161 162 163 164 165 166 167 169 170 171 172 173 174 176
 177 178 179 181 182 184 185 186 187 188 189 190 191 192 195 196 197 199
 200 201 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218
 219 221 222 223 224 226 227 228 230 231 232 233 234 235 236 238 240 241
 242 243 244 245 247 248 249 250 251 252 254 255 256 257 258 259 260 261
 264 265 267 268 269 270 271 272 273 274 275 276 277 279 280 281 282 283
 284 285 287 288 289 290 292 294 295 298 300 301 303 304 305 306 308 309
 310 311 313 314 315 317 319 320 322 325 326 327 329 330 331 332 333 334
 337 339 341 342 343 344 345 346 347 349 350 351 353 354 355 356 358 359
 361 362 364 366 369 371 372 373 374 376 378 379 380 381 382 383 384 385
 387 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405
 406 407 408 409 410 411 412 413 414 416 417 418 419 421 422 424 425 426
 427 428 429 430 431 434 435 437 438 441 442 443 445 446 447 448 449 451
 452 453 454 455 456 457 458 459 460 461 462 464 465 466 467 468 470 472
 474 475 476 477 478 479 482 483 484 486 488 491 492 493 494 495 497 498
 500 501 503 504 505 506 507 508 509 510 511 512 514 515 516 517 518 520
 522 523 524 525 526 527 528 529 530 531 532 534 535 537 538 539 540 541
 542 545 547 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563
 564 565 566 567 568] [  3   5  16  23  26  36  37  38  45  48  50  59  67  74  92
94  96  97
 103 111 116 119 122 123 139 143 146 147 150 152 157 168 175 180 183 193
 194 198 202 220 225 229 237 239 246 253 262 263 266 278 286 291 293 296
 297 299 302 307 312 316 318 321 323 324 328 335 336 338 340 348 352 357
 360 363 365 367 368 370 375 377 386 388 415 420 423 432 433 436 439 440
 444 450 463 469 471 473 480 481 485 487 489 490 496 499 502 513 519 521
 533 536 543 544 546 548]
[  0   1   2   3   5   6   7   8   9  10  11  13  14  15  16  17  18  19
  20  21  22  23  24  25  26  27  29  30  31  33  34  36  37  38  39  40
  42  43  45  46  48  49  50  52  54  55  56  57  58  59  60  62  63  64
  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
  84  86  87  88  89  90  91  92  93  94  96  97  99 101 102 103 104 105
 106 108 109 110 111 113 114 116 117 118 119 121 122 123 124 126 128 130
 131 132 135 137 138 139 140 141 143 144 145 146 147 148 149 150 152 153
 154 155 156 157 158 160 161 162 163 165 166 167 168 172 173 174 175 176
 177 180 181 182 183 184 185 187 188 189 190 191 192 193 194 195 196 198
 199 201 202 203 204 205 207 208 209 210 211 212 214 216 217 218 220 222
 225 227 228 229 230 231 234 235 236 237 238 239 241 243 244 245 246 247
 248 249 250 251 252 253 255 257 259 260 261 262 263 264 265 266 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 284 286 287 288 289 290
 291 293 295 296 297 298 299 300 301 302 303 304 305 307 308 309 310 311
 312 313 315 316 318 319 320 321 322 323 324 325 326 328 329 330 331 332
 333 334 335 336 337 338 339 340 341 343 344 345 346 348 350 352 353 355
 357 360 362 363 364 365 366 367 368 369 370 372 374 375 377 378 379 380
 381 382 384 385 386 387 388 389 390 391 393 394 395 396 398 399 401 402
 404 407 408 409 410 411 413 414 415 418 420 421 422 423 424 425 426 427
 428 431 432 433 434 435 436 439 440 441 442 444 445 446 449 450 453 454
 455 457 458 459 461 462 463 464 466 468 469 470 471 473 474 475 476 477
 478 480 481 482 483 484 485 486 487 489 490 491 493 494 495 496 497 498
 499 500 501 502 503 504 507 508 510 511 512 513 516 517 519 520 521 522
 523 526 527 528 529 530 531 532 533 535 536 537 538 539 540 541 542 543
 544 545 546 547 548 549 550 551 552 553 554 555 556 557 560 561 562 563
 564 565 566 567 568] [  4  12  28  32  35  41  44  47  51  53  61  65  85  95  98
100 107 112
 115 120 125 127 129 133 134 136 142 151 159 164 169 170 171 178 179 186
 197 200 206 213 215 219 221 223 224 226 232 233 240 242 254 256 258 267
```

```
 282 283 285 292 294 306 314 317 327 342 347 349 351 354 356 358 359 361
 371 373 376 383 392 397 400 403 405 406 412 416 417 419 429 430 437 438
 443 447 448 451 452 456 460 465 467 472 479 488 492 505 506 509 514 515
 518 524 525 534 558 559]
[  0   2   3   4   5   6   7   9  10  11  12  15  16  17  18  19  22  23
  24  25  26  28  29  30  31  32  33  35  36  37  38  39  41  42  44  45
  46  47  48  49  50  51  53  54  55  56  57  59  60  61  63  65  66  67
  68  69  70  72  73  74  75  76  77  78  79  81  82  83  84  85  86  88
  89  90  92  93  94  95  96  97  98 100 101 103 104 107 108 109 110 111
 112 113 114 115 116 117 118 119 120 122 123 124 125 126 127 129 131 132
 133 134 136 137 139 140 141 142 143 144 145 146 147 148 149 150 151 152
 153 154 155 157 158 159 163 164 165 167 168 169 170 171 172 173 174 175
 176 177 178 179 180 181 182 183 184 185 186 188 192 193 194 195 196 197
 198 199 200 202 203 204 206 208 209 210 211 213 215 218 219 220 221 222
 223 224 225 226 227 228 229 231 232 233 234 235 237 238 239 240 242 244
 245 246 247 248 249 250 253 254 255 256 257 258 261 262 263 264 265 266
 267 268 271 272 274 275 277 278 280 281 282 283 284 285 286 287 289 290
 291 292 293 294 296 297 298 299 301 302 304 305 306 307 310 311 312 314
 316 317 318 319 320 321 322 323 324 325 327 328 329 331 332 333 334 335
 336 338 340 341 342 346 347 348 349 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 367 368 369 370 371 373 374 375 376 377 380 381
 382 383 384 386 388 390 392 393 394 395 396 397 398 399 400 402 403 404
 405 406 407 408 410 411 412 414 415 416 417 419 420 421 422 423 424 425
 426 428 429 430 431 432 433 434 436 437 438 439 440 441 442 443 444 446
 447 448 449 450 451 452 453 456 457 460 462 463 464 465 467 468 469 470
 471 472 473 477 479 480 481 482 485 486 487 488 489 490 492 494 495 496
 497 498 499 500 501 502 503 505 506 507 509 511 512 513 514 515 517 518
 519 520 521 523 524 525 526 527 528 530 531 532 533 534 535 536 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 553 555 556 557 558 559
 561 562 564 565 567 568] [  1   8  13  14  20  21  27  34  40  43  52  58  62  64
71  80  87  91
  99 102 105 106 121 128 130 135 138 156 160 161 162 166 187 189 190 191
 201 205 207 212 214 216 217 230 236 241 243 251 252 259 260 269 270 273
 276 279 288 295 300 303 308 309 313 315 326 330 337 339 343 344 345 350
 366 372 378 379 385 387 389 391 401 409 413 418 427 435 445 454 455 458
 459 461 466 474 475 476 478 483 484 491 493 504 508 510 516 522 529 537
 552 554 560 563 566]
```

As indicated, the output that should be shown after implementing the code shown above are the indices of the different instances in the training and testing sets. It is important to note that the numbers for testing do not match in any of the 5 cases.



In addition to the above, it is very common to repeat the partitioning (splitting) process into k-sets a certain number of times so as to add more statistical rigor to the results obtained. This means that, as in the case of the hold-out method, the aim is to eliminate dependence

on the random partitioning process because this can cause the most difficult test instances to always fall into the same set.

The procedure described above is referred to as *repeated k-fold cross-validation* and is shown in the following code example:

In [ ]:
```
from sklearn.model_selection import RepeatedKFold

#Parameters
rd = 42
partitions = 5
repeats = 3

#Very similar to the above, but including an additional repeats parameter
rkf = RepeatedKFold(n_splits=partitions, n_repeats=repeats, random_state=rd)
for index, (train, test) in enumerate(rkf.split(X,y)):
  if index % partitions == 0:
    print("Repetition ", (index // partitions)+1)
  print("%s %s" % (train, test))
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
```

Important note: The total number of partitions and/or iterations cannot be estimated a priori and will depend mainly on the number of instances available. In most cases, 10 partitions and 3 iterations are used by default. However, when only a few instances are available, as little as 5 partitions and 3–5 iterations can be used.

5.5 Stratified cross-validation (classification only)

This alternative is a very suitable variation for the classification of the *k-fold cross-validation* and returns stratified-type partitions. This term means that each set contains approximately the same percentage of samples from each target class as the full set does.

This is essential to avoid possible bias in the models and to avoid producing misleading conclusions. It is especially relevant when working with unbalanced data, in other words, with a very small percentage of samples from one of the classes.

As indicated at the beginning of this activity, cases of the type considered here are often observed in problems in the context of biology and/or healthcare.

In the following example, the solution is implemented using this new improved version of *k-fold*. The call format is practically equivalent to the previous case of k-fold cross-validation. To understand the differences in behavior, we can compare the two partitioning schemes, checking the final distribution of examples in each class.

To do so, we use a `NumPy` instruction called `bincount`, which simply counts the number of occurrences of each value in the list. The distribution of instances in the two classes for the test partitions is shown in the example below. Although the breast cancer dataset does not present a high imbalance of classes, we can observe that a bias towards one of the classes (in the second partition) may have been unintentionally created.

In [8]:
```
from sklearn.model_selection import StratifiedKFold, KFold
import numpy as np
```

```
#Parameters
rd = 42
partitions = 5

#Parameters rd = 42 partitions = 5 #We perform stratified partitioning
skf = StratifiedKFold(n_splits=partitions,shuffle=True,random_state=rd)

#Let's check for differences with KFold
kf = KFold(n_splits=partitions,shuffle=True,random_state=rd)

#Next we do each subdivision.
print("Check class distribution in SKF:")
for train, test in skf.split(X,y):
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
  print('train - {} | test - {}'.format(np.bincount(y_train.iloc[:,0]), np.bincount

print("Check class distribution in KF:")
for train, test in kf.split(X, y):
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
  print('train - {} | test - {}'.format(np.bincount(y_train.iloc[:,0]), np.bincount
```

```
Check class distribution in SKF:
train - [169 286] | test - [43 71]
train - [169 286] | test - [43 71]
train - [170 285] | test - [42 72]
train - [170 285] | test - [42 72]
train - [170 286] | test - [42 71]
Check class distribution in KF:
train - [169 286] | test - [43 71]
train - [175 280] | test - [37 77]
train - [169 286] | test - [43 71]
train - [169 286] | test - [43 71]
train - [166 290] | test - [46 67]
```

5.6 Leave One Out Validation (LOOV)

This is, together with *hold-out* the simplest of the validation techniques. In this particular case, each sample is considered as a single test set, whereas the remaining samples form the training set.

LOOV must be used with special care. Specifically, instead of building *k* models (such as in k-fold cross validatoin), now *n* models are obtained, being *n* the size of the dataset. Taken the former into account, and as each model is learned from the full dataset (minus one sample), the computational cost of this validation is simply huge.

Users must acknolwedge that, when it comes to performance metrics, this validation technique offers a high variance. This behavior is expected since all models are basically the same (they were built with almost the same instances).

LOOV can be simulated from `KFold(n_splits=n)` with `n` the total number of samples from the dataset.

In [9]:
```
#Parameters
rd = 42
partitions = len(X)

#Parameters rd = 42 partitions = all samples
loov = KFold(n_splits=partitions,shuffle=True,random_state=rd)
```

```
#Next we do each subdivision.
for train, test in loov.split(X,y):
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
  #Below the learning and validation should be taken place

print("Example last partition: Total sizeTr[{}] sizeTst[{}]".format(len(y_train), l
```

```
Example last partition: Total sizeTr[568] sizeTst[1]
```

BIBLIOGRAPHICAL REFERENCES

- Han, J., Kamber, M., Pei, J. (2011). Data Mining: Concepts and Techniques. San Francisco, CA, USA: Morgan Kaufmann Publishers. ISBN: 0123814790, 9780123814791
- Hastie, T., Tibshirani, R., Friedman, J. H. (2001). The Elements of Statistical Learning. Springer Verlag.
- Witten, I. H., Frank, E., Hall, M. A., Pal, C. J. (2017). Data mining: practical machine learning tools and techniques. Amsterdam; London: Morgan Kaufmann. ISBN: 9780128042915 0128042915
- Molnar, C. (2023). Interpretable Machine Learning A Guide for Making Black Box Models Explainable. https://christophm.github.io/interpretable-ml-book/ (consulted October the 26th 2023)
- Scikit-Learn: Cross-validation: evaluating estimator performance https://scikit-learn.org/stable/modules/cross_validation.html (visited on July 25th 2020)

Additional references

- Alpaydin, E. (2016). Machine Learning: The New AI. MIT Press. ISBN: 9780262529518
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). Springer.
- Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation nd Model Selection. Proceedings of the International Joint Conference on Artificial Intelligence IJCAI), 1995 (p./pp. 1137--1145).
- Rao, R. B. & Fung, G. (2008). On the Dangers of Cross-Validation. An Experimental Evaluation. SDM (p./pp. 588-596), : SIAM. ISBN: 978-1-61197-278-8