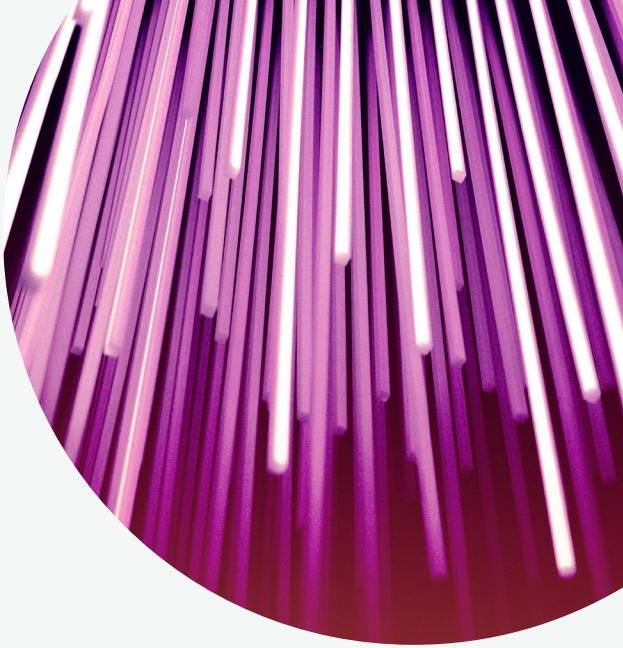
devoteam



January 2022

A History of Algorithms.

Understanding Data Science's Main Algorithms

Creative tech for Better Change

A History of Algorithms

Contents

Preface

Another way of looking at algorithms...

Discovering or reviewing the fundamental algorithms of data science... These are essential tools for 21st-century businesses, and a key skill for data professionals.

Algorithms are essential tools for solving data-related problems. In particular, they make it possible to process data and analyse it. These formulae have been created since the birth of computing, and are an integral part of the modern business world. Their use is experiencing a boom. This has been made possible by the expansion of computing capacities and the amount of data available, which has exploded. In the past two years, more data has been produced and collected than in the previous century.

Algorithms are increasingly being integrated into data analysis tools which are currently on the market. In fact, they are becoming more and more accessible, and it seems that they can be used, to a certain extent, without being an expert in the field. In this context, the data scientist in particular is essential for taking the analysis further and dealing with the complexity.

As data professionals, our specialities are diversifying and are increasingly in demand. Data scientists, data analysts, visualisation engineers, data product owners, data engineers, architects, etc., have one main objective: to support businesses in understanding their environment and realising their ambitions. Indeed, data and knowing how to handle it are considerable assets in the race for competitiveness, which has been amplified by the digital boom.

Algorithmics is a major subject that calls for key skills and expertise, above all to address the current needs of the business world. It potentially relates to all sectors, and levels of maturity in relation to the subject vary greatly from one organisation to another.

This booklet is a tool intended to help you discover or review the most commonly-used algorithms and their underlying mathematical processes. It is also an opportunity to see them in a different light, to move away from the actual codes a little and to look at simple use cases, as well as reading the opinions of the experts who have made this booklet possible. We hope that it will be useful to you in your daily use of algorithms.

Aurélien Bénard
Tech Lead and Manager of the
Data Science Guild

Algorithm No. 1 - The Decision Tree: the essential algorithm

The first chapter of this didactic booklet is about one of the simplest algorithms: the decision tree. A decision tree allows you to use your data to determine clear business rules, according to a target variable that you are seeking to explain. It is rarely used unaltered in machine learning, and is an elementary and essential tool that you must master to understand the algorithms that we will see later: the random forest (Algorithm No. 2) and the isolation forest (Algorithm No. 3).

Working principle

A decision tree makes it possible to make a target variable stand out from other, so-called explanatory, variables.

From a mathematical perspective: given a matrix (X) with m observations and n variables, associated with a vector (Y) to be explained, a relationship between X and Y must be found.

To do this, the algorithm will seek to divide the individuals into groups of individuals that are as similar as possible in terms of the variable to be predicted.

As a result, the algorithm produces a tree that reveals hierarchical relationships between the variables. It is therefore possible to quickly understand the business rules explaining your target variable.

Construction of rules

The decision tree is an iterative algorithm which, at each iteration, separates the individuals into k groups (generally k=2 or a “binary tree”), to explain the target variable.

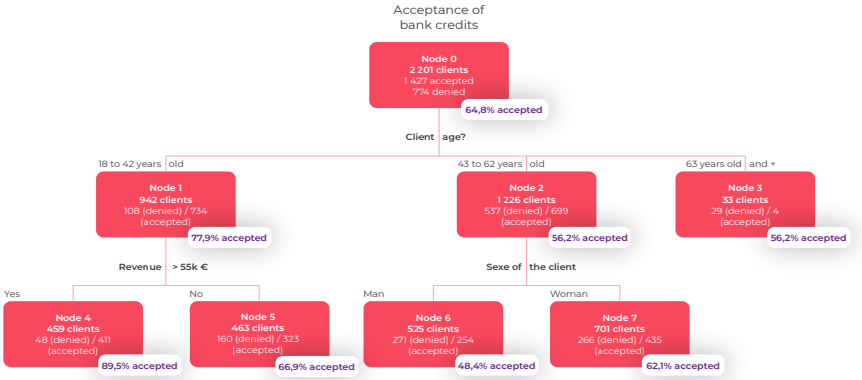
The first division (or “split”) is achieved by choosing the explanatory variable that will allow the best separation of individuals. This division creates sub-populations corresponding to the first node of the tree.

The splitting process is then repeated several times for each sub-population (previously calculated nodes), until the splitting process ends.

What are your chances of being accepted for a loan at the bank?

Let’s take a look at a specific case to illustrate all of the above. The diagram below helps us to understand the likelihood of being accepted for a bank loan. This is a hypothetical example to demonstrate the principle of reading a decision tree. Imagine that a manager of a large bank wants to know their own rules for accepting or not accepting a bank loan on the basis of the customer’s profile.

To do this, they appoint an advisor, who is responsible for interviewing customers. This adviser (who is something of a statistician at heart) summarises the results for the manager by proposing the decision tree below.



Let's take a closer look... In the roots of the tree, 2,201 customer files are under review.

Of these files, 1427 will be accepted (64.8%) and 774 will be rejected (35.2%). The explanatory variable that best separates the accepted files (our target variable) from the other files is customer age. Therefore, among 942 customers aged between 18 and 42 years (42.8% of all customers), the credit acceptance rate reaches 77.9% (i.e. 734 customers); while among the 33 customers aged 63+ years, the credit acceptance rate is just 12.1%. The main variable separating the population of customers aged between 18 and 42 years (Node 1) is income. As you can see, the 459 customers with an annual net income above EUR 55K have a credit acceptance rate of 89.5%.

Among customers aged between 43 and 62 years, the main explanatory variable for credit acceptance is gender. Thus, the rate of acceptance of a loan is 62.1% for women, compared to 48.4% for men in the same age group.

The limitations of this algorithm

Decision trees can sometimes lead to overfitting. This means that although the algorithm finds a rule that seems perfect for understanding and describing the data, this rule cannot be generalised. Worse still, in some cases, it is possible that the found rule may change drastically if a few more observations are added to your initial data.

Going a step further

There are a few key questions that you should ask yourself when creating a decision tree.

- Which explanatory input variables should you choose to create your tree? You will need to explain a target variable in terms of other variables. Try to find a causal process in your data (is there a causal relationship between the different variables in your data?).
- How can you process continuous data (e.g.: the height of an individual, the price of a property, etc.) and qualitative data (e.g.: the socio-professional category, etc.)? You need to pre-process the variables and select the decision tree model that works best.
- How can you define the optimal size of a tree? You have to think about pruning the tree (cutting it to a certain height). Indeed, a tree that is too profound (i.e. with lots of nodes) is always synonymous with overfitting.

To continue answering these questions, find out more about the three algorithms of the decision tree family: CHAID, CART and C4.5.

To summarise...

If you fully understand how this first algorithm works, you will easily understand the set methods which put several trees into competition, in particular the random forest (Algorithm No. 2) and the isolation forest (Algorithm No. 3).

The opinion of Idriss Brahimi
Data Scientist

"The decision tree is easy to understand. With other algorithms, however, a compromise between interpretability and performance is needed."

Algorithm No. 2 - Random Forest: intuitive and easy to train

The random forest is a vital algorithm in machine learning. It is also known as a "random decision forest." Suggested by Leo Breiman in 2001, this algorithm is based on the grouping of multiple decision trees. It is fairly intuitive and easy to train, and it produces generalisable results. The only downside is that the random forest is a black box that gives results that are difficult to read, i.e. not very explanatory.

Nevertheless, it is possible to limit this by using other machine learning techniques. This will be the focus of Algorithm No. 7, LIME - a key algorithm for making an explanatory model.

Random forest - working principle

A random forest is made up of a set of independent decision trees.

Each tree has a partial vision of the problem due to a double random selection:

- random selection with replacement of observations (rows of your database). This process is known as **tree bagging.**
- random selection of variables (columns of your database). This process is known as **feature sampling.**

All of these independent decision trees are ultimately put together. The prediction made by the random forest for unknown data is therefore the average (or the vote, in the case of a classification problem) of all the trees.

The basic idea of this algorithm is quite intuitive. For example, if your credit application is rejected by your bank, chances are that you will consult one or more other banks. Indeed, a single opinion is not usually enough to make the best decision.

The random forest works on this same principle. Rather than having a complex estimator which is capable of doing everything, the random forest uses several simple estimators (of lower individual quality). Each estimator has a fragmented view of the problem. All of these estimators are ultimately brought together to obtain a global view of the problem. The combination of all these estimators makes the prediction highly efficient.

Genesis of the algorithm

The major flaw of the decision tree is that its performance is highly dependent on the initial data sample. For example, the addition of some new data to the knowledge base can radically modify the model and change the results.

To fight against this flaw, we can use a multitude of trees - a forest of trees! - hence the name “random forest.”

The term “random” comes from the random double draw process that is applied to each tree, in relation to both the variables and the observations.

Practical illustration of the algorithm

A formula to remember:

Random forest = tree bagging + feature sampling.

Tree bagging

"Bagging" is short for "bootstrap aggregation." It is a process of randomly selecting observation samples (rows of data), determined by 3 key steps:

1. Constructing n decision trees, by randomly selecting n observation samples.
2. Training each decision tree;
3. To make a prediction on new data, each of n trees must be used, and the majority is determined from n predictions.

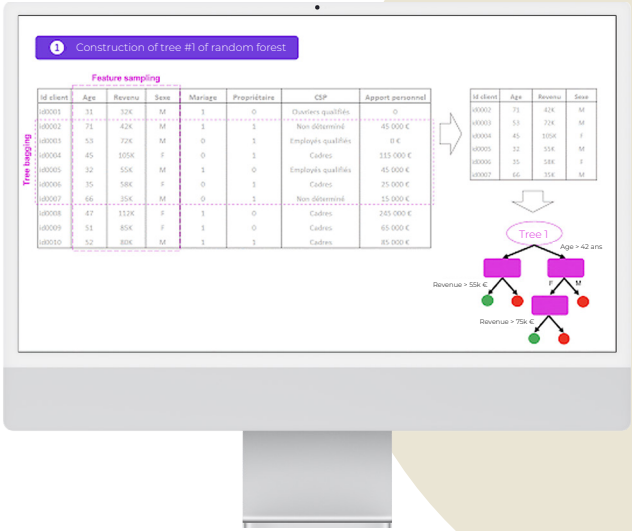
Feature sampling

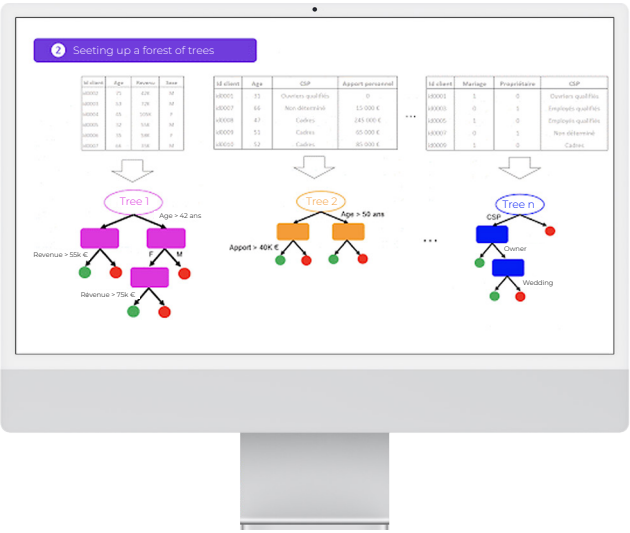
It is a process of randomly selecting variables (columns of data). By default, n variables for a problem with n variables in total are selected from the root of the decision tree.

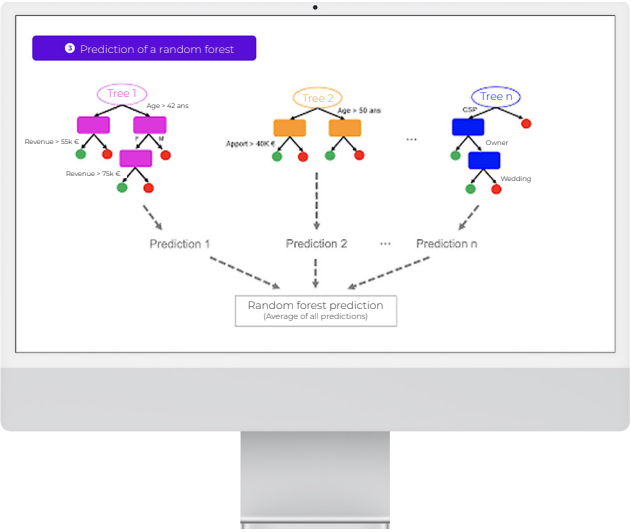
Going back to the previous example of credit acceptance, the basic idea of feature sampling is to ask each bank to study your loan application based on limited access to customer information. One bank will make its decision on the basis of only having, for

example, access to information relating to the age, CSP and annual income of the customer. Meanwhile, another bank will only have access to information relating to the marital status, gender and current credit rating of the customer.

This process makes it possible to weaken the correlation between the decision trees that could interfere with the quality of the result. In statistics, we say that feature sampling makes it possible to reduce the variance of the data set created.




2 Seeting up a forest of trees
Tree 1
Age > 42 ans
Revenue > 55k €
F
M
Revenue > 75k €
Tree 2
Age > 50 ans
Apport > 40K €
Tree n
CSP
Owner
Wedding


3 Prediction of a random forest
Tree 1
Age > 42 ans
Revenue > 55k €
F
M
Revenue > 75k €
Tree 2
Age > 50 ans
Apport > 40K €
Tree n
CSP
Owner
Wedding
Prediction 1
Prediction 2
Prediction n
Random forest prediction
(Average of all predictions)

Split/division criteria

As you know, a decision tree creates sub-populations by successively separating the leaves of a tree.

There are different separation criteria for constructing a tree:

- The Gini criterion organises the separation of the leaves of a tree by focussing on the class most represented in the data set. This must be separated as quickly as possible.
- The entropy criterion is based on the measurement of the prevalent disorder (as in thermodynamics) in the population studied. The construction of the tree aims to lower the global entropy of the leaves of the tree at each stage.

To summarise...

If you understand how the random forest algorithm works, you are ready to discover gradient boosting, which is also an ensemble method.

In the next chapter, you will find out how forest isolation works. This is a modern algorithm, and one of the most frequently used in anomaly or fraud detection cases.

The opinion of Idriss Brahimi
Data Scientist

"The random forest performs better than the decision tree. It is often used in machine learning competitors, and remains a model that is difficult to interpret."

Algorithme N°3 - Isolation Forest: for isolating anomalies

The main purpose of an anomaly detection algorithm is to identify atypical data that do not conform to other data.

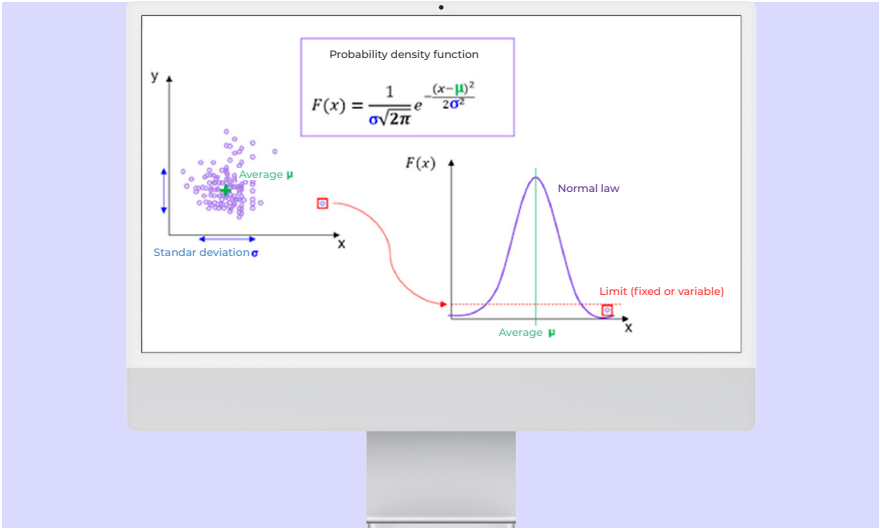
The problem is not simple. That is because it is not necessarily known in advance what characterises an anomaly. It is therefore up to the algorithm to learn an appropriate metric, with which to detect anomalies within the data.

Use cases encountered by our customers include:

- bank transaction fraud issues;
- the detection of material failures or damage as part of the optimisation of the predictive maintenance framework.

The detection of anomalies is usually an unsupervised learning technique that involves detecting data samples with characteristics which are very different to those of other samples.

For this, a first approach (theorised, for example, by the lean manufacturing and 6 sigma methods) involves calculating the mean and the standard deviation of the data to determine a “probability density” function. This function is then used to calculate the probability of the existence of an atypical sample of data. The rule is as follows: the sample is considered abnormal if this probability is below a certain threshold.



A threshold makes it possible to isolate anomalies - it can be fixed or adaptable. An adaptable threshold is constantly updated and adapts to the data in real time.

These techniques are very effective, but other more modern methods with which to recognise anomalies in data have been developed in recent years.

The isolation forest is used to calculate an anomaly score for each observation in a data set. This score gives a measure of the normality of each observation based on the data set. To calculate this score, the algorithm isolates the relevant data in a recursive way. It chooses a variable at random and sets a cut-off threshold at random, before evaluating whether this makes it possible to isolate a particular observation.

This type of algorithm is preferable in the case of a large-scale mathematical problem (with a large number of observations and variables).

Operation illustration

In the following example, forest isolation is applied to data that contains two variables (x and y) and a single anomaly (the extreme point that appears in red on the graph).

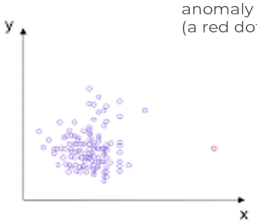
Principle of the algorithm

Objective:

We are going to make a series of random cuts (also called «splits») and we are going to count the number of cuts we need to make to succeed in isolating a particular observation.

The smaller the number of splits, the higher the chance that the isolated observation is an anomally.

Example: we will try to isolate an anomaly (red dot) from the dataset (a red dot) in the dataset below.



x	y
1,42	1,61
1,71	1,97
1,55	1,75
1,39	1,69
1,46	2,01
...	...
4,1	1,43
1,43	1,33

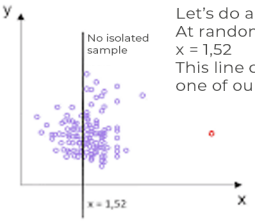
Objective:

We are going to make a series of random cuts (also called «splits») and we are going to count the number of cuts we need to make to succeed in isolating a particular observation.

The smaller the number of splits, the higher the chance that the isolated observation is an anomally.

Step 1:

Let's do a random 1st split in the data. At random, the variable X is split in x = 1,52
This line does not allow to isolate one of our samples. We continue...

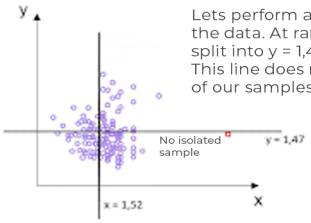


Objective:

We are going to make a series of random cuts (also called «splits») and we are going to count the number of cuts we need to make to succeed in isolating a particular observation.

The smaller the number of splits, the higher the chance that the isolated observation is an anomally.

Step 2:



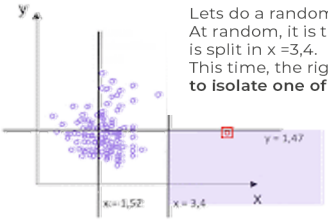
Lets perform a 2nd random split in the data. At random, the variable y is split into y = 1,47.
This line does not allow to isolate one of our samples. We continue...

Objective:

We are going to make a series of random cuts (also called «splits») and we are going to count the number of cuts we need to make to succeed in isolating a particular observation.

The smaller the number of splits, the higher the chance that the isolated observation is an anomally.

Step 3:



Lets do a random 3rd split in the data. At random, it is the variable y which is split in x =3,4.
This time, the right hand side **allows to isolate one of our samples!**

Result : You have understood the idea! In our example, just 3 splits (also known as “separations”) are enough to isolate the sample in red (a supposed anomaly, due to its distance from the other data). However, in the mass of data (i.e. “normal” data), it would have taken perhaps between 100 and 400 splits to isolate a sample.

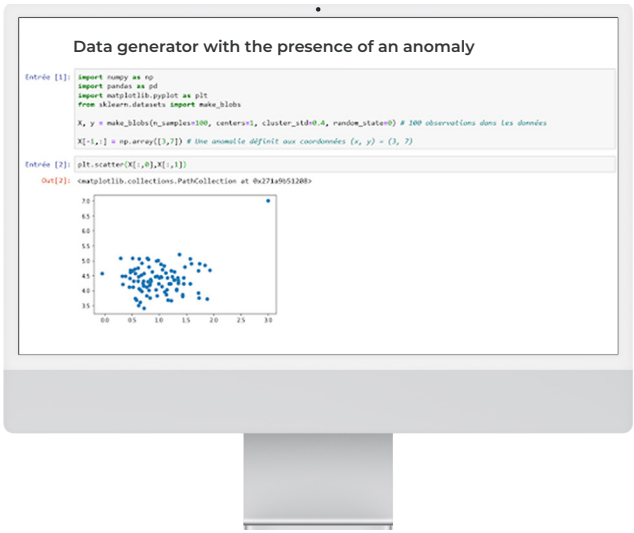
Remember: the lower the number of splits, the higher the probability that the observed sample is an anomaly.

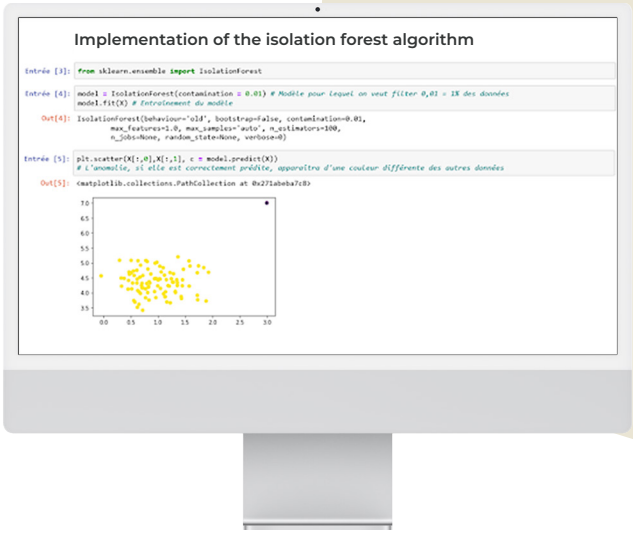
A forest of trees?

It should be noted that with the random splits process, it is possible to incorrectly isolate a sample from the mass of data ("normal" data). It is unlikely, but it can happen. The solution to overcome this risk of incorrectly isolating a normal sample involves generating several decision trees as estimators. Each of these trees will perform a sequence of random splits. The average of all of the results will then be considered. We can therefore disqualify the few minor errors that can be made by some of these estimators because "the majority will prevail." Again, ensemble techniques give our results a vital robustness.

Using the algorithm in Python

The sklearn.ensemble.IsolationForest class is used to instantiate a forest isolation model.





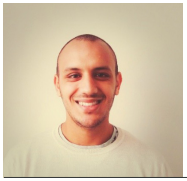
The result is conclusive: the algorithm has understood that there is an outlying point (an anomaly) in the data.

Note that in the implementation of sklearn, the threshold of points to be considered as abnormal is defined by the contamination rate. In our example, the rate of contamination is equal to 0.01 (i.e. 1%). Our data is therefore considered to contain 1% anomalies.

To summarise...

These first three chapters have been devoted to algorithms based on decision trees. These algorithms are widely used, and are extremely effective for solving data problems.

By the way, if you are hoping to work as a data scientist, it is highly likely that interviewers will ask you to describe how one of these algorithms works.



The opinion of Idriss Brahimi, Data Scientist

The anomaly score for each observation depends on the “contamination” parameter, which is set during the training phase. This implies that even before constructing the machine learning model, it is essential to have a good idea of the percentage of abnormal data.

Algorithm No. 4 - Linear regression to understand the main principles of machine learning

Many people think that linear regression is straightforward and that it is simply a question of finding a straight line through a scatter plot. They are wrong! We can even say that linear regression is a deceptively simple model. Let's find out why together.

Modelling well means modelling a phenomenon based on the simplest possible - and most easily interpretable - law. With this in mind, we are going to present the theoretical steps with which to analyse the implementation of a linear model for calculating the life of a mechanical component, in order to optimise the preventive maintenance of our customer.

We have decided to maintain a statistical approach, but mathematical formulae are nothing to be afraid of! This booklet is intended for the general public, and we hope that it will be a good way for you to understand in detail some fundamentals of machine learning.

The best linear regression model can be built by following three key steps:

1. Start by defining a cost function. This is a mathematical function that measures the errors we make when approximating data. It is also known as model-induced error.
2. To minimise this cost function, we must find the right parameters of our model to minimise the modelling error.
3. Select a method of resolving the problem. There are two methods:

- a digital resolution method, gradient descent;
- an analytical method, the “least squares” method.

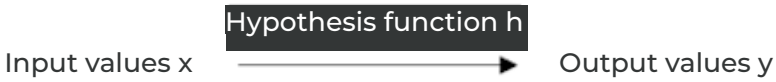
Step 1: Preparation of the cost function.

To create a cost function, we must start by defining an assumption function.

This hypothesis can be summed up as follows: the model depends on a set of n input variables, labelled x1, x2, ... xn. (These input variables correspond to the known data. For example, the number of inhabitants of a geographical area, the salary of an employee, or any other known variable.)

These input variables will influence an unknown target variable (Y), which we are seeking to predict.

From a mathematical perspective, we seek to determine the best hypothesis function which will make it possible to find an approximate linear relationship between the input variables and the target variable (Y).

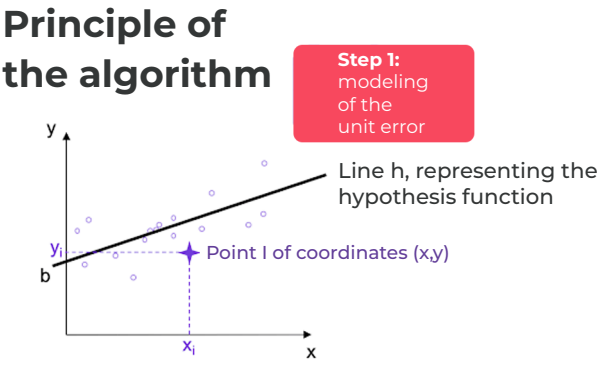


To simplify the approach, we take the simplest case of univariate linear regression. In this case, linear regression applies to a single input variable, and the hypothesis function (h) is written: h(X) = a x + b.

Our challenge is then to find the best approximation, i.e. the best pair (a, b) so that h is as near as possible to all of the points of our data. In other words, we will determine from the data the best linear relationship between the input value (X) and the target variable value (Y), by training the hypothesis function (h).

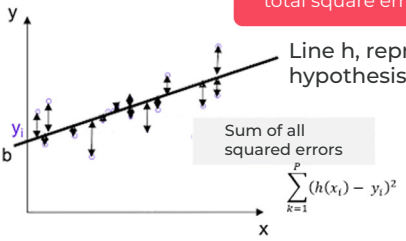
This hypothesis function is therefore a function that demonstrates the error between the prediction of the model and the actual data.

The hypothesis function assigns to each point X a value defined by h(xi), which is more or less the same as the target variable (yi). We can therefore determine the margin of error for xi as follows: h(xi) - yi. Each margin of error can be either positive or negative. Consequently, the sum of the margins of error may offset each other. It is, therefore, necessary to ensure that the contribution of each error is systematically penalised. The margin of error is then squared (see figure below).



Principle of the algorithm

Step 2: modeling of the total square error



We find the sum of all the unit errors for all the data points, to determine a quadratic error (squared error):

$$\sum_{k=1}^{P}(h(x_i) - y_i)^2$$

The cost function is then determined by weighting the sum of the squared errors by the number of points p in the learning base:

$$C(h) = \frac{1}{2p}\sum_{k=1}^{P}(h(x_i) - y_i)^2$$

In our univariate linear regression case, the cost function is determined as follows:

$$C(a, b) = \frac{1}{2p}\sum_{k=1}^{P}(a\,x_i + b - y_i)^2$$

The values of X and Y are given. In terms of its construction, cost function C is a function of the parameters of the hypothesis function (h). And, as shown in the figure above, the parameters of C define an affine line with:

- b, the ordinate at the origin of function h;
- a, the leading coefficient (or slope) of straight-line h.

Note: the same principle applies for multiple linear regression (i.e. linear regression with n input variables).

Step 2: Minimisation of the cost function.

Determining the best parameters (a, b) for the hypothesis function (h) comes down to finding the best straight line - the one that minimises the sum of all unit errors.

From a mathematical perspective, it is a matter of finding the minimum of the cost function.

Squaring the sum of the unit errors does two things:

- On the one hand, it ensures that the cost function is appropriately penalised by each unit error. Indeed, all errors are positive.
- On the other hand, it guarantees that the cost function is convex (if the cost function admits a minimum, this minimum is the global minimum of the function). We will outline the notion of convexity (see below).

We will also introduce the numerical method of gradient descent, which is used to mathematically find the minimum of the cost function, i.e. the best pair (a, b).

It is an iterative method that can be summed up as follows: if you drop a ball from the top of a hill, the ball will take the best slope at all times as it rolls to the bottom of the hill. The convexity of the cost function corresponds to the fact that we are certain that the hill is not uneven, with up-slope areas.

The mathematical formulation of this gradient descent problem is written as follows:

Step 1: Initialisation of the pair

Step 2: Iteration until convergence:

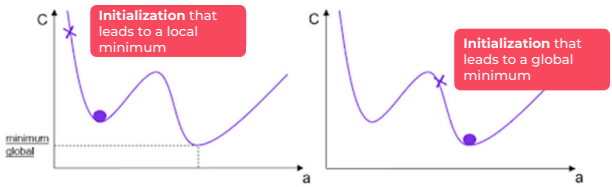
$$\begin{pmatrix} a_i \\ b_i \end{pmatrix} := \begin{pmatrix} a_i \\ b_i \end{pmatrix} - \alpha \frac{\partial}{\partial \beta_i} C(a,\ b)$$

In each iteration, the best slope is found with our function C, which goes over the iterations toward the minimum of the cost function.

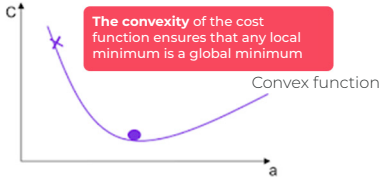
The major problem lies in the case of a non-convex cost function. Indeed, in this case, it may be that according to different initialisations of the descent of the gradient, we converge towards a local minimum for C.

The convexity of the cost function makes it possible to overcome this problem, as any local minimum is the global minimum of the function (refer to the figure below for a better understanding).

Case of a non-convex cost function



Case of a convex cost function



In the gradient descent formula above, the speed of convergence is determined by the factor α in front of the partial derivative C. This factor is called the learning rate, and represents the speed of modification of each parameter during each iteration.

- The larger α is, the greater the modification of the parameters between two successive iterations (therefore, the greater the probability of "missing" the minimum or of diverging).
- Conversely, the smaller α is, the more likely we are to converge to the minimum (in contrast, the convergence process takes longer).

Gradient descent is the numerical resolution approach to finding a solution to the modelling problem. This method makes it possible to find, in an iterative way, the best model that minimises the error by seeking the best slope up to the global minimum of the cost function.

This method is particularly suitable for large volumes of data, and makes it possible to reach a solution as quickly as possible.

There is another approach: an analytical approach.

This approach involves mathematically solving linear regression. Without going into too much mathematical detail, the so-called least squares method provides an analytical solution to a problem. If you want to know more, a form of analytical resolution for linear regression is written like this:

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

It's up to you to elaborate on this, if you need to.

And in practice...

- One of the delicate points of the implementation of a linear regression model comes from the instability of the prediction when faced with the integration of new observations in the data (in other words, the coefficients of each explanatory variable can change drastically if a few additional observations are added to the data),
- It is not usually easy to choose the explanatory input variables to take into account in the model. Questions must then be asked about the nature of the process between the variables. What are the causes for, and effects of, including a specific variable in the model? Immutable relationships between variables must also be found. (These are like the laws of physics, which are universal and therefore do not change - think of the laws of attraction, vibration, transmutation of energy, etc.)
- An additional step of normalising the input variables is vital in the case of multivariate linear regression. This normalisation involves transforming all the variables at the input of the model so that they will evolve on the same scale (to enable the gradient descent algorithm to work correctly),
- The linear model is not suitable for all the physical phenomena involved (e.g.: the phenomena of thermal heating are generally modelled by quadratic relations between electrical and physical measurements). If the phenomenon cannot be modelled by a linear relationship between input and target variables, it is then necessary to find a polynomial function. (But in this case, beware of overfitting - i.e. finding false relationships in your data!)
- Remember the positive points of the linear model: it is an interesting approach because the model is simple to explain to businesses, and the model is explanatory (the coefficients of each normalised variable indicate the importance of the variable in the relationship).

To summarise...

We have seen that linear regression allows you to familiarise yourself with the main principles of constructing a machine learning model. In a later chapter, we will look at how to improve the stability of linear regression models by penalising LASSO, Ridge or ELASTIC NET.

The opinion of Houssam Alrachid, Data Scientist

"When selecting multiple linear regression, adding independent variables increases the variance explained in the dependent variable. Therefore, adding too many independent variables without any theoretical justification can result in an overfitting model."

Algorithm No. 5 - Understanding the “k-nearest neighbours” method

The k-nearest neighbours (k-NN) algorithm is relatively simple to understand. First, you need to know that it is a supervised learning algorithm that allows us to solve not only a classification problem, but also a regression problem.

As a reminder, supervised learning is when an algorithm needs labelled input data (i.e. inputs with a label) and output data (i.e. data that the algorithm will have to predict later).

The idea is as follows: in a learning phase, the algorithm must be trained to learn the relationship between the data inputs and outputs.

This is a key point. In fact, supervised learning is based on the organisation of data according to a series of {input-output} pairs.

Once the algorithm has been trained, it can predict an output value, solely on the basis of the input values.

The k-nearest neighbours method and its principles

Once an algorithm learning phase has been completed, the algorithm is able to make a prediction based on a new unknown observation by finding the observation that is nearest to it in the training data set.

The algorithm then assigns the label of this training data to the new observation, which was previously unknown.

The k in the formula “k-nearest neighbours” means that instead of just the nearest neighbour of the unknown observation, a fixed number (k) of neighbours in the training data set can be considered.

Ultimately, we can make a prediction based on the majority class in this neighbourhood.

This principle may seem complicated to explain, but take a look at the example below which simply illustrates the general idea of this method.

A simple illustration of the algorithm’s principle of operation :

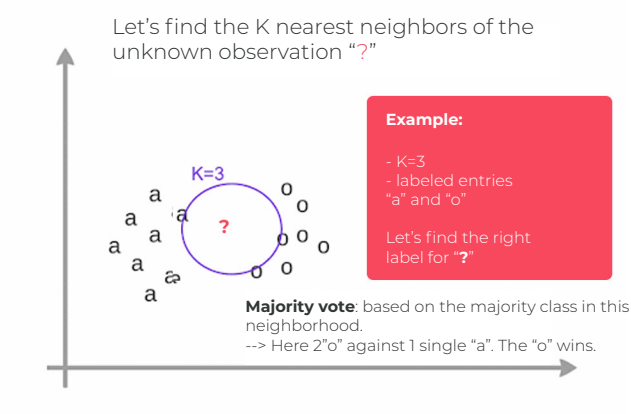


Figure 1: An example of the k-NN algorithm and its working principle

In summary, the steps of the algorithm are as follows:

From the input data:

- A definition function for the distance (also known as a similarity function) is selected between observations. (We will come back to this in the next paragraph.)
- We set a value for "k," the number of nearest neighbours.

Pseudo code :

We must do the following for a new unknown input observation for which we want to predict the output variable.

- Step 1 - Calculate all the distances between this observation and the other observations in the data set.
- Step 2 - Select the observations from the data set which are the "nearest" to the predicted observation.

Note: To determine whether one value is "nearer" than another, a distance calculation-based similarity function is used (see the next paragraph).

- Step 3 - Take the values of the selected observations:

  If a regression is performed, the algorithm will calculate the average (or the median) of the values of the selected observations.

  If a classification is performed, the algorithm will assign the label of the majority class to the previously unknown data.

- Step 4 - Return the value calculated in step 3 as the value that was predicted by the algorithm for the previously unknown input observation.

For this algorithm, the choice of the number (k) and the choice of the similarity function are steps which can cause the results to vary greatly.

Metrics for evaluating the similarity between observations

As we have just said, to measure the proximity between observations, a similarity function must be imposed on the algorithm.

This function calculates the distance between two observations and estimates the affinity between the observations like this: “The nearer two points are to each other, the more similar they are.”

However, this trivial statement comes with some mathematical nuances, and there are many similarity functions in related literature.

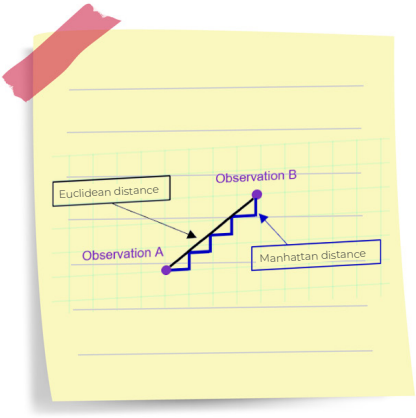


Figure 2: Distance between two points: Euclidean Distance vs Taxicab Geometry/ Manhattan Distance (lots of paths between A and B)

Euclidean Distance is one of the best-known similarity functions. We used this function in the example above.

It is the most intuitive similarity function because it formalises the idea of distance - the straight line distance between two points in space.

Manhattan Distance (also known as Taxicab Geometry) corresponds to a movement at a right angle on a chessboard.

If you are interested, I invite you to look at the mathematical definitions of other metrics such as the Minkowski Function, the Jaccard Function, the Hamming Distance, the Levenshtein Distance, the Jaro Metric, or even the Jaro-Winker Distance.

You need to know that the distance function is generally chosen according to the types of data that we are handling and the mathematical or physical meaning of the problem to be solved. The Euclidean Distance is a good choice to start with for quantitative data (e.g.: height, weight, salary, turnover, etc.), as it is very schematic.

The Manhattan distance can be interesting for data that are not of the same type (e.g. data that have not been put on the same scale).

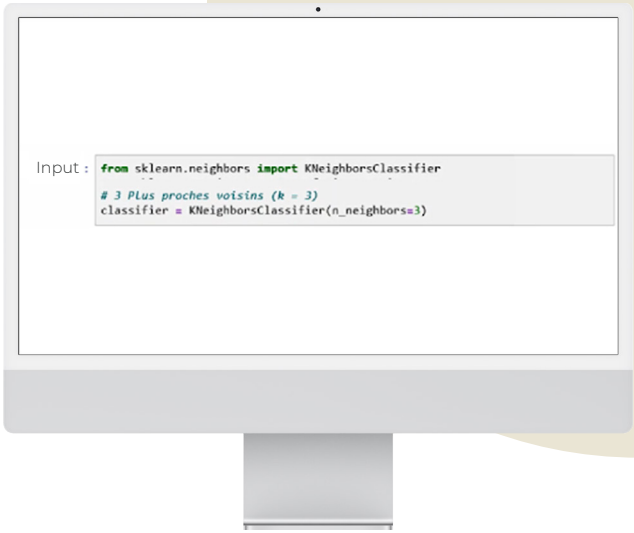
Also, I strongly advise you to deepen your subject knowledge in order to master the classic use cases of the various similarity functions, according to the problem to be solved.

Your turn to code... (Python)

This paragraph provides some elements to help you to code the algorithm yourself. All of the machine learning models in scikit-learn are implemented in their own classes, which are called the Estimator classes.

The class for the k-nearest neighbours algorithm is KNeighboursClassifier, from the neighbours module.

Before we can use the template, it needs to be instantiated as follows:

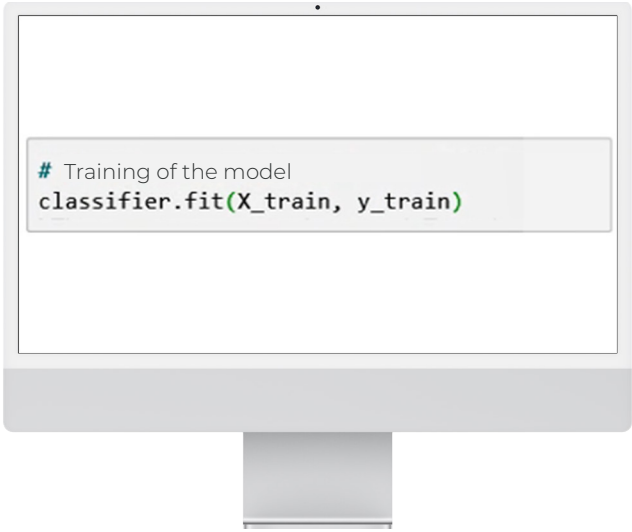


In our example, the classifier object encapsulates the algorithm that will be used to create the model from the training data, as well as the algorithm that will make predictions for new observations.

To create the k-NN model, we must first split the data into 2 databases: one for training the algorithm, and one for testing the performance obtained.

Note: test_size = 0.2 = 20% gives the size of the test database, i.e. 20% of the initial data sample.

Next, we use the fit method of the classifier object, which takes as arguments the training dataset X_train and the test dataset y_train.



The fit method returns the k-NN object itself.

The default algorithm configuration is as follows:

KNeighboursClassifier (algorithm='auto', leaf_size=30, metric='minkowski', n_jobs=1, n_neighbours=3, p=2, weights='uniforms')

You will notice that all the parameters except for the number of nearest neighbours have their default value set to n_ neighbours=3.

To make a prediction about new unknown data, we create a vector that contains the known input data. The X_new vector is below.

We will then use the predict method of the classifier object.

The limitations of the K-nearest neighbours algorithm

Although the operation of the nearest neighbours algorithm is easy to understand, the main drawback of the method is its cost in terms of complexity. Indeed, the algorithm searches for the k-nearest neighbours for each observation. Therefore, if the database is very large (a few million rows), the calculation time can be extremely long. Special attention must therefore be paid to the size of the input data set.

Another focal point: the choice of similarity function can lead to very different results between models.

It is therefore recommended to create a rigorous state of the art to better choose the similarity distance according to the problem to be solved. Above all, it is recommended to test several combinations to estimate the impact on the results.

Despite everything, remember that if the k-NN algorithm is interesting from a didactic point of view, it is almost never used as it is by data scientists.

The opinion of Safa Hassine, Data Scientist

"For high-dimensional data, it is necessary to use dimensionality reduction techniques (such as PCA, ACM, CCA) before applying the k-NN algorithm, otherwise the Euclidian Distance will be useless for measuring similarity distances in large space. A good use case for KNN is to see which common items customers buy together, to encourage them to complete their basket. On the one hand, this creates a personalised customer experience by facilitating the search for necessary items; and one on the other, it increases income."


LIVE DATA
LIVE DATA
LIVE DATA
105.32
87.15
60.36
35.22
20.12

Algorithm No. 6 – And how about if we used LASSO? To understand linear regularisation techniques

Before reading this chapter, we advise you to carefully re-read the section about Algorithm No. 4 and linear regression. We have seen that linear regression models are based on the minimisation of the residual error for the estimation of coefficients.

But it has also been specified that the cost function could generate strong instability in the results of the estimate.

But what exactly does that mean? It means that a few slight changes to the data can produce very different patterns.

For example, you have obtained a linear regression model from a database. Then, following a few corrections to your database, 2% of your data changes radically. You recalculate your regression model, even though you are confident that there will be no changes to your model.

And what a surprise! Your model (i.e. the value of the coefficients of each explanatory variable) has been totally transformed.

Fortunately, there are techniques to stabilise linear regression models and therefore avoid nasty surprises like this. We are going to look at three regularisation techniques: Ridge, LASSO and Elastic Net.

Remember: The best linear regression model can be built by following three key steps.

1. The first step is to create a cost function. This is a mathematical function that measures the errors we make when approximating data. This is also known as model-induced error.
2. Then comes the minimisation step of this cost function: we must find the best possible parameters for our model in order to minimise the modelling error.
3. Then a method of resolving the problem must be selected. There are two methods:

    - a numerical resolution method, gradient descent
    - an analytical method, the “least squares” method

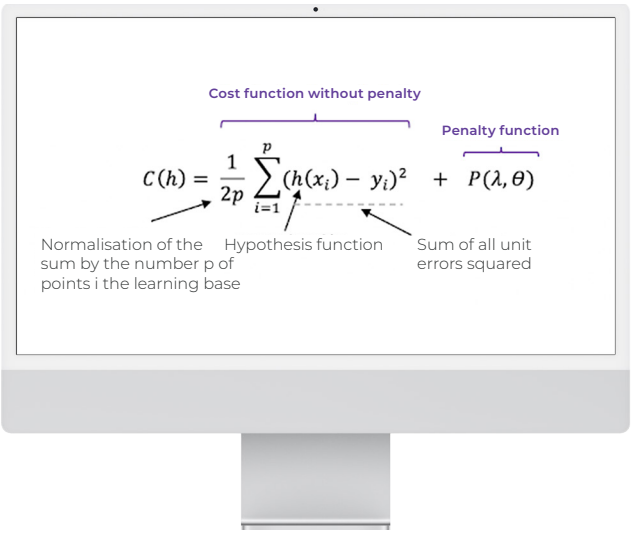
Working principle of regularisation methods

Regularisation techniques are used in the context of linear regression, and to limit the problems caused by the instability of predictions. These methods make it possible to distort the space of solutions, to prevent the appearance of values which are too high. We use the word “shrinkage” to evoke this spatial transformation of the space of search for solutions.

This is a question of slightly modifying the cost function of the linear regression problem by supplementing it with a penalty term.

If the three key steps for constructing a regression model remain unchanged, it is still necessary to adapt the cost function a little.

The cost function with penalisation is written as follows:



The values of X and Y are given. In terms of its construction, cost function C is a function of the parameters of the hypothesis function (h).

And the parameters of C define an affine line.

This is the function $P(\lambda, \theta)$ that manages the penalty according to a lambda parameter which is set empirically in order to obtain the best results.

We suggest that you take a detailed look at three methods based on this principle.

First regularisation method: penalised Ridge regression

Ridge regression is one of the most intuitive penalisation methods. It is used to limit the instability of predictions linked to explanatory variables that are too correlated.

This penalisation function is based on the so-called L2 norm which corresponds to the Euclidean Distance. The Ridge regression is therefore the equivalent of minimising the following cost function:

$$C(h) = \frac{1}{2p}\sum_{i=1}^{p}(h(x_i) - y_i)^2 \quad + \lambda \sum_{j=1}^{n} \theta_j^2$$

The Ridge penalty will decrease the distance between the possible solutions, based on the Euclidean Measure.

Configuration of the lambda parameter:

- When lambda is near to zero, the classic solution is used, without penalties.
- When lambda is infinite, the penalty is such that all parameters are set to zero.
- When lambda is increased, the bias of the solution is increased, but the variance is reduced (cf. the definition of the bias-variance trade-off).

As with classic linear regression, Ridge regression can be solved by gradient descent, and by iterating until the convergence of cost function C.

Therefore, Ridge regression makes it possible to circumvent problems of collinearity (where explanatory variables are very strongly correlated) in a context where the number of explanatory variables at the input of the problem is high.

The main weakness of this method relates to the difficulties of interpretation because without selection, all the variables are used in the model.

LASSO penalisation method

You might know the term “lasso” from stories about the Wild West. In this context, though, it stands for “Least Absolute Shrinkage and Selection Operator.” The acronym LASSO contains terms relating to the notion of shrinkage of the search space, and other terms relating to a variable selection operation (“selection operator”).

The LASSO method introduces the following penalty term to the cost function formulation:

$$P(\lambda, \theta) = \lambda \sum_{j=1}^{n} | \theta_j |$$

This time, another norm is used. The L1 norm corresponds to the Manhattan norm (distance corresponding to a movement at right angles on a chessboard - unlike a Euclidean Distance, which corresponds to a movement in a straight line).

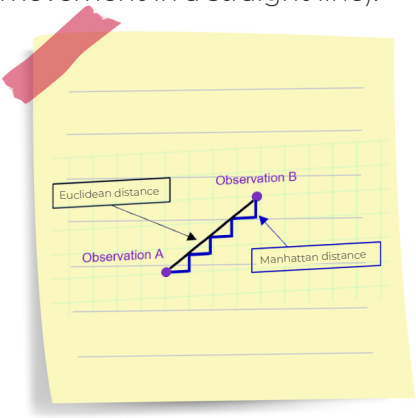


Figure 1 : Distance between two points: Euclidean Distance vs Taxicab Geometry (lots of paths between A and B)

It is a much less intuitive distance than the Euclidean Distance, which allows a penalty and decreases the distance between the possible solutions on the basis of the L1 norm.

The cost function to be minimised by LASSO is written as follows:

$$C(h) = \frac{1}{2p} \sum_{i=1}^{p} (h(x_i) - y_i)^2 \quad + \lambda \sum_{j=1}^{n} | \Theta_j |$$

Note that there is no analytical solution for LASSO, and so we can use an iterative algorithm or the gradient descent method to solve this equation.

LASSO does have some good qualities: it is a form of penalisation which makes it possible to set certain coefficients of explanatory variables to zero (unlike Ridge regression, which can lead to coefficients near to 0, but never exactly zero).

LASSO is therefore an algorithm that also allows the simplification of the model, by eliminating variables.

We will now geometrically illustrate the effects of a Ridge Vs LASSO regularisation on the model parameters with the two graphs below.

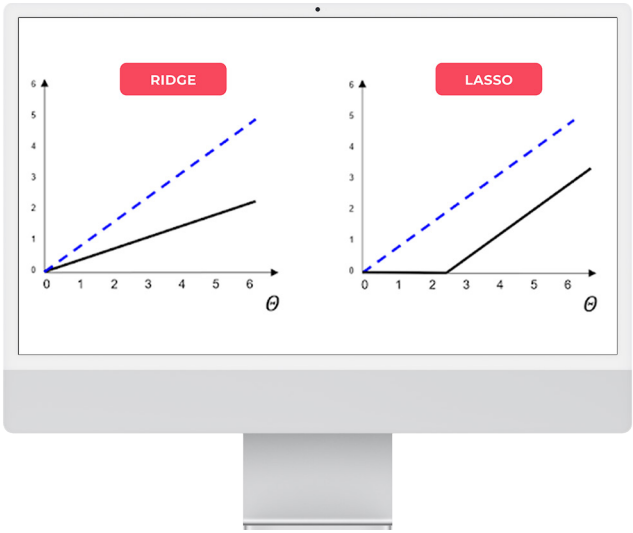


Figure 2: Geometric comparison between Ridge and LASSO regularisation on model parameters

The black line represents the regularisation function, while the dotted blue line represents an unregulated line. It can be seen that the Ridge regression scales the coefficients by dividing them by a constant factor, whereas LASSO subtracts a constant factor by truncating to 0 below a certain value.

Elastic Net = Ridge + LASSO

In practice, Ridge regression gives better results than penalised LASSO regression, especially if the explanatory variables of the problem to be solved are highly correlated (this is the classic use case for this penalisation method).

But Ridge regression does not reduce the number of variables. To find a compromise between the two penalisation techniques, Elastic Net regularisation combines the two approaches.

The cost function is determined:

$$C(h) = \frac{1}{2p}\sum_{i=1}^{p}(h(x_i) - y_i)^2 \quad + \lambda \sum_{j=1}^{n}[\frac{1}{2}(1 - \alpha)\theta_j^2 + \alpha\,|\theta_j|]$$

Where the alpha parameter is a parameter defining the balance between Ridge and LASSO.

- For alpha = 1, the cost function matches that of LASSO.
- For alpha = 0, the Ridge regression is found.

It is possible to adjust the penalty depending on the application case.

- When alpha approaches 1, we can have a behaviour near to the LASSO while eliminating the problems relating to strong correlations between explanatory variables.
- When alpha increases from 0 to 1 (for a given lambda), the number of variables removed from the model (leading to a zero coefficient) increases until a smallest model is obtained, obtained by LASSO.

To summarise...

To conclude, regularisation makes it possible to shrink the space formed by the solutions of the modelling problem by linear regression. To do this, we add a term that will penalise the coefficients to the cost function. Minimising the cost function will therefore minimise the regression coefficients.

3 types of regularisation have been presented in this chapter: the Ridge, LASSO and Elastic Net methods.

The opinion of Houssam Alrachid,
Data Scientist

"Sometimes, LASSO regression can cause bias in the model, meaning that the prediction depends too much on a particular variable. In these cases, Elastic Net better combines LASSO and Ridge regularisation, but does not easily eliminate the high collinearity coefficient."

Algorithm No. 7 - LIME or SHAP to understand and interpret your machine learning models

Machine learning (ML) models are becoming more and more complex. Indeed, a sophisticated model (e.g. XGBoost boosting or deep learning) generally leads to more accurate predictions than a simple model (e.g. linear regression or decision tree).

There is therefore a trade-off between the performance of a model and its interpretability. What a model gains in performance, it loses in interpretability (conversely).

What exactly is “model interpretability”?

Interpretability is defined as the ability of humans to understand the reasons for a decision made by a model. This criterion has become instrumental for many reasons.

On a scientific level, the development of knowledge and progress depends on a deep understanding of the phenomenon studied. It is therefore unimaginable for a data scientist to let a machine learning model operate without seeking to determine the influential variables, or without aiming to verify the consistency of the results against expert knowledge of the field, etc. It is about understanding, and having confidence in - and proof of - the consistency of the model.

On an ethical level: let's imagine a situation where someone is suffering from cancer. They are denied treatment because of the sole decision of an algorithm. Moreover, this algorithm is complex and therefore no surgeon is able to justify such a decision. This situation is not acceptable.

On a legislative level: Article 22 of the General Data Protection Regulation (GDPR) states that a person must not be the subject of a decision based exclusively on automated processing and emanating solely from the decision of a machine.

In this chapter, we will present two methods with which to interpret machine learning models: the LIME and SHAP algorithms. These two methods operate at the output of a complex model, a black box with poorly understood functioning.

Before looking at the specificities of each algorithm, we suggest that you quickly review what characterises the main interpretability methods.

Interpretability methods

The different interpretability methods can be defined according to the following typologies:

- Agnostic versus specific interpretation methods: Agnostic methods can be used for any type of model. In contrast, specific models can only be used to interpret a specific family of algorithms.
- Intrinsic versus post-hoc methods: In intrinsic methods, interpretability is directly linked to the simplicity of the model; while in post-hoc methods, the model cannot be interpreted because it is too complex from the outset.

- Local versus global methods: Local methods give an interpretation for a single observation, or a small number of them. In contrast, global interpretation methods make it possible to explain all the observations at the same time, globally.
- A priori versus a posteriori methods: A priori approaches are used without assumptions regarding the data, and prior to the creation of the model. In contrast, a posteriori approaches are used after the model has been created.

The current state of the art around the interpretability of machine learning models shows that there is a strong desire to mix the different methods: {intrinsic & global} or {post hoc & global & agnostic} or even { post-hoc & local & agnostic}.

LIME

The LIME algorithm (Local Interpretable Model-agnostic Explanations) is a local model that seeks to explain the prediction of an individual by analysing its surroundings.

LIME has the particularity of being a model:

- Interpretable. It provides a qualitative understanding between the input variables and the response. The input-output relationships are easy to understand.
- Locally simple. The model is globally complex, so it is necessary to look for simpler answers locally.
- Agnostic. It is able to explain any machine learning model.

To do that:

- 1st step: the LIME algorithm generates new data, in a neighbourhood which is near to that of the individual to be explained.
- 2nd step: LIME gives rise to a transparent model on the basis of the predictions of the complex “black box” model that we are trying to interpret. It, therefore, learns using a simple and therefore interpretable model (for example, linear regression or a decision tree).

The transparent model, therefore, acts as a substitute model with which to interpret the results of the original complex model.

The main drawback of the LIME method relates to its local operation. In addition, LIME does not make it possible to generalise the interpretability resulting from the local model on a more global level.

Application with Python

There is a LIME library in Python. According to the input data type, you can use:

- lime.lime_tabular for data tables
- lime.lime_image for image databases
- lime.lime_text for a text corpus.

Our use case: explaining the churn score of a particular customer, based on their transactional data. Different questions need to be asked: How can different score classes be explained? What makes this customer's score different from those of other customers? What behaviour causes this particular customer to have a certain score?

LIME has enabled us to explain the predictions of a complex XGBoost model, which boosts machine learning.

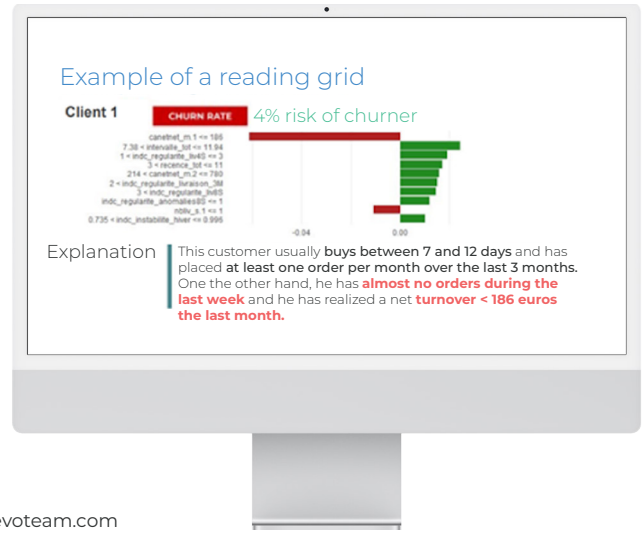
```
import lime.lime_tabular

classifier_lime = lime.lime_tabular.
LimeTabularExplainer(train_data_X.values,
                                mode='regression',
                                training_labels=train_data_y,
                                feature_names= data_
dataset.feature_names,
                                    categorical_
features=['CHAS'])

lime_results = classifier_lime.explain_instance(test_
data_X.values[0], sklearn_regressor.predict)


lime_results.show_in_notebook()
```

Results:



The difference in colour shows us which are the favourable and unfavourable factors for the interpretation of the churn:

- in green: favourable factors help to increase the predicted value.

In our example, the customer makes regular purchases. There are no anomalies for this customer.

- in red: unfavourable factors help to increase the customer's churn risk score.

In our example, the customer has hardly made any purchases in the past week. This should sound a warning - the customer may be losing interest in the brand. In addition, the customer's lifetime value over the last month is low (€186) compared to their previous purchases. We have to find a way to win them back.

SHAP

The implementation of SHAP is based on a method used for estimating Shapley values. There are different estimation methods such as KernelSHAP (a method inspired by LIME) or TreeSHAP (a decision tree-based method).

Shapley value estimation principle

For a given individual, the Shapley value of a variable (or several variables) is its contribution to the difference between the value predicted by the model and the mean of the predictions of all the individuals.

To do that:

- Step 1 in calculating Shapley values for a particular individual: Simulate different combinations of values for the input variables
- Step 2: For each combination, calculate the difference between the predicted value and the average of the predictions. The Shapley value of a variable, therefore, corresponds to the average of the value contribution, according to the various combinations.

Here is a simple example

Imagine that we have created a model for predicting apartment prices in Paris. For a given observation, the model predicts the class “Price > €12,500/m2” with a score of 70% for an apartment with a balcony (presence_balcony variable = 1). When changing the balcony value to 0 (no balcony), the score drops by 20%, and the contribution of the balcony value = 1 is 20%.

Things are indeed more complicated in practice, because to obtain a correct estimate of the Shapley values, we need to add all of the values of each variable of the model, and divide the sum by the number of values in the data set. The calculation time could then become extremely important.

Some lines of Python coding for testing SHAP

In Python, alibi and SHAP libraries implement the methods for estimating the values of Shaply KernelSHAP, TreeSHAP (for use cases on the basis of data tables) and DeepSHAP (for deep learning use cases).

```
import shap

classifier_shap = shap.KernelExplainer(sklearn_
regressor.predict, data_train_X)
```

```
shap_results = classifier_shap.shap_values(data_
test_X.iloc[0])

shap.waterfall_plot(classifier_shap.expected_
value,shap_values,data_test_X.iloc[0])
```

To summarize...

In machine learning, it is important to find a fair compromise between a powerful model versus an interpretable model.

With LIME or SHAP, you can make an initially complex model interpretable and therefore promote its adoption by business users. If understanding a model is essential in a scientific process, legal constraints now require that a decision must not be made based solely on the result of an automatic algorithm.

Remember that the SHAP algorithm currently meets the normative requirements imposed by GDPR.

If you are interested in this subject, we invite you to read about other ways of interpreting machine learning models such as: techniques based on the analysis of "ICE and PDP graphs," so-called "permutation feature Importance," "counterfactual explanations," or even "anchors."

The opinion of d'Alaeddine Othmani,
Data Scientist

"Predictive models which are too complex tend to be penalised. This is not only because they cannot be generalised for specific, real-world situations, but also because they are difficult to interpret. When such interpretability methods are associated, with complex prediction models, their contribution is invaluable. This enable us to avoid the cost of insights that would not be available to commercial decision makers."

The last word from Sylvain Le Corff, teacher of statistics at Telecom Sud Paris

“Every data scientist has to master these tools”

This booklet provides an overview of the algorithms that form the foundations of data science and machine learning. It reminds us that an AI solution is indeed based on many technical and mathematical elements (modelling of a regression or classification problem, defining criteria to be optimised, etc.).

All data scientists must master these tools, as well as the different principles and assumptions guaranteeing their correct use. It also highlights that the development of a complete solution is based on a number of choices that enable the definition of an original solution: choice of model, choice of regularisation, determination of a potential approximation (Monte Carlo type) of the cost function, selection of variables... The various digital professions focus on understanding and mastering such tools, in order to ensure that the final functionalities of the algorithm meet the needs of the customer.

Although knowledge of these tools is invaluable, many challenges remain. Above all, these relate to measuring the uncertainty of the predictions provided by AI models. It is still a question of developing simulation methods in order to facilitate the provision of statistical guarantees, which do not exist for many state-of-the-art solutions.

Finally, it seems important to remember the crucial role of data visualisation, analysis and pre-processing. The “data-centric AI competition” promoted by Andrew Ng has recently highlighted this. Numerous models are available, but their performance can be improved with a deeper understanding of the data sets. This highlights once again the crucial role of data analysts and not just that of the complexity of the models.

Sylvain Le Corff, teacher of statistics at Telecom Sud Paris

Acknowledgements

Special thanks must go to Sylvain Le Corff, teacher of statistics at Telecom Sud Paris, and to the Data Science team: Aurélien Bénard, Houssam Alrachid, Alaeddine Othmani, Safa Hassine and Idriss Brahimi.

About Devoteam

Devoteam is a leading consulting firm focused on digital strategy, tech platforms and cybersecurity. By combining creativity, tech and data insights, we empower our customers to transform their business and unlock the future.

With 25 years' experience and more than 8,500 employees across Europe, the Middle East and Africa, Devoteam promotes responsible tech for people and works to create better change.

Creative tech for Better Change


Creative tech for Better Change