




















# **Processo de aplicação de modelos de machine learning**

Bizagi Modeler

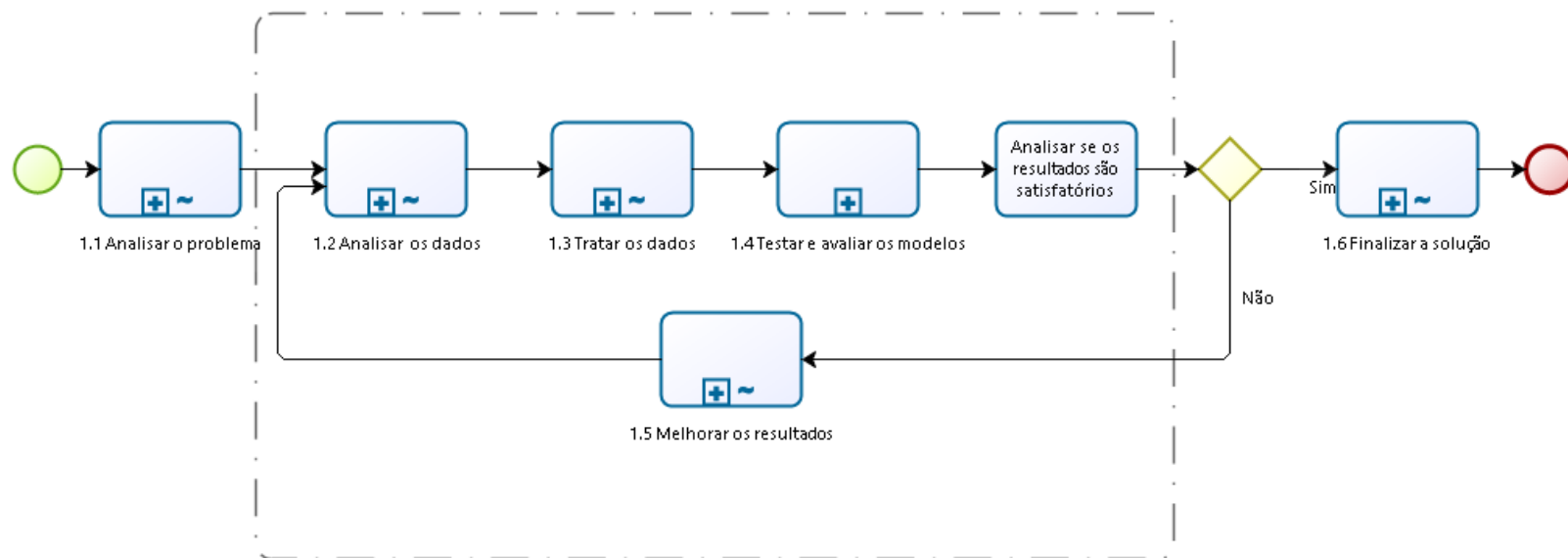


## Índice

PROCESSO DE APLICAÇÃO DE MODELOS DE MACHINE LEARNING .....	1
BIZAGI MODELER .....	1
1 1. SOLUCIONAR PROBLEMA COM MACHINE LEARNING .....	5
1.1 MAIN PROCESS .....	6
1.1.1 Elementos do processo .....	6
1.1.1.1  1.4 Testar e avaliar os modelos .....	6
1.1.1.2  Analisar se os resultados são satisfatórios .....	6
1.1.1.3  1.5 Melhorar os resultados .....	6
1.1.1.4  1.1 Analisar o problema .....	6
1.1.1.5  1.2 Analisar os dados .....	7
1.1.1.6  1.3 Tratar os dados .....	7
1.1.1.7  1.6 Finalizar a solução .....	7
1.2 1.4 TESTAR E AVALIAR OS MODELOS .....	8
1.2.1 Elementos do processo .....	8
1.2.1.1  Definir como os modelos serão testados e avaliados .....	8
1.2.1.2  Executar testes .....	10
1.2.1.3  Registrar resultados .....	10
1.2.1.4  Analisar resultados .....	10
1.2.1.5  Registrar análise dos resultados .....	10
1.2.1.6  Definir que modelos serão utilizados .....	10
1.2.1.7  Registrar modelos .....	11
1.3 1.5 MELHORAR OS RESULTADOS .....	11
1.3.1 Elementos do processo .....	11
1.3.1.1  Aprimorar os modelos .....	12
1.3.1.2  Registrar modificações .....	12
1.3.1.3  Registrar modelos .....	12
1.3.1.4  Registrar modelos .....	12
1.3.1.5  Construir ensembles .....	12

1.3.1.6	<input type="checkbox"/> Extreme feature engineering .....	13
1.4	1.1 ANALISAR O PROBLEMA .....	14
1.4.1	Elementos do processo .....	14
1.4.1.1	<input type="checkbox"/> Analisar quais são as premissas do problema .....	14
1.4.1.2	<input type="checkbox"/> Registrar qualidade das soluções pré existentes .....	15
1.4.1.3	<input type="checkbox"/> Descrever formalmente o problema .....	15
1.4.1.4	<input type="checkbox"/> Analisar por que o problema precisa ser resolvido.....	15
1.4.1.5	<input type="checkbox"/> Analisar quais soluções já existem.....	15
1.4.1.6	<input type="checkbox"/> Analisar como a solução será utilizada .....	16
1.4.1.7	<input type="checkbox"/> Descrever informalmente o problema.....	16
1.4.1.8	<input type="checkbox"/> Analisar problemas similares.....	16
1.5	1.2 ANALISAR OS DADOS.....	17
1.5.1	Elementos do processo .....	17
1.5.1.1	<input type="checkbox"/> Analisar quais as fontes de dados disponíveis.....	17
1.5.1.2	<input type="checkbox"/> Registrar fontes de dados.....	17
1.5.1.3	<input type="checkbox"/> Descrever os dados .....	17
1.5.1.4	<input type="checkbox"/> Visualizar os dados.....	18
1.6	1.3 TRATAR OS DADOS .....	19
1.6.1	Elementos do processo .....	19
1.6.1.1	<input type="checkbox"/> Selecionar os dados.....	19
1.6.1.2	<input type="checkbox"/> Limpar os dados .....	19
1.6.1.3	<input type="checkbox"/> Transformar os dados .....	20
1.7	1.6 FINALIZAR A SOLUÇÃO .....	21
1.7.1	Elementos do processo .....	21
1.7.1.1	<input type="checkbox"/> Operacionalizar o modelo .....	21
1.7.1.2	<input type="checkbox"/> Apresentar os resultados.....	22

# 1 1. SOLUCIONAR PROBLEMA COM MACHINE LEARNING



**Versão:** 1.0

**Autor:** abevieiramota@gmail.com

**Descrição**

Processo de solução de problemas com machine learning.

## 1.1 MAIN PROCESS

---

### 1.1.1 ELEMENTOS DO PROCESSO

#### 1.1.1.1 1.4 Testar e avaliar os modelos

[Ver detalhes](#)

**Descrição**

Aplicar um conjunto de algoritmos de machine learning e definir um baseline para a qualidade dos modelos.

#### 1.1.1.2 Analisar se os resultados são satisfatórios

#### 1.1.1.3 1.5 Melhorar os resultados

[Ver detalhes](#)

**Descrição**

Melhorar a qualidade dos resultados.

**Pedido ad-hoc**

Paralelo

#### 1.1.1.4 1.1 Analisar o problema

[Ver detalhes](#)

**Descrição**

Entender e definir o problema a ser solucionado.

You can use the most powerful and shiniest algorithms available, but the results will not be helpful to you if you are solving the wrong problem.

**Pedido ad-hoc**

Paralelo

1.1.1.5



1.2 Analisar os dados

[Ver detalhes](#)

### Descrição

Levantar que dados estão disponíveis e entendê-los.

### Pedido ad-hoc

Paralelo

1.1.1.6



1.3 Tratar os dados

[Ver detalhes](#)

### Descrição

Transformar os dados disponíveis em estruturas apropriadas para a aplicação de algoritmos de machine learning.

### Pedido ad-hoc

Paralelo

1.1.1.7



1.6 Finalizar a solução

[Ver detalhes](#)

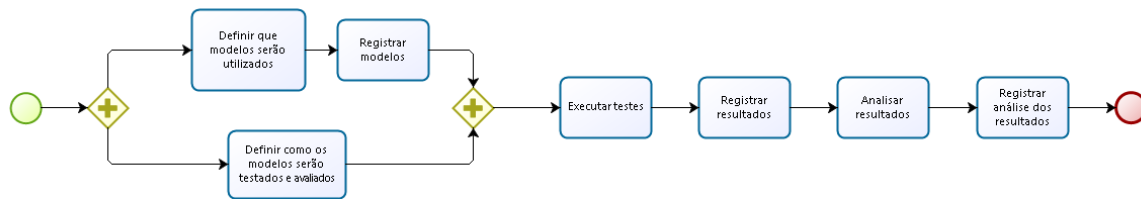
### Descrição

Descrever o problema e a solução desenvolvida para as partes interessadas.

### Pedido ad-hoc

Paralelo

## 1.2 1.4 TESTAR E AVALIAR OS MODELOS



Powered by  
bizaqi  
Modeler

### 1.2.1 ELEMENTOS DO PROCESSO

#### 1.2.1.1 Definir como os modelos serão testados e avaliados

##### Descrição

You need to define a test harness. The test harness is the data you will train and test an algorithm against and the measure(s) you will use to assess its performance. It is important to define your test harness well so that you can focus on evaluating different algorithms and thinking deeply about the problem.

The goal of the test harness is to be able to quickly and consistently test algorithms against a fair representation of the problem being solved. The outcome of testing multiple algorithms against the harness will be an estimation of how a variety of algorithms perform on the problem against a chosen measures. You will know which algorithms might be worth tuning on the problem and which should not be considered further.

The results will also give you an indication of how learnable the problem is. If a variety of different learning algorithms universally perform poorly on the problem, it may be an indication of a lack of structure available to algorithms to learn. This may be because there is a lack of learnable structure in the selected data or it may be an opportunity to try different transforms to expose the structure to the learning algorithms. Test Options

The test options you use when evaluating machine learning algorithms can mean the difference between over-learning, a mediocre result and a usable state-of-the-art result that you can confidently shout from the roof tops (you really do feel like doing that sometimes). Randomness The root of the difficulty in choosing the right test options is randomness. Most (almost all) machine learning algorithms use randomness in some way. The randomness may be explicit in the algorithm or may be in the sample of the data selected to train the algorithm.

This does not mean that the algorithms produce random results, it means that they produce results with some noise or variance. We call this type of limited variance, stochastic and the algorithms that exploit it, stochastic algorithms. Train and Test on Same Data If you have a dataset, you may want to train the model on the dataset and then report the results of the model tested on that dataset. That's how good the model is, right?

The problem with this approach of evaluating algorithms is that you indeed will know the performance of the algorithm on the dataset, but do not have any indication of how the algorithm will perform on data that the model was not trained on (so-called unseen data).

This matters, only if you want to use the model to make predictions on unseen data. Split Test

A simple way to use one dataset to both train and estimate the performance of the algorithm on unseen data is to split the dataset. You take the dataset, and split it into a training dataset and a test dataset. For example, you randomly select 66% of the instances for training and use the remaining 34% as a test dataset. Split tests are fast when you have a lot of data or when training a model is expensive (it resources or time). A split test on a large dataset can produce an accurate estimate of the actual performance of the algorithm.



If you run an algorithm on the training dataset and then assess the model on the test dataset, you will be able to compute a performance accuracy, for example 87%. How good is the algorithm on the data? Can we confidently say it can achieve an accuracy of 87%?

A problem is that if we split the training dataset again into a different 66%/34% split, we would get a different result from our algorithm. This is called model variance. Multiple Split Tests

A solution to our problem with the split test getting different results on different splits of the dataset is to reduce the variance of the random process and repeat it many times. We can collect the results from a fair number of runs (say 10) and take the average.

For example, let's say we split our dataset 66%/34%, ran our algorithm and got an accuracy and we did this 10 times with 10 different splits. We might have 10 accuracy scores as follows: 87, 87, 88, 89, 88, 86, 88, 87, 88, 87. The average performance of our model is 87.5, with a standard deviation of about 0.85.

A problem with multiple split tests is that it is possible that some data instances are never included for training or testing, whereas others may be selected multiple times. The effect is that this may skew results and may not give an meaningful idea of the accuracy of the algorithm.

Cross Validation A solution to the problem of ensuring each instance is used for training and testing an equal number of times while reducing the variance of an accuracy score is to use cross validation. Specifically k-fold cross validation, where k is the number of splits to make in the dataset.

For example, let's choose a value of k=10 (which is very common). This will split the dataset into 10 parts (10 folds) and the algorithm will be run 10 times. Each time the algorithm is run, it will be trained on 90% of the data and tested on 10%, and each run of the algorithm will change which 10% of the data the algorithm is tested on.

In this example, each data instance will be used as a training instance exactly 9 times and as a test instance 1 time. The accuracy will not be a mean and a standard deviation, but instead will be an exact accuracy score of how many correct predictions were made.

The k-fold cross validation method is the go-to method for evaluating the performance of an algorithm on a dataset. You want to choose k-values that give you a good sized training and test dataset for your algorithm. Not too disproportionate (too large or small for training or test datasets). If you have a lot of data, you may have to resort to either sampling the data or reverting to a split test.

Cross validation does give an unbiased estimation of the algorithms performance on unseen data, but what if the algorithm itself uses randomness. The algorithm would produce different results for the same training data each time it was trained with a different random number seed (start of the sequence of pseudo-randomness). Cross validation does not account for variance in the algorithm's predictions.

Another point of concern is that cross validation itself uses randomness to decide how to split the dataset into k-folds. Cross validation does not estimate how the algorithm performs with different sets of folds. This only matters if you want to understand how robust the algorithm is on the dataset. Multiple Cross Validation

A way to account for the variance in the algorithm itself is to run cross validation multiple times and take the mean and the standard deviation of the algorithms accuracy from each run. This will give you an estimate of the performance of the algorithm on the dataset and an estimation of how robust (the size of the standard deviation) the performance is.

If you have one mean and standard deviation for algorithm A and another mean and standard deviation for algorithm B and they differ (for example, algorithm A has a higher accuracy), how do you know if the difference is meaningful?






This only matters if you want to compare the results between algorithms. Statistical Significance

A solution to comparing algorithm performance measures when using multiple runs of k-fold cross validation is to use statistical significance tests (like the Student's t-test).

The results from multiple runs of k-fold cross validation is a list of numbers. We like to summarize these numbers using the mean and standard deviation. You can think of these numbers as a sample from an underlying population. A statistical significance test answers the question: are two samples drawn from the same population? (no difference). If the answer is "yes", then, even if the mean and standard deviations differ, the difference can be said to be not statistically significant.

We can use statistical significance tests to give meaning to the differences (or lack there of) between algorithm results when using multiple runs (like multiple runs of k-fold cross validation with different random number seeds). This can be useful when we want to make accurate claims about results (algorithm A was better than algorithm B and the difference was statistically significant)

This is not the end of the story, because there are different statistical significance tests (parametric and nonparametric) and parameters to those tests (p-value). I'm going to draw the line here because if you have followed me this far, you now know enough about selecting test options to produce rigorous (publishable!) results.

- 1.2.1.2  Executar testes
- 1.2.1.3  Registrar resultados
- 1.2.1.4  Analisar resultados
- 1.2.1.5  Registrar análise dos resultados
- 1.2.1.6  Definir que modelos serão utilizados

## Descrição

Spotchecking algorithms is about getting a quick assessment of a bunch of different algorithms on your machine learning problem so that you know what algorithms to focus on and what to discard. Spot-Checking

On a new problem, you need to quickly determine which type or class of algorithms is good at picking out the structure in your problem and which are not. The solution is to try lots of algorithms and let the results guide your decisions on what algorithms or classes of algorithms to spend time tuning. The alternative to spot checking is that you feel overwhelmed by the vast number of algorithms and algorithm types that you could try, that you end up trying very few or going with what has worked for you in the past. This results in wasted time and subpar results. Benefits of Spot-Checking Algorithms

There are 3 key benefits of spotchecking algorithms on your machine learning problems: •

Speed: You could spend a lot of time playing around with different algorithms, tuning parameters and thinking about what algorithms will do well on your problem. I have been there and end up testing the same algorithms over and over because I have not been systematic. A single spotcheck experiment can save hours, days and even weeks of noodling around. •

Objective: There is a tendency to go with what has worked for you before. We pick our favorite algorithm (or algorithms) and apply them to every problem we see. The power of machine learning is that there are so many different ways to approach a given problem. A spotcheck experiment allows you to automatically and objectively discover those algorithms that are the best at picking out the structure in the problem so you can focus your attention. •

Results: Spotchecking algorithms gets you usable results, fast. You may discover a good enough solution in the first spot check experiment. Alternatively, you may quickly learn that your dataset does not expose enough structure for any mainstream algorithm to do well. Spotchecking gives you the results you need to decide whether to move forward and optimize a given model or backward and revisit the presentation of the problem. Tips for Spot-Checking Algorithms

There are some things you can do when you are spotchecking algorithms to ensure you are getting useful and actionable results. Below are 5 tips to ensure you are getting the most from spotchecking machine learning algorithms on your problem. •

Algorithm Diversity: You want a good mix of algorithm types. I like to include instance based methods (like LVQ and knn), functions and kernels (like neural nets, regression and SVM), rule systems (like Decision Table and RIPPER) and decision trees (like CART, ID3 and C4.5). •

Best Foot Forward: Each algorithm needs to be given a chance to put it's best foot forward. This does not mean performing a sensitivity analysis on the parameters of each algorithm, but using experiments and heuristics to give each algorithm a fair chance. For example if kNN is in the mix, give it 3 chances with k values of 1, 5 and 7. •

Formal Experiment: Don't play. There is a huge temptation to try lots of different things in an informal manner, to play around with algorithms on your problem. The idea of spotchecking is to get to the methods that do well on the problem, fast. Design the

experiment, run it, then analyze the results. Be methodical. I like to rank algorithms by their statistical significant wins (in pairwise comparisons) and take the top 3 to 5 as a basis for tuning. •

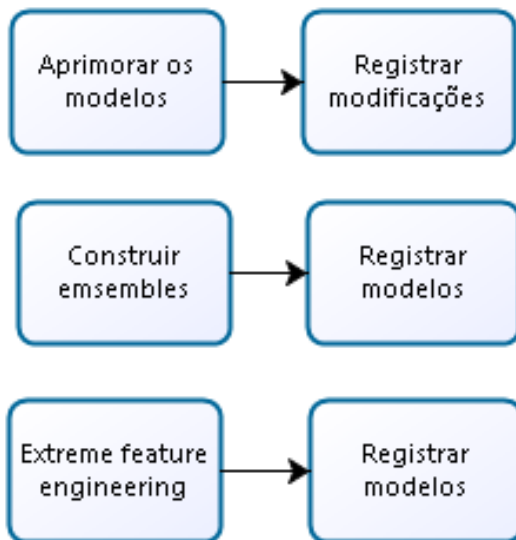
Jumping off Point: The best performing algorithms are a starting point not the solution to the problem. The algorithms that are shown to be effective may not be the best algorithms for the job. They are most likely to be useful pointers to types of algorithms that perform well on the problem. For example, if kNN does well, consider followup experiments on all the instance based methods and variations of kNN you can think of. •

Build Your Shortlist: As you learn and try many different algorithms you can add new algorithms to the suite of algorithms that you use in a spotcheck experiment. When I discover a particularly powerful configuration of an algorithm, I like to generalize it and include it in my suite, making my suite more robust for the next problem.

Start building up your suite of algorithms for spot check experiments.

#### 1.2.1.7 Registrar modelos

## 1.3 1.5 MELHORAR OS RESULTADOS



### 1.3.1 ELEMENTOS DO PROCESSO

### 1.3.1.1 Aprimorar os modelos

#### Descrição

The place to start is to get better results from algorithms that you already know perform well on your problem. You can do this by exploring and fine tuning the configuration for those algorithms. Machine learning algorithms are parameterized and modification of those parameters can influence the outcome of the learning process. Think of each algorithm parameter as a dimension on a graph with the values of a given parameter as a point along the axis. Three parameters would be a cube of possible configurations for the algorithm, and n-parameters would be an n-dimensional hypercube of possible configurations for the algorithm.

The objective of algorithm tuning is to find the best point or points in that hypercube for your problem. You will be optimizing against your test harness, so again you cannot underestimate the importance of spending the time to build a trusted test harness.

You can approach this search problem by using automated methods that impose a grid on the possibility space and sample where good algorithm configurations might be. You can then use those points in an optimization algorithm to zoom in on the best algorithm performance.

You can repeat this process with a number of well performing methods and explore the best you can achieve with each. I strongly advise that the process is automated and reasonably coarse grained as you can quickly reach points of diminishing returns (fractional percentage performance increases) that may not translate to the production system.

The more tuned the parameters of an algorithm, the more biased the algorithm will be to the training data and test harness. This strategy can be effective, but it can also lead to more fragile models that overfit your test harness and don't perform as well in practice.

### 1.3.1.2 Registrar modificações

### 1.3.1.3 Registrar modelos

### 1.3.1.4 Registrar modelos

### 1.3.1.5 Construir emsembles

#### Descrição

Ensemble methods are concerned with combining the results of multiple methods in order to get improved results. Ensemble methods work well when you have multiple "good enough" models that specialize in different parts of the problem.

This may be achieved through many ways. Three ensemble strategies you can explore are: •

Bagging: Known more formally as Bootstrapped Aggregation is where the same algorithm has different perspectives on the problem by being trained on different subsets of the training data. •

Boosting: Modules are linked in a chain, where one model learns to fix the mistakes made by the model ahead of it, and so on down the line. •

Blending: Known more formally as Stacked Aggregation or Stacking is where a variety of models whose predictions are taken as input to a new model that learns how to combine the predictions into an overall prediction.

It is a good idea to get into ensemble methods after you have exhausted more traditional methods. There are two good reasons for this, they are generally more complex than traditional methods and the traditional methods give you a good base level from which you can improve and draw from to create your ensembles.

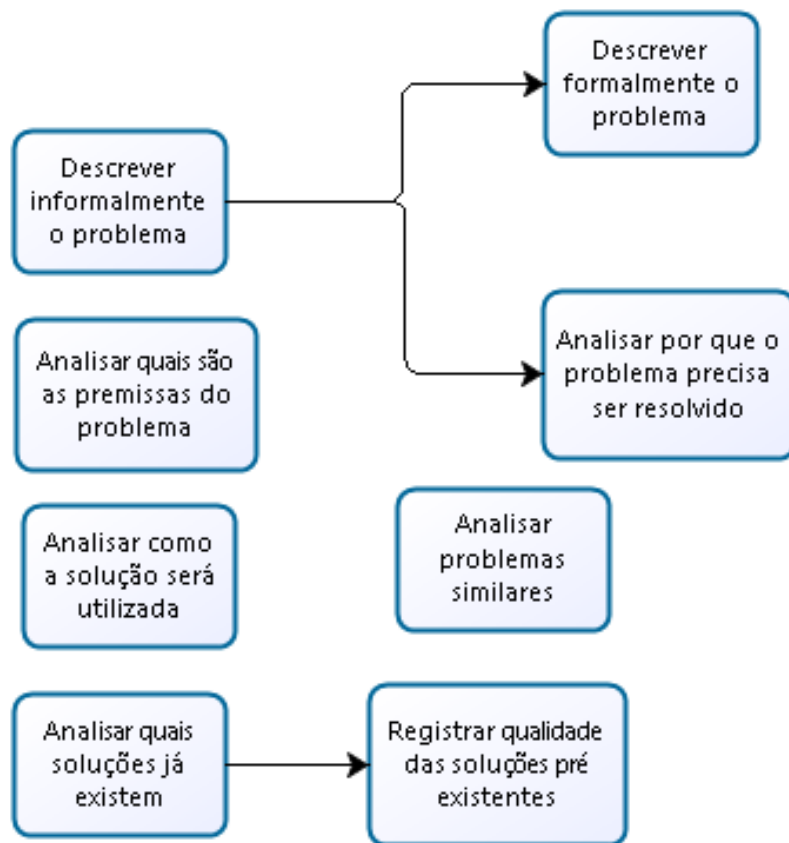
**Descrição**

The previous two strategies have looked at getting more from machine learning algorithms. This strategy is about exposing more structure in the problem for the algorithms to learn. In Data Preparation we learned about feature decomposition and aggregation in order to better normalise the data for machine learning algorithms. In this strategy we push that idea to the limits. I call this strategy extreme feature engineering, when really the term "feature engineering" would suffice. Think of your data as having complex multidimensional structures embedded in it that machine learning algorithms know how to find and exploit to make decisions. You want to best expose those structures to algorithms so that the algorithms can do their best work. A difficulty is that some of those structures may be too dense or too complex for the algorithms to find without help. You may also have some knowledge of such structures from your domain expertise. Take attributes and decompose them wildly into multiple features. Technically, what you are doing with this strategy is reducing dependencies and nonlinear relationships into simpler independent linear relationships. This might be a foreign idea, so here are two examples: •

- **Categorical:** You have a categorical attribute that had the values [red, green blue], you could split that into 3 binary attributes of red, green and blue and give each instance a 1 or 0 value for each.
- **Real:** You have a real valued quantity that has values ranging from 0 to 1000. You could create 10 binary attributes, each representing a bin of values (099 for bin 1, 100199 for bin 2, etc.) and assign each instance a binary values (1/0) for the bins.

I recommend performing this process one step at a time and creating a new test/train dataset for each modification you make and then test algorithms on the dataset. This will start to give you an intuition for attributes and features in the database that are exposing more or less information to the algorithms and the effects on performance measures. You can use these results to guide further extreme decompositions or aggregations.

## 1.4 1.1 ANALISAR O PROBLEMA



### 1.4.1 ELEMENTOS DO PROCESSO

#### 1.4.1.1 Analisar quais são as premissas do problema

##### Descrição

Create a list of assumptions about the problem and its phrasing. These may be rules of thumb and domain specific information that you think will get you to a viable solution faster.

It can be useful to highlight questions that can be tested against real data because breakthroughs and innovation occur when assumptions and best practices are demonstrated to be wrong in the face of real data. It can also be useful to highlight areas of the problem specification that may need to be challenged, relaxed or tightened. For example: •

The specific words used in the tweet matter to the model. • The specific user that retweets does not matter to the model. • The number of retweets may matter to the model. • Older tweets are less predictive than more recent tweets.

#### 1.4.1.2 Registrar qualidade das soluções pré existentes

##### **Descrição**

baseline

#### 1.4.1.3 Descrever formalmente o problema

##### **Descrição**

Tom Mitchell defines a useful formalism for describing a machine learning problem:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Use this formalism to define the T, P, and E for your problem.

For example: • Task (T): Classify a tweet that has not been published as going to get retweets or not. •

Experience (E): A corpus of tweets for an account where some have retweets and some do not. •

Performance (P): Classification accuracy, the number of tweets predicted correctly out of all tweets considered, as a percentage.

#### 1.4.1.4 Analisar por que o problema precisa ser resolvido

##### **Descrição**

The next step is to think deeply about why you want or need the problem solved. Motivation

Consider your motivation for solving the problem. What need will be fulfilled when the problem is solved?

For example, you may be solving the problem as a learning exercise. This is useful to clarify as you can decide that you don't want to use the most suitable method to solve the problem, but instead you want to explore methods that you are not familiar with in order to learn new skills.

Alternatively, you may need to solve the problem as part of a duty at work. Benefits

Consider the benefits of having the problem solved. What capabilities does it enable?

It is important to be clear on the benefits of the problem being solved to ensure that you capitalise on them. These benefits can be used to sell the project to colleagues and management to get buy-in and additional time or budget resources.

If it benefits you personally, then be clear on what those benefits are and how you will know when you have got them. For example, if it's a tool or utility, then what will you be able to do with that utility that you can't do now and why is that meaningful to you? Use

Consider how the solution to the problem will be used and what type of lifetime you expect the solution to have. As programmers we often think the work is done as soon as the program is written, but really the project is just beginning its maintenance lifetime.

The way the solution will be used will influence the nature and requirements of the solution you adopt.

This is covered in the Present Results step, but for now consider whether you are looking to write a report to present results or you want to operationalize the solution. If you want to operationalize the solution, consider the functional and nonfunctional requirements you have for a solution, just like a software project.

#### 1.4.1.5 Analisar quais soluções já existem

##### **Descrição**

baseline

#### 1.4.1.6 Analisar como a solução será utilizada

##### **Descrição**

embedded q+

#### 1.4.1.7 Descrever informalmente o problema

##### **Descrição**

1. What is the problem? 2. Why does the problem need to be solve? 3. How would I solve the problem?

Describe the problem as though you were describing it to a friend or colleague. This can provide a great starting point for highlighting areas that you might need to expand upon. It also provides the basis for a one sentence description you can use to share your understanding of the problem.

#### 1.4.1.8 Analisar problemas similares

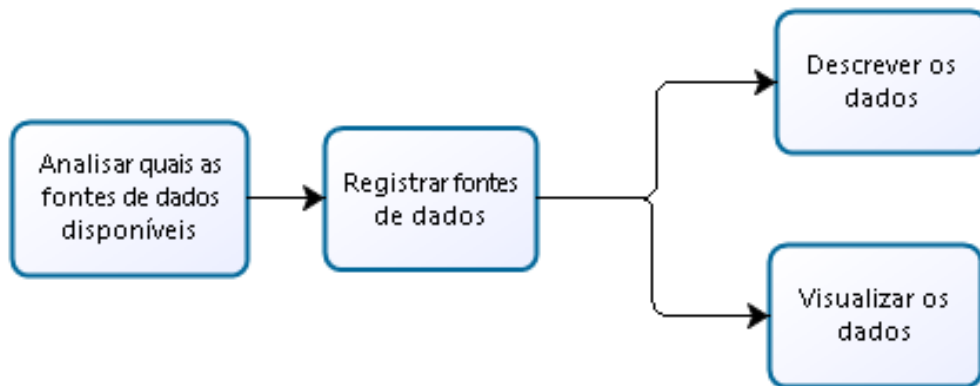
##### **Descrição**

What other problems have you seen or can you think of that are like the problem you are trying to solve? Other problems can inform the problem you are trying to solve by highlighting limitations in your phrasing of the problem such as time dimensions and conceptual drift (where the concept being modelled changes over time). Other problems can also point to algorithms and data transformations that could be adopted to spot check performance. For example: •

A related problem would be email spam discrimination that uses email body data as input data and needs binary classification decision.



## 1.5 1.2 ANALISAR OS DADOS



Powered by  
**bizagi**  
Modeler

---

### 1.5.1 ELEMENTOS DO PROCESSO

#### 1.5.1.1 Analisar quais as fontes de dados disponíveis

##### **Descrição**

tipo(web, documento físico, questionário etc) e atributos específicos(web.url, como acessar etc)

#### 1.5.1.2 Registrar fontes de dados

##### **Descrição**

data.sources

#### 1.5.1.3 Descrever os dados

##### **Descrição**

Summarizing the data is about describing the actual structure of the data. I typically use a lot of automated tools to describe things like attribute distributions. The minimum aspects of the data I like to summarize are the structure and the distributions. Data Structure Summarizing the data structure is about describing the number and types of attributes. For example, going through this process highlights ideas for transforms in the Data Preparation step for converting attributes from one type to another (such as real to ordinal or nominal to binary).

Some motivating questions for this step include: • How many attributes and instances are there? • What are the data types of each attribute (e.g. nominal, ordinal, integer, real, etc.)? Data Distributions Summarizing the distributions of each attributes can also flush out ideas for possible data transforms in the Data Preparation step, such as the need and effects of Discretize, Normalization and Standardization. I like to capture a summary of the distribution of each real valued attribute. This typically includes the minimum, maximum, median, mode, mean, standard deviation and number of missing values. Some motivating questions for this step include: • Create a five-number summary of each realvalued attribute. • What is the distribution of the values for the class attribute? Knowing the distribution of the class attribute (or mean of a regression output attribute) is useful because you can use it to define the minimum accuracy of a predictive model. For example, if there is a binary classification problem (2 classes) with the distribution of 80% apples and 20% bananas, then a predictor can predict "apples" for every test instance and be assured to achieve an accuracy of 80%. This is the worst case algorithm that all algorithms in the test harness must beat when Evaluating Algorithms. Additionally, if I have the time or interest, I like to generate a summary of the pairwise attribute correlations using a parametric (Pearson's) and nonparametric (Spearman's) correlation coefficient. This can highlight attributes that might be candidates for removal (highly correlated with each other) and others that may be highly predictive (highly correlated with the class attribute).

data.structures

#### 1.5.1.4



#### Visualizar os dados

### Descrição

Visualizing the data is about creating graphs that summarize the data, capturing them and studying them for interesting structure that can be described.

There are seemingly an infinite number of graphs you can create (especially in software like R), so I like to keep it simple and focus on histograms and scatter plots. Attribute Histograms

I like to create histograms of all attributes and mark class values. I like this because I used Weka a lot when I was learning machine learning and it does this for you. Nevertheless, it's easy to do in other software like R. Having a discrete distribution graphically can quickly highlight things like the possible family of distribution (such as Normal or Exponential) and how the class values map onto those distributions.

Some motivating questions for this step include: • What families of distributions are shown (if any)? •

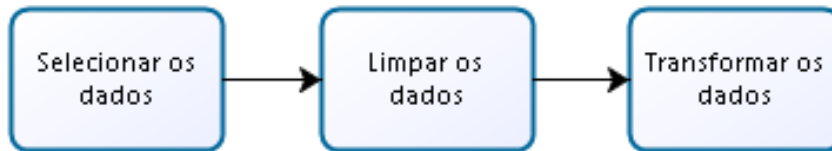
Are there any obvious structures in the attributes that map to class values? Pairwise Scatter-plots Scatter plots show one attribute on each axis. In addition, a third axis can be added in the form of the color of the plotted points mapping to class values. Pairwise scatter plots can be created for all pairs of attributes. These graphs can quickly highlight 2dimensional structure between attributes (such as correlation) as well as crossattribute trends in the mapping of attribute to class values.

Some motivating questions for this step include: • What interesting twodimensional structures are shown? •

What interesting relationships between the attributes to class values are shown?

data.visualization

## 1.6 1.3 TRATAR OS DADOS



Powered by  
**bizagi**  
Modeler

### 1.6.1 ELEMENTOS DO PROCESSO

#### 1.6.1.1 Selecionar os dados

##### Descrição

que fontes/dados/atributos serão utilizados

This step is concerned with selecting the subset of all available data that you will be working with. There is always a strong desire for including all data that is available, that the maxim “more is better” will hold. This may or may not be true.

You need to consider what data you actually need to address the question or problem you are working on. Make some assumptions about the data you require and be careful to record those assumptions so that you can test them later if needed. Below are some questions to help you think through this process:

- What is the extent of the data you have available? For example through time, database tables, connected systems. Ensure you have a clear picture of everything that you can use. •

What data is not available that you wish you had available? For example data that is not recorded or cannot be recorded. You may be able to derive or simulate this data. •

What data don't you need to address the problem? Excluding data is almost always easier than including data. Note down which data you excluded and why.

It is only in small problems, like competition or toy datasets where the data has already been selected for you.

#### 1.6.1.2 Limpar os dados

##### Descrição

remover dados corrompidos

After you have selected the data, you need to consider how you are going to use the data. This preprocessing step is about getting the selected data into a form that you can work with.

Three common data preprocessing steps are formatting, cleaning and sampling: •

Formatting: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file. •

Cleaning: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be

sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely. •

Sampling: There may be far more selected data available than you need to work with.

More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

It is very likely that the machine learning tools you use on the data will influence the preprocessing you will be required to perform. You will likely revisit this step.

### 1.6.1.3 Transformar os dados

#### Descrição

The final step is to transform the process data. The specific algorithm you are working with and the knowledge of the problem domain will influence this step and you will very likely have to revisit different transformations of your preprocessed data as you work on your problem.

Three common data transformations are scaling, attribute decompositions and attribute aggregations. This step is also referred to as feature engineering. •

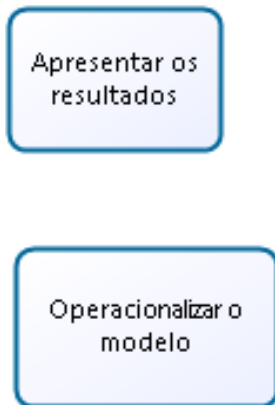
Scaling: The preprocessed data may contain attributes with mixtures of scales for various quantities such as dollars, kilograms and sales volume. Many machine learning methods like data attributes to have the same scale such as between 0 and 1 for the smallest and largest value for a given feature. Consider any feature scaling you may need to perform. •

Decomposition: There may be features that represent a complex concept that may be more useful to a machine learning method when split into the constituent parts. An example is a date that may have day and time components that in turn could be split out further. Perhaps only the hour of day is relevant to the problem being solved. Consider what feature decompositions you can perform.

Aggregation: There may be features that can be aggregated into a single feature that would be more meaningful to the problem you are trying to solve. For example, there may be data instances for each time a customer logged into a system that could be aggregated into a count for the number of logins allowing the additional instances to be discarded. Consider what type of feature aggregations you could perform.

You can spend a lot of time engineering features from your data and it can be beneficial to the performance of an algorithm. Start small and build on the skills you learn.

## 1.7 1.6 FINALIZAR A SOLUÇÃO



Powered by  
**bizagi**  
Modeler

---

### 1.7.1 ELEMENTOS DO PROCESSO

#### 1.7.1.1 Operacionalizar o modelo

##### **Descrição**

You have found a model that is good enough at addressing the problem you face that you would like to put it into production. This may be an operational installation on your workstation in the case of a fun side project, all the way up to integrating the model into an existing enterprise application. The scope is enormous. In this section you will learn three key aspects to operationalizing a model that you could consider carefully before putting a system into production. Three areas that you should think carefully about are the algorithm implementation, the automated testing of your model and the tracking of the models performance through time. These three issues will very likely influence the type of model you choose. Algorithm Implementation It is likely that you were using a research library to discover the best performing method. The algorithm implementations in research libraries can be excellent, but they can also be written for the general case of the problem rather than the specific case with which you are working. Think very hard about the dependencies and technical debt you may be creating by putting such an implementation directly into production. Consider locating a productionlevel library that supports the method you wish to use. You may have to repeat the process of algorithm tuning if you switch to a production level library at this point. You may also consider implementing the algorithm yourself. This option may introduce risk depending on the complexity of the algorithm you have chosen and the implementation tricks it uses. Even with open source code, there may be a number of complex operations that may be very difficult to internalize and reproduce confidently. Model Tests Write automated tests that verify that the model can be constructed and achieve a minimum level of performance repeatedly. Also write tests for any data preparation steps. You may wish to control the randomness used by the algorithm (random number seeds) for each unit test run so that tests are 100% reproducible.

Tracking Add infrastructure to monitor the performance of the model over time and raise alarms if accuracy drops below a minimum level. Tracking may occur in realtime or with samples of live data on a recreated model in a separate environment. A raised alarm may be an indication that that structure learned by the model in the data have changed (concept drift) and that the model may need to be updated or tuned.

There are some model types that can perform online learning and update themselves. Think carefully in allowing models to update themselves in a production environment. In some circumstances, it can be wiser to manage the model update process and switch out models (their internal configuration) as they are verified to be more performant.

### 1.7.1.2



### Apresentar os resultados

#### Descrição

Once you have discovered a good model and a good enough result (or not, as the case may be), you will want to summarize what was learned and present it to stakeholders. This may be yourself, a client or the company for which you work.

Use a powerpoint template and address the sections listed below. You might like to write up a one-pager and use each part as a section header. Try to follow this process even on small experimental projects you do for yourself such as tutorials and competitions. It is easy to spend an inordinate number of hours on a project and you want to make sure to capture all the great things you learn along the way. Below are the sections you can complete when reporting the results for a project. •

Context (Why): Define the environment in which the problem exists and set up the motivation for the research question. •

Problem (Question): Concisely describe the problem as a question that you went out and answered. •

Solution (Answer): Concisely describe the solution as an answer to the question you posed in the previous section. Be specific. • Findings: Bulleted lists of discoveries you made along the way that interest the audience. They may be discoveries in the data, methods that did or did not work or the model performance benefits you achieved along your journey. •

Limitations: Consider where the model does not work or questions that the model does not answer. Do not shy away from these questions, defining where the model excels is more trusted if you can define where it does not excel.

Conclusions (Why+Question+Answer): Revisit the why, research question and the answer you discovered in a tight little package that is easy to remember and repeat for yourself and others.

The type of audience you are presenting to will define the amount of detail you go into. Having the discipline to complete your projects with a report on results, even on small side projects will accelerate your learning in the field. I highly recommend sharing results of side projects on blogs or with communities and elicit feedback that you can capture and take with you into the start of your next project.