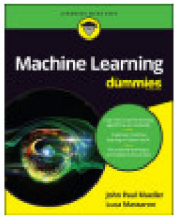
dummies
A Wiley Brand

Cheat Sheet

Machine Learning For Dummies Cheat Sheet



From **Machine Learning For Dummies**

By **John Paul Mueller, Luca Massaron**

Machine learning is an incredible technology that you use more often than you think today and with the potential to do even more tomorrow. The interesting thing about machine learning is that both R and Python make the task easier than more people realize because both languages come with a lot of built-in and extended support (through the use of libraries, datasets, and other resources). With that in mind, this cheat sheet helps you access the most commonly needed reminders for making your machine learning experience fast and easy.

Choosing the Right Algorithm for Machine Learning

Machine learning involves the use of many different algorithms. This table gives you a quick summary of the strengths and weaknesses of various algorithms.

Algorithm	Best at	Pros	Cons
Random Forest	Apt at almost any machine learning problem Bioinformatics	Can work in parallel Seldom overfits Automatically handles missing values No need to transform any variable No need to tweak parameters Can be used by almost anyone with excellent results	Difficult to interpret Weaker on regression when estimating values at the extremities of the distribution of response values Biased in multiclass problems toward more frequent classes

Gradient Boosting	Apt at almost any machine learning problem Search engines (solving the problem of learning to rank)	It can approximate most nonlinear function Best in class predictor Automatically handles missing values No need to transform any variable	It can overfit if run for too many iterations Sensitive to noisy data and outliers Doesn't work well without parameter tuning
Linear regression	Baseline predictions Econometric predictions Modelling marketing responses	Simple to understand and explain It seldom overfits Using L1 & L2 regularization is effective in feature selection Fast to train Easy to train on big data thanks to its stochastic version	You have to work hard to make it fit nonlinear functions Can suffer from outliers
Support Vector Machines	Character recognition Image recognition Text classification	Automatic nonlinear feature creation Can approximate complex nonlinear functions	Difficult to interpret when applying nonlinear kernels Suffers from too many examples, after 10,000 examples it starts taking too long to train

K-nearest Neighbors	Computer vision Multilabel tagging Recommender systems Spell checking problems	Fast, lazy training Can naturally handle extreme multiclass problems (like tagging text)	Slow and cumbersome in the predicting phase Can fail to predict correctly due to the curse of dimensionality
Adaboost	Face detection	Automatically handles missing values No need to transform any variable It doesn't overfit easily Few parameters to tweak It can leverage many different weak-learners	Sensitive to noisy data and outliers Never the best in class predictions
Naive Bayes	Face recognition Sentiment analysis Spam detection Text classification	Easy and fast to implement, doesn't require too much memory and can be used for online learning Easy to understand Takes into account prior knowledge	Strong and unrealistic feature independence assumptions Fails estimating rare occurrences Suffers from irrelevant features

Neural Networks	Image recognition Language recognition and translation Speech recognition Vision recognition	Can approximate any nonlinear function Robust to outliers Works only with a portion of the examples (the support vectors)	Very difficult to set up Difficult to tune because of too many parameters and you have also to decide the architecture of the network Difficult to interpret Easy to overfit
Logistic regression	Ordering results by probability Modelling marketing responses	Simple to understand and explain It seldom overfits Using L1 & L2 regularization is effective in feature selection The best algorithm for predicting probabilities of an event Fast to train Easy to train on big data thanks to its stochastic version	You have to work hard to make it fit nonlinear functions Can suffer from outliers
SVD	Recommender systems	Can restructure data in a meaningful way	Difficult to understand why data has been restructured in a certain way
PCA	Removing collinearity Reducing dimensions of the dataset	Can reduce data dimensionality	Implies strong linear assumptions (components are a weighted summations of features)

K-means	Segmentation	Fast in finding clusters Can detect outliers in multiple dimensions	Suffers from multicollinearity Clusters are spherical, can't detect groups of other shape Unstable solutions, depends on initialization

Getting the Right Library for Machine Learning

When working with R and Python for machine learning, you gain the benefit of not having to reinvent the wheel when it comes to algorithms. There is a library available to meet your specific needs — you just need to know which one to use. This table provides you with a listing of the libraries used for machine learning for both R and Python. When you want to perform any algorithm-related task, simply load the library needed for that task into your programming environment.

Algorithm	Python implementation	R implementation
Adaboost	sklearn.ensemble.AdaBoostClassifier sklearn.ensemble.AdaBoostRegressor	library(ada) : ada
Gradient Boosting	sklearn.ensemble.GradientBoostingClassifier sklearn.ensemble.GradientBoostingRegressor	library(gbm) : gbm
K-means	sklearn.cluster.KMeans sklearn.cluster.MiniBatchKMeans	library(stats) : kmeans
K-nearest Neighbors	sklearn.neighbors.KNeighborsClassifier sklearn.neighbors.KNeighborsRegressor	library(class): knn

Linear regression	sklearn.linear_model.LinearRegression	library(stats) : lm
	sklearn.linear_model.Ridge	library(stats) : glm
	sklearn.linear_model.Lasso	library(MASS) : lm.ridge
	sklearn.linear_model.ElasticNet	library(lars) : lars
	sklearn.linear_model.SGDRegressor	library(glmnet) : glmnet
Logistic regression	sklearn.linear_model.LogisticRegression	library(stats) : glm
	sklearn.linear_model.SGDClassifier	library(glmnet) : glmnet
Naive Bayes	sklearn.naive_bayes.GaussianNB	library(klaR) : NaiveBayes
	sklearn.naive_bayes.MultinomialNB	library(e1071) : naiveBayes
	sklearn.naive_bayes.BernoulliNB	
Neural Networks	sklearn.neural_network.BernoulliRBM	library(neuralnet) : neuralnet
	(in version 0.18 of Scikit-learn, a new implementation of supervised neural network will be introducted)	library(AMORE) : train library(nnet) : nnet
PCA	sklearn.decomposition.PCA	library(stats): princomp library(stats) : stats
Random Forest	sklearn.ensemble.RandomForestClassifier	library(randomForest) : randomForest
	sklearn.ensemble.RandomForestRegressor	
	sklearn.ensemble.ExtraTreesClassifier	
	sklearn.ensemble.ExtraTreesRegressor	

Support Vector Machines	sklearn.svm.SVC	library(e1071) : svm
	sklearn.svm.LinearSVC	
	sklearn.svm.NuSVC	
	sklearn.svm.SVR	
	sklearn.svm.LinearSVR	
	sklearn.svm.NuSVR	
	sklearn.svm.OneClassSVM	
SVD	sklearn.decomposition.TruncatedSVD	library(irlba) : irlba
	sklearn.decomposition.NMF	library(svd) : svd

Locating the Algorithm You Need for Machine Learning

There are a number of different algorithms you can use for machine learning. However, finding the specific algorithm you want to know about can be difficult. This table provides you with the online location for information about the algorithms used in machine learning.

Algorithm	Type	Python/R URL
Naive Bayes	Supervised classification, online learning	**http://scikit-learn.org/stable/modules/naive_bayes.html** **https://cran.r-project.org/web/packages/bnlearn/index.html**
PCA	Unsupervised	**http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PC** **https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html**
SVD	Unsupervised	**http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.** **https://cran.r-project.org/web/packages/svd/index.html**

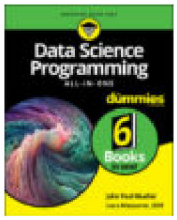
K-means	Unsupervised	**http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.ht** **https://cran.r-project.org/web/packages/broom/vignettes/kmeans.html**
K-nearest Neighbors	Supervised regression and classification	**http://scikit-learn.org/stable/modules/neighbors.html** **https://cran.r-project.org/web/packages/kknn/index.html**
Linear Regression	Supervised regression, online learning	**http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegressior** **https://cran.r-project.org/web/packages/phylolm/index.html**
Logistic Regression	Supervised classification, online learning	**http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressic** **https://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_logistic_regression_glm.pdf**
Neural Networks	Unsupervised Supervised regression and classification	**http://scikit-learn.org/dev/modules/neural_networks_supervised.html** **https://cran.r-project.org/web/packages/neuralnet/index.html**
Support Vector Machines	Supervised regression and classification	**http://scikit-learn.org/stable/modules/svm.html** **https://cran.r-project.org/web/packages/e1071/index.html**
Adaboost	Supervised classification	**http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.h** **https://cran.r-project.org/web/packages/adabag/index.html**
Gradient Boosting	Supervised regression and classification	**http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingCl** **https://cran.r-project.org/web/packages/gbm/index.html**
Random Forest	Supervised regression and classification	**http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClass** **https://cran.r-project.org/web/packages/randomForest/index.html**

About the Book Author

John Paul Mueller is a prolific freelance author and technical editor. He's covered everything from networking and home security to database management and heads-down programming.

Luca Massaron is a data scientist who specializes in organizing and interpreting big data, turning it into smart data with data mining and machine learning techniques.

A Brief Guide to Understanding Bayes' Theorem



By **John Paul Mueller, Luca Massaron**

Before you begin using Bayes' Theorem to perform practical tasks, knowing a little about its history is helpful. The reason this knowledge is so useful is because Bayes' Theorem doesn't seem to be able to do everything it purports to do when you first see it, which is why many statisticians rejected it outright.

©Shutterstock/Gearstd

After you do have a basic knowledge of how Bayes' Theorem came into being, you need to look at the theorem itself. The following sections provide you with a history of Bayes' Theorem that then moves into the theorem itself. Here, Bayes' Theorem is presented from a practical perspective.

A little Bayes history

You might wonder why anyone would name an algorithm Naïve Bayes (yet you find this algorithm among the most effective **machine learning algorithms in packages** such as Scikit-learn). The *naïve* part comes from its formulation; it makes some extreme simplifications to standard probability calculations. The reference to Bayes in its name relates to the Reverend Bayes and his theorem on probability.

The Reverend Thomas Bayes (1701–1761) was an English statistician and a philosopher who formulated his theorem during the first half of the eighteenth century. Bayes' Theorem is based on a thought experiment and then a demonstration using the simplest of means.

Reverend Bayes wanted to determine the probability of a future event based on the number of times it occurred in the past. It's hard to contemplate how to accomplish this task with any accuracy.

The demonstration relied on the use of two balls. An assistant would drop the first ball on a table where the end position of this ball was equally possible in any location, but not tell Bayes its location. The assistant would then drop a second ball, tell Bayes the position of the second ball, and then provide the position of the first ball relative to the location of this second ball

The assistant would then drop the second ball a number of additional times — each time telling Bayes the location of the second ball and the position of the first ball relative to the second. After each toss of the second ball, Bayes would attempt to guess the position of the first. Eventually, he was to guess the position of the first ball based on the evidence provided by the second ball.

The theorem was never published while Bayes was alive. His friend Richard Price found Bayes' notes after his death in 1761 and published the material for Bayes, but no one seemed to read it at first. **Bayes' Theorem** has deeply revolutionized the theory of probability by introducing the idea of conditional probability — that is, probability conditioned by evidence.

The critics saw problems with Bayes' Theorem that you can summarize as follows:

- Guessing has no place in rigorous mathematics.

- If Bayes didn't know what to guess, he would simply assign all possible outcomes an equal probability of occurring.
- Using the prior calculations to make a new guess presented an insurmountable problem.

Often, it takes a problem to illuminate the need for a previously defined solution, which is what happened with Bayes' Theorem. By the late eighteenth century, the need to study astronomy and make sense of the observations made by the

- Chinese in 1100 BC
- Greeks in 200 BC
- Romans in AD 100
- Arabs in AD 1000

became essential. The readings made by these other civilizations not only reflected social and other biases but also were unreliable because of the differing methods of observation and the technology use.

You might wonder why the study of astronomy suddenly became essential, and the short answer is money. **Navigation of the late eighteenth century relied heavily on accurate celestial observations**, so anyone who could make the readings more accurate could reduce the time required to ship goods from one part of the world to another.

Pierre-Simon Laplace wanted to solve the problem, but he couldn't just dive into the astronomy data without first having a means to dig through all that data to find out which was correct and which wasn't. He encountered Richard Price, who told him about Bayes' Theorem.

Laplace used the theorem to solve an easier problem, that of the births of males and females. Some people had noticed that more boys than girls were born each year, but no proof existed for this observation. Laplace used Bayes' Theorem to prove that more boys are born each year than girls based on birth records.

Other statisticians took notice and started using the theorem, often secretly, for a host of other calculations, such as the calculation of the masses of Jupiter and Saturn from a wide variety of **observations by Alexis Bouvard**.

The basic Bayes theorem formula

When thinking about Bayes' Theorem, it helps to start from the beginning — that is, probability itself. *Probability* tells you the likelihood of an event and is expressed in a numeric form.

REMEMBER

The probability of an event is measured in the range from 0 to 1 (from 0 percent to 100 percent) and it's empirically derived from counting the number of times a specific event happens with respect to all the events. You can calculate it from data!

When you observe events (for example, when a feature has a certain characteristic) and you want to estimate the probability associated with the event, you count the number of times the characteristic appears in the data and divide that figure by the total number of observations available. The result is a number ranging from 0 to 1, which expresses the probability.

When you estimate the probability of an event, you tend to believe that you can apply the probability in each situation. The term for this belief is *a priori* because it constitutes the first estimate of probability with regard to an event (the one that comes to mind first).

For example, if you estimate the probability of an unknown person's being a female, you might say, after some counting, that it's 50 percent, which is the prior, or the first, probability that you will stick with.

The prior probability can change in the face of evidence, that is, something that can radically modify your expectations.

For example, the evidence of whether a person is male or female could be that the person's hair is long or short. You can estimate having long hair as an event with 35 percent probability for the general population, but within the female population, it's 60 percent. If the percentage is higher in the female population, contrary to the general probability (the prior for having long hair), that's useful information for making a prediction.

Imagine that you have to guess whether a person is male or female and the evidence is that the person has long hair. This sounds like a predictive problem, and in the end, this situation is similar to predicting a categorical variable from data: We have a target variable with different categories and you have to guess the probability of each category based on evidence, the data. Reverend Bayes provided a useful formula:

```
P(B|E) = P(E|B)*P(B) / P(E)
```

The formula looks like statistical jargon and is a bit counterintuitive, so it needs to be explained in depth. Reading the formula using the previous example as input makes the meaning behind the formula quite a bit clearer:

- **P(B|E):** The probability of being a female (the belief B) given long hair (the evidence E). This part of the formula defines what you want to predict. In short, it says to predict y given x where y is an outcome (male or female) and x is the evidence (long or short hair).
- **P(E|B):** The probability of having long hair, the evidence of when a person is female. In this case, you already know that it's 60 percent. In every data problem, you can obtain this figure easily by simple cross-tabulation of the features against the target outcome.
- **P(B):** The probability of being a female, which has a 50 percent general chance (a prior).
- **P(E):** The probability of having long hair in general, which is 35 percent (another prior).

TIP

When reading parts of the formula such as P(B|E), you should read them as follows: probability of B given E. The | symbol translates as given. A probability expressed in this way is a conditional probability, because it's the probability of a belief, B, conditioned by the evidence presented by E. In this example, plugging the numbers into the formula translates into

```
60% * 50% / 35% = 85.7%
```

Therefore, getting back to the previous example, even if being a female is a 50 percent probability, just knowing evidence like long hair takes it up to 85.7 percent, which is a more favorable chance for the guess. You can be more confident in guessing that the person with long hair is a female because you have a bit less than a 15 percent chance of being wrong.

About the Book Author

John Mueller has produced 114 books and more than 600 articles on topics ranging from functional programming techniques to working with Amazon Web Services (AWS). **Luca Massaron,** a Google Developer Expert (GDE),??interprets big data and transforms it into smart data through simple and effective data mining and machine learning techniques.