

Machine Learning Basics:

An Illustrated Guide for Non-Technical Readers



A GUIDEBOOK BY DATAIKU

www.dataiku.com

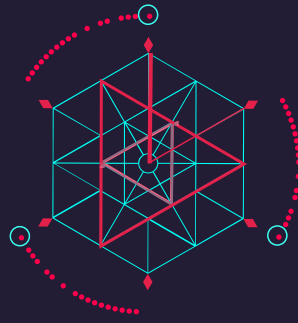
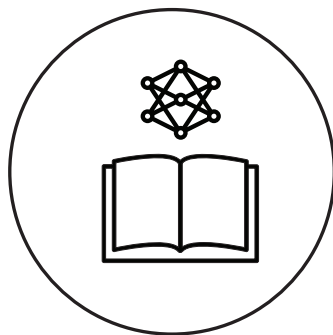


TABLE OF CONTENTS

3	MACHINE LEARNING CONCEPTS FOR EVERYONE
4	AN INTRODUCTION TO KEY DATA SCIENCE CONCEPTS
6	TOP PREDICTION ALGORITHMS
8	HOW TO EVALUATE MODELS
10	INTRODUCING THE K-FOLD STRATEGY AND THE HOLD-OUT STRATEGY
12	K-MEANS CLUSTERING ALGORITHM IN ACTION
14	FOR FURTHER EXPLORATION
15	ABOUT DATAIKU





Machine Learning Concepts for Everyone

According to Google Trends, interest in the term machine learning (ML) has increased over 300 percent since Dataiku was founded in 2013. We've watched ML go from the realm of a relatively small number of data scientists to the mainstream of analysis and business. And while this has resulted in a plethora of innovations and improvements among our customers and for organizations worldwide, it's also provoked reactions ranging from curiosity to anxiety among people everywhere.

We've decided to make this guide because we've noticed that there aren't too many resources out there that answer the question, "What is machine learning?" while using a minimum of technical terms. Actually, the basic concepts of machine learning aren't very difficult to grasp when they're explained simply.

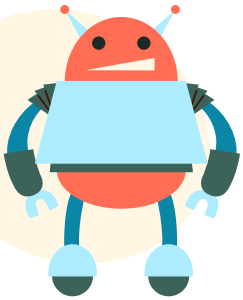
In this guidebook, we'll start with some definitions and then move on to explain some of the most common algorithms used in machine learning today, such as linear regression and tree-based models. Then we'll dive a bit deeper into how we go about deciding how to evaluate and fine-tune these models. Next, we'll take a look at clustering models, and then we'll finish with some resources that you can explore if you want to learn more.

We hope you enjoy the guidebook and that no matter how much or how little you have familiarized yourself with machine learning, you find it valuable!



An Introduction to Key Data Science Concepts

Definitions of 10 fundamental terms for data science and machine learning



What is machine learning?

The answer is, in one word, **algorithms**.

Machine learning is, more or less, a way for computers to learn things without being specifically programmed. But how does that actually happen?

Algorithms are sets of rules that a computer is able to follow. Think about how you learned to do long division -- maybe you learned to take the denominator and divide it into the first digits of the numerator, then subtracting the subtotal and continuing with the next digits until you were left with a remainder.

Well, that's an algorithm, and it's the sort of thing we can program into a computer, which can perform these sorts of calculations much, much faster than we can.

What does **machine learning** look like?

In machine learning, our goal is either prediction or clustering. Prediction is a process where, from a set of input variables, we estimate the value of an output variable. For example, using a set of characteristics of a house, we can predict its sale price. Prediction problems are divided into two main categories:

0 1 2 3 4
5 6 7 8 9 6 8 7 9 5
4 3 2 1 0

Regression problems, where the variable to predict is numerical (e.g., the price of a house).



Classification problems, where the variable to predict is part of one of some number of pre-defined categories, which can be as simple as "yes" or "no." (for example, predict whether a certain piece of equipment will experience a mechanical failure)

The most prominent and common algorithms used in machine learning historically and today come in three groups: **linear models, tree-based models, and neural networks**.

The terms we've chosen to define here are commonly used in machine learning. Whether you're working on a project that involves machine learning or if you're just curious about what's going on in this part of the data world, we hope you'll find these definitions clear and helpful.



AN INTRODUCTION TO KEY DATA SCIENCE CONCEPTS

Definitions of 10 fundamental terms for data science and machine learning.



model [n]

['mɒdəl] / noun

1. a mathematical representation of a real world process; a predictive model forecasts a future outcome based on past behaviors.



training [v]

['treɪnɪŋ] / verb

1. the process of creating a model from the training data. The data is fed into the training algorithm, which learns a representation for the problem, and produces a model. Also called "learning".



classification [n]

[klæsəfə'keɪʃən] / noun

1. a prediction method that assigns each data point to a predefined category, e.g., a type of operating system.



algorithm [n]

['ælɡə'rɪðəm] / noun

1. a set of rules used to make a calculation or solve a problem.



regression [n]

[rɛ'ɡreʃən] / noun

1. a prediction method whose output is a real number, that is, a value that represents a quantity along a line. Example: predicting the temperature of an engine or the revenue of a company.



target [n]

['tɑːɡət] / noun

1. in statistics, it is called the dependent variable; it is the output of the model or the variable you wish to predict.



test set [n]

['test set] / noun

1. a dataset, separate from the training set but with the same structure, used to measure and benchmark the performance of various models.



overfitting [v]

['oʊvər'fɪtɪŋ] / verb

1. a situation in which a model that is too complex for the data has been trained to predict the target. This leads to an overly specialized model, which makes predictions that do not reflect the reality of the underlying relationship between the features and target.

feature [n]

['fi:tʃər] / noun

1. also known as an independent variable or a predictor variable, a feature is an observable quantity, recorded and used by a prediction model. You can also engineer features by combining them or adding new information to them.



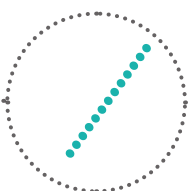
Top Prediction Algorithms

Pros and cons of some of the most common machine learning algorithms

Now, let's take a look at some of the major types of machine learning algorithms. We can group them into three buckets:

linear models - tree-based models - and neural networks

Linear Model Approach



A **linear model** uses a simple formula to find the “best fit” line through a set of data points. This methodology dates back over 200 years, and it has been used widely throughout statistics and machine learning. It is useful for statistics because of its simplicity -- the variable you want to predict (the dependent variable) is represented as an equation of variables you know (independent variables), and so prediction is just a matter of inputting the independent variables and having the equation spit out the answer.

For example, you might want to know how long it will take to bake a cake, and your regression analysis might yield an equation $t = 0.5x + 0.25y$, where t is the baking time in hours, x is the weight of the cake batter in kg, and y is a variable which is 1 if it is chocolate and 0 if it is not. If you have 1 kg of chocolate cake batter (we love cake), then you plug your variables into our equation, and you get $t = (0.5 \times 1) + (0.25 \times 1) = 0.75$ hours, or 45 minutes.

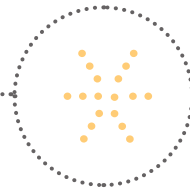
Tree-Based Model Approach



When you hear tree-based, think decision trees, i.e., a sequence of branching operations.

A **decision tree** is a graph that uses a branching method to show each possible outcome of a decision. Like if you're ordering a salad, you first decide the type of lettuce, then the toppings, then the dressing. We can represent all possible outcomes in a decision tree. In machine learning, the branches used are binary yes/no answers.

Neural Networks









Neural networks refer to a biological phenomenon comprised of interconnected neurons that exchange messages with each other. This idea has now been adapted to the world of machine learning and is called ANN (Artificial Neural Networks).

Deep learning, which you've heard a lot about, can be done with several layers of neural networks put one after the other.

ANNs are a family of models that are taught to adopt cognitive skills.



TOP PREDICTION ALGORITHMS

	TYPE	NAME	DESCRIPTION	ADVANTAGES	DISADVANTAGES
Linear		Linear regression	The “best fit” line through all data points. Predictions are numerical.	Easy to understand -- you clearly see what the biggest drivers of the model are.	<ul style="list-style-type: none"> X Sometimes too simple to capture complex relationships between variables. X Does poorly with correlated features.
		Logistic regression	The adaptation of linear regression to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	<ul style="list-style-type: none"> X Sometimes too simple to capture complex relationships between variables. X Does poorly with correlated features.
Tree-based		Decision tree	A series of yes/no rules based on the features , forming a tree, to match all possible outcomes of a decision.	Easy to understand.	<ul style="list-style-type: none"> X Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
		Random Forest	Takes advantage of many decision trees, with rules created from subsamples of features. Each tree is weaker than a full decision tree, but by combining them we get better overall performance.	A sort of “wisdom of the crowd”. Tends to result in very high quality models. Fast to train.	<ul style="list-style-type: none"> X Models can get very large. X Not easy to understand predictions.
		Gradient Boosting	Uses even weaker decision trees, that are increasingly focused on “hard” examples.	High-performing.	<ul style="list-style-type: none"> X A small change in the feature set or training set can create radical changes in the model. X Not easy to understand predictions.
Neural networks		Neural networks	Interconnected «neurons» that pass messages to each other. Deep learning uses several layers of neural networks stacked on top of one another.	Can handle extremely complex tasks - no other algorithm comes close in image recognition.	<ul style="list-style-type: none"> X Very slow to train, because they often have a very complex architecture. X Almost impossible to understand predictions.

How to Evaluate Models

Metrics and methodologies for choosing the best model

By now, you might have already created a machine learning model. But now the question is: **how can you tell if it's a good model?**

It depends on **what kind of model you've built.**

Metrics for Evaluating Models

We've already talked about training sets and test sets -- this is when you break your data into two parts: one to train your model and the other to test it. After you've trained your model using the training set, you want to test it with the test set. Makes sense, right? **So now, which metrics should you use with your test set?**

There are several metrics for evaluating machine learning models, depending on whether you are working with a regression model or a classification model.

For regression models, you want to look at mean squared error and R^2 . **Mean squared error** is calculated by computing the square of all errors and averaging them over all observations. The lower this number is, the more accurate your predictions were. **R^2** (pronounced **R-Squared**) is the percentage of the observed variance from the mean that is explained (that is, predicted) by your model. R^2 always falls between 0 and 1, and a higher number is better.

For classification models, the most simple metric for evaluating a model is accuracy. **Accuracy** is a common word, but in this case we have a very specific way of calculating it. Accuracy is the percentage of observations which were correctly predicted by the model. Accuracy is simple to understand, but should be interpreted with caution, in particular when the various classes to predict are unbalanced.

Another metric you might come across is the **ROC AUC**, which is a measure of accuracy and stability. AUC stands for "area under the curve". A higher ROC AUC generally means you have a better model. **Logarithmic loss**, or log loss, is a metric often used in competitions like those run by Kaggle, and it is applied when your classification model outputs not strict classifications (e.g., true and false) but class membership probabilities (e.g., a 10 percent chance of being true, a 75 percent chance of being true, etc.). Log loss applies heavier penalties to incorrect predictions that your model made with high confidence.

Overfitting and Regularization

When you train your model using the training set, the model learns the underlying patterns in that training set in order to make predictions. But the model also learns peculiarities of that data that don't have any predictive value. And when those peculiarities start to influence the prediction, we'll do such a good job at explaining our training set that the performance on the test set (and on any new data, for that matter) will suffer. This is called overfitting, and it can be one of the biggest challenges to building a predictive model. The remedy for overfitting is called **regularization**, which is basically just the process of simplifying your model or of making it less specialized.

For linear regression, regularization takes the form of **L2 and L1 regularization**. The mathematics of these approaches are out of our scope in this post, but conceptually they're fairly simple. Imagine you have a regression model with a bunch of variables and a bunch of coefficients, in the model $y = C_1a + C_2b + C_3c \dots$, where the C s are coefficients and a , b , and c are variables. What L2 regularization does is reduce the magnitude of the coefficients, so that the impact of individual variables is somewhat dulled.

Now, imagine that you have a lot of variables -- dozens, hundreds, or even more -- with small but non-zero coefficients. L1 regularization just eliminates a lot of these variables, working under the assumption that much of what they're capturing is just noise.

For decision tree models, regularization can be achieved through setting tree depth. A deep tree -- that is, one with a lot of decision nodes -- will be complex, and the deeper it is, the more complex it is. By limiting the depth of a tree, by making it more shallow, we accept losing some accuracy, but it will be more general.



HOW TO EVALUATE MODELS

The validation step is where you optimize the parameters for each algorithm you want to use. The two most common approaches are k-fold cross validation and a validation set — which approach you use will depend on your requirements and constraints!

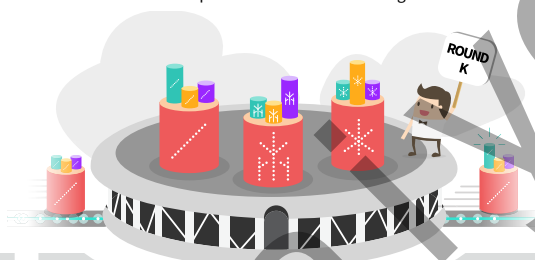
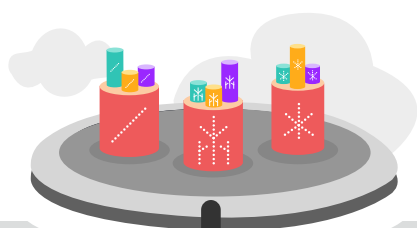
VALIDATION SET

A validation set reduces the amount of data you can use, but it is simple and cheap

VS

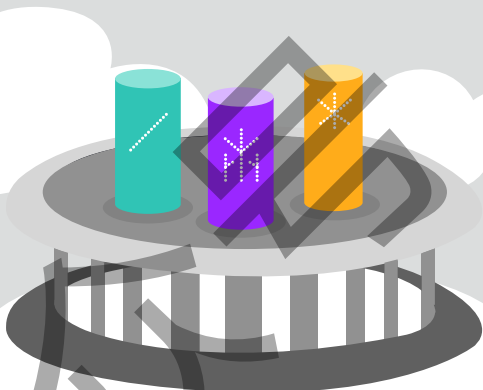
K-FOLD CROSS VALIDATION

K-folds will provide better results, but it is more expensive and time-consuming



2. TEST STEP

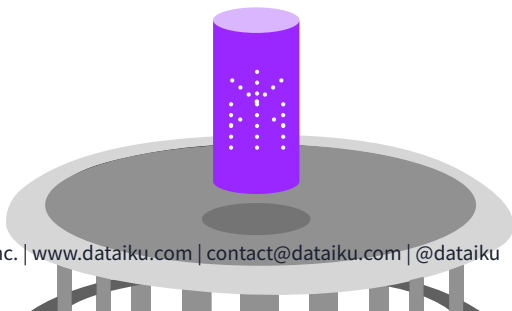
The test step is where you take the best version of each algorithm and apply it to your test set — a set of data that has not been used in either the training or validating of the models. Your challenge here is to decide which metric to use for evaluation.



1. Linear	0.83
2. Tree	0.78
3. Other	0.71




3. YOUR BEST MODEL

Based on the metrics you chose, you will be able to evaluate one algorithm against another and see which performed best on your test set. Now you're ready to deploy the model on brand new data!








has become too complex. The remedy for overfitting is called **regularization**, which is the process of simplifying the model, of making it less specialized.

Legend

	Linear Model
	Tree Model
	Other Algorithms

Common metrics for evaluating models

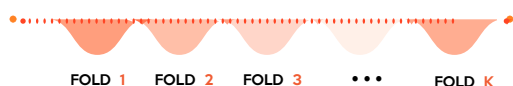
	Mean-squared error
	R-Squared
	Accuracy
	ROC AUC
	Log loss

	Human	Machine
Validation Step	Selects algorithms Selects metrics	Runs calculation
Test Step	Selects metrics	Runs calculation
Model	Decides how to apply the model	Scores New data

Introducing the K-fold Strategy and the Hold-Out Strategy

Comparing two popular methodologies

When you build a model, **don't wait until you've already run it on the test set** to discover that it's overfitted. Instead, the best practice is **to evaluate regularization techniques on your model** before touching the test set. In some ways, this validation step is just stage two of your training.

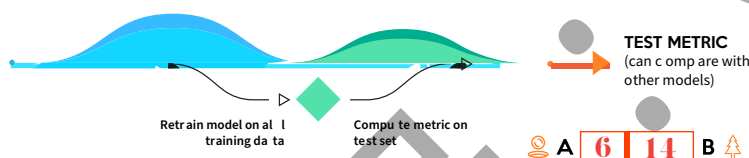
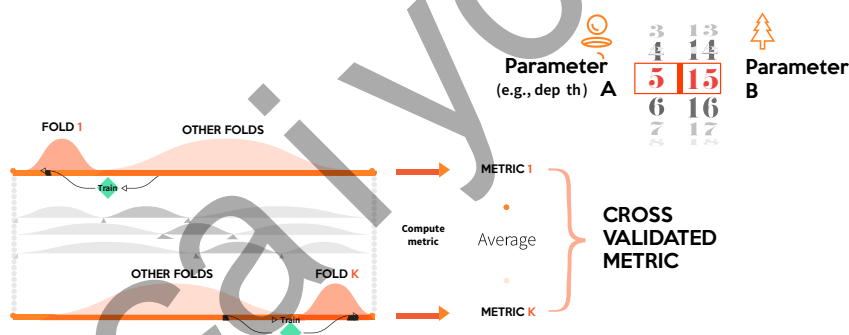


The preferred method of validating a model is called K-fold Cross-Validation.

To do this, you **take your training set and split it into some number -- called K (hence the name) -- sections, or folds.**

Then, for each combination of parameters (e.g., tree depth of five and 15 trees), test the model on the fold and **calculate the error after training it on the rest of the training set** NOT part of the fold -- **and then continue doing this until you've calculated the error on all K folds.**

The average of these errors is your **cross-validated error** for each combination of parameters.



Then, **choose the parameters with the best cross-validated error**, train your model on the full training set, and then compute your error on the test set -- which until now hasn't been touched.

This test error can now be used to compare with other algorithms.

The drawback of K-fold cross-validation is that it can take up a lot of time and computing resources. A more efficient though less robust approach is **to set aside a section of your training set and using it as a validation set -- this is called the hold-out strategy.**

The hold-out validation set could be the same size as your test set, so you might wind up with a split of 60-20-20 among your training set, validation set, and test set.

Then, for each parameter combination, calculate the error on your validation set, and then choose the model with the lowest error to then calculate the test error on your test set.

This way, you will have the confidence that you have properly evaluated your model before applying it in the real world.

Pros of the hold-out strategy

Fully independent data; only needs to be run once so has lower computational costs.

Cons of the hold-out strategy

Performance evaluation is subject to higher variance given the smaller size of the data.

Pros of the K-fold strategy

Prone to less variation because it uses the entire training set.

Cons of the K-fold strategy

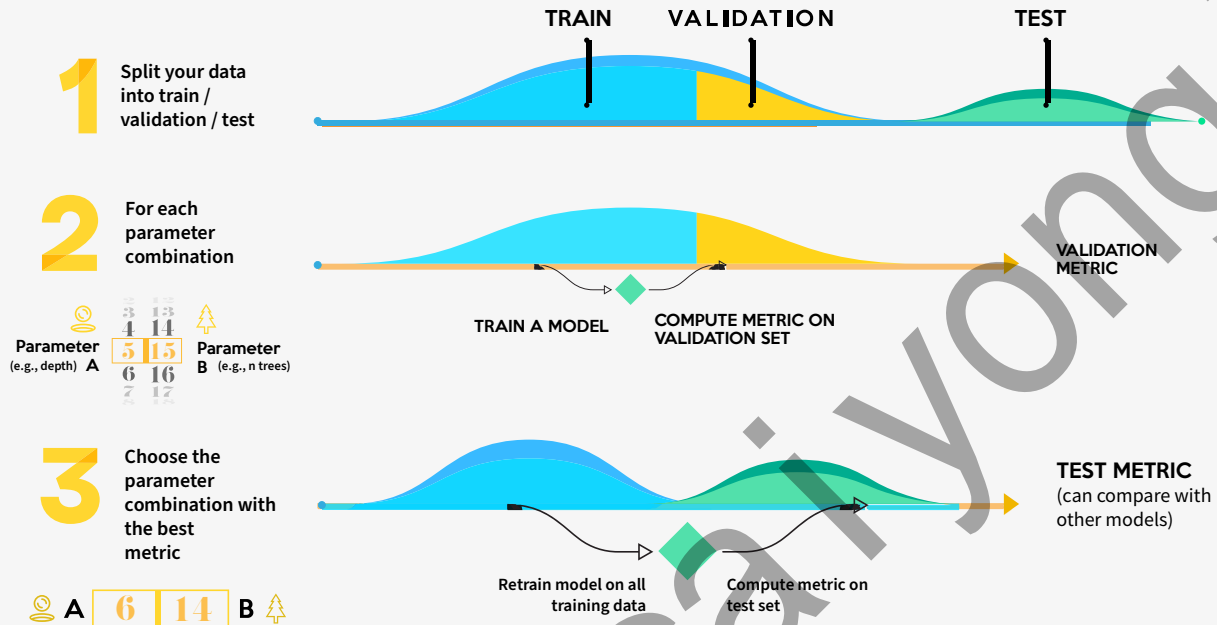
Higher computational costs; the model needs to be trained K times at the validation step (plus one more at the test step).



HOW TO VALIDATE YOUR MODEL

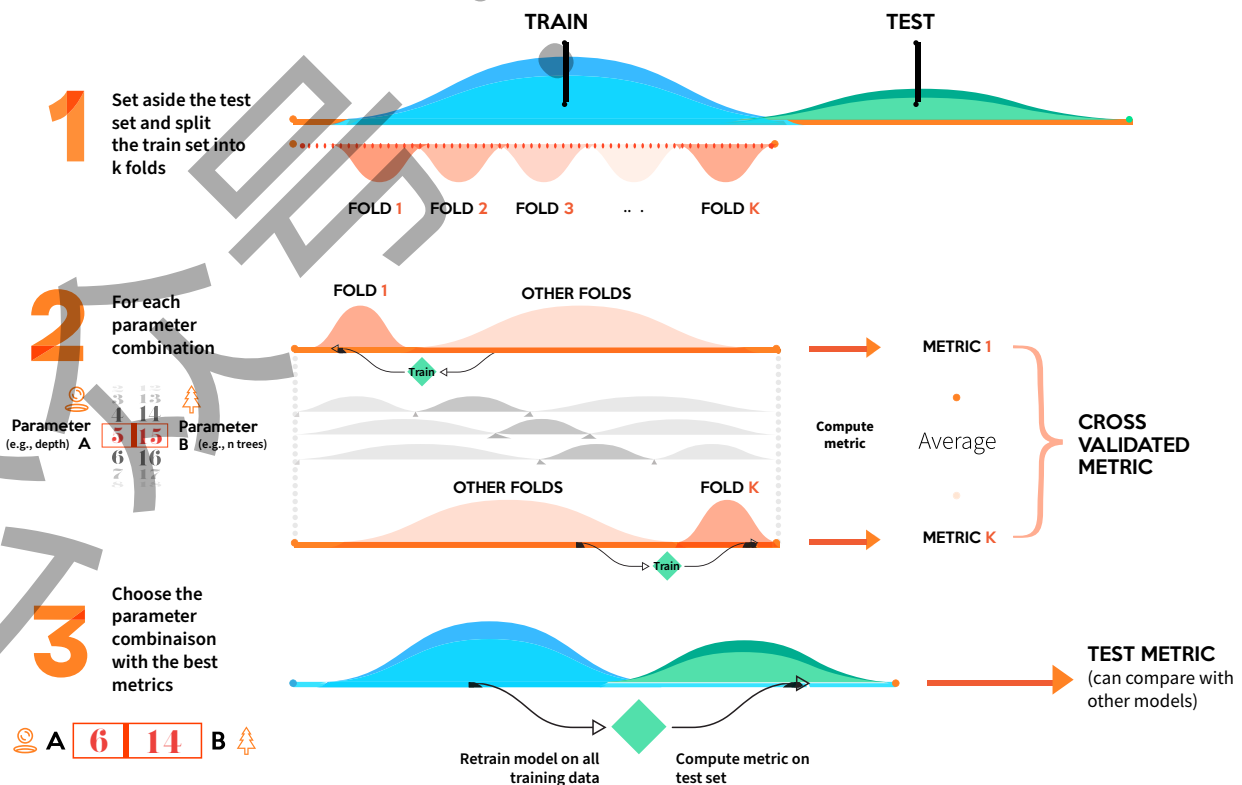
HOLDOUT STRATEGY

For a given model



K-FOLD STRATEGY

For a given model



Unsupervised Learning and Clustering

An overview of the most common example of unsupervised learning

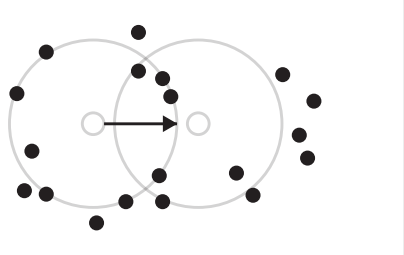
What do we mean by **unsupervised learning**?

By unsupervised, it means that we're not trying to predict a variable; instead, we want to discover hidden patterns within our data that will let us to identify groups, or clusters, within that data.



Previously, we've discussed regression and classification algorithms. These are types of supervised learning; that is, we have a specific variable we are looking to predict based on other, independent variables. Here we'll discuss clustering models, which is one of the simplest types of unsupervised learning.

Clustering is often used in marketing in order to group users according to multiple characteristics, such as location, purchasing behavior, age, and gender. It can also be used in scientific research; for example, to find population clusters within DNA data.

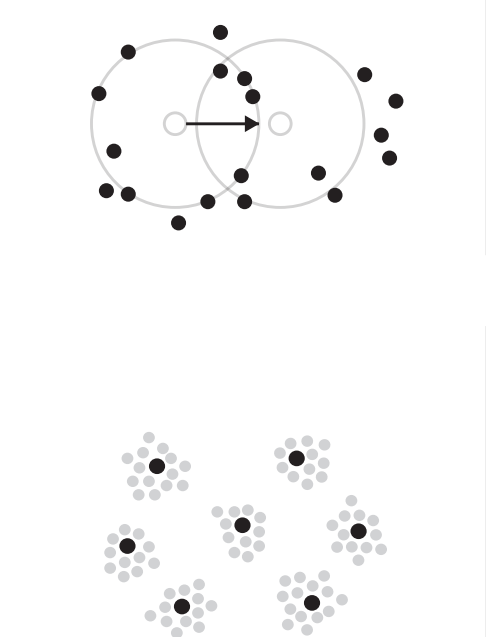


One simple clustering algorithm is DBSCAN. In DBSCAN, you select a distance for your radius, and you select a point within your dataset -- then, all other data points within the radius's distance from your initial point are added to the cluster. You then repeat the process for each new point added to the cluster, and you repeat until no new points are within the radii of the most recently added points. Then, you choose another point within the dataset and build another cluster using the same approach.

DBSCAN is intuitive, but its effectiveness and output rely heavily on what you choose for a radius, and there are certain types of distributions that it won't react well to.

The most widespread clustering algorithm is called k-means clustering.

In k-means clustering, we pre-define the number of clusters we want to create -- the number we choose is the k, and it is always a positive integer.



To run k-means clustering, we begin by randomly placing k starting points within our dataset. These points are called centroids, and they become the prototypes for our k clusters. We create these initial clusters by assigning every point within the dataset to its nearest centroid. Then, with these initial clusters created, we calculate the midpoint of each of them, and we move each centroid to the midpoint of its respective cluster. After that, since the centroids have moved, we can then reassign each data point to a centroid,

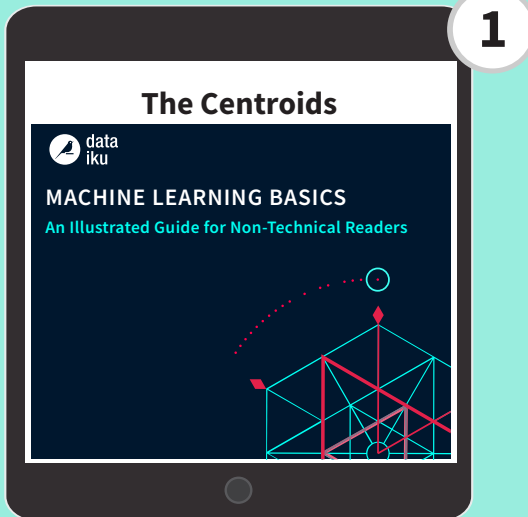
create an updated set of clusters, and calculate updated midpoints. **We continue iterating**

for a predetermined number of times -- 300 is pretty standard. By the time we get to the end, the centroids should move minimally, if at all.



K-MEANS CLUSTERING ALGORITHM IN ACTION

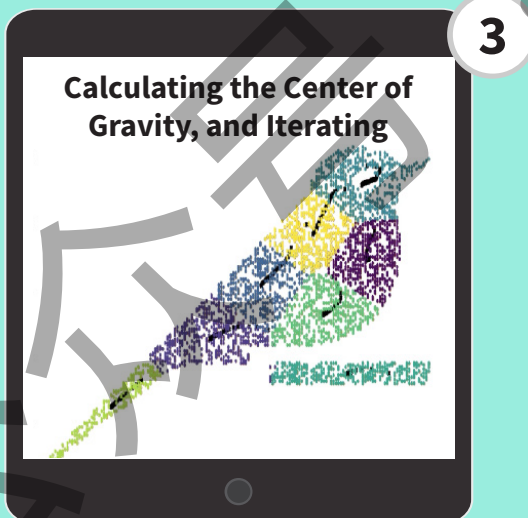
A popular clustering algorithm, k-means clustering identifies clusters via an iterative process. The «k» in k-means is the number of clusters, and it is chosen before the algorithm is run.



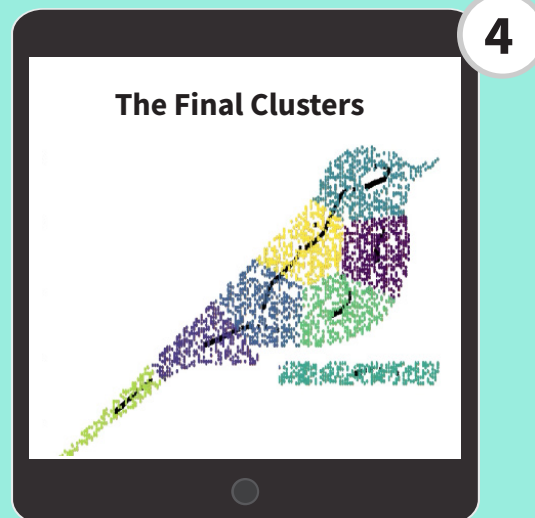
1. First, **we choose the number of clusters we want** -- in this case, we choose eight. Thus, **eight centroids are randomly chosen** within our dataset.



2. Each data point is assigned to its closest centroid -- this creates the first set of clusters, which are represented by different colors.



3. The **midpoint** -- also called the center of gravity -- **for each cluster is calculated**, and the centroids are moved to these points. Then, new clusters are formed, and the process is iterated.



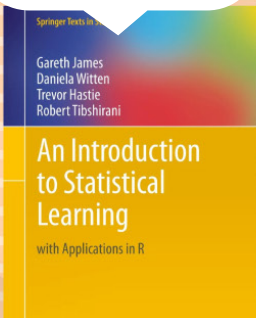
4. The algorithm is terminated after a **pre-determined number of iterations** -- in this case, we use 300, which is a common setting. The result: our final clusters!

For further exploration

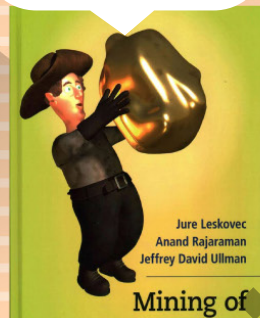
Our favorite resources for diving into machine learning

BOOKS

A theoretical, but clear and comprehensive, textbook: [An Introduction to Statistical Learning](#) by Hastie, Tibshirani, and Friedman



Anand Rajaraman and Jeffrey Ullman's [book \(or PDF\)](#), [Mining of Massive Datasets](#), for some advanced but very practical use cases and algorithms



[Python Machine Learning](#): a practical guide around scikit-learn. "This was my first machine learning book, and I owe a lot to it" says one of our senior data scientists.

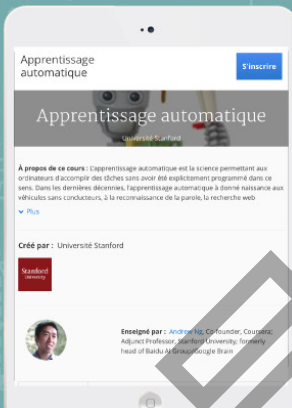


COURSES

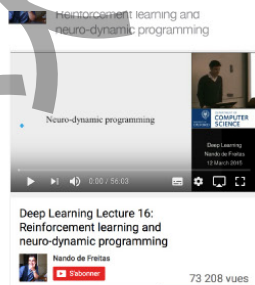


[Andrew Ng's Coursera/Stanford course on machine learning](#) is basically a requirement!

"Intro to Machine Learning" on Udacity is a good [introduction to machine learnings](#) for people who already have basic Python knowledge. The very regular quizzes and exercises make it particularly interactive.



VIDEOS & OTHER



Oxford professor [Nando de Freitas's 16-episode deep learning lecture series](#) on YouTube



Open-source machine learning libraries, such as scikit-learn (and their [great user guide](#)), Keras, TensorFlow, and MLlib





Your Path to Enterprise AI

Dataiku is one of the world's leading AI and machine learning platforms, supporting agility in organizations' data efforts via collaborative, elastic, and responsible AI, all at enterprise scale. Hundreds of companies use Dataiku to underpin their essential business operations and ensure they stay relevant in a changing world.

300+
CUSTOMERS

30,000+
ACTIVE USERS

*data scientists, analysts, engineers, & more

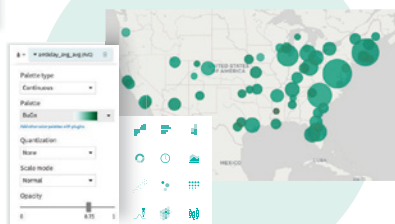
1. Clean & Wrangle

Name	Sex	Age
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22
Robertson, M.	male	22

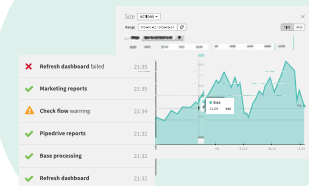
2. Build + Apply Machine Learning



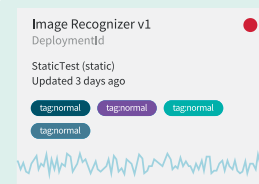
3. Mining & Visualization



5. Monitor & Adjust



4. Deploy to production



GUIDEBOOK

www.dataiku.com