dataiku

Machine Learning Basics:
An Illustrated Guide
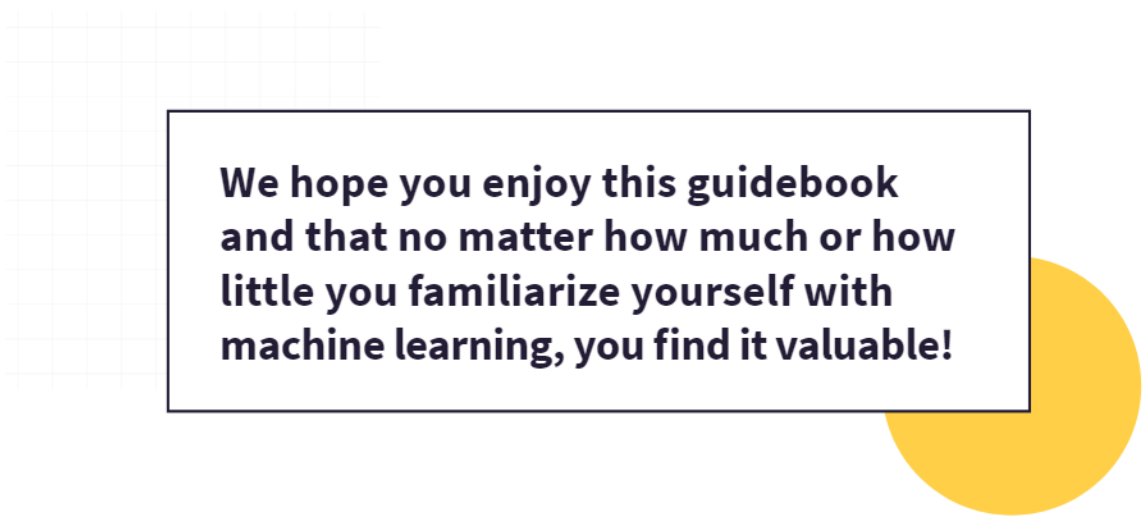# for Non-Technical Readers

Introduction: Machine Learning Concepts for Everyone

According to Google Trends, interest in the term machine learning (ML) has increased over 490% 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.

While this has resulted in a plethora of innovations and improvements among our customers and for organizations worldwide, it has 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 many resources out there that answer the question, "What is machine learning?" while using a minimum of technical terms. The basic concepts of machine learning are actually not 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 finally we'll finish with some resources that you can explore to learn more.

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

An Introduction to Key Data Science Concepts

## A First Look at 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?

As humans, we learn through past experiences. We use our senses to obtain these "experiences" and use them later to survive. Machines learn through commands provided by humans. These sets of rules are known as algorithms.
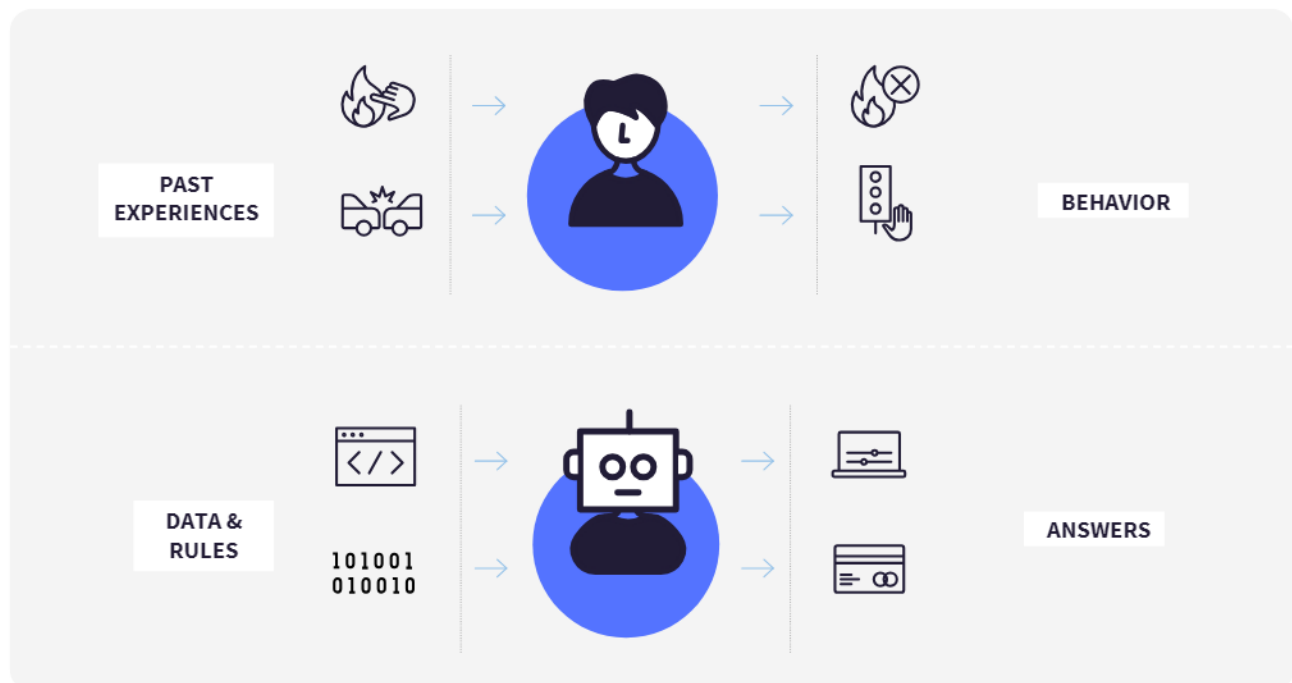
Algorithms are sets of rules that a computer is able to follow. Think about how you learned to do long division — maybe you learned to divide the denominator into the first digits of the numerator, subtract the subtotal, and continue with the next digits until you were left with a remainder.

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.

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. This technique is used for data that has a precise mapping between input and output, referred to as labeled data. This is known as supervised learning. For example, using a set of characteristics of a house, we can predict its sale price. We will discuss unsupervised learning later in the guidebook when exploring clustering methods.

➔ **LABELED DATA**

Patient information				Label	
AGE	GENDER	SMOKING	VACCINATION	SICK	
18	M	0	1	0	.... Class label
41	F	0	0	0	1= SICK
33	F	1	0	1	0 = NOT SICK
24	M	0	1	0	
65	M	1	0	1	
19	F	0	1	0	

➔ **UNLABELED DATA**

| Customer information				
AGE	GENDER	MARITAL STATUS	OCCUPATION	PRODUCT CODE
18	Single	0	Student	S212
41	Married	0	Teacher	M211
33	Single	1	Nurse	S600
24	Single	0	Store manager	0S703
65	Married	1	Retired	M107

Customer information
+
Product Purchase
=
Pattern/ Similarities

Prediction problems are divided into two main categories:

Regression problems, where the variable to predict is numerical. For example: Understand how the number of sleep hours and study hours (the predictors) determine a students' test scores.

→ **PREDICTION: REGRESSION**

Hours of study

PREDICTIORS

Hours of sleep

71
78
57
63

RESULT
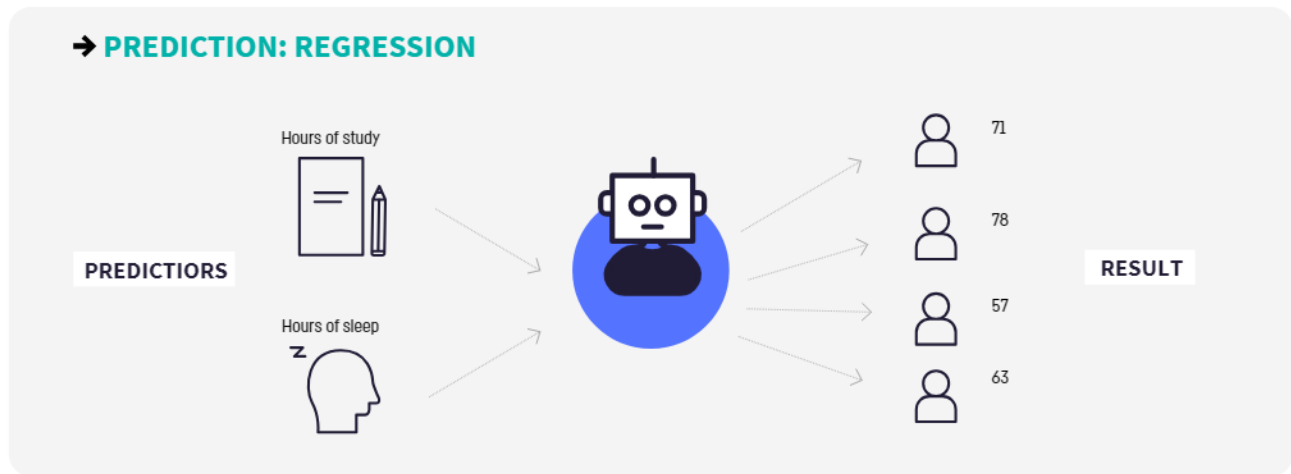
Classification problems, where the variable to predict is part of one of some number of pre-defined categories, which can be as simple as "yes" or "no." For example: Understand how the number of sleep hours and study hours (the predictors) determine if a student will Succeed or Fail, where "Succeed" and "Fail" are two class labels.

→ **PREDICTION: CLASSIFICATION**

Hours of study

PREDICTIORS

Hours of sleep

Succeed

Fail
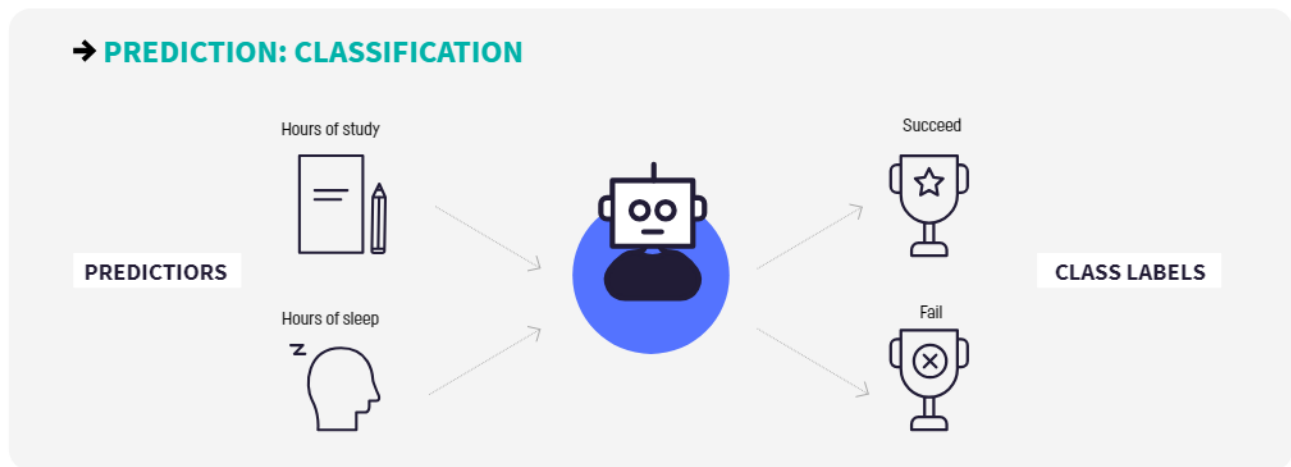
CLASS LABELS

One way to remember this distinction is that classification is about predicting a category or class label, while regression is about predicting a quantity. 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. We'll come back to this and explain these groups more in-depth, but first, let's take a step back and define a few key terms.

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.

Definitions of 10 Fundamental Terms for Data Science and Machine Learning

MODEL [ˈmɑdəl] / noun

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

TRAINING [ˈtreɪnɪŋ] / 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 [ˌklæsəfəˈkeɪʃən] / noun

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

TRAINING SET [ˈtreɪnɪŋ sɛt] / noun

1. a dataset used to find potentially predictive relationships that will be used to create a model.

FEATURE [ˈfitfər] / 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.

ALGORITHM [ˈælgəˌrɪðəm] / noun

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

REGRESSION [rəˈgrɛʃən] / noun

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

TARGET [ˈtɑrgət] / 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 [tɛst sɛt] / 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 [ˌoʊvərˈfɪtɪŋ] / 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.
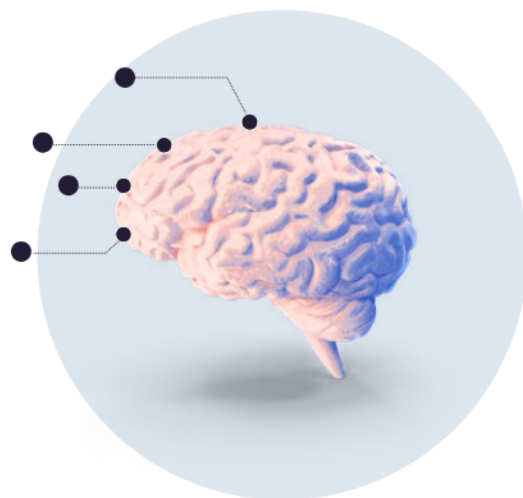
Top Prediction Algorithms

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.

Note that linear regressions can be simple or multiple. In multiple linear regression, the value of the target variable changes based on the value of more than one independent variable, or x.

Linear Models

Katie Gross

"Remember, linear models generate a formula to create a best-fit line to predict unknown values. Linear models are considered "old school" and often not as predictive as newer algorithm classes, but they can be trained relatively quickly and are generally more straightforward to interpret, which can be a big plus!"

➜ 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.

Tree-Based Models

Katie Gross

"Tree-based models are very popular in machine learning. The decision tree, the foundation of tree-based models, is quite straightforward to interpret, but generally a weak predictor. Ensemble models can be used to generate stronger predictions from many trees, with random forest and gradient boosting as two of the most popular. All tree-based models can be used for regression or classification and can handle non-linear relationships quite well."
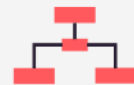
➜ NEURAL NETWORKS

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.

	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.	Sometimes too simple to capture complex relationships between variables. Does poorly with correlated features.
Linear		**Logistic Regression**	The adaptation of **linear regression** to problems of classification (e.g., yes/no questions, groups, etc.	Also easy to understand.	Sometimes too simple to capture complex relationships between variables. 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.	Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
Tree-Based		**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.	Models can get very large. Not easy to understand predictions.
Tree-Based		**Gradient Boosting**	Uses even weaker decision trees, that are increasingly **focused on "hard" examples.**	High-performing.	A small change in the feature set or training set can create radical changes in the model. 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.	Very slow to train, because they often have a very complex architecture. 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?

Not even sure where to start when it comes to building, much less evaluating models? Dataiku can help! 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. Throughout this guide, we'll show you tips like the following on how tools like Dataiku can help you if you're new to machine learning.
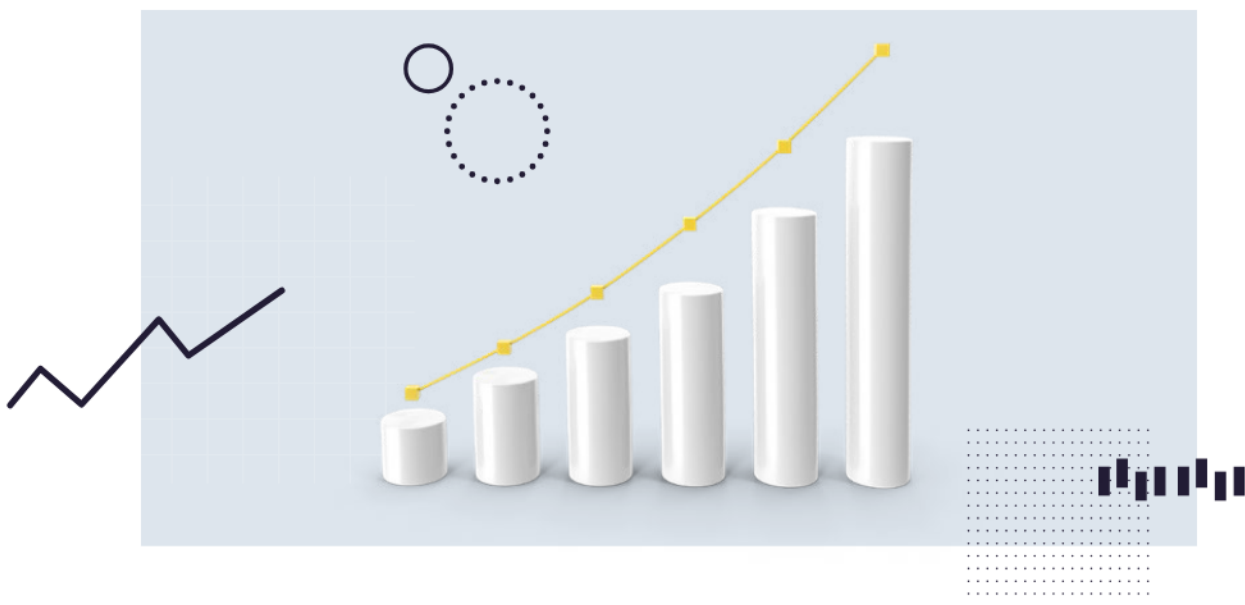
Result Tab

Dataiku is the world's leading AI and machine learning platform, supporting agility in organizations' data efforts via collaborative, elastic, and responsible AI, all at enterprise scale.

In Dataiku the Models page of a visual analysis consists of a Result tab that is useful for comparing model performance across different algorithms and training sessions. By default, the models are grouped by Session. However, we can select the Models view to assess all models in one window, or the Table view to see all models along with more detailed metrics.

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 R2. 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 are.

R2 (pronounced R-Squared) is the percentage of the observed variance from the mean that is explained (that is, predicted) by your model. R2 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% chance of being true, a 75% chance of being true, etc.). Log loss applies heavier penalties to incorrect predictions that your model made with high confidence.

Dataiku displays algorithm-dependent interpretation panels. For example, a linear model, such as Logistic Regression, will display information about the model's coefficients, instead of variable importance, as in the case of an XGBoost model.
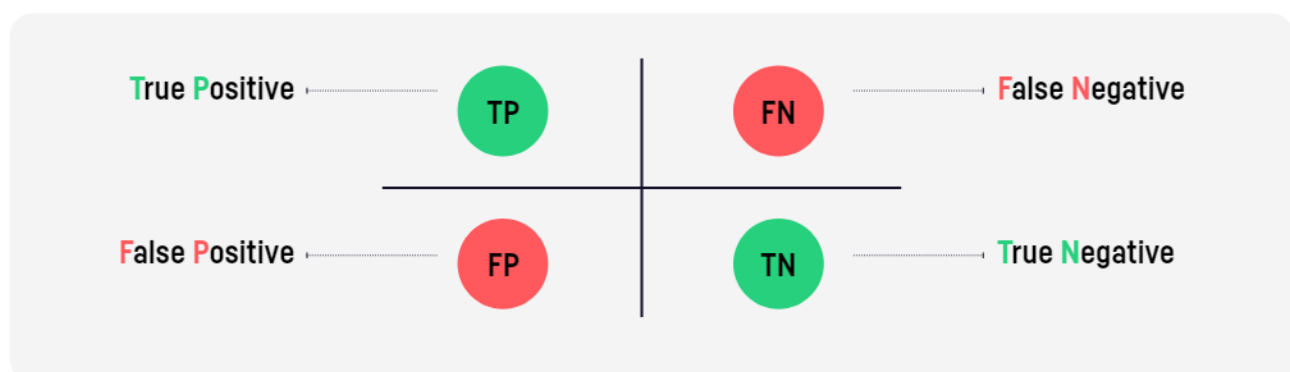
→ THE CONFUSION MATRIX

One tool used to evaluate and compare models is a simple table known as a confusion matrix. The number of columns and rows in the table depends on the number of possible outcomes. For binary classification, there are only two possible outcomes, so there are only two columns and two rows.

The labels that make up a confusion matrix are TP, or true positive, FN, or false negative, FP, or false positive, and TN, or true negative.

There will always be errors in prediction models. When a model incorrectly predicts an outcome as true when it should have predicted false, it is labeled as FP, or false positive. This is known as a type I error. When a model incorrectly predicts an outcome as false when it should have predicted true, it is labeled as FN, or false negative. This is known as a type II error.

Depending on our use case, we have to decide if we are more willing to accept higher numbers of type I or type II errors. For example, if we were classifying patient test results as indicative of cancer or not, we would be more concerned with a high number of false negatives. In other words, we would want to minimize the number of predictions where the model falsely predicts that a patient's test result is not indicative of cancer.

Similarly, if we were classifying a person as either guilty of a crime, or not guilty of a crime, we would be more concerned with high numbers of false positives. We would want to reduce the number of predictions where the model falsely predicts that a person is guilty.

True Positive ··········	TP		FN	·········· False Negative
False Positive ··········	FP		TN	·········· True Negative

> **Confusion Matrix**
>
> The Confusion matrix compares the actual values of the target variable with the predicted values. In addition, Dataiku displays some associated metrics, such as "precision," "recall," and the "F1-score." By default, Dataiku displays the Confusion matrix and associated metrics at the optimal threshold (or cut-off). However, manual changes to the cut-off value are reflected in both the Confusion matrix and associated metrics in real-time.

➔ 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 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 $y = C1a + C2b + C3c...$, where the Cs 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, making it more shallow, we accept losing some accuracy, but it will be more general.

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!
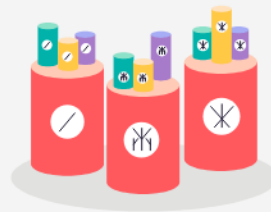
1. Your Approach

| **VALIDATION SET** | **VS** | **K-FOLD CROSS VALIDATION** |

A validation set reduces the amount of data you can use, but it is simple and cheap
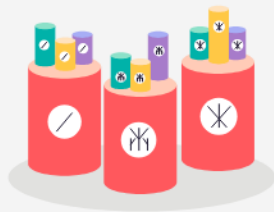
K-folds provide better results, but are more expensive and take more time

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.

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!

Legend

/	LINEAR MODEL
	TREE MODEL
	OTHER ALGORITHMS

Common Metrics for Evaluating Models

$\diamond^2$	MEAN-SQUARED ERROR
R^2	R-SQUARED
	ACCURACY
	ROC AUC
	LOG LOSS

	HUMAN	MACHINE
Validation step	Selects algorithms Select 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 2 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 validation step is where we will optimize the hyperparameters for each algorithm we want to use. Let's take a look at the definition of parameter and hyperparameter. In machine learning, tuning the hyperparameters is an essential step in improving machine learning models.

Model parameters are attributes about a model after it has been trained based on known data. You can think of model parameters as a set of rules that define how a trained model makes predictions. These rules can be an equation, a decision tree, many decision trees, or something more complex. In the case of a linear regression model, the model parameters are the coefficients in our equations that the model "learns" — where each coefficient shows the impact of a change in an input variable on the outcome variable.

A machine learning practitioner uses an algorithms' hyperparameters as levers to control how a model is trained by the algorithm. For example, when training a decision tree, one of these controls, or hyperparameters, is called max_depth. Changing this max_depth hyperparameter controls how deep the eventual model may go.

It is an algorithm's responsibility to find the optimal parameters for a model based on the training data, but it is the machine learning practitioner's responsibility to choose the right hyperparameters to control the algorithm.

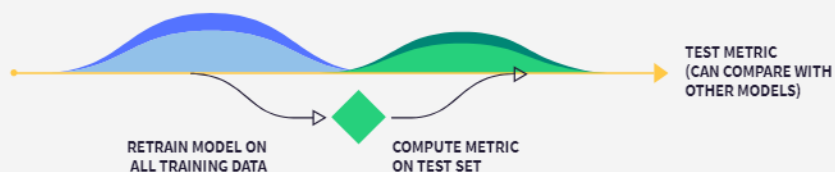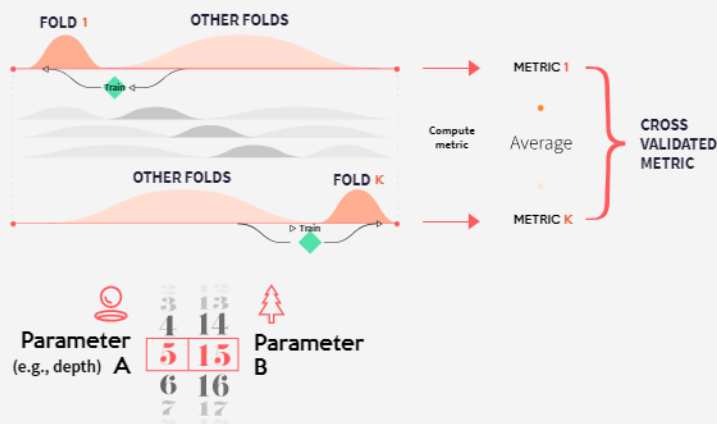**FOLD 1 FOLD 2 FOLD 3 . . FOLD K**
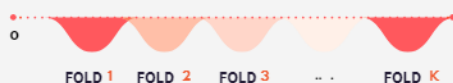
→ The preferred method of validating a model is called K-fold Cross-Validation. **To do this, you take your training set and split it into some number — called K (hence the name) — sections, or folds.**

↓

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.**

FOLD 1 OTHER FOLDS

train

OTHER FOLDS FOLD K

▷ Train

Compute metric Average

METRIC 1

METRIC K

CROSS VALIDATED METRIC

Parameter (e.g., depth) A

3   13
4   14
5   15
6   16
7   17

Parameter B

RETRAIN MODEL ON ALL TRAINING DATA

COMPUTE METRIC ON TEST SET

TEST METRIC (CAN COMPARE WITH OTHER MODELS)

↓

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 use it as a validation set — this is called the hold-out strategy.

- The held-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).

Train/Test Set

Data science and machine learning platforms can do a lot of the heavy lifting for you when it comes to train/test data. For example, during the model training phase, Dataiku "holds out" on the test set, and the model is only trained on the train set. Once the model is trained, Dataiku evaluates its performance on the test set.
This ensures that the evaluation is done on data that the model has never seen before.

By default, Dataiku randomly splits the input dataset into a training and a test set, and the fraction of data used for training can be customized (though again, 80% is a standard fraction of data to use for training). For more advanced practitioners, there are lots more settings in Dataiku that can be tweaked for the right train/test set specifications.
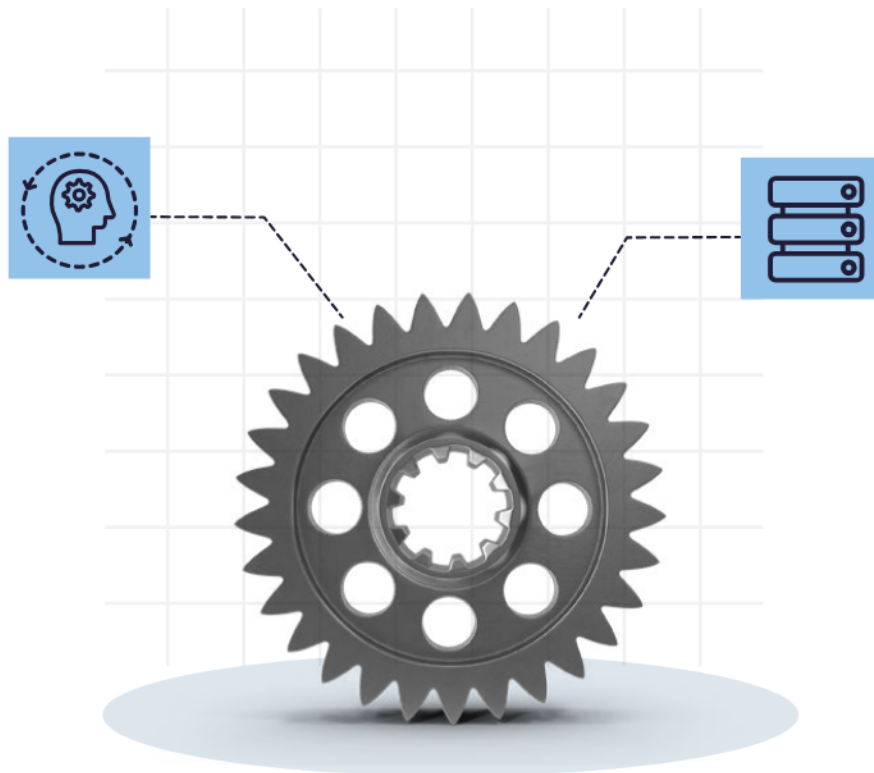
How to Validate Your Model

→ HOLDOUT STRATEGY

For a given model
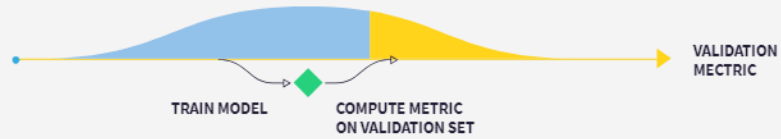
1 Split your data into train/ validation / test

TRAIN VALIDATION TEST

2 For each parameter combination

Parameter (e.g., depth) **A** Parameter **B** (e.g., n trees)

8 13
4 14
5 15
6 16
7 17

TRAIN MODEL   COMPUTE METRIC ON VALIDATION SET

VALIDATION MECTRIC

3 Choose the parameter combination with the best metric

A **6** **14** B

RETRAIN MODEL ON ALL TRAINING DATA   COMPUTE METRIC ON TEST SET

TEST METRIC (CAN COMPARE WITH OTHER MODELS)
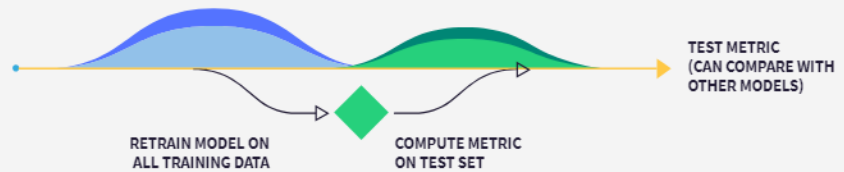
→ K-FOLD STRATEGY

For a given model

1 Set aside the best set and split the train set into k folds
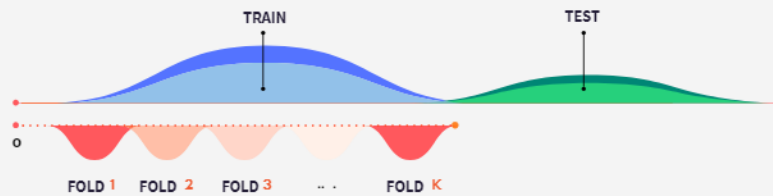
TRAIN TEST

0 FOLD 1 FOLD 2 FOLD 3 ... FOLD K

2 For each parameter combination

Parameter (e.g., depth) **A** Parameter **B** (e, n trees)

8 13
4 14
5 15
6 16
7 17

FOLD 1   OTHER FOLDS   Train   METRIC 1

OTHER FOLDS   FOLD K   Train   METRIC K

Compute metric   Average   CROSS VALIDATED METRIC

3 Choose the parameter combination with the best metrics

A **6** **14** B

Retrain model on all training data   Compute metric on test set

TEST METRIC (CAN COMPARE WITH OTHER MODELS)
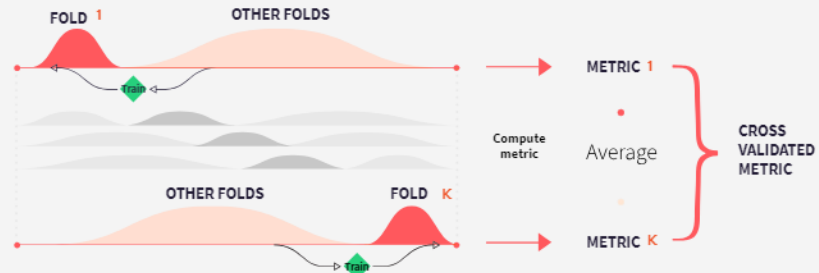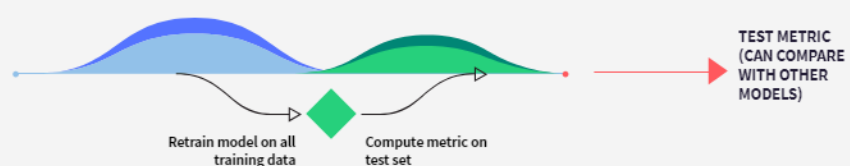
Unsupervised Learning and Clustering

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 identify groups, or clusters, within that data.

UNSUPERVISED LEARNING

Unlabeled

OBSERVATIONS		
S212	M	18
S212	F	41
S600	F	33
M107	M	24
M107	M	65

SUPERVISED LEARNING

Labeled

ID	X	Y
ID1	1	1.2
ID2	2	2.3
ID3	3	2.4
ID4	4	2.9
ID5	5	4.2

CLUSTERING

PREDICTION

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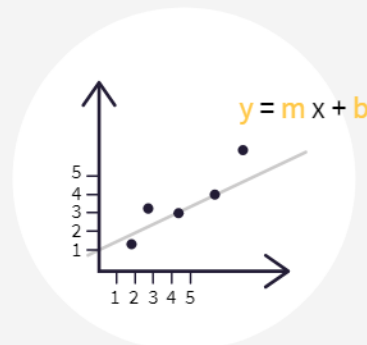
Clustering is often used in marketing in order to group users according to multiple characteristics, such as location, purchasing behavior, age, and gender. It can also be used in scientific research; for example, to find population clusters within DNA data.

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!

→ HIERARCHICAL CLUSTERING

Hierarchical clustering is another method of clustering. Here, clusters are assigned based on hierarchical relationships between data points. There are two key types of hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). Agglomerative is more commonly used as it is mathematically easier to compute, and is the method used by python's scikit-learn library, so this is the method we'll explore in detail.

Here's how it works:

1. Assign each data point to its own cluster, so the number of initial clusters (K) is equal to the number of initial data points (N).

2. Compute distances between all clusters.

3. Merge the two closest clusters.

4. Repeat steps two and three iteratively until all data points are finally merged into one large cluster.
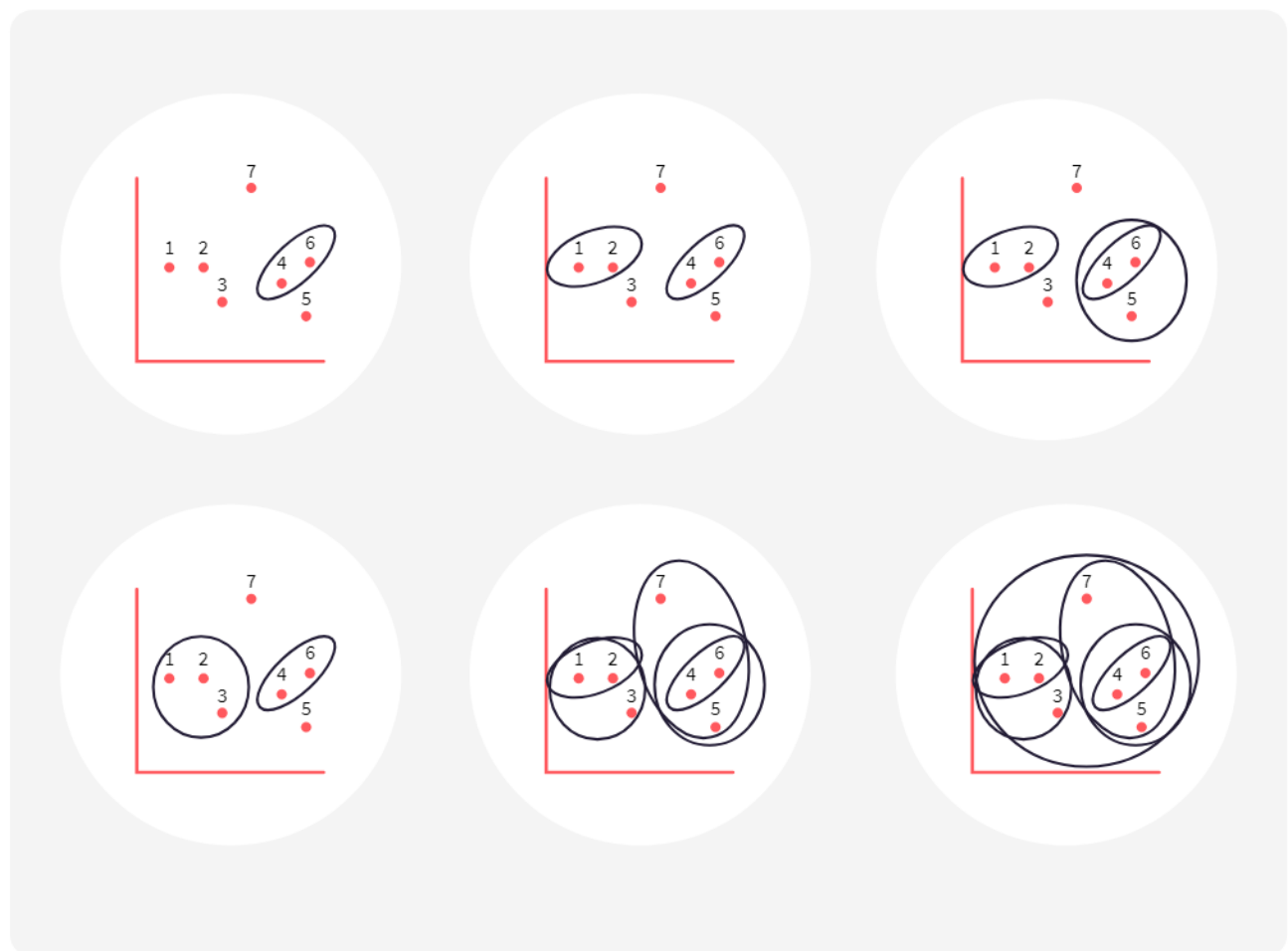
Conclusion

You now have all the basics you need to start testing machine learning.

Don't worry, we won't leave you empty handed though. Here are a few ideas for projects that are relatively straightforward that you can get started with today.

⮕ **How To: Execute Anomaly Detection at Scale**

⮕ **How To: Address Churn with Predictive Analytics**

⮕ **How To: Improve Data Quality With an Efficient Data Labeling Process**

⮕ **How To: Operationalize Data Science**

⮕ **How To: Future Proof your Operations with Predictive Maintenance**

⮕ **How To: Drive Serendipitous Discovery with Recommendation Engines**

Practical Next Steps

It's now time for you to put everything you learned here into practice!

The Machine Learning Basics, Continued: Building Your First Machine Learning Model guidebook will walk you through some of the main considerations when building your first predictive machine learning model, including: defining the goal, preparing the data, building, tuning, interpreting the model.

Meet Katie Gross

Throughout this guidebook, you may have noticed some "In plain English" features and wondered what those were. These side blurbs are extracts from Dataiku Lead Data Scientist Katie Gross' "In plain English" blog series during which she goes over high level machine learning concepts and breaks them down into plain English that everyone can understand.

> You can check out all of these deep dive blogs, as well as a webinar with GigaOm Research featuring Katie Gross **here** to continue your learning journey.

For further exploration

→ **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 learning for people who already have basic Python knowledge. The very regular quizzes and exercises make it particularly interactive.
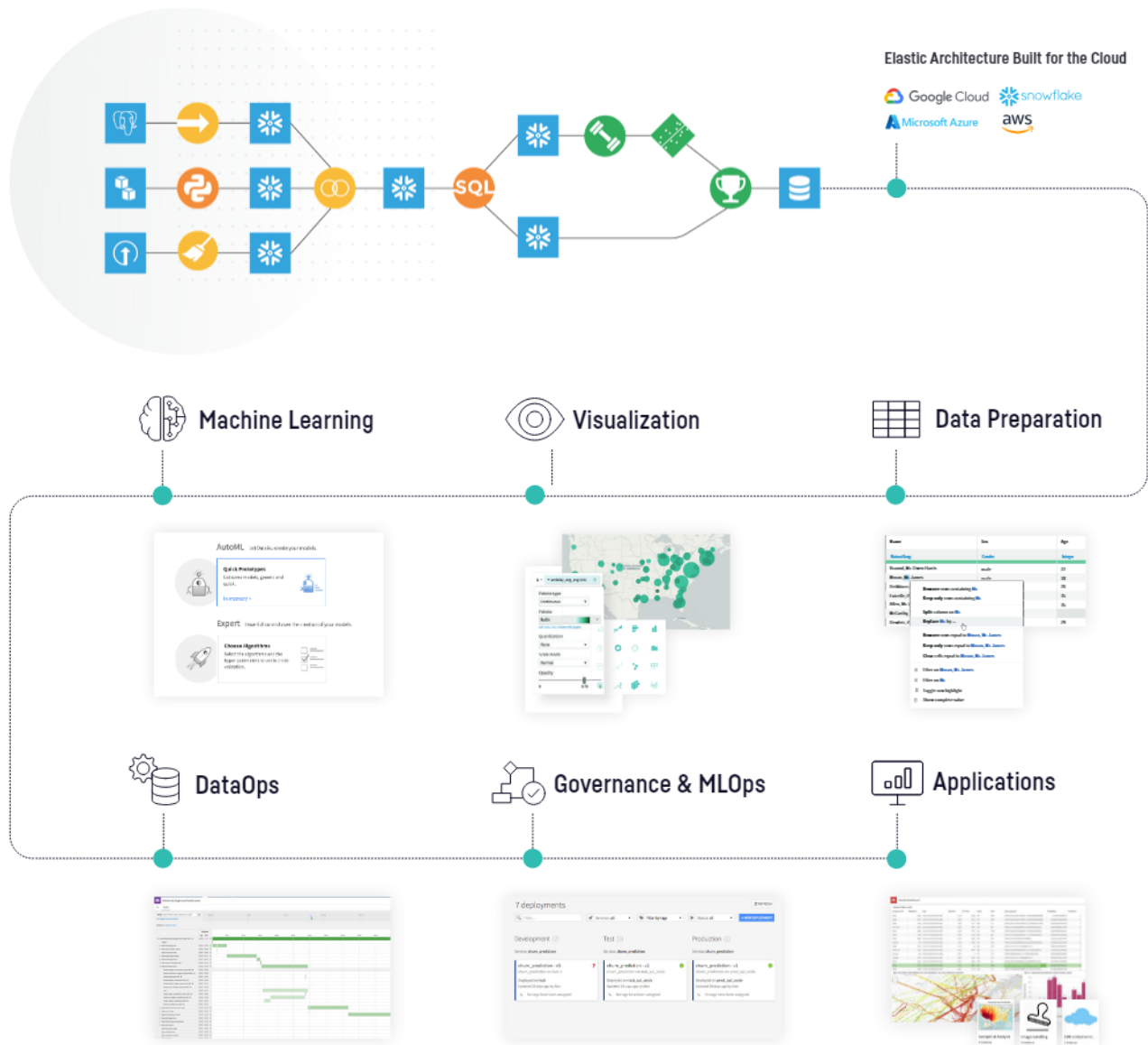
→ **VIDEOS & OTHER**

- **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.

Dataiku Academy

In May 2020, Dataiku launched **Dataiku Academy,** a free online learning tool to enable all users to build their skills around Dataiku. Dataiku Academy is designed for every type of user — from non-technical roles to the hardcore coders — and meant to guide them through self-paced, interactive online training.

Everyday AI, Extraordinary People

Elastic Architecture Built for the Cloud

Google Cloud · snowflake · Microsoft Azure · aws

Machine Learning

Visualization

Data Preparation

DataOps

Governance & MLOps

Applications

45,000+
ACTIVE USERS

450+
CUSTOMERS

Dataiku is the platform for Everyday AI, systemizing the use of data for exceptional business results. Organizations that use Dataiku elevate their people (whether technical and working in code or on the business side and low- or no-code) to extraordinary, arming them with the ability to make better day-to-day decisions with data.