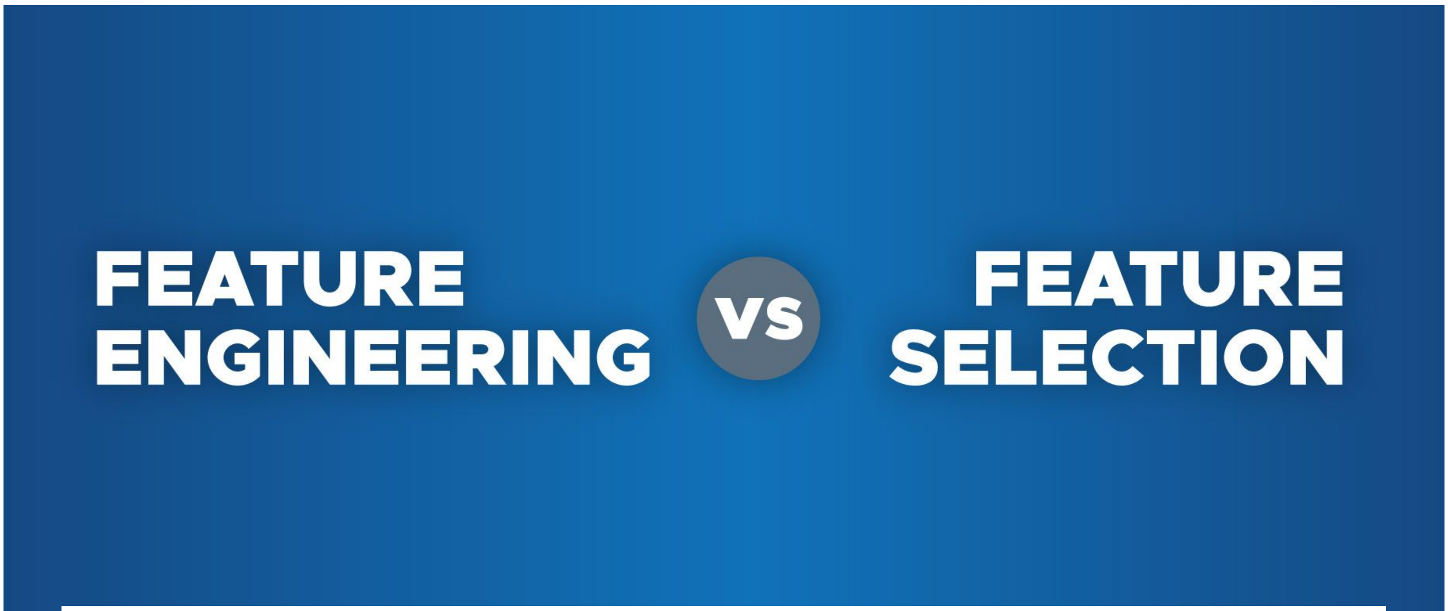
4 JANUARY 2018 / ENGINEERING

Feature Engineering vs Feature Selection

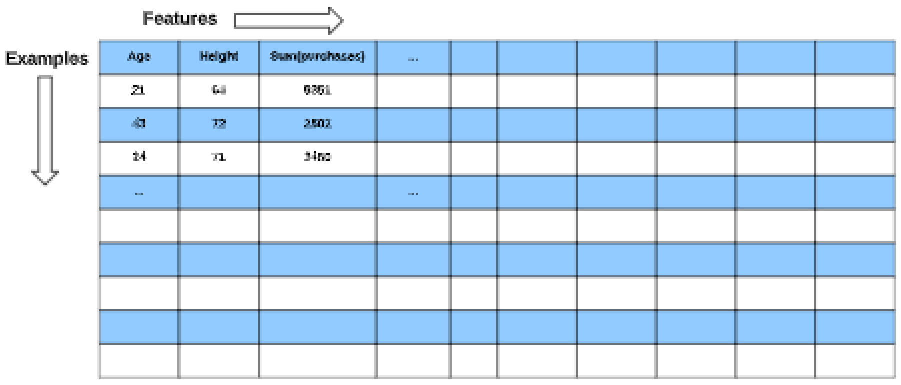


All machine learning workflows depend on feature engineering and feature selection. However, they are often erroneously equated by the data science and machine learning communities. Although they share some overlap, these two ideas have different objectives. Knowing these distinct goals can tremendously improve your data science workflow and pipelines.

This post is intended for those who have done some machine learning before but want to improve their models. First, we'll cover what features and feature matrices are, then we'll walk through the differences between feature engineering and feature selection.

A Brief Overview of Features

Machine learning is the process of generalizing from a set of training data to predict or infer an output. Machine learning algorithms normally take in a collection of numeric examples as input. This process, known as "fitting" or "training," is completed to build a model that the algorithms can use to predict output in the future. These numeric examples are stacked on top of each other, creating a two-dimensional "feature matrix." Each row of this matrix is one "example," and each column represents a "feature."


Features
Examples

Age	Height	Sum(purchases)	...						
21	64	6351							
43	72	2502							
24	71	3460							
...			...						

Feature matrix diagram. Each row represents an example, and each column represents a feature describing that example.

Knowing the role of these features is vital to understanding machine learning. They represent a transformation of the input data to a format that is suitable as input for the algorithms. Sometimes, if the input already contains single numeric values for each example (such as the dollar amount of a credit card transaction), no transformation is needed.

For deep learning in particular, features are usually simple since the algorithms generate their own internal transformations. This approach require large amounts of data and come at the expense of

interpretability. However, these trade-offs are often worthwhile in image processing or natural language processing use cases.

For most other use cases companies face, feature engineering is necessary to convert data into a machine learning-ready format. The choice of features is crucial for both interpretability and performance. Without feature engineering, we wouldn't have the accurate machine learning systems deployed by major companies today.

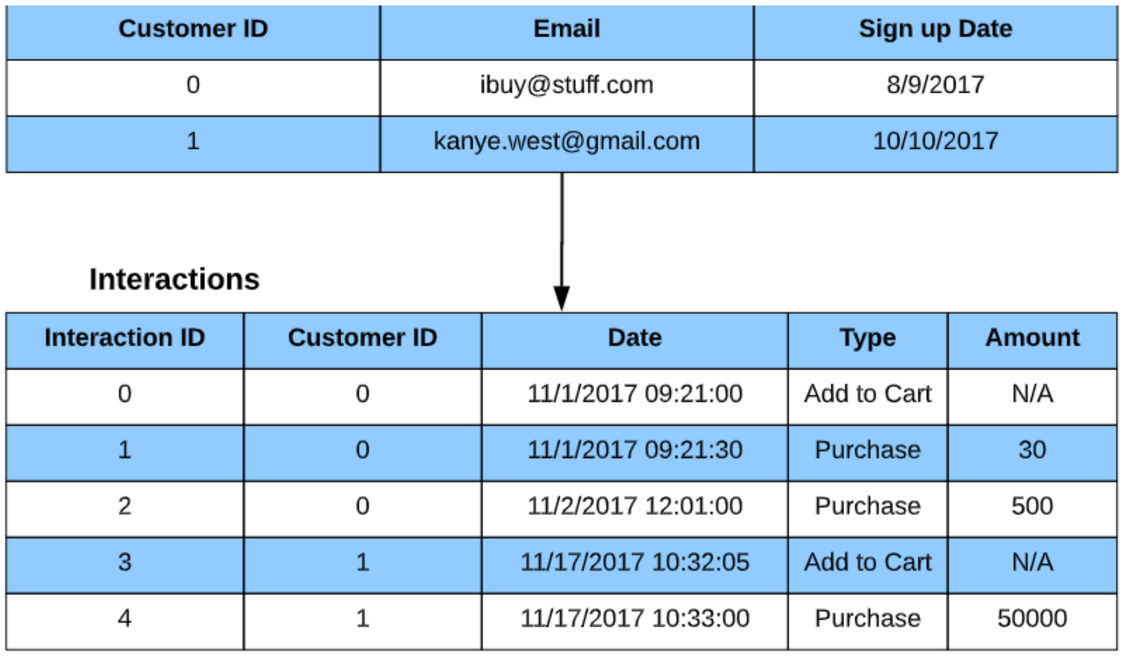
What is Feature Engineering?

Feature engineering is the process of using domain knowledge to extract new variables from raw data that make machine learning algorithms work.

In a typical machine learning use case, data scientists predict quantities using information drawn from their company's data sources. These sources could be various disparate log files or databases. But in general, they contain many tables connected by certain columns. For instance, an ecommerce website's database would have a table called "Customers", containing a single row for every customer that visited the site.

It could also have a table called "Interactions", containing a row for each interaction (click or page visit) that the customer made on the site. This table also contains information about when the interaction took place and the type of event that the interaction represented (is it a "Purchase" event, a "Search" event, or an "Add to Cart" event?). These two tables are related by the "Customer ID" column.

Customers



Customer ID	Email	Sign up Date
0	ibuy@stuff.com	8/9/2017
1	kanye.west@gmail.com	10/10/2017

Interactions

Interaction ID	Customer ID	Date	Type	Amount
0	0	11/1/2017 09:21:00	Add to Cart	N/A
1	0	11/1/2017 09:21:30	Purchase	30
2	0	11/2/2017 12:01:00	Purchase	500
3	1	11/17/2017 10:32:05	Add to Cart	N/A
4	1	11/17/2017 10:33:00	Purchase	50000

This dataset describes interactions with an e-commerce company's website, including the email and sign-up date of each customer. It also includes a log table detailing each interaction each customer made with the site, and the times they occurred.

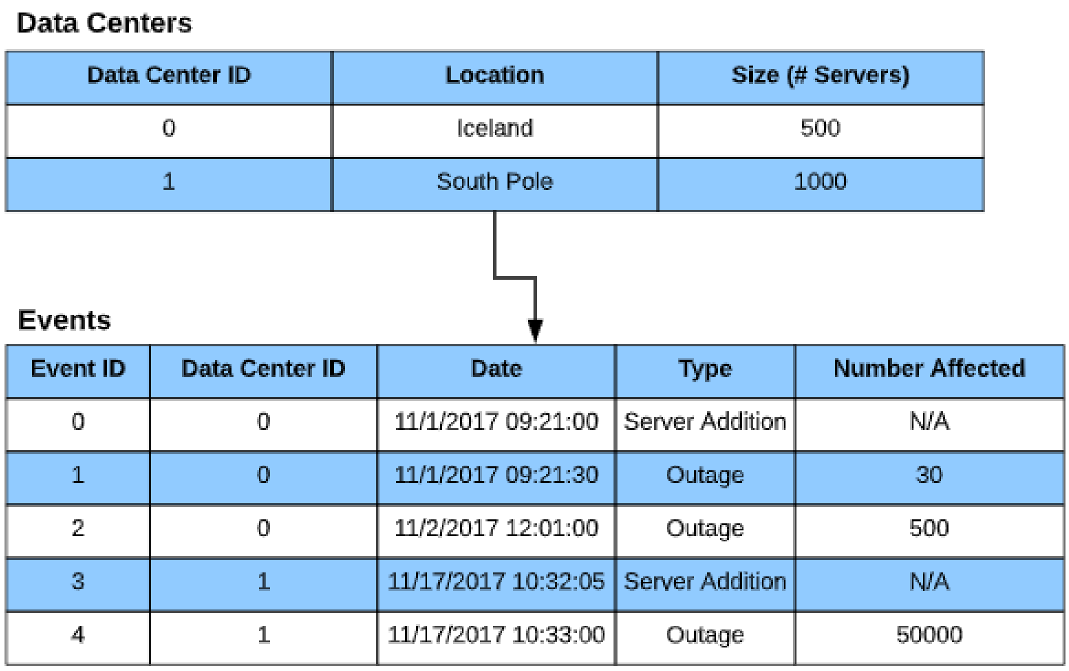
In order to predict when a customer will purchase an item next, we would like a single numeric feature matrix with a row for every customer. We can then use this in a machine learning algorithm. However, the table that looks the most like that (Customers) does not contain much relevant information. We can construct a few features from it, such as the number of days since the customer signed up, but our options are limited at this point.

To improve predictive power, we need to take advantage of the historical data in the Interactions table. Feature engineering makes this possible. We can compute aggregate statistics for each customer by using all values in the Interactions table with that customer's ID. Here are some potentially useful aggregate features about their historical behavior:

- Average time between past purchases
- Average amount of past purchases
- Maximum amount of past purchases

- Time elapsed since last purchase
- Total number of past purchases

To compute all of these features, we would have to find all interactions related to a particular customer. Then we'd filter out the interactions whose "Type" is not "Purchase," and compute a function that returns a single value using the available data. Notice that in general, this process is unique for each use case and dataset. These features are highly specific and wouldn't make much sense for a dataset from a different industry, like one describing network outages.



Data Centers

Data Center ID	Location	Size (# Servers)
0	Iceland	500
1	South Pole	1000

Events

Event ID	Data Center ID	Date	Type	Number Affected
0	0	11/1/2017 09:21:00	Server Addition	N/A
1	0	11/1/2017 09:21:30	Outage	30
2	0	11/2/2017 12:01:00	Outage	500
3	1	11/17/2017 10:32:05	Server Addition	N/A
4	1	11/17/2017 10:33:00	Outage	50000

This dataset describes a company's data centers, including each one's location and size. It also includes a log table detailing events that happened to each data center and the times they occurred.

However, in the network outage dataset, features using similar functions can still be built. We could transform the Location column to be a True/False value that indicates whether the data center is in the Arctic circle. We can also build features by utilizing

aggregation functions similar to the ones used for e-commerce, such as the following:

- Average time between past outages
- Average number of affected servers in past outages
- Maximum number of affected servers in past outages
- Time elapsed since last outage
- Total number of past outages

This type of feature engineering is necessary to effectively use machine learning algorithms and thus build predictive models. Deducing the right set of features to create leads to the biggest gains in performance. This is what data scientists focus on the majority of the time.

The right transformations depend on many factors: the type/structure of the data, the size of the data, and of course, the goals of the data scientist. In practice, these transformations run the gamut: time series aggregations like what we saw above (average of past data points), image filters (blurring an image), and turning text into numbers (using advanced natural language processing that maps words to a vector space) are just a few examples. We developed Featuretools to relieve some of the implementation burden on data scientists and reduce the total time spent on this process through feature engineering automation.

Feature engineering transformations can be “unsupervised.” This means that computing them does not require access to the outputs, or labels, of the problem at hand. They are also usually interpretable. Of the examples mentioned above, the historical aggregations of customer data or network outages are interpretable. An image filter is not, since each feature would represent a pixel of data.

Finally, it is worth noting that formal methods for feature engineering are not as common as those for feature selection. *Feature selection has a long history of formal research, while feature engineering has remained ad hoc and driven by human intuition until only recently.* This is rapidly changing, however — Deep Feature Synthesis, the algorithm behind Featuretools, is a prime example of this.

What Is Feature Selection?

For any given dataset, many possible features can be chosen. A crucial point to consider is which features to use. There are an infinite number of transformations possible. Even if we restrict ourselves to the space of common transformations for a given type of dataset, we are still often left with thousands of possible features.

It would be great if we could plug all of these features in to see which worked. But in reality, the algorithms don't work well when they ingest too many features. This is especially true when the number of features is greater than the number of data points. So how can we solve this?

Enter feature selection.

There are numerous feature selection algorithms that convert a set with too many features into a manageable subset. Similar to feature engineering, different feature selection algorithms are optimal for different types of data. And as always, the goals of the data scientist have to be accounted for as well when choosing the feature selection algorithm. But before all of this, feature engineering should always come first. Without good features, it doesn't matter what you select.

Selecting the most predictive features from a large space is tricky — the more training examples you have, the better you can perform, but the computation time will increase. Several overarching methods exist which fall into one of two categories:

Supervised Selection

This type of method involves examining features in conjunction with a trained model where performance can be computed. Since features are selected based on the model's actual performance, these strategies tend to work well. However, their downside is the exorbitant amount of time they take to run.

A strategy that is guaranteed to work but is prohibitively expensive for all but the smallest feature sets is known as "exhaustive search." Essentially, this strategy attempts every combination of features, checks the performance of each, and chooses the best one. If you have 1,000 features and only want 10, then you'd have to try out 2.6 x 10^23 different combinations. This would be an extremely inefficient use of time. More complex but suboptimal algorithms can run in a reasonable amount of time. These methods have the benefit of being interpretable.

Unsupervised Selection

These methods perform statistical tests on features to determine which are similar or which don't convey much information. Some, like the Variance (or CoVariance) Selector, keep an original subset of features intact, and thus are interpretable. Others, such as Principal Component Analysis (PCA), perform dimensionality reduction and thus produce mostly uninterpretable output.

Fundamental Tools of Data Science

When data scientists want to increase the performance of their

models, feature engineering and feature selection are often the first place they look to improve. Understanding them helps significantly in virtually any data science task you take on.

Feature engineering enables you to build more complex models than you could with only raw data. It also allows you to build interpretable models from any amount of data.

Feature selection will help you limit these features to a manageable number. This will reduce the risk of overwhelming the algorithms or the people tasked with interpreting your model. It also trims down computation time since you won't perform as many data transformations.

If you're just getting started with either feature engineering or feature selection, try to find a simple dataset, build as simple of a model as you can (if using Python, try scikit-learn), and experiment by adding new features.

Featuretools provides a great intro for learning how to build your own features for machine learning. It allows you to see how the addition or removal of features affects the output and thus determine which features are contributing more than others.

Getting a good grasp on what feature engineering and feature selection are can be overwhelming at first, but doing so will impeccably improve your data science skills.

Ben Schreck
Read more posts by this author.

Read More