

# Best Practices for Feature Engineering

July 26, 2017

 [Share](#)

 [Google](#)

 [Linkedin](#)

 [Tweet](#)

Feature engineering, the process creating new input features for machine learning, is one of the most effective ways to improve predictive models.

*Coming up with features is difficult, time-consuming, requires expert knowledge.  
“Applied machine learning” is basically feature engineering. ~ Andrew Ng*

Through feature engineering, you can isolate key information, highlight patterns, and bring in domain expertise.

Unsurprisingly, it can be easy to get stuck because feature engineering is so open-ended.

In this guide, we'll discuss 20 best practices and heuristics that will help you navigate feature engineering.



## Recommended Reading

[Best Practices for Feature Engineering](#)

[The Beginner's Guide to Kaggle](#)

[How to Handle Imbalanced Classes in Machine Learning](#)

[9 Mistakes to Avoid When Starting Your Career in Data Science](#)

[WTF is the Bias-Variance Tradeoff? \(Infographic\)](#)

[Free Data Science Resources for Beginners](#)

[Dimensionality Reduction Algorithms: Strengths and Weaknesses](#)



## Feature Engineering Best Practices

EliteDataScience

### What is Feature Engineering?

Feature engineering is an informal topic, and there are many possible definitions. The machine learning workflow is fluid and iterative, so there's no one "right answer."

We explain our approach in more detail in our [free 7-day email crash course](#).

In a nutshell, we define feature engineering as **creating new features from you existing ones to improve model performance**.

A typical data science process might look like this:

1. Project Scoping / Data Collection
2. Exploratory Analysis
3. Data Cleaning
4. **Feature Engineering**

### Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

**Sign Up Now**

Free 7-Day Crash Course: Data Science & Machine Learning

5. Model Training (including cross-validation to tune hyper-parameters)
6. Project Delivery / Insights

**Free 7-Day Crash Course: Data Science & Machine Learning!**

## What is Not Feature Engineering?

That means there are certain steps we do not consider to be feature engineering:

- We do not consider **initial data collection** to be feature engineering.
- Similarly, we do not consider **creating the target variable** to be feature engineering.
- We do not consider removing duplicates, handling missing values, or fixing mislabeled classes to be feature engineering. We put these under **data cleaning**.
- We do not consider **scaling or normalization** to be feature engineering because these steps belong inside the cross-validation loop (i.e. after you've already built an analytical base table).
- Finally, we do not consider **feature selection or PCA** to be feature engineering. These steps also belong inside your cross-validation loop.

Again, this is simply *our* categorization. Reasonable data scientists may disagree that's perfectly fine.

With those disclaimers out of the way, let's dive into the best practices and heuristics.

## Indicator Variables

The first type of feature engineering involves using indicator variables to isolate important information.

Now, some of you may be wondering, "shouldn't a good algorithm learn the key information on its own?"

## Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

**Sign Up Now**

**Free 7-Day Crash Course: Data Science & Machine Learning!**

Well, not always. It depends on the amount of data you have and the strength of competing signals. You can help your algorithm "focus" on what's important by highlighting it beforehand.

- **Indicator variable from thresholds:** Let's say you're studying alcohol preferences by U.S. consumers and your dataset has an `age` feature. You can create an indicator variable for `age >= 21` to distinguish subjects who were over the legal drinking age.
- **Indicator variable from multiple features:** You're predicting real-estate prices and you have the features `n_bedrooms` and `n_bathrooms`. If houses with 2 beds and 2 baths command a premium as rental properties, you can create an indicator to flag them.
- **Indicator variable for special events:** You're modeling weekly sales for an commerce site. You can create two indicator variables for the weeks of Black and Christmas.
- **Indicator variable for groups of classes:** You're analyzing website conversion and your dataset has the categorical feature `traffic_source`. You could create an indicator variable for `paid_traffic` by flagging observations with traffic source values of `"Facebook Ads"` or `"Google Adwords"`.

## Interaction Features

The next type of feature engineering involves highlighting interactions between two or more features.

Have you ever heard the phrase, "the sum is greater than the parts?" Well, some features can be combined to provide more information than they would as individuals.

Specifically, look for opportunities to take the sum, difference, product, or quotient of multiple features.

Free 7-Day Crash Course: Data Science & Machine Learning!

## Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

[Sign Up Now](#)

Free 7-Day Crash Course: Data Science & Machine Learning!

*\*Note: We don't recommend using an automated loop to create interactions for a lot of features. This leads to "feature explosion."*

- **Sum of two features:** Let's say you wish to predict revenue based on preliminary sales data. You have the features `sales_blue_pens` and `sales_black_pens`. You could sum those features if you only care about overall `sales_pens`.
- **Difference between two features:** You have the features `house_built_date` and `house_purchase_date`. You can take their difference to create the feature `house_age_at_purchase`.
- **Product of two features:** You're running a pricing test, and you have the feature `price` and an indicator variable `conversion`. You can take their product to create the feature `earnings`.
- **Quotient of two features:** You have a dataset of marketing campaigns with features `n_clicks` and `n_impressions`. You can divide clicks by impression to create `click_through_rate`, allowing you to compare across campaigns of different volume.

## Feature Representation

This next type of feature engineering is simple yet impactful. It's called feature representation.

Your data won't always come in the ideal format. You should consider if you'd gain more information by representing the same feature in a different way.

- **Date and time features:** Let's say you have the feature `purchase_datetime`. It might be more useful to extract `purchase_day_of_week` and `purchase_hour_of_day`. You can also aggregate observations to create features such as `purchases_over_last_30_days`.

Free 7-Day Crash Course: Data Science & Machine Learning!

## Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

Sign Up Now

Free 7-Day Crash Course: Data Science & Machine Learning!

Free 7-Day Crash Course: Data Science & Machine Learning!

- **Numeric to categorical mappings:** You have the feature `years_in_school` might create a new feature `grade` with classes such as `"Elementary School"`, `"Middle School"`, and `"High School"`.
- **Grouping sparse classes:** You have a feature with many classes that have low sample counts. You can try grouping similar classes and then grouping the remaining ones into a single `"Other"` class.
- **Creating dummy variables:** Depending on your machine learning implementation, you may need to manually transform categorical features into dummy variables. You should always do this *after* grouping sparse classes.

# External Data

An underused type of feature engineering is bringing in external data. This can lead to some of the biggest breakthroughs in performance.

For example, one way quantitative hedge funds perform research is by layering different streams of financial data.

Many machine learning problems can benefit from bringing in external data. Here are some examples:

- **Time series data:** The nice thing about time series data is that you only need one feature, some form of `date`, to layer in features from another dataset.
- **External API's:** There are plenty of API's that can help you create features. For example, the [Microsoft Computer Vision](#) API can return the number of faces in an image.
- **Geocoding:** Let's say you have `street_address`, `city`, and `state`. Well, you can [geocode](#) them into `latitude` and `longitude`. This will allow you to calculate features such as local demographics (e.g. `median_income_within_2_miles`) with the help of [another dataset](#).

# Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

First Name

Email Address

Sign Up Now

Free 7-Day Crash Course: Data Science & Machine Learning!

- **Other sources of the same data:** How many ways could you track a Facebook campaign? You might have Facebook's own tracking pixel, Google Analytics possibly another third-party software. Each source can provide information that the others don't track. Plus, any differences between the datasets could be informative (e.g. bot traffic that one source ignores while another source keeps).

## Error Analysis (Post-Modeling)

The final type of feature engineering we'll cover falls under a process called error analysis. This is performed *after* training your first model.

Error analysis is a broad term that refers to analyzing the misclassified or high error observations from your model and deciding on your next steps for improvement.

Possible next steps include collecting more data, splitting the problem apart, or engineering new features that address the errors. To use error analysis for feature engineering, you'll need to understand *why* your model missed its mark.

Here's how:

- **Start with larger errors:** Error analysis is typically a manual process. You will have to scrutinize every observation. We recommend starting with those that have higher error scores. Look for patterns that you can formalize into new features.
- **Segment by classes:** Another technique is to segment your observations and compare the average error within each segment. You can try creating indicator variables for the segments with the highest errors.
- **Unsupervised clustering:** If you have trouble spotting patterns, you can run an unsupervised clustering algorithm on the misclassified observations. We don't recommend blindly using those clusters as a new feature, but they can make it easier

**Free 7-Day Crash Course: Data Science & Machine Learning!**

## Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

**Sign Up Now**

**Free 7-Day Crash Course: Data Science & Machine Learning!**



to spot patterns. Remember, the goal is to understand *why* observations were misclassified.

- **Ask colleagues or domain experts:** This is a great complement to any of the other three techniques. Asking a domain expert is especially useful if you've identified a pattern of poor performance (e.g. through segmentations) but don't yet understand why.

## Conclusion

As you see, there are many possibilities for feature engineering. We've covered 20 best practices and heuristics, but they are by no means exhaustive!

Remember these general guidelines as you start to experiment on your own:

### Good features to engineer...

- Can be computed for future observations.
- Are usually intuitive to explain.
- Are informed by domain knowledge or exploratory analysis.
- Must have the potential to be predictive. Don't just create features for the sake of it.
- **Never touch the target variable.** This is a trap that beginners sometimes fall into. Whether you're creating indicator variables or interaction features, never use the target variable. That's like "cheating" and it would give you very misleading results.

**Finally, don't worry if this feels overwhelming right now!** You'll naturally get better at feature engineering through practice and experience.

First, if this is your first exposure to some of these tactics, we highly recommend trying out a dataset and solidifying what you've learned. Here are some more resources that can help you in your journey:



**Free 7-Day Crash Course: Data Science & Machine Learning!**

## Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

**Sign Up Now**

**Free 7-Day Crash Course: Data Science & Machine Learning!**





Fun Machine Learning Projects for Beginners



7-Day Email Crash Course on Applied Machine Learning



Machine Learning Masterclass - Learn by completing end-to-end projects



Have any questions about feature engineering? Did we miss one of your favorite heuristics? Let us know in the comments!



Share



Google



Linkedin



Tweet

## Leave A Response

Name\*

Email\*

Website

Submit Comment

\* Denotes Required Field

Free 7-Day Crash Course: Data Science & Machine Learning!

## Free 7-Day Crash Course: Data Science & Machine Learning!

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

First Name

Email Address

Sign Up Now



Free 7-Day Crash Course: Data Science

## **Free 7-Day Crash Course: Data Science & Machine Learning!**

Jumpstart your data science and machine learning journey with our practical, super intuitive course.

**Sign Up Now**