Discover Feature Engineering, How to Engineer Features and How to Get Good at It

by **Jason Brownlee** on September 26, 2014 in **Data Preparation**

Last Updated on August 15, 2020

Feature engineering is an informal topic, but one that is absolutely known and agreed to be key to success in applied machine learning.

In creating this guide I went wide and deep and synthesized all of the material I could.

You will discover what feature engineering is, what problem it solves, why it matters, how to engineer features, who is doing it well and where you can go to learn more and get good at it.

If you read one article on feature engineering, I want it to be this one.

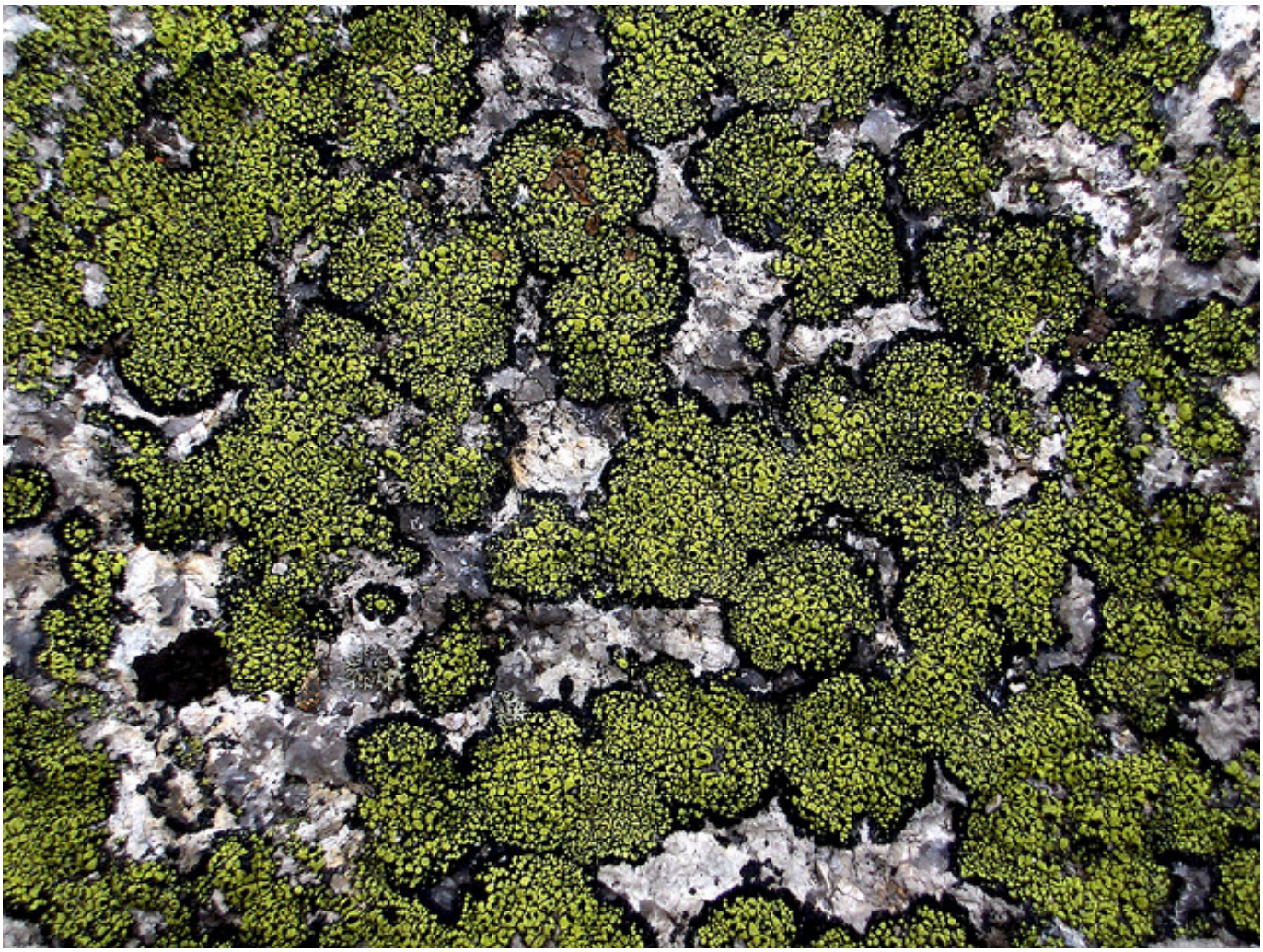
> *feature engineering is another topic which doesn't seem to merit any review papers or books, or even chapters in books, but it is absolutely vital to ML success. […] Much of the success of machine learning is actually success in engineering features that a learner can understand.*

— Scott Locklin, in "Neglected machine learning ideas"

Kick-start your project with my new book Data Preparation for Machine Learning, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

Problem that Feature Engineering Solves



Feature engineering is hard.
Photo by Vik Nanda, some rights reserved

When your goal is to get the best possible results from a predictive model, you need to get the most from what you have.

This includes getting the best results from the algorithms you are using. It also involves getting the most out of the data for your algorithms to work with.

How do you get the most out of your data for predictive modeling?

This is the problem that the process and practice of feature engineering solves.

> *Actually the success of all Machine Learning algorithms depends on how you present the data.*

— Mohammad Pezeshki, answer to "What are some general tips on feature selection and engineering

that every data scientist should know?”

Importance of Feature Engineering

The features in your data will directly influence the predictive models you use and the results you can achieve.

You can say that: the better the features that you prepare and choose, the better the results you will achieve. It is true, but it also misleading.

The results you achieve are a factor of the model you choose, the data you have available and the features you prepared. Even your framing of the problem and objective measures you’re using to estimate accuracy play a part. Your results are dependent on many inter-dependent properties.

You need great features that describe the structures inherent in your data.

Better features means flexibility.

You can choose “the wrong models” (less than optimal) and still get good results. Most models can pick up on good structure in data. The flexibility of good features will allow you to use less complex models that are faster to run, easier to understand and easier to maintain. This is very desirable.

Better features means simpler models.

With well engineered features, you can choose “the wrong parameters” (less than optimal) and still get good results, for much the same reasons. You do not need to work as hard to pick the right models and the most optimized parameters.

With good features, you are closer to the underlying problem and a representation of all the data you have available and could use to best characterize that underlying problem.

Better features means better results.

> *The algorithms we used are very standard for Kagglers. […] We spent most of our efforts in feature engineering.*

— Xavier Conort, on "Q&A with Xavier Conort" on winning the Flight Quest challenge on Kaggle

What is Feature Engineering?

Here is how I define feature engineering:

> *Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.*

You can see the dependencies in this definition:

- The performance measures you've chosen (RMSE? AUC?)
- The framing of the problem (classification? regression?)
- The predictive models you're using (SVM?)
- The raw data you have selected and prepared (samples? formatting? cleaning?)

> *feature engineering is manually designing what the input x's should be*

— Tomasz Malisiewicz, answer to "What is feature engineering?"

Feature Engineering is a Representation Problem

Machine learning algorithms learn a solution to a problem from sample data.

In this context, feature engineering asks: what is the best representation of the sample data to learn a solution to your problem?

It's deep. Doing well in machine learning, even in artificial intelligence in general comes back to representation problems. It's hard stuff, perhaps unknowable (or at best intractable) to know the best representation to use, *a priori*.

> *you have to turn your inputs into things the algorithm can understand*

— Shayne Miel, answer to "What is the intuitive explanation of feature engineering in machine

learning?"

Feature Engineering is an Art

It is an art like engineering is an art, like programming is an art, like medicine is an art.

There are well defined procedures that are methodical, provable and understood.

The data is a variable and is different every time. You get good at deciding which procedures to use and when, by practice. By empirical apprenticeship. Like engineering, like programming, like medicine, like machine learning in general.

Mastery of feature engineering comes with hands on practice, and study of what others that are doing well are practicing.

> *…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.*

— Pedro Domingos, in "A Few Useful Things to Know about Machine Learning" (PDF)

Sub-Problems of Feature Engineering

It is common to think of feature engineering as one thing.

For example, for a long time for me, feature engineering was feature construction.

I would think to myself "*I'm doing feature engineering now*" and I would pursue the question "*How can I decompose or aggregate raw data to better describe the underlying problem?*" The goal was right, but the approach was one of many.

In this section we look at these many approaches and the specific sub-problems that they are intended to address. Each could be an in depth article of their own as they are large and important areas of practice and study.

Feature: An attribute useful for your modeling task

Let's start with data and what is a feature.

Tabular data is described in terms of observations or instances (rows) that are made up of variables or attributes (columns). An attribute could be a feature.

The idea of a feature, separate from an attribute, makes more sense in the context of a problem. A feature is an attribute that is useful or meaningful to your problem. It is an important part of an

observation for learning about the structure of the problem that is being modeled.

I use "*meaningful*" to discriminate attributes from features. Some might not. I think there is no such thing as a non-meaningful feature. If a feature has no impact on the problem, it is not part of the problem.

In computer vision, an image is an observation, but a feature could be a line in the image. In natural language processing, a document or a tweet could be an observation, and a phrase or word count could be a feature. In speech recognition, an utterance could be an observation, but a feature might be a single word or phoneme.

Feature Importance: An estimate of the usefulness of a feature

You can objectively estimate the usefulness of features.

This can be helpful as a pre-cursor to selecting features. Features are allocated scores and can then be ranked by their scores. Those features with the highest scores can be selected for inclusion in the training dataset, whereas those remaining can be ignored.

Feature importance scores can also provide you with information that you can use to extract or construct new features, similar but different to those that have been estimated to be useful.

A feature may be important if it is highly correlated with the dependent variable (the thing being predicted). Correlation coefficients and other univariate (each attribute is considered independently) methods are common methods.

More complex predictive modeling algorithms perform feature importance and selection internally while constructing their model. Some examples include MARS, Random Forest and Gradient Boosted Machines. These models can also report on the variable importance determined during the model preparation process.

Feature Extraction: The automatic construction of new features from raw data

Some observations are far too voluminous in their raw state to be modeled by predictive modeling algorithms directly.

Common examples include image, audio, and textual data, but could just as easily include tabular data with millions of attributes.

Feature extraction is a process of automatically reducing the dimensionality of these types of observations into a much smaller set that can be modelled.

For tabular data, this might include projection methods like Principal Component Analysis and unsupervised clustering methods. For image data, this might include line or edge detection. Depending on the domain, image, video and audio observations lend themselves to many of the same types of DSP methods.

Key to feature extraction is that the methods are automatic (although may need to be designed and constructed from simpler methods) and solve the problem of unmanageably high dimensional data, most typically used for analog observations stored in digital formats.

Feature Selection: From many features to a few that are useful

Not all features are created equal.

Those attributes that are irrelevant to the problem need to be removed. There will be some features that will be more important than others to the model accuracy. There will also be features that will be redundant in the context of other features.

Feature selection addresses these problems by automatically selecting a subset that are most useful to the problem.

Feature selection algorithms may use a scoring method to rank and choose features, such as correlation or other feature importance methods.

More advanced methods may search subsets of features by trial and error, creating and evaluating models automatically in pursuit of the objectively most predictive sub-group of features.

There are also methods that bake in feature selection or get it as a side effect of the model. Stepwise regression is an example of an algorithm that automatically performs feature selection as part of the model construction process.

Regularization methods like LASSO and ridge regression may also be considered algorithms with feature selection baked in, as they actively seek to remove or discount the contribution of features as part of the model building process.

Read more in the post: An Introduction to Feature Selection.

Feature Construction: The manual construction of new features from raw data

The best results come down to you, the practitioner, crafting the features.

Feature importance and selection can inform you about the objective utility of features, but those features have to come from somewhere.

You need to manually create them. This requires spending a lot of time with actual sample data (not aggregates) and thinking about the underlying form of the problem, structures in the data and how best to expose them to predictive modeling algorithms.

With tabular data, it often means a mixture of aggregating or combining features to create new features, and decomposing or splitting features to create new features.

With textual data, it often means devising document or context specific indicators relevant to the problem. With image data, it can often mean enormous amounts of time prescribing automatic filters to pick out relevant structures.

This is the part of feature engineering that is often talked the most about as an artform, the part that is attributed the importance and signalled as the differentiator in competitive machine learning.

It is manual, it is slow, it requires lots of human brain power, and it makes a big difference.

> *Feature engineering and feature selection are not mutually exclusive. They are both useful. I'd say feature engineering is more important though, especially because you can't really automate it.*

— Robert Neuhaus, answer to "Which do you think improves accuracy more, feature selection or feature engineering?"

Feature Learning: The automatic identification and use of features in raw data

Can we avoid the manual load of prescribing how to construct or extract features from raw data?

Representation learning or feature learning is an effort towards this goal.

Modern deep learning methods are achieving some success in this area, such as autoencoders and restricted Boltzmann machines. They have been shown to automatically and in a unsupervised or semi-supervised way, learn abstract representations of features (a compressed form), that in turn have supported state-of-the-art results in domains such as speech recognition, image classification, object recognition and other areas.

We do not have automatic feature extraction or construction, yet, and we will probably never have automatic feature engineering.

The abstract representations are prepared automatically, but you cannot understand and leverage what has been learned, other than in a black-box manner. They cannot (yet, or easily) inform you and the process on how to create more similar and different features like those that are doing well, on a given

problem or on similar problems in the future. The acquired skill is trapped.

Nevertheless, it's fascinating, exciting and an important and modern part of feature engineering.

Process of Feature Engineering

Feature engineering is best understood in the broader process of applied machine learning.

You need this context.

Process of Machine Learning

The process of applied machine learning (for lack of a better name) that in a broad brush sense involves lots of activities. Up front is problem definition, next is data selection and preparation, in the middle is model preparation, evaluation and tuning and at the end is the presentation of results.

Process descriptions like data mining and KDD help to better understand the tasks and subtasks. You can pick and choose and phrase the process the way you like. I've talked a lot about this before.

A picture relevant to our discussion on feature engineering is the front-middle of this process. It might look something like the following:

1. (tasks before here…)
2. **Select Data**: Integrate data, de-normalize it into a dataset, collect it together.
3. **Preprocess Data**: Format it, clean it, sample it so you can work with it.
4. **Transform Data**: *Feature Engineer happens here*.
5. **Model Data**: Create models, evaluate them and tune them.
6. (tasks after here…)

The traditional idea of "*Transforming Data*" from a raw state to a state suitable for modeling is where feature engineering fits in. Transform data and feature engineering may in fact be synonyms.

This picture helps in a few ways.

You can see that before feature engineering, we are munging out data into a format we can even look at, and just before that we are collating and denormalizing data from databases into some kind of central picture.

We can, and should go back through these steps as we identify new perspectives on the data.

For example, we may have an attribute that is an aggregate field, like a sum. Rather than a single sum, we may decide to create features to describe the quantity by time interval, such as season. We need to step backward in the process through Preprocessing and even Selecting data to get access to the "real

raw data" and create this feature.

We can see that feature engineering is followed by modeling.

It suggests a strong interaction with modeling, reminding us of the interplay of devising features and testing them against the coalface of our test harness and final performance measures.

This also suggests we may need to leave the data in a form suitable for the chosen modeling algorithm, such as normalize or standardize the features as a final step. This sounds like a preprocessing step, it probably is, but it helps us consider what types of finishing touches are needed to the data before effective modeling.

Iterative Process of Feature Engineering

Knowing where feature engineering fits into the context of the process of applied machine learning highlights that it does not standalone.

It is an iterative process that interplays with data selection and model evaluation, again and again, until we run out of time on our problem.

The process might look as follows:

1. **Brainstorm features**: Really get into the problem, look at a lot of data, study feature engineering on other problems and see what you can steal.
2. **Devise features**: Depends on your problem, but you may use automatic feature extraction, manual feature construction and mixtures of the two.
3. **Select features**: Use different feature importance scorings and feature selection methods to prepare one or more "views" for your models to operate upon.
4. **Evaluate models**: Estimate model accuracy on unseen data using the chosen features.

You need a well defined problem so that you know when to stop this process and move on to trying other models, other model configurations, ensembles of models, and so on. There are gains to be had later in the pipeline once you plateau on ideas or the accuracy delta.

You need a well considered and designed test harness for objectively estimating model skill on unseen data. It will be the only measure you have of your feature engineering process, and you must trust it not to waste your time.

General Examples of Feature Engineering

Let's make the concepts of feature engineering more concrete.

In this section we will consider tabular data like that you might have in an excel spreadsheet. We will

look at some examples of manual feature construction that you might like to consider on your own problems.

When I hear "*feature engineering is critically important*", this is the type of feature engineering I think of. It is the most common form that I am familiar with and practice.

Which of these is best? You cannot know before hand. You must try them and evaluate the results to achieve on your algorithm and performance measures.

Decompose Categorical Attributes

Imagine you have a categorical attribute, like "*Item_Color*" that can be *Red*, *Blue* or *Unknown*.

Unknown may be special, but to a model, it looks like just another colour choice. It might be beneficial to better expose this information.

You could create a new binary feature called "*Has_Color*" and assign it a value of "*1*" when an item has a color and "*0*" when the color is unknown.

Going a step further, you could create a binary feature for each value that *Item_Color* has. This would be three binary attributes: *Is_Red*, *Is_Blue* and *Is_Unknown*.

These additional features could be used instead of the *Item_Color* feature (if you wanted to try a simpler linear model) or in addition to it (if you wanted to get more out of something like a decision tree).

Decompose a Date-Time

A date-time contains a lot of information that can be difficult for a model to take advantage of in it's native form, such as ISO 8601 (i.e. 2014-09-20T20:45:40Z).

If you suspect there are relationships between times and other attributes, you can decompose a date-time into constituent parts that may allow models to discover and exploit these relationships.

For example, you may suspect that there is a relationship between the time of day and other attributes.

You could create a new numerical feature called *Hour_of_Day* for the hour that might help a regression model.

You could create a new ordinal feature called *Part_Of_Day* with 4 values *Morning*, *Midday*, *Afternoon*, *Night* with whatever hour boundaries you think are relevant. This might be useful for a decision tree.

You can use similar approaches to pick out time of week relationships, time of month relationships and various structures of seasonality across a year.

Date-times are rich in structure and if you suspect there is time dependence in your data, take your time and tease them out.

Reframe Numerical Quantities

Your data is very likely to contain quantities, which can be reframed to better expose relevant structures. This may be a transform into a new unit or the decomposition of a rate into time and amount components.

You may have a quantity like a weight, distance or timing. A linear transform may be useful to regression and other scale dependent methods.

For example, you may have *Item_Weight* in grams, with a value like 6289. You could create a new feature with this quantity in kilograms as 6.289 or rounded kilograms like 6. If the domain is shipping data, perhaps kilograms is sufficient or more useful (less noisy) a precision for *Item_Weight*.

The *Item_Weight* could be split into two features: *Item_Weight_Kilograms* and *Item_Weight_Remainder_Grams*, with example values of 6 and 289 respectively.

There may be domain knowledge that items with a weight above 4 incur a higher taxation rate. That magic domain number could be used to create a new binary feature *Item_Above_4kg* with a value of "*1*" for our example of 6289 grams.

You may also have a quantity stored as a rate or an aggregate quantity for an interval. For example, *Num_Customer_Purchases* aggregated over a year.

In this case you may want to go back to the data collection step and create new features in addition to this aggregate and try to expose more temporal structure in the purchases, like perhaps seasonality. For example, the following new binary features could be created: *Purchases_Summer*, *Purchases_Fall*, *Purchases_Winter* and *Purchases_Spring*.

Concrete Examples of Feature Engineering

A great place to study examples of feature engineering is in the results from competitive machine learning.

Competitions typically use data from a real-world problem domain. A write-up of methods and approach is required at the end of a competition. These write-ups give valuable insight into effective real-world machine learning processes and methods.

In this section we touch on a few examples of interesting and notable post-competition write-ups that focus on feature engineering.

Predicting Student Test Performance in KDD Cup 2010

The KDD Cup is a machine learning competition held for attendees of the ACM Special Interest Group on Knowledge Discovery and Data Mining conferences, each year.

In 2010, the focus of the competition was the problem of modeling how students learn. A corpus of student results on algebraic problems was provided to be used to predict those students' future performance.

The winner of the competition were a group of students and academics at the National Taiwan University. Their approach is described in the paper "Feature Engineering and Classifier Ensemble for KDD Cup 2010".

The paper credits feature engineering as a key method in winning. Feature engineering simplified the structure of the problem at the expense of creating millions of binary features. The simple structure allowed the team to use highly performant but very simple linear methods to achieve the winning predictive model.

The paper provides details of how specific temporal and other non-linearities in the problem structure were reduced to simple composite binary indicators.

This is an extreme and instructive example of what is possible with simple attribute decomposition.

Predicting Patient Admittance in the Heritage Health Prize

The heritage health prize was a 3 million dollar prize awarded to the team who could best predict which patients would be admitted to hospital within the next year.

The prize had milestone awards each year where the top teams would be awarded a prize and their processes and methods made public.

I remember reading the papers released at the first of the three milestones and being impressed with the amount of feature engineering involved.

Specifically, the paper "Round 1 Milestone Prize: How We Did It – Team Market Makers" by Phil Brierley, David Vogel and Randy Axelrod. Most competitions involve vast amounts of feature engineering, but it struck me how clearly this paper made the point.

The paper provides both tables of attributes and SQL required to construct the attributes.

The paper gives some great real-world examples of feature engineering by simple decomposition. There are a lot of counts, mins, maxes, lots of binary attributes, and discretized numerical attributes. Very simple methods used to great effect.

More Resources on Feature Engineering

We have covered a lot of ground in this article and I hope you have a much greater appreciation of what feature engineering is, where it fits in, and how to do it.

This is really the start of your journey. You need to practice feature engineering and you need to study great practitioners of feature engineering.

This section provides some resources that might help you on your journey.

Books

I cannot find any books or book chapters on the topic.

There are however some great books on feature extraction. If you are working with digital representations of analog observations like images, video, sound or text, you might like to dive deeper into some feature extraction literature.

- Feature Extraction, Construction and Selection: A Data Mining Perspective
- Feature Extraction: Foundations and Applications
- Feature Extraction & Image Processing for Computer Vision

There are also lots of books on feature selection. If you are working to reduce your features by removing those that are redundant or irrelevant, dive deeper into feature selection.

- Feature Selection for Knowledge Discovery and Data Mining
- Computational Methods of Feature Selection

Papers and Slides

It is a hard topic to find papers on.

Again, there are plenty of papers of feature extraction and chapters in books of feature selection, but not much of feature engineering. Also feature engineering has a meaning in software engineering as well, one that is not relevant to our discussion.

Here are some generally relevant papers:

- JMLR Special Issue on Variable and Feature Selection

Here are some generally relevant and interesting slides:

- Feature Engineering (PDF), Knowledge Discover and Data Mining 1, by Roman Kern, Knowledge Technologies Institute

- Feature Engineering and Selection (PDF), CS 294: Practical Machine Learning, Berkeley
- Feature Engineering Studio, Course Lecture Slides and Materials, Columbia
- Feature Engineering (PDF), Leon Bottou, Princeton

Links

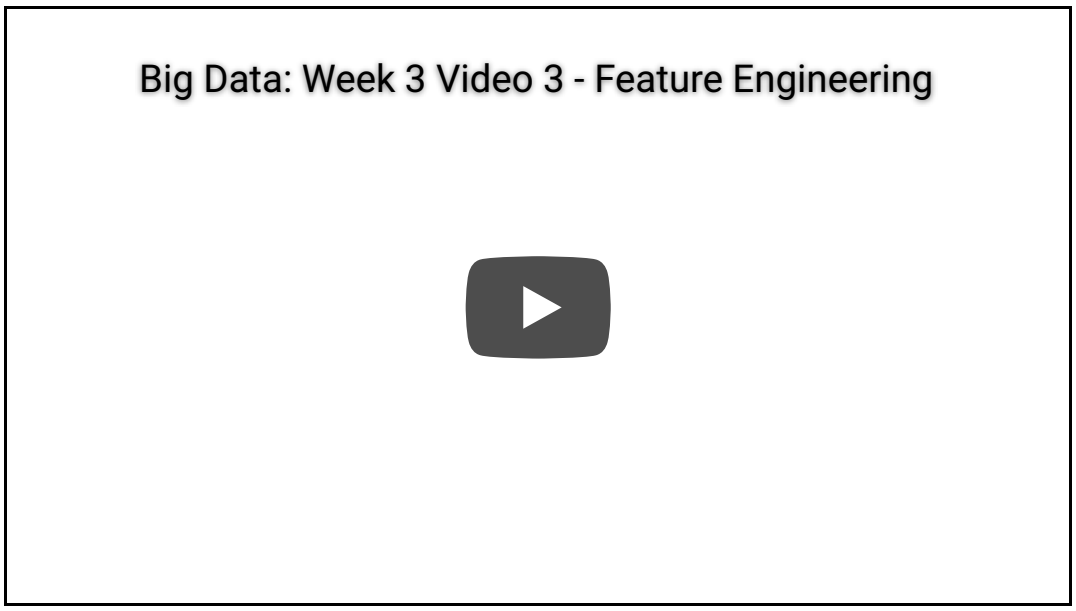
There blog posts here and there. The most useful links are tutorials that work through a problem and clearly articulate the intentional feature engineering.

Below are some generally interesting links:

- Feature Engineering: How to perform feature engineering on the Titanic competition (a getting started competition on Kaggle). There is more data munging than feature engineering, but it's still instructive.
- ~~IPython Notebook by Guibing Guo, dedicated to explaining feature engineering. A bit messy, but worth a skim~~. (link appears broken, sorry.)

Videos

There are a few videos on the topic of feature engineering. The best by far is titled "Feature Engineering" by Ryan Baker. It's short (9 minutes or so) and I recommend watching it for some good practical tips.



If you think I missed a key concept or resource, please leave a comment.

Update 2015: I notice that there is now a Wikipedia article on Feature Engineering and it copies large chunks of this post. Oh well.

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.

View all posts by Jason Brownlee →

‹ Compare Models And Select The Best Using The Caret R Package

A Data-Driven Approach to Choosing Machine Learning Algorithms ›

119 Responses to *Discover Feature Engineering, How to Engineer Features and How to Get Good at It*

Sam September 26, 2014 at 8:03 am #

REPLY

Chapter 10 of this book : http://books.google.co.uk/books?id=Ofp4h_oXsZ4C&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false has

a nice section on features.

Also I think you can sometimes think of ensemble methods as feature learning.

jasonb September 26, 2014 at 8:09 am #

REPLY

"Machine Learning: The Art and Science of Algorithms that Make Sense of Data" is a great book, thanks for the tip Sam.

I totally agree. Things get messy when you take an objective look at regularization and even ensemble methods as you suggestion. I think that is why we tend to wards a vague name like "feature engineering" rather than a specific subtopic when talking about this stuff in practice.

Anurag September 26, 2014 at 4:21 pm #

REPLY

As important as it sounds, I rarely hear about Feature Engineering as a term even in Practical Machine Learning. Most of the things covered here are already taken care of in regularization, or other phases as well.
Nevertheless, I vouch for the distinction made here and I think that Feature Engineering reserves a place not only in Practical Machine Learning but also in theory.

jasonb September 27, 2014 at 6:26 am #

REPLY

Nice point Anurag.

Akash Deep Singh September 27, 2014 at 8:37 pm #

REPLY

An excellent, well woven article with a great insight about Feature Extraction and Analysis. I love the article for its 'no-math' approach yet forming a brief coherent description about such a vast topic.
The list of resources are good too. They are on my 'next to read list' after I finish the machine learning books.
Thank you Jason!

jasonb September 28, 2014 at 5:30 am #

REPLY

Thanks Akash, let me know how you go with the resources.

I found the book “Feature Extraction: Foundations and Applications” fascinating, hard to stop reading.

Damian Mingle September 28, 2014 at 1:23 am #

REPLY

Nothing better than a Saturday morning reading Jason’s work-up on feature engineering. It has always baffled me why many individuals agree that this is such an important topic, but how there is very little formal information on this topic.

However, this write-up has been saved by me. It has a variety of sources on the topic and gives some really applicable steps to foster a practitioners intuition.

Great job…

jasonb September 28, 2014 at 5:31 am #

REPLY

Thanks Damian!

Roselyn Isimeto September 29, 2014 at 5:58 am #

REPLY

Many thanks Jason, for all your teachings on ML.

jasonb September 29, 2014 at 7:25 am #

REPLY

You’re welcome Roselyn.

Udi October 7, 2014 at 5:50 am #

REPLY

Great article (as always)

One question, could you,please explain moe about the following:

“These additional features could be used instead of the Item_Color feature (if you wanted to try a simpler linear model) or in addition to it (if you wanted to get more out of something like a decision tree).”

Thanks

Udi

jasonb October 7, 2014 at 7:05 am #

REPLY

Thanks Udi!
That line is suggesting that you can use the new features like “Has_Color” and “Is_Red” in your data when training your model in addition to or instead of the “Item_Color” attribute. Experimentation is required with your specific dataset and choice of models to see what gives the best results.

Udi October 10, 2014 at 7:47 pm #

REPLY

Thanx Jason, however I’m interested in understanding why a decision tree will benifet from the additional feature, while simpler linear model won’t.

Thanks a lot
Udi

Cemal February 19, 2015 at 2:54 am #

REPLY

Hi Jason,

I have the same question as Udi “…why a decision tree will benifet from the additional feature, while simpler linear model won’t”

You did not answer this question.

Thanks!

Jason Brownlee February 19, 2015 at 8:39 am #

REPLY

The premise I was getting at was that decomposition decreases the complexity of the model by adding complexity in attributes.

If the decomposition is non-linear for example, the model no longer requires the sophistication of relating the non-linear attributes values to the class and other attributes and instead can work with the decomposed elements directly.

Simpler models are better, and often the simplification of using decomposed attributes allows the use of simpler models, such as linear models if the decomposed attributes aid in

linear separability of class values for example.

Cemal February 19, 2015 at 8:26 pm #

Thank you. I have an additional question regarding redundancy.

You suggest that in addition to the original feature “Item_Color” one can use the new features “Has_Color”, “Is_Red”, “Is_Blue” and “Is_Unknown”.

My first thought would be that these new features are redundant because the information is already in the original feature. The new features just provide a different representation of the information.

Could you please elaborate on this ?

Kira November 7, 2014 at 1:34 am #

REPLY

Very nice article, thank you Jason!!! Easy to read for non-computer scientists as well

jasonb November 7, 2014 at 5:08 am #

REPLY

Great to hear, thanks Kira.

Juane November 25, 2014 at 6:57 pm #

REPLY

A clear, content rich article that gives great examples and further reading. Nice work Jason!

Jason Brownlee November 25, 2014 at 7:47 pm #

REPLY

Thanks Juane!

Ziauddin Syed March 5, 2015 at 11:48 pm #

REPLY

Thanx Jason for a explaining feature engineering in a simple language 🙂

Jason Brownlee March 6, 2015 at 5:44 am #

REPLY

You're welcome!

Larry April 2, 2015 at 3:43 pm #

REPLY

very nice article Jason. do you have articles on the different techniqiues in testing and validating the models.

Accepted December 2, 2015 at 1:23 pm #

REPLY

I also want to learn some articles about cross-validation. Thank you!

Andre May 27, 2015 at 9:51 am #

REPLY

Jason, thanks for publishing all this useful knowledge. I'm working through my first ML competition and have read through many of your posts already that answer the questions I have.

Mark July 28, 2015 at 11:28 am #

REPLY

Well written, Jason! I 100% agree, feature engineering is very important and very seldom written about. Kaggle write-ups agree: http://blog.kaggle.com/2014/08/01/learning-from-the-best/.

Cong August 21, 2015 at 9:14 am #

REPLY

Great article and reference links. Do you have some advice about feature engineering for time-series data, e.g. stock price, etc?

Guibing November 25, 2015 at 12:09 pm #

REPLY

http://nbviewer.ipython.org/url/luckymoon.me/docs/Features.ipynb

for the broken link. I wrote it as a note but didn't know it was referenced here.

Chirag Mandot May 3, 2016 at 2:59 am #

REPLY

Hi Jason
This article is amazing. I am currently working on making a Feature Engineering engine, which plots the entire process you described by looking at the source/input data. Its both automated and a manual process. I will share the details once its ready.
Once again thanks for writing this.

Chaks September 22, 2016 at 1:31 pm #

REPLY

Hi Jason, Can you show me how to use PCA in combination with LR or CART? Thanks

Jason Brownlee September 22, 2016 at 5:30 pm #

REPLY

I have a short example of PCA for data prep here that may help:
http://machinelearningmastery.com/pre-process-your-dataset-in-r/

Jason Zhang November 1, 2016 at 3:30 am #

REPLY

It is a great article. Thanks a lot.

Jason Brownlee November 1, 2016 at 8:01 am #

REPLY

I'm glad you found it useful Jason.

tatta November 8, 2016 at 7:30 pm #

REPLY

Thanks Jason, a very useful article with a lot of great resources. Much appreciated.

Jason Brownlee November 9, 2016 at 9:49 am #

REPLY

I'm glad you found it useful tatta.

Marcos November 21, 2016 at 10:18 pm #

REPLY

Great article! Here there is a new book that you could add to your reference section of this article: http://shop.oreilly.com/product/0636920049081.do

Jason Brownlee November 22, 2016 at 7:06 am #

REPLY

Thanks for the pointer Marcos. I've got the early access but I've not read it yet. It's high on my list.

Olalekan Fuad Elesin November 22, 2016 at 3:21 am #

REPLY

This is really cool tutorial, Jason. Many thanks. Currently working on some dataset that has only timestamp and value. With what I've learn't, engineering features (timestamp) to extract hour_of_day, season_of_year, reading and reading_remainders will be very useful.

vielen Dank

Jason Brownlee November 22, 2016 at 7:07 am #

REPLY

Very nice!

I plan to write a feature engineering post on time series data soon.

Sander January 31, 2017 at 12:19 pm #

REPLY

very good thank you,
by the are there some code examples for
https://en.wikipedia.org/wiki/Deep_feature_synthesis
http://www.feature-labs.com/

seems to be I sow in past some non commercial version of this method

Jason Brownlee February 1, 2017 at 10:37 am #

REPLY

Hi Sander, sorry I am not familiar with deep feature synthesis.

Ravi Shankar February 5, 2017 at 5:28 am #

REPLY

From your articles, I do understand the importance of recognizing features in ML applications. Though very new to ML, I got fasicinated to look this solve some of the challenging problems in my world. I am currently looking at a dataset for diagnosing product issues. Simply put for every issue, there is an ID, problem description and resolution.

Problem description is line of text. Can this be defined as a feature? I can always see more issues where description is the same or close. But the line of text should be mapped to a number!

Instead, should I extract more out of text as features by text analysis so that they can be translated to numbers which enable ML algorithms to work easily.

Thanks again for the good sources which is helping me orient to the exciting field.

Jason Brownlee February 6, 2017 at 9:41 am #

REPLY

Great question Ravi.

You can extract features from a line of text, you can also treat each word or pairs/triples of words as features.

You may want to read up on good representations for text data in the field on natural language processing (NLP) like bag of words and word embeddings.

Ed Smith February 12, 2017 at 4:29 pm #

REPLY

Thanks for this article, extremely helpful!

Jason Brownlee February 13, 2017 at 9:12 am #

REPLY

I'm glad to hear it Ed.

Zubair April 18, 2017 at 8:14 pm #

REPLY

Hi Jason, Thanks for this post

I have a question about following

'need a well considered and designed test harness for objectively estimating model skill on unseen data'

How to achieve this, would this be the test data after the train/test split or do you mean more than that?

Jason Brownlee April 19, 2017 at 7:51 am #

REPLY

Great question, see this process:
http://machinelearningmastery.com/start-here/#process

Bala June 20, 2017 at 10:39 pm #

REPLY

Excellent article Jason. Very helpful, especially the examples. I am new to ML, maybe this is a very basic question but still i am asking.
The Item_color can be mentioned as Has_color and Has_unknown , which is a very good feature to introduce.
Then we would have two features right ?One the original Item_color and the newly added one. How do we deal with this.
Is it like we do not consider the original feature while modeling (use the new one) or we use both ?

Jason Brownlee June 21, 2017 at 8:16 am #

REPLY

Yes, you can replace the original feature or keep it. Test and use model skill to help you discover what is predictive.

Saiskanth June 22, 2017 at 8:39 pm #

REPLY

Nice article. Are there any packages in R for feature engineering?

Jason Brownlee June 23, 2017 at 6:41 am #

REPLY

I don't know, sorry.

Mayukh Sarkar July 8, 2017 at 5:42 pm #

REPLY

Feature engineering is a process. There is no one shot package to help you out..I urge you understand what is feature engineering more. It's process. Not some algorithm that is available with a package.

Jason Brownlee July 9, 2017 at 10:52 am #

REPLY

I agree.

Eugene Vyborov July 31, 2017 at 2:27 pm #

REPLY

Very interesting and valuable article, thank you. Feature selection is a very practical piece of the entire process, meaning that it is more than everything else requires a deep understanding of the subject under research.

Jason Brownlee July 31, 2017 at 3:50 pm #

REPLY

I agree that it is key!

Kono September 9, 2017 at 5:02 am #

REPLY

Hey Jason,

Nice post and it is the 2nd Google search result for "feature engineering" keyword. A question here:

In general, how to deal with count based features such as click count, reservation count etc.? The problem with such features is their values keep increasing, which will render the model useless over time. Even if the time frame is limited to the previous month, it will be subject to seasonality.

Jason Brownlee September 9, 2017 at 12:01 pm #

REPLY

Perhaps use thresholds and binary flags? (above x, below x)

Perhaps use discrete bins? (low, medium, high)

Perhaps use a rescaling (log?)

Aceit September 10, 2017 at 9:07 pm #

REPLY

I really liked the post. It is very comprehensive. Indeed, I really like posts from Dr.Brownlee because it has both simplicity yet do not lose the comprehensiveness.

I do have questions about the post, though.
Although there may not be absolute answers for these questions, your opinions will still be very helpful.

1) When would be better to leave the nominal data as it is instead of transferring to one-hot encoding?
– For most of the machine learning algorithms I am aware of, there is no algorithm that prefers nominal data type to binary features that were converted by one-hot encoding.(As long as it's not ordinal). Your example of eye color will be one of them, but we can have countries, states, a day of week and many other examples. Those categories in nominal data cannot have ordinal relationship, and therefore, suggested to be transformed to individual feature by one-hot encoding. If so, when would be a good choice to leave the nominal feature as it is? Is there any machine learning algorithm that solves the problem better with nominal feature instead of one-hot category features?

2) Selecting the algorithm
When there is a question about which algorithm to select, the most common answer is that 'it depends on the data set'. Easy to say, but in fact, I often feel the answer is irresponsible.
For the data set with few features with only numerical variables, all machine learning learning algorithms will welcome it. Someone may say linear regression will be the best because of its simplicity, but let's be honest. With our computing power, choosing XGboost won't take too much resources instead of using linear regression.

The problem occurs when we have mixed data type variables(both numeric and categorical), and when we have many features(high dimension) and sparse data(many zeros in each feature).
The problem is again, there is no machine learning algorithm that welcomes those kind of data set. I have been searching for tips to handle mixed data type(let's say 80% of features are one-hot and 20% are numerical) and sparse data(if we convert nominal data to one-hot, this will happen).
No one has really given me a clear answer for this question.

Some people say regular regression algorithm works fine with many categorical data(one-hot encoded), but some say complicated model such as XGboost works better. But there is an absent of reasoning.

I will be really excited to hear from you about this.

If there are any theoretical reasoning, that will be VERY H
ELPFUL.

Thank you in advance.

Jason Brownlee September 11, 2017 at 12:07 pm #

REPLY

Good questions. There is no best representation or algorithm and we cannot choose them analytically a priori.

We must discover what works best for our specific data. That is the task of applied machine learning.

See this post:
http://machinelearningmastery.com/a-data-driven-approach-to-machine-learning/

Lubaba October 8, 2017 at 8:27 am #

REPLY

hi sir
how can we engineer our features using weka tool?

Jason Brownlee October 8, 2017 at 8:44 am #

REPLY

You would use filters I expect.

Sorry, I don't have an example.

Bhartendu Dwivedi October 12, 2017 at 7:07 am #

REPLY

It was well descriptive as well as comprehensive. It would have been much better if you give some engineering related problem along with the evolution of feature in that. Tough good article (y)

Jason Brownlee October 13, 2017 at 5:41 am #

REPLY

Thanks for the suggestion.

Ben November 23, 2017 at 7:58 am #

REPLY

Thanks for this article Jason. It is a real eye-opener to a whole lots of ideas in feature engineering. However, I am presently trying to work on Android permissions attribute, but I still don't fully understand how to transform these attributes into trainable features. My goal is to develop a model for detecting Android malware. Please do you have any suggestion or guide to further help me grasp this concept? From your work, you gave an example of transforming categorical attributes into binary features, please if you could help me out I would greatly appreciate. Thanks Jason.

Jason Brownlee November 23, 2017 at 10:43 am #

REPLY

I'm not familiar with the data, I don't have good suggestions sorry.

Perhaps you can find examples of similar data that has been framed for machine learning and use that as inspiration?

Max Kanter December 2, 2017 at 7:15 am #

REPLY

Even though feature engineering is so important for machine learning, it's surprising how little formal attention and study has been given to it, so great to see an article like this!

This post speaks well to using an existing feature matrix and then performing feature engineering to improve it before applying ML. However, another important scenario I see come up over and over again is the case where you have multi-table or very fine-grained datasets that need feature engineering done. For instance, a relational database of customer information or log files of user interactions with a product. What tools do people here use for dataset likes this?

My colleagues and I ran into the problem of performing feature engineering to these datasets so frequently, we released an open source python library for automated feature engineering called Featuretools (www.featuretools.com). It's based data science research at MIT that resulted in an algorithm called Deep Feature Synthesis [1]. Hopefully, it can be useful to people here.

[1] https://dai.lids.mit.edu/wp-content/uploads/2017/10/DSAA_DSM_2015.pdf

---

Jason Brownlee December 2, 2017 at 9:07 am #

REPLY

Thanks for sharing the link, though what you describe becomes feature engineering more than feature selection:
http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

Haim January 9, 2018 at 4:26 am #

REPLY

How to distinguish between a good feature to a bad one? Is correlation (or squared-correlation) enough?

Jason Brownlee January 9, 2018 at 5:35 am #

REPLY

The best way is that it improves the predictive skill of the model.

Haim January 9, 2018 at 8:41 am #

REPLY

Is it possible to examine the feature before running the model (assuming the data is huge)? I'm asking this because sometimes I see that the least correlated feature is the most important one when presenting feature importance graph.

Jason Brownlee January 9, 2018 at 3:18 pm #

REPLY

For sure.

Ephraïm January 17, 2018 at 1:42 am #

REPLY

Hi Jason, I'm working on features reduction for an existing dataset for Logistic Regression. Can I remove individual variables used to create combined ones ? By combined variables, I mean the aggregate variables (sum or mean of some independant variables).
Thanks.
By the way, great article.

Jason Brownlee January 17, 2018 at 9:59 am #

REPLY

Sure, try it and see how it impacts your model.

Alee January 17, 2018 at 9:05 pm #

REPLY

Sir, I have raw data from which i have to extract features. I have myself done data collection. I have to perform binary classification . So I wanted to know is it necessary that initially many features should be there and after that we select features useful for us.
And second question, is it necessary to have large quantity of data to apply machine learning algorithm? I have less data but I have covered various patterns i.e. quality of training data is good.
Thanks

Jason Brownlee January 18, 2018 at 10:08 am #

REPLY

I would recommend extracting as many features as you can think of to see what might add value.

See this post on how much data you need:
https://machinelearningmastery.com/much-training-data-required-machine-learning/

Minh Kharetti February 14, 2018 at 11:05 pm #

REPLY

This is great Jason, thanks for putting in all the work to find, summarize and present this information on such an important topic. It really brings home the point of the importance of this stage of the process, and how it separates good ML teams from great ones. I'm bookmarking this page so I can come back to it, read key points and get through all the links.

Jason Brownlee February 15, 2018 at 8:43 am #

REPLY

Thanks, I'm glad it helped.

Michael Swan February 23, 2018 at 8:25 am #

REPLY

Hi Jason,

You stated in this article that "we will probably never have automatic feature engineering", however I actually found your article after looking into Feature Labs and their work on automatic feature engineering (see Featuretools). It appears they are attempting to synthesize the "human intuition" aspect of feature engineering at least partially. Do you have any thoughts on this? Has your opinion about automated (or near automated) feature engineering changed since you wrote this article?

Jason Brownlee February 23, 2018 at 12:04 pm #

REPLY

Interesting.

I guess deep learning is also a solution to this, e.g. feature learning. But really to a lesser degree. Humans are so much more efficient at this, especially if they have a little domain knowledge.

Linda March 13, 2018 at 3:01 pm #

REPLY

The way which you have run is good because you won't solve the disable machines. Many Engineer does not think this type of thoughts.Latest deep learning ways are achieving some success in this area, such as autoencoders and restricted Boltzmann machines.

Jason Brownlee March 13, 2018 at 3:06 pm #

REPLY

Do you have some examples?

Bob Copeland March 18, 2018 at 3:26 am #

REPLY

I'm probably just basically stupid, however, I'm starting from the bottom up on the "Titanic". I'm totally clueless & haven't any comprehension of the different systems to get a "perfect" answer but I'm somewhat logical. It seems to me from an intellectual perspective that most of the systems are a 50% yes or no coin toss and there isn't any known consistent logical relationship between whatever factors you select. I do realize in A. I. that you use a multitude of examples & that process gets you closer to an answer through "weights" or "diddling" from my humorous perspective. I humorously look at it as trying to decide which person in a group is going to definitely die on a certain date.

Jason Brownlee March 18, 2018 at 6:07 am #

REPLY

We can use the baseline rate, such as predicting the mean outcome as a baseline model.

If a method cannot outperform the baseline, it has no value or no skill.

I believe on the titanic dataset, results are far better than a baseline.

Perhaps this process will help you work through your dataset:
https://machinelearningmastery.com/start-here/#process

Rohit September 14, 2018 at 2:36 pm #

REPLY

Hi Jason. Some basic feature transformations usually suggested are log transformations, or taking a higher polynomial of the same variable and adding it to the dataset (like in this answer: https://datascience.stackexchange.com/a/10664). Since these new variables are generated from the existing ones, do they not introduce multicollinearity which should be worried about? Could you throw some light on this point?

Jason Brownlee September 15, 2018 at 6:00 am #

REPLY

Yes, they can, depending on the order. Nevertheless, they can improve the predictive skill of the model.

Dilsha Dominic October 28, 2018 at 5:38 pm #

REPLY

Thanks a lot Jason

Jason Brownlee October 29, 2018 at 5:54 am #

REPLY

I'm happy it helped.

Sekhar November 27, 2018 at 2:30 am #

REPLY

Hi Jason,

Need your help please..

I have been given a normalized data set (original data is not provided) asked to identify new features based on any existing relationship.

I cannot find any relation ship by seeing the data. How should I go forward.. Will Cluster Analysis help to find the new features?

Please Advise

Thanks

Sekhar

Jason Brownlee November 27, 2018 at 6:36 am #

REPLY

What do you mean by new features? Like feature engineering?

You could use mean, min, max, stdev, mode, median, etc.

Kahina January 17, 2019 at 11:50 am #

REPLY

Thank you soooo much for sharing this great work

Jason Brownlee January 17, 2019 at 1:45 pm #

REPLY

I'm glad it helped.

Rebwar Mala Nabi February 3, 2019 at 9:31 am #

REPLY

Very Interesting article. However, I am wondering whether it is possible to have feature engineering for stock price prediction using ensemble methods.

Jason Brownlee February 4, 2019 at 5:43 am #

REPLY

Perhaps, I don't know about stock price prediction, sorry.

abid April 5, 2019 at 10:15 am #

REPLY

I would like to know is there a method or a function in python to calculate the classification error is not global which actually equals (FDR = 1- precision) but the error for each features as the importance for each features . Thank you in advance

Jason Brownlee April 5, 2019 at 2:01 pm #

REPLY

No, I have not seen such a thing, other than perhaps for very simple linear models that attribute the amount of variance in a prediction to a coefficient/feature.

Prem July 11, 2019 at 9:31 pm #

REPLY

Hi Jason,
In feature engineering, if we find only 10 features out of 30 has got information value or importance measure, others nothing, can we delete them from the dataset.

Jason Brownlee July 12, 2019 at 8:42 am #

REPLY

Think of feature engineering and feature selection as “suggestions”.

Test to confirm the change to input features improves model skill.

Prem Alphonse July 12, 2019 at 10:47 am #

REPLY

Thanks Jason, so does that mean even after feature selection, all the features reach the model and used by the model.

Jason Brownlee July 13, 2019 at 6:49 am #

No, I don’t understand what you mean?

ashesh April 26, 2019 at 1:09 am #

REPLY

I simply loved it. I came across it while doing a coursera course ”
How to Win a Data Science Competition: Learn from Top Kagglers”. The course mentions this link in its reading section. I was feeling stuck on a problem in Kaggle with no significant improvements with tweaking hyper-parameters and changing models. I was feeling the need to engineer more features. However, with my limited knowledge, I had acquired a negative connotation about feature engineering, especially the manual portion. It had its roots in news on deep learning based autoencoders being able

to extract features automatically. While that is true, it can't be applied on every problem. And so the knack of getting insights from the data and creating effective features is very much relevant today. Thanks for writing such an amazing post 🙂

Jason Brownlee April 26, 2019 at 8:35 am #

REPLY

Thanks I'm glad it helped!

Ram May 13, 2019 at 10:05 pm #

REPLY

It is a great article on feature engineering. Does the feature engineering depends on Performance measures (RMSE ? AUC?) and framing of problem ? Can you explain what type of dependency ?

Jason Brownlee May 14, 2019 at 7:43 am #

REPLY

Often we choose features based on their impact on model performance, by some metric that is important to the domain.

Miguel A September 7, 2019 at 7:48 am #

REPLY

Great material , Thank you for sharing.

Jason Brownlee September 8, 2019 at 5:08 am #

REPLY

Thanks, I'm glad it helped.

Andy December 8, 2019 at 4:47 pm #

REPLY

Thanks for your sharing, it's very methodical. Can't wait to read what you recommend.

Jason Brownlee December 9, 2019 at 6:46 am #

REPLY

Thanks.

Niklas December 27, 2019 at 1:41 am #

REPLY

This is another example of great stuff. I don't want to burden you with comments as I prefer to have you generate ever more quality stuff. But I got a reason to post now so thought I would use it to add my thanks. Please multiply my thanks by 1.13 times days of effort to arrive at the appropriate amount of thanks I want to direct your way.

It seems like the 'How we did it' paper link is dead. I found the below link to the paper I believe you are referencing.
https://foreverdata.org/1015/content/milestone1-2.pdf

Jason Brownlee December 27, 2019 at 6:36 am #

REPLY

Thanks Niklas, fixed!

MC February 14, 2020 at 8:09 pm #

REPLY

thanks for the detailed post

could you refer me to unsupervised feature selection\extraction ?

worked examples, python packages etc.

thank you very much

Jason Brownlee February 15, 2020 at 6:24 am #

REPLY

Start here:
https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

Ilona May 27, 2020 at 6:33 am #

REPLY

Hi Jason,

thanks for your great blog post;

for your interest:
the link “Feature Engineering and Classifier Ensemble for KDD Cup 2010” seems not be usable as expected, I have got the paper here:
https://pslcdatashop.web.cmu.edu/KDDCup/workshop/papers/kdd2010ntu.pdf

Jason Brownlee May 27, 2020 at 8:06 am #

REPLY

Thanks, fixed!

seshu duddu July 12, 2020 at 8:15 pm #

REPLY

any suggestions how to be good at feature engineering

Jason Brownlee July 13, 2020 at 6:00 am #

REPLY

Practice on many different datasets.
Fit models on the results and see what works.
Read what techniques people use on competitions.

Bệnh viện Chấn thương Chỉnh hình July 24, 2020 at 3:04 pm #

REPLY

I’ve read a few just right stuff here. Certainly worth bookmarking for revisiting.I wonder how so much effort you place to make any such great informative site.

Jason Brownlee July 25, 2020 at 6:09 am #

REPLY

Thanks!

Ismat MS December 6, 2020 at 2:02 am #

REPLY

Thank you so much for the article. I was trying to find “how to derive features” and reached your post. For a beginner in ML and using text from the healthcare domain, there’s much to learn. But when you describe feature construction as an art, it brings me back to the foundation of Content Analysis methodology for analysing text. In my case, rather than looking at the feature at the document, or phrase, or sentence level (often used for sentiment analysis), I look at the word level.

I think I read somewhere that the basic foundation is often the same, but may have different names. Thanks again!

Jason Brownlee December 6, 2020 at 7:05 am #

REPLY

You’re welcome.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome!
I'm *Jason Brownlee* PhD
and I **help developers** get results with **machine learning**.

Read more

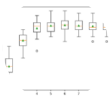
Never miss a tutorial:

Picked for you:

How to Choose a Feature Selection Method For Machine Learning

Data Preparation for Machine Learning (7-Day Mini-Course)

How to Calculate Feature Importance With Python



Recursive Feature Elimination (RFE) for Feature Selection in Python

How to Remove Outliers for Machine Learning

Loving the Tutorials?

The Data Preparation EBook is
where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE