The first part of initiating the Machine Learning problem and solving its aspects is to define the problem. Problem solving is the most important skill in Machine Learning.  To become a good Machine Learning analyst, the analyst has to acquire problem solving skills. Most of the time, the emphasis is on learning a programming language rather than on inculcating the problem solving skills.

The difficulty faced by analysts in understanding the real world problems and developing computer solutions has motivated the world to train on Machine Learning/ Artificial Intelligence/ Data Science as a diverse subset than computer science.

It deals with the techniques needed to practice computational thinking, the art of using computers to solve problems and the ways the computers can be used to solve problems.

1. **Problem Definition**

**How to Define your Machine Learning Problem**

The first step in any project is defining your problem. You can use the most powerful and shiniest algorithms available, but the results will be meaningless if you are solving the wrong problem.

In this post you will learn the process for thinking deeply about your problem before you get started. This is unarguably the most important aspect of applying machine learning.

[](https://3qeqpr26caki16dnhd19sv6by6v-wpengine.netdna-ssl.com/wp-content/uploads/2013/12/question.jpg)

## **Problem Definition Framework**

I use a simple framework when defining a new problem to address with machine learning. The framework helps me to quickly understand the elements and motivation for the problem and whether machine learning is suitable or not.

The framework involves answering three questions to varying degrees of thoroughness:

* **Step 1**: What is the problem?
* **Step 2**: Why does the problem need to be solved?
* **Step 3**: How would I solve the problem?

## Step 1: What is the Problem

The first step is defining the problem. I use a number of tactics to collect this information.

### Informal description

Describe the problem as though you were describing it to a friend or colleague. This can provide a great starting point for highlighting areas that you might need to fill. It also provides the basis for a one sentence description you can use to share your understanding of the problem.

For example: *I need a program that will recommend me videos of once taste/ genres*

## Step 2: Why does the the problem need to be solved?

The second step is to think deeply about why you want or need the problem solved.

### Motivation

Consider your motivation for solving the problem. What need will be fulfilled when the problem is solved?

For example, you may be solving the problem as a learning exercise. This is useful to clarify as you can decide that you don’t want to use the most suitable method to solve the problem, but instead you want to explore methods that you are not familiar with in order to learn new skills.

### Solution Benefits

Consider the benefits of having the problem solved. What capabilities does it enable?

It is important to be clear on the benefits of the problem being solved to ensure that you capitalize on them. These benefits can be used to sell the project to colleagues and management to get buy in and additional time or budget resources.

If it benefits you personally, then be clear on what those benefits are and how you will know when you have got them. For example, if it’s a tool or utility, then what will you be able to do with that utility that you can’t do now and why is that meaningful to you?

### Solution Use

Consider how the solution to the problem will be used and what type of lifetime you expect the solution to have. As programmers we often think the work is done as soon as the program is written, but really the project is just beginning its maintenance lifetime.

The way the solution will be used will influence the nature and requirements of the solution you adopt.

Consider whether you are looking to write a report to present results or you want to operationalize the solution. If you want to operationalize the solution, consider the functional and nonfunctional requirements you have for a solution, just like a software project

## Step 3: How would I solve the problem?

In this third and final step of the problem definition, explore how you would solve the problem manually.

List out step-by-step what data you would collect, how you would prepare it and how you would design a program to solve the problem. This may include prototypes and experiments you would need to perform which are a gold mine because they will highlight questions and uncertainties you have about the domain that could be explored.

This is a powerful tool. It can highlight problems that actually can be solved satisfactorily using a manually implemented solution. It also flushes out important domain knowledge that has been trapped up until now like where the data is actually stored, what types of features would be useful and many other details.

## Summary

In this post you learned the value of being clear on the problem you are solving. You discovered a three step framework for defining your problem with practical tactics at at step:

* **Step 1: What is the problem?** Describe the problem informally and formally and list assumptions and similar problems.
* **Step 2: Why does the problem need to be solve?** List your motivation for solving the problem, the benefits a solution provides and how the solution will be used.
* **Step 3: How would I solve the problem?** Describe how the problem would be solved manually to flush domain knowledge.

1. Data Analysis
2. EDA Concluding Remarks

**Step 2**: Prepare your data.

* [How to Prepare Data For Machine Learning](http://machinelearningmastery.com/how-to-prepare-data-for-machine-learning/)
* [How to Identify Outliers in your Data](http://machinelearningmastery.com/how-to-identify-outliers-in-your-data/)

# How to Prepare Data for Machine Learning

Machine learning algorithms learn from data. It is critical that you feed them the right data for the problem you want to solve.

Even if you have good data, you need to make sure that it is in a useful scale, format and even that meaningful features are included.

In this post you will learn how to prepare data for a machine learning algorithm. This is a big topic and you will cover the essentials.

## Data Preparation Process

The more disciplined you are in your handling of data, the more consistent and better results you are like likely to achieve. The process for getting data ready for a machine learning algorithm can be summarized in three steps:

* **Step 1**: Select Data
* **Step 2**: Preprocess Data
* **Step 3**: Transform Data

## Step 1: Select Data

This step is concerned with selecting the subset of all available data that you will be working with. There is always a strong desire for including all data that is available, that the maxim “more is better” will hold. This may or may not be true.

You need to consider what data you actually need to address the question or problem you are working on. Make some assumptions about the data you require and be careful to record those assumptions so that you can test them later if needed.

Below are some questions to help you think through this process:

* What is the extent of the data you have available? For example through time, database tables, connected systems. Ensure you have a clear picture of everything that you can use.
* What data is not available that you wish you had available? For example data that is not recorded or cannot be recorded. You may be able to derive or simulate this data.
* What data don’t you need to address the problem? Excluding data is almost always easier than including data. Note down which data you excluded and why.

It is only in small problems, like competition or toy datasets where the data has already been selected for you.

## Step 2: Preprocess Data

After you have selected the data, you need to consider how you are going to use the data. This preprocessing step is about getting the selected data into a form that you can work.

Three common data preprocessing steps are formatting, cleaning and sampling:

* **Formatting**: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.
* **Cleaning**: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.
* **Sampling**: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

## Step 3: Transform Data

The final step is to transform the process data. The specific algorithm you are working with and the knowledge of the problem domain will influence this step and you will very likely have to revisit different transformations of your preprocessed data as you work on your problem.

Three common data transformations are scaling, attribute decompositions and attribute aggregations. This step is also referred to as feature engineering.

* **Scaling**: The preprocessed data may contain attributes with a mixtures of scales for various quantities such as dollars, kilograms and sales volume. Many machine learning methods like data attributes to have the same scale such as between 0 and 1 for the smallest and largest value for a given feature. Consider any feature scaling you may need to perform.
* **Decomposition**: There may be features that represent a complex concept that may be more useful to a machine learning method when split into the constituent parts. An example is a date that may have day and time components that in turn could be split out further. Perhaps only the hour of day is relevant to the problem being solved. consider what feature decompositions you can perform.
* **Aggregation**: There may be features that can be aggregated into a single feature that would be more meaningful to the problem you are trying to solve. For example, there may be a data instances for each time a customer logged into a system that could be aggregated into a count for the number of logins allowing the additional instances to be discarded. Consider what type of feature aggregations could perform.

You can spend a lot of time engineering features from your data and it can be very beneficial to the performance of an algorithm. Start small and build on the skills you learn.

## Summary

In this post you learned the essence of data preparation for machine learning. You discovered a three step framework for data preparation and tactics in each step:

* **Step 1: Data Selection** Consider what data is available, what data is missing and what data can be removed.
* **Step 2: Data Preprocessing** Organize your selected data by formatting, cleaning and sampling from it.
* **Step 3: Data Transformation** Transform preprocessed data ready for machine learning by engineering features using scaling, attribute decomposition and attribute aggregation.

Data preparation is a large subject that can involve a lot of iterations, exploration and analysis. Getting good at data preparation will make you a master at machine learning. For now, just consider the questions raised in this post when preparing data and always be looking for clearer ways of representing the problem you are trying to solve.

# How to Identify Outliers in your Data

## Outliers

Many machine learning algorithms are sensitive to the range and distribution of attribute values in the input data.

Outliers in input data can skew and mislead the training process of machine learning algorithms resulting in longer training times, less accurate models and ultimately poorer results.

Even before predictive models are prepared on training data, outliers can result in misleading representations and in turn misleading interpretations of collected data. Outliers can skew the summary distribution of attribute values in descriptive statistics like mean and standard deviation and in plots such as histograms and scatterplots, compressing the body of the data.

Finally, outliers can represent examples of data instances that are relevant to the problem such as anomalies in the case of fraud detection and computer security.

## Outlier Modeling

Outliers are extreme values that fall a long way outside of the other observations. For example, in a normal distribution, outliers may be values on the tails of the distribution.

The process of identifying outliers has many names in data mining and machine learning such as outlier mining, outlier modeling and novelty detection and anomaly detection.

* **Extreme Value Analysis**: Determine the statistical tails of the underlying distribution of the data. For example, statistical methods like the z-scores on univariate data.
* **Probabilistic and Statistical Models**: Determine unlikely instances from a probabilistic model of the data. For example, gaussian mixture models optimized using expectation-maximization.
* **Linear Models**: Projection methods that model the data into lower dimensions using linear correlations. For example, principle component analysis and data with large residual errors may be outliers.
* **Proximity-based Models**: Data instances that are isolated from the mass of the data as determined by cluster, density or nearest neighbor analysis.
* **Information Theoretic Models**: Outliers are detected as data instances that increase the complexity (minimum code length) of the dataset.
* **High-Dimensional Outlier Detection**: Methods that search subspaces for outliers give the breakdown of distance based measures in higher dimensions

4. Pre-processing Pipeline

* [Improve Model Accuracy with Data Pre-Processing](http://machinelearningmastery.com/improve-model-accuracy-with-data-pre-processing/)
* [Discover Feature Engineering](http://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/)
* [Data Leakage in Machine Learning](http://machinelearningmastery.com/data-leakage-machine-learning/)

# Improve Model Accuracy with Data Pre-Processing

Data preparation can make or break the predictive ability of your model.

You must pre-process your raw data before you model your problem. The specific preparation may depend on the data that you have available and the machine learning algorithms you want to use.

Sometimes, pre-processing of data can lead to unexpected improvements in model accuracy. This may be because a relationship in the data has been simplified or unobscured.

Data preparation is an important step and you should experiment with data pre-processing steps that are appropriate for your data to see if you can get that desirable boost in model accuracy.

There are three types of pre-processing you can consider for your data:

* Add attributes to your data
* Remove attributes from your data
* Transform attributes in your data

## Add Data Attributes

Advanced models can extract the relationships from complex attributes, although some models require those relationships to be spelled out plainly. Deriving new attributes from your training data to include in the modeling process can give you a boost in model performance.

* **Dummy Attributes**: Categorical attributes can be converted into n-binary attributes, where n is the number of categories (or levels) that the attribute has. These denormalized or decomposed attributes are known as dummy attributes or dummy variables.
* **Transformed Attribute**: A transformed variation of an attribute can be added to the dataset in order to allow a linear method to exploit possible linear and non-linear relationships between attributes. Simple transforms like log, square and square root can be used.
* **Missing Data**: Attributes with missing data can have that missing data imputed using a reliable method, such as k-nearest neighbors.

## Remove Data Attributes

Some methods perform poorly with redundant or duplicate attributes. You can get a boost in model accuracy by removing attributes from your data.

* **Projection**: Training data can be projected into lower dimensional spaces, but still characterize the inherent relationships in the data. A popular approach is Principal Component Analysis (PCA) where the principal components found by the method can be taken as a reduced set of input attributes.
* **Spatial Sign**: A spatial sign projection of the data will transform data onto the surface of a multidimensional sphere. The results can be used to highlight the existence of outliers that can be modified or removed from the data.
* **Correlated Attributes**: Some algorithms degrade in importance with the existence of highly correlated attributes. Pairwise attributes with high correlation can be identified and the most correlated attributes can be removed from the data.

## Transform Data Attributes

Transformations of training data can reduce the skewness of data as well as the prominence of outliers in the data. Many models expect data to be transformed before you can apply the algorithm.

* **Centering**: Transform the data so that it has a mean of zero and a standard deviation of one. This is typically called data standardization.
* **Scaling**: A standard scaling transformation is to map the data from the original scale to a scale between zero and one. This is typically called data normalization.
* **Remove Skew**: Skewed data is data that has a distribution that is pushed to one side or the other (larger or smaller values) rather than being normally distributed. Some methods assume normally distributed data and can perform better if the skew is removed. Try replacing the attribute with the log, square root or inverse of the values.
* **Box-Cox**: A Box-Cox transform or family of transforms can be used to reliably adjust data to remove skew.
* **Binning**: Numeric data can be made discrete by grouping values into bins. This is typically called data discretization. This process can be performed manually, although is more reliable if performed systematically and automatically using a heuristic that makes sense in the domain.

## Summary

Data pre-process is an important step that can be required to prepare raw data for modeling, to meet the expectations of data for a specific machine learning algorithms, and can give unexpected boosts in model accuracy.

In this post we discovered three groups of data pre-processing methods:

* Adding Attributes
* Removing Attributes
* Transforming Attributes

# Discover Feature Engineering

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.

## Problem that Feature Engineering Solves

When your goal is to get the best possible results from a [predictive model](https://machinelearningmastery.com/gentle-introduction-to-predictive-modeling/), 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.

## 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.

### 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?

### 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.

## 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](http://en.wikipedia.org/wiki/Feature_(machine_learning)).

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](http://en.wikipedia.org/wiki/Random_forest#Variable_importance) 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](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/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.

### 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.

### 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](http://en.wikipedia.org/wiki/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](http://machinelearningmastery.com/what-is-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](http://machinelearningmastery.com/process-for-working-through-machine-learning-problems/).

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.

# Data Leakage in Machine Learning

Data leakage is a big problem in machine learning when developing [predictive models](https://machinelearningmastery.com/gentle-introduction-to-predictive-modeling/).

Data leakage is when information from outside the training dataset is used to create the model.

## Goal of Predictive Modeling

The goal of predictive modeling is to develop a model that makes accurate predictions on new data, unseen during training.

## What is Data Leakage in Machine Learning?

Data leakage can cause you to create overly optimistic if not completely invalid predictive models.

Data leakage is when information from outside the training dataset is used to create the model. This additional information can allow the model to learn or know something that it otherwise would not know and in turn invalidate the estimated performance of the mode being constructed.

### Data Leakage is a Problem

It is a serious problem for at least 3 reasons:

1. **It is a problem if you are running a machine learning competition**. Top models will use the leaky data rather than be good general model of the underlying problem.
2. **It is a problem when you are a company providing your data**. Reversing an anonymization and obfuscation can result in a privacy breach that you did not expect.
3. **It is a problem when you are developing your own predictive models**. You may be creating overly optimistic models that are practically useless and cannot be used in production.

5. Building Machine Learning Models

**Step 3**: Spot-check algorithms.

* [How to Evaluate Machine Learning Algorithms](http://machinelearningmastery.com/how-to-evaluate-machine-learning-algorithms/)
* [Why you should be Spot-Checking Algorithms on your Machine Learning Problems](http://machinelearningmastery.com/why-you-should-be-spot-checking-algorithms-on-your-machine-learning-problems/)
* [How To Choose The Right Test Options When Evaluating Machine Learning Algorithms](http://machinelearningmastery.com/how-to-choose-the-right-test-options-when-evaluating-machine-learning-algorithms/)
* [A Data-Driven Approach to Choosing Machine Learning Algorithms](http://machinelearningmastery.com/a-data-driven-approach-to-machine-learning/)

**Step 4**: Improve results.

* [How to Improve Machine Learning Results](http://machinelearningmastery.com/how-to-improve-machine-learning-results/)
* [Machine Learning Performance Improvement Cheat Sheet](http://machinelearningmastery.com/machine-learning-performance-improvement-cheat-sheet/)
* [How To Improve Deep Learning Performance](http://machinelearningmastery.com/improve-deep-learning-performance/)

6. Concluding Remarks

vsbsbsbsb