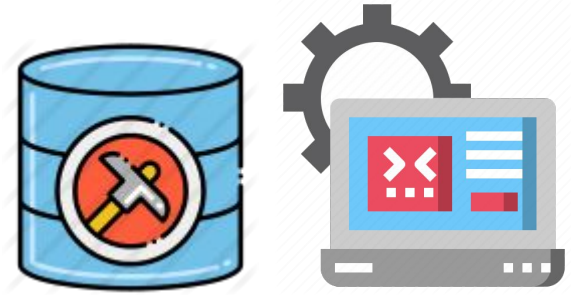
SOFTANBEES
SIMPLIFYING DATA SOLUTIONS

Feature Engineering & Data Preprocessing

with

R



Feature Engineering & Data Preprocessing

Two of the most important steps in developing a machine learning model is **feature engineering** and **preprocessing.**

Feature engineering is the process of using domain knowledge to extract features from raw data. These features can be used to improve the performance of machine learning algorithms.
Feature engineering can be considered as applied machine learning itself.

Data preprocessing is a **data mining** technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.
Data preprocessing is a proven method of resolving such issues.

In simpler words,
Feature engineering consists of the creation of features whereas **preprocessing** involves cleaning the data.

The What, Why And How Of Feature Engineering

The WHAT

Feature engineering involves leveraging data mining techniques to extract features from raw data along with the use of domain knowledge. Feature engineering is useful to improve the performance of machine learning algorithms and is often considered as applied machine learning.

Feature engineering involves several processes.
Feature selection, construction, transformation, and extraction are some key aspects of feature engineering.

Feature selection involves choosing a set of features from a large collection. Selecting the important features and reducing the size of the feature set makes computation in machine learning and data analytic algorithms more feasible. Feature selection also improves the quality of the output obtained from algorithms.

Feature transformation involves creating features using existing data by the use of mathematical operations.

Feature construction is the process of developing new features apart from the ones generated in feature transformation, that are appropriate variables of the process under study.

Feature extraction is a process of reducing the dimensionality of a dataset. Feature extraction involves combining the existing features into new ones thereby reducing the number of features in the dataset. This reduces the amount of data into manageable sizes for algorithms to process, without distorting the original relationships or relevant information.

The What, Why And How Of Feature Engineering

The WHY

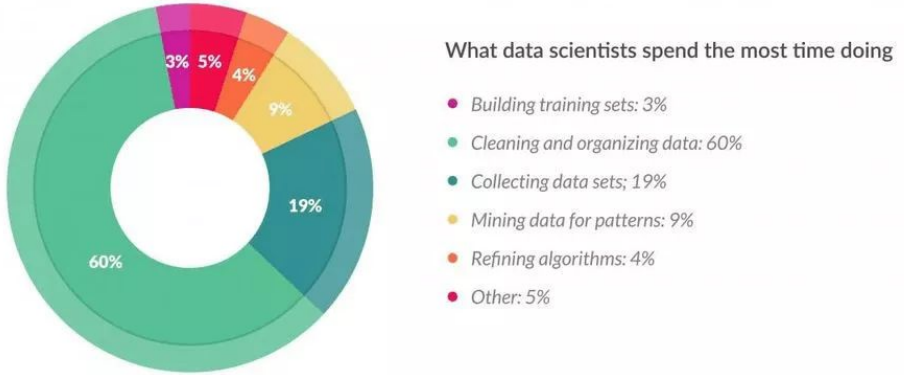
The intention of feature engineering is to achieve two primary goals:

1. Preparing an input dataset that is compatible with and best fits the machine learning algorithm.

2. Improving the performance of machine learning models

When feature engineering processes are executed well, the resulting dataset will be **optimal** and contain all the essential factors that bear an **impact** on the business problem. These datasets in turn result in best possible predictive models and most beneficial insights.

Data scientists spend **80%** of their time on data preparation. The importance of feature engineering is realised through its time-efficient approach to preparing data that brings consistent output.



The What, Why And How Of Feature Engineering

The HOW

There are a few common feature engineering techniques used. We'll Describe them here.

Imputation

One of the most common problems in machine learning is the absence of values in the datasets. Irrespective of the cause, absence of values affects the performance of machine learning algorithms.

Some platforms do not accept datasets with missing data. By using the method of Imputation, values are introduced into the dataset that are coherent with the existing values. Although there are many imputation methods, replacing missing values with the median of the column or the maximum value occurred is a common imputation method.

One-Hot Encoding

This is one of the common encoding methods used in feature engineering. One-hot encoding is a method of assigning binary values (0's and 1's) to values in the columns. In this method, all values above the threshold are converted to 1, while all values equal to or below the threshold are converted as 0. This changes the feature values to a numerical format which is much easier for algorithms to understand without compromising the value of the information and the relationship between the variables and the feature.

Grouping Operations

In machine learning algorithms, a variable or instance is represented in rows and features are represented in columns. Many datasets rarely fit into the simplistic arrangement of rows and columns as each column has multiple rows of an instance. To handle such cases, data is grouped in such a fashion that every variable is represented by only one row. The intention of grouping operations is to arrive at an aggregation that establishes the most viable relationship with features.

The What, Why And How Of Feature Engineering

The HOW

There are a few common feature engineering techniques used. We'll Describe them here.

Bag of Words

Bag of Words (BoW) is a counting algorithm that evaluates the number of repetitions of a word in a document. This algorithm is useful in identifying similarities and differences in documents for applications like search and document classification.

Feature Hashing

Feature hashing is an important technique used to scale-up machine learning algorithms by vectorizing features. The technique of feature hashing is commonly used in document classification and sentiment analysis where tokens are converted into integers. Hash values are derived by applying hash functions to features that are used as indices to map data.

Log Transformation

A measure of asymmetry in a dataset is known as Skewness, which is defined as the extent to which a given distribution of data varies from a normal distribution. Skewness of data affects the prediction models in ML algorithms. To resolve this, Log Transformations are used to reduce the skewness of data. The less skewed distributions are, the better is the ability of algorithms to interpret patterns.

Automated Feature Engineering

Automated feature engineering is a new technique that is becoming a standard part of machine learning workflow. The traditional approach is a time consuming, error-prone process and is specific to the problem at hand and has to change with every new dataset. This will increase the efficiency of data scientists by helping them spend more time on other elements of machine learning and would enable citizen data scientists to do feature engineering using a framework based approach.

The What, Why And How Of Data Preprocessing

The WHAT

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

The most common problems you can find with raw data can be divided into 3 groups:
Missing data, Noisy data & Inconsistent Data.

Missing data:
This can be seen as inaccurate data since the information that isn't there creates gaps that might be relevant to the final analysis. Missing data often appears when there's a problem in the collection phase, such as a glitch that caused a system's downtime, mistakes in data entry, or issues with biometrics use, among others.

Noisy data:
This group encompasses erroneous data and outliers that we can find in the data set but that is just meaningless information. Here we can see noise made of human mistakes, rare exceptions, mislabels, and other issues during data gathering.

Inconsistent data:
Inconsistencies happen when we keep files with similar data in different formats and files. Duplicates in different formats, mistakes in codes of names, or the absence of data constraints often lead to inconsistent data, that introduces deviations that we have to deal with before analysis.

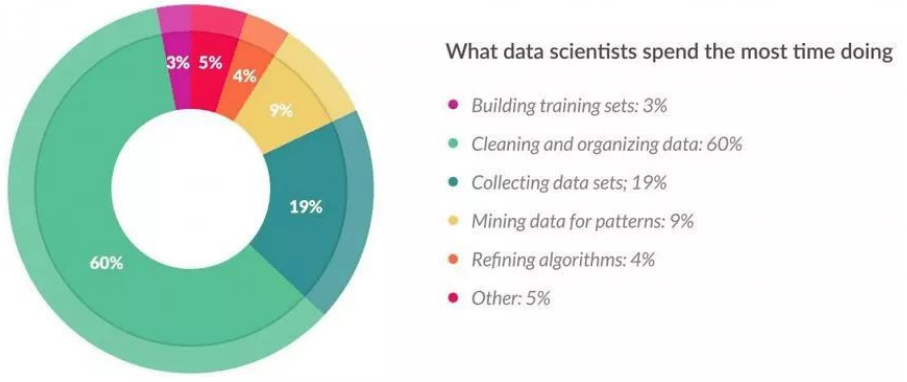
The What, Why And How Of Data Preprocessing

The WHY

Data preprocessing is very important. Since mistakes, redundancies, missing values, and inconsistencies all compromise the integrity of the set, we need to fix all those issues for a more accurate outcome. Imagine we are training a Machine Learning algorithm to deal with our customers' purchases with a faulty dataset. Chances are that the system will develop biases and deviations that will produce a poor user experience.
Thus, before using that data for the purpose we want, we need it to be as organized and "clean" as possible. There are several ways to do so, depending on what kind of problem we're tackling.

Data scientists spend **60%** of their time on data preparation. The importance of data preprocessing is realised through its time-efficient approach to preparing data that brings consistent output.

Virtually any type of data analytics, data science or AI development requires some type of data preprocessing to provide reliable, precise and robust results for enterprise applications. Good preprocessing can help align the way data is fed into various algorithms for building machine learning or deep learning models.



The What, Why And How Of Data Preprocessing

The HOW

There are a few common preprocessing techniques used. We'll Describe them here.

Identify and sort out missing data

There are a variety of reasons that a data set might be missing individual fields of data. Data scientists need to decide whether it is better to discard records with missing fields, ignore them or fill them in with a probable value. For example, in an IoT application that records temperature, it may be safe to add in the average temperature between the previous and subsequent record when required.

Noisy data

Real-world data is often noisy, which can distort an analytic or AI model. For example, a temperature sensor might erroneously report a temperature as 250 degrees Fahrenheit, while previous and subsequent measurements might be about 75 degrees. A variety of statistical approaches can be used to reduce the noise, including binning, regression and clustering.

Identify and remove duplicates

When two records seem to repeat, an algorithm needs to determine if the same measurement was recorded twice or the records represent different events. In some cases, there may be slight differences in a record because one field was recorded incorrectly. In other cases, different records might represent a father and son living in the same house, which really do represent separate individuals. Techniques for identifying and removing or joining duplicates can help to automatically address these types of problems.

The What, Why And How Of Data Preprocessing

The HOW

There are a few common preprocessing techniques used. We'll Describe them here.

Feature scaling or normalization

Often, multiple variables change over different scales, or one will change linearly while another will change exponentially. For example, salary might be measured in thousands of dollars, while age will be represented in double digits. Scaling helps to transform the data in a way that its easier for algorithms to tease apart a meaningful relationship between variables.

Discretization

It's often useful to lump raw numbers into discrete intervals. For example, income might be broken into five ranges that are representative of people who typically apply for a given type of loan. This can reduce the overhead of training a model or running inferences against it.

Data reduction

A data scientist may wish to combine a variety of data sources for creating a new AI or analytics model. Some of the variables may not be correlated with a given outcome -such as the likelihood of loan repayment & may be safely discarded. Other variables might be relevant, but only in terms of relationship - such as the ratio of debt to credit & may be combined into a single variable.

Feature encoding

Another aspect of feature engineering lies in organizing unstructured data into a structured format. Unstructured data formats can include text, audio and video. For example, the process of developing natural language processing algorithms typically starts by using data transformation algorithms like Word2vec to translate words into numerical vectors. This make is easy to represent to the algorithm that words like "mail" and "parcel" are similar, while a word like "house" is completely different. Similarly, a facial recognition algorithm might re-encode raw pixel data into vectors representing the distances between parts of the face.

Libraries in R which are used in feature engineering

Data.table
purrr
RecordLinkage
stringr
tm
Recipes
Superml
Timetk
caret

Libraries in R which are used in Data Preprocessing

outliers
Dplyr
Lubridate
Mlr
Janitor
Magrittr
Tidyr
Prrr
Sqldf
Caret
RANN
Catools
tidymodels

Feature Selection Algorithms

Reason to use feature engineering

There are three reasons why we don't give all the features to the ML algorithm and let it decide which feature is important.

1. Curse of dimensionality — Overfitting
If we have more columns in the data than the number of rows, we will be able to fit our training data perfectly, but that won't generalize to the new samples. And thus we learn absolutely nothing.

2. Occam's Razor:
We want our models to be simple and explainable. We lose explainability when we have a lot of features.

3. Garbage In Garbage out:
Most of the times, we will have many non-informative features. For Example, Name or ID variables. Poor-quality input will produce Poor-Quality output.

Also, a large number of features make a model bulky, time-taking, and harder to implement in production.

Feature Selection Methods

There are a lot of ways in which we can think of feature selection, but most feature selection methods can be divided into three major buckets

Filter based: We specify some metric and based on that filter features. An example of such a metric could be correlation/chi-square.

Wrapper-based: Wrapper methods consider the selection of a set of features as a search problem. Example: Recursive Feature Elimination

Embedded: Embedded methods use algorithms that have built-in feature selection methods. For instance, Lasso and RF have their own feature selection methods.

Feature Selection Algorithms

Recursive Feature Elimination

This is a wrapper based method. Wrapper methods consider the selection of a set of features as a search problem.
From sklearn Documentation:
The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a coef_ attribute or through a feature_importances_ attribute. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Chi-Squared

This is another filter-based method.
In this method, we calculate the chi-square metric between the target and the numerical variable and only select the variable with the maximum chi-squared values.
The chi-squared statistic also works in a hand-wavy way with non-negative numerical and categorical features.

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Pearson Correlation

This is a filter-based method.
We check the absolute value of the Pearson's correlation between the target and numerical features in our dataset. We keep the top n features based on this criterion.

$$r = \frac{N\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{[N\Sigma x^2 - (\Sigma x)^2][N\Sigma y^2 - (\Sigma y)^2]}}$$

Lasso

This is an Embedded method.Embedded methods use algorithms that have built-in feature selection methods.
For example, Lasso and RF have their own feature selection methods. Lasso Regularizer forces a lot of feature weights to be zero.

Tree-based

This is an Embedded method.Embedded methods use algorithms that have built-in feature selection methods.

We calculate feature importance using node impurities in each decision tree. In Random forest, the final feature importance is the average of all decision tree feature importance.

Tidymodels for data preprocessing

Step 1

Splitting the training, testing and validation datasets.

Step 2

Preprocessing and Feature Engineering

This part mostly concerns what we can *do* to our variables to make the models more effective.
This is mostly related to the predictors. Operations that we might use are:

- transformations of individual predictors or groups of variables
- alternate encodings of a variable
- elimination of predictors (unsupervised)

Understanding Step 2

Reasons for Modifying the Data

- Some models (*K*-NN, SVMs, PLS, neural networks) require that the predictor variables have the same units. **Centering** and **scaling** the predictors can be used for this purpose.
- Other models are very sensitive to correlations between the predictors and **filters** or **PCA signal extraction** can improve the model.
- As we'll see in an example, changing the scale of the predictors using a **transformation** can lead to a big improvement.
- In other cases, the data can be **encoded** in a way that maximizes its effect on the model. Representing the date as the day of the week can be very effective for modeling public transportation data.
- Many models cannot cope with missing data so **imputation** strategies might be necessary.
- Development of new *features* that represent something important to the outcome (e.g. compute distances to public transportation, university buildings, public schools, etc.)

Tidymodels for data preprocessing

Step 3
Categorical Encoding Procedure

One common procedure for modeling is to create numeric representations of categorical data. This is usually done via *dummy variables*: a set of binary 0/1 variables for different levels of an R factor.

Most dummy variable procedures would make *two* numeric variables from this predictor that are 1 when the observation has that level, and 0 otherwise.

A *zero-variance* predictor that has only a single value (zero) would be the result.

Many models (e.g. linear/logistic regression, etc.) would find this numerically problematic and issue a warning and **NA** values for that coefficient. Trees and similar models would not notice.

There are two main approaches to dealing with this:

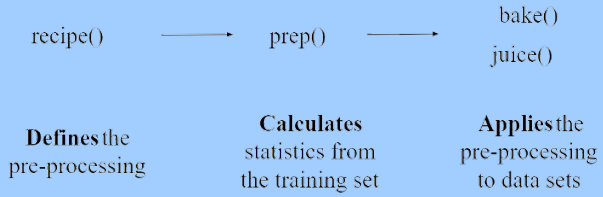
- Run a filter on the training set predictors prior to running the model and remove the zero-variance predictors.
- Recode the factor so that infrequently occurring predictors (and possibly new values) are pooled into an "other" category

Tidymodels for data preprocessing

Step 4
Recipes

Recipes are an alternative method for creating the data frame of predictors for a model. They allow for a sequence of *steps* that define how data should be handled.
For a formula: **log10(Sale_Price) ~ Longitude + Latitude?** These steps are:

- Assign Sale_Price to be the outcome
- Assign Longitude and Latitude as predictors
- Log transform the outcome



juice() is used to get the pre-processed training set (basically for free)

bake() is used to pre-process a *new* data set

recipe() - data is used *only* to determine column names and types. A 0-row data frame could even be used.
prep() - training is the entire training set, used to estimate parameters in each step (like means or standard deviations).
bake() - new_data is data to apply the pre-processing to, using the *same estimated parameters* from when prep() was called on the training set.
The step() functions help us make a better recipe by preprocessing the data. Such as:

- step_rm() removes variables; here we'll use it to remove the original date variable since we no longer want it in the model.
- step_dummy() converts characters or factors (i.e., nominal variables) into one or more numeric binary model terms for the levels of the original data.
- step_normalize() centers and scales numeric variables.

Tidymodels for data preprocessing

Step 5
Principal Component Analysis for dimensionality reduction

In any dataset, there are potential clusters of *highly correlated variables*. It would be nice if we could combine/amalgamate the variables of clusters into a single variable that represents them. Another way of putting this is that we would like to create artificial features of the data that account for a certain amount of *variation* in the data.

There are a few different methods that can accomplish this; such as principal component analysis (PCA).

Principal component analysis (PCA) is a multivariate statistical technique that can be used to create artificial new variables from an existing set.

Conceptually, PCA determines which variables account for the most correlation in the data and creates a new variable that is a linear combination of all the predictors.

- This is called the *first principal component* (aka `PC1`).
- This linear combination emphasizes the variables that are the most correlated.

The variables constructing `PC1` are then *removed from the data*. The second PCA component is the linear combination that accounts for the most left-over correlation in the data (and so on).

The main takeaways:

- The components account for as much as the variation in the original data as possible.
- Each component is uncorrelated with the others.
- The new variables are *linear combinations* of all of the input variables and are effectively unitless (It is generally a good idea to center and scale your predictors because of this).

For our purposes, we would use PCA on the *predictors* to:

- Reduce the number of variables exposed to the model (but this is not feature selection).
- Combat excessive correlations between the predictors (aka multicollinearity).

In this way, the procedure is often called *signal extraction* but this is poorly named since there is no guarantee that the new variables will have an association with the outcome.

PCA does a *rotation* of the data so that the *variation* in one dimension is maximized.

The rotation also makes the new variables *uncorrelated*.

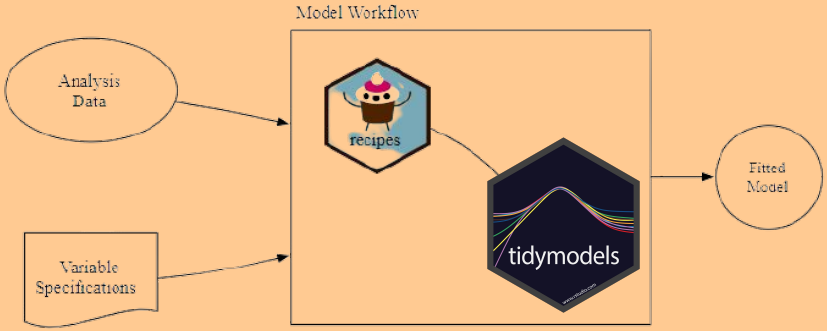
Tidymodels for data preprocessing

Step 6
Combining the Recipe with a Model

We are

- estimating transformations on some variables,
- making nonlinear features for the geocoding variables, and
- estimating new encodings for some factor variables.

These should definitely be included in our overall modeling process.



Step 7

Workflows

The workflows package enables a handy type of object that can bundle pre-processing and models together.

- You don't have to keep track of separate objects in your workspace.
- The recipe prepping and model fitting can be executed using a single call to fit() instead of prep()-juice()-fit().
- The recipe baking and model predictions are handled with a single call to predict() instead of bake()-predict().
- Workflows *will* be able to add post-processing operations in upcoming versions. An example of post-processing would be to modify (and tune) the probability cutoff for two-class models.