DATA SCIENCE INFINITY

DATA PREP & CLEANING FOR MACHINE LEARNING

AN OVERVIEW

Why?

Data Preparation & Cleaning is an **extremely important** part of the overall Machine Learning process, one that **must** be considered before ever looking to build or train a model.

A common phrase in Machine Learning is...

"Garbage in...Garbage Out!"

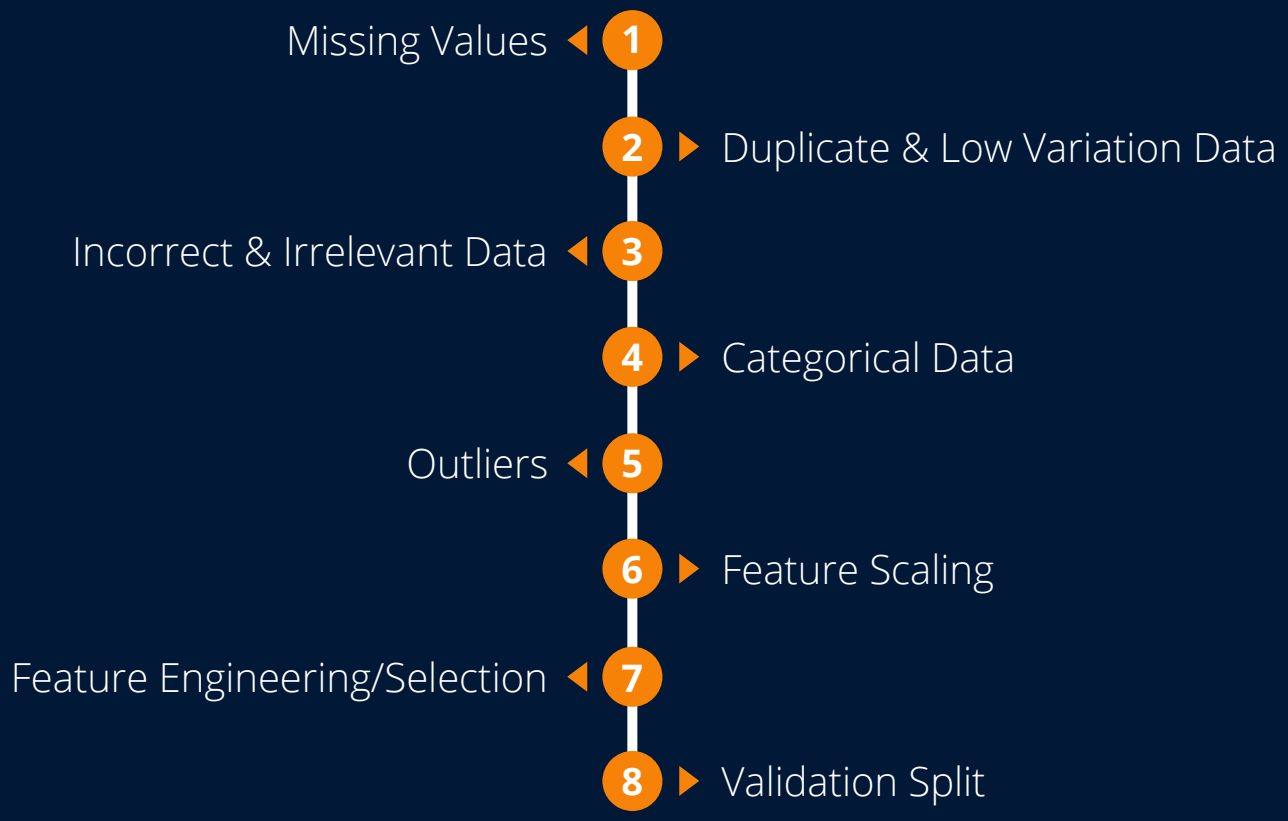
If the data isn't clean or isn't prepared in an appropriate way, even the fanciest of algorithms or models will struggle to learn.

On top of this, ensuring the data is clean can actually be one of the biggest boosters of model performance and accuracy!

DATA SCIENCE INFINITY

8-Step Checklist

This 8-Step Data Preparation & Cleaning checklist will ensure you're always giving your ML model the best chance to learn & perform!



Note: Which steps are applicable can depend on the data you're using, the problem you're solving, and on the type of model you're applying! However, in the vast majority of cases you can't go wrong at least considering these 8 steps!

Missing Values

1

In many cases, your chosen model or algorithm simply won't know how to process missing values - and you will be **returned an error**.

Even if the missing values don't lead to an error you do always want to ensure that you are **passing the model the most useful information to learn from** - and you should consider whether missing values meet that criteria or not.

The two most common approaches for dealing with missing values:

Removal: Often you will simply remove any observations (rows) where one or missing values are present. You can also remove entire columns if no information is present.

Imputation: This is where you input or "impute" replacement values where they were originally missing. This can be based upon the column mean, median, or most common value or more advanced approaches that take into account other present data points to give an estimation of what the missing value might be!

Duplicate & Low Variation Data

When looking through your data, don't just look for missing values - keep an eye out for duplicate data, or data that has low variation.

Duplicate data is most commonly rows of data that are exactly the same across all columns. This duplicate rows do not add anything to the learning process of the model or algorithm, but do add storage & processing overhead.

In the vast majority of cases you can remove duplicate rows prior to training your model.

Low variation data is where a column in your dataset contain only one (or few) unique value(s).

An example: There is a column in a house price dataset called "property_type". Every row in this column has the value "house". This column won't add any value to the learning process, so can be removed.

Incorrect & Irrelevant Data

Irrelevant data is anything that isn't related specifically to the problem you're looking to solve. For example, if you're predicting house prices, but your dataset contains commercial properties as well - these would need to be removed.

Incorrect data can be hard to spot! An example could be looking for values that shouldn't be possible such a negative house price values.

For categorical or text variables you should spend time analysing the unique values within a column.

For example, you were predicting car prices and had a column in the dataset called "car_colour". Upon inspection you find values of "Orange", "orange", "orang" are all present. These need to be rectifed prior to training otherwise you will be limited the potential learning that an take place!

Always explore your data thoroughly!

Categorical Data

Generally speaking, ML models like being fed numerical data. They are less fond of categorical data!

Categorical data is anything that is listed as groups, classes, or text. A simple example would be a column for gender which contains values of either "Male" or "Female".

Your model or algorithm won't know how to assign some numerical importance to these values, so you often want to turn these groups or classes *into* numerical values.

A common approach is called **One Hot Encoding** where you create new columns, one for *each unique class* in your categorical column. You fill these new columns with values of 1 or 0 depending on which is true for each observation.

Other encoding techniques you can consider are; Label Encoding, Binary Encoding, Target Encoding, Ordinal Encoding, & Feature Hashing.

Outliers

There is no formal definition for an outlier. You can think of them as any data point that is very different to the majority.

How you deal with outliers is dependent on the problem you are solving, and the model you are applying. For example, if your data contained one value that was 1000x any other, this could badly affect a Linear Regression model which tries to generalise a rule across all observations. A Decision Tree would be unaffected however, as it deals with each observation independently.

In practice, outliers are commonly isolated using the number of standard deviations from the mean, or a rule based upon the interquartile range.

In cases where you want to mitigate the effects of outliers, you may look to simply **remove** any observations (rows) that contain outlier values in one or more of the columns or you may look to **replace their values** to reduce their effect.

Always remember - just because a value is very high, or very low, that **does not mean** it is wrong to be included.

Feature Scaling

Feature Scaling is where you force all the values from a column in your data to exist on the same scale. In certain scenarios it will help the model assess the relationships between variables more fairly, and more accurately.

The two most common scaling techniques are:

Standardisation: rescales all values to have a mean of 0 and standard deviation of 1. In other words, the majority of your values end up between -4 and +4

Normalisation: rescales data so that it exists in a range between 0 and 1

Feature Scaling is **essential** for distance-based models such as k-means or k-nearest-neighbours.

Feature Scaling is **recommended** for any algorithms that utilise Gradient Descent such as Linear Regression, Logistic Regression, and Neural Networks.

Feature Scaling is **not necessary** for tree-based algorithms such as Decision Trees & Random Forests.

Feature Engineering & Selection

Feature Engineering is the process of using further knowledge to supplement or transform the original feature set.

The key to good Feature Engineering is to create or refine features that the algorithm or model can understand *better* or that it will find *more useful* than the raw features for solving the particular problem at hand.

Feature Selection is where you only keep a subset of the most informative variables. This can be done using human intuition, or dynamically based upon statistical analysis.

A smaller feature set can lead to improved model accuracy through reduced noise. It can mean a lower computational cost, and improved processing speed. It can also make your models easier to understand & explain to stakeholders & customers!

Validation Split

The Validation Split is where you partition your data into a **training set**, and a **validation set** (and sometimes a **test set** as well).

You train the model with the **training set only**. The validation and/or test sets, are held-back from training and are used to assess model performance. They provide a true understanding of how accurate predictions are on **new or unseen data**.

An approach called **k-fold cross validation** can provide you with an even more robust understanding of model performance.

Here, the entire dataset is again partitioned into training & validation sets, and the model is trained and assessed like before. However, this process is done multiple times with the training and test sets being rotated to encompass different sets of observations within the data. You do this **k** times - with your final predictive accuracy assessment being based upon the average of each of the iterations.