thinktank.

Why do some Predictive Algorithms Fail?

White Paper

This White Paper is part of the ‘Think Tank’ series by Metrix Data Science.

Getting it Wrong: Passenger frustration

Algorithms often inaccurately predict. A common trap many algorithms fall into is a phenomenon called overfitting, more on this later. Sometimes the consequences of misfiring algorithms are harmless but occasionally they are disastrous.

Take airlines as an example. Airlines regularly oversell seats knowing that there will be last-minute cancellations. There is a story of one airline that developed a suite of models based on data taken from previous bookings and cancellation data. The analyst who built the model was a Harvard graduate but lacked industry experience. The analyst developed an exceptionally complex regression model with some 30-40 variables, which should have been a red light in itself, and the model could explain 99% of the variation in cancellations. The airline subsequently adopted this model and used it to predict passenger numbers. The result was that many flights were over-booked by 50% and only a 20% drop off rate materialised, resulting in utter chaos as many flights were over-subscribed. Retrospective interrogation of the model identified overfitting as the culprit.

Background

A common question data scientists ask themselves is: how complex does my algorithm need to be to understand the relationship I am investigating? Models are, by definition, simplified versions of reality. Probability plays such a large role in understanding how models work precisely because it is important to quantify how likely it is that any inference is down to random chance. The desire to cheat the laws of probability and produce a model which almost perfectly fits the data runs into the intractable issue of overfitting.

This paper will regularly reference training and test data. To explain, statistical modelling typically involves establishing two groups of data. The training set is used to build the model i.e. the model parameters are derived using this data. The testing set is used to evaluate the predictiveness of the model i.e. how accurately can the model predict new data.

Overfitting is a situation where the algorithm is too complex for the data it is trying to understand. It is best demonstrated by thinking about a relationship between two variables X and Y shown in figure 1. The line with peaks and troughs which covers all data points is known as a polynomial model and these humps in the curve are the result of an excessively complex model specification. This model perfectly describes the training data since there is no error (error refers to the difference between the actual data points and the values the model predicts) as the curve goes through every data point. While very good at predicting the training data, when applied to different data (i.e. the test data), the model will not be able to make accurate predictions because it has been excessively fine-tuned to fit the training dataset.

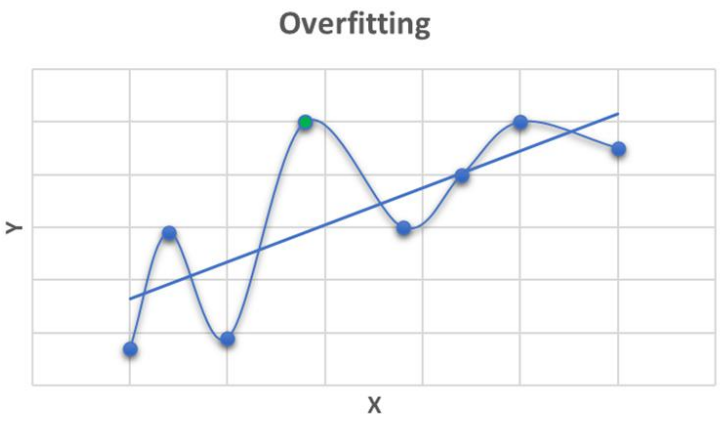


Figure 1

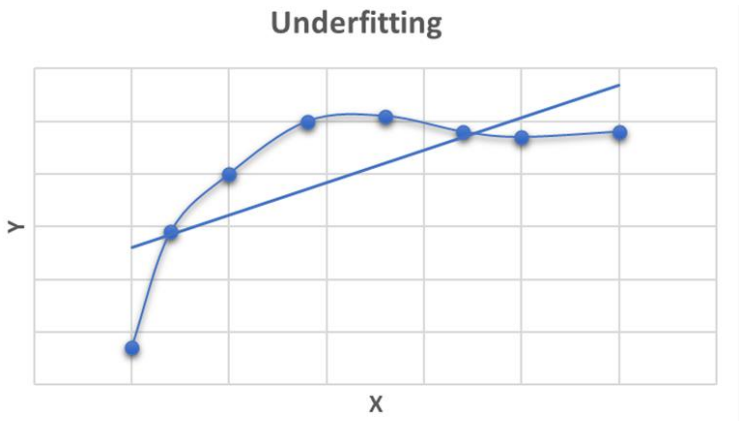


Figure 2

This situation arises precisely because the underlying distribution of the data has been incorrectly identified. In the case of figure 1 the underlying distribution of the data is linear, hence a straight-line curve fits the data fairly well. There are data points, such as the green data point in figure 1, with a high error value. However, having a model with some error is a necessary sacrifice to produce a generalisable model which can predict values of Y from new values of X.

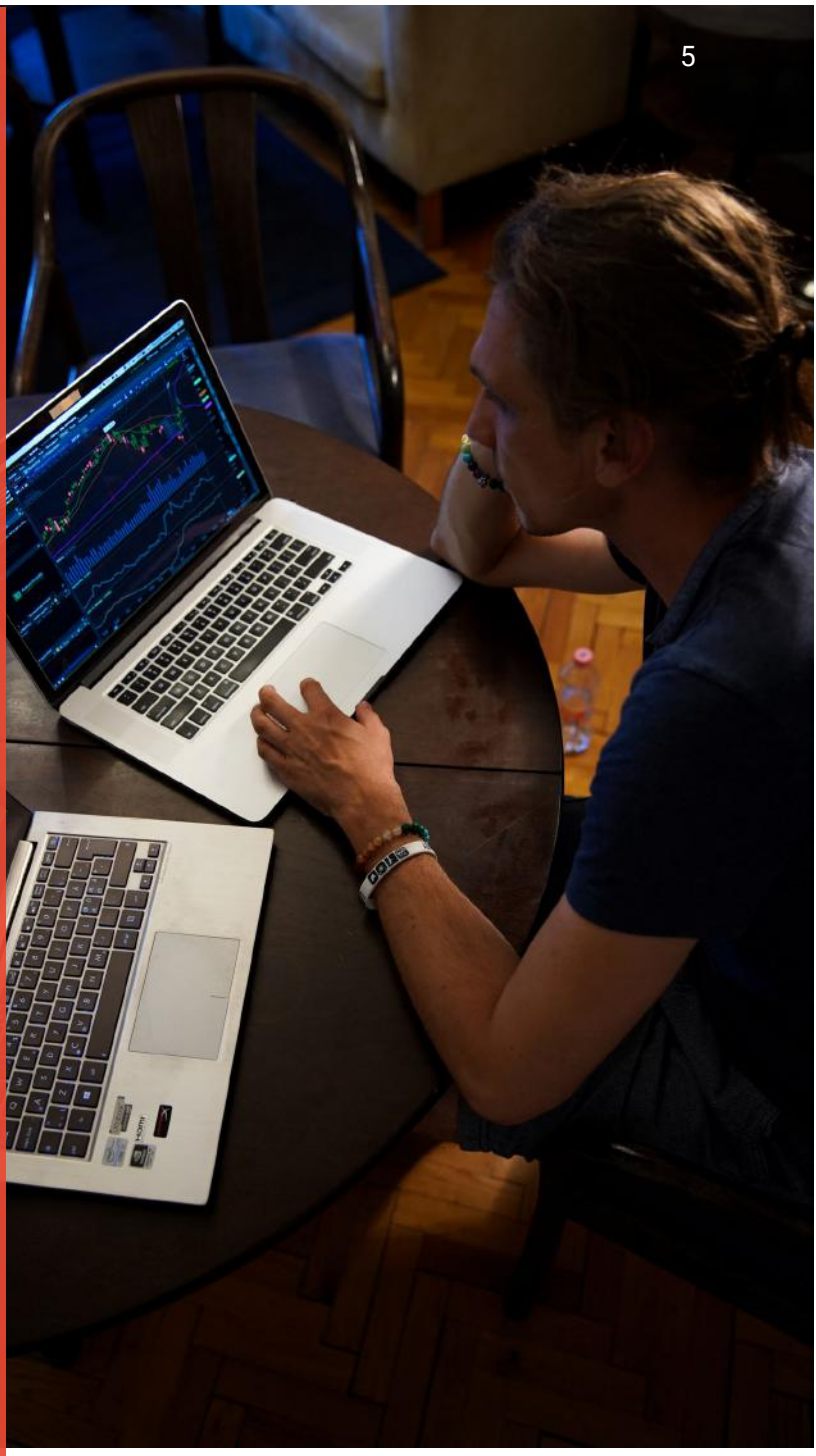
This may lead to the conclusion that the simpler the model the better since you are less likely to experience overfitting. However, this is not always the case because on the other end of the spectrum is underfitting. This occurs when the model specification is overly simplified and does not pick up on the true data distribution. Figure 2 shows how fitting a linear trend line through data that are clearly non-linear would result in a model which will not perform accurate prediction.

How to detect Over/Under Fitting

An underfit model is very easy to detect. The model is used to make predictions for both the training and the test data. For underfitting, the error statistic for training data and the test data will be very high. In other words, the model is completely unfit for explaining the relationship in both training and test data – the worst of both worlds. This situation corresponds to the area on the left-hand side of the graph (see figure 3 below). The model complexity is low (i.e. the model is too simplistic to understand the data) and the error for test and training data is high. Having a high error for the training data alone would be a sufficient red flag to completely disregard this model.

Overfitting concerns the right-hand side of the graph where the model is overly complex, resulting in a large difference between the testing data and training data error. In this situation, the model is so good at predicting the training data that its parameters do badly when tested on new data. Such a gulf in error which favours the training data is a strong indication of overfitting.

The sweet spot occurs in the middle where the error for both data sets is sufficiently low and the difference is not particularly marked. It will always be the case that the training error is lower since the algorithm is always going to have a better understanding of the data it was trained on.



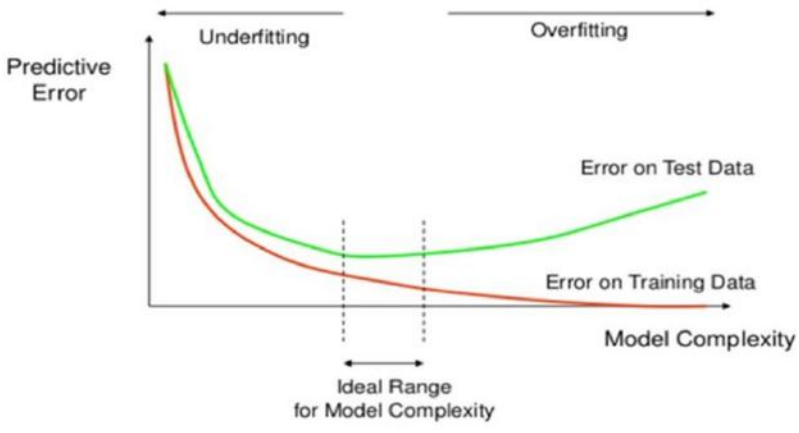
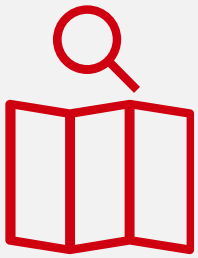


Figure 3

How to solve over/ underfitting



The key to solving this problem is a full understanding of the underlying distribution of your data. Identifying the distribution of the data should be prioritised before conducting modelling. This can be achieved through constructing scatter diagrams and running tests for various quirks within the data.

This being said, in the real-world datasets seldom perfectly fit a certain distribution. Especially when data is split into smaller training and test sets. Therefore it is always instructive to calculate the error statistics for the training and test datasets. This will provide an indication of where the model fits in the spectrum of over-under fitting. Once this is established the answer is to:

- Simplify your model if it is overfit
- Increase the complexity of your model if it is underfit

There are various ways to achieve this. Adding/reducing variables can be the quickest way to change the complexity of a model. The more variables the higher the complexity. In the context of a regression, this can involve adding interaction terms, squared terms or log transformations. When dealing with times series data, seasonality can render linear regression models useless unless you add a sinusoidal component that models the data based on a sin wave. This added complexity is very likely to improve the model's predictiveness.

Changing the modelling technique can also vary model complexity. Some models are by their nature more complex. For example, in terms of classification algorithms, there is a sliding scale of complexity. Logistic regression is fairly low complexity, ensemble methods like random forest are in the medium complexity range and a large neural network represents high a complexity model.

Summary

Data scientists tread a precarious tight rope between creating algorithms that are very good at predicting and algorithms that are generalisable. Too much of one takes away from the other. This is why models are constantly iterated and tweaked as more information becomes available and even then, there will always be a degree of error to any predictive model. At the heart of this is the over/under fitting debate is knowing how to identify the existence of either of these problems . Playing with model specification and modelling methodology can unlock the potential of the model to be both accurate and generalisable.