Principles of Model Selection

Welcome to the first module in PA II - **Model Selection**. In this module, you will learn the fundamental principles governing almost all modelling techniques in machine learning.

After studying the basic principles of model selection in this session, you will learn how to evaluate models using cross-validation.

In the previous course on predictive analytics, you may have come across situations where a model performs well on the training data but not on the test data. Also, you will shortly learn some new classification models such as SVMs, decision trees, random forests etc. Given a classification problem, how do you choose one model among all these choice

Thus, you saw that modeling should not be just a trial and error process, but rather a well thought out one based on sound domain knowledge.

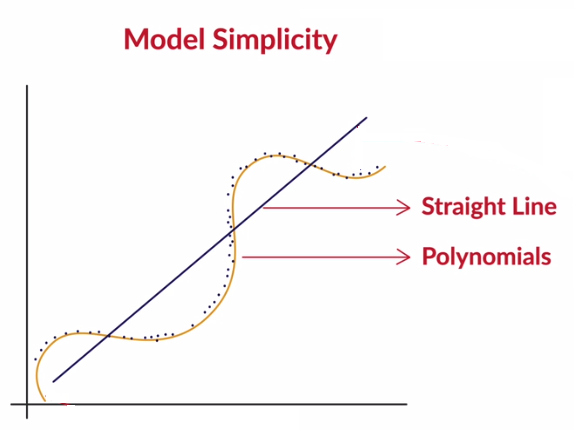
As you learnt, the central issue in all of the machine learning is 'how do we extrapolate learnings from a finite amount of **available data** to all possible inputs ‘of the same kind'?' Training data is always finite, yet the model is supposed to learn everything about the task at hand from it and perform well on unseen data.

Though there is **no one answer to this difficult question**, there are tried and tested principles which can be used to choose, create and evaluate models.

How do you ensure, and be confident, that the model is as good as it seems on the training data and deploy it to make predictions on real, unseen data?

Often, it is mistaken that if a model performs well on the training data, it will produce good results on test data as well. Very often, that is not the case.

As you learnt, a **straight line model** is considered as the **simpler model** because it doesn't try to fit through every training point and results in some positive error i.e∑ni=0(εi)>0∑i=0n(εi)>0  as shown in the figure below. Whereas the polynomial model is considered as a **complex model** which tends to make zero error(∑ni=0(εi)∼0∑i=0n(εi)∼0) in training data. Thus, it becomes important that the simpler models are bound to perform better on unseen data sets other than the polynomial model.



**Figure-1: Model Simplicity**

The dilemma of choosing a model, as discussed in this lecture, is not just confined to **regression problems**. In classification problems, you can build **extremely complex models** having **close to 100% accuracy**, but is that the right choice? How do you avoid the temptation of choosing an 'accurate' model over one which is simpler but less accurate?

There is the clear **trade-off**: On one hand, you have good performance (on the training data), on the other, you have stability.

**Model Complexity**

From your school or college time, you can probably recall those few fellows who seemed to study less but understood much more than others. They seem to never care about memorizing or mechanically practising what was being taught, yet are able to explain complex problems in physics or mathematics with simplicity and elegance.

Assuming that people learn using ‘**mental models**’, do these students have remarkably different mental models than those who solve a bunch of books and focus on memorization? How can they learn so much from a finite amount of information and apply that to solve unseen complex problems?

In this lecture, prof. Raghaavn will explain the meaning of **model simplicity**, **complexity** as well as the **pros and cons** associated with them. As a by-product, you will also understand that the best way to ‘learn’ is ‘to keep your mental models simple’.

As you learnt, a **simpler model is usually more generic than a complex model**. This becomes important because generic models are bound to perform better on unseen data sets.

Apart from being generalisable, there's another benefit of having a simple model - they require less training data to learn.

Till now, you have understood two key benefits of a simpler model  -

1. They are more generalisable.
2. They require less training data points.

Finally, you learned 4 benefits of using a simpler model as opposed to a complex one:

1. A simpler model is usually more **generic** than a complex model. This becomes important because generic models are bound to perform better on unseen datasets.
2. A simpler model requires **less training data points.**This is important because in many cases, one has to work with limited data points.
3. A simple model is **more robust and does not change significantly**if the training data points undergo small changes.
4. A simple model may make more errors in the training phase but it is bound to outperform complex models when it sees new data. Thus, they **perform better on unseen data.**

# Overfitting

In this lecture, you will understand one of the most common problems that arise because of model complexity - **overfitting**.  Overfitting, as the name suggests, is a phenomenon where the model '**mugs up**' the dataset rather than learning generalisable patterns from it. Let's understand this in more detail.

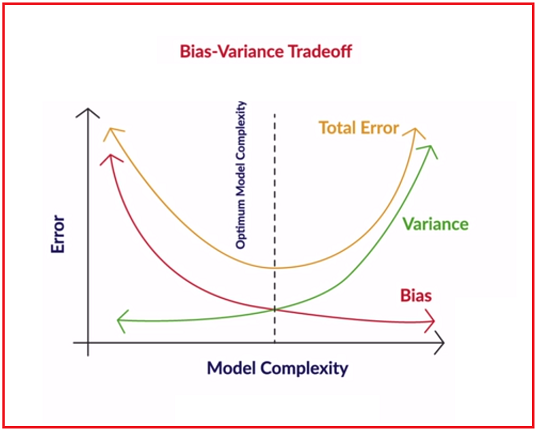
Overfitting is a phenomenon where a model becomes too specific to the data it is trained on and fails to generalise to other unseen data points in the larger domain. A model that has become too specific to a training data set has actually ‘learnt’ not just the hidden patterns in the data but also the noise and the inconsistencies in the data. In a typical case of overfitting, the model performs very well on the training data but fails miserably on the test data.

So far, we have discussed the pros and cons of simple and complex models. On one hand, simplicity is generalizable and robust and on the other hand, some problems are inherently complex in nature. There is a trade-off between the two, which is known as the **bias-variance tradeoff** in machine learning.

We considered the example of a model memorizing the entire training dataset. If you change the dataset a little, this model will need to change drastically. The model is, therefore, **unstable and sensitive to changes in training data**, and this is called **high variance**.

The ‘**variance**’ of a model is the **variance in its output** on some test data with respect to the changes in the training data. In other words, variance here refers to the **degree of changes in the model itself**with respect to changes in training data. Basically, It also refers to changes in the model as a whole when trained on different datasets, rather than the variance in the predicted values of a single model.

**Bias** quantifies the prejudiced assumptions you have made about a particular problem. Extremely simple models are likely to fail in predicting complex real world phenomena. Simplicity has its own disadvantages. It's like trying to perform image classification of Google images using a simple logistic regression model with two variables. In this case, the model is underestimating the amount of complexity involved in the task and is **highly biased.**



**Bias Variance Tradeoff**

Bias = correctness of the model

Variance – consistency of the model

Imagine solving digital image processing problems using simple linear regression when much more complex models like neural networks are typically successful in these problems. We say that the linear model has a high bias since it is way too simple to be able to learn the complexity involved in the task.

In an ideal case, we want to reduce both the bias and the variance, because the expected total error of a model is the sum of the errors in bias and the variance, as shown in the figure above.

Although, in practice, we often cannot have a low bias and low variance model. As the model complexity goes up, the bias reduces while the variance increases, hence the trade-off.

# Model Evaluation and Cross Validation

We now shift our attention towards model evaluation. The key thing to remember is that a model should never be evaluated on data it has already seen before. With that in mind, you will have either one of two cases - 1) The training data is abundant and 2) The training data is limited.

The first case is straightforward because you can use as many observations as you like to both train and test the model. In the second case, however, you will need to find some ‘hack’ so that the model can be evaluated on unseen data and at the same time doesn’t eat up the data available for training. This hack, commonly used in statistics, is called **cross-validation**.

**A Note on Terminology: Training, Validation and Test sets**

Note that the terminology used in the vanilla train-test set approach is often loosely used. The accurate terminology is as follows:

When you divide the available data into 70:30 or 80:20, the majority data is called the **training set,**the remaining part is called the **validation set**(rather than the 'test' set) using which you validate the model while training (and make changes in model hyperparameters accordingly). The **test data,** on the other hand, is something that the model never uses while training or validation, i.e. it is kept isolated from the model throughout the model building process.

Though most people often refer to the validation set as the 'test' set, it is good to be aware of the technically correct terminology.

Thus, you saw that cross-validation is especially useful in cases when you have a limited amount of training data. However, you can use it even when you have abundant data.

In the upcoming modules, such as SVMs, decision trees, and random forests, you will use cross-validation heavily to assess model performance. Therefore, for keeping the modules within a certain time-limit, we have not included a hands-on lab session in this module. You will do cross-validation hands on in R as you learn about these different algorithms in the upcoming modules.

# Summary

In this session, you learnt the most fundamental principles of machine learning which you should now be able to apply while building models. The most important points to re-iterate are:

**Model Simplicity and complexity**

* A model should be as simple as necessary, but no simpler
* When in doubt, choose a simpler model
* Advantages of simplicity are generalisability, robustness, making few assumptions and less data required for learning

**Overfitting**

* A model memorizes the data rather than intelligently learning the underlying trends in it
* It arises because it is possible to memorize data, and this is a problem, because the real test happens on unseen, real world data

**Bias-Variance Tradeoff**

* Bias measures how accurately a model can describe the actual task at hand
* Variance measures how flexible the model is with respect to changes in the training data
* As complexity increases, bias reduces and variance increases, and we aim to find the optimal point where the total model error is the least

**Cross Validation**

* It is a statistical technique which enables us to make extremely efficient use of available data
* It divides the data into several pieces, or 'folds', and uses each piece as test data one at a time

"Model variance" measures the variability in estimates of the model with respect to:

**All possible datasets of a given size, i.e. the variability of the model with respect to different training sets**

*Variance measures how much the model changes with respect to the training data.*

Select the valid reason for why cross validation is required / effective:

**Data is adequate but not abundant**

**:**Data is rarely abundant, and that's when you need to perform cross-validation.

Inconsistent results across the models generated during cross-validation is an indication of (select all the correct options):

**High model variance**

Yes, model variance is high when the model changes with change in training data, which happens in cross validation.

**Model complexity too high**

Complexity leads to high model variance, which results in inconsistent results (drastic changes in the model) with changes in training data.