Overfitting and Underfitting in Machine Learning

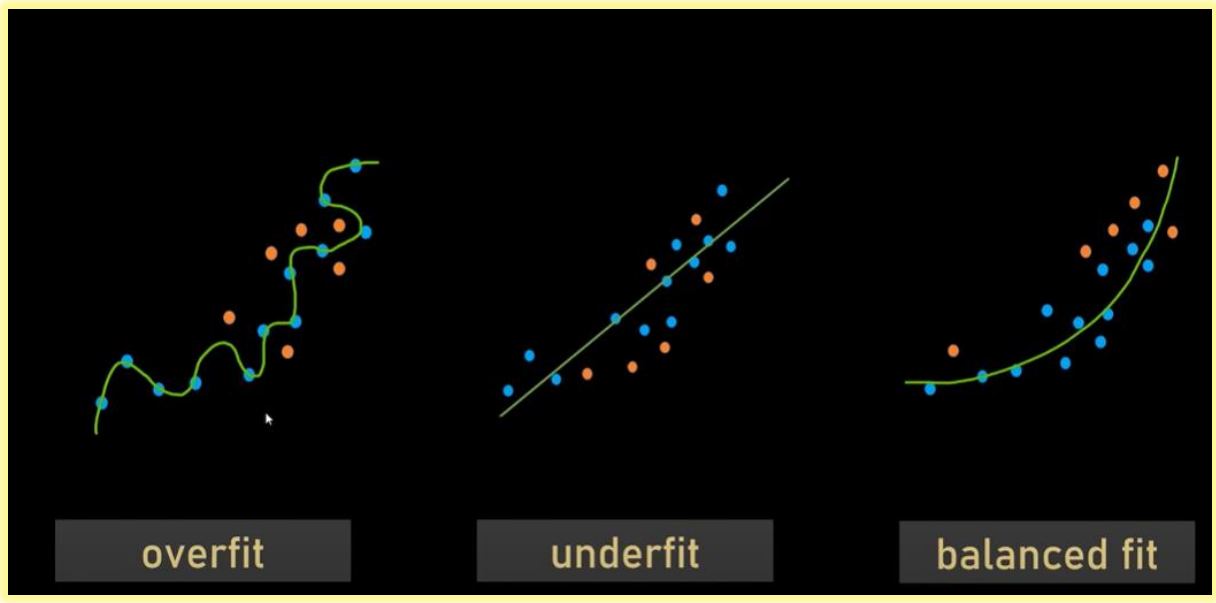
Overfitting and Underfitting are the two main problems that occur in machine learning and degrade the performance of the machine learning models.

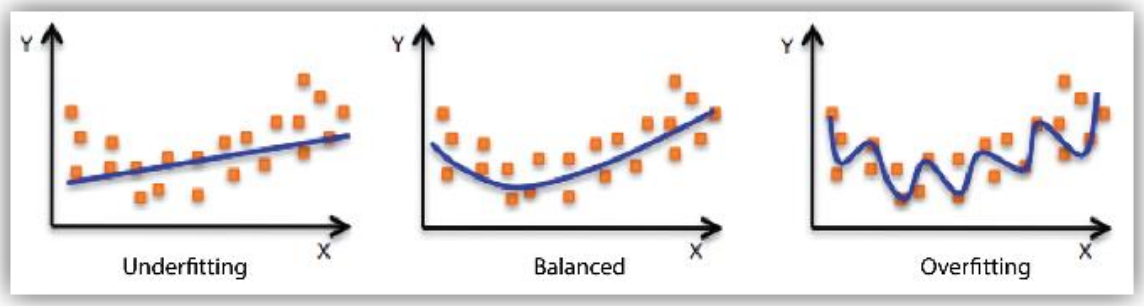
The main goal of each machine learning model is **to generalize well**.
Here **generalization** defines the ability of an ML model to provide a suitable output by adapting the given set of unknown input. It means after providing training on the dataset, it can produce reliable and accurate output. Hence, the underfitting and overfitting are the two terms that need to be checked for the performance of the model and whether the model is generalizing well or not.

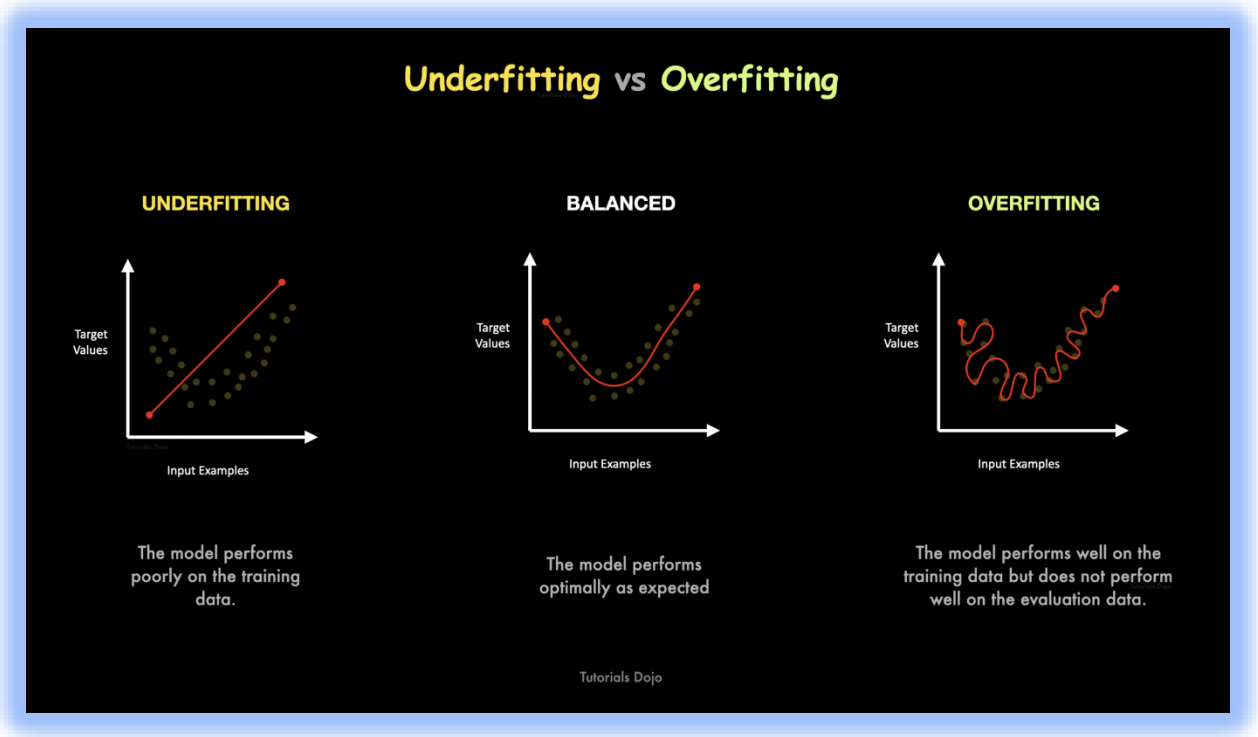
Before understanding the overfitting and underfitting, let's understand some basic term that will help to understand this topic well:

Before diving further let's understand two important terms:

- **Bias:** Assumptions made by a model to make a function easier to learn. It is actually the error rate of the training data. When the error rate has a high value, we call it High Bias and when the error rate has a low value, we call it low Bias.
- **Variance:** The difference between the error rate of training data and testing data is called variance. If the difference is high then it's called high variance and when the difference of errors is low then it's called low variance. Usually, we want to make a low variance for generalized our model.







We can determine whether a predictive model is **underfitting** or **overfitting** the training data by looking at the prediction error on the training data and the evaluation data.

Your model is ***underfitting*** the training data when the model performs poorly on the training data. This is because the model is unable to capture the relationship between the input examples (often called X) and the target values (often called Y).

Your model is ***overfitting*** your training data when you see that the model performs well on the training data but does not perform well on the evaluation data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples.

How to Detect Overfitting and Underfitting

A key challenge with Overfitting and Underfitting with machine learning in general, is that we can't know how well our model will perform on new data until we actually test it.

So to better demonstrate it we can use an example.

Let's say we want to predict if a student will land a job interview based on his/her resume.
Now, assume we train a model from a dataset of 10,000 resumes and their outcomes.
Next, we try the model out on the original dataset, and it predicts outcomes with 99% accuracy… wow!

But now comes the bad news. When we run the model on a new **"unseen"** dataset of resumes,
we only get 50% accuracy… uh-oh!

Our model doesn't generalize well from our training data to unseen data.

Now we know that our model is **Overfitted**.

But if we try the model out on the original dataset, and it predicts outcomes with 55% accuracy or something.

Then our model is **Underfitted**.

Reasons for Overfitting and Underfitting

Overfitting happens when the size of training data used is not enough, or when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have **low bias** and **high variance**. These models are very complex like Decision trees which are prone to overfitting.

While **Underfitting** happens when a model unable to capture the underlying pattern of the data. These models usually have **high bias** and **low variance**. It happens when

we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kind of models are very simple to capture the complex patterns in data like Linear and logistic regression. It can also happen when the size of training data used is not enough.

How to Prevent Overfitting and Underfitting

Detecting overfitting and underfitting is useful, but it doesn't solve the problem. Fortunately, you have several options to try.

Here are a few of the most popular solutions for Overfitting:

- **Cross-validation**
  Cross-validation is a powerful preventative measure against overfitting.
  The idea is clever: Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model.
  In standard k-fold cross-validation,
- **Train with more data**
  It won't work every time, but training with more data can help algorithms detect the signal better.
  Of course, that's not always the case. If we just add more noisy data, this technique won't help. That's why you should always ensure your data is clean and relevant.
- **Remove features**
  Some algorithms have built-in feature selection.
  For those that don't, you can manually improve their generalizability by removing irrelevant input features.
  An interesting way to do so is to tell a story about how each feature fits into the model. This is like the data scientist's spin on software engineer's rubber duck debugging technique, where they debug their code by explaining it, line-by-line, to a rubber duck.

If anything doesn't make sense, or if it's hard to justify certain features, this is a good way to identify them.

- **Regularization**
  Regularization refers to a broad range of techniques for artificially forcing your model to be simpler.
  The method will depend on the type of learner you're using. For example, you could prune a decision tree, use dropout on a neural network, or add a penalty parameter to the cost function in regression.

Oftentimes, the regularization method is a hyperparameter as well, which means it can be tuned through cross-validation.

- **Ensembling**
  Ensembles are machine learning methods for combining predictions from multiple separate models. There are a few different methods for ensembling, but the two most common are:
  - **Bagging** attempts to reduce the chance overfitting complex models.
  - **Boosting** attempts to improve the predictive flexibility of simple models.

Now for the most popular solutions for Underfitting:

- **Increase model complexity**
  As model complexity increases, performance on the data used to build the model (training data) improves.
- **Increase number of features, performing feature engineering**
- **Remove noise from the data**
- **Train with more data** Same as Overfitting.

Balanced Model