

Underfitting vs. Overfitting (vs. Best Fitting) in Machine Learning

The Challenge of Underfitting and Overfitting in Machine Learning

Can you explain what is underfitting and overfitting in the context of machine learning? Describe it in a way even a non-technical person will grasp.

Your ability to explain this in a non-technical and easy-to-understand manner might well decide your fit for the data science role!

Even when we're working on a machine learning project, we often face situations where we are encountering unexpected performance or error rate differences between the training set and the test set (as shown below). How can a model perform so well over the training set and just as poorly on the test set?

```
1 from sklearn.metrics import classification_report
2 print(classification_report(y_train, predicted_values))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14234
1	1.00	1.00	1.00	3419
accuracy			1.00	17653
macro avg	1.00	1.00	1.00	17653
weighted avg	1.00	1.00	1.00	17653

```
1 predicted_values = classifier.predict(x_test)
2 print(classification_report(y_test, predicted_values))
```

	precision	recall	f1-score	support
0	0.87	0.86	0.87	3559
1	0.44	0.46	0.45	855
accuracy			0.78	4414
macro avg	0.66	0.66	0.66	4414
weighted avg	0.79	0.78	0.79	4414

This happens very frequently whenever I am working with tree-based predictive models. Because of the way the algorithms work, you can imagine how tricky it is to avoid falling into the overfitting trap!

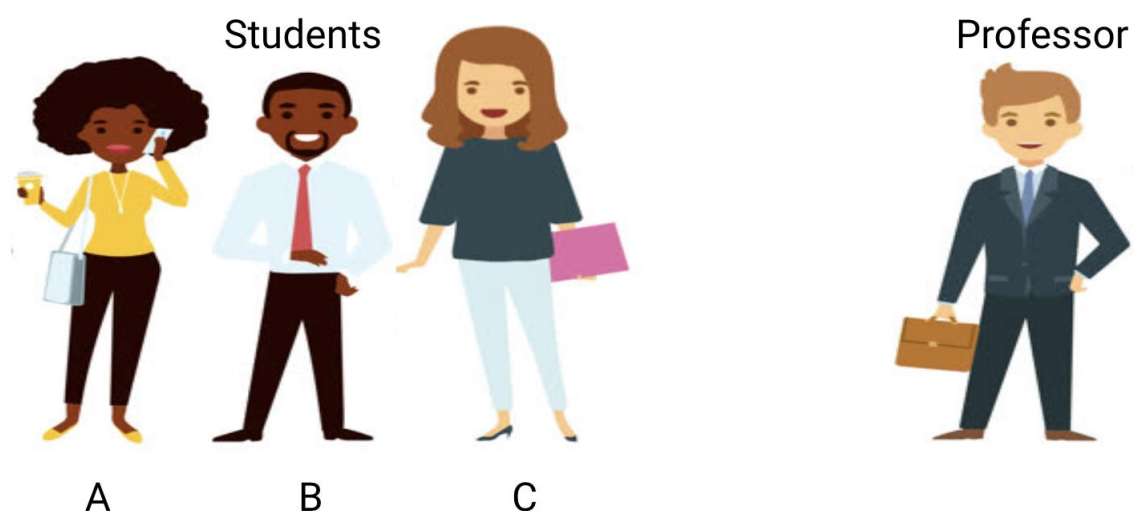
Moreover, it can be quite daunting when we are unable to find the underlying reason why our predictive model is exhibiting this anomalous behavior.

Here's my personal experience – ask any seasoned data scientist about this, they typically start talking about some array of fancy terms like Overfitting, Underfitting, Bias, and Variance. But little does anyone talk about the intuition behind these machine learning concepts. Let's rectify that, shall we?

Let's Take an Example to Understand Underfitting vs. Overfitting

I want to explain these concepts using a real-world example. A lot of folks talk about the theoretical angle but I feel that's not enough – we need to visualize how underfitting and overfitting actually work.

So, let's go back to our college days for this.



Consider a math class consisting of 3 students and a professor.

Now, in any classroom, we can broadly divide the students into 3 categories. We'll talk about them one-by-one.



A

- Hobby = chatting
- Not interested in class
- Doesn't pay much attention to professor

Let's say that student A resembles a student who does not like math. She is not interested in what is being taught in the class and therefore does not pay much attention to the professor and the content he is teaching.



B

- Hobby = to be best in class.
- Mugs up everything professor says.
- Too much attention to the class work.

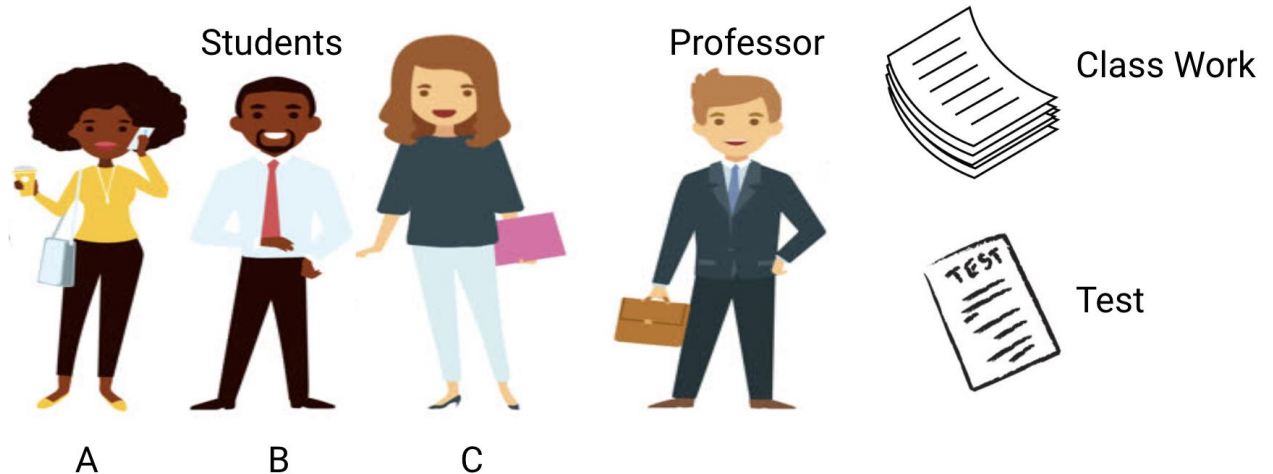
Let's consider student B. He is the most competitive student who focuses on memorizing each and every question being taught in class instead of focusing on the key concepts. Basically, he isn't interested in learning the problem-solving approach.



C

- Hobby = learning new things
- Eager to learn concepts.
- Pays attention to class and learns the idea behind solving a problem.

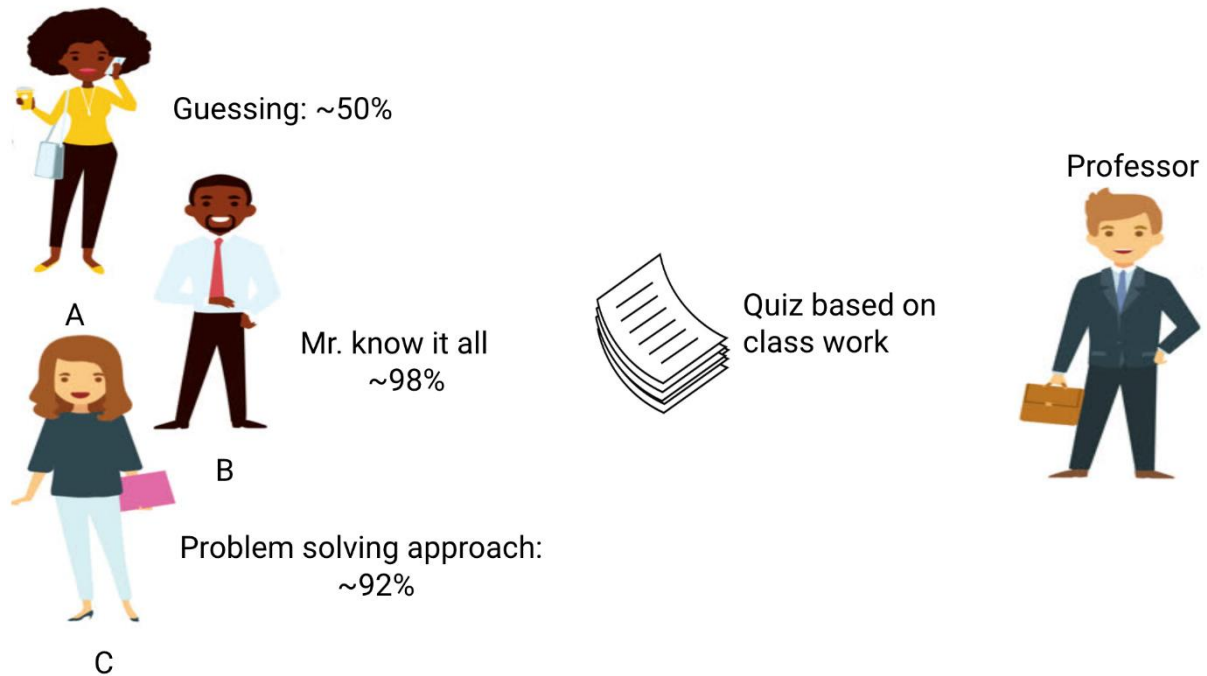
Finally, we have the ideal student C. She is purely interested in learning the key concepts and the problem-solving approach in the math class rather than just memorizing the solutions presented.



We all know from experience what happens in a classroom. The professor first delivers lectures and teaches the students about the problems and how to solve them. At the end of the day, the professor simply takes a quiz based on what he taught in the class.

The obstacle comes in the semester3 tests that the school lays down. This is where new questions (unseen data) comes up. The students haven't seen these questions before and certainly haven't solved them in the classroom. Sounds familiar?

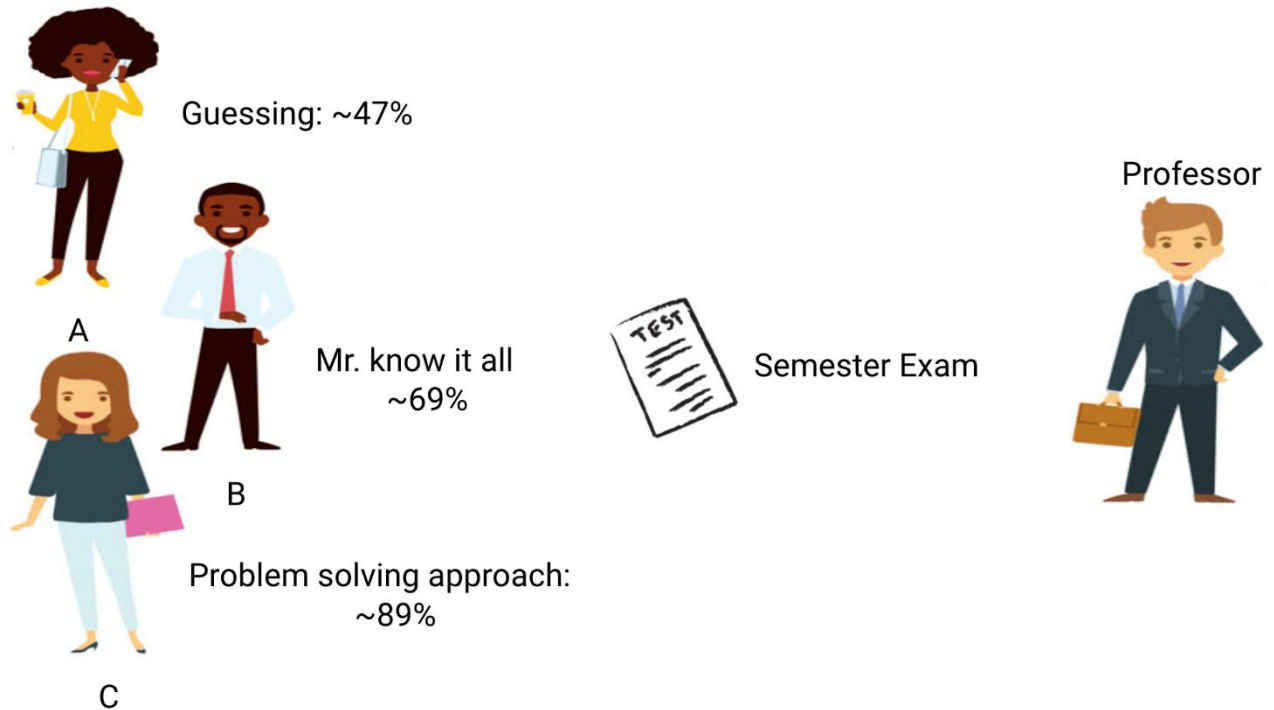
So, let's discuss what happens when the teacher takes a classroom test at the end of the day:



- Student A, who was distracted in his own world, simply guessed the answers and got approximately 50% marks in the test
- On the other hand, the student who memorized each and every question taught in the classroom was able to answer almost every question by memory and therefore obtained 98% marks in the class test
- For student C, she actually solved all the questions using the problem-solving approach she learned in the classroom and scored 92%

We can clearly infer that the student who simply memorizes everything is scoring better without much difficulty.




Now here's the twist. Let's also look at what happens during the monthly test, when students have to face new unknown questions which are not taught in the class by the teacher.



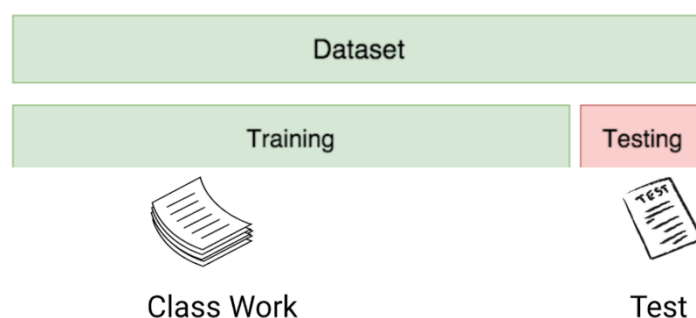
- In the case of student A, things did not change much and he still randomly answers questions correctly ~50% of the time.
- In the case of Student B, his score dropped significantly. Can you guess why? This is because he always memorized the problems that were taught in the class but this monthly test contained questions which he has never seen before. Therefore, his performance went down significantly
- In the case of Student C, the score remained more or less the same. This is because she focused on learning the problem-solving approach and therefore was able to apply the concepts she learned to solve the unknown questions

How Does this Relate to Underfitting and Overfitting in Machine Learning?

You might be wondering how this example relates to the problem which we encountered during the train and test scores of the decision tree classifier? Good question!

		
A	B	C
Not interested in learning	Memorizing the lessons	Conceptual Learning
Class test ~50%	Class test ~98%	Class test ~92%
Test ~47%	Test ~69%	Test ~89%

So, let's work on connecting this example with the results of the decision tree classifier that I showed you earlier.



First, the classwork and class test resemble the training data and the prediction over the training data itself respectively. On the other hand, the semester test represents the test set from our data which we keep aside before we train our model (or unseen data in a real-world machine learning project).

Now, recall our decision tree classifier I mentioned earlier. It gave a perfect score over the training set but struggled with the test set. Comparing that to the student examples we just discussed, the classifier

establishes an analogy with student B who tried to memorize each and every question in the training set.

Similarly, our decision tree classifier tries to learn each and every point from the training data but suffers radically when it encounters a new data point in the test set. It is not able to generalize it well.



A

Not interested in learning

Class test ~50%
Test ~47%

Under-fit/ biased learning



B

Memorizing the lessons

Class test ~98%
Test ~69%

Over-fit/ Memorizing



C

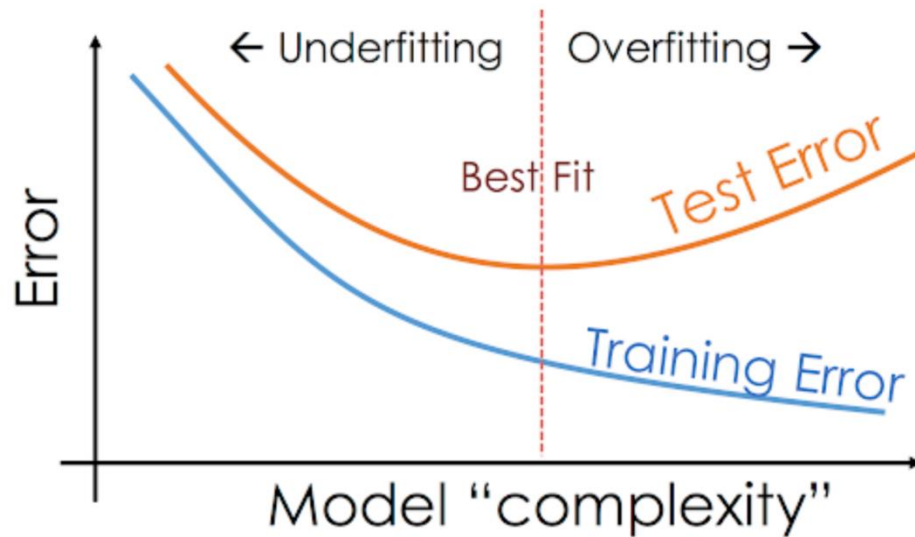
Conceptual Learning

Class test ~92%
Test ~89%

Best-fit

This situation where any given model is performing too well on the training data but the performance drops significantly over the test set is called an overfitting model.

For example, non-parametric models like decision trees, KNN, and other tree-based algorithms are very prone to overfitting. These models can learn very complex relations which can result in overfitting. The graph below summarises this concept:



On the other hand, if the model is performing poorly over the test and the train set, then we call that an underfitting model. An example of this situation would be building a linear regression model over non-linear data.

What Are Overfitting and Underfitting in Machine Learning?

As you enter the realm of Machine Learning, several ambiguous terms will introduce themselves. Terms like Overfitting, Underfitting, and bias-variance trade-off. These concepts lie at the core of the field of Machine Learning in general. In this post, I explain those terms with an example.

Why should we even care?

Arguably, Machine Learning models have one sole purpose; to generalize well. I have mentioned this in several previous posts, but it never hurts to emphasize on it.

Generalization is the model's ability to give sensible outputs to sets of input that it has never seen before.

Normal programs cannot do such a thing, as they can only give outputs “robotically” to the inputs they know. Performance of the model as well as the application as a whole relies heavily on the generalization of the model. If the model generalizes well, it

serves its purpose. A lot of techniques to evaluate this performance have been introduced, [starting with the data itself](#).

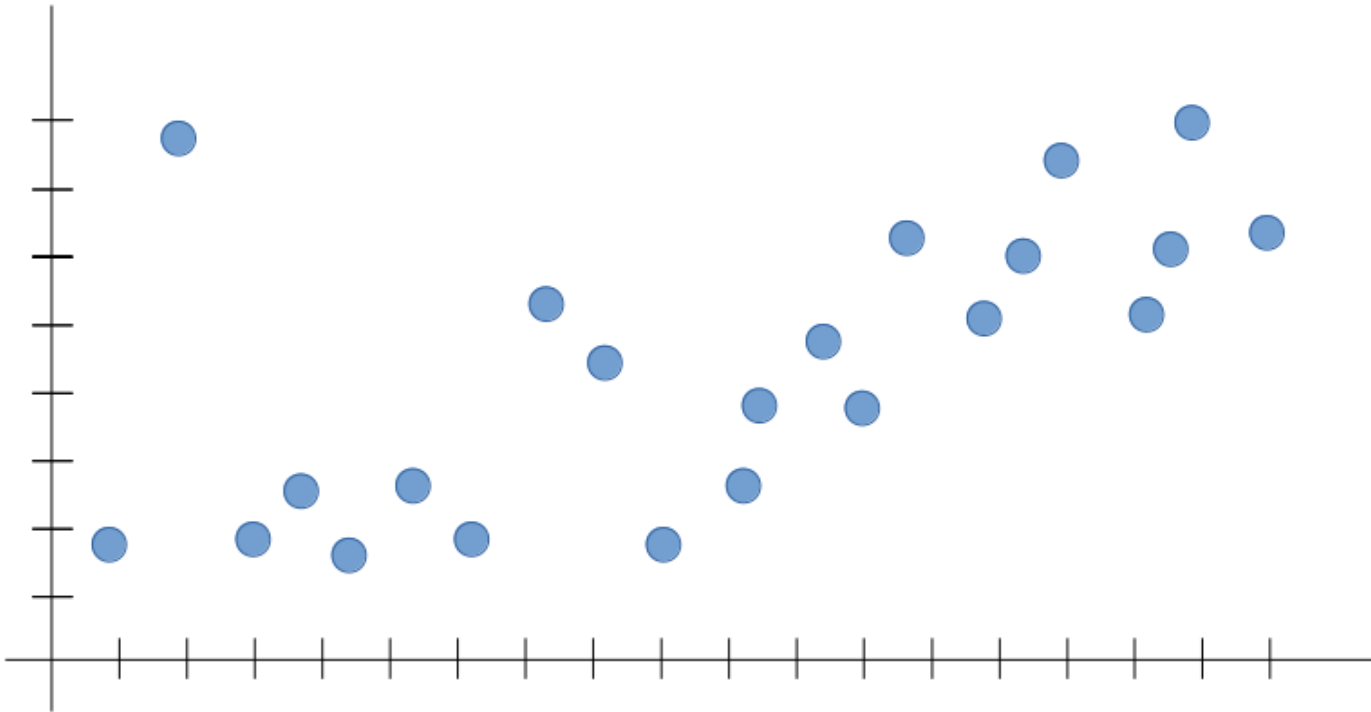
Building on that idea, terms like **overfitting and underfitting refer to deficiencies that the model's performance might suffer from**. This means that knowing “how off” the model's predictions are is a matter of knowing how close it is to overfitting or underfitting.

A model that generalizes well is a model that is neither underfit nor overfit.

This may not make that much sense yet, but I need you to keep this sentence in mind throughout this post, as it's the big picture regarding our topic. The rest of the post will make links between whatever you learn and how it fits within this big picture.

Our Example

Let's say we're trying to build a Machine Learning model for the following data set.



Please take note that I believe newcomers in the field should have more hands-on experience than research. Therefore, mathematical technicalities like the functions involved and the such will not be touched on in this post. For now, let's just keep in mind that the x-axis is the input value and y-axis is the output value in the data set.

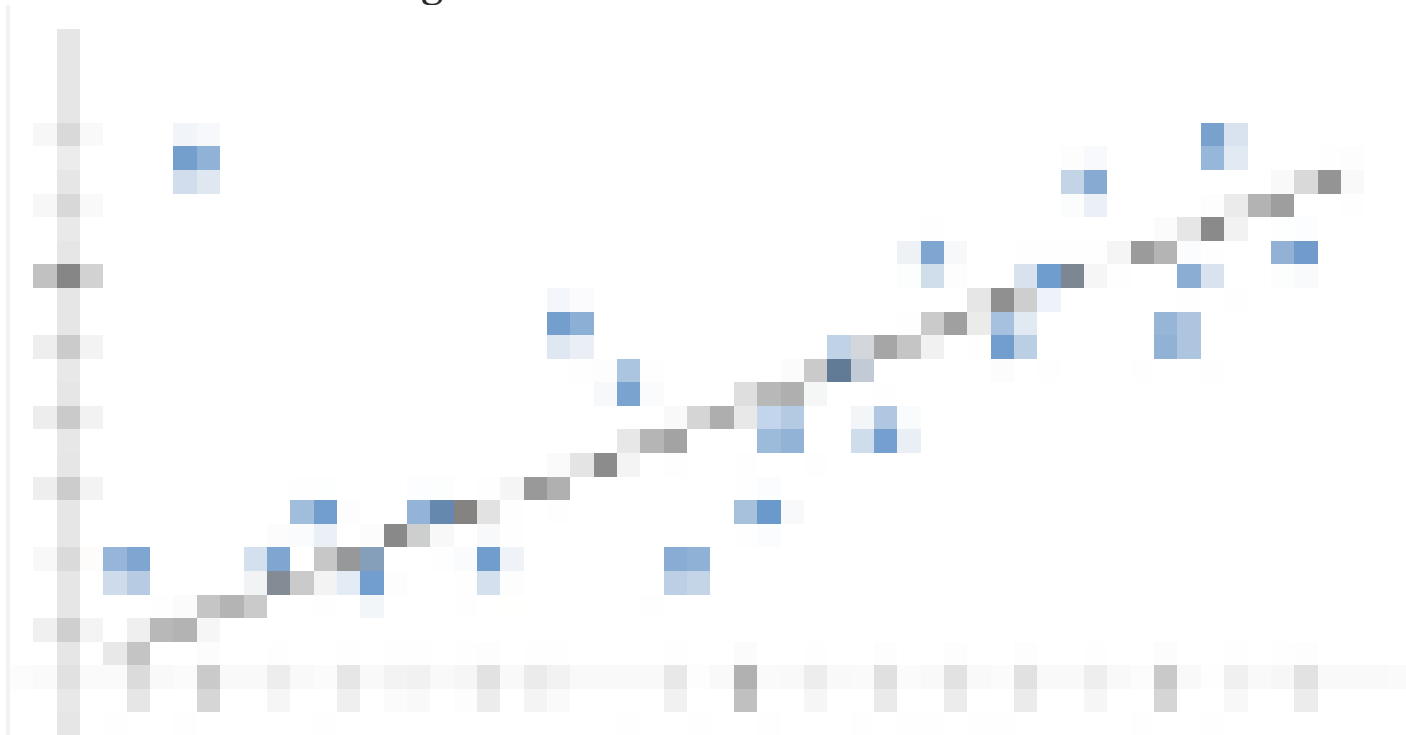
If you've had any previous experience with Machine Learning model training, you probably know that we have a few options here. However, for simplicity reasons, let's choose [Univariate Linear Regression](#) in our example. **Linear Regression allows us to map numeric inputs to numeric outputs, fitting a**

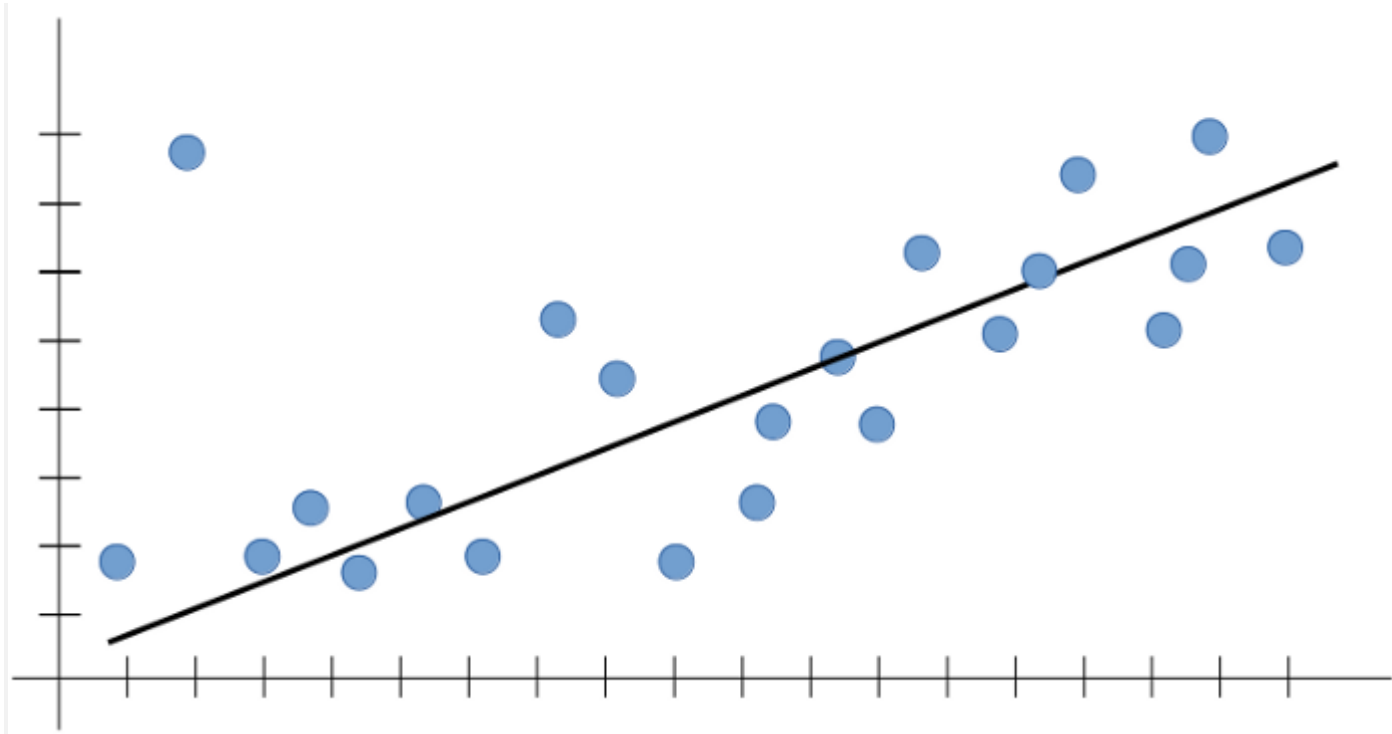
line into the data points. This line-fitting process is the medium of both overfitting and underfitting.

The training stage

Training the Linear Regression model in our example is all about minimizing the total distance (i.e. cost) between the line we're trying to fit and the actual data points. This goes through multiple iterations until we find the relatively “optimal” configuration of our line within the data set. This is exactly where overfitting and underfitting occur.

In Linear Regression, we would like our model to follow a line similar to the following:





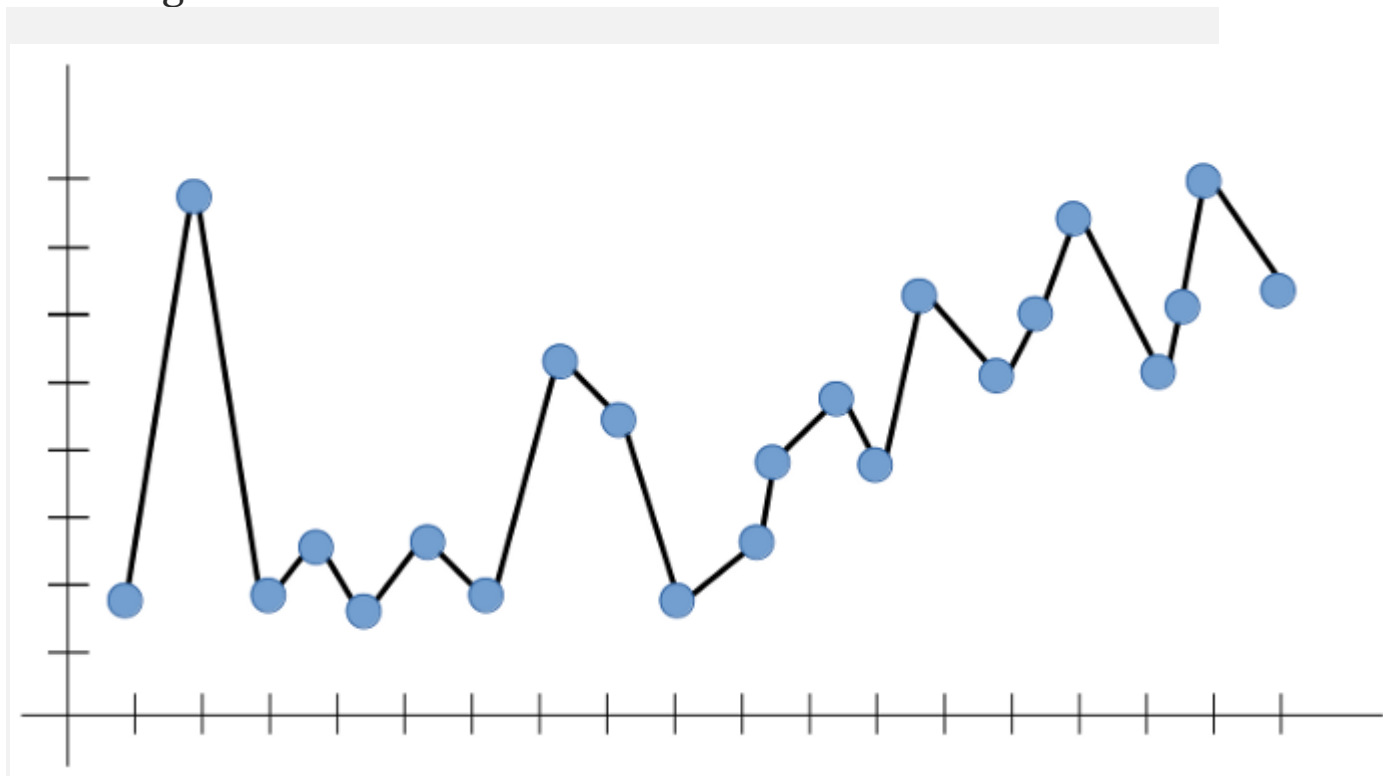
Even though the overall cost is not minimal (i.e. there is a better configuration in which the line could yield a smaller distance to the data points), the line above fits within the trend very well, making the model reliable. Let's say we want to infer an output for an input value that is not currently resident in the data set (i.e. generalize). **The line above could give a very likely prediction for the new input, as, in terms of Machine Learning, the outputs are expected to follow the trend seen in the training set.**

Overfitting

When we run our training algorithm on the data set, we allow the overall cost (i.e. distance from each point to the line) to

become smaller with more iterations. Leaving this training algorithm run for long leads to minimal overall cost. However, this means that the line will be fit into all the points (including noise), catching secondary patterns that may not be needed for the generalizability of the model.

Referring back to our example, if we leave the learning algorithm running for long, it could end up fitting the line in the following manner:



This looks good, right? Yes, but is it reliable? Well, not really.

The essence of an algorithm like Linear Regression is to capture the dominant trend and fit our line within that trend. In the

figure above, the algorithm captured all trends — but not the dominant one. If we want to test the model on inputs that are beyond the line limits we have (i.e. generalize), what would that line look like? There is really no way to tell. Therefore, the outputs aren't reliable. **If the model does not capture the dominant trend that we can all see (positively increasing, in our case), it can't predict a likely output for an input that it has never seen before — defying the purpose of Machine Learning to begin with!**

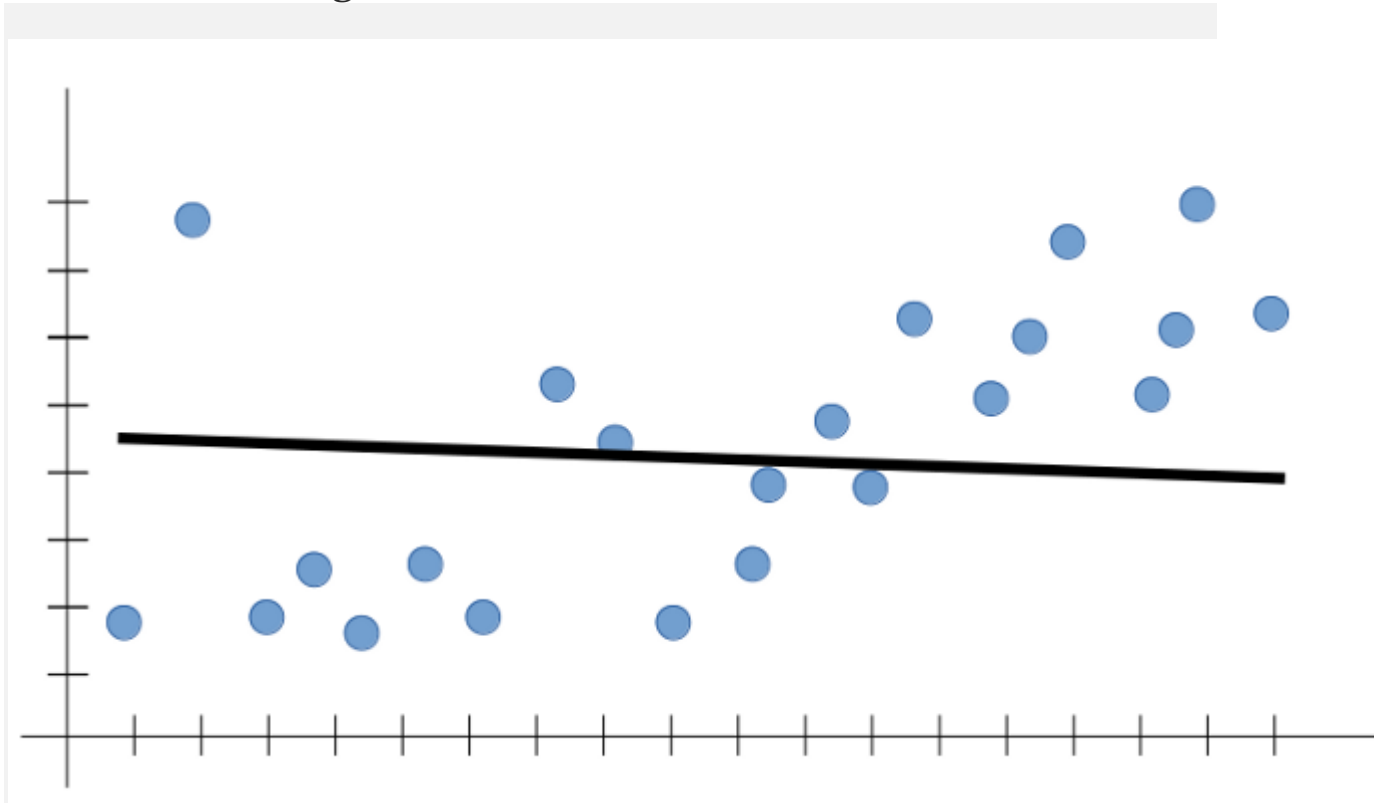
Overfitting is the case where the overall cost is really small, but the generalization of the model is unreliable. This is due to the model learning “too much” from the training data set.

This may sound preposterous, as why would we settle for a higher cost when we can just find the minimal one?
Generalization.

The more we leave the model training the higher the chance of overfitting occurring. **We always want to find the trend, not fit the line to all the data points.** Overfitting (or high variance) leads to more bad than good. What use is a model that has learned very well from from the training data but still can't make reliable predictions for new inputs?

Underfitting

We want the model to learn from the training data, but we don't want it to learn too much (i.e. too many patterns). One solution could be to stop the training earlier. However, this could lead the model to not learn enough patterns from the training data, and possibly not even capture the dominant trend. This case is called underfitting.



Underfitting is the case where the model has “not learned enough” from the training data,

resulting in low generalization and unreliable predictions.

As you probably expected, underfitting (i.e. high bias) is just as bad for generalization of the model as overfitting. In high bias, the model might not have enough flexibility in terms of line fitting, resulting in a simplistic line that does not generalize well.

Bias-variance trade-off

So what is the right measure? Depending on the model at hand, a performance that lies between overfitting and underfitting is more desirable. This trade-off is the most integral aspect of Machine Learning model training. As we discussed, Machine Learning models fulfill their purpose when they generalize well. Generalization is bound by the two undesirable outcomes — high bias and high variance. Detecting whether the model suffers from either one is the sole responsibility of the model developer.