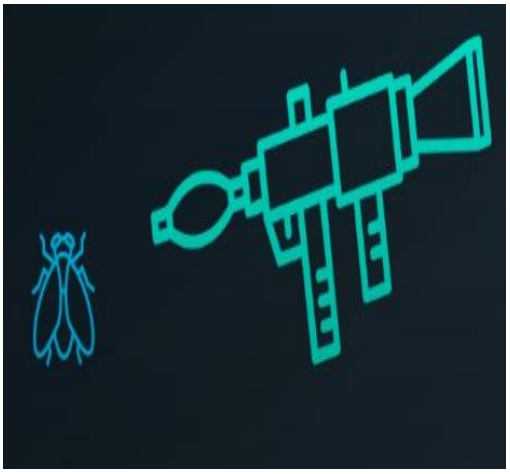
Training and Tuning Lesson:

Types of Errors

There are two types of errors:

1. **OVERFITTING**: it is overcomplicating the problem we are trying to solve, it is like trying to kill Godzilla with a flyswatter.

2. **UNDERFITTING**: it is oversimplifying. the problem we are trying to solve, it is like trying to kill a fly with a bazooka.



In machine learning, these two errors are very easy to make.

Let us take this classification example, we want to find a property that separates the set on the left from the set on the right.

What if we say the right contains animals, and the left contains other things? That would be oversimplifying (**UNDERFITTING**), we call this type of error (**Error due to bias**).

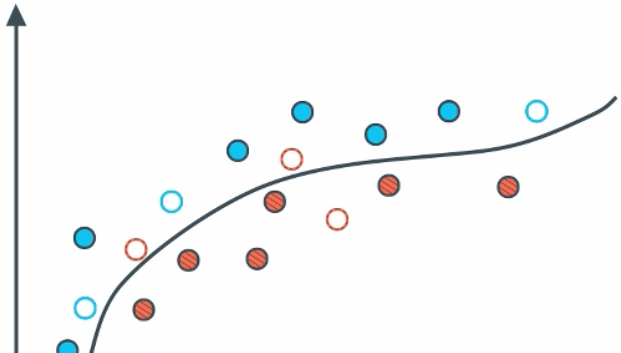
UNDERFITTING does poorly with both training and testing sets.

What if we say the right contains dogs wagging their tails, and others on the left? this would work on our training set but fail immediately once we put a dog who does not wag its tail, this model is too specific (**OVERFITTING**), this type of error is called (**error due to variance**).

OVERFITTING Errors does well with training set, but not testing set.

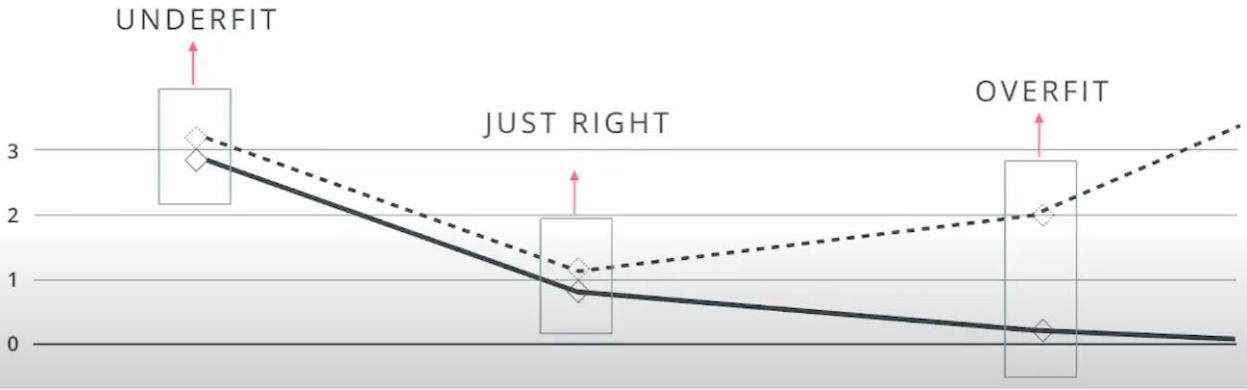
And the good model here is the one who put dogs on the right, and everything else on the left.

Example:



Number of training errors: 0

Number of testing errors: 2



UNDERFIT Models gives us high training and testing errors.

OVERFIT Models give us very low training but high testing error (memorize the training data).

Models in the middle is just right and gives us low training and testing errors.

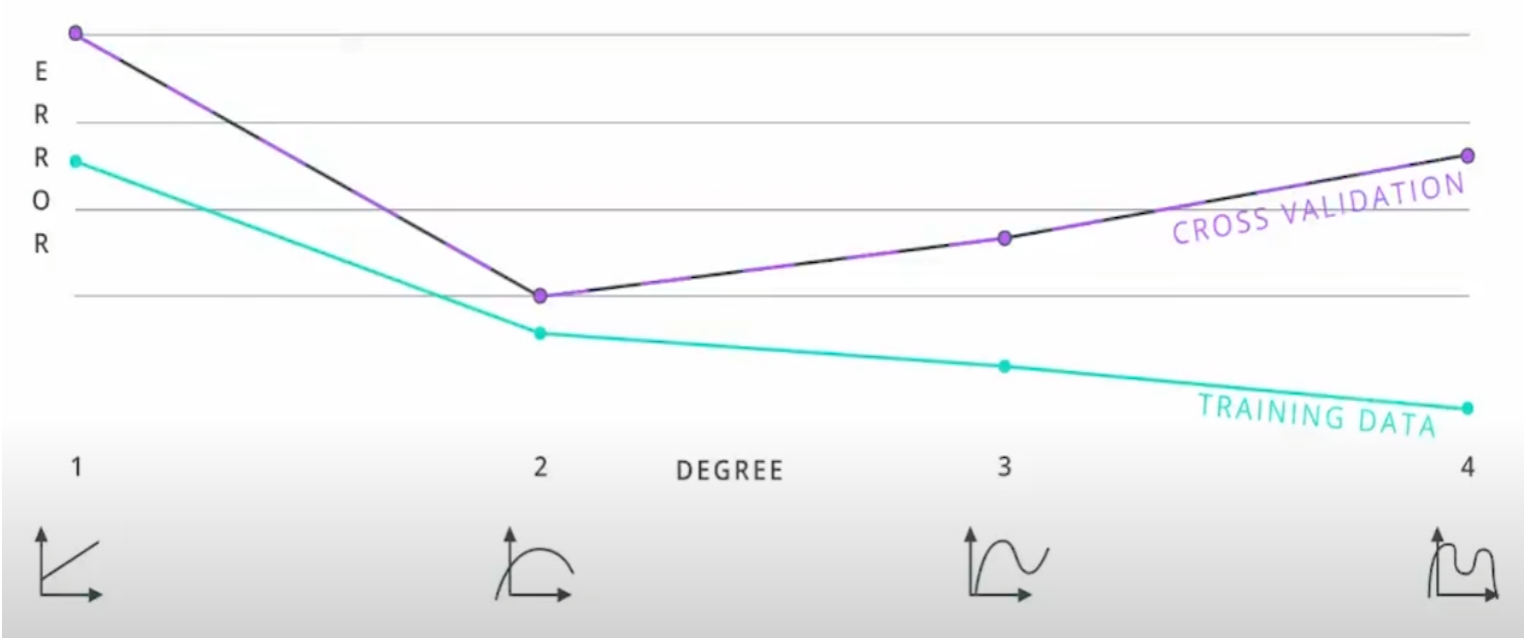
To determine the complexity of our model we need to split our data even more,

We need to have **Croess Validation** Data along with **Training** and **Testing**.

The Reason is to be able to determine the complexity without breaking the **Golden Rule** (“Never Use your testing data for training”).

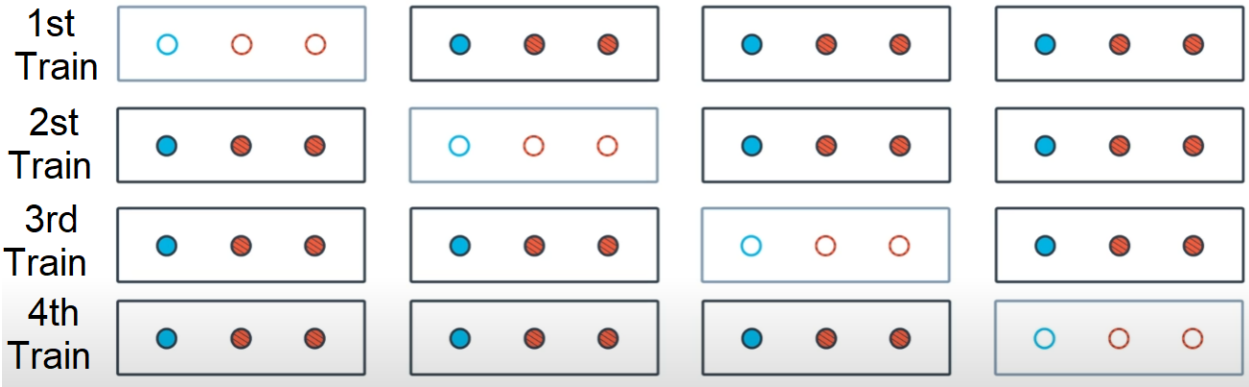
Now to determine which polynomial degree is fit without model the most using the graph underneath, (1 degree **“Linear”**, 2 degrees, 3 degrees or 4 degree).

We can see in this graph it is **UNDERFIT** in 1 degree, **OVERFIT** in the 4 degrees, and the best point is in 2 degrees.



K-Fold Cross Validation

When separate our data to test and training data it seems that we are wasting a lot of data that we could use for training our algorithm, there is a way to use these data without cheating, and it is **K-FOLD Cross Validation.**



The empty points are testing points,

The filled points are training.

What we do is break our data into "K" Buckets, in the example above K=4,

then we train our model K times and in each time using different bucket as testing set and use the remaining buckets as a training set,

finally, we get calculate the average of the results to get our final model.

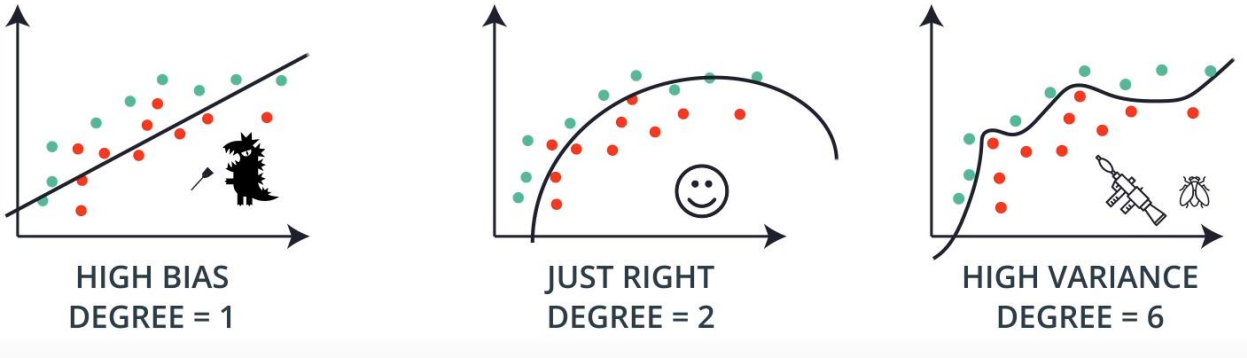
In sklearn it is done like this:

```
from sklearn.model_selection import KFold
kf = KFold(12, 3)                    ← parameters are (size of data , size of testing set)
for trian_indices, test_indices in kf:
    print train_indices, test_indices
```

OUTPUT:

```
[3 4 4 6 7 8 9 10 11] [0 1 2]
[0 1 2 6 7 8 9 10 11] [3 4 5]
[0 1 2 3 4 5 9 10 11] [6 7 8]
[0 1 2 3 4 5 6 4  8 ] [9 10 11]
```

Learning Curves



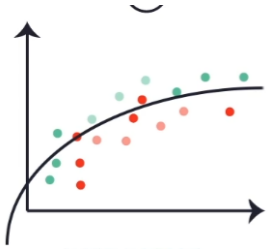
In this section we will learn a way to tell **OVERFITTING**, **UNDERFITTING** and **Good Models.**

The way is to train the model with very few points and increase the number of points gradually.

CV Error = Cross-Validation Error

In **UNDERFITTING Model**, the training error will start small and keep increasing, and the CV Error will start large and keep decreasing until maybe they convert at some point, the point where they convert will be a high point anyway, since this model is **UNDERFITTING,** we don't expect it to have small error.

In the **Good Model,** it is similar to the **UNDERFITTING** Model, except when the Training and CV Error converge it will be at a lower point, since the model is good, and we expect it to have a small error.

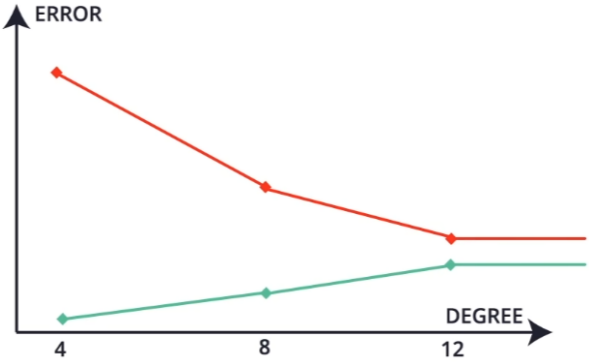


JUST RIGHT
DEGREE = 2
(parabolas)

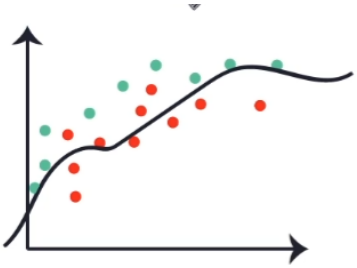
With 4 Training Points
Training Error: Tiny
CV Error: Large

With 8 Training Points
Training Error: Small
CV Error: Medium

With 12 Training Points
Training Error: Small
CV Error: Small



With **OVERFITTING Model**, the training error will never grow too large, and the CV Error will never get too low, which means the two lines will never get close to each other.

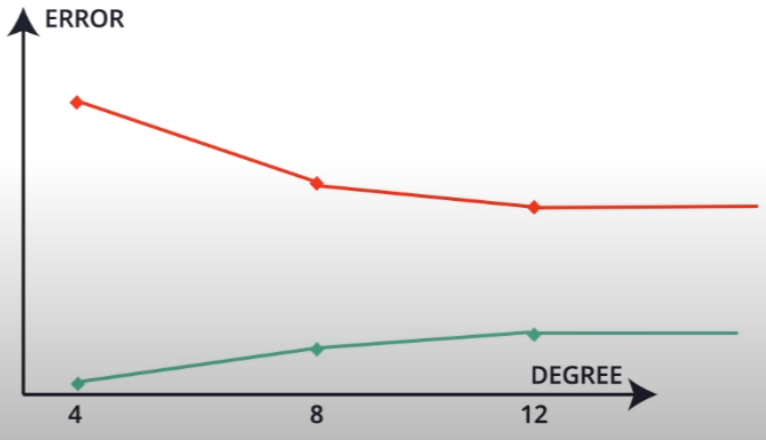


HIGH BIAS
DEGREE = 12 (CURVES)

With 4 Training Points
Training Error: Tiny
CV Error: Large

With 8 Training Points
Training Error: Small
CV Error: Large

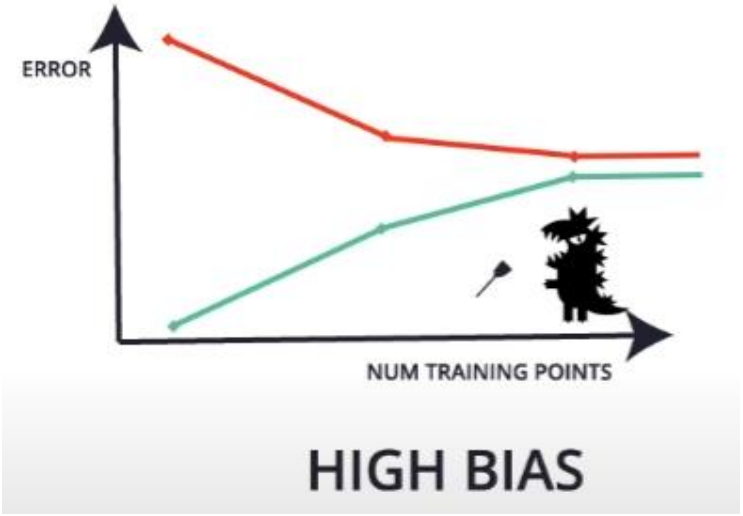
With 12 Training Points
Training Error: Tiny
CV Error: Large



In summary:

In **HIGH BIAS** or **UNDERFITTING,**

The curves get close to each other and converge to a high point.

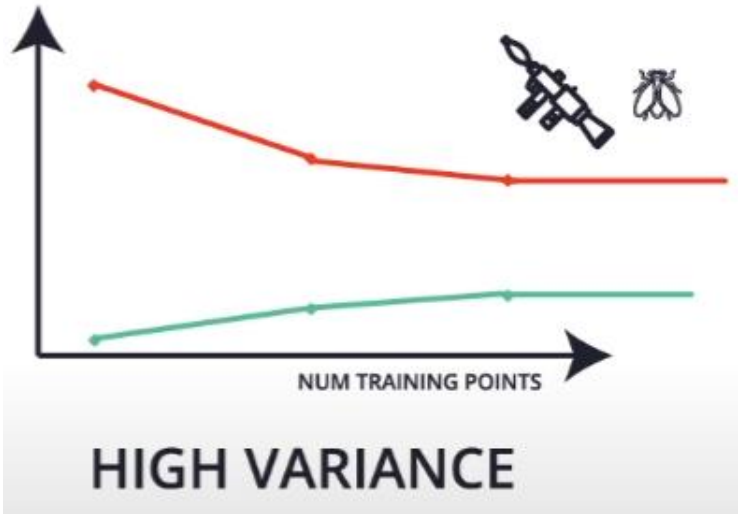


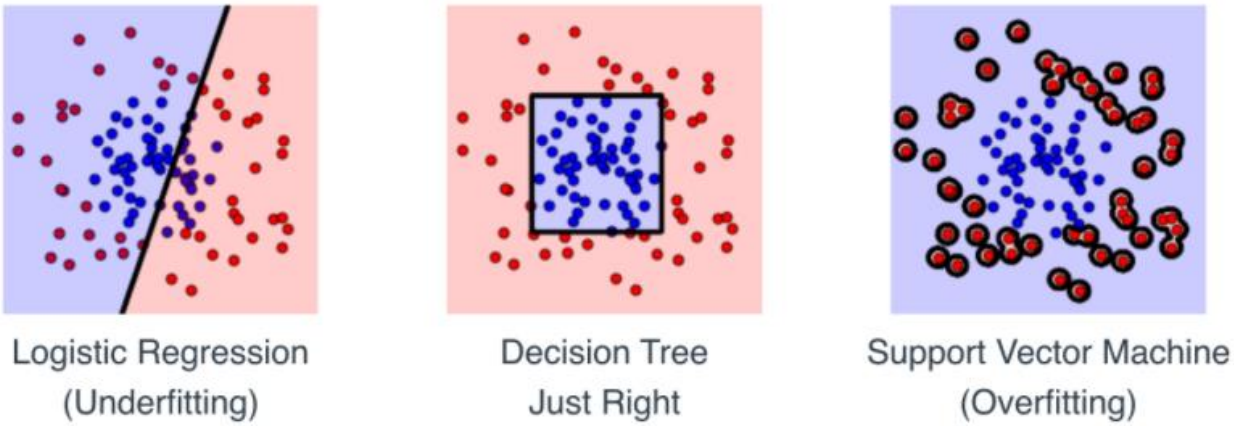
In **Good Model,**

The curves get close to each other and converge to a low point.

In **HIGH VARIANCE** or **OVERFIT,**

The curves do not get close to Each other.



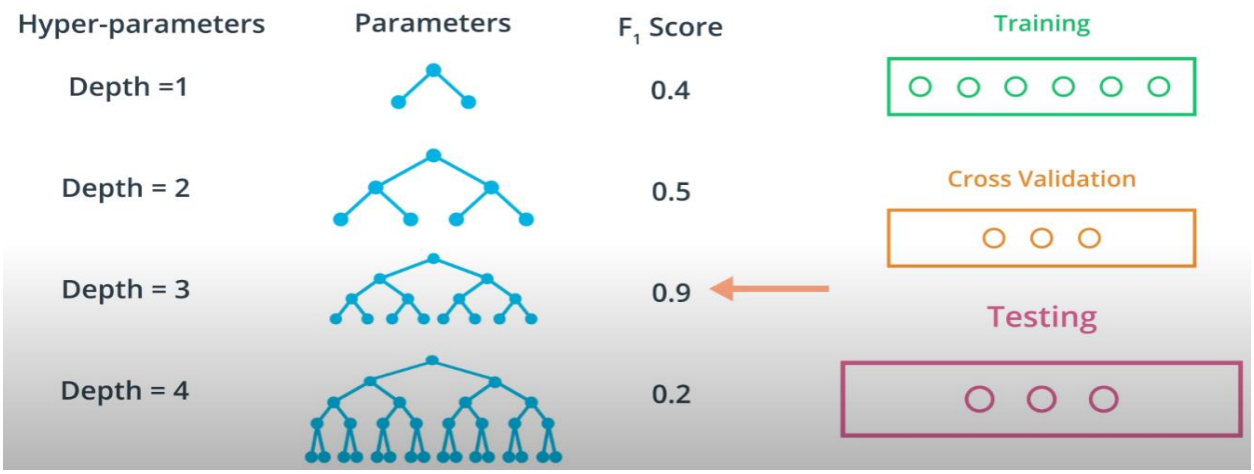


When we look at the models above, does it make? It does, right? We can see that the data is correctly bounded by a circle, or a square. What our models do, is the following:

- The **Logistic Regression model** uses a line, which is too simplistic. It doesn't do very well on the training set. Thus, it underfits.
- The **Decision Tree model** uses a square, which is a pretty good fit, and generalizes well. Thus, this model is good.
- The **Support Vector Machine model** actually draws a tiny circle around each point. This is clearly just memorizing the training set, and won't generalize well. Thus, it overfits.

GRID SEARCH

This example is with **decision tree**, here we train depth 1, 2, 3 ,4 with training set, then calculate F1 using CV set, and take the highest F1 score as the best, and finally make sure the model is good with testing set.



Here we have only 1 **hyper-parameter** (Depth), But what if we have more than one? Let take a **SUPPORT VECTOR MACHINE** example:

We have **SVC** with two hyper parameters (Kernel and C)

What we should do is **grid search**, and the way is to make a table with all possibility and take the best one.

We train everyone, calculate F1 score, and take the best and test it.