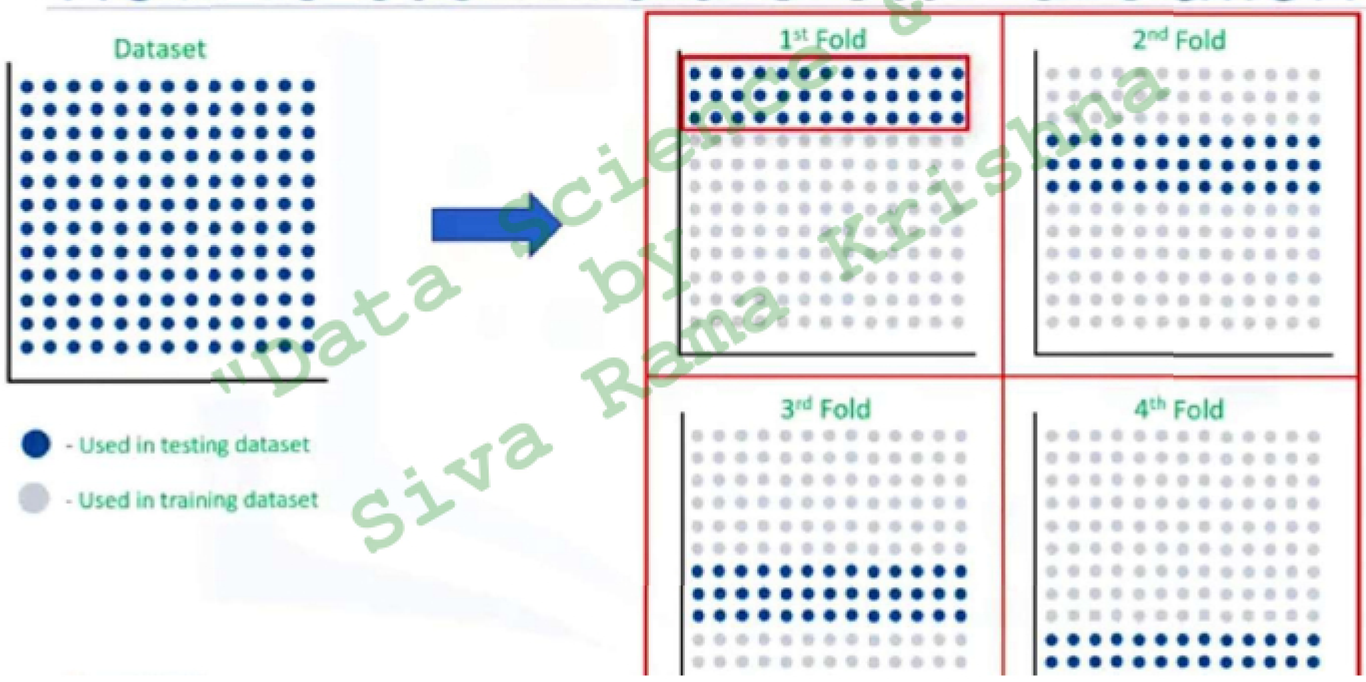
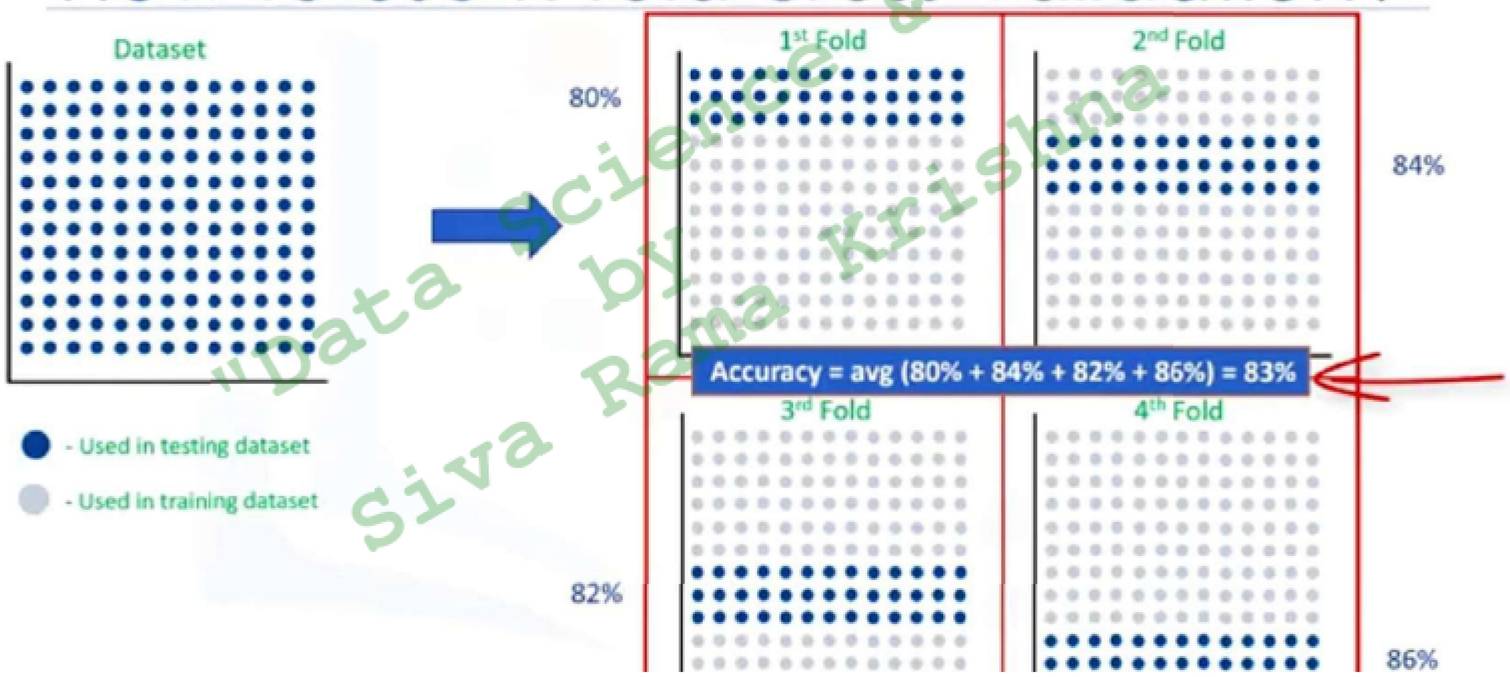
How to use K-fold cross-validation?



How to use K-fold cross-validation?



Accuracy = avg (80% + 84% + 82% + 86%) = 83%

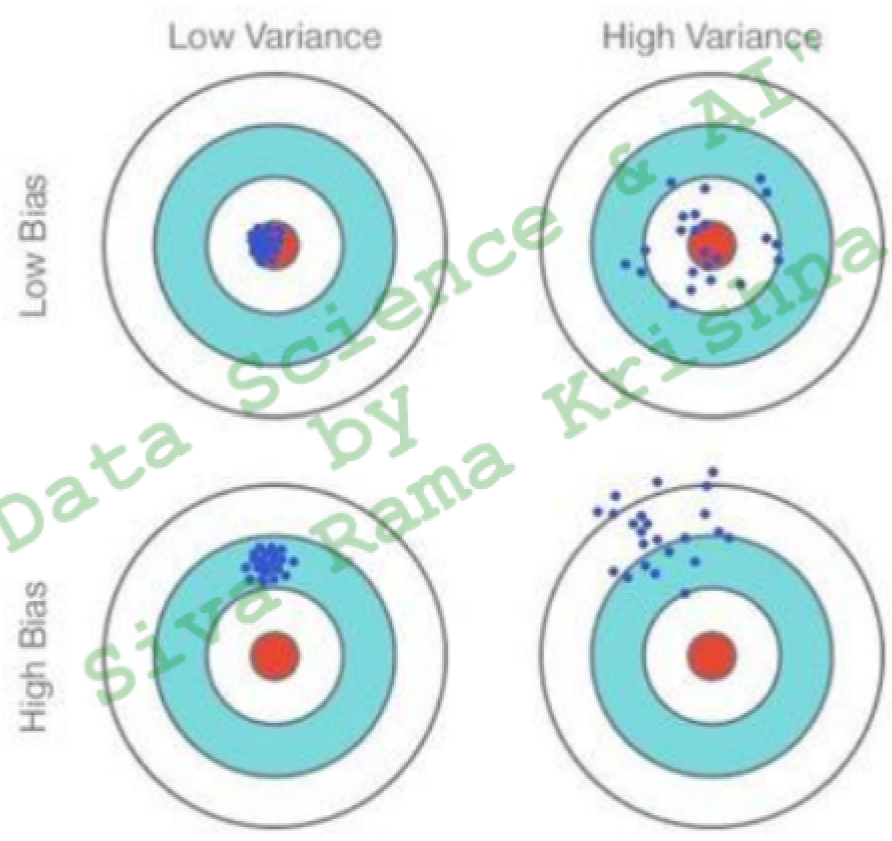
Bias-Variance Trade Off

Overfitting versus Underfitting

- In general, increasing model complex search for better performance leads to a **Bias-Variance trade-off**.
- We want to have a model that can generalize well to new unseen data, but can also account for variance and patterns in the known data.
- Extreme bias or extreme variance both lead to bad models.
- We can visualize this effect by considering a model that underfits (high bias) or a model that overfits (high variance).

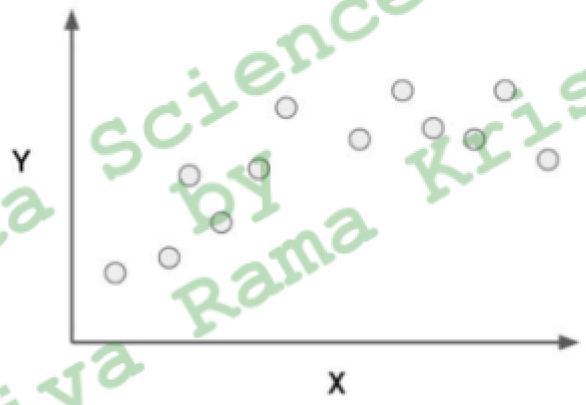
Bias - Our model will not be trained well with the training data. There will be high training error when we train our model with the data.

Variance – If you train your data on training data and obtain a very low error, upon changing the data and then training the same previous model you experience high error, this is variance.

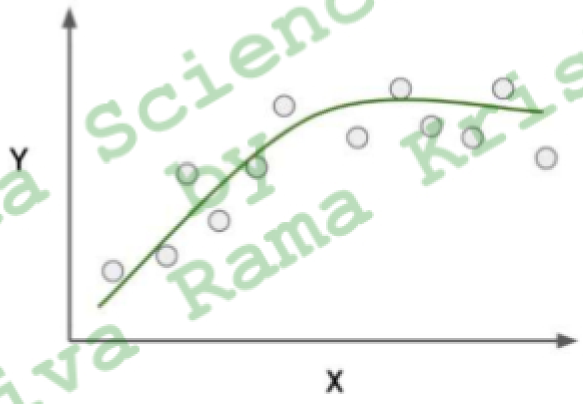

Low Variance
High Variance
Low Bias
High Bias

- **Overfitting**
  - The model fits too much to the noise from the data.
  - This often results in **low error on training sets but high error on test/validation sets.**

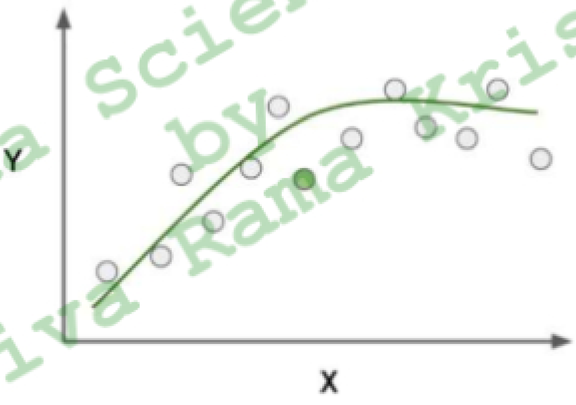
Data



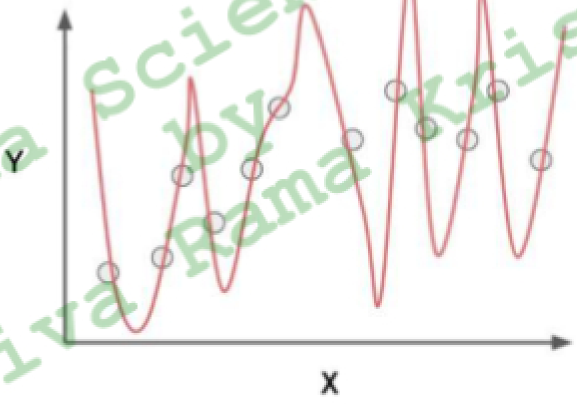
Good Model



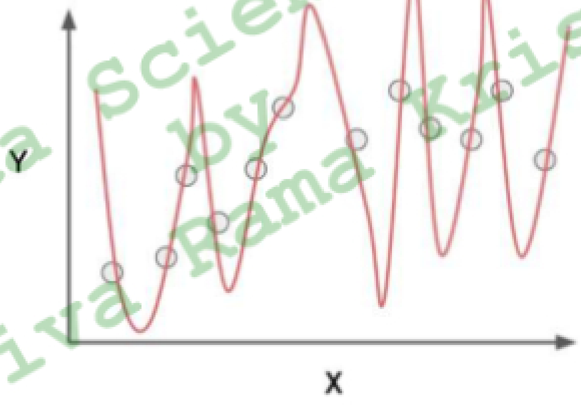
Good Model



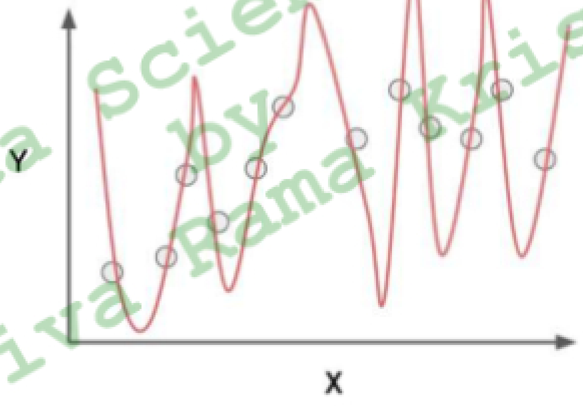
- **What is the error on the training data here?**



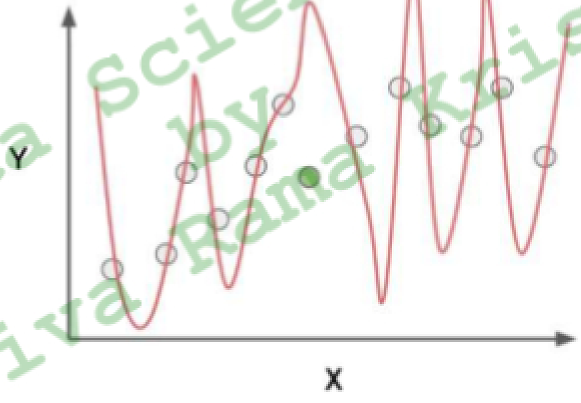
- **Error on train is zero! Model fits perfectly!**



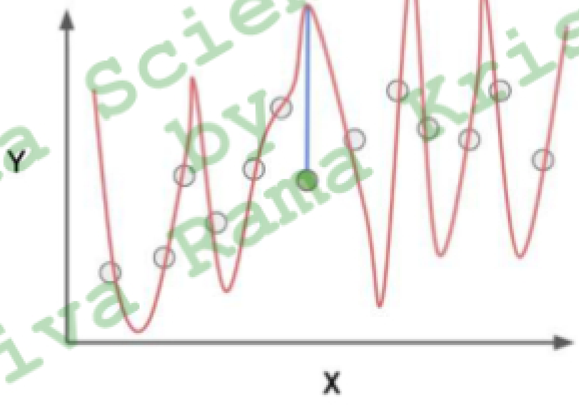
- **Error on train is zero! Model fits perfectly!**



- **But what about on a new unseen data point?**

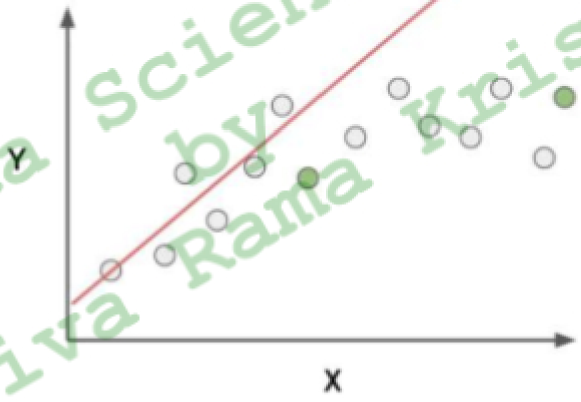


- **Overfitting can cause large test errors!**



- **Overfitting**
  - Model is fitting too much to noise and variance in the training data.
  - Model will perform very well on training data, but have poor performance on new unseen data.

Underfitting

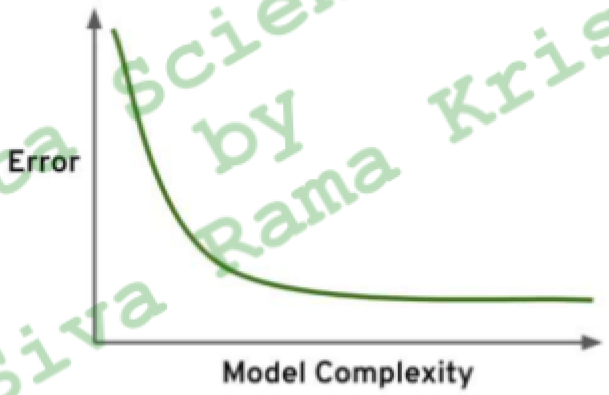


- **Underfitting**
  - Model does not capture the underlying trend of the data and does not fit the data well enough.
  - Low variance but high bias.
  - Underfitting is often a result of an excessively simple model.
  - Model has high bias and is generalizing too much.
  - Underfitting can lead to poor performance in both training and testing data sets.

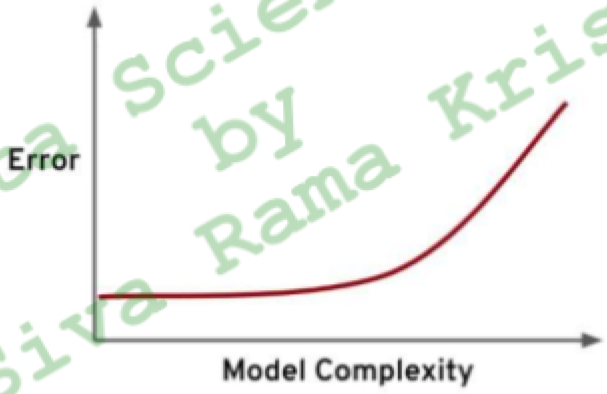
- **Overfitting versus Underfitting**
  - Overfitting can be harder to detect, since good performance on training data could lead to a model that appears to be performing well.

- This data was easy to visualize, but how can we see underfitting and overfitting when dealing with multi dimensional data sets?
- First let's imagine we trained a model and then measured its error versus model complexity (e.g. higher order polynomials).

- Good Model



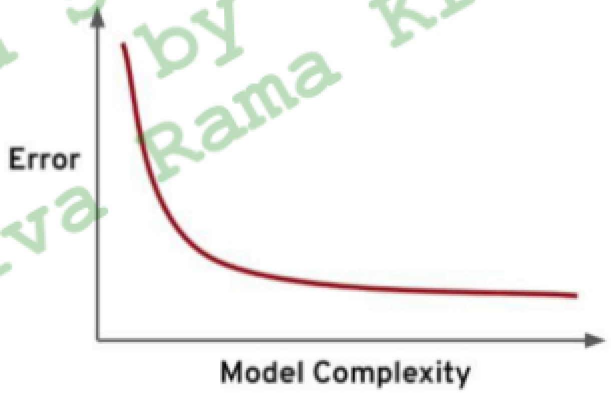
- Bad Model



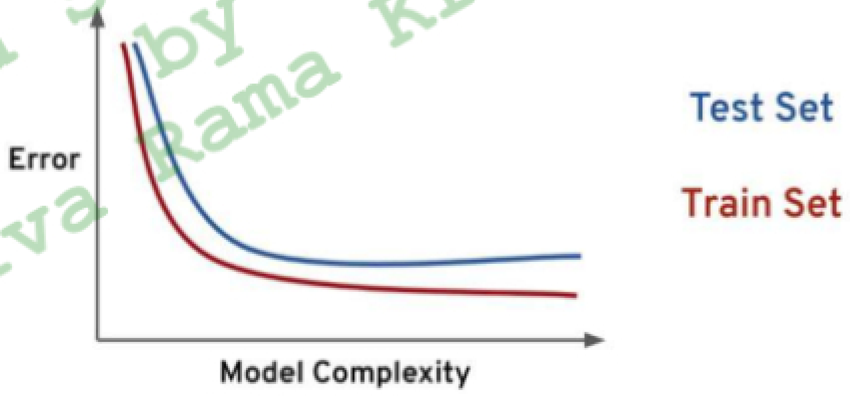
- When thinking about **overfitting** and **underfitting** we want to keep in mind the relationship of model performance on the training set versus the test/validation set.

- Let’s imagine we split our data into a **training set** and a **test set**

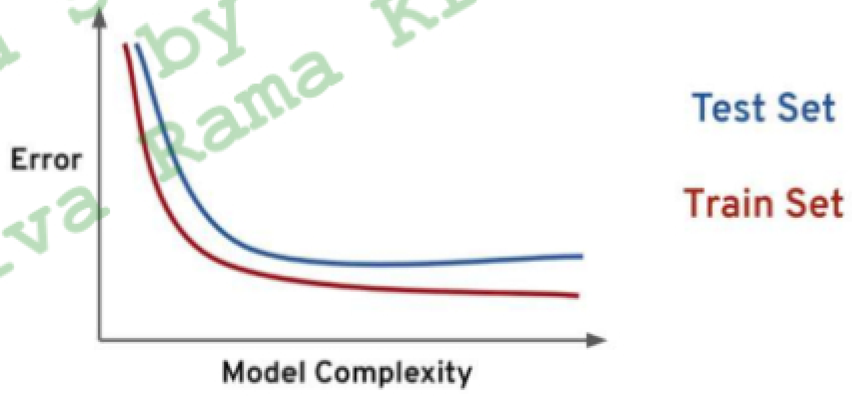
- We first see performance on the **training set**



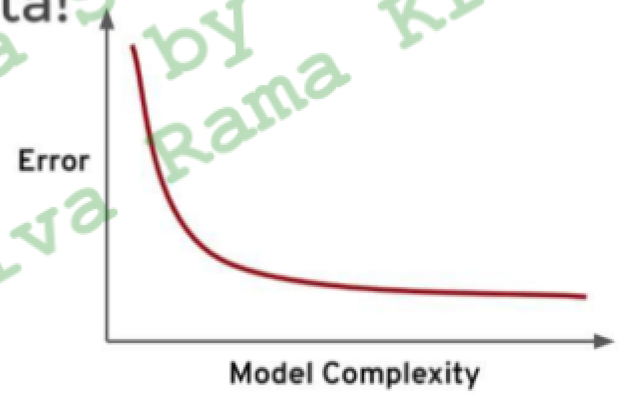
- Next we check performance on the **test set**



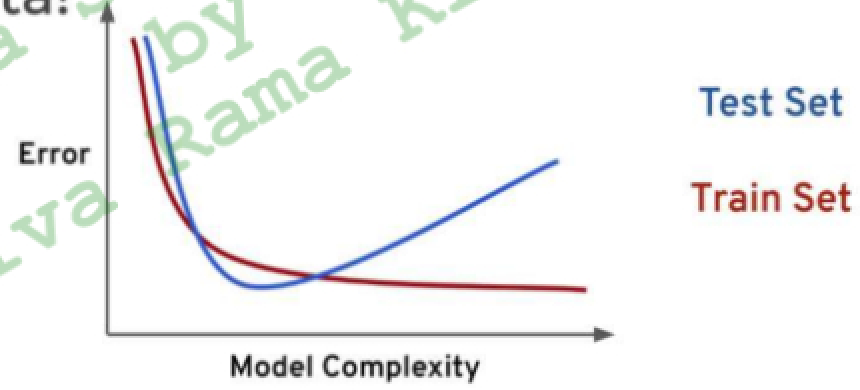
- Ideally the model would perform well on both, with similar behavior.



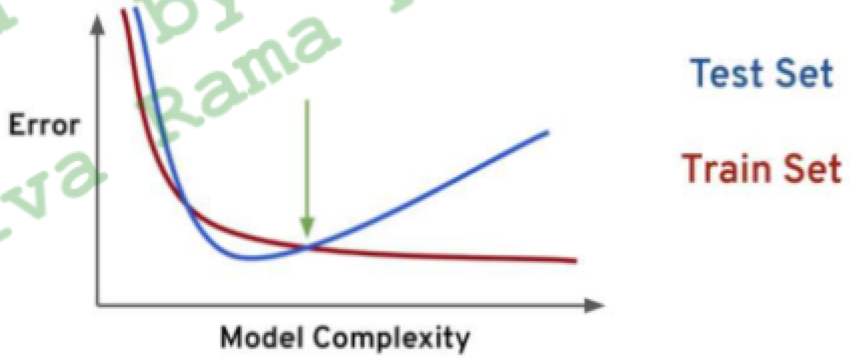
- But what happens if we overfit on the training data? That means we would perform poorly on new test data!



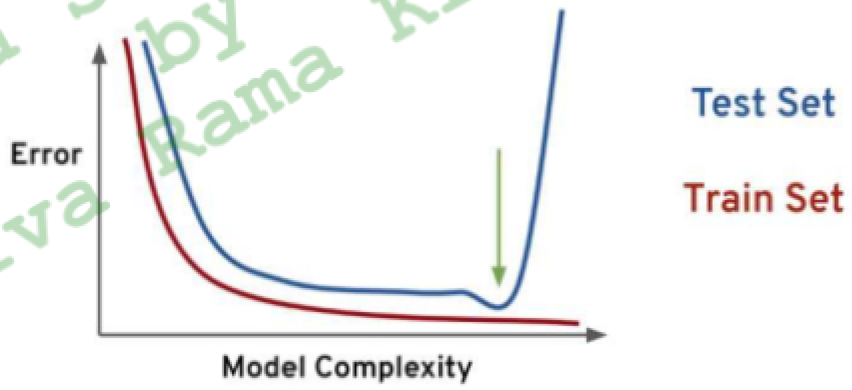
- But what happens if we overfit on the training data? That means we would perform poorly on new test data!



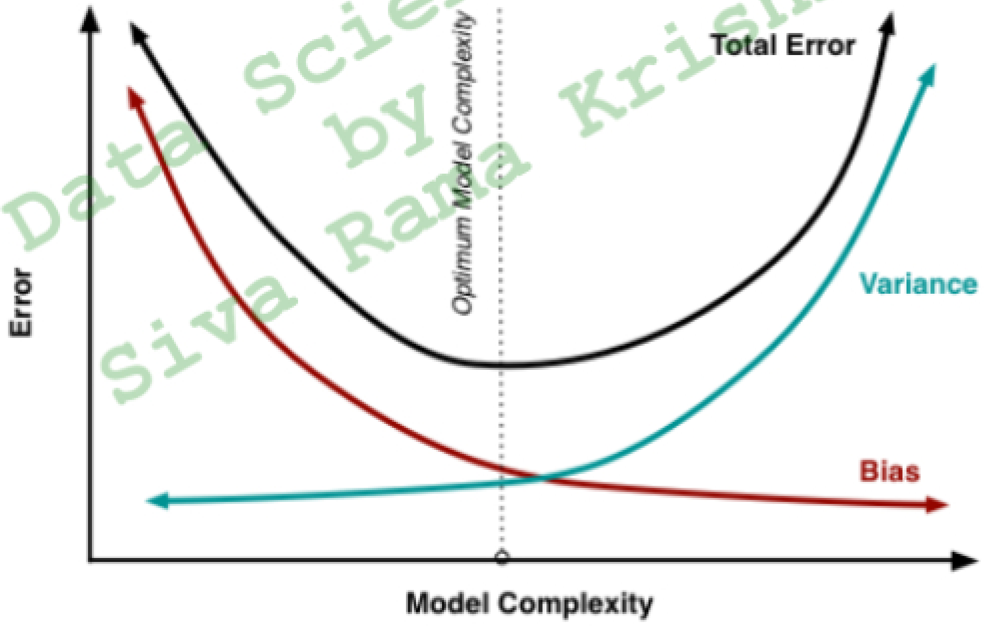
- This is a good indication too much complexity, you should look for the point to determine appropriate values!



- For certain algorithms this test error jump can be sudden instead of gradual.



- This means when deciding optimal m complexity **and** wanting to fairly evaluate our model’s performance, we can consider both the train error and test error to select an ideal complexity.



Techniques to reduce overfitting :

1. Increase training data.
2. Reduce model complexity.
3. Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
4. Ridge Regularization and Lasso Regularization
5. Use dropout for neural networks to tackle overfitting.

Machine Learning algorithms have their own hyperparameters that can increase complexity