

# Introduction to Data Science

Dr. Irfan Yousuf

Department of Computer Science (New Campus)

UET, Lahore

(Week 15; April 21 – 25, 2025)

# Outline

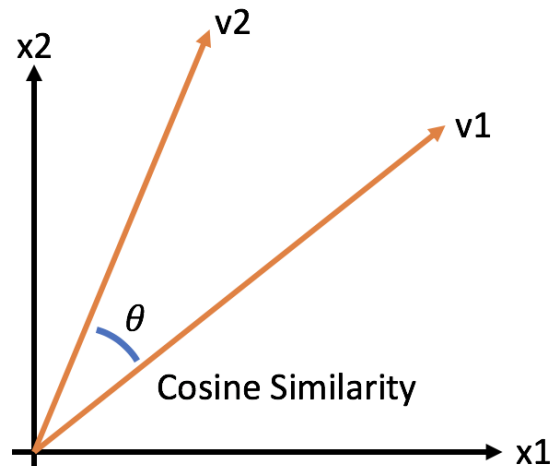
- Similarity Measures
- Model Building in Machine Learning
- Overfitting vs. Underfitting

# Similarity Measure

- In statistics, a similarity measure or similarity function is a real-valued function that **quantifies the similarity** between two objects.
- Usually, such measures are in some sense **the inverse of distance metrics**: they take on **large values for similar objects** and either zero or a **negative value for very dissimilar** objects.
- A similarity measure is a data mining or machine learning context **is a distance with dimensions** representing features of the objects.

# Cosine Similarity

- Cosine similarity measures the **similarity between two vectors** of an inner product space.
- It is measured by the cosine of the angle between two vectors and determines whether two vectors are **pointing in roughly the same direction**.
- It is often used to measure document similarity in text analysis



# Cosine Similarity

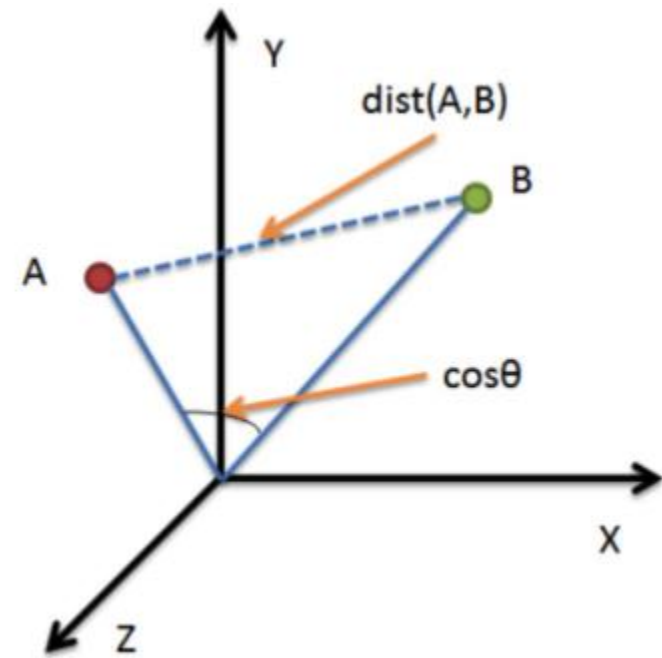
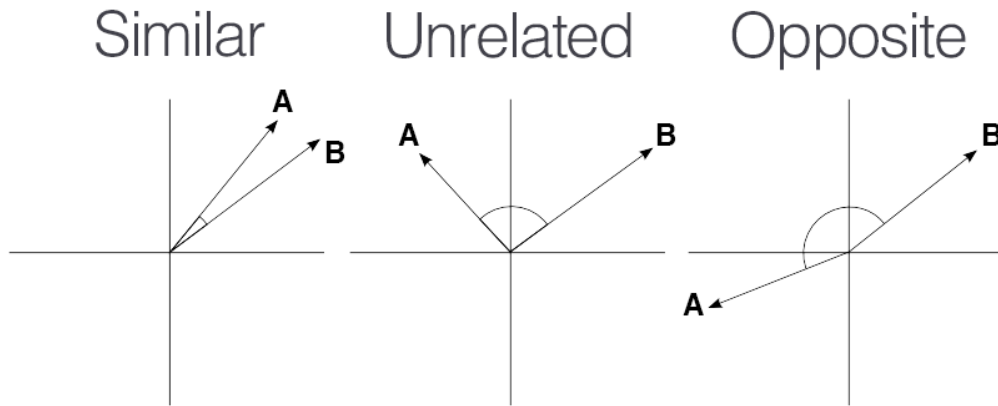
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

$$\text{similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

where

- $\theta$  is the angle between the vectors,
- $A \cdot B$  is dot product between A and B and calculated as  
 $A \cdot B = A^T B = \sum_{i=1}^n A_i B_i = A_1 B_1 + A_2 B_2 + \dots + A_n B_n,$
- $\|A\|$  represents the L2 norm or magnitude of the vector which is calculated as  
 $\|A\| = \sqrt{A_1^2 + A_2^2 + \dots + A_n^2}.$

# Cosine Similarity



Cosine similarity measures the cosine of the angle between two multi-dimensional vectors. The smaller the angle, the higher the cosine similarity. Unlike measuring Euclidean distance, **cosine similarity captures the orientation of the documents and not the magnitude**. Cosine similarity pays more attention to the difference in the direction of two vectors than the distance or length.

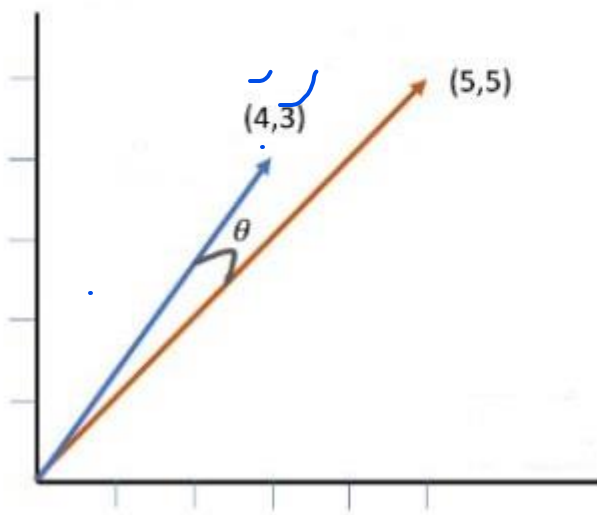
# Cosine Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

$$\text{similarity} = \cos \theta = \frac{b \cdot c}{\|b\| \|c\|}$$

$b \cdot c \Rightarrow$  Is the Dot product of the two vectors

$\|b\| \|c\| \Rightarrow$  Is the product of each vector's magnitude



Calculating:

$$b \cdot c = \sum_{i=1}^n b_i c_i = (4 \times 5) + (3 \times 5) = 35$$

$$\|b\| = \sqrt{4^2 + 3^2} = 5$$

$$\|c\| = \sqrt{5^2 + 5^2} = 5\sqrt{2}$$

$$\text{similarity} = \frac{35}{5 \times 5\sqrt{2}} \sim 0.989$$

# Cosine Similarity for Text

*Document 1: Deep Learning can be hard*

*Document 2: Deep Learning can be simple*

**Step 1: First we obtain a vectorised representation of the texts**

## Vectorised Representation

Aa Word	☰ Document 1	☰ Document 2
Deep	1	1
Learning	1	1
Can	1	1
Be	1	1
Hard	1	0
Simple	0	1



# Cosine Similarity for Text

*Document 1: [1, 1, 1, 1, 1, 0] let's refer to this as A*

*Document 2: [1, 1, 1, 1, 0, 1] let's refer to this as B*

Above we have two vectors (A and B) that are in a 6 dimension vector space

**Step 2: Find the cosine similarity**

**cosine similarity (CS) = (A . B) / (||A|| ||B||)**

- Calculate the dot product between A and B:  $1.1 + 1.1 + 1.1 + 1.1 + 1.0 + 0.1 = 4$
- Calculate the magnitude of the vector A:  $\sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0^2} = 2.2360679775$
- Calculate the magnitude of the vector B:  $\sqrt{1^2 + 1^2 + 1^2 + 1^2 + 0^2 + 1^2} = 2.2360679775$
- Calculate the cosine similarity:  $(4) / (2.2360679775 * 2.2360679775) = 0.80$  (80% similarity between the sentences in both document)

# Cosine Similarity for Text

Doc1: Data is the oil of the digital economy

Doc2: Data is a new oil

	data	digital	economy	is	new	of	oil	the
doc_1	1	1	1	1	0	1	1	2
doc_2	1	0	0	1	1	0	1	0

$$\begin{aligned}A \cdot B &= \sum_{i=1}^n A_i B_i \\&= (1 * 1) + (1 * 0) + (1 * 0) + (1 * 1) + (0 * 1) + (1 * 0) + (1 * 1) + (2 * 0) \\&= 3\end{aligned}$$

$$\sqrt{\sum_{i=1}^n A_i^2} = \sqrt{1+1+1+1+0+1+1+4} = \sqrt{10}$$

$$\sqrt{\sum_{i=1}^n B_i^2} = \sqrt{1+0+0+1+1+0+1+0} = \sqrt{4}$$

$$\text{cosine similarity} = \cos\theta = \frac{A \cdot B}{|A||B|} = \frac{3}{\sqrt{10} \cdot \sqrt{4}} = 0.4743$$

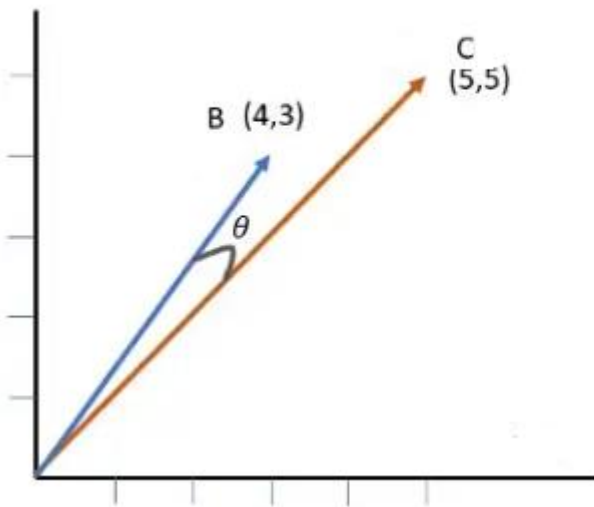
# Cosine Similarity for Movie Ratings

User	Movie1	Movie2
B	4	3
C	5	5

$$\vec{b} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

$$\vec{c} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$20 + 5$$



Calculating:

$$b \cdot c = \sum_{i=1}^n b_i c_i = (4 \times 5) + (3 \times 5) = 35$$

$$\|b\| = \sqrt{4^2 + 3^2} = 5$$

$$\|c\| = \sqrt{5^2 + 5^2} = 5\sqrt{2}$$

$$\text{similarity} = \frac{35}{5 \times 5\sqrt{2}} \sim 0.989$$

# Cosine Similarity for Movie Ratings

User	Movie1	Movie2	Movie3
B	4	3	5
C	5	5	1

$$\vec{b} = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$$

$$\vec{c} = \begin{bmatrix} 5 \\ 5 \\ 1 \end{bmatrix}$$

$$b \cdot c = (4 \times 5) + (3 \times 5) + (5 \times 1) = 40$$

$$\|b\| = \sqrt{4^2 + 3^2 + 5^2} = 5\sqrt{2}$$

$$\|c\| = \sqrt{5^2 + 5^2 + 1^2} = \sqrt{51}$$

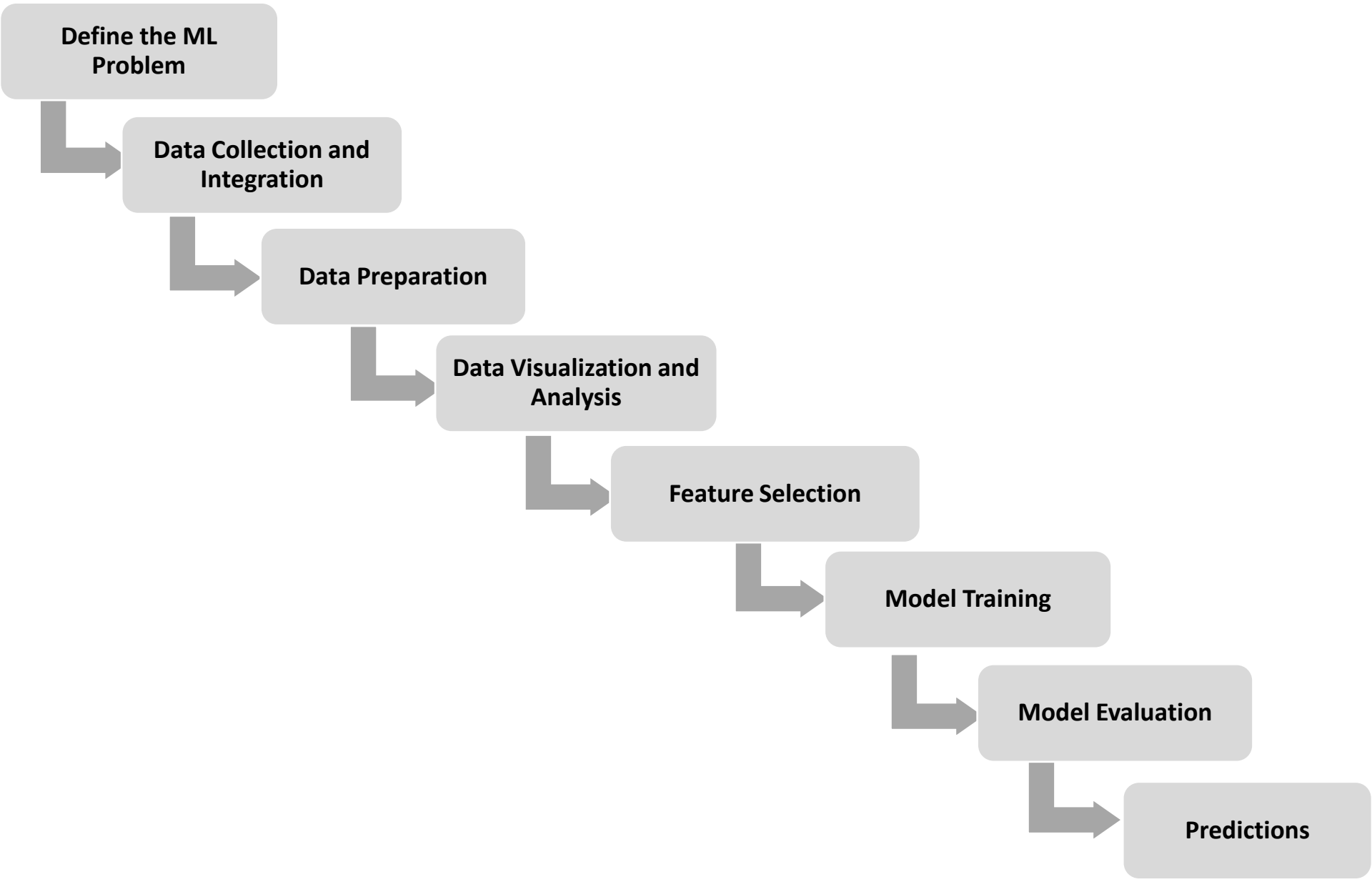
$$\text{similarity} = \frac{40}{5\sqrt{2} \times \sqrt{51}} \sim 0.792$$

# **Model Building in Machine Learning**



# Model Building in Machine Learning

- A machine learning model is built by learning and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfill its purpose.

# Model Building in Machine Learning



# Step 1: Define the Problem

- We need to define our objectives and evaluate the problem that we are facing.
  - Generally, our predictions are divided into three different main categories depending on the ML problem we need to answer;
    - Classification
    - Clustering
    - Regression
- 
- 



## **Step 2: Data Collection and Integration**

- Data is everywhere so it can be collected from multiple sources like internet, databases or other types of storage.
- Chances are, that some of the data you collect going to be noisy - your data is possibly incomplete even irrelevant.
- So, wherever it comes from, it will need to be compiled - get integrated.
- The quality and quantity of data that you gather will directly determine how accurate your model can be.

# Step 3: Data Preparation

- Data you collected is
  - Noisy
  - Incomplete
  - Irrelevant
- You must clean it.
- You also need to normalize the data.

## Step 4: Data Visualization and Analysis

- Perform Exploratory Data Analysis (EDA)
- It's a technique that helps you understand the relationships within your dataset.
- This leads to better features, better models.
- When you can see the data in a chart or plotted out, you can help unveil previously unseen patterns.
- It reveals corrupt data or outliers that you don't want, properties that could be very significant in your analysis

# Step 5: Feature Selection

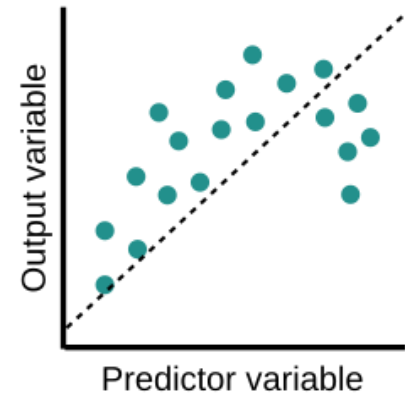
- A feature is a characteristic that might help when solving the problem.
- We will look for features that correlate to our desired output.
- A crucial part in this step is Feature Engineering – the process of manipulating the original data into new and potentially a lot of more useful features.

# Step 6: Training

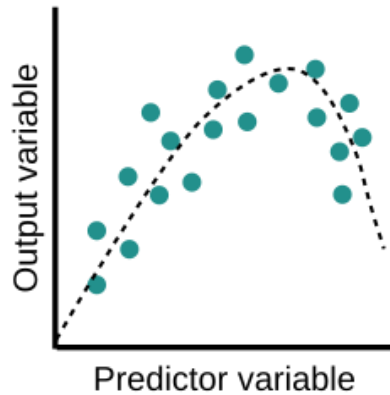
- The training often considered the main part of machine learning.
- **Overfitting:** the model learns the particulars of dataset too well.
- **Underfitting:** will not have enough features to model the data properly
- **Bias:** which is the gap between predicted value and actual value.
- **Variance:** how dispersed your predicted values are.

# Step 6: Training

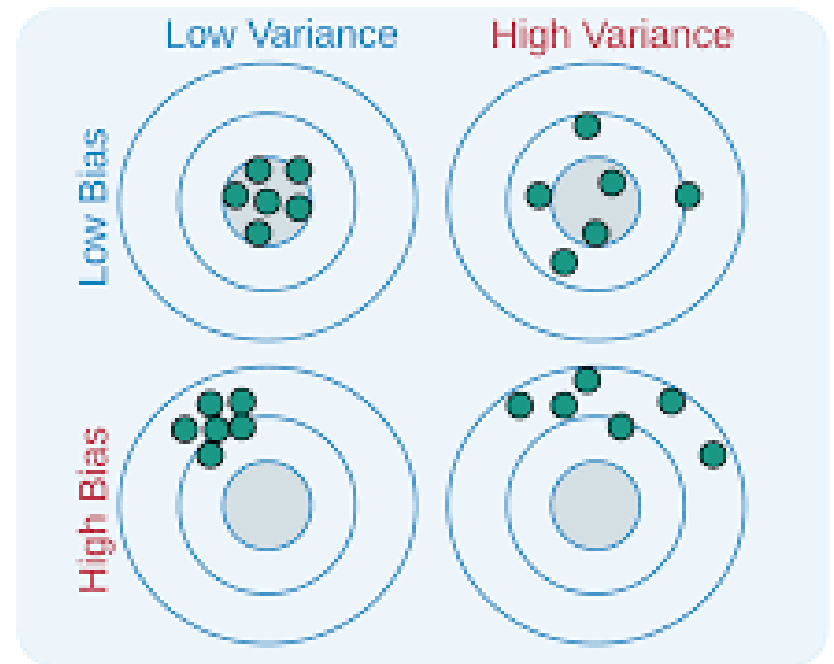
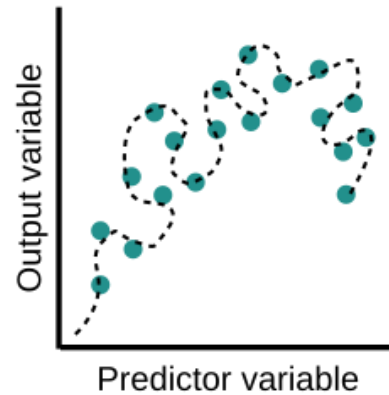
Underfit



Optimal



Overfit

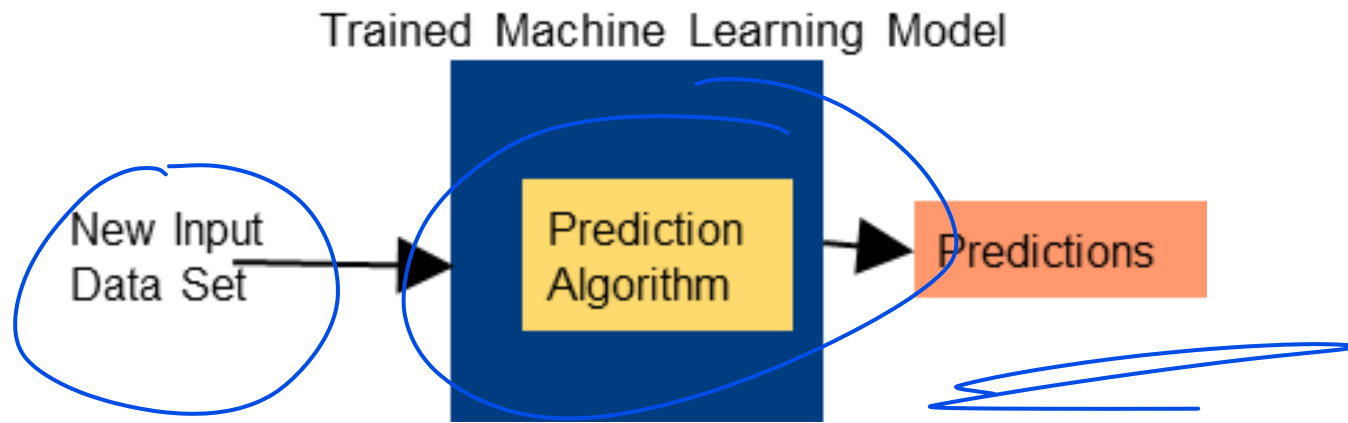


## Step 7: Model Evaluation

- One of the most effective ways to evaluate your model's accuracy, precision, and ability to recall involves looking at something called a confusion matrix.
- The confusion matrix analyzes the model and shows how many of the data points were predicted correctly and incorrectly.

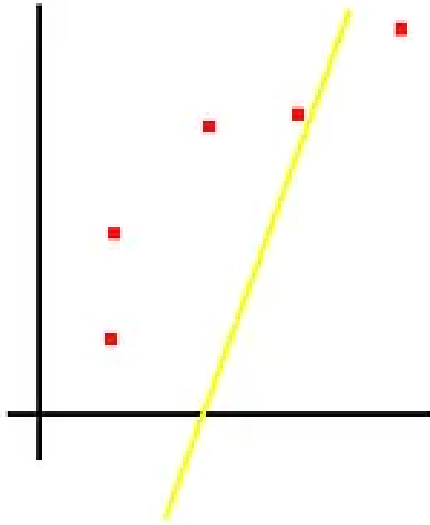
# Step 8: Predictions

- Process unseen data and predict something.

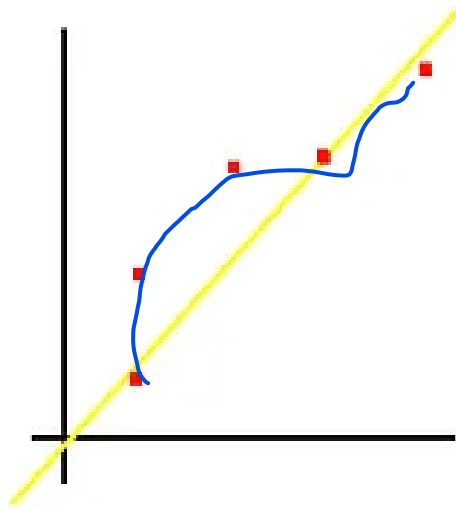




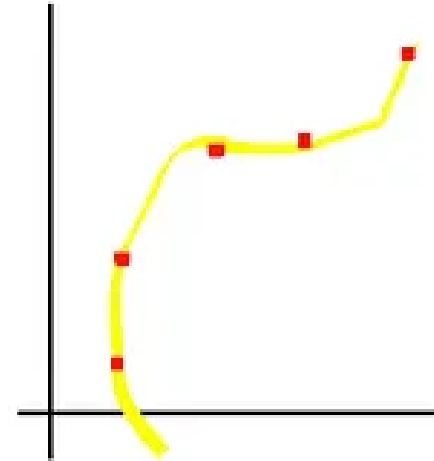
# Overfitting and Underfitting



Underfitting



Perfect Fit



Overfitting

# Overfitting

- Overfitting refers to the scenario where a machine learning **model can't generalize or fit well on unseen dataset.**
- A clear sign of machine learning overfitting is if its error on the testing or validation dataset is much greater than the error on training dataset.
- Overfitting is a term used in statistics that refers to a modeling error that occurs when a function corresponds too closely to a dataset.
- As a result, **overfitting may fail to fit additional data**, and this may affect the accuracy of predicting future observations.

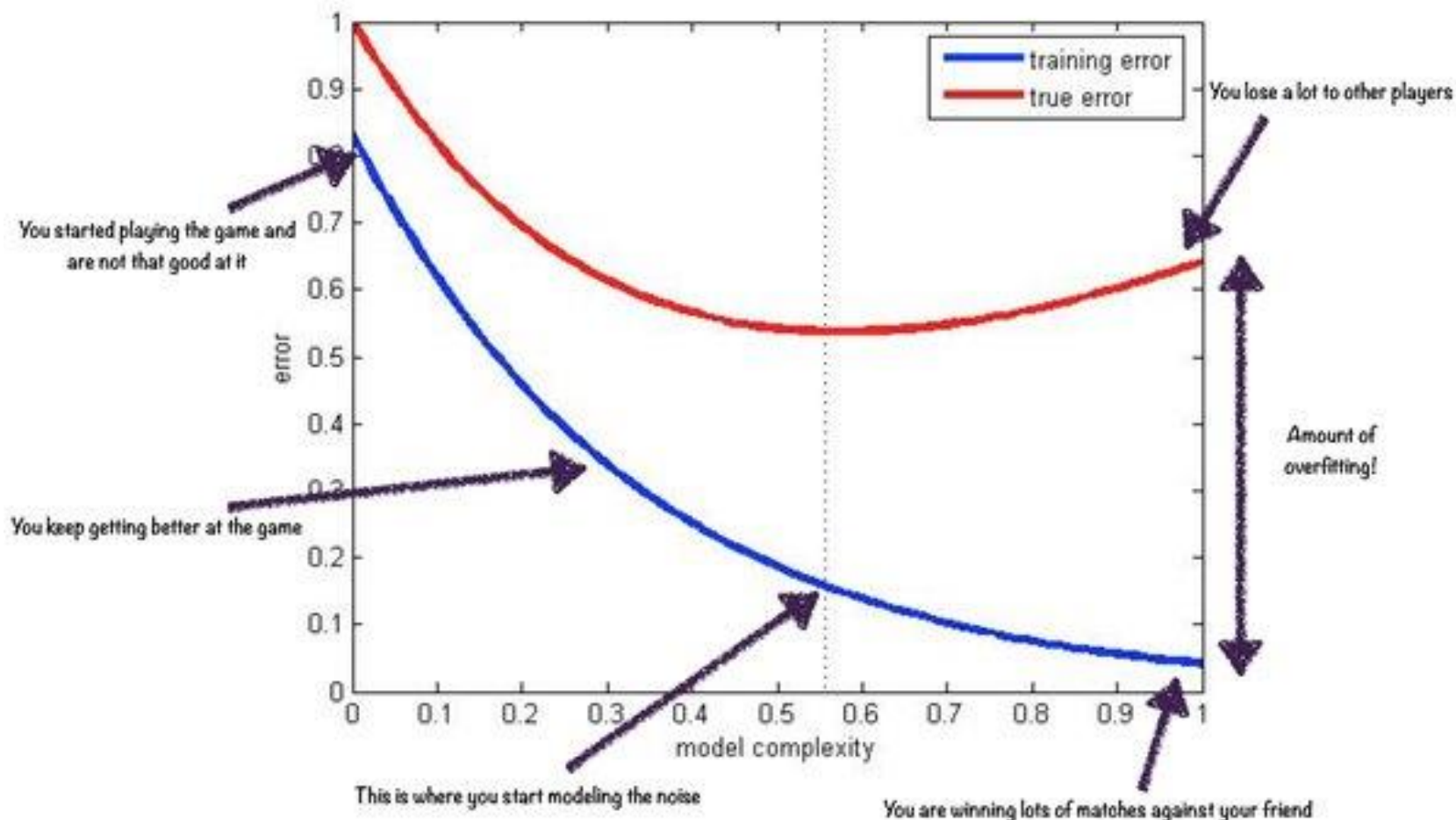
# Overfitting

- Overfitting happens when a model **learns the detail and noise in the training dataset** to the extent that it negatively impacts the performance of the model on a new dataset.
- This means that the **noise or random fluctuations in the training dataset is picked up and learned as concepts by the model.**
- The problem is that these concepts do not apply to new datasets and negatively impact the model's ability to generalize.

# Overfitting

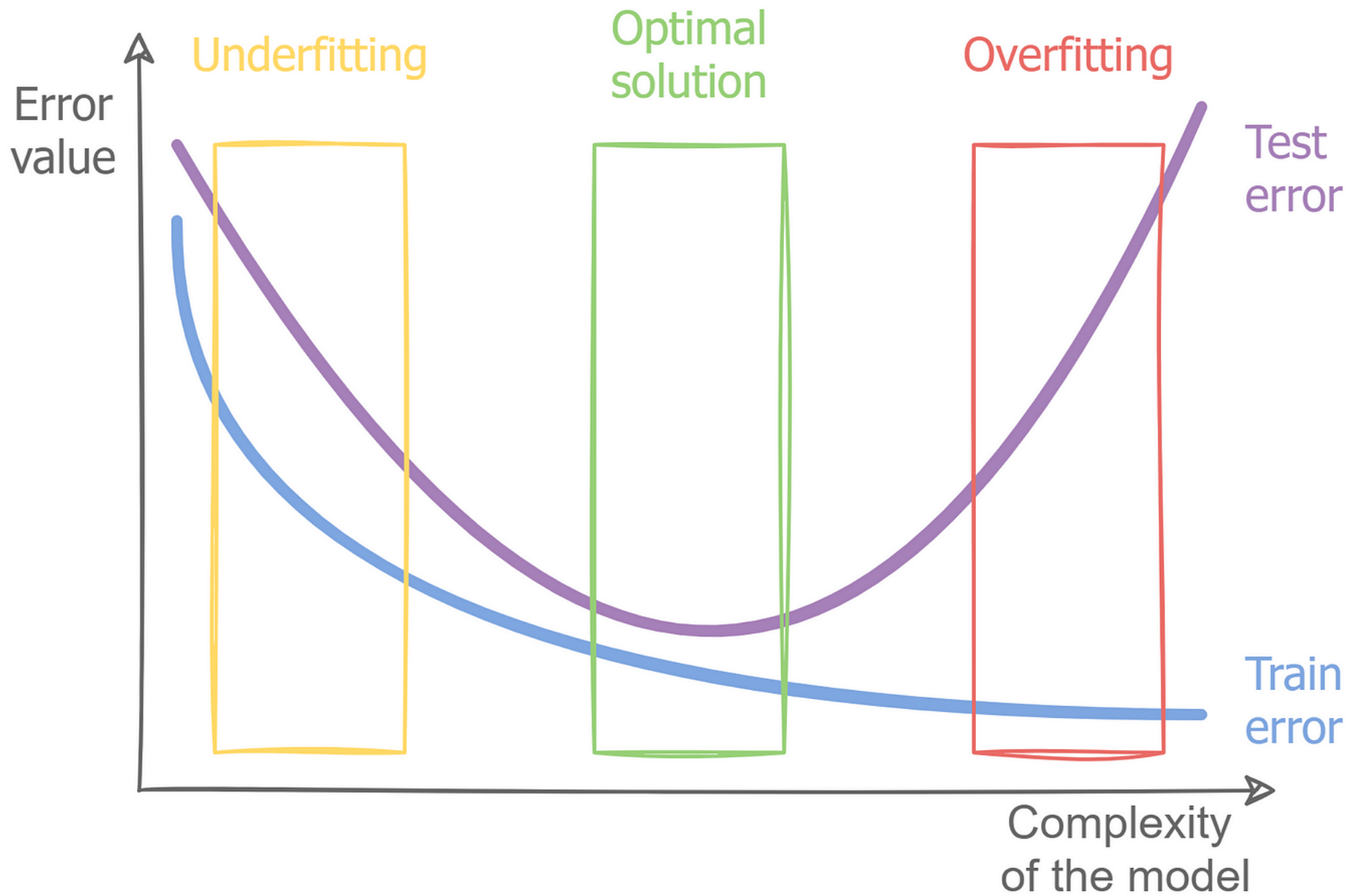
- Say you start playing FIFA with your friend on the Xbox. You lose first. But slowly you learn the tricks of the game and keep getting better at it. You get so good at the game that you start defeating your friend in every other match. You are pretty happy with yourself because you came a long way from knowing nothing.
- One a fine day, your friend is not around and you have to play with someone else. And you lose horribly. You begin to wonder what went wrong. When this phenomenon happens in machine learning, people call it overfitting.

# Overfitting

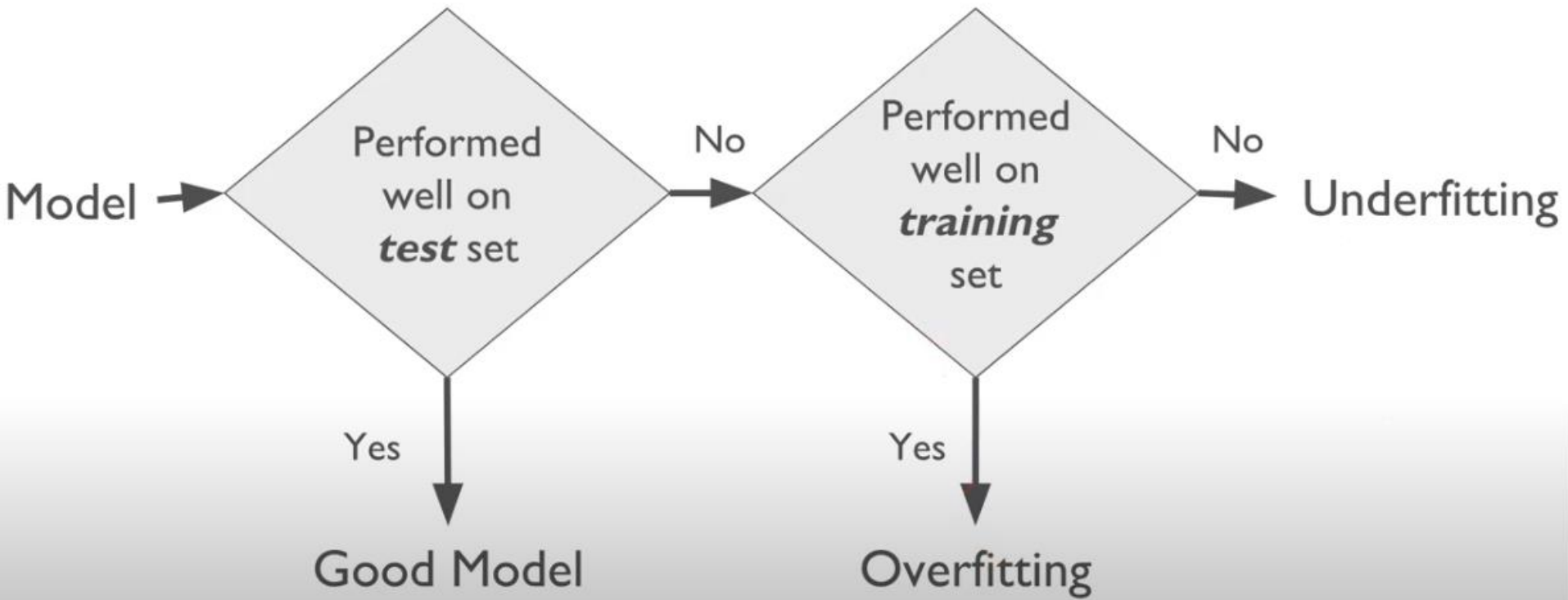


# Underfitting

- Underfitting refers to a model that **can neither model the training dataset nor generalize to new dataset.**
- An underfit machine learning model is not a suitable model and will be obvious as **it will have poor performance on the training dataset.**



# Overfitting and Underfitting

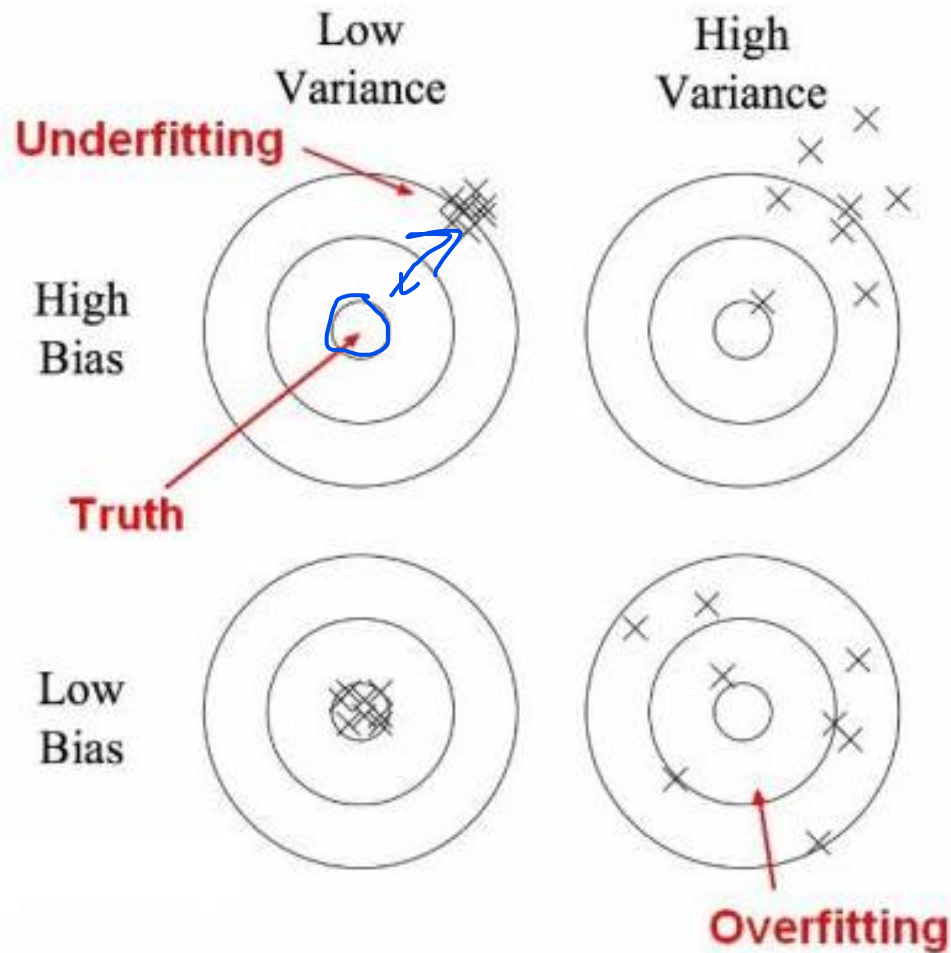




# Reasons for Overfitting and Underfitting

- **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**.
- 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.

# Reasons for Overfitting and Underfitting



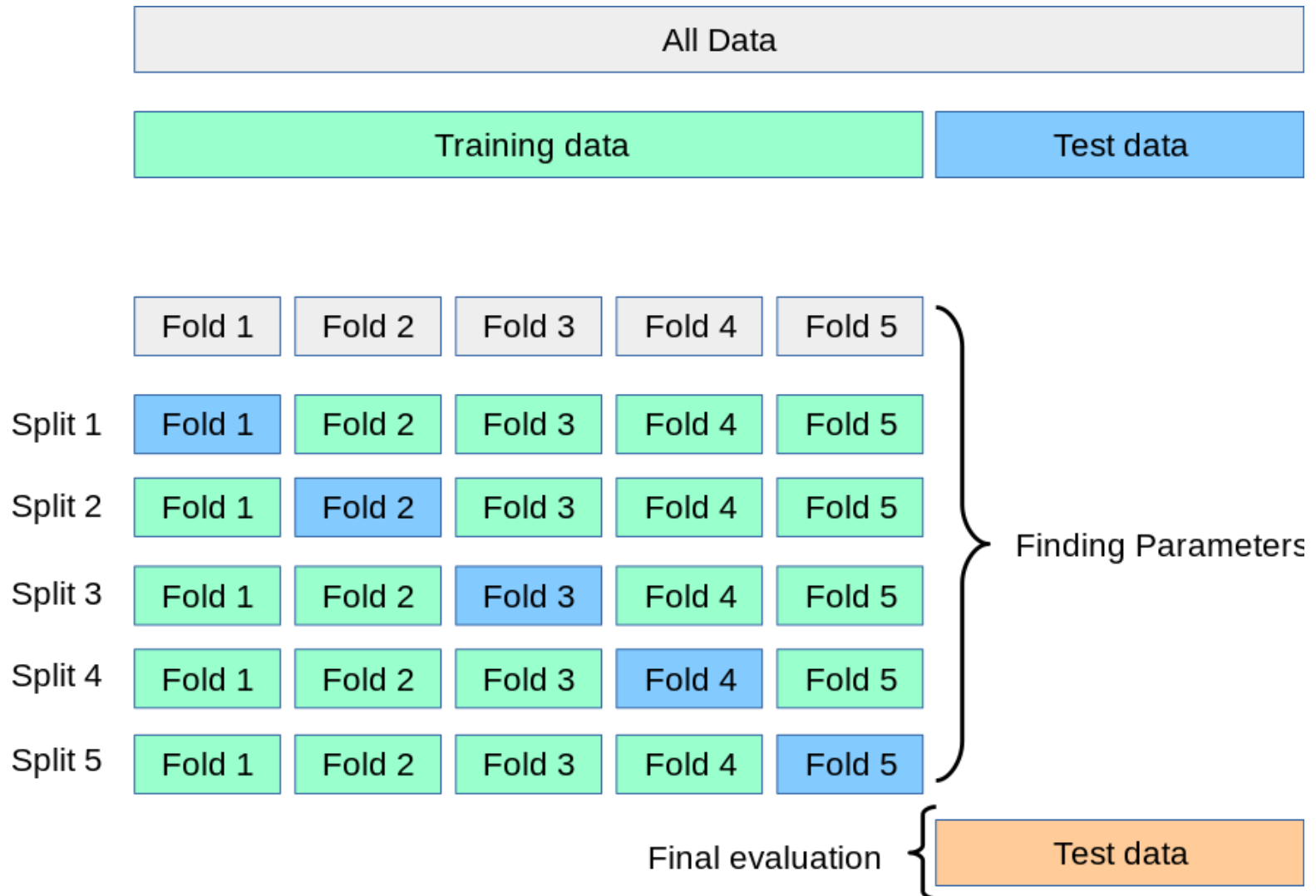
# Reasons for Overfitting and Underfitting

- **Specificity:** Ability to perform focused learning on training data such that the model learns the minute details of the data.
- **Generalizability:** Ability to generalize model for unseen test data upon training by seen data.
- **OverFitting:** High Specificity, Low Generalizability
- **UnderFitting:** Low Specificity, High Generalizability

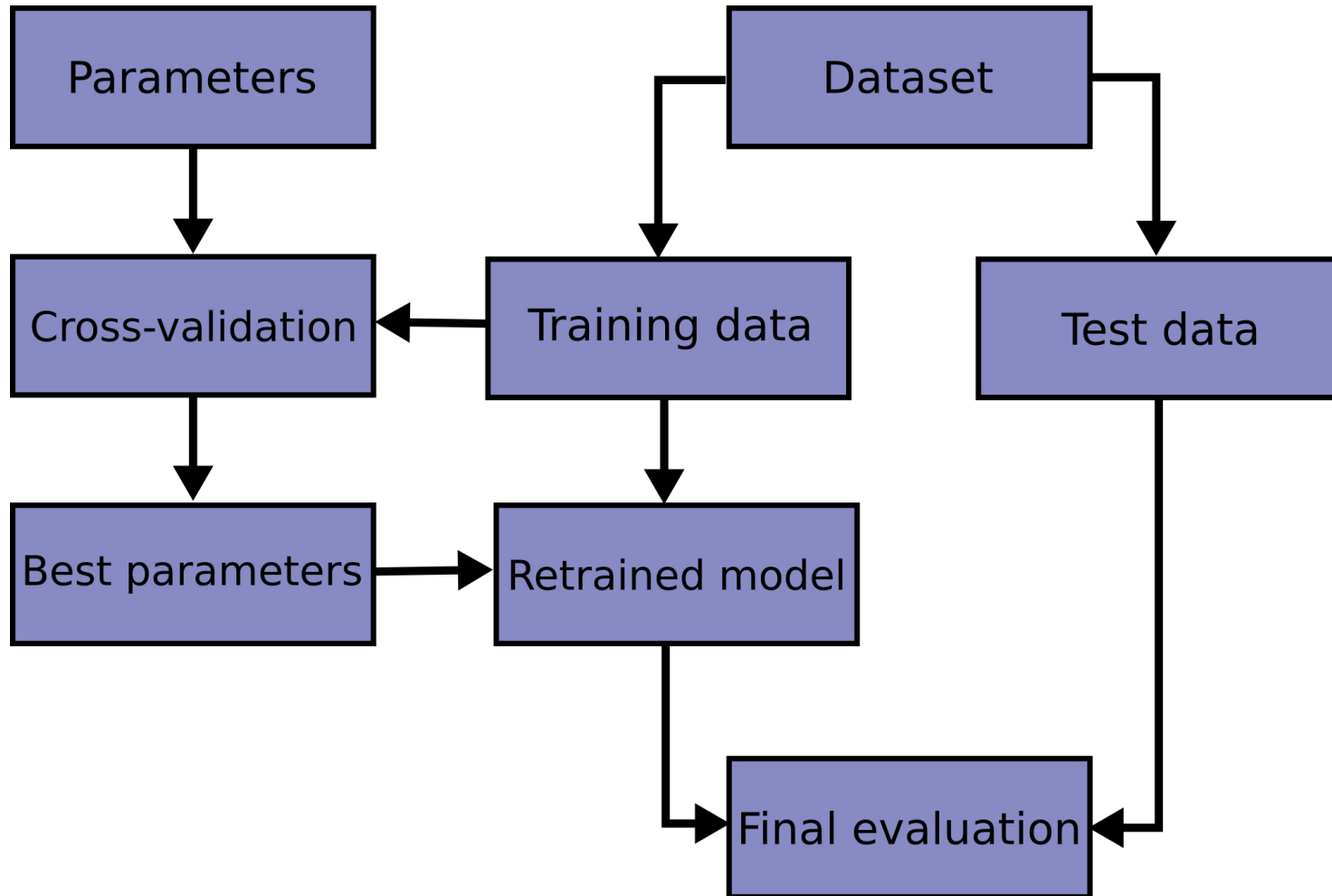
# How to Prevent Overfitting

- **Cross-validation** is a powerful preventative measure against overfitting.
- 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, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining fold as the test set.

# How to Prevent Overfitting



# How to Prevent Overfitting



# How to Prevent Overfitting

- **Train with more data:** It won't work every time, but training with more data can help algorithms detect the signal better.
- **Feature Selection:** We can improve generalizability of a model by removing irrelevant input features.

# How to Prevent Underfitting

- **Increase model complexity:** As model complexity increases, performance on the data used to build the model (training data) improves.
- Remove noise from the data.
- Increase number of features.
- Increase the duration of training



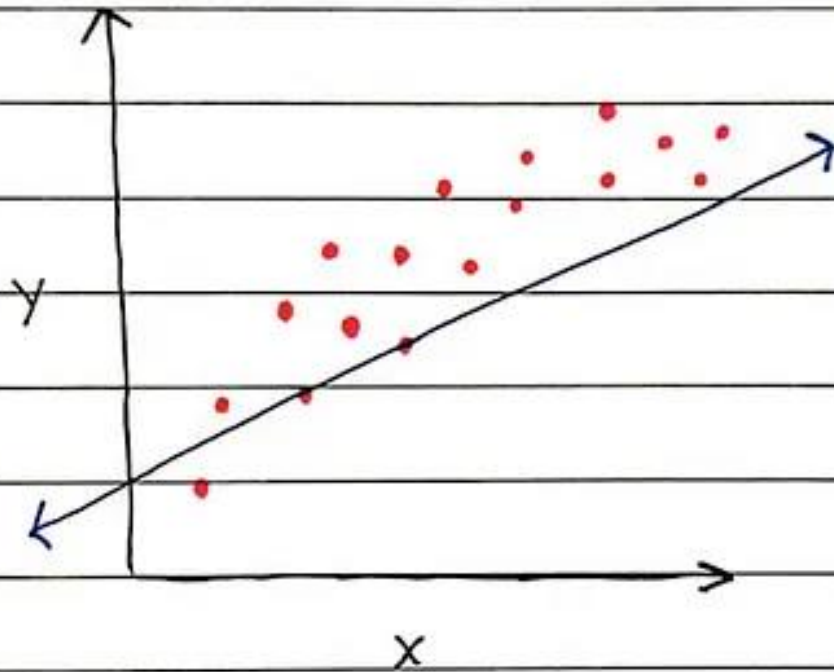
# Bias and Variance Trade-off

- In general, a machine learning model analyses the data, find patterns in it and make predictions.
- While training, the model learns these patterns in the dataset and applies them to test data for prediction.
- While making predictions, a difference occurs between prediction values made by the model and actual values/expected values, and this difference is known as **bias errors or Errors due to bias.**

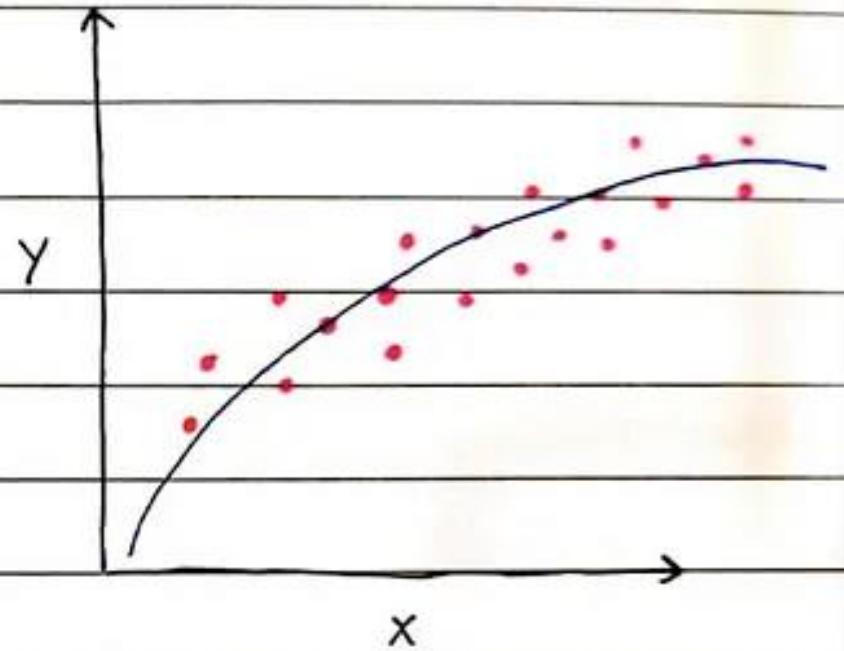
# Bias

- It can be defined as an inability of an algorithm to capture the true relationship between the data points.
- Each algorithm begins with some amount of bias because bias occurs from assumptions in the model, which makes the target function simple to learn.
- **Low Bias:** A low bias model will make fewer assumptions about the form of the target function.
- **High Bias:** A model with a high bias makes more assumptions, and the model becomes unable to capture the important features of our dataset. A high bias model also cannot perform well on new data.

# Bias



HIGH BIAS



LOW BIAS

# Variance

- The variance would specify the amount of variation in the prediction if different training data was used.
- In simple words, variance tells that how much a random variable is different from its expected value.
- Ideally, a model should not vary too much from one training dataset to another, which means the algorithm should be good in understanding the hidden mapping between inputs and output variables.
- Variance errors are either of low variance or high variance.

# Variance

- **Low variance** means there is a small variation in the prediction of the target function with changes in the training data set.
- **High variance** shows a large variation in the prediction of the target function with changes in the training dataset.





# Bias and Variance Trade-off

- Let us consider the problem of fitting a curve through a given set of points.
- The points are drawn from a sinusoidal function (the true  $f(x)$  )

# Bias and Variance Trade-off

We consider two models :

$$\begin{array}{l} \textit{Simple} \\ (\textit{degree : 1}) \end{array} \quad y = \hat{f}(x) = w_1x + w_0$$


$$\begin{array}{l} \textit{Complex} \\ (\textit{degree : 25}) \end{array} \quad y = \hat{f}(x) = \sum_{i=1}^{25} w_i x^i + w_0$$


Note that in both cases we are making an assumption about how  $y$  is related to  $x$ . We have no idea about the true relation  $f(x)$

The training data consists of 500 points

# Bias and Variance Trade-off

We sampled 500 points from the training data and train a simple and a complex model

We repeat the process 'k' times to train multiple models (each model sees a different sample of the training data)




# Bias and Variance Trade-off

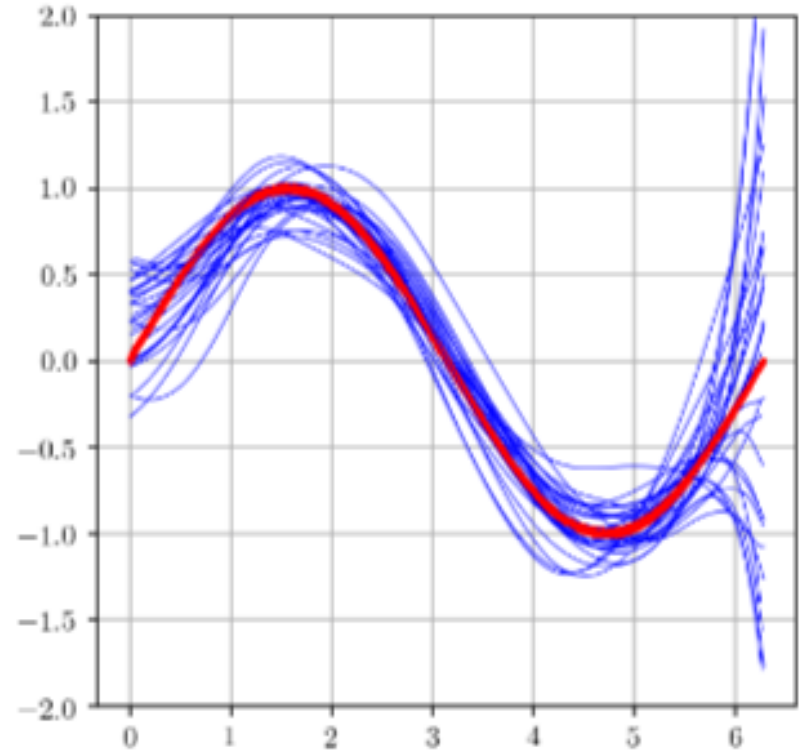
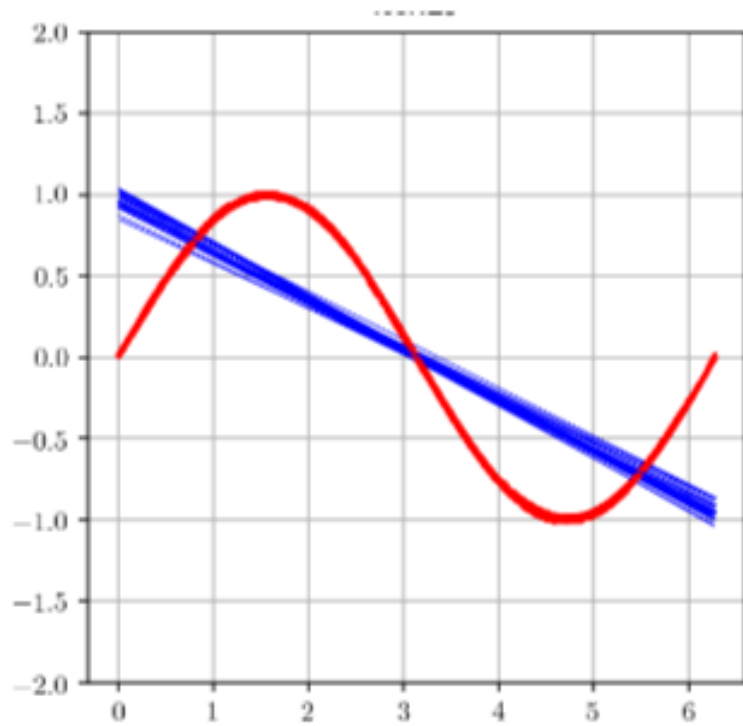
Simple models trained on different samples of the data do not differ much from each other

However they are very far from the true sinusoidal curve (under fitting)

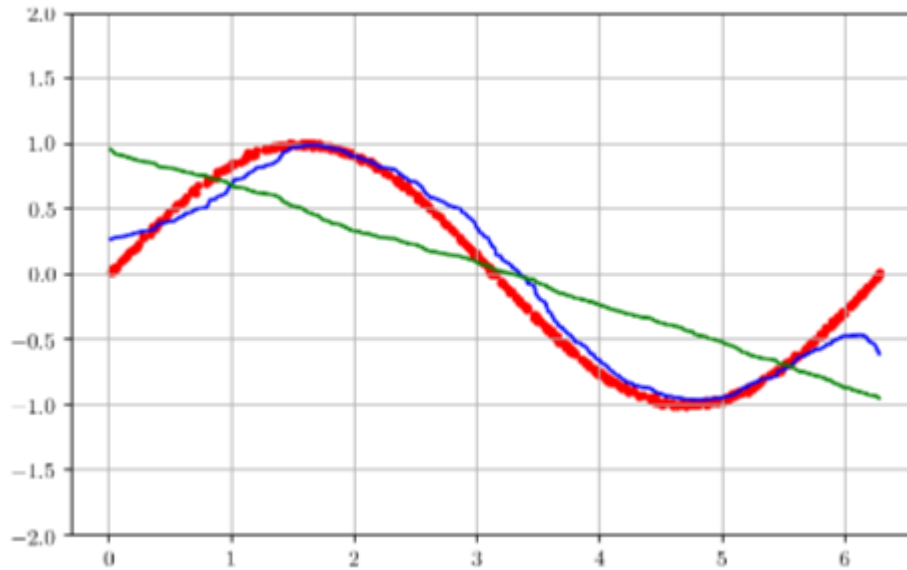
On the other hand, complex models trained on different samples of the data are very different from each other (high variance)



# Bias and Variance Trade-off



# Bias and Variance Trade-off



Green Line: Average value of  $\hat{f}(x)$  for the simple model

Blue Curve: Average value of  $\hat{f}(x)$  for the complex model

Red Curve: True model ( $f(x)$ )

# Bias and Variance Trade-off

Let  $f(x)$  be the true model (sinusoidal in this case) and  $\hat{f}(x)$  be our estimate of the model (simple or complex, in this case) then,

$$\text{Bias } \hat{f}(x) = E[\hat{f}(x)] - f(x)$$

$E[\hat{f}(x)]$  is the average (or expected) value of the model

We can see that for the simple model the average value (green line) is very far from the true value  $f(x)$  (sinusoidal function)

Mathematically, this means that the simple model has a high bias

The expected value is the mean of the possible values a random variable can take, weighted by the probability of those outcomes. Expected value = Weighted Average

# Bias and Variance Trade-off

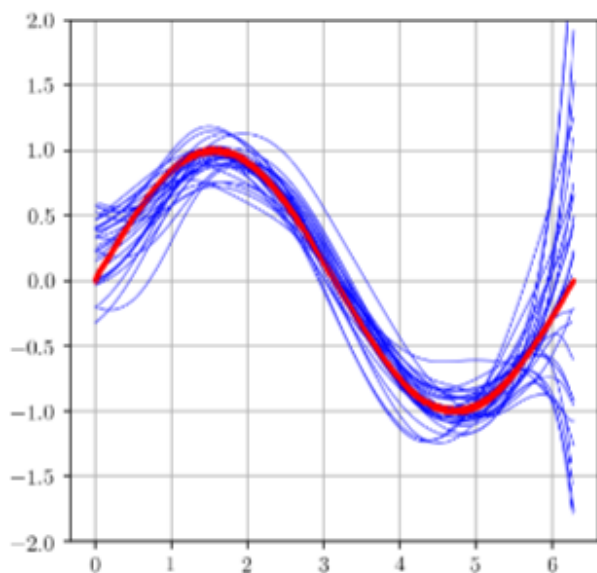
The variance of a random variable  $X$  is the expected value of the squared deviation from the mean of  $X$ ,  $\mu = \mathbb{E}[X]$ :

$$\text{Var}(X) = \mathbb{E}[(X - \mu)^2].$$

$$\begin{aligned}\text{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2 - 2X\mathbb{E}[X] + \mathbb{E}[X]^2] \\ &= \mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X]^2 \\ &= \mathbb{E}[X^2] - 2\mathbb{E}[X]^2 + \mathbb{E}[X]^2 \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2\end{aligned}$$

The variance of  $X$  is equal to the mean of the square of  $X$  minus the square of the mean of  $X$ .

# Bias and Variance Trade-off



$$\text{Variance } \hat{f}(x) = E \left[ (\hat{f}(x) - E[\hat{f}(x)])^2 \right]$$

(Standard definition from statistics)

Roughly speaking it tells us how much the different  $\hat{f}(x)$ 's (trained on different samples of the data) differ from each other)

It is clear that the simple model has a low variance whereas the complex model has a high variance

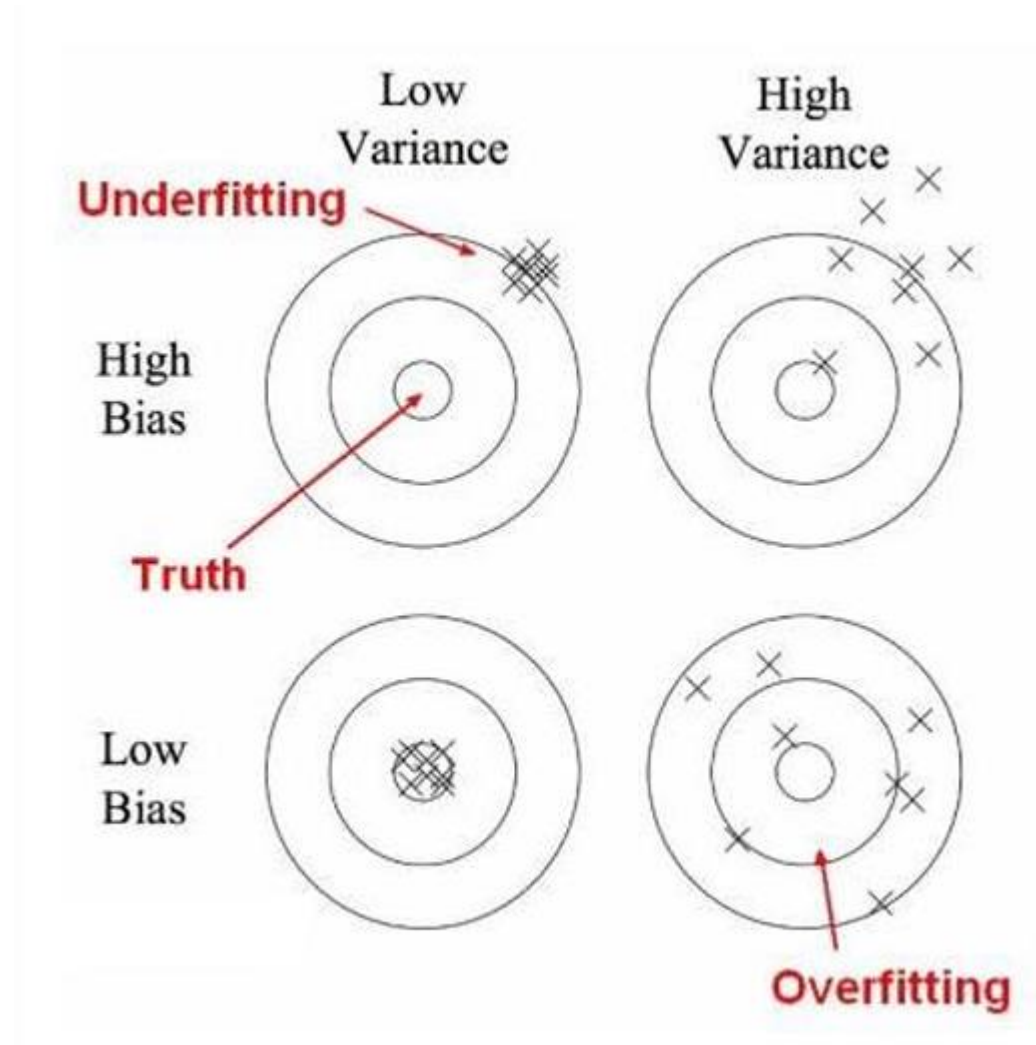
# Bias and Variance Trade-off

Simple model: high bias, low variance

Complex model: low bias, high variance

There is always a trade-off between the bias and variance

# Bias and Variance Trade-off





# Summary

- Model Building in Machine Learning