Bias-Variance Tradeoff

- Bias
- Variance
- Overfitting
- Underfitting
- Good Fit
- How to tackle overfitting

Approximate a Target Function in Machine Learning

- Supervised Machine learning approximating a target function (f) that maps input variables (X) to an output variable (Y).

$$Y = f(X)$$

- Generalization is important because the data we collect is only a sample, it is incomplete and noisy.

- **Generalization refers** to how well the concepts learned by a model apply to specific examples not seen by the model (**how well the model generalizes to new data**).

What is “Error” in Machine Learning?

- Error is the difference in the expected output and the predicted output of the model.
- Measures - how well the model performs over a given set of data
- To calculate error - Loss/Cost Function (Mean Squared Error (MSE)) is used

$$MSE = 1/n \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$$

n = *Number of data-points over which the error is calculated*
y = *The expected output of the model*
$\bar{y}$ = *The predicted output of the model*

Types of errors

- Errors in any supervised Machine Learning algorithm comprises of 3 parts:
  - **Bias error** - reducible errors - we can attempt to minimize as much as possible
  - **Variance error** - reducible errors - we can attempt to minimize as much as possible
  - **Noise** - irreducible error - cannot eliminate

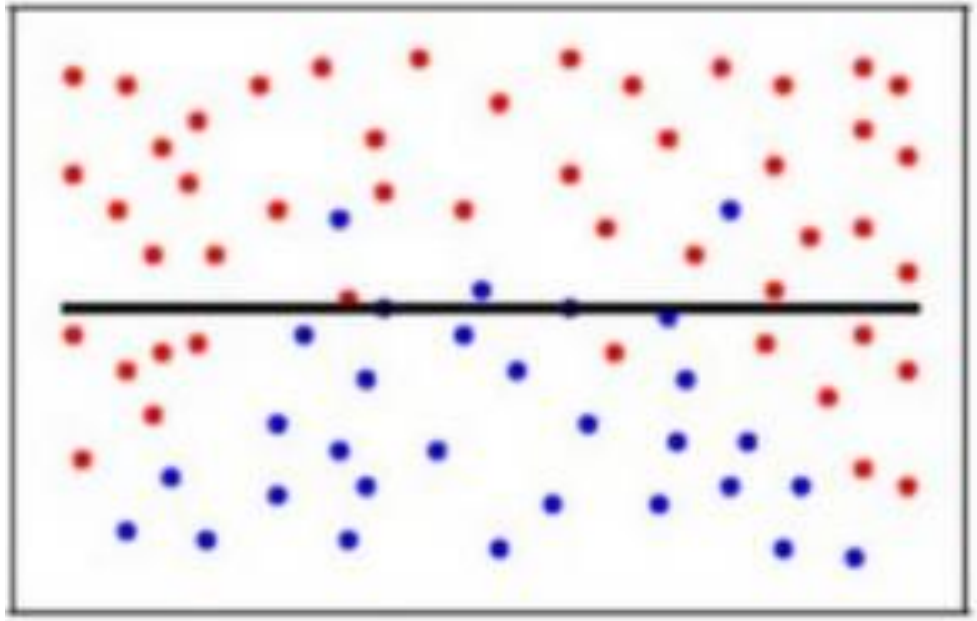
Total Error = Bias Error + Variance Error + Noise

What is Bias?

- The model makes certain assumptions when it trains on the data provided.
- When it is introduced to the testing/ validation data, these assumptions may not always be correct.
- When there is a high bias error, it results in a very simplistic model that does not consider the variations very well (low variance).
- Since it does not learn the training data very well, it is called **Underfitting.**

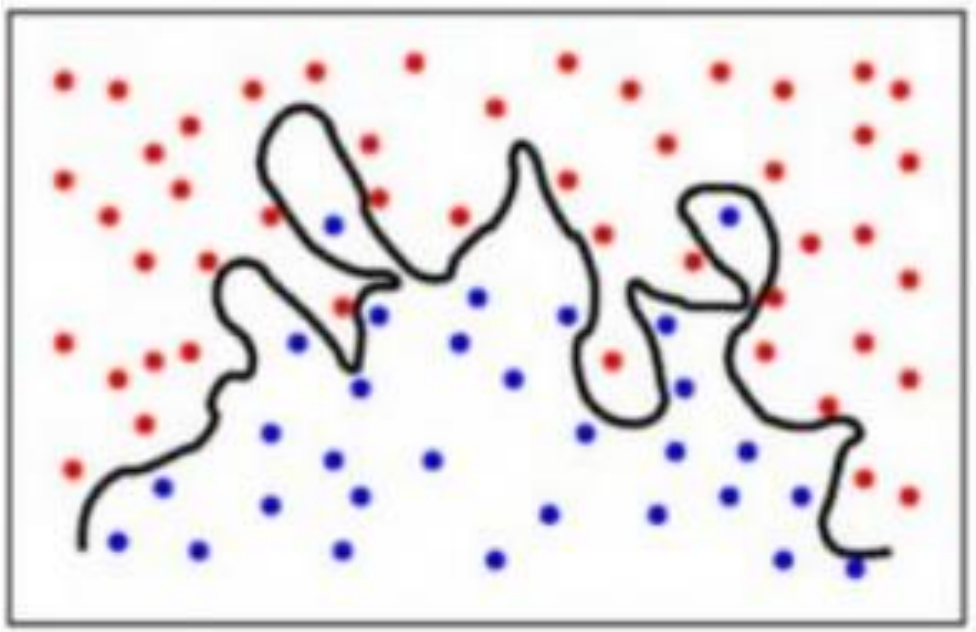
$$y = f(x) + e$$

Here ‘e’ is the error that is normally distributed



What is a Variance?

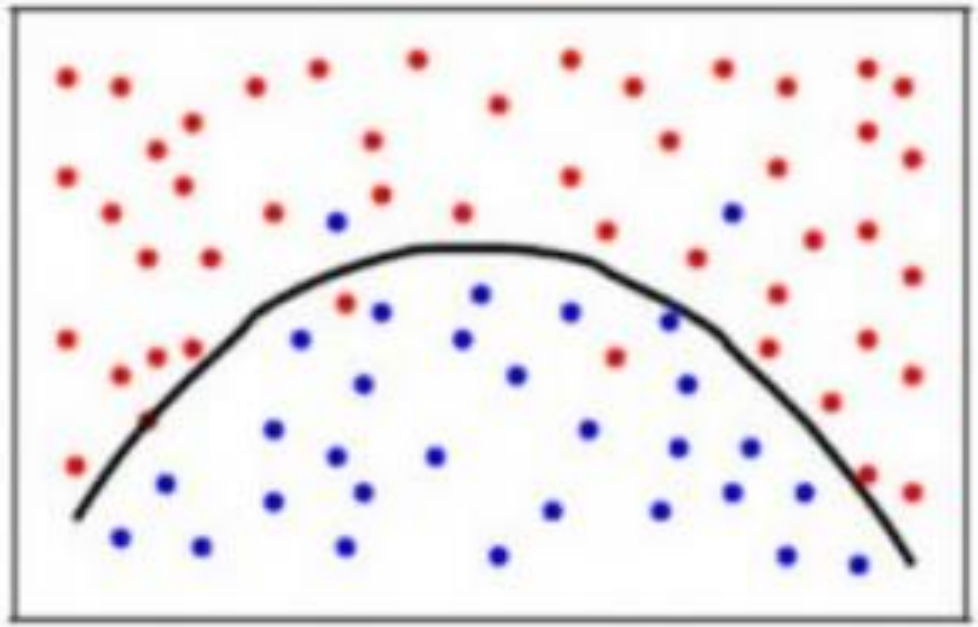
- When the model takes into account the fluctuations in the data – is called the Variance.
- The model learns too much from the training data.
- When testing with new data, it is unable to predict accurately based on it.
- In the case of high variance, the model learns too much from the training data, it is called **overfitting.**

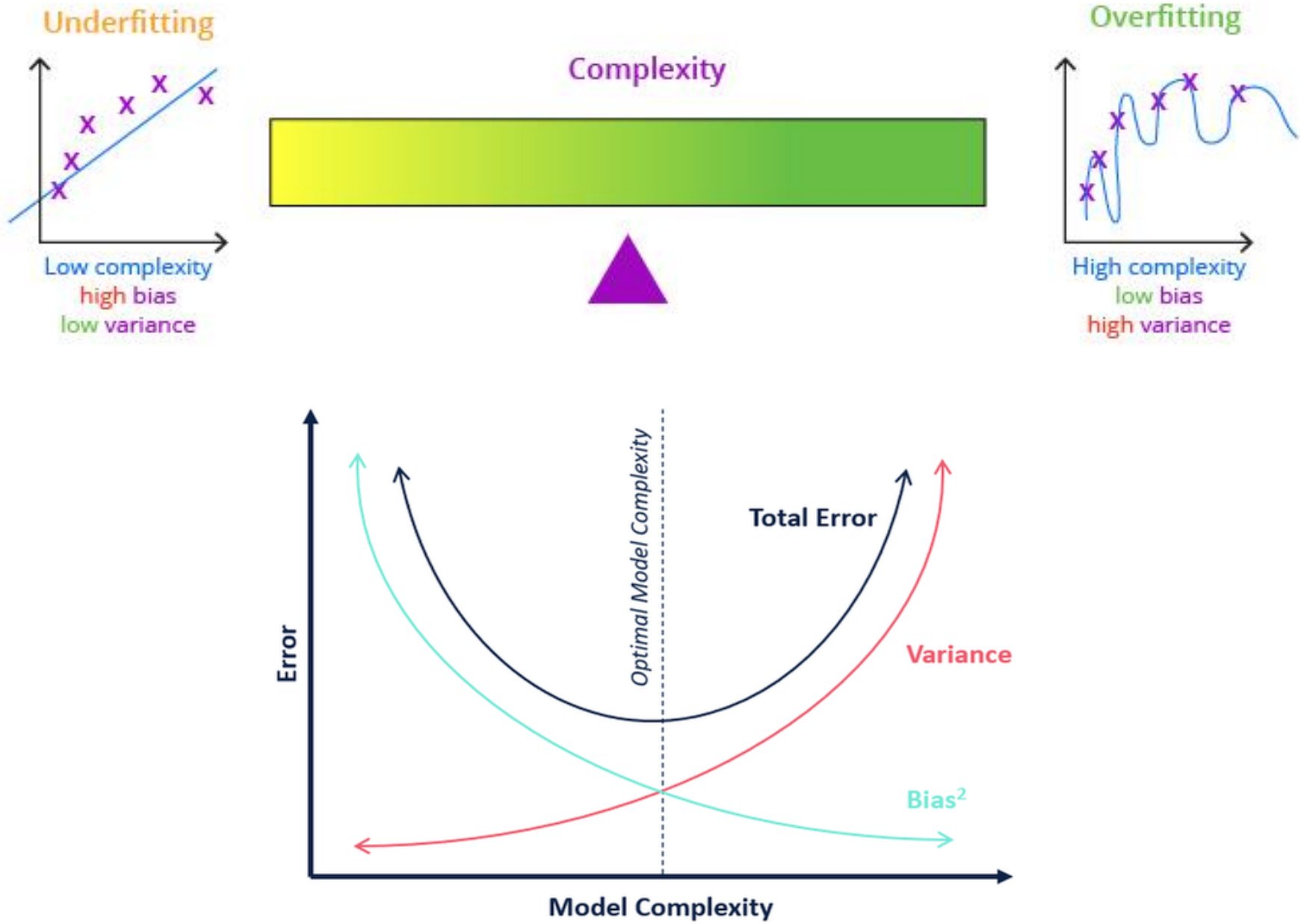


Bias-Variance Tradeoff

- Though some points are classified incorrectly, the model generally fits most of the data points accurately.

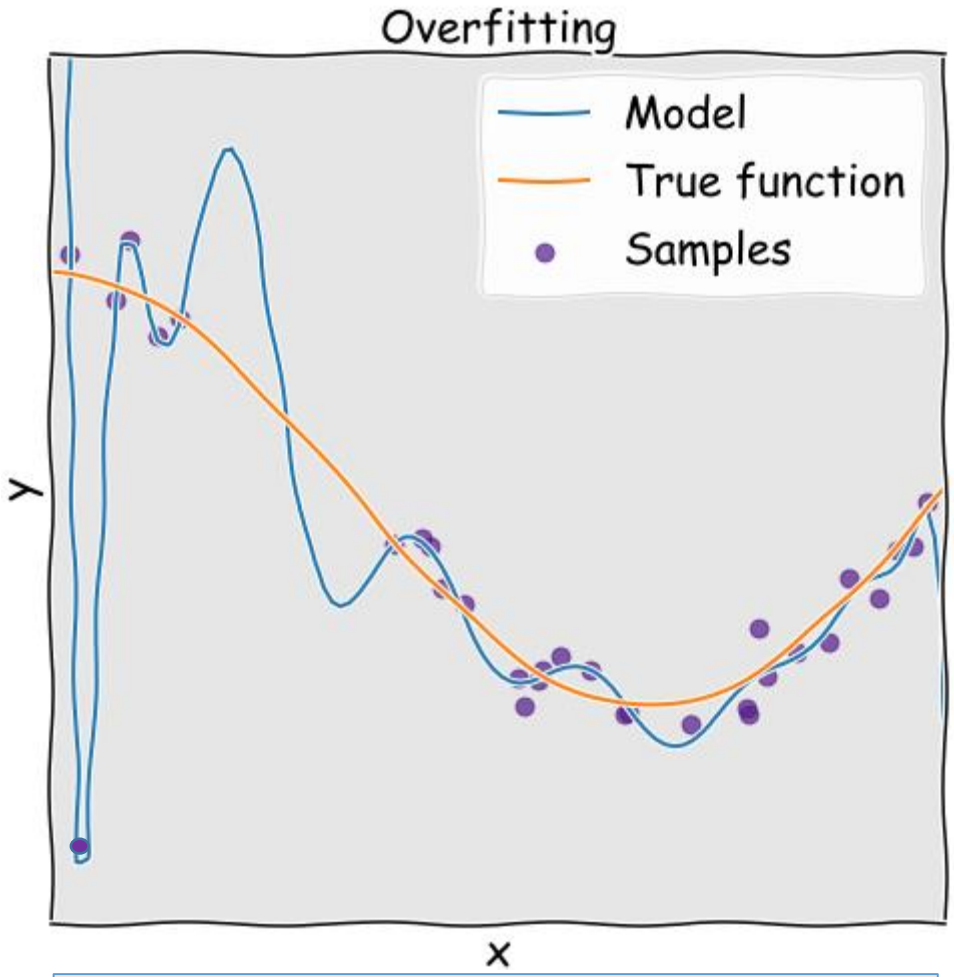
- The balance between the Bias error and the Variance error is the **Bias-Variance Tradeoff**.




Underfitting
Low complexity
high bias
low variance
Complexity
Overfitting
High complexity
low bias
high variance
Error
Optimal Model Complexity
Total Error
Variance
Bias²
Model Complexity

Overfitting

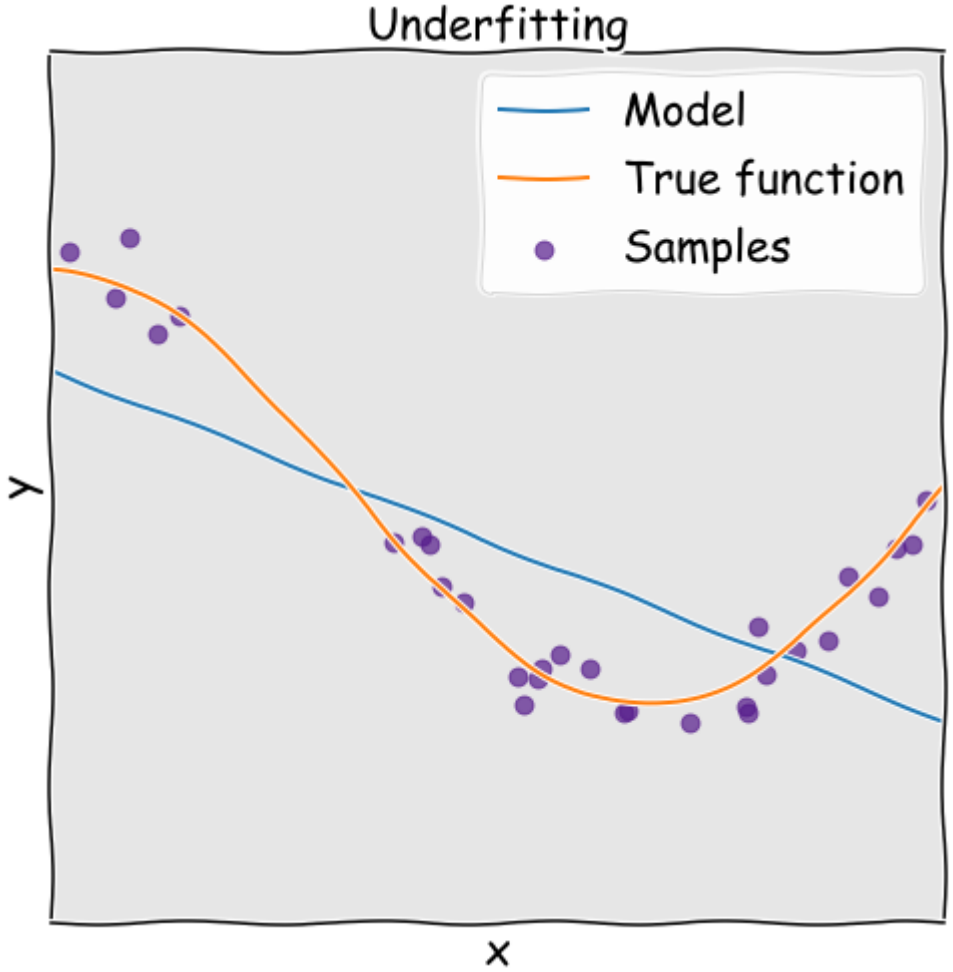
- When a model learns the pattern and noise in the data to such extent that it hurts the performance of the model on the new dataset, is termed overfitting.
- The model fits the data so well that it interprets noise as patterns in the data.
- Example: decision boundary is generated by non-linear models such as **decision trees**



Overfitting - the model function has too much complexity (parameters) to fit the true function correctly

Underfitting

- When the model neither learns from the training dataset nor generalizes well on the test dataset, it is termed as underfitting.
- If the performance is not good to try other models and you will certainly get good results.
- Hence, underfitting is not often discussed as often as overfitting is discussed.



Underfitting - the model function does not have enough complexity (parameters) to fit the true function correctly

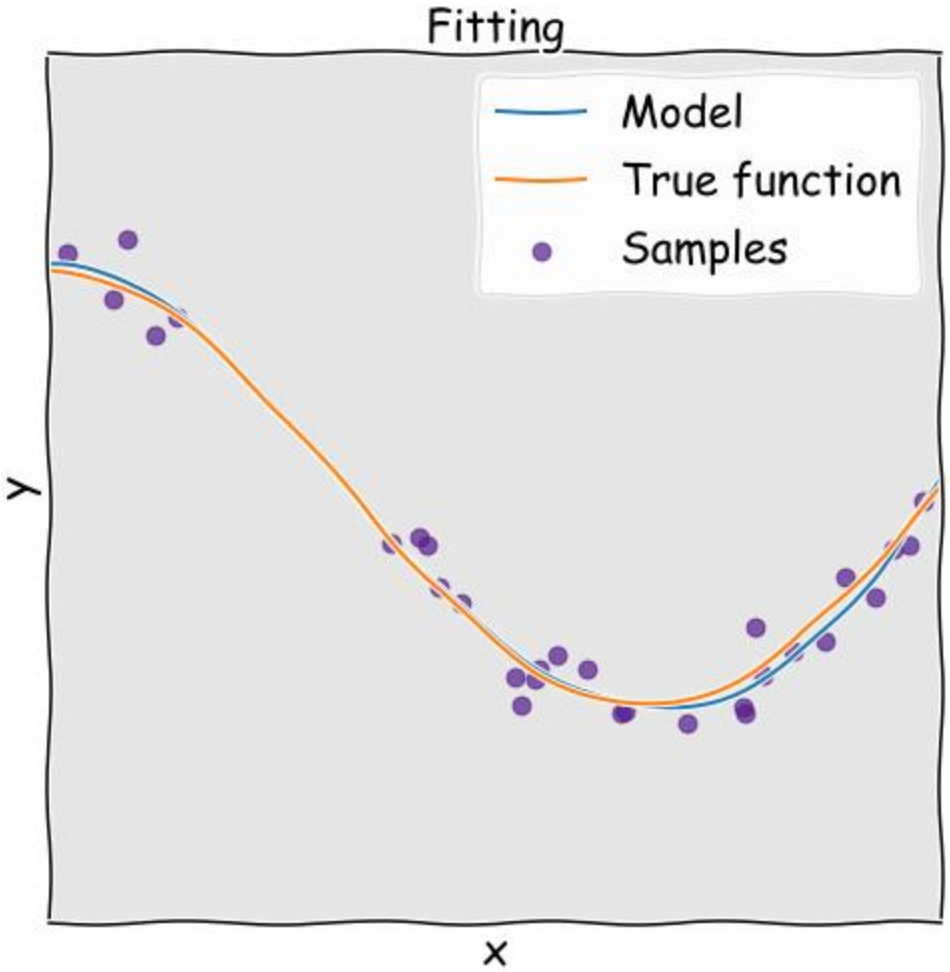
Good Fit

- *The spot in the middle of underfitting and overfitting is the good fit.*

- In the real world, getting a perfect fit model is very difficult.

- Overfitting:
  - Good performance on the training data, poor generalization to other data.

- Underfitting:
  - Poor performance on the training data and poor generalization to other data



An example of a well chosen model - the model function the right complexity (parameters) to fit the true function correctly.

How to tackle overfitting

- Prevent the model from overfitting by using techniques like **K-fold cross-validation and hyperparameter tuning**.

- Generally, we use K-fold cross-validation to do hyperparameter tuning.

K-fold cross-validation

- Divide the dataset into three parts.
- Training part : 80% of data
- Test part : 20% of data

- During training
  - Training dataset is divided into 80:20 ratio and
  - 80% of data is used for training and 20% is used for cross-validation.

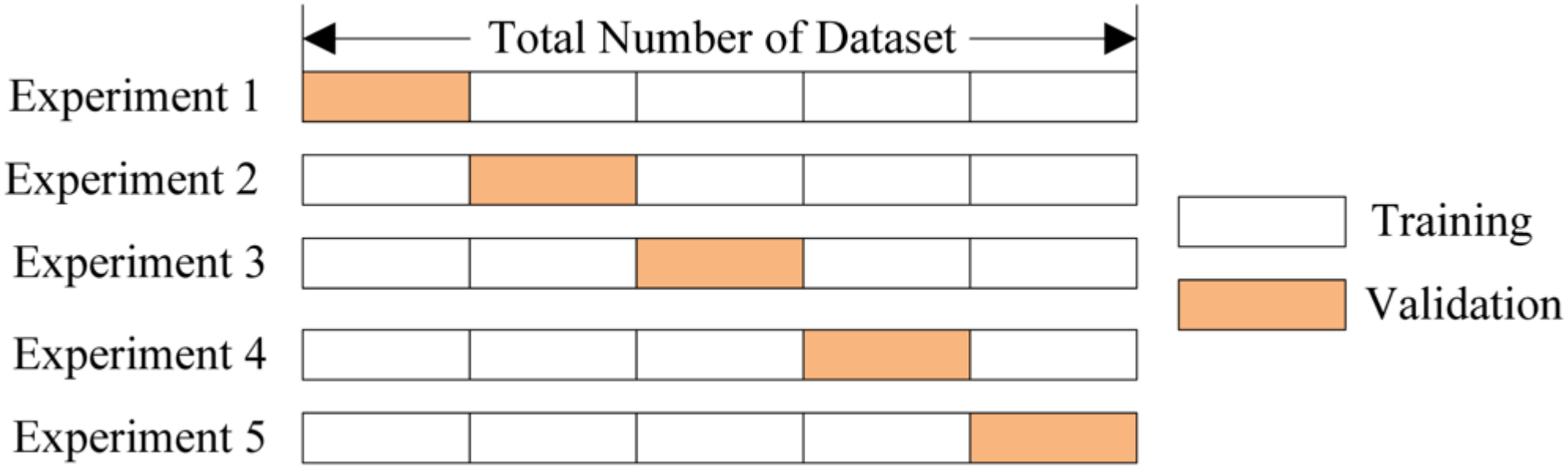
- In decision trees, hyperparameters are
  - **depth of the tree** and
  - number of data points at each node after which the node will be split.

How k-fold cross-validation works ?

- Example of 5-fold cross-validation.
- Training data set will be divided into 5 parts and
  - randomly 4 parts will be selected for training and 1 part for validation.

- This will be repeated 5 times and the average of the metric will be the final metric for that epoch.
- In one epoch the depth of the tree and split value is fixed.

- Example:
  - In an epoch, we can have the depth of the tree to be 100 and split value at 200.
- Let the training dataset D is divided into D1, D2, D3, D4, and D5.



Five-Fold Cross-Validation

How k-fold cross-validation works ?

Tree Depth	N-Split Value	Training Dataset	Validation Dataset	Metric Value	Final Metric Value
100	200	D1, D2, D3, D4	D5	a5	A1 = (a1+a2+a3+a4+a5)/5
100	200	D1, D2, D3, D5	D4	a4	
100	200	D1, D2, D4, D5	D3	a3	
100	200	D1, D3, D4, D5	D2	a2	
100	200	D2, D3, D4, D5	D1	a1	

- Let us take accuracy as a metric for now.
- After training A1 is the training accuracy.
- If both the training accuracy and test accuracy are close then the model has not overfit.

How k-fold cross-validation works ?

- If the training result is very good and the test result is poor then the model has **overfitted**.

- If the training accuracy and test accuracy is low then the model has **underfit**.

- If the model is underfitting or overfitting then we **change the value of the hyperparameter** and again retrain the model until we get a good fit.

Hyperparameter Tuning

- In this, we take a range of all the hyperparameter and then we monitor the cross-validation accuracy on all the possible combinations of hyperparameters.
- We take that hyperparameter that gave the best accuracy (here we have taken accuracy as a metric).
- Then we train the model with that hyperparameter and then test it.
- Following is how cross-validation accuracy is calculated for each hyperparameter combination.

Epoch	Tree Depth	N-Split Value	Training Dataset	Validation Dataset	Metric Value	Final Metric Value
1	100	200	D1, D2, D3, D4	D5	a5	A1 = (a1+a2+a3+a4+a5)/5
	100	200	D1, D2, D3, D5	D4	a4	
	100	200	D1, D2, D4, D5	D3	a3	
	100	200	D1, D3, D4, D5	D2	a2	
	100	200	D2, D3, D4, D5	D1	a1	
2	500	200	D1, D2, D3, D4	D5	a5	A2 = (a1+a2+a3+a4+a5)/5
	500	200	D1, D2, D3, D5	D4	a4	
	500	200	D1, D2, D3, D5	D3	a3	
	500	200	D1, D3, D4, D5	D2	a2	
	500	200	D2, D3, D4, D5	D1	a1	

Avoid Overfitting - Reduce Number of features

- For lower dimensional datasets
  - it is possible to plot the hypothesis to check if is overfit or not.
- When the dimensionality of the dataset increases beyond the limit of visualization
  - same strategy cannot be applied.
  - So, plotting hypothesis may not always work.
- Need other methods to address overfitting.

- There are two options to avoid the overfitting:
  - **Reduce the number of features**
  - **Regularization**

Reduce the number of features

- Mostly overfitting is observed when the **number of features is really high** and the number of training examples are less.
- In such cases **reducing the number of features** can help avoid the issue of overfitting.
- Reducing the number of features can be done manually, or using **model selection algorithms** which help decide which features to eliminate.
- **This also presents a disadvantage as removing features is sometimes equivalent to removing information.**

Avoid overfitting - Regularization

- Keep all the features but reduce magnitude/values of parameters θ_j.
- This technique works well, when there exists a lot features and each contributes a bit to the prediction,
- **Regularization works well when there are a lot of slightly useful features.**

Regularization

- So the main ideas in regularization are:
- maintain small values for the parameters $\theta_0,\theta_1,\cdots,\theta_n$
  - which keeps the hypothesis simple
  - and less prone to overfitting
- Mathematically, regularization is acheived by modifying the cost function as follows,

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

λ is a regularization parameter

If value of θ_j is increased, it would consequently increase the cost

λ controls the trade off between the goal of fitting the training set well and the goal of keeping the parameters small and the hypothesis simple.

if the value of λ is kept very large, it would fail to fit the dataset and would give a underfit hypothesis.

Thank you