# PART-1(Bias-Variance Tradeoff in Machine Learning)

## Impact of Increasing Model Complexity on Bias and Variance

As we increase the complexity of a machine-learning model by adding more features or including higher-order polynomial terms in a regression model, the effects on bias and variance can be described through the **bias-variance trade-off**.

### 1. Bias

Bias refers to the error introduced by approximating a real-world problem with a simpler model.

- Low-complexity models (e.g., linear models) typically have **high bias**, as they underfit the data and fail to capture underlying patterns. - As model complexity increases, bias **decreases**, since the model can fit more complex patterns.

### 2. Variance

Variance refers to the model's sensitivity to small fluctuations in the training data.

- High-complexity models, especially those with many features or higher-order terms, have **high variance** since they can overfit the training data, learning patterns that do not generalize well to unseen data. - As model complexity increases, variance **increases**, as the model becomes too sensitive to the specific training data.

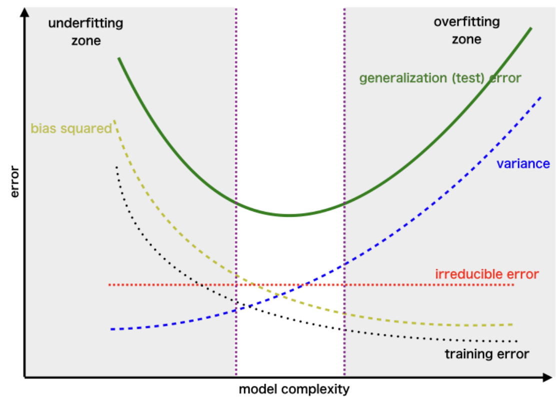
### 3. Bias-Variance Tradeoff

The bias-variance tradeoff represents the balance between bias and variance:

- Initially, as complexity increases, both training and test errors decrease due to a reduction in bias.
- After a certain point, increasing complexity results in overfitting, which causes an increase in test error due to higher variance.

# Graphical Representation

The bias-variance tradeoff can be represented graphically as shown below:




In this graph: - The **x-axis** represents model complexity. - The **y-axis** represents the error (or loss). - The **training error** decreases as complexity increases. - The **test error** initially decreases but then increases due to overfitting. - **Bias** starts high and decreases, while **variance** starts low and increases.

## Mathematical Formulation

The total error can be decomposed into three parts:

$$\text{Total Error} = \underbrace{\text{Bias}^2}_{\text{underfitting}} + \underbrace{\text{Variance}}_{\text{overfitting}} + \underbrace{\text{Irreducible Error}^2}_{\text{Noise}}$$

- **Bias**: Measures how far the model's predictions are from the true values.
- **Variance**: Measures the model's sensitivity to changes in the training data.
- **Irreducible error**: Represents the inherent noise in the data.

# PART-2(Email Filtering System Evaluation)

## Confusion Matrix

Based on the problem description, we can summarize the classification results in the following confusion matrix:

| | Predicted Spam | Predicted Legitimate |
|---|---|---|
| Actual Spam | True Positives (TP) = 200 | False Negatives (FN) = 50 |
| Actual Legitimate | False Positives (FP) = 20 | True Negatives (TN) = 730 |

## 1. Accuracy

Accuracy is the proportion of correctly classified emails (both spam and legitimate) out of the total number of emails:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Substituting the values:

$$\text{Accuracy} = \frac{200 + 730}{200 + 730 + 20 + 50} = \frac{930}{1000} = 0.93$$

Thus, the model's accuracy is **93%**.

## 2. Precision

Precision measures how many of the emails classified as spam were actually spam:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Substituting the values:

$$\text{Precision} = \frac{200}{200 + 20} = \frac{200}{220} \approx 0.909$$

So, the model's precision is **90.9%**.

## 3. Recall

Recall (also known as Sensitivity or True Positive Rate) measures how many actual spam emails were correctly classified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Substituting the values:

$$\text{Recall} = \frac{200}{200 + 50} = \frac{200}{250} = 0.8$$

Thus, the model's recall is **80%**.

## 4. F1-Score

The F1-score is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Substituting the values:

$$F1 = 2 \times \frac{0.909 \times 0.8}{0.909 + 0.8} = 2 \times \frac{0.7272}{1.709} \approx 0.851$$

Thus, the F1-score is **85.1%**.

## 5. False Positive Rate

The false positive rate measures how many legitimate emails were incorrectly classified as spam:

$$\text{FPR} = \frac{FP}{FP + TN}$$

Substituting the values:

$$\text{FPR} = \frac{20}{20 + 730} = \frac{20}{750} \approx 0.0267$$

Thus, the false positive rate is **2.67%**.

## Conclusion

The email filtering system has the following performance metrics:

- **Accuracy**: 93%
- **Precision**: 90.9%

- **Recall**: 80%
- **F1-Score**: 85.1%
- **False Positive Rate**: 2.67%

The model performs well overall, but there is a tradeoff between precision and recall. Only a small portion of legitimate emails are mistakenly flagged as spam, but the model could improve its ability to correctly classify all spam emails.

# PART-3(Finding the Equation of the Regression Line)

## Given Data

The given data is as follows:

| $x$ | $y$ |
|---|---|
| 3 | 15 |
| 6 | 30 |
| 10 | 55 |
| 15 | 85 |
| 18 | 100 |

The number of data points $n = 5$.

## Step 1: Calculate the Required Sums

We need to calculate the following sums:

$$\sum x = 3 + 6 + 10 + 15 + 18 = 52$$

$$\sum y = 15 + 30 + 55 + 85 + 100 = 285$$

$$\sum xy = (3 \times 15) + (6 \times 30) + (10 \times 55) + (15 \times 85) + (18 \times 100) = 45 + 180 + 550 + 1275 + 1800 = 3850$$

$$\sum x^2 = (3^2) + (6^2) + (10^2) + (15^2) + (18^2) = 9 + 36 + 100 + 225 + 324 = 694$$

## Step 2: Calculate the Slope ($m$) and Intercept ($c$)

The slope $m$ of the regression line is given by:

$$m = \frac{n \sum xy - \sum x \sum y}{n \sum x^2 - (\sum x)^2}$$

Substituting the values:

$$m = \frac{5 \times 3850 - 52 \times 285}{5 \times 694 - 52^2}$$

$$m = \frac{19250 - 14820}{3470 - 2704} = \frac{4430}{766} \approx 5.78$$

The y-intercept $c$ is calculated using the formula:

$$c = \frac{\sum y - m \sum x}{n}$$

Substituting the values:

$$c = \frac{285 - 5.78 \times 52}{5} = \frac{285 - 300.56}{5} = \frac{-15.56}{5} \approx -3.11$$

## Step 3: The Equation of the Regression Line

Thus, the equation of the regression line is:

$$y = 5.78x - 3.11$$

## Step 4: Using the Regression Line

To predict the value of $y$ for a given $x$, substitute the value of $x$ into the regression line equation. For example, for $x = 12$:

$$y = 5.78 \times 12 - 3.11 = 69.36 - 3.11 = 66.25$$

Thus, the predicted value of $y$ when $x = 12$ is approximately 66.25.

# PART-4 (Toy Example: Empirical Risk and Generalization)

## Problem

Given a training dataset with features $X$ and labels $Y$, let $\hat{f}(X)$ be the prediction of a model $f$, and let $L(\hat{f}(X), Y)$ be the loss function. Suppose we have two models, $f_1$ and $f_2$, and the empirical risk for $f_1$ is lower than that for $f_2$. We provide a toy example where model $f_1$ has a lower empirical risk on the training set but may not necessarily generalize better than model $f_2$.

## Toy Example

Consider a small dataset for a regression task:

| $X$ | $Y$ |
|---|---|
| 1 | 1.5 |
| 2 | 2.0 |
| 3 | 2.5 |
| 4 | 3.5 |
| 5 | 5.0 |

We will now create two models, $f_1$ and $f_2$, and analyze their empirical risk and generalization ability.

## Model $f_1$: High Complexity (Overfitting)

Model $f_1$ is a high-degree polynomial regression, such as a 4th-degree polynomial. It fits the training data perfectly, with an equation that could look like:

$$f_1(X) = 0.05X^4 - 0.3X^3 + 0.7X^2 - 0.4X + 1.5$$

The predicted values of $f_1$ on the training set are:

$$\hat{Y}_1 = [1.55, 1.94, 2.38, 3.12, 5.01]$$

The empirical risk, computed as the Mean Squared Error (MSE), for $f_1$ is:

$$MSE(f_1) = 0.033$$

Although the empirical risk is very low, this model is highly complex and likely overfits the noise in the data, which means it may not generalize well to unseen data.

## Model $f_2$: Low Complexity (Better Generalization)

Model $f_2$ is a simple linear regression model, capturing the overall trend in the data. Its equation could be:

$$f_2(X) = 0.85X + 0.75$$

The predicted values of $f_2$ on the training set are:

$$\hat{Y}_2 = [1.60, 2.45, 3.30, 4.15, 5.00]$$

The empirical risk (MSE) for $f_2$ is:

$$MSE(f_2) = 0.255$$

Despite having a higher empirical risk on the training data, $f_2$ is likely to generalize better because it avoids overfitting and has lower variance.

## Testing the Models

To further support our claim, we test both models on a new data point that was not part of the training set, say $X = 6$, where the true value of $Y$ is $Y = 6.0$.

For model $f_1$:

$$f_1(6) = 0.05(6)^4 - 0.3(6)^3 + 0.7(6)^2 - 0.4(6) + 1.5 = 9.23$$

For model $f_2$:

$$f_2(6) = 0.85(6) + 0.75 = 5.85$$

We observe that model $f_1$ predicts 9.23, which is much farther from the true value of 6.0, while model $f_2$ predicts 5.85, which is closer to the true value.

Thus, the generalization ability of $f_2$ is better on unseen data, even though its empirical risk on the training set is higher.

## Conclusion

In this example, model $f_1$ has a lower empirical risk on the training set because it overfits the data, but it does not generalize as well as model $f_2$. Testing on a new data point further shows that model $f_2$ provides a better prediction. This illustrates that a lower training error does not necessarily imply better performance on unseen data, especially when the model is overly complex and prone to overfitting.