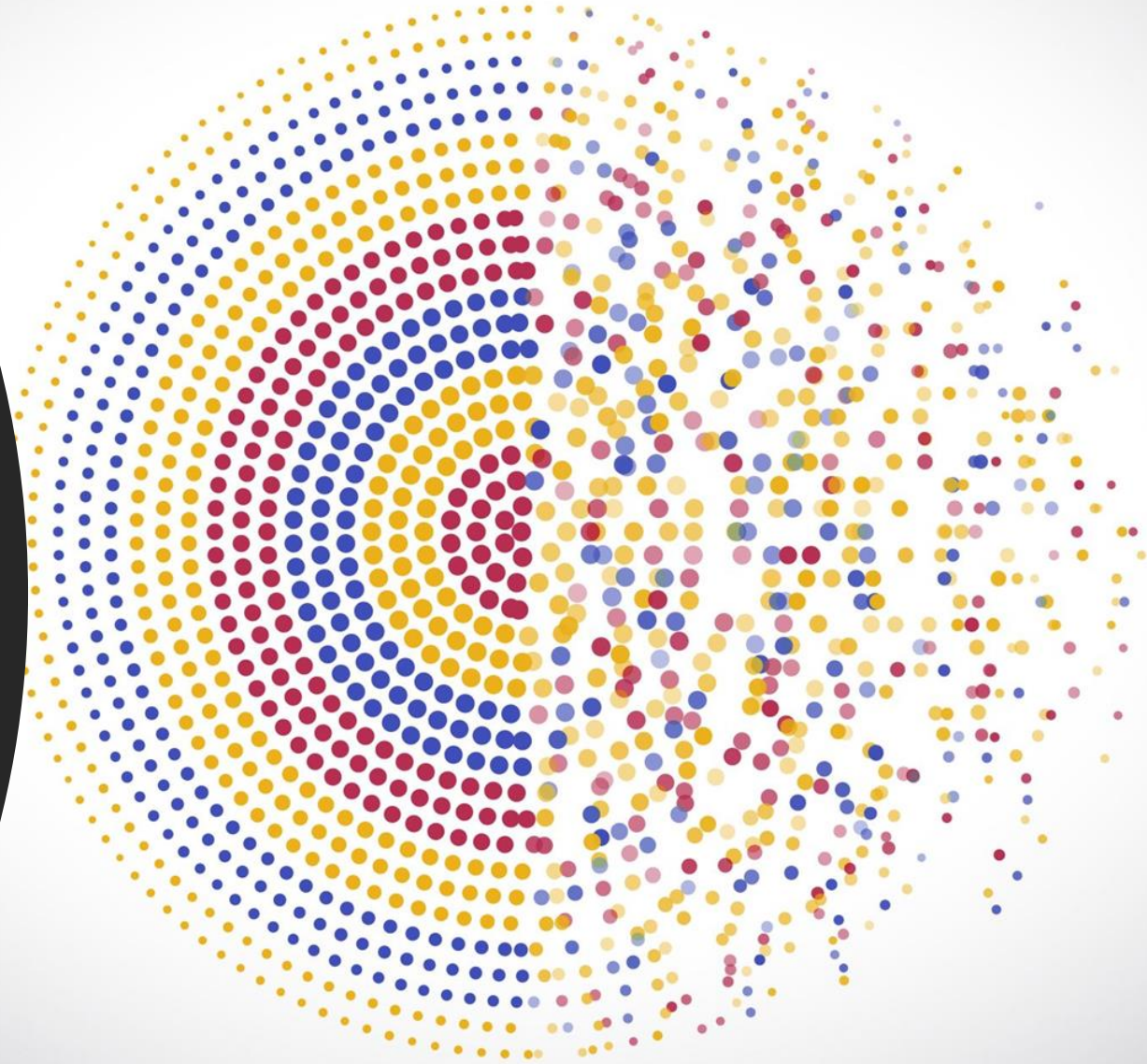
Model Evaluation

Dr. Mohamed AlHajri



Content

- Training, Validation, and Testing Dataset
- Overfitting and Underfitting
- K-fold cross-validation
- Model Evaluation

Training, Validation, and Testing

- Training Data: Used to train the model, allowing it to learn the patterns and relationships in the data. The model optimizes its parameters by minimizing the **training error**.

  Training Error: *The error (or loss) calculated on the training data. If the model fits the training data well, this error will be low.*

- Validation Data: A separate subset of data not used during the initial training process. Its primary purpose is to fine-tune the model's hyperparameters and monitor model performance to prevent overfitting.

  Validation Error: *The error calculated on the validation data. This error helps monitor how well the model generalizes beyond the training data.*

- Test Data: After the model is trained and validated, the test data is used to evaluate the model's performance on unseen data, providing a final estimate of the model's generalization ability.

Training, Validation, and Testing

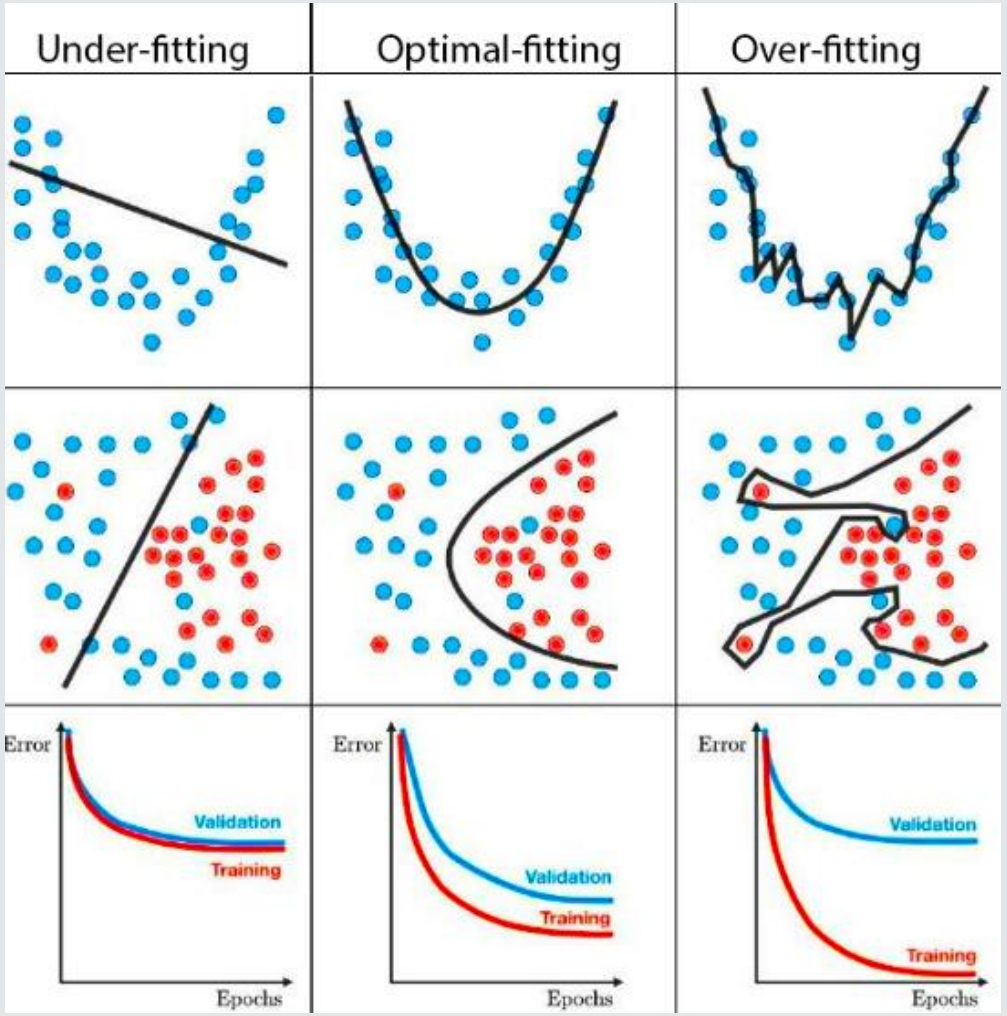
- Suppose we have a dataset of 1000 samples. We might split the data into:
  - *70% for training (700 samples)*
  - *15% for validation (150 samples)*
  - *15% for testing (150 samples)*

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import numpy as np
# Load dataset
iris = load_iris()
X, y = iris.data, iris.target
# Split data: 70% train, 15% validation, 15% test
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
print(f"Training set size: {len(X_train)}")
print(f"Validation set size: {len(X_val)}")
print(f"Test set size: {len(X_test)}")
```

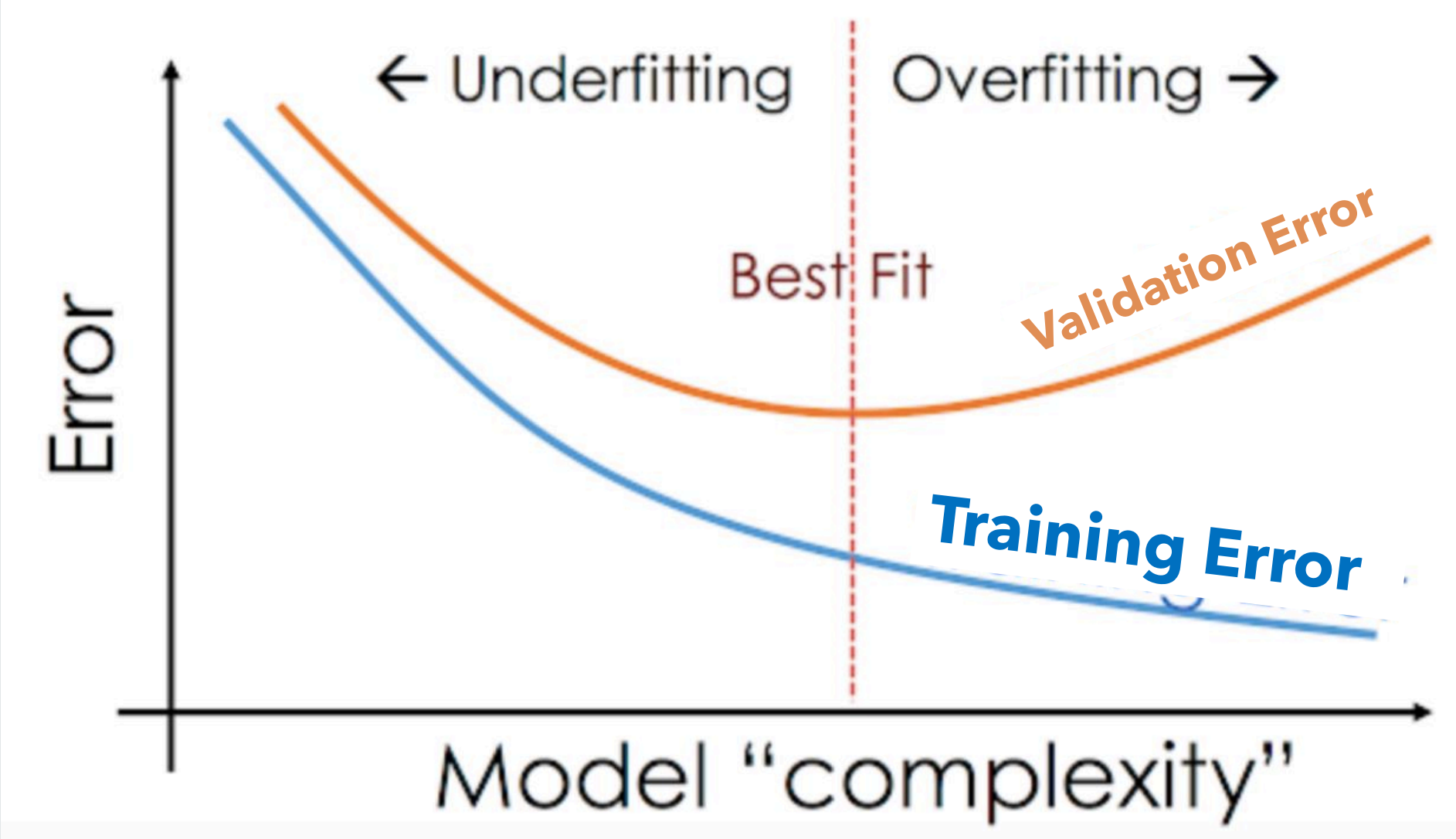
Underfitting and Overfitting

- **Underfitting:** A situation where the model is too simple to capture the underlying structure of the data. It results in **both high training and testing error**.

- **Overfitting:** A situation where the model becomes too complex and captures noise in the training data, which leads to **low training error** but **high testing error** due to poor generalization.

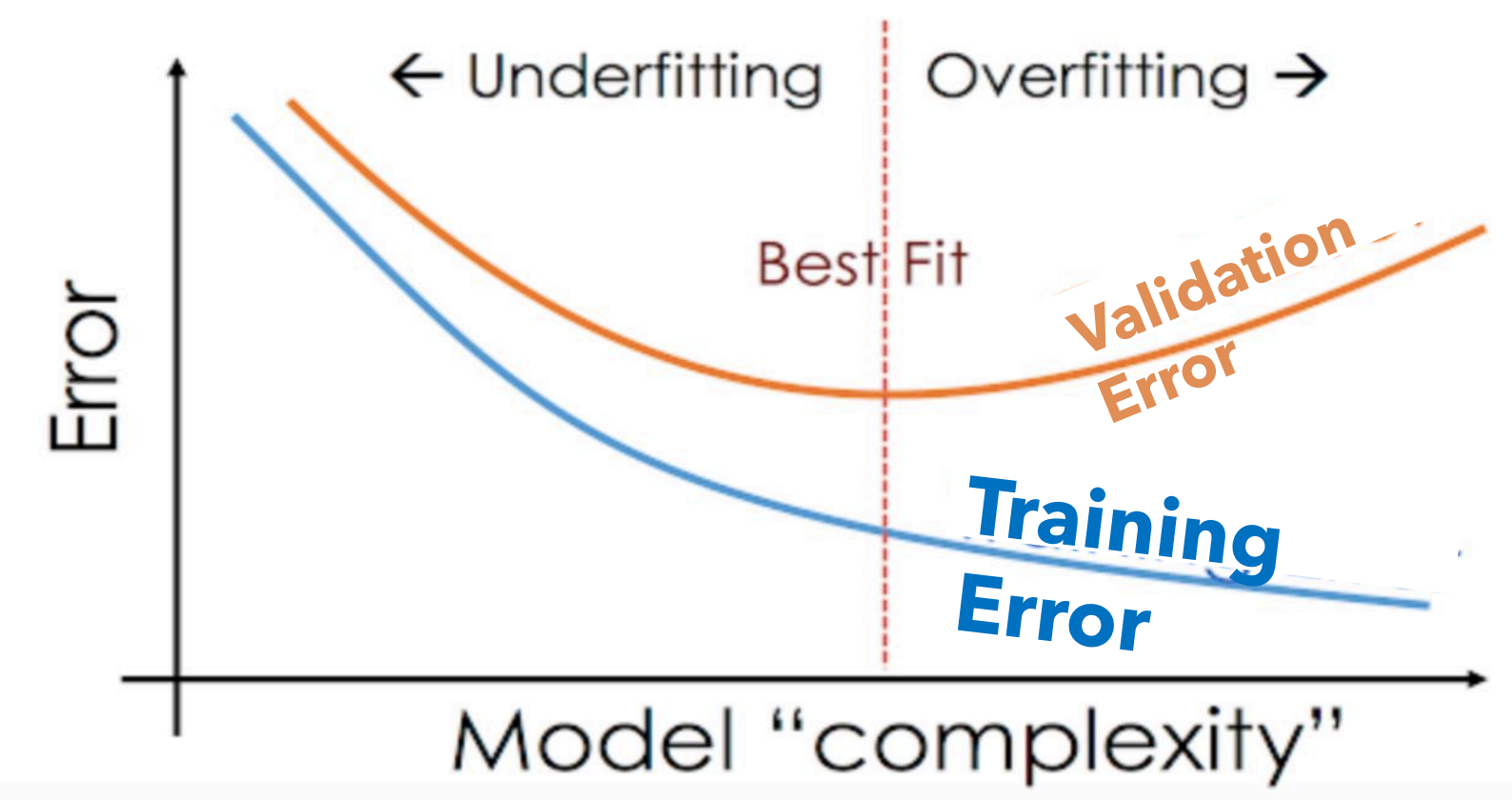


Underfitting and Overfitting



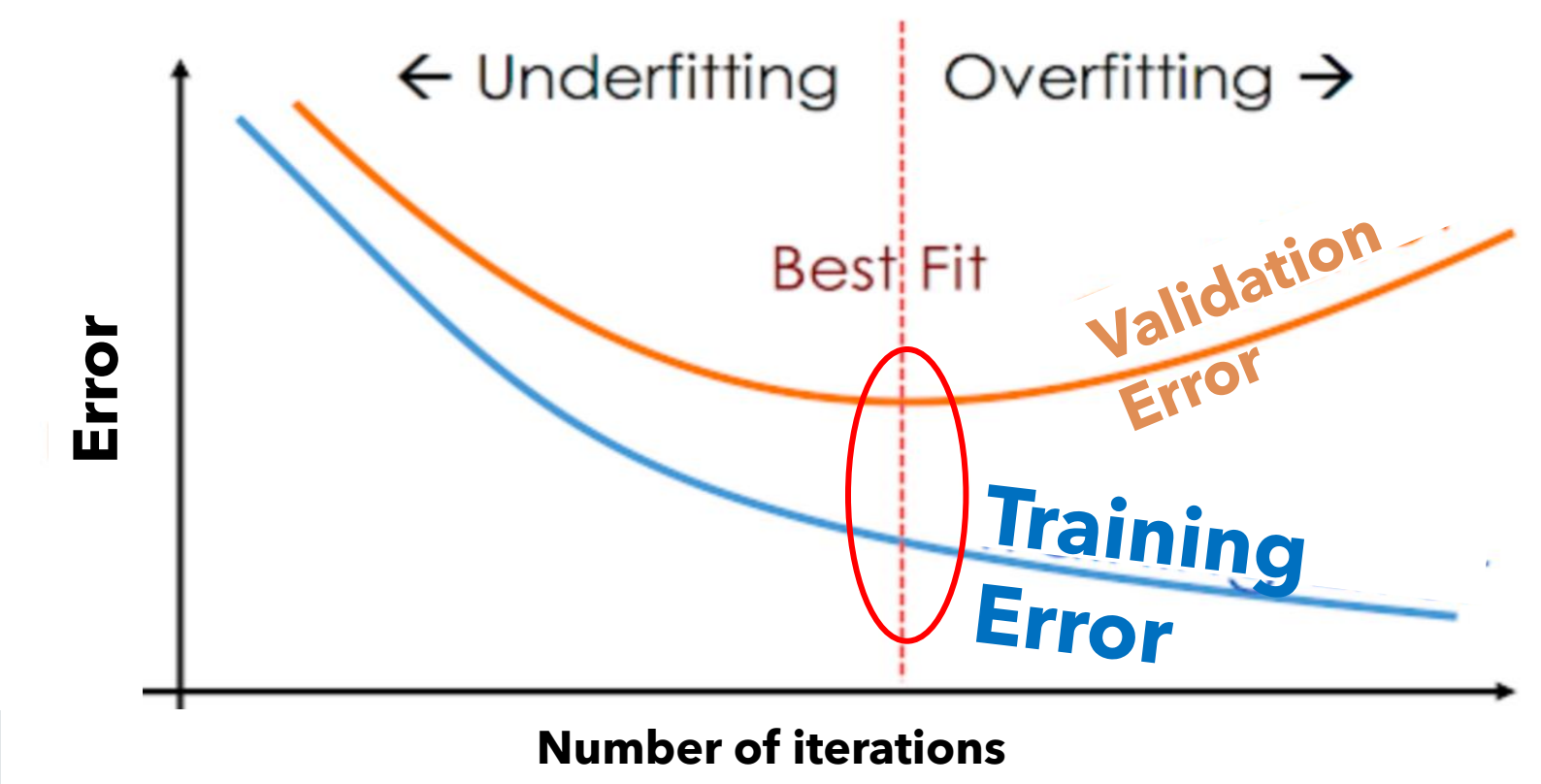
Importance of Validation Dataset

- **Prevent Overfitting**



Importance of Validation Dataset

- **Early Stopping**



Importance of Validation Dataset

- **Model Selection**

Model	Training Error (%)	Validation Error (%)	
Model 1	12	11	Lowest Validation Error
Model 2	17	18	
Model 3	6	19	Overfitting
Model 4	23	23.5	
Model 5	11	17	Overfitting

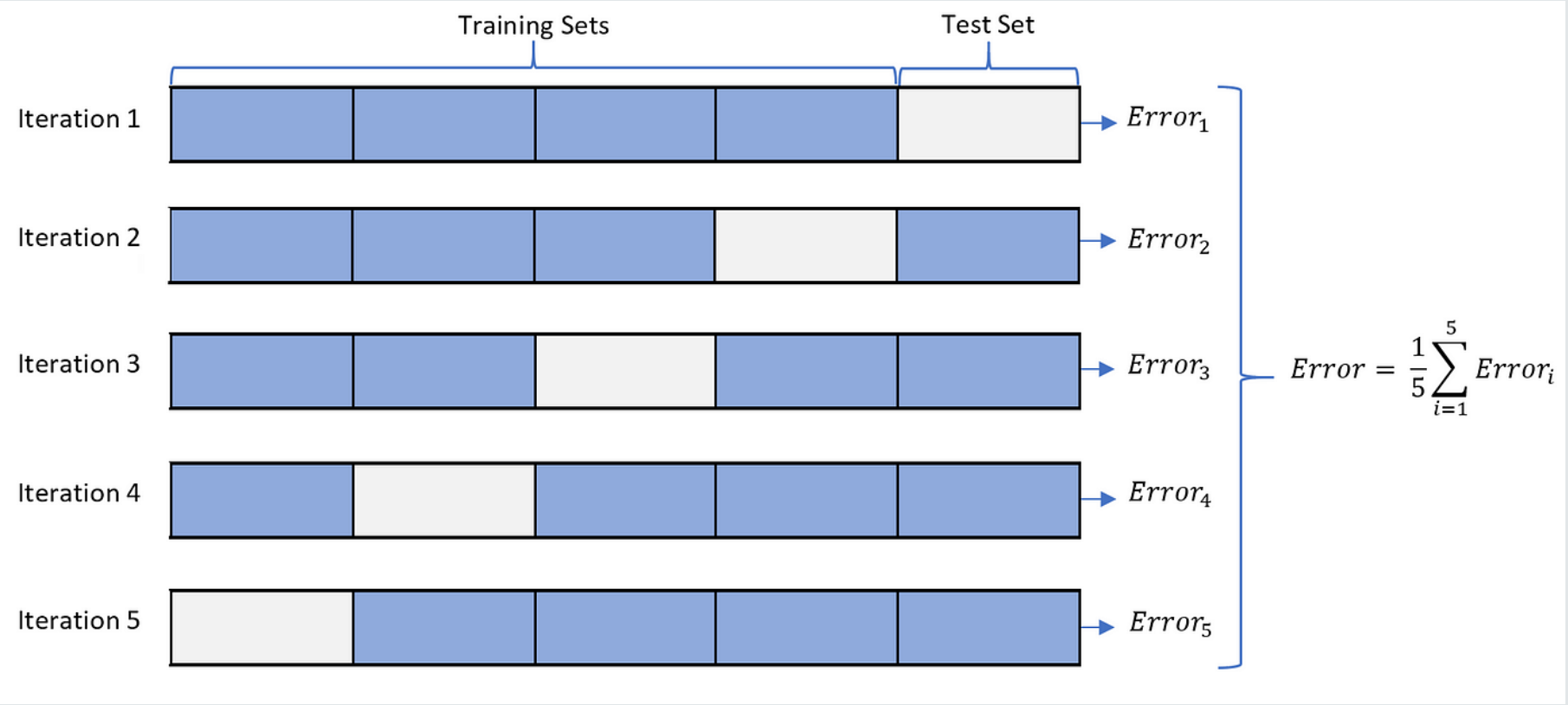
Importance of Validation Dataset

- **Model Tuning**

Parameters	Training Error (%)	Validation Error (%)	
Model 1 (5,7)	12	11	Lowest Validation Error
Model 1 (3,6)	17	18	
Model 1 (2,3)	6	19	Overfitting
Model 1 (1,1)	23	23.5	
Model 1 (3,9)	11	17	Overfitting

K-fold Cross Validation

- K-Fold Cross-Validation: This technique splits the dataset into K equally sized subsets or folds. The model is trained K times, using a different fold as the validation set and the remaining $K-1$ folds for training. The final performance metric is the average of the results across all folds.



K-fold Cross Validation

- **More Reliable Performance Estimates:** K-fold cross-validation evaluates the model on different subsets of the data, providing a more reliable estimate of how the model will perform on unseen data. This is especially helpful when data is limited.
- **Reduced Overfitting Risk:** Since the model is evaluated on different portions of the data, K-fold cross-validation helps detect overfitting. If the model performs consistently well across folds, it is less likely to be overfitting to a specific subset of the data.
- **Better Model Selection:** By evaluating the model on multiple folds, K-fold cross-validation provides more reliable metrics for comparing different models or hyperparameter configurations, ensuring the selected model is likely to generalize well.

Evaluation Metrics

- Model evaluation is critical for understanding how well a machine learning model performs on both training and unseen data. This section covers essential metrics used to evaluate classification and regression models, along with their mathematical formulations, limitations, and detailed numerical examples.

Evaluation Metrics - Classification

- **Accuracy**: measures the fraction of correct predictions over the total predictions.
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

Where:

- **TP (True Positives)**: Correctly predicted positive cases.
- **TN (True Negatives)**: Correctly predicted negative cases.
- **FP (False Positives)**: Incorrectly predicted positive cases.
- **FN (False Negatives)**: Incorrectly predicted negative cases.
- **Limitation**: Accuracy can be misleading for imbalanced datasets, where one class dominates the data.

Evaluation Metrics - Classification

- **Precision:** focuses on the proportion of true positives among all predicated positives.

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

$$Type\ I\ Error = 1 - Precision$$

- **Limitation:** Precision is not enough when false negatives are important, as it doesn't capture how many actual positives are missed

Evaluation Metrics - Classification

- **Recall:** measures the model's ability to capture actual positive cases.

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

$$Type\ II\ Error = 1 - Recall$$

- **Limitation:** High recall can come at the expense of precision, meaning the model might predict many false positives

Evaluation Metrics - Classification

- F_1 score is the harmonic mean of precision and recall, providing a balance between the two.

$$F_1 = 2\frac{Precision.Recall}{Precision + Recall}$$

- **Limitation:** The F_1 score does not account for true negatives and may not provide a complete picture in highly imbalanced datasets.

Evaluation Metrics - Classification

- **Why the harmonic mean?**
- The **harmonic mean** is used in the F_1 **score** because it provides a more balanced measure when combining **precision** and **recall**, especially when one of the values is much smaller than the other. The harmonic mean penalizes extreme values more than the arithmetic mean or geometric mean, ensuring that both precision and recall contribute meaningfully to the final score.

Evaluation Metrics - Classification

- **Arithmetic Mean and its Limitation:**
- The arithmetic mean of precision and recall would give equal weight to both metrics. However, it might be misleading when one value is significantly lower than the other, as it wouldn't properly account for the imbalance between the two.
- For example, if $precision = 0.1$ and $recall = 0.9$, the arithmetic mean would be:

$$Arithmetic\ Mean = \frac{0.1 + 0.9}{2} = 0.5$$

- This value suggests a balanced performance, even though precision is very low.

Evaluation Metrics - Classification

- **Geometric Mean** is used for **multiplicative** processes, especially when growth or returns are compounded.
- For example, if $precision = 0.1$ and $recall = 0.9$, the arithmetic mean would be:

$$Geometric\ Mean = \sqrt{0.9 * 0.1} = 0.3$$

- **Harmonic Mean :**
- The harmonic mean of precision and recall.
- For example, if $precision = 0.1$ and $recall = 0.9$, the arithmetic mean would be:

$$Harmonic\ Mean = 2 * \frac{0.9 * 0.1}{0.9 + 0.1} = 0.18$$

Equality holds when *recall = precision*

$$Harmonic\ mean \leq Geometric\ mean \leq Arithmetic\ mean$$

Evaluation Metrics - Classification

- Numerical Example 1:

	Predicated Positive	Predicted Negative
Actual Positive	TP = 50	FN = 10
Actual Negative	FP = 10	TN = 50

$$Accuracy = \frac{50 + 50}{50 + 50 + 10 + 10} = \frac{100}{120} = 0.833$$

$$Precision = \frac{50}{50 + 10} = \frac{50}{60} = 0.833$$

$$Recall = \frac{50}{50 + 10} = \frac{50}{60} = 0.833$$

$$F_1 = 2\frac{0.833 \ * 0.833}{0.833 + 0.833} = 0.833$$

Evaluation Metrics - Classification

- Numerical Example 2 (High Precision, Low Recall):

	Predicated Positive	Predicted Negative
Actual Positive	TP = 10	FN = 40
Actual Negative	FP = 2	TN = 48

$$Accuracy = \frac{10 + 48}{10 + 48 + 40 + 2} = \frac{58}{100} = 0.58$$

$$Precision = \frac{10}{10 + 2} = \frac{10}{12} = 0.833$$

High precision doesn't reflect the fact that the model misses many positive cases (low recall).

$$Recall = \frac{10}{40 + 10} = \frac{10}{50} = 0.2$$

The model fails to capture most actual positives, as indicated by the low recall.

$$F_1 = 2\frac{0.833 * 0.833}{0.833 + 0.833} = 0.833$$

The low F1 score reflects the trade-off between high precision and low recall.

Evaluation Metrics - Classification

- Numerical Example 3 (High Recall, Low Precision):

	Predicated Positive	Predicted Negative
Actual Positive	TP = 45	FN = 5
Actual Negative	FP = 40	TN = 10

$$Accuracy = \frac{45 + 10}{45 + 10 + 5 + 40} = \frac{55}{100} = 0.55$$

$$Precision = \frac{45}{45 + 40} = \frac{45}{85} = 0.529$$

$$Recall = \frac{45}{45 + 5} = \frac{45}{50} = 0.9$$

$$F_1 = 2\frac{0.529 \ * 0.9}{0.529 + 0.9} = 0.666$$

Evaluation Metrics - Classification

- Numerical Example 4 (Swap FN with FP):

	Predicated Positive	Predicted Negative
Actual Positive	TP = 45	FN = 40
Actual Negative	FP = 5	TN = 10

$$Accuracy = \frac{45 + 10}{45 + 10 + 5 + 40} = \frac{55}{100} = 0.55$$

$$Precision = \frac{45}{45 + 5} = \frac{45}{50} = 0.9$$

$$Recall = \frac{45}{45 + 40} = \frac{45}{85} = 0.529$$

$$F_1 = 2\frac{0.529 \ * 0.9}{0.529 + 0.9} = 0.666$$

Weighted F1 Score can solve this probelm by having different weights on the recall and precision

$$F_\beta = (1 + \beta^2)\frac{Precision.Recall}{(\beta^2 Precision) + Recall}$$

$$F_2 = 5\frac{0.529 * 0.9}{(4 * 0.9) + 0.529} = 0.576$$

$$F_2 = 5\frac{0.529 * 0.9}{(4 * 0.529) + 0.9} = 0.789$$

Evaluation Metrics - Regression

- **Mean Squared Error (MSE)** is the average of the squared differences between the predicted values and the actual values.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i is the actual value, $\hat{y}_i$ is the predicated value, and N is the number of data points

- **Limitation**: MSE heavily penalizes large errors, making it sensitive to outliers.

Evaluation Metrics - Regression

- **Mean Absolute Error (MAE)** is the average of the absolute differences between the predicted values and the actual values.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

where y_i is the actual value, $\hat{y}_i$ is the predicated value, and N is the number of data points

Evaluation Metrics - Regression

- Numerical Example 1
- Suppose we have the following actual and predicted values:

$$y = [3, 0.5, 2, 7]$$
$$\hat{y} = [2.5, 0, 2, 8]$$

- **MSE:**

$$MSE = \frac{1}{4}[(3 - 2.5)^2 + (0.5 - 0)^2 + (2 - 2)^2 + (7 - 8)^2)] = 0.375$$

- **MAE:**

$$MAE = \frac{1}{4}[|3 - 2.5| + |0.5 - 0| + |2 - 2| + |7 - 8|] = 0.5$$

Evaluation Metrics - Regression

- Numerical Example 2 (Effect of Outlier)
- Suppose we have the following actual and predicted values:

$$y = [3, 0.5, 2, 7, 100]$$

$$\hat{y} = [2.5, 0, 2, 8, 8]$$

- **MSE:**

$$MSE = \frac{1}{5}[(3 - 2.5)^2 + (0.5 - 0)^2 + (2 - 2)^2 + (7 - 8)^2 + (100 - 8)^2)] = 1693.1$$

- **MAE:**

$$MAE = \frac{1}{5}[|3 - 2.5| + |0.5 - 0| + |2 - 2| + |7 - 8| + |100 - 8|] = 18.8$$