CADT
Cambodia Academy of Digital Technology

Training and Evaluation

Lecturer: Soklay HENG

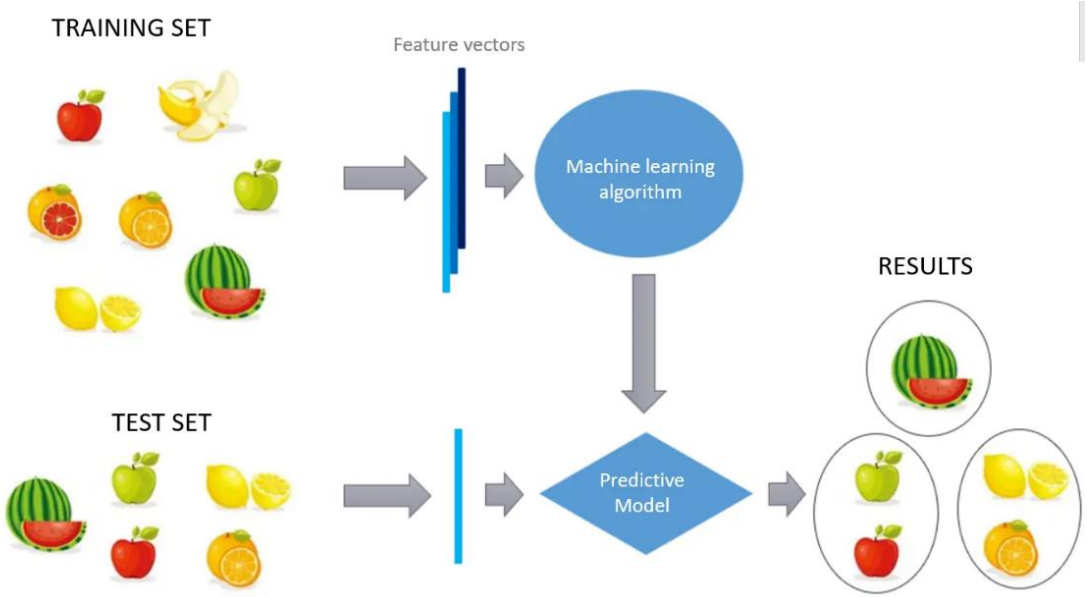
Academic Year: 2026-2027

Content

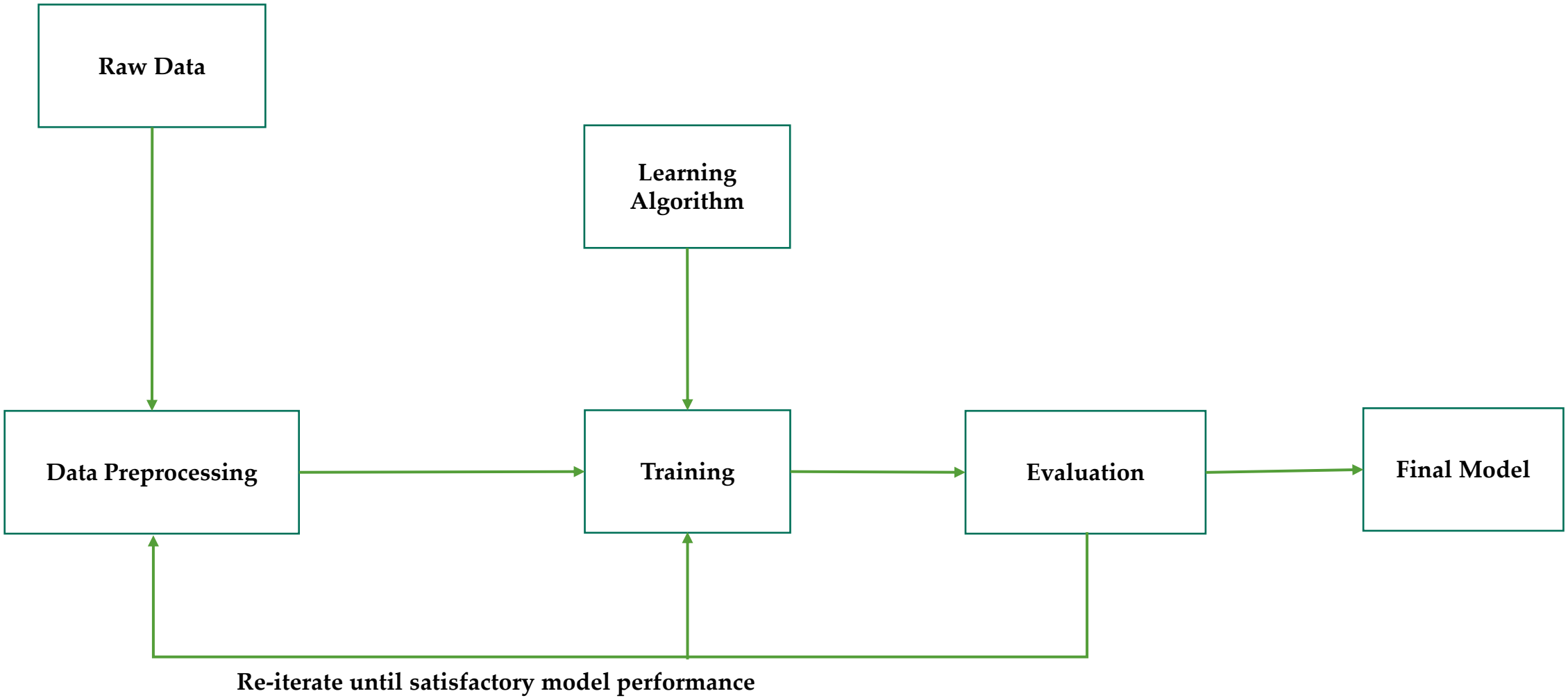
1. Machine Learning Phases
2. Training
3. Evaluation

1. Machine Learning Phases

- **Learning Phase** or **Training Phase** involves main steps including data preprocessing, training and evaluation, which aims to produce the final model with respect to the requirement.
- **Prediction Phase**, **Testing Phase** or **Inference Phase** refers to when the final model is used to predict the new unseen data.

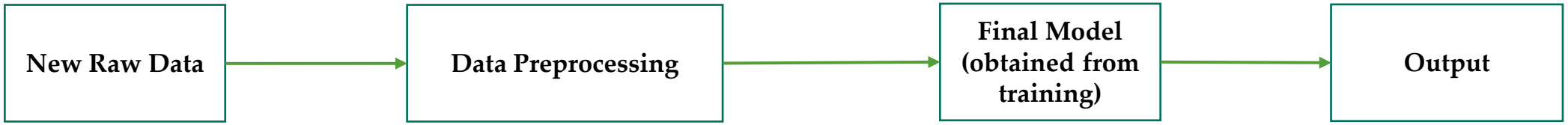


1.1 Learning Phase



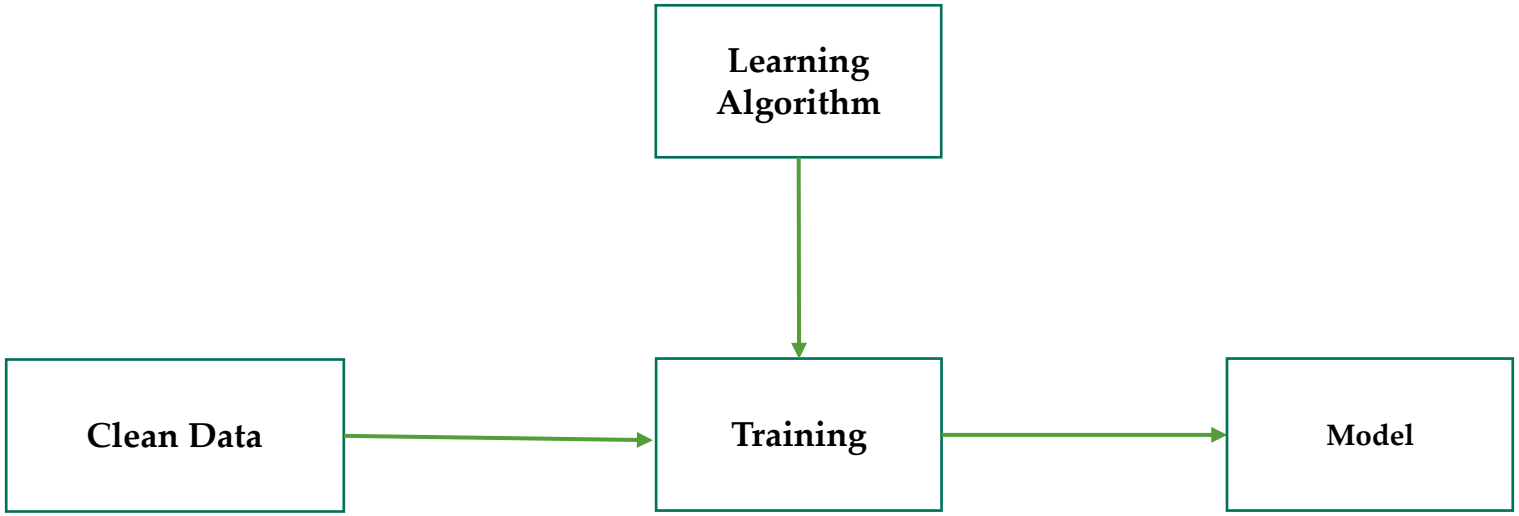
1.2 Prediction Phase

- When testing with the new data, these data are required to be preprocessed with the same methods that are used in Learning Phase, otherwise both data will be inconsistent.



2. Training

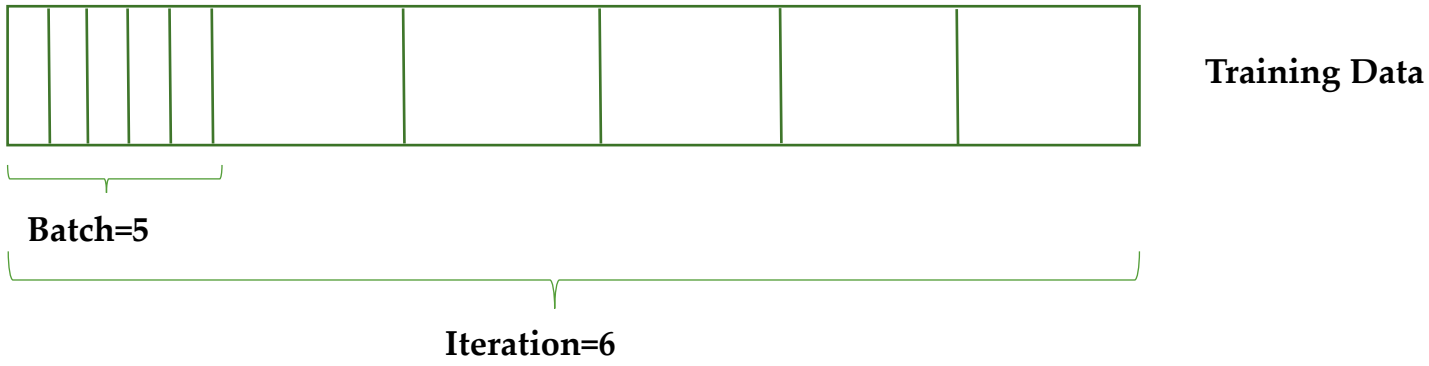
- The training step can be done when the learning algorithms are defined and the data is ready after being analyzed and preprocessed, with respect to our constrains and problem.
- We will take a look some **terminologies** and the **two approaches** which are used for training the model and understand their advantages and downsides.



2.1 Batch vs. Iteration vs. Epoch

- One **epoch** is when the entire training dataset is completely iterated.
- **Batch** refers to the collection of examples which are presented during each step of the training.
- **Iteration** refers to the number of batches which are needed to complete in one epoch.

Example: we can divide the training data of 30 examples into batches of 5 then it will take 6 iterations to complete 1 epoch.

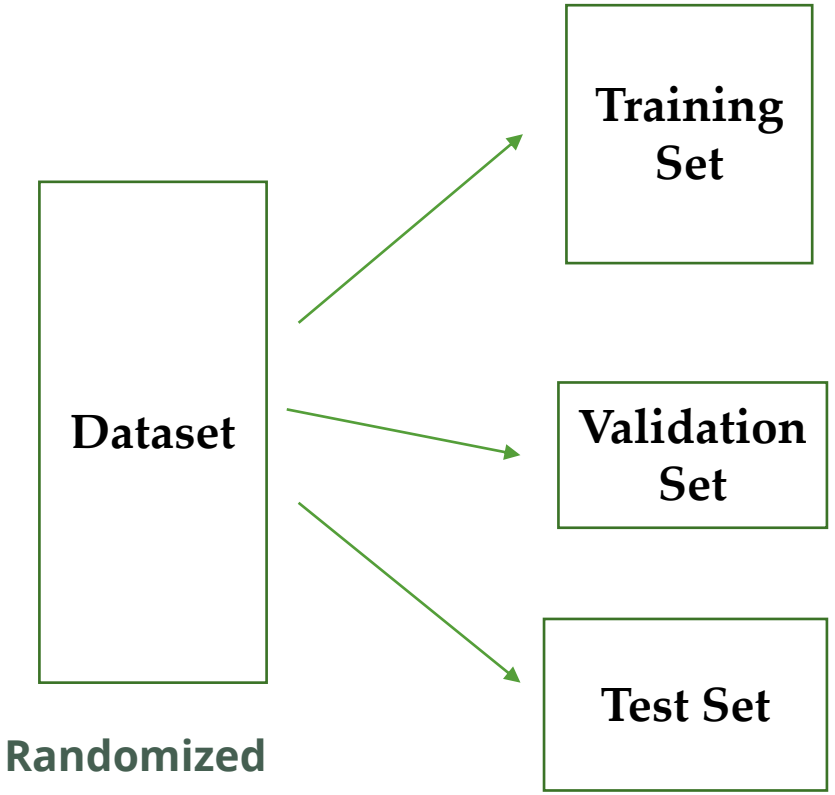


2.2 Parameters vs. Hyperparameters

In Machine Learning, there are two types of parameters:

- **Parameters** refers to the **weights** and **biases** of the model which are used for making predictions. These parameters can be learned directly from training data.
- **Hyperparameters** refers to higher-level structural **settings of learning algorithms** (i.e. learning rate, batch size, dropout ratio…). They are defined before training the model as they cannot be learned from training data. However, they can be tuned to find the most suitable value for the dataset.

2.3 Train-Test Split



- **Training Set**: a set of data used to train the model by learning its parameters such as weights and biases.
- **Validation Set (Optional)**: a set of data used to provide an unbiased evaluation of a model fit on the training set. It is normally used to tune the hyperparameters or to select the best model.
- **Testing/Dev Set**: a set of data used to measure and evaluate the performance (i.e., generalization and predictive power) of the trained model. It must not be used in model learning phase.

2.3.1 Train-Test Split - Drawback

- Even though Train-Test Split approach is easily to be implemented, it provides **high variance** estimate since changing which observations happens to be in training and testing dataset can significantly change testing accuracy. This is known as **Selection Bias**.
- **Selection bias** is the bias introduced by the selection of individuals, groups or data for analysis in such a way that proper randomization is not achieved, thereby the sample obtained is not representative of the population intended to be analyzed.

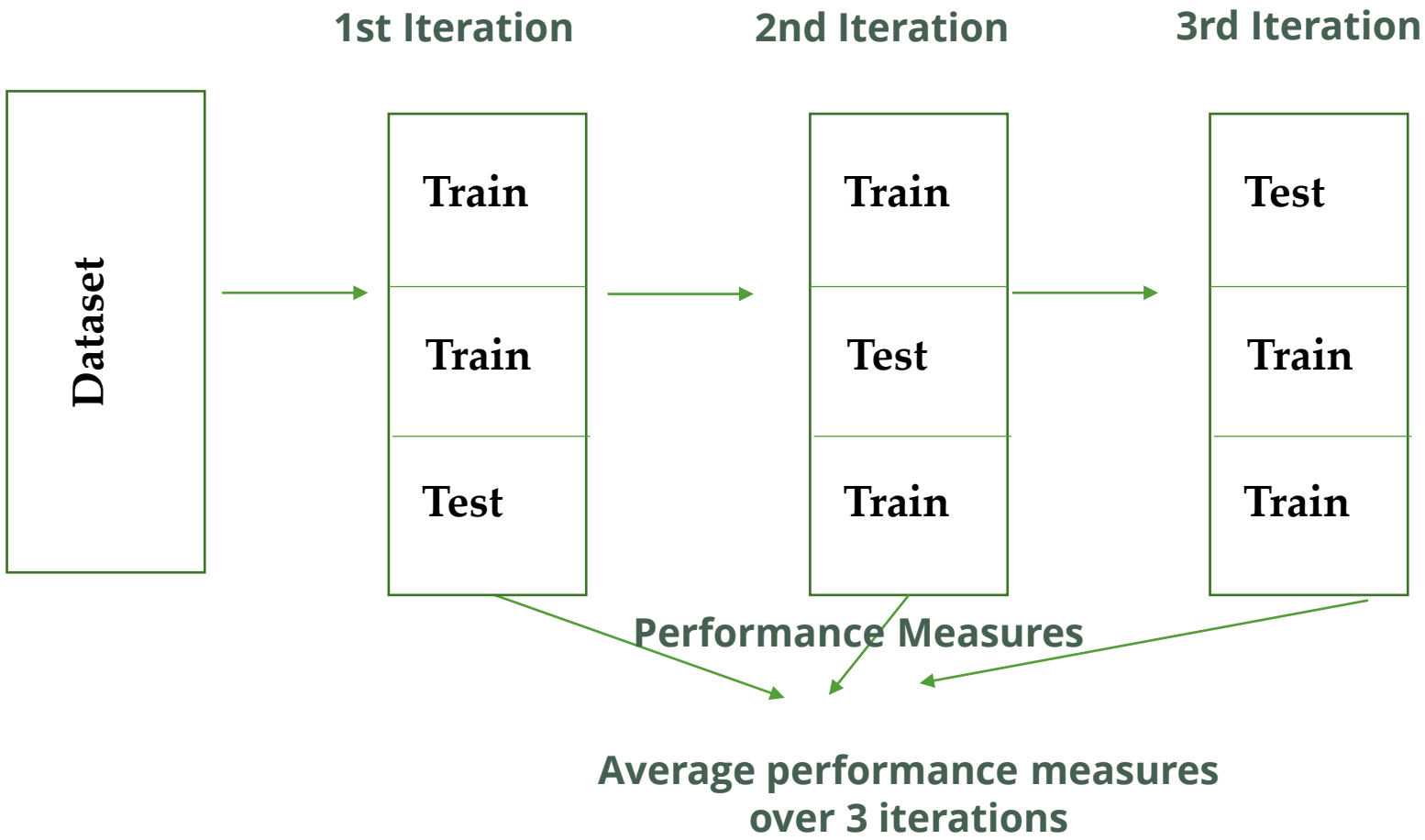
2.3.2 K-Fold Cross-Validation

- Why just choose one particular "split" of the data? – In principle, we should do this multiple times since the performance may be different for each split.

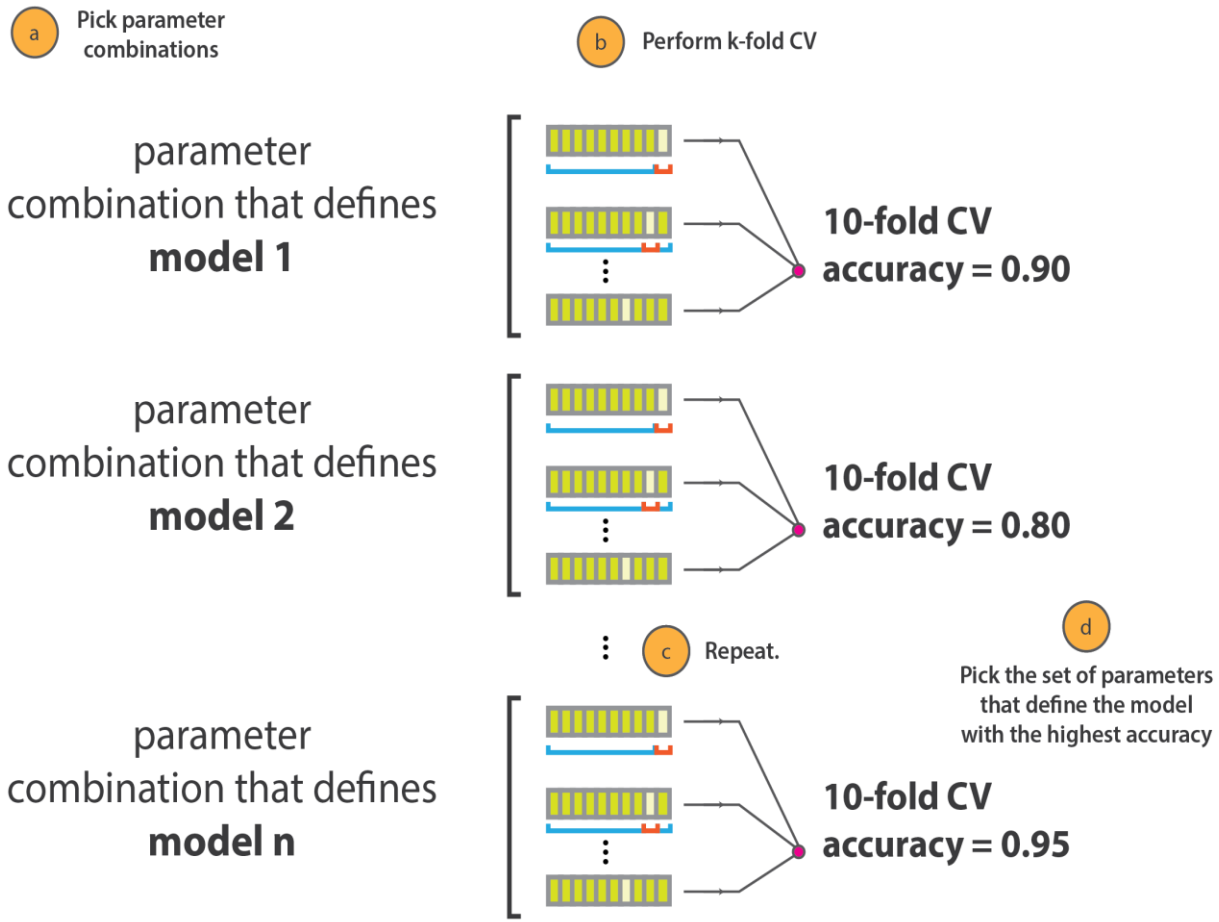
- **K-Fold Cross-Validation** (e.g., k=10) is done as follow:
  1. Randomly partition full dataset into k equal sized partitions.
  2. Select one partition as test data and remaining k-1 as training data.
  3. Train model and determine test performance with test data.
  4. Repeat the process k times, selecting a different partition each time for test data.
  5. Average test performance result to obtain the performance of model.

- Can also do "**leave-one-out CV**" where k = n

2.3.2 Example of 3-Fold CV



2.4 Tuning Hyperparameters

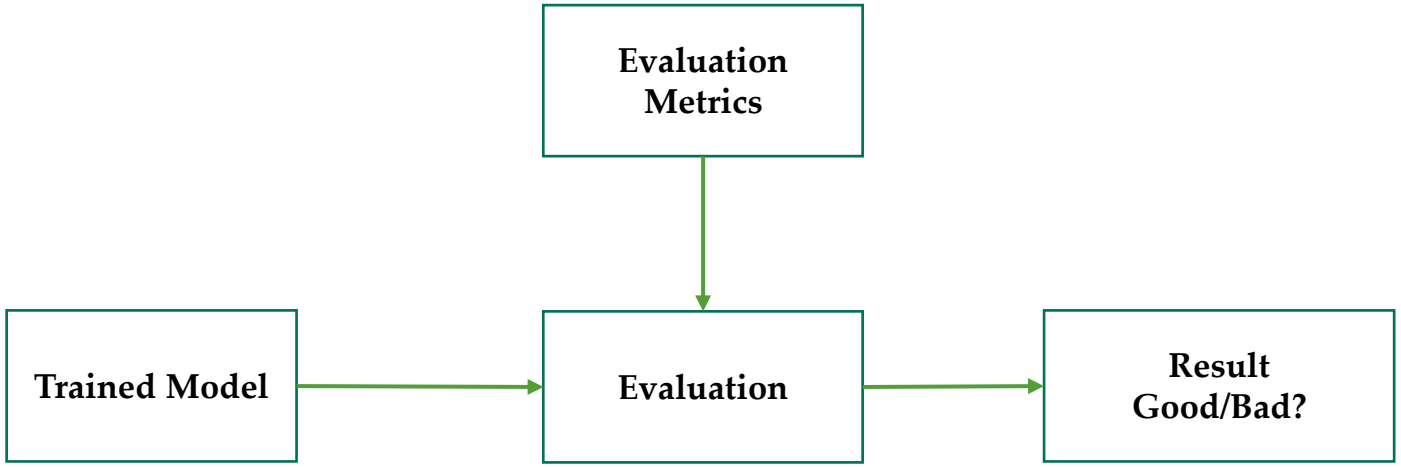


K-Fold Cross-Validation can be used to tune the hyperparameters, which is done as follows:

1. Pick a combination of parameters.
2. Perform K-Fold CV and compute the test performance.
3. Repeat step 1 and 2 with other parameter combinations.
4. Pick the set of parameters having the highest test performance.

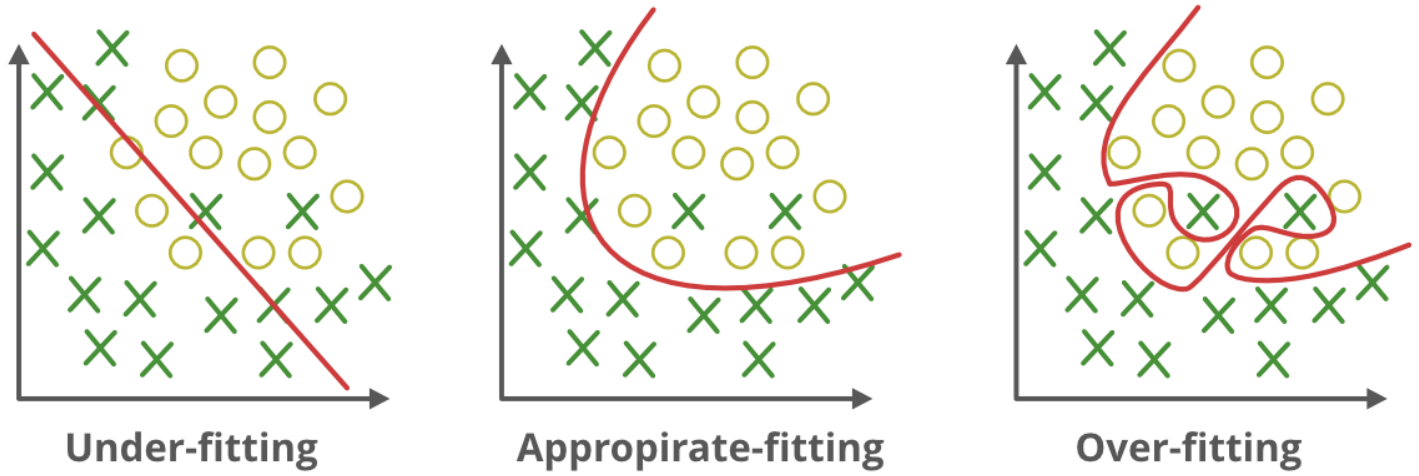
3. Evaluation

- When we obtained the model from the training step, we need to evaluate the accuracy of model with respect to the desired and expected requirement.
- We will present the **terminologies** which is used to describe the quality of the model and the **evaluation metrics** which are used as the **indicators** to measure the performance of the model.



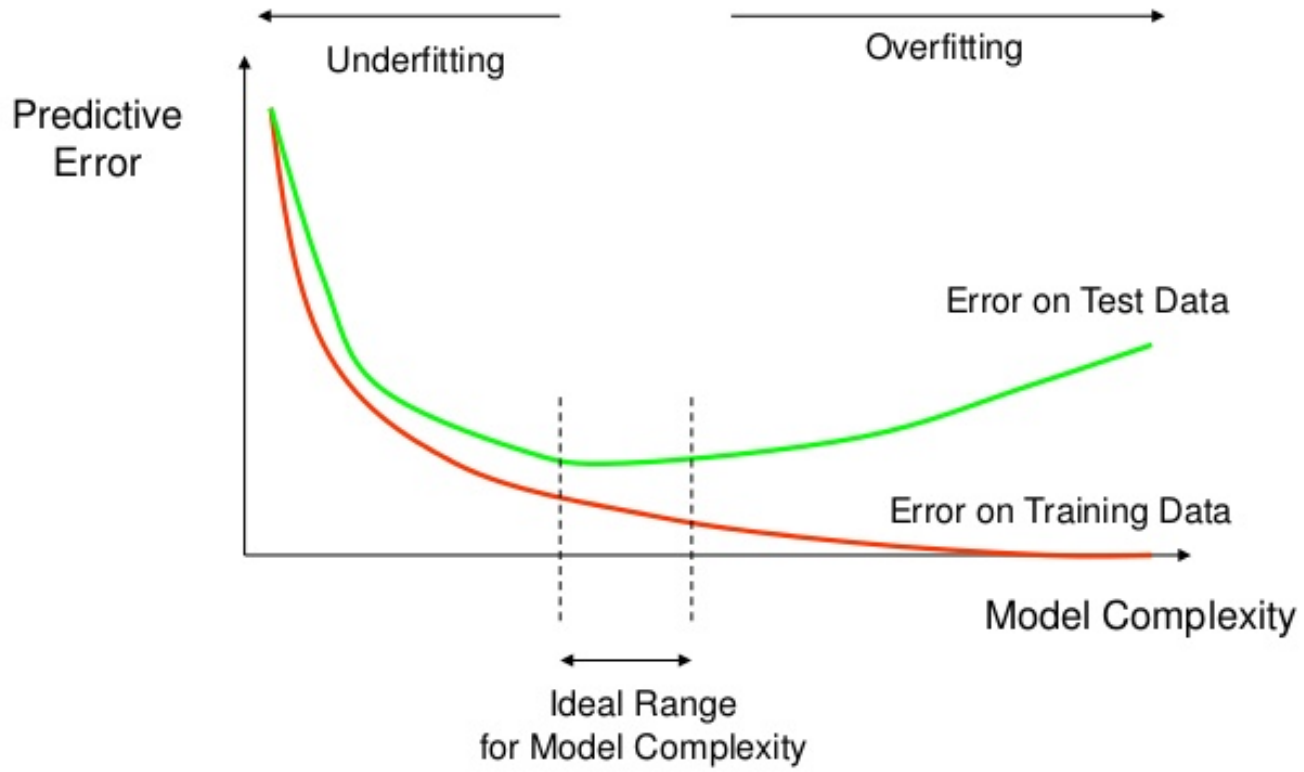
3.1 Underfitting vs. Overfitting

- **Underfitting:** When a model performs poorly on training data. This is because the model is unable to capture the relationship between the input examples and the target values.
- **Overfitting:** When a model performs well on training data but performs poor on testing data. This is because the model memorizes or focuses too much on the data it has seen during training, thus it is unable to generalize to unseen examples (testing data).



3.1.1 Underfitting vs. Overfitting - Tradeoff

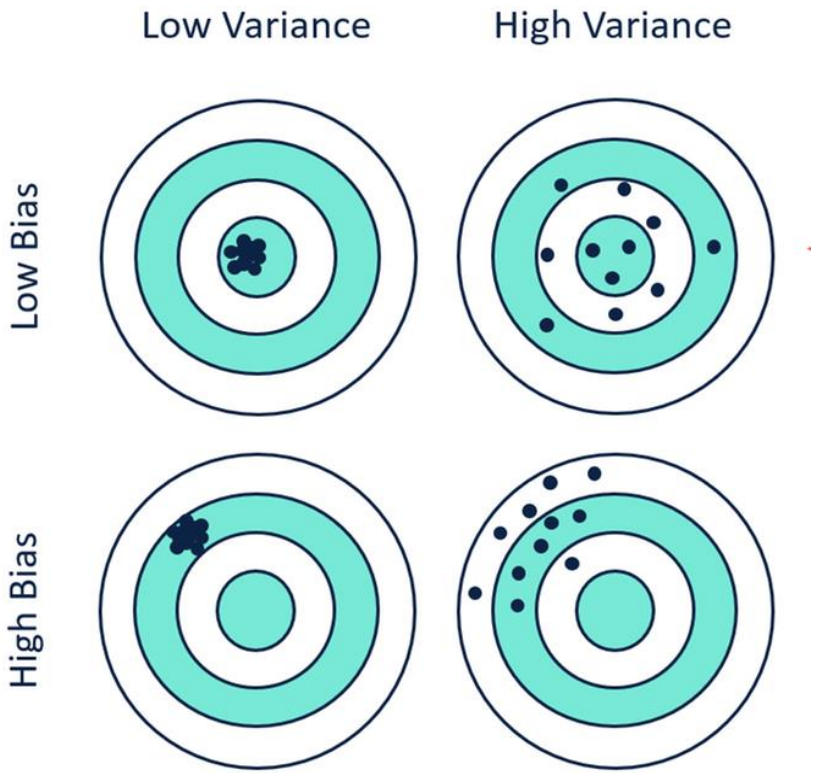
In order to achieve a optimal model, we need to take into account the compromise between Underfitting and Overfitting.



3.1.2 Preventing Underfitting and Overfitting

- **Underfitting**
  - Make sure there is enough training data so that the error is sufficiently minimized.
  - Increase the number of features to increase the model complexity.
- **Overfitting**
  - Decrease the number of features. As the number of features increase, the model complexity also increases, thus creating a higher chance of overfitting.
  - Apply **Regularization term** to the error/cost function to forcing the model to be simpler.
  - Early stop the training before the model start overfitting the training data.
  - Use **Ensemble Learning** to reduce chance of overfitting.

3.2 Bias vs. Variance

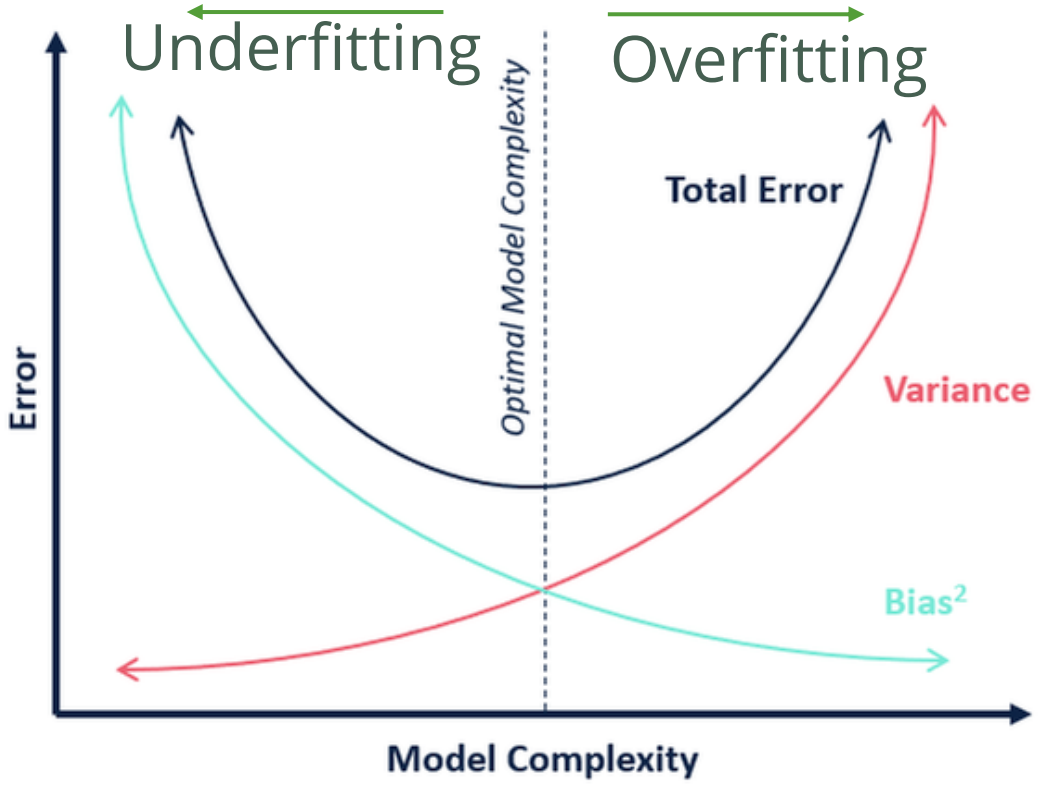


- **Bias** is the difference between the average predictions and the correct values. Model with high bias pays very little attention to the training data and oversimplifies the data, thus it leads to high error on training and testing data (**Underfitting**).

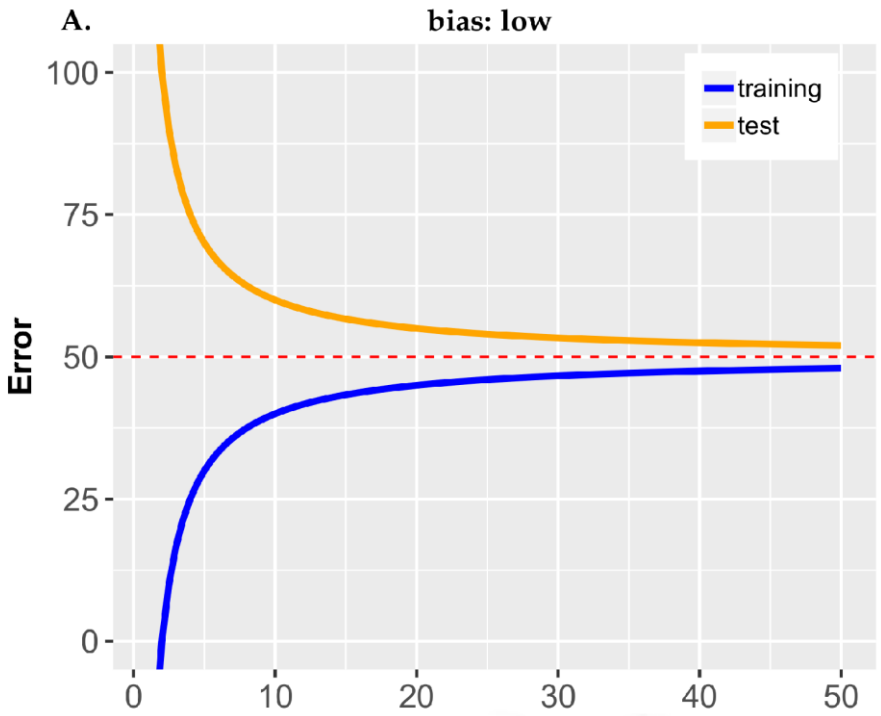
- **Variance** is the variability of model prediction on the data. Model with high variance pays much attention to training data and does not generalize well on the unseen data. Such model perform very well on training data but has high error rates on test data (**Overfitting**).

3.2.1 Bias vs. Variance - Tradeoff

In practice, we need to accept the trade-off between bias and variance since we cannot have both values low. We want to aim for something in the middle by which we can get the optimal model as possible.



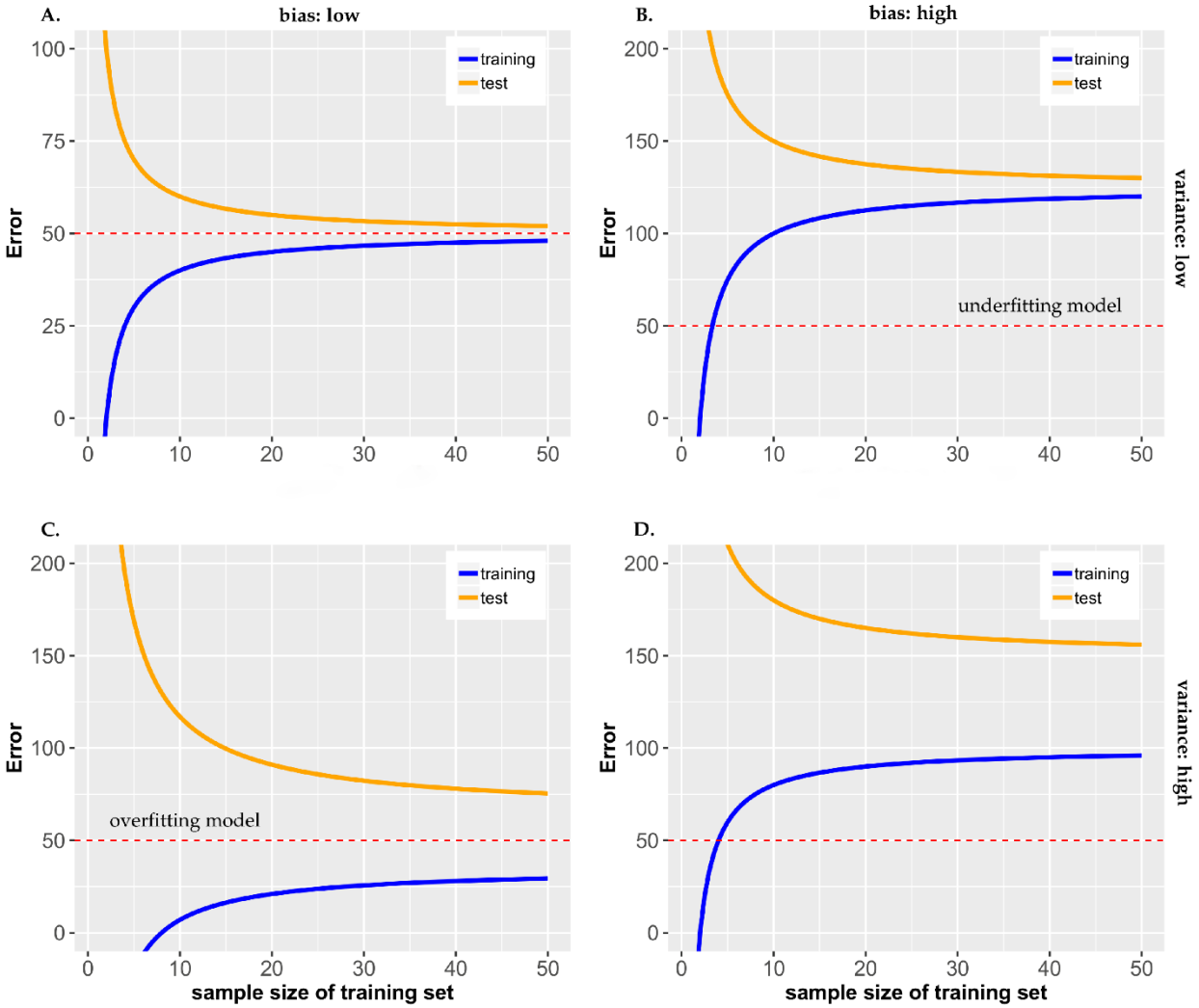
3.3 Learning Curve



Learning curve shows the testing and training error/score of a model for varying numbers of training samples. It helps to find out how much the model benefits from adding more training data and whether it suffers more from a variance error or a bias error.

If both testing and training error/score converge to a low value with increasing size of the training set, it will not benefit much from more training data.

3.3 Learning Curve



- This is an example of using learning curve to evaluate the model.
- The figure shows that the optimal model is on the top-left, which has the **low bias** and **low variance** as much as possible.
- It shows that the model trend to converge from sample size of 40.

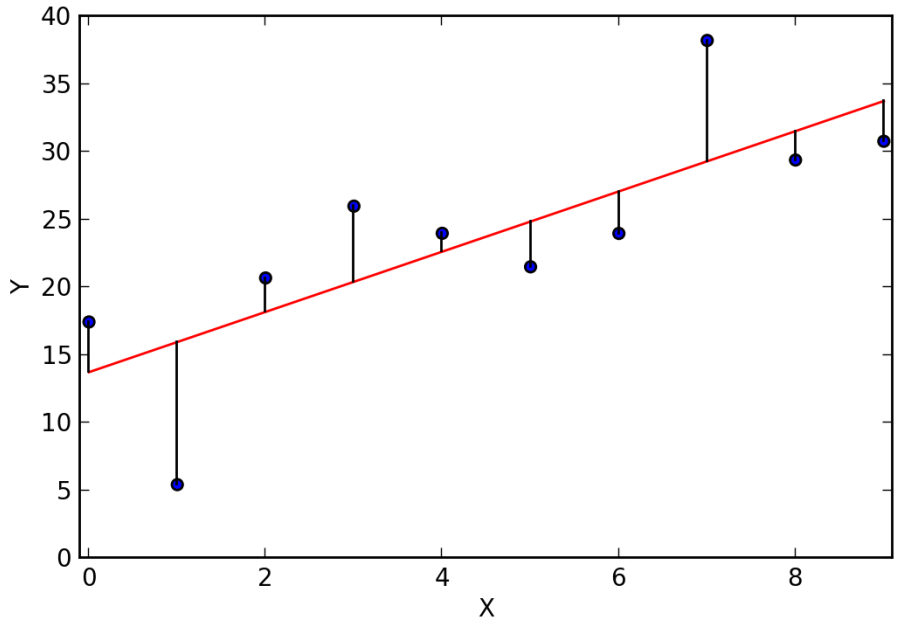
3.4 Evaluation Measures

- **Evaluation Measure/Metric** is an indicator explaining the performance of the model. The important aspect of evaluation metrics is their capability to discriminate among model results.
- The idea of building Machine Learning models works on a constructive feedback principle. We build a model, get feedback from metrics, make improvements and continue until we achieve a desirable accuracy/performance.
- The metrics are used based on the characteristic of the algorithm:
  - **Regression**: Mean Squared Error (MSE), Mean Absolute Error (MAE), R^2
  - **Classification**: Accuracy, Recall, Precision, F1-Score

3.4.1 Mean Squared Error

- **Mean Squared Error** (MSE) or **Mean Squared Deviation** (MSD) measures the average of error squares between predicted/estimated values $\hat{y}$ and reference/true values y.
- The smaller MSE, the better performance of the model in predicting values. Depending on your data, it may be impossible to get a very small value for MSE.

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



3.4.1 Mean Squared Error

Example: Find the mean squared error for the following set of values:

x	y	$\hat{y}$
43	41	43.6
44	45	44.4
45	49	45.2
46	47	46
47	44	46.8

3.4.1 Mean Squared Error

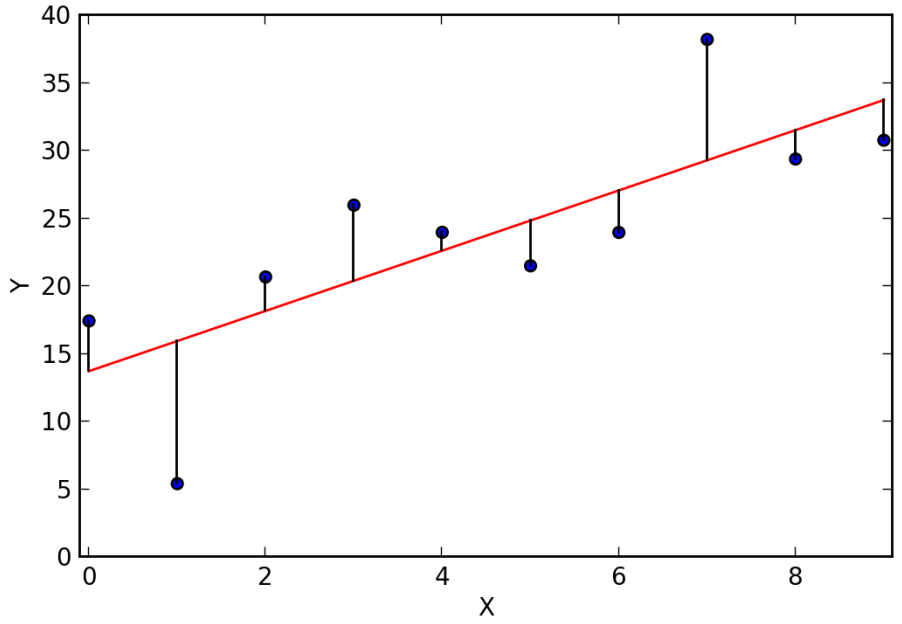
x	y	$\hat{y}$	$y - \hat{y}$	$(y - \hat{y})^2$
43	41	43.6	-2.6	6.76
44	45	44.4	0.6	0.36
45	49	45.2	3.8	14.44
46	47	46	1	1
47	44	46.8	-2.8	7.84

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{5}(6.76 + 0.36 + 14.44 + 1 + 7.84) = 6.08$$

3.4.2 Mean Absolute Error

- **Mean Absolute Error** (MAE) measures the average magnitude of the errors between predicted/estimated values ŷ and reference/true values y.
- The smaller MAE, the better performance of the model in predicting values.

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



3.4.2 Mean Absolute Error

Example: Find the mean absolute error for the following set of values:

x	y	$\hat{y}$
43	41	43.6
44	45	44.4
45	49	45.2
46	47	46
47	44	46.8

3.4.2 Mean Absolute Error

x	y	$\hat{y}$	$y - \hat{y}$	$\lvert y - \hat{y} \rvert$
43	41	43.6	-2.6	2.6
44	45	44.4	0.6	0.6
45	49	45.2	3.8	3.8
46	47	46	1	1
47	44	46.8	-2.8	2.8

$$MSE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| = \frac{1}{5}(2.6 + 0.6 + 3.8 + 1 + 2.8) = 2.16$$

3.4.3 $\mathbf{R^2}$

- $\boldsymbol{R^2}$ (**R-squared**) or **Coefficient of Determination** measures proportion of the variance in the dependent variable that is explained or predicted by the regression model.

$$R^2 = 1 - \frac{SS_r}{SS_t} = \frac{SS_t - SS_r}{SS_t} = \frac{\text{Explained Variation}}{\text{Total Variation}}$$

- $\boldsymbol{SS_t}$ is the total sum of squared errors when mean of observed values as the predicted value.

$$SS_t = \sum_{i=1}^{N} (y_i - \bar{y})^2$$

- $\boldsymbol{SS_r}$ is the total sum of squared errors between the reference values and the predicted values.

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

3.4.3 $\mathbf{R^2}$

- R-squared measures the performance proportion of the regression model based on the baseline model which always predicts $\bar{y}$.
- In most cases, R-squared is between **0** and **1** (100%).
  - **0** indicates that the dependent variable cannot be predicted from independent variables ($SS_t = SS_r, \hat{y} = \bar{y}$).
  - **1** indicates that the dependent variable is predicted without error from independent variables ($SS_r = 0, y = \hat{y}$). The two variables are perfectly correlated without variance.
- If R-squared has negative value, it means that the model has worse predictions than the baseline model.

3.4.3 $\mathbf{R^2}$

Example: Find the R^2 for the following set of values:

x	y	$\hat{y}$
43	41	43.6
44	45	44.4
45	49	45.2
46	47	46
47	44	46.8

3.4.3 $\mathbf{R^2}$

x	y	$\hat{y}$	$(y-\hat{y})^2$	$\bar{y}$	$(y-\bar{y})^2$
43	41	43.6	6.76	45.2	17.64
44	45	44.4	0.36	45.2	0.04
45	49	45.2	14.44	45.2	14.44
46	47	46	1	45.2	3.24
47	44	46.8	7.84	45.2	1.44
			$SS_r = 30.4$		$SS_t = 36.8$

$$R^2 = 1 - \frac{SS_r}{SS_t} = 1 - \frac{30.4}{36.8} = 0.1739$$

3.4.4 Confusion Matrix

Actual Class

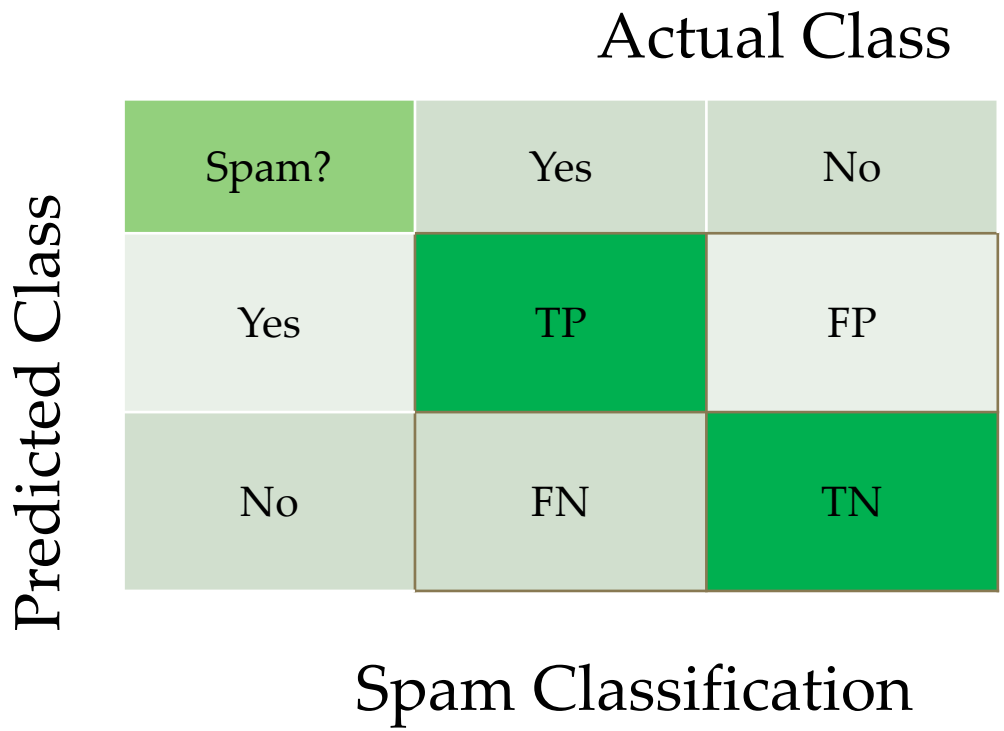
Predicted Class: Spam?	Yes	No
Yes	True Positive (TP)	False Positive (FP)
No	False Negative (FN)	True Negative (TN)

Spam Classification

- **Confusion matrix** is a contingency table which allows visualizing performance of a classifier. Each row of the table represents the instances in a predicted class while each column represents the instances in an actual class (or vice visa).
- **True positive** and **true negatives** are the observations that are correctly predicted and therefore shown in green.

3.4.4.1 Accuracy

- **Accuracy** is a measure of the fraction of times that the model predicted correctly (both true positive and true negative) out of total number of predictions.



$$Accuray = \frac{TP + TN}{TP + FP + FN + TN}$$

3.4.4.1 Accuracy - Can be Misleading

- Assume that your model trains itself to only identify **True Negative** and misses out **True Positive**. This happens when there is a class imbalance in training data.

Actual Class

Spam?	Yes	No
Yes	TP=0	FP=0
No	FN=80	TN=185

Predicted Class

Spam Classification

$$Accuray = \frac{185}{265} = 0.7$$

3.4.4.2 Precision

- **Precision** calculates the fraction of times the model predicts positive cases correctly from total number of **positive cases** it predicted.
- It attempts to answer the question of ”What proportion of positive identifications was actually correct?”

Actual Class

Spam?	Yes	No
Yes	TP	FP
No	FN	TN

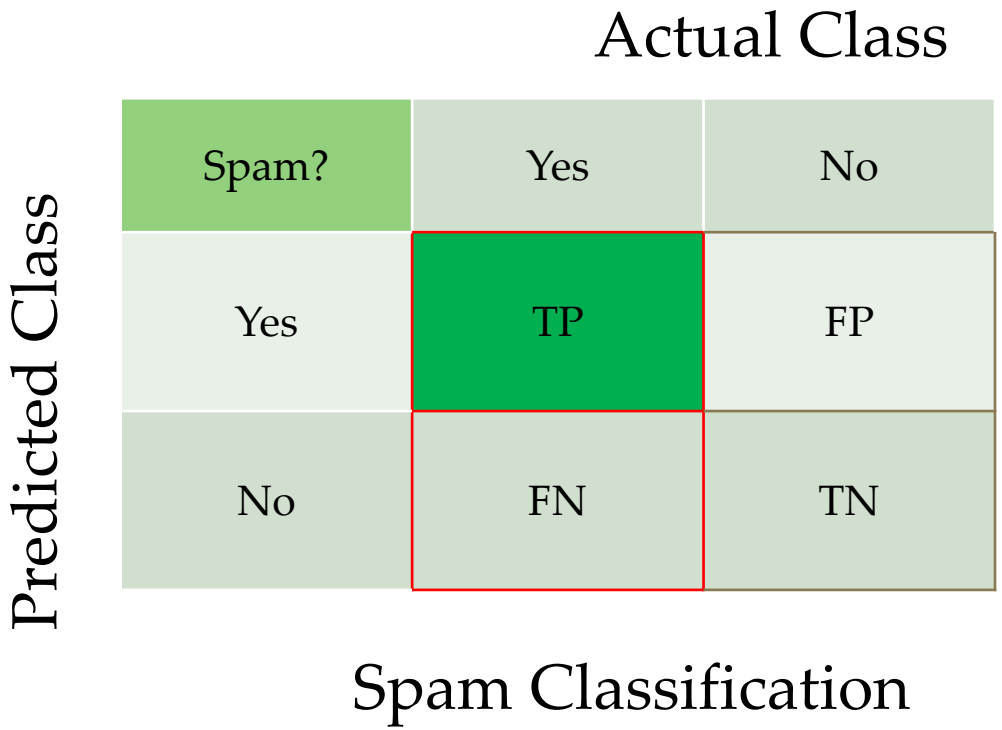
Predicted Class

Spam Classification

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

3.4.4.3 Recall

- **Recall** measures the fraction of times the model predicts positive cases correctly from total number of **actual positive cases**.
- It attempts to answer the question of ”What proportion of actual positives was identified correctly?”



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

3.4.4.4 F1-Score

- **F1-Score** is the **weighted average** between Precision and Recall to seek the balance between these two metrics.
- It is a better metric when there are imbalanced classes compared with **Accuracy**.

$$F1\text{-}Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall}$$

3.4.4.4 Example

Example: Calculate accuracy, precision, recall and F1-score for the following confusion matrix of spam email detection.

Spam?	Yes	No
Yes	TP=45	FN=20
No	FP=5	TN=30