

MACHINE LEARNING

CHAP 3 **Evaluation of Machine Learning models**

Jean Tabet
j.tabet@ul.edu.lb

➔ EVALUATION OF MACHINE LEARNING MODELS

Outline

- ➊ Introduction – ML Project Life Cycle
- ➋ Evaluation Metrics of a Classification algorithm
- ➌ Evaluation Metrics of a Regression algorithm
- ➍ Protocol - K-Fold Cross-Validation

➔ EVALUATION OF MACHINE LEARNING MODELS

Introduction

Business Understanding

- Understand the goals of the project and data mining

Data Understanding

- Initial data collection
- Data description
- Data Mining
- Data Quality Verification

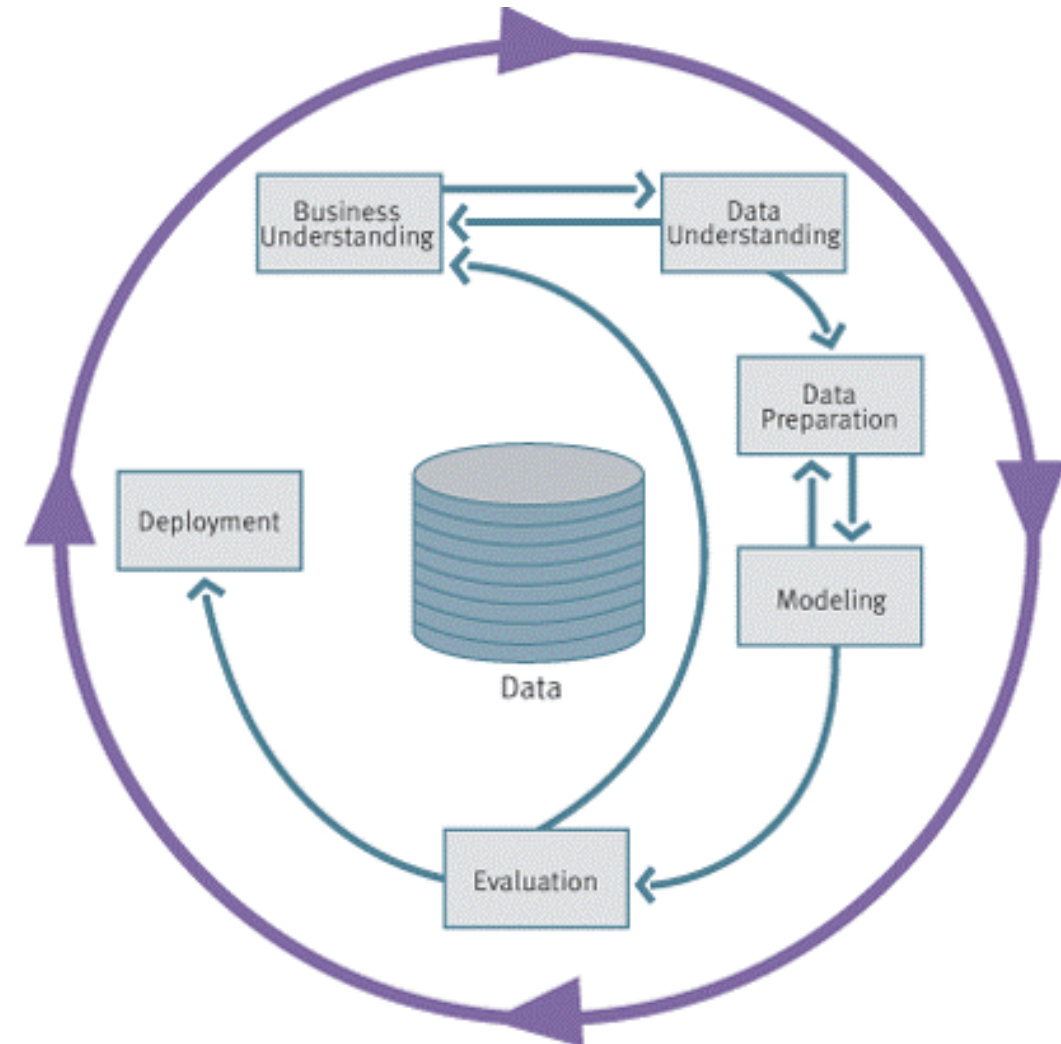
Data Preparation

- Data Selection, Data Cleaning, Feature engineering, Data Integration, Data Formatting

Modeling: Creating ML Models, Applying ML Models, Evaluating Models

Evaluation: Business goals respect

Deployment: Production



➔ EVALUATION OF CLASSIFICATION MODELS

Confusion Matrix

- A matrix containing as many rows as columns as classes,
- $C[i, j]$ = Number of observations of class i that are predicted in class j .

Binary Classification Case

		Predictions		
		1	0	Total
Actual	1	Number of 1 predicted correctly True Positives (TP)	Number of 1 predicted as 0 False Negatives (FN)	P
	0	Number of 0 predicted as 1 False Positives (FP)	Number of 0 predicted correctly True Negatives (TN)	N
	Total			

P : Number of positive observations **N** : Number of Negative Observations

```
sklearn.metrics.confusion_matrix(y_true, y_pred, *, labels=None, sample_weight=None, normalize=None)
```

➔ EVALUATION OF CLASSIFICATION MODELS

Example

We consider a classifier that classifies an email into two classes: Spam and not-spam. The classifier was tested on a set consisting of 100 Not-Spam emails and 100 Spam emails.

- Out of the 100 spam emails, the classifier predicted 95 as spam.
- Out of the 100 Not-Spam emails: the classifier predicted only 3 as spam.

Show the confusion matrix.

		Predicted Classes	
		Spam	Non-Spam
Real Classes	Spam	95	5
	Non-Spam	3	97



		Predictions		
Observations		+	-	Total
	+	TP	FN	P
	-	FP	TN	N
	Total			

➔ EVALUATION OF CLASSIFICATION MODELS

Sensitivity & precision

Recall (or **Sensitivity**) is the proportion of correct positive predictions among positive observations.

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

It is called the True Positive Rate (TPR).

		Predictions		
Observations		+	-	Total
	+	TP	FN	P
	-	FP	TN	N
	Total			

It's very easy to have a good recall by predicting that all examples are positive. This criterion cannot be used alone. It is often added:

Precision is the proportion of correct positive predictions among positive predictions.

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

➔ EVALUATION OF CLASSIFICATION MODELS

F1-score

F1-score is the harmonic average of precision and recall.

$$F1 = 2 * \frac{PRECISION * RECALL}{PRECISION + RECALL}$$

Example

Assuming that: Precision = 77% and Recall = 80%. When your classifier declares an image to be a 5, it's only right 77% of the time. In addition, it only detects 80% of the 5 !!!

$$F1 = 2 * \frac{PRECISION * RECALL}{PRECISION + RECALL} = 1 = 2 * \frac{77 * 80}{77 + 80} = 78.5$$

➔ EVALUATION OF CLASSIFICATION MODELS

Error Rate (ERR) and Accuracy (ACC)

ERR : Number of all incorrect predictions / size of dataset

$$\text{ERR} = \frac{FN + FP}{P + N}$$

ACC: Number of all correct predictions / size of dataset

$$\text{ACC} = \frac{TP + TN}{P + N}$$

$$\text{ERR} : 1 - \text{ACC}$$

		Predictions		
		+	-	Total
Observations	+	TP	FN	P
	-	FP	TN	N
	Total			

➔ EVALUATION OF CLASSIFICATION MODELS

True Negative Rate (TNR) and True Positive Rate (TPR)

Specificity: True Negative Rate (TNR)

$$SP = \frac{TN}{TN+FP}$$

False Positive Rate:

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

$$FPR : 1 - SP$$

		Predictions			
Observations		+	-	Total	
	+	TP	FN	P	Recall = TP/P
	-	FP	TN	N	SP = TN/N
	Total	TP+FP	FN+TN	P+N	
		Precision			ACC = (TP+TN)/(P+N)

➔ EVALUATION OF CLASSIFICATION MODELS

Example

Let's take the example of a clinical trial to illustrate these different criteria. This is not a machine learning model, but a cervical cancer screening smear. The results of an experiment conducted on 4,000 women aged 40 years and older are presented in the following confusion matrix.

- Recall?
- Specificity?
- Accuracy?
- Is it a good screening tool?

	Smear +	Smear -	Total
Cancer	190	10	200
No cancer	210	3590	3800
Total	400	3600	4000

The screening smear is similar to an ML algo.

➔ EVALUATION OF CLASSIFICATION MODELS

Example

Observations	Predictions			
	+	-	Total	
	+	TP	FN	P
	-	FP	TN	N
	Total	TP+FP	FN+TN	P+N
	Precision			ACC = (TP+TN)/(P+N)

	Smear +	Smear -	Total
Cancer	190	10	200
No cancer	210	3590	3800
Total	400	3600	4000

$$\text{Recall} = 190/200 = 95 \%$$

$$\text{Precision} = 190/400 = 47.5 \%$$

$$\text{Specificity} = 3590/3800 = 94.5 \%$$

Yes, this is a **good screening tool** because:

- Very high sensitivity (95%)
- High specificity (~94.5%)
- High overall accuracy
- Very few missed cancer cases

It behaves similarly to a good ML classifier:
high recall with acceptable false positives.

➔ EVALUATION OF CLASSIFICATION MODELS

Example

For screening tests, the most important metric is Recall (Sensitivity).

We want:

Very high recall (few false negatives)

Missing cancer cases is dangerous.

Recall = 95% → Good

Only 10 false negatives out of 200 cancer cases.

False positives = 210

In screening, false positives are acceptable because they can be followed by confirmatory tests.

		Predictions		
Observatios		Smear +	Smear -	Total
	Cancer	190	10	200
	No cancer	210	3590	3800
	Total	400	3600	4000

➔ EVALUATION OF CLASSIFICATION MODELS

ROC curve

- A graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied.
- Effective method of evaluating the performance of a model in how well it distinguishes the positive class.
- The idea of the ROC curve is to vary the "threshold" from 1 to 0 and, for each case, calculate the Recall (TPR) and the FPR (1-specificity) that are plotted in a graph: on the x-axis the FPR, on the y-axis the Recall.

➔ EVALUATION OF CLASSIFICATION MODELS

ROC curve

The results of a binary classification experiment are given, evaluated on 6 samples



Label	+	-	+	+	-	-
Score	0.99	0.95	0.51	0.45	0.10	0.01

Assignment Rule

If $\text{Score}(X_i) \geq \text{threshold} \rightarrow X_i$ is predicted as (+)

Otherwise X_i is predicted as (-)

Plot

We vary the "threshold" from 1 to 0 and, for each case, calculate the Recall (TPR) and the FPR (1-specificity) and plot them: on the x-axis the FPR, on the y-axis the Recall.

➔ EVALUATION OF CLASSIFICATION MODELS

ROC curve

Label	+	-	+	+	-	-
Score	0.99	0.95	0.51	0.45	0.10	0.01

	+	-	Total
+	0	3	3
-	0	3	3
Total			

Threshold1 = 1

TPR = 0

FPR = 0

	+	-	Total
+	1	2	3
-	0	3	3
Total			

Threshold2 = 0,99

TPR = 1/3

FPR = 0

	+	-	Total
+	1	2	3
-	1	2	3
Total			

Threshold3 = 0,95

TPR = 1/3

FPR = 1/3

	+	-	Total
+	2	1	3
-	1	2	3
Total			

Threshold4 = 0,51

TPR = 2/3

FPR = 1/3

➔ EVALUATION OF CLASSIFICATION MODELS

ROC curve

Label	+	-	+	+	-	-
Score	0.99	0.95	0.51	0.45	0.10	0.01

	+	-	Total
+	3	0	3
-	2	1	3
Total			

Threshold6 = 0,1

TPR = 1

FPR = 2/3

	+	-	Total
+	3	0	3
-	1	2	3
Total			

Threshold5 = 0,45

TPR = 1

FPR = 1/3

	+	-	Total
+	3	0	3
-	3	0	3
Total			

Threshold7 = 0,01

TPR = 1

FPR = 1

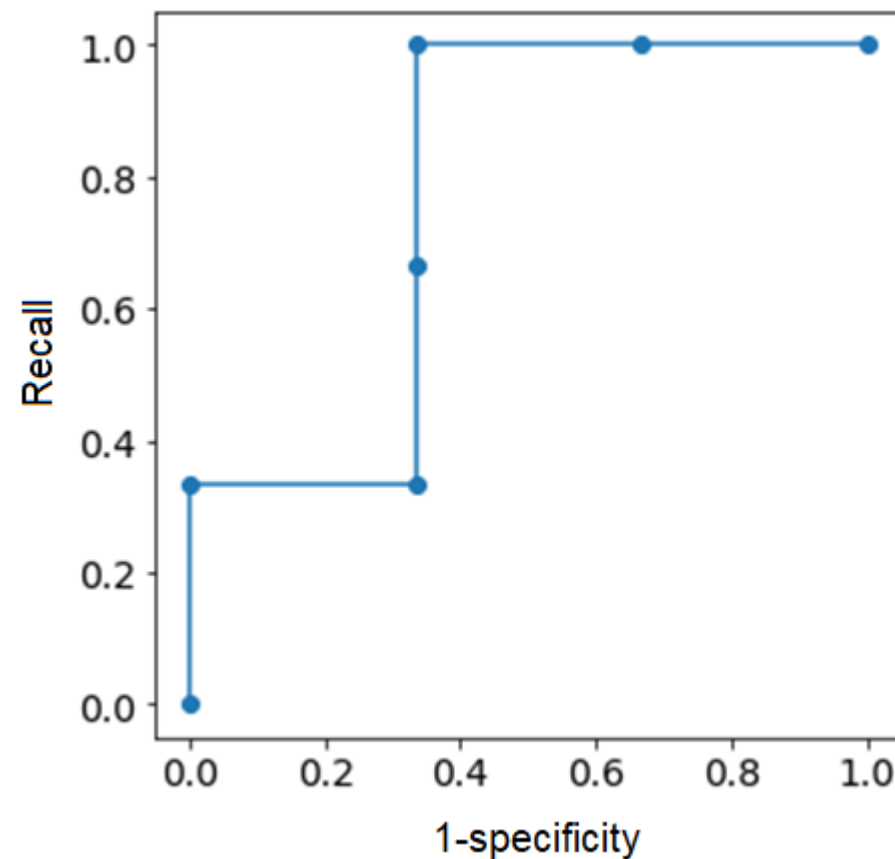
➔ EVALUATION OF CLASSIFICATION MODELS

ROC curve

Label	+	-	+	+	-	-
Score	0.99	0.95	0.51	0.45	0.10	0.01

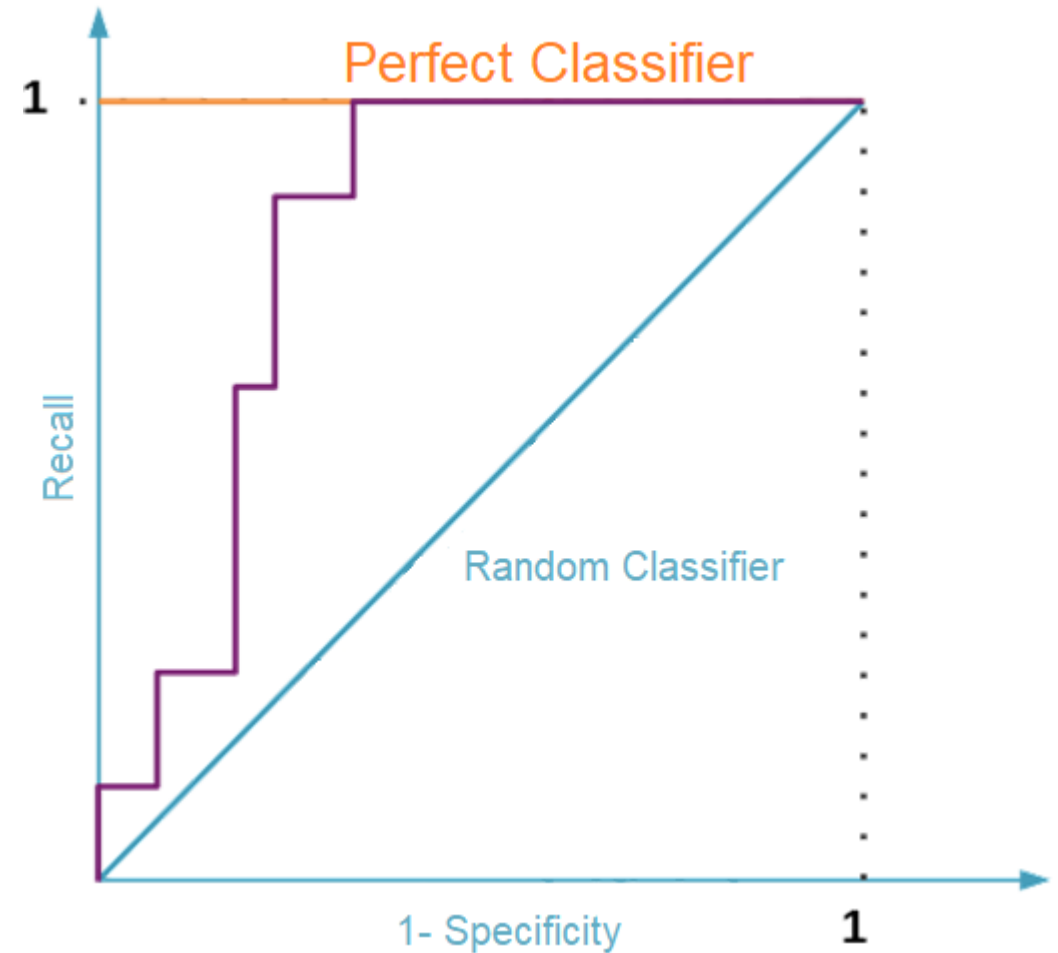
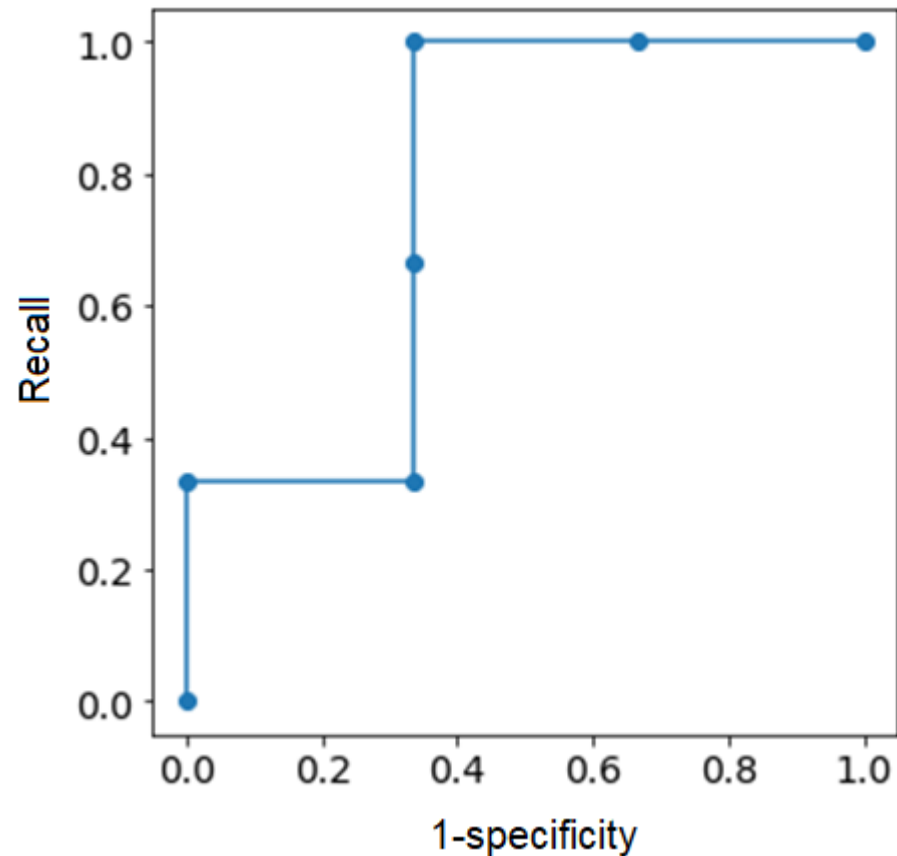
Rank scores in descending order

TP/P	0	1/3	1/3	2/3	1	1	1
FP/P	0	0	1/3	1/3	1/3	2/3	1



➔ EVALUATION OF CLASSIFICATION MODELS

ROC curve



➔ EVALUATION OF CLASSIFICATION MODELS

AUC curve

A global comparison tool is **Area Under Curve (AUC)**

It represents the area under the ROC curve.

The higher its value, the more interesting (performing) the model is between 0.0 and 1.0.

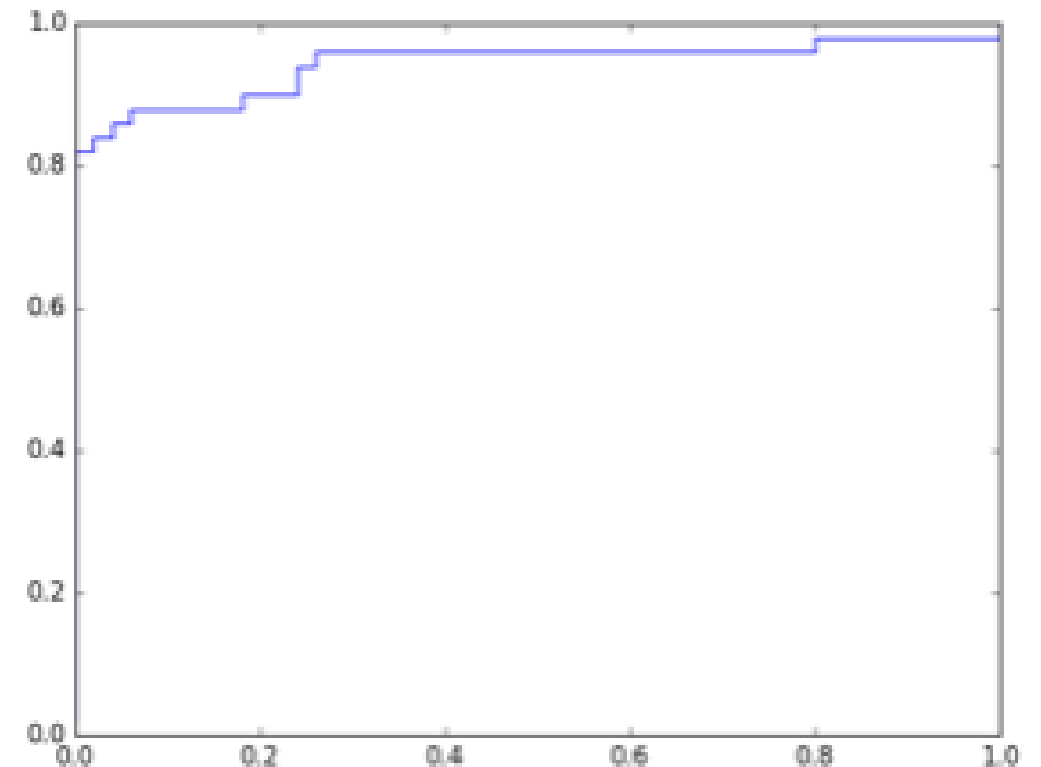
1.0 = perfect model

0.5 = random guessing

< 0.5 = worse than random

AUC summarizes the ROC curve into a single metric:

Probability that the model ranks a random positive higher than a random negative.



➔ EVALUATION OF CLASSIFICATION MODELS

Python

```
y_scores = model.predict_proba(X_test)[: , 1]  # probability for class=1
```

```
from sklearn.metrics import roc_curve
```

```
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
```

```
optimal_idx = np.argmax(tpr - fpr)
```

```
optimal_threshold = thresholds[optimal_idx]
```

```
import matplotlib.pyplot as plt
```

```
plt.plot(fpr, tpr, label="ROC curve")
```

```
plt.plot([0,1], [0,1], linestyle="--", color="gray", label="Random")
```

```
plt.xlabel("False Positive Rate")
```

```
plt.ylabel("True Positive Rate")
```

```
plt.title("ROC Curve")
```

```
plt.legend()
```

```
plt.show()
```

```
from sklearn.metrics import roc_auc_score
```

```
auc = roc_auc_score(y_test, y_scores)
```

```
print("AUC:", auc)
```

➔ EVALUATION THE PERFORMANCE OF ML MODEL

Generalization

- Generalization is the ability of a model to make correct predictions on new data, which was not used to build it.
- A good model avoids overfitting or underfitting.

Underfitting

- Too simple model
- It adjusts very little to the training data
- Poor performance on the training set
- Poor performance on new data.

Overfitting

- A very complex model
- It adjusts too much to the training data
- Good performance on the training set
- Poor performance on new data.

➔ EVALUATION THE PERFORMANCE OF ML MODEL

Train-Validation-Test Splitting

From the full dataset, we divide data into:

1. Training Set

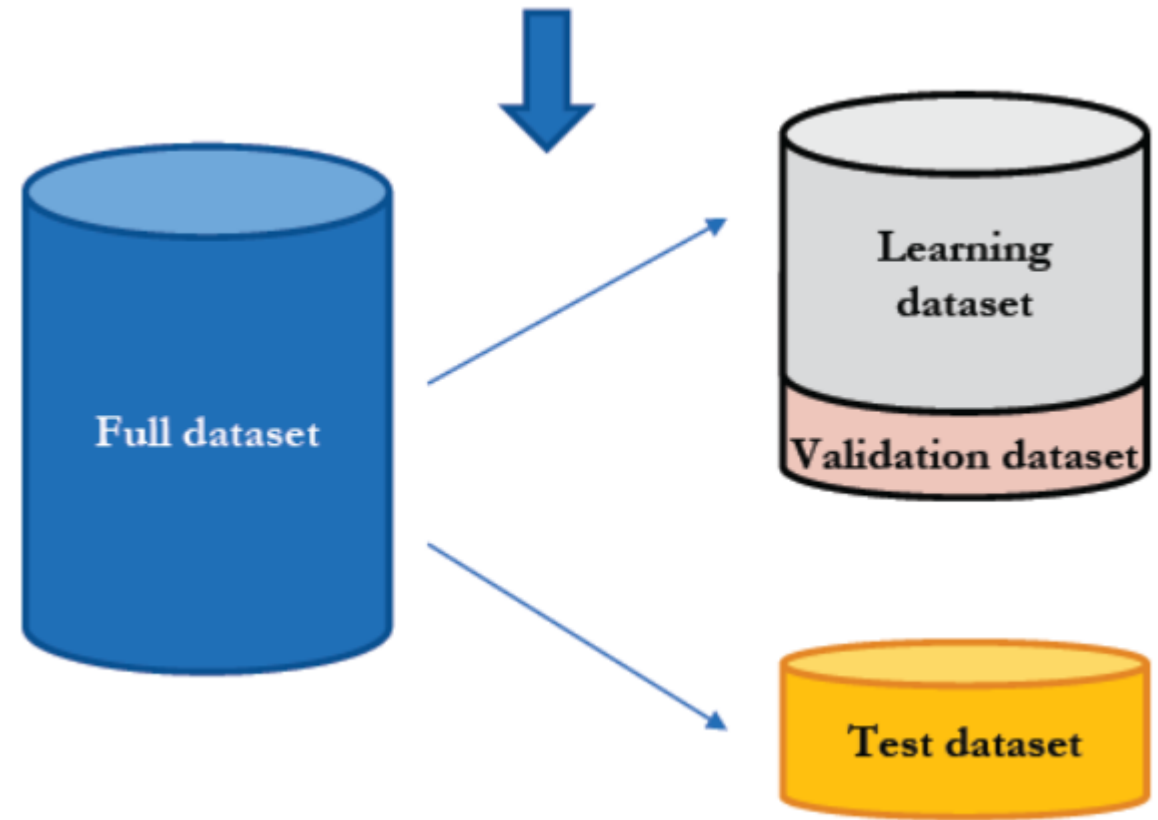
- Used to fit the model parameters
- The model learns patterns from this data

2. Validation Set

- Used to tune hyperparameters
- Helps select: regularization and model complexity
- We evaluate performance many times here

3. Test Set

- Used only once
- Measures the true generalization error
- Simulates new unseen data



➔ EVALUATION THE PERFORMANCE OF ML MODEL

Generalization

Don't touch the test set during tuning. Using the test set to pick hyperparameters leaks information and gives an overly optimistic score.

How to split: First, split the dataset into Training and Test once.

- If data is plentiful, you can also carve out a separate Validation set from the training data.
- If data is not very large, don't waste a fixed validation chunk—run K-fold cross-validation on the training set instead.

Workflow:

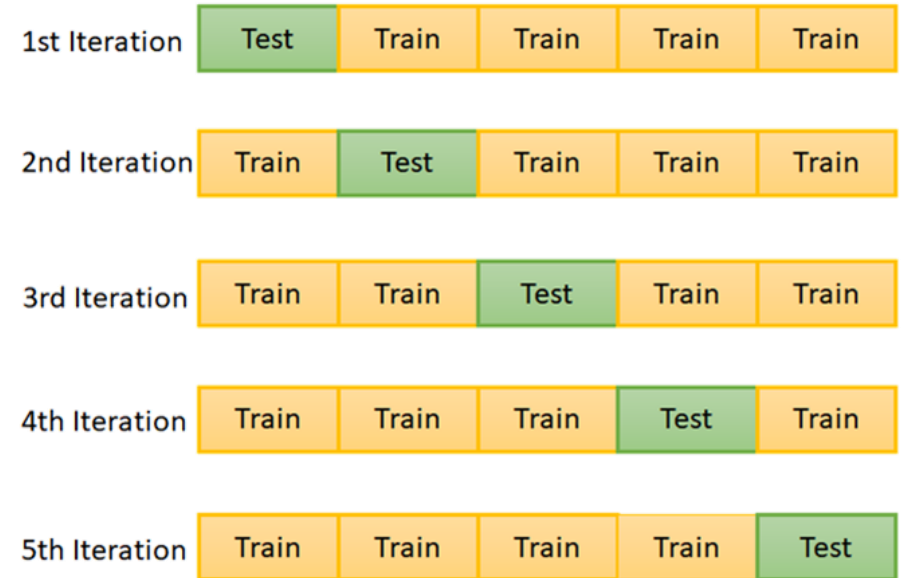
- On the training set, do K-fold CV to compare models/hyperparameters and pick the best.
- Retrain that choice on the entire training set.
- Evaluate once on the untouched test set to report final performance.

This maximizes training data, gives a stable estimate during tuning, and keeps the test score an unbiased measure of generalization.

➔ EVALUATION THE PERFORMANCE OF ML MODEL

K-folds cross-validation

- Use the entire dataset for training and validation.
- Divide the dataset into k folds that are approximately equal. At each iteration, each of the k subset is used as a test set. The rest (the union of the other k-1 parts) is used for training.
- Evaluate the predictions made on the data set.
- Average performance obtained on k folds.
- Why it's better than a single train/validation split
Lower variance estimate (every sample is used for validation once and for training K-1 times).
- Data-efficient—no points are wasted.
- Choosing K: 5 or 10.



➔ EVALUATION THE PERFORMANCE OF ML MODEL

K-folds cross-validation

- k is not a parameter but a hyperparameter, which means that unlike conventional parameters, it cannot be learned automatically by the algorithm from the training data.
- Hyperparameters are used to characterise the model (e.g. complexity, speed of convergence, etc). These parameters (in this case the number of neighbours k) are not found using the training data, but rather we optimise them using the validation dataset.
- In practice, the recommended optimum value for K is the square root of N , where N is the total number of samples.
- It is advisable to consider odd values of k : to avoid the indeterminate case.

➔ EVALUATION OF REGRESSION MODELS

Linear regression

Explanatory variables,
Characteristics
Attributes

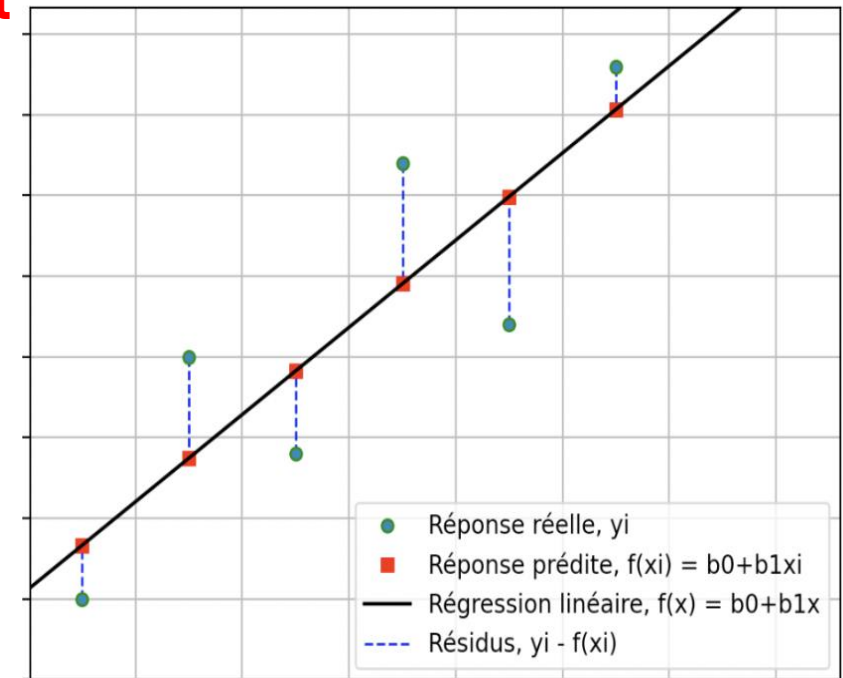


X^1	...	X^j	...	X^n	Y
X_{11}	1	X_{1j}		X_{1n}	y_1
...	...			...	...
X_{i1}		X_{ij}		X_{in}	y_i
...					
X_{m1}	1	X_{mj}		X_{mn}	y_m

Variable to predict
Target Variable
Dependent Variable
(Continuous)

Regression

Dependent
variable



Independent variable

Input - Dataset

Output of ML

➔ EVALUATION OF REGRESSION MODELS

Linear regression: Squared Error

i : Observation index; n : Number of Observations

x_i : Input Parameter: Input Variable

y_i : Actual Dependent Response/Variable Observed

$\hat{y}_i$: Estimated (predicted) response by linear regression

1. Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n [y_i - \hat{y}_i]^2$$

2. Root Mean Square Error (RMSE)

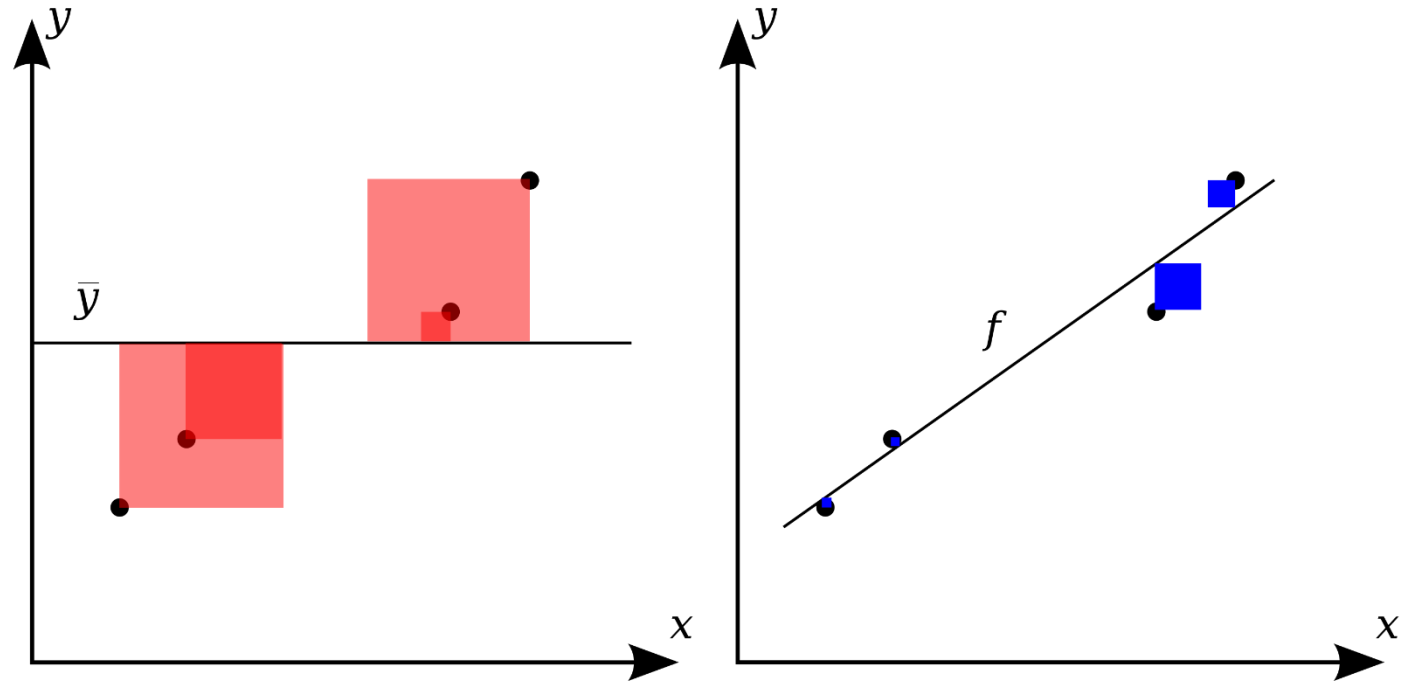
$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n [y_i - \hat{y}_i]^2}$$

➔ EVALUATION OF REGRESSION MODELS

Linear regression: Determination coefficient R^2

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

The more the linear regression (right) fits the data compared to the simple mean (on the left graph), the closer the value of R^2 is to 1.



- The areas of the blue squares represent the square of the residuals relative to the linear regression.
- The areas of the red squares represent the square of the residuals relative to the mean value.

➔ EVALUATION OF REGRESSION MODELS

Linear regression: Adjusted determination coefficient ($\bar{R}^2$)

Adjusted R^2 tells you how much of the variance in y is explained by your model after penalizing for the number of predictors. Unlike ordinary R^2 , it doesn't automatically increase when you add variables that don't help.

It takes into account the number of variables. Indeed, the main disadvantage of R^2 is that it grows with the number of explanatory variables.

$$\bar{R}^2 = 1 - (1 - R^2) \frac{n-1}{n-p}$$

With p the number of variables