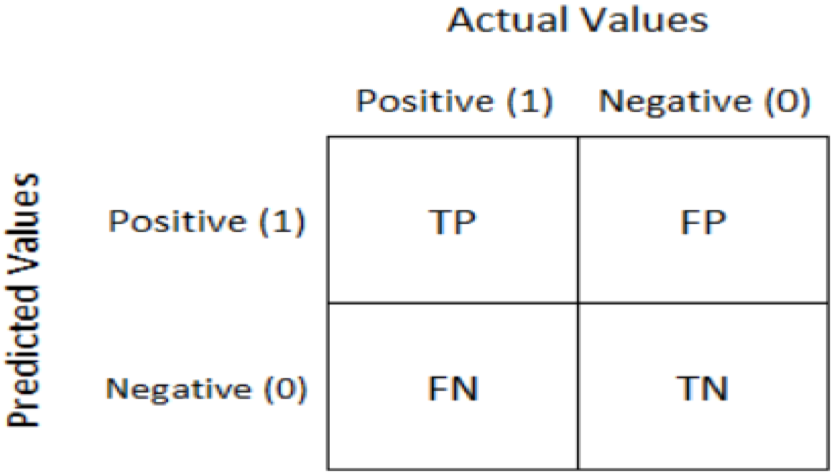
BINARY CLASSIFICATION AND PERFORMANCE MEASURES

PERFORMANCE MEASURES

CONFUSION MATRIX

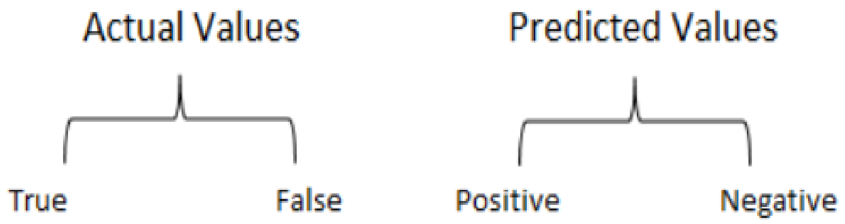
- A confusion matrix is a performance measurement technique for Machine learning classification.
- It is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes.
- For a binary classification problem, it is a two-by-two table that contains four outcomes produced by a binary classifier.



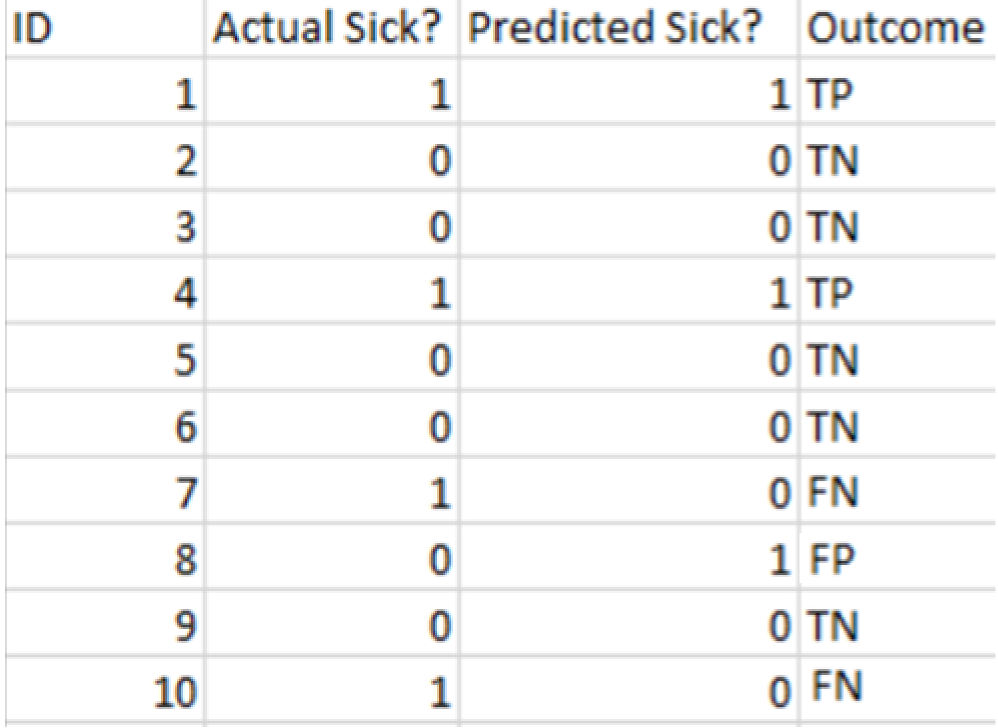
		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

CONFUSION MATRIX

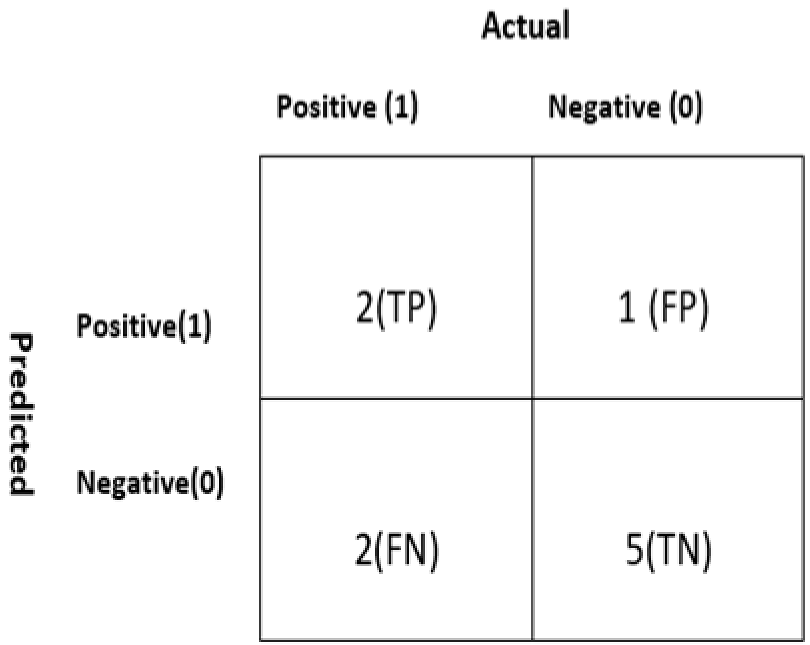
- **True Positive(TP):**
  - *Interpretation*: You predicted positive and it's true.
  - You had predicted that France would win the world cup, and it won.
- **True Negative(TN):**
  - *Interpretation:* You predicted negative and it's true.
  - You had predicted that England would not win, and it lost.
- **False Positive(FP): (Type 1 Error)**
  - *Interpretation*: You predicted positive and it's false.
  - You had predicted that England would win, but it lost.
- **False Negative(FN): (Type 2 Error)**
  - *Interpretation*: You predicted negative and it's false.
  - You had predicted that France would not win, but it won.
- Just Remember, We describe predicted values as Positive and Negative and actual values as True and False.



Example 1



ID	Actual Sick?	Predicted Sick?	Outcome
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	1	0	FN
8	0	1	FP
9	0	0	TN
10	1	0	FN



		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	2(TP)	1 (FP)	3
	Negative(0)	2(FN)	5(TN)	7
	Total	4	6	10

Example 2

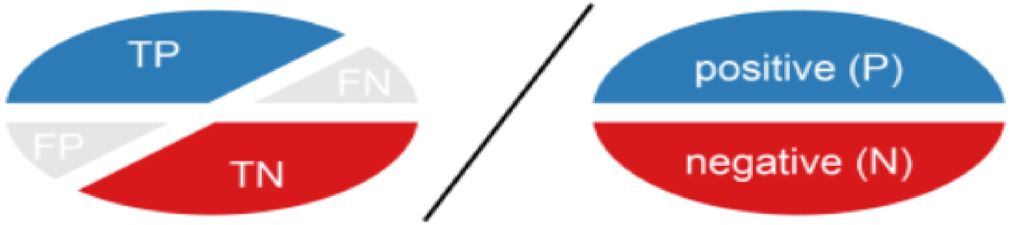
y	y pred	output for threshold 0.6	
0	0.5	0	→ TN
1	0.9	1	→ TP
0	0.7	1	→ FP
1	0.7	1	→ TP
1	0.3	0	→ FN
0	0.4	0	→ TN
1	0.5	0	→ FN

		Actual	
		Positive (1)	Negative (0)
Predicted	Positive(1)	2(TP)	1(FP)
	Negative(0)	3(FN)	2(TN)

ACCURACY

- Accuracy (ACC) is calculated as the number of all correct predictions divided by the total number of the dataset.
- The best accuracy is 1.0, whereas the worst is 0.0.
- Overall, how often is the classifier correct?

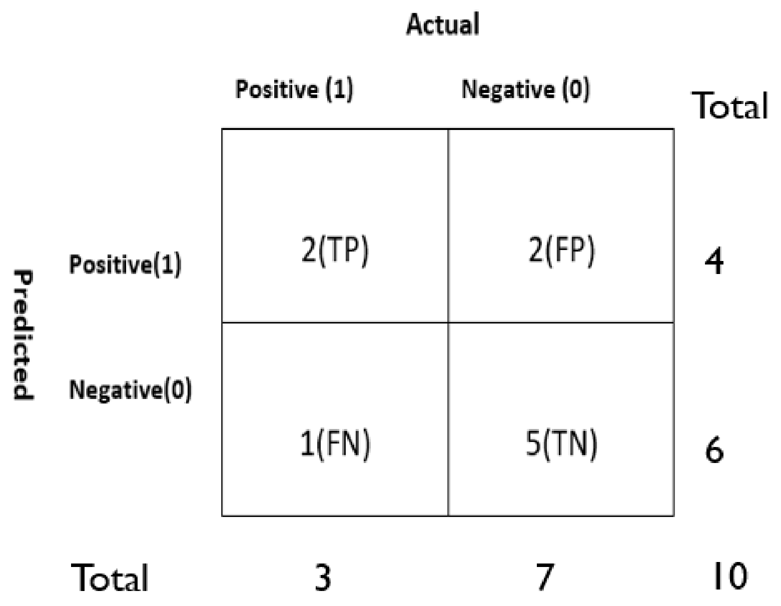
Accuracy: (TP + TN) / (P + N)



Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

ACCURACY(CONT)

- The accuracy would be calculated by the following formula



		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	2(TP)	2(FP)	4
	Negative(0)	1(FN)	5(TN)	6
Total		3	7	10

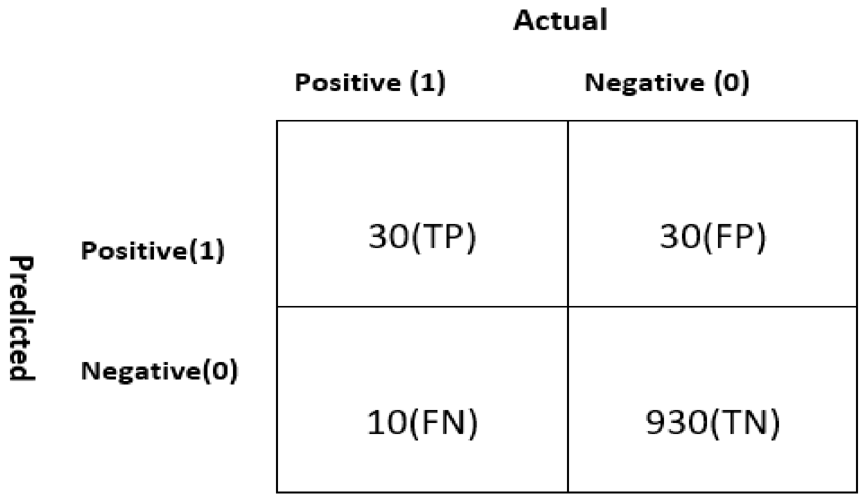
- $ACC = \frac{TP + TN}{TP + TN + FN + FP} = \frac{TP + TN}{P + N}$

ACC= (2+5)/10 = 0.7

So the model is saying I can predict sick people 70% of the time.

ACCURACY(CONT)

- Let’s take another example.

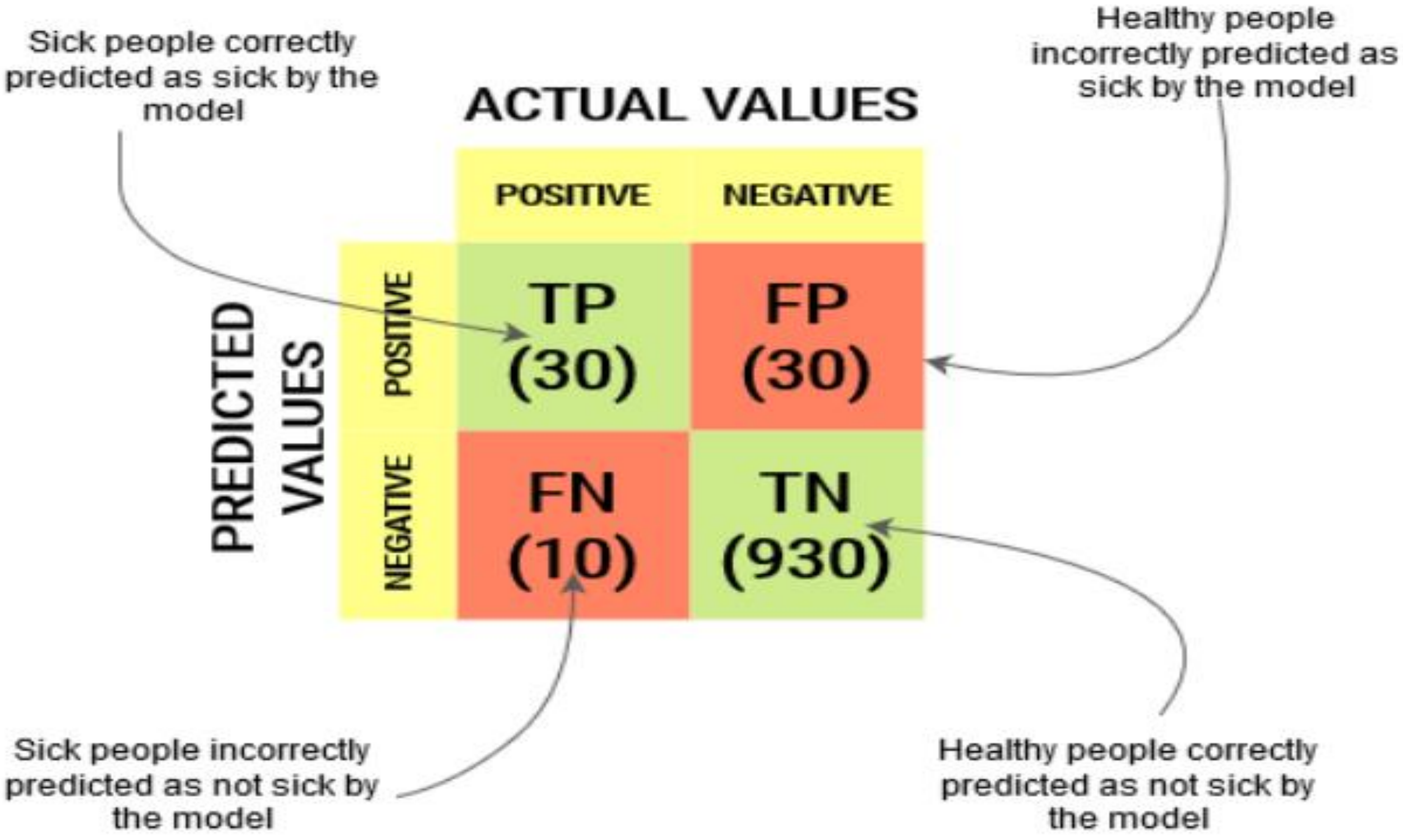


		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	30(TP)	30(FP)	60
	Negative(0)	10(FN)	930(TN)	940
	Total	40	960	1000

ACC=(TP+TN)/P+N = (30+930)/1000 = 0.96

- Our model is saying “I can predict sick people 96% of the time”.

CONFUSION MATRIX



		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP (30)	FP (30)
	NEGATIVE	FN (10)	TN (930)

Sick people correctly predicted as sick by the model

Healthy people incorrectly predicted as sick by the model

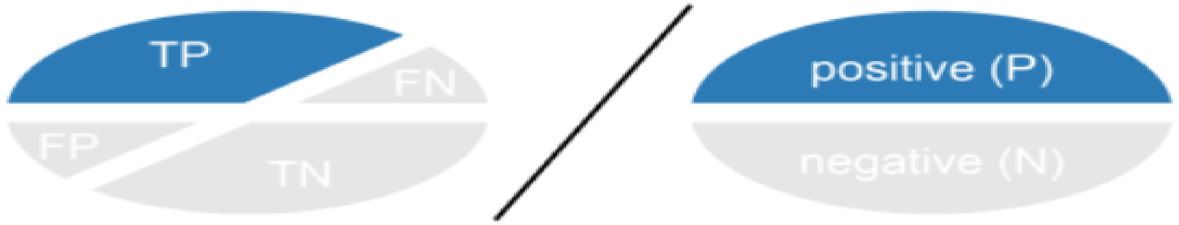
Sick people incorrectly predicted as not sick by the model

Healthy people correctly predicted as not sick by the model

RECALL

- Recall (REC) is calculated as the number of correct positive predictions divided by the total number of positives.
- It is also called sensitivity (SN) or true positive rate (TPR).
- The best recall is 1.0, where as the worst is 0.0.
- Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

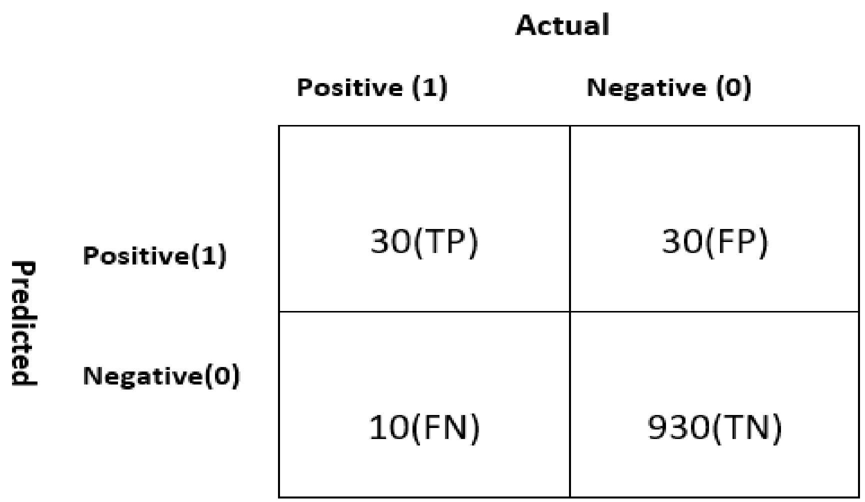
Sensitivity: TP / P



Sensitivity is calculated as the number of correct positive predictions (TP) divided by the total number of positives (P).

RECALL(CONT)

- The recall would be calculated by the following formula



		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	30(TP)	30(FP)	60
	Negative(0)	10(FN)	930(TN)	940
	Total	40	960	1000

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

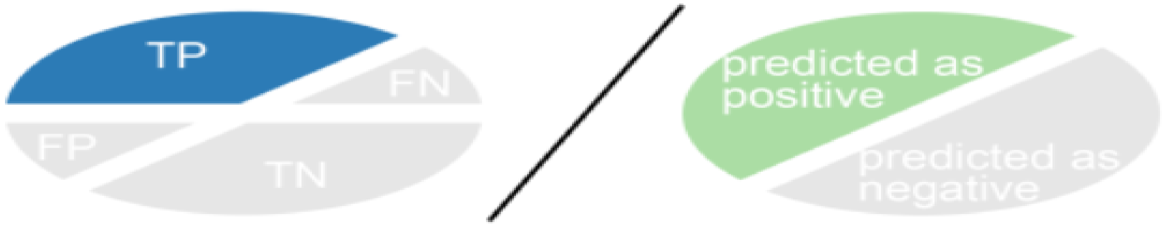
REC= 30/(30+10) = 0.75

- 75% of the positives were successfully predicted by our model. Awesome!

PRECISION

- Precision (PREC) is calculated as the number of correct positive predictions divided by the total number of positive predictions.
- It is also called positive predictive value (PPV).
- The best precision is 1.0, whereas the worst is 0.0.
- Precision tells us how many of the correctly predicted cases actually turned out to be positive.

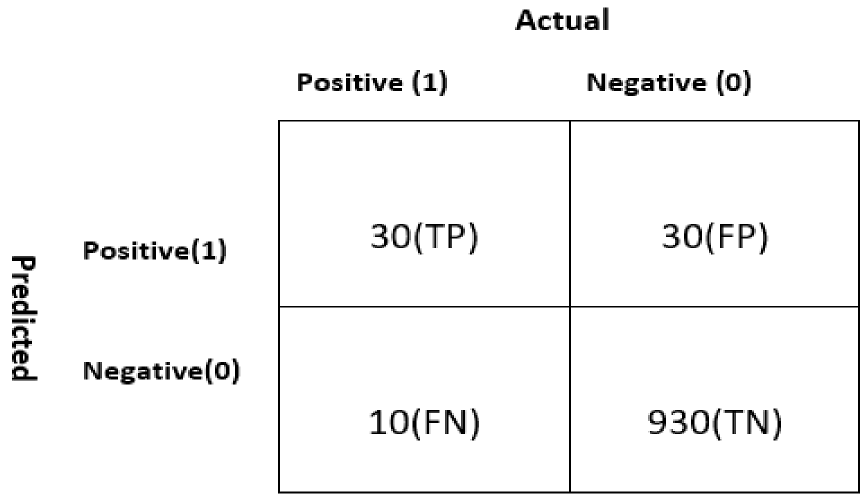
Precision: TP / (TP + FP)



Precision is calculated as the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP).

PRECISION(CONT.)

- The precision would be calculated by the following formula



		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	30(TP)	30(FP)	60
	Negative(0)	10(FN)	930(TN)	940
	Total	40	960	1000

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

Precision= 30/(30+30)

- 50% percent of the correctly predicted cases turned out to be positive cases
- This would determine whether our model is reliable or not.

F-SCORE

- In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa.
- The F1-score captures both the trends in a single value.
- **F1-score is a harmonic mean of Precision and Recall**, and so it gives a combined idea about these two metrics.
- It is maximum when Precision is equal to Recall.

- $$F_\beta = \frac{(1+\beta^2)(\mathrm{PREC} \cdot \mathrm{REC})}{(\beta^2 \cdot \mathrm{PREC} + \mathrm{REC})}$$

F-SCORE(CONT)

β is commonly 0.5, 1, or 2.

Gives more weight to precision

- $$F_{0.5} = \frac{1.25 \cdot \mathrm{PREC} \cdot \mathrm{REC}}{0.25 \cdot \mathrm{PREC} + \mathrm{REC}}$$

Provides a balance between precision and recall

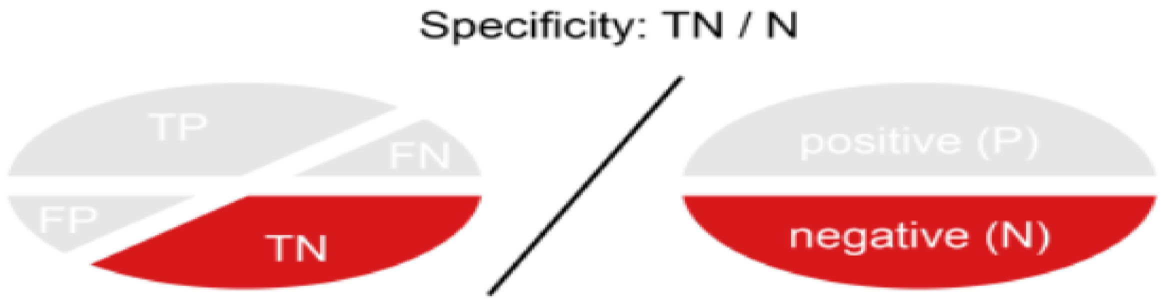
- $$F_1 = \frac{2 \cdot \mathrm{PREC} \cdot \mathrm{REC}}{\mathrm{PREC} + \mathrm{REC}}$$

Gives more weight to recall

- $$F_2 = \frac{5 \cdot \mathrm{PREC} \cdot \mathrm{REC}}{4 \cdot \mathrm{PREC} + \mathrm{REC}}$$

SPECIFICITY (TRUE NEGATIVE RATE)

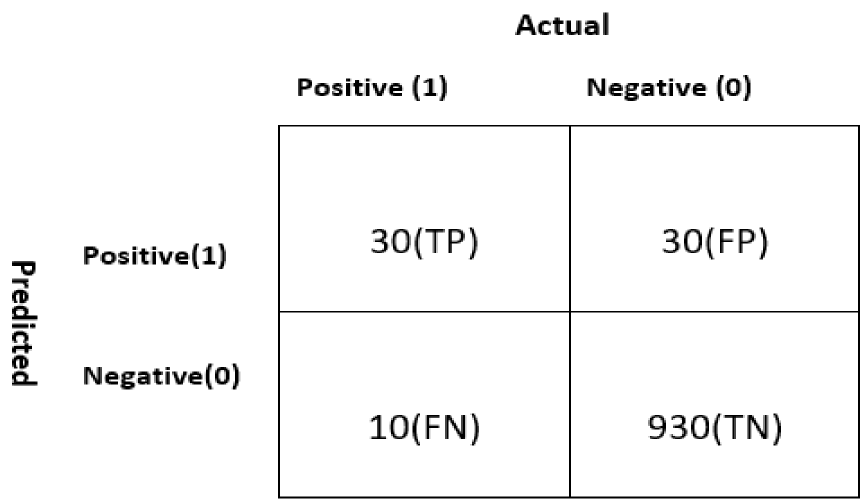
- Specificity (SP) is calculated as the number of correct negative predictions divided by the total number of negatives.
- It is also called true negative rate (TNR).
- The best specificity is 1.0, whereas the worst is 0.0.


Specificity: TN / N

Specificity is calculated as the number of correct negative predictions (TN) divided by the total number of negatives (N).

SPECIFICITY (TRUE NEGATIVE RATE)

- The specificity would be calculated by the following formula



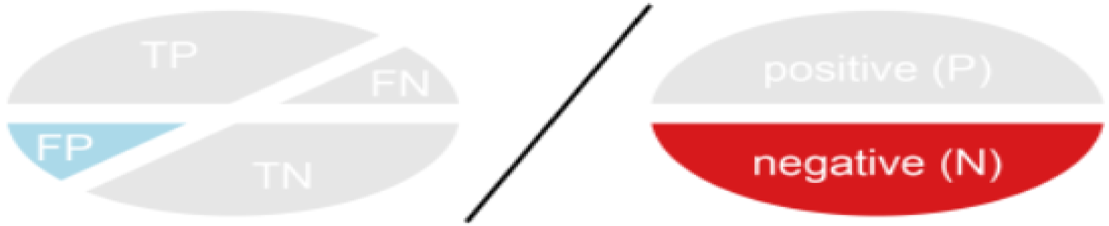
		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	30(TP)	30(FP)	60
	Negative(0)	10(FN)	930(TN)	940
	Total	40	960	1000

- $\mathrm{SP} = \dfrac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = \dfrac{\mathrm{TN}}{\mathrm{N}}$

SP=930/960 = 0.96

FALSE POSITIVE RATE

- False positive rate (FPR) is calculated as the number of incorrect positive predictions divided by the total number of negatives.
- The best false positive rate is 0.0 whereas the worst is 1.0.
- It can also be calculated as 1 – specificity.

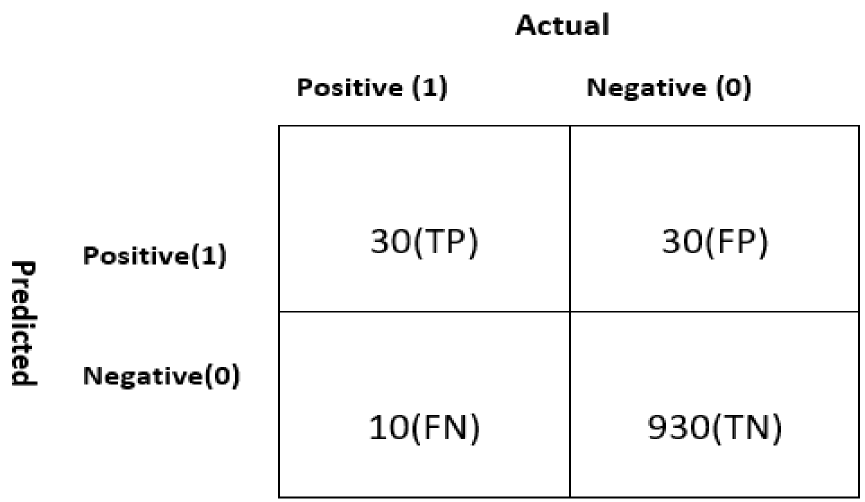


False positive rate: FP / N

False positive rate is calculated as the number of incorrect positive predictions (FP) divided by the total number of negatives (N).

FALSE POSITIVE RATE

- The specificity would be calculated by the following formula



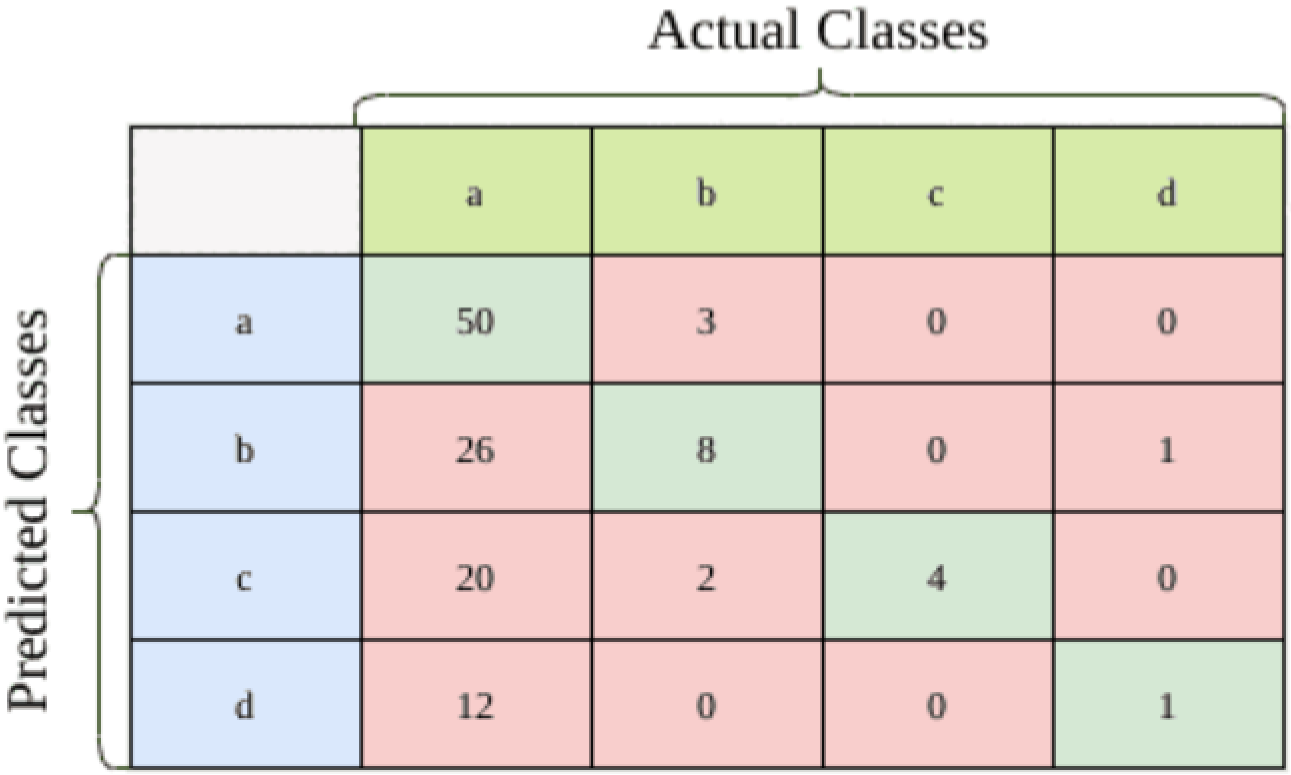
		Actual		
		Positive (1)	Negative (0)	Total
Predicted	Positive(1)	30(TP)	30(FP)	60
	Negative(0)	10(FN)	930(TN)	940
	Total	40	960	1000

- $\mathrm{FPR} = \dfrac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{SP}$

FPR=30/960 = 0.031

MULTI-CLASS CLASSIFICATION PROBLEM

- As an illustration, let’s consider the confusion matrix below with a total of 127 samples:



Actual Classes

Predicted Classes

	a	b	c	d
a	50	3	0	0
b	26	8	0	1
c	20	2	4	0
d	12	0	0	1

$$\text{Precision}(class = a) = \frac{TP(class = a)}{TP(class = a) + FP(class = a)} = \frac{50}{53} = 0.943$$

$$\text{Recall}(class = a) = \frac{TP(class = a)}{TP(class = a) + FN(class = a)} = \frac{50}{108} = 0.463$$

Then, we apply the formula for *class a*:

$$\text{F-1 Score}(class = a) = \frac{2 \times \text{Precision}(class = a) \times \text{Recall}(class = a)}{\text{Precision}(class = a) + \text{Recall}(class = a)} = \frac{2 \times 0.943 \times 0.463}{0.943 + 0.463} = 0.621$$

Similarly, we first calculate the precision and recall values for the other classes:

$$\text{Precision}(class = b) = \tfrac{8}{35} = 0.228 \quad \text{Recall}(class = b) = \tfrac{8}{13} = 0.615$$

$$\text{Precision}(class = c) = \tfrac{4}{26} = 0.154 \quad \text{Recall}(class = c) = \tfrac{4}{4} = 1.000$$

$$\text{Precision}(class = d) = \tfrac{1}{13} = 0.077 \quad \text{Recall}(class = d) = \tfrac{1}{2} = 0.500$$

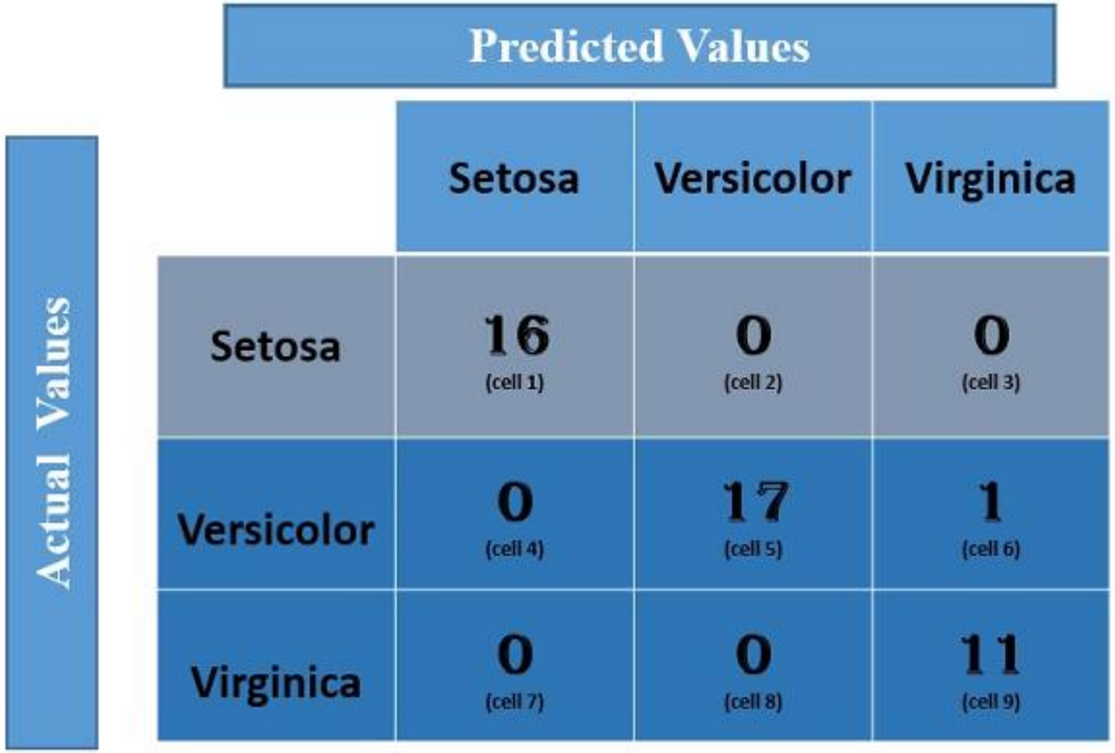
The calculations then lead to per-class F-1 scores for each class:

$$\text{F-1 Score}(class = b) = \frac{2 \times 0.228 \times 0.615}{0.228 + 0.615} = 0.333$$

$$\text{F-1 Score}(class = c) = \frac{2 \times 0.154 \times 1.000}{0.154 + 1.000} = 0.267$$

$$\text{F-1 Score}(class = d) = \frac{2 \times 0.077 \times 0.500}{0.077 + 0.500} = 0.133$$

ACTIVITY

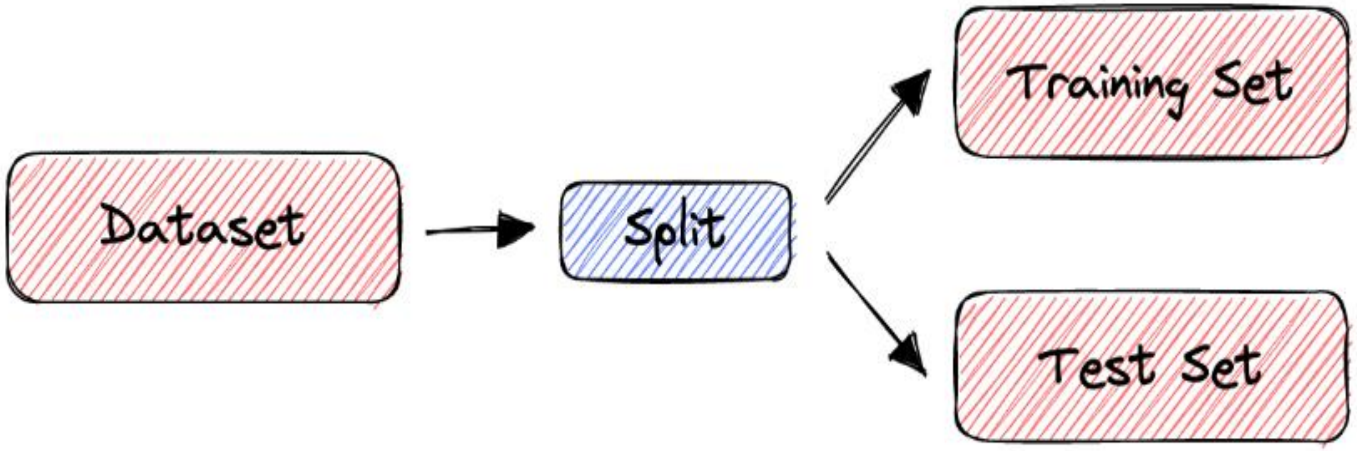


		Predicted Values		
		Setosa	Versicolor	Virginica
Actual Values	Setosa	16 (cell 1)	0 (cell 2)	0 (cell 3)
	Versicolor	0 (cell 4)	17 (cell 5)	1 (cell 6)
	Virginica	0 (cell 7)	0 (cell 8)	11 (cell 9)

TEST AND TRAIN SPLIT

- An important decision when developing any machine learning model is how to evaluate its final performance
- To get an unbiased estimate of the model's performance, we need to evaluate it on the data we didn't use for training.
- The simplest way to split the data is to use the train-test split method.
- It randomly partitions the dataset into two subsets (called training and test sets) so that the predefined percentage of the entire dataset is in the training set.
- Then, we train our machine learning model on the training set and evaluate its performance on the test set. In this way, we are always sure that the samples used for training are not used for evaluation and vice versa.
- Usually, we use 80/20 split.

TEST AND TRAIN SPLIT



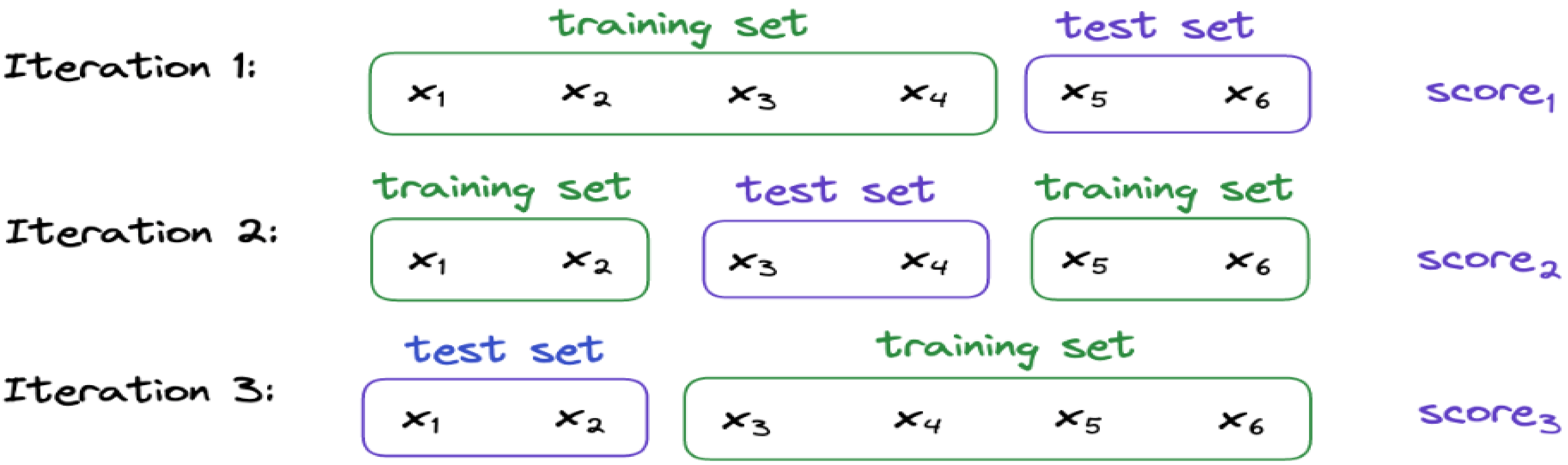
CROSS VALIDATION

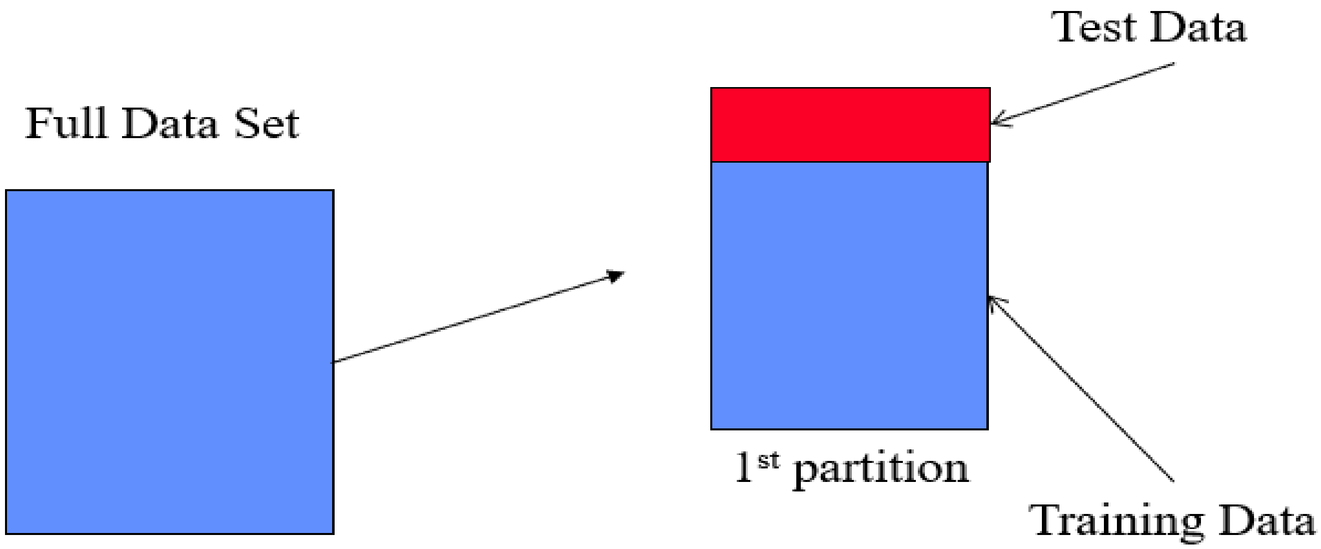
- The train-split method has certain limitations. When the dataset is small, the method is prone to high variance.
- To deal with this issue, we use cross-validation to evaluate the performance of a machine-learning model.
- In cross-validation, we don't divide the dataset into training and test sets only once.
- Instead, we repeatedly partition the dataset into smaller groups and then average the performance in each group. That way, we reduce the impact of partition randomness on the results.

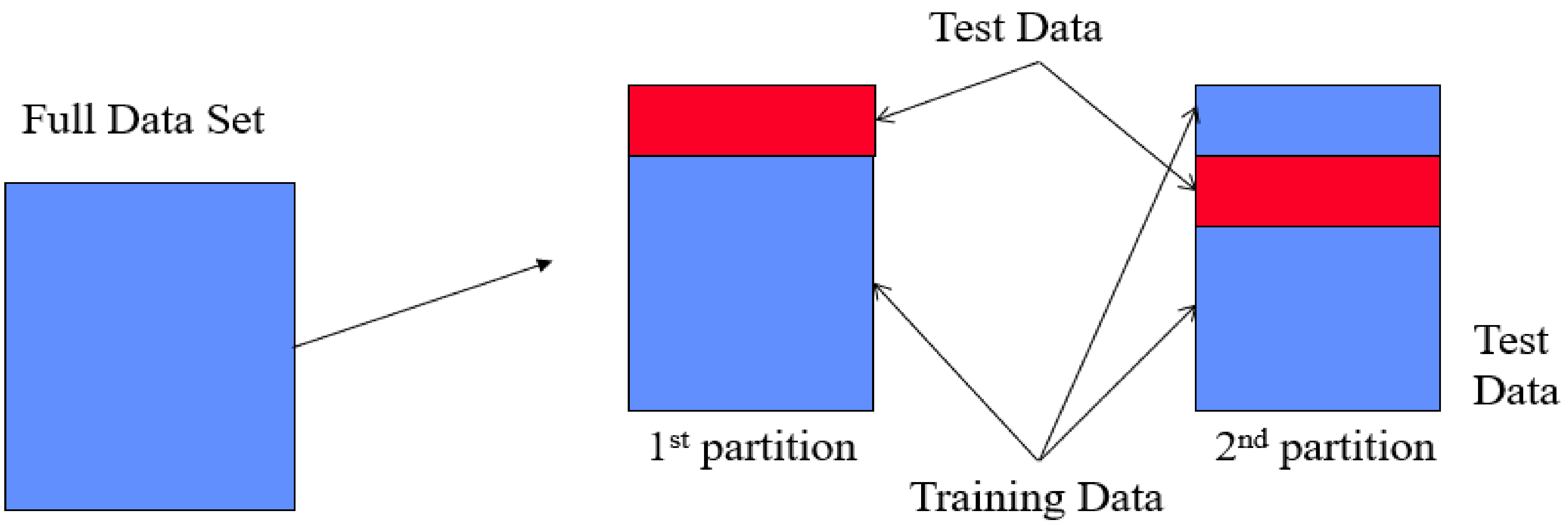
K-FOLD CROSS VALIDATION METHOD

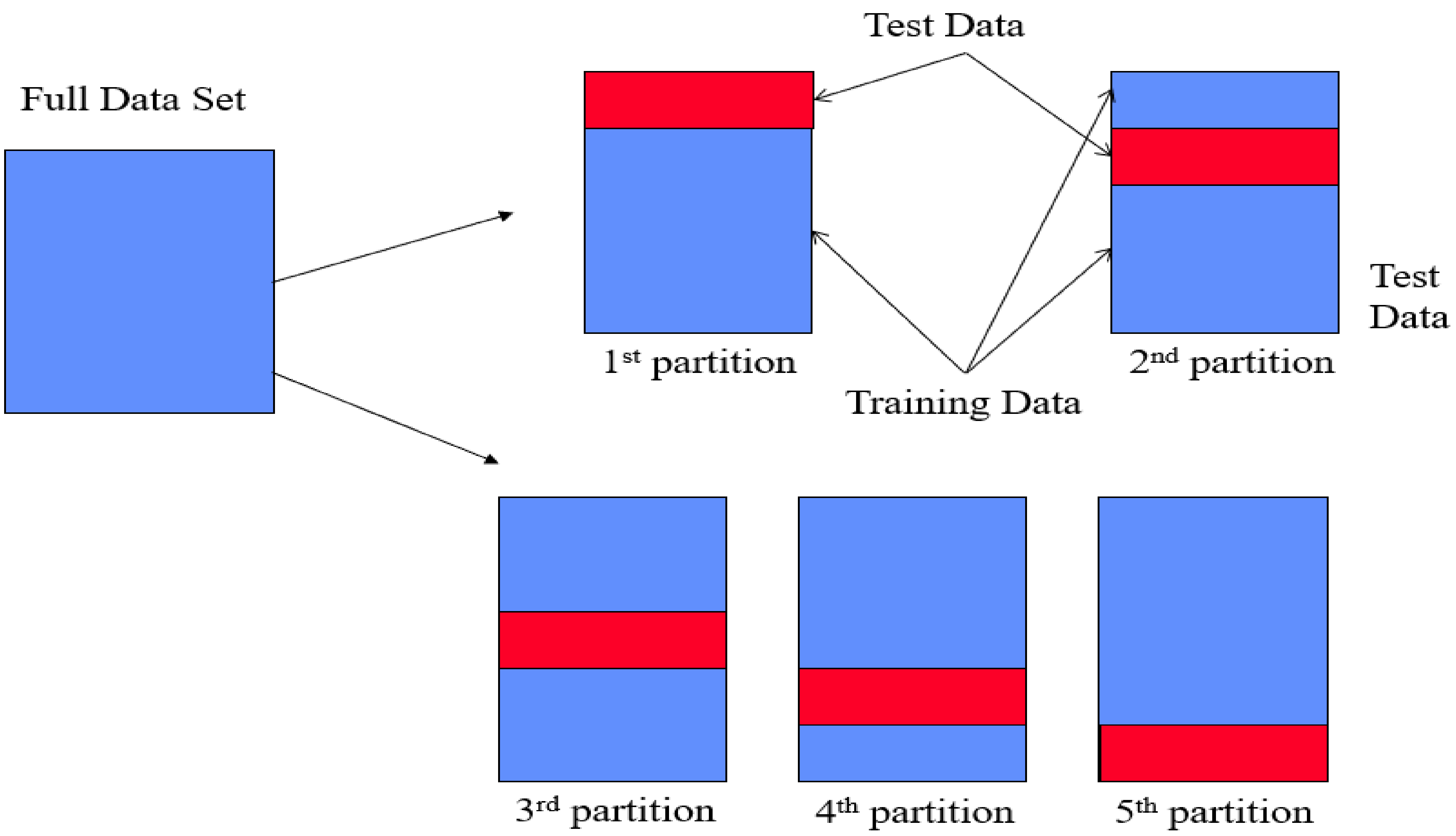
- “k-fold Cross-Validation” (e.g., k=10)
  - randomly partition our full data set into k disjoint subsets (each roughly of size n/k, n = total number of training data points)
    - for i = 1:10 (here k = 10)
      - train on 90% of data,
      - Acc(i) = accuracy on other 10%
    - end
    - Cross-Validation-Accuracy = $1/k \ \Sigma_i$ Acc(i)
  - choose the method with the highest cross-validation accuracy
  - common values for k are 5 and 10
  - Can also do “leave-one-out” where k = n

$$S = \{x_1, x_2, x_3, x_4, x_5, x_6\} \longrightarrow \begin{aligned} S_1 &= \{x_1, x_2\} \\ S_2 &= \{x_3, x_4\} \\ S_3 &= \{x_5, x_6\} \end{aligned}$$




Full Data Set
Test Data
1st partition
Training Data

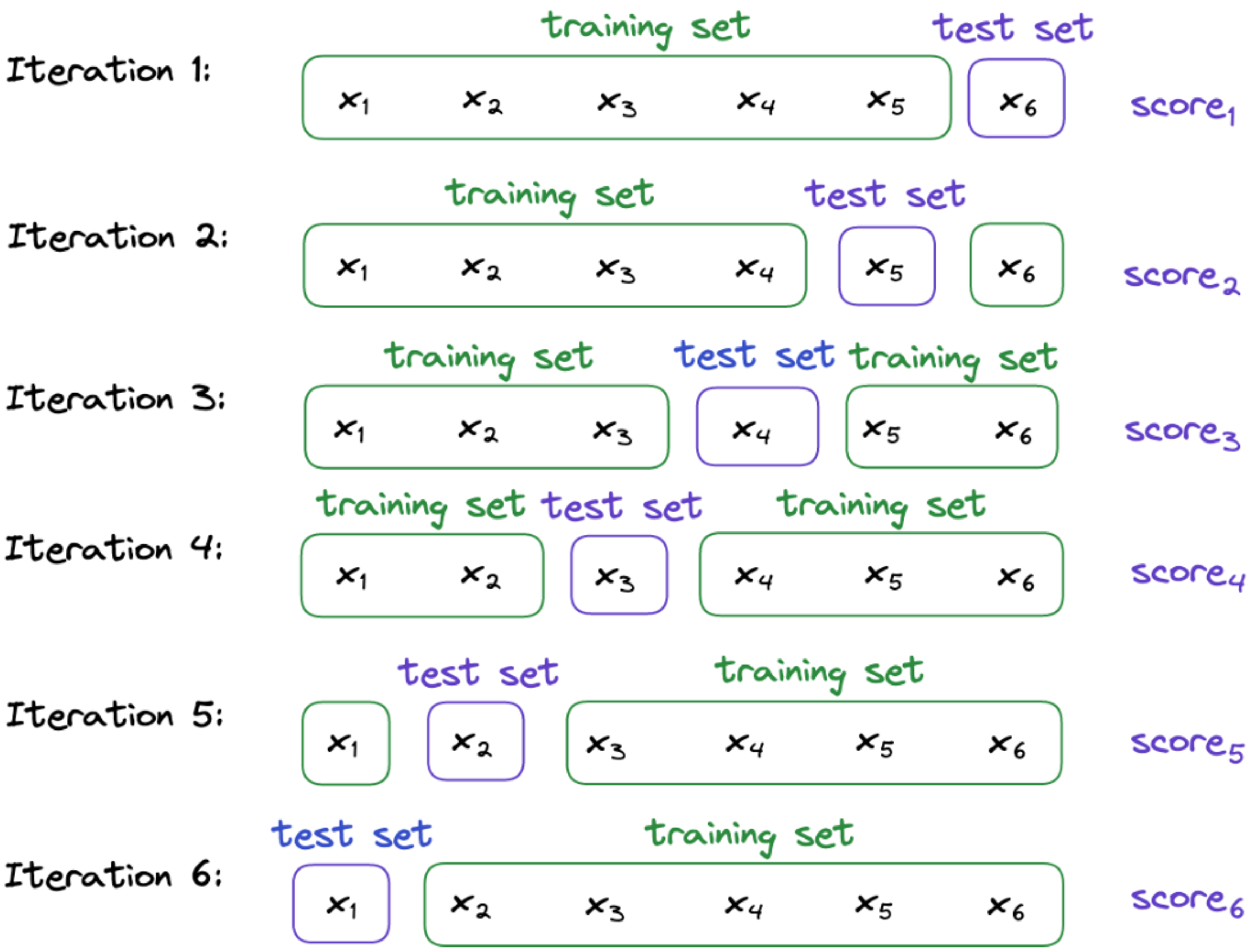

Full Data Set
Test Data
1st partition
Training Data
2nd partition
Test Data


Full Data Set
Test Data
1st partition
2nd partition
Test Data
Training Data
3rd partition
4th partition
5th partition

LEAVE-ONE-OUT METHOD

- Leave-one-out cross-validation, or LOOCV, is a configuration of k-fold cross-validation where *k* is set to the number of examples in the dataset.
- Each time, only one sample is used as a test set while the rest are used to train our model.
- LOOCV is an extreme version of k-fold cross-validation that has the maximum computational cost. It requires one model to be created and evaluated for each example in the training dataset.
- Given the computational cost, LOOCV is not appropriate for very large datasets such as more than tens or hundreds of thousands of examples

$$S_1 = \{x_1\}$$
$$S_2 = \{x_2\}$$
$$S_3 = \{x_3\}$$
$$S_4 = \{x_4\}$$
$$S_5 = \{x_5\}$$
$$S_6 = \{x_6\}$$



Iteration 1: training set [x_1 x_2 x_3 x_4 x_5] test set [x_6] score_1

Iteration 2: training set [x_1 x_2 x_3 x_4] test set [x_5] [x_6] score_2

Iteration 3: training set [x_1 x_2 x_3] test set [x_4] training set [x_5 x_6] score_3

Iteration 4: training set [x_1 x_2] test set [x_3] training set [x_4 x_5 x_6] score_4

Iteration 5: [x_1] test set [x_2] training set [x_3 x_4 x_5 x_6] score_5

Iteration 6: test set [x_1] training set [x_2 x_3 x_4 x_5 x_6] score_6