UNIVERSIDAD
NACIONAL
DE COLOMBIA

# CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES
Evaluación de Desempeño

John William Branch

Esmeide Alberto Leal Narváez

Gidia Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

## Agenda

- **Introduction**
- **Classification**
    - Confusion Matrix
    - Precision, Recall, Specificity, F1 Score
    - Precision-Recall AUC
    - ROC/AUC Curve
    - Hold-out, Cross Validation, Leave-one-out (Accuracy Estimation)
- **Examples**
- **Regression**
    - MAE, MSE
    - RMSE, RMSLE, R-Square
- **Examples**

## Introduction

- **What is an evaluation metric?**
- A way to quantify performance of a machine learning model

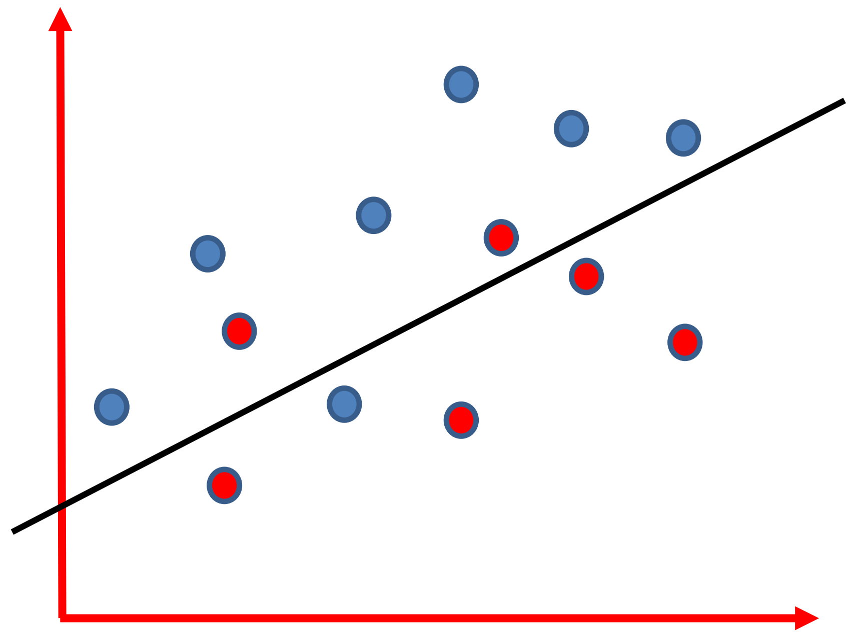
- Evaluation metric ≠ Loss function

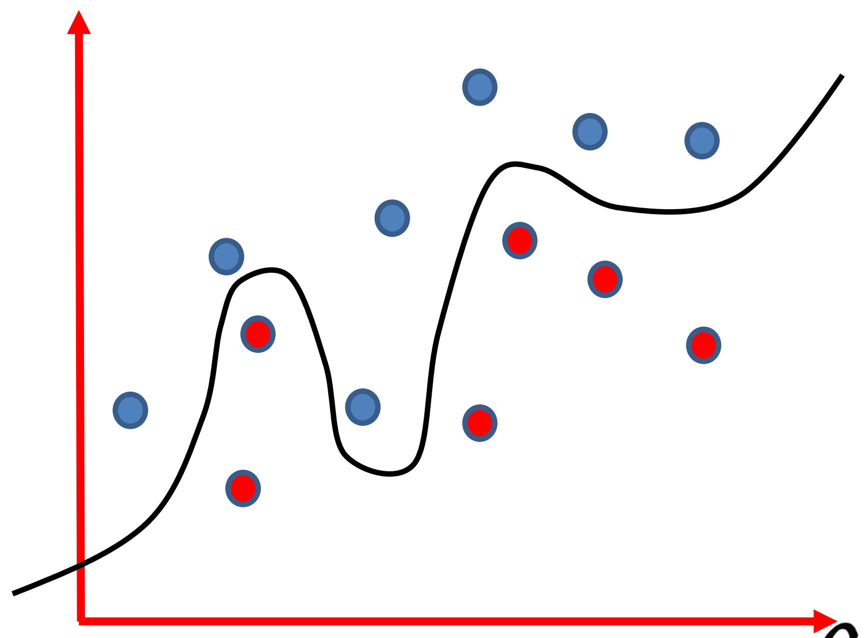
- There are various metrics which we can use to evaluate the performance of ML algorithms, classification as well as regression algorithms.

- We must carefully choose the metrics for evaluating ML model performance because
  - How the performance of ML algorithms is measured and compared will be dependent entirely on the metric you choose.
  - How you weight the importance of various characteristics in the result will be influenced completely by the metric you choose.

## Introduction

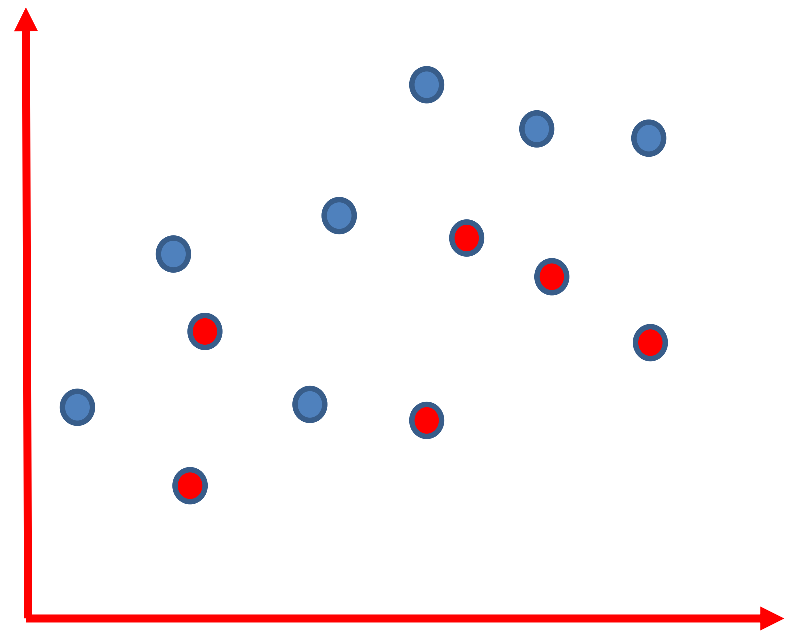
- Testing: How well is my model doing?
    - Is good or not
- How do I improve it based on these metrics?

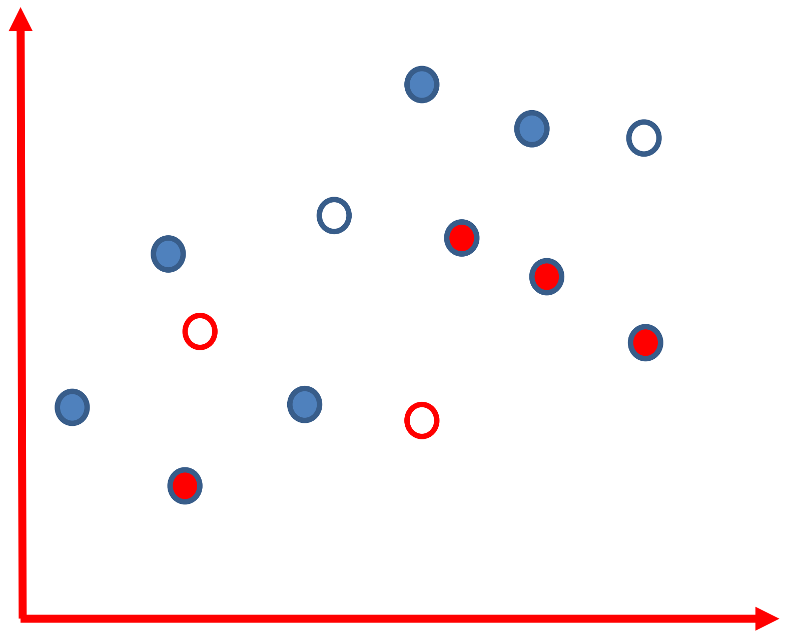


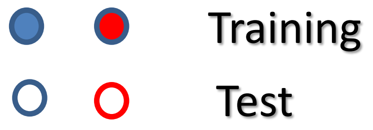


## Introduction

- Testing: How well is my model doing?
    - Is good or not
- How do I improve it based on these metrics?

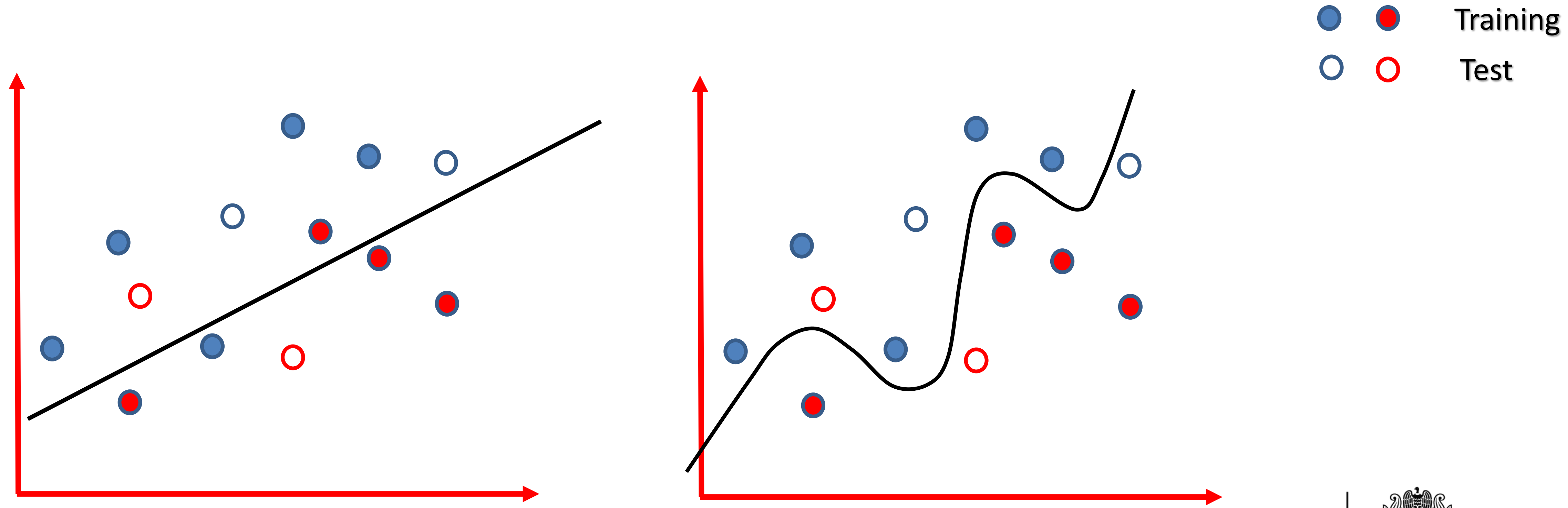






## Introduction

- Testing: How well is my model doing?
    - Is good or not
- How do I improve it based on these metrics?

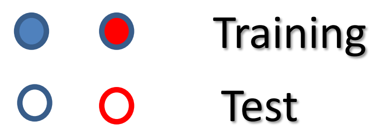


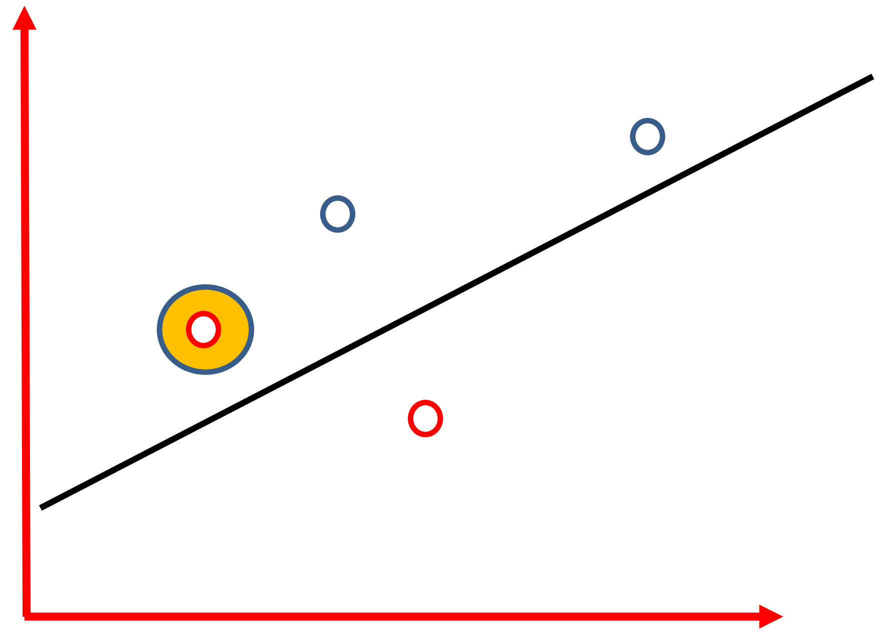
Gidia Grupo de I+D en Inteligencia Artificial

Universidad Nacional de Colombia

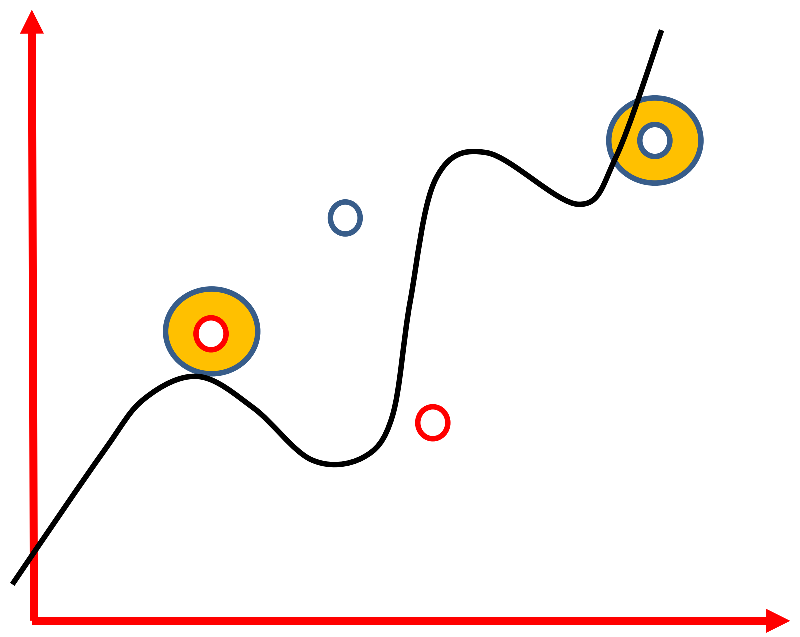
# Introduction

- Testing: How well is my model doing?
  - Is good or not
- How do I improve it based on these metrics?





Generalize



Overfitting - Memorize

## Confusion Matrix

- A confusion matrix is a predictive analytics tool
- Not a metric.
- Specifically, it is a table that displays and compares actual values with the model's predicted values.
- Within the context of machine learning, a confusion matrix is utilized to analyze how a machine learning classifier performed on a dataset.
- Help to understand some other metrics.
- A confusion matrix generates a visualization of metrics like precision, accuracy, specificity, and recall.

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

- Formal Definition
  - The confusion matrix, **T**, is a *n x n matrix, where n is the number of classes of our data. The element T(i, j) of the confusion matrix is defined as the number of samples that belong to class* $C_i$ *and were classified as* $C_j$**.** *A perfect classification means that T(i,i) is* $N_i$ *and T(i,j)=0 for* $i \neq j$, *where* $N_i$ *is the number of samples of class* $C_i$.

## Confusion Matrix

- Definition

Predicted

Actual

| | $C_1$ | $C_2$ | … | $C_n$ |
|---|---|---|---|---|
| $C_1$ | | | | |
| $C_2$ | | | | |
| … | | | | |
| $C_n$ | | | | |

## Confusion Matrix

- Definition

Predicted

Actual

| | $C_1$ | $C_2$ | … | $C_n$ |
|---|---|---|---|---|
| $C_1$ | 1350 | 80 | | 10 |
| $C_2$ | 77 | 950 | | 5 |
| … | | | | |
| $C_n$ | 30 | 64 | | 895 |

Example: there are 80 samples of class 1 that have been classified as class 2.

Grupo de I+D en Inteligencia Artificial

Universidad Nacional de Colombia

## Confusion Matrix

- Two Class (Detection)

Predicted

Actual

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | | |
| $C_1$ | | |

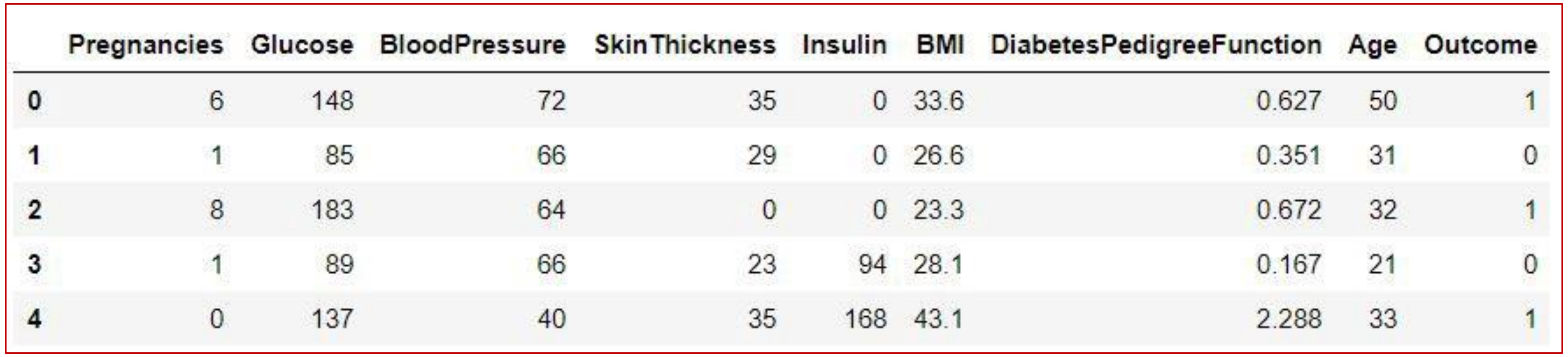
- $C_0$ = Target Class
- $C_1$ = Not Target Class

- Example: Disease Detection, Anomaly Detection, Face Detection, etc.

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

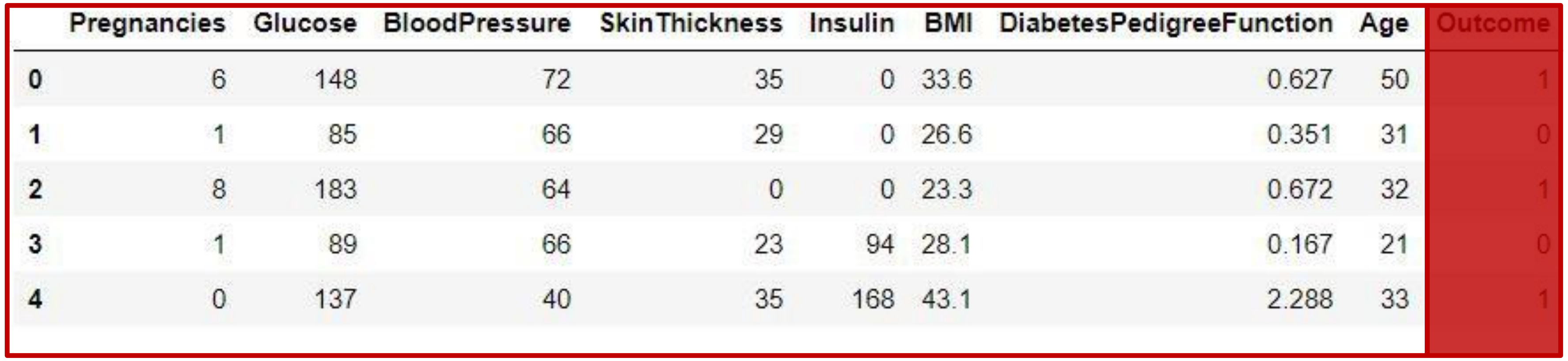
- Toy Example
  - Imagine that we have this diabetes dataset



| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

## Confusion Matrix

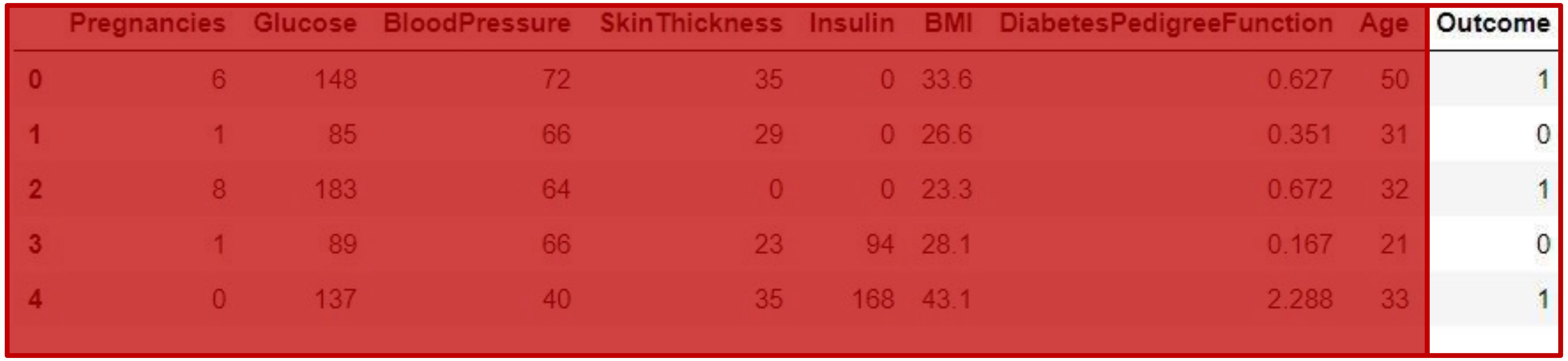
- Toy Example
  - We have some clinical measurements . . .



| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

## Confusion Matrix

- Toy Example
  - We want to apply a machine learning method to them to predict whether or not someone will develop diabetes disease.

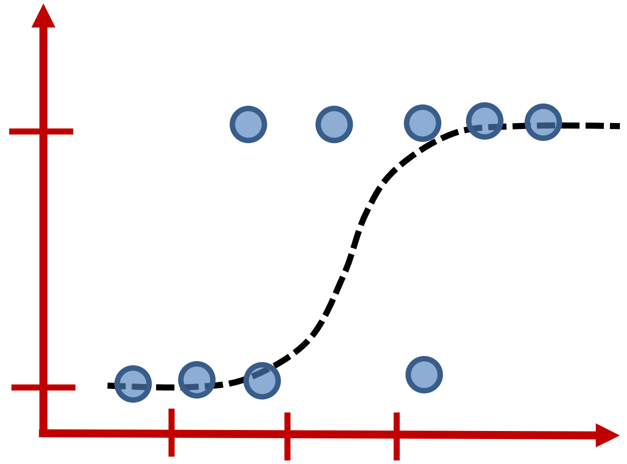


| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

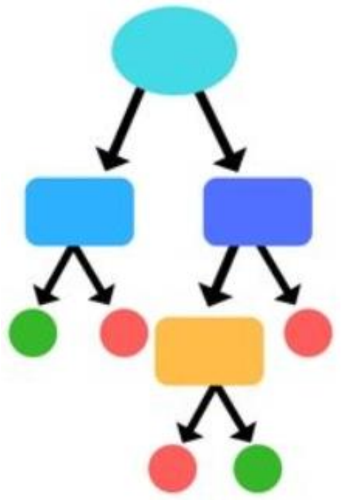
Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

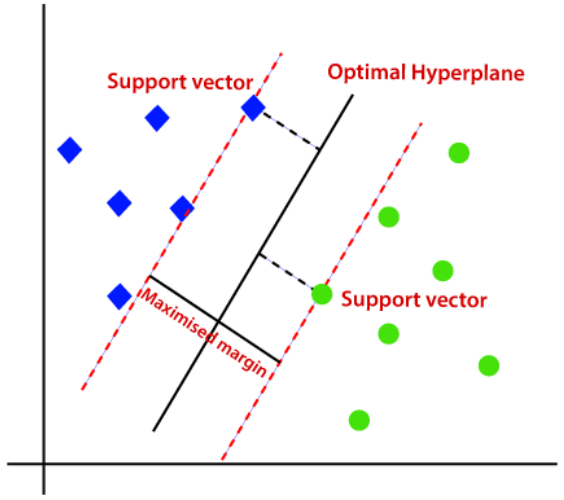
- To do this, we could use, different machine learning methods



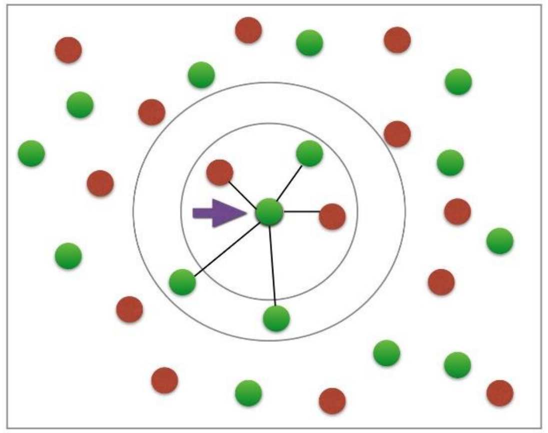
Logistic Regression



Random Forest



SVM



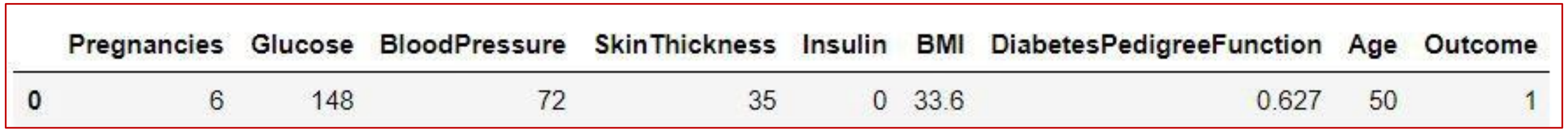
K-NN

etc.

**How to decide which one is the best with our data?**

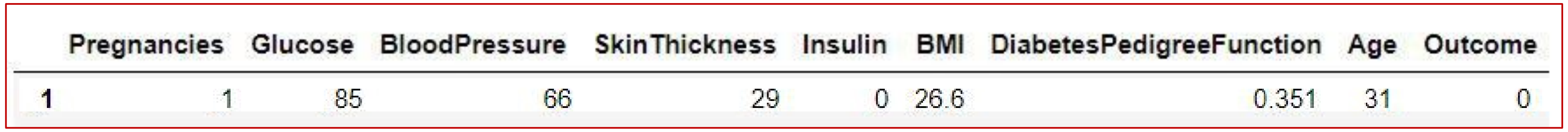
## Confusion Matrix

- Toy Example
  - We start by dividing the data into Training and Testing sets



| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |

Training Data

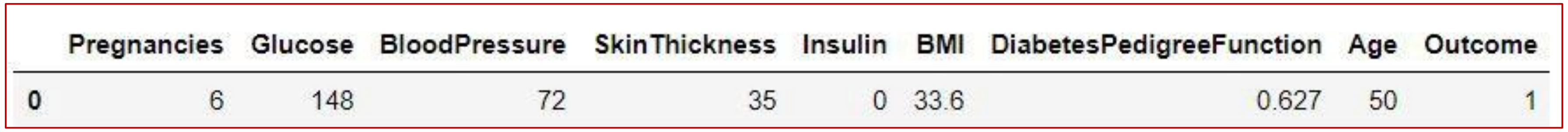


| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |

Testing Data

## Confusion Matrix

- Toy Example
  - We start by dividing the data into Training and Testing sets



| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |

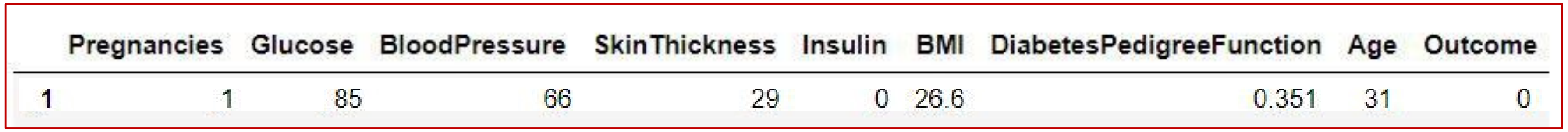
Training Data

Gidia Grupo de I+D en Inteligencia Artificial

Universidad Nacional de Colombia

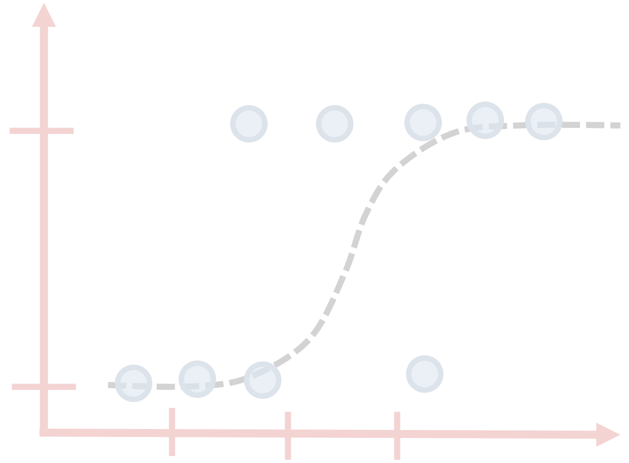
## Confusion Matrix

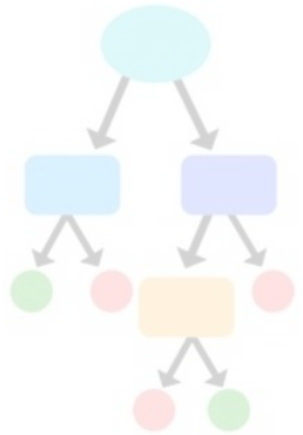
- Toy Example
  - We start by dividing the data into Training and Testing sets

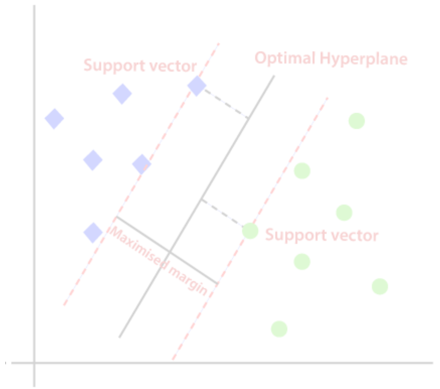


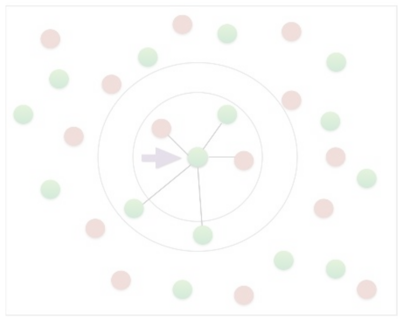
| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |

Testing Data



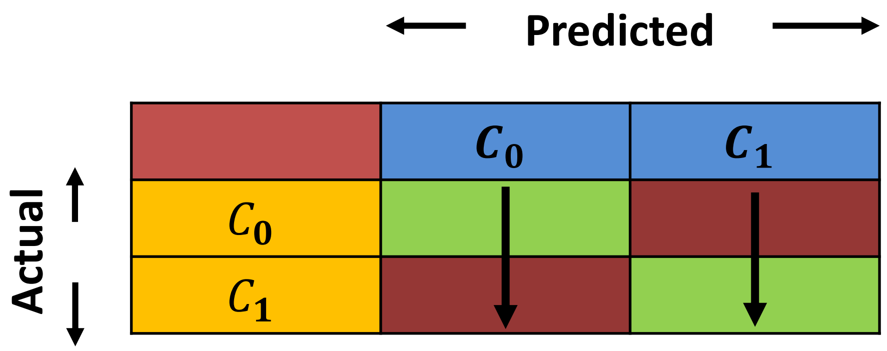






## Confusion Matrix

- We create a confusion matrix for each method

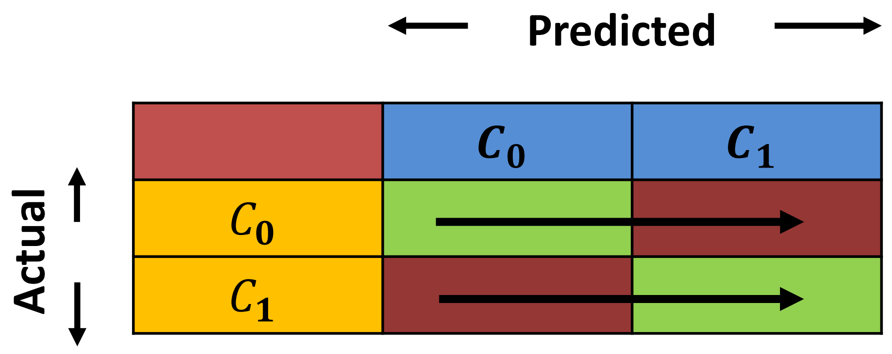


- The cols in the confusion matrix correspond to what machine learning algorithm predicted

- $C_0$= Has Diabetes
- $C_1$= Does Not Have Diabetes

## Confusion Matrix

- We create a confusion matrix for each method



- The rows in the confusion matrix correspond to the known truth

- $C_0$= Has Diabetes
- $C_1$= Does Not Have Diabetes

# Confusion Matrix

- Two Class

**Predicted**

**Actual**

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP | FN |
| $C_1$ | FP | TN |

- **True Positive** (**TP**): number of target correctly classified (**Patients with diabetes that were correctly identified by the algorithm**)
- **True Negative** (**TN**): number of non-target correctly classified (**Patients without diabetes that were correctly identified by the algorithm**)
- **False Positive** (**FP**): number of non-targets classified as targets. The false positives are known as “false alarms” and “Type I Error” (**Patients without diabetes, but the algorithm says they do**)
- **False Negative** (**FN**): number of targets classified as non-targets. The false negatives are known as “Type II Error” (**Patients with diabetes, but the algorithm said they don't have diabetes**)

## Confusion Matrix

- Two Class

Predicted

Actual

|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP | FN |
| $C_1$ | FP | TN |

**P = TP + FN** (Positive Instances)

**N = FP + TN** (Negative Instances)

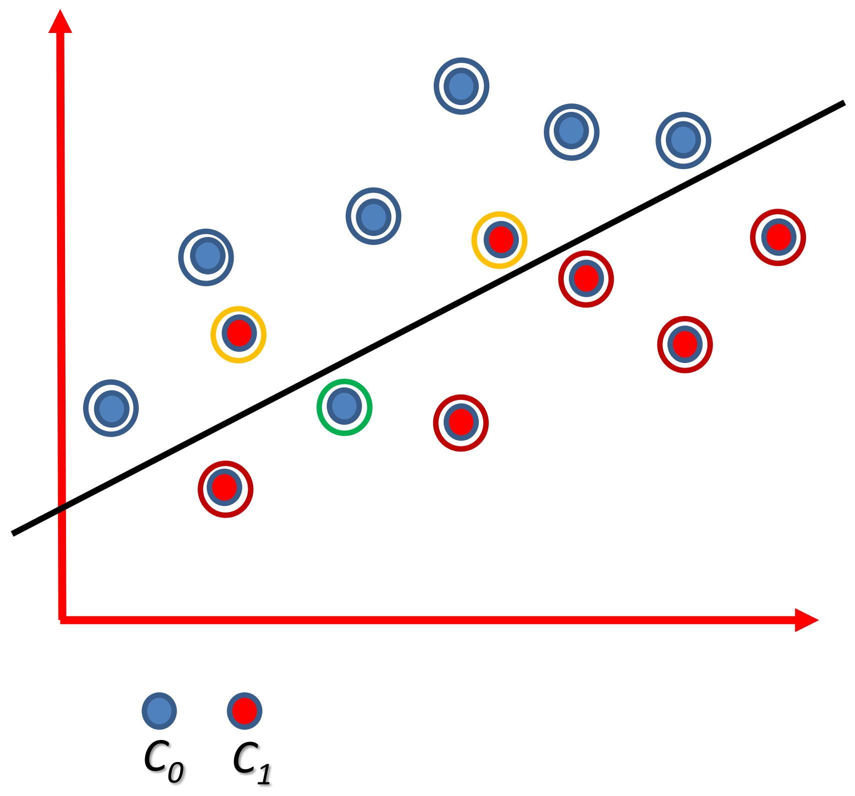
**D = TP + FP** (Detections)

- **True Positive** (**TP**): number of target correctly classified (**Patients with diabetes that were correctly identified by the algorithm**)
- **True Negative** (**TN**): number of non-target correctly classified (**Patients without diabetes that were correctly identified by the algorithm**)
- **False Positive** (**FP**): number of non-targets classified as targets. The false positives are known as “false alarms” and “Type I Error” (**Patients without diabetes, but the algorithm says they do**)
- **False Negative** (**FN**): number of targets classified as non-targets. The false negatives are known as “Type II Error” (**Patients with diabetes, but the algorithm said they don't have diabetes**)

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

Example



Predicted

Actual

|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP = 6 | FN = 1 |
| $C_1$ | FP = 2 | TN = 5 |

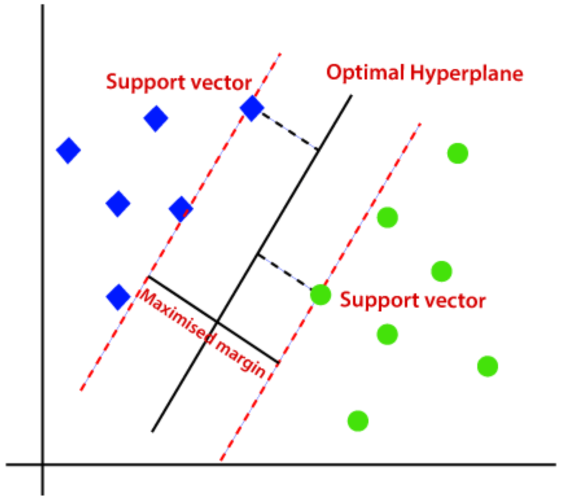
## Confusion Matrix

- Example

Predicted →

Actual ↕

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | |
| $C_1$ | | 98 |



- There are 130 True Positives, patients with diabetes that were correctly classified

- There are 98 True Negatives, patients without diabetes that were correctly classified

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

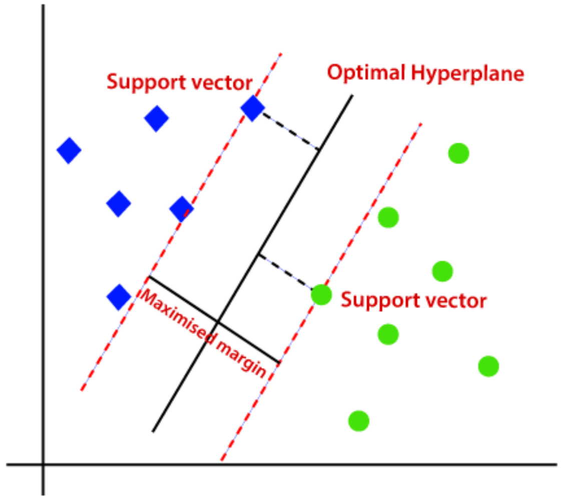
- Example

Predicted

Actual

|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ |  | 29 |
| $C_1$ | 11 |  |

- There are 19 patients, that the algorithm misclassified, saying that they have diabetes, when they don't. (False Negative)



- There are 11 patients, that the algorithm misclassified, saying that they don't have diabetes, when they do

## Confusion Matrix

- Example

Predicted →, Actual ↓

|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |

- The diagonal of the confusion matrix (blue cells) indicate how many times the samples were correctly classified.

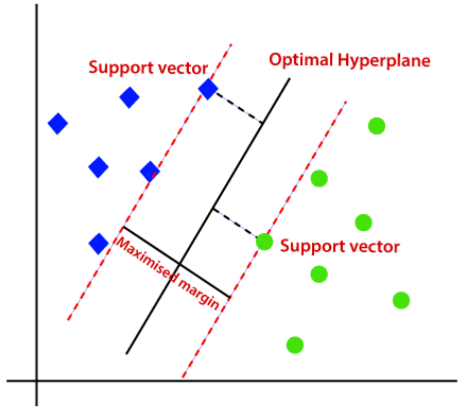
- The secondary diagonal of the confusion matrix (green cells) are samples that the algorithm misclassified.

Gidia Grupo de I+D en Inteligencia Artificial

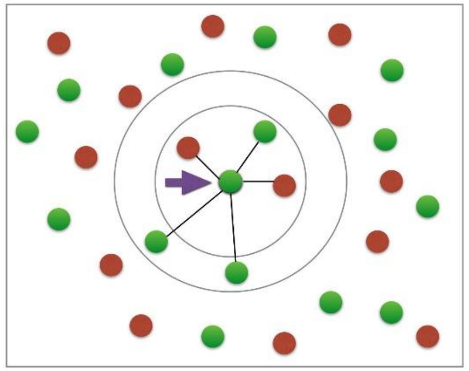
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

- Example



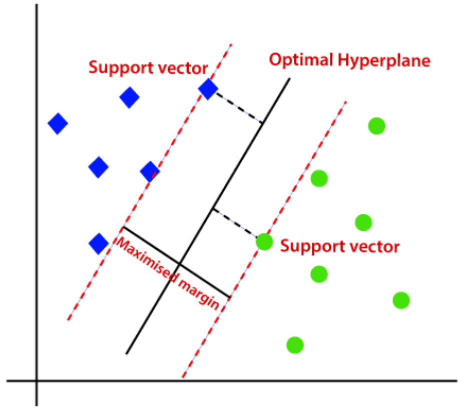
| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |



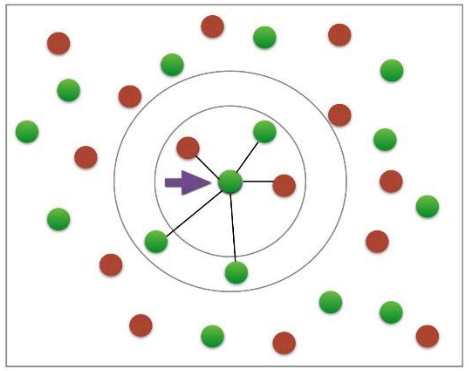
| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 106 | 38 |
| $C_1$ | 45 | 58 |

## Confusion Matrix

- Example



| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |



| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 106 | 38 |
| $C_1$ | 45 | 58 |

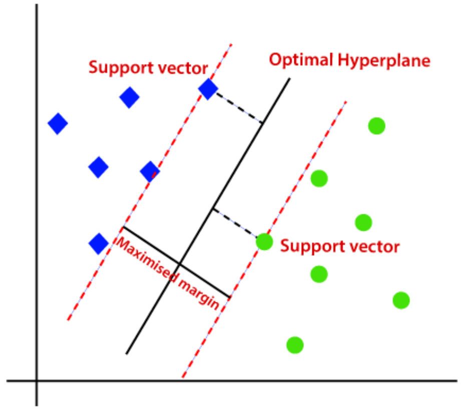
- The k-NN algorithm, had worse performance than the SVM algorithm, predicting patients with diabetes.

Gidia Grupo de I+D en Inteligencia Artificial

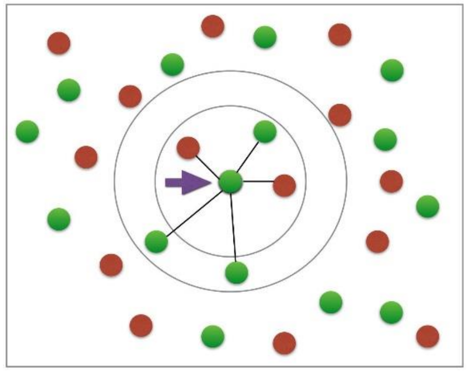
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

- Example



|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |



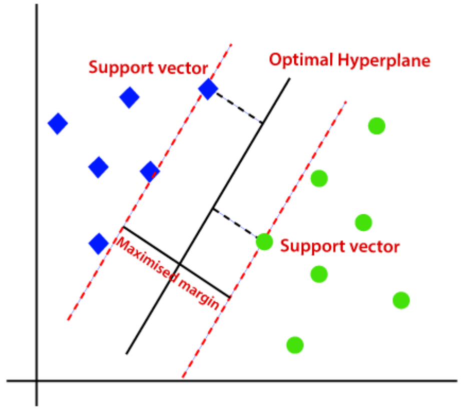
|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 106 | 38 |
| $C_1$ | 45 | 58 |

- The k-NN algorithm, had worse performance than the SVM algorithm, predicting patients without diabetes.

Gidia Grupo de I+D en Inteligencia Artificial
Universidad Nacional de Colombia

## Confusion Matrix

- Example



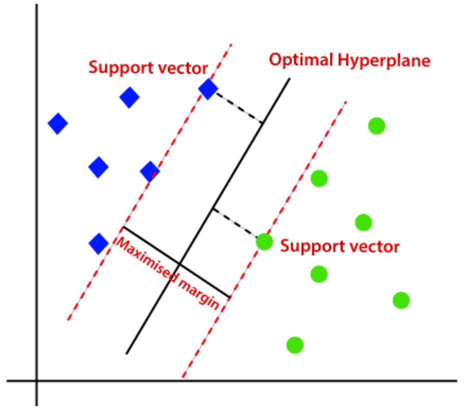
|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |

- Conclusion: we choose the SVM algorithm, instead The k-NN algorithm.

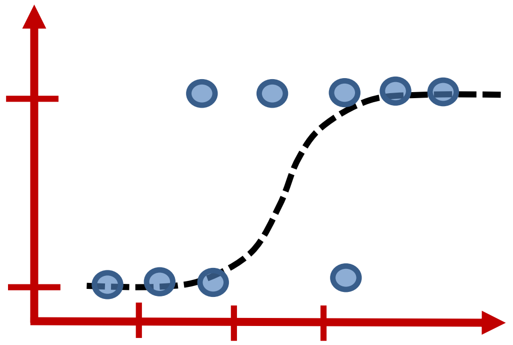
Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Confusion Matrix

- Example



| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |

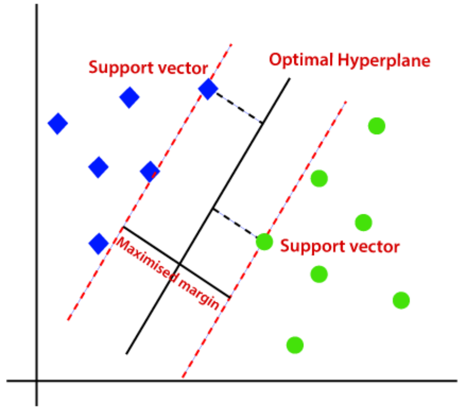


| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 127 | 29 |
| $C_1$ | 12 | 101 |

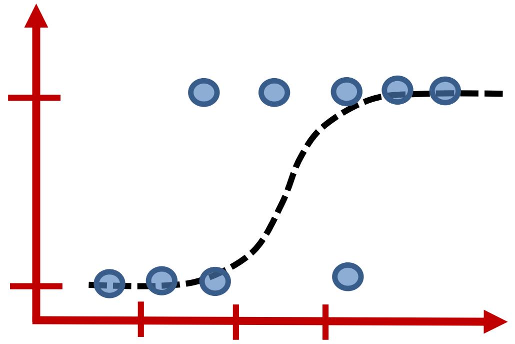
- Now we use logistic regression to testing the dataset and create its confusion matrix.

## Confusion Matrix

- Example



|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |



|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 127 | 29 |
| $C_1$ | 12 | 101 |

- Is difficult to choose what algorithm have best performance, then we use a more sophisticated metrics to help us to take a decision

## Confusion Matrix

```
from sklearn.metrics import confusion_matrix

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

results = confusion_matrix(X_actual, Y_predic)

print ('Confusion Matrix :')
print(results)
```

Output

```
Confusion Matrix:
[
   [3 3]
   [1 3]
]
```

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Metrics

- Definitions

True positive rate, known as Sensitivity or Recall:

$$TPR = S_n = Re = \frac{TP}{P} = \frac{TP}{TP+FN}$$

Precision or Positive Predictive Value:

$$Pr = \frac{TP}{D} = \frac{TP}{TP+FP}$$

True negative rate, known as Specificity:

$$TNR = Sp = \frac{TN}{N} = \frac{TN}{TN+FP}$$

False negative rate, known as Miss Rate:

$$FNR = MR = \frac{FN}{P} = \frac{FN}{TP+FN}$$

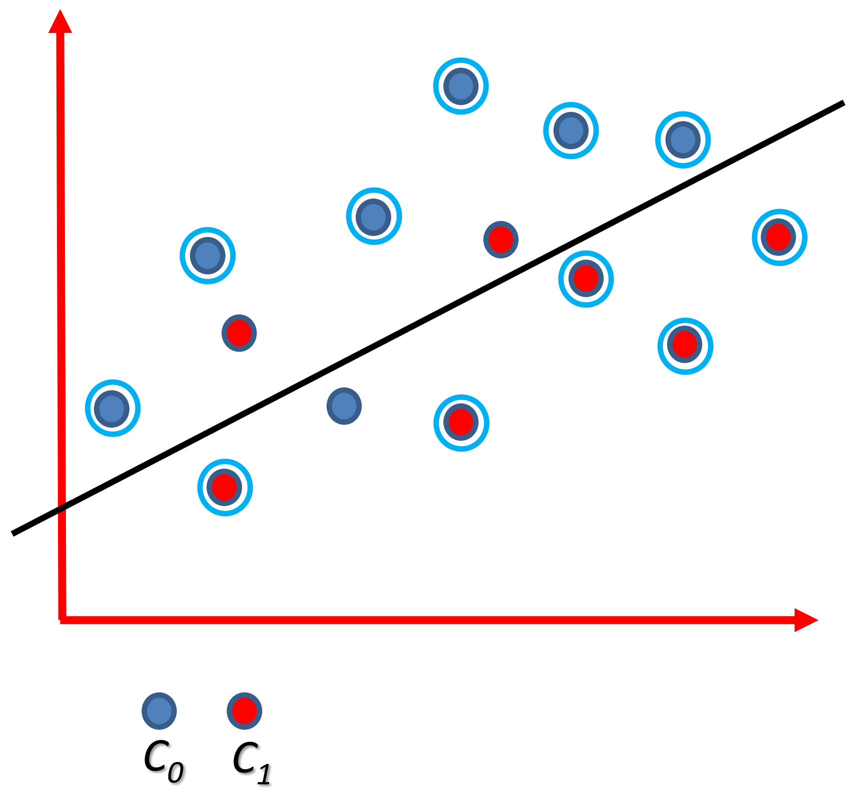
False positive rate, known as 1-Specificity:

$$FPR = 1 - Sp = \frac{FP}{N} = \frac{FP}{TN+FP}$$

Accuracy:

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

## Accuracy



Accuracy:

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

$$\text{Accuracy} = \frac{\text{Correctly Classified points}}{\text{All points}}$$

$$\text{Accuracy} = \frac{11}{11 + 3}$$

Accuracy = 78.57%

- Accuracy = Out of the all data, how many did we classified correctly?
- Is the correct metric?

## Accuracy

- Example

- Imagine the following problem: Suppose we must classify 10.000 peoples and we need to classify as Chinese or not Chinese.
- Now, we have that of these 10.000 people 99,990 are not Chinese and 10 are Chinese
- **Our classifier is a simple program that only returns false when evaluating anyone.**
- Evaluating Accuracy

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Accuracy

- From the confusion matrix, we can estimate Accuracy

Predicted →, Actual ↓

|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP | FN |
| $C_1$ | FP | TN |

Predicted →, Actual ↓

|  | $C_0$ | $C_1$ |  |
|---|---|---|---|
| $C_0$ | TP | FN | P |
| $C_1$ | FP | TN | N |

Accuracy:

$$ACC = \frac{\boxed{TP}+\boxed{TN}}{\boxed{P}+\boxed{N}} = \frac{99{,}990}{10{,}000} = 99{,}9\%$$

- Accuracy = Out of the all the Chinese people, how many did we classified correctly?
- What is wrong with this?  and why we need other performance measures?

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

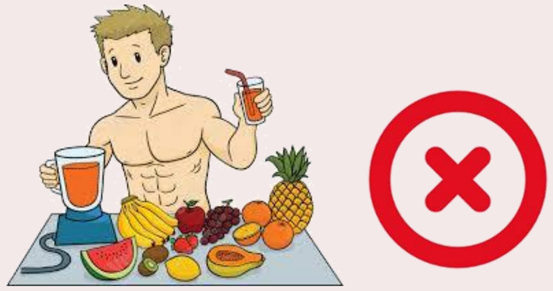
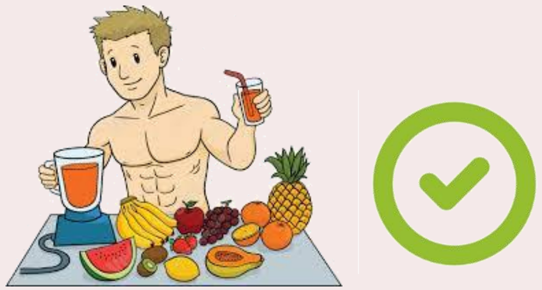
## Accuracy

```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

results = confusion_matrix(X_actual, Y_predic)

print ('Confusion Matrix :')
print(results)
print ('Accuracy Score is',accuracy_score(X_actual, Y_predic))
```

Output

```
Confusion Matrix:
[
   [3 3]
   [1 3]
]

Accuracy Score is 0.6
```

# Precision - Recall

| | Diagnosed Sick | Diagnosed Healthy |
|---|---|---|
| Sick | ✓ | ✗ |
| Healthy |  ✗ |  ✓ |

**Medical Model**

## Precision - Recall

| | Diagnosed Sick | Diagnosed Healthy |
|---|---|---|
| Sick | | ✗ |
| Healthy | ✗ | |

**Medical Model**

## Precision - Recall

| | Sent to Spam Folder | Sent to Inbox |
|---|---|---|
| Spam | ✓ | ✗ |
| Not Spam | ✗ | ✓ |

**Spam Detector**

## Precision - Recall

| | Sent to Spam Folder | Sent to Inbox |
|---|---|---|
| Spam | | |
| Not Spam | | |

**Spam Detector**

## Precision - Recall

- **Medical Model**
- False Positives Ok
- False Negative **NOT** Ok

- Find all the sick people
- Ok if not all are sick

- **High Recall**

- **Spam Detector**
- False Positives **NOT** Ok
- False Negative  Ok

- You don’t necessarily need to find all spam but they better all be spam

- **High Precision**

## Precision - Recall

- From the confusion matrix, we can estimate Precision and Recall

← Predicted →

Actual ↑↓

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP | FN |
| $C_1$ | FP | TN |

D

← Predicted →

Actual ↑↓

| | $C_0$ | $C_1$ | |
|---|---|---|---|
| $C_0$ | TP | FN | P |
| $C_1$ | FP | TN | |

Precision or Positive Predictive Value:

$$Pr = \frac{TP}{D} = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FP}}$$

True positive rate, known as Sensitivity or Recall:

$$TPR = S_n = Re = \frac{TP}{P} = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FN}}$$

- **Precision** = Out of all the people classified Chinese, how many are actually Chinese? (Or classified correctly)

- **Recall** = Out of all the people that are actually Chinese, how many are classified as Chinese?

## Precision - Recall

Predicted

Actual

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP | FN |
| $C_1$ | FP | TN |

- Precision = Of the people classified Chinese, how many are actually Chinese?
- Number of people classified Chinese = TP+ FP
- Number of people actually Chinese (when classified as Chinese) = TP

- Recall = Of the people that are actually Chinese, how many are classified as Chinese?
- Number of people actually Chinese = TP+ FN
- Number of people classified Chinese (when actually Chinese) = TP

## Precision - Recall

← **Predicted** →

**Actual** ↑↓

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP = 0 | FN = 10 |
| $C_1$ | FP = 0 | TN = 9,990 |

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{0}{0} \ (not\ difined) \qquad Recall = \frac{0}{0 + 10} = 0$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = 99{,}9\%$$

- Now, we redefine the classifier: **Our classifier is a simple program that only returns true when evaluating anyone.**

← **Predicted** →

**Actual** ↑↓

| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP = 10 | FN = 0 |
| $C_1$ | FP = 9,990 | TN = 0 |

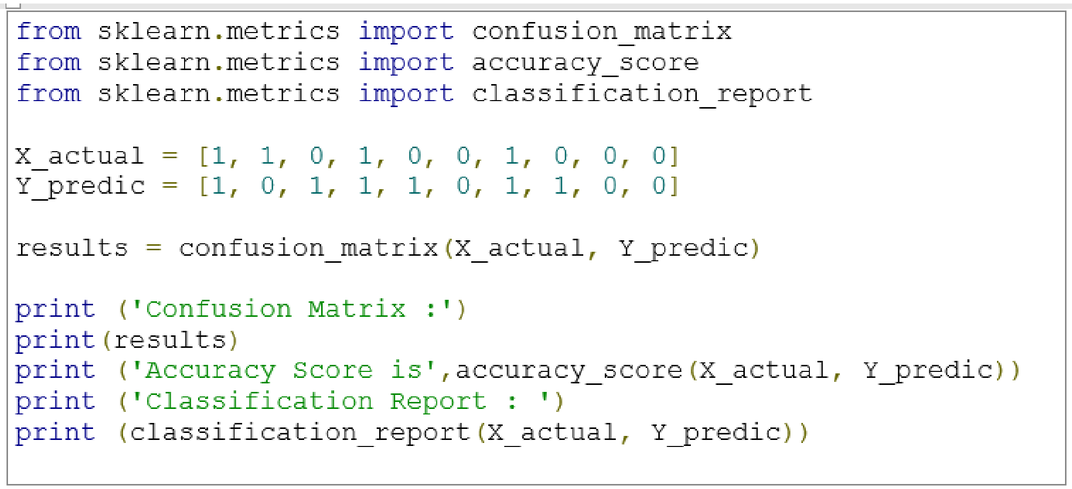
$$Precision = \frac{10}{10 + 9{,}990} = 0{,}001 \qquad Recall = \frac{10}{10 + 0} = 1$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = 0.1\%$$

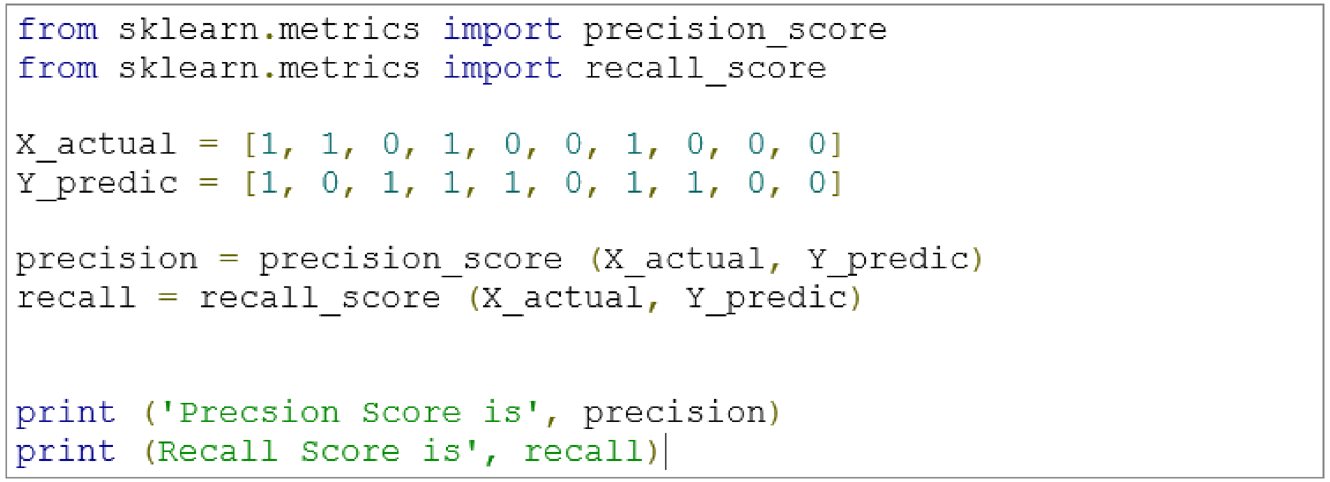
Gidia Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

## Precision - Recall



```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

results = confusion_matrix(X_actual, Y_predic)

print ('Confusion Matrix :')
print(results)
print ('Accuracy Score is',accuracy_score(X_actual, Y_predic))
print ('Classification Report : ')
print (classification_report(X_actual, Y_predic))
```



```python
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

precision = precision_score (X_actual, Y_predic)
recall = recall_score (X_actual, Y_predic)


print ('Precsion Score is', precision)
print (Recall Score is', recall)
```

Output

```
Confusion Matrix:
[
   [3 3]
   [1 3]
]

Accuracy Score is 0.6


Classification Report :
             precision     recall     f1-score      support
      0        0.75         0.50      0.60          6
      1        0.50         0.75      0.60          4
micro avg      0.60         0.60      0.60          10
macro avg      0.62         0.62      0.60          10
weighted avg   0.65         0.60      0.60          10
```

## F1-Score/-F$_\beta$-Score (F-Score/F-Measure)

- Comparing Systems

- System 1
- Precision: 70%
- **Recall: 60%**

- System 2
- **Precision: 80%**
- Recall: 50%

- Which of these two systems perform better?

- The answer to this question is the computing of a single measure from the precision and recall measure, which is called the F beta measure

$$F_\beta = \frac{1}{\beta \times \frac{1}{Precision} + (1-\beta) \times \frac{1}{Recall}}$$

$\beta$=0.5 = F-Measure

$$F_\beta = \frac{1}{0.5 \times \frac{1}{0.7} + (1-0.5) \times \frac{1}{0.6}} = 0.6461$$

$>$

$$F_\beta = \frac{1}{0.5 \times \frac{1}{0.8} + (1-0.5) \times \frac{1}{0.5}} = 0.6153$$

*Medical model*

$\beta$=0.95

$$F_\beta = \frac{1}{0.95 \times \frac{1}{0.7} + (1-0.95) \times \frac{1}{0.6}} = 0.6942$$

$<$

$$F_\beta = \frac{1}{0.95 \times \frac{1}{0.8} + (1-0.95) \times \frac{1}{0.5}} = 0.7766$$

*Spam model*

## Specificity

- From the confusion matrix, we can estimate two metrics: Recall and Specificity

**Predicted**

**Actual**

|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | TP | FN |
| $C_1$ | FP | TN |

- Specificity tell us what percentage of patients without diabetes were correctly identified
- Of all the people who are healthy, how many of those did we correctly predict?

True negative rate, known as Specificity:

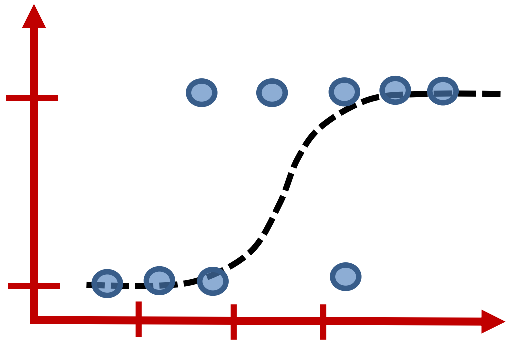
$$TNR = Sp = \frac{TN}{N} = \frac{\boxed{TN}}{\boxed{TN}+\boxed{FP}}$$

## Specificity

```
from sklearn.metrics import specificity_score

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]
Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

specificity = specificity_score (X_actual, Y_predic)


print ('Specificity Score is', specificity)
```

## Recall - Specificity

- Example



| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 127 | 29 |
| $C_1$ | 12 | 101 |

- $C_0$= Has Diabetes
- $C_1$= Does Not Have Diabetes

True positive rate, known as Sensitivity or Recall:

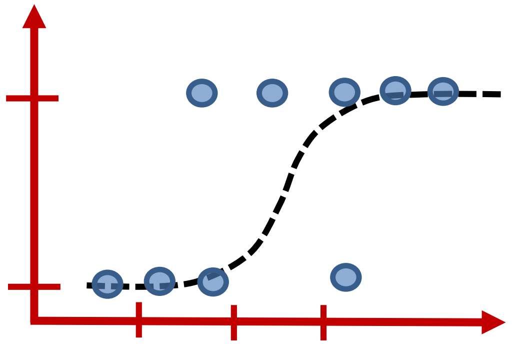
$$TPR = S_n = Re = \frac{TP}{P} = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FN}} = \frac{127}{127+28} = 0.81$$

- Recall tell us that 81% of the patients with diabetes were correctly identified

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Recall - Specificity

- Example



|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 127 | 29 |
| $C_1$ | 12 | 101 |

True negative rate, known as Specificity:

$$TNR = Sp = \frac{TN}{N} = \frac{\boxed{TN}}{\boxed{TN} + \boxed{FP}} = \frac{101}{101 + 12} = 0.89$$

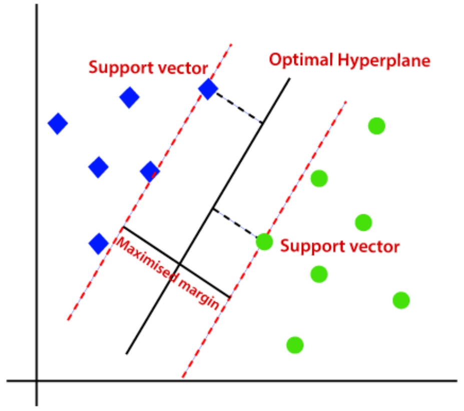
- Specificity tell us that 89% of the patients without diabetes were correctly identified

Gidia Grupo de I+D en Inteligencia Artificial

Universidad Nacional de Colombia

## Recall - Specificity

- Example



| | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |

True positive rate, known as Sensitivity or Recall:

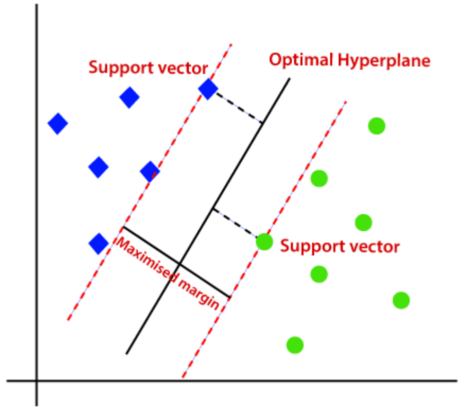
$$TPR = S_n = Re = \frac{TP}{P} = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FN}} = \frac{130}{130+29} = 0.83$$

- Recall tell us that 83% of the patients with diabetes were correctly identified

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Recall - Specificity

- Example



|  | $C_0$ | $C_1$ |
|---|---|---|
| $C_0$ | 130 | 27 |
| $C_1$ | 14 | 98 |

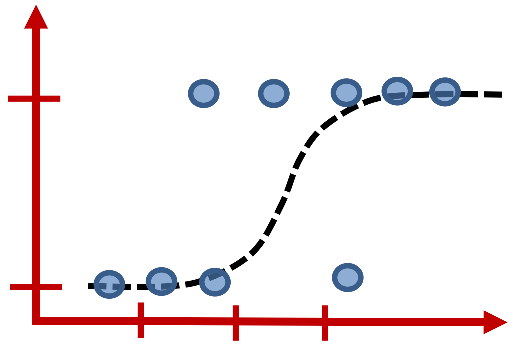
True negative rate, known as Specificity:

$$TNR = Sp = \frac{TN}{N} = \frac{\boxed{TN}}{\boxed{TN}+\boxed{FP}} = \frac{98}{98+14} = 0.87$$

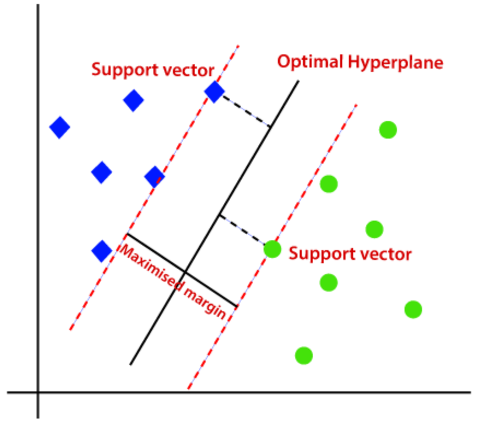
- Specificity tell us that 87% of the patients without diabetes were correctly identified

## Recall - Specificity

- Example



- Recall = 0.81
- Specificity = 0.89

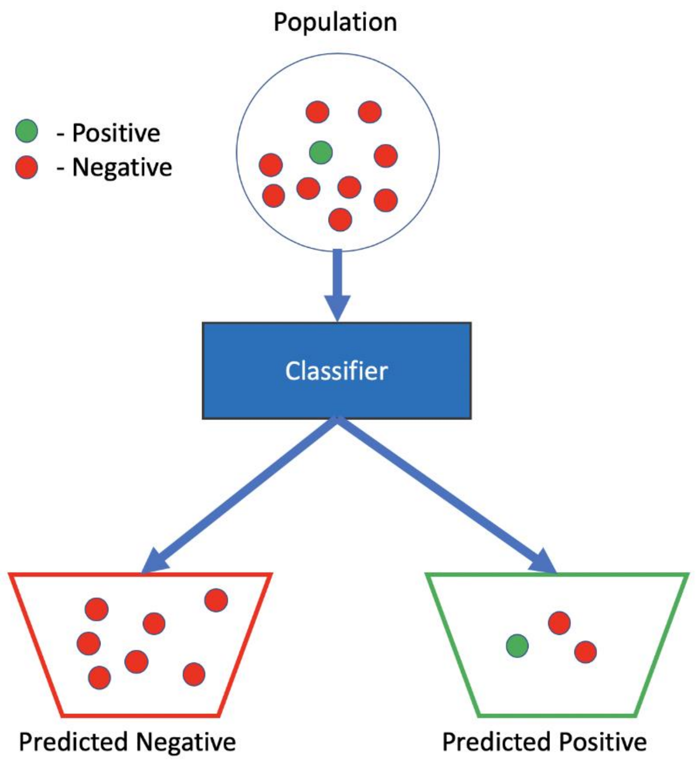


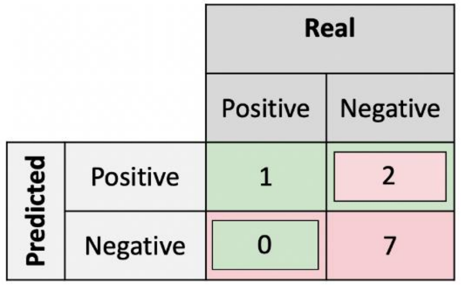
- Recall = 0.83
- Specificity = 0.87

- What to choose?
- We choose the logistic regression model, because correctly identifying patients without diabetes; this is the case if is more important than identifying patients with diabetes.
- We choose the SVM model, because correctly identifying patients with diabetes; this is the case if is more important than identifying patients without diabetes.

## Recall, Precision, Specificity

**Example 1 — Low Precision, High Recall, and High Specificity**





| | | Real | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | 1 | 2 |
| | Negative | 0 | 7 |

$$\boldsymbol{precision} = \frac{tp}{tp+fp} = \frac{1}{3} = \mathbf{33\%}$$
$$\boldsymbol{recall} = \frac{tp}{tp+fn} = \frac{1}{1} = \mathbf{100\%}$$
$$\boldsymbol{specificity} = \frac{tn}{tn+fp} = \frac{7}{9} = \mathbf{78\%}$$
$$\boldsymbol{sensitivity} = recall = \mathbf{100\%}$$

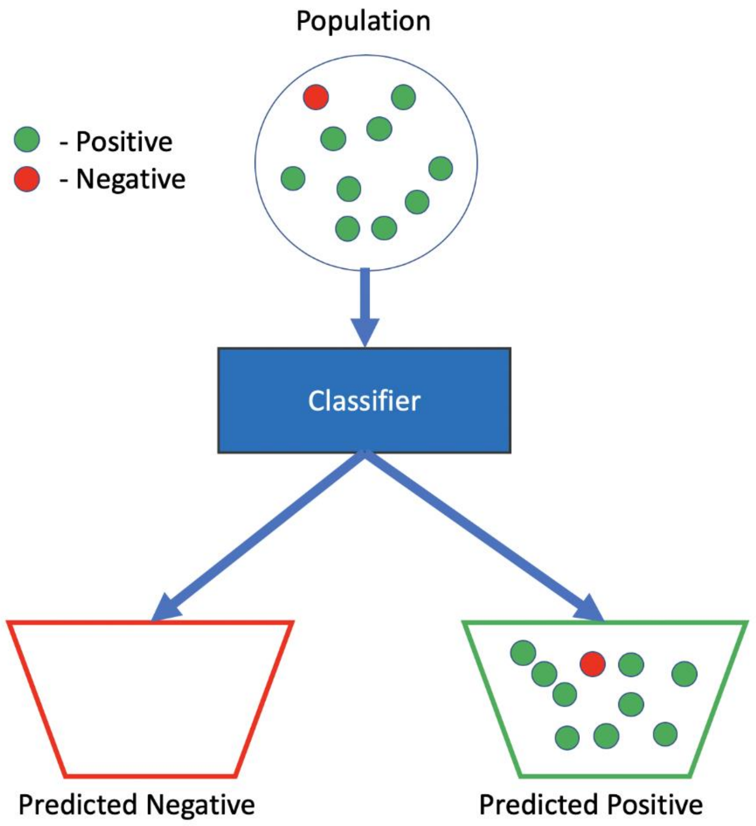
taken from
[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

Gidia Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

## Recall, Precision, Specificity

**Example 2 — High Precision, High Recall, and Low Specificity**



|  |  | **Real** |  |
|---|---|---|---|
|  |  | Positive | Negative |
| **Predicted** | Positive | 9 | 1 |
|  | Negative | 0 | 0 |

$$\textbf{\textit{precision}} = \frac{tp}{tp+fp} = \frac{9}{10} = \mathbf{90\%}$$

$$\textbf{\textit{recall}} = \frac{tp}{tp+fn} = \frac{9}{9} = \mathbf{100\%}$$

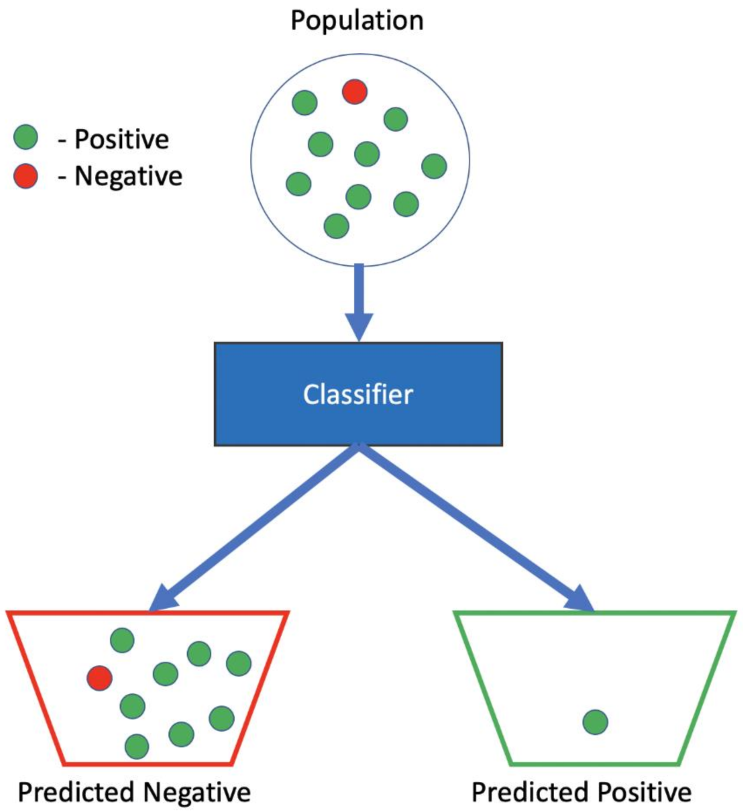
$$\textbf{\textit{specificity}} = \frac{tn}{tn+fp} = \frac{0}{1} = \mathbf{0\%}$$

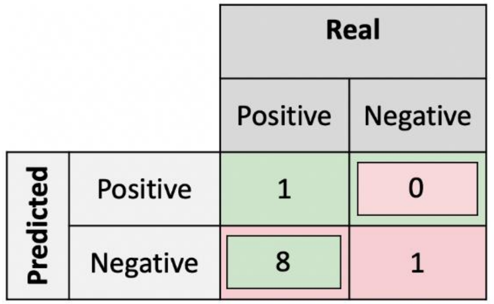
$$\textbf{\textit{sensitivity}} = recall = \mathbf{100\%}$$

taken from
[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

## Recall, Precision, Specificity

**Example 3 — High Precision, Low Recall, and High Specificity**





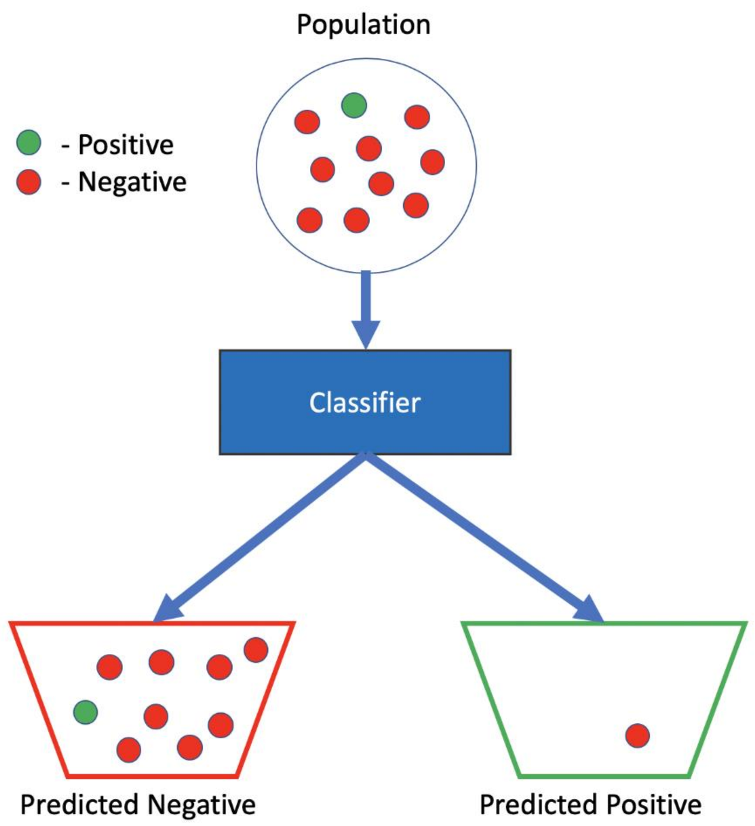
| | | Real | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | 1 | 0 |
| | Negative | 8 | 1 |

$$\boldsymbol{precision} = \frac{tp}{tp+fp} = \frac{1}{1} = \mathbf{100\%}$$
$$\boldsymbol{recall} = \frac{tp}{tp+fn} = \frac{1}{9} = \mathbf{11\%}$$
$$\boldsymbol{specificity} = \frac{tn}{tn+fp} = \frac{1}{1} = \mathbf{100\%}$$
$$\boldsymbol{sensitivity} = recall = \mathbf{11\%}$$

taken from
[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

## Recall, Precision, Specificity

**Example 4 — Low Precision, Low Recall, and High Specificity**



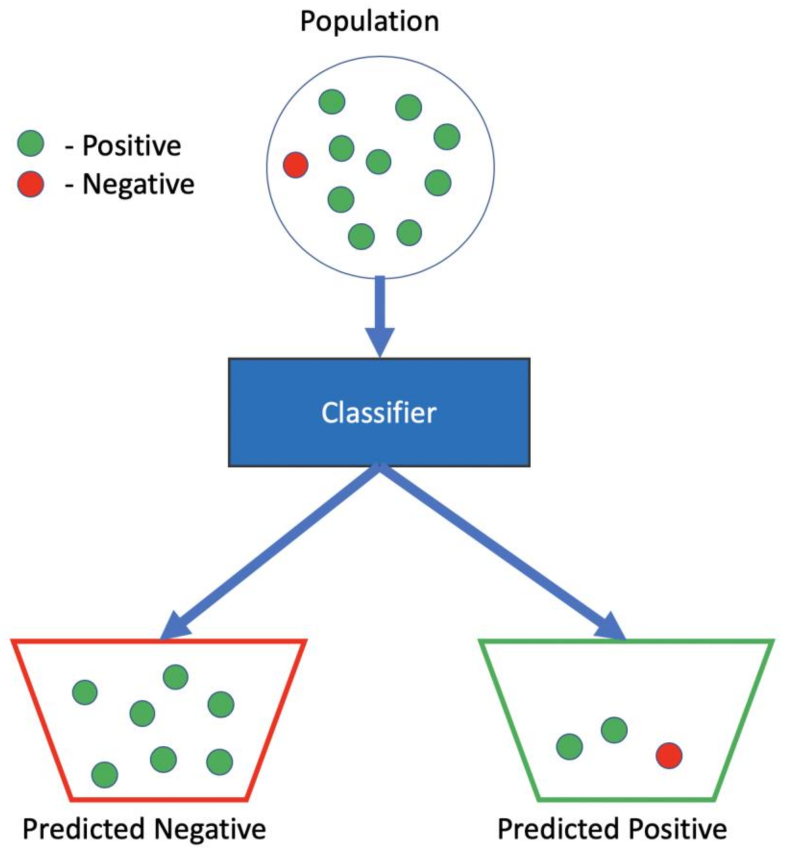
|  |  | Real |  |
|---|---|---|---|
|  |  | Positive | Negative |
| Predicted | Positive | 0 | 1 |
|  | Negative | 1 | 8 |

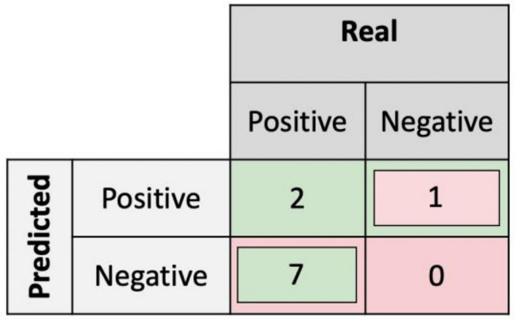
$$precision = \frac{tp}{tp + fp} = \frac{0}{1} = \mathbf{0\%}$$
$$recall = \frac{tp}{tp + fn} = \frac{0}{1} = \mathbf{0\%}$$
$$specificity = \frac{tn}{tn + fp} = \frac{8}{9} = \mathbf{89\%}$$
$$sensitivity = recall = \mathbf{0\%}$$

taken from
[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

## Recall, Precision, Specificity

**Example 5 — High Precision, Low Recall, and Low Specificity**





| | | Real | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | 2 | 1 |
| | Negative | 7 | 0 |

$$\boldsymbol{precision} = \frac{tp}{tp+fp} = \frac{2}{3} = \mathbf{66\%}$$

$$\boldsymbol{recall} = \frac{tp}{tp+fn} = \frac{2}{9} = \mathbf{22\%}$$

$$\boldsymbol{specificity} = \frac{tn}{tn+fp} = \frac{0}{1} = \mathbf{0\%}$$

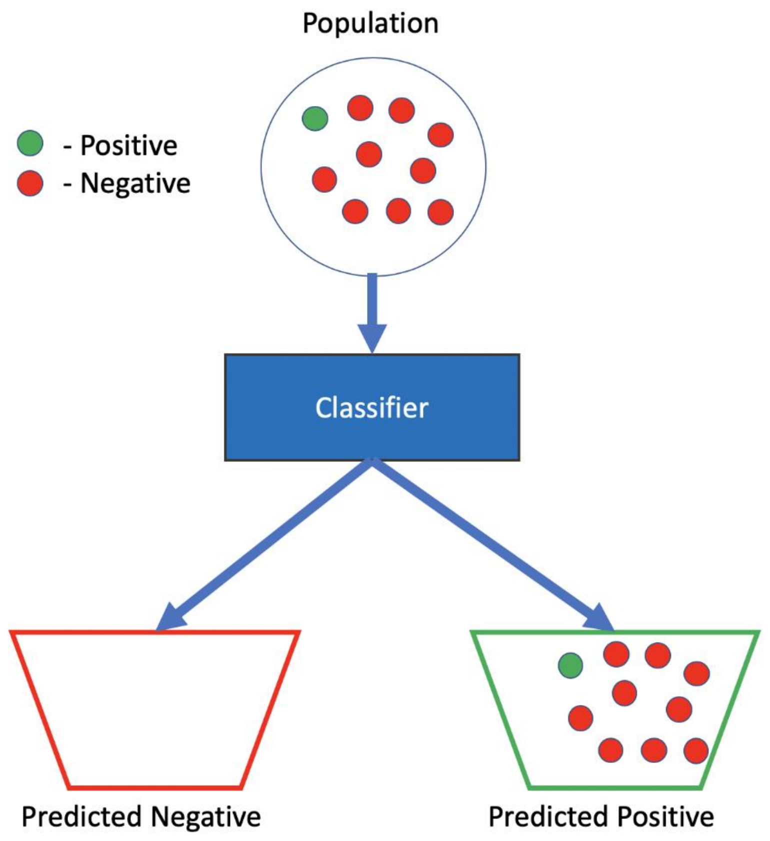
$$\boldsymbol{sensitivity} = recall = \mathbf{22\%}$$

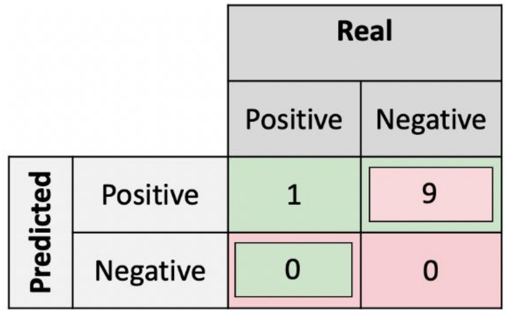
taken from

[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

## Recall, Precision, Specificity

**Example 6 — Low Precision, High Recall, and Low Specificity**





| | | Real | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | 1 | 9 |
| | Negative | 0 | 0 |

$$\boldsymbol{precision} = \frac{tp}{tp+fp} = \frac{1}{10} = \mathbf{10\%}$$

$$\boldsymbol{recall} = \frac{tp}{tp+fn} = \frac{1}{1} = \mathbf{100\%}$$

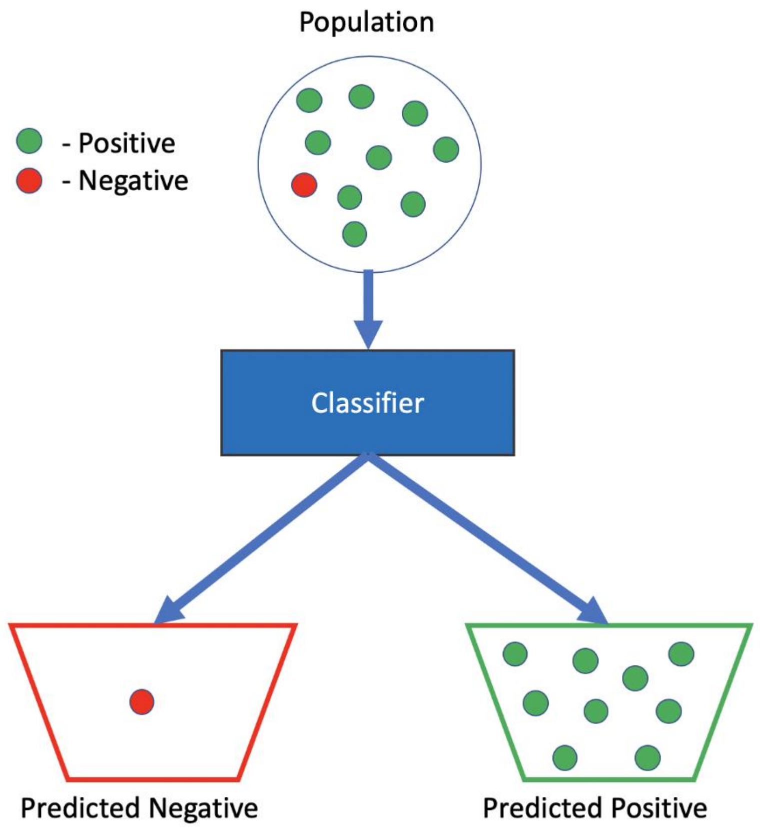
$$\boldsymbol{specificity} = \frac{tn}{tn+fp} = \frac{0}{9} = \mathbf{0\%}$$

$$\boldsymbol{sensitivity} = recall = \mathbf{100\%}$$

taken from
[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

## Recall, Precision, Specificity

**Example 7 — High Precision, High Recall, and High Specificity**



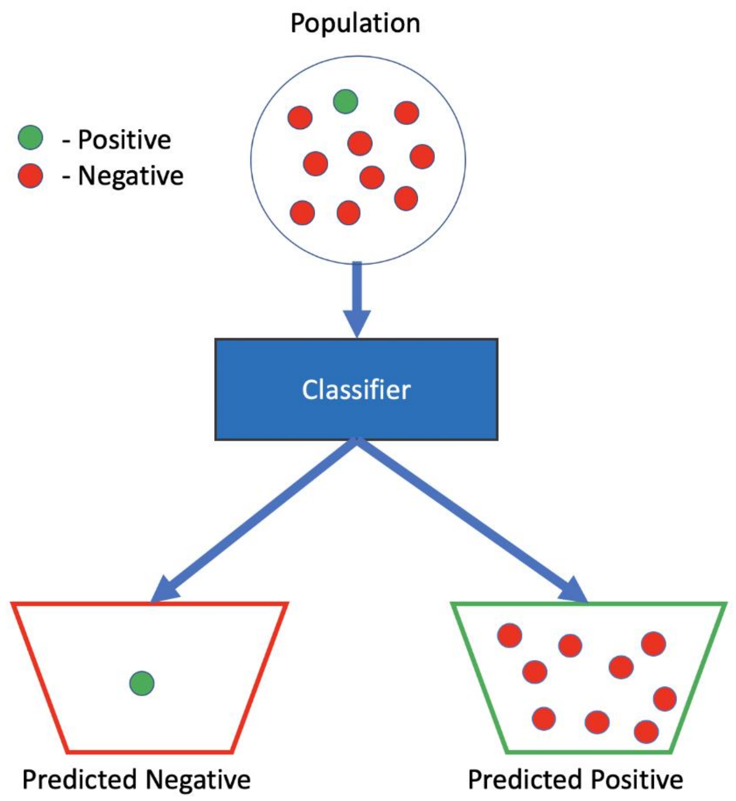
| | | Real | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | 9 | 0 |
| | Negative | 0 | 1 |

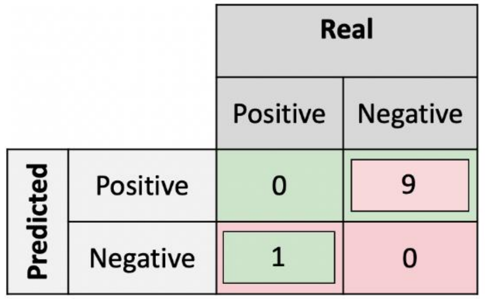
$$\boldsymbol{precision} = \frac{tp}{tp+fp} = \frac{9}{9} = \mathbf{100\%}$$
$$\boldsymbol{recall} = \frac{tp}{tp+fn} = \frac{9}{9} = \mathbf{100\%}$$
$$\boldsymbol{specificity} = \frac{tn}{tn+fp} = \frac{1}{1} = \mathbf{100\%}$$
$$\boldsymbol{sensitivity} = recall = \mathbf{100\%}$$

taken from
Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium

## Recall, Precision, Specificity

**Example 8 — Low Precision, Low Recall, and Low Specificity**





| | | Real | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | 0 | 9 |
| | Negative | 1 | 0 |

$$\textbf{\textit{precision}} = \frac{tp}{tp+fp} = \frac{0}{9} = \mathbf{0\%}$$

$$\textbf{\textit{recall}} = \frac{tp}{tp+fn} = \frac{0}{1} = \mathbf{0\%}$$

$$\textbf{\textit{specificity}} = \frac{tn}{tn+fp} = \frac{0}{9} = \mathbf{0\%}$$

$$\textbf{\textit{sensitivity}} = recall = \mathbf{0\%}$$

taken from
[Data Science in Medicine — Precision & Recall or Specificity & Sensitivity? | by Alon Lekhtman | Medium]

## Summary

- **Accuracy is a great measure** but only **when you have symmetric datasets** (false negatives & false positives counts are close), also, **false negatives & false positives have similar costs.**

- If the cost of false positives and false negatives are different then F1 is your option. **F1 is best if you have an uneven class distribution**.

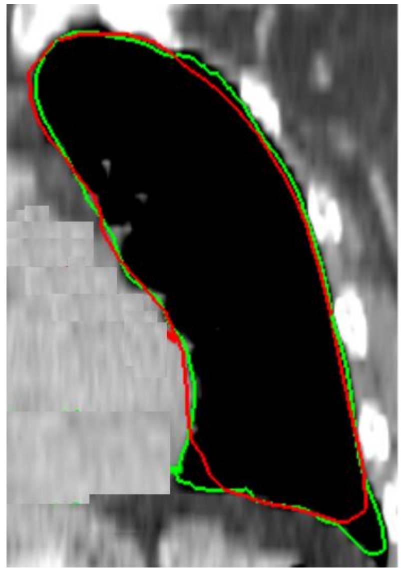
- **Choose Recall** if the idea of false positives is far better than false negatives.

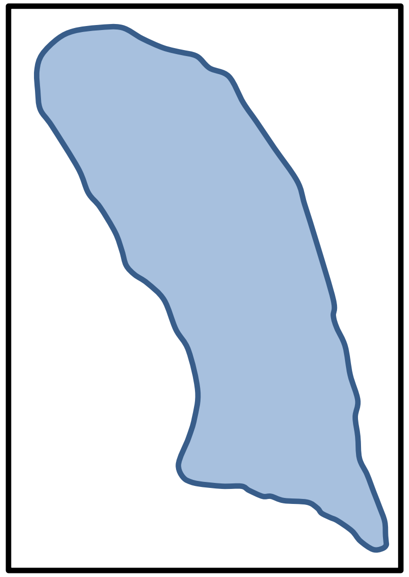
- **Choose precision if you want to be more confident of your true positives**. for example, Spam emails.

- **Choose Specificity if you want to cover all true negatives**, meaning you don't want any false alarms, you don't want any false positives.

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

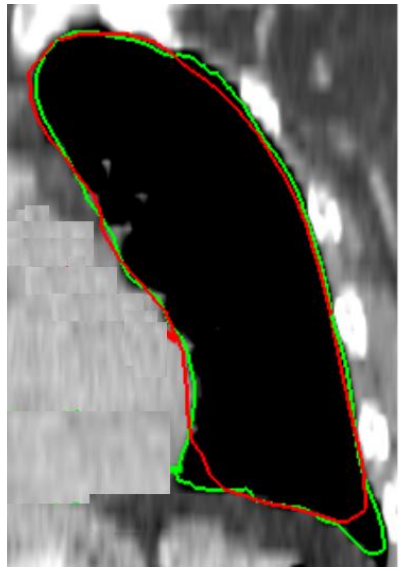
## Example

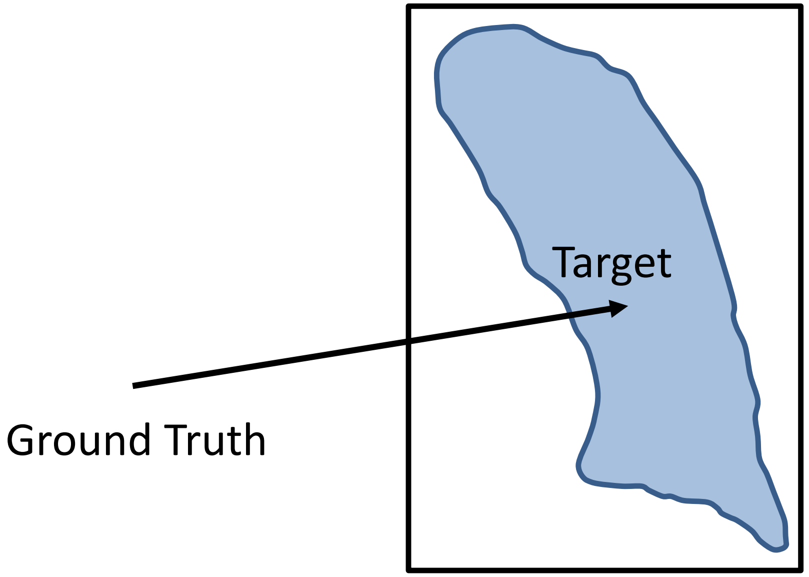




● Class 1

## Example

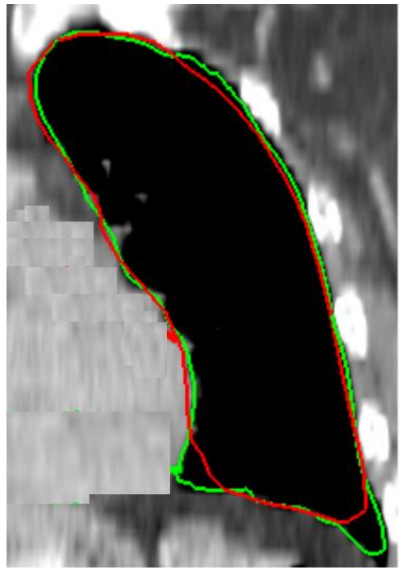


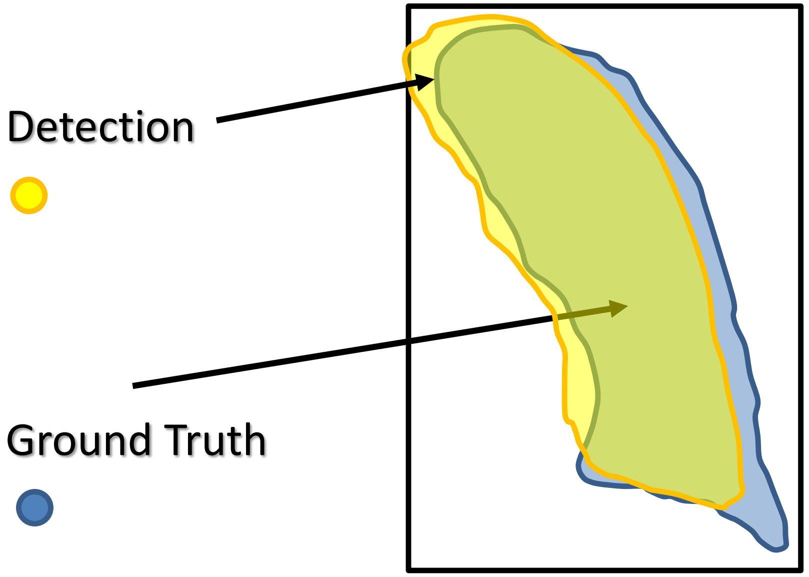


● Class 1 or Target

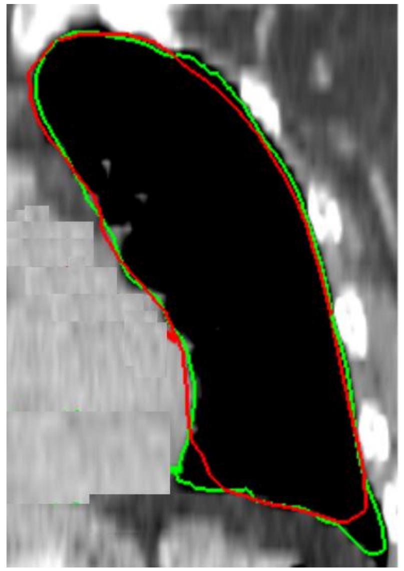
Blue pixels: Positive instances (P)
White pixels: Negative instances (N)

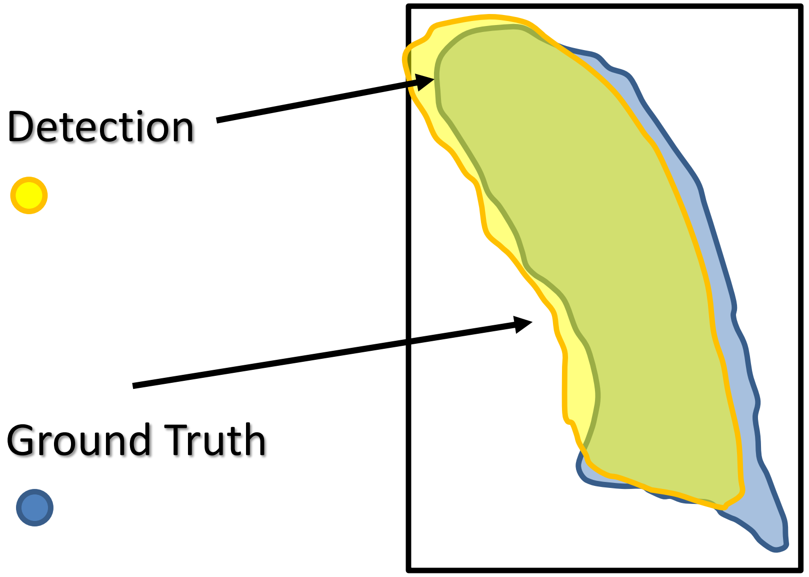
## Example





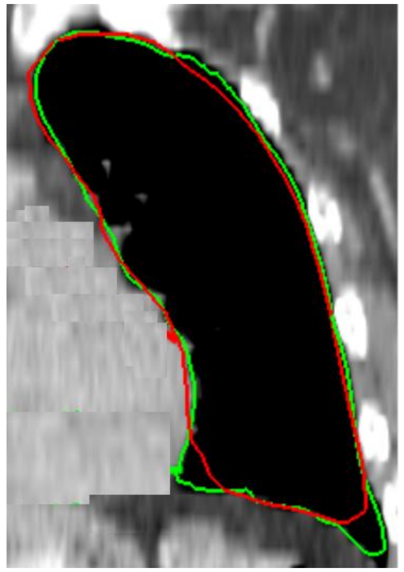
## Example





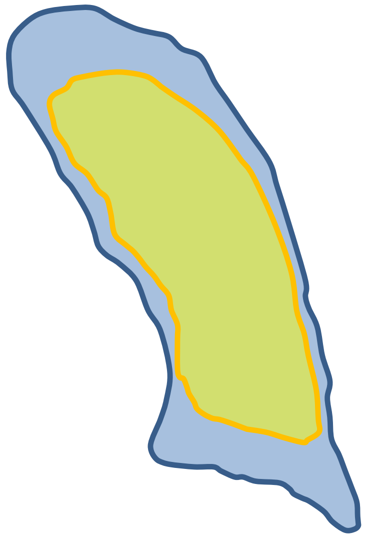
True positives (TP) : ●
False positives (FP) : ●
False negatives (FN) : ●
True negatives (TN) : ○

## Example

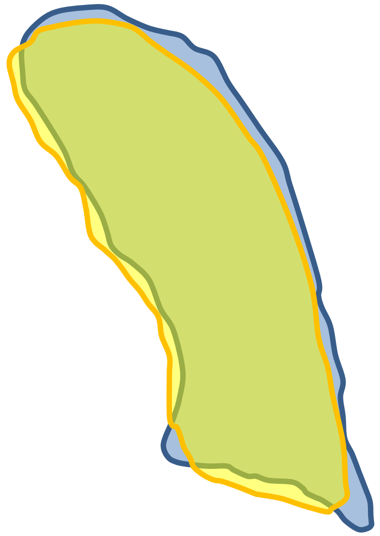


IDEAL
TPR = 100%
FPR = 0%

Extreme
No false positive

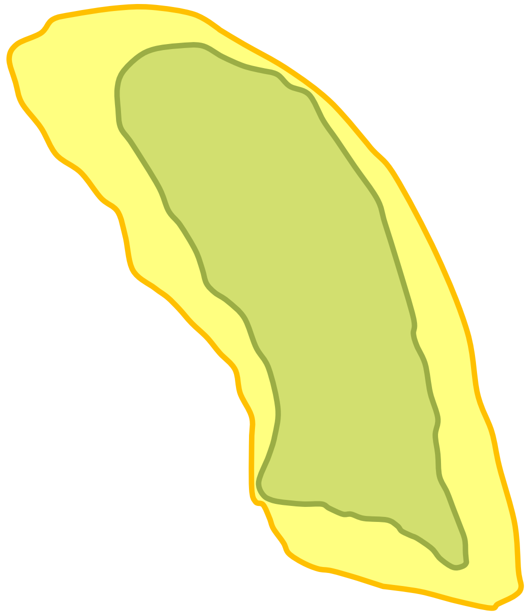


No false alarm



Reality:
Trade-off between
FPR and TPR

Extreme
All positive samples
are detected



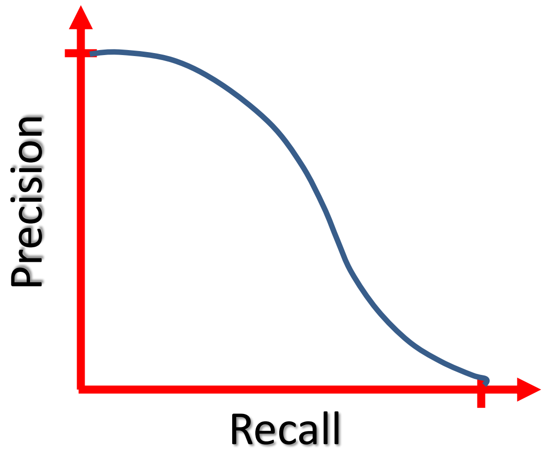
All targets detected

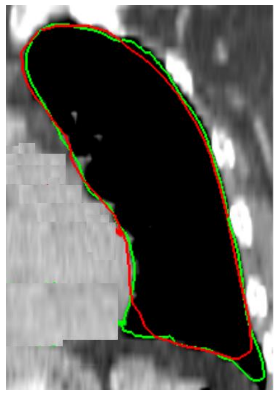
Gidia Grupo de I+D en Inteligencia Artificial

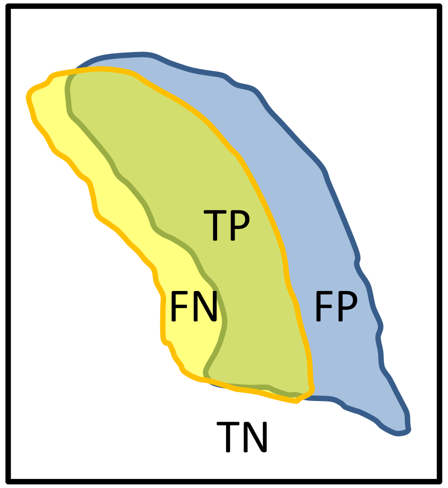
UNIVERSIDAD NACIONAL DE COLOMBIA

# Precision-Recall Curve

- The precision and recall metrics are related so that if you train your classifier to increase accuracy, it will decrease recall and vice versa.
- The PR curve is the result of drawing the graph between precision and recall. This graph allows us to see from which recall we have a precision degradation and vice versa.
- Ideally, a curve that is as close as possible to the upper right corner (high precision and high recall)







| | |
|---|---|
| True positives (TP) : | ● |
| False positives (FP) : | ● |
| False negatives (FN) : | ● |
| True negatives (TN) : | ○ |

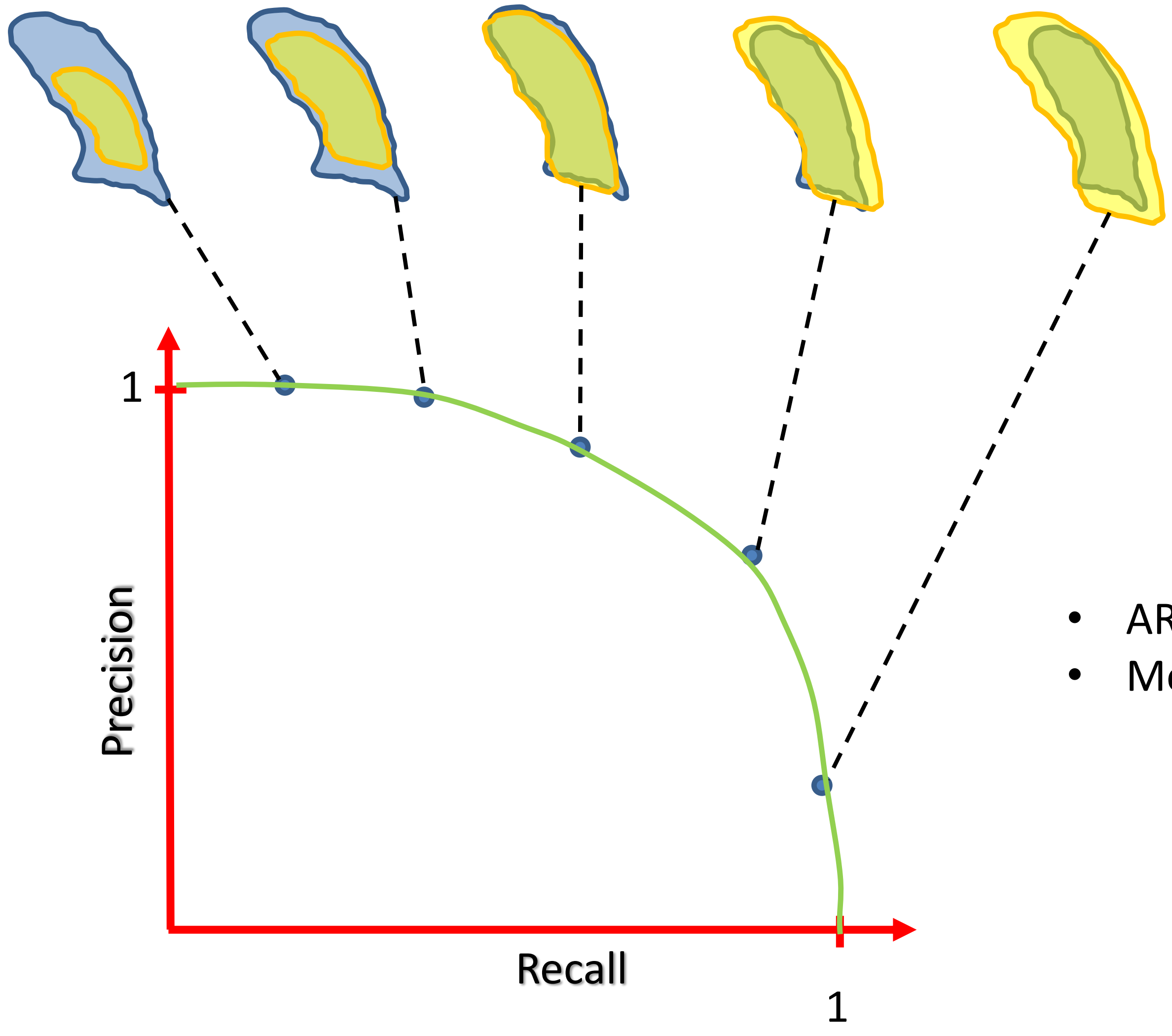
$$Pr = \frac{TP}{D} = \frac{TP}{TP+FP}$$

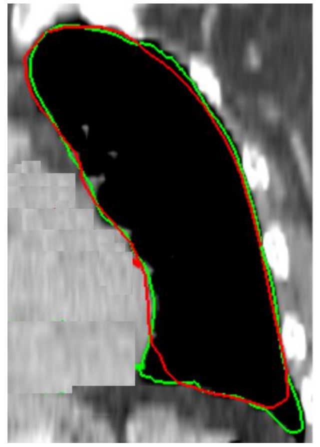
$$Re = \frac{TP}{P} = \frac{TP}{TP+FN}$$

## Precision-Recall Curve

$$Pr = \frac{TP}{D} = \frac{\boxed{TP}}{\boxed{TP} + \boxed{FP}}$$

$$Re = \frac{TP}{P} = \frac{\boxed{TP}}{\boxed{TP} + \boxed{FN}}$$





- AREA UNDER CURVE – AUC:
- Mean Precision Average (Integral)

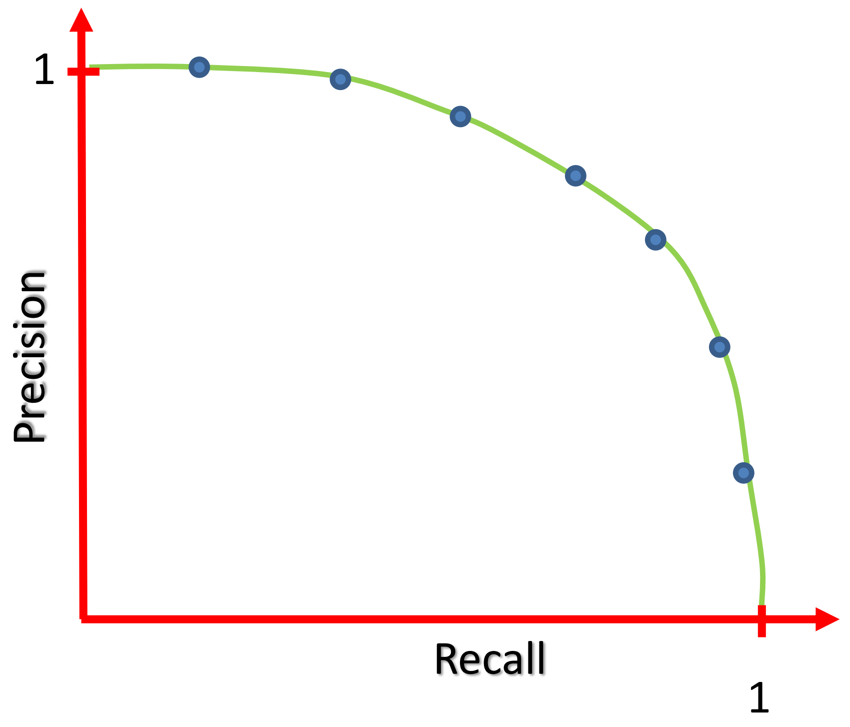
Gidia Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

# Precision-Recall Curve

$$Pr = \frac{TP}{D} = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FP}}$$

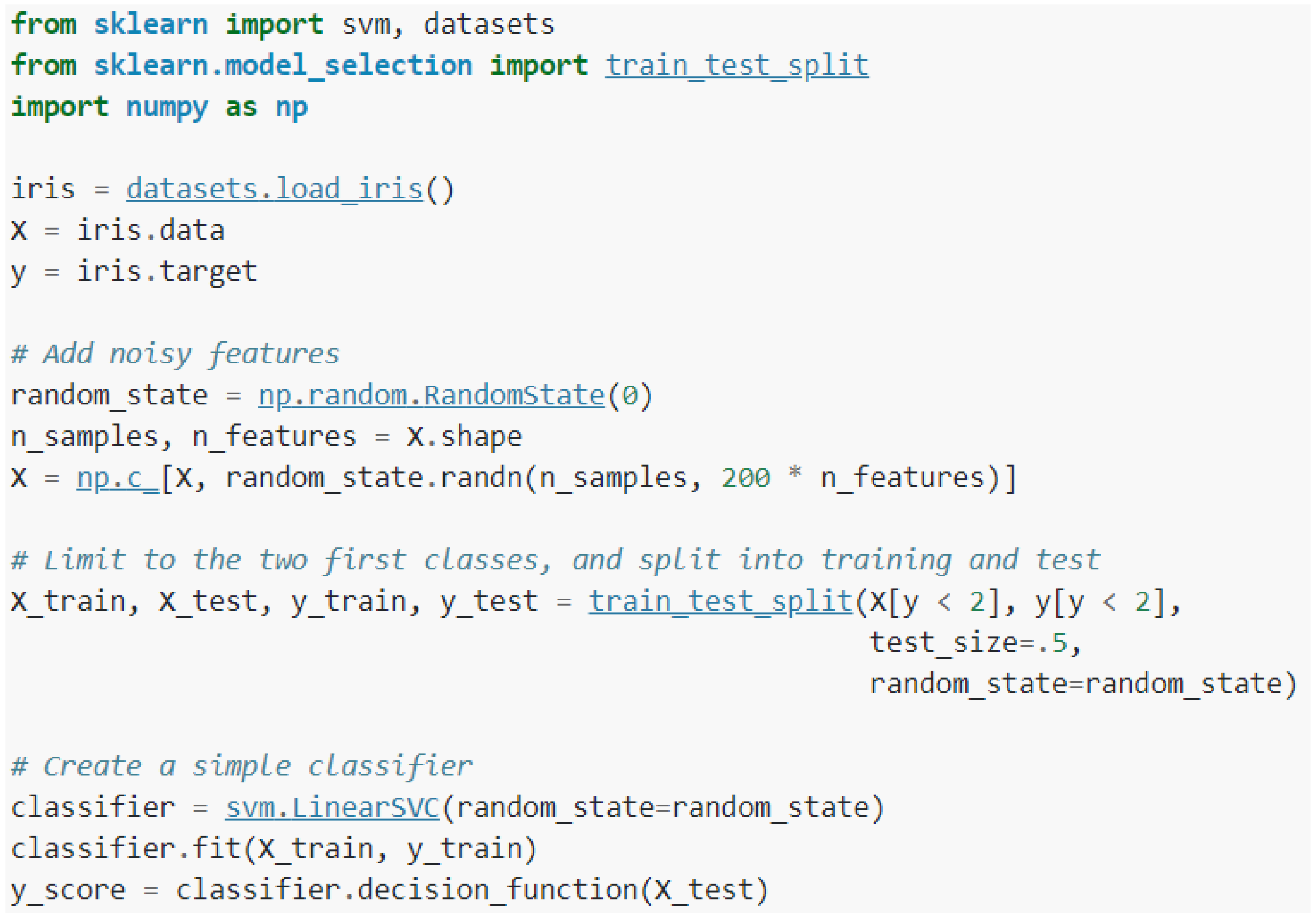
$$Re = \frac{TP}{P} = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FN}}$$



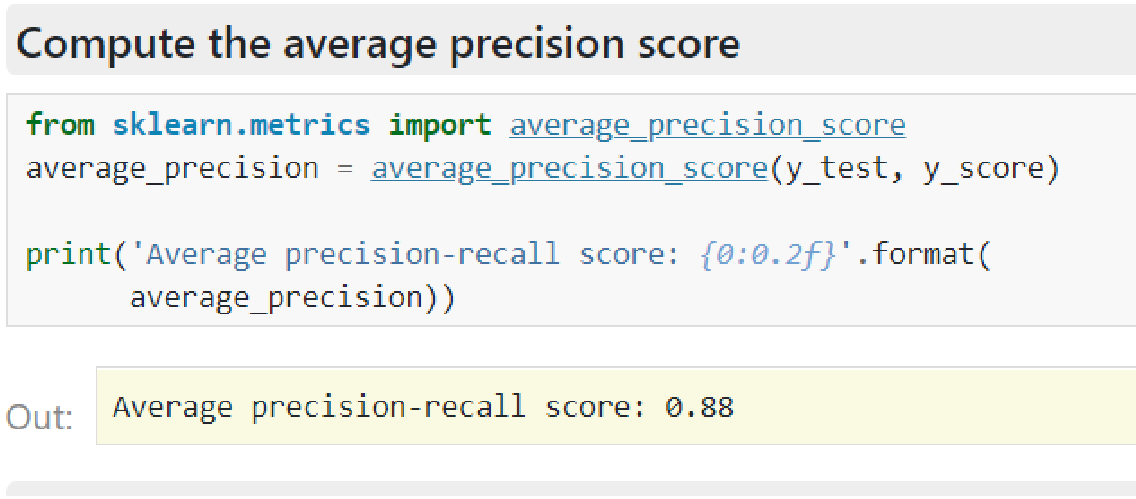
| Threshold |
|---|
| 0.2 |
| 0.3 |
| 0.4 |
| 0.5 |
| 0.7 |
| 0.8 |
| 0.9 |

```
[[ 305,  102],
 [ 393, 1200]]

[[ 263,  144],
 [ 210, 1383]]

[[ 216,  191],
 [ 107, 1486]]

[[ 184,  223],
 [  50, 1543]]

[[ 109,  298],
 [  10, 1583]]

[[  63,  344],
 [   2, 1591]]

[[  20,  387],
 [   0, 1593]]
```

## Precision-Recall Curve



```python
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
import numpy as np

iris = datasets.load_iris()
X = iris.data
y = iris.target

# Add noisy features
random_state = np.random.RandomState(0)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

# Limit to the two first classes, and split into training and test
X_train, X_test, y_train, y_test = train_test_split(X[y < 2], y[y < 2],
                                                    test_size=.5,
                                                    random_state=random_state)

# Create a simple classifier
classifier = svm.LinearSVC(random_state=random_state)
classifier.fit(X_train, y_train)
y_score = classifier.decision_function(X_test)
```



### Compute the average precision score

```python
from sklearn.metrics import average_precision_score
average_precision = average_precision_score(y_test, y_score)

print('Average precision-recall score: {0:0.2f}'.format(
      average_precision))
```

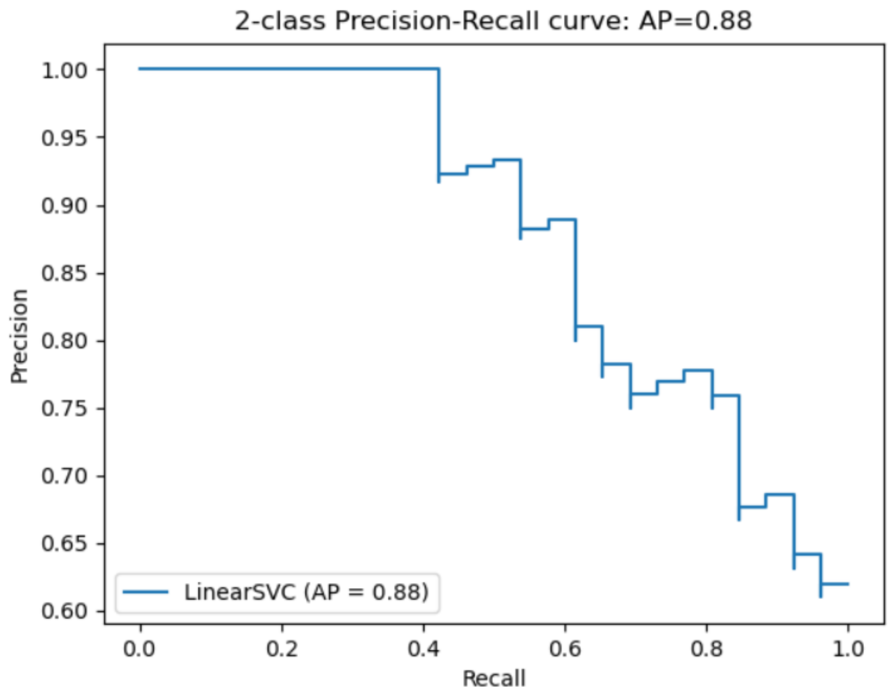
Out:

```
Average precision-recall score: 0.88
```

## Precision-Recall Curve

Plot the Precision-Recall curve

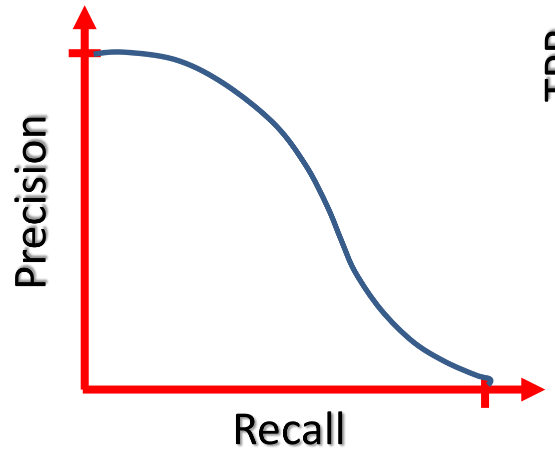
```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve
import matplotlib.pyplot as plt

disp = plot_precision_recall_curve(classifier, X_test, y_test)
disp.ax_.set_title('2-class Precision-Recall curve: '
                   'AP={0:0.2f}'.format(average_precision))
```

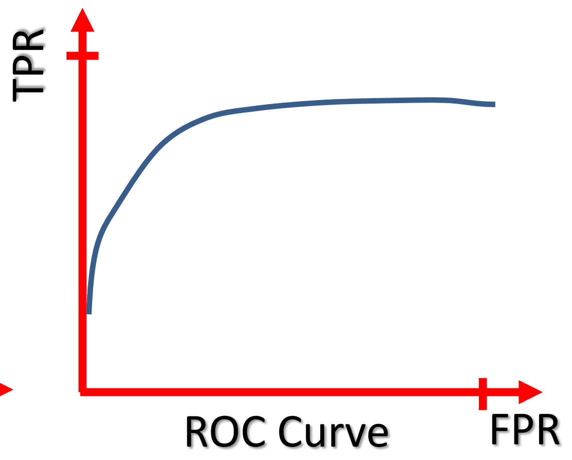


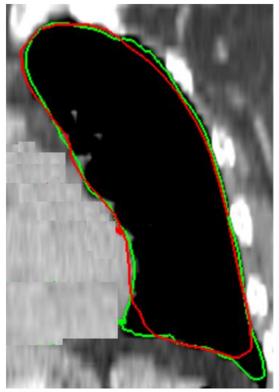
Out:
```
Text(0.5, 1.0, '2-class Precision-Recall curve: AP=0.88')
```

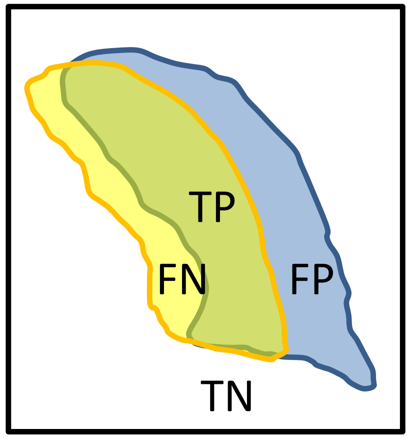
## ROC Curve – Receiver Operation Characteristic

- The ROC (receiver operating characteristic) curve is similar to the PR curve but changing some values.
- Relate the recall to the false positive rate. In other words, it relates the sensitivity of our model to optimistic failures (classify negatives as positive).
- It makes sense since, generally, if we increase the recall, our model will tend to be more optimistic and will introduce more false positives in the classification.









True positives (TP) : 🟢
False positives (FP) : 🟡
False negatives (FN) : 🔵
True negatives (TN) : ○

$$Pr = \frac{TP}{D} = \frac{TP}{TP+FP}$$

$$Re = \frac{TP}{P} = \frac{TP}{TP+FN}$$

$$FPR = 1 - Sp = \frac{FP}{N} = \frac{FP}{TN+FP}$$

$$TPR = S_n = Re = \frac{TP}{P} = \frac{TP}{TP+FN}$$

- Sensitivity = TPR y Specificity = 1 - FPR

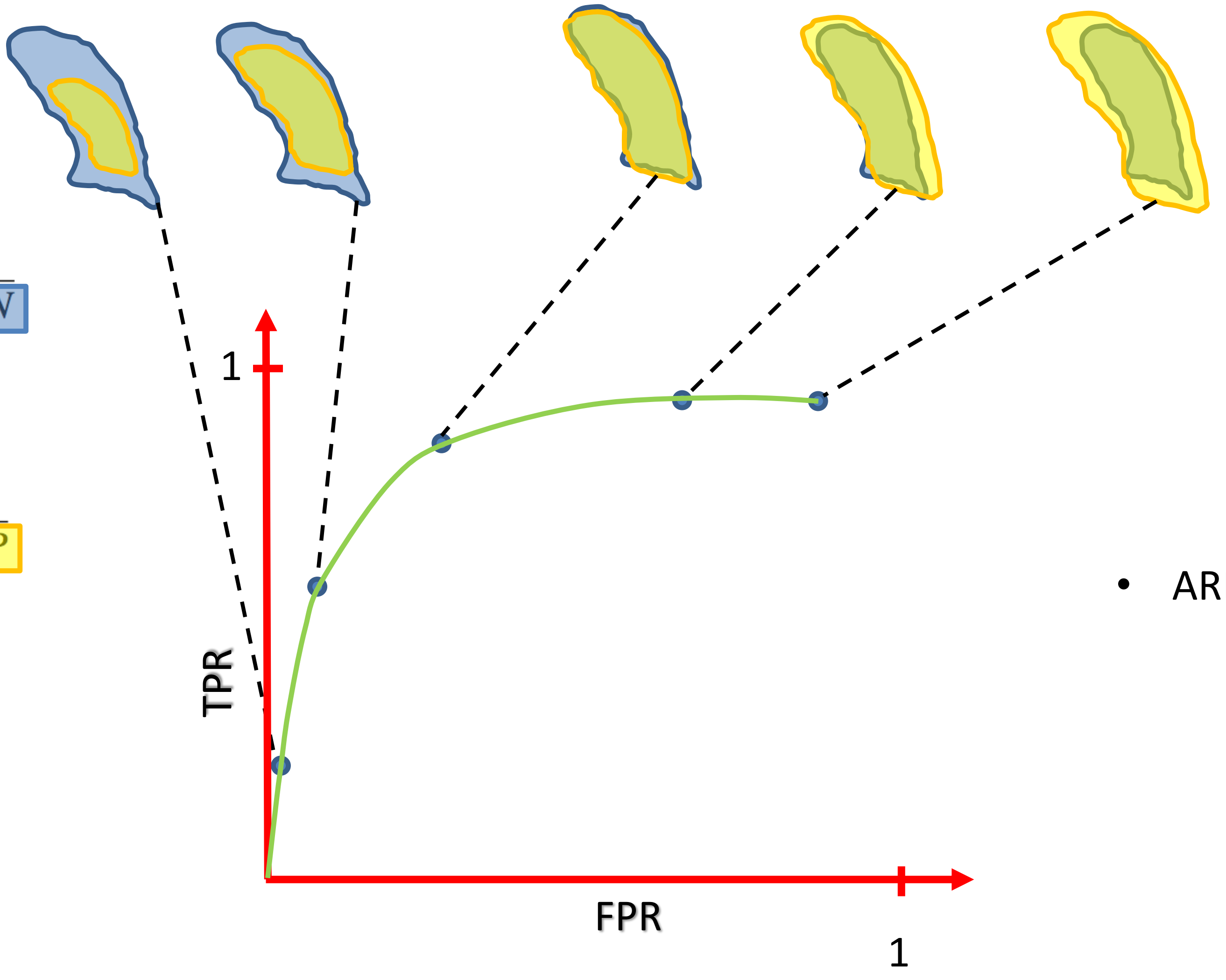
Gidia Grupo de I+D en Inteligencia Artificial

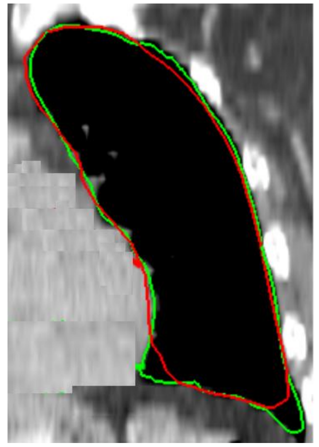
UNIVERSIDAD NACIONAL DE COLOMBIA

# ROC Curve – Receiver Operation Characteristic

$$TPR = \frac{\boxed{TP}}{\boxed{TP}+\boxed{FN}}$$

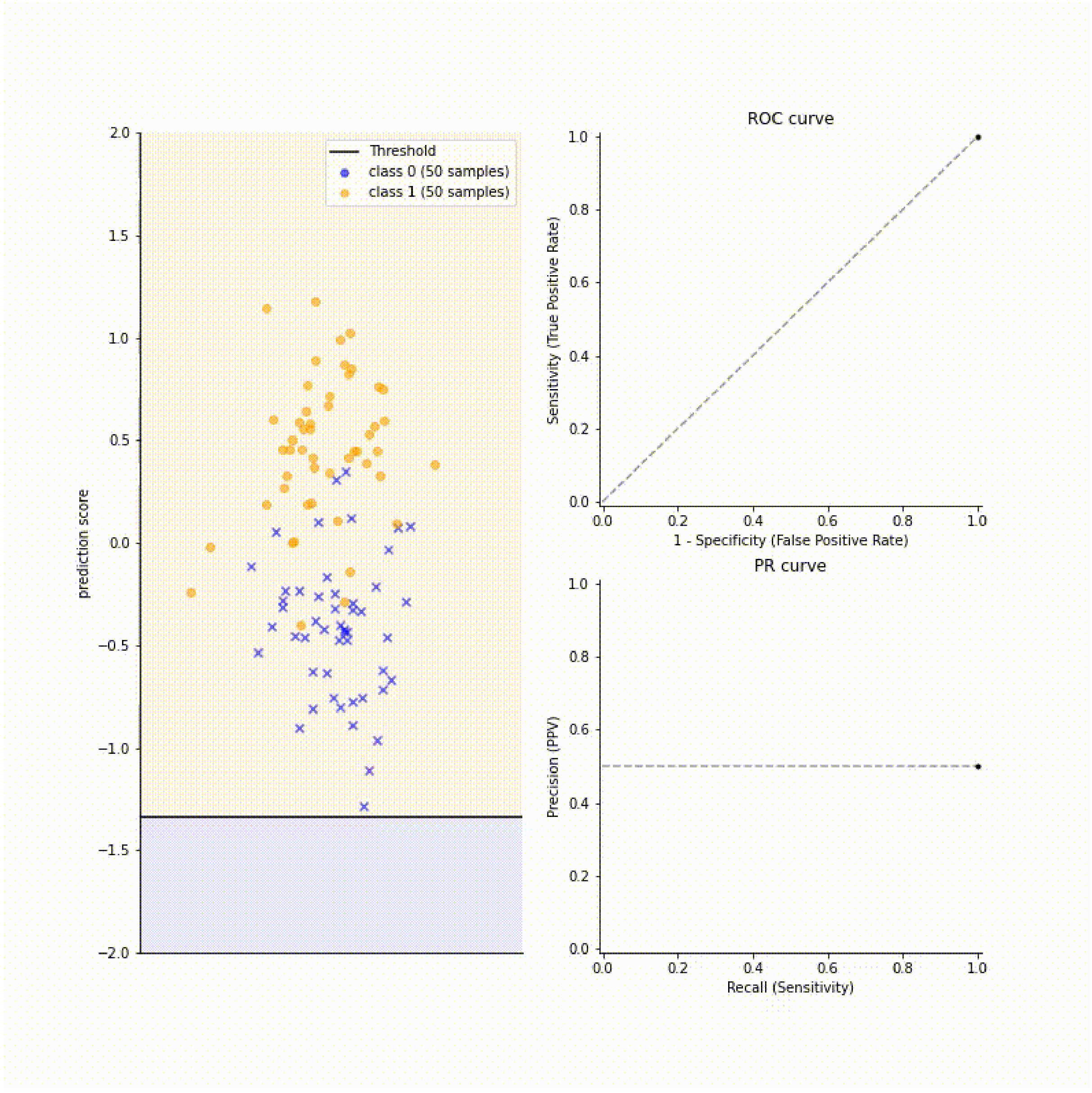
$$FPR = \frac{\boxed{FP}}{\boxed{TN}+\boxed{FP}}$$





- AREA UNDER CURVE – AUC

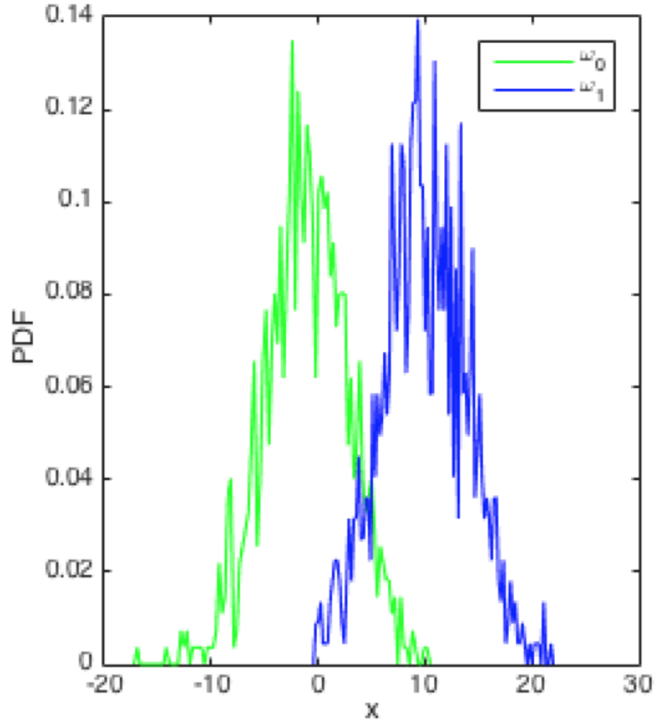
## Precision-Recall Curve and ROC Curve

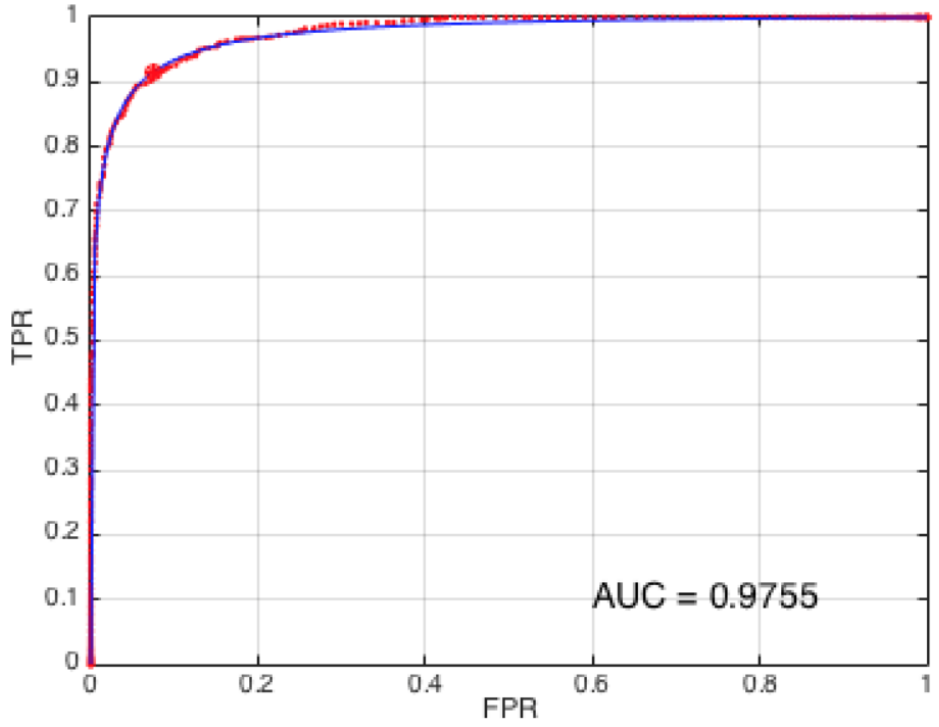


Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

## ROC Curve Example

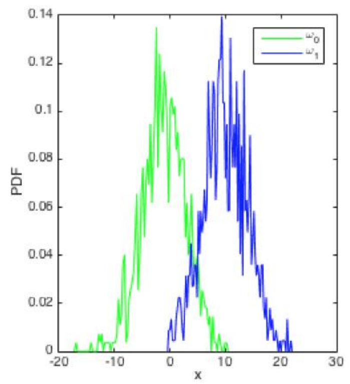


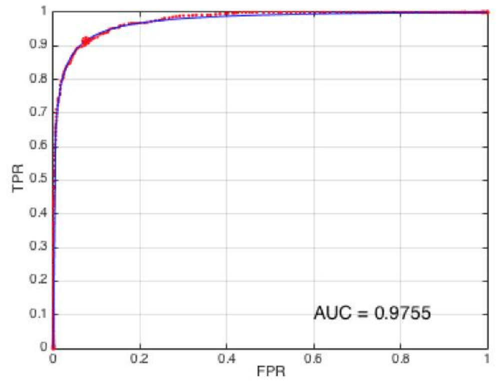


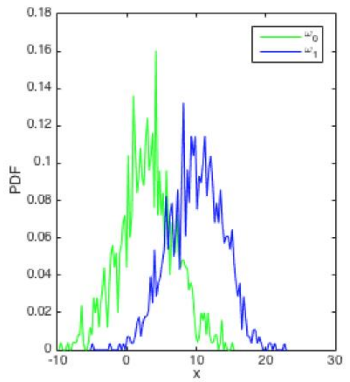
Refe: Prof. D. Mery

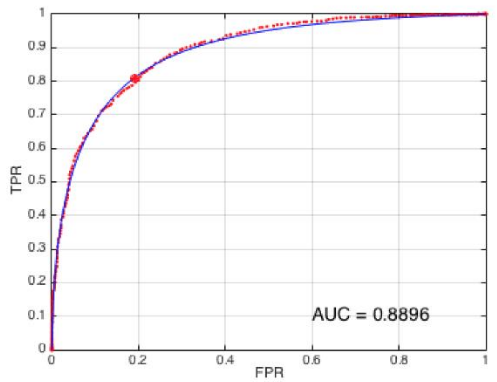
Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

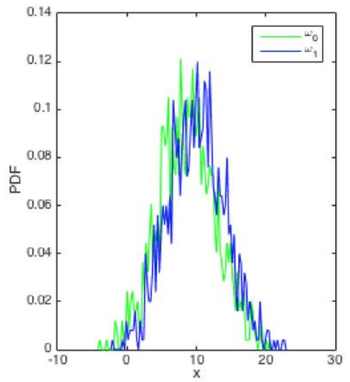
## ROC Curve Example

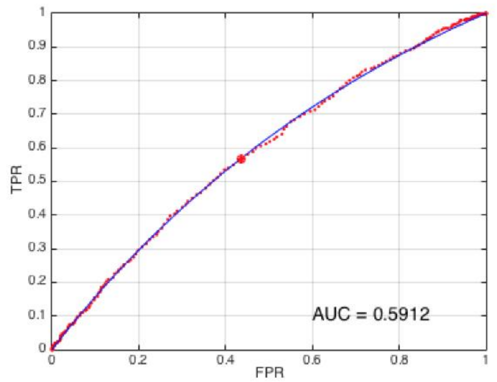












**Refe: Prof. D. Mery**

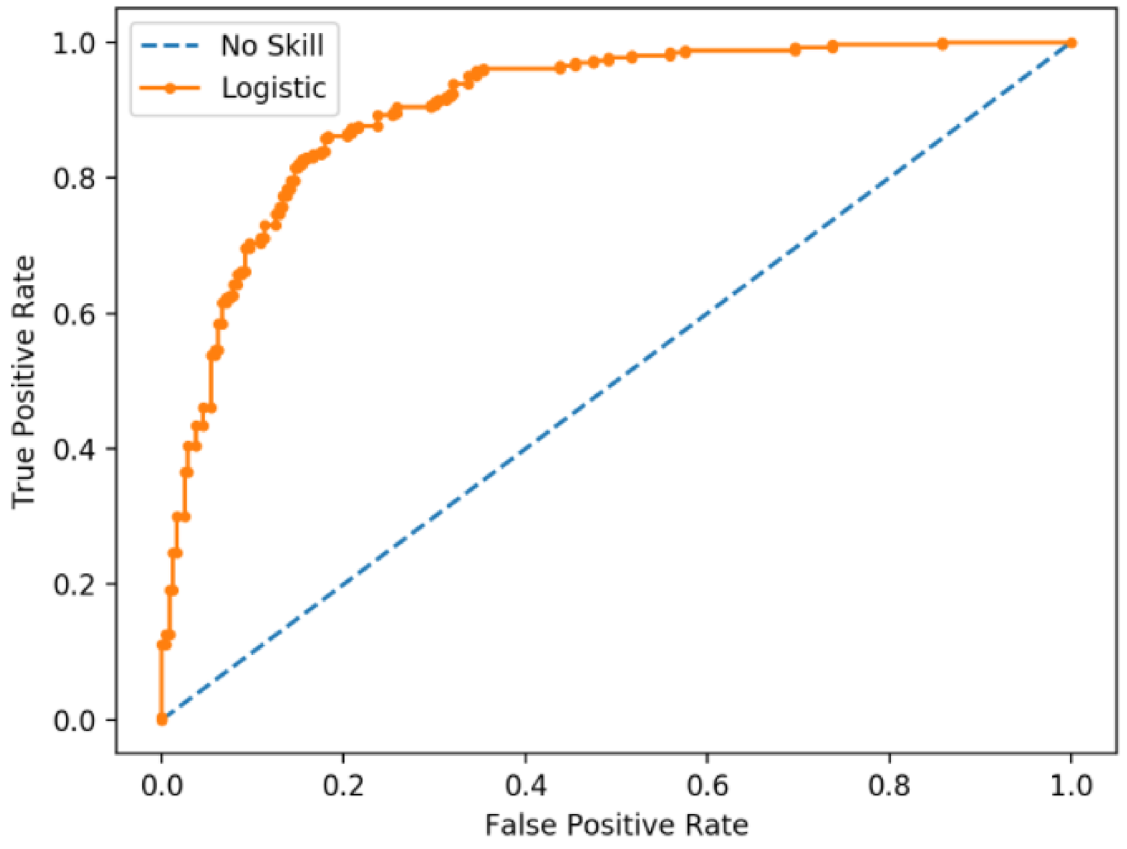
Gidia Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

# ROC Curve

```python
# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from matplotlib import pyplot
# generate 2 class dataset
X, y = make_classification(n_samples=1000, n_classes=2, random_state=1)
# split into train/test sets
trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, random_state=2)
# generate a no skill prediction (majority class)
ns_probs = [0 for _ in range(len(testy))]
# fit a model
model = LogisticRegression(solver='lbfgs')
model.fit(trainX, trainy)
# predict probabilities
lr_probs = model.predict_proba(testX)
# keep probabilities for the positive outcome only
lr_probs = lr_probs[:, 1]
# calculate scores
ns_auc = roc_auc_score(testy, ns_probs)
lr_auc = roc_auc_score(testy, lr_probs)
# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(testy, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(testy, lr_probs)
# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
# show the legend
pyplot.legend()
# show the plot
pyplot.show()
```

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.903
```



ROC Curve Plot for a No Skill Classifier and a Logistic Regression Model

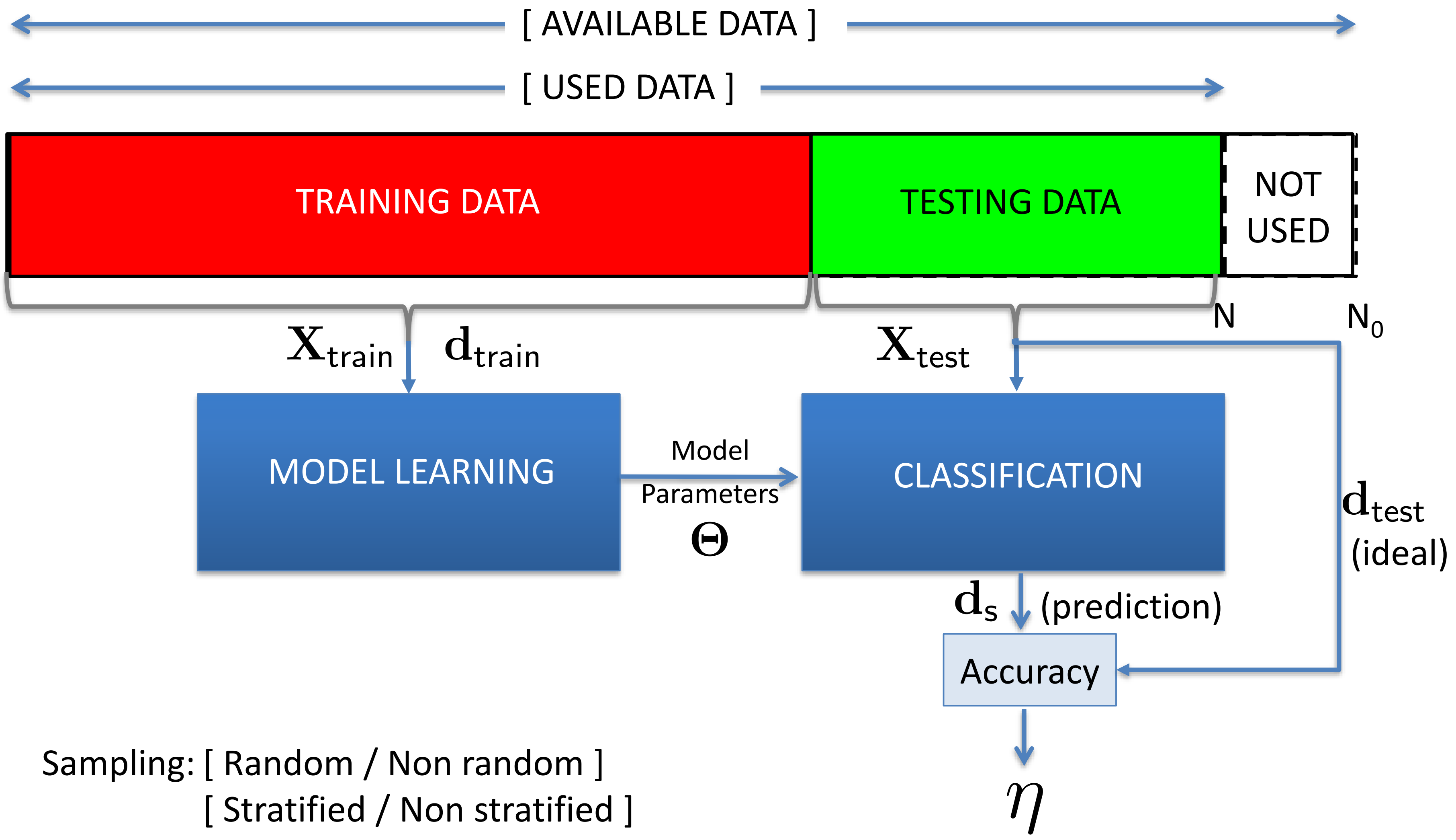
## Differences between ROC and PR curves

- In general, we will use the PR curve or the Average Precision when we have problems with unbalanced datasets, that is, when the positive class occurs rarely.

- When there are few positive examples, the ROC curve or the ROC AUC can give a high value, however, the PR curve will be far from its optimal value, revealing an indicator of precision related to the low probability of the positive class.

- It will be an interesting option to use the ROC curve and the AUC ROC when we have a more balanced dataset or we want to reveal an indicator more related to false alarms (false positives).

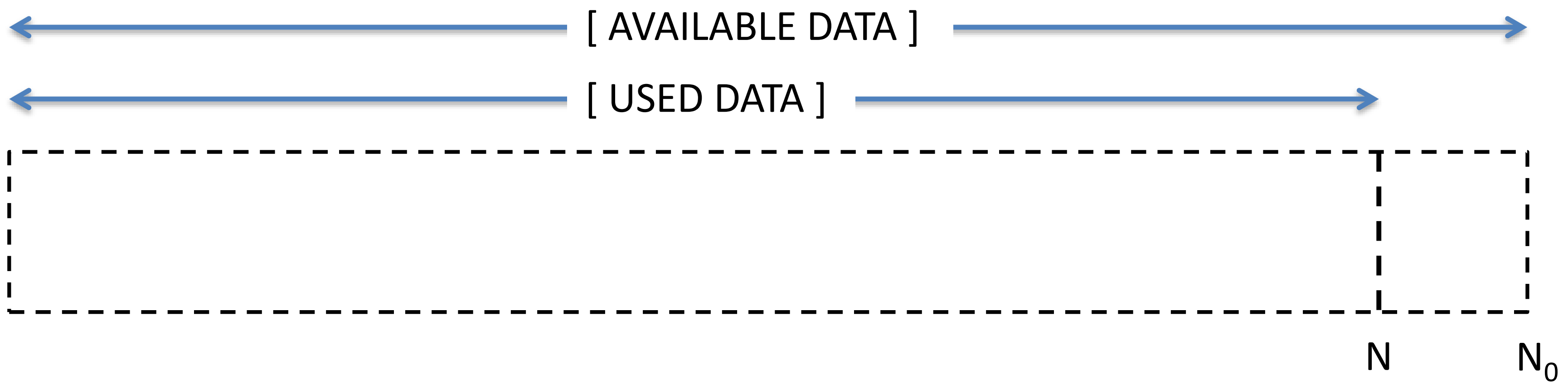
# Estimating the Accuracy of a Classifier



Sampling: [ Random / Non random ]
[ Stratified / Non stratified ]

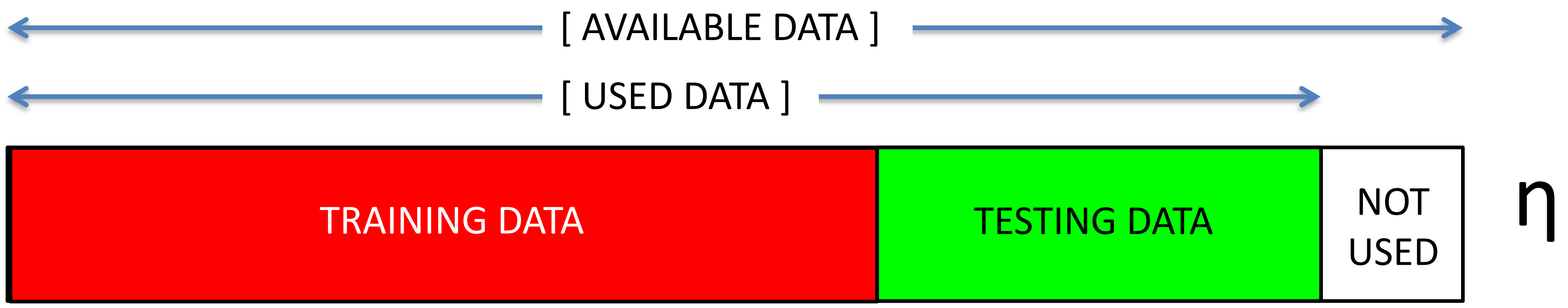
Refe: Prof. D. Mery

## Estimating the Accuracy of a Classifier



Refe: Prof. D. Mery

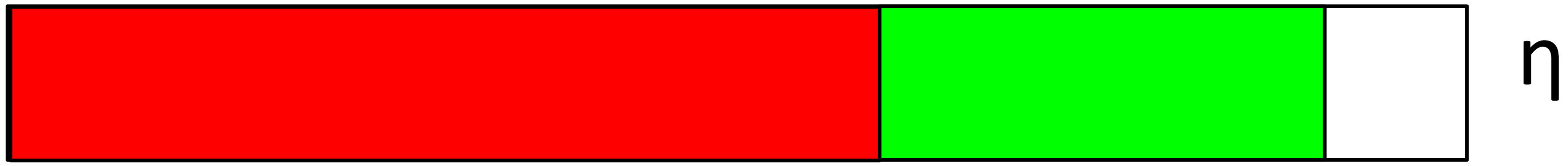
# Estimating the Accuracy of a Classifier



Refe: Prof. D. Mery

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

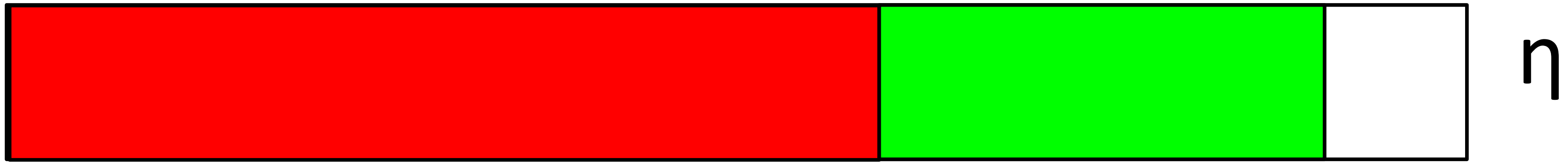
## Estimating the Accuracy of a Classifier



Refe: Prof. D. Mery

# Estimating the Accuracy of a Classifier

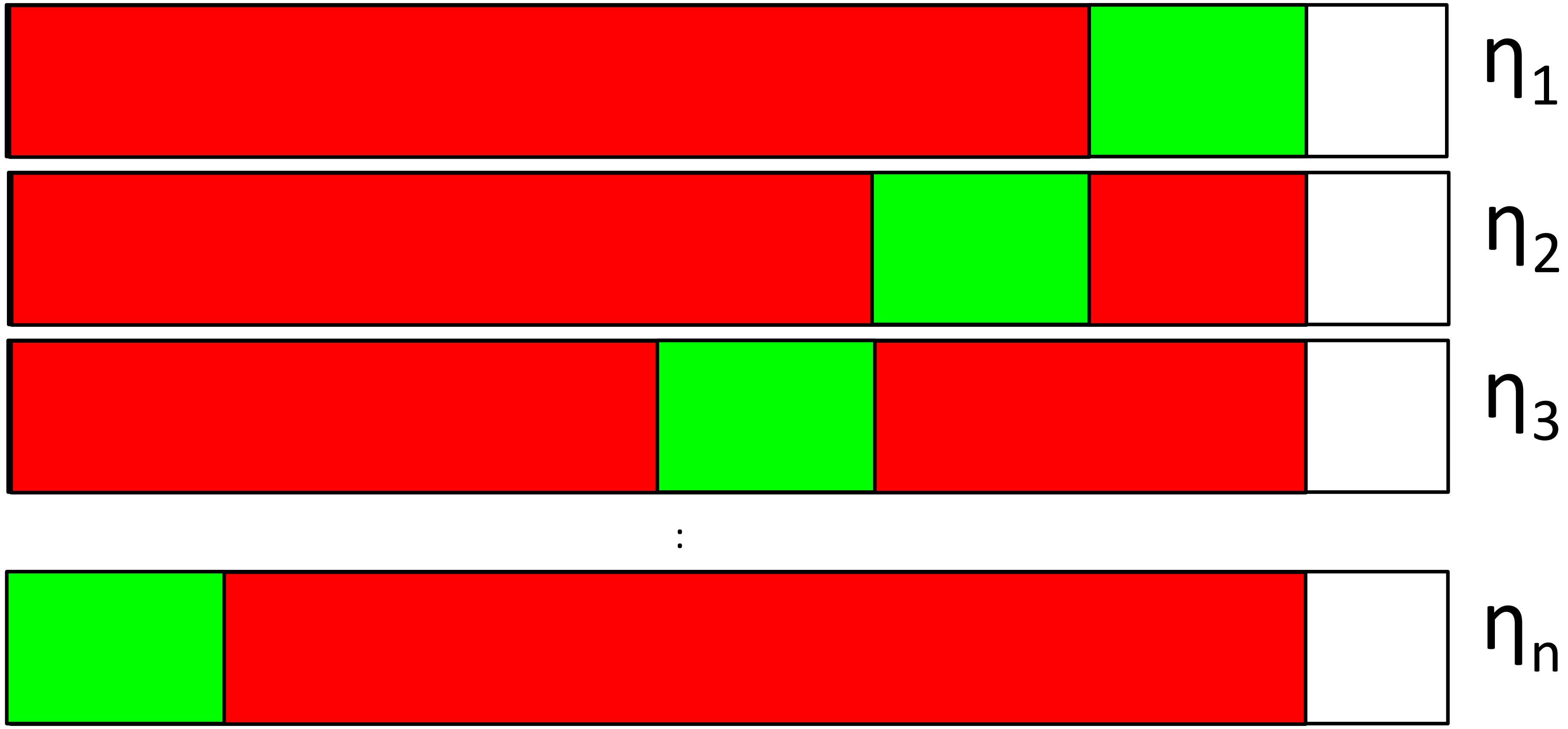
HO: HOLD OUT



Refe: Prof. D. Mery

# Estimating the Accuracy of a Classifier

CV: CROSS VALIDATION – n folds



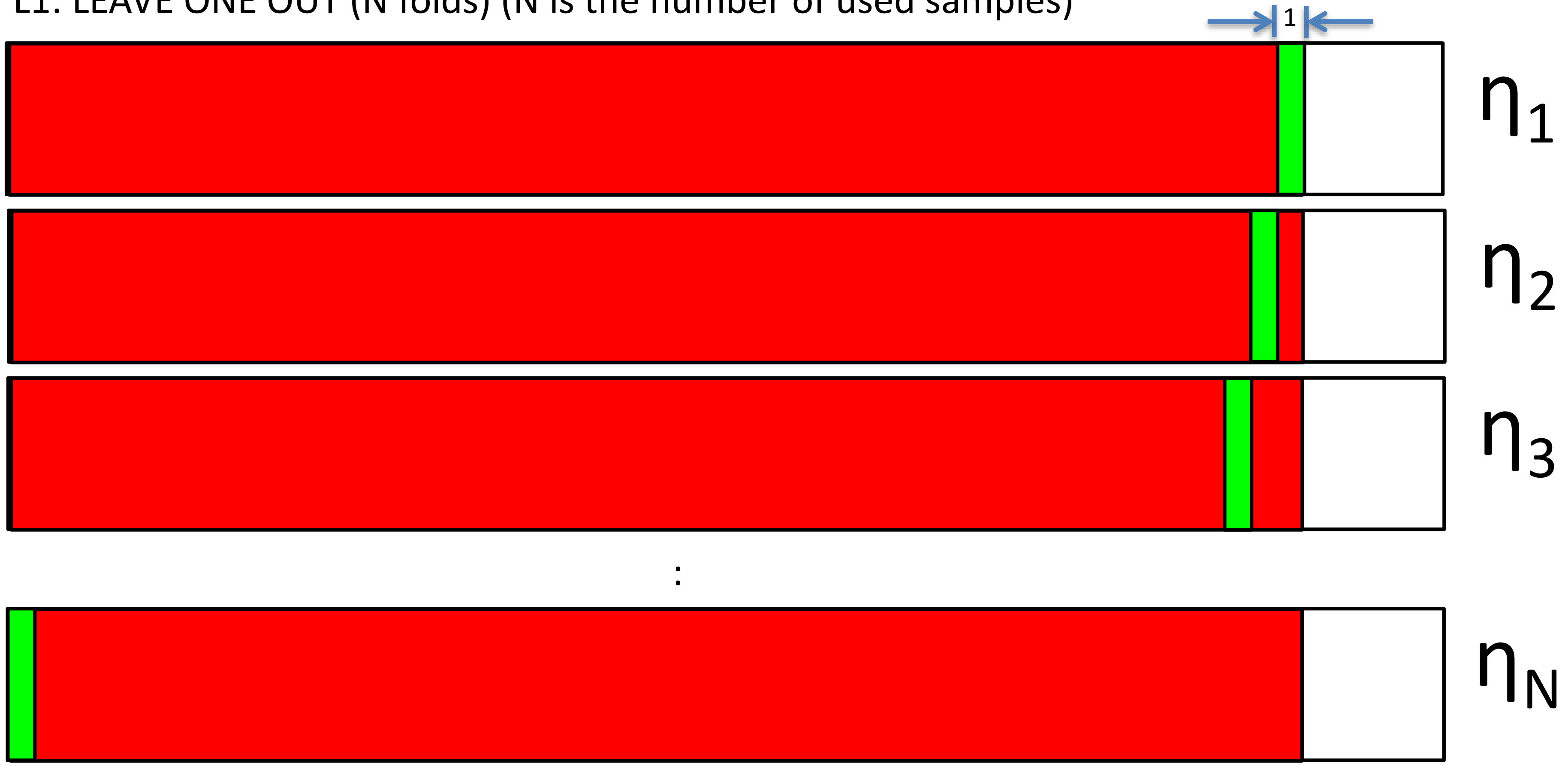
$$\eta = (\eta_1 + \eta_2 + \eta_3 + \ldots + \eta_n)/n$$

Refe: Prof. D. Mery

Gidia Grupo de I+D en Inteligencia Artificial
UNIVERSIDAD NACIONAL DE COLOMBIA

## Estimating the Accuracy of a Classifier

L1: LEAVE ONE OUT (N folds) (N is the number of used samples)

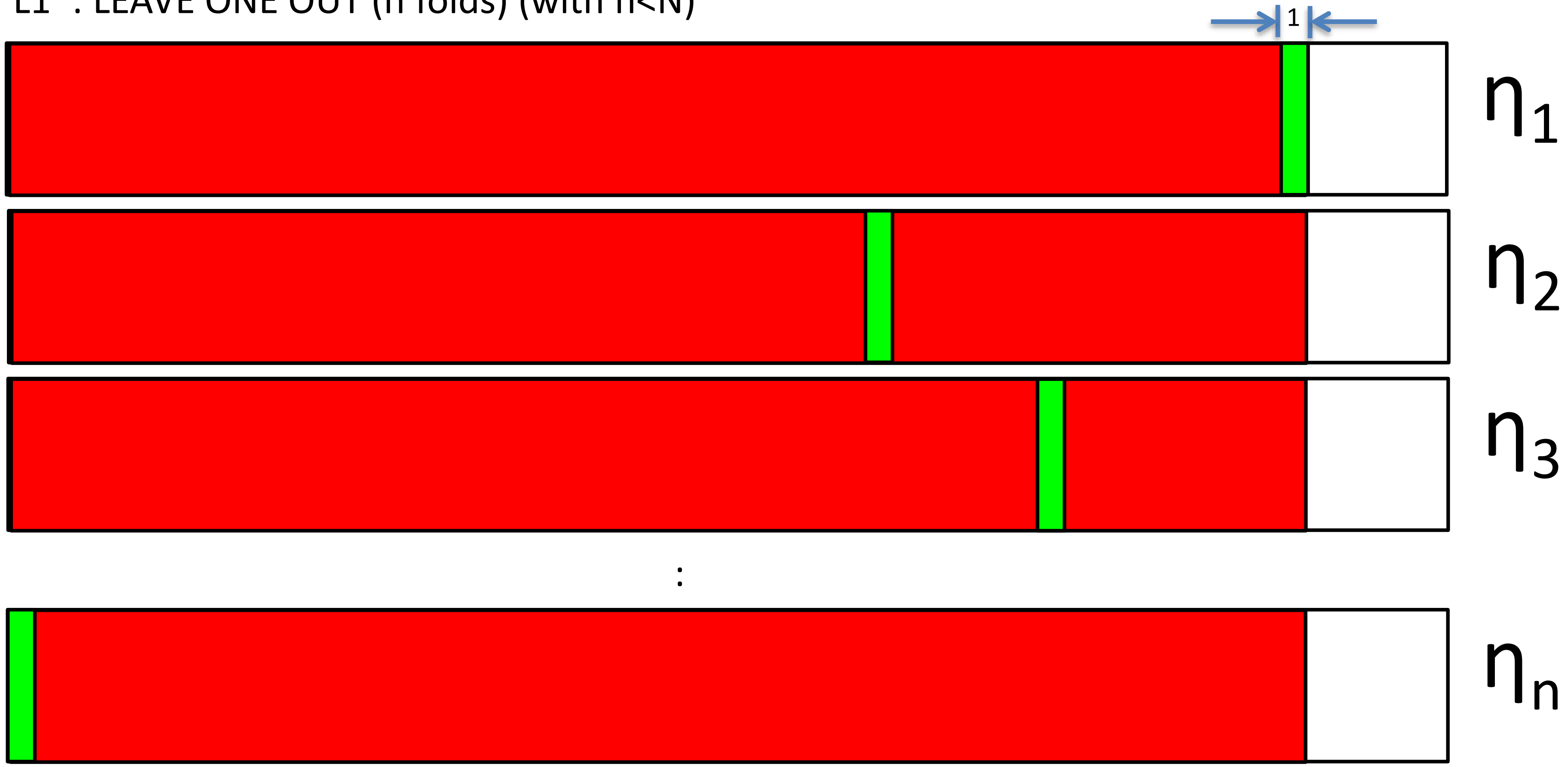


$$\eta = (\eta_1 + \eta_2 + \eta_3 + \ldots + \eta_N)/N$$

Refe: Prof. D. Mery

## Estimating the Accuracy of a Classifier

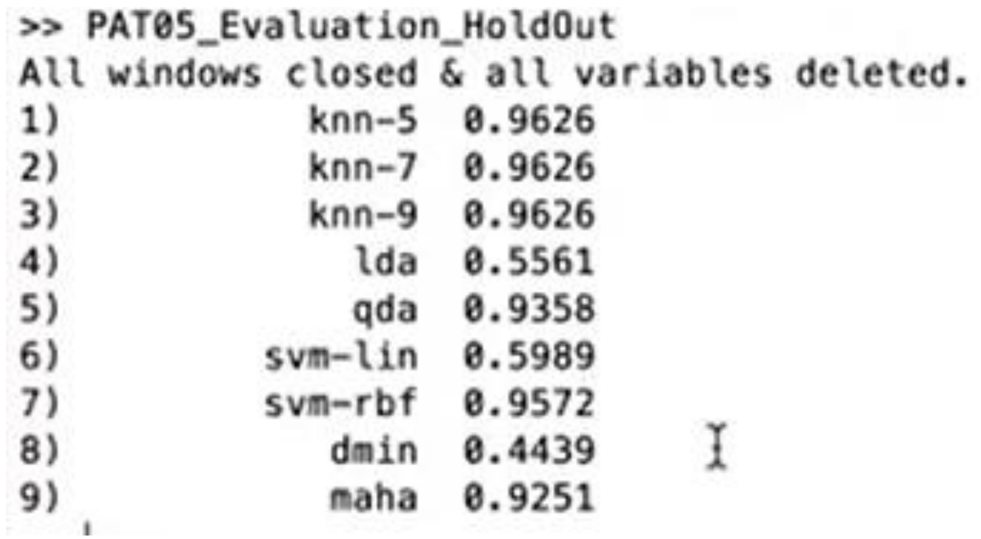
L1*: LEAVE ONE OUT (n folds) (with n<N)



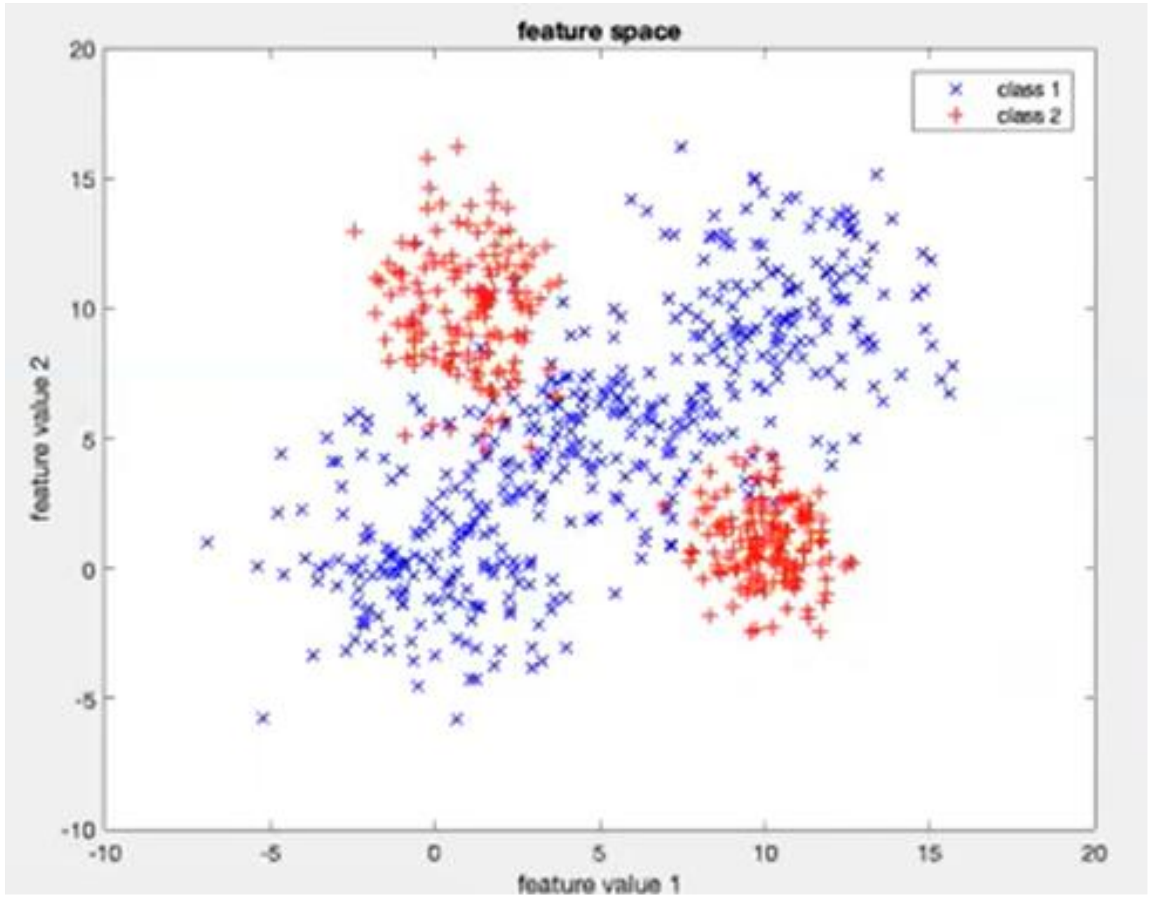
$$\eta = (\eta_1 + \eta_2 + \eta_3 + \ldots + \eta_n)/n$$

Refe: Prof. D. Mery

# Estimating the Accuracy of a Classifier

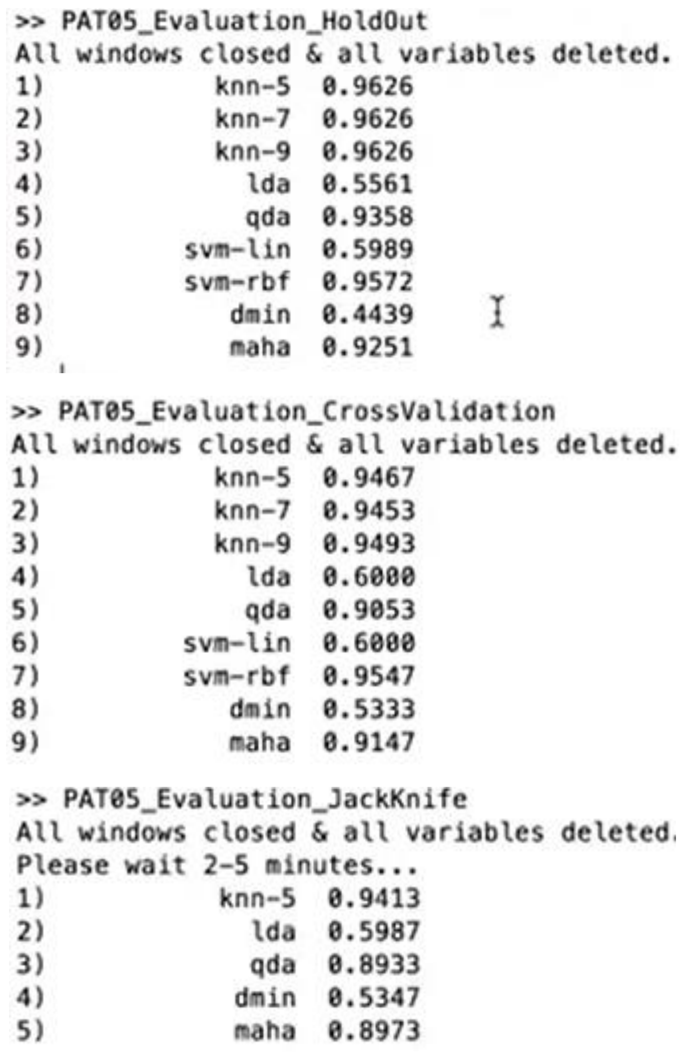


```
>> PAT05_Evaluation_HoldOut
All windows closed & all variables deleted.
1)           knn-5  0.9626
2)           knn-7  0.9626
3)           knn-9  0.9626
4)             lda  0.5561
5)             qda  0.9358
6)         svm-lin  0.5989
7)         svm-rbf  0.9572
8)            dmin  0.4439
9)            maha  0.9251
```

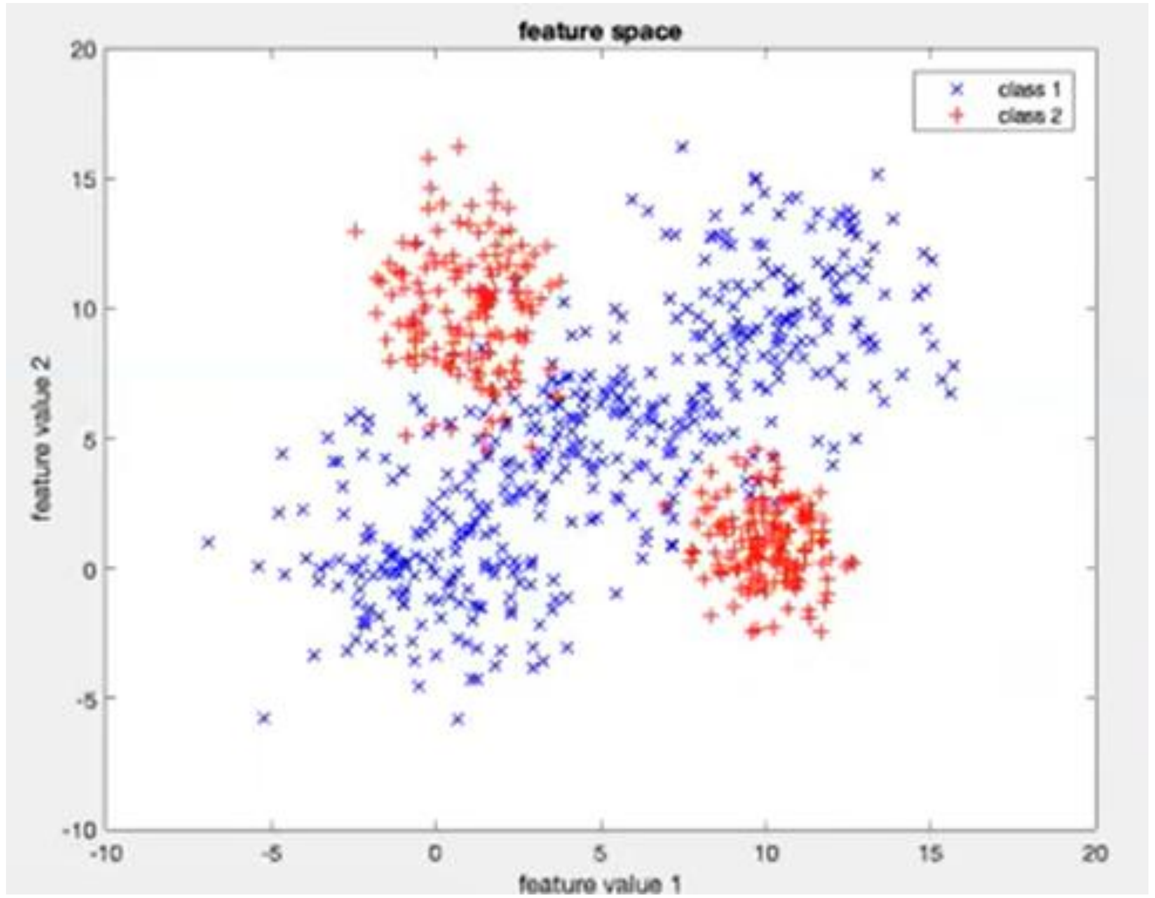


Refe: Prof. D. Mery

# Estimating the Accuracy of a Classifier



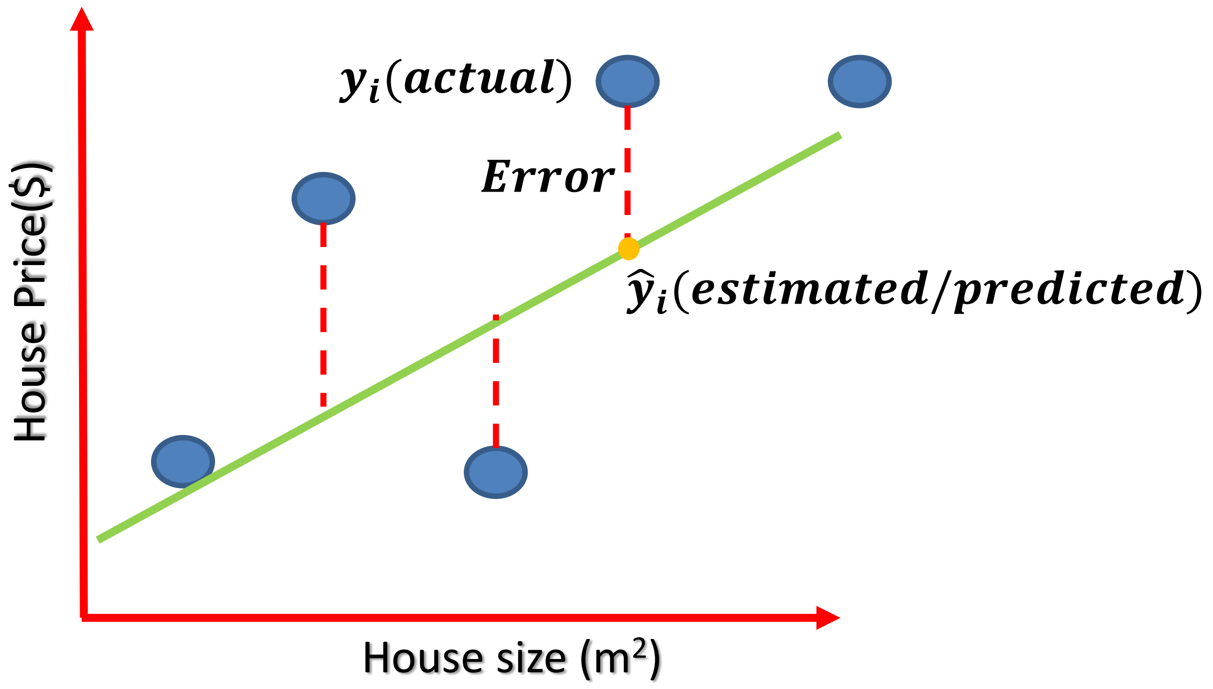
```
>> PAT05_Evaluation_HoldOut
All windows closed & all variables deleted.
1)           knn-5  0.9626
2)           knn-7  0.9626
3)           knn-9  0.9626
4)             lda  0.5561
5)             qda  0.9358
6)         svm-lin  0.5989
7)         svm-rbf  0.9572
8)            dmin  0.4439
9)            maha  0.9251

>> PAT05_Evaluation_CrossValidation
All windows closed & all variables deleted.
1)           knn-5  0.9467
2)           knn-7  0.9453
3)           knn-9  0.9493
4)             lda  0.6000
5)             qda  0.9053
6)         svm-lin  0.6000
7)         svm-rbf  0.9547
8)            dmin  0.5333
9)            maha  0.9147

>> PAT05_Evaluation_JackKnife
All windows closed & all variables deleted.
Please wait 2-5 minutes...
1)           knn-5  0.9413
2)             lda  0.5987
3)             qda  0.8933
4)            dmin  0.5347
5)            maha  0.8973
```



**Refe: Prof. D. Mery**

## Regression Metrics

- how to assess model performance?
- After the model fitting, we would like to assess the performance of the model by comparing model predictions to actual (True) data.



$$Residuals(Error) = \hat{y}_i - y_i$$

Gidia Grupo de I+D en Inteligencia Artificial

UNIVERSIDAD NACIONAL DE COLOMBIA

## Regression Metrics

- **Mean Absolute Error (MAE):** is obtained by calculating the absolute difference between the model predictions and the true (actual) values.
- **MAE** is a measure of the average magnitude of error generated by the regression model
- **MAE** is calculated as follow:

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

- If **MAE** is zero, this means that the model predictions are perfect

## Regression Metrics

- **Mean Square Error (MSE):** is similar to Mean Absolute Error (**MAE)**, bur instead of using absolute values, square of the difference between the model predictions and the training dataset (true values) is calculated.
- **MSE** values are generally larger compared to the **MAE** since the residuals are being squared.
- In case of outliers, **MSE** will become much larger compared to **MAE**.
- In MSE, error increases in a quadratic fashion while in **MAE**, is proportional fashion
- **MSE** is calculated as follow:

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

- **Root Mean Square Error (RMSE):** is the standard deviation of the residuals.
- **RMSE** provide an estimate of how large the residuals are being dispersed
- **RMSE** is calculated as follow:

$$\boldsymbol{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

## Regression Metrics

- **Mean Absolut Percentage Error (MAPE):** the **MAE** values can range from 0 to infinity which make difficult to interpret the results as compared to the training data.
- **MAPE** is equivalent to **MAE** but provides the error in a percentage form and therefore overcome **MAE** limitations.
- **MAPE** have some limitations if the data point values is zero (there is a division involved)
- **MAPE** is calculated as follow:

$$\boldsymbol{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| / y_i$$

- **Mean Percentage Error (MPE):** is similar to **MAPE** but without the absolute operation.
- **MPE** is useful to provide an insight equivalent of how many positive errors as compared to negative ones
- **MPE** is calculated as follow:

$$\boldsymbol{MPE} = \frac{100\%}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\ / y_i$$

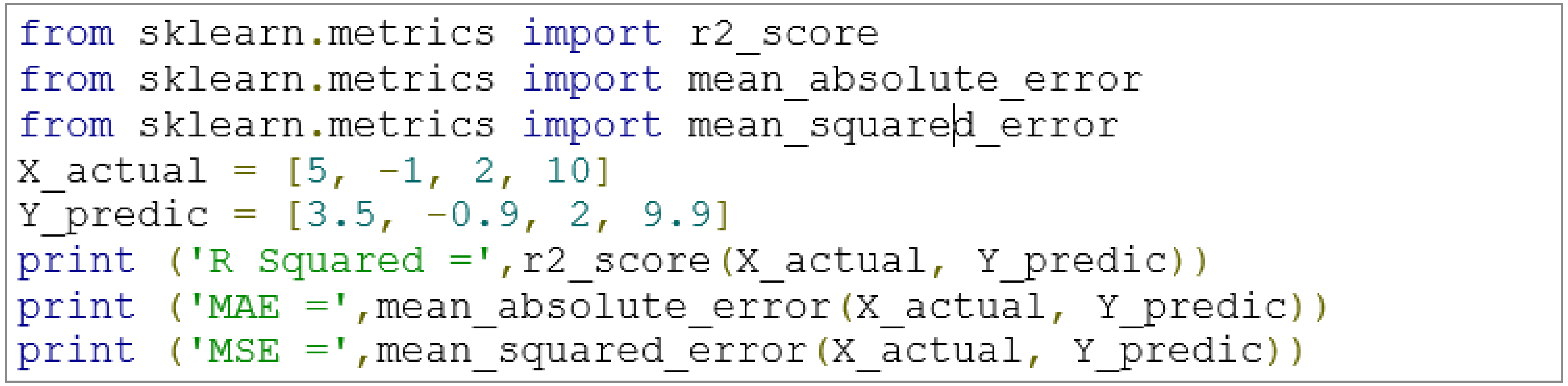
## Regression Metrics

- **R-Square** or the coefficient of determination represents the proportion of variance (of y) that has been explained by the independent variable in the model.
- If $R^2 = 80$ this means that 80% of the instances in house prices is due to the increase in the size (m²).

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

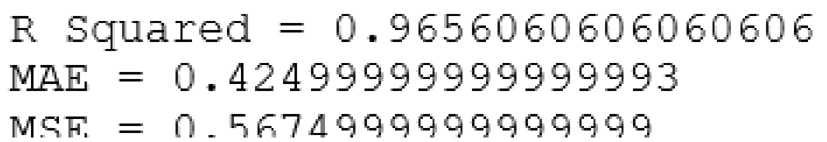
- In the above equation, numerator is MSE and the denominator is the variance in $Y$ values.

## Regression Metrics



```python
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
X_actual = [5, -1, 2, 10]
Y_predic = [3.5, -0.9, 2, 9.9]
print ('R Squared =',r2_score(X_actual, Y_predic))
print ('MAE =',mean_absolute_error(X_actual, Y_predic))
print ('MSE =',mean_squared_error(X_actual, Y_predic))
```

Output



```
R Squared = 0.9656060606060606
MAE = 0.42499999999999993
MSE = 0.5674999999999999
```