



**Mondragon  
Unibertsitatea**

Goi Eskola  
Politeknikoa

# **Fundamentos para la validación de modelos II**

Fundamentos del Aprendizaje  
Automático



**Mondragon  
Unibertsitatea**

Goi Eskola  
Politeknikoa

# **Validación y selección de modelo**

# Índice

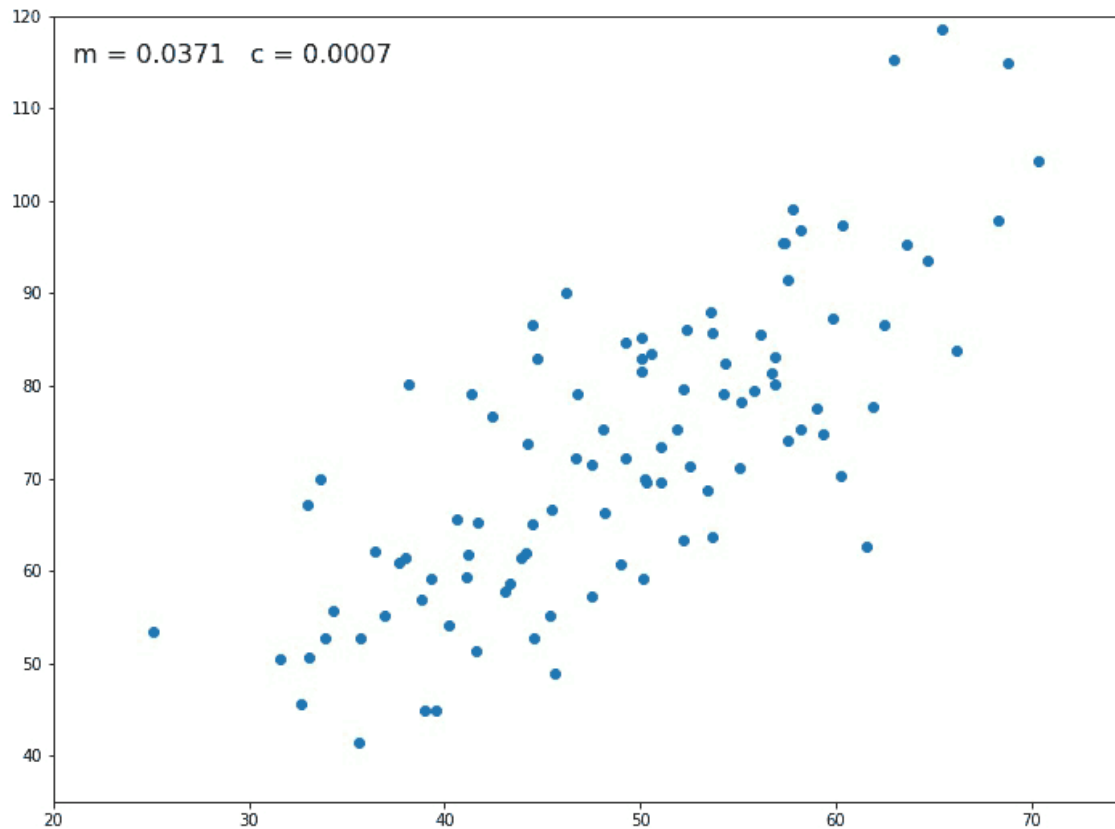
1. Regresión
2. Clasificación
- ~~3. Clustering~~



# **Regresión**

ejemplo: regresión lineal

$$Y = mX + c$$



Objetivo: valores de  $m$  y  $c$  que mejor se adapten a los puntos: **error mínimo**

# Regresión

- Mean Absolute Error (MAE)
  - Le media de la diferencia absoluta entre los valores reales y los predichos
  - Sean  $y$  e  $\hat{y}$  los valores real y predicho

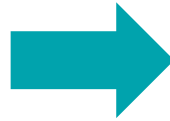
$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} |y_i - \hat{y}_i|$$

- Mejor valor possible: 0 (mínimo)
- Misma unidad que la variable de respuesta

# Regresión

- Error cuadrático medio (Mean squared error MSE)
  - Sean  $y$  e  $\hat{y}$  los valores real y predicho

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2$$



$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}}$$

- Mejor valor posible: 0

- Root Mean Squared Error

- Escala similar a MAE

# Regresión

- Python scikit-learn

## Regression metrics

See the [Regression metrics](#) section of the user guide for further details.

|   |   |
|---|---|
| <code>metrics.explained_variance_score</code> (y_true, y_pred)  | Explained variance regression score function                  |
| <code>metrics.max_error</code> (y_true, y_pred)                 | max_error metric calculates the maximum residual error.       |
| <code>metrics.mean_absolute_error</code> (y_true, y_pred)       | Mean absolute error regression loss                           |
| <code>metrics.mean_squared_error</code> (y_true, y_pred[, ...]) | Mean squared error regression loss                            |
| <code>metrics.mean_squared_log_error</code> (y_true, y_pred)    | Mean squared logarithmic error regression loss                |
| <code>metrics.median_absolute_error</code> (y_true, y_pred)     | Median absolute error regression loss                         |
| <code>metrics.r2_score</code> (y_true, y_pred[, ...])           | R^2 (coefficient of determination) regression score function. |





# **Clasificación**

# Clasificación

1. Generalidades
2. Matriz de confusión
3. Scores relacionados con la matriz de confusión
  1. Acierto y error
  2. Ratios de aciertos y errores por clases
  3. Precisión
  4. F1-Score
  5. Area bajo la curva ROC
  6. Gráfica Precisión-Recall y precisión media
4. Scikit-learn

# Generalidades

- Las funciones para selección de modelo y evaluación basadas en validación cruzada son válidas para clasificación:
  - Hay que elegir el *score* adecuado usando el parámetro ***scoring***
- Objetivo:** obtener el valor más alto posible

| Scoring               | Function                                     | Comment                                     |
|-----------------------|--|---|
| <b>Classification</b> |  |   |
| 'accuracy'            | <code>metrics.accuracy_score</code>          |   |
| 'balanced_accuracy'   | <code>metrics.balanced_accuracy_score</code> | for binary targets                          |
| 'average_precision'   | <code>metrics.average_precision_score</code> |   |
| 'brier_score_loss'    | <code>metrics.brier_score_loss</code>        |   |
| 'f1'                  | <code>metrics.f1_score</code>                | for binary targets                          |
| 'f1_micro'            | <code>metrics.f1_score</code>                | micro-averaged                              |
| 'f1_macro'            | <code>metrics.f1_score</code>                | macro-averaged                              |
| 'f1_weighted'         | <code>metrics.f1_score</code>                | weighted average                            |
| 'f1_samples'          | <code>metrics.f1_score</code>                | by multilabel sample                        |
| 'neg_log_loss'        | <code>metrics.log_loss</code>                | requires <code>predict_proba</code> support |
| 'precision' etc.      | <code>metrics.precision_score</code>         | suffixes apply as with 'f1'                 |
| 'recall' etc.         | <code>metrics.recall_score</code>            | suffixes apply as with 'f1'                 |
| 'roc_auc'             | <code>metrics.roc_auc_score</code>           |   |

# Matriz de confusión (CM)

- Recoge los aciertos y errores de cada clase

|             |           | Predicción                |                           |
|-------------|-----------|---------------------------|---------------------------|
|             |           | Positivos                 | Negativos                 |
| Observación | Positivos | Verdaderos Positivos (VP) | Falsos Negativos (FN)     |
|             | Negativos | Falsos Positivos (FP)     | Verdaderos Negativos (VN) |

# Matriz de confusión (CM)

- Recoge los aciertos y errores de cada clase
- Ignora el desbalanceo entre las clases

|             |           | Predicción                |                           |
|-------------|-----------|---------------------------|---------------------------|
|             |           | Positivos                 | Negativos                 |
| Observación | Positivos | Verdaderos Positivos (VP) | Falsos Negativos (FN)     |
|             | Negativos | Falsos Positivos (FP)     | Verdaderos Negativos (VN) |

# Matriz de confusión (CM)

- Recoge los aciertos y errores de cada clase
- Ignora el desbalanceo entre las clases
- Se puede extender a múltiples clases

|             |           | Predicción                |                           |
|-------------|-----------|---------------------------|---------------------------|
|             |           | Positivos                 | Negativos                 |
| Observación | Positivos | Verdaderos Positivos (VP) | Falsos Negativos (FN)     |
|             | Negativos | Falsos Positivos (FP)     | Verdaderos Negativos (VN) |

# Matriz de confusión (CM)

- Recoge los aciertos y errores de cada clase
- Ignora el desbalanceo entre las clases
- Se puede extender a múltiples clases

|             |           | Predicción                |                           |
|-------------|-----------|---------------------------|---------------------------|
|             |           | Positivos                 | Negativos                 |
| Observación | Positivos | Verdaderos Positivos (VP) | Falsos Negativos (FN)     |
|             | Negativos | Falsos Positivos (FP)     | Verdaderos Negativos (VN) |

|             |   | Predicted |   |   |       |
|-------------|---|-----------|---|---|-------|
|             |   | A         | B | C |       |
| True labels | A | 2         | 2 | 0 | 4     |
|             | B | 1         | 2 | 0 | 3     |
|             | C | 0         | 0 | 3 | 3     |
|             |   | 3         | 4 | 3 | Total |

# Matriz de confusión (CM)

- Recoge los aciertos y errores de cada clase
- Ignora el desbalanceo entre las clases
- Se puede extender a múltiples clases

|             |           | Predicción                |                           |
|-------------|-----------|---------------------------|---------------------------|
|             |           | Positivos                 | Negativos                 |
| Observación | Positivos | Verdaderos Positivos (VP) | Falsos Negativos (FN)     |
|             | Negativos | Falsos Positivos (FP)     | Verdaderos Negativos (VN) |

|             |   | Predicted |   |   |       |
|-------------|---|-----------|---|---|-------|
|             |   | A         | B | C |       |
| True labels | A | 2         | 2 | 0 | 4     |
|             | B | 1         | 2 | 0 | 3     |
|             | C | 0         | 0 | 3 | 3     |
|             |   | 3         | 4 | 3 | Total |

|        |       | Prediction  |             |             |     |             |       |
|--------|-------|-------------|-------------|-------------|-----|-------------|-------|
|        |       | $c_1$       | $c_2$       | $c_3$       | ... | $c_n$       | Total |
| Actual | $c_1$ | $TP_1$      | $FN_{12}$   | $FN_{13}$   | ... | $FN_{1n}$   | $N_1$ |
|        | $c_2$ | $FN_{21}$   | $TP_2$      | $FN_{23}$   | ... | $FN_{2n}$   | $N_2$ |
|        | $c_3$ | $FN_{31}$   | $FN_{32}$   | $TP_3$      | ... | $FN_{3n}$   | $N_3$ |
|        | ...   | ...         | ...         | ...         | ... | ...         | ...   |
|        | $c_n$ | $FN_{n1}$   | $FN_{n2}$   | $FN_{n3}$   | ... | $TP_n$      | $N_n$ |
| Total  |       | $\hat{N}_1$ | $\hat{N}_2$ | $\hat{N}_3$ | ... | $\hat{N}_n$ | N     |



# Matriz de confusión (CM)

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import confusion_matrix
# A sample toy binary classification dataset
X, y = datasets.make_classification(n_classes=2, random_state=0)
svm = LinearSVC(random_state=0)
def tn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 0]
def fp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[0, 1]
def fn(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 0]
def tp(y_true, y_pred): return confusion_matrix(y_true, y_pred)[1, 1]
scoring = {'tp': make_scorer(tp), 'tn': make_scorer(tn),
           'fp': make_scorer(fp), 'fn': make_scorer(fn)}
cv_results = cross_validate(svm.fit(X, y), X, y,
                           scoring=scoring, cv=5)

# Getting the test set true positive scores
print(cv_results['test_tp'])

# Getting the test set false negative scores
print(cv_results['test_fn'])
```

# Scores basados en la CM

- Accuracy

|                       | Predicted label<br>class 1                     | Predicted label<br>class 2                     |
|-----------------------|--|--|
| True label<br>class 1 | <b>correct</b><br>true positive<br>for class 1 | <b>wrong</b><br>false positive<br>for class 2  |
| True label<br>class 2 | <b>wrong</b><br>false positive<br>for class 1  | <b>correct</b><br>true positive<br>for class 2 |

$$\text{accuracy} = \frac{\text{orange} + \text{blue}}{\text{orange} + \text{yellow} + \text{blue} + \text{green}}$$

## Accuracy (ACC)

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

## Error (ERR)

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

# Scores basados en la CM



## Problemas con desbalanceo

- Fabricamos 1000 piezas
  - Defectivo: 3.8%
  - No defectivo 96.2 %
- Accuracy: 96.2%
  - Nuestro modelo siempre dice que no tenemos defectivo

# Scores basados en la CM

- Precisión (PRE)



|                       | Predicted label<br>class 1                     | Predicted label<br>class 2                     |
|-----------------------|--|--|
| True label<br>class 1 | <b>correct</b><br>true positive<br>for class 1 | <b>wrong</b><br>false positive<br>for class 2  |
| True label<br>class 2 | <b>wrong</b><br>false positive<br>for class 1  | <b>correct</b><br>true positive<br>for class 2 |

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

$$\text{class 1 precision} = \frac{\text{orange square}}{\text{orange square} + \text{yellow square}}$$

$$\text{class 2 precision} = \frac{\text{blue square}}{\text{blue square} + \text{green square}}$$

# Scores basados en la CM

- Recall (REC) = Sensitivity (SEN) = True Positive Rate (TPR)

|                       | Predicted label<br>class 1                     | Predicted label<br>class 2                     |
|-----------------------|--|--|
| True label<br>class 1 | <b>correct</b><br>true positive<br>for class 1 | <b>wrong</b><br>false positive<br>for class 2  |
| True label<br>class 2 | <b>wrong</b><br>false positive<br>for class 1  | <b>correct</b><br>true positive<br>for class 2 |

$$SEN = TPR = REC = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$\text{class 1 recall} = \frac{\text{orange square}}{\text{orange square} + \text{green square}}$$

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

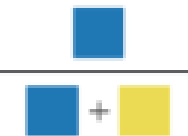
$$\text{class 2 recall} = \frac{\text{blue square}}{\text{blue square} + \text{yellow square}}$$

# Scores basados en la CM

- Specitifity (SPC) = True Negative Rate (TNR)

|                       | Predicted label<br>class 1                     | Predicted label<br>class 2                     |
|-----------------------|--|--|
| True label<br>class 1 | <b>correct</b><br>true positive<br>for class 1 | <b>wrong</b><br>false positive<br>for class 2  |
| True label<br>class 2 | <b>wrong</b><br>false positive<br>for class 1  | <b>correct</b><br>true positive<br>for class 2 |

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

specitifity = 

# Scores basados en CM

- **F1-score**

- Puede ser una buena métrica si buscamos balance entre la precisión y el recall y las clases nos están distribuidas de forma equitativa

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

# Scores basados en CM

- F1-score

|             | Precisión | Recall | Media ((p + r) / 2) |
|-------------|-----------|--------|---------------------|
| Algoritmo 1 | 0.5       | 0.4    | 0.45                |
| Algoritmo 2 | 0.7       | 0.1    | 0.4                 |
| Algoritmo 3 | 0.02      | 1.0    | 0.51                |



# Scores basados en CM

- F1-score

|             | Precisión | Recall | Media ((p + r) / 2) |
|-------------|-----------|--------|---------------------|
| Algoritmo 1 | 0.5       | 0.4    | 0.45                |
| Algoritmo 2 | 0.7       | 0.1    | 0.4                 |
| Algoritmo 3 | 0.02      | 1.0    | 0.51                |

# Scores basados en CM

- F1-score

|             | Precisión | Recall | Media ((p + r) / 2) | F1     |
|-------------|-----------|--------|---------------------|--------|
| Algoritmo 1 | 0.5       | 0.4    | 0.45                | 0.444  |
| Algoritmo 2 | 0.7       | 0.1    | 0.4                 | 0.175  |
| Algoritmo 3 | 0.02      | 1.0    | 0.51                | 0.0392 |

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

# Scores basados en CM

- Objetivo final:
  - Alta precisión + recall alto **para cada clase**:
    - Cada clase es clasificada correctamente por el modelo

# Scores basados en CM

|                             | Predicted label<br>defective | Predicted label<br>not defective |
|-----------------------------|------------------------------|----------------------------------|
| True label<br>defective     | 0                            | 380                              |
| True label<br>not defective | 0                            | 9620                             |

Acc = 96,2%

Precisión no defectivo = 1

Recall no defectivo = 1

F1 = no computable

$$\text{accuracy} = \frac{9620 + 0}{9620 + 380 + 0 + 0}$$

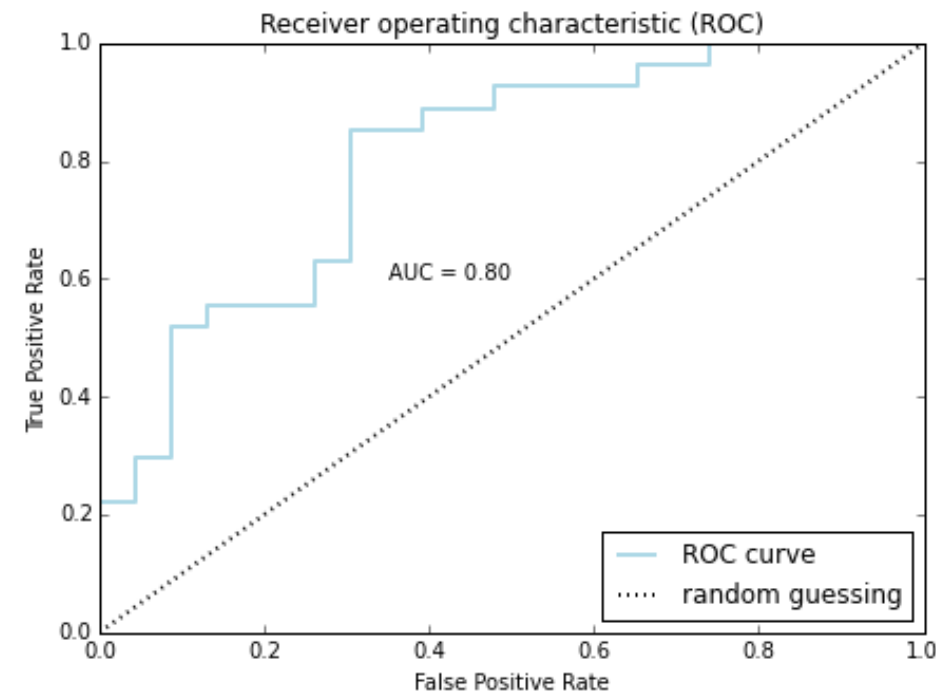
$$\text{defective recall} = \frac{0}{0 + 380}$$

$$\text{not defective precision} = \frac{9620}{380 + 9620}$$

$$\text{not defective recall} = \frac{9620}{9620 + 0}$$

# Scores basados en CM

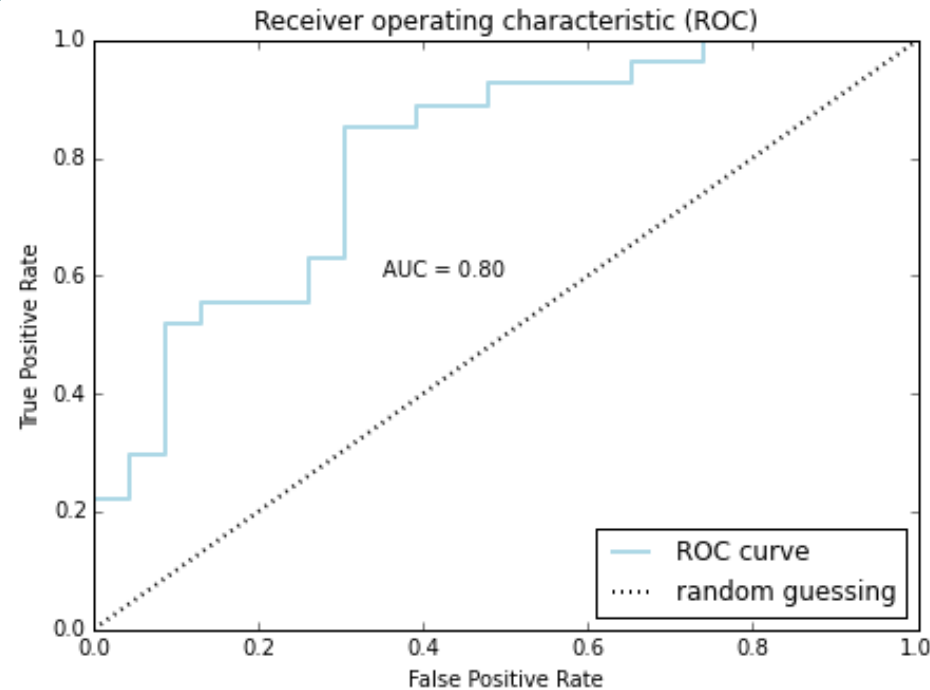
## Area Under the ROC Curve (AUC)



# Scores basados en CM

## Area Under the ROC Curve (AUC)

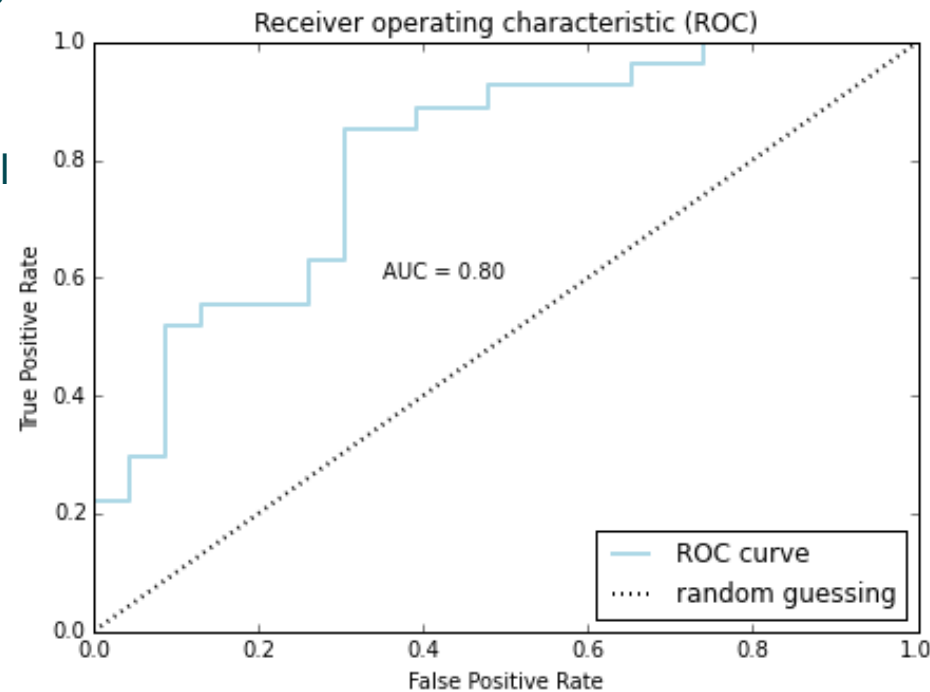
- Contiene la proporción de aciertos en la clase positiva (VP o TP) para diferentes niveles de fallos en ella (FP)



# Scores basados en CM

## Area Under the ROC Curve (AUC)

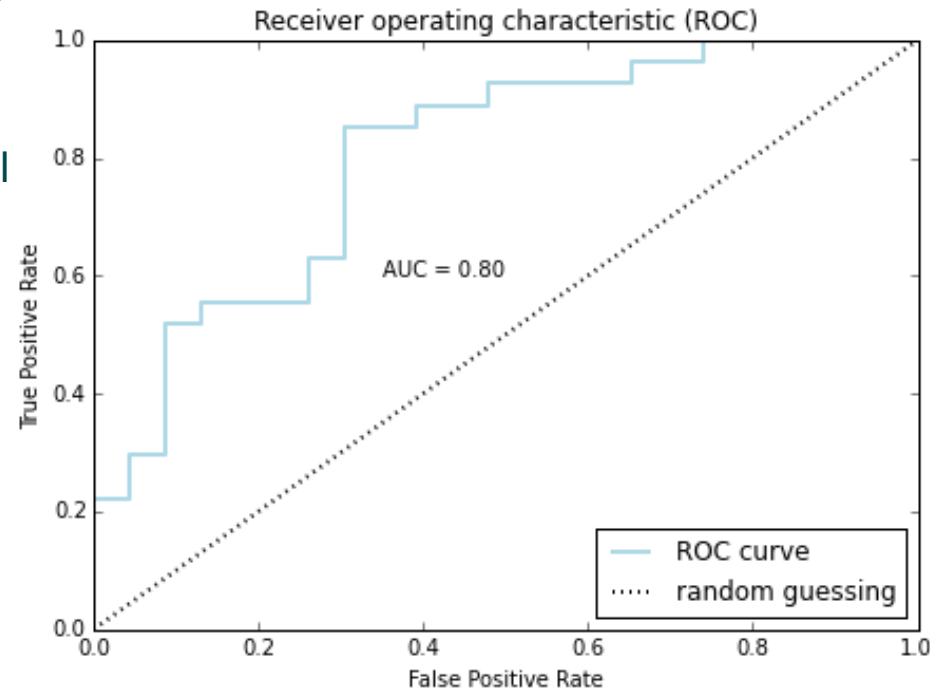
- Contiene la proporción de aciertos en la clase positiva (VP o TP) para diferentes niveles de fallos en ella (FP)
- El área por debajo de la curva indica la bondad del clasificador.



# Scores basados en CM

## Area Under the ROC Curve (AUC)

- Contiene la proporción de aciertos en la clase positiva (VP o TP) para diferentes niveles de fallos en ella (FP)
- El área por debajo de la curva indica la bondad del clasificador.
- El clasificador perfecto (sin fallo) tendría una proporción de TP de 1 para cualquier FP (área = 1).

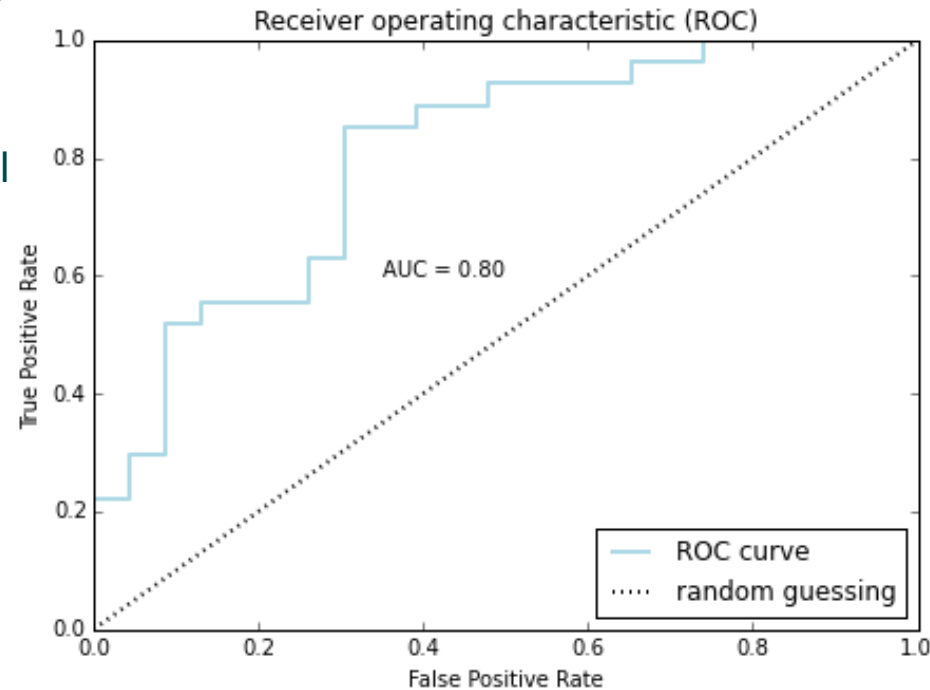




# Scores basados en CM

## Area Under the ROC Curve (AUC)

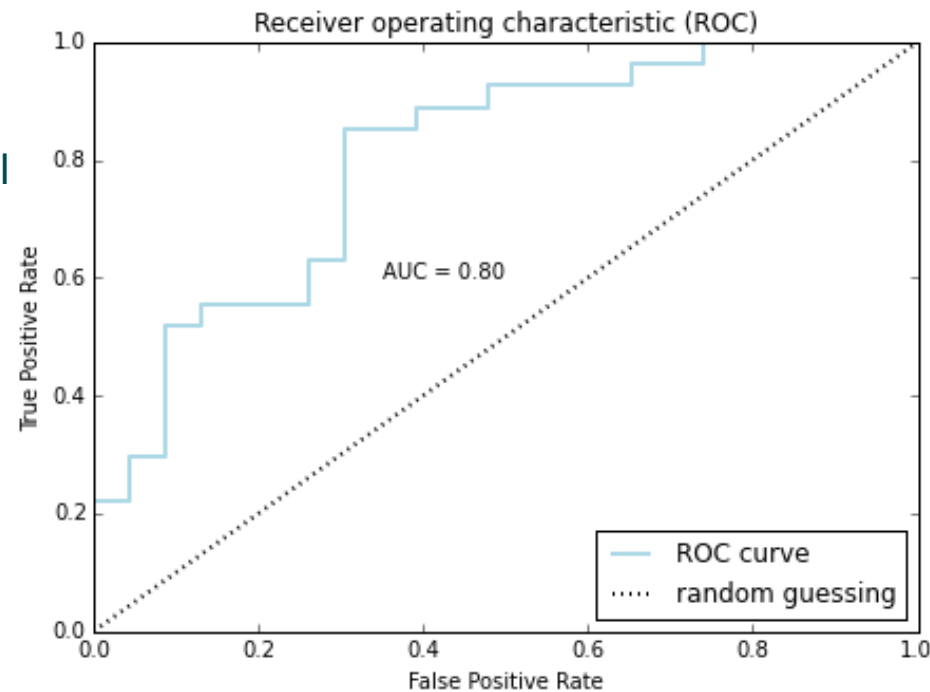
- Contiene la proporción de aciertos en la clase positiva (VP o TP) para diferentes niveles de fallos en ella (FP)
- El área por debajo de la curva indica la bondad del clasificador.
- El clasificador perfecto (sin fallo) tendría una proporción de TP de 1 para cualquier FP (área = 1).
- Un clasificador aleatorio (lanzamiento de moneda se correspondería con la línea roja (área = 0.5)



# Scores basados en CM

## Area Under the ROC Curve (AUC)

- Contiene la proporción de aciertos en la clase positiva (TP) para diferentes niveles de fallos en ella (FP)
- El área por debajo de la curva indica la bondad del clasificador.
- El clasificador perfecto (sin fallo) tendría una proporción de TP de 1 para cualquier FP (área = 1).
- Un clasificador aleatorio (lanzamiento de moneda se correspondería con la línea roja (área = 0.5)
- Suele considerarse como bueno un valor de AUC por encima de 0.75



# **Scores basados en CM**

## **Precision-Recall plot**

# Scores basados en CM

## Precision-Recall plot

Se obtienen pares de valores de PRE y REC para diferentes *thresholds* que cubran la totalidad del rango  $[0, 1]$  para ambos *scores*

# Scores basados en CM

- Precision-Recall plot
- Trade off precisión y recall
  - Regresión logística  $0 \leq h_0(x) \leq 1$
  - Predice 1 si  $h_0(x) \geq 0.5$
  - Predice 0 si  $h_0(x) < 0.5$
- Supongamos que queremos predecir  $y = 1$  (cáncer)

# Scores basados en CM

- Precision-Recall plot

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

- Trade off precisión y recall

Regresión logística  $0 \leq h_0(x) \leq 1$

Predice 1 si  $h_0(x) \geq 0.9$

Predice 0 si  $h_0(x) < 0.9$

$$SEN = TPR = REC = \frac{TP}{P} = \frac{TP}{FN + TP}$$

|           |   | T  | F  |
|-----------|---|----|----|
| Observado | T | TP | FN |
|           | F | FP | TN |

- Supongamos que queremos predecir  $y = 1$  (cáncer), solo si estamos muy seguros de que lo tienen
  - Hemos disminuido los Falsos Positivos => mayor precisión (-FP)
  - Menos recall => teniendo cáncer les comunicamos que no lo tienen (+ FN)

# Scores basados en CM

- Precision-Recall plot

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

- Trade off precisión y recall

Regresión logística  $0 \leq h_0(x) \leq 1$

Predice 1 si  $h_0(x) \geq 0.1$

Predice 0 si  $h_0(x) < 0.1$

$$SEN = TPR = REC = \frac{TP}{P} = \frac{TP}{FN + TP}$$

|           |   | T  | F  |
|-----------|---|----|----|
| Observado | T | TP | FN |
|           | F | FP | TN |

– Queremos evitar que se nos escapen casos de cáncer

- Hemos aumentado los Falsos Positivos => menor precisión (+FP)
- Mayor recall => menos casos detectados incorrectamente (- FN)

# Scores basados en CM

## Precision-Recall plot

Se obtienen pares de valores de PRE y REC para diferentes *thresholds* que cubran la totalidad del rango  $[0, 1]$  para ambos *scores*

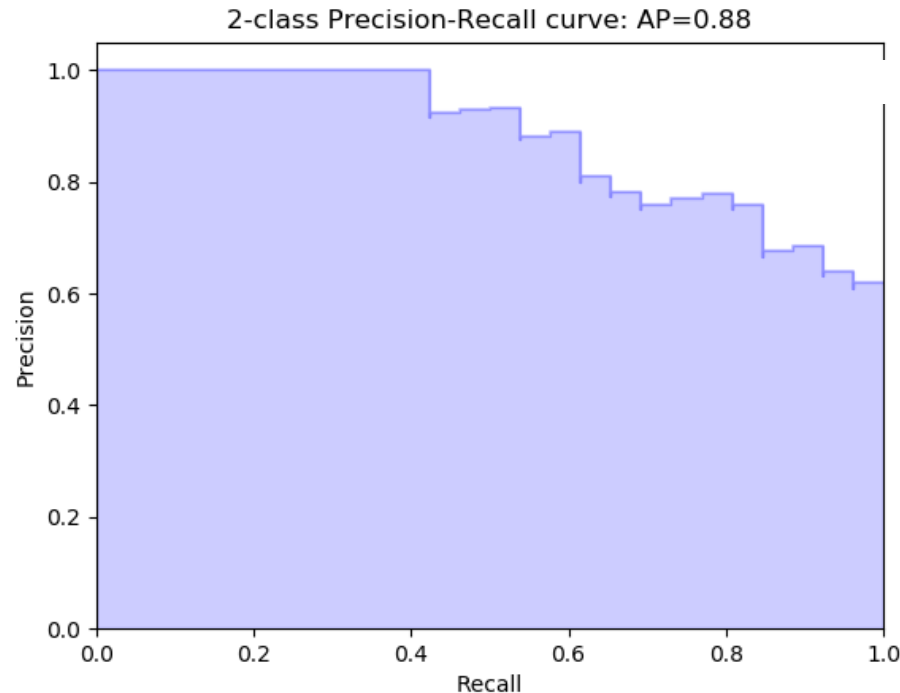


# Scores basados en CM

## Precision-Recall plot & Average Precision score (AP)

Se obtienen pares de valores de PRE y REC para diferentes *thresholds* que cubran la totalidad del rango [0, 1] para ambos scores (izda)

Se plasman en una gráfica, llamada ***precision-recall plot*** los pares para diferentes *thresholds*



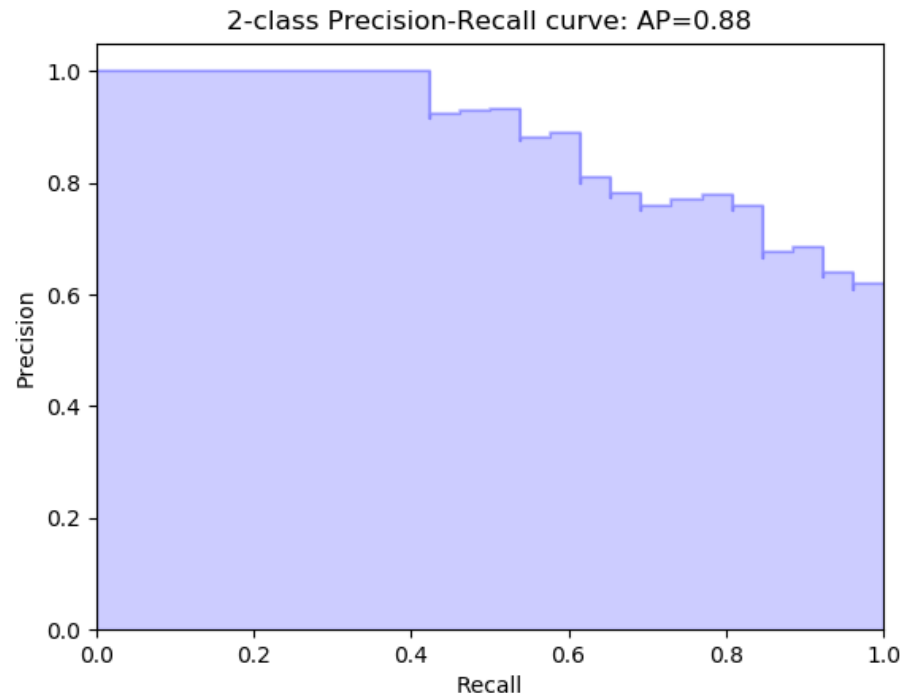
# Scores basados en CM

## Precision-Recall plot & Average Precision score (AP)

Se obtienen pares de valores de PRE y REC para diferentes *thresholds* que cubran la totalidad del rango [0, 1] para ambos scores (izda)

Se plasman en una gráfica, llamada ***precision-recall plot*** los pares para diferentes *thresholds*

El área bajo esa curva poligonal a trozos se llama ***Average Precision score***



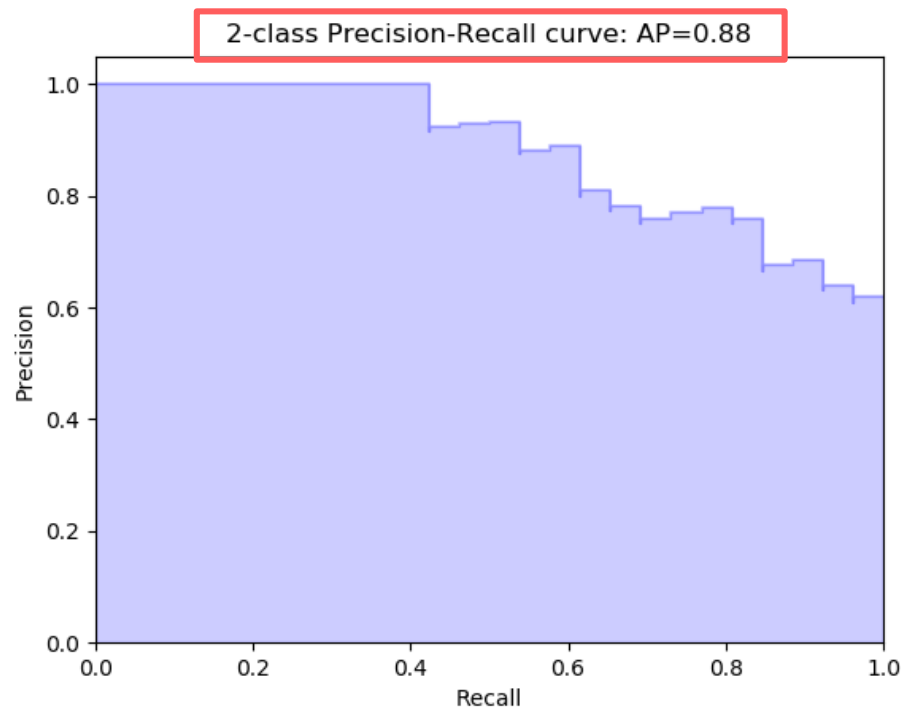
# Scores basados en CM

## Precision-Recall plot & Average Precision score (AP)

Se obtienen pares de valores de PRE y REC para diferentes *thresholds* que cubran la totalidad del rango [0, 1] para ambos scores

Se plasman en una gráfica, llamada ***precision-recall plot*** los pares para diferentes *thresholds*

El área bajo esa curva poligonal a trozos se llama ***Average Precision score***



# Métricas de clasificación en scikit-learn

|  |  |
|--|--|
| <code>metrics.accuracy_score (y_true, y_pred[, ...])</code>    | Accuracy classification score.   |
| <code>metrics.auc (x, y[, reorder])</code>                     | Compute Area Under the Curve (AUC) using the trapezoidal rule                                    |
| <code>metrics.average_precision_score (y_true, y_score)</code> | Compute average precision (AP) from prediction scores  |
| <code>metrics.balanced_accuracy_score (y_true, y_pred)</code>  | Compute the balanced accuracy  |
| <code>metrics.brier_score_loss (y_true, y_prob[, ...])</code>  | Compute the Brier score.   |
| <code>metrics.classification_report (y_true, y_pred)</code>    | Build a text report showing the main classification metrics                                      |
| <code>metrics.cohen_kappa_score (y1, y2[, labels, ...])</code> | Cohen's kappa: a statistic that measures inter-annotator agreement.                              |
| <code>metrics.confusion_matrix (y_true, y_pred[, ...])</code>  | Compute confusion matrix to evaluate the accuracy of a classification                            |
| <code>metrics.f1_score (y_true, y_pred[, labels, ...])</code>  | Compute the F1 score, also known as balanced F-score or F-measure                                |
| <code>metrics.fbeta_score (y_true, y_pred, beta[, ...])</code> | Compute the F-beta score   |
| <code>metrics.hamming_loss (y_true, y_pred[, ...])</code>      | Compute the average Hamming loss.  |
| <code>metrics.hinge_loss (y_true, pred_decision[, ...])</code> | Average hinge loss (non-regularized)   |
| <code>metrics.jaccard_similarity_score (y_true, y_pred)</code> | Jaccard similarity coefficient score   |
| <code>metrics.log_loss (y_true, y_pred[, eps, ...])</code>     | Log loss, aka logistic loss or cross-entropy loss.   |
| <code>metrics.matthews_corrcoef (y_true, y_pred[, ...])</code> | Compute the Matthews correlation coefficient (MCC)   |
| <code>metrics.precision_recall_curve (y_true, ...)</code>      | Compute precision-recall pairs for different probability thresholds                              |
| <code>metrics.precision_recall_fscore_support (...)</code>     | Compute precision, recall, F-measure and support for each class                                  |
| <code>metrics.precision_score (y_true, y_pred[, ...])</code>   | Compute the precision  |
| <code>metrics.recall_score (y_true, y_pred[, ...])</code>      | Compute the recall   |
| <code>metrics.roc_auc_score (y_true, y_score[, ...])</code>    | Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. |
| <code>metrics.roc_curve (y_true, y_score[, ...])</code>        | Compute Receiver operating characteristic (ROC)  |
| <code>metrics.zero_one_loss (y_true, y_pred[, ...])</code>     | Zero-one classification loss.  |



# Clustering

To be seen...



**Mondragon  
Unibertsitatea**

Goi Eskola  
Politeknikoa

**Aitor Agirre / Carlos Cernuda**

aaguirre@mondragon.edu