Métricas de Clasificación
Classifications Metrics

En cualquier proyecto de Machine Learning (o Aprendizaje Automático), **una vez entrenado un algoritmo,** siempre llega la parte de **evaluarlo**, y hacernos la pregunta de ¿**cómo sabemos si nuestro modelo generaliza bien y no hace overfitting?** En este caso trataremos la evaluación de **algoritmos de clasificación**, tarea que se engloba dentro del Aprendizaje Supervisado.

Matriz de confusión

Una vez se ha entrenado un modelo, podemos **obtener predicciones** sobre unos datos. En el caso de un problema binario, nuestro modelo **clasificaría como 0 o 1** un conjunto de datos, y a partir de esto podemos ya crear la **matriz de confusión**.

	1	0
1	TP	FP
0	FN	TN

Métricas de Clasificación
Classifications Metrics

En esta tabla, los números de la **primera columna, representan los valores predichos**, mientras que los valores de la **primera fila, representan los valores reales.**

TP (True Positive) – Son los valores que el algoritmo clasifica como positivos y que realmente son positivos.

TN (True Negative) – Son valores que el algoritmo clasifica como negativos (0 en este caso) y que realmente son negativos.

FP (False Positive) – Falsos positivos, es decir, valores que el algoritmo clasifica como positivo cuando realmente son negativos.

FN (False Negative) – Falsos negativos, es decir, valores que el algoritmo clasifica como negativo cuando realmente son positivos.

Esta matriz es la base sobre la que se construyen todas las métricas de clasificación.

Lo veremos mejor con un ejemplo práctico.

Valor Real	Predicción
1	1
1	1
1	0
0	1
1	0
0	0

Métricas de Clasificación
Classifications Metrics

La **primera columna representa los valores reales**, es decir los que debería haber predicho el modelo, mientras que la **segunda columna representa los valores de la predicción del modelo**. Como podemos ver, los valores no son los mismos para las dos columnas, por lo que nuestro modelo no predice todo correctamente. A partir de esta tabla podríamos crear la siguiente matriz de confusión.

	1	0
1	TP = 2	FP = 1
0	FN = 2	TN = 1

La librería Scikit-Learn proporciona funciones para obtener una **matriz de confusión de forma sencilla.**

```
from sklearn.metrics import confusion_matrix

y_true = [1, 1, 1, 0, 1, 0]

y_pred = [1, 1, 0, 1, 0, 0]

tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
```

Métricas de Clasificación
Classifications Metrics
Principales Métricas de clasificación

A partir de estos valores que obtenemos de la matriz de confusión, podemos obtener diferentes métricas que nos permitirán evaluar nuestro modelo. A continuación hablaremos de las métricas más comunes.

Accuracy

La traducción de accuracy al español sería **precisión**, pero ya que hay otra métrica que también tiene la misma traducción, he optado por dejarlo en inglés. La métrica accuracy representa **el porcentaje total de valores correctamente clasificados, tanto positivos como negativos**.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Es recomendable utilizar esta métrica en problemas en los que los **datos están balanceados,** es decir, que haya **misma cantidad de valores de cada etiqueta** (en este caso mismo número de 1s y 0s).

Utilizando el ejemplo anterior, tendríamos un accuracy de 3/6, es decir 50%. Utilizando Scikit-Learn podemos calcularlo de forma muy sencilla.

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
```

Precisión

La métrica de precisión e**s utilizada para poder saber qué porcentaje de valores que se han clasificado como positivos son realmente positivos.**

Precision = TP / (TP + FP)

Siguiendo con el mismo ejemplo, tendríamos una precision de 2/3, es decir 66,6%. Calcular esta métrica también es muy sencillo.

```
from sklearn.metrics import precision_score

precision_score(y_true, y_pred)
```

Métricas de Clasificación
Classifications Metrics

Recall

La métrica de recall, t**ambién conocida como el ratio de verdaderos positivos**, es utilizada para saber **cuantos valores positivos son correctamente clasificados.**

Recall = TP / (TP + FN)

Siguiendo el ejemplo, tendríamos un recall de 2/4, es decir 50%.

```
from sklearn.metrics import recall_score

recall_score(y_true, y_pred)
```

F1 Score

Esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está **desbalanceado**. Esta métrica combina el **precision y el recall, para obtener un valor mucho más objetivo**.

F1 = 2 * ((recall * precision)/(recall + precision))

Siguiendo el mismo ejemplo, tendríamos un F1 de 2 * ((0,50 * 0,666)/(0,50 + 0,666)), que da como resultado 57,1%.

```
from sklearn.metrics import f1_score

f1_score(y_true, y_pred)
```

Métricas de Clasificación
Classifications Metrics

Curva ROC

Una curva ROC (Receiver Operating Characteristic) es un gráfico muy utilizado para evaluar modelos de Machine Learning para **problemas de clasificación.**

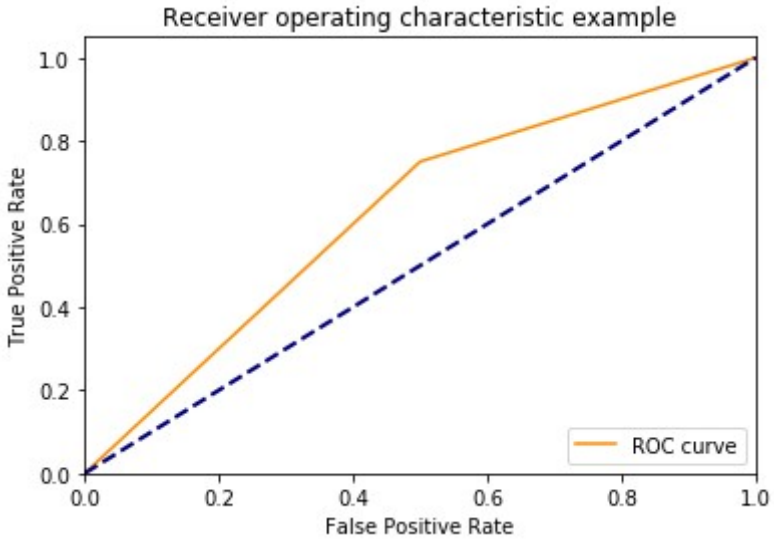
La gráfica representa el **porcentaje de verdaderos positivos (True Positive Rate), también conocido como Recall, contra el ratio de falsos positivos (False Positive Rate).**

La diferencia con el resto de métricas, es que en este caso, **el umbral por el que se clasifica un elemento como 0 o 1, se va modificando, para poder ir generando todos los puntos de la gráfica**.

En este caso, vamos a modificar un poco las predicciones, para que se pueda apreciar de mejor forma la gráfica.

```
import matplotlib.pyplot as plt

from sklearn.metrics import roc_curve

y_true = [1, 1, 1, 0, 1, 0]

y_pred = [1, 1, 0, 1, 1, 0]

fpr, tpr, thresholds = roc_curve(y_true, y_pred)

plt.figure()

plt.plot(fpr, tpr, color='darkorange',

label='ROC curve')

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic example')

plt.legend(loc="lower right")

plt.show()
```

Métricas de Clasificación
Classifications Metrics



AUC

A partir de la gráfica anterior, se puede obtener una métrica sólida, muy útil para problemas de clasificación binaria.

Esta métrica es el AUC (Area Under Curve), que significa, el **área bajo la curva**.

El valor de esta métrica se encuentra en un rango entre 0 y 1, donde 0 es como si tuviéramos un modelo aleatorio, es decir, que si tiraramos los dados al aire tendríamos mejor resultado, y 1 es un resultado óptimo que indica que nuestro modelo generaliza muy bien.

Calcular esta métrica no es tan sencillo como las otras, ya que entraría en juego el cálculo de integrales, por lo que podemos utilizar de nuevo la librería Scikit-Learn para calcularlo de forma sencilla.

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test, y_pred)
```

Métricas de Clasificación
Classifications Metrics
Repaso breve de las métricas de clasificación

En este post hemos visto diferentes métricas para evaluar un algoritmo de clasificación. Nos hemos centrado en problemas de clasificación binaria, pero todos son adaptables a problemas de múltiples clases.

¿qué métrica es la mejor? La respuesta es que no hay una verdad absoluta.

Si tienes datos **desbalanceados**, la mejor métrica sería un **F1-score**, ya que el **Accuracy y AUC dan mayor relevancia a los datos que aparecen más**.

Todas estas métricas se encuentran implementadas en sklearn, así que si no quieres preocuparte de si lo estás calculando bien, sólo tienes que llamar a sus respectivas funciones.