Ingeniería en Inteligencia Artificial,
Machine Learning
Sem: 2025-1, 5BM1, Práctica 5, Fecha: 20 de Octubre 2024

Laboratorio 5: medidas de desempeño en ML

Machine Learning

Grupo: 5BM1

Profesor: Andrés Floriano García

Integrantes:
Juan Manuel Alvarado Sandoval
Alexander Iain Crombie Esquinca
Herrera Saavedra Jorge Luis
Quiñones Mayorga Rodrigo

Contents

1 Introducción a la matriz de confusión y métricas de evaluación

La matriz de confusión es una herramienta esencial en el campo del aprendizaje automático para evaluar el desempeño de los modelos de clasificación. Esta matriz permite visualizar el rendimiento de un modelo al comparar las predicciones realizadas con los resultados reales, organizando la información en cuatro categorías principales: Verdaderos Positivos (TP), Verdaderos Negativos (TN), Falsos Positivos (FP) y Falsos Negativos (FN). A partir de esta matriz, es posible calcular diversas métricas que proporcionan una evaluación más detallada del modelo, tales como la *Precision*, el *Recall*, el *F1-Score*, y la *Accuracy*, entre otras.

Estas métricas no solo permiten identificar qué tan acertado es un modelo, sino también comprender cómo se comporta en distintas situaciones, como cuando se enfrenta a clases desbalanceadas o cuando las consecuencias de los errores difieren en gravedad. Por ejemplo, la *Precision* se enfoca en la proporción de predicciones positivas que realmente son correctas, mientras que el *Recall* evalúa la capacidad del modelo para identificar correctamente todos los ejemplos positivos. La combinación de estas métricas ayuda a tener una visión integral de la calidad de las predicciones, permitiendo ajustes en los umbrales de decisión o modificaciones en el diseño del modelo para mejorar su desempeño.

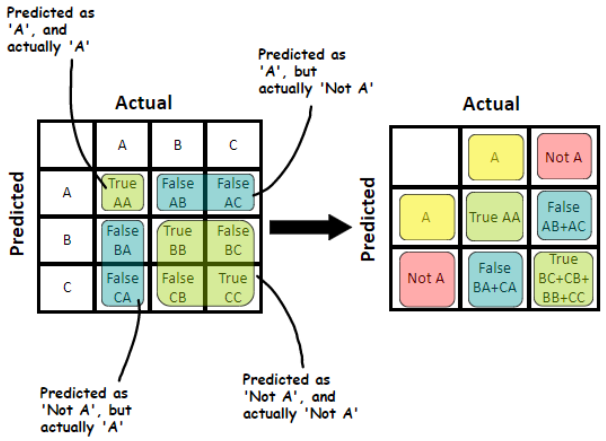

Figure 1: Matriz de confusión de 3 clases.

2 Métricas de evaluación basadas en la matriz de confusión

A partir de la matriz de confusión, se pueden derivar varias métricas clave que permiten cuantificar el rendimiento de un clasificador. Algunas de las métricas más relevantes son:

- **Precision** (Valor Predictivo Positivo): Indica la proporción de predicciones positivas que son correctas. Se define como:

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

- **Recall** (Tasa de Verdaderos Positivos o Sensibilidad): Mide la capacidad del modelo para identificar correctamente las instancias positivas. Se calcula como:

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

- **True Positive Rate** (Tasa de Verdaderos Positivos o Sensibilidad): Representa la proporción de verdaderos positivos en relación con todos los ejemplos reales positivos.

- **True Negative Rate** (Tasa de Verdaderos Negativos o Especificidad): Mide la capacidad del modelo para identificar correctamente las instancias negativas. Se calcula como:

$$\text{True Negative Rate} = \frac{TN}{TN + FP}$$

- **False Positive Rate** (Tasa de Falsos Positivos): Indica la proporción de instancias negativas que fueron incorrectamente clasificadas como positivas:

$$\text{False Positive Rate} = \frac{FP}{FP + TN}$$

- **False Negative Rate** (Tasa de Falsos Negativos): Representa la proporción de instancias positivas que fueron clasificadas incorrectamente como negativas:

$$\text{False Negative Rate} = \frac{FN}{TP + FN}$$

- **F1-Score**: Es la media armónica entre la *Precision* y el *Recall*, proporcionando un balance entre ambas métricas:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Accuracy**: Mide la proporción de predicciones correctas sobre el total de instancias:

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

- **Error Rate**: Representa la tasa de error del modelo, es decir, la proporción de predicciones incorrectas:

$$\text{Error Rate} = 1 - \text{Accuracy}$$

3 Evidencia de la Práctica

3.1 Parte 1

```python
def confusion_matrix2(true, pred):
    # Identificamos las clases únicas
    classes = sorted(set(true + pred))  # Ordenamos las clases para mantener el orden ascendente
    num_classes = len(classes)

    # Creamos la matriz de confusión
    mat = np.zeros((num_classes, num_classes))

    # Creamos un mapeo de las clases a índices (por si las clases no empiezan en 0 o 1)
    class_to_index = {cls: idx for idx, cls in enumerate(classes)}

    # Recorremos las etiquetas verdaderas y predichas al mismo tiempo
    for t, p in zip(true, pred):
        true_idx = class_to_index[t]    # Índice de la clase verdadera
        pred_idx = class_to_index[p]    # Índice de la clase predicha
        mat[true_idx, pred_idx] += 1    # Actualizamos la matriz de confusión

    return mat

#ejemplo de implementación de la matriz:

true = [1, 0, 1, 1, 0, 0, 1]  # etiquetas verdaderas
predicted = [1, 0, 1, 0, 0, 1, 1]  # etiquetas predichas

conf=confusion_matrix2(true, predicted)
print(conf)
```

```
[[2. 1.]
 [1. 3.]]
```

Figure 2: Implementación de la función sin bibliotecas.

```python
def calculate_tp_tn_fp_fn(conf_matrix):
    # Asegurarnos que conf_matrix es un array de numpy
    if not isinstance(conf_matrix, np.ndarray):
        raise ValueError("conf_matrix debe ser un array de numpy")

    num_classes = conf_matrix.shape[0]  # Número de clases
    tp = np.zeros(num_classes)
    tn = np.zeros(num_classes)
    fp = np.zeros(num_classes)
    fn = np.zeros(num_classes)

    # Calcular TP, TN, FP, FN para cada clase
    for i in range(num_classes):
        tp[i] = conf_matrix[i, i]  # True Positives para la clase i, la diagonal

        # False Positives para la clase i: suma de la columna i excepto el valor diagonal
        fp[i] = np.sum(conf_matrix[:, i]) - conf_matrix[i, i]

        # False Negatives para la clase i: suma de la fila i excepto el valor diagonal
        fn[i] = np.sum(conf_matrix[i, :]) - conf_matrix[i, i]

        # True Negatives para la clase i: suma de todos los elementos fuera de la fila i y la columna i
        tn[i] = np.sum(conf_matrix) - (tp[i] + fp[i] + fn[i])

    # Sumamos los valores de TP, TN, FP, FN para obtener el total
    return sum(tp), sum(tn), sum(fp), sum(fn)
```

Figure 3: Función que calcula tp,tn,fp y fn sin bibliotecas.

```python
# Calcular métricas

num_classes = conf.shape[0]  # Número de clases
accuracy = (tp + tn) / (tp + tn + fp + fn)
error= 1-accuracy
precision= tp/(tp+fp)
recall= tp/(tp+fn)
ppv= tp/(tp+fp)
tpr= tp/(tp+fn)
tnr= tn/(tn+fp)
fpr= fp/(fp+tn)
fnr= fn/(fn+tp)
f1= (2*tp)/(2*tp+fp+fn)
print(f"Classes: {num_classes:.1f}")
print(f"Accuracy: {accuracy:.4f}")
print(f"Error: {error:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"PPV: {ppv:.4f}")
print(f"TPR: {tpr:.4f}")
print(f"TNR: {tnr:.4f}")
print(f"FPR: {fpr:.4f}")
print(f"FNR: {fnr:.4f}")
print(f"F1 score: {f1:.4f}")
```

```
fp: 2.0 tn 5.0 fp: 2.0 fn 2.0

Classes: 2.0
Accuracy: 0.7143
Error: 0.2857
Precision: 0.7143
Recall: 0.7143
PPV: 0.7143
TPR: 0.7143
TNR: 0.7143
FPR: 0.2857
FNR: 0.2857
F1 score: 0.7143
```


Figure 4: Cálculo de metricas obtenidas sin bibliotecas

3.2 Parte 2

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

Figure 5: Métricas disponibles en librerias que ofrece python .

```
Accuracy: 0.7988826815642458
Error: 0.2011173184357542
Precision: 0.7638888888888888
Recall: 0.7432432432432432
F1-Score: 0.7534246575342466
TPR: 0.7432432432432432
TNR: 0.8380952380952381
FPR: 0.1619047619047619
FNR: 0.25675675675675674
PPV: 0.7638888888888888
```

Figure 6: Impresión de métricas para la clasificación obtenidas mediante métodos de librerias.

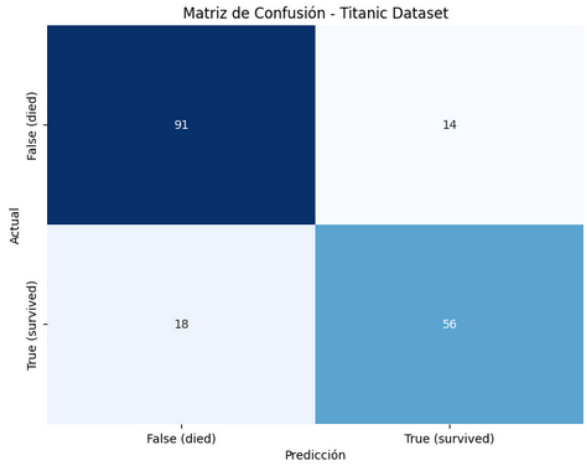

Figure 7: Matriz de confusión desplegada de matplotlib para titanic-dataset.

4 Enlace al repositorio

Haz click para seguir el enlace

5 Referencias

- Will Cukierski. (2012). *Titanic - Machine Learning from Disaster*. Kaggle. Recuperado de: https://kaggle.com/competitions/titanic