Autor:
Iván García Santillán

Algoritmos predictivos de clasificación y evaluación de modelos

Introducción

En esta unidad se utiliza herramientas y conjuntos de datos públicos para el entrenamiento de algunos algoritmos de aprendizaje automático supervisado (clasificación) siguiendo una metodología para el desarrollo de proyectos de análisis de datos. Los algoritmos de clasificación que abordaremos son árboles de decisión y Redes neuronales artificiales para problemas de clasificación binaria y/o multi-clasificación, todos ellos utilizando el lenguaje Python y el framework TensorFlow-Keras y otras librerías como Scikit-Learn. Además, se enfatizará en la evaluación de los algoritmos utilizando métricas y gráficas de rendimiento de los modelos entrenados.

Palabras clave

Aprendizaje automático, árboles de decisión, redes neuronales artificiales, evaluación de modelos.

Reto

¿Qué métricas y gráficas resultan más adecuados para evaluar el rendimiento de algoritmos de clasificación binaria/multiclase?

Desarrollo

3. Algoritmos predictivos y evaluación de modelos

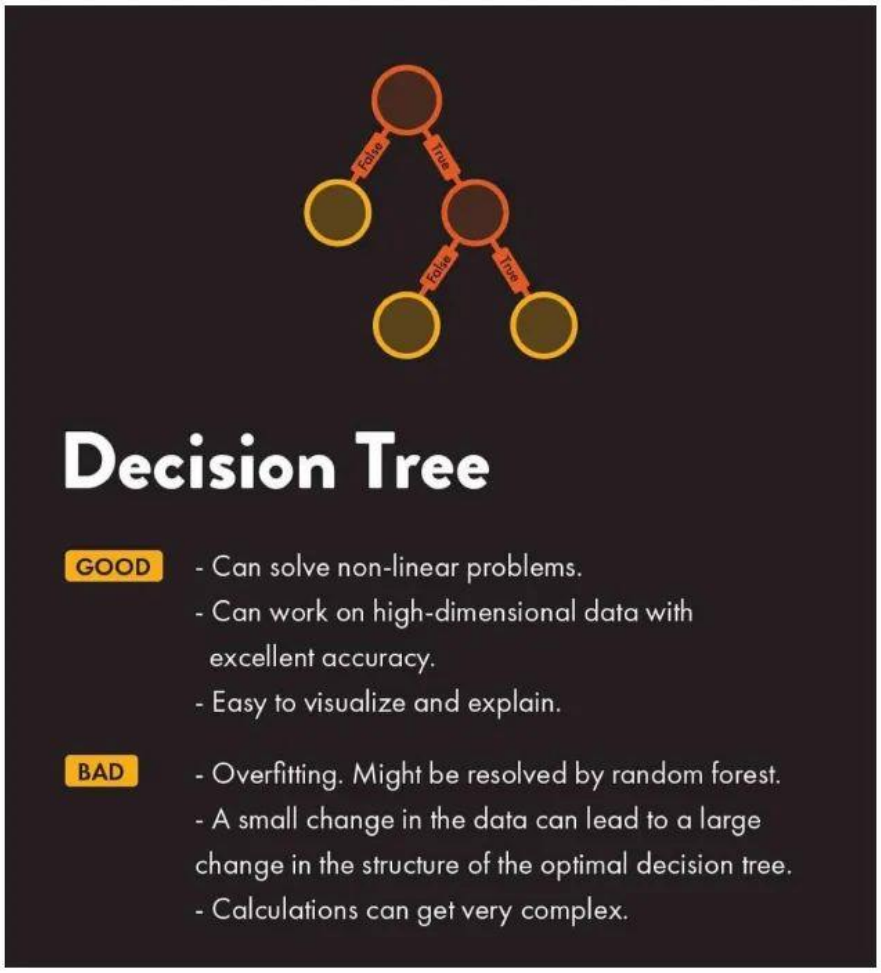
3.1 Algoritmos predictivos de clasificación (continuación)

3.1.1 Árbol de decisión

Es un método de aprendizaje automático utilizado para la toma de decisiones (o hacer predicciones) y la resolución de problemas de clasificación y regresión. Los árboles de decisión tienen una amplia gama de aplicaciones prácticas. En el sector financiero, se utilizan para evaluar el riesgo de crédito de los clientes y en el trading para tomar decisiones de inversión basadas en patrones históricos. En el ámbito de la salud, ayudan a diagnosticar enfermedades al clasificar los síntomas y signos clínicos. En el comercio minorista y marketing son útiles para segmentar clientes y predecir su comportamiento de compra, lo que facilita la personalización de estrategias de marketing. También desempeñan un papel crucial en el control de la calidad en la fabricación, identificando factores clave que influyen en los defectos del producto. Su popularidad se debe a su facilidad de interpretación y efectividad, ya que los árboles de decisión pueden manejar tanto datos numéricos como categóricos y son capaces de modelar interacciones complejas entre variables.

En la Figura 16 se indican algunos pros y contras del algoritmo de árbol de decisión.

Figura 16. Pros y contras del algoritmo de árbol de decisión



Fuente: ML4Devs (2020)

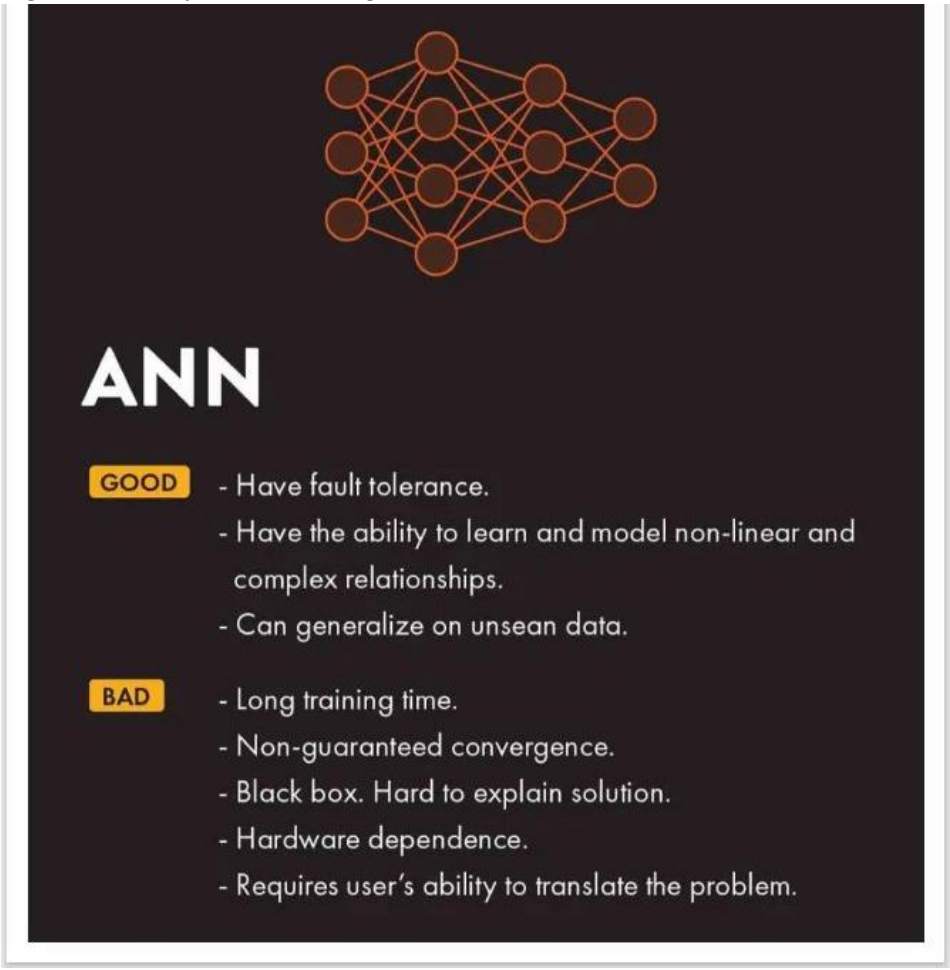
3.1.2 Redes neuronales artificiales

Una Red Neuronal Artificial (RNA), o artificial neural network (ANN), es un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Está compuesta por unidades de procesamiento llamadas neuronas artificiales o nodos, que están organizadas en capas interconectadas. Las redes neuronales artificiales son utilizadas para abordar una amplia variedad de tareas de aprendizaje automático, incluyendo clasificación, regresión, agrupamiento, procesamiento de imágenes, procesamiento de lenguaje natural y más. Por ejemplo, en el campo de la visión por computadora, son esenciales para el reconocimiento de imágenes y rostros, permitiendo innovaciones como los

sistemas de seguridad biométrica y las aplicaciones de filtros en redes sociales. En el procesamiento del lenguaje natural, impulsan traductores automáticos y asistentes virtuales, facilitando la comunicación y la interacción con la tecnología. En el sector financiero, estas redes se utilizan para la predicción de series temporales en mercados bursátiles y para la detección de fraudes, analizando patrones complejos en transacciones. En medicina, contribuyen significativamente en el diagnóstico y pronóstico de enfermedades, analizando datos médicos y patrones en imágenes diagnósticas como radiografías y resonancias magnéticas. Además, en el transporte y ámbito del vehículo autónomo, son fundamentales para la interpretación de los datos sensoriales y la toma de decisiones en tiempo real.

En la Figura 17 indican algunos pros y contras del algoritmo ANN.

Figura 17. Pros y contras del algoritmo ANN



Fuente: ML4Devs (2020)

3.2 Evaluación de modelos predictivos y métricas de rendimiento

Estimar la bondad de un clasificador sirve para medir su capacidad de predicción sobre nuevas instancias (generalización). La validación se realiza brevemente basándose en la tasa de error, entendido como la clasificación incorrecta.

Tasa de error = # de errores cometidos / # total de casos

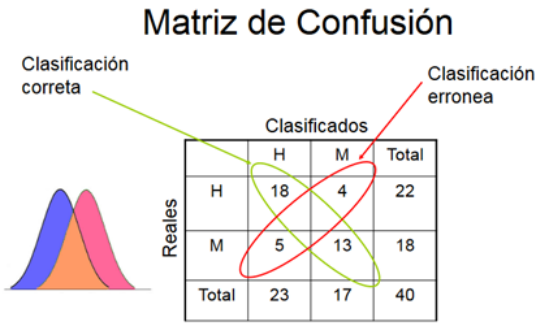
Una **matriz de confusión** permite ver, mediante una tabla de contingencias, la distribución de los errores (tipo I o falsos positivos y tipo II o falsos negativos) cometidos por un clasificador a lo largo de las distintas categorías del problema, ver Figura 18. De una matriz de confusión se pueden extraer otros conceptos enriquecedores a la hora de comprender la distribución y naturaleza de los errores cometidos por el clasificador, así como algunas otras métricas de evaluación como: *exactitud (accuracy), precisión (precision), sensibilidad (recall), especificidad (specificity), F1-score, Curva ROC, área bajo la curva (AUC), coef. Kappa, coef. de determinación R^2.*

Figura 18. Matriz de confusión de un clasificador binario

Fuente: UAB (2016)

En la Figura 19 se muestra un ejemplo de un clasificador binario y la tasa de error cometido.

Figura 19. Ejemplo de matriz de confusión binaria



Tasa de error global = (5 + 4) / 40 = 0.225

= 22.5%

Entendiendo los términos TP, FP, FN, TN:

- Verdadero positivo (TP): el modelo predijo positivo y es cierto (asignaciones positivas correctas)
- Verdadero Negativo (TN): El modelo predijo negativo y es cierto (asignaciones negativas correctas)
- Falso positivo (FP, error tipo 1): el modelo predijo positivo y es falso (asignaciones positivas incorrectas)
- Falso negativo (FN, error tipo 2): el modelo predijo negativo y es falso (asignaciones negativas incorrectas).

En un problema de clasificación binario, se denomina como la clase positiva (P ó 1) a los ejemplos que deseamos ser capaces de identificar. Los demás ejemplos los asignaremos a la clase negativa (N ó 0). Ejemplos:

- Drowsiness detection: drowsy (1), non-drowsy (0)
- Breast cancer detection: malignant (1), benign (0)
- Fraudulent transaction detector: fraud (1), non-fraud (0)
- Spam filter: spam (1), non-spam (0)

Métricas de evaluación a partir de la matriz de confusión 2×2:

- **Exactitud (*accuracy*):** calcula la tasa de aciertos totales (predicciones correctas) y da como resultado la eficacia general del algoritmo.
- **Precisión (*precision*):** calcula la relación entre las predicciones positivas correctas y los pronósticos positivos totales. Indica la calidad de la respuesta del clasificador.

- **Sensibilidad (*Recall o TPR -True Positive Rate-*):** calcula la tasa de positivos verdaderos (TP) e indica la eficiencia en la clasificación de todos los elementos que son de la clase positiva (+).
- **Especificidad:** calcula la tasa de negativos verdaderos (TN) e indica la eficiencia en la clasificación de todos los elementos que son de la clase negativa (-).
- **Medida F1:** conocida como media armónica de precisión y sensibilidad. Calcula el equilibrio entre la precisión y la sensibilidad, que varía de 0 a 1, siendo 1 el mejor valor. especialmente cuando se trata de promediar ratios o tasas. La media armónica se utiliza principalmente en situaciones donde las medias aritméticas no son adecuadas, especialmente cuando se desea promediar tasas o ratios. Se define como:

$$Media\ armónica = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

En la Tabla 5 se presenta la matriz de confusión para la detección de somnolencia. Fíjese que la clase negativa está en la primera columna, a diferencia de la mostrada en la Figura 18. Ambas formas de organización son usadas y se deben considerar en el momento de calcular las métricas de evaluación de un algoritmo.

Tabla 5. Estructura de la Matriz de confusión para la detección de somnolencia.

		PREDICHO	
		No-Somnolencia	Somnolencia
REAL	No-Somnolencia	TN	FP
	Somnolencia	FN	TP

Las fórmulas para calcular las métricas de evaluación (a partir de la Tabla 5) son las siguientes:

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

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

$$sensitivity\ (o\ recall) = \frac{TP}{\text{TP} + \text{FN}}$$

$$specificity = \frac{TN}{\text{TN} + \text{FP}}$$

$$F1\ score = \frac{2 * precisión * recall}{precisión + recall}$$

En la Tabla 6 se muestra un ejemplo de un clasificador binario para el cálculo de las métricas de evaluación.

Tabla 6. Matriz de confusión para la detección de somnolencia.

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

¿Qué se puede aprender de esta matriz de confusión?

- De esos 165 casos, el clasificador predijo "sí" 110 veces y "no" 55 veces.
- En realidad, 105 pacientes de la muestra padecen la enfermedad y 60 pacientes no padecen.

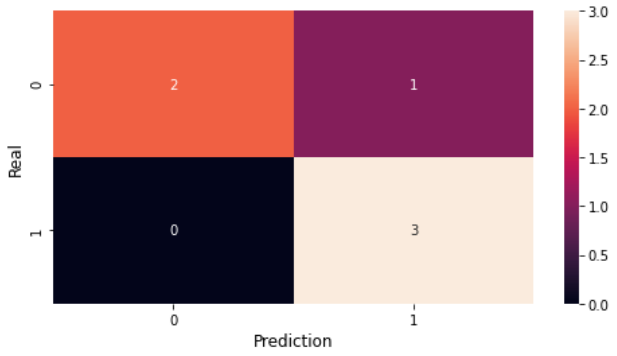
Métricas:

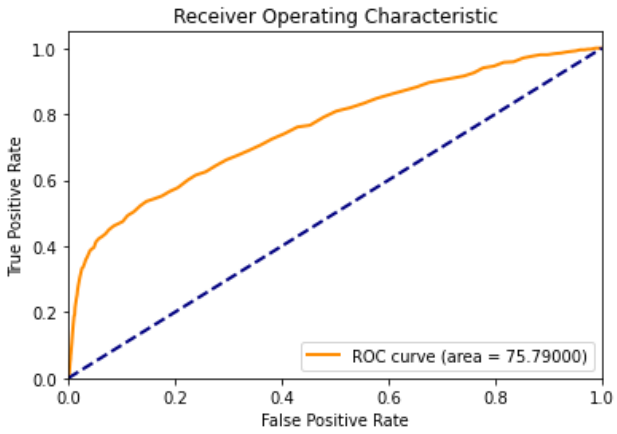
- Accuracy: (TP+TN)/total = (100+50)/165 = 0.91
- Error Rate: (FP+FN)/total = (10+5)/165 = 0.09
- Error rate = 1 - Accuracy
- Precision: TP/predicted yes = 100/110 = 0.91
- Sensitivity" or "Recall" (True Positive Rate): TP/actual yes = 100/105 = 0.95
- Specificity (True Negative Rate:): TN/actual no = 50/60 = 0.83
- F1-score = (2*0.91*0.95)/(0.91+0.95)= 0.93

A continuación, se presenta a modo de ejemplo la evaluación de modelos en Python, la librería Scikit-learn y los resultados obtenidos (Figura 20) en un dataset pequeño conteniendo 6 observaciones:

```
# Dataset:
y_true = [0, 1, 0, 0, 1, 1]
y_pred = [0, 0, 1, 0, 0, 1]
# Matriz de confusión:
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_true, y_pred)
# Exactitud:
from sklearn.metrics import accuracy_score
accuracy_score(y_true, y_pred)
# Sensibilidad:
from sklearn.metrics import recall_score
recall_score(y_true, y_pred)
# Especificidad:
from sklearn.metrics import recall_score
recall_score(y_true, y_pred, pos_label=0)
# Precisión:
from sklearn.metrics import precision_score
precision_score(y_true, y_pred)
# Puntuación F1:
from sklearn.metrics import f1_score
f1_score(y_true, y_pred)
# Área bajo la curva:
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_true, y_pred)
# R Score:(R^2 coefficient of determination)
from sklearn.metrics import r2_score
r2_score(y_true, y_pred)
# ROC Curve (binary classification problem):
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
plt.plot(roc_curve(y_true, y_pred)[0], roc_curve(y_true, y_pred)[1], color='darkorange',lw=2,
label='ROC curve (area = %0.2f)' % auc)
```

Figura 20. Resultados del Código Python mostrando la matriz de confusión y la curva ROC.





La matriz de confusión y la curva ROC se puede adaptar para problemas de clasificación con más de 2 clases. Sin embargo, esto queda como trabajo autónomo del maestrante.

Trabajo autónomo:

Calcule la matriz de confusión y las métricas de evaluación del siguiente detector de somnolencia (2x2) dado en la Tabla 7:

Tabla 7. Valores predichos y reales para un clasificador de somnolencia (10 observaciones)

Actual class	Predicted class
1	1
0	1
0	0
1	1
1	1
0	1
0	0

0	0
1	1
1	0

		Predicted	
		Non-Drowsy (0)	Drowsy (1)
Actual	Non-Drowsy (0)		
	Drowsy (1)		

Conclusiones

En esta unidad se ha mencionado los algoritmos predictivos de clasificación: Árboles de decisión y Redes neuronales artificiales. Ellos tienen un sin número de aplicaciones prácticas, abarcando numerosos sectores e industrias. Sin embargo, para obtener un mejor rendimiento de cada uno de ellos es importante experimentar con los parámetros de cada algoritmo. Además, hay que considerar las ventajas y desventajas que presentan cada uno de ellos. En la Tabla 8 se indican algunos criterios a considerar para la selección adecuada de un algoritmo de clasificación de aprendizaje automático.

Tabla 8. Criterios para la selección de un algoritmo de clasificación

Algorithm	Prediction Speed	Training Speed	Memory Usage	Required Tuning	General Assessment
Logistic Regression (and Linear SVM)	Fast	Fast	Small	Minimal	Good for small problems with linear decision boundaries
Decision Trees	Fast	Fast	Small	Some	Good generalist, but prone to overfitting
(Nonlinear) SVM (and Logistic Regression)	Slow	Slow	Medium	Some	Good for many binary problems, and handles high-dimensional data well
Nearest Neighbor	Moderate	Minimal	Medium	Minimal	Lower accuracy, but easy to use and interpret
Naïve Bayes	Fast	Fast	Medium	Some	Widely used for text, including spam filtering
Ensembles	Moderate	Slow	Varies	Some	High accuracy and good performance for small- to medium-sized datasets
Neural Network	Moderate	Slow	Medium to Large	Lots	Popular for classification, compression, recognition, and forecasting

Fuente: Mathworks (2023)

Además, otra característica deseable es la **interpretabilidad** del modelo, es decir, la capacidad de explicar la predicción o decisión tomada, contribuyendo a que sean confiables, justificables y transparentes. Las ANN y SVM se consideran como cajas negras, mientras que los árboles de decisión, la regresión logística, y K-NN son mucho más interpretables. Un modelo de aprendizaje automático se considera una "**caja negra**" cuando su funcionamiento interno es difícil de interpretar o entender, incluso para expertos. Esto significa que, aunque el modelo pueda hacer predicciones precisas, es complicado explicar cómo y por qué llega a esas predicciones.

Finalmente, se debe intentar lograr un equilibrio entre la precisión (%), el tamaño (MB) y la velocidad de inferencia (ms) del modelo entrenado, considerando en donde será puesto en producción: aplicación Web, móvil, sistema embebido.

ANEXOS

Bibliografía:

Berzal, Fernando (2019). Redes neuronales & Deep Learning. USA: Independently published.

Brownlee, J. (2023). Machine Learning Mastery. https://machinelearningmastery.com/

García-Santillán, I. (2023). Ciencia y analítica de datos. Maestría en Inteligencia Artificial aplicada. Universidad Hemisferios.

InstitutoEmprende (2023). Cómo crear un árbol de decisión. https://institutoemprende.com/como-crear-un-arbol-de-decisiones/

Lara, J. (2014). Minería de Datos. Madrid: CEF-Udima

MathWorks (2023). Machine Learning in Matlab. https://la.mathworks.com/help/stats/machine-learning-in-matlab.html

MaximaFormacion (2023). Qué son los árboles de decisión y para qué sirven. https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/

ML4Devs (2020). An Engineer's Trek into Machine Learning. https://www.ml4devs.com/articles/machine-learning-intro-for-developers/

Sierra, B. (2006). Aprendizaje automático : conceptos básicos y avanzados. Madrid: Prentice-Hall.

Torres, J. (2020). Python Deep Learning: Introducción práctica con Keras y TensorFlow 2. Marcombo.

UAB. (2016). Curso de Detección de objetos. Universidad Autónoma de Barcelona.