Autor:
Iván García Santillán

Algoritmos predictivos de clasificación: Árbol de decisión

Introducción

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 la calidad del control 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. Aunque también se debe considerar algunas desventajas como que pueden ser propensos al sobreajuste (*overfitting*) y una precisión disminuida en ciertos conjuntos de datos.

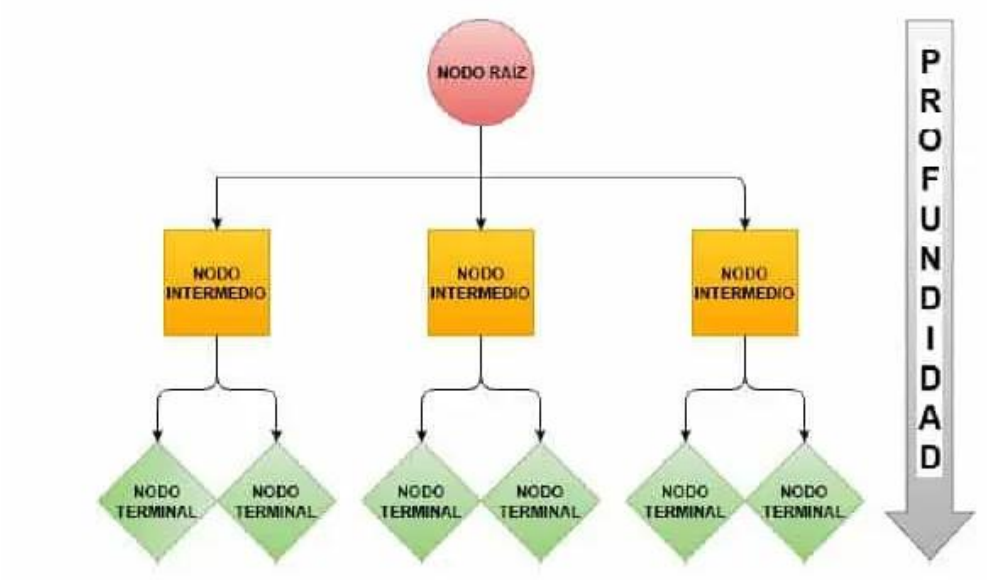
A continuación, se detallan el funcionamiento del algoritmo y un ejemplo en Python, Keras y TensorFlow para la predicción del cáncer de mama utilizando un data set público.

Contenido

El algoritmo de árbol de decisión se basa en la estructura de un árbol, donde cada nodo representa una decisión basada en una característica específica, y las ramas

representan las posibles salidas o resultados de esa decisión, como se indica en la Figura 21.

Figura 21. Estructura básica de un Árbol de decisión



Fuente: MaximaFormacion (2023)

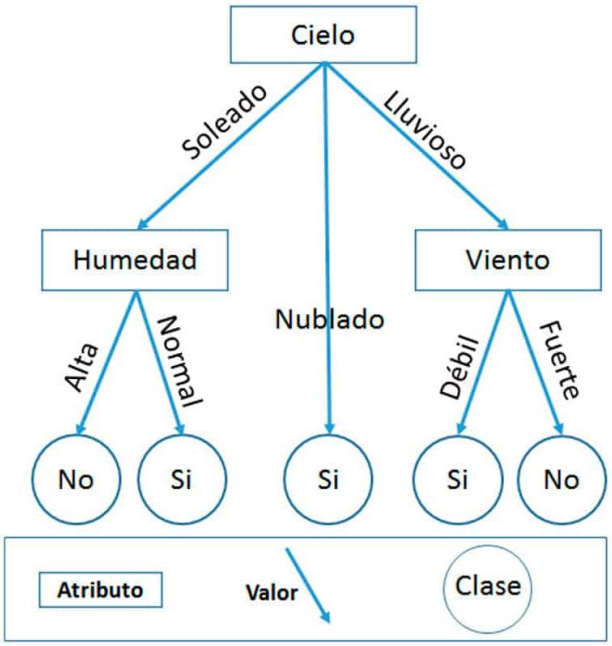
Algunos conceptos clave relacionados con el algoritmo de árbol de decisión son los siguientes:

- Nodo Raíz: El nodo superior del árbol, que representa la característica que mejor divide el conjunto de datos inicial en función de ciertos criterios (como la ganancia de información o la impureza).
- Nodos Internos (intermedio): Los nodos en el medio del árbol que representan decisiones basadas en características específicas.
- Hojas (nodo terminal): Los nodos finales del árbol que representan las etiquetas de clasificación o los valores de regresión resultantes.
- División: El proceso de dividir un nodo en dos o más nodos hijos según una característica específica. La elección de la característica y el criterio de división es un aspecto importante del algoritmo.
- Criterios de Decisión: Los criterios utilizados para decidir cómo dividir los nodos, como la ganancia de información (entropía), la impureza Gini o el error

cuadrático medio (*mse*), dependiendo del tipo de problema (clasificación o regresión).

Un ejemplo de árbol de decisión para predecir si jugar o no un partido de tenis considerando variables atmosféricas se presenta en la Figura 22.

Figura 22. Árbol de decisión para predecir si jugar o no un partido de tenis



Fuente: InstitutoEmprende (2023)

Los árboles de decisión son populares en el aprendizaje automático debido a su simplicidad y capacidad para manejar conjuntos de datos tanto categóricos como numéricos. Además, son fácilmente interpretables y explicables, lo que los hace útiles y confiables en aplicaciones donde la **transparencia** del modelo es importante, tales como el diagnóstico y tratamiento médico, préstamos y créditos, sentencias judiciales y libertad condicional, contratación y evaluación de empleados, Evaluación de riesgos y tarifas de pólizas, segmentación de clientes y recomendaciones de productos, evaluación de Estudiantes y personalización del aprendizaje, entre otros.

A continuación, se presenta un ejemplo del algoritmo de árbol de decisión en Python para la predicción del cáncer de mama (benigno=0, maligno=1)

utilizando el conjunto de datos Breast Cancer Wisconsin (Diagnostic) disponible en la librería scikit-learn.

```
# Decision Tree for breast cancer prediction

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

# Carga el conjunto de datos Breast Cancer
dataset = load_breast_cancer()
X = dataset.data # 569x30
y = dataset.target # 569x1

# Divide el conjunto de datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normaliza los datos para que tengan una escala similar
scaler = MinMaxScaler(feature_range=(0,1)) # [0, 1]
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Crea y entrena el modelo de árbol de decisión
model = DecisionTreeClassifier(max_depth=4, criterion = 'gini')
model.fit(X_train, y_train)

# Realiza predicciones usando el conjunto de prueba
y_pred = model.predict(X_test)

# Convierte las probabilidades en etiquetas binarias (0 o 1)
y_pred = (y_pred > 0.5)

# Muestra el informe de evaluación del modelo entrenado
print(classification_report(y_test, y_pred))

# Matriz de confusión:
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

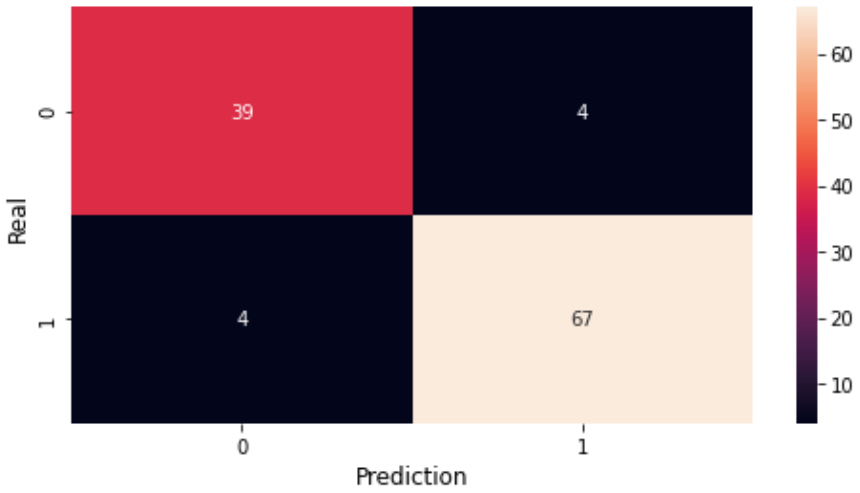
```
cm = confusion_matrix(y_test, y_pred)
print("confusion matrix: \n", cm)
# gráfica de la CM
plt.figure(figsize = (8,4))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Prediction', fontsize = 12)
plt.ylabel('Real', fontsize = 12)
plt.show()

# Exactitud:
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print("accuracy: ", acc)

# Sensibilidad:
from sklearn.metrics import recall_score
recall = recall_score(y_test, y_pred)
print("recall: ", recall)

# Precisión:
from sklearn.metrics import precision_score
precision = precision_score(y_test, y_pred)
print("precision: ", precision)

# Especificidad
# 'specificity' is just a special case of 'recall'.
# specificity is the recall of the negative class
specificity = recall_score(y_test, y_pred, pos_label=0)
print("specificity: ", specificity)

# Puntuación F1:
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred)
print("f1 score: ", f1)

# Área bajo la curva:
from sklearn.metrics import roc_auc_score
auc = roc_auc_score(y_test, y_pred)
print("auc: ", auc)

# Curva ROC
```

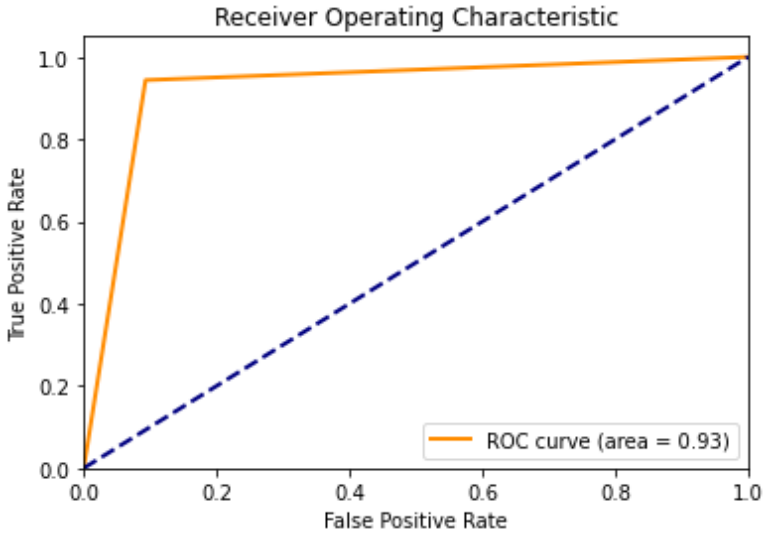
```
from sklearn.metrics import roc_curve
plt.figure()
lw = 2
plt.plot(roc_curve(y_test, y_pred)[0], roc_curve(y_test, y_pred)[1],
color='darkorange',lw=lw, label='ROC curve (area = %0.2f)' %auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()

# R Score (R^2 coefficient of determination)
from sklearn.metrics import r2_score
R = r2_score(y_test, y_pred)
print("R2: ", R)

# Visualizar un árbol de decisión usando matplotlib
from sklearn.tree import plot_tree
# Crear la figura y el eje
fig, ax = plt.subplots(figsize=(12, 8))
# Dibujar el árbol de decisión
plot_tree(model,
          feature_names = dataset.feature_names,
          class_names = dataset.target_names,
          filled=True,
          rounded=True,
          ax=ax)
# Mostrar la gráfica
plt.show()

# Calcular y visualizar la importancia de las variables en la predicción del modelo
importances = model.feature_importances_
# Crear un DataFrame para visualizar las importancias
import pandas as pd
feature_importances = pd.DataFrame({
   'Variable': dataset.feature_names,
   'Importancia': importances
}).sort_values(by='Importancia', ascending=False)
```

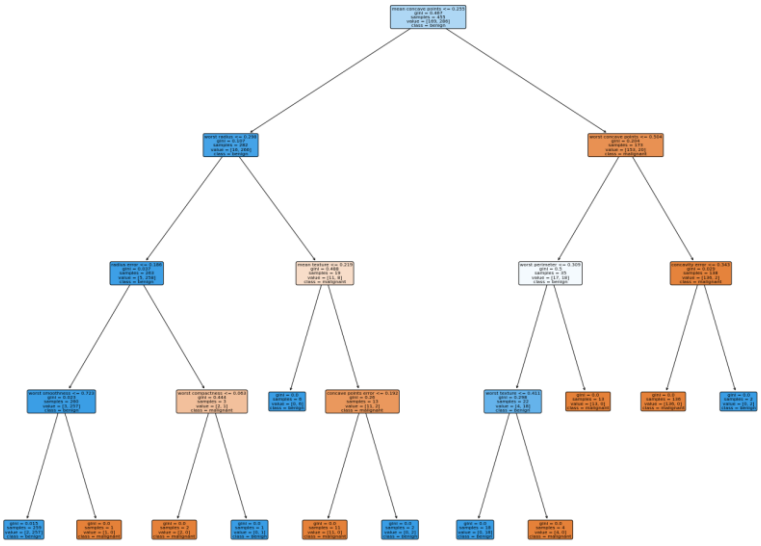
```
print(feature_importances)

# Visualizar las importancias de las variables
plt.figure(figsize=(12, 8))
plt.barh(feature_importances['Variable'], feature_importances['Importancia'])
plt.xlabel('Importancia')
plt.ylabel('Variables')
plt.title('Importancia de las variables')
plt.gca().invert_yaxis()
plt.show()
```

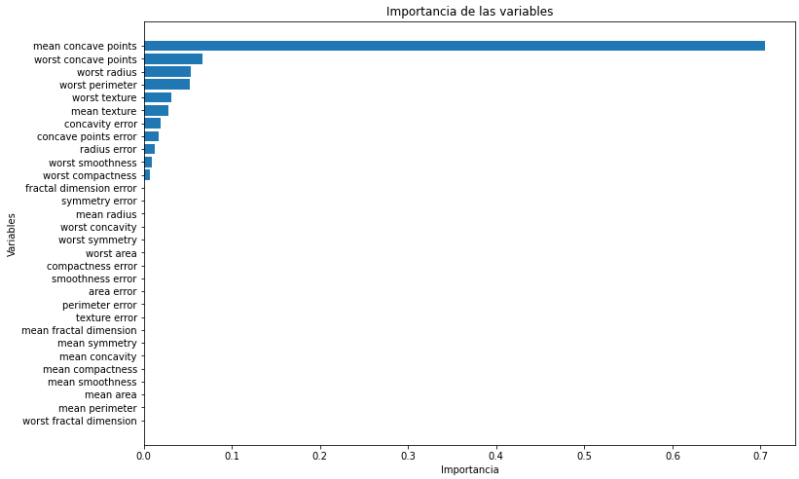
Los resultados del modelo de árbol de decisión se presentan a continuación:

accuracy: 0.93
recall: 0.94
precision: 0.94
specificity: 0.90
f1 score: 0.94
auc: 0.93
R2: 0.70




Receiver Operating Characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0





Un clasificador adecuado es aquel que alcanza una exactitud (accuracy) superior al 85%. Por lo que, este modelo de árbol de decisión entrenado cumple con este criterio de calidad.

En un árbol de decisión, tanto los parámetros como los hiperparámetros juegan roles importantes en la construcción y el ajuste del modelo. Los **parámetros** de un árbol de decisión son los valores aprendidos durante el entrenamiento, estos parámetros incluyen:

- Divisiones (Splits): Puntos de división específicos en cada nodo que determinan cómo se dividen los datos.
- Hojas (Leaves): Las predicciones finales del árbol que representan las clases (en clasificación) o los valores (en regresión).

Los **hiperparámetros** de un árbol de decisión son configurados antes del entrenamiento y afectan cómo se construye el árbol. Los principales hiperparámetros en un árbol de decisión incluyen:

- **criterion**: La función utilizada para medir la calidad de una división.
  gini: Gini impurity (impureza de Gini).
  entropy: Información de entropía.
  mse: Mean squared error (error cuadrático medio) para regresión.
  mae: Mean absolute error (error absoluto medio) para regresión.
- **max_depth**: La profundidad máxima del árbol. Limitar la profundidad ayuda a evitar el sobreajuste.

Tenga en cuenta que se pueden ajustar varios hiperparámetros del algoritmo (como max_depth, criterion, etc.) para obtener un modelo óptimo. Revise la documentación (ayuda en línea) del algoritmo para más detalles y experimente con ellos.

Adicionalmente, el árbol de decisión permite calcular y visualizar la **importancia de cada característica** predictora (*feature*) en la predicción del modelo. En otras palabras, mide la contribución de cada característica a la predicción del modelo. Se normaliza de manera que la suma de todas las importancias es igual a 1.
Se puede usar estos valores para lo siguiente:

- Interpretación del modelo: Identificar cuáles características son más relevantes para la toma de decisiones del modelo.
- Selección de características: Decidir qué características podrían ser eliminadas (si tienen una importancia muy baja) para simplificar el modelo sin una pérdida significativa de precisión.
- Visualización: Crear gráficos de barras que muestren la importancia de cada característica, lo cual es útil para presentar y explicar los resultados del modelo (interpretabilidad).

Finalmente, aunque un árbol de decisión es un modelo fundamental en la caja de herramientas de análisis predictivo, como cualquier otra técnica, presenta tanto ventajas como desventajas que se resumen a continuación:

Ventajas de los árboles de decisión:

Fácil interpretación y visualización: Los árboles de decisión son fáciles de entender y explicar, incluso para usuarios sin un profundo conocimiento técnico. Su representación gráfica, que muestra claramente cómo se toman las decisiones, es intuitiva y útil para la toma de decisiones basada en datos. El árbol se puede visualizar usando algunas librerías como *matplotlib* y *graphviz*. Además, se puede calcular y visualizar la importancia de cada característica predictora en la predicción del modelo.

Manejo de datos no lineales: Pueden capturar relaciones no lineales sin necesidad de transformar las variables, lo que los hace aplicables a una amplia

variedad de datos. El término "modelar relaciones no lineales" se refiere a la capacidad de un modelo estadístico o de aprendizaje automático para capturar y representar relaciones entre variables que no son simplemente proporcionales o directas. En una **relación no lineal**, los cambios en una variable no producen cambios proporcionales y predecibles en otra variable, lo que significa que la relación entre variables no puede ser descrita adecuadamente por una línea recta, sino por otro tipo de relación como cuadrática, exponencial, logarítmica, sinusoidal, etc.

No requieren escalado de características: A diferencia de otros algoritmos que requieren normalización o escalado de datos, los árboles de decisión funcionan bien con características en su escala original.

Pueden manejar tanto variables numéricas como categóricas: Los árboles de decisión pueden manejar datos que incluyan tanto variables numéricas como categóricas sin la necesidad de crear variables ficticias o dummies (label encoding o one-hot encoding).

Robustos a outliers: Los árboles de decisión son bastante robustos a la presencia de outliers, ya que las divisiones se hacen basadas en porcentajes de muestras y no en valores específicos.

Desventajas de los Árboles de Decisión:

Sobreajuste: Una de las mayores desventajas de los árboles de decisión es su tendencia al sobreajuste, especialmente con árboles muy profundos. Los árboles que son demasiado complejos tienden a aprender detalles del conjunto de datos de entrenamiento que no generalizan bien a nuevos datos.

Inestabilidad: Pequeñas variaciones en los datos pueden resultar en la generación de árboles completamente diferentes. Esta inestabilidad se puede mitigar en parte utilizando ensambles de árboles de decisión, como Random Forests o Gradient Boosting (XGBoost).

Problemas con datos desbalanceados: Al igual que otros algoritmos, los árboles de decisión pueden crear sesgos si las clases están muy desbalanceadas.

Dificultad con datos de alta dimensionalidad: Aunque pueden manejar muchas características, su rendimiento puede degradarse si muchas de estas características son irrelevantes o redundantes.

Conclusiones

En esta lectura se ha revisado los fundamentos del algoritmo de Árbol de decisión para abordar problemas de clasificación binaria utilizando Python. Los árboles de decisión ofrecen una metodología comprensible y versátil para abordar problemas de clasificación y regresión, pero su uso debe ser cuidadoso para evitar problemas como el sobreajuste y la inestabilidad. La utilización de técnicas de poda de árboles, la limitación de la profundidad del árbol y el uso de métodos de ensamble son estrategias comunes para mejorar el rendimiento de los árboles de decisión en la práctica.

Por lo que, se motiva a los estudiantes a revisar material y ejemplos adicionales disponibles libremente en Internet para profundizar en la comprensión del algoritmo y su aplicación eficaz en diferentes problemas de variada complejidad.

Bibliografía:

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

Gerón, A. (2023). Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow: Conceptos, herramientas y técnicas para conseguir sistemas inteligentes. Tercera Edición. Anaya Multimedia.

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

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