Trabajo Final Integrador - Aplicación de Inteligencia Artificial

Fecha de entrega: viernes, 10 de noviembre de 2023.

Alumno: Rodriguez, Ezequiel Facundo.

Consigna:

1) Elegir un dataset de su interés. Puede ser uno público o privado al que tengan acceso. (En caso de ser datos privados, la cátedra no divulga ningún tipo de información presentada en los trabajos).
2) Realizar un análisis exploratorio de los datos, identificando correlaciones, comportamientos y patrones de interés. Aplicar las transformaciones y procesamiento que considere necesario.
3) Definir un problema concreto a resolver utilizando técnicas de AI sobre los datos seleccionados. Por ejemplo: clasificación, regresión, agrupamiento, etc.
4) Seleccionar el tipo de modelo más adecuado para la tarea y entrenarlo con parte de los datos. Evaluar su rendimiento.
5) Ajustar los hiperparámetros y/o la arquitectura del modelo para mejorar los resultados. Analizar y discutir su desempeño.
6) Entregar un informe en formato PDF explicando las decisiones tomadas en cada paso. Incluir las referencias a librerías y métodos utilizados.
7) Entregar un Notebook de Jupyter con el código de la resolución, donde se pueda reproducir el proceso completo.
8) Incluir un enlace a los datos si son públicos o adjuntarlos al envío en caso contrario.

Elección dataset

Para aplicar los conceptos vistos sobre la cursada el dataset seleccionado contiene información sobre automóviles usados publicados para la venta. Dicho dataset cuenta con 10.000 registros con las siguientes variables:

- ‘Unnamed: 0’: id del registro.
- ‘mileage_per_year’: cantidad de millas recorridas en el lapso de un año.
- ‘model_year’: año de fabricación del automóvil.
- ‘price’: precio al cual se publicó el automóvil.
- ‘sold: indica si el automóvil fue vendido o no.

Los tipos de datos de cada una de las variables son:

- ‘Unnamed: 0’: int64.
- ‘mileage_per_year’: int64.

- ‘model_year’: int64.
- ‘price’: float64.
- ‘sold: object.

Ninguna de las variables cuenta con no nulos (NaN), por lo tanto no se debe realizar tratamiento sobre los mismos.

Análisis exploratorio de los datos

Como primer medida, se tradujo los nombres de las columnas al español, mediante el uso de un diccionario:

```
dic_col = {
   'mileage_per_year': 'millas_anuales',
   'model_year': 'modelo',
   'price': 'precio',
   'sold': 'vendido'
}
```

Luego, se modificó el tipo de dato en ‘vendido’, pasando de ‘yes’ y ‘no’ como valores asignados a 1 y 0 respectivamente. Esto en pos de poder utilizar más adelante la variable para el análisis como valor numérico.

Con respecto a la variable ‘modelo’, se utilizó la misma para el cálculo de una nueva variable denominada ‘edad_del_modelo’: a fines prácticos, resulta más útil tener la 'edad' del modelo que su año de fabricación, por lo que se realiza ese cambio.

En cuanto a ‘millas_anuales’, se transformó los valores de dicha variable a kilómetros, ya que es la unidad de longitud utilizada en nuestro país para distancias grandes. Para ello, se creó una nueva variable denominada ‘km_anuales’, multiplicando el valor de ‘millas_anuales’ por 1.60934 para su correcta conversión.

Por último, se procedió con la eliminación de las variables que no serán tenidas en cuenta y/o utilizadas para la resolución del problema: en este caso, se eliminaron las variables ‘Unnamed: 0’, ‘millas_anuales’ y ‘modelo’. Por lo que las variables con las que cuenta el dataset luego de las diferentes acciones realizadas es:

- ‘precio’: float64.
- ‘edad_del_modelo’: int64.
- ‘km_anuales’: float64.
- ‘vendido’: int64.

Problema a resolver y selección modelo

En este caso, lo que se busca es predecir si un automóvil usado puesto en venta será vendido o no en base a su edad, sus kilómetros anuales y su precio. Dicha predicción tiene solo dos resultados posibles: el automóvil es vendido o no es vendido, por lo tanto estamos ante un problema de clasificación binaria. Para la resolución del mismo, se optó por utilizar árboles de decisión: estos funcionan de manera similar a un conjunto de reglas "if-then", donde el objetivo es dividir un conjunto de datos en subconjuntos más pequeños basándose en ciertos criterios. Comienza evaluando desde un nodo raíz que representa la totalidad de los datos, el cual trabaja usando la variable que mejor divide los datos. A partir

de allí, cada nodo interno representa una decisión basada en el valor de una característica. Los nodos hoja representan la predicción o clasificación final, las cuales están cada una asociada con una etiqueta específica para la variable objetivo. Este tipo de clasificador utiliza criterios de impureza (como la entropía o la ganancia de información) para determinar la mejor manera de dividir los datos en cada nodo. Una vez que el árbol está construido, se utiliza para hacer predicciones en nuevos datos evaluando la instancia a través de las decisiones tomadas en cada nodo hasta llegar a una hoja.

Se seleccionó este tipo de clasificador debido a varias características de los mismos: su facilidad de interpretación y comprensión, lo cual resulta conveniente en caso de tener que presentar el modelo ante alguien que no está familiarizado con las diferentes técnicas de clasificación; su insensibilidad ante la escala de los datos (lo cual se analizará posteriormente), es decir, que no necesita que las características del modelo están escaladas o normalizadas antes de entrenarlo; su buen funcionamiento ante conjuntos de datos que no fueron completamente limpiados o preprocesados, dando un ahorro significativo de tiempo ; y su rapidez tanto para el proceso de entrenamiento como para el proceso de predicción del modelo.

Antes de separar los datos para entrenamiento y para prueba se dividen los datos en 2: por un lado, en ‘Y’, la columna de la salida esperada (vendido), y en ‘X’ las demás variables que forman parte del dataset. Luego, a cada una de ellas se las divide en train y test, en base a ciertos parámetros: el porcentaje destinado a test/validación (test_size), el cual para este caso se utiliza un 0.25, y un valor aleatorio cualquiera (random_state), el cual sirve para garantizar repetibilidad.

Una vez que se tienen los conjuntos X_test, y_test, X_train e y_train, se procede a aplicar el modelo seleccionado, en este caso es ‘DecisionTreeClassifier’ proveniente de la librería ‘sklearn.tree’. Al mismo se le aplica un hiperparametro, max_depth = 3, el cual establece que la profundidad máxima del árbol sea 3, lo que permite que el modelo no genere constantemente nodos hasta llegar a hojas puras para entregar resultados, lo cual implicaría un mayor costo de procesamiento que muy probablemente no se condice con una mejora proporcional en los resultados del modelo.

Luego de entrenar al modelo con los datos separados previamente, se realizan las predicciones sobre el conjunto de datos separados para dicha tarea, y se procede a evaluar las distintas métricas que arroja el modelo. Para este caso, las métricas a analizar serán:

- Precisión: es la proporción de predicciones positivas que fueron correctas. Para este caso, seria cuantos automóviles que el modelo predijo cómo vendidos fueron realmente vendidos.
- Recall: es la proporción de casos reales positivos que fueron correctamente identificados. Para este caso, sería la proporción de cuantos de los automóviles vendidos realmente pudo identificar el modelo.
- F1-score: se utiliza para resumir la precisión y sensibilidad de un modelo en una sola métrica.
- Support: indica el número de muestras de cada clase que se usaron para calcular las métricas.

- Macro avg: Promedio simple de métricas por clase que no considera tamaños de clase. Devuelve la métrica promedio sin importar la distribución.
- Weighted avg: Promedio ponderado por clase. Si considera tamaños de clase.

Para estas métricas, cuanto más cercano sea su valor a 1, mejor se considera el modelo aplicado.

Estas métricas pueden también observarse a través de una matriz de confusión, la cual compara los valores de la clase real y de la clase predicha, otorgando así una visual de cómo está trabajando sobre las predicciones el modelo.

Por último, utilizando la librería ‘Graphviz’, se graficaron los diferentes árboles obtenidos para el modelo, cada uno con sus nodos, ramas correspondientes, cantidad de de muestras, condición a evaluar y clase predicha. Esto permite ver de manera visual el modelo, dando una idea de cómo trabaja y evalúa cada variable para llegar a los diferentes resultados predichos.

Ajustes sobre el modelo y evaluación de los mismos

Para el primer caso, se utilizaron los datos luego del análisis exploratorio de los mismos, sin hacer un preprocesamiento de los valores, como puede ser un tratamiento y exclusión de outliers o valores atípicos, o un escalamiento o normalización de los mismos. Para este caso, los resultados obtenidos fueron:

Reporte de clasificación modelo sin escalar:

Classification report				
	precision	recall	f1-score	support
0	0.72	0.81	0.76	1031
1	0.85	0.77	0.81	1432
accuracy			0.79	2463
macro avg	0.78	0.79	0.78	2463
weighted avg	0.79	0.79	0.79	2463

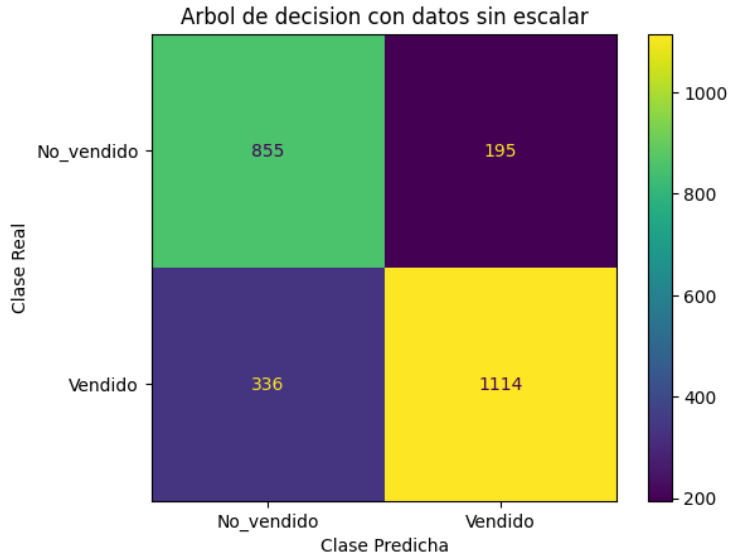
Donde 1 y 0 corresponden a las clases de la variable ‘vendido’.

Muestra métricas:

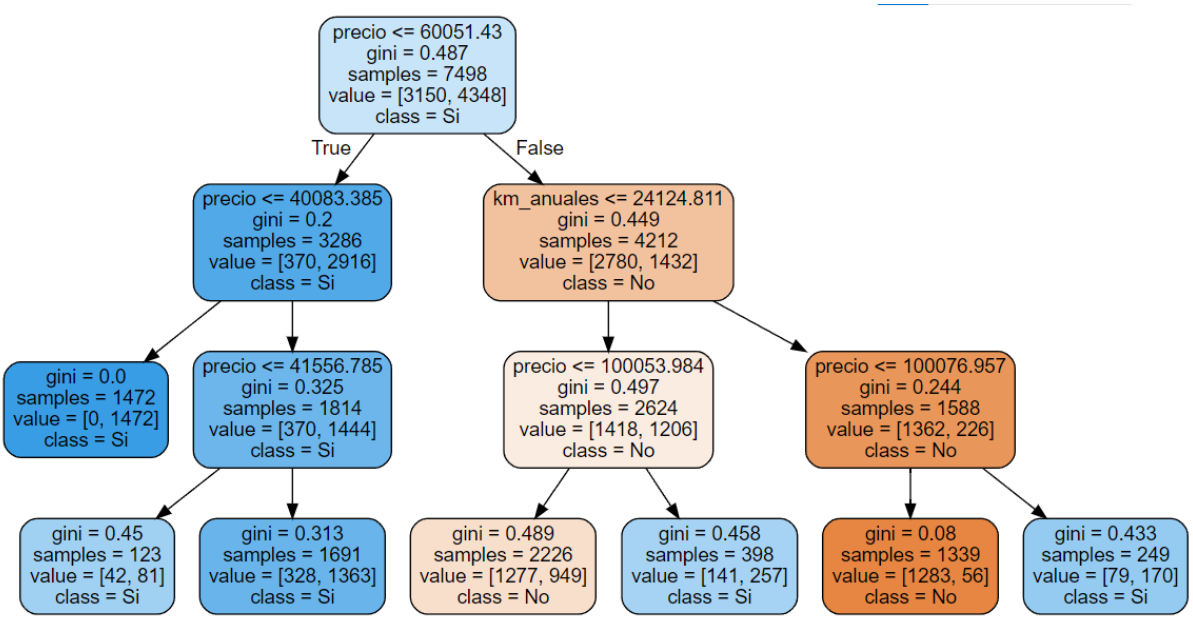
- Accuracy: 78.77%
- Recall: 78.77%
- Precision: 79.40%
- F1-Score: 78.89%
- MCC: 57.38%
- Tiempo de entrenamiento: 0.0306 s

- Tiempo de predicción: 0.0120 s
- Tiempo total: 0.0426 s

Matriz de confusión:



Árbol de decisión resultante:



Para el segundo caso, se procedió a escalar los datos, lo cual consiste en eliminar el valor medio de los datos y llevarlos a escala de forma que su desviación estándar sea 1. Una vez realizado este primer paso, se procedió con el entrenamiento y las predicciones del modelo, el cual arrojó los siguientes resultados:

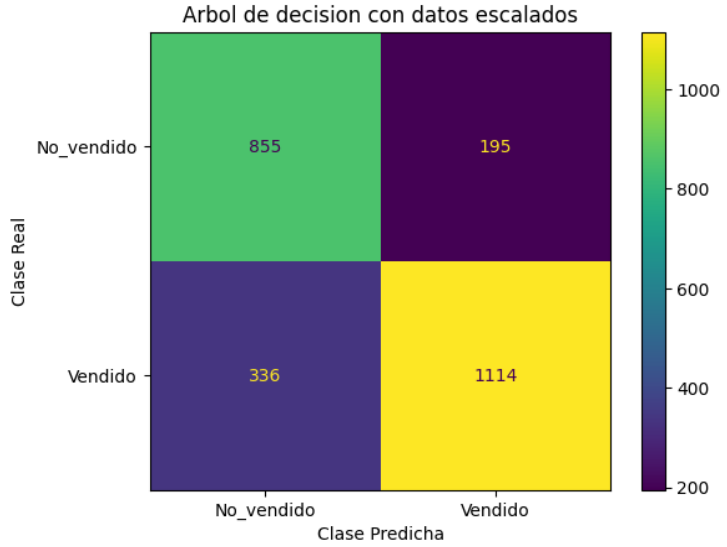
Reporte de clasificación modelo escalado:

Classification report				
	precision	recall	f1-score	support
0	0.72	0.81	0.76	1050
1	0.85	0.77	0.81	1450
accuracy			0.79	2500
macro avg	0.78	0.79	0.79	2500
weighted avg	0.80	0.79	0.79	2500

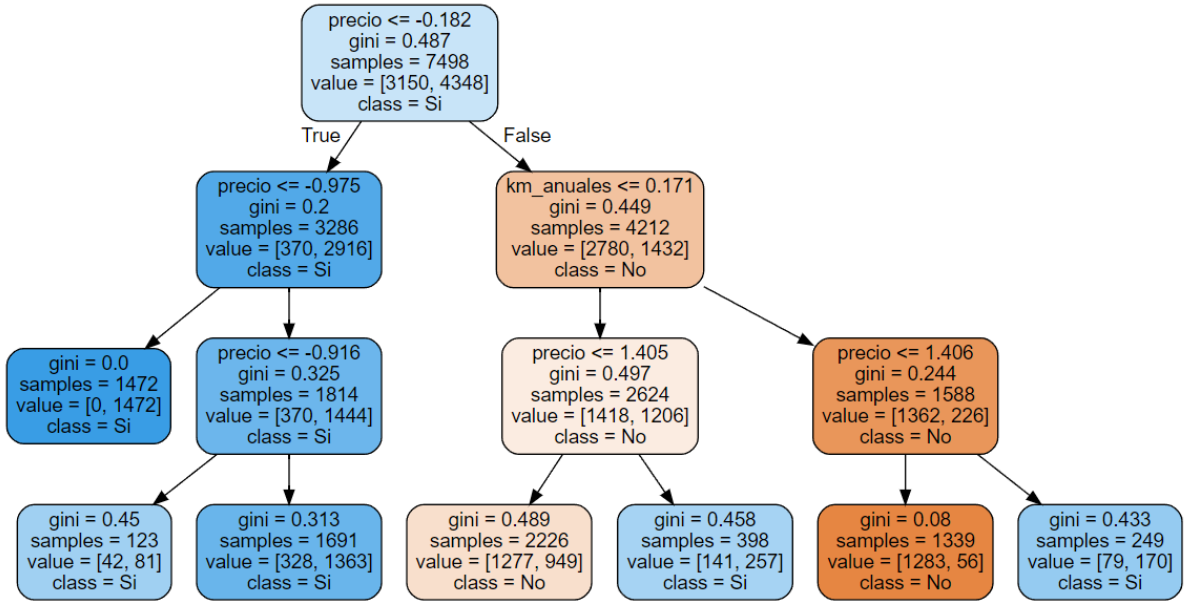
Muestra métricas:

- Accuracy: 78.76%
- Recall: 78.76%
- Precision: 79.51%
- F1-Score: 78.89%
- MCC: 57.57%
- Tiempo de entrenamiento: 0.0245 s
- Tiempo de predicción: 0.0009 s
- Tiempo total: 0.0254 s

Matriz de confusión:



Árbol de decisión resultante:



Comparando las métricas entre el modelo con los datos sin escalar y el modelo con los datos escalados se puede ver que arrojan resultados similares, es decir, el modelo elegido ajusta de igual manera sin importar la naturaleza o escala de los datos. Por lo que se demuestra lo que se afirmó entre las ventajas para elegir el tipo de modelo de árbol de clasificación.

Por último, se procedió a analizar las variables mediante el uso de gráficos de caja o boxplot, con el fin de identificar aquellas variables que contenían outliers, los cuales se suponen que pueden llevar a que el modelo no se ajuste de la mejor manera. En la variable ‘km_anuales’ se podían observar en el gráfico varios puntos por fuera de los valores típicos, por lo que se procedió a aquellos valores que superaran los límites superior e inferior, los cuales se calculan en base a los valores de los cuartiles 1 y 3 y el intervalo intercuartil. Una vez identificados estos puntos, se eliminaron y se avanzó con el entrenamiento y predicciones del modelo, arrojando los siguientes resultados:

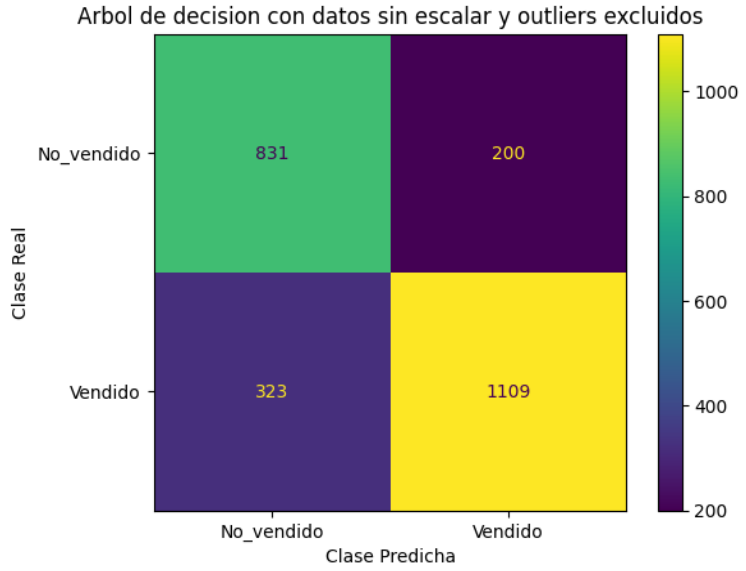
Reporte de clasificación modelo sin escalar excluyendo outliers:

Classification report				
	precision	recall	f1-score	support
0	0.72	0.81	0.76	1031
1	0.85	0.77	0.81	1432
accuracy			0.79	2463
macro avg	0.78	0.79	0,78	2463
weighted avg	0,79	0.79	0.79	2463

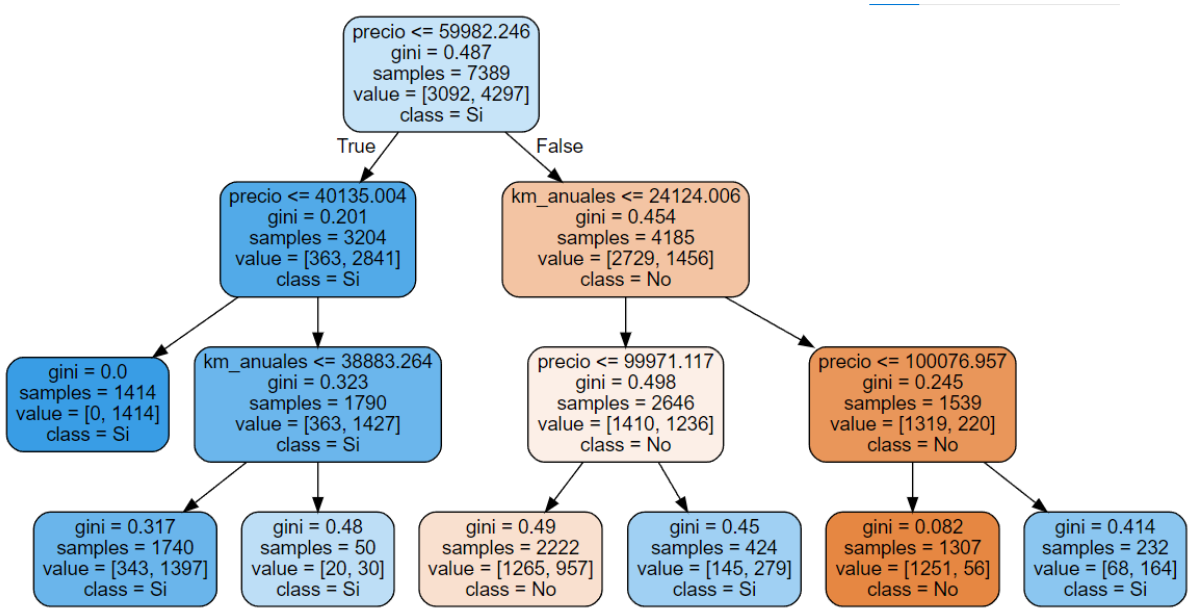
Muestra métricas:

- Accuracy: 78.77%
- Recall: 78.77%
- Precision: 79.40%
- F1-Score: 78.89%
- MCC: 57.38%
- Tiempo de entrenamiento: 0.0163 s
- Tiempo de predicción: 0.0021 s
- Tiempo total: 0.0183 s

Matriz de confusión:



Árbol de decisión resultante:



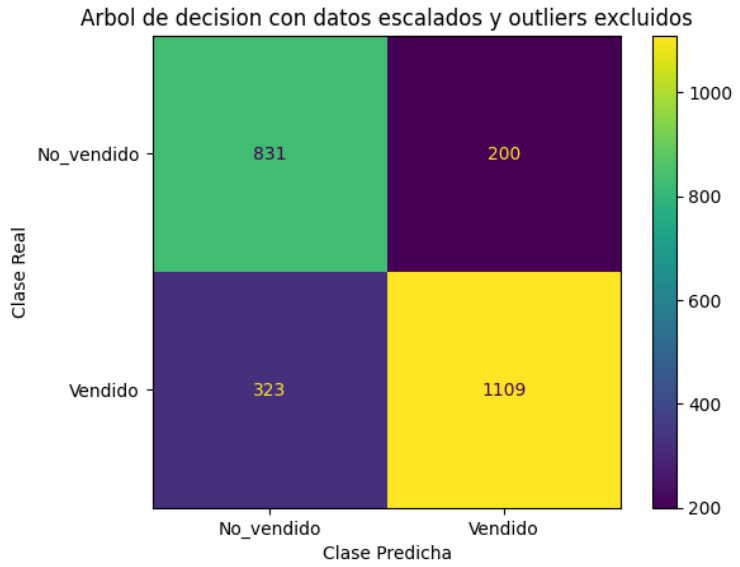
Reporte de clasificación modelo escalado excluyendo outliers:

Classification report				
	precision	recall	f1-score	support
0	0.72	0.81	0.76	1031
1	0.85	0.77	0.81	1432
accuracy			0.79	2463
macro avg	0.78	0.79	0,78	2463
weighted avg	0,79	0.79	0.79	2463

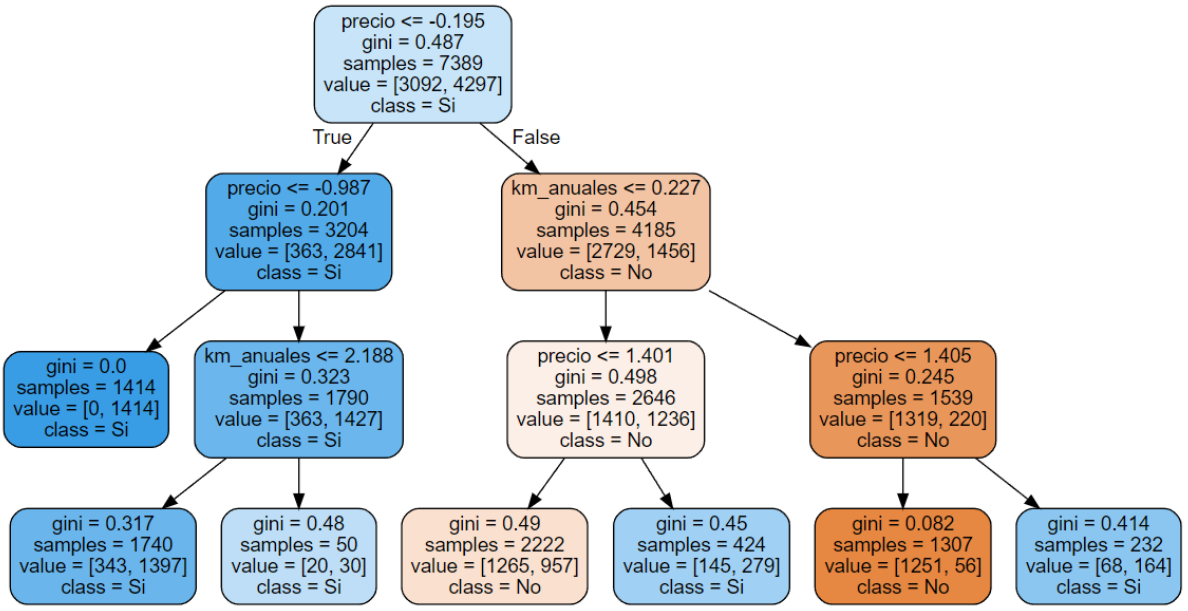
Muestra métricas:

- Accuracy: 78.77%
- Recall: 78.77%
- Precision: 79.40%
- F1-Score: 78.89%
- MCC: 57.38%
- Tiempo de entrenamiento: 0.0208 s
- Tiempo de predicción: 0.0008 s
- Tiempo total: 0.0216 s

Matriz de confusión:



Árbol de decisión resultante:



Ahora se puede observar que ambos modelos, con o sin outliers y con o sin escalamiento, arrojan resultados similares, lo que confirma que el utilizar árboles de decisión permite lo mencionado anteriormente con la escala y además permite evitar el preprocesamiento sin perder ajuste en el modelo.

Con un accuracy cercano al 0.8 para todos los casos evaluados, se puede afirmar que el modelo se ajusta de buena manera a los datos presentados y resulta un buen predictor para el problema planteado al inicio de este informe.

Link dataset:
https://gist.githubusercontent.com/ahcamachod/1595316a6b37bf39baac355b081d9c3b/raw/98bc94de744764cef0e67922ddfac2a226ad6a6f/car_prices.csv

Librerías utilizadas

- Pandas: para la manipulación y el análisis de datos mediante el uso de dataframes.
- Numpy: para realizar cálculos lógicos y matemáticos sobre cuadros y matrices.
- Matplotlib: para crear gráficos en base a datos almacenados.
- Seaborn: para crear gráficos estadísticos.
- Datetime: para manipular fechas y horas.
- Time: para realizar funciones relacionadas con el tiempo.
- Scikit-Learn: para aprendizaje automático y modelado estadístico, incluyendo clasificación, regresión, agrupación, y reducción de dimensionalidad.
- Graphviz: para crear gráficos y presentar información estructural en forma de diagramas.