Técnicas de Inteligencia Artificial

Actividad 1. Laboratorio: Árboles de decisión, reglas y ensemble learning

Presentado por: Leonard Jose Cuenca Roa.

Fecha: 11/06/2023

- **Descripción de la fuente de datos empleada**.

El dataset empleado en esta actividad contiene información sobre la evaluación de vehículos, diseñado para clasificar su aceptabilidad contiene una serie de atributos que permiten validar diversas características. Cada instancia representa un vehículo y sus atributos de entrada describen aspectos clave como el costo de compra, el costo de mantenimiento, el número de puertas, la capacidad de personas, el tamaño del maletero y la seguridad. El objetivo principal de este conjunto de datos es permitir la investigación y el desarrollo de modelos de clasificación que puedan determinar si un vehículo cumple con criterios de aceptabilidad buenos, inaceptables, aceptables o muy buenos. Este dataset me apoyará en aplicar las bases y técnicas de inteligencia artificial.

- **Caracterización del *dataset* utilizado en modo texto y gráfico.**

Se confirmó que todas las columnas del dataset son de naturaleza categórica. Esta categorización se basa en la ausencia de valores recurrentes, progresivos, datos de tipo temporal, o cualquier otro formato que sugiera una interpretación numérica para cálculos o resúmenes estadísticos directos. Las características como 'Buying', 'Maintenance', 'Doors', 'Person', 'lug_boot', y 'safety' presentan categorías discretas.

Existencia de Valores Desconocidos:

Tras ejecutar un script de validación para detectar la presencia de valores nulos, vacíos o incompletos, se determinó que el dataset no contiene valores nulos. Deseo resaltar en caso de un flujo alterno donde se hubieran identificado valores faltantes, la estrategia a seguir incluiría opciones como la eliminación de las instancias incompletas, la imputación de valores utilizando técnicas estadísticas por ejemplo, la media o la moda o la proyección mediante métodos probabilísticos para mantener la integridad del conjunto de datos.

Decisión de Codificación de Características:

Dado que las librerías de **scikit-learn** operan principalmente con datos numéricos, fue indispensable realizar una codificación de las características categóricas. Se optó por utilizar **LabelEncoder** para las variables que presentan un orden inherente o naturaleza ordinal, tales como 'Buying', 'Maintenance', 'safety', y 'lug_boot' (con categorías como 'vhigh', 'high', 'med', 'low' o 'small', 'big'). Esta elección se consideró adecuada al preservar la relación ordinal entre

las categorías. Para los atributos '**Doors**' y '**Person**', aunque sus valores son numéricos ('2', '3', '4', '5more'), se decidió tratarlos también como categóricos y se aplicó **LabelEncoder**. Esta decisión se fundamenta en que, en el contexto de este problema, estos valores no representan magnitudes continuas o sobre las cuales se deban realizar operaciones aritméticas directas, por ejemplo no se espera que **'4'** puertas sea el doble de **'2'** puertas en un sentido puramente numérico para el modelo, sino una categoría distinta.

Finalmente, el atributo **'class'** fue identificado y designado como la variable objetivo para la clasificación, también siendo codificado numéricamente con **LabelEncoder** para su uso en los modelos.

- **Parámetros relevantes utilizados en los diferentes algoritmos.**

En la configuración del **DecisionTreeClassifier**, utilicé los siguientes parámetros clave para optimizar su rendimiento y controlar su complejidad: el parámetro **random_state** se estableció en **42**, asegurando la reproducibilidad de los resultados al fijar una semilla para el generador de números aleatorios interno, lo que garantiza que el árbol de decisión generado será idéntico en ejecuciones repetidas con los mismos datos. Para controlar la complejidad del modelo y prevenir el **sobreajuste (overfitting)**, se definió una profundidad máxima **max_depth** de **10**. Esta limitación restringe el número de niveles que el árbol puede crecer, **evitando que memorice ruidos o patrones específicos** del conjunto de entrenamiento y promoviendo una mejor generalización a nuevos datos. Adicionalmente, el parámetro **min_samples_split** lo que implica que un nodo interno solo se dividirá si contiene un mínimo de **5 muestras**. Esta medida ayuda a controlar la **granularidad** de las divisiones, asegurando que solo se realicen divisiones significativas y contribuyendo a la simplicidad y robustez del modelo. Finalmente, se especificó que cada nodo hoja (terminal) del árbol debe contener un mínimo de 2 muestras mediante el parámetro **min_samples_leaf**. Esto es vital para mejorar la robustez del modelo y reducir el riesgo de **sobreajuste**, al asegurar que cada rama final represente un conjunto razonable de instancias, previniendo así la creación de hojas que contengan una o muy pocas muestras, por ultimo **criterion** le asigné el valor de **gini** es el valor por defecto para medir la probabilidad de clasificación de las instancias elegidas al azar.

En la configuración del **RandomForestClassifier**, se estableció **n_estimators** en **100**, lo que significa que el modelo construirá y combinará un centenar de árboles de decisión individuales. Esta elección tiene el propósito de asegurar un proceso de clasificación más robusto y preciso. **Un número elevado de árboles contribuye significativamente a prevenir el sobreajuste y a reducir la varianza**, ya que las predicciones finales se obtienen

promediando o mediante el voto de un conjunto diverso de modelos. Adicionalmente, el parámetro **random_state** se fijó en **42**. Esta semilla para el generador de números aleatorios es crucial **para garantizar la reproducibilidad de los resultados**. Otro parámetro **max_depth** en **10**, Este parámetro controla la **profundidad máxima de cada árbol** de decisión en el bosque. Es decir, la longitud máxima de la ruta desde la raíz hasta la hoja más profunda de un árbol, el **min_samples_split** en **5** Este parámetro especifica el número mínimo de muestras que debe tener un nodo para poder ser dividido y por ultimo **min_samples_leaf** en **2** define el número mínimo de muestras que debe tener una hoja, Es decir, cada división debe asegurar que tanto el nodo hijo izquierdo como el derecho tengan al menos 2 muestras**.**

- **Resultados obtenidos por los diferentes algoritmos escogidos de forma gráfica y comparada/superpuesta.**

El análisis exploratorio del **dataset**, evidenciado en las gráficas de distribución, reveló patrones importantes. La **Distribución de la Clase de Aceptabilidad de Coches** muestra un claro desequilibrio, donde la clase '**unacc**' (inaceptable) es la más prevalente con aproximadamente **1200** instancias, seguida por '**acc**' (aceptable) con cerca de **400**. En contraste, las clases '**vgood**' (muy bueno) y '**good**' (bueno) son minoritarias, cada una apenas superando las 50 instancias. Por otro lado, las gráficas de **Distribución de Costo de Compra ('Buying')** y **Distribución de Número de Puertas ('Doors')** indican una distribución uniforme en sus respectivas categorías, con aproximadamente 430 instancias por cada nivel de costo o número de puertas, lo que sugiere consistencia en estas características.

En cuanto a los **Resultados Obtenidos por los Diferentes Algoritmos**, la comparación de efectividad entre modelos ilustra un desempeño ligeramente superior del **Árbol de Decisión**, con una precisión aproximada del **0.950**, superando al **Bosque aleatorio** que alcanzó alrededor del **0.941**. Se debe reflejar que ambos algoritmos lograron su desempeño en clasificar el dataset de vehículos, ambos reflejan porcentajes muy altos, pero el que tuvo mejor resultado fue el de **Árbol de decisión** bien sea por su simplicidad, pero no descarto ninguno de los dos ya que uno de los modelos es mas usado por su poca complejidad en comparación con el de **Bosque** aleatorio agregando que ambos tienen su impacto durante la ejecución.

- **Discusión de los resultados obtenidos y argumentos sobre cómo mejorar de dichos resultados.**

Como resultado y reflexión del análisis comparativo de los modelos de clasificación, se puedo mencionar varias conclusiones significativas. El **Árbol de Decisión (Decision Tree**

Classifier), aunque destaca por su flexibilidad, facilidad de interpretación y bajo requerimiento de preparación de datos, demostró un margen ligeramente alto, en comparación con **Bosque aleatorio (Random Forest Classifier**). En contraste, el modelo **Random Forest Classifier** exhibió un rendimiento ligeramente inferior. Su naturaleza de **ensemble learning** lo hace intrínsecamente menos propenso al **sobreajuste** y considerablemente más robusto en sus operaciones, manejando de forma efectiva datos no lineales. Las métricas obtenidas confirmaron que **Random Forest** esta un poco por debajo en comparación con **Decision Tree** en precisión general y crucialmente, en la capacidad de clasificar correctamente clases minoritarias como ‘good’, se debe resaltar que considero que ninguno es mejor que el otro ambos se complementan para llegar a una conclusión sobre el estudio principal el dataset de vehículos gracias a estos modelos puedo conformar la siguiente premisas.

El análisis del conjunto de datos reveló que la seguridad (**Safety**) es el factor más crítico para los usuarios, aportando un 35% de importancia. Los niveles bajos de seguridad son rara vez aceptables. En segundo lugar, la capacidad de personas (**Person**) tiene un 25% de importancia; los vehículos con capacidad para dos personas son inaceptables, mientras que aquellos con capacidad para "más" personas son mejor valorados. El precio de compra (**Buying**) contribuye con un 20% de importancia, siendo los precios "muy altos" fuertemente penalizados y los precios "bajos" asociados a clasificaciones positivas. Finalmente, el mantenimiento (**Maintenance**) representa un 15% de importancia. Otros atributos como el tamaño del maletero (**lug_boot**) y el número de puertas (Doors) tienen una importancia mínima (3% y 2% respectivamente), lo que sugiere que son menos relevantes para los usuarios en la toma de decisiones

Se menciona algunas técnicas para mejorar los resultados obtenidos en ambos algoritmos considero lo siguiente: **Aumento del Volumen de Datos;** Considerar la posibilidad de obtener o generar más datos para el entrenamiento de los modelos, especialmente para las clases minoritarias. **Técnicas de Balanceo de Clases;** Dada la clara identificación de un desequilibrio significativo de clases en el dataset (donde algunas clases tienen muchas más instancias que otras, como 'unacc' frente a 'good' o 'vgood'), la aplicación de técnicas de balanceo es crucial. En particular, la técnica SMOTE (**Synthetic Minority Over-sampling Technique)** es altamente recomendable, ya que puede generar instancias sintéticas para las clases minoritarias, ayudando a los modelos a aprender sus patrones de manera más efectiva y a mejorar su rendimiento global.