Técnicas de Inteligencia Artificial

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

Presentado por: Leonard Jose Cuenca Roa.

Fecha: 20/06/2023

- **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.

  (Flujo Alterno - Solo para información, podrías omitirlo o dejarlo si quieres demostrar tu conocimiento del proceso: En un escenario 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 (e.g., 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 (e.g., 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 5. 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 se configuró en 4, lo que implica que un nodo interno solo se dividirá si contiene un mínimo de 4 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 = 2. 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.

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. Dada la naturaleza aleatoria inherente al **RandomForestClassifier** tanto en el muestreo de datos para entrenar cada árbol como en la selección de características para las divisiones de los nodos, establecer random_state asegura que el proceso de entrenamiento y las predicciones resultantes sean consistentes en ejecuciones repetidas.

- **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 superior del **Random Forest**, con una precisión aproximada del 0.95, superando al **Decision Tree** que alcanzó alrededor del 0.88. Esta diferencia subraya la mayor tasa general de clasificaciones correctas del Random Forest en el conjunto de prueba. Al examinar las Matrices de Confusión, se observa que el Decision Tree tuvo dificultades significativas para aprender los patrones de la clase 'good', posiblemente debido al desequilibrio de clases, lo que resultó en una clasificación deficiente para esta categoría. En contraste,

el **Random Forest demostró una gestión mucho más efectiva de la clase 'good'**, lo cual evidencia su mayor robustez y capacidad para aprender de clases minoritarias, a diferencia de un único árbol de decisión.

- **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 de precisión aceptable, pero con una clara propensión al sobreajuste y una cierta inestabilidad, especialmente evidente en su incapacidad para clasificar la clase 'good'.

En contraste, el modelo Random Forest Classifier exhibió un rendimiento superior. 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 supera al Decision Tree** en precisión general y crucialmente, en la capacidad de clasificar correctamente clases minoritarias como ‘good'.

Se menciona algunas técnicas para mejorar los resultados obtenidos en ambos algoritmos:

Ajuste de Hiperparámetros (Hyperparameter Tuning): Es fundamental aplicar técnicas como GridSearchCV o RandomizedSearchCV de scikit-learn. Estas herramientas permitirían explorar sistemáticamente diferentes combinaciones de parámetros para cada modelo, con el fin de encontrar la configuración óptima que maximice su rendimiento y generalización.

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.