Proyecto - Adult-census-income classification

Ángel Alonso Galarza Chávez

Diciembre 2024

Profesor: Dra. Xiaoou Li
Curso: Mineria de Datos

1 Introducción

La clasificación consiste en entrenar modelos para aprender a reconocer patrones en los datos y asignar nuevos elementos a categorías específicas. Consiste en aprender de la estructura de un conjunto de datos, que se encuentran particionando en clases, el aprendizaje de estas categorías se logra construyendo un modelo de clasificación que se utilizara para estimar los identificadores de las clases que existen en un conjunto de datos. Para la creación de este modelo se requiere de un análisis que extrae la información útil para poder clasificar cada nueva instancia a la clase en la que pertenece. [2, 1]

Este proyecto tiene como objetivo profundizar en los conceptos y etapas involucradas en la construcción de modelos de clasificación. Exploraremos técnicas de preprocesamiento de datos, selección de características, entrenamiento y evaluación de modelos, con el fin de construir un clasificador robusto y preciso.

2 Descripcion del conjunto de datos

El objetivo del análisis es observar las características de los datos para obtener información útil para la construcción de un modelo de clasificación. Se emplean herramientas de visualización como histogramas, gráficos de caja, entre otros y técnicas de estadística como la correlación. Esta información es útil para observar si el conjunto de datos presenta valores faltantes o errores en los datos, y preparar un plan de como tratar estos datos y seleccionar los atributos que aporten información que nos sea de ayuda para la construcción del modelo de clasificación.

El conjunto de datos Adult-Income Dataset obtenidos del sitio web [4], contiene 15 atributos, los atributos son categóricos o numéricos, los atributos de tipo numérico son age, fnlwgt, education-num, capital-gain, capital-loss y hours-per-week, cada uno de estos atributos tienen una escala diferente como se puede apreciar en la figura 1. Los atributos categóricos del conjunto de datos son workclass, education, marital-status, occupation, relationship, race, sex, native-country y target, estos atributos contienen información como el tipo de trabajo, estado civil, el país de origen, toda esa información puede ser útil para la construcción del modelo.

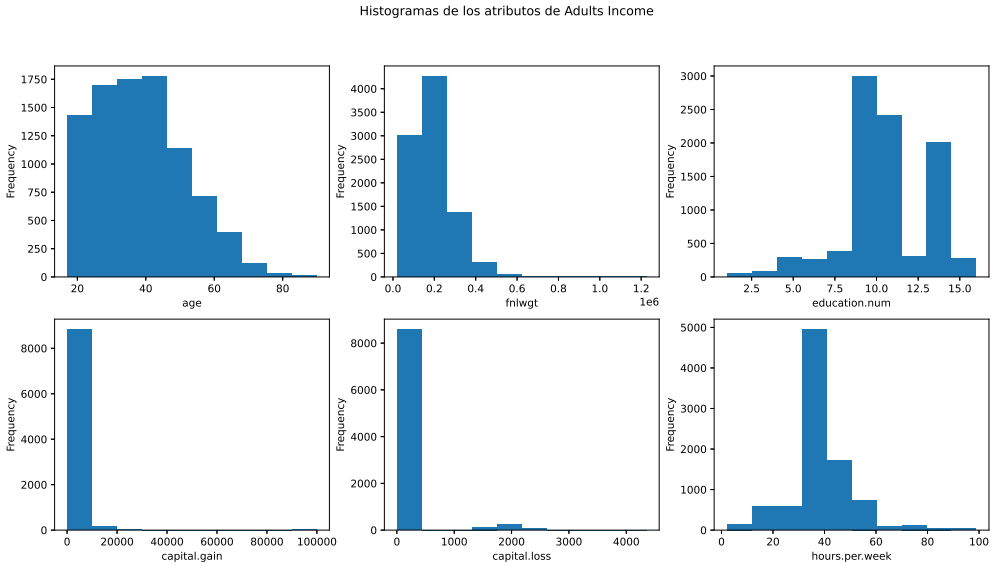


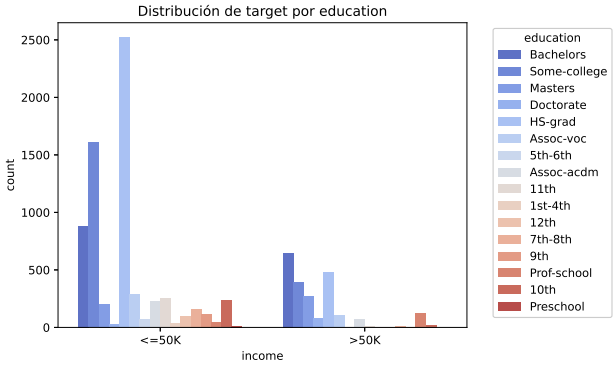
Figura 1: Histogramas de los atributos numericos

Se tiene dos archivos, uno para el entrenamiento del modelo que contiene 9049 registros con 15 columnas. De igual manera, el archivo para prueba contiene las mismas columnas a excepcion de la columna income que sera la columna que predeciremos el valor.
El conjunto presenta una gran cantidad de atributos y con ello también una gran cantidad de valores en cada atributo, como se vio previamente en la figura 1 existen atributos que no tienen una distribución normal, tal es el caso de los atributos 'capital gain' y 'capital loss' donde su valor mas frecuente es el 0 y el resto de valores mayores a 0 no representan una cantidad significativa.

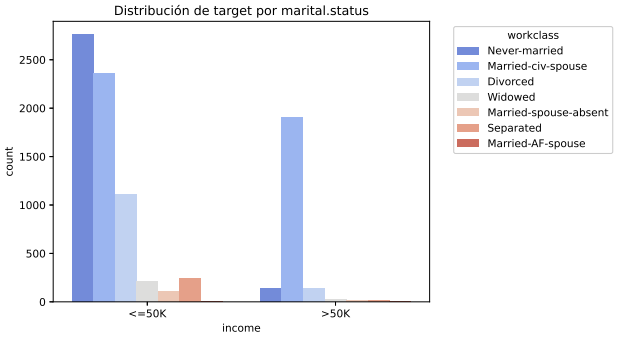
2.1 Atributos categóricos

Las siguientes figuras presentan una exploración visual de los atributos categóricos del conjunto de datos. A través de estas visualizaciones, buscaremos identificar patrones, tendencias y posibles relaciones entre las variables. El análisis de estas distribuciones nos permitirá identificar patrones relevantes para el entrenamiento de los modelos de clasificación.

El atributo 'education' proporciona información sobre el nivel educativo de los individuos como se puede apreciar en la figura 2a. Aunque podría ser útil para la clasificación, se ha decidido eliminar debido a la existencia de 'education num', por lo que el atributo 'education' se considera redundante debido a la presencia de 'education num', que captura de manera más precisa la información jerárquica sobre el nivel educativo.
La variable 'marital status' aporta información valiosa para distinguir entre las clases. La alta frecuencia de 'never married' en la clase 0 y de 'married-civ-spouse' en la clase 0 y 1 como se aprecia en la figura 2b sugiere una fuerte asociación entre el estado civil, sin embargo existen otros valores que contiene el atributo. Por lo tanto, se mantendrá esta variable en su formato original para aprovechar al máximo su potencial discriminatorio.

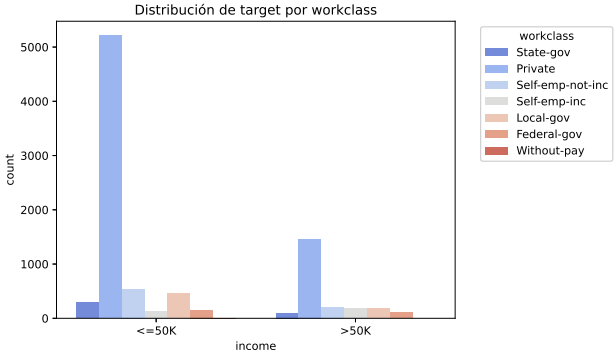


(a) Atributo Education

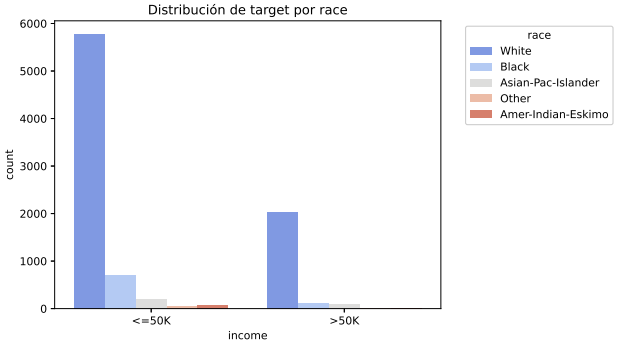


(b) Atributo Marital Status

El atributo 'workclass' indica el tipo de empleo de cada individuo como se muestra en la figura 3a. Si bien 'private' es la categoría más frecuente, se ha decidido mantener todas las categorías de la variable 'workclass', incluyendo aquellas con menor frecuencia. Esto se debe a que la diversidad de tipos de empleo puede ser un factor determinante para la clasificación.
Por otro lado, 'race' en la figura 3b presenta una distribución desbalanceada, con una alta frecuencia de 'white'. Si bien podría codificarse binariamente, se ha optado por mantenerla en su formato original para permitir al modelo capturar posibles interacciones entre la raza y otras variables, y así obtener una mejor comprensión de los patrones subyacentes en los datos..

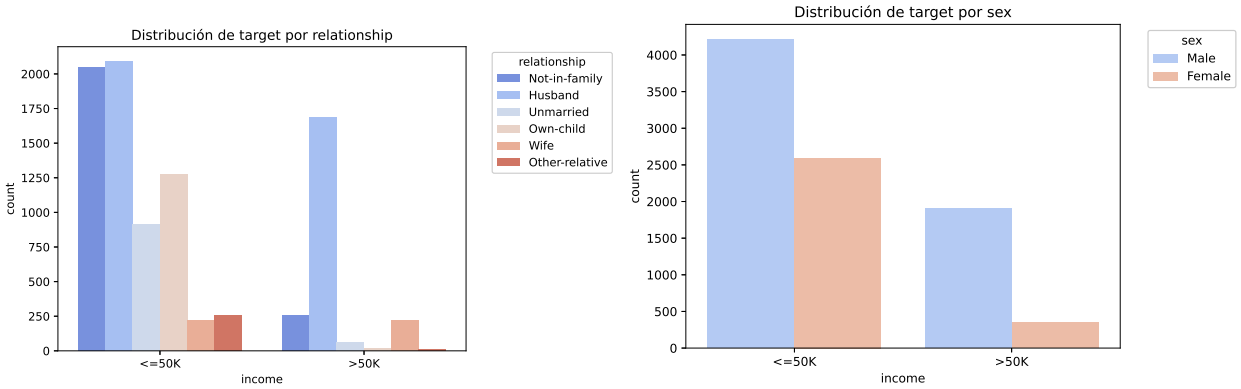


(a) Atributo Workclass



(b) Atributo Race

El atributo 'relationship' proporciona información sobre el estado civil de los individuos como se aprecia en la figura 4a. Si bien existe una alta frecuencia de 'Husband' en ambas clases, los valores 'own-child' y 'other-relative' son menos comunes en la clase 1.
Por otro lado, el atributo 'sex', en la figura 4b muestra una distribución similar en ambas clases, aunque con diferentes proporciones. Dada la naturaleza binaria de 'sex', se puede representar de forma numérica asignando 0 a 'male' y 1 a 'female'.



(a) Atributo Relationship

(b) Atributo Sex

El atributo categórica 'occupation' aporta información valiosa para la clasificación como se puede apreciar en la figura 5, ya que refleja la diversidad de profesiones de los individuos. Dada la naturaleza nominal de este atributo y la imposibilidad de establecer un orden jerárquico entre las diferentes ocupaciones, se optó por no aplicar un preprocesamiento de reducción de datos.

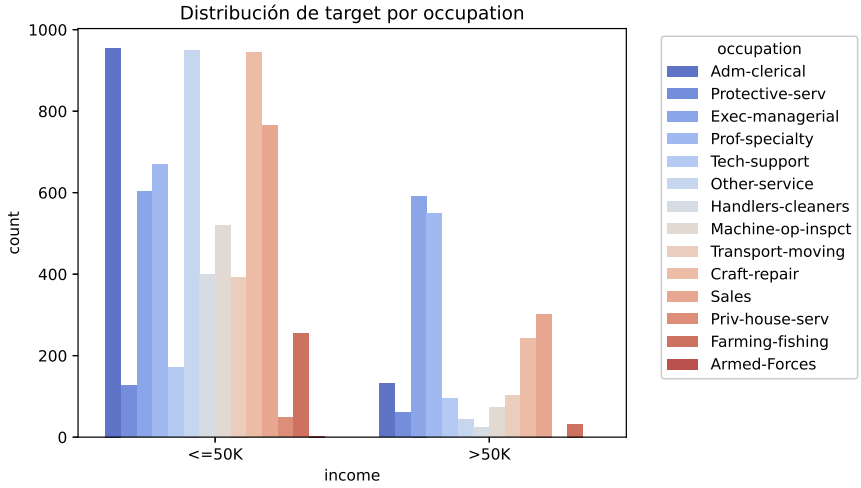


Figura 5: Atributo Occupation

3 Clasificadores

3.1 K-nearest-neighbor

El algoritmo de K-nearest-neighbor se ha utilizado ampliamente en el área de reconocimiento de patrones. K-nearest-neighbor se basan en la comparacion del los datos del conjunto de entrenamiento con el conjunto de prueba, buscando similadridades entre esos datos. Cada dato contiene atributos que representa un punto en un espacio n-dimensional. Cuando se le da una tupla desconocida, el algoritmo K-nearest-neighbor más cercanos busca en el espacio de patrones las k puntos de entrenamiento que están más cerca del dato desconocida. Estas k datos de entrenamiento son los k "vecinos más cercanos" del dato desconocida [2]. Para clasificar una nueva instancia, el algoritmo K-Nearest Neighbors identifica los k puntos de entrenamiento más cercanos a ella en el espacio de características. A continuación, se realiza una votación entre las etiquetas de clase de estos k vecinos más cercanos, asignando a la nueva instancia la clase que obtenga la mayoría de votos. Es fundamental que el valor de k sea un número impar para evitar empates en la votación, garantizando así

una clasificación única.

3.2 Decision Tree

Los arboles de decision son un algoritmo de clasificacion, en la que el proceso de clasificacion se realiza por medio de un conjunto de decisiones con jerarquia que se construyen a partir de los atributos de un conjunto de datos, dicha estructura se asemaja a un arbol binario, lo que lo convierte en un algoritmo facil de intrepretar. Los arboles de decision se copmrende de nodos que son las decisiones, cada nodo tiene un criterio de division, que es la condicion sobre una o mas variables que evaluan cada instancia nueva, los nodos hojas de los arboles de decision son las clases que perteneceran las nuevas instancias de los datos [1]. El objetivo de los arboles de decision es clasificar correctamente las nuevas instancias con sus respectivas clases, para lograr dicho objetivo se requeire de una serie de pasos para entrenar y validar los modelos de clasificacion [6].

3.3 Random Forest

El algoritmo Random Forest intenta mejorar el rendimiento de generalización mediante la construcción de un conjunto de árboles de decisión. Random Forest se basan en la idea de que se toman múltiples muestras del conjunto de datos original, pero con una particularidad: cada muestra se obtiene seleccionando aleatoriamente elementos del conjunto original con reemplazo. Esto significa que un mismo elemento puede aparecer varias veces en una misma muestra, y otros pueden no aparecer en absoluto [6].

Se seleccionaron los algoritmos KNN, Árbol de Decisión y Random Forest debido a su eficacia en tareas de clasificación binaria. KNN destaca por su simplicidad y capacidad para capturar patrones no lineales. Los Árboles de Decisión ofrecen modelos intuitivos y jerárquicos, mientras que Random Forest, una extensión de estos, proporciona mayor robustez y precisión al combinar múltiples árboles.

4 Experimentación y resultados

Para el desarrollo de este proyecto, se empleó una serie de bibliotecas compuesta por Pandas [7], Scikit-learn [5], Matplotlib [3] y Seaborn [8]. Pandas se utilizó para la carga, limpieza y transformación de los datos, facilitando el preprocesamiento necesario. Scikit-learn proporcionó las herramientas para implementar y evaluar diversos modelos de aprendizaje automático, incluyendo K-Nearest Neighbors, Árboles de Decisión y Random Forest. Además, se aprovechó su funcionalidad para optimizar los hiperparámetros de los modelos. Por último, Matplotlib y Seaborn permitieron visualizar los datos y los resultados de los modelos, facilitando la interpretación y análisis.
En la experimentacion, se emplearon distintas tecnicas para la construccion de los distintos modelos de clasificacion. Obtener una representación de los datos con una menor dimensionalidad puede ayudar a obtener mejores resultados de precisión y exactitud para el modelo de clasificación.

4.1 Preprocesamiento

El preprocesamiento de los datos es una etapa fundamental en cualquier proyecto de aprendizaje automático. A través de diversas técnicas, transformamos los datos a un formato adecuado para ser procesado por los algoritmos de clasificación. En este caso, nos encontramos con conjuntos de datos que contienen tanto variables numéricas como categóricas.
Para las variables categóricas, se aplicó la técnica de one-hot encoding. Esta técnica consiste en crear nuevas columnas binarias (con valores 0 o 1) por cada categoría única de la variable original. Esta representación permite a los algoritmos de aprendizaje automático tratar las categorías como características numéricas. Como resultado de aplicar one-hot encoding, el número de columnas aumentó de 15 a 86. Este incremento es común cuando se trabaja con variables categóricas que tienen muchas categorías únicas. Sin embargo, esta transformación es necesaria para garantizar que el modelo pueda aprender las relaciones entre las variables y realizar predicciones precisas.

4.2 Construcción de los modelos

Se realizo una partición estratificada del conjunto de datos en un 70% para entrenamiento y un 30% para evaluación, asegurando una representación proporcional de cada clase en ambos subconjuntos. Esta estrategia permitió obtener una estimación mas precisa del desempeño general de los modelos.

Los algoritmos de aprendizaje automático supervisado dependen en gran medida de la correcta configuración de sus hiperparámetros para alcanzar un rendimiento óptimo. Estos parámetros, que no se aprenden directamente de los datos sino que se establecen antes del entrenamiento, controlan la complejidad del modelo y su capacidad de generalizar a nuevos datos.

La búsqueda de la combinación óptima de hiperparámetros es un desafío crucial en el desarrollo de modelos de clasificación. Scikit-learn ofrece diversas técnicas para abordar esta tarea, destacando la búsqueda exhaustiva y la validación cruzada. La búsqueda exhaustiva explora sistemáticamente todas las posibles combinaciones de valores para los hiperparámetros, mientras que la validación cruzada es una técnica fundamental para obtener estimaciones más precisas del rendimiento de un modelo. Al dividir el conjunto de datos en varias partes y entrenar el modelo en diferentes combinaciones de estas partes, se reduce el riesgo de sobreajuste y se obtiene una mejor idea de cómo el modelo se generalizará a datos no vistos.

4.2.1 K-nearest-neighbor

"n_neighbors" este hiperparámetro determina el número de vecinos más cercanos que se considerarán para la clasificación, "wights" controla cómo se ponderan los votos de los vecinos más cercanos y "metric" define la función de distancia utilizada para medir la cercanía entre los puntos.

Los valores para los hiperparámetros para hacer el ajuste del algoritmos K-nearest-neighbor son los siguientes:

- **n_neighbors:** [3, 5, 7, 9, 11, 13, 15]
- **weights:** Uniform y Distance
- **metrict:** Euclidean y Manhattan

4.2.2 Arbol de Decision

Para construir el modelo de clasificación, se emplearon dos implementaciones de algoritmos CART que proporciona la biblioteca scikit-learn, optimizada y con soporte para los criterios de impureza Gini e entropía.
A continuación se mostrar los hiperparámetros utilizados para la construcción del modelo de clasificación.
"criterion" este hiperparámetro define el criterio de impureza utilizado para seleccionar la mejor división en cada nodo del árbol, "max_depth" limita la profundidad máxima de cada árbol, "min_samples_split" define el número mínimo de muestras requeridas para dividir un nodo interno y "min_samples_leaf" define el número mínimo de muestras requeridas en cada hoja.
Los valores para los hiperparámetros para hacer el ajuste a los algoritmos CART son los siguientes:

- **criterion:** Gini index y Entropia.
- **max_depth:** [4, 8, 12, 16, 20, 24]
- **min_samples_split:** [2, 5, 10, 20, 25]
- **min_samples_leaf:** [1, 2, 4, 6, 10, 15]

4.2.3 Random Forest

Para la construccion del modelo de clasificacion utilizando el algoritmo Random Forest se utilizaron los siguientes hiperparametros. El hiperparámetro "n_estimators" define el número de árboles de decisión que se construirán en el bosque aleatorio, "max_features" controla el número de características que se consideran al dividir cada nodo en un árbol de decisión, "max_depth" define la profundidad máxima de cada árbol

de decisión "max_leaf_nodes" y limita el número máximo de nodos hoja en cada árbol. A continuación se muestran el rango de los valores de los hiperparámetros para la construcción del modelo Random Forest:

- **n_estimators:** [25, 50, 100, 150, 175, 190, 200]
- **max_depth:** [3, 6, 9, 12, 16, 24, 32]
- **min_features:** ['sqrt', 'log2']
- **max_leaf_nodes:** [3, 6, 9, 12, 16, 24, 32]

4.3 Resultados

Los resultados de la etapa de entrenamiento que se muestran en la tabla 1, el modelo de Random Forest es el que presenta el mejor desempeño general. Alcanzó una precisión del 88% y un recall del 96% para la clase 0, y del 79% y 53% para la clase 1, respectivamente. Su F1-score, que combina precisión y recall, también fue el más alto en ambas clases, con valores de 0.92 y 0.64.
El modelo de Árbol de Decisión obtuvo resultados cercanos, pero ligeramente inferiores al de Random Forest. Por último, el modelo K-nearest-neighbor mostró un desempeño considerablemente menor, especialmente en la clase 1, donde sus métricas de precisión, recall y F1-score fueron significativamente más bajas.

A continuación se presentan las curvas ROC de los tres modelos de clasificacion.

La curva ROC de la figura 6 nos indica que el modelo de clasificación K-nearest-neighbor, durante su entrenamiento, ha mostrado una buena capacidad para clasificar correctamente los ejemplos positivos sin generar demasiados falsos positivos.

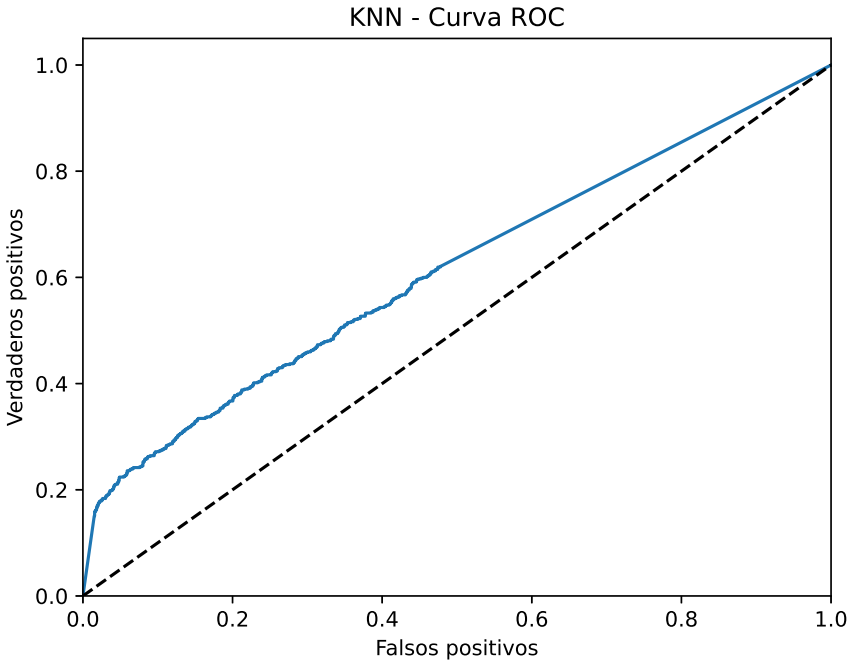


Figura 6: KNN - Curva ROC en la etapa de entrenamiento

La forma de la curva de la figura 7 sugiere que el modelo tiene una buena capacidad para distinguir entre las clases positivas y negativas, nos proporciona una visión clara del desempeño del modelo de árbol de decisión en la clasificación binaria. Ademas se puede apreciar que obtiene una mejor clasificación en comparación al modelo que utiliza K-nearest-neighbor, teniendo un mayor curva.

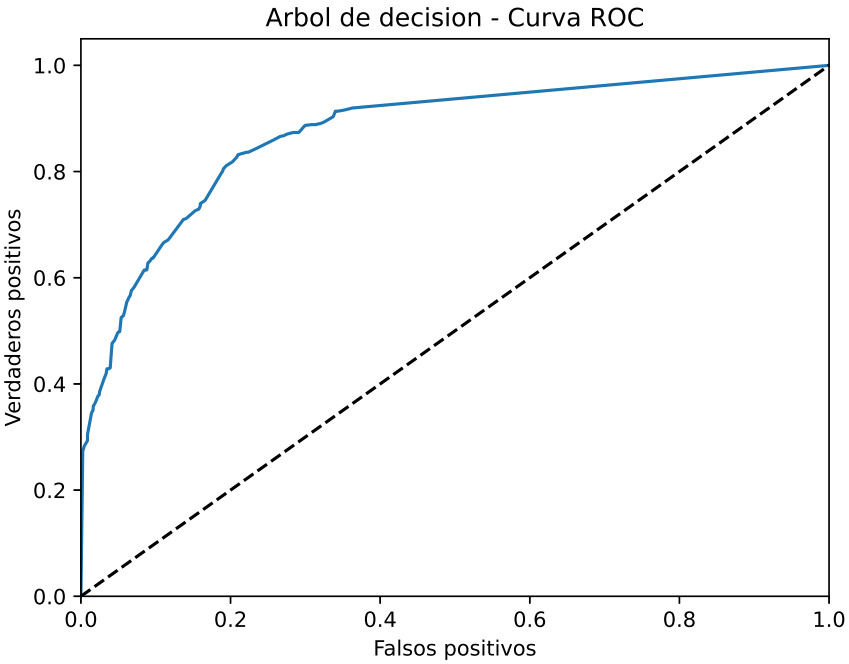


Figura 7: Árbol de decisión - Curva ROC en la etapa de entrenamiento

Por ultimo, basándonos en la curva ROC de la figura 8 y en las características generales del modelo Random Forest, podemos concluir que este modelo presenta un excelente desempeño en la tarea de clasificación. Con esto podemos confirmar que modelo Random Forest confirma la superioridad de este algoritmo.

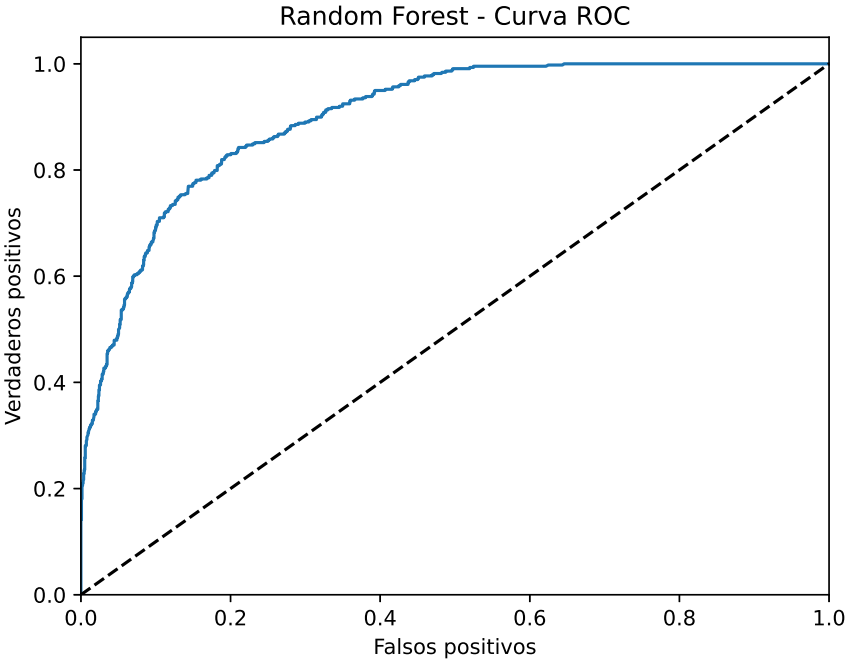


Figura 8: Random Forest - Curva ROC en la etapa de entrenamiento

Table 1: Tabla de resultados del conjunto de entrenamiento

Modelo	Clase	Precision	Recall	F1-score
K Nearest Neighbor	0	0.80	0.85	0.82
	1	0.43	0.35	0.38
Decision Tree	0	0.87	0.93	0.90
	1	0.73	0.58	0.65
Random Forest	0	**0.88**	**0.96**	**0.92**
	1	**0.79**	**0.53**	**0.64**

La tabla de resultados con el conjunto de datos de prueba de la tabla 2, utilizando la métrica F1-score, confirma las tendencias observadas en la tabla 1, en la etapa de entrenamiento. El modelo K-nearest-neighbors obtuvo el F1-score más bajo, con un valor de 0.73. Le sigue el modelo de Árbol de Decisión con un F1-score de 0.81. Finalmente, el modelo de Random Forest se mantiene como el mejor modelo, alcanzando un F1-score de 0.84.

Table 2: Tabla de resultados del conjunto de datos de prueba

Modelo	F1-score
K Nearest Neighbor	0.7341
Decision Tree	0.8185
Random Forest	**0.8496**

5 Conclusión

Basándose en los resultados presentados, se puede concluir que el modelo de Random Forest es el más adecuado para este problema de clasificación binaria. Su capacidad de generalizar y su buen desempeño en ambas clases lo convierten en una excelente opción. Sin embargo, es importante recordar que la elección del mejor modelo siempre depende del conjunto de datos específico y del problema a resolver, con un preprocesamiento diferente puede afectar a los modelos de clasificación, mejorando su desempeño o no logrando generalizar bien los datos. Es por ello que es recomendable realizar una experimentación con distintos preprocesamiento de datos empleando diversas técnicas para la transformación y con ello obtener un modelo robusto y con mejores resultados.

6 Referencias

[1] Charu C Aggarwal. *Data mining*. en. 2015th ed. Cham, Switzerland: Springer International Publishing, Apr. 2015.

[2] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining concepts and techniques, third edition*. 2012. URL: `http://www.amazon.de/Data-Mining-Concepts-Techniques-Management/dp/0123814790/ref=tmm_hrd_title_0?ie=UTF8&qid=1366039033&sr=1-1`.

[3] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: `10.1109/MCSE.2007.55`.

[4] Xiaoou Li. *Aadult-census-income classification*. `https://kaggle.com/competitions/aadult-census-income-classification`. Kaggle. 2024.

[5] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[6] Pang-Ning Tan et al. *Introduction to data mining, global edition*. en. 2nd ed. London, England: Pearson Education, Mar. 2023.

[7] The pandas development team. *pandas-dev/pandas: Pandas*. Version v2.2.3. Sept. 2024. DOI: 10.5281/zenodo.13819579. URL: https://doi.org/10.5281/zenodo.13819579.

[8] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.