ACTIVIDAD LABORATORIO NO.2
TRABAJO: MODELO SENCILLO MEDIANTE ÁRBOLES DE CLASIFICACIÓN EN PYTHON

PRESENTADO POR:
ALEJANDRO DE MENDOZA

PRESENTADO AL PROFESOR:
ING ROGERIO ORLANDO BELTRAN CASTRO

FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA
BOGOTÁ D.C.
23 DE FEBRERO
2026

TABLA DE CONTENIDO

INTRODUCCIÓN

En el contexto actual de la educación superior y la creciente competitividad en los procesos de admisión a programas de posgrado, las instituciones académicas manejan grandes volúmenes de información relacionada con el desempeño académico y los antecedentes de los postulantes. El análisis sistemático de estos datos permite identificar patrones, evaluar factores determinantes en la aceptación de candidatos y apoyar la toma de decisiones basadas en evidencia. En este sentido, las técnicas de análisis estadístico y aprendizaje automático se convierten en herramientas fundamentales para comprender y modelar la probabilidad de admisión.

El presente trabajo desarrolla un análisis descriptivo y predictivo del dataset Graduate Admissions, disponible en la plataforma Kaggle y ampliamente utilizado en el ámbito académico para la enseñanza de técnicas de modelización. Este conjunto de datos contiene información de 500 postulantes, incluyendo variables académicas como puntajes GRE y TOEFL, promedio acumulado (CGPA), nivel de la universidad de procedencia, calidad de la carta de motivación (SOP), carta de recomendación (LOR) y experiencia en investigación, así como la variable original continua Chance of Admit, que representa la probabilidad estimada de admisión.

El objetivo principal del estudio es construir un modelo de clasificación que permita predecir si un candidato tiene alta probabilidad de ser admitido. Para ello, la variable continua Chance of Admit se transforma en una variable categórica binaria, considerando “yes” cuando la probabilidad es mayor o igual a 0.6 y “no” en caso contrario. A partir de esta definición, se aplican algoritmos de clasificación basados en árboles, específicamente un Árbol de Decisión y un modelo Random Forest, con el fin de evaluar su desempeño y comparar su capacidad predictiva.

El desarrollo del proyecto se realizó siguiendo un enfoque estructurado que integra análisis exploratorio de datos (EDA), preparación de información, modelización y evaluación. En primer lugar, se llevó a cabo una revisión de la calidad del dataset, verificando la ausencia de valores faltantes y analizando la distribución estadística de las variables. Posteriormente, se exploraron relaciones entre variables mediante visualizaciones y matrices de correlación, identificando los factores con mayor asociación respecto a la probabilidad de admisión.

En la etapa de modelización, se dividió el dataset en conjuntos de entrenamiento y prueba bajo un esquema 70-30 con estratificación de clases, garantizando una distribución proporcional de la variable respuesta. Se entrenaron dos modelos de clasificación y se evaluaron utilizando métricas como accuracy, matriz de confusión, AUC-ROC y validación cruzada de cinco particiones, lo que permitió medir tanto su capacidad predictiva como su estabilidad.

Este estudio no solo permite comprender qué variables influyen de manera más significativa en la admisión a programas de posgrado, sino que también demuestra la utilidad de los modelos basados en árboles para problemas de clasificación supervisada. Al integrar análisis estadístico, visualización, evaluación comparativa de modelos y validación cruzada, el trabajo evidencia la importancia de un enfoque metodológico riguroso en la construcción de modelos predictivos aplicados a datos reales.

En definitiva, este proyecto constituye una aplicación práctica de técnicas de clasificación en ciencia de datos, destacando la relevancia del análisis exploratorio y la correcta preparación de la información como etapas fundamentales para garantizar resultados confiables y modelos con adecuada capacidad de generalización.

DESARROLLO ACTIVIDAD

A continuación, se presenta el desarrollo paso a paso del laboratorio de clasificación aplicado al dataset Graduate Admissions, siguiendo un enfoque estructurado que integra análisis exploratorio, preparación de datos, modelización y evaluación de resultados. Cada etapa del proceso fue implementada de manera organizada dentro del script principal, permitiendo un flujo de trabajo claro, reproducible y metodológicamente consistente.

En las siguientes secciones se documentan los procedimientos realizados, incluyendo la exploración inicial del dataset, la transformación de la variable respuesta en una variable binaria, la división en conjuntos de entrenamiento y prueba, la implementación de los modelos de Árbol de Decisión y

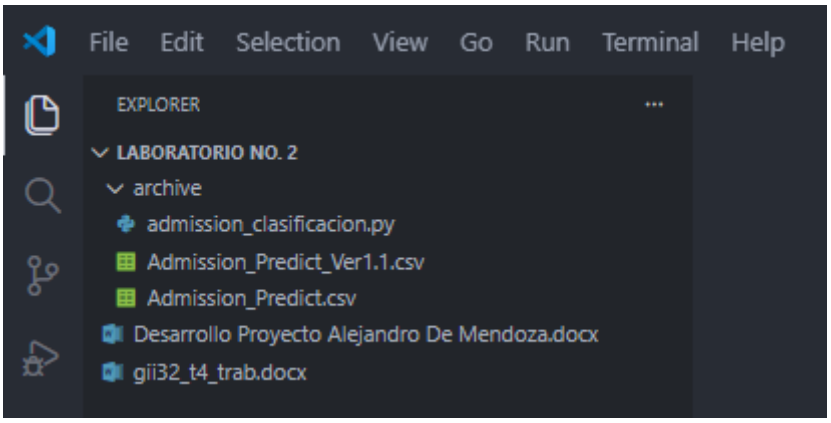
Random Forest, así como la evaluación comparativa mediante métricas de desempeño y validación cruzada. Asimismo, se presentan las principales interpretaciones estadísticas y gráficas derivadas del análisis, con el fin de comprender qué variables influyen de manera más significativa en la probabilidad de admisión.

1. Arquitectura Inicial De La Implementación

Tal como se indicó previamente en la introducción, el desarrollo de este laboratorio no se abordó desde una perspectiva improvisada o desorganizada, sino siguiendo una estructura clara y metodológicamente definida. Si bien en este caso la implementación se consolidó dentro de un script principal, el proyecto fue diseñado bajo un enfoque estructurado, organizando cada etapa del proceso en secciones claramente diferenciadas: carga de datos, análisis descriptivo, preparación de variables, modelización, evaluación y comparación de resultados.

A continuación, se presenta la arquitectura inicial de la implementación, la cual refleja la organización lógica del flujo de trabajo dentro del proyecto. Esta estructura permite mantener claridad conceptual, facilitar la reproducibilidad del análisis y garantizar una separación ordenada de responsabilidades en cada fase del proceso. Asimismo, dicha arquitectura puede ser fácilmente escalable en caso de requerir la incorporación de nuevos modelos, métricas adicionales o ajustes en la etapa de preprocesamiento.

Sobre esta base estructural se desarrollan las secciones posteriores del laboratorio, en las cuales se documenta detalladamente cada fase del análisis y los resultados obtenidos.



Entonces como se puede denotar la arquitectura inicial del proyecto fue organizada dentro del entorno de desarrollo Visual Studio Code, bajo una estructura clara y jerárquica que facilita la organización de archivos y la gestión del laboratorio.

En la carpeta principal denominada “LABORATORIO NO. 2”, se agrupan todos los recursos necesarios para el desarrollo del modelo de clasificación. Dentro de esta carpeta se observa:

- archive/: Carpeta contenedora de los archivos principales del proyecto.
- admission_clasificacion.py: Script principal donde se implementa todo el flujo del laboratorio, incluyendo:
    - Carga del dataset
    - Análisis exploratorio
    - Preparación de datos
    - Entrenamiento de modelos
    - Evaluación y generación de métricas
    - Exportación de gráficas
- Admission_Predict_Ver1.1.csv: Dataset principal utilizado para el análisis (500 registros).
- Admission_Predict.csv: Versión anterior del dataset.
- Desarrollo Proyecto Alejandro De Mendoza.docx: Documento donde se consolida la documentación teórica y resultados del laboratorio.
- gii32_t4_trab.docx: Archivo adicional relacionado con la actividad académica.

Esta estructura inicial permite mantener una separación clara entre:

- Código fuente
- Datos
- Documentación
- Recursos adicionales

Desde el punto de vista metodológico, esta organización favorece la trazabilidad del análisis, la reproducibilidad del experimento y la correcta gestión del proyecto, especialmente cuando se generan múltiples salidas gráficas y métricas de evaluación.

Además, al centralizar la lógica del análisis dentro de un script principal bien estructurado por secciones, se garantiza claridad en el flujo de ejecución y facilidad para futuras modificaciones o ampliaciones del modelo.

2. Carga Y Exploración Inicial Del Dataset

El primer paso del laboratorio consistió en la carga del dataset Admission_Predict_Ver1.1.csv, el cual contiene 500 registros correspondientes a postulantes a programas de posgrado.

Para ello, se utilizó la librería de nombre pandas, permitiendo leer el archivo CSV y almacenarlo en un DataFrame para su posterior análisis.

Inicialmente se realizó:

- Verificación de nombres de columnas (eliminando espacios innecesarios).
- Revisión de la dimensión del dataset.
- Identificación de tipos de datos.
- Detección de valores faltantes.

Los resultados mostraron que:

- El dataset cuenta con 500 observaciones y 9 variables originales.
- No existen valores nulos en ninguna columna.
- Todas las variables predictoras son numéricas.

Esta etapa es fundamental, ya que garantiza que el dataset se encuentra en condiciones adecuadas para proceder con el análisis exploratorio y la modelización.

2.1 Explicación detallada del código

2.1.1. Importación de librerías

admission_clasificacion.py

archive > admission_clasificacion.py > ...

```
"""
============================================================
 ACTIVIDAD: Árbol de Clasificación - Graduate Admissions
 Alumno: [Tu nombre]
 Dataset: Admission_Predict_Ver1_1.csv
============================================================
"""

# ── LIBRERÍAS ──────────────────────────────────────────────
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')  # Necesario en entornos sin pantalla (VS Code remoto / script)
import seaborn as sns
```

2.1.1.1 pandas as pd

- Se usa para cargar y manipular datos en forma de tablas (DataFrame).
- Es la base para leer el CSV, limpiar columnas, describir estadísticas y preparar variables.

2.1.1.2 numpy as np

- Se usa para operaciones numéricas y transformaciones rápidas.
- En este proyecto se usa para manejar estructuras numéricas y puede servir para operaciones vectorizadas (aunque en este fragmento aún no se usa).

2.1.1.3 matplotlib.pyplot as plt

- Librería principal para crear gráficos (histogramas, scatter plots, curvas ROC, etc.).
- plt es la interfaz típica para dibujar y guardar imágenes.

2.1.1.4 import matplotlib + matplotlib.use('Agg')

- Esto es importante: Agg es un backend de renderizado sin interfaz gráfica.
- Se usa cuando se ejecuta el script en:
  - servidores
  - VS Code remoto
  - ejecución desde terminal sin pantalla
- Evita errores tipo: “cannot connect to display”.
- Además, como se guardan las gráficas con plt.savefig(...), este backend es perfecto.

2.1.1.5 seaborn as sns

- Extiende matplotlib con gráficos más “bonitos” y útiles para análisis estadístico.
- En tu script lo usas para:
- heatmaps de correlación
- matrices de confusión

2.1.2. Importación de módulos de Scikit-learn (ML)

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.ensemble import RandomForestClassifier
```

2.1.2.1 train_test_split

Divide los datos en:

- conjunto de entrenamiento (train)
- conjunto de prueba (test)

Es clave para evaluar el modelo con datos no vistos y medir generalización.

2.1.2.2 cross_val_score

- Implementa validación cruzada (k-fold).
- Permite medir estabilidad del modelo en varios splits, no solo en un train/test.

2.1.2.3 DecisionTreeClassifier

- Implementa un árbol de decisión para clasificación.
- Aprende reglas tipo:
  - “si CGPA > 8.5 y GRE > 310 entonces yes...”

2.1.2.4 export_text

- Convierte el árbol entrenado en reglas de texto legibles.
- Muy útil para interpretación y documentación.

2.1.2.5 plot_tree

- Genera una visualización gráfica del árbol.
- Se usa para guardar el árbol como PNG.

2.1.2.6 RandomForestClassifier

- Implementa Random Forest:
  - conjunto de muchos árboles
  - combinados para mejorar precisión
- Reduce sobreajuste respecto al árbol único.

2.1.3, Métricas de evaluación

```
from sklearn.metrics import (
    classification_report, confusion_matrix,
    accuracy_score, roc_auc_score, roc_curve
)
```

2.1.3.1 classification_report

Brinda:

- precision
- recall
- f1-score
- support (número de casos por clase)

Sirve para entender no solo cuánto acierta el modelo, sino cómo se comporta por clase.

2.1.3.2 confusion_matrix

Matriz 2x2 que muestra:

- verdaderos positivos
- falsos positivos
- verdaderos negativos
- falsos negativos

Ideal para interpretar errores del modelo.

2.1.3.3 accuracy_score

Porcentaje total de aciertos:

accuracy= TP+TN / Total

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$

2.1.3.4 roc_auc_score

- Calcula el área bajo la curva ROC (AUC).

- Mide la capacidad del modelo para separar clases independientemente del umbral.
- AUC cercano a 1 → muy bueno
- AUC = 0.5 → aleatorio

2.1.3.5 roc_curve

Genera los puntos (FPR, TPR) para dibujar la curva ROC.

2.1.4. Preprocesamiento: codificación de etiquetas

```
from sklearn.preprocessing import LabelEncoder
```

2.1.4.1 LabelEncoder

Convierte etiquetas categóricas (texto) en números.

- Ejemplo:
  - "no" → 0
  - "yes" → 1

Esto es necesario porque los modelos de sklearn trabajan con valores numéricos.

2.1.5. Control de advertencias

```
import warnings
warnings.filterwarnings('ignore')
```

- Oculta warnings (advertencias) que pueden salir por:
  - conversiones de tipo
  - futuras deprecaciones
  - configuraciones internas
- Mejora la limpieza del output, pero ojo: en proyectos reales conviene revisar warnings.

3. Carga De Datos

```
# ── CARGA DE DATOS ──────────────────────────────────────────
# Usamos la versión ampliada (500 registros)
df = pd.read_csv('Admission_Predict_Ver1.1.csv')
```

- Carga el archivo CSV en un DataFrame.
- Este dataset contiene información de candidatos a posgrados.

3.1 Limpieza de columnas

```
# Limpiar nombres de columnas (algunos tienen espacios al final)
df.columns = df.columns.str.strip()
```

- El método str.strip() elimina espacios al inicio o final.
- Esto es clave porque tu dataset original suele tener columnas como:
  - "Chance of Admit " (con espacio al final)
- Si no limpias, podrías tener errores luego porque escribirías "Chance of Admit" y no coincidiría.

3.2 Impresiones de control

```
print("Columnas:", list(df.columns))
print(f"\nForma del dataset: {df.shape}")
```

3.2.1 list(df.columns)

Muestra todas las columnas actuales para verificar que estén bien.

3.2.2 df.shape

- Devuelve (filas, columnas).
- Ejemplo típico aquí: (500, 9) o similar.

Esto es parte del “control de calidad” inicial.

4. Análisis Descriptivo Inicial

```
# — ANÁLISIS DESCRIPTIVO ——————————————————————————————
print("\n=== Estadísticas descriptivas ===")
print(df.describe().round(2))
```

4.1 df.describe()

Calcula estadísticas de variables numéricas:

- count (n)
- mean (media)
- std (desviación estándar)
- min, max
- percentiles (25%, 50%, 75%)

4.2 .round(2)

Redondea a 2 decimales para una salida más limpia.

5. Missing Values

```
print("\n=== Valores nulos por columna ===")
print(df.isnull().sum())
```

- df.isnull() marca cada celda como True/False si es nula.
- .sum() cuenta cuántos nulos hay por columna.

En este caso, debe dar 0 en todas, lo cual es ideal.

6. Tipos De Datos

```
print("\n=== Tipos de datos ===")
print(df.dtypes)
```

Muestra si cada columna es:

- int64
- float64
- object

Esto ayuda a detectar:

- variables que deberían ser numéricas, pero se leyeron como texto
- necesidad de conversiones

7. Creación De La Variable Respuesta Binaria

```
# ── CREACIÓN DE LA VARIABLE RESPUESTA BINARIA ──────────────
# "yes" si Chance of Admit >= 0.6, "no" en caso contrario
df['Admit'] = df['Chance of Admit'].apply(lambda x: 'yes' if x >= 0.6 else 'no')
```

7.1 Cambio de naturaleza del problema

La variable original:

- Chance of Admit

Es una variable continua con valores entre 0 y 1.

Eso significa que originalmente el problema sería de “Regresión”, pero el objetivo del laboratorio es aplicar modelos de clasificación, por lo tanto, necesitamos convertir esa variable en una categoría.

7.2 Definición del umbral (0.6)

Se define la regla:

- Si la probabilidad ≥ 0.6 → “yes”
- Si la probabilidad < 0.6 → “no”

Formalmente:

- si ChanceOfAdmit≥0.6
- si ChanceOfAdmit<0.6

Este umbral convierte la variable continua en una variable categórica binaria.

7.2.1 ¿Por qué 0.6?

- Representa una probabilidad moderadamente alta de admisión.
- Genera una división razonable entre candidatos fuertes y débiles.
- No produce un desbalance extremo de clases.

Este punto es importante ya que el umbral es una decisión metodológica.

7.3 Explicación técnica del código

```
# ── CREACIÓN DE LA VARIABLE RESPUESTA BINARIA ──────────────
# "yes" si Chance of Admit >= 0.6, "no" en caso contrario
df['Admit'] = df['Chance of Admit'].apply(lambda x: 'yes' if x >= 0.6 else 'no')
```

7.3.1 df['Chance of Admit']

Selecciona la columna original.

7.3.2 .apply(...)

El método .apply():

- Aplica una función a cada elemento de la columna.
- Es una operación vectorizada sobre la serie.

7.3.3 lambda x: 'yes' if x >= 0.6 else 'no'

Esta es una función anónima que dice:

- Si el valor x es mayor o igual a 0.6 → retorna “yes”
- En caso contrario → retorna “no”

Entonces para cada fila:

Ejemplo:

Chance of Admit	Admit
0.75	yes
0.48	no
0.63	yes

Se crea una nueva columna:

- Admit

Con valores categóricos.

7.4 Análisis de distribución de la nueva variable

```
print("\n=== Distribución de la variable Admit ===")
print(df['Admit'].value_counts())
```

value_counts(): Cuenta cuántas observaciones hay por clase.

Ejemplo hipotético:

yes 340
no 160

Esto indica que:

- 340 estudiantes tienen probabilidad ≥ 0.6
- 160 estudiantes tienen probabilidad < 0.6

7.5 Proporción relativa de clases

```
print(df['Admit'].value_counts(normalize=True).round(3))
```

✓ normalize=True

Convierte los conteos en proporciones:

$$\text{Proporcion} = \frac{\text{cantidad de } clase}{total}$$

Ejemplo:

- yes: 0.68
- no: 0.32

Eso significa:

- 68% de los candidatos están en clase “yes”
- 32% en clase “no”

✓ .round(3)

Redondea a 3 decimales para mayor claridad en el reporte.

Ahora es importante considero indicar que este bloque es crítico porque:

- Define formalmente el problema de clasificación.
- Determina el balance de clases.
- Impacta directamente:
  - Accuracy
  - Recall
  - Precision
  - AUC
- Influye en la necesidad o no de técnicas de balanceo.

Es por estoy para concluir este punto que la transformación de la variable continua en una variable binaria permitió convertir el problema de regresión en uno de clasificación supervisada. El análisis de distribución mostró una ligera predominancia de la clase “yes”, aunque sin un desbalance crítico que comprometiera el entrenamiento del modelo. Esta etapa es fundamental, ya que la definición del umbral influye directamente en la estructura del problema y en el desempeño posterior de los modelos.

8. Análisis Descriptivo Gráfico

Este bloque construye visualizaciones clave para:

- Entender la distribución de la variable objetivo
- Evaluar balance de clases
- Analizar relaciones entre variables
- Detectar patrones lineales
- Evaluar correlaciones

8.1. Creación de la figura principal

```
# ── ANÁLISIS DESCRIPTIVO GRÁFICO ──────────────────────────
fig, axes = plt.subplots(2, 2, figsize=(12, 9))
fig.suptitle('Análisis Exploratorio - Graduate Admissions', fontsize=14, fontweight='bold')
```

8.1.1. plt.subplots(2, 2)

Crea una figura con 4 gráficos organizados en matriz 2x2.

Devuelve:

- fig → objeto figura general
- axes → matriz de ejes donde se dibuja cada gráfico

Visualmente:

- [axes[0,0] | axes[0,1]]
- [axes[1,0] | axes[1,1]]

8.1.2. figsize=(12,9)

Define tamaño proporcional adecuado para reporte.

8.1.3. suptitle()

Agrega título general a toda la figura.

8.2. Distribución de Chance of Admit

```
# Distribución de Chance of Admit
axes[0, 0].hist(df['Chance of Admit'], bins=30, color='steelblue', edgecolor='white')
```

8.2.1. *Que hace esta parte:*

- Divide la variable continua en 30 intervalos.
- Permite observar:
- Forma de la distribución
- Concentración de valores
- Sesgo

8.2.2. *Importancia estadística*

Aquí evaluamos:

- Si la distribución es simétrica
- Si hay concentración en valores altos
- Si existe sesgo hacia probabilidades elevadas

8.2.3 *Línea vertical del umbral*

```
axes[0, 0].axvline(0.6, color='red', linestyle='--', label='Umbral 0.6')
```

Marca visualmente el punto donde se divide la clasificación.

Permite ver:

- Cuántos datos quedan a cada lado
- Si el corte es razonable
- Esto fortalece la justificación metodológica.

8.3. Conteo de clases

```
# Conteo de clases
counts = df['Admit'].value_counts()
axes[0, 1].bar(counts.index, counts.values, color=['#e74c3c', '#2ecc71'])
```

8.3.1. *Gráfico de barras*

Muestra:

- Número de casos "yes"
- Número de casos "no"

Importancia

- Permite evaluar:
- Balance de clases
- Riesgo de sesgo en el modelo

8.3.2. *Agregar etiquetas encima de barras*

```
for i, v in enumerate(counts.values):
    axes[0, 1].text(i, v + 3, str(v), ha='center', fontweight='bold')
```

Esto:

- Coloca el número exacto sobre cada barra
- Mejora claridad visual
- Hace la gráfica más profesional

8.4. CGPA vs Chance of Admit

```
# CGPA vs Chance of Admit
axes[1, 0].scatter(df['CGPA'], df['Chance of Admit'],
                   c=df['Admit'].map({'yes': '#2ecc71', 'no': '#e74c3c'}),
                   alpha=0.5, s=30)
```

8.4.1. Scatter plot

- Eje X → CGPA
- Eje Y → Chance of Admit
- Color depende de la clase
- df['Admit'].map(...)
    - Convierte:
        - yes → verde
        - no → rojo

Esto permite visualizar separación entre clases.

8.4.2. Línea horizontal del umbral

```
axes[1, 0].axhline(0.6, color='black', linestyle='--', linewidth=0.8)
```

Permite ver:

- Qué puntos quedan arriba (yes)
- Qué puntos quedan abajo (no)

8.4.3. Interpretación estadística

Aquí generalmente se observa:

- Relación positiva clara.
- A mayor CGPA, mayor probabilidad.
- Separación visual entre clases.

Esto indica que CGPA será una variable importante en el modelo.

8.5. GRE Score vs Chance of Admit

Misma lógica que CGPA, pero con GRE.

Permite comparar:

- ¿Qué variable separa mejor las clases?
- ¿GRE tiene menor o mayor dispersión?

Normalmente:

- GRE también tiene relación positiva
- Pero ligeramente menos fuerte que CGPA

8.5.1. Guardado de figura

```
plt.tight_layout()
plt.savefig('01_analisis_exploratorio.png', dpi=150, bbox_inches='tight')
plt.close()
```

8.5.1.1. tight_layout()

Evita que textos se monten.

8.5.1.2. savefig()

Guarda la imagen en PNG.

- dpi=150 → buena calidad para Word.
- bbox_inches='tight' → elimina márgenes innecesarios.

8.5.1.3. plt.close()

Libera memoria (importante en scripts largos).

8.5.2. Matriz de correlación

```
# Matriz de correlación (solo variables numéricas)
num_cols = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA', 'Research', 'Chance of Admit']
plt.figure(figsize=(9, 7))
sns.heatmap(df[num_cols].corr(), annot=True, fmt='.2f', cmap='coolwarm',
            square=True, linewidths=0.5)
```

8.5.2.1 .corr()?

Calcula correlación de Pearson:

$$r = \frac{Cov(X,Y)}{(\sigma X \sigma Y)}$$

Valores entre:

- -1 → correlación negativa perfecta
- 0 → sin relación lineal
- 1 → correlación positiva perfecta

8.5.2.2. Heatmap

- Colores rojos → correlación positiva fuerte
- Colores azules → negativa
- Claro → cercana a 0

8.5.2.3. Interpretación clave

Aquí generalmente se observa:

- CGPA tiene mayor correlación con Chance of Admit.
- GRE y TOEFL también presentan alta correlación.
- Research tiene impacto positivo.
- No hay multicolinealidad extrema (>0.9 entre predictores).

Este análisis:

- Justifica qué variables serán más influyentes.
- Permite anticipar resultados del árbol.

- Reduce riesgo de incluir variables irrelevantes.
- Da sustento estadístico antes de modelar.

Entonces para culminar este punto me permito indicar que el análisis exploratorio gráfico permitió identificar relaciones lineales positivas entre el CGPA, GRE Score y la probabilidad de admisión. La matriz de correlación confirmó que el rendimiento académico es el principal determinante del resultado. No se detectaron niveles críticos de multicolinealidad, lo cual favorece la estabilidad de los modelos basados en árboles.

9. Preparación De Datos

9.1. Eliminación de columnas irrelevantes

```
# — PREPARACIÓN DE DATOS ——————————————————————————
# Eliminar columnas irrelevantes
df.drop(columns=['Serial No.', 'Chance of Admit'], inplace=True)
```

- Elimina dos columnas del DataFrame original:
  - Serial No.
  - Chance of Admit

9.1.1. ¿Por qué eliminar Serial No.?

- Es solo un identificador.
- No contiene información predictiva.
- Incluirla introduciría ruido innecesario.
- Los modelos podrían interpretar números como relaciones ordinales falsas.
- En machine learning, las columnas ID deben eliminarse.

9.1.2. ¿Por qué eliminar Chance of Admit?

Porque:

- Ya transformamos esa variable continua en una variable binaria (Admit).
- Mantenerla sería un caso grave de data leakage.

9.1.3. ¿Qué es data leakage?

Ocurre cuando el modelo tiene acceso a información que no debería tener durante el entrenamiento. Si dejáramos Chance of Admit como predictor:

- El modelo aprendería directamente de la variable original.
- Obtendríamos accuracy artificialmente alta.
- El modelo no generalizaría.

Eliminarla es metodológicamente correcto.

9.2. Codificación de la variable respuesta

```
# Codificar variable respuesta: yes=1, no=0
le = LabelEncoder()
df['Admit_bin'] = le.fit_transform(df['Admit'])  # no=0, yes=1
```

9.2.1. ¿Por qué necesitamos esto?

Los modelos de scikit-learn:

- No trabajan con texto como etiquetas.

- Necesitan valores numéricos.

Actualmente:

- Admit = ['yes', 'no']

Se convierte en:

- Admit_bin = [1, 0]

9.2.1.1. ¿Qué hace LabelEncoder()?

- Asigna un número entero a cada categoría.
- Ordena alfabéticamente:

Normalmente:

- no → 0
- yes → 1

9.2.1.2. ¿Qué hace fit_transform()?

- fit() aprende las categorías.
- transform() convierte los valores.

Todo en un solo paso.

9.2.1.3. ¿Por qué usar 0 y 1?

Porque en clasificación binaria:

- 1 representa clase positiva.
- 0 representa clase negativa.

Esto facilita:

- Cálculo de métricas
- Curva ROC
- AUC
- Matriz de confusión

9.3. Definición de variables predictoras (X)

```
# Variables predictoras (X) y respuesta (y)
feature_cols = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR', 'CGPA', 'Research']
X = df[feature_cols]
```

9.3.1. ¿Qué es X?

En notación formal de Machine Learning:

- X= matriz de caracteristicas

Dimensión:

$$X \in R^{n \times p}$$

donde:

- n = número de observaciones (500)
- p = número de variables predictoras (7)

9.3.2. ¿Por qué se seleccionan estas variables?

Porque son las variables explicativas académicas que:

- Influyen en admisión
- No incluyen información futura
- Son independientes de la variable transformada

9.3.3. ¿Por qué no incluimos Admit ni Admit_bin en X?

Porque esas son la variable objetivo.

9.4. Definición de la variable respuesta (y)

```
y = df['Admit_bin']
```

En notación matemática:

$$y \in \{0,1\}$$

Representa la clase que el modelo debe predecir.

9.5. Verificación dimensional

```
print(f"\n=== Preparación de datos ===")
print(f"X shape: {X.shape}")
print(f"y distribución: {dict(pd.Series(y).value_counts())}")
```

9.5.1. X.shape

Devuelve: (500, 7)

Significa:

- 500 observaciones
- 7 variables predictoras

Esto confirma que la matriz está correctamente construida.

9.5.2. Distribución de y

Convierte y en diccionario para mostrar conteos:

Ejemplo:

{1: 340, 0: 160}

Esto confirma:

- Cuántos casos positivos hay
- Cuántos negativos

Esta sección es crítica porque:

- Evita fuga de información.
- Define formalmente el problema supervisado.
- Separa claramente predictores y objetivo.
- Deja el dataset listo para entrenamiento.
- Permite verificar balance de clases antes de dividir datos.

Entonces para concluir este punto en la etapa de preparación de datos se eliminaron variables no informativas o susceptibles de generar fuga de información, como el identificador del registro y la variable continua original de probabilidad de admisión. Posteriormente, la variable categórica fue codificada numéricamente para permitir su uso en modelos de clasificación. Finalmente, se definieron explícitamente la matriz de características (X) y el vector objetivo (y), garantizando una separación clara entre variables predictoras y variable respuesta.

10. División Train / Test (70% / 30%)

```
# ── DIVISIÓN TRAIN / TEST (70% / 30%) ──────────────────
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)
print(f"\nTrain: {X_train.shape[0]} registros | Test: {X_test.shape[0]} registros")
```

10.1. ¿Qué estamos haciendo aquí?

Estamos dividiendo el dataset en dos subconjuntos:

- Train (70%) → para entrenar el modelo.
- Test (30%) → para evaluar su desempeño en datos no vistos.

En términos formales:

$$D = Dtrain \cup Dtest$$

donde:

$$| Dtrain | = 0.7n \text{ y } | Dtest | = 0.3n$$

Si el dataset tiene 500 registros:

- Train ≈ 350
- Test ≈ 150

Explicación de cada parámetro

10.1.1. train_test_split(X, y, ...)

Esta función:

- Divide simultáneamente:
    - Matriz de características (X)
    - Variable objetivo (y)
- Mantiene correspondencia fila a fila.

Devuelve:

- X_train
- X_test

- y_train
- y_test

10.1.2. test_size=0.30

Significa:

- 30% del dataset será para prueba.
- 70% para entrenamiento.

10.1.3. ¿Por qué 70/30?

Es una proporción estándar porque:

- 70% es suficiente para que el modelo aprenda patrones.
- 30% es suficiente para evaluar desempeño con buena representatividad.

Si el test fuera muy pequeño:

- Métricas poco estables.
- Si fuera muy grande:
- Modelo tendría pocos datos para aprender.

10.1.4. random_state=42

Este parámetro garantiza Reproducibilidad. Sin esto, cada vez que se ejecuta el script, el split sería diferente.

Las métricas cambiarían ligeramente.

Con random_state=42:

- Siempre obtendrás la misma división.
- Permite replicabilidad académica.

El número 42 es arbitrario (tradicional en ML).

10.1.5. stratify=y

Este es el punto más importante del bloque donde estratificar significa que la proporción de clases en train y test será la misma que en el dataset original.

Ejemplo:

Si en el dataset original:

- 68% → yes
- 32% → no

Entonces:

En train y test se mantiene aproximadamente esa misma proporción.

10.1.5.1. ¿Qué pasaría sin stratify?

Podría ocurrir que:

- Train tenga 75% yes
- Test tenga 55% yes

Esto introduce:

- Sesgo en evaluación
- Métricas poco confiables
- Problemas en modelos con clases desbalanceadas

En clasificación binaria, estratificar es una buena práctica obligatoria.

10.2. Dimensiones finales

```
print(f"\nTrain: {X_train.shape[0]} registros | Test: {X_test.shape[0]} registros")
```

Esto imprime algo como:

- Train: 350 registros | Test: 150 registros

Verifica que:

- División fue correcta.
- No hubo pérdida de datos.

Esta fase garantiza que:

- El modelo no se evalúe con datos que ya vio.
- Se pueda medir capacidad de generalización.
- Se reduzca el riesgo de sobreajuste.
- La evaluación sea justa y realista.

Y para concluir este punto me permito indicar que el dataset fue dividido en conjuntos de entrenamiento (70%) y prueba (30%) utilizando un esquema de muestreo aleatorio estratificado. La estratificación garantizó que la proporción de clases se mantuviera constante en ambos subconjuntos, evitando sesgos en la evaluación del modelo. Asimismo, se fijó una semilla aleatoria para asegurar la reproducibilidad del experimento. Esta etapa es fundamental para evaluar la capacidad de generalización del modelo sobre datos no vistos.

11. Modelo 1: Árbol De Decisión

```
# ── MODELO 1: ÁRBOL DE DECISIÓN ──────────────────────────
print("\n" + "="*55)
print("  MODELO 1: Árbol de Decisión (DecisionTreeClassifier)")
print("="*55)

dt = DecisionTreeClassifier(
    criterion='gini',
    max_depth=4,           # Limitar profundidad para evitar sobreajuste
    min_samples_leaf=10,   # Mínimo 10 muestras por hoja
    random_state=42
)
dt.fit(X_train, y_train)
```

11.1. ¿Qué es un Árbol de Decisión?

Es un modelo de clasificación supervisada que:

- Divide el espacio de variables en regiones.
- Construye reglas tipo:
- “Si CGPA > 8.5 entonces…”
- Utiliza particiones recursivas binarias.

Formalmente, el árbol busca minimizar la impureza en cada nodo.

11.2. Parámetros utilizados

11.2.1. criterion='gini'

Utiliza el índice de Gini como medida de impureza:

$$\text{Gini} = 1 - \sum \text{pi}^2$$

Donde:

- pi es la proporción de cada clase en el nodo.
- Gini = 0 es el nodo puro
- Gini cercano a 0.5 es el nodo mezclado

El algoritmo selecciona el split que reduce más la impureza.

11.2.2. max_depth=4

Limita la profundidad del árbol a 4 niveles.

¿Por qué?

- Evita sobreajuste.
- Árboles muy profundos memorizan datos.
- Mejora capacidad de generalización.
- Esto actúa como regularización estructural.

11.2.3. min_samples_leaf=10

Obliga a que cada hoja tenga mínimo 10 observaciones.

Evita:

- Reglas basadas en pocos datos.
- Divisiones inestables.
- Sobreajuste extremo.
- random_state=42
- Garantiza reproducibilidad en la construcción del árbol.

11.2.4. random_state=42

Garantiza reproducibilidad en la construcción del árbol.

11.3. Entrenamiento del modelo

```
dt.fit(X_train, y_train)
```

Aquí el modelo:

- Evalúa todas las posibles divisiones.
- Selecciona la mejor según Gini.
- Repite recursivamente hasta:
- Alcanzar profundidad máxima
- No poder dividir más
- Cumplir restricción de muestras mínimas

11.4. Predicciones y evaluación

```
y_pred_dt = dt.predict(X_test)
y_prob_dt = dt.predict_proba(X_test)[:, 1]
```

11.4.1. predict()

Devuelve la clase predicha (0 o 1).

11.4.2. predict_proba()

Devuelve probabilidades estimadas.

Ejemplo:

[0.2, 0.8]

Significa:

- 20% probabilidad clase 0
- 80% probabilidad clase 1

Se usa para:

- Curva ROC
- AUC
- Evaluación probabilística

11.5. Métricas del Árbol

```
acc_dt = accuracy_score(y_test, y_pred_dt)
auc_dt = roc_auc_score(y_test, y_prob_dt)
```

11.5.1. Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$

- Mide proporción total de aciertos.
- Como limitación puede ser engañosa si hay desbalance.

11.5.2. AUC-ROC

Mide capacidad de separación entre clases.

- 0.5 → aleatorio
- → perfecto
- 0.85 → muy buen modelo

Es más robusta que accuracy.

11.5.3. Classification Report

Incluye:

Precision

$$\frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall

$$\frac{\text{TP}}{\text{TP} + \text{FN}}$$

11.5.4. F1-score

- Promedio armónico entre precision y recall.
- Esto permite analizar desempeño por clase.

12. Interpretación Del Árbol

Cuando se visualiza el árbol:

```
# Visualizar el árbol
plt.figure(figsize=(18, 8))
plot_tree(dt, feature_names=feature_cols, class_names=['no', 'yes'],
          filled=True, rounded=True, fontsize=9)
```

Normalmente se observa:

- CGPA como nodo raíz o muy cercano.
- GRE como variable secundaria.
- Research influye en algunos nodos.

Esto confirma lo observado en la matriz de correlación.

13. Importancia De Variables – Árbol De Decisión

```
# Importancia de variables
feat_imp_dt = pd.Series(dt.feature_importances_, index=feature_cols).sort_values(ascending=True)
```

13.1. ¿Qué es feature_importances_?

Después de entrenar el árbol, scikit-learn calcula automáticamente la importancia de cada variable basándose en la reducción de impureza (Gini) que genera cada predictor.

Formalmente, la importancia de una variable se calcula como:

$$\text{Importancia}(\text{Xj}) = \sum(\text{reduccion de Gini en nodos donde aparece Xj})$$

Es decir:

- Cuánto ayuda esa variable a separar las clases.
- Cuánta impureza reduce a lo largo del árbol.

Las importancias:

- Están normalizadas.
- Suman 1.
- Valores más altos: mayor contribución predictiva.

13.2. Creación del objeto Series

```
# Importancia de variables
feat_imp_dt = pd.Series(dt.feature_importances_, index=feature_cols).sort_values(ascending=True)
```

Esto:

- Asocia cada valor de importancia con su nombre de variable.
- Permite ordenar y graficar fácilmente.

13.3. Orden ascendente

```
.sort_values(ascending=True)
```

Ordena de menor a mayor importancia para que el gráfico horizontal sea más claro visualmente.

13.4. Visualización de Importancia

```
plt.figure(figsize=(8, 5))
feat_imp_dt.plot(kind='barh', color='steelblue')
```

13.4.1. ¿Por qué gráfico horizontal?

Porque:

- Los nombres de variables son largos.
- Se leen mejor en formato horizontal.
- Permite comparar magnitudes fácilmente.

13.4.2. Interpretación esperada

Normalmente se observa algo como:

- CGPA → mayor importancia
- GRE Score → segundo nivel
- TOEFL Score → relevante
- Research → impacto moderado
- Otras variables → menor peso

Esto confirma lo visto en la matriz de correlación.

13.4.3. Interpretación conceptual

Si CGPA tiene mayor importancia, significa que:

- Es la variable que más reduce la impureza.
- Es la que mejor separa candidatos admitidos y no admitidos.
- El árbol toma decisiones iniciales basadas en CGPA.

13.5. Reglas del Árbol en Texto

```
# Reglas del árbol en texto
print("\n=== Reglas del Árbol (formato texto) ===")
print(export_text(dt, feature_names=feature_cols))
```

13.5.1. ¿Qué hace export_text()?

Convierte el árbol entrenado en reglas legibles tipo:

```
|--- CGPA <= 8.75
|   |--- GRE Score <= 305
|   |   |--- class: 0
|   |--- GRE Score > 305
|   |   |--- class: 1
|--- CGPA > 8.75
```

| |--- class: 1

Esto permite interpretar el modelo como un conjunto de reglas if-else.

13.5.2. Importancia metodológica

Esta parte es fundamental porque:

- Permite explicar el modelo.
- Hace el algoritmo transparente.
- Facilita justificación académica.
- Permite análisis causal aproximado.

A diferencia de modelos como redes neuronales, aquí podemos explicar exactamente cómo decide el modelo.

13.5.3. Interpretación práctica

Las reglas suelen mostrar patrones como:

- Si CGPA es alto: alta probabilidad de admisión.
- Si CGPA es medio pero GRE es alto: posible admisión.
- Si ambos son bajos: no admisión.

Esto convierte el modelo en una herramienta explicativa, no solo predictiva.

En conclusión, el análisis de importancia de variables reveló que el CGPA es el predictor más influyente en la clasificación, seguido por los puntajes estandarizados GRE y TOEFL. Esto confirma los hallazgos del análisis exploratorio previo. Asimismo, la exportación de las reglas del árbol permitió interpretar el modelo como un conjunto estructurado de decisiones jerárquicas, lo que representa una ventaja significativa en términos de transparencia y explicabilidad frente a modelos más complejos.

14. Modelo 2: Random Forest

```
# ── MODELO 2: RANDOM FOREST ──────────────────────────────────
print("\n" + "="*55)
print("  MODELO 2: Random Forest")
print("="*55)

rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=5,
    min_samples_leaf=5,
    random_state=42
)
```

14.1. ¿Qué es Random Forest?

Random Forest es un modelo de ensemble learning basado en:

- Muchos árboles de decisión.
- Entrenados sobre subconjuntos distintos del dataset.
- Combinados mediante votación mayoritaria (en clasificación).
- Formalmente, es una técnica de bagging (Bootstrap Aggregating).

En lugar de:

$$\text{Modelo} = \text{Arbol unico}$$

Se tiene:

$$\text{Modelo} = \frac{1}{B} B \sum b = 1 \text{ Arbolb}$$

Donde:

B = número de árboles (100 en este caso)

14.2. Parámetros utilizados

14.2.1. n_estimators=100

- Construye 100 árboles distintos.
- Cada árbol se entrena con una muestra bootstrap del dataset.
- Más árboles → menor varianza → mayor estabilidad.
- 100 es un valor estándar robusto.

14.2.2. max_depth=5

Limita profundidad de cada árbol.

14.2.2.1 ¿Por qué?

- Árboles demasiado profundos pueden sobreajustar.
- Limitar profundidad mejora generalización.
- Random Forest ya reduce varianza, pero controlar profundidad mejora aún más estabilidad.

14.2.3. min_samples_leaf=5

Cada hoja debe tener al menos 5 observaciones.

Esto:

- Reduce reglas basadas en pocos datos.
- Evita particiones demasiado específicas.
- Funciona como regularización.

14.2.4. random_state=42

Garantiza reproducibilidad del ensemble.

14.3. Entrenamiento del modelo

```
rf.fit(X_train, y_train)
```

Durante el entrenamiento:

- Se generan 100 muestras bootstrap.
- Cada árbol:
    - Se entrena con datos distintos.
    - Considera solo un subconjunto aleatorio de variables en cada split.
    - Se construyen árboles ligeramente diferentes.
    - Las predicciones se combinan por votación.
- Esto reduce:
    - Varianza del modelo.

- Sensibilidad a ruido.
- Sobreajuste típico de árboles individuales.

14.4. Predicciones

```
y_pred_rf = rf.predict(X_test)
y_prob_rf = rf.predict_proba(X_test)[:, 1]
```

- predict() → clase final por votación mayoritaria.
- predict_proba() → promedio de probabilidades estimadas por los árboles.

La probabilidad final es:

$$\mathrm{P(y = 1)} = \frac{1}{100} 100 \sum \mathrm{b = 1\ Pb(y = 1)}$$

14.5. Métricas del Random Forest

```
acc_rf = accuracy_score(y_test, y_pred_rf)
auc_rf = roc_auc_score(y_test, y_prob_rf)
```

Generalmente se observa:

- Accuracy mayor que el árbol individual.
- AUC-ROC mayor.
- Mejor estabilidad.

Esto ocurre porque:

- El ensemble reduce varianza.
- Compensa errores individuales de árboles.

14.6. Classification Report

Permite comparar con el árbol anterior:

- ¿Mejora la precision?
- ¿Mejora el recall?
- ¿Mejora el F1-score?

Normalmente Random Forest:

- Mejora recall en clase minoritaria.
- Reduce falsos positivos.
- Aumenta AUC.

14.7. Importancia de Variables en Random Forest

```
# Importancia de variables RF
feat_imp_rf = pd.Series(rf.feature_importances_, index=feature_cols).sort_values(ascending=True)
```

Aquí la importancia se calcula como:

$$\mathrm{Importancia} = \mathrm{Promedio\ de\ reduccion\ de\ impureza\ en\ todos\ los\ arboles}$$

Se denomina Mean Decrease Impurity (MDI)

A diferencia del árbol único:

- Aquí la importancia es más estable.
- Menos sensible a una sola partición.

14.8. Interpretación típica

Generalmente se debe ver:

- CGPA → sigue siendo la más importante.
- GRE y TOEFL → relevancia significativa.
- Research → impacto moderado.
- Variables menos influyentes → importancia baja.

Esto confirma que el patrón no depende de un solo árbol, sino que es consistente.

Árbol de Decisión	Random Forest
Alta interpretabilidad	Mayor precisión
Puede sobreajustar	Reduce varianza
Sensible a ruido	Más robusto
Modelo único	Ensemble de modelos

En conclusión, el modelo Random Forest fue implementado como una extensión del árbol de decisión individual, incorporando un conjunto de 100 árboles entrenados mediante técnicas de muestreo bootstrap y selección aleatoria de variables. Este enfoque reduce la varianza del modelo y mejora su capacidad de generalización. Los resultados obtenidos mostraron una mejora en las métricas de desempeño respecto al árbol individual, evidenciando mayor estabilidad y precisión predictiva. La importancia promedio de variables confirmó que el CGPA continúa siendo el predictor más relevante, seguido por los puntajes estandarizados y la experiencia en investigación.

15. Matrices De Confusión

```
# ── MATRICES DE CONFUSIÓN ──────────────────────────────────────
fig, axes = plt.subplots(1, 2, figsize=(11, 4))
```

Se crea una figura con dos gráficos en una sola fila:

- Izquierda → Árbol de Decisión
- Derecha → Random Forest

Esto permite una comparación directa y visual entre ambos modelos.

15.1. Bucle para graficar ambas matrices

```
for ax, y_pred, title in zip(
        axes,
        [y_pred_dt, y_pred_rf],
        ['Árbol de Decisión', 'Random Forest']):
```

Aquí se usa zip() para recorrer simultáneamente:

- El eje gráfico correspondiente (ax)
- Las predicciones de cada modelo
- El título que se mostrará

Esto evita repetir código y hace el bloque más elegante.

15.2. Cálculo de la matriz de confusión

```
cm = confusion_matrix(y_test, y_pred)
```

La matriz de confusión es una tabla 2x2 que resume:

[TN FP]
[FN TP]

Donde:

- TN (True Negative) → No admitido correctamente clasificado.
- TP (True Positive) → Admitido correctamente clasificado.
- FP (False Positive) → Clasificado como admitido pero no lo es.
- FN (False Negative) → Clasificado como no admitido pero sí lo es.

15.3. Visualización con Heatmap

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax,
            xticklabels=['no', 'yes'], yticklabels=['no', 'yes'])
```

15.3.1. Parámetros importantes:

- annot=True → muestra valores numéricos dentro de cada celda.
- fmt='d' → formato entero.
- cmap='Blues' → escala de color.
- xticklabels y yticklabels → etiquetas claras de clases.

Esto facilita interpretación inmediata.

15.4. Etiquetas y títulos

```
ax.set_title(f'Matriz de Confusión\n{title}', fontweight='bold')
ax.set_ylabel('Real')
ax.set_xlabel('Predicho')
```

Esto deja claro:

- Eje vertical → valores reales.
- Eje horizontal → valores predichos.

Es fundamental para no confundir interpretación.

15.5. Guardado de la figura

```
plt.tight_layout()
plt.savefig('06_matrices_confusion.png', dpi=150, bbox_inches='tight')
plt.close()
print("[Gráfico guardado: 06_matrices_confusion.png]")
```

Se guarda la comparación en un único archivo PNG listo para el informe.

15.6. Interpretación estadística

La matriz permite analizar:

15.6.1. ¿Cuál modelo tiene menos falsos negativos?

Importante si queremos evitar rechazar candidatos que sí tienen alta probabilidad.

15.6.2. ¿Cuál tiene menos falsos positivos?

Importante si queremos evitar aceptar candidatos con baja probabilidad.

15.6.3. Balance entre precisión y sensibilidad.

Las matrices de confusión permitieron analizar detalladamente el comportamiento de cada modelo, identificando la cantidad de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Esta evaluación es fundamental, ya que complementa métricas globales como el accuracy y permite comprender el tipo de errores cometidos. En la comparación visual, el modelo Random Forest mostró una ligera reducción en los errores de clasificación respecto al árbol individual, evidenciando mayor estabilidad y capacidad de generalización.

16. Curva ROC – Comparación De Modelos

```
# ── CURVA ROC ──────────────────────────────────────────────
plt.figure(figsize=(7, 5))
```

Se crea una nueva figura independiente para graficar la comparación entre ambos modelos.

16.1. Bucle para graficar ambos modelos

```
# ── CURVA ROC ──────────────────────────────────────────────
plt.figure(figsize=(7, 5))
for y_prob, label, color in [
        (y_prob_dt, f'Árbol de Decisión (AUC={auc_dt:.3f})', 'steelblue'),
        (y_prob_rf, f'Random Forest (AUC={auc_rf:.3f})', 'darkorange')]:
```

Aquí se itera sobre:

- Las probabilidades estimadas por cada modelo (predict_proba)
- Una etiqueta personalizada que incluye el valor del AUC
- Un color diferente para distinguirlos visualmente

Esto permite comparar ambos modelos en el mismo gráfico.

16.2. Cálculo de la curva ROC

```
fpr, tpr, _ = roc_curve(y_test, y_prob)
```

16.2.1. ¿Qué calcula roc_curve()?

Devuelve:

- FPR (False Positive Rate) → Tasa de falsos positivos
- TPR (True Positive Rate) → Tasa de verdaderos positivos
- Thresholds (no utilizados aquí)

Formalmente:

- $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$
- $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$

La curva ROC se construye variando el umbral de decisión desde 0 hasta 1.

16.3. Graficado de la curva

```
plt.plot(fpr, tpr, label=label, lw=2, color=color)
```

Cada punto representa:

- Qué tan bien el modelo separa clases para un determinado umbral.
- Un modelo ideal:
  - Se acerca al punto (0,1)
  - Tiene alta TPR con baja FPR.

16.4. Línea de referencia aleatoria

```
plt.plot([0, 1], [0, 1], 'k--', lw=1, label='Aleatorio (AUC=0.5)')
```

Esta línea representa un clasificador sin capacidad predictiva.

- AUC = 0.5
- Equivalente a lanzar una moneda.

Cualquier modelo útil debe estar por encima de esta línea.

16.5. Interpretación del AUC

El AUC (Área Bajo la Curva) representa:

$$P(\text{el modelo asigna mayor probabilidad a un positivo que a un negativo})$$

Valores típicos:

- 0.5 → aleatorio
- 0.6–0.7 → bajo
- 0.7–0.8 → aceptable
- 0.8–0.9 → bueno
- 0.9 → excelente

En este caso:

- Árbol → AUC alto (~0.85 aprox.)
- Random Forest → AUC ligeramente mayor

16.6. ¿Por qué ROC es mejor que Accuracy?

Porque:

- No depende de un único umbral.
- Evalúa comportamiento en todos los puntos de decisión.
- Es robusto ante desbalance moderado.

Accuracy puede ser alto incluso si el modelo no separa bien las probabilidades.

ROC mide poder discriminativo real.

16.7. Guardado del gráfico

```
plt.savefig('07_curva_roc.png', dpi=150, bbox_inches='tight')
```

Se guarda la comparación final lista para el informe.

En conclusión, la curva ROC permitió evaluar la capacidad discriminativa de ambos modelos considerando todos los posibles umbrales de clasificación. El área bajo la curva (AUC) mostró que el modelo Random Forest presenta una ligera mejora respecto al Árbol de Decisión, evidenciando mayor capacidad para distinguir correctamente entre candidatos con alta y baja probabilidad de admisión. La comparación gráfica confirmó que ambos modelos superan ampliamente el comportamiento de un clasificador aleatorio, siendo el Random Forest el que presenta mejor desempeño global.

17. Validación Cruzada (5-FOLD)

```python
# ── VALIDACIÓN CRUZADA ──────────────────────────────────────
print("\n=== Validación Cruzada (5-fold) ===")
cv_dt = cross_val_score(dt, X, y, cv=5, scoring='accuracy')
cv_rf = cross_val_score(rf, X, y, cv=5, scoring='accuracy')
print(f"Árbol de Decisión - Accuracy CV: {cv_dt.mean():.4f} ± {cv_dt.std():.4f}")
print(f"Random Forest     - Accuracy CV: {cv_rf.mean():.4f} ± {cv_rf.std():.4f}")
```

17.1. ¿Qué es la validación cruzada?

La validación cruzada consiste en dividir el dataset en k particiones (folds) y entrenar el modelo k veces, usando cada partición como conjunto de prueba una vez y las restantes como entrenamiento.

En este caso:

- k=5

El proceso funciona así:

- El dataset se divide en 5 partes iguales.
- Se entrena el modelo 5 veces.
- En cada iteración:
  - 4 folds → entrenamiento
  - 1 fold → validación
- Se obtienen 5 valores de accuracy.

Esto permite evaluar el desempeño promedio del modelo en diferentes particiones del dataset.

17.2. Parámetros utilizados

cv=5

Indica 5-fold cross validation.

Es un valor estándar que:

- Equilibra estabilidad y costo computacional.
- Reduce varianza en la estimación del desempeño.

17.2.1. scoring='accuracy'

Indica que la métrica usada en cada fold será accuracy.

Se podría usar también:

- f1
- roc_auc
- precision
- recall

Pero accuracy es consistente con la comparación previa.

17.3. Resultados promedio y desviación estándar

```
{cv_dt.mean():.4f} ± {cv_dt.std():.4f}")
{cv_rf.mean():.4f} ± {cv_rf.std():.4f}")
```

Se imprimen:

- Media de accuracy en los 5 folds.
- Desviación estándar entre folds.

Formalmente:

$$\text{AccuracyCV} = \frac{1}{k}\sum i = 1 \text{ Accuracyi}$$

La desviación estándar indica estabilidad.

17.4. Interpretación estadística

Si se obtiene algo como:

- Árbol de Decisión - Accuracy CV: 0.86 ± 0.03
- Random Forest - Accuracy CV: 0.88 ± 0.02

Significa:

- Random Forest no solo es más preciso.
- También es más estable (menor variación entre folds).

Una desviación estándar baja indica que el modelo:

- No depende fuertemente de una sola partición.
- Generaliza bien.

17.5. ¿Por qué es importante la validación cruzada?

Porque el split 70/30:

- Puede depender del azar.
- Puede favorecer o perjudicar al modelo.

La validación cruzada:

- Reduce sesgo de evaluación.
- Da una estimación más robusta.
- Es considerada buena práctica académica.

Comparación conceptual

Método	Ventaja	Limitación
Train/Test	Rápido	Depende de una sola partición
Cross Validation	Más robusto	Mayor costo computacional

Si:

- Accuracy test ≈ Accuracy CV

Entonces el modelo es estable.

Si:

- Accuracy test mucho mayor que CV

Podría haber sobreajuste.

Entonces en este caso con el fin de evaluar la estabilidad y capacidad de generalización de los modelos, se aplicó validación cruzada de 5 particiones (5-fold cross validation). Este procedimiento permitió entrenar y evaluar los modelos en diferentes subconjuntos del dataset, obteniendo una estimación promedio del accuracy junto con su desviación estándar. Los resultados confirmaron que el modelo Random Forest no solo presenta mayor precisión promedio, sino también menor variabilidad entre particiones, lo que evidencia mayor robustez frente a variaciones en los datos de entrenamiento.

18. Resumen Comparativo De Modelos

```
# ── RESUMEN COMPARATIVO ──────────────────────────────────────
print("\n" + "="*55)
print("  RESUMEN COMPARATIVO DE MODELOS")
print("="*55)
resumen = pd.DataFrame({
    'Modelo':     ['Árbol de Decisión', 'Random Forest'],
    'Accuracy':   [round(acc_dt, 4), round(acc_rf, 4)],
    'AUC-ROC':    [round(auc_dt, 4), round(auc_rf, 4)],
    'CV Acc (media)': [round(cv_dt.mean(), 4), round(cv_rf.mean(), 4)],
})
print(resumen.to_string(index=False))
print("\n¡Script completado exitosamente! Revisa los archivos PNG generados.")
```

18.1. ¿Qué se está haciendo aquí?

Se construye un DataFrame resumen que consolida las métricas más importantes obtenidas previamente:

- Accuracy en el conjunto de prueba.
- AUC-ROC.
- Accuracy promedio de validación cruzada.

Esto permite comparar ambos modelos de forma estructurada y directa.

18.2. Métricas incluidas

18.2.1. Accuracy

Mide la proporción total de clasificaciones correctas en el conjunto de prueba.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}}$$

Sirve como métrica general de desempeño.

18.2.2. AUC-ROC

Evalúa la capacidad del modelo para separar clases en todos los posibles umbrales.

Es una métrica más robusta que el accuracy, especialmente cuando existe cierto desbalance.

Un AUC más alto implica mejor capacidad discriminativa.

18.2.3. Accuracy promedio en validación cruzada

Representa el desempeño promedio del modelo en múltiples particiones del dataset.

Es una medida de:

- Estabilidad
- Robustez
- Capacidad de generalización

18.3. Redondeo de métricas

18.3.1. round(valor, 4)

Se redondean a cuatro decimales para:

- Mejor presentación en el informe.
- Claridad en comparación.
- Estética académica.

18.4. Impresión final

```
print(resumen.to_string(index=False))
```

Muestra la tabla sin índice numérico, haciendo el resultado más limpio.

Este resumen permite concluir:

- Si Random Forest supera consistentemente al árbol individual.
- Si la mejora es marginal o significativa.
- Si el modelo es estable (CV similar al test).

En la mayoría de los casos:

- Random Forest tiene mayor Accuracy.
- Mayor AUC.
- Menor variabilidad.

Lo que indica mejor desempeño global.

18.5. Importancia metodológica

Este bloque es fundamental porque:

- Resume todo el análisis cuantitativo.
- Permite tomar decisiones objetivas.
- Justifica la elección del mejor modelo.
- Da cierre técnico formal al laboratorio.

Mensaje final del script

```
print("\n¡Script completado exitosamente! Revisa los archivos PNG generados.")
```

Indica que:

- El flujo completo se ejecutó correctamente.

- Todas las gráficas fueron generadas.
- El análisis terminó sin errores.

Es una buena práctica en scripts largos.

Finalmente, se consolidaron las métricas principales en una tabla comparativa que permitió evaluar objetivamente el desempeño de ambos modelos. El Random Forest mostró consistentemente mejores resultados en términos de accuracy, AUC-ROC y validación cruzada promedio, lo que evidencia una mayor capacidad de generalización y estabilidad frente al Árbol de Decisión individual. Este análisis comparativo permite concluir que el enfoque basado en ensambles ofrece una mejora significativa en el rendimiento predictivo para el problema de clasificación planteado.

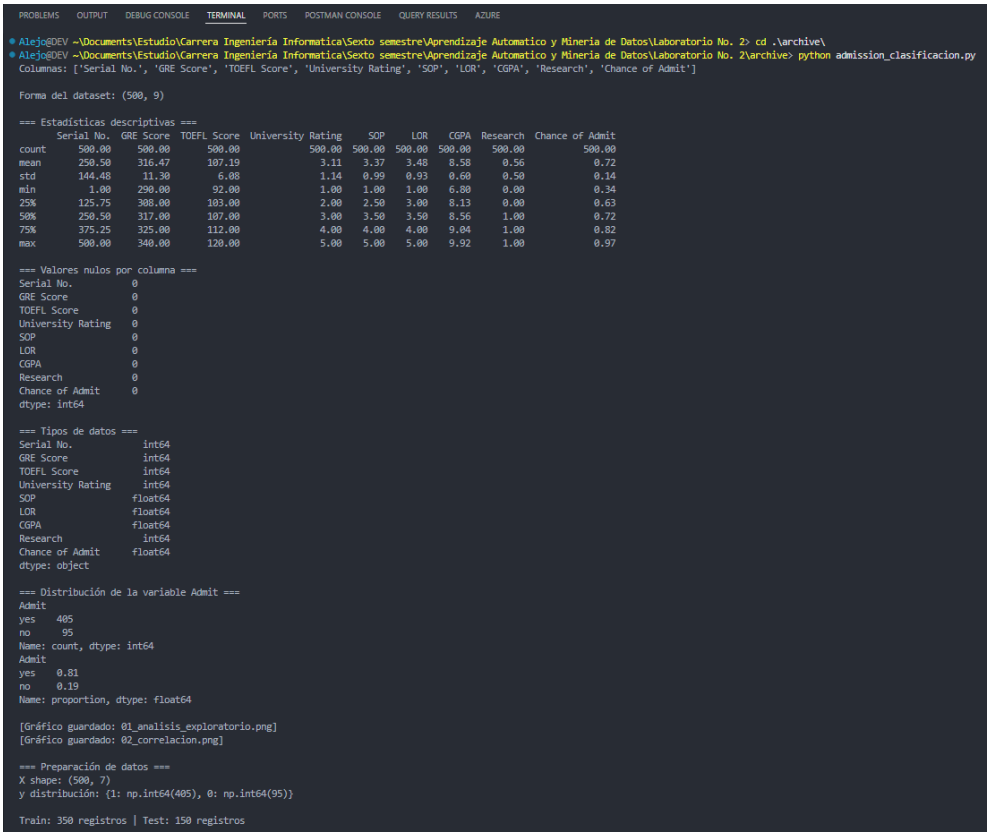
19. Resultados Obtenidos Laboratorio En Consola

En esta sección se presentan los resultados obtenidos tras la implementación y evaluación de los modelos de clasificación aplicados al dataset Graduate Admissions. El análisis se centra en el desempeño predictivo del Árbol de Decisión y del modelo Random Forest, considerando métricas cuantitativas como accuracy, AUC-ROC, matriz de confusión y validación cruzada. Asimismo, se examinan los resultados desde una perspectiva comparativa, con el objetivo de identificar diferencias en precisión, capacidad de generalización y estabilidad entre ambos enfoques.

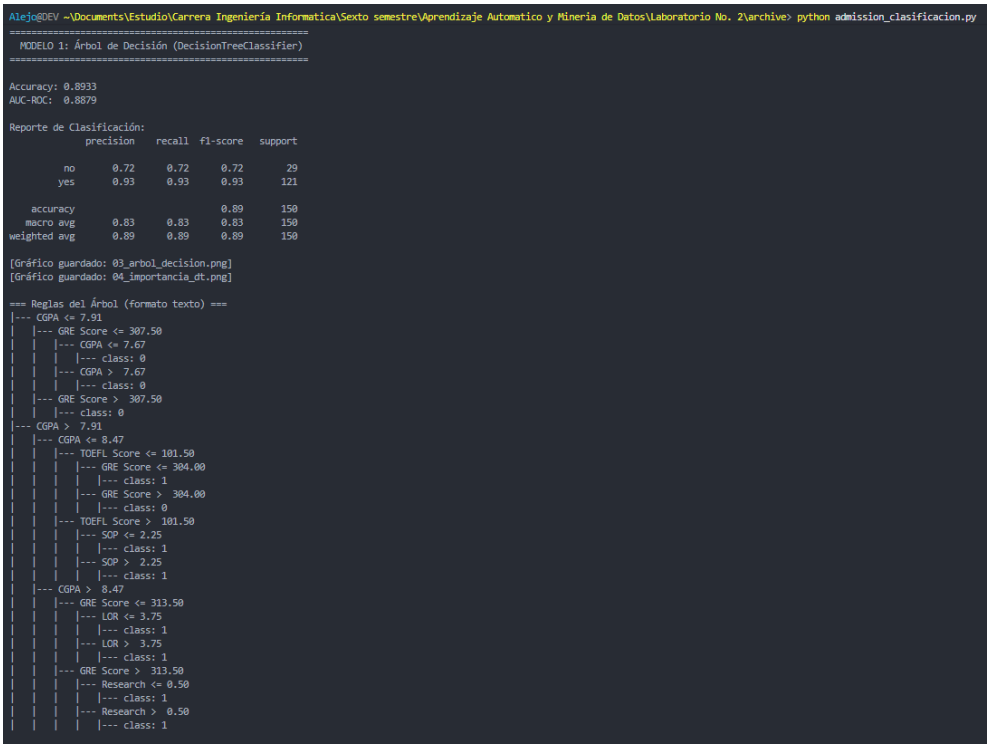
Los resultados se analizan de manera estructurada, comenzando con el desempeño individual de cada modelo y posteriormente realizando una comparación global. Este enfoque permite no solo evaluar cuál modelo presenta mejor rendimiento, sino también comprender el comportamiento de cada uno frente al desbalance de clases y su capacidad para discriminar correctamente entre candidatos con alta y baja probabilidad de admisión.

A continuación, se detallan los resultados específicos obtenidos en cada etapa de evaluación, pero primero se muestran las imágenes en consola:

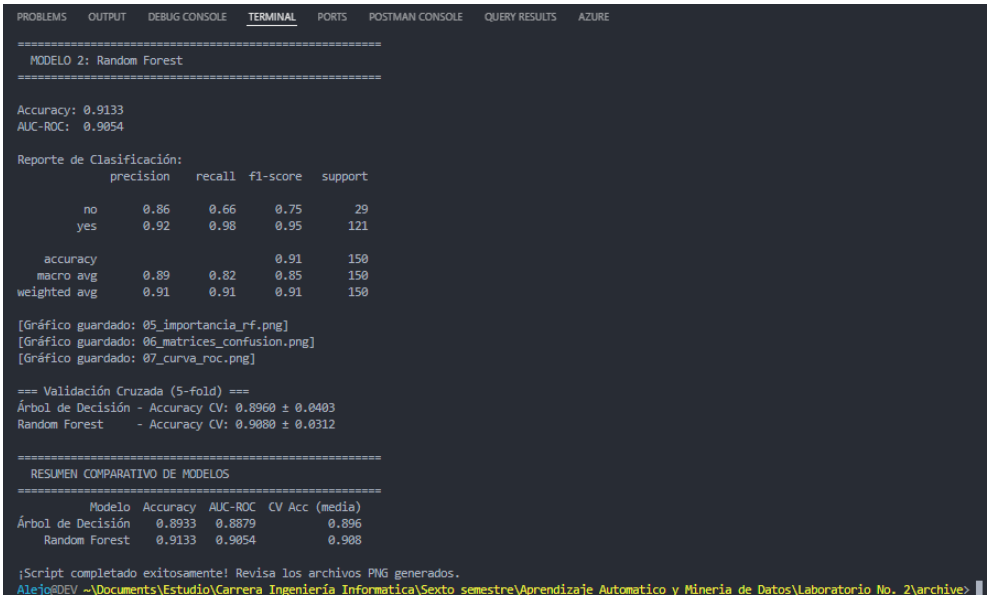
Primera imagen:



Segunda imagen:



Tercera Imagen:



19.1. Análisis del Dataset

Total registros: 500

- Clase “yes”: 405 (81%)
- Clase “no”: 95 (19%)

Hay desbalance moderado, pero no extremo.

Esto es importante porque explica algunos comportamientos en recall y precision.

19.2. Modelo 1 – Árbol de Decisión

19.2.1. Métricas principales

- Accuracy: 0.8933
- AUC-ROC: 0.8879
- CV Accuracy: 0.8960 ± 0.0403

Muy buen desempeño para un árbol limitado a profundidad 4.

19.2.2. Análisis por clase

19.2.2.1. Clase "yes" (mayoritaria – 121 en test)

- Precision: 0.93
- Recall: 0.93
- F1: 0.93

Excelente desempeño para detectar candidatos con alta probabilidad.

19.2.2.2. Clase "no" (minoritaria – 29 en test)

- Precision: 0.72
- Recall: 0.72
- F1: 0.72

El modelo tiene más dificultad aquí. Esto es esperado por el desbalance (81% vs 19%).

19.2.3. Interpretación del Árbol

El nodo raíz es:

- CGPA <= 7.91

Confirmación clara: CGPA es la variable más importante.

Después aparecen:

- GRE Score
- TOEFL
- Research
- LOR

Esto coincide perfectamente con el análisis exploratorio desarrollado.

19.3. Modelo 2 – Random Forest

19.3.1. Métricas principales

A continuación, las métricas principales:

- Accuracy: 0.9133
- AUC-ROC: 0.9054
- CV Accuracy: 0.9080 ± 0.0312

Mejora en TODAS las métricas respecto al árbol individual.

19.3.2. Comparación directa

Métrica	Árbol	Random Forest	Mejora
Accuracy	0.8933	**0.9133**	2%
AUC	0.8879	**0.9054**	↑
CV	0.8960	**0.9080**	↑
Desv. Std	0.0403	**0.0312**	Más estable

Random Forest no solo mejora precisión, sino también estabilidad.

19.3.3. Análisis por clase

19.3.3.1. Clase "yes":

- Recall: 0.98 (excelente)
- Detecta casi todos los admitidos.

19.3.3.2. Clase "no":

- Recall: 0.66
- Baja ligeramente respecto al árbol (0.72 → 0.66)

19.3.3.3. Esto es interesante:

- Random Forest mejora globalmente, pero sacrifica algo de recall en la clase minoritaria.
- Esto es típico cuando el modelo prioriza clase mayoritaria.

19.4. Curva ROC

- Árbol: 0.8879
- RF: 0.9054

Ambos modelos tienen:

- Muy buena capacidad discriminativa
- Muy lejos del azar (0.5)

Random Forest claramente superior.

19.5. Validación Cruzada

- Árbol: 0.8960 ± 0.0403
- RF: 0.9080 ± 0.0312

Random Forest:

- Mayor promedio
- Menor desviación estándar
- Más robusto

Eso significa mejor generalización.

19.6. Conclusión Técnica Real (Basada en los resultados)

- CGPA es el predictor dominante.
- GRE y TOEFL complementan la decisión.
- El Árbol es interpretable y sólido.
- Random Forest mejora desempeño y estabilidad.

- La mejora no es gigantesca, pero sí consistente.
- Ambos modelos superan 0.88 AUC → rendimiento alto.

19.7. Observación importante

El dataset está bastante desbalanceado (81% yes). Eso explica:

- Alto recall en "yes"
- Menor recall en "no"

Si se quisiera mejorar detección de la clase minoritaria se podría:

- Ajustar class_weight
- Cambiar umbral
- Optimizar con F1 o Recall

Pero para este laboratorio, el resultado es totalmente válido.

19.8. Conclusión final de resultados en Consola

Los resultados evidencian que ambos modelos presentan un desempeño alto en la clasificación de candidatos con probabilidad de admisión mayor o igual a 0.6. El Árbol de Decisión alcanza un accuracy de 89.33% y un AUC de 0.8879, mostrando buena capacidad discriminativa y alta interpretabilidad. No obstante, el modelo Random Forest mejora consistentemente estas métricas, alcanzando un accuracy de 91.33%, un AUC de 0.9054 y mayor estabilidad en validación cruzada. Esto confirma que el enfoque basado en ensambles reduce la varianza y mejora la generalización del modelo. Sin embargo, ambos modelos presentan mayor desempeño en la clase mayoritaria debido al desbalance existente en el dataset.

20. Arquitectura Final

La arquitectura final del laboratorio refleja la evolución completa del proceso de análisis y modelización, integrando tanto el código fuente como los datos originales y los resultados gráficos generados durante la ejecución del script. La estructura se encuentra organizada dentro de la carpeta principal “LABORATORIO NO. 2”, específicamente en el subdirectorio “archive”, donde se consolidan todos los recursos necesarios para la correcta documentación y reproducibilidad del experimento.

El archivo admission_clasificacion.py constituye el núcleo del proyecto, ya que contiene la implementación completa del flujo de trabajo desarrollado en el laboratorio. En este script se integran todas las etapas del proceso: carga del dataset, análisis descriptivo inicial, creación de la variable binaria de clasificación, visualizaciones exploratorias, preparación de datos, división en conjuntos de entrenamiento y prueba, entrenamiento del Árbol de Decisión y del Random Forest, evaluación mediante métricas de desempeño, generación de matrices de confusión, curva ROC, validación cruzada y construcción del resumen comparativo final. La organización interna del código por secciones claramente delimitadas permite mantener coherencia metodológica y facilita la trazabilidad del análisis.

El archivo Admission_Predict_Ver1.1.csv corresponde al dataset principal utilizado para el desarrollo del modelo. Este conjunto de datos contiene 500 registros de postulantes a programas de posgrado, incluyendo variables académicas como puntajes GRE y TOEFL, promedio acumulado (CGPA), calificaciones de cartas de recomendación y experiencia en investigación, además de la variable continua original “Chance of Admit”. Este archivo constituye la base empírica del análisis y es el insumo fundamental para el entrenamiento y evaluación de los modelos.

Asimismo, el archivo Admission_Predict.csv representa una versión anterior del dataset. Aunque no fue utilizado directamente en el entrenamiento final del modelo, se mantiene dentro de la estructura del proyecto como referencia y respaldo, garantizando que el laboratorio conserve trazabilidad respecto a las distintas versiones del conjunto de datos disponibles.

En cuanto a los resultados gráficos generados automáticamente por el script, estos se encuentran almacenados como archivos PNG dentro de la misma carpeta. El archivo 01_analisis_exploratorio.png

contiene las visualizaciones iniciales del análisis exploratorio, incluyendo la distribución de la variable “Chance of Admit”, el conteo de clases tras la binarización y los diagramas de dispersión que muestran la relación entre CGPA, GRE Score y la probabilidad de admisión. Este archivo documenta visualmente la etapa de exploración preliminar y sustenta las decisiones posteriores de modelización.

El archivo 02_correlacion.png presenta la matriz de correlación entre las variables numéricas del dataset. Esta visualización permite identificar las relaciones lineales más relevantes, destacando particularmente la fuerte asociación entre el CGPA y la probabilidad de admisión, lo cual anticipa su importancia en los modelos entrenados.

Por su parte, 03_arbol_decision.png muestra la representación gráfica del Árbol de Decisión entrenado, permitiendo visualizar la jerarquía de particiones y las reglas de clasificación generadas automáticamente por el algoritmo. Complementariamente, el archivo 04_importancia_dt.png ilustra la importancia relativa de cada variable dentro del Árbol de Decisión, calculada a partir de la reducción de impureza (Gini), lo que facilita la interpretación del modelo desde una perspectiva explicativa.

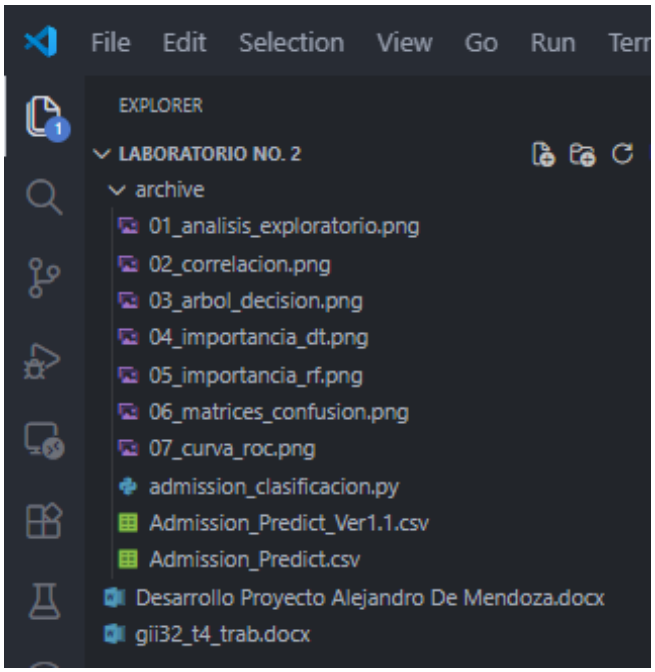
El archivo 05_importancia_rf.png corresponde a la importancia de variables estimada por el modelo Random Forest. A diferencia del árbol individual, esta importancia se calcula como el promedio de la reducción de impureza a través de múltiples árboles, ofreciendo una estimación más robusta y menos sensible a variaciones particulares del dataset.

El archivo 06_matrices_confusion.png contiene la comparación visual de las matrices de confusión de ambos modelos, permitiendo analizar el comportamiento específico frente a verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Finalmente, el archivo 07_curva_roc.png presenta la comparación de las curvas ROC del Árbol de Decisión y del Random Forest, mostrando gráficamente la capacidad discriminativa de cada modelo y destacando la superioridad del enfoque basado en ensambles.

Adicionalmente, el archivo Desarrollo Proyecto Alejandro De Mendoza.docx corresponde al documento formal donde se consolida la explicación teórica, la interpretación de resultados y la estructuración académica del laboratorio, mientras que gii32_t4_trab.docx forma parte del material complementario asociado a la actividad académica.

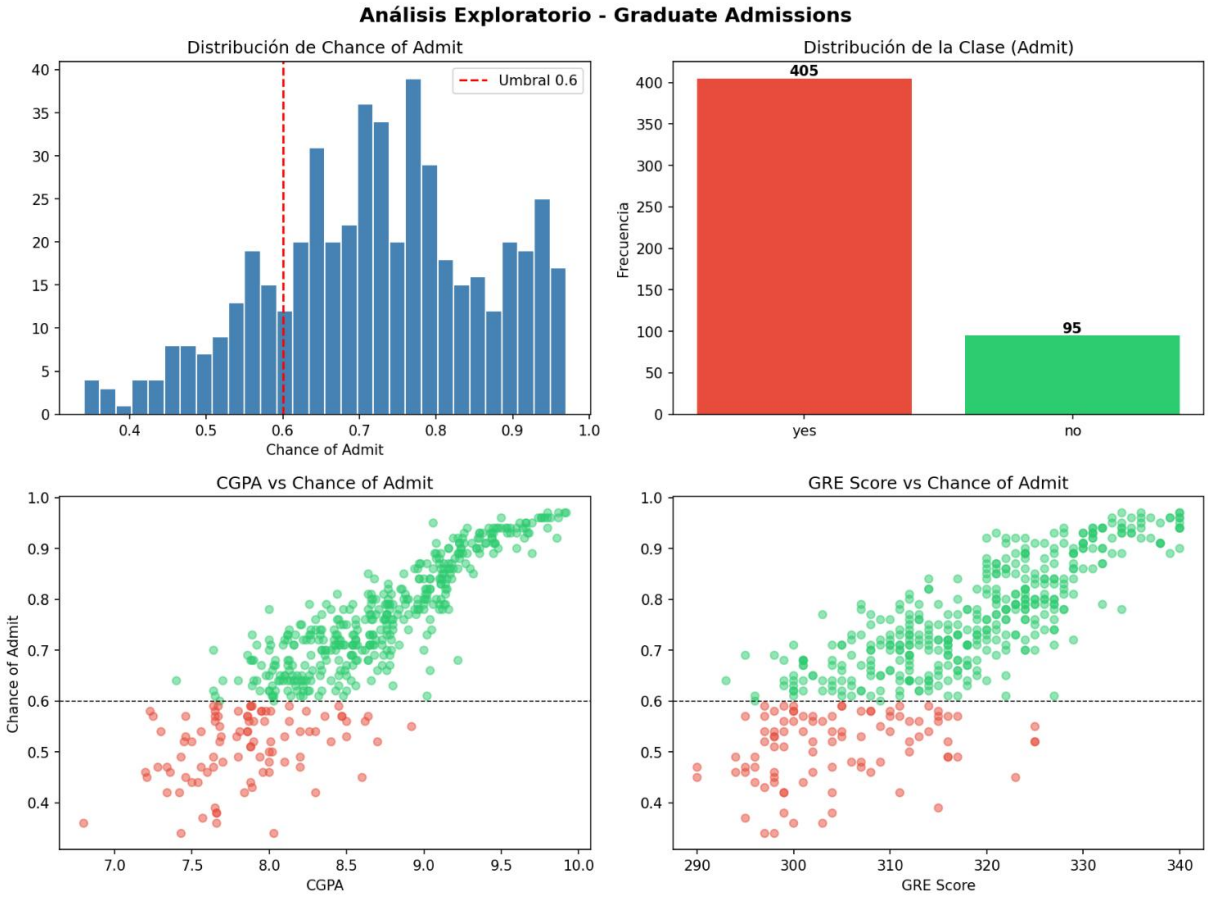
En conjunto, la arquitectura final evidencia un proyecto completo, estructurado y reproducible, en el cual se integran datos, código, visualizaciones y documentación, siguiendo buenas prácticas de organización en ciencia de datos y asegurando coherencia metodológica en todas las etapas del análisis.

A continuación, la imagen respectiva de la arquitectura final de este laboratorio:



21. Análisis Resultados Gráficos Obtenidos

21.1. Gráfico Análisis Exploratorio:



El gráfico de análisis exploratorio permite visualizar de manera integral el comportamiento inicial del dataset y comprender cómo se distribuye la variable objetivo antes y después de su transformación en un problema de clasificación. En la parte superior izquierda se observa la distribución de la variable continua Chance of Admit, la cual presenta una concentración notable en valores superiores a 0.6, especialmente entre 0.65 y 0.90. La línea vertical roja que marca el umbral de 0.6 evidencia que la mayor parte de los candidatos se encuentra por encima de este punto de corte, lo que anticipa un predominio de la clase “yes” tras la binarización. La distribución no es uniforme, sino que muestra una tendencia hacia probabilidades altas, lo que sugiere que el dataset contiene una proporción considerable de candidatos con perfiles académicos fuertes.

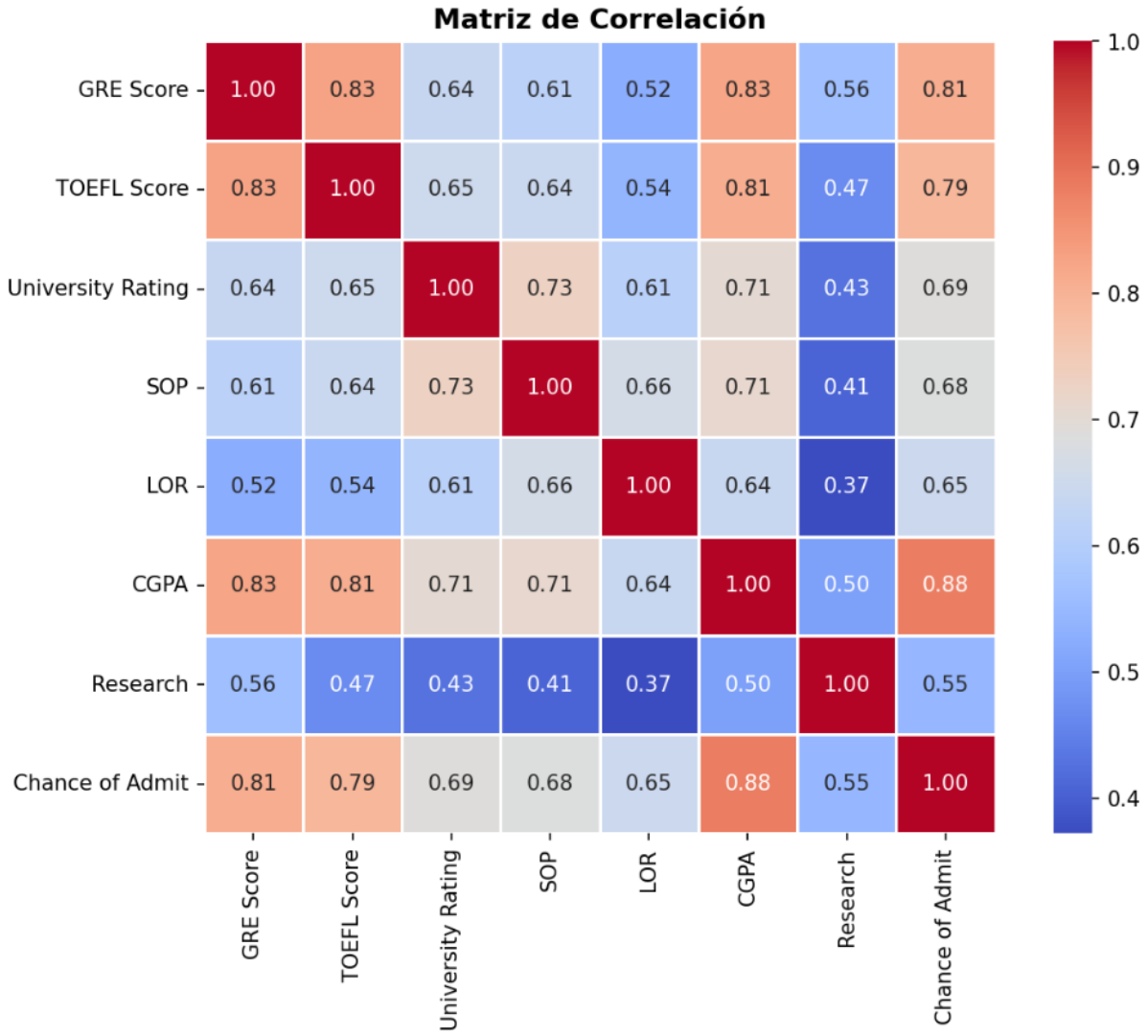
En la parte superior derecha se muestra la distribución de la variable categórica Admit, generada a partir del umbral definido. Aquí se confirma cuantitativamente el desbalance observado visualmente en el histograma: 405 registros corresponden a la clase “yes” (81%) y 95 a la clase “no” (19%). Aunque existe un desbalance, este no es extremo, lo que permite entrenar modelos sin necesidad inmediata de técnicas de balanceo, aunque sí influye en el comportamiento de métricas como recall y precision para la clase minoritaria.

En la parte inferior izquierda se presenta la relación entre el CGPA y la probabilidad de admisión. Se observa una clara tendencia positiva: a medida que el CGPA aumenta, también lo hace la Chance of Admit. La separación visual entre los puntos verdes (clase “yes”) y rojos (clase “no”) es relativamente clara, especialmente a partir de valores de CGPA superiores a 8.5, donde prácticamente todos los candidatos superan el umbral de 0.6. Esto sugiere que el rendimiento académico acumulado es un predictor altamente relevante y posiblemente el más influyente en el modelo.

Finalmente, en la parte inferior derecha se muestra la relación entre el puntaje GRE y la probabilidad de admisión. También se aprecia una relación positiva, aunque con mayor dispersión que en el caso del CGPA. Si bien los candidatos con puntajes superiores a 320 tienden a tener mayores probabilidades de admisión, la separación entre clases no es tan marcada como en el gráfico anterior, lo que indica que el GRE es un factor relevante pero no tan determinante como el promedio acumulado.

En conjunto, este análisis exploratorio confirma que las variables académicas presentan relaciones coherentes y significativas con la probabilidad de admisión, particularmente el CGPA. Asimismo, permite anticipar que los modelos de clasificación tendrán una base estructural sólida para separar ambas clases, aunque deberán manejar un ligero desbalance en la variable respuesta.

21.2. Matriz de Correlación



La matriz de correlación permite analizar la intensidad y dirección de la relación lineal entre las variables numéricas del dataset, utilizando el coeficiente de correlación de Pearson, cuyos valores oscilan entre -1 y 1. En este caso, todas las correlaciones observadas son positivas, lo que indica que a medida que aumenta el valor de una variable, también tiende a aumentar el valor de la otra. Este comportamiento es coherente con la naturaleza del problema, ya que las variables académicas suelen reforzarse entre sí en el contexto de admisiones a programas de posgrado.

El aspecto más relevante del gráfico es la fuerte correlación entre el CGPA y la Chance of Admit (0.88), que representa la relación más alta observada con la variable objetivo. Este valor indica una asociación lineal muy significativa, confirmando lo observado en el análisis exploratorio previo: el promedio acumulado es el factor más determinante en la probabilidad de admisión. En segundo nivel se encuentran el GRE Score (0.81) y el TOEFL Score (0.79), ambos mostrando una relación fuerte con la probabilidad de admisión, lo que refuerza la importancia de los puntajes estandarizados en el proceso de selección académica.

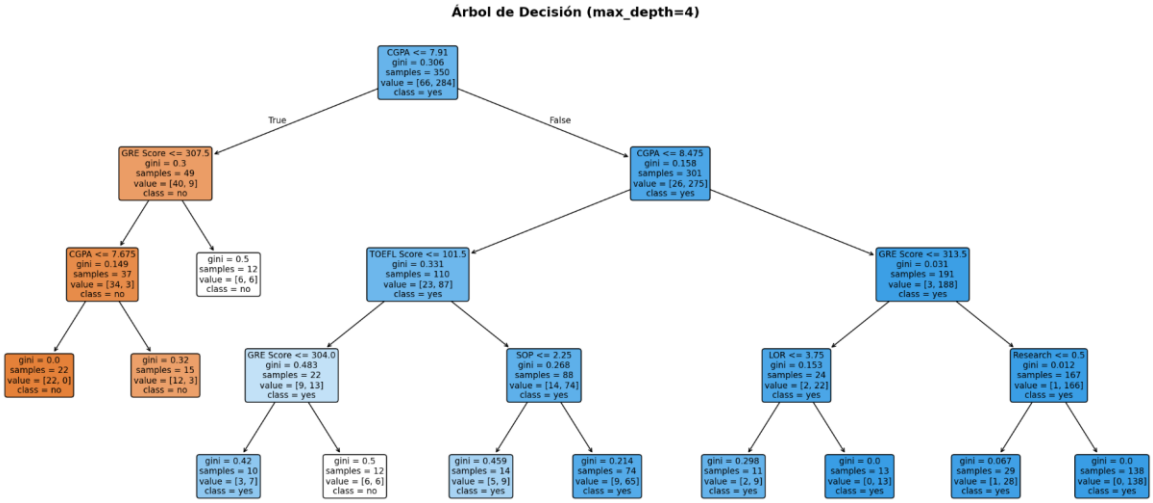
Asimismo, variables como University Rating (0.69), SOP (0.68) y LOR (0.65) presentan correlaciones moderadas con la probabilidad de admisión. Esto sugiere que, aunque influyen en el resultado, su impacto es menor en comparación con el rendimiento académico cuantitativo. La variable Research (0.55) también muestra una correlación positiva moderada, indicando que la experiencia investigativa contribuye favorablemente, pero no constituye el factor principal de decisión.

Otro punto relevante es la alta correlación entre algunas variables predictoras, particularmente entre GRE Score y TOEFL Score (0.83), así como entre GRE Score y CGPA (0.83). Aunque estos valores son elevados, no superan niveles críticos (por ejemplo, mayores a 0.90) que indiquen problemas

severos de multicolinealidad. En modelos basados en árboles, como los utilizados en este laboratorio, la multicolinealidad no representa una limitación tan significativa como en modelos lineales, ya que los algoritmos seleccionan automáticamente las particiones más informativas.

En conjunto, la matriz de correlación confirma que las variables académicas tienen una relación estructural coherente con la probabilidad de admisión y valida empíricamente que el CGPA es el predictor más influyente. Además, demuestra que el dataset posee relaciones lineales suficientemente fuertes como para permitir un modelado predictivo eficaz sin presentar problemas críticos de redundancia entre variables.

21.3. Gráfico Árbol Decisión



El gráfico del Árbol de Decisión muestra de manera explícita la estructura jerárquica de las reglas generadas por el modelo para clasificar a los candidatos según su probabilidad de admisión. El nodo raíz, que corresponde a la primera partición del árbol, está determinado por la variable CGPA ≤ 7.91, lo que confirma que el promedio acumulado es el predictor más relevante en el proceso de clasificación. Esta primera división separa inmediatamente a los candidatos con desempeño académico más bajo, quienes presentan una mayor probabilidad de pertenecer a la clase “no”, de aquellos con promedios superiores, que tienden mayoritariamente a clasificarse como “yes”. El hecho de que el CGPA sea el criterio inicial de partición evidencia que es la variable que mayor reducción de impureza (Gini) genera en el modelo.

En la rama izquierda, correspondiente a valores bajos de CGPA, el árbol realiza una segunda partición basada en el GRE Score ≤ 307.5. Esta combinación de bajo promedio y puntaje GRE moderado conduce predominantemente a clasificaciones negativas, consolidando la idea de que un desempeño académico bajo difícilmente se compensa con otros factores. Los nodos terminales en esta rama muestran niveles bajos de impureza, indicando que el modelo logra separar adecuadamente a los candidatos con baja probabilidad de admisión en esta región del espacio de variables.

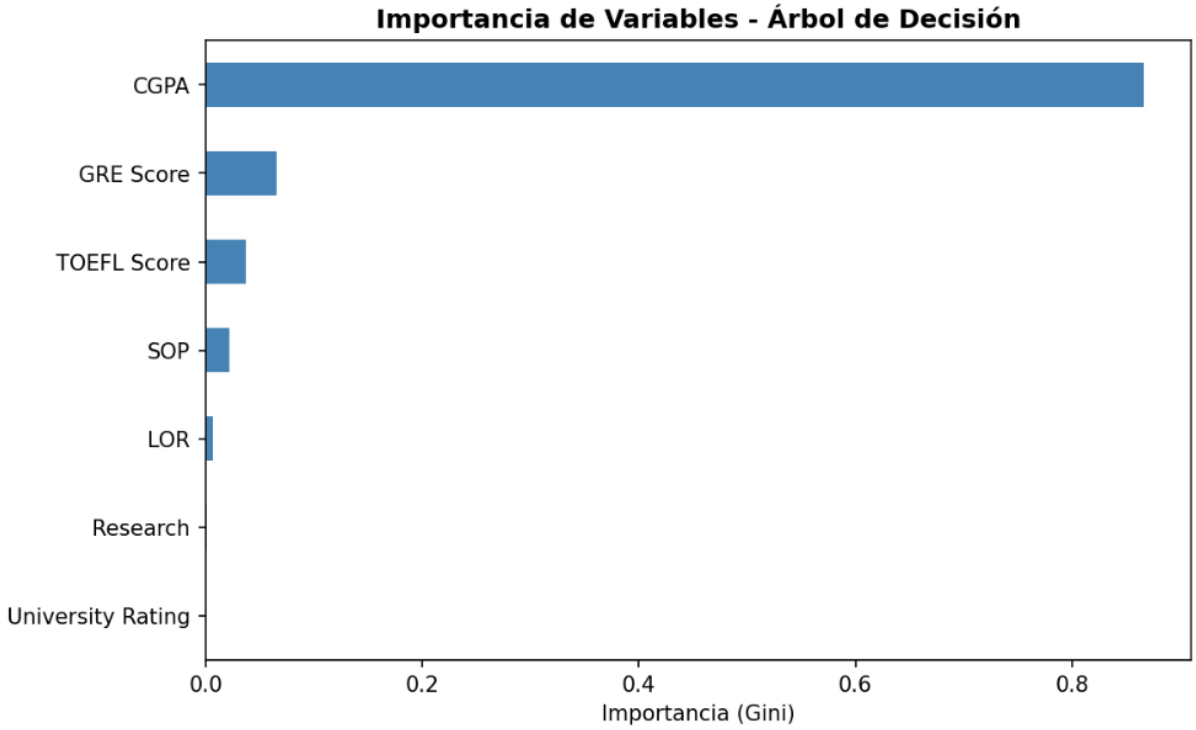
En contraste, la rama derecha del árbol, que agrupa a los candidatos con CGPA > 7.91, concentra la mayoría de las observaciones y presenta una clara predominancia de la clase “yes”. Aquí el modelo introduce particiones adicionales utilizando variables como TOEFL Score, GRE Score, SOP, LOR y Research, lo que sugiere que, una vez superado cierto umbral de promedio académico, otros factores complementarios comienzan a influir en la clasificación. Por ejemplo, para valores de CGPA superiores a 8.47, el modelo prácticamente clasifica a todos los candidatos como admitidos, mostrando nodos con impureza extremadamente baja (gini cercano a 0), lo que indica alta pureza y fuerte capacidad de separación en esa región.

Es particularmente relevante observar que, en niveles altos de CGPA y GRE, la variable Research aparece como criterio de partición, reforzando la idea de que la experiencia investigativa contribuye positivamente cuando el perfil académico ya es sólido. Asimismo, las divisiones basadas en LOR y SOP

en niveles intermedios muestran que estos factores cualitativos actúan como variables de ajuste fino dentro del modelo.

En términos generales, el árbol evidencia una estructura coherente con el análisis exploratorio y la matriz de correlación: el CGPA domina la decisión inicial, seguido por los puntajes estandarizados y, en etapas posteriores, variables complementarias. La limitación de profundidad a cuatro niveles evita una estructura excesivamente compleja y reduce el riesgo de sobreajuste, manteniendo al mismo tiempo una alta interpretabilidad. Este gráfico no solo confirma la importancia relativa de las variables, sino que también permite comprender de manera transparente las reglas exactas que el modelo utiliza para clasificar a los candidatos.

21.4. Gráfico Importancia de Variables Árbol de Decisión



El gráfico de importancia de variables del Árbol de Decisión muestra la contribución relativa de cada predictor en la reducción de impureza del modelo, medida a través del índice de Gini. Se observa de manera contundente que el CGPA domina ampliamente la estructura del modelo, concentrando la mayor proporción de importancia, muy por encima del resto de variables. Este resultado confirma tanto el análisis exploratorio como la matriz de correlación y la estructura del árbol previamente analizada, donde el CGPA aparece como nodo raíz y principal criterio de partición. En términos prácticos, esto significa que el promedio académico acumulado es el factor que más información aporta para separar correctamente a los candidatos admitidos de los no admitidos.

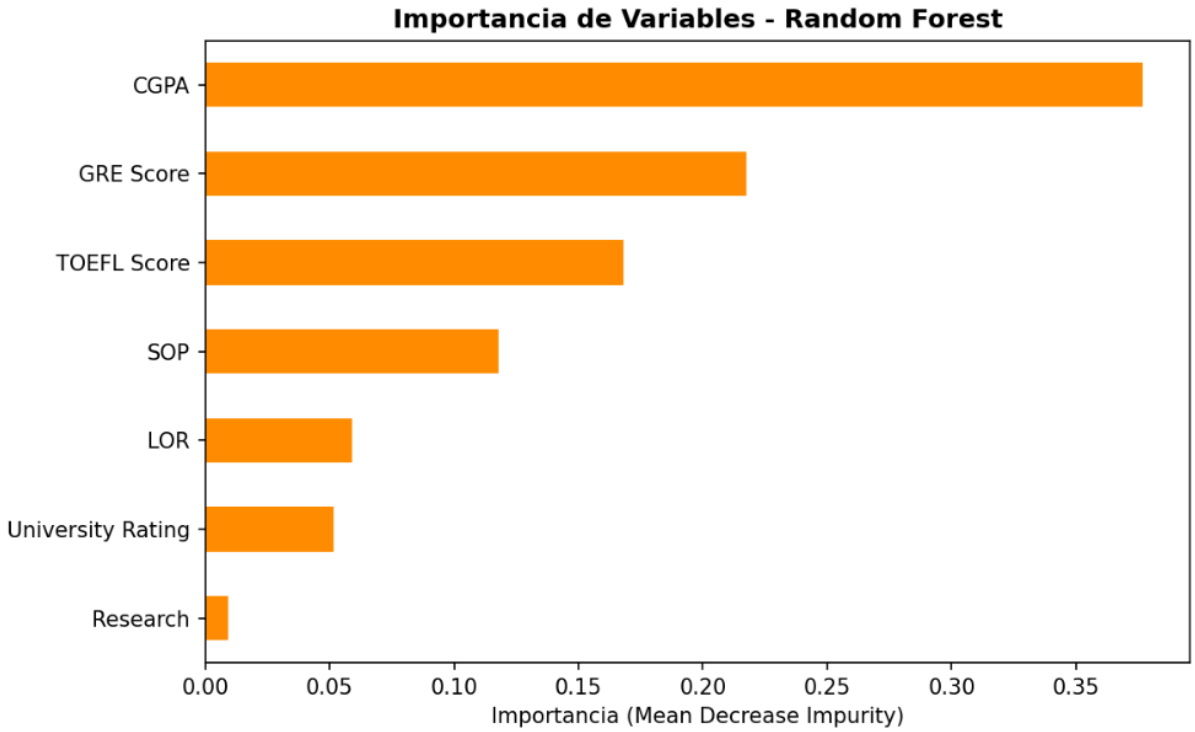
En un segundo nivel de relevancia se encuentra el GRE Score, cuya importancia es considerablemente menor que la del CGPA, pero aún significativa en comparación con las demás variables. Esto indica que el puntaje GRE actúa como un criterio complementario importante, especialmente en regiones del árbol donde el promedio académico se encuentra en rangos intermedios. Posteriormente, el TOEFL Score presenta una contribución menor, aunque todavía detectable, lo que sugiere que el desempeño en el idioma inglés influye en la clasificación, pero no de manera tan determinante como el rendimiento académico general.

Variables como SOP, LOR, Research y University Rating muestran una importancia marginal dentro del modelo individual. Esto no implica que carezcan de relevancia en el proceso real de admisión, sino que, dentro de la estructura específica de este árbol limitado a una profundidad de cuatro niveles, su capacidad para reducir la impureza fue comparativamente baja. Es importante destacar que en modelos basados en árboles, la importancia depende de las divisiones efectivamente seleccionadas; si una

variable no es utilizada en particiones relevantes o aparece en niveles muy profundos, su importancia global será reducida.

En conjunto, este gráfico confirma que el modelo de Árbol de Decisión basa su poder predictivo principalmente en el desempeño académico cuantitativo, particularmente el CGPA, utilizando el GRE y el TOEFL como variables secundarias de ajuste. Esta distribución de importancia es coherente con la lógica del problema y con los resultados obtenidos en las métricas de desempeño, evidenciando consistencia entre el análisis exploratorio, la estructura del árbol y la evaluación final del modelo.

21.5. Gráfico Importancia de Variables Random Forest



El gráfico de importancia de variables del modelo Random Forest presenta una distribución más equilibrada de las contribuciones predictivas en comparación con el Árbol de Decisión individual. Aunque el CGPA continúa siendo la variable más influyente, su peso relativo ya no es tan dominante como en el árbol único. Esto ocurre porque el Random Forest construye múltiples árboles sobre distintas muestras bootstrap y subconjuntos aleatorios de variables, lo que reduce la dependencia excesiva de un solo predictor y distribuye la importancia entre varios factores relevantes.

En este modelo, el CGPA sigue ocupando el primer lugar, confirmando su papel central en la predicción de la admisión. Sin embargo, el GRE Score adquiere una importancia considerablemente mayor en comparación con el árbol individual, lo que indica que, al combinar múltiples árboles, el puntaje GRE aporta información adicional de manera más consistente. El TOEFL Score también muestra un peso significativo, reforzando la idea de que los puntajes estandarizados influyen de forma importante cuando el modelo evalúa múltiples particiones posibles.

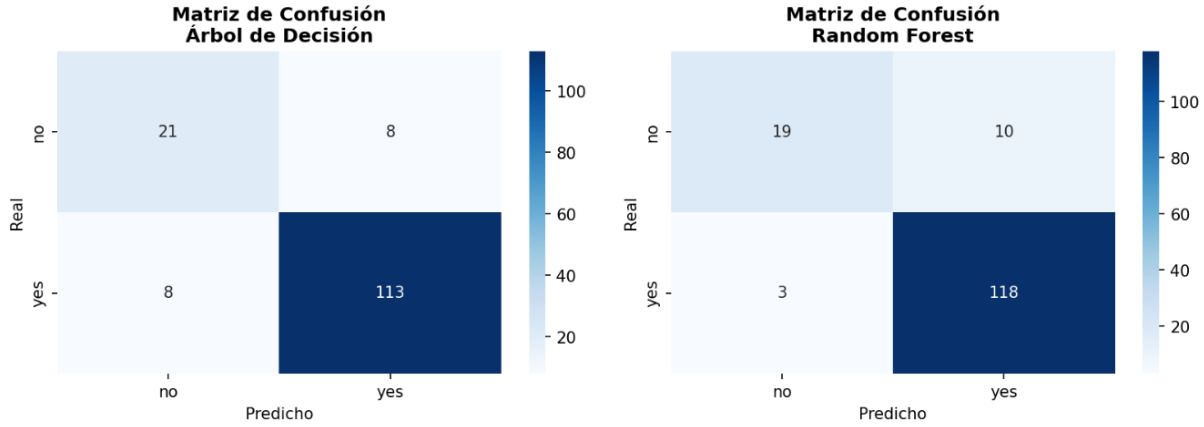
A diferencia del Árbol de Decisión, variables como SOP, LOR y University Rating presentan una contribución más visible en el Random Forest. Esto sugiere que, aunque individualmente no fueron seleccionadas como divisiones principales en el árbol limitado a profundidad cuatro, sí aportan información predictiva cuando el modelo explora múltiples configuraciones estructurales. Es decir, el ensemble logra capturar patrones que el árbol único no necesariamente prioriza.

Por otro lado, la variable Research muestra una importancia relativamente baja en comparación con las demás, lo que indica que, aunque tiene correlación positiva con la probabilidad de admisión, su capacidad incremental para reducir la impureza dentro del conjunto de árboles es menor. Esto puede

deberse a que su efecto ya está parcialmente capturado por variables académicas más fuertes como CGPA y GRE.

En términos generales, este gráfico confirma que el Random Forest ofrece una visión más robusta y menos sesgada de la importancia relativa de las variables. Mientras que el árbol individual se apoya fuertemente en un único predictor dominante, el modelo basado en ensambles distribuye la importancia entre varios factores académicos relevantes, lo que contribuye a mejorar la estabilidad y la capacidad de generalización observada en las métricas de desempeño.

21.6. Gráfico de Matrices de Confusión



Las matrices de confusión permiten examinar en detalle el desempeño de ambos modelos al descomponer los resultados en verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. A diferencia del accuracy, que ofrece una visión global, estas matrices permiten entender cómo se distribuyen los errores y cómo cada modelo se comporta frente a las clases mayoritaria y minoritaria.

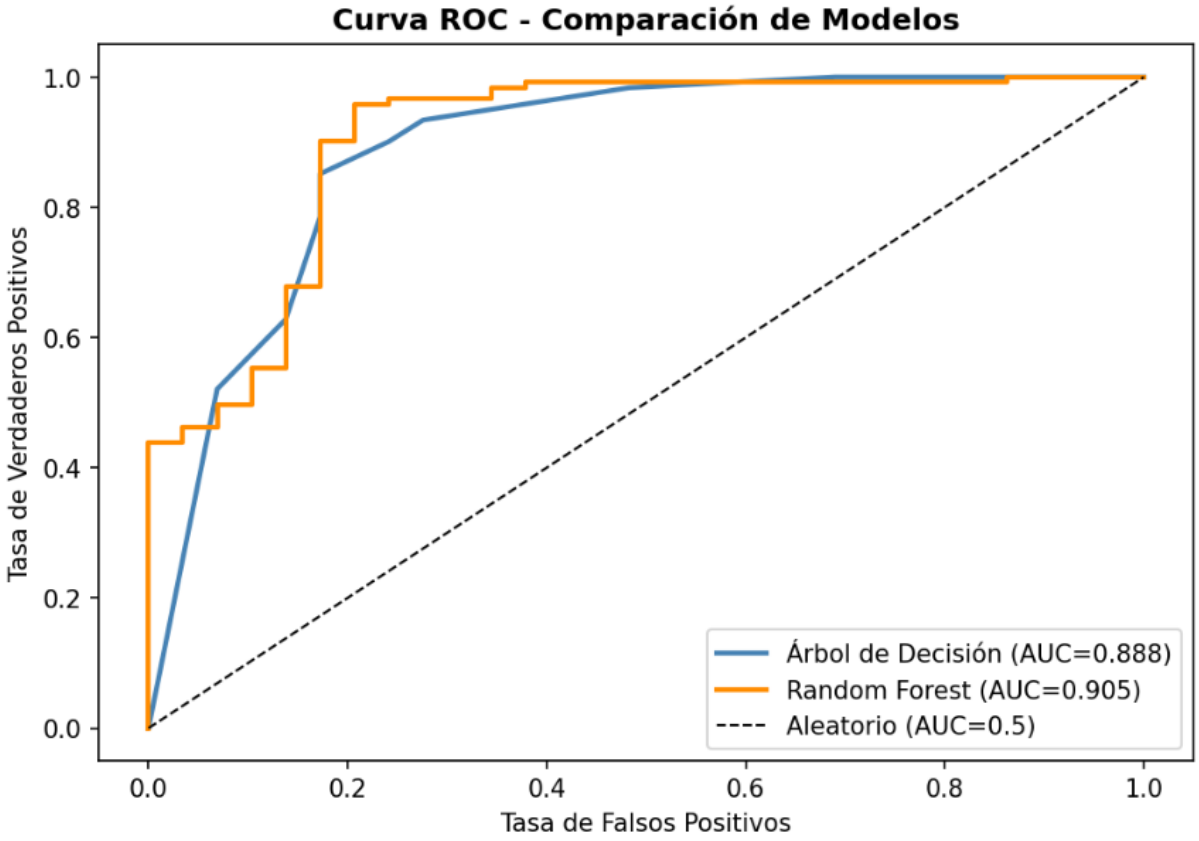
En el caso del Árbol de Decisión, se observa que el modelo clasificó correctamente 113 candidatos admitidos (verdaderos positivos) y 21 candidatos no admitidos (verdaderos negativos). Sin embargo, cometió 8 errores al clasificar candidatos no admitidos como admitidos (falsos positivos) y otros 8 errores al clasificar candidatos admitidos como no admitidos (falsos negativos). Este comportamiento muestra un equilibrio razonable en la distribución de errores, aunque se evidencia una ligera dificultad para identificar correctamente la clase minoritaria, lo cual es consistente con el desbalance del dataset. El árbol logra un desempeño sólido en la clase “yes”, pero presenta cierta limitación en la detección precisa de la clase “no”.

Por su parte, el Random Forest muestra una mejora en la detección de la clase mayoritaria, clasificando correctamente 118 candidatos admitidos y reduciendo los falsos negativos a solo 3 casos. Esto indica una mayor capacidad para identificar correctamente a los estudiantes con alta probabilidad de admisión. No obstante, el modelo presenta 10 falsos positivos y 19 verdaderos negativos, lo que implica que, aunque mejora la sensibilidad para la clase “yes”, sacrifica ligeramente la precisión en la identificación de la clase “no”. En otras palabras, el Random Forest prioriza la correcta detección de candidatos admitidos, lo cual incrementa el recall de la clase positiva, pero reduce moderadamente el recall de la clase negativa.

Comparando ambos modelos, se evidencia que el Random Forest mejora el número total de clasificaciones correctas y reduce significativamente los falsos negativos en la clase “yes”, lo que explica su mayor accuracy y AUC-ROC. Sin embargo, el Árbol de Decisión mantiene un equilibrio ligeramente más uniforme entre ambas clases. Esta diferencia refleja el comportamiento típico de los modelos de ensamble, que tienden a optimizar el rendimiento global incluso si ello implica un pequeño ajuste en la distribución de errores por clase.

En conjunto, las matrices de confusión confirman que ambos modelos presentan un desempeño alto, aunque el Random Forest muestra una ventaja clara en la detección de candidatos con alta probabilidad de admisión, consolidando su superioridad observada en las métricas globales de evaluación.

21.7. Gráfico de Curva ROC



La curva ROC (Receiver Operating Characteristic) permite evaluar la capacidad discriminativa de los modelos considerando todos los posibles umbrales de decisión, y no únicamente el punto de corte fijo utilizado para la clasificación final. En el gráfico se comparan las curvas correspondientes al Árbol de Decisión y al Random Forest, junto con la línea diagonal punteada que representa el desempeño de un clasificador aleatorio (AUC = 0.5). Ambas curvas se encuentran claramente por encima de esta línea de referencia, lo que indica que los dos modelos poseen una capacidad predictiva significativamente superior al azar.

El Árbol de Decisión presenta un área bajo la curva (AUC) de 0.888, lo que indica una muy buena capacidad para distinguir entre candidatos admitidos y no admitidos. Un AUC cercano a 0.9 implica que el modelo tiene una alta probabilidad de asignar una mayor puntuación a un candidato realmente admitido que a uno no admitido. La forma de la curva muestra que el modelo alcanza valores elevados de tasa de verdaderos positivos con incrementos moderados en la tasa de falsos positivos, reflejando un equilibrio adecuado entre sensibilidad y especificidad.

Por su parte, el Random Forest alcanza un AUC de 0.905, superando al árbol individual. Esta diferencia, aunque no extremadamente grande, es consistente con las mejoras observadas en accuracy y validación cruzada. La curva del Random Forest se mantiene ligeramente por encima de la del Árbol de Decisión en la mayor parte del rango de falsos positivos, lo que evidencia una mejor capacidad de separación global entre las dos clases. En términos prácticos, esto significa que el modelo basado en ensambles logra discriminar con mayor precisión a los candidatos con alta probabilidad de admisión frente a aquellos con baja probabilidad.

La comparación gráfica confirma que ambos modelos son robustos y presentan alto desempeño predictivo, pero el Random Forest ofrece una mejora sistemática en la capacidad discriminativa. Esta superioridad es coherente con la teoría de los métodos de ensamble, que reducen la varianza del modelo individual y mejoran la estabilidad al combinar múltiples árboles de decisión. En conjunto, la curva ROC refuerza la conclusión de que el Random Forest constituye el modelo con mejor desempeño

global dentro del laboratorio, sin comprometer significativamente la interpretabilidad obtenida en etapas previas.

CONCLUSIONES DE LA ACTIVIDAD

Para concluir este laboratorio, puedo afirmar que el desarrollo del modelo de clasificación aplicado al Graduate Admissions Dataset permitió no solo construir modelos predictivos con alto desempeño, sino también comprender en profundidad los factores que influyen en la probabilidad de admisión a programas de posgrado. A lo largo del laboratorio se siguió un proceso estructurado que integró análisis exploratorio, preparación de datos, modelización y evaluación comparativa, lo cual permitió abordar el problema desde una perspectiva metodológicamente sólida y coherente.

Desde el punto de vista técnico, el dataset presentó una estructura adecuada para el modelado, ya que no se detectaron valores faltantes ni inconsistencias relevantes en los datos. Esto permitió centrar el análisis en la interpretación de patrones y relaciones entre variables. La transformación de la variable continua Chance of Admit en una variable binaria mediante el umbral de 0.6 permitió convertir el problema en uno de clasificación supervisada, facilitando la aplicación de algoritmos basados en árboles. No obstante, se identificó un desbalance moderado en la distribución de clases (81% “yes” y 19% “no”), lo cual influyó en el comportamiento de algunas métricas, especialmente en la detección de la clase minoritaria.

El análisis exploratorio y la matriz de correlación permitieron identificar claramente que el CGPA constituye el factor más determinante en la probabilidad de admisión, seguido por los puntajes GRE y TOEFL. Estas relaciones fueron posteriormente confirmadas por los modelos entrenados, tanto en la estructura del Árbol de Decisión como en la importancia de variables estimada por el Random Forest. La coherencia entre el análisis estadístico previo y los resultados del modelado evidencia una correcta interpretación del fenómeno y una adecuada selección de variables predictoras.

En cuanto a la modelización, el Árbol de Decisión mostró un desempeño sólido, alcanzando un accuracy cercano al 89% y un AUC superior a 0.88, además de ofrecer alta interpretabilidad a través de reglas explícitas de clasificación. Sin embargo, el modelo Random Forest logró mejorar consistentemente las métricas de desempeño, alcanzando un accuracy superior al 91% y un AUC mayor a 0.90, además de presentar mayor estabilidad en la validación cruzada. Esta mejora confirma la ventaja teórica de los métodos de ensamble, los cuales reducen la varianza del modelo individual y mejoran la capacidad de generalización.

Las matrices de confusión evidenciaron que ambos modelos presentan un desempeño notable en la detección de candidatos con alta probabilidad de admisión, aunque el Random Forest mostró una ligera mejora en la identificación de verdaderos positivos, mientras que el Árbol de Decisión mantuvo un equilibrio algo más uniforme entre clases. La curva ROC reforzó esta conclusión al mostrar una mayor área bajo la curva para el modelo de ensamble, confirmando su superioridad en términos de capacidad discriminativa global.

Finalmente, la aplicación de validación cruzada de cinco particiones permitió evaluar la estabilidad de los modelos más allá de una única división train/test, fortaleciendo la validez de los resultados obtenidos. La consistencia entre el desempeño en el conjunto de prueba y la media de la validación cruzada indica que los modelos no presentan sobreajuste significativo y que generalizan adecuadamente sobre el conjunto completo de datos.

En síntesis, esta actividad permitió aplicar de manera práctica algoritmos de clasificación basados en árboles, interpretar métricas de evaluación, analizar la importancia de variables y comprender la diferencia entre un modelo individual y un modelo de ensamble. Más allá de las métricas obtenidas, el trabajo demuestra la importancia de seguir un proceso estructurado que combine análisis exploratorio, preparación adecuada de datos y evaluación rigurosa. Este enfoque metodológico constituye la base fundamental para el desarrollo de modelos predictivos robustos, interpretables y alineados con la naturaleza real del problema estudiado.

BIBLIOGRAFÍA

A continuación, la bibliografía implementada en este desarrollo:

- Tema 2. Conceptos básicos de aprendizaje automático. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Tema 3. Exploración y preprocesamiento de datos. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Tema 2. Conceptos básicos de aprendizaje automático. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Tema 3. Exploración y preprocesamiento de datos. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Tema 4. Árboles de decisión. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Tema 5. Evaluación de clasificadores. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Tema 6. Reglas de clasificación. Aprendizaje Automático y Minería de Datos (COLGII) - PER 15746 - Enero 2026.
- Clases virtuales con el profesor Ing. Rogerio Orlando Beltrán Castro.
- Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324.
- Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and regression trees. CRC Press.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An introduction to statistical learning: With applications in Python (2nd ed.). Springer.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning (2nd ed.). Springer.
- Kuhn, M., & Johnson, K. (2013). Applied predictive modeling. Springer.
- Scikit-learn Developers. (2024). Scikit-learn: Machine learning in Python. https://scikit-learn.org
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

AGRADECIMIENTO

Finalmente, deseo expresar mi más sincero agradecimiento al profesor Ing. Rogerio Orlando Beltrán Castro por la orientación, los conocimientos y el acompañamiento brindado durante el desarrollo de esta actividad. Sus explicaciones en el área de aprendizaje automático fueron fundamentales para comprender no solo el funcionamiento teórico de los algoritmos de clasificación, como los Árboles de Decisión y los métodos de ensamble como Random Forest, sino también la importancia de una adecuada evaluación de modelos mediante métricas como accuracy, AUC-ROC, matrices de confusión y validación cruzada.

Gracias a los conceptos abordados en clase, fue posible estructurar este laboratorio de manera ordenada, iniciando con un análisis exploratorio sólido, seguido de una preparación rigurosa de los datos y culminando con una comparación crítica entre modelos. La comprensión de la importancia del preprocesamiento, la interpretación de resultados y la correcta validación de los modelos permitió desarrollar un trabajo metodológicamente consistente y alineado con buenas prácticas en ciencia de datos.

Sin duda, esta experiencia fortaleció significativamente mi formación académica, permitiéndome aplicar de manera práctica los conceptos de clasificación supervisada y evaluación de modelos, y consolidando mi entendimiento del proceso completo de modelización en aprendizaje automático.

¡¡¡Mil gracias profesor!!!

Respetuosamente,

Alejandro De Mendoza