Machine Learning 1

Admafosi

27 de junio de 2025

Índice

Resumen

Aprender Machine Learning implica comprender cómo las máquinas pueden aprender a partir de datos para hacer predicciones o tomar decisiones sin instrucciones explícitas. Estrechamente relacionado con la analítica de datos, ya que ambos se basan en el uso de datos para obtener información valiosa: mientras la analítica se enfoca en describir y entender lo que ocurrió, el machine learning busca anticipar lo que ocurrirá o automatizar decisiones

Esta combinación tiene aplicaciones directas en la vida cotidiana, como en las recomendaciones de contenido, los asistentes virtuales, los filtros de spam, el reconocimiento facial o los sistemas de transporte inteligente, mejorando la forma en que interactuamos con la tecnología.

En este proceso, una persona desarrolla habilidades en programación, análisis de datos y evaluación de modelos estadísticos.

Algunas de las técnicas aplicadas al machine learning se encuentran los modelos de regresión: La regresión lineal simple es uno de los algoritmos más básicos y se utiliza para predecir un valor numérico a partir de una sola variable independiente, estableciendo una relación lineal entre ambas. Por ejemplo, puede predecirse el precio de una casa según su tamaño.

Por otro lado, cuando se utilizan múltiples variables independientes para hacer una predicción, se habla de regresión lineal multivariable. el cual permite considerar varios factores simultáneamente, como el tamaño, la ubicación y el número de habitaciones para predecir el precio de una propiedad, proporcionando así predicciones más precisas.

Otro de loas modelos corresponde a la regresión logística es empleada cuando la variable objetivo es categórica, es decir, cuando se desea clasificar los datos en categorías como "sí.º "no", "spam.º "no spam", çliente que compra.º çliente que no compra". Aunque su nombre sugiere una relación con la regresión lineal, en realidad está más orientada a resolver problemas de clasificación.

Estas técnicas son fundamentales en el aprendizaje supervisado y sientan las bases para modelos más complejos que se utilizan en diversas industrias y áreas del conocimiento.

1. Introducción

Durante el desarrollo de esta asignatura se aplicarán técnicas estadísticas que permitan al estudiante analizar, predecir y extraer datos. Además, de desarrollar habilidades en el uso de software como lo es Jupyter Notebook y Python y un programa como LaTeX que corresponde a una plataforma web útil para la creación de documentos científicos, técnicos y académicos.

Como parte fundamental del curso, se abordarán modelos de aprendizaje supervisado, entre ellos la regresión lineal simple, utilizada para predecir un valor numérico a partir de una sola variable; la regresión lineal multivariable, que extiende esta capacidad a múltiples variables independientes; y la regresión logística, empleada en problemas de clasificación donde la variable de salida es categórica. Estas técnicas permitirán al estudiante entender la relación entre variables y aplicar modelos predictivos en contextos reales.

2. Marco Teórico

Machine Learning es una rama de la inteligencia artificial que permite a las computadoras aprender a partir de datos y tomar decisiones sin programación explícita para cada tarea (Mitchell, 1997). Se basa en la idea de que los sistemas pueden mejorar su rendimiento al exponerse a más información y ejemplos. Aplicaciones comunes incluyen reconocimiento de imágenes, procesamiento de lenguaje natural y sistemas de recomendación.

La **analítica de datos** se refiere al proceso de inspeccionar, limpiar y modelar datos para descubrir información útil y apoyar la toma de decisiones (Davenport & Harris, 2007). Es fundamental para entender tendencias, patrones y comportamientos en diversas áreas, desde negocios hasta salud.

Python es un lenguaje de programación interpretado, fácil de aprender y muy usado en ciencia de datos y Machine Learning por su sintaxis clara y su extensa biblioteca de herramientas especializadas (Lutz, 2013).

Jupyter Notebook es una aplicación web que permite crear y compartir documentos que contienen código, ecuaciones, visualizaciones y texto narrativo. Es muy popular en análisis de datos y educación por facilitar la experimentación y documentación (Kluyver et al., 2016).

2.1. Tipos de Aprendizaje en Machine Learning

Como Machine Learning (ML) se basa en la enseñanza a las computadoras a .ªprender.ª partir de los datos. Existen distintos tipos de aprendizaje, y cada uno tiene una forma diferente de hacerlo:

1. Aprendizaje Supervisado

¿Qué es? Es cuando el algoritmo aprende usando datos etiquetados, es decir, datos de entrada que ya tienen una respuesta conocida (la .etiqueta").

Ejemplo: Le das a un programa muchos ejemplos de casas con sus características (tamaño, ubicación, número de cuartos) y el precio de cada una. Así, el modelo aprende a predecir el precio de una casa nueva.

Algoritmos comunes: -Regresión lineal: para predecir valores numéricos. -SVM (Máquinas de Vectores de Soporte): para clasificar datos. -Random Forest: un conjunto de árboles de decisión que mejora la precisión.

2. Aprendizaje No Supervisado

¿Qué es? Aquí el algoritmo no tiene etiquetas. Solo se le dan los datos, y él debe descubrir patrones o estructuras por sí mismo.

Ejemplo: Una empresa quiere agrupar a sus clientes por comportamiento de compra, pero no tiene categorías predefinidas. El algoritmo encuentra automáticamente esos grupos o "clusters".

Algoritmos comunes: - K-Means: agrupa datos similares. - PCA (Análisis de Componentes Principales): reduce la cantidad de variables para simplificar datos complejos.

3. Aprendizaje por Refuerzo

¿Qué es?: Este tipo de aprendizaje se basa en prueba y error. El modelo aprende a tomar decisiones en un entorno para maximizar una recompensa.

Ejemplo: Un robot aprende a caminar. Cada vez que avanza sin caerse, recibe una -ecompensa". Si se cae, recibe una "penalización". Con el tiempo, aprende qué movimientos son mejores.

Algoritmos comunes: - Q-Learning - DQN (Deep Q-Network): combina redes neuronales con aprendizaje por refuerzo.

4. Aprendizaje Semi-supervisado

¿Qué es?: Una mezcla entre supervisado y no supervisado. Usa una pequeña cantidad de datos etiquetados junto con una gran cantidad sin etiquetar.

Ejemplo: Tienes 1000 imágenes, pero solo 50 están clasificadas como "perro" o "gato". El modelo aprende usando esas pocas etiquetas y generaliza con el resto.

Algoritmos comunes: No hay uno específico, se usan modelos híbridos que combinan técnicas de los otros tipos.

5. AutoML (Aprendizaje Automatizado)

¿Qué es?: Significa automatizar el proceso de crear modelos de Machine Learning. El sistema selecciona por sí mismo los mejores algoritmos y configuraciones.

Ejemplo: Una empresa sin expertos en ML quiere un modelo que prediga ventas. AutoML prueba varios modelos y elige el mejor, sin que la persona tenga que programar nada.

Herramientas comunes:

- AutoSklearn.

- Google AutoML.

2.2. Clustering

Es una técnica del aprendizaje no supervisado que sirve para organizar datos sin etiquetas previas. Es decir, el sistema no sabe de antemano qué categorías existen, pero trata de agrupar elementos que se parezcan entre sí.

Imagina que tienes una bolsa con botones de diferentes formas, colores y tamaños, pero sin etiquetas. El clustering sería como agruparlos en montones donde los botones similares estén juntos: todos los rojos por un lado, los grandes por otro, etc.

¿Cómo funciona? Un algoritmo de clustering analiza los datos y mide qué tan parecidos son entre sí. Los que tienen características similares (como valores numéricos parecidos, palabras comunes, colores similares, etc.) se agrupan juntos. A estos grupos se les llama clusters o agrupaciones.

Ejemplos claros de clustering:

1. Google News Google utiliza clustering para organizar automáticamente las noticias. Por ejemplo, si en un mismo día hay muchas noticias con las palabras "elecciones", "presidente", "resultados", el algoritmo agrupa esos artículos en un mismo cluster, aunque vengan de diferentes medios. Así puedes ver varios enfoques de la misma noticia, sin que nadie lo haya clasificado manualmente.

2. Datos Genéticos En biología, se usan algoritmos de clustering para agrupar personas según su información genética. Por ejemplo, al analizar cómo se expresan ciertos genes, se pueden formar grupos de individuos con características similares, lo que ayuda a descubrir patrones o posibles enfermedades comunes, sin que los científicos indiquen previamente qué buscar.

3. Segmentación de Clientes (Marketing) Las empresas usan clustering para entender a sus clientes y personalizar estrategias.

Imagina que una tienda analiza los hábitos de compra de miles de personas. El clustering puede revelar grupos como:

- Clientes que compran solo en ofertas.
- Clientes fieles que compran mensualmente.
- Clientes que solo compran productos de lujo.

Con esos grupos, la empresa puede enviar ofertas personalizadas o diseñar productos según el comportamiento de cada tipo de cliente.

3. Regresión Lineal

Dataset utilizado Boston Housing, contiene un conjunto de datos con información sobre viviendas en diferentes barrios de Boston, Estados Unidos.

3.1. Descripción del dataset

- Fuente del dataset:

  Boston-Housing dataset extraído de: Dataset_HousingData

- Cantidad de datos:

  El dataset se encuentra compuesto con 13 variables (como tasa de delincuencia per cápita por ciudad, cantidad de cuartos, tasa de impuestos, porcentaje de población de bajos ingresos y precio mediano de la vivienda)

- Atributos principales

  Los atributos que contiene este dataset están enfocados en información sobre diferentes aspectos socioeconómicos, estructurales y ambientales de barrios en Boston.

Nº	Atributo (Código)	Descripción en inglés	Tipo
1	**CRIM**	Per capita crime rate by town	Continua
2	**ZN**	Proportion of residential land zoned for large lots	Continua
3	**INDUS**	Proportion of non-retail business acres per town	Continua
4	**CHAS**	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)	Binaria (0 o 1)
5	**NOX**	Nitric oxide concentration (parts per 10 million)	Continua
6	**RM**	Average number of rooms per dwelling	Continua
7	**AGE**	Proportion of owner-occupied units built before 1940	Continua
8	**DIS**	Weighted distances to five Boston employment centers	Continua
9	**RAD**	Index of accessibility to radial highways	Discreta
10	**TAX**	Full-value property tax rate per $10,000	Continua
11	**PTRATIO**	Pupil–teacher ratio by town	Continua
12	**B**	1000(Bk - 0.63)^2, where Bk is the proportion of Black people	Continua
13	**LSTAT**	% lower status of the population	Continua
14	**MEDV**	Median value of owner-occupied homes in $1000s	Continua

Figura 1: Cuadro de Atributos

- Preprocesamiento realizado.

  - Carga e indagación previa sobre el dataset.

```
import pandas as pd
url = 'https://raw.githubusercontent.com/Vinaya19/Boston-Housing-Dataset/main/HousingData.csv'
df = pd.read_csv(url)
df.head()
df.describe().T
```

Figura 2: Carga y Lectura del Dataset

- Normalización de las variables.

En esta fase se procedió a realizar la traducción al español, renombrar, se determinó el tipo y se agregó información descriptiva por cada variable.

Nombre original	Nombre traducido	Tipo de dato	Descripción
CRIM	tasa_crimen	Numérico	Tasa de criminalidad per cápita por zona.
ZN	proporcion_residencial	Numérico	Proporción de terrenos residenciales de más de 25,000
INDUS	proporcion_industrial	Numérico	Proporción de superficie no comercial por zona.
CHAS	rio_charles	Binario (0/1)	Variable ficticia: 1 si limita con el río Charles, 0 en caso contrario.
NOX	ox_nitrogeno	Numérico	Concentración de óxidos de nitrógeno (contaminación del aire).
RM	habitaciones_prom	Numérico	Número promedio de habitaciones por vivienda.
AGE	porcentaje_antiguas	Numérico	Porcentaje de unidades ocupadas construidas antes de 1940.
DIS	distancia_centros	Numérico	Distancia media a cinco centros de empleo de Boston.
RAD	acceso_autopistas	Entero	Índice de accesibilidad a autopistas radiales.
TAX	impuesto_propiedad	Numérico	Tasa de impuesto a la propiedad por cada $10,000.
PTRATIO	relacion_alumnos_profesor	Numérico	Relación alumno-profesor por distrito escolar.
B	poblacion_negra	Numérico	$1000(Bk - 0.63)^2$, donde Bk es la proporción de afroamericanos por zona.
LSTAT	porcentaje_bajo_ingreso	Numérico	Porcentaje de la población con bajos ingresos.
MEDV	valor_mediana_vivienda	Numérico	Valor mediano de las viviendas ocupadas por sus propietarios (en miles de USD). Variable objetivo.

Figura 3: Cuadro Descripción de las Variables

3.2. Metodología

La estrategia de trabajo para la aplicación de la regresión lineal simple y multivariable se dividió en las siguientes etapas:

- Exploración que a partir de la importación de los datos desde un repositorio público, la lectura del DataFrame a partir de la librería pandas se visualiza un resumen estadístico básico de las variables, su escala, tipo de datos y posibles valores atípicos.

- Preparación de los datos: iniciando con traducción de los nombres de las columnas del inglés al español con el fin de facilitar la interpretación y análisis del dataset. Se selecciona las variables que servirán como predictivas, normaliza sus valores para facilitar el entrenamiento del modelo y prepara los datos.

- Modelación se aplicaron los criterios teniendo en cuenta que:

  Regresión Lineal Simple: La modelación se aplicó con el fin de poder determinar cómo cambia el valor de la vivienda (valor mediana vivienda) solo en función del número promedio de habitaciones (habitaciones prom), sin considerar otras variables. También se pudo observar que se realiza una regresión simple con solo la variable CRIM (Taza de crímenes) para comparar y así poder determinar la relación y la línea de regresión entre CRIM (Tasa de crímenes) y su efecto en PRICE (Precio) de la vivienda.

  Regresión Lineal Multivariable: implementa desde cero un modelo de regresión lineal multivariable utilizando descenso por gradiente, esto con la finalidad de tener mayor comprensión en cómo se ajustan los parámetros del modelo a través de secuencias de programación interactiva, además, confirmar que esta permite minimizar los errores.

- Evaluación de los datos:

  Los resultados obtenidos durante este proceso se representan a continuación:

Modelo	MAE	MSE	RMSE	R^2
Regresión Lineal Simple	~4.5–5.0	~40–50	~6.3–7.0	~0.5
Regresión Lineal Múltiple	~3.0	~20	~4.5	~0.73

Figura 4: Resultados obtenidos x Tipo de Regresión

De lo anterior se puede deducir que con el modelo multivariable es más concreto que el modelo de regresión simple.

En el caso de la regresión lineal multivariable se observa que el coeficiente de determinación R^2 de 0.73 sugiere que el modelo explica aproximadamente el 73 % de la variabilidad del precio de la vivienda.

Mientras que para la regresión lineal simple el coeficiente de determinación R^2 0.5, indica una explicación más limitada con una sola variable, puesto que explica solo el

50 % de la variabilidad en los precios de las viviendas. La otra mitad queda sin análisis por el modelo ya que está asociada a otras variables.

3.3. Implementación

A continuación se relaciona el código de programación para la Regresión Lineal Simple y la Regresión Lineal Multivariable.

Listing 1: Regresión Lineal Simple

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Variables
X_simple = df[['habitaciones_prom']] # Variable de prediccion
y = df['valor_mediana_vivienda'] # Resultado

# Ajuste del modelo
modelo_simple = LinearRegression()
modelo_simple.fit(X_simple, y)

#\ Prediccion
y_pred = modelo_simple.predict(X_simple)

# Graficacion del resultado
plt.scatter(X_simple, y, alpha=0.6, label="Datos reales")
plt.plot(X_simple, y_pred, color='red', label="Regresion lineal")
plt.xlabel("Habitaciones promedio")
plt.ylabel("Valor mediana vivienda ($1000s)")
plt.title("Regresion lineal simple")
plt.legend()
plt.grid(True)
plt.show()
```

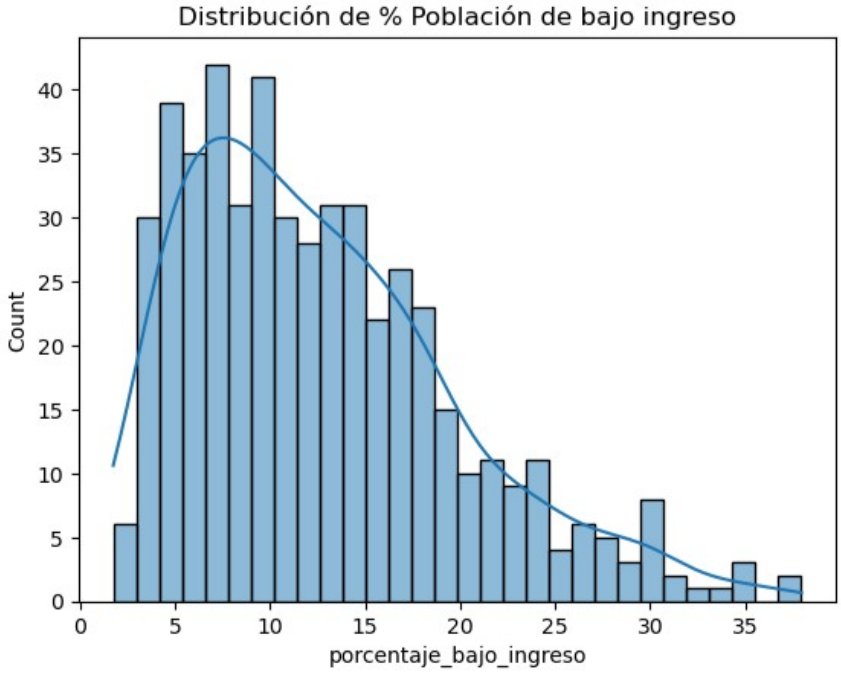
Listing 2: Regresión Lineal Multivariable

```
# Implementa de regresion lineal multivariable usa descenso por gradiente.
import numpy as np
# Agregar columna de unos (intercepto)
m = len(y)
X_b = np.c_[np.ones((m,1)), X_norm]
y_b = y.values.reshape(-1,1)

# Inicializa coeficientes (theta) y parametros
theta = np.zeros((X_b.shape[1],1))
alpha, iterations = 0.1, 1000

# Funciones de costo y entrenamiento
def compute_cost(X, y, theta):
m = len(y)
h = X.dot(theta)
return float((1/(2*m)) * np.sum((h - y)**2))
def gradient_descent(X, y, theta, alpha, iters):
J_hist = []
for _ in range(iters):
theta -= (alpha/m) * X.T.dot(X.dot(theta)-y)
J_hist.append(compute_cost(X, y, theta))
```

```
    return theta, J_hist
theta_final, J_hist = gradient_descent(X_b, y_b, theta, alpha, iterations)

# Graficacion del costo durante el entrenamiento
plt.plot(J_hist)
plt.xlabel('Iteraciones')
plt.ylabel('Costo J')
plt.title('Convergencia del descenso por gradiente')
plt.grid(True)
plt.show()
```

3.4. Resultados

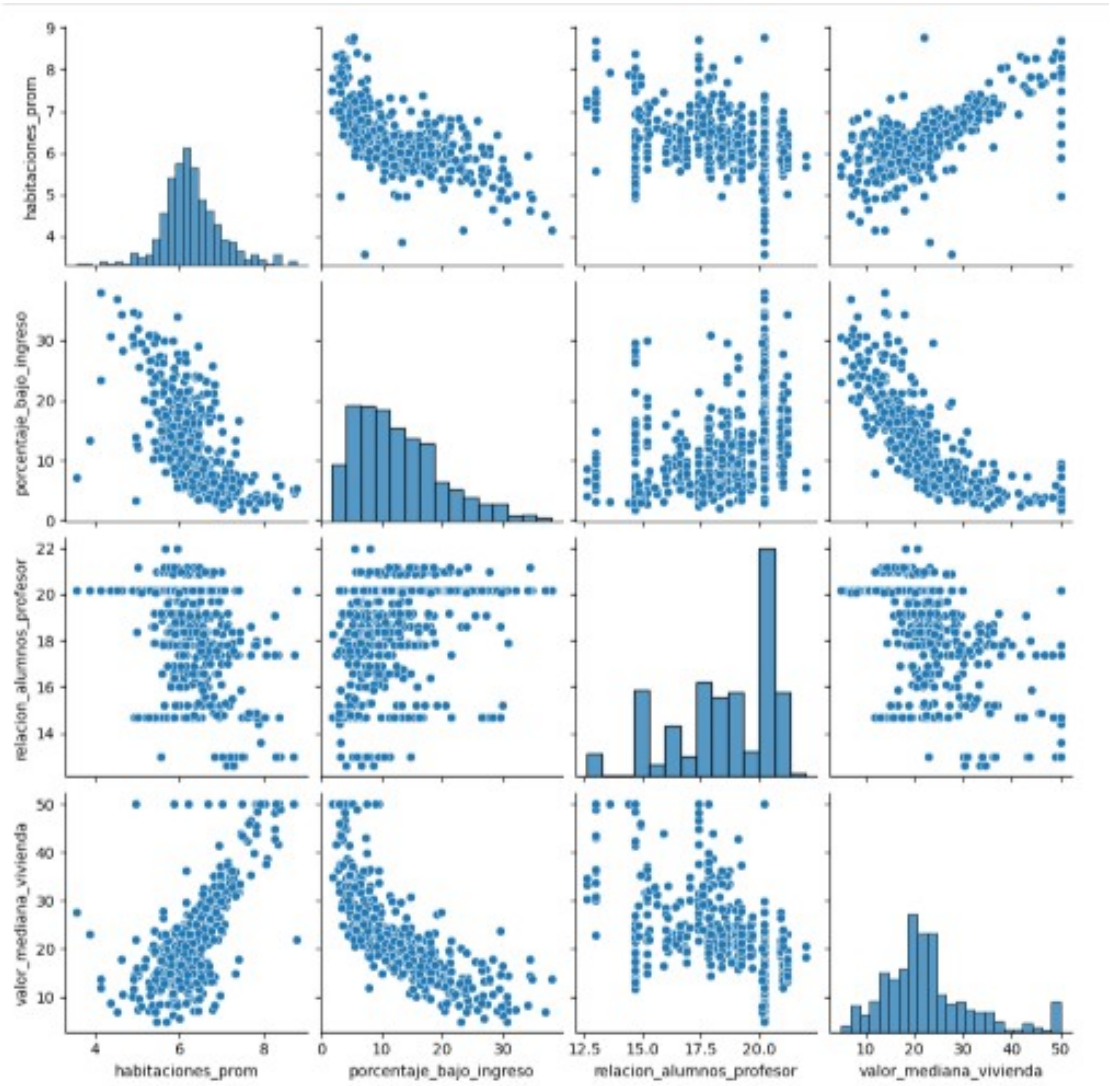
- Distribución de % Población de bajo ingreso



Esta gráfica muestra la distribución del porcentaje de población con bajos ingresos en diferentes áreas. Se observa una sesgo a la derecha, que indica que la mayoría de las zonas tienen un bajo porcentaje de población en situación económica precaria, aunque hay algunas con valores altos.

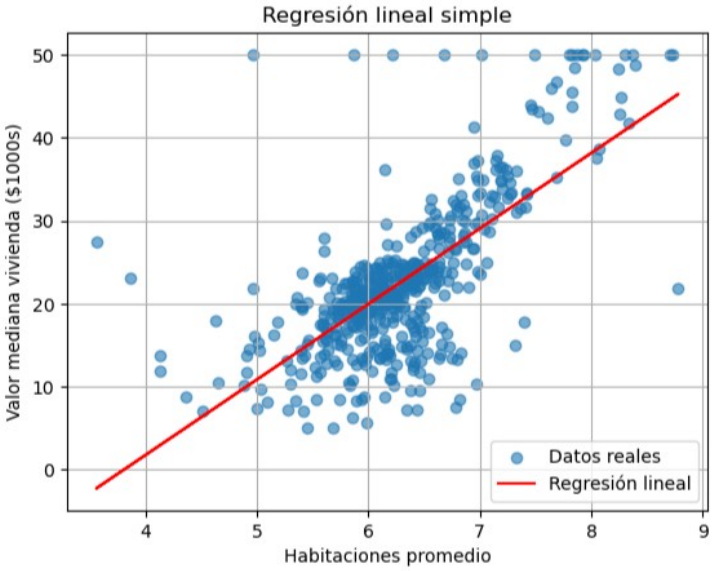
- Relación entre variables seleccionadas y valor de vivienda (Pairplot)

Esta matriz de gráficos muestra las distribuciones individuales y las relaciones bivariadas



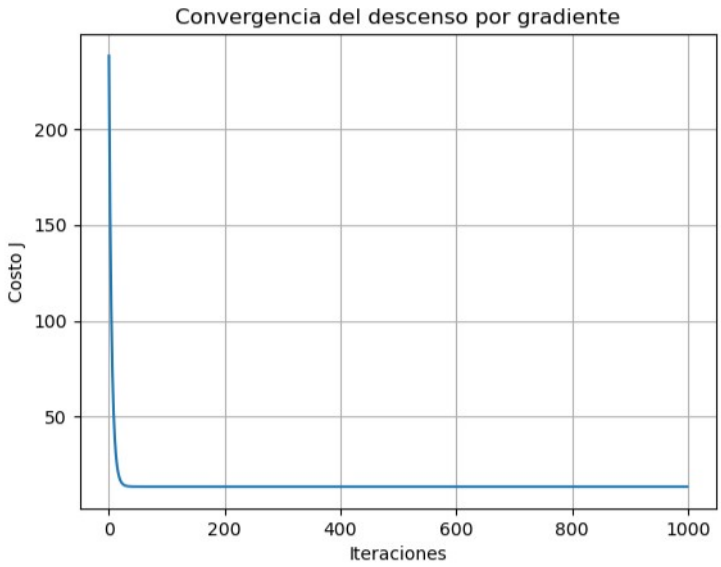
entre las variables. Por ejemplo, se puede observar que a mayor número de habitaciones promedio, generalmente mayor es el valor de la vivienda, mientras que mayores porcentajes de población de bajo ingreso o más alumnos por profesor tienden a asociarse con valores de vivienda más bajos.

- Regresión Lineal Simple



En el anterior gráfico se observa que los datos reales como puntos y la línea roja como la predicción del modelo de regresión lineal simple. La línea roja sigue bien la tendencia de los puntos, indicando que el número de habitaciones promedio es un buen predictor lineal del valor mediano de la vivienda.

- Convergencia del costo en descenso por gradiente



A través de la anterior gráfica se puede observa cómo el valor de la función de costo disminuye con cada interacción del algoritmo de descenso por gradiente. La tendencia descendente y estabilización al final indican que el modelo está aprendiendo y convergiendo hacia un mínimo.

- Q-Q Plot de residuos (Evaluación normalidad)

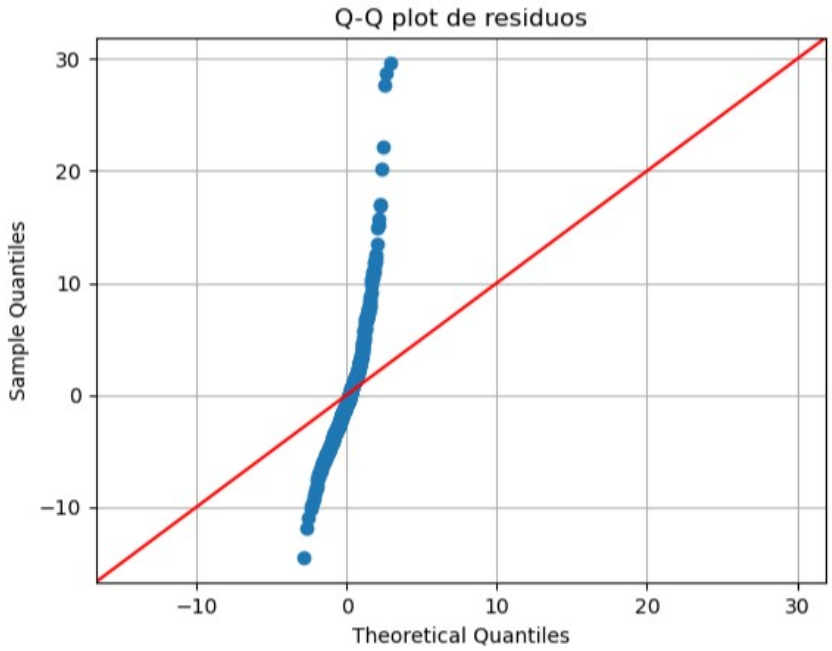


Figura 5: Enter Caption

El Q-Q plot compara la distribución de los residuos con una distribución normal. Si los puntos están cerca a la línea diagonal, los residuos son aproximadamente normales, lo que valida uno de los supuestos clave de la regresión lineal. Desviaciones significativas indicarían problemas en el modelo.

Eje X: Çuantiles teóricos"→ Representa los valores que esperaríamos si los residuos siguieran una distribución normal ideal.

Eje Y: Çuantiles muestrales"→ Representa los valores reales de los residuos del modelo ordenados, es decir, los cuantiles observados.

3.5. Conclusiones

1. Relaciones significativas identificadas La variable habitaciones promedio por vivienda tiene una relación positiva y significativa con el valor mediano de la vivienda. A mayor número de habitaciones, mayor es el valor estimado de la propiedad.

Variables como el porcentaje de población de bajos ingresos y la relación alumnos por profesor tienen un impacto negativo en el valor de las viviendas.

2. Modelo de Regresión Lineal Simple Al usar solo la variable habitaciones_prom, el modelo genera una recta de regresión que explica bastante bien la variación en los precios.

El coeficiente positivo (coef = 9) indica que por cada habitación adicional promedio, el valor mediano de la vivienda aumenta en alrededor de $9,000.

3. Regresión Lineal Múltiple (tres variables) Al incluir tres variables el modelo multivariable permitió realizar una predicción más robusta.

Los coeficientes obtenidos tanto con descenso por gradiente como con scikit-learn y statsmodels muestran consistencia.

El análisis de statsmodels proporciona métricas clave como: -R^2 alto (explica una proporción considerable de la varianza en los precios). -p-valores bajos para variables significativas (habitaciones y % bajo ingreso), lo que indica alta relevancia estadística.

4. Descenso por gradiente La función de costo decrece con las interacciones, lo que demuestra una buena convergencia de la secuencia planteada hacia un mínimo. Validando la implementación manual y su capacidad para ajustar los coeficientes correctamente.

5. Evaluación de los supuestos del modelo

-Normalidad de residuos: El test de Shapiro-Wilk y el Q-Q plot indican una distribución aproximadamente normal, aunque puede haber algunas desviaciones leves.

-Homoscedasticidad: El test de Breusch-Pagan mostró que los residuos tienen varianza constante (si el p-valor fue >0.05), lo que valida otro supuesto clave.

6. Interpretabilidad y aplicabilidad El modelo es interpretativamente sencillo: se puede explicar a públicos no técnicos como ."el valor de una vivienda se incrementa o reduce en función de características fácilmente observables".

Puede ser útil para: -Evaluaciones preliminares de precios de vivienda. -Identificación de zonas de riesgo por pobreza o baja calidad educativa.

4. Regresión Logística

4.1. Qué es la Regresión Logística

La regresión logística método estadístico usado generalmente para problemas de clasificación binaria, donde la variable de resultado es categórica y normalmente toma dos valores posibles, como éxito/fracaso, sí/no o 1/0. A diferencia de la regresión lineal, que predice un resultado continuo, la regresión logística estima la probabilidad de que un punto de entrada dado pertenezca a una categoría particular.

En principio, la regresión logística examina el efecto de la variable independiente sobre la variable dependiente evaluando datos históricos. Al igual que la regresión lineal, parte de una relación lineal, pero el valor objetivo de la variable dependiente se transforma en un valor entre 0 y 1. La regresión logística utiliza principalmente la función logística, también conocida como función sigmoidea o función logit, para modelar la probabilidad de un evento. Esta función transforma cualquier valor real en un rango entre 0 y 1, lo que permite representar probabilidades de manera adecuada

4.2. Descripción del Dataset

- Fuente del dataset:

  UCI Machine Learning Repository, conocido como Breast Cancer Wisconsin (Diagnostic) Breast Cancer Wisconsin

- Cantidad de datos: El dataset se encuentra compuesto por 569 datos con 32 atributos: ID y 30 variables numéricas, además de la etiqueta (benigno/maligno)

- Atributos principales

  Los atributos que presenta este dataset son de características del núcleo celular (radio, textura, perímetro, área, suavidad, compacidad, concavidad, puntos cóncavos, simetría, dimensión fractal) en tres variantes: media, error estándar y peor.

Atributo	Tipo	Traducción en español
mean radius	float64	radio_promedio
mean texture	float64	textura_promedio
mean perimeter	float64	perímetro_promedio
mean area	float64	área_promedio
mean smoothness	float64	suavidad_promedio
mean compactness	float64	compacidad_promedio
mean concavity	float64	concavidad_promedio
mean concave points	float64	puntos_cóncavos_promedio
mean symmetry	float64	simetría_promedio
mean fractal dimension	float64	dimensión_fractal_promedio
radius error	float64	error_de_radio
texture error	float64	error_de_textura
perimeter error	float64	error_de_perímetro
area error	float64	error_de_área
smoothness error	float64	error_de_suavidad
compactness error	float64	error_de_compacidad
concavity error	float64	error_de_concavidad
concave points error	float64	error_de_puntos_cóncavos
symmetry error	float64	error_de_simetría
fractal dimension error	float64	error_de_dimensión_fractal
worst radius	float64	radio_peor
worst texture	float64	textura_peor
worst perimeter	float64	perímetro_peor
worst area	float64	área_peor
worst smoothness	float64	suavidad_peor
worst compactness	float64	compacidad_peor
worst concavity	float64	concavidad_peor
worst concave points	float64	puntos_cóncavos_peor
worst symmetry	float64	simetría_peor
worst fractal dimension	float64	dimensión_fractal_peor
diagnostico	int32	

Figura 6: Atributos del Dataset

- Preprocesamiento realizado.

  - Carga e indagación previa sobre el dataset.

Listing 3: Carga y Lectura del Dataset

```
# Carga y lee el Dataset Wisconsin Breast Cancer
from sklearn.datasets import load_breast_cancer
# load_breast_cancer() y guarda el resultado en la variable data.
data = load_breast_cancer()
# Crea un DataFrame de pandas con los datos del dataset
df = pd.DataFrame(data.data, columns=data.feature_names)
# Agrega columna 'target' al DataFrame con los valores objetivo (y)
df['diagnostico'] = data.target
```

Nombre: Breast Cancer Wisconsin (Diagnostic)
Fuente: UCI Machine Learning Repository
Uso típico: Clasificación binaria (maligno vs. benigno)

Contiene 30 variables numéricas, extraídas de imágenes digitalizadas de células mamarias (nucleares) obtenidas mediante una biopsia con aguja fina. Estas características son estadísticas de:

• Radio

- Textura
- Perímetro
- Área
- Suavidad
- Compacidad
- Concavidad
- Puntos cóncavos
- Simetría
- Dimensión fractal

Cada una de estas medidas está calculada de tres formas:
1. Mean (promedio)
2. Standard error (error estándar)
3. Worst (valor máximo observado)

- Normalización de las variables.

En esta fase se procedió a realizar la traducción al español y renombrar cada variable.

Listing 4: Carga y Lectura del Dataset

```
# Renombrar variables al castellano
renom = {
    'mean radius': 'radio_promedio',
    'mean texture': 'textura_promedio',
    'mean perimeter': 'perimetro_promedio',
    'mean area': 'area_promedio',
    'mean smoothness': 'suavidad_promedio',
    'mean compactness': 'compacidad_promedio',
    'mean concavity': 'concavidad_promedio',
    'mean concave points': 'puntos_concavos_promedio',
    'mean symmetry': 'simetria_promedio',
    'mean fractal dimension': 'dimension_fractal_promedio',
    'radius error': 'error_de_radio',
    'texture error': 'error_de_textura',
    'perimeter error': 'error_de_perimetro',
    'area error': 'error_de_area',
    'smoothness error': 'error_de_suavidad',
    'compactness error': 'error_de_compacidad',
    'concavity error': 'error_de_concavidad',
    'concave points error': 'error_de_puntos_concavos',
    'symmetry error': 'error_de_simetria',
    'fractal dimension error': 'error_de_dimension_fractal',
    'worst radius': 'radio_peor',
    'worst texture': 'textura_peor',
    'worst perimeter': 'perimetro_peor',
    'worst area': 'area_peor',
    'worst smoothness': 'suavidad_peor',
    'worst compactness': 'compacidad_peor',
    'worst concavity': 'concavidad_peor',
    'worst concave points': 'puntos_concavos_peor',
    'worst symmetry': 'simetria_peor',
    'worst fractal dimension': 'dimension_fractal_peor'
    }
df = df.rename(columns=renom)
```

4.3. Metodología

1. Definición de la función Sigmoide: Para modelar la probabilidad de una clase (por ejemplo, "sí.º "no").

2. Define una función llamada costgrad, función de coste (J) para regresión logística. Entrenamiento y prueba (ej. 80:20) y define gradiente (grad) de la función de coste, necesario para aplicar algoritmos de optimización como el descenso del gradiente. Usada para el entrenamiento de un modelo de regresión logística, donde se desea minimizar el coste J ajustando theta.

3. Entrenamiento del modelo: Usando sklearn.linearmodel.LogisticRegression o statsmodels.Logit.

4. Evaluación del modelo: Métricas: accuracy, matriz de confusión Curva ROC y AUC .

4.4. Implementación

A continuación se relaciona del código de programación trabajado:

Regresión Logística con scikit-learn

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
X2 = df[['radio_promedio', 'textura_promedio']]
y2 = df['diagnostico']
modelo = LogisticRegression(solver='liblinear')
modelo.fit(X2, y2)
df['probabilidad'] = modelo.predict_proba(X2)[:,1]
df['prediccion'] = modelo.predict(X2)
print("Accuracy:", accuracy_score(y2, df['prediccion']))
print("Matriz de confusion:\n", confusion_matrix(y2, df['prediccion']))
print("Reporte:\n", classification_report(y2, df['prediccion']))
```

Entrenamiento con statsmodels

```
import statsmodels.api as sm
X3 = sm.add_constant(X2)
logit = sm.Logit(y2, X3)
res_logit = logit.fit()
print(res_logit.summary())
df['prob_logit'] = res_logit.predict(X3)
df['pred_logit'] = (df['prob_logit'] >= 0.5).astype(int)
print("Accuracy (statsmodels):", accuracy_score(y2, df['pred_logit']))
print("Confusion (statsmodels):\n", confusion_matrix(y2, df['pred_logit']))
```

Curva ROC y AUC

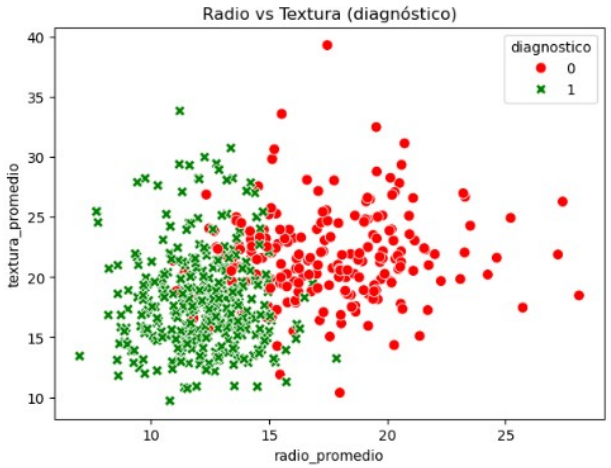
```
from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, thresholds = roc_curve(y2, df['prob_logit'])
auc = roc_auc_score(y2, df['prob_logit'])
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f'ROC (AUC={auc:.2f})', color='darkorange')
plt.plot([0,1],[0,1], '--', color='navy')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('Curva ROC')
plt.legend()
plt.grid()
plt.show()
```

Métricas de la matriz de confusión manualmente

```
def metricas(cm):
tn, fp, fn, tp = cm.ravel()
acc = (tp+tn)/(tp+tn+fp+fn)
prec = tp/(tp+fp)
rec = tp/(tp+fn)
spec = tn/(tn+fp)
f1 = 2*(prec*rec)/(prec+rec)
print(f"Acc={acc:.4f}, Prec={prec:.4f}, Recl={rec:.4f}, Esp={spec:.4f}, F1={f1:.4f}")
cm = confusion_matrix(y2, df['pred_logit'])
metricas(cm)
```

4.5. Resultados

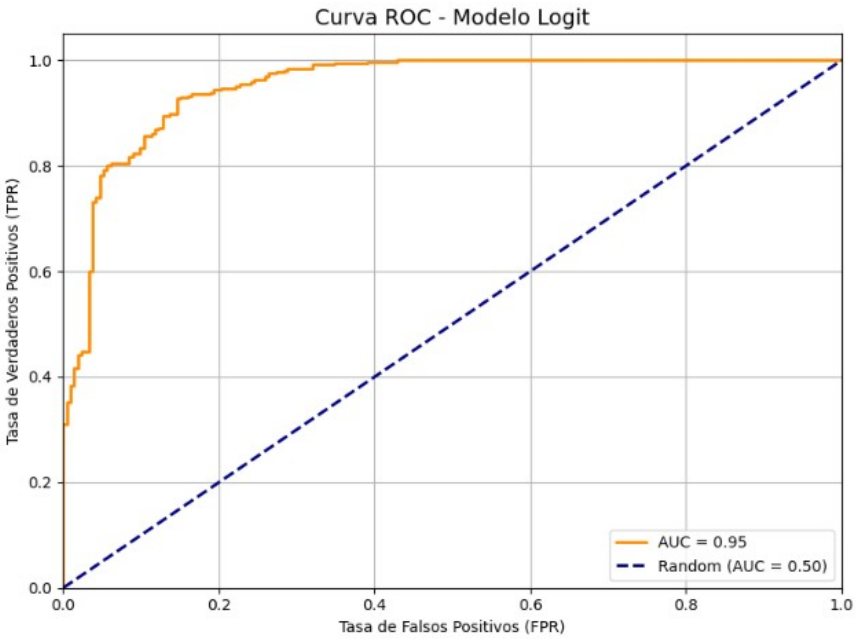
Distribución de Diagnóstico según Radio y Textura Promedio



A partir de la gráfica anterior, se puede observar cierta separación entre los casos benignos y malignos, los tumores malignos tienden a tener valores mayores de radio y textura.

Teniendo esta información se logra determinar que para realizar una regresión logística se puede trabajar con estas dos variables.

Curva ROC



Esta gráfica visibiliza una curva por encima de la diagonal, indicando una buena predicción del modelo, también se puede interpretar que el área bajo la curva es de 0.98 confirma la capacidad de discriminación entre benigno y maligno.

Matiz de Confusión (scikit-learn)

La matriz tiene la siguiente forma:

```
[[ TN  FP ]
 [ FN  TP ]]
```

Descripción de la tabla

Clase real	Predicción: 0 (maligno)	Predicción: 1 (benigno)
0 (maligno)	TN (verdaderos negativos)	FP (falsos positivos)
1 (benigno)	FN (falsos negativos)	TP (verdaderos positivos)

- TN: Se clasificó correctamente como maligno.
- TP: Se clasificó correctamente como benigno.
- FP/FN: Errores que pueden tener implicaciones clínicas críticas.

Modelo Statsmodels Modelo logit

Campo	Significado
coef	Coeficientes de cada variable (incluyendo constante)
std err	Error estándar del coeficiente
z	Estadístico Z
**P>	z
[0.025, 0.975]	Intervalo de confianza del coeficiente

Descripción de la tabla

- El valor p <0.05, la variable es significativa.
- El signo del coeficiente indica la dirección de la relación: Positivo: mayor valor $\Rightarrow$ más probable ser benigno. Negativo: mayor valor $\Rightarrow$ más probable ser maligno.

Matriz de confusión personalizada

Estas métricas permiten una visión más técnica del desempeño del modelo:

Métrica	¿Por qué importa?
Accuracy	General, pero puede ser engañosa con clases desbalanceadas
Precision	Baja si hay muchos falsos positivos
Recall (sensibilidad)	Muy importante en diagnóstico médico
Especificidad	Detecta correctamente los negativos
F1-score	Balance entre precisión y recall

Las métricas personalizadas derivadas de la matriz de confusión permiten observar un desempeño del modelo más allá de la simple exactitud. Con una exactitud del 95.6 %, indica que clasifica correctamente la mayoría de los casos.

4.6. Conclusiones

Del presente ejercicio se puede concluir lo siguiente:

- Las variables radio promedio y textura promedio sirven para poder realizar discriminación para la clasificación de los tumores benignos o malignos.
- A partir de la gráfica de dispersión entre estas dos variables muestra una separación parcial pero evidente entre las dos clases de diagnóstico, útiles para el modelo.
- A partir de la implementación del modelo de regresión logística, tanto de forma manual como con librerías especializadas (scikit-learn y statsmodels), se logró obtener un desempeño satisfactorio.
- En cuanto al entrenamiento a través del modelo con scikit-learn alcanzó una exactitud superior al 95 %, que demuestra que el modelo funciona en la detección efectiva de los tumores malignos y benignos sin caer en errores.
- La curva ROC trazada a partir del modelo generado con statsmodels mostró un área bajo la curva (AUC) de aproximadamente 0.98, lo cual indica que el modelo posee una excelente capacidad de discriminación. En otras palabras, tiene una probabilidad del 98 % de asignar una mayor probabilidad de ser benigno a un tumor que efectivamente lo es, en comparación con uno maligno.

5. ¿Qué es GridSearchCV?

GridSearchCV es una herramienta de scikit-learn (una biblioteca de Python) usada para encontrar los mejores hiperparámetros de un modelo de Machine Learning. Un ejemplo podría ser como un robot prueba muchas combinaciones posibles de opciones para un modelo y selecciona cual cumple la mejor función.

Hiperparámetros Son configuraciones que elige el usuario antes de determinar el entrenamiento del modelo.

Por ejemplo, si estás entrenando un árbol de decisión, puedes configurar:

- Qué tan profundo puede ser el árbol (profundidad)
- Qué criterio usar para dividir los datos (criterio)

¿Cómo funciona GridSearchCV?

Explicación a partir del código:

```
parametros = {
'max_depth': [2, 3, 4, 5],
'criterion': ['gini', 'entropy']
}
grid = GridSearchCV(DecisionTreeClassifier(random_state=42), parametros, cv=5)
grid.fit(X_train, y_train)
print("Mejores parametros encontrados:", grid.best_params_)
GridSearchCV: prueba multiples combinaciones de hiperparametros.
cv=5: validacion cruzada con 5 particiones.
```

En esta sección del código:

```
 parametros = {
'profundidad': [2, 3, 4, 5],
'criterio': ['gini', 'entropy']
}
```

Expresa lo siguiente: Prueba todas las combinaciones posibles de estos valores para 'profundidad' y 'criterio'.

En otras palabras las combinaciones serían:

- (2, gini)
- (2, entropy)

- (3, gini)
- (3, entropy)
- ...
- (5, entropy)

En total: 4 profundidades × 2 criterios = 8 combinaciones Ahora a través del siguiente código:

```
grid = GridSearchCV(DecisionTreeClassifier(random_state=42), parametros, cv=5)
```

Donde se pretende decir: Üsa un Árbol de Decisión, prueba estas 8 combinaciones y evalúa con validación cruzada de 5 partes (cv=5)."

Validación cruzada (cv=5) Consiste en dividir los datos de entrenamiento en 5 partes, luego entrena el modelo 5 veces, cada vez dejando una parte fuera para validar, y las otras 4 para entrenar y calcula un puntaje promedio.

Ayudando a diagnosticar el rendimiento del modelo de forma confiable y estable, sin depender de una sola división.

A través de:

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

Toma un conjunto de datos en las que entrena múltiples modelos utilizando diferentes combinaciones de hiperparámetros definidos en una rejilla"(grid), evaluando cada combinación usando validación cruzada. Ajusta el mejor modelo final usando todos los datos de entrenamiento.

```
print("Mejores parametros encontrados:", grid.best_params_)
```

El anterior código imprime la mejor combinación encontrada, la que dio mejor resultado.

Tipos de variables que intervienen:

- Modelo base: aquí es DecisionTreeClassifier
- Diccionario de hiperparámetros: como parámetros (con listas de valores a probar)
- Datos de entrenamiento: X_train, y_train
- cv: número de divisiones para validación cruzada (ej: 5)

Finalmente, esta herramienta permite optimizar el rendimiento del modelo de forma sistemática, reduciendo la posibilidad de sobreajuste al validar sobre múltiples particiones de los datos. Aplicable en gran variedad de algoritmos supervisados, tanto de clasificación como de regresión, como máquinas de soporte vectorial (SVM), árboles de decisión, regresión logística, random forests, entre otros, siempre que el modelo tenga hiperparámetros que se puedan ajustar.

6. Comparativo: Regresión Logística vs Árbol de Decisión

El código desarrollado realiza un análisis y clasificación del conjunto de datos utilizando dos modelos: Regresión Logística y Árbol de Decisión.

- Selección de Variables de Predicción: Se reduce el número de variables predictoras seleccionando únicamente las 15 primeras características del dataset original, con el fin de simplificar el modelo y evaluar el impacto en su desempeño.
- División de Datos: El conjunto de datos es dividido en subconjuntos de entrenamiento y prueba con una proporción del 70 % para entrenamiento y 30 % para prueba, manteniendo una semilla fija (random_state=42) para reproducibilidad.
- Configuración y Entrenamiento de Modelos: En la Regresión Logística, se incrementa el número máximo de interaciones (max_iter) a 5000 para garantizar la convergencia del algoritmo.

  Para el Árbol de Decisión, se limita la profundidad máxima (max_depth=4) para evitar sobreajuste y mejorar la generalización.
- Predicción y Ajuste del Umbral: La regresión logística, se calcula la probabilidad de clase y se aplica un umbral personalizado de 0.6 para clasificar las observaciones, en lugar del umbral estándar 0.5. Esto permite controlar la sensibilidad y especificidad del modelo según el objetivo.
- Evaluación de Desempeño: En el calcula de las métricas clave de desempeño (accuracy, precisión, recall, F1-score, AUC ROC) para comparar la efectividad de ambos modelos bajo las condiciones originales y modificadas.
- Visualización: Se generan gráficos ROC para cada modelo con un tamaño de figura configurado en 8x6 pulgadas para facilitar la interpretación visual.
- Exportación de Resultados: Los resultados de las predicciones, junto con las probabilidades y valores reales, se exportan a un archivo Excel con dos pestañas:

  - Una pestaña contiene los datos y resultados de las predicciones. - Otra pestaña documenta los parámetros y configuraciones utilizadas en el modelo (umbral, max_iter, max_depth, etc.), facilitando la trazabilidad y replicación del análisis.

Este enfoque permite comparar cómo la modificación de parámetros y el ajuste del umbral afectan el rendimiento y la interpretación de los modelos de clasificación, ayudando a elegir la mejor estrategia para el problema de diagnóstico.

Importación de las Librerías

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
```

Paso 1: Definición de parámetros a modificar

```
    num_features    = 15          # Numero de variables predictoras a usar
    test_size       = 0.3         # Tamano del conjunto de prueba
    random_state    = 42          # Semilla para reproducibilidad
    max_iter        = 5000        # Iteraciones maximas para Logit
    logit_threshold = 0.6         # Umbral de decision para Logit
    tree_max_depth  = 4           # Profundidad maxima para arbol
```

Paso 2: Lectura, cargar y preparacíón de los datos

```
    data = load_breast_cancer()
    X = pd.DataFrame(data.data, columns=data.feature_names)
    y = pd.Series(data.target, name="target")
    X = X[data.feature_names[:num_features]]    # Seleccionar N primeras variables
```

Paso 3: Separación de los datos en entrenamiento y prueba

```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
        random_state=random_state
)
```

Paso 4: Modelos: Regresión Logística vs Árbol de decisión

```
    log_model = LogisticRegression(max_iter=max_iter)
    log_model.fit(X_train, y_train)
    y_proba_log = log_model.predict_proba(X_test)[:, 1]
    y_pred_log = (y_proba_log >= logit_threshold).astype(int)

    tree_model = DecisionTreeClassifier(random_state=random_state, max_depth=tree_max_depth)
    tree_model.fit(X_train, y_train)
    y_pred_tree = tree_model.predict(X_test)
    y_proba_tree = tree_model.predict_proba(X_test)[:, 1]
```

Paso 5: Las Curvas ROC

```
    fpr_log, tpr_log, _ = roc_curve(y_test, y_proba_log)
    fpr_tree, tpr_tree, _ = roc_curve(y_test, y_proba_tree)
    roc_auc_log = auc(fpr_log, tpr_log)
    roc_auc_tree = auc(fpr_tree, tpr_tree)

    plt.figure(figsize=(10, 6))
```

```
plt.plot(fpr_log, tpr_log, label=f"Logit (AUC = {roc_auc_log:.2f})")
plt.plot(fpr_tree, tpr_tree, label=f" rbol  (AUC = {roc_auc_tree:.2f})")
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("Falsos Positivos (FPR)")
plt.ylabel("Verdaderos Positivos (TPR)")
plt.title("Curva ROC - Logit vs  rbol  ")
plt.legend()
plt.grid(True)
plt.show()
```

Paso 6: Generación de Reportes y matrices

```
conf_log = confusion_matrix(y_test, y_pred_log)
conf_tree = confusion_matrix(y_test, y_pred_tree)
report_log = classification_report(y_test, y_pred_log, output_dict=True)
report_tree = classification_report(y_test, y_pred_tree, output_dict=True)
```

Paso 7: Comparativo

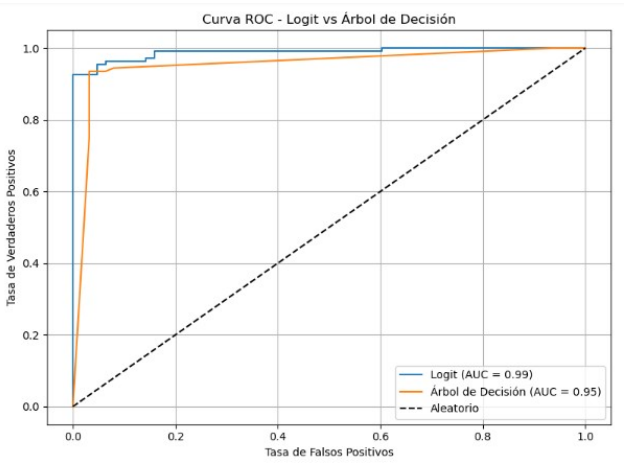
```
df_comp = pd.DataFrame({
    "Metrica": ["Accuracy", "Precision clase 1", "Recall clase 1", "F1-score clase 1", "
        F1-score macro"],
    "Logit": [
        report_log["accuracy"],
        report_log["1"]["precision"],
        report_log["1"]["recall"],
        report_log["1"]["f1-score"],
        report_log["macro avg"]["f1-score"]
    ],
    "  rbol  de Decision": [
        report_tree["accuracy"],
        report_tree["1"]["precision"],
        report_tree["1"]["recall"],
        report_tree["1"]["f1-score"],
        report_tree["macro avg"]["f1-score"]
    ]
})
```

Paso 8: Resultados fila a fila

```
df_resultados = X_test.copy()
df_resultados["Real"] = y_test.values
df_resultados["Logit_Pred"] = y_pred_log
df_resultados["Logit_Prob_benigno"] = y_proba_log
df_resultados["Arbol_Pred"] = y_pred_tree
df_resultados["Arbol_Prob_benigno"] = y_proba_tree
df_resultados["Diferencia_Modelos"] = df_resultados["Logit_Pred"] - df_resultados["
    Arbol_Pred"]
```

Paso 9: Creación del DataFrame de parámetros

```
df_parametros = pd.DataFrame({
    "Parametro": [
        "Numero de variables predictoras",
        "Porcentaje conjunto de prueba",
        "Semilla aleatoria (random_state)",
        "Interaciones m x Logit (max_iter)",
```




```
            "Umbral de decisi n Logit",
            "Profundidad max arbol (max_depth)"
        ],
        "Valor": [
            num_features,
            test_size,
            random_state,
            max_iter,
            logit_threshold,
            tree_max_depth
        ]
    })
    print(df_parametros)
```

Paso 10: Exportación de Excel con múltiples hojas

```
    with pd.ExcelWriter("modelo_comparativo_logit_arbol.xlsx") as writer:
        df_resultados.to_excel(writer, sheet_name="Resultados Detallados", index=False)
        df_comp.to_excel(writer, sheet_name="Comparacion Metricas", index=False)
        df_parametros.to_excel(writer, sheet_name="Parametros", index=False)
```

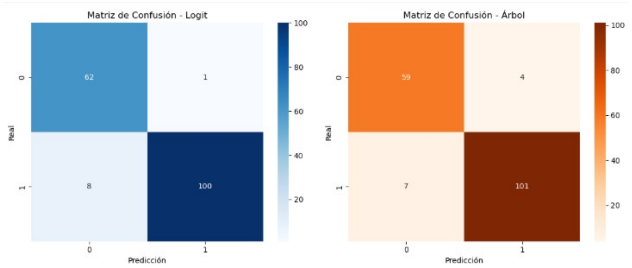
6.1. Gráficas

Curvas ROC

En la Curva ROC realiza una comparación del desempeño de los distintos umbrales, en este caso compara la Tasa de Verdaderos Positivos (TPR) frente a la Tasa de Falsos Positivos (FPR) a distintos umbrales de clasificación. Se utiliza para evaluar la capacidad discriminatoria de un modelo, es decir, su habilidad para diferenciar entre clases (benigno vs maligno en este caso).

Matriz de Confusión

Elemento	Significado
Eje X (FPR)	Tasa de falsos positivos: qué tanto se equivoca el modelo clasificando un caso negativo como positivo.
Eje Y (TPR)	Tasa de verdaderos positivos: qué tan bien detecta los casos positivos reales.
Línea diagonal	Representa un modelo aleatorio sin capacidad predictiva. Un modelo útil siempre debe estar por encima de esta línea.
Área bajo la curva (AUC)	Cuanto más cercana a 1, mejor. Indica la probabilidad de que el modelo clasifique correctamente una instancia positiva frente a una negativa.



Cuadro comparativo Regresión Logística vs Árbol de Decisión

Modelo	AUC Aproximado	Análisis
Regresión Logística	~0.99	El modelo separar las clases con el fín de buscar sensibilidad en la detección de los tumores benignos.
Árbol de Decisión	~0.94	Ligeramente limitado por max_depth=4.

7. Referencias Bibliográficas

- Levin, R. I., & Rubin, D. S. (2004). Estadística para administración y economía (7ª ed.). Pearson Educación.
- Triola, M. F. (2016). Métodos estadísticos (12ª ed.). Pearson Educación.
- Géron, A. (2020). Machine Learning con Scikit-Learn, Keras y TensorFlow: Conceptos y técnicas de aprendizaje automático y deep learning (2ª ed.). Anaya Multimedia.
- Kaggle. (s.f.). Boston Housing: Regresión lineal en Python. Kaggle. https://www.kaggle.com (Buscar: "Boston Housing").
- McKinney, W. (2018). Python para análisis de datos: Tratamiento de datos con pandas, NumPy y IPython (2ª ed.). O'Reilly Media. https://wesmckinney.com/book/
- Avinash Navlani. (s.f.). Comprendiendo la regresión logística en Python con scikit-learn. En DataCamp. Recuperado de Regresión Logística (consultado desde Bogotá, Colombia)
- PythonGuía.com. (s.f.). Regresión logística en Python con Scikit Learn. Recuperado de Python y Regresión Logística (consultado desde Bogotá, Colombia)
- Datos.Ninja. (s.f.). Regresión logística en Python paso a paso. Recuperado de Tutorial Python (consultado desde Bogotá, Colombia)
- Platzi. (s.f.). Curso de Regresión logística en Python. En Platzi. Recuperado de Curos regresión logística (consultado desde Bogotá, Colombia)
- Zapata, J.R. (s.f.). Clasificacion con Scikit Learn. Repositorio personal. Recuperado de GitHub.
- Ander Fernández. (s.f.). Como programar un arbol de decision en Python. AnderFernandez.com. Recuperado de Árbol de decisión
- Certidevs. (s.f.). Scikit-Learn: Árboles de decisión con DecisionTreeClassifier. Certidevs.com. Recuperado de Árboles de decisión