UNIVERSIDAD SIMÓN BOLÍVAR

Proyecto I

Junior Lara y Astrid Alvarado

June 3, 2025

1 RESUMEN

Este reporte presenta los resultados y análisis de un proyecto centrado en la implementación y experimentación de modelos de regresión lineal utilizando el algoritmo de Descenso del Gradiente. El estudio se llevó a cabo en Python, empleando herramientas como Jupyter Notebook y bibliotecas como Pandas, Numpy y Matplotlib para el procesamiento y visualización de datos. El proyecto abordó dos aspectos principales: la implementación técnica del algoritmo y su aplicación en conjuntos de datos reales, con énfasis en la importancia de la normalización (Z-score). ✓

Además, se exploró el impacto de la tasa de aprendizaje en el rendimiento del algoritmo, observando que valores intermedios, más ajustados, permitían una convergencia rápida y estable, mientras que valores extremos (muy pequeños o muy grandes) resultaban en un rendimiento deficiente. El proyecto también incluyó la limpieza y preparación de datos, como la eliminación de valores nulos y la selección de atributos basada en su correlación con la variable objetivo, mejorando así la capacidad predictiva del modelo.

faltaron: datasets estudiados
resultados obtenidos

2 DETALLES DE IMPLEMENTACIÓN Y EXPERIMENTACIÓN

2.1 Implementación

El proyecto fue realizado mediante la utilización del lenguaje de programación `Python`. En particular, el equipo trabajó con las siguientes versiones del lenguaje:

- `Python 3.10.12`
- `Python 3.12.3` ✓

Asimismo, el proyecto requirió la instalación de las siguientes librerías:

- `Pandas 2.2.3`
- `Numpy 2.2.6` ✓
- `Matplotlib 3.10.3`

En conjunto de sus respectivas dependencias. Por tal motivo, se hizo uso de entornos virtuales con el fin de que ambos integrantes trabajaran bajo las mismas versiones de las librerías y evitar posibles problemas de incompatibilidad.

Adicionalmente, todo el proyecto fue elaborado mediante el uso de la herramienta `Jupyter Notebook` con el fin de aprovechar la creación de documentos computacionales que combinan código, texto, visualizaciones y ecuaciones.

2.1.1 Detalles del Algoritmo "Descenso del Gradiente"

La implementación de este algoritmo consta de dos (2) funciones implementadas en el archivo `linear_regression.py`:

- `compute_cost(prediction, y)`, que calcula la función de coste $J()$. Toma como argumento el vector de predicciones y el vector de valores objetivos, representados por arreglos de `Numpy`. La misma retorna el **Error Cuadrático Medio** de los vectores. Cabe destacar que para evitar desbordamiento en el cómputo de la diferencia de los vectores se hizo uso de la función `np.clip` con el fin de mantener los valores dentro de un intervalo $[-1e^{10}, 1e^{10}]$
- `gradient_descent(X, y, theta, alpha, num_iterations, bias, epsilon)`, que implementa el algoritmo de descenso del gradiente. Esta función toma como parámetros los siguientes valores:
  - `X`, la matriz de características.
  - `y`, el vector de valores objetivo.
  - `theta`, el vector de pesos.
  - `alpha`, la tasa de aprendizaje.
  - `num_iterations`, la cantidad de iteraciones a realizar.
  - `bias`, el sesgo inicial con valor 0.1 por defecto.
  - `epsilon`, la tolerancia para la convergencia con valor $1e^{-3}$

  Y devuelve la tripleta `(theta, bias, cost_history)`, los cuales representan al vector de pesos actualizados, el sesgo final y los costos obtenidos a lo largo del algoritmo.

2.2 Experimentación

2.2.1 Detalles de Normalización de los datos

En los incisos en los que se requirió la normalización de los datos, se realizó mediante la **normalización Z-score**:

$$X_{norm} = \frac{X - \mu}{\sigma} \tag{2.1}$$

Siendo μ la media de los valores y σ su desviación estándar. Para la parte 3, dado que los datos se componen de valores númericos y valores categóricos, se hizo un filtrado para trabajar únicamente con valores numéricos y así poder normalizar los mismos.

Además, se omitieron las columnas `PID, Order` dado que son valores únicos asociados a los datos, por lo que normalizarlos no es requerido. Del mismo modo, la columna `Fireplaces` se decidió no normalizar dado que esta contenía valores en el intervalo $[0,3]$, por lo que la normalización no era relevante. Otro punto a favor de la no normalización de estos datos radica en que era necesario mantener dicha naturaleza para poder calcular la columna `FireYN`.

2.2.2 Detalles de Limpieza de los datos

La limpieza de datos para la parte 3 se realizó mediante el uso de los DataFrame de `Pandas` que permite la manipulación de datos a través de indexación booleana y proyección de columnas. En general, se siguió el proceso de limpieza sugerido en [1], incluyendo la escogencia de una muestra aleatoria de 200 datos. Sin embargo, para el inciso 3.d se requirió de un filtrado adicional con el fin de eliminar las celdas vacías de la columna `Garage Yr Blt`, para así poder hacer la experimentación sin inconsistencias o errores en el código.

pq solo 200?

debían filtrarse antes.
Ya se habían quedado con muy pocos datos

Cabe destacar que para escoger la muestra se utilizó un valor como semilla con el fin de asegurar la reproducibilidad en ejecuciones posteriores.

2.2.3 Detalles de Separación de datos (80%-20%)

La separación de los datos para la parte 3.c se hizo gracias a las funcionalidades que `Pandas` ofrece. En general, se mezclaron los datos de forma aleatoria estableciendo una semilla con el fin de reproducir los mismos resultados en ejecuciones posteriores. Con esto, se calculó el punto de corte del DataFrame como sigue

$$cutoff = \lfloor 0.8 \cdot size \rfloor$$

Donde $size$ es la cantidad de filas que tiene el DataFrame. Una vez hecho esto, se separan los datos mediante la funcionalidad de `slices` que ofrece `Python`, dejando el 80% de los datos para el entrenamiento y el 20% restante para realización de pruebas.

2.2.4 Detalles del Modelo 2

En la parte 3 se pide la creación de un modelo para los datos ofrecidos en [1]. En concreto, se solicitó que se creara un modelo utilizando las columnas `Total Basement, Gr Liv Area` y 3 atributos adicionales, por lo que para este proceso de decidió escoger los atributos según su correlación con los valores objetivos (`Sale Prices`).

Al seleccionar atributos con mayor correlación respecto a la variable objetivo, se eligen aquellas características que más influyen en la predicción de `Sale Prices`, mejorando así la capacidad predictiva del modelo, ya que se priorizan las variables que aportan más información relevante. Por lo tanto, se calculó la correlación de cada atributo numérico, exceptuando a los ya usados en el modelo 1, con `Sale Prices` para luego seleccionar los tres atributos con mayor valor absoluto de correlación.

Cabe destacar que la correlación se calculó mediante la función `corr()` que ofrece `Pandas`.

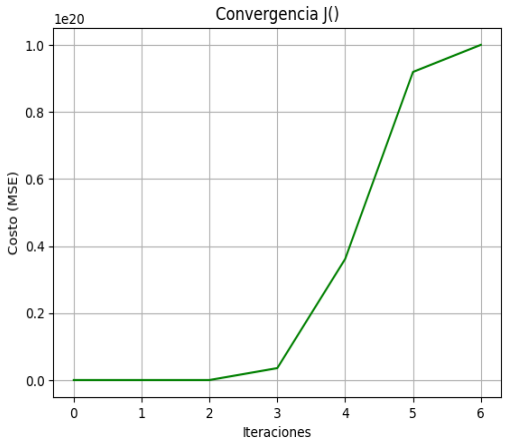
3 Resultados

A continuación se analizará los resultados de los experimentos realizados con la implementación propia del algoritmo **Descenso del Gradiente**

3.1 Parte 2

3.1.1 Peso corporal en función del peso de cerebro

La curva de convergencia muestra que la función de costo $J(\theta)$ no disminuye como se esperaría teóricamente en un descenso del gradiente bien ajustado, sino que presenta un crecimiento



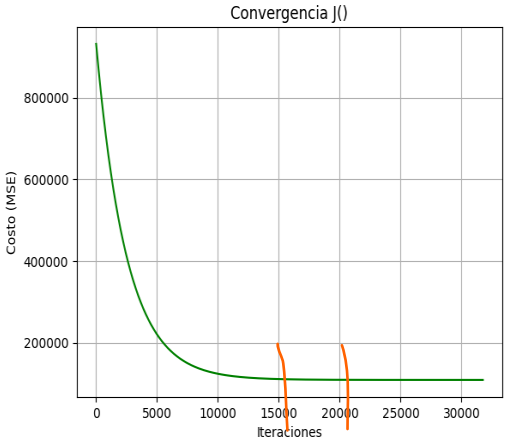
(a) Curva de convergencia de la función de coste

(b) Scatterplot de los datos y la curva de predicción

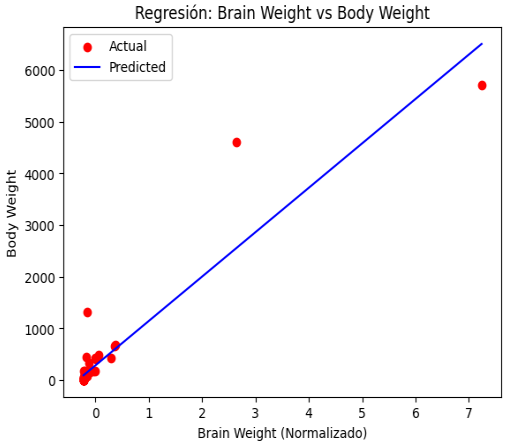
Figure 3.1: Gráficos obtenidos al ejecutar el algoritmo (datos sin normalizar)

desde 0 hasta valores muy altos. Esto ocurre porque los datos no están normalizados y tienen escalas muy grandes, lo que provoca que el algoritmo sea inestable y no logre minimizar el costo correctamente para el valor de alpha utilizado. Se evidencian 6 iteraciones de las 50000 que recibe el algoritmo como argumento, esto dada la implementación del algoritmo que presenta una condición de parada cuando la diferencias de los costos sea menor a epsilon.

El scatterplot muestra los datos reales y las predicciones del modelo. Sin embargo, debido a la falta de normalización y la escala extremadamente grande de los datos (del orden de 1×10^{19}), la curva de regresión no se ajusta adecuadamente a los puntos.



(a) Curva de convergencia de la función de coste



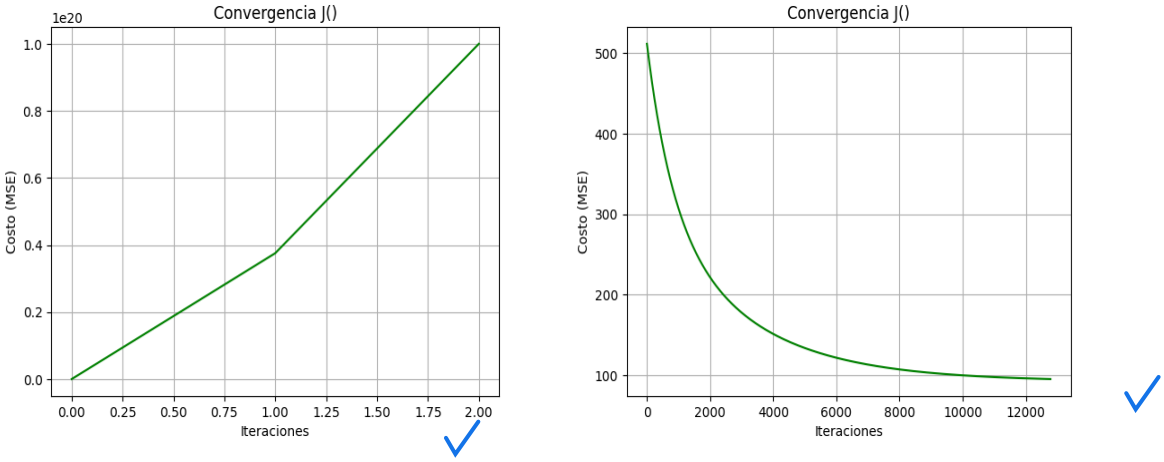
(b) Scatterplot de los datos y la curva de predicción

Figure 3.2: Gráficos obtenidos al ejecutar el algoritmo (datos normalizados)

Al analizar la curva de convergencia con datos normalizados, se observa que la función de costo $J(\theta)$ disminuye de manera suave y progresiva a lo largo de las iteraciones, mostrando un comportamiento estable y típico del descenso del gradiente. Esto contrasta fuertemente con el caso de los datos sin normalizar, donde la curva de costo crecía rápidamente y el algoritmo se detenía prematuramente debido a inestabilidad numérica.

En el scatterplot, la curva de regresión ajustada con datos normalizados se adapta mucho mejor a la tendencia de los datos, y las predicciones se distribuyen de forma coherente respecto a los valores reales. Esto indica que el modelo logra aprender correctamente la relación lineal entre las variables.

3.1.2 Tasa de homicidios por millón de habitantes



(a) Datos sin normalizar

(b) Datos normalizados

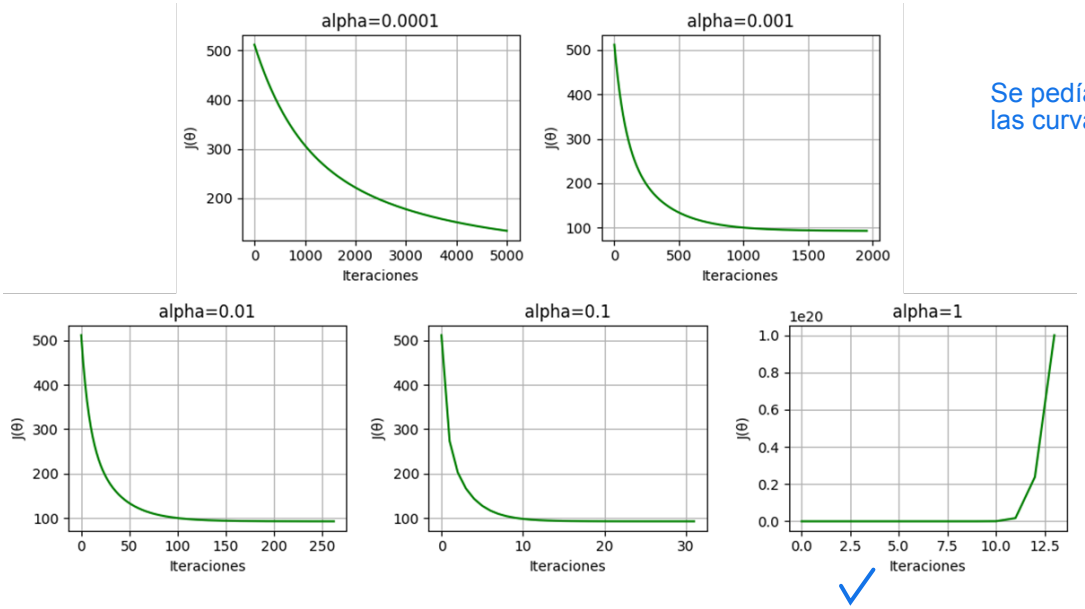
Figure 3.3: Curva de convergencia de la función de coste

- Sin normalizar: La curva de convergencia muestra que la función de costo $J(\theta)$ crece de manera abrupta y el algoritmo se detiene rápidamente, usualmente en la segunda iteración, a pesar de haber solicitado 50,000 iteraciones. Esto indica inestabilidad numérica: los valores de las variables son tan grandes y están en escalas tan distintas que el descenso del gradiente no puede avanzar correctamente y el costo se dispara.
- Normalizados: El costo disminuye progresivamente y el algoritmo logra avanzar durante un número considerable de iteraciones, cumpliendo su objetivo de optimización. Esto evidencia que la normalización permite que el descenso del gradiente sea estable y eficiente.

La normalización no solo es útil, sino que en la práctica suele ser necesaria cuando se trabaja con descenso del gradiente y variables en escalas muy diferentes. Sin normalizar, el algoritmo puede volverse inestable, divergir o detenerse prematuramente, como se observa en los gráficos. Normalizar garantiza estabilidad numérica, una convergencia más rápida y resultados más confiables. Por lo tanto, la normalización es una etapa fundamental en el preprocesamiento para modelos lineales entrenados con descenso del gradiente.

Se puede notar que: Dónde? discusión antes de la figura

- Para valores pequeños de alpha (0.0001 y 0.001), la función de costo disminuye de manera estable pero lentamente, requiriendo muchas iteraciones para acercarse al mínimo.
- Para valores intermedios (0.01 y 0.1), la convergencia es mucho más rápida y eficiente, alcanzando valores bajos de costo en pocas iteraciones.
- Para valores grandes de alpha (1), la curva muestra inestabilidad o incluso divergencia, lo que puede llevar a que el costo no disminuya adecuadamente o incluso aumente.



Se pedían todas las curvas en 1 gráfica

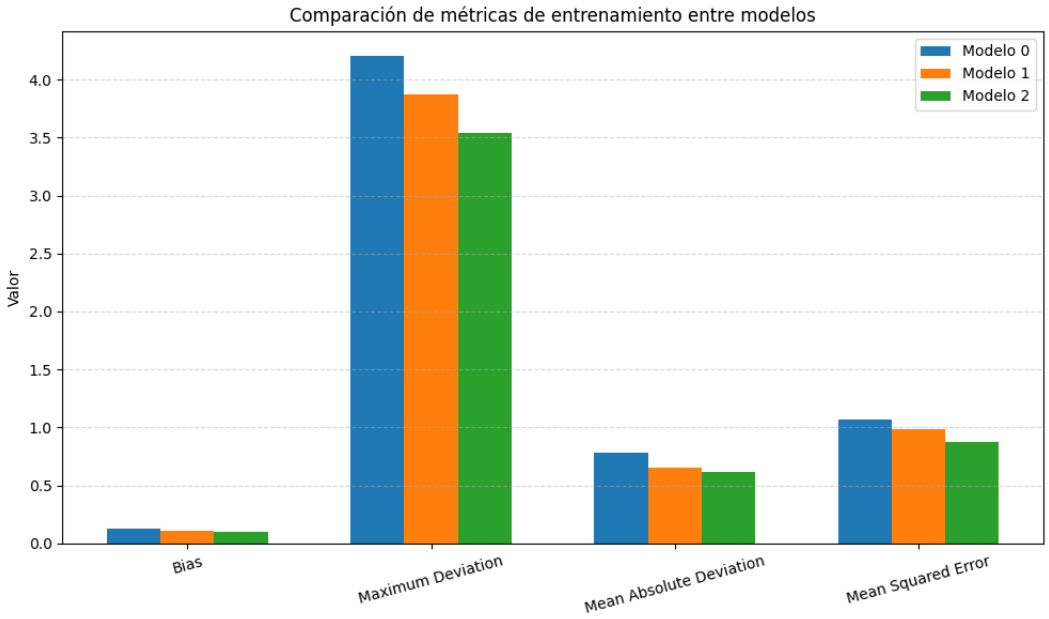
Figure 3.4: Curva de convergencia de la función de coste con distintos valores de `alpha`

La tasa de aprendizaje controla el tamaño de los pasos que da el algoritmo en cada iteración.

- Si alpha es muy pequeño, el algoritmo avanza muy lento y tarda mucho en converger.
- Si alpha es adecuado, el algoritmo converge rápido y de manera estable.
- Si alpha es muy grande, el algoritmo puede volverse inestable y no converger, o incluso divergir.

Por lo tanto, elegir un valor apropiado de alpha es fundamental para lograr una convergencia eficiente y estable en el descenso del gradiente.

3.2 Parte 3



Los valores de cada métrica tienen diferentes escalas. Por eso, deberían estar en gráficas diferentes

Figure 3.5: Comparación de métricas para los datos Train

Train vs Test??

Teniendo que:

- Para Bias se puede ver que los datos de entrenamiento tienen un valor positivo, siendo el del modelo 2 el más pequeño, por lo que indica una menor sobreestimación del precio.
- Para Maximum Deviation se puede ver que el modelo 2 tiene un menor valor de peor error cometido. Esto quiere decir que este modelo tiene una mejor capacidad de predicción.
- Para Mean Absolute Deviation se puede apreciar que el modelo 2 tiene un menor valor de error medio, lo que indica que este es más preciso en las predicciones realizadas. Hay que tener en cuenta que este valor no toma en cuenta las sobreestimaciones o subestimaciones realizadas.
- Para Mean Square Error se tiene que el modelo 2 tiene un menor error cuadrático medio, señal de que las estimaciones están más cerca del valor objetivo.

Los modelos se evalúan sobre es TEST set!!

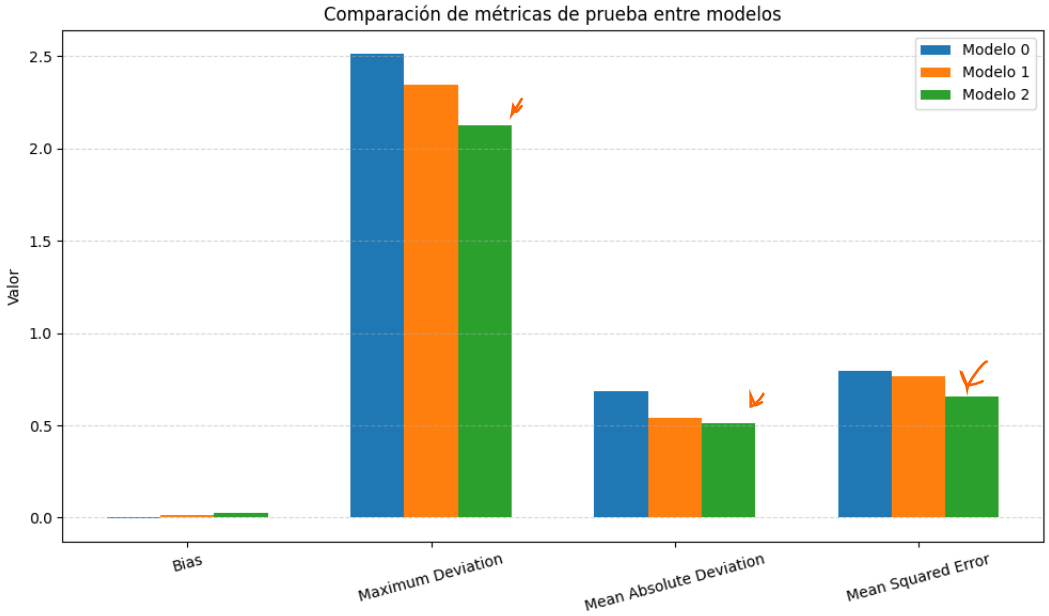


Figure 3.6: Comparación de métricas para los datos Test

Teniendo que:

- Para Bias se observa el siguiente ranking Modelo 0 > Modelo 1 > Modelo 2. De esta forma se indica que Modelo 0 posee una sobreestimación más alta que los de mas modelos.
- Para Maximum Deviation se observa el siguiente ranking Modelo 0 > Modelo 1 > Modelo 2. De esta forma se indica que el Modelo 0 posee una peor predicción en los datos que los demas modelos, lo que favorece al modelo 2.
- Para Mean Absolute Deviation se observa el siguiente ranking Modelo 0 > Modelo 1 > Modelo 2. De esta forma se indica que el Modelo 0 posee un error medio absoluto más alto que los demás modelos, lo que favorece al modelo 2.
- Para Mean Square Error se observa el siguiente ranking Modelo 0 > Modelo 1 > Modelo 2. De esta forma se indica que el Modelo 0 posee un error cuadritico más alto que los demás modelos, lo que favorece al modelo 2.

Mejor Modelo??

4 Conclusión

Este proyecto confirmó que el éxito del Descenso del Gradiente en regresión lineal depende de tres pilares: preprocesamiento de datos, selección adecuada de atributos y ajuste de hiperparámetros. La normalización (Z-score) demostró ser esencial para garantizar estabilidad y convergencia, mientras que la elección de una tasa de aprendizaje (α) equilibrada evitó divergencias sin sacrificar velocidad. Además, la limpieza de datos y la selección de características basada en correlación mejoraron la eficiencia y precisión del modelo.
Recomendaciones clave:

- Normalizar siempre los datos antes de aplicar Descenso del Gradiente, especialmente cuando las características tienen escalas dispares.
- Validar α en un rango amplio (e.g., 0.001 a 0.1) mediante técnicas como grid search para equilibrar velocidad y estabilidad.
- Priorizar atributos de alta correlación con la variable objetivo, pero complementar con análisis de multicolinealidad para evitar sobreajuste.
- Usar entornos virtuales y semillas aleatorias (como se hizo con Pandas) para garantizar reproducibilidad en entornos colaborativos.

Experiencias clave:

- La división 80%-20% de los datos aseguró que los modelos no sobreajustaran, validándose con métricas consistentes (Figura 3.5).
- Herramientas como Jupyter Notebook agilizaron la integración de código, visualizaciones y análisis, mientras que el uso de np.clip evitó desbordamientos numéricos.
- La selección de características mediante corr() de Pandas optimizó el Modelo 2, pero reveló limitaciones al excluir variables categóricas, sugiriendo explorar técnicas de codificación en futuras iteraciones.

5 Referencias

[1] De Cock, Dean. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education, Volume 19, Number 3, (2011). [Fecha de consulta: 2 de Junio 2025]. Disponible en: http://ww2.amstat.org/publications/jse/v19n3/decock.pdf