Instituto Politécnico Nacional
Escuela Superior de Cómputo
(ESCOM)

Ingeniería en Inteligencia Artificial
Grupo: 5BM1
Unidad de aprendizaje: Aprendizaje de Máquina
Práctica #4 modelos de regresión para machine learning
Profesor: Abdiel Reyes Vera
Alumno:
Velázquez Arrieta Eduardo Uriel
Fecha de entrega: 22 de Septiembre 2025

índice de contenido

Índice de imágenes

1. Introducción

Los humanos cuentan con dos características primordiales que nos ayudaron a sobrevivir y a escalar en la cadena alimenticia; no, no estamos hablando del pulgar y una postura erguida para poder observar a nuestras presas de mejor manera, sino más bien de la corteza cerebral, ya que esta se encarga de realizar dos acciones fundamentales, predecir y clasificar.

Ahora nos preguntamos ¿por qué estas dos acciones son clave para la supervivencia y evolución?; esto es simple, clasificar es algo tan primitivo como poder saber que frutas, hojas, etc; son comibles y cuales otras nos pueden llegar a causar malestares o inclusive la muerte. Por otra parte, el poder predecir es algo que hacemos de manera casi automática, algo así como respirar; por ejemplo, si vemos que el día está nublado, hay relámpagos y el ambiente se siente húmedo podemos predecir una lluvia y así con otras muchas situaciones que se nos presentan en el día a día.

Pero ¿qué relación guarda una cualidad tan primitiva de los humanos con la inteligencia artificial?; esto es una pregunta relativamente compleja, pero que la respuesta se encuentra dentro de la misma, para esto tenemos que retomar los inicios de la computación, donde era casi inimaginable el hecho de que una máquina pudiera realizar cálculos numéricos miles de veces más rápido que una persona común; pero estos tenían una gran limitante, no podían imitar ciertas acciones que son básicas para los humanos, algunos ejemplos son: medir la fuerza con la que se debe tomar un huevo para que no se rompa, reconocer una cara, o en nuestro caso particular y el que más nos interesa, poder clasificar y predecir con base en ciertos datos y umbrales.

Como ya sabemos la inteligencia artificial trata de simular los procesos cognitivos que los humanos, tienen; aunque suena a un tema novedoso y nuevo, la realidad es que desde los inicios de la computación se abordaban estos problemas y se desarrollaron algoritmos con el fin de imitar el proceso cognitivo humano; pero por varias situaciones tales como el limitado poder de las computadoras de ese momento y la falta de información, siendo que esta era limitada por lo que era muy complejo validar y entrenar los algoritmos(o modelos), haciendo que el desarrollo y estudio de la inteligencia artificial haya pasado por varios inviernos (así es, no solo uno fueron varios; dejando en claro que han habido muchas limitaciones para su desarrollo). Algunos de los primeros algoritmos y modelos que se desarrollaron fueron los de clasificación y regresión, los cuales tienen como base un modelo matemático, el cual trataremos y explicaremos en este documento.

Se desarrollaron 4 modelos basados en regresión lineal, Random Forest Regressor, KNeighborsRegressor, Linear Regressor y MLPRegressor (Multi-layer Perceptron Regressor) de los cuales se entrenaron todos de manera supervisada y solo dos de manera no supervisada (en el documento se explicará qué es el entrenamiento supervisado y no supervisado), los cuales son KNeighborsRegressor y MLPRegressor.

También trabajamos con seis datasets diferentes, los cuales son bastante usados en el mundo de machine-learning, esto por la versatilidad y el conjunto de features o características que los conforman.

2. Desarrollo

Para hacer un código más práctico y versátil se decidió entrenar a todos los modelos en conjunto, simplemente cambiando los diferentes datasets; primero se explicarán en detalle el funcionamiento de cada modelo y el por que solo se ciertos modelos se pueden entrenar como no supervisados, pero primero la diferencia entre estos.

2.1 Modelo supervisado

Un modelo supervisado cumple con las características de que los datos con los que se van a entrenar se relacionan con clases que ya han sido previamente agrupadas con base en los features del dataset, generalmente sirven con valores discretos o binarios, los cuales nos indican si el dato de entrada pertenece o no a alguna clase.

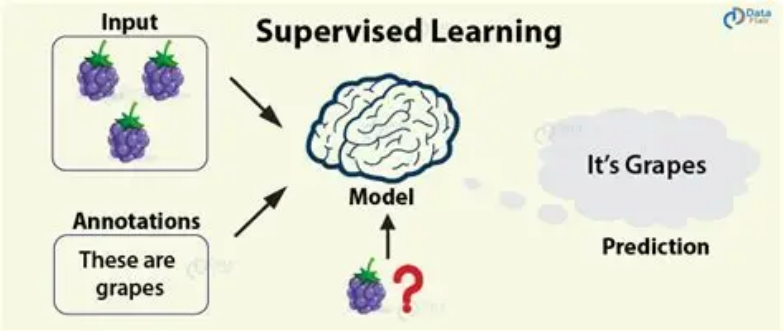

Imagen 1 ejemplo de modelo supervisado

2.2 Modelo no supervisado

La principal diferencia respecto al modelo supervisado es que este no tiene los datos previamente agrupados, haciendo que este se encargue también de clasificar los datos y agrupar a las nuevas entradas en las clases que ya se han creado previamente.

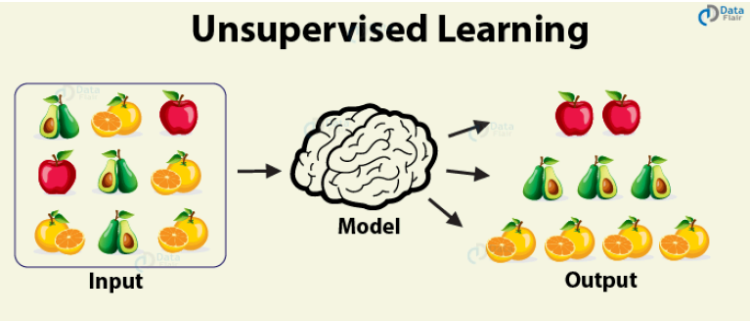

Imagen 2 ejemplo de modelo no supervisado

2.3 Linear Regressor

Linear regressor o regresión lineal es un algoritmo de aprendizaje automático-supervisado que aprende de los conjuntos de datos etiquetados y asigna los puntos de datos con las funciones lineales más optimizadas, lo que permite realizar predicciones en nuevos conjuntos de datos. Supone que existe una relación lineal entre la entrada y la salida, lo que significa que la salida cambia a un ritmo constante a medida que cambia la entrada. Esta relación se representa mediante una línea recta.

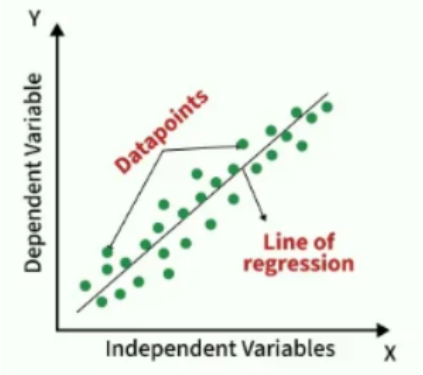

Imagen 3 funcionamiento de linear regressor

2.4 KNeighbors Regressor

Generalmente KNN se usa como un algoritmo de clasificación de clases, pero también es posible usarlo como un algoritmo de regresión para valores continuos, la idea es predecir el conjunto al que pertenece un nuevo conjunto de datos de entrada, esto promediando con las K vecinos más cercanos en el espacio de características

La distancia entre los puntos de datos se mide típicamente mediante la distancia euclidiana, aunque se pueden utilizar otras métricas de distancia.

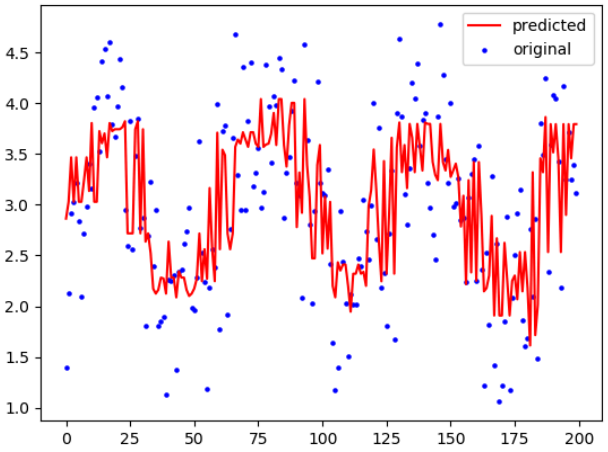

Imagen 4 ejemplo de KNN

Como ya se mencionó, KNN puede ser un algoritmo de clasificación, lo que nos da entrada para poder hacer un modelo no supervisado.

2.5 Random Forest Regressor

Es un algoritmo de aprendizaje que trabaja en conjunto combinando las predicciones de múltiples árboles de decisiones, esto para generar un resultado más preciso y estable. Al momento de entrenar el modelo, se divide el dataset original y se reparte de manera aleatoria entre todos los miembros del bosque, una vez se entrenaron los árboles, cada uno hace una predicción final; la predicción final para las tareas de regresión es el promedio de todas las predicciones de los árboles individuales y este proceso se denomina Agregación.

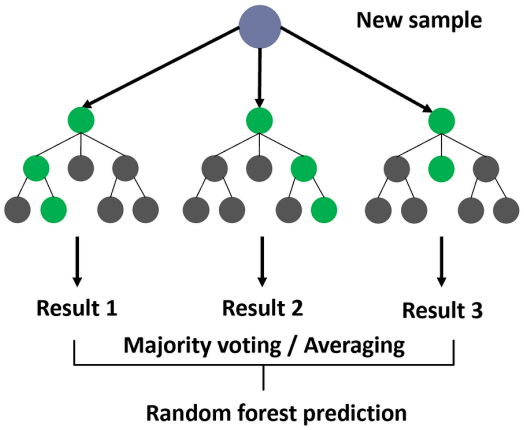

Imagen 5 ejemplo de Random Forest

2.6 MLP Regressor

Como sabemos, los MLP constan de múltiples capas de nodos interconectados, o neuronas. Cada neurona de una capa recibe la entrada de la capa anterior, aplica una función de transformación a la entrada y pasa la salida a la siguiente. Básicamente, existen tres capas diferentes: la capa de entrada, la capa oculta y la capa de salida. Las capas ocultas pueden ser de cualquier tamaño.

El MLP de clasificación y el MLP de regresión no son muy diferentes, pero, por supuesto, presentan algunas diferencias. En primer lugar, si se desea predecir un único valor (ingresos generados por publicidad), solo se necesita una neurona de salida; si se desean predecir varios valores, se pueden agregar varias neuronas de salida.

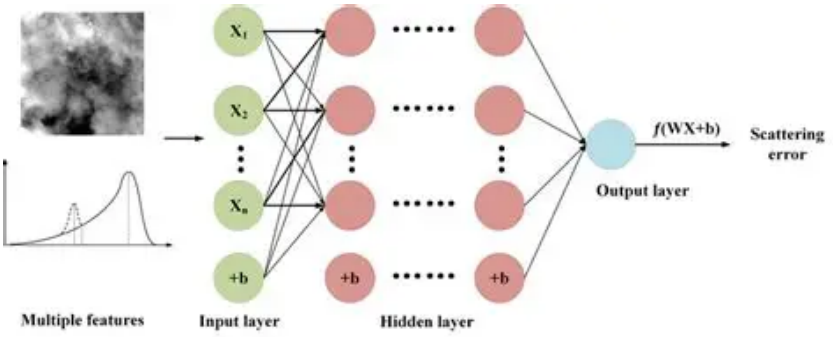

Imagen 6 funcionamiento de MLP

2.7 Datasets a utilizar

Los datasets a utilizar son: iris dataset, boston house, diabetes dataset, wine quality, car price y concrete compressive; los dataset son fáciles de trabajar debido a los features con las que cuentan; también debido a que son bastante comunes en el mundo de machine learning, por lo que podemos comparar las métricas de rendimiento para poder saber que tan eficiente y preciso es el modelo.

2.8 Funcionamiento del código

El código se divide en varias partes, desde importar las librerías, hasta el entrenamiento de los modelos.

2.8.1 Importación de librerías

Primero importamos todas las librerías, desde los modelos para entrenar, los datasets (solo los que estén disponibles en la librería), matplot para poder hacer las gráficas, y numpy para poder hacer los cálculos de las métricas que nos dirán qué tan acertados son los modelos y por último pandas para poder leer y dividir el dataset.

```python
#importamos las libreriass y los dataset(al menos los que esten disponibles en sklearn)
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import load_iris, load_diabetes, load_wine
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Imagen 7 importación de librerías

2.8.2 Segmentación de datos

Segmentos los datos y dividimos la información en variables X, y; en caso de que el dataset solo cargamos y asignamos las variables respectivamente, en caso de que no estén los dataset los importamos manualmente y dividimos de la misma manera la información(menos el dataset de car_price debido a que es una fuente externa los datos no están normalizados).

```python
#iris
iris = load_iris()
X_iris ,y_iris = iris.data, iris.target

#diabetes
diabetes = load_diabetes()
X_diabetes , y_diabetes = diabetes.data, diabetes.target

#wine
wine = load_wine()
X_wine, y_wine = wine.data , wine.target

# Boston Housing
url_boston = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
boston = pd.read_csv(url_boston)
X_boston = boston.drop(columns=["medv"])
y_boston = boston["medv"]

#car price
df_car = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vTnbmOG6vWzL8Nr34mXBPrMuNTYvRIP-Q3b9-1Xmfi70ZaXCJXLnJEupH0sGe4pkpDXgOHovSzKg5NV/pub?output=csv')
categorical_cols = df_car.select_dtypes(include=['object']).columns
df_car_encoded = pd.get_dummies(df_car, columns=categorical_cols, drop_first=True)
X_car = df_car_encoded.drop(columns=["Price"])
y_car = df_car_encoded["Price"]

#concrete data
df_concrete = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vRfvjStVFNK-8P3oDmxtq8eSjVoNVxcvrwrBRB8KvkucfAubsx5tSygFb_4WuDoo8x53VYx8KL94CI8/pub?output=csv')
X_concrete = df_concrete.drop(columns=["Strength"])
y_concrete = df_concrete["Strength"]
```

Imagen 8 segmentación de datos

2.8.3 División del dataset

En esta parte del código dividimos el dataset y lo fraccionamos en 80% para entrenar el modelo y 20% para evaluar qué tan preciso es, la literatura indica que el valor puede ser de una proporción de 70%-30% hasta 80%-20%.

```
X_train_iris , X_test_iris, y_train_iris, y_test_iris = train_test_split(X_iris,y_iris,test_size=.2,random_state=42)
X_train_boston , X_test_boston, y_train_boston, y_test_boston = train_test_split(X_boston,y_boston,test_size=.2,random_state=42)
X_train_diabetes , X_test_diabetes, y_train_diabetes, y_test_diabetes = train_test_split(X_diabetes,y_diabetes,test_size=.2,random_state=42)
X_train_wine , X_test_wine, y_train_wine, y_test_wine = train_test_split(X_wine,y_wine,test_size=.2,random_state=42)
X_train_car , X_test_car, y_train_car, y_test_car = train_test_split(X_car,y_car,test_size=.2,random_state=42)
X_train_concrete , X_test_concrete, y_train_concrete, y_test_concrete = train_test_split(X_concrete,y_concrete,test_size=.2,random_state=42)
```

Imagen 9 fragmentación de datos

2.8.4 Entrenamiento de modelos supervisados

2.8.5 Diccionario de modelos

Primero creamos un diccionario con el nombre del modelo y los features esperados, aunque parezcan pocos realmente recibe la mayoría de parámetros de manera default o por defecto, también se crea otro diccionario para guardar los resultados y evaluar el rendimiento de cada modelo.

```
def modelos_supervisados(X_train, X_test,y_train,y_test,data_info):
  modelos = {
    "RandomForestRegressor": RandomForestRegressor(random_state=42),
    "KNeighborsRegressor": KNeighborsRegressor(),
    "LinearRegression": LinearRegression(),
    "MLPRegressor": MLPRegressor(max_iter=1000,random_state=42)
  }

  rest = {"Modelo": [], "mse": [], "R2": []}
```

Imagen 10 diccionario de modelos

2.8.6 Entrenamiento de modelos

Con un for recorremos el diccionario y enviamos los parámetros para entrenar cada uno de los modelos(dependiendo el modelo se agregan los diferentes parámetros), una vez entrenado se evalúan con MSE y R^2.

```
for nombre, modelo in modelos.items():
  modelo.fit(X_train, y_train)
  y_pred = modelo.predict(X_test)
  mse = mean_squared_error(y_test, y_pred)
  r2 = r2_score(y_test, y_pred)

  rest["Modelo"].append(nombre)
  rest["mse"].append(mse)
  rest["R2"].append(r2)

resultados_df = pd.DataFrame(rest)
```

Imagen 11 entrenamiento de modelos

2.8.7 Comparación de modelos

En el array de resultados comparamos los resultados de cada uno de los modelos, esto con el fin de comparar las métricas individuales y entender que depende el contexto o el problema que estemos tratando vamos a tener que seleccionar alguno de los modelos, o simplemente podemos usar varios como en este caso para poder comparar cual es el que mejor se acopla a nuestras necesidades.

```python
plt.style.use('seaborn-v0_8-whitegrid')
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
axes[0].bar(resultados_df['Modelo'], resultados_df['mse'], color='blue')
axes[0].set_title(f'Error Cuadrático Medio (MSE) para el dataset {data_info}')
axes[0].set_ylabel('MSE')
axes[0].tick_params(axis='x', rotation=45)

axes[1].bar(resultados_df['Modelo'], resultados_df['R2'], color='red')
axes[1].set_title(f'Coeficiente de Determinación (R²) para el dataset {data_info}')
axes[1].set_ylabel('R² Score')
axes[1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

Imagen 12 comparación y graficación de modelos

2.8.8 Resultados de los modelos supervisados

Se mostrarán dos ejemplos de dataset comparando los 4 modelos de aprendizaje, tomando como criterio el MSE y R^2.

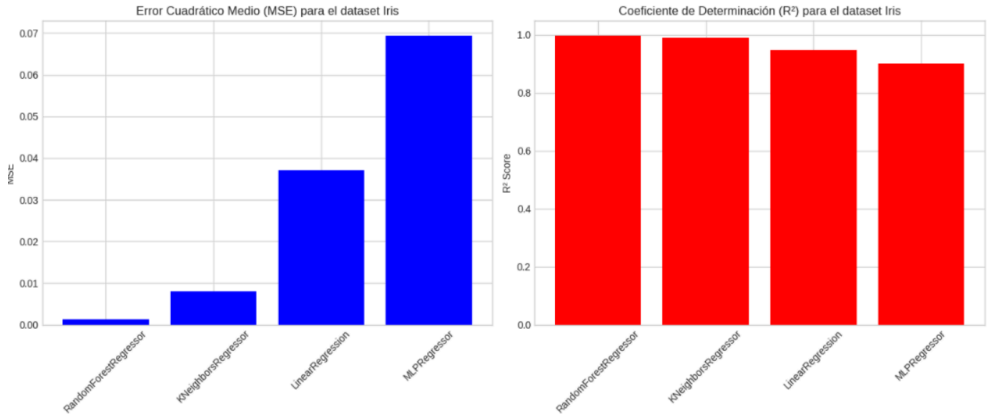

Imagen 13 comparación de modelos con el iris dataset

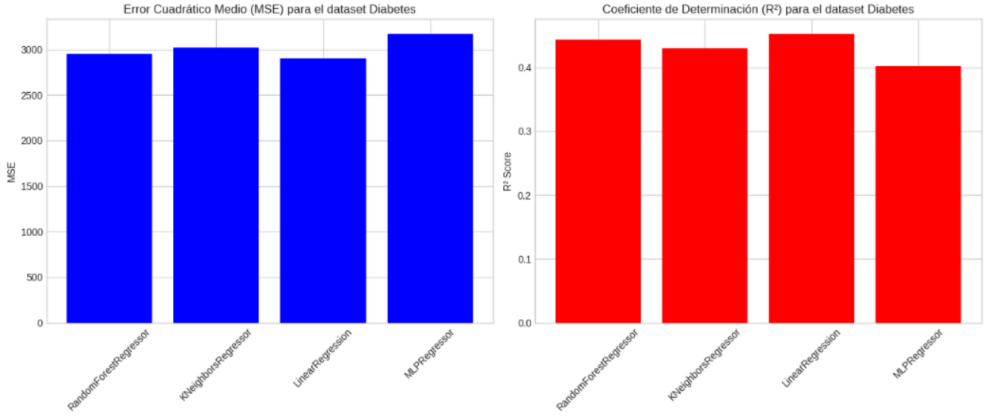

Imagen 14 comparación de modelos con el diabetes dataset

2.8.9 Entrenamiento modelos no supervisados

Debido a la naturaleza de los mismos modelos, nos percatamos de que estos no se pueden entrenar de manera no supervisada, por lo que entrenamos a Kmeans (una variante de KNN) para ejemplificar cómo es el entrenamiento de un modelo no supervisado.

Esto es algo completamente intuitivo, ya que en la documentación de scikit-learn se menciona que estos modelos son supervisados y si o si necesitan tener una entrada con clasificación para poder funcionar de manera correcta; aunado a esto, si vemos la forma en

que se extrae la información de un dataset para un modelo no supervisado es completamente diferente como se muestra en las siguientes imágenes.

```
# Boston Housing
url_boston = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
boston = pd.read_csv(url_boston)
X_boston = boston.drop(columns=["medv"])
y_boston = boston["medv"]
```

Imagen 13 comparación de datasets (supervisado)

```
url_boston = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
boston = pd.read_csv(url_boston)
X_no_supervisado = boston
```

Imagen 14 comparación de datasets (no supervisado)

Como se observa, para trabajar con un dataset supervisado es necesario clasificar los datos para que los modelos puedan trabajar correctamente, mientras que los no supervisados no, simplemente se toman todas las variables en un solo vector y el propio modelo se encarga de clasificar.

Conclusiones

En esta práctica, exploramos la aplicación de varios modelos de regresión en el ámbito del aprendizaje automático, abordando tanto los modelos supervisados como una incursión en la ejemplificación de los no supervisados. Como se detalló en la introducción, la capacidad de predecir y clasificar es intrínseca a la inteligencia humana y, de manera similar, fundamental para la inteligencia artificial. A través de este trabajo, hemos podido observar cómo algoritmos como Linear Regressor, KNeighbors Regressor, Random Forest Regressor y MLP Regressor buscan emular estas capacidades cognitivas, adaptándose a diferentes tipos de datos y problemas.

El desarrollo de la práctica se centró en la implementación y evaluación de estos modelos utilizando una variedad de datasets reconocidos, como iris, diabetes, wine quality, boston house, car price y concrete compressive. La elección de estos conjuntos de datos no fue arbitraria, sino que respondió a su versatilidad y a la riqueza de sus características, lo que nos permitió realizar comparaciones de rendimiento significativas. La metodología implementada incluyó la segmentación de datos, la división en conjuntos de entrenamiento y prueba (80%-20%), y el entrenamiento sistemático de cada modelo.

Un punto crucial de aprendizaje fue la distinción entre el aprendizaje supervisado y no supervisado. Si bien se intentó aplicar todos los modelos de regresión de manera no supervisada, se confirmó que su diseño inherente y la documentación de Scikit-learn los orientan hacia un entrenamiento supervisado, donde se requiere de datos previamente etiquetados para su correcto funcionamiento. Para ilustrar el concepto de aprendizaje no supervisado, se recurrió a un modelo como K-Means, que evidenció la diferencia fundamental en la preparación y el enfoque de los datos para este tipo de algoritmos. La representación visual de la segmentación de datos para ambos enfoques reforzó esta comprensión.

Los resultados obtenidos, presentados a través de métricas como MSE y R^2, destacaron que el rendimiento de cada modelo varía considerablemente dependiendo del dataset utilizado. No existe un modelo universalmente "mejor", sino que la eficacia de cada uno está intrínsecamente ligada a las características específicas de los datos y al problema que se intenta resolver. La comparación detallada de los modelos con los datasets de Iris y Diabetes, por ejemplo, mostró cómo ciertos algoritmos pueden sobresalir en la identificación de patrones y la realización de predicciones en unos contextos más que en otros.

En retrospectiva, esta práctica no solo ha consolidado nuestra comprensión de los modelos de regresión, sino que también ha subrayado la importancia de una selección y configuración cuidadosas de los algoritmos de machine learning. La flexibilidad del código desarrollado, permitiendo el entrenamiento conjunto de múltiples modelos con distintos datasets, es un testimonio de la versatilidad necesaria en este campo. La inteligencia artificial continúa avanzando, y la comprensión de estos modelos fundamentales es un paso esencial para abordar problemas cada vez más complejos y construir sistemas inteligentes más robustos y adaptativos.

Referencias

Mucci, T. (2025, febrero 25). La historia de la inteligencia artificial. Ibm.com.

https://www.ibm.com/mx-es/think/topics/history-of-artificial-intelligence

User guide. (s/f). Scikit-Learn. Recuperado el 20 de septiembre de 2025, de

https://scikit-learn.org/stable/user_guide.html

Linear regression in machine learning. (2018, septiembre 13). GeeksforGeeks.

https://www.geeksforgeeks.org/machine-learning/ml-linear-regression/

K-nearest neighbors (KNN) regression with scikit-learn. (2024, junio 17).

GeeksforGeeks.

https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbors-knn-regression-with-scikit-learn/

Sidharth, G. N. (2023, mayo 10). Regression with Multilayer Perceptron(MLP) Using Python. Quark Machine Learning.

https://www.quarkml.com/2023/05/regression-with-multilayer-perceptron-using-python.html

Sklearn: unsupervised knn vs k-means. (2018, 6 julio). Data Science Stack Exchange.

https://datascience.stackexchange.com/questions/34073/sklearn-unsupervised-knn-s-k-means