Regresión Lineal

Estudiantes:

- Daniel Alejandro Albarracín Vargas
- Juan Sebastián Garzón Gómez
- Nicolás López Sánchez
- Lina Mariana Pinzón Pinzón

CADI: Profundización II (Machine Learning)

Link Repositorio GitHub: https://github.com/lopezns/MachineLearning

Ejercicio 6

> **6. Sports - Relationship between training duration and performance in races (time in minutes)**
>
Training Duration (hours)	Race Time (minutes)
1	32
4	24
2	28
6	20
3	26
5	22
7	18
8	15
9	14
3.5	27
2.5	30
7.5	16
6.5	19
4.5	25
5.5	23
1.5	31

Training Duration (hours)	Race Time (minutes)
8.5	14
9.5	13
2.2	29
5.2	21
3.8	28
7.2	17
4.8	24
6.8	22
9.8	12

Tabla de Contenido

1. *Introducción*

El objetivo de este informe es analizar la relación entre la duración del entrenamiento y el rendimiento en carreras, utilizando técnicas de regresión lineal. En particular, se busca evaluar cómo los cambios en la duración del entrenamiento pueden afectar el tiempo de carrera, que se mide en minutos. Este análisis se realiza a través de un modelo de regresión lineal, el cual permite entender y predecir el impacto de diferentes duraciones de entrenamiento en el rendimiento de los atletas. El informe incluye una explicación detallada del código utilizado para generar los datos, entrenar el modelo y evaluar los resultados.

2. *Descripción del Ejercicio*

El ejercicio consiste en utilizar un modelo de regresión lineal para explorar la relación entre la duración del entrenamiento y el tiempo en carreras. Se comienza con la importación de bibliotecas esenciales para el manejo de datos y la creación de gráficos, como NumPy, Matplotlib y scikit-learn. Se generan datos originales y aleatorios sobre duraciones de entrenamiento y tiempos de carrera, y se combinan para formar un conjunto de datos más amplio. A continuación, se entrena un modelo de regresión lineal utilizando una parte de los datos y se evalúa su rendimiento con el conjunto de prueba. Finalmente, se visualizan los resultados mediante gráficos que comparan los datos reales con las predicciones del modelo.

3. *Desarrollo y Resultados*

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

- *import numpy as np:* Se importa la biblioteca NumPy, que es muy usada en Python para trabajar con arreglos y realizar operaciones matemáticas. Aquí, se importa con el alias np para facilitar su uso.
- *import matplotlib.pyplot as plt:* Se importa Matplotlib (específicamente el módulo pyplot) para crear gráficos. plt es el alias que se usa comúnmente para referirse a este módulo.
- *from sklearn.model_selection import train_test_split:* Importa la función train_test_split del módulo sklearn.model_selection. Esta función divide los datos en dos partes: conjunto de entrenamiento y conjunto de prueba. Se utiliza para entrenar un modelo con una parte de los datos y probarlo con la otra parte.
- *from sklearn.linear_model import LinearRegression:* Importa la clase LinearRegression de sklearn.linear_model, que permite construir un modelo de regresión lineal. Este modelo se ajusta para predecir una variable objetivo en función de una o más variables independientes.

```
training_duration_orig = np.array([60, 240, 120, 360, 180, 300, 420, 480,
540, 210, 150, 450, 390, 270, 330, 90, 510, 570, 132, 312, 228, 432, 288,
408, 588])
race_time_orig = np.array([32, 24, 28, 20, 26, 22, 18, 15, 14, 27, 30, 16,
19, 25, 23, 31, 14, 13, 29, 21, 28, 17, 24, 22, 12])
```

- *training_duration_orig:* Se crea un arreglo de NumPy que contiene las duraciones de entrenamiento de varios atletas en minutos. Estos son los datos originales.
- *race_time_orig:* Otro arreglo de NumPy que contiene los tiempos de carrera (también en minutos) que corresponden a cada duración de entrenamiento en training_duration_orig.

```
def generate_random_data(n_samples=75, seed=42):
    np.random.seed(seed)
    training_duration = np.random.normal(loc=300, scale=150, size=n_samples)
    training_duration = np.clip(training_duration, 60, 600)
    coef = -0.03
    intercept = 40
    race_time = coef * training_duration + intercept + np.random.normal(scale=2,
size=n_samples)
    return training_duration, race_time
```

- *def generate_random_data(n_samples=75, seed=42):* Define una función que recibe dos argumentos opcionales: n_samples (número de muestras a generar, por defecto 75) y seed (semilla para generar datos aleatorios, que garantiza la reproducibilidad de los resultados).

- *np.random.seed(seed):* Establece una semilla para el generador de números aleatorios, lo que asegura que los datos generados sean los mismos cada vez que se ejecute el código.

- *training_duration = np.random.normal(loc=300, scale=150, size=n_samples)* Genera n_samples de datos aleatorios que siguen una distribución normal con:
  - Media (loc=300): El promedio de los tiempos de entrenamiento es 300 minutos.
  - Desviación estándar (scale=150): La dispersión de los datos alrededor de la media es de 150 minutos.

- *training_duration = np.clip(training_duration, 60, 600):* Usa la función clip para limitar los valores generados entre 60 y 600 minutos. Cualquier valor menor a 60 se reemplaza por 60, y cualquier valor mayor a 600 se reemplaza por 600.

- *coef = -0.03 y intercept = 40:* Define el coeficiente de regresión (pendiente de la recta) y el intercepto (donde la recta cruza el eje vertical).

- *race_time = coef training_duration + intercept + np.random.normal(scale=2, size=n_samples):* Calcula el tiempo de carrera utilizando una fórmula lineal simple con un pequeño ruido aleatorio agregado (distribución normal con desviación estándar de 2). La relación es negativa (a medida que el entrenamiento aumenta, el tiempo de carrera disminuye).

- *return training_duration, race_time:* Devuelve las duraciones de entrenamiento generadas y los tiempos de carrera

```
training_duration_rand, race_time_rand = generate_random_data(n_samples=75)
```

- Llama a la función generate_random_data con 75 muestras para generar duraciones de entrenamiento y tiempos de carrera aleatorios. Los resultados se guardan en training_duration_rand y race_time_rand.

```
training_duration = np.concatenate((training_duration_orig,
training_duration_rand))
race_time = np.concatenate((race_time_orig, race_time_rand))
```

- *np.concatenate:* Combina los datos originales (training_duration_orig y race_time_orig) con los generados aleatoriamente (training_duration_rand y race_time_rand) para crear conjuntos de datos más grandes.

```
x = training_duration.reshape(-1, 1)
y = race_time
```

- *x = training_duration.reshape(-1, 1):* Reorganiza (redimensiona) el arreglo de duraciones de entrenamiento en un arreglo de 2 dimensiones con una columna (esto es necesario porque scikit-learn espera que las características tengan 2 dimensiones). El -1 permite que NumPy ajuste el número de filas automáticamente.

- *y = race_time:* Asigna los tiempos de carrera al vector y, que será la variable objetivo (dependiente) en la regresión.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)
```

- *train_test_split:* Divide los datos en dos subconjuntos:
  - *x_train y y_train:* El 80% de los datos que se usarán para entrenar el modelo.
  - *x_test y y_test:* El 20% restante se usará para probar el modelo.
- *test_size=0.2:* Indica que el 20% de los datos será usado para pruebas.
- *random_state=42:* Asegura que la división sea reproducible, es decir, cada vez que corras el código, obtendrás la misma división de datos.

```
model = LinearRegression()
model.fit(x_train, y_train)
```

- *model = LinearRegression():* Crea un modelo de regresión lineal usando la clase LinearRegression.
- *model.fit(x_train, y_train):* Ajusta (entrena) el modelo utilizando los datos de entrenamiento (x_train y y_train). El modelo aprende la relación entre las duraciones de entrenamiento y los tiempos de carrera.

```
y_pred = model.predict(x_test)
```

- *model.predict(x_test):* Usa el modelo entrenado para predecir los tiempos de carrera (y_pred) en el conjunto de prueba (x_test). Estas son las predicciones que luego compararemos con los valores reales.

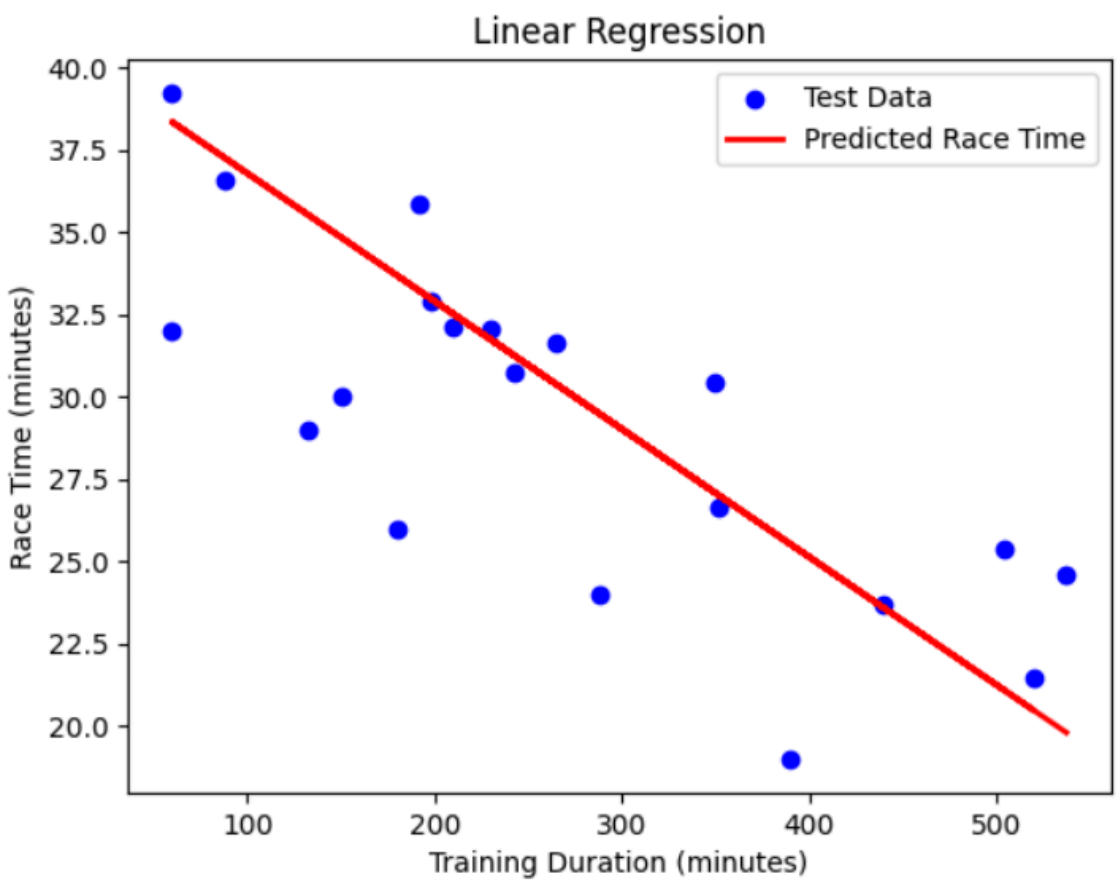
```
r2 = model.score(x_test, y_test)
print("R^2 Score:", r2)
coeficient = model.coef_[0]
print("Coeficient:", coeficient)
intercept = model.intercept_
print("Intercept:", intercept)
```

- *model.score(x_test, y_test):* Calcula el coeficiente de determinación R2R^2R2, que mide qué tan bien el modelo se ajusta a los datos de prueba. Un valor más cercano a 1 indica un mejor ajuste.
- *model.coef_[0]:* Devuelve el coeficiente de regresión (la pendiente de la recta ajustada).
- *model.intercept_:* Devuelve el intercepto del modelo, que es el valor de y cuando x es igual a 0.

```
plt.scatter(x_test, y_test, color='blue', label='Test Data')
plt.plot(x_test, y_pred, color='red', linestyle='-', linewidth=2,
label='Predicted Race Time')
plt.xlabel('Training Duration (minutes)')
plt.ylabel('Race Time (minutes)')
plt.title('Linear Regression')
plt.legend()
plt.show()
```

- *plt.scatter:* Crea un gráfico de dispersión (puntos) con los valores de prueba reales (x_test vs y_test), dibujando los puntos en azul.
- *plt.plot:* Dibuja una línea roja para mostrar las predicciones del modelo (x_test vs y_pred).
- *plt.xlabel y plt.ylabel:* Añade etiquetas a los ejes del gráfico.
- *plt.title:* Añade un título al gráfico.
- *plt.legend:* Muestra la leyenda del gráfico para distinguir los puntos reales de los predichos.
- *plt.show():* Muestra el gráfico final.

Gráfica



Explicación de la Gráfica

- *Relación entre la duración del entrenamiento y el tiempo de carrera:*
  La gráfica muestra una relación inversa: a medida que la duración del entrenamiento aumenta, el tiempo de carrera tiende a disminuir. Esto significa que cuanto más tiempo entrena una persona, más rápido puede correr la carrera (menos minutos).
  La pendiente negativa de la línea roja (que va descendiendo de izquierda a derecha) refleja esta relación inversa.

- *Variabilidad en los datos (puntos azules dispersos):*
  Los puntos azules no están perfectamente alineados con la línea roja, lo que es normal. Esto indica que, aunque existe una tendencia general, los resultados reales no son idénticos a las predicciones del modelo debido a la variabilidad natural (como fatiga, condiciones físicas, etc.) y el ruido que añadimos en el modelo.
  Esta dispersión también muestra que la regresión lineal no es un modelo perfecto para predecir el tiempo de carrera solo basado en la duración del entrenamiento, pero sí captura la tendencia principal.

- *Línea de regresión:*
  La línea roja representa lo que el modelo predice que debería ser el tiempo de carrera para una determinada cantidad de entrenamiento. Si la predicción es buena, la línea estará cerca de los puntos azules.
  Cuanto más cercanos estén los puntos azules a la línea roja, mejor será la precisión del modelo.

---

4. **Conclusiones**

Los resultados del análisis muestran que la duración del entrenamiento tiene un impacto significativo en el tiempo de carrera. El modelo de regresión lineal ajustado indica una relación negativa entre estas variables, lo que significa que a medida que aumenta la duración del entrenamiento, el tiempo de carrera tiende a disminuir. El coeficiente de regresión obtenido sugiere una disminución constante en el tiempo de carrera por cada minuto adicional de entrenamiento, mientras que el intercepto proporciona el tiempo de carrera base sin entrenamiento. El coeficiente de determinación (R^2) indica que el modelo se ajusta razonablemente bien a los datos, confirmando que la duración del entrenamiento puede ser un factor predictivo útil en la mejora del rendimiento en carreras.