# Regresión lineal simple

lab1.py

```python
# Importar las bibliotecas necesarias
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

En este fragmento, se importan las bibliotecas necesarias para el análisis y la construcción del modelo de regresión lineal simple.

- numpy es utilizada para operaciones matemáticas, especialmente para manejar matrices y vectores.
- matplotlib.pyplot es utilizada para visualizar los datos y los resultados del modelo.
- train_test_split y LinearRegression son importadas desde sklearn.model_selection y sklearn.linear_model, respectivamente, para dividir los datos en conjuntos de entrenamiento y prueba y para construir el modelo de regresión lineal.

```python
# Generar datos de ejemplo
np.random.seed(0)
X = 2 * np.random.rand(100, 1)  # Tamaño de la vivienda (variable independiente)
y = 5 + 3 * X + np.random.randn(100, 1)  # Precio de la vivienda (variable dependiente)
```

En este fragmento, se generan datos de ejemplo para simular el precio de las viviendas en función de su tamaño.

Se utilizan funciones de NumPy para generar un conjunto de datos “X” que representa el tamaño de las viviendas y un conjunto de datos y que representa el precio de las viviendas. Se introduce ruido aleatorio en los datos de salida “y”.

```
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear un modelo de regresión lineal
model = LinearRegression()

# Entrenar el modelo
model.fit(X_train, y_train)
```

Aquí se dividen los datos generados en fragmento 2 en conjuntos de entrenamiento y prueba usando la función train_test_split de scikit-learn.

Los datos se dividen en X_train, X_test, y_train y y_test para su posterior entrenamiento y evaluación del modelo.

```
# Coeficientes del modelo
print("Intercepto (error cuadratico):", model.intercept_)
print("Coeficiente de determinación:", model.coef_[0])

# Hacer predicciones
y_pred = model.predict(X_test)
```

Se crea un objeto de modelo de regresión lineal utilizando LinearRegression.

Se entrena el modelo utilizando los datos de entrenamiento (“X_train” y “y_train”) utilizando el método “fit”.

```
# Visualización de los datos y la línea de regresión
plt.scatter(X_test, y_test, color='black')
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.xlabel('Tamaño de la vivienda')
plt.ylabel('Precio de la vivienda')
plt.title('Regresión Lineal Simple')
plt.show()
```

Se imprimen el intercepto y el coeficiente de determinación del modelo, que proporcionan información sobre el ajuste del modelo a los datos.

Se realizan predicciones sobre los datos de prueba (X_test) utilizando el método predict del modelo.

Se visualizan los datos de prueba (X_test y y_test) y las predicciones del modelo (y_pred) utilizando un gráfico de dispersión y una línea de regresión.

## Decisiones de Diseño:

- Uso de Bibliotecas Especializadas: Se utilizan bibliotecas especializadas como NumPy, Matplotlib y scikit-learn para aprovechar la funcionalidad optimizada y las implementaciones eficientes de algoritmos de aprendizaje automático.
- División de Datos: Se dividen los datos en conjuntos de entrenamiento y prueba para evaluar el rendimiento del modelo en datos no vistos. Esto ayuda a detectar el sobreajuste del modelo.
- Visualización de Resultados: Se utiliza Matplotlib para visualizar los datos y los resultados del modelo. La visualización es una parte importante del análisis de datos y ayuda a comprender mejor el comportamiento del modelo y la relación entre las variables.
- Comunicación Clara: Se imprimen métricas de evaluación del modelo (intercepto, coeficiente de determinación) para comunicar el rendimiento del modelo de manera clara y concisa.