

SISTEMAS EXPERTOS

REGRESIÓN LINEAL SIMPLE

Por Esteban Madriz y ChatGPT



DESCRIPCIÓN:

Construir un modelo de regresión lineal simple para predecir una variable dependiente (por ejemplo, precio vivienda) basándose en una variable independiente (por ejemplo, tamaño vivienda). Este ejemplo utiliza Python y la biblioteca scikit-learn, que es una biblioteca muy popular en el ámbito de ciencia de datos y machine learning



IMPORTACION DE LIBRERÍAS

VISUAL STUDIO CODE

```
lab1.py > ...  
1  # Importar las bibliotecas necesarias  
2  import numpy as np  
3  import matplotlib.pyplot as plt  
4  from sklearn.metrics import mean_squared_error, r2_score  
5  from sklearn.model_selection import train_test_split  
6  from sklearn.linear_model import LinearRegression  
7
```



EXPLICACIÓN

- **numpy** (alias np): Se utiliza para el manejo de arrays y operaciones matemáticas en Python, lo cual es fundamental para la manipulación de datos numéricos en el contexto de tu ejercicio.
- **matplotlib.pyplot** (alias plt): Es una biblioteca de visualización que te permite crear gráficos y visualizar los datos y los resultados de tu modelo de una manera fácil y eficiente.
- **train_test_split de sklearn.model_selection**: Esta función, proveniente del módulo model_selection de scikit-learn, te permite dividir tus datos en conjuntos de entrenamiento y prueba. Esto es crucial para evaluar el rendimiento de tu modelo en datos no vistos.
- **LinearRegression de sklearn.linear_model**: Es la clase que utilizaremos para crear nuestro modelo de regresión lineal en este ejercicio. Esta clase nos permite entrenar un modelo de regresión lineal y hacer predicciones con él.
- **mean_squared_error y r2_score de sklearn.metrics**: Estas funciones, pertenecientes al módulo metrics de scikit-learn, se utilizan para evaluar el rendimiento de nuestro modelo de regresión lineal. mean_squared_error calcula el error cuadrático medio entre las predicciones y los valores verdaderos, mientras que r2_score calcula el coeficiente de determinación, que es una medida de qué tan bien se ajustan las predicciones a los valores reales.






GENERACIÓN DE DATOS

- **`np.random.seed(0)`**: Establece la semilla para el generador de números aleatorios de NumPy, lo que garantiza que los resultados sean reproducibles.
- **`np.random.rand(100, 1)`**: Genera una matriz de tamaño (100, 1) con números aleatorios distribuidos uniformemente en el intervalo [0, 1), utilizados para representar la variable independiente X.
- **`np.random.randn(100, 1)`**: Genera una matriz de tamaño (100, 1) con números aleatorios distribuidos según una distribución normal estándar, que se usa para representar el ruido aleatorio en la variable dependiente y.

```
# Generar datos sintéticos para el ejemplo
np.random.seed(0)
X = 2 * np.random.rand(100, 1) # Variable independiente (tamaño de la vivienda)
y = 5 + 3 * X + np.random.randn(100, 1) # Variable dependiente (precio de la vivienda)
```

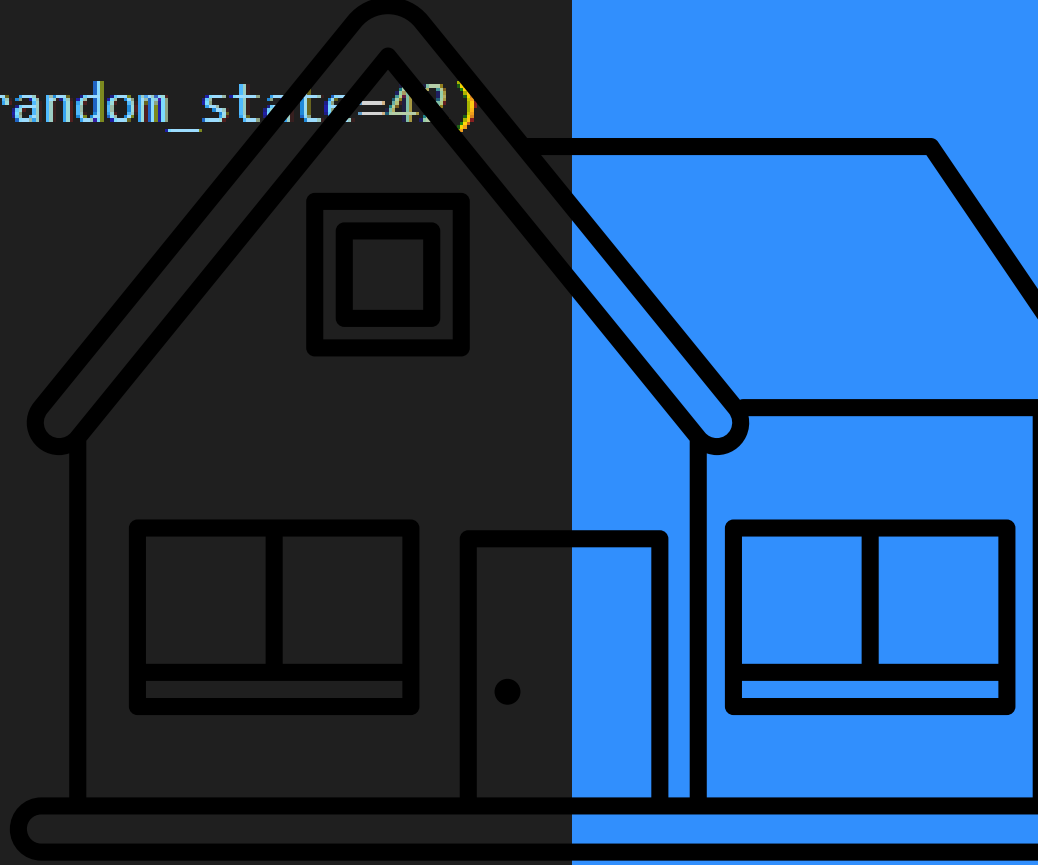


```
# Dividir los datos en conjunto de entrenamiento y conjunto de prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Entrenar el modelo con el conjunto de entrenamiento
model.fit(X_train, y_train)

# Hacer predicciones con el conjunto de prueba
y_pred = model.predict(X_test)
```

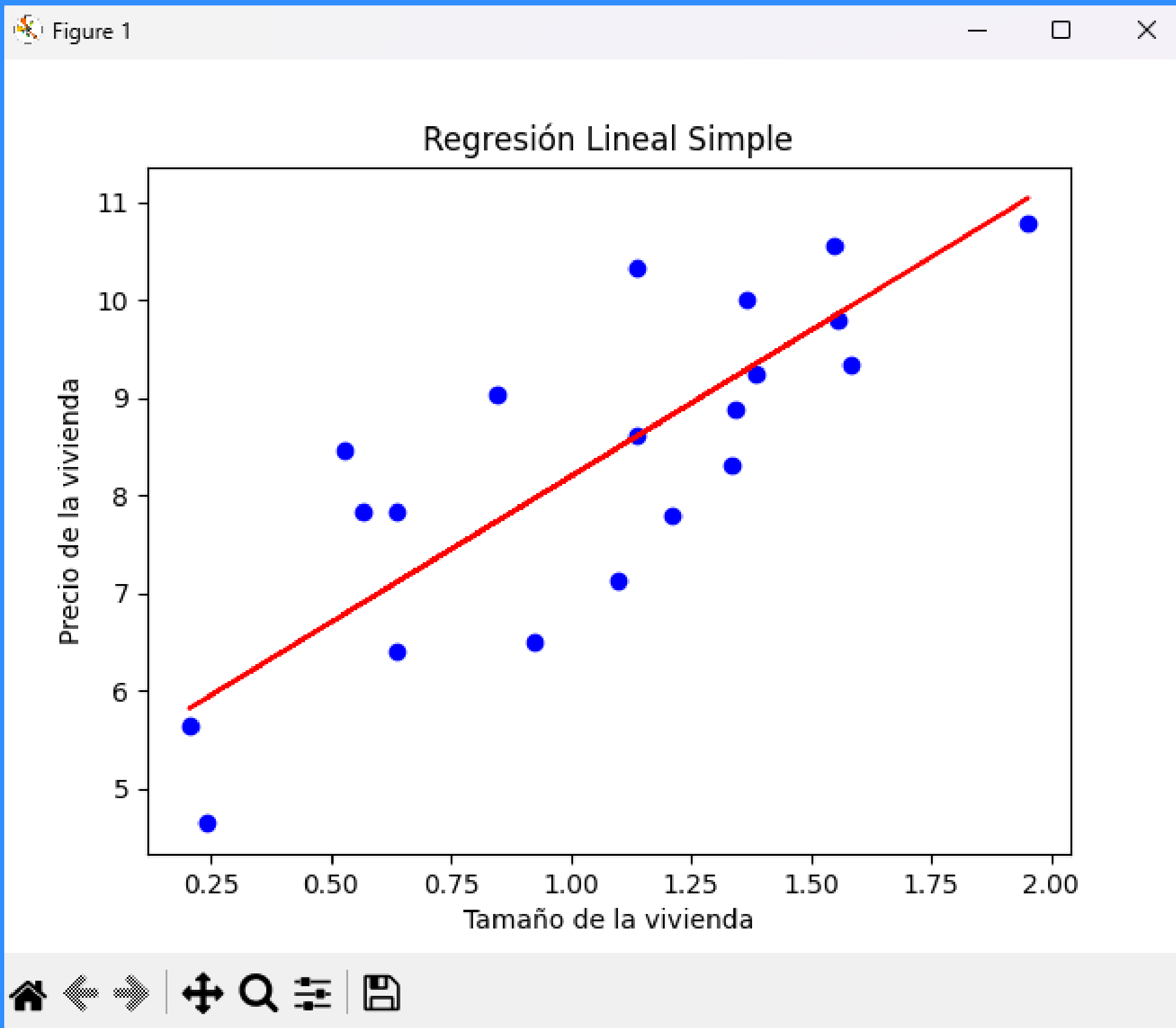


- En el primer código se dividen los datos en conjuntos de entrenamiento y prueba, donde `X_train` y `y_train` son los datos de entrenamiento y `X_test` y `y_test` son los datos de prueba.
- En el segundo se crea un modelo de regresión lineal utilizando la clase `LinearRegression` de `scikit-learn`.
- En el tercero se entrena el modelo utilizando los datos de entrenamiento (`X_train` y `y_train`) para ajustar los coeficientes de la regresión lineal.
- En el cuarto se realiza predicciones utilizando el modelo entrenado sobre los datos de prueba (`X_test`) y almacena las predicciones en `y_pred`.

```
print("MSE:", mean_squared_error(y_test, y_pred))  
print("R2:", r2_score(y_test, y_pred))  
  
# Visualizar los resultados  
plt.scatter(X_test, y_test, color='blue')  
plt.plot(X_test, y_pred, color='red')  
plt.title('Regresión Lineal Simple')  
plt.xlabel('Tamaño de la vivienda')  
plt.ylabel('Precio de la vivienda')  
plt.show()
```



- Las dos primeras líneas imprimen el error cuadrático medio (MSE) y el coeficiente de determinación (R^2) para evaluar el rendimiento del modelo. `mean_squared_error` calcula el MSE entre las etiquetas verdaderas (`y_test`) y las predicciones (`y_pred`), mientras que `r2_score` calcula el R^2 , que es una medida de qué tan bien se ajustan las predicciones a los valores reales.
- La siguiente sección visualiza los resultados del modelo. `plt.scatter` crea un diagrama de dispersión de los datos de prueba (`X_test` y `y_test`) en color azul. Luego, `plt.plot` traza la línea de regresión utilizando los datos de prueba (`X_test`) y las predicciones (`y_pred`) en color rojo. Los métodos `plt.title`, `plt.xlabel` y `plt.ylabel` agregan etiquetas al gráfico para indicar el título, el eje x (Tamaño de la vivienda) y el eje y (Precio de la vivienda), respectivamente.
- Finalmente, `plt.show()` muestra el gráfico completo con todas las visualizaciones realizadas.



Finalmente podemos ver como muestra datos de distintos precios de viviendas con respecto a su tamaño y por consola nos tira el MSE y R2



```
PS C:\Users\Esteb\OneDrive\Escritorio\Lab 1 S Expertos> python lab1.py
MSE: 0.9177532469714293
R2: 0.6521157503858555
```