Trabajo Práctico 2

Regresión del valor medio de casas en distritos de California

Abril Noguera - Pablo Brahim - Fermin Rodriguez - Kevin Pennington

Se requiere construir una regresión que nos permita predecir el valor medio de las casas en distritos de California, EEUU (medidos en cientos de miles de dólares $100,000). Este dataset se deriva del censo de 1990 de EEUU, donde cada observación es un bloque. Un bloque es la unidad geográfica más pequeña para la cual la Oficina del Censo de EEUU publica datos de muestra (un bloque típicamente tiene una población de 600 a 3000 personas).

Los atributos, en el orden en que se guardaron en el dataset, son:

- `MedInc`: Ingreso medio en el bloque
- `HouseAge`: Edad mediana de las casas en el bloque
- `AveRooms`: Número promedio de habitaciones por hogar.
- `AveBedrms`: Número promedio de dormitorios por hogar.
- `Population`: Población del bloque
- `AveOccup`: Número promedio de miembros por hogar.
- `Latitude`: Latitud del bloque
- `Longitude`: Longitud del bloque

Y el target es:

- `MedHouseVal`: Mediana del costo de casas en el bloque (en unidades de a $100.000)

Para este TP, se proporciona una notebook (`ayuda.ipynb`) con la lectura del dataset, la separación de los datos, entre otras ayudas para resolver este trabajo práctico.

In [142…

```
from sklearn.datasets import fetch_california_housing
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, FunctionTransformer
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_scor
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV, KFold
```

In [143…

```python
# Leemos el dataset
california_housing = fetch_california_housing()

# Y obtenemos los atributos y target
X = california_housing.data
y = california_housing.target

# Transformamos en Pandas
X = pd.DataFrame(X, columns=california_housing['feature_names'])
y = pd.Series(y, name=california_housing['target_names'][0])

# Unimos a X e y, esto ayuda a la parte de la gráfica del mapa de calor de c
df = pd.concat([X, y], axis=1)
```

Link al Repositorio

Adjunto el link al repositorio con la resulución completa. Repositorio de Github

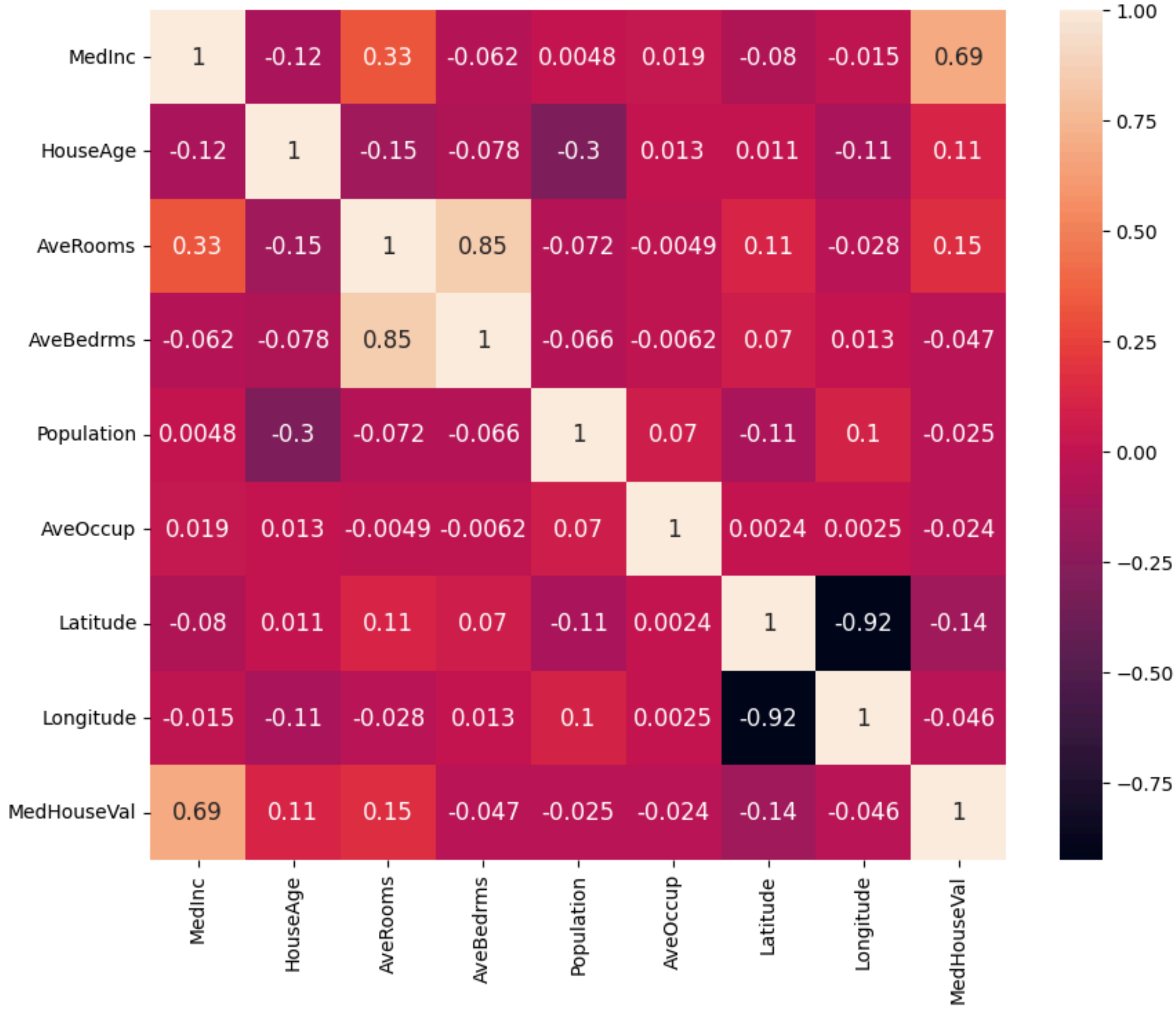
---

1. Obtener la correlación entre los atributos y los atributos con el target. ¿Cuál atributo tiene mayor correlación lineal con el target y cuáles atributos parecen estar más correlacionados entre sí? Se puede obtener los valores o directamente graficar usando un mapa de calor.

Correlaciones sobre la base completa

In [144…

```python
correlations = df.corr()
plt.figure(figsize=[10,8])
sns.heatmap(correlations, annot=True, annot_kws={'size':12})
plt.show()
```



El atributo con mayor correlación lineal con el target **es el ingreso medio del bloque (MeInc)** con un coeficiente de **0.69**. La correlación es bastante alta lo cual es esperable ya que, a mayor ingreso del bloque, mayor debe ser el costo medio de las propiedades.

Entre los atributos no se observa en general correlaciones muy grandes. El -0.92 es el mayor valor pero es la correlación entre las coordenadas de California, que no refleja una relación causal, sino que responde a la disposición diagonal del Estado. Otra correlación alta entre atributos es la de numero promedio de habitaciones y dormitorios. Finalmente, otro valor que resalta es el promedio de habitaciones e ingreso medio del bloque con un coeficiente de 0.33.

Correlaciones eliminando outliers

Se calculan nuevamente las correlaciones manuales excluyendo el 1% superior de valores atípicos.

In [145…
```
df.describe(percentiles=[0.01, 0.5, 0.99])
```

Out[145…

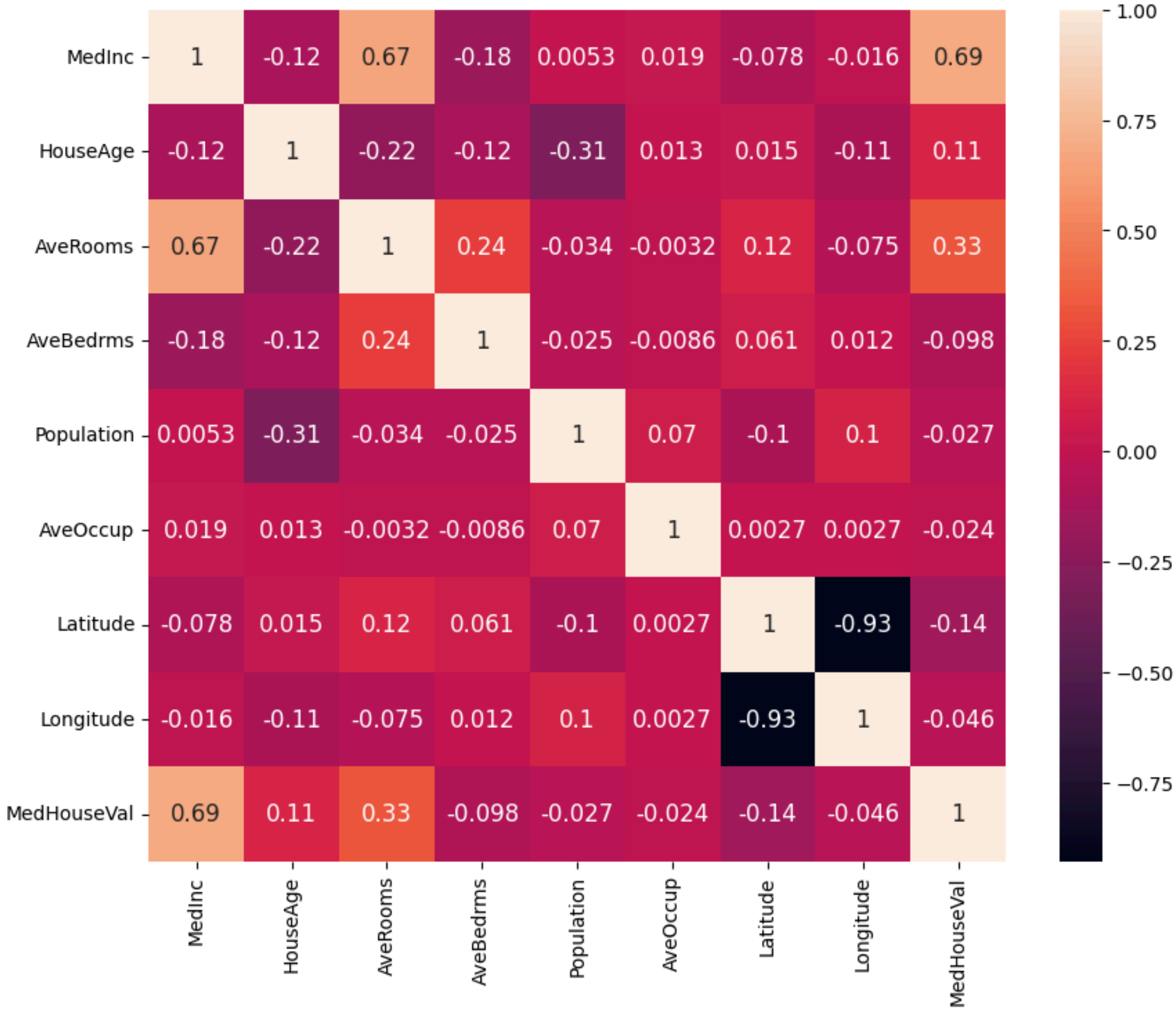
	MedInc	HouseAge	AveRooms	AveBedrms	Population
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744
std	1.899822	12.585558	2.474173	0.473911	1132.462122
min	0.499900	1.000000	0.846154	0.333333	3.000000
1%	1.069631	4.000000	2.581133	0.872840	88.000000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000
99%	10.596540	52.000000	10.357033	2.127541	5805.830000
max	15.000100	52.000000	141.909091	34.066667	35682.000000

Las variables **AveRooms, AveBedrms, AveOccup y Population** presentan valores máximos considerablemente más altos que su percentil 99, lo cual indica una fuerte presencia de valores atípicos extremos en estas distribuciones.

In [146…

```
df_aux = df.loc[ (df["AveRooms"] < df.quantile(0.99)["AveRooms"])]
plt.figure(figsize=[10,8])
correlations =df_aux.corr()
sns.heatmap(correlations, annot=True, annot_kws={'size':12})
print(f"Se excluyeron {df.shape[0] - df_aux.shape[0]} valores del data frame
```

```
Se excluyeron 207 valores del data frame
```

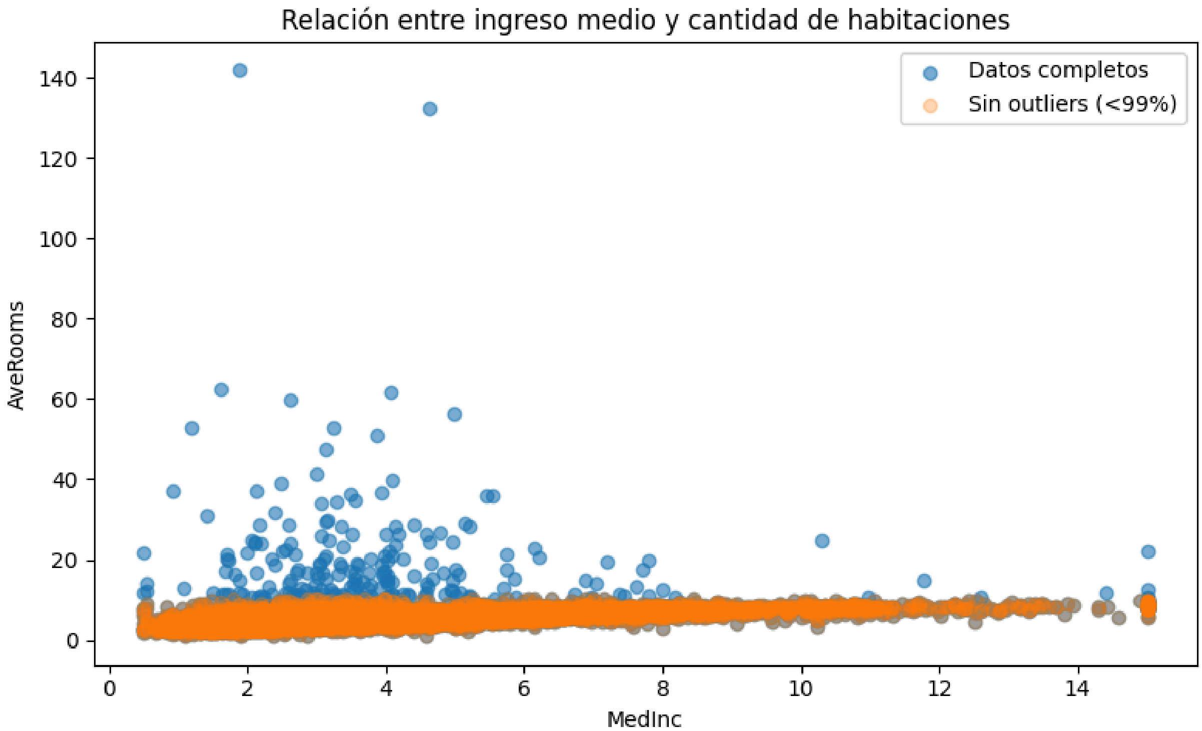


La correlación entre **AveRooms y MedInc** pasó de 0.33 a 0.67, lo cual indica que los valores atípicos estaban ocultando una relación positiva significativa entre el tamaño promedio de las viviendas y el ingreso medio del bloque.

La relación entre **AveRooms y AveBedrms** se redujo de 0.85 a 0.24, lo que indica que los valores extremos inflaban artificialmente esta relación. Al eliminarlos, se revela que la asociación entre ambas variables no es tan fuerte como parecía inicialmente.

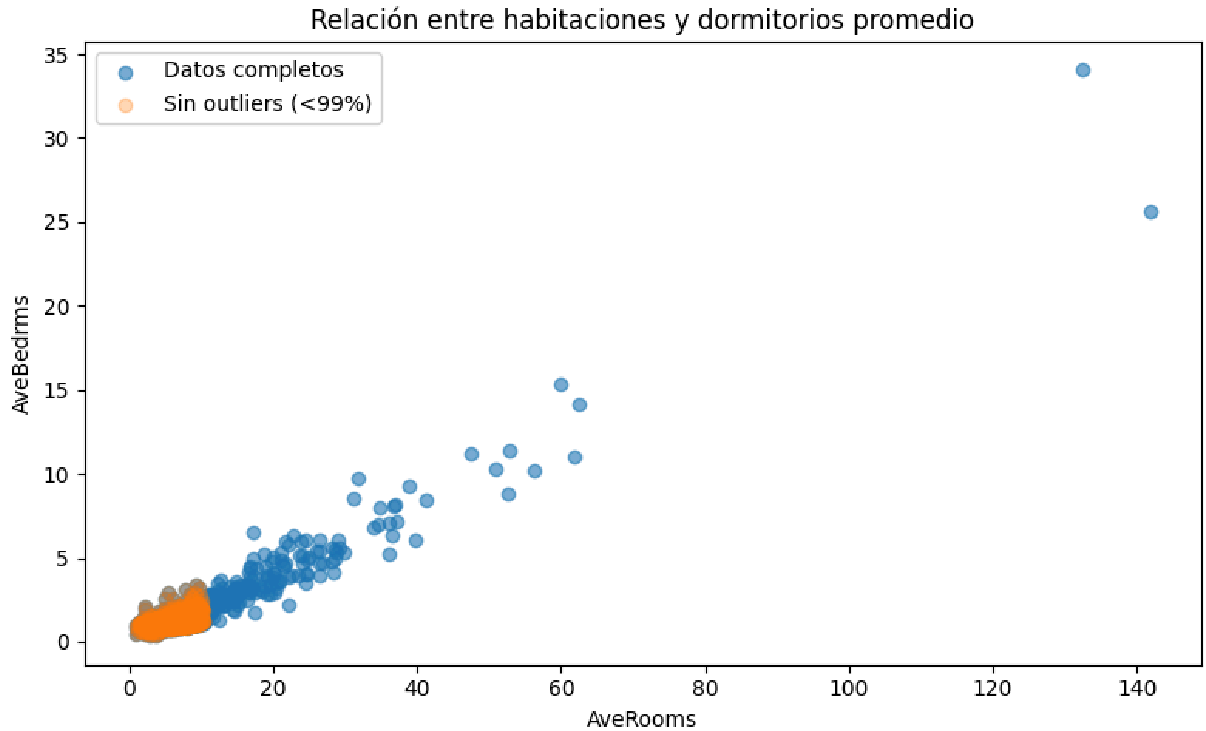
In [147…

```
plt.figure(figsize=(8, 5))
plt.scatter(df.MedInc, df.AveRooms, label="Datos completos", alpha=0.6)
plt.scatter(df_aux.MedInc, df_aux.AveRooms, alpha=0.3, label="Sin outliers (

plt.xlabel("MedInc")
plt.ylabel("AveRooms")
plt.title("Relación entre ingreso medio y cantidad de habitaciones")
plt.legend()
plt.grid(False)
plt.tight_layout()
plt.show()
```



In [148…

```
plt.figure(figsize=(8, 5))
plt.scatter(df.AveRooms, df.AveBedrms, label="Datos completos", alpha=0.6)
plt.scatter(df_aux.AveRooms, df_aux.AveBedrms, alpha=0.3, label="Sin outlier

plt.xlabel("AveRooms")
plt.ylabel("AveBedrms")
plt.title("Relación entre habitaciones y dormitorios promedio")
plt.legend()
plt.grid(False)
plt.tight_layout()
plt.show()
```



---

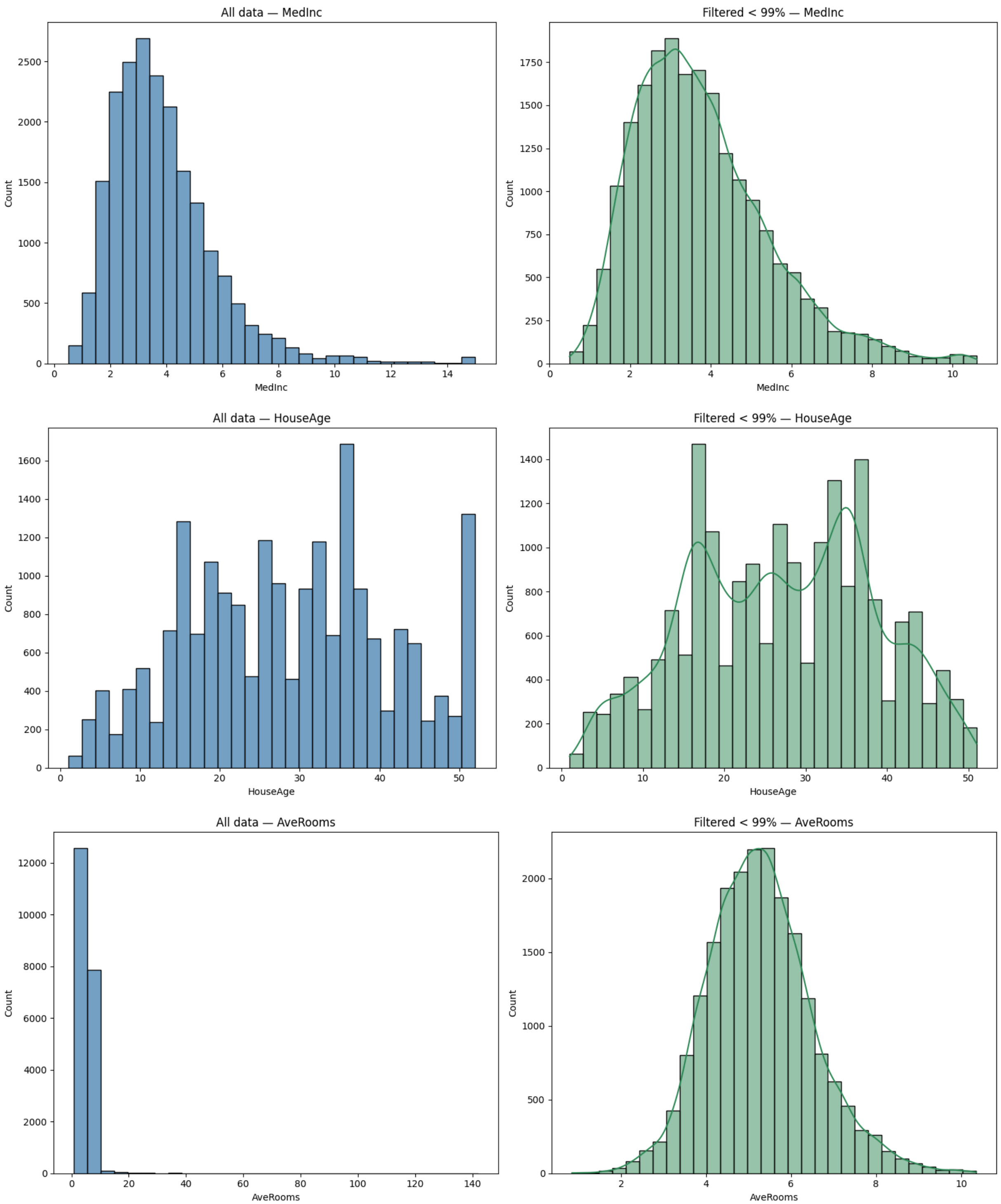
2. Graficar los histogramas de los diferentes atributos y el target. ¿Qué tipo de forma de histograma se observa? ¿Se observa alguna forma de campana que nos indique que los datos pueden provenir de una distribución gaussiana, sin entrar en pruebas de hipótesis?

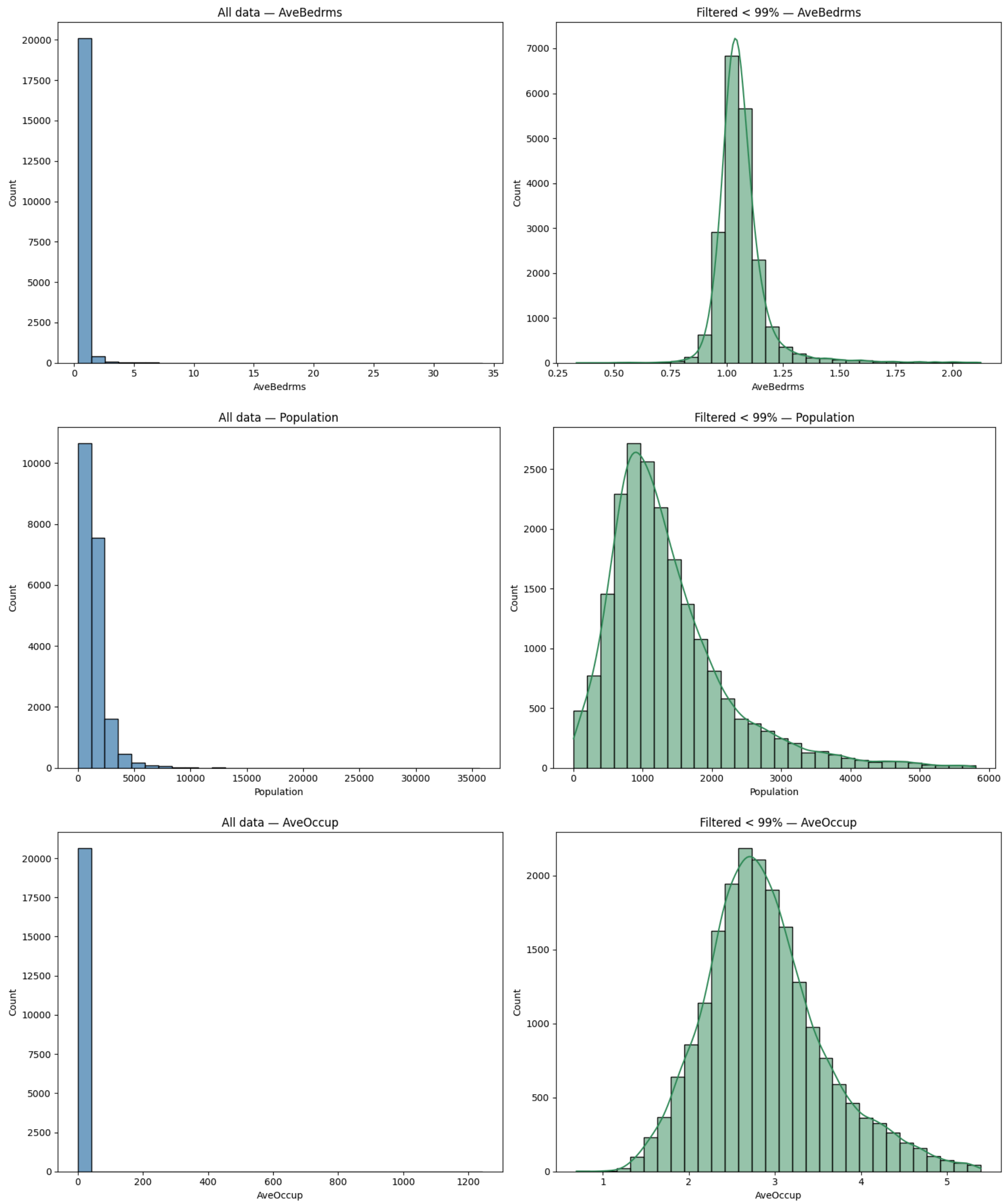
In [149…

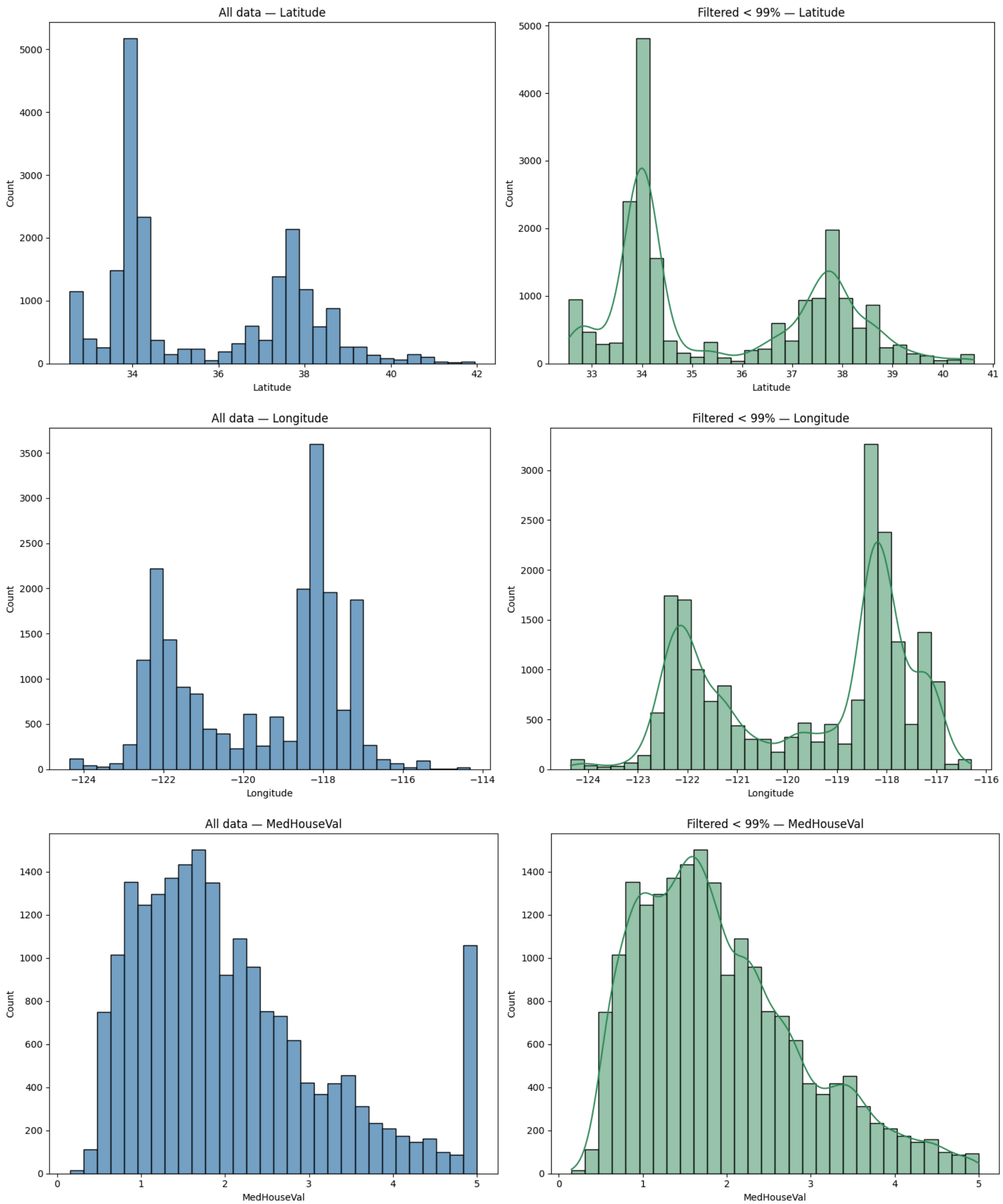
```
def plot_hist(df, column, quantile=0.99, bins=30):

    threshold = df[column].quantile(quantile)
    df_filtered = df[df[column] < threshold]

    fig, axes = plt.subplots(1, 2, figsize=(15, 6))

    sns.histplot(df[column], ax=axes[0], bins=bins, color="steelblue")
    axes[0].set_title(f"All data — {column}")

    sns.histplot(df_filtered[column], ax=axes[1], bins=bins, color="seagreen
    axes[1].set_title(f"Filtered < {quantile:.0%} — {column}")

    plt.tight_layout()
    plt.savefig("a.png")
    plt.show()
```

In [150…

```
for col in df.columns:
    plot_hist(df,col)
```


All data — MedInc
Count
MedInc
0
2
4
6
8
10
12
14
0
500
1000
1500
2000
2500
Filtered < 99% — MedInc
Count
MedInc
0
250
500
750
1000
1250
1500
1750
All data — HouseAge
Count
HouseAge
0
10
20
30
40
50
200
400
600
800
1000
1200
1400
1600
Filtered < 99% — HouseAge
Count
HouseAge
All data — AveRooms
Count
AveRooms
20
40
60
80
100
120
140
2000
4000
6000
8000
10000
12000
Filtered < 99% — AveRooms
Count
AveRooms


All data — AveBedrms
Count
0 2500 5000 7500 10000 12500 15000 17500 20000
0 5 10 15 20 25 30 35
AveBedrms
Filtered < 99% — AveBedrms
Count
0 1000 2000 3000 4000 5000 6000 7000
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
AveBedrms
All data — Population
Count
0 2000 4000 6000 8000 10000
0 5000 10000 15000 20000 25000 30000 35000
Population
Filtered < 99% — Population
Count
0 500 1000 1500 2000 2500
0 1000 2000 3000 4000 5000 6000
Population
All data — AveOccup
Count
0 2500 5000 7500 10000 12500 15000 17500 20000
0 200 400 600 800 1000 1200
AveOccup
Filtered < 99% — AveOccup
Count
0 500 1000 1500 2000
1 2 3 4 5
AveOccup

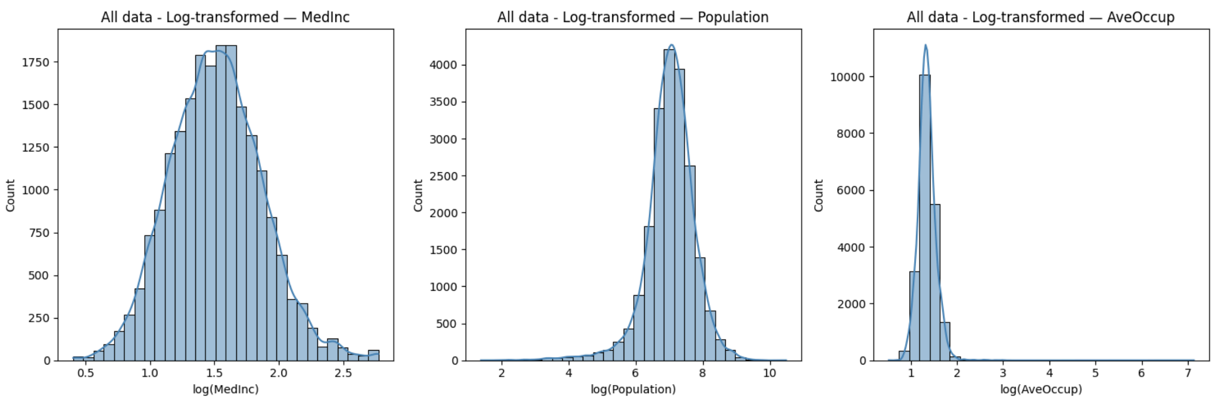


En general (excluyendo latitud y longitud) los datos presentan forma de campana, aunque no se ajustan perfectamente a una distribución normal. Esto se hace mas visible filtrando el 99% inferior de los datos.

Sin embargo, la mayoría de las variables exhiben una asimetría o skew hacia la derecha. **MediaInc, Population y AveOccup** son los atrubutos donde este comportamiento se hace mas evidente. Podrían seguir una distribución log-normal, lo que podria comprobarse aplicando el logaritmo y evaluando si es normal.

In [151…

```
df_log = df[["MedInc", "Population", "AveOccup"]].apply(lambda x: np.log(x +

# Graficamos histogramas de las variables transformadas
plt.figure(figsize=(15, 5))

for i, column in enumerate(df_log.columns):
    plt.subplot(1, 3, i + 1)
    sns.histplot(df_log[column], kde=True, bins=30, color='steelblue')
    plt.title(f'All data - Log-transformed — {column}')
    plt.xlabel(f'log({column})')
    plt.ylabel('Count')

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



Al transformar **MedInc** aplicando el logaritmo se presenta una distribución bastante simétrica con forma de campana. La variable ahora se asemeja mucho a una normal. En cambio las variables **Population y AveOccup** siguen presentando sesgos.

Antes del modelado, se eliminaron los outliers de la variable **AveRooms** (filtrando el 1% superior).

In [152…

```
mask = X["AveRooms"] < X["AveRooms"].quantile(0.99)
X_filtered = X[mask].copy()
y_filtered = y[mask].copy()
```

---

3. Calcular la regresión lineal usando todos los atributos. Con el set de entrenamiento, calcular la varianza total del modelo y la que es explicada con el modelo. ¿El modelo está capturando el comportamiento del target? Expanda su respuesta.

In [153…

```
# Dividimos el dataset en train y test
X_train, X_test, y_train, y_test = train_test_split(X_filtered,  y_filtered,
```

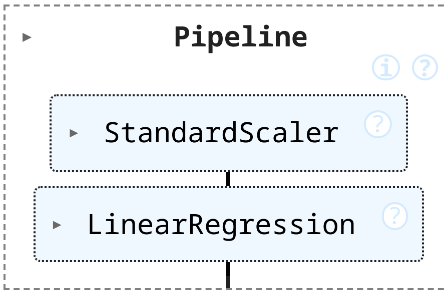
In [154…

```python
# Creamos el pipeline
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', LinearRegression())
])
```

In [155…

```python
# Entrenamos el modelo
pipe.fit(X_train, y_train)
```

Out[155…



In [156…

```python
# Predecir y evaluar
y_pred_train = pipe.predict(X_train)

# Calcular varianza total y explicada
var_total = np.var(y_train)
var_residual = np.var(y_train - y_pred_train)
var_explicada = var_total - var_residual
r2 = r2_score(y_train, y_pred_train) # porcentaje de varianza explicada

# Resultados
print("Resultados del modelo en Train:")
print(f"Varianza total del target: {var_total:.4f}")
print(f"Varianza explicada por el modelo: {var_explicada:.4f}")
print(f"R²: {r2:.2%}")
```

```
Resultados del modelo en Train:
Varianza total del target: 1.3254
Varianza explicada por el modelo: 0.8170
R²: 61.64%
```

El modelo está explicando un **58.71%** de la variabilidad del target, lo cual no es perfecto pero tampoco bajo. Esto quiere decir que, aunque el modelo logra reconocer ciertos patrones importantes en los datos, todavía hay una parte considerable que no está pudiendo explicar. Es decir, hay aspectos del comportamiento del target que el modelo no está captando del todo.

---

4. Calcular las métricas de MSE, MAE y R^2 del set de evaluación.

In [157…

```python
# 1. Predecir sobre el set de test
y_pred = pipe.predict(X_test)
```

```python
# 2. Calcular métricas
mse_reg = mean_squared_error(y_test, y_pred)
mae_reg = mean_absolute_error(y_test, y_pred)
r2_reg = r2_score(y_test, y_pred)

# 3. Mostrar resultados
print("Resultados del modelo en Test:")
print(f"MSE (Error cuadrático medio): {mse_reg:.4f}")
print(f"MAE (Error absoluto medio):  {mae_reg:.4f}")
print(f"R² (Varianza explicada):     {r2_reg:.2%}")
```

```
Resultados del modelo en Test:
MSE (Error cuadrático medio): 0.5186
MAE (Error absoluto medio):  0.5248
R² (Varianza explicada):     61.50%
```

El desempeño del modelo sobre el conjunto de test es coherente con lo observado en entrenamiento. El R^2 de **58.64%** indica que el modelo mantiene una capacidad de generalización razonable, aunque puede mejorar.

---

5. Crear una regresión de Ridge. Usando una validación cruzada de 5-folds y usando como métrica el MSE, calcular el mejor valor de α, buscando entre [0, 12.5]. Graficar el valor de MSE versus α.

In [158…

```python
# Definimos el rango de alphas
alphas = np.linspace(0, 12.5, 100)

# Creamos el pipeline
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', Ridge())
])

# Definimos los hiperparámetros a testear
param_grid = {'regressor__alpha': alphas}

# Validación cruzada
cv = KFold(n_splits=5, shuffle=True, random_state=42)

# GridSearch con scoring de MSE (negativo)
grid = GridSearchCV(pipe, param_grid, cv=cv, scoring='neg_mean_squared_error
grid.fit(X_train, y_train)
```

Out[158…

```
▸ GridSearchCV
▸ best_estimator_:
  Pipeline
  ▸ StandardScaler
  ▸ Ridge
```

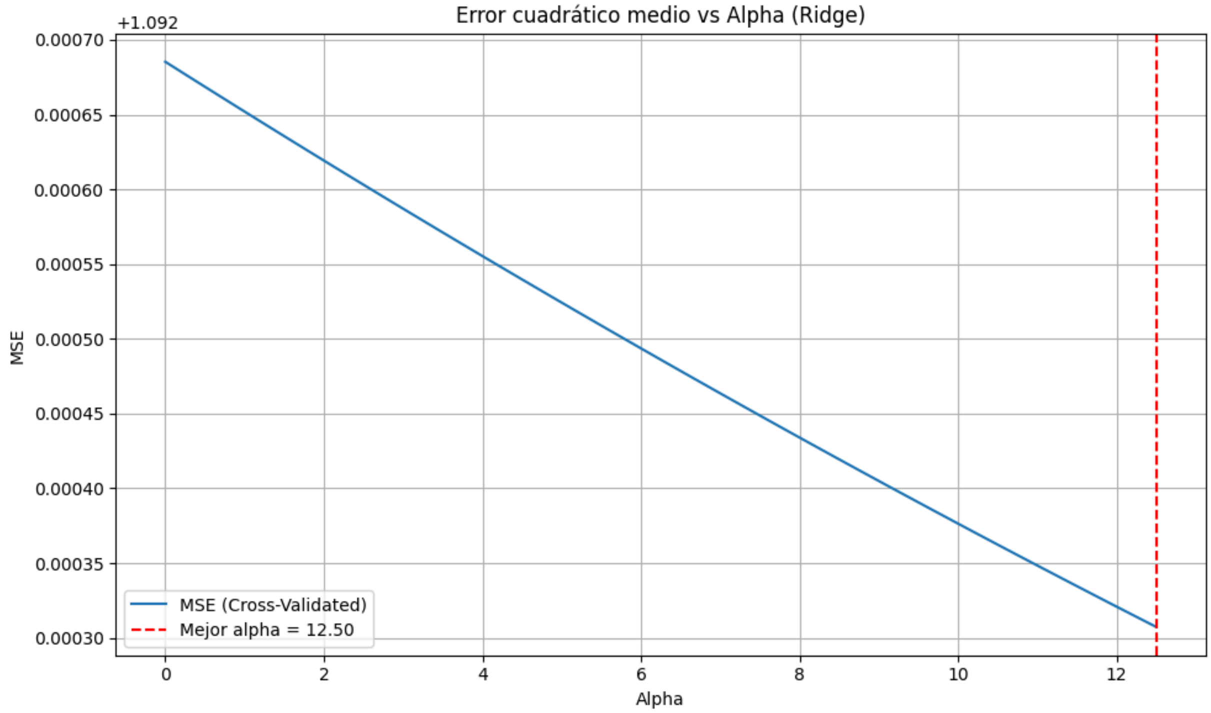
In [159…

```
best_alpha = grid.best_params_['regressor__alpha']
print("Mejor alpha:", best_alpha)
```

```
Mejor alpha: 12.5
```

In [160…

```
# Obtenemos los MSEs negativos y los convertimos a positivos
mean_mse = -grid.cv_results_['mean_test_score']

# Graficamos
plt.figure(figsize=(10, 6))
plt.plot(alphas, mean_mse, label='MSE (Cross-Validated)')
plt.axvline(x=best_alpha, color='r', linestyle='--', label=f'Mejor alpha = {
plt.title('Error cuadrático medio vs Alpha (Ridge)')
plt.xlabel('Alpha')
plt.ylabel('MSE')
plt.legend()
plt.grid()

# Forzar que matplotlib no use offset notation
plt.ticklabel_format(style='plain', axis='y')
plt.tight_layout()
plt.show()
```

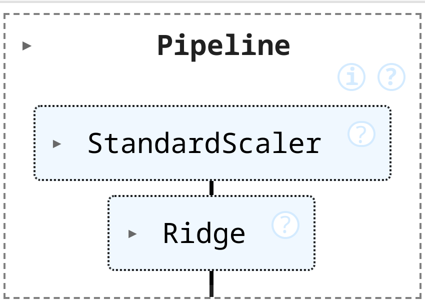


El gráfico muestra una tendencia descendente del error cuadrático medio (MSE) a medida que el valor de alpha aumenta dentro del rango [0, 12.5]. Esto indica que el modelo se beneficia de una mayor regularización, ya que el MSE sigue disminuyendo en ese intervalo. El valor óptimo hallado (alpha = 12.5) está en el extremo superior del rango explorado, lo que sugiere que probablemente no se ha alcanzado el mínimo global del MSE.

In [161…

```
# Entrenamos el modelo final con ese alpha
modelo_final = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', Ridge(alpha=best_alpha))
])

modelo_final.fit(X_train, y_train)
```

Out[161…



In [162…

```
# Predecir y evaluar
y_pred_train = modelo_final.predict(X_train)

# Calcular varianza total y explicada
var_total = np.var(y_train)
var_residual = np.var(y_train - y_pred_train)
var_explicada = var_total - var_residual
```

```
r2 = r2_score(y_train, y_pred_train) # porcentaje de varianza explicada

# Resultados
print("Resultados del modelo en Train:")
print(f"Varianza total del target: {var_total:.4f}")
print(f"Varianza explicada por el modelo: {var_explicada:.4f}")
print(f"R²: {r2:.2%}")
```

```
Resultados del modelo en Train:
Varianza total del target: 1.3254
Varianza explicada por el modelo: 0.8169
R²: 61.64%
```

El modelo entrenado con regresión Ridge y el mejor valor de regularización (alpha) obtenido por validación cruzada logra explicar el **61.64%** de la variabilidad del target en el conjunto de entrenamiento

In [163…

```
# 1. Predecir sobre el set de test
y_pred = modelo_final.predict(X_test)

# 2. Calcular métricas
mse_ridge = mean_squared_error(y_test, y_pred)
mae_ridge = mean_absolute_error(y_test, y_pred)
r2_ridge = r2_score(y_test, y_pred)

# 3. Mostrar resultados
print("Resultados del modelo en Test:")
print(f"MSE (Error cuadrático medio): {mse_ridge:.4f}")
print(f"MAE (Error absoluto medio):  {mae_ridge:.4f}")
print(f"R² (Varianza explicada):     {r2_ridge:.2%}")
```

```
Resultados del modelo en Test:
MSE (Error cuadrático medio): 0.5187
MAE (Error absoluto medio):  0.5247
R² (Varianza explicada):     61.49%
```

El modelo muestra un rendimiento muy similar al observado en el conjunto de entrenamiento, con un R^2 de **61.49%**.

---

6. Comparar, entre la regresión lineal y la mejor regresión de Ridge, los resultados obtenidos en el set de evaluación. ¿Cuál da mejores resultados (usando MSE y MAE)? Conjeturar por qué el mejor modelo mejora. ¿Qué error puede haberse reducido?

In [164…

```
# Comparación
print("Comparación en set de evaluación:\n")
print(f"{'Métrica':<10}{'Lineal':<10}{'Ridge':<10}{'Mejora (%)':<10}")
print(f"{'-'*40}")
print(f"{'MSE':<10}{mse_reg:.4f}{mse_ridge:>10.4f}{(mse_reg-mse_ridge)/mse_r
print(f"{'MAE':<10}{mae_reg:.4f}{mae_ridge:>10.4f}{(mae_reg-mae_ridge)/mae_r
print(f"{'R²':<10}{r2_reg:.2%}{r2_ridge:>10.2%}{(r2_ridge-r2_reg)*100:>10.2f
```

```
Comparación en set de evaluación:

Métrica   Lineal    Ridge     Mejora (%)
----------------------------------------
MSE       0.5186    0.5187     -0.02%
MAE       0.5248    0.5247      0.01%
R²        61.50%    61.49%     -0.01pp
```

En este caso, la regresión Ridge no logra mejorar significativamente los resultados respecto a la regresión lineal. Las métricas obtenidas en el set de evaluación son prácticamente idénticas: el MSE y MAE varían en menos de un 0.02%, y el R^2 es prácticamente el mismo (61.50% vs. 61.49%).

Esto sugiere que la regularización introducida por Ridge no aporta beneficios relevantes en este escenario, posiblemente porque:

- El modelo lineal ya tiene un buen ajuste sin signos de sobreajuste.
- No hay colinealidades fuertes que perjudiquen el entrenamiento.
- El número de variables es moderado y el preprocesamiento fue efectivo.

Link al Notebook

Se puede encontrar el trabajo completo en el siguiente link: Repositorio GitHub