testando

October 27, 2025

[2]:
```
import pandas as pd

# Substitua 'nome_do_arquivo.csv' pelo nome real do seu arquivo
df = pd.read_csv('housing.csv')
```

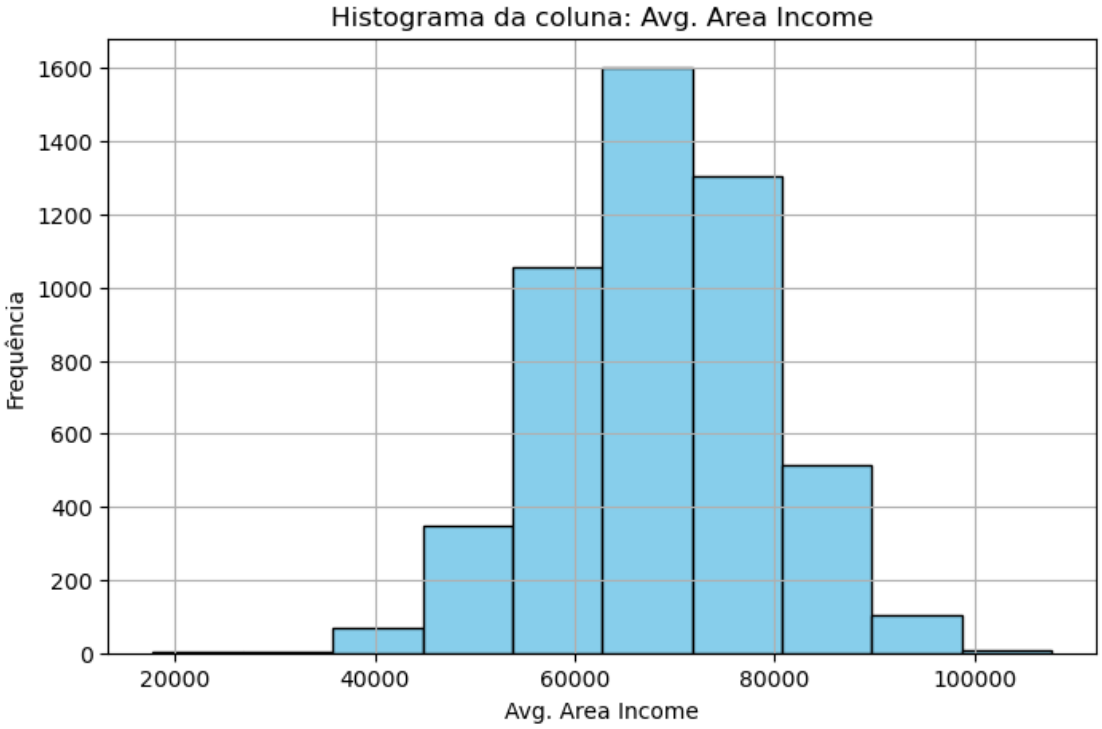
[4]:
```
df.head(5)
```

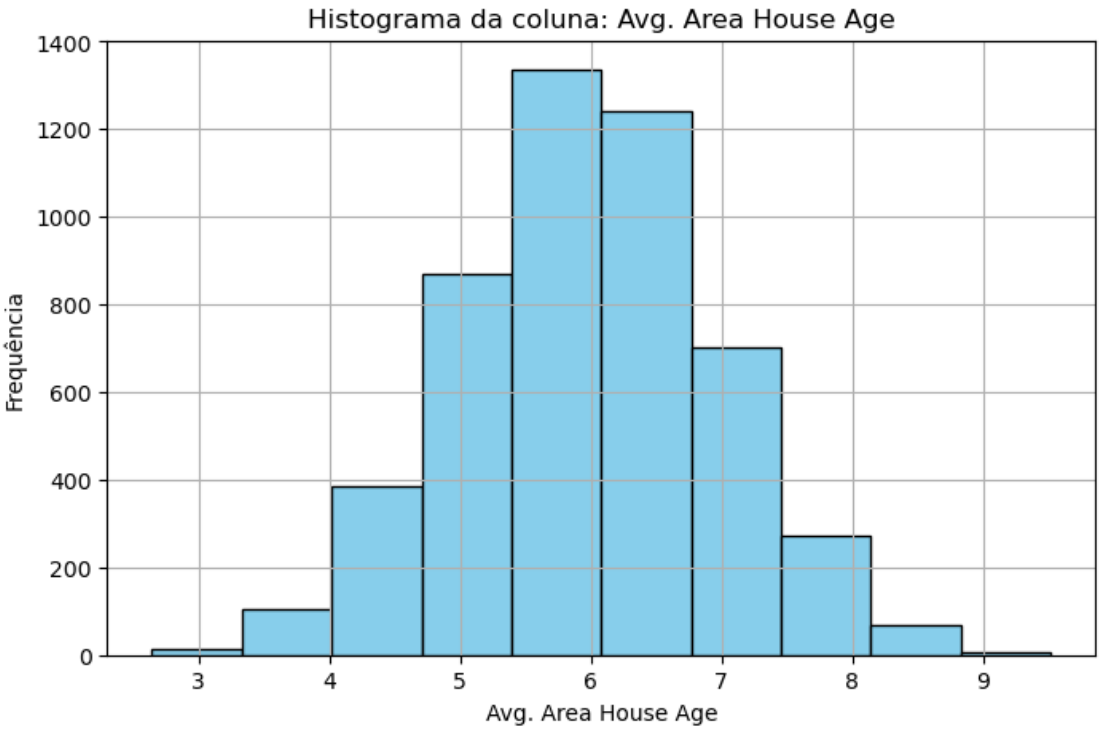
[4]:

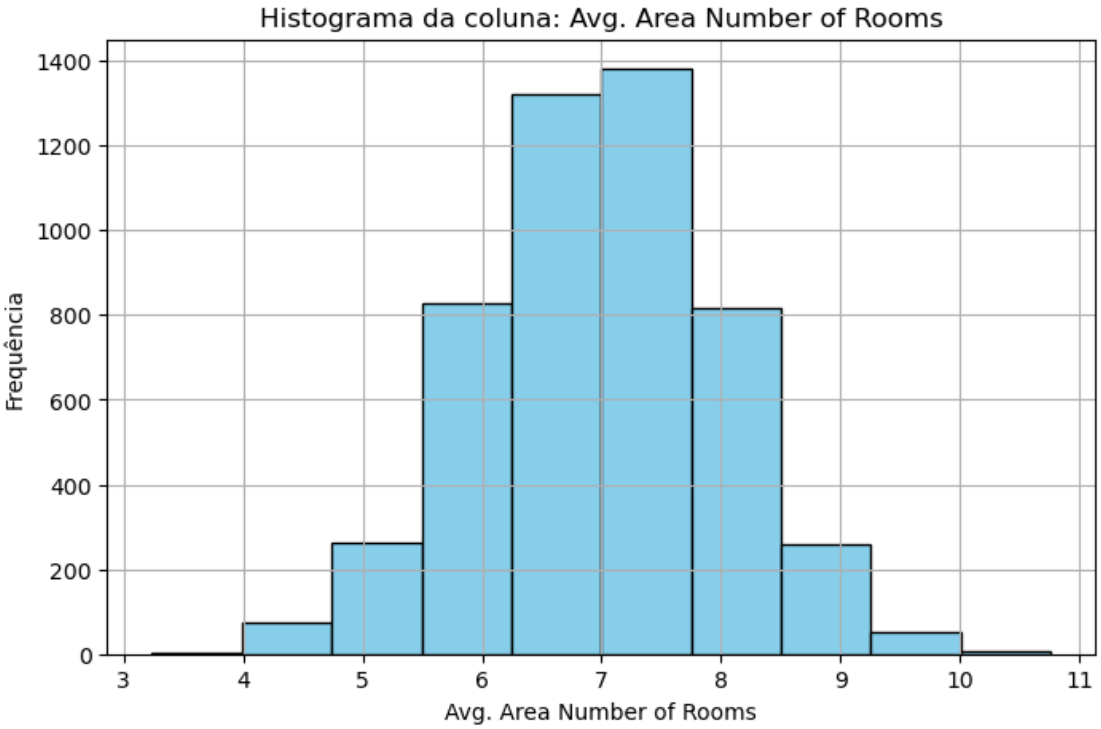
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA…
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\nFPO AE 09386

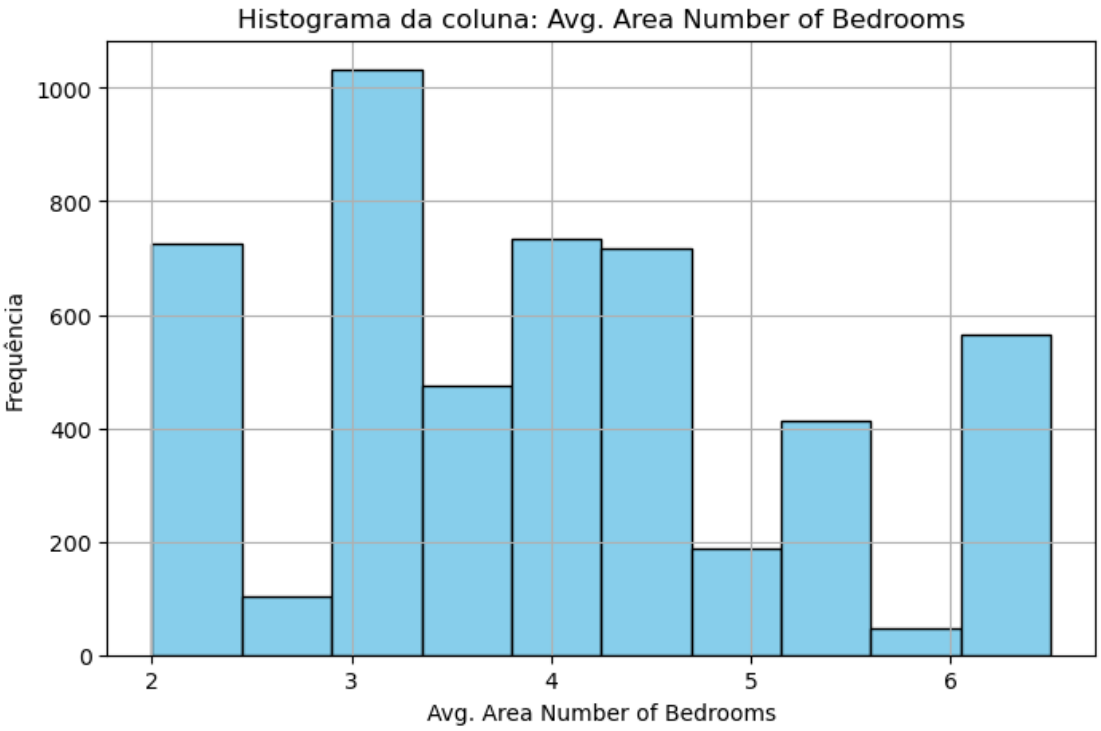
[13]:
```
import matplotlib.pyplot as plt

colunas_numericas = df.select_dtypes(include='number').columns

colunas_numericas = df.select_dtypes(include='number').columns

# Gerar histogramas para cada coluna numérica
for coluna in colunas_numericas:
    plt.figure(figsize=(8, 5))
```

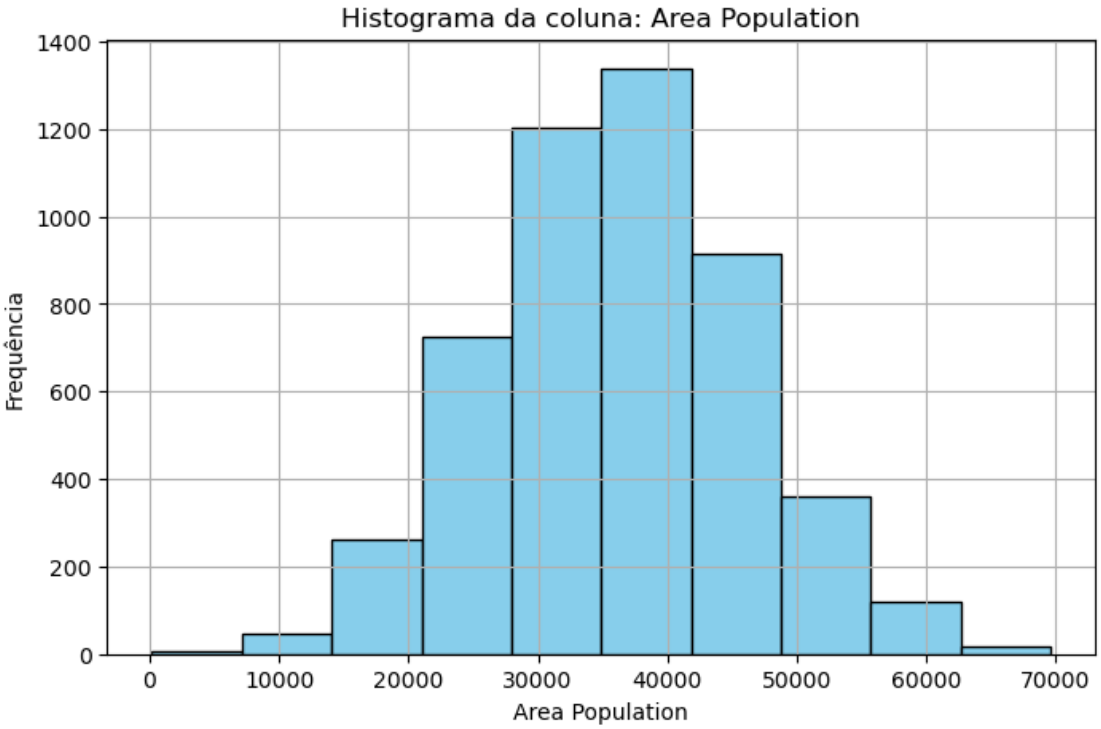
```
    plt.hist(df[coluna], bins=10, color='skyblue', edgecolor='black')
    plt.title(f'Histograma da coluna: {coluna}')
    plt.xlabel(coluna)
    plt.ylabel('Frequência')
    plt.grid(True)
    plt.show()
```

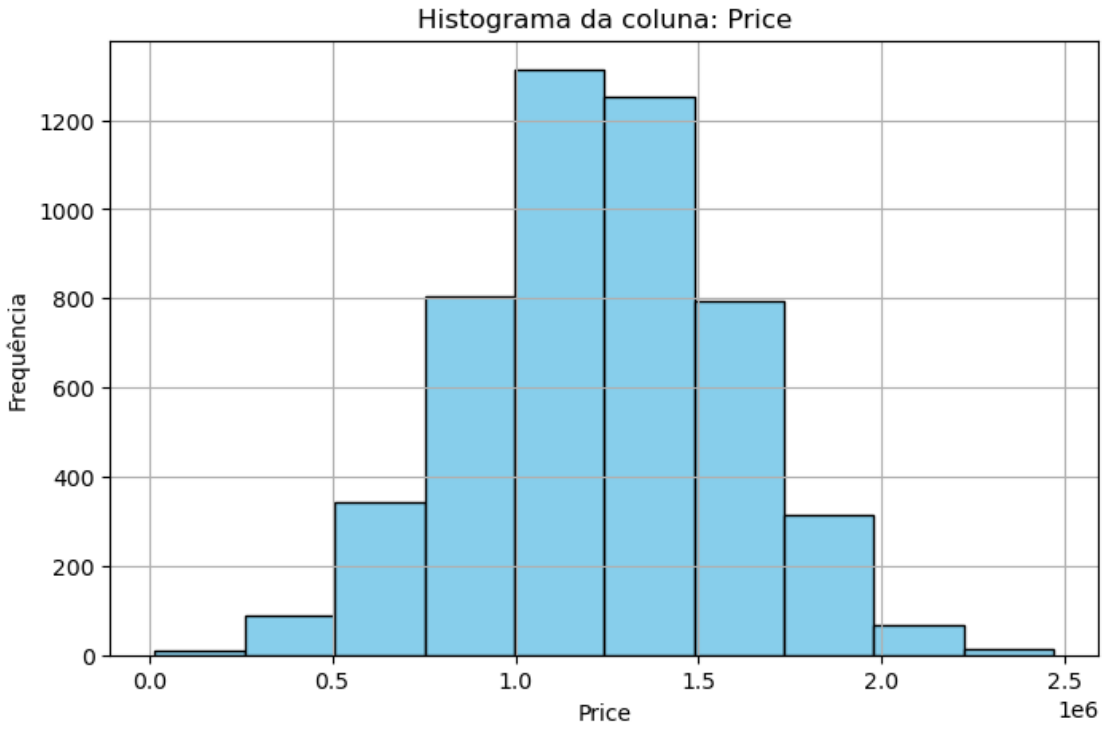



Histograma da coluna: Avg. Area House Age
Frequência
Avg. Area House Age
0
200
400
600
800
1000
1200
1400
3
4
5
6
7
8
9


Histograma da coluna: Avg. Area Number of Rooms
Frequência
Avg. Area Number of Rooms
0
200
400
600
800
1000
1200
1400
3
4
5
6
7
8
9
10
11

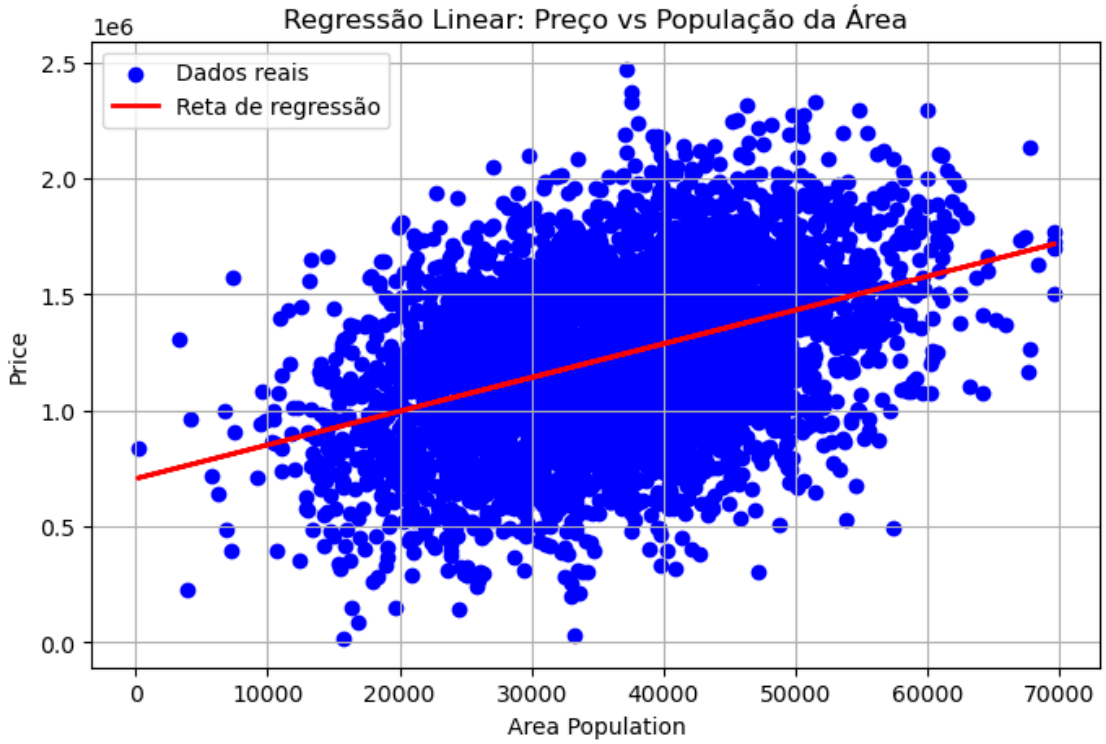

Histograma da coluna: Avg. Area Number of Bedrooms
Frequência
Avg. Area Number of Bedrooms
0
200
400
600
800
1000
2
3
4
5
6


Histograma da coluna: Area Population
Frequência
Area Population
0
200
400
600
800
1000
1200
1400
0
10000
20000
30000
40000
50000
60000
70000

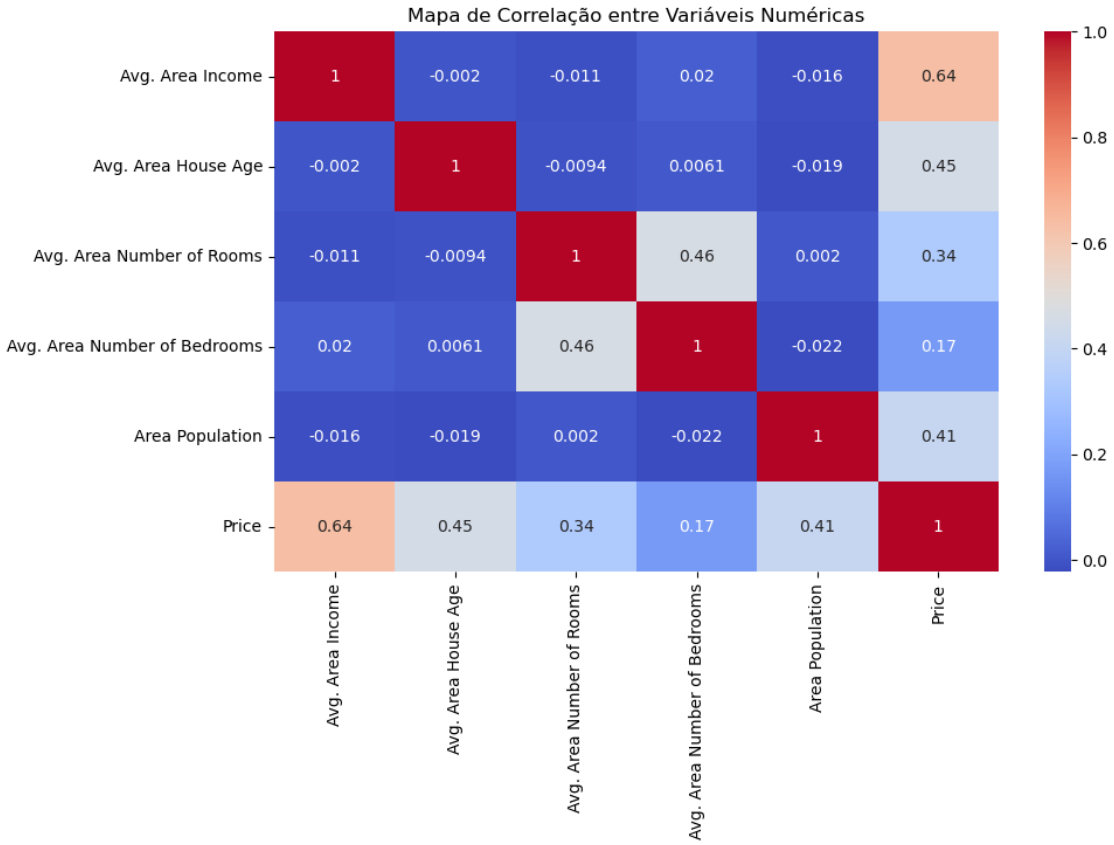


[16]:
```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Usar uma variável para prever o preço (exemplo: Area Population)
X = df[['Area Population']]
y = df['Price']

# Treinar o modelo
modelo = LinearRegression()
modelo.fit(X, y)

# Previsões
y_pred = modelo.predict(X)

# Plotar os dados e a reta
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color='blue', label='Dados reais')
plt.plot(X, y_pred, color='red', linewidth=2, label='Reta de regressão')
plt.title('Regressão Linear: Preço vs População da Área')
plt.xlabel('Area Population')
plt.ylabel('Price')
```

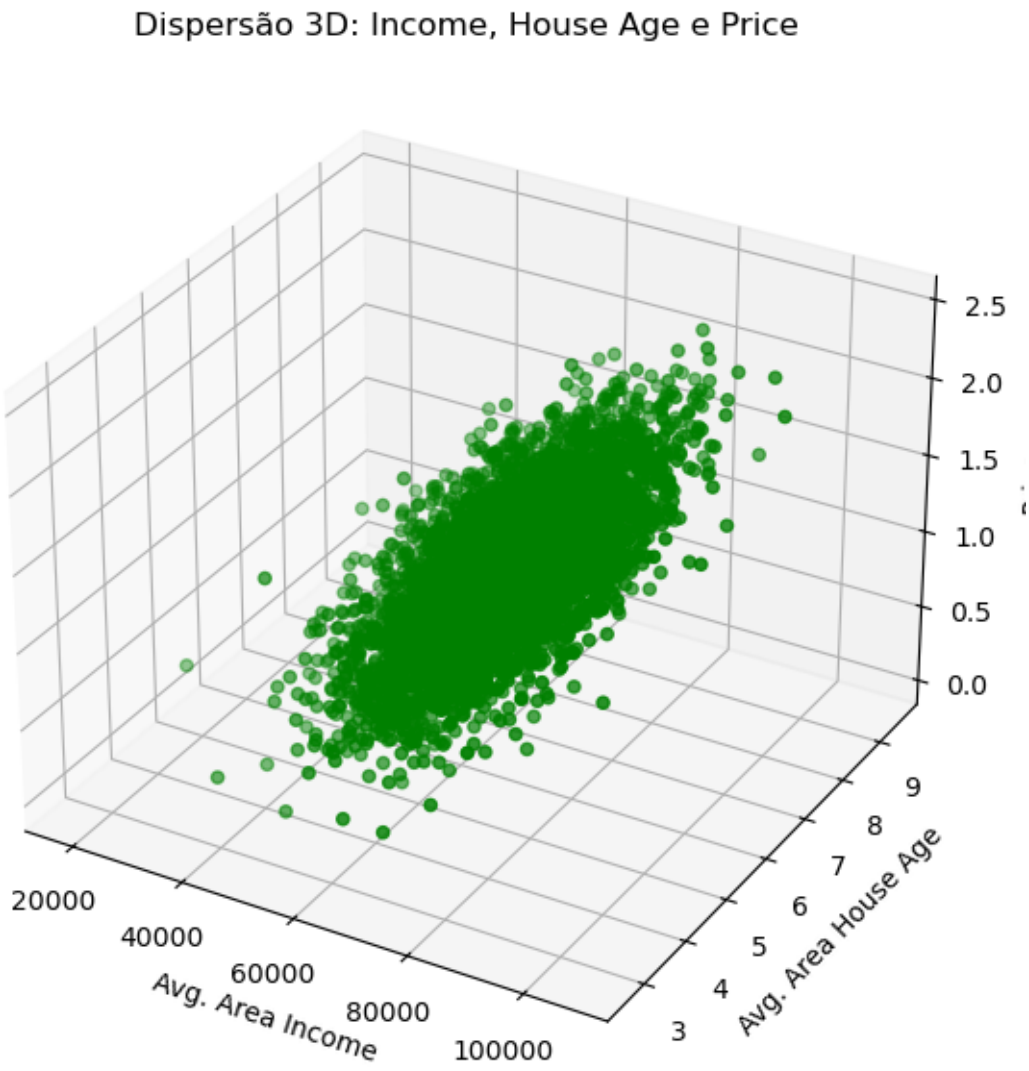
```
plt.legend()
plt.grid(True)
plt.show()
```



[19]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Selecionar apenas colunas numéricas
df_numerico = df.select_dtypes(include='number')

# Gerar o heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df_numerico.corr(), annot=True, cmap='coolwarm')
plt.title('Mapa de Correlação entre Variáveis Numéricas')
plt.show()
```



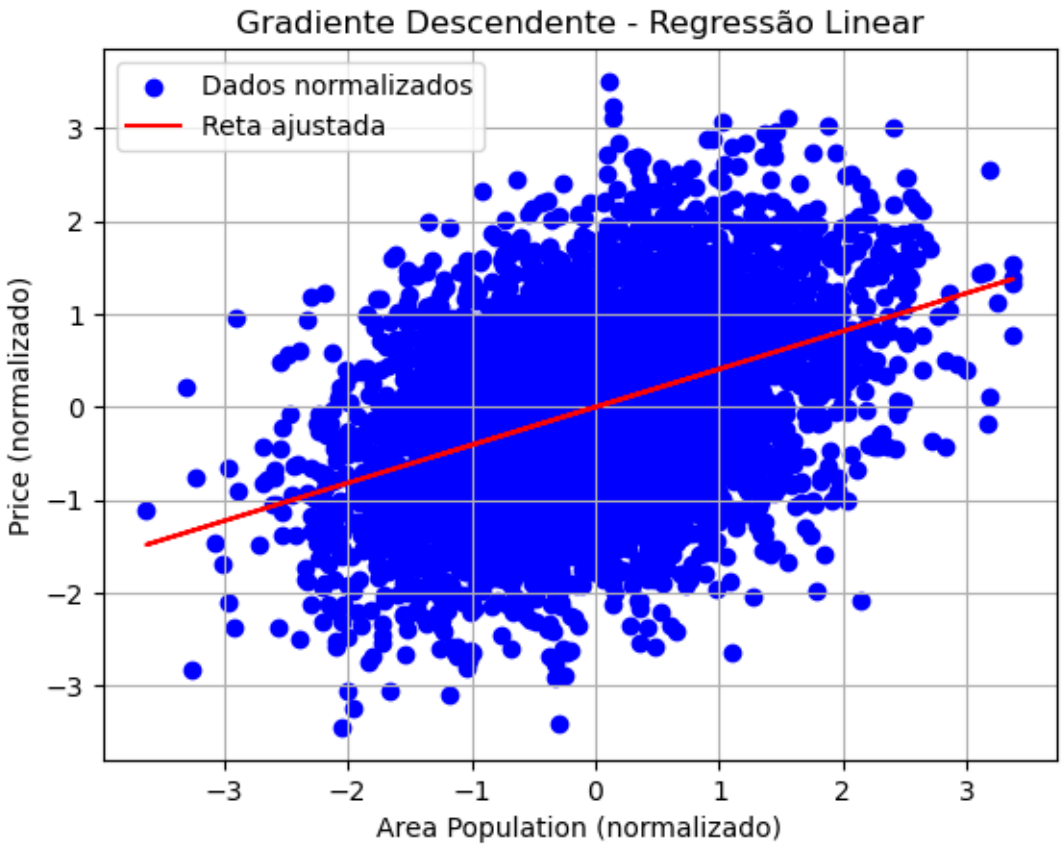
[20]:
```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['Avg. Area Income'], df['Avg. Area House Age'], df['Price'],␣
 ↪c='green', marker='o')
ax.set_xlabel('Avg. Area Income')
ax.set_ylabel('Avg. Area House Age')
ax.set_zlabel('Price')
ax.set_title('Dispersão 3D: Income, House Age e Price')
plt.show()
```



[21]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Selecionar variável independente e dependente
X = df['Area Population'].values
y = df['Price'].values

# Normalizar os dados
X = (X - X.mean()) / X.std()
y = (y - y.mean()) / y.std()

# Inicializar parâmetros
```

```
m = 0  # inclinação
b = 0  # intercepto
alpha = 0.01  # taxa de aprendizado
epochs = 1000  # número de iterações

# Gradiente descendente
for i in range(epochs):
    y_pred = m * X + b
    error = y_pred - y
    cost = (error ** 2).mean()

    # Derivadas
    dm = (2 * (error * X).mean())
    db = (2 * error.mean())

    # Atualização dos parâmetros
    m -= alpha * dm
    b -= alpha * db

# Exibir resultados
print(f"m (inclinação): {m}")
print(f"b (intercepto): {b}")
print(f"Custo final: {cost}")

# Visualizar a reta ajustada
plt.scatter(X, y, color='blue', label='Dados normalizados')
plt.plot(X, m * X + b, color='red', label='Reta ajustada')
plt.title('Gradiente Descendente - Regressão Linear')
plt.xlabel('Area Population (normalizado)')
plt.ylabel('Price (normalizado)')
plt.legend()
plt.grid(True)
plt.show()
```

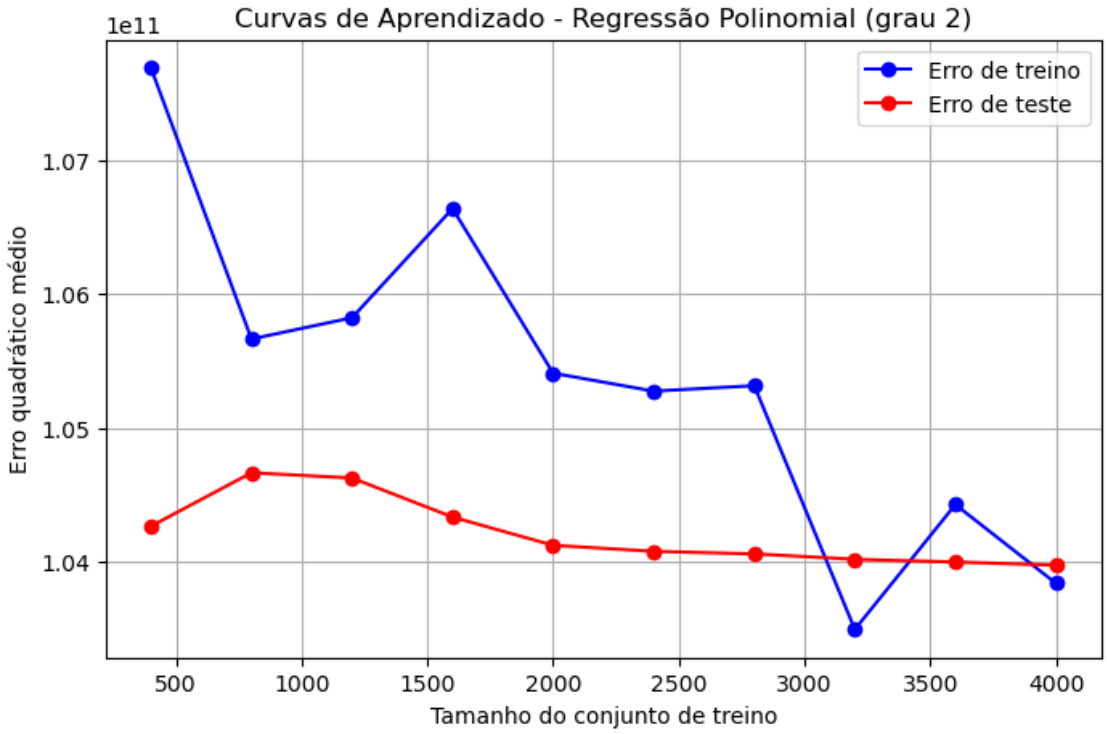
```
m (inclinação): 0.4085558786333454
b (intercepto): 5.563549621001604e-17
Custo final: 0.8330820934723008
```



[36]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.metrics import mean_squared_error, r2_score

X = df[['Area Population']]
y = df['Price']

# Transformar em polinômio de grau 2
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Dividir em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,␣
 ↪random_state=42)

# Treinar o modelo
```

```python
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Previsões
y_pred = modelo.predict(X_test)

# Avaliação
print("Erro quadrático médio (MSE):", mean_squared_error(y_test, y_pred))
print("Coeficiente de determinação (R²):", r2_score(y_test, y_pred))

# Curvas de aprendizado
train_sizes, train_scores, test_scores = learning_curve(
    modelo, X_poly, y, cv=5, scoring='neg_mean_squared_error',
    train_sizes=np.linspace(0.1, 1.0, 10), random_state=42
)

# Média e desvio padrão
train_scores_mean = -np.mean(train_scores, axis=1)
test_scores_mean = -np.mean(test_scores, axis=1)

# Gráfico
plt.figure(figsize=(8, 5))
plt.plot(train_sizes, train_scores_mean, 'o-', color='blue', label='Erro de␣
 ↪treino')
plt.plot(train_sizes, test_scores_mean, 'o-', color='red', label='Erro de␣
 ↪teste')
plt.title('Curvas de Aprendizado - Regressão Polinomial (grau 2)')
plt.xlabel('Tamanho do conjunto de treino')
plt.ylabel('Erro quadrático médio')
plt.legend()
plt.grid(True)
plt.show()
```

```
Erro quadrático médio (MSE): 101934056935.76347
Coeficiente de determinação (R²): 0.1714864341215081
```



[44]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

X = df[['Area Population']]
y = df['Price']

# Dividir em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Modelos
modelos = {
    'Ridge': Ridge(alpha=1.0),
    'Lasso': Lasso(alpha=0.1),
    'ElasticNet': ElasticNet(alpha=0.1, l1_ratio=0.5)
}

# Treinar e avaliar
for nome, modelo in modelos.items():
    modelo.fit(X_train, y_train)
```

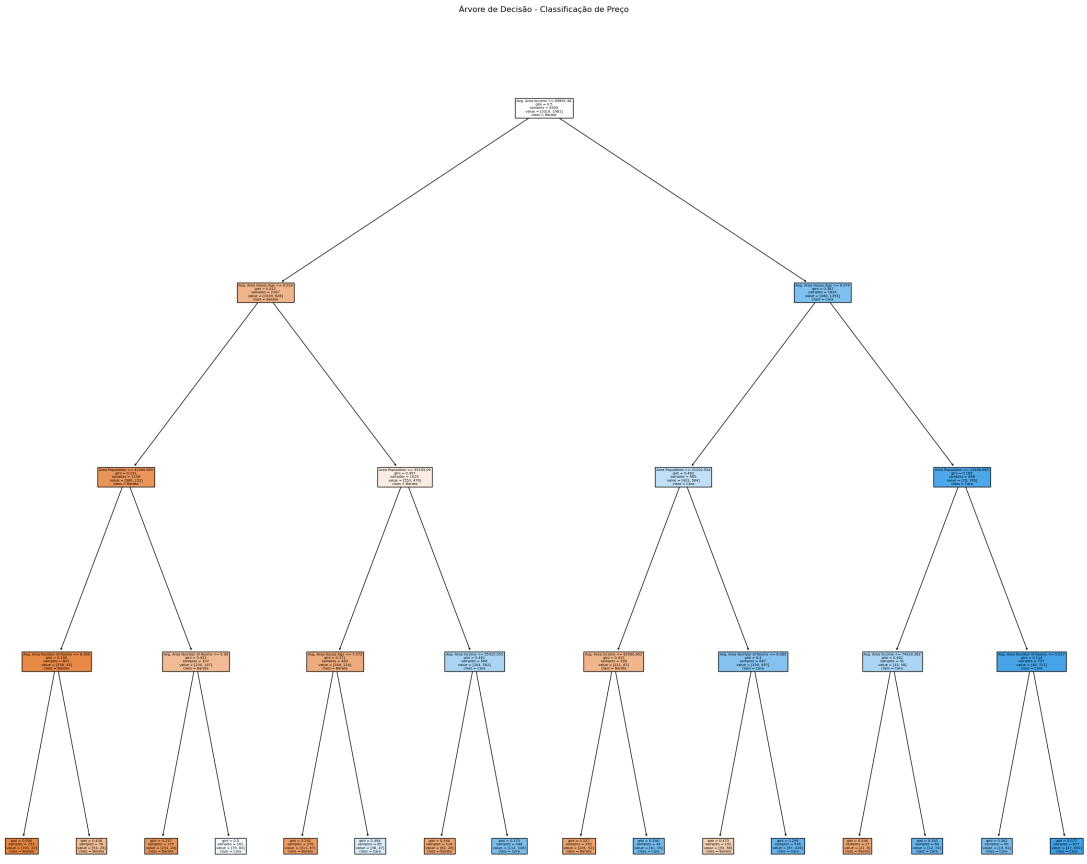
```
    y_pred = modelo.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"{nome} → MSE: {mse:.2f}, R²: {r2:.4f}")
```

```
Ridge → MSE: 101884635124.57, R²: 0.1719
Lasso → MSE: 101884635124.67, R²: 0.1719
ElasticNet → MSE: 101884635125.39, R²: 0.1719
```

[45]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt


# Criar uma variável de classificação: "cara" (1) se Price > média, "barata"␣
 ↪(0) se Price <= média
df['Classe'] = (df['Price'] > df['Price'].mean()).astype(int)

# Selecionar variáveis independentes
X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
        'Avg. Area Number of Bedrooms', 'Area Population']]
y = df['Classe']

# Dividir em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Criar e treinar o modelo
modelo = DecisionTreeClassifier(max_depth=4, random_state=42)
modelo.fit(X_train, y_train)

# Avaliar o modelo
y_pred = modelo.predict(X_test)
print("Matriz de Confusão:\n", confusion_matrix(y_test, y_pred))
print("\nRelatório de Classificação:\n", classification_report(y_test, y_pred))

# Visualizar a árvore
plt.figure(figsize=(30, 25))
plot_tree(modelo, feature_names=X.columns, class_names=['Barata', 'Cara'],␣
 ↪filled=True)
plt.title('Árvore de Decisão - Classificação de Preço')
plt.show()
```

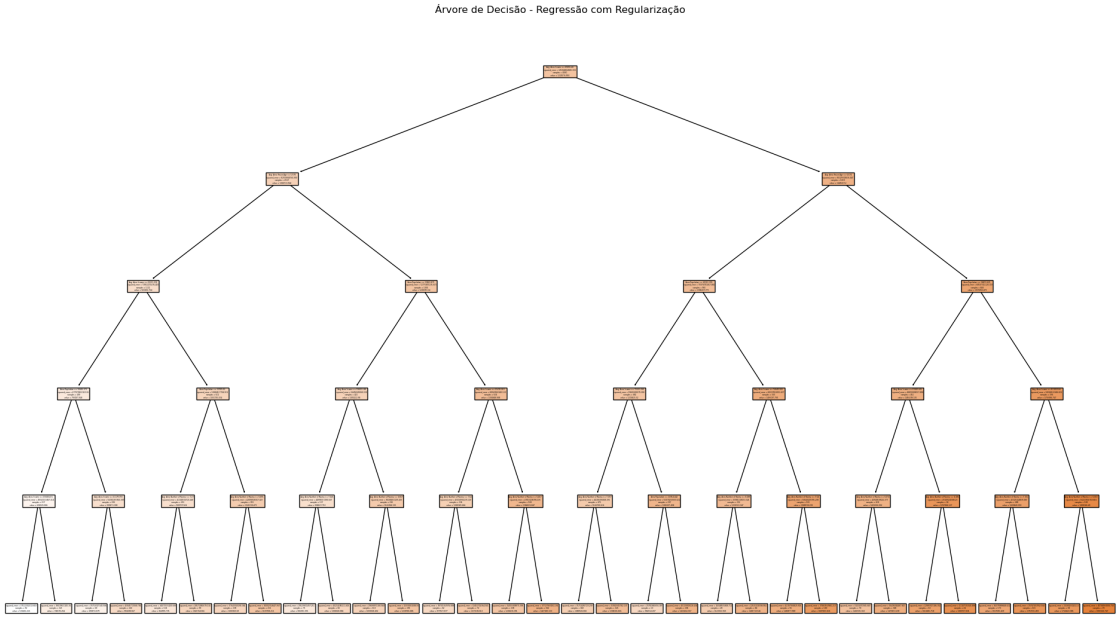
```
Matriz de Confusão:
 [[359 119]
```

```
 [ 93 429]]

Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.79      0.75      0.77       478
           1       0.78      0.82      0.80       522

    accuracy                           0.79      1000
   macro avg       0.79      0.79      0.79      1000
weighted avg       0.79      0.79      0.79      1000
```



[35]:

```
Erro quadrático médio (MSE): 42485358826.53633
Coeficiente de determinação (R²): 0.6546816913096776
```



[39]:

Estatísticas Descritivas e de Forma:

	Média	Mediana	Desvio Padrão	\
Avg. Area Income	6.858311e+04	6.880429e+04	10657.991214	
Avg. Area House Age	5.977222e+00	5.970429e+00	0.991456	
Avg. Area Number of Rooms	6.987792e+00	7.002902e+00	1.005833	
Avg. Area Number of Bedrooms	3.981330e+00	4.050000e+00	1.234137	
Area Population	3.616352e+04	3.619941e+04	9925.650114	
Price	1.232073e+06	1.232669e+06	353117.626584	
Classe	5.006000e-01	1.000000e+00	0.500050	

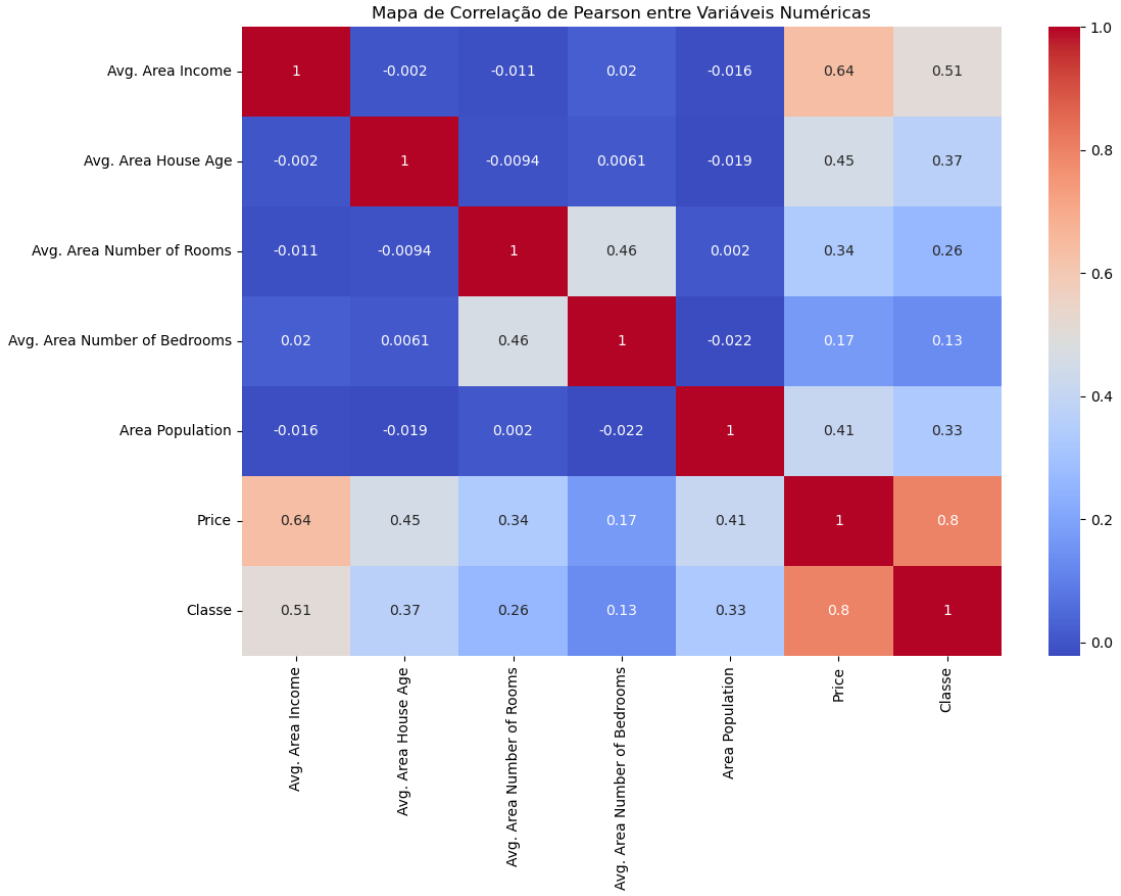
	Variância	Mínimo	Máximo	\
Avg. Area Income	1.135928e+08	17796.631190	1.077017e+05	
Avg. Area House Age	9.829854e-01	2.644304	9.519088e+00	
Avg. Area Number of Rooms	1.011700e+00	3.236194	1.075959e+01	
Avg. Area Number of Bedrooms	1.523095e+00	2.000000	6.500000e+00	
Area Population	9.851853e+07	172.610686	6.962171e+04	
Price	1.246921e+11	15938.657920	2.469066e+06	
Classe	2.500496e-01	0.000000	1.000000e+00	

	Amplitude	Assimetria (Skewness)	Curtose
Avg. Area Income	8.990512e+04	-0.033710	0.044329
Avg. Area House Age	6.874784e+00	-0.007212	-0.084554
Avg. Area Number of Rooms	7.523394e+00	-0.040984	-0.075777

```
Avg. Area Number of Bedrooms  4.500000e+00              0.376128 -0.702064
Area Population               6.944910e+04              0.050634 -0.007926
Price                         2.453127e+06             -0.002717 -0.056063
Classe                        1.000000e+00             -0.002400 -1.999994
```

Correlação de Pearson:

```
                            Avg. Area Income  Avg. Area House Age  \
Avg. Area Income                    1.000000            -0.002007
Avg. Area House Age                -0.002007             1.000000
Avg. Area Number of Rooms          -0.011032            -0.009428
Avg. Area Number of Bedrooms        0.019788             0.006149
Area Population                    -0.016234            -0.018743
Price                               0.639734             0.452543
Classe                              0.505707             0.365857

                            Avg. Area Number of Rooms  \
Avg. Area Income                            -0.011032
Avg. Area House Age                         -0.009428
Avg. Area Number of Rooms                    1.000000
Avg. Area Number of Bedrooms                 0.462695
Area Population                              0.002040
Price                                        0.335664
Classe                                       0.262750

                            Avg. Area Number of Bedrooms  Area Population  \
Avg. Area Income                                0.019788        -0.016234
Avg. Area House Age                             0.006149        -0.018743
Avg. Area Number of Rooms                       0.462695         0.002040
Avg. Area Number of Bedrooms                    1.000000        -0.022168
Area Population                                -0.022168         1.000000
Price                                           0.171071         0.408556
Classe                                          0.129665         0.328662

                              Price    Classe
Avg. Area Income           0.639734  0.505707
Avg. Area House Age        0.452543  0.365857
Avg. Area Number of Rooms  0.335664  0.262750
Avg. Area Number of Bedrooms  0.171071  0.129665
Area Population            0.408556  0.328662
Price                      1.000000  0.799461
Classe                     0.799461  1.000000
```




```
Variáveis mais correlacionadas com 'Price':

Classe                          0.799461
Avg. Area Income                0.639734
Avg. Area House Age             0.452543
Area Population                 0.408556
Avg. Area Number of Rooms       0.335664
Avg. Area Number of Bedrooms    0.171071
Name: Price, dtype: float64
```

[46]:
```
# Importação de bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.preprocessing import PolynomialFeatures
```

```
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier,␣
 ↪plot_tree
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.metrics import mean_squared_error, r2_score,␣
 ↪classification_report, confusion_matrix

# Carregar os dados
#df = pd.read_csv('nome_do_arquivo.csv')  # Substitua pelo nome real do seu␣
 ↪arquivo

# Selecionar colunas numéricas
df_num = df.select_dtypes(include='number')

#  Estatísticas descritivas e de forma
estatisticas = pd.DataFrame({
    'Média': df_num.mean(),
    'Mediana': df_num.median(),
    'Desvio Padrão': df_num.std(),
    'Variância': df_num.var(),
    'Mínimo': df_num.min(),
    'Máximo': df_num.max(),
    'Amplitude': df_num.max() - df_num.min(),
    'Assimetria': df_num.apply(skew),
    'Curtose': df_num.apply(kurtosis)
})
print(" Estatísticas:\n", estatisticas)

#  Correlação de Pearson
correlacao = df_num.corr()
print("\n Correlação de Pearson:\n", correlacao)

# Heatmap de correlação
plt.figure(figsize=(12, 8))
sns.heatmap(correlacao, annot=True, cmap='coolwarm')
plt.title('Mapa de Correlação de Pearson')
plt.show()

#  Regressão Linear Simples
X_lin = df[['Area Population']]
y = df['Price']
modelo_lin = LinearRegression()
modelo_lin.fit(X_lin, y)
y_pred_lin = modelo_lin.predict(X_lin)
print("\n Regressão Linear → R²:", r2_score(y, y_pred_lin))

#  Regressão Polinomial + Curvas de Aprendizado
poly = PolynomialFeatures(degree=2)
```

```
X_poly = poly.fit_transform(X_lin)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,␣
 ↪random_state=42)
modelo_poly = LinearRegression()
modelo_poly.fit(X_train, y_train)
y_pred_poly = modelo_poly.predict(X_test)
print("\n Regressão Polinomial → R²:", r2_score(y_test, y_pred_poly))

# Curvas de aprendizado
train_sizes, train_scores, test_scores = learning_curve(
    modelo_poly, X_poly, y, cv=5, scoring='neg_mean_squared_error',
    train_sizes=np.linspace(0.1, 1.0, 10)
)
train_scores_mean = -np.mean(train_scores, axis=1)
test_scores_mean = -np.mean(test_scores, axis=1)
plt.figure(figsize=(8, 5))
plt.plot(train_sizes, train_scores_mean, 'o-', label='Erro de treino')
plt.plot(train_sizes, test_scores_mean, 'o-', label='Erro de teste')
plt.title('Curvas de Aprendizado - Regressão Polinomial')
plt.xlabel('Tamanho do treino')
plt.ylabel('Erro quadrático médio')
plt.legend()
plt.grid(True)
plt.show()

#  Modelos Regularizados
X_multi = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of␣
 ↪Rooms',
              'Avg. Area Number of Bedrooms', 'Area Population']]
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X_multi, y, test_size=0.2,␣
 ↪random_state=42)
modelos = {
    'Ridge': Ridge(alpha=1.0),
    'Lasso': Lasso(alpha=0.1),
    'ElasticNet': ElasticNet(alpha=0.1, l1_ratio=0.5)
}
for nome, modelo in modelos.items():
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    print(f"{nome} → MSE: {mean_squared_error(y_test, y_pred):.2f}, R²:␣
 ↪{r2_score(y_test, y_pred):.4f}")

#  Árvore de Decisão - Regressão
modelo_tree_reg = DecisionTreeRegressor(max_depth=5, min_samples_split=10,␣
 ↪min_samples_leaf=5, random_state=42)
modelo_tree_reg.fit(X_train, y_train)
```

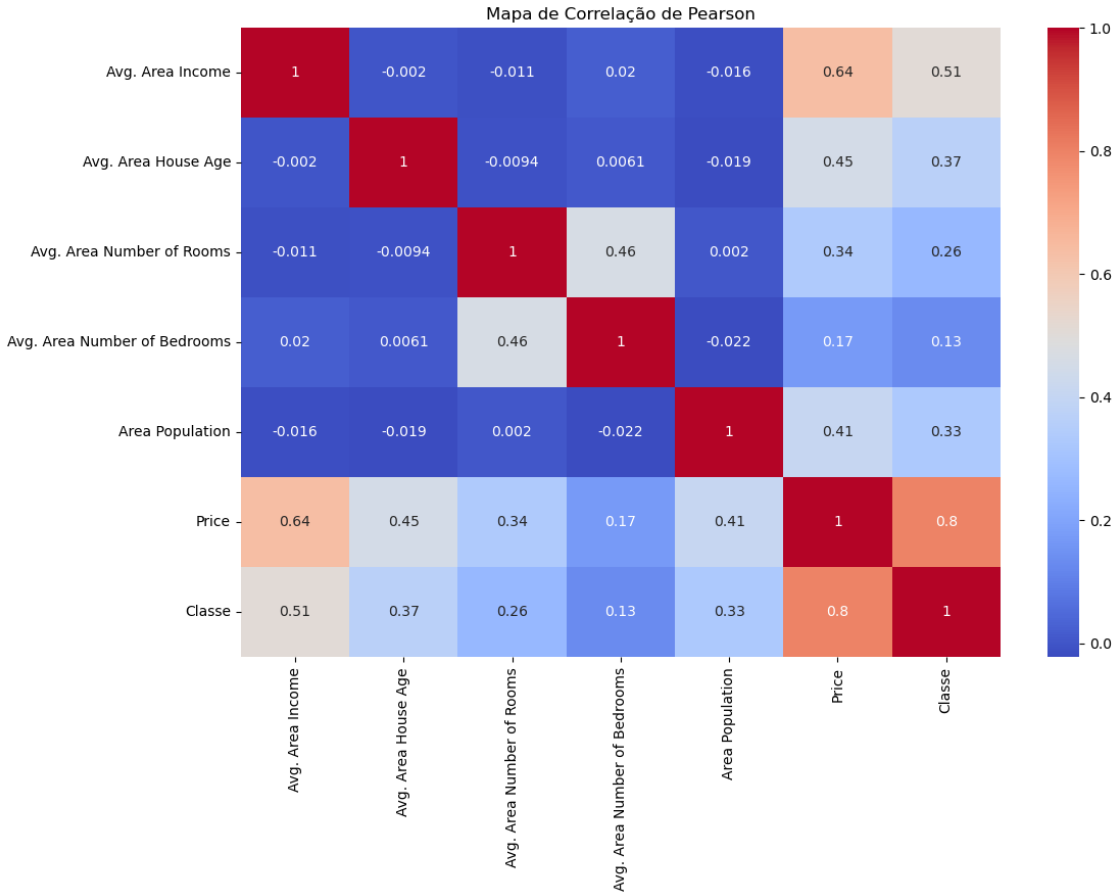
```
y_pred_tree = modelo_tree_reg.predict(X_test)
print("\n Árvore de Regressão → R²:", r2_score(y_test, y_pred_tree))
plt.figure(figsize=(24, 14))
plot_tree(modelo_tree_reg, feature_names=X_multi.columns, filled=True)
plt.title('Árvore de Decisão - Regressão')
plt.show()

#  Árvore de Decisão - Classificação
df['Classe'] = (df['Price'] > df['Price'].mean()).astype(int)
X_class = X_multi
y_class = df['Classe']
X_train, X_test, y_train, y_test = train_test_split(X_class, y_class,␣
 ↪test_size=0.2, random_state=42)
modelo_tree_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
modelo_tree_clf.fit(X_train, y_train)
y_pred_clf = modelo_tree_clf.predict(X_test)
print("\n Árvore de Classificação:\n", classification_report(y_test,␣
 ↪y_pred_clf))
plt.figure(figsize=(24, 14))
plot_tree(modelo_tree_clf, feature_names=X_class.columns,␣
 ↪class_names=['Barata', 'Cara'], filled=True)
plt.title('Árvore de Decisão - Classificação')
plt.show()
```

```
 Estatísticas:
                                    Média       Mediana  Desvio Padrão  \
Avg. Area Income              6.858311e+04  6.880429e+04   10657.991214
Avg. Area House Age           5.977222e+00  5.970429e+00       0.991456
Avg. Area Number of Rooms     6.987792e+00  7.002902e+00       1.005833
Avg. Area Number of Bedrooms  3.981330e+00  4.050000e+00       1.234137
Area Population               3.616352e+04  3.619941e+04    9925.650114
Price                         1.232073e+06  1.232669e+06  353117.626584
Classe                        5.006000e-01  1.000000e+00       0.500050

                                 Variância        Mínimo        Máximo  \
Avg. Area Income              1.135928e+08  17796.631190  1.077017e+05
Avg. Area House Age           9.829854e-01      2.644304  9.519088e+00
Avg. Area Number of Rooms     1.011700e+00      3.236194  1.075959e+01
Avg. Area Number of Bedrooms  1.523095e+00      2.000000  6.500000e+00
Area Population               9.851853e+07    172.610686  6.962171e+04
Price                         1.246921e+11  15938.657920  2.469066e+06
Classe                        2.500496e-01      0.000000  1.000000e+00

                                 Amplitude  Assimetria   Curtose
Avg. Area Income              8.990512e+04   -0.033710  0.044329
Avg. Area House Age           6.874784e+00   -0.007212 -0.084554
Avg. Area Number of Rooms     7.523394e+00   -0.040984 -0.075777
```

```
Avg. Area Number of Bedrooms  4.500000e+00    0.376128 -0.702064
Area Population               6.944910e+04    0.050634 -0.007926
Price                         2.453127e+06   -0.002717 -0.056063
Classe                        1.000000e+00   -0.002400 -1.999994
```

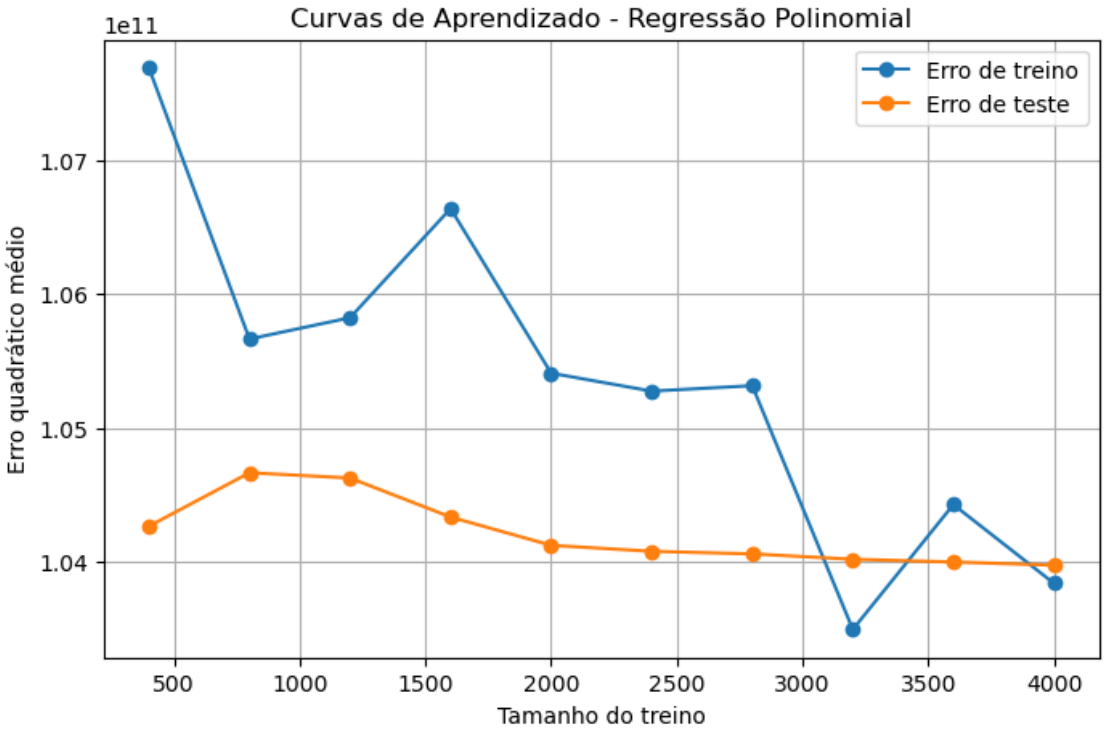
Correlação de Pearson:

```
                             Avg. Area Income  Avg. Area House Age  \
Avg. Area Income                     1.000000            -0.002007
Avg. Area House Age                 -0.002007             1.000000
Avg. Area Number of Rooms           -0.011032            -0.009428
Avg. Area Number of Bedrooms         0.019788             0.006149
Area Population                     -0.016234            -0.018743
Price                                0.639734             0.452543
Classe                               0.505707             0.365857

                             Avg. Area Number of Rooms  \
Avg. Area Income                             -0.011032
Avg. Area House Age                          -0.009428
Avg. Area Number of Rooms                     1.000000
Avg. Area Number of Bedrooms                  0.462695
Area Population                               0.002040
Price                                         0.335664
Classe                                        0.262750

                             Avg. Area Number of Bedrooms  Area Population  \
Avg. Area Income                                 0.019788        -0.016234
Avg. Area House Age                              0.006149        -0.018743
Avg. Area Number of Rooms                        0.462695         0.002040
Avg. Area Number of Bedrooms                     1.000000        -0.022168
Area Population                                 -0.022168         1.000000
Price                                            0.171071         0.408556
Classe                                           0.129665         0.328662

                                Price    Classe
Avg. Area Income             0.639734  0.505707
Avg. Area House Age          0.452543  0.365857
Avg. Area Number of Rooms    0.335664  0.262750
Avg. Area Number of Bedrooms 0.171071  0.129665
Area Population              0.408556  0.328662
Price                        1.000000  0.799461
Classe                       0.799461  1.000000
```

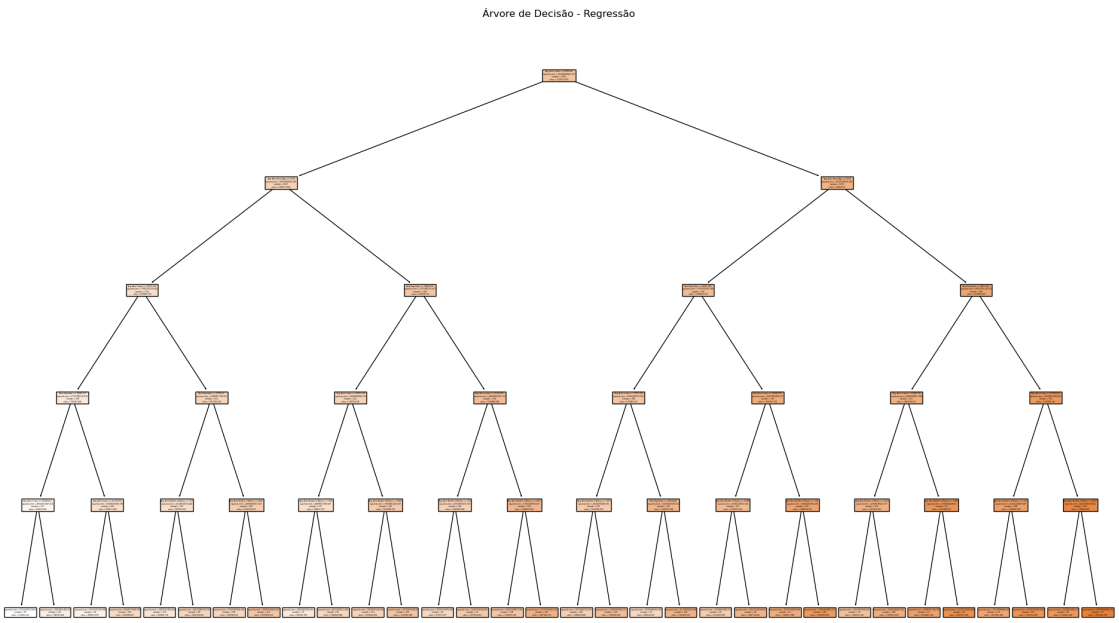


Regressão Linear → R^2: 0.16691790652769944

Regressão Polinomial → R^2: 0.1714864341215081

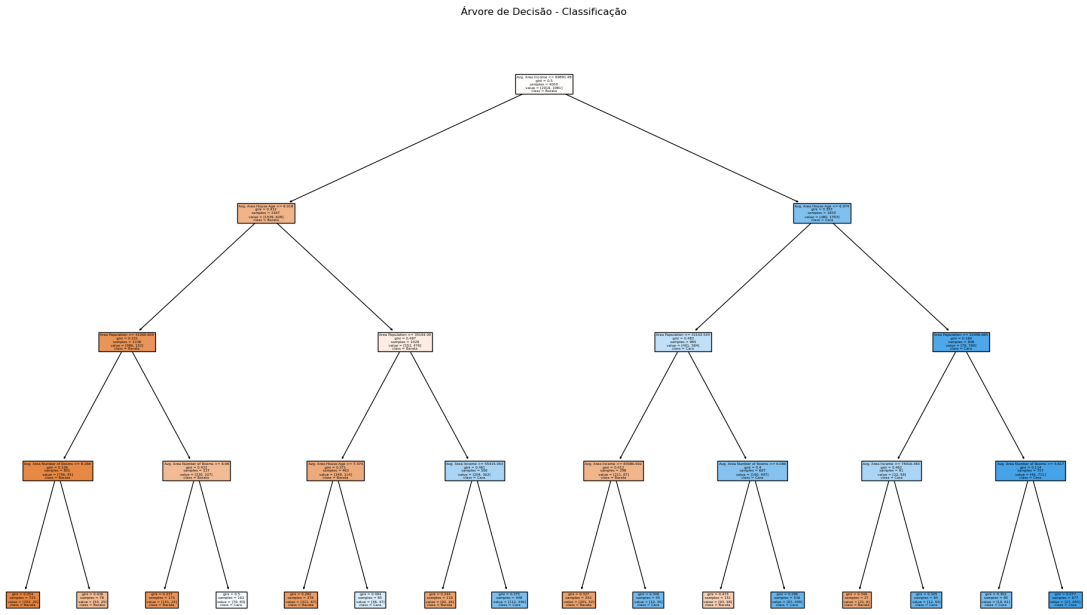



```
Ridge → MSE: 10089716571.16, R²: 0.9180
Lasso → MSE: 10089010509.60, R²: 0.9180
ElasticNet → MSE: 10323866841.71, R²: 0.9161

 Árvore de Regressão → R²: 0.6546816913096776
```



Árvore de Classificação:

	precision	recall	f1-score	support
0	0.79	0.75	0.77	478
1	0.78	0.82	0.80	522
accuracy			0.79	1000
macro avg	0.79	0.79	0.79	1000
weighted avg	0.79	0.79	0.79	1000



[41]:
```
# ================================
#   Importação de bibliotecas
# ================================
import pandas as pd  # manipulação de dados tabulares
import numpy as np  # operações matemáticas e estatísticas
import matplotlib.pyplot as plt  # criação de gráficos
import seaborn as sns  # gráficos estatísticos avançados
from scipy.stats import skew, kurtosis  # medidas de forma (assimetria e␣
 ↪curtose)

# Modelos de regressão e regularização
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.preprocessing import PolynomialFeatures  # transforma dados para␣
 ↪regressão polinomial

# Árvores de decisão
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier,␣
 ↪plot_tree

# Utilitários de modelagem
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.metrics import mean_squared_error, r2_score,␣
 ↪classification_report, confusion_matrix

# ================================
```

```
#  Carregamento e seleção de dados
# ================================
#df = pd.read_csv('nome_do_arquivo.csv')  # substitua pelo nome real do seu␣
 ↪arquivo CSV
df_num = df.select_dtypes(include='number')  # seleciona apenas colunas␣
 ↪numéricas

# ================================
#  Estatísticas descritivas e de forma
# ================================
estatisticas = pd.DataFrame({
    'Média': df_num.mean(),  # média de cada coluna
    'Mediana': df_num.median(),  # mediana
    'Desvio Padrão': df_num.std(),  # dispersão dos dados
    'Variância': df_num.var(),  # variância
    'Mínimo': df_num.min(),  # menor valor
    'Máximo': df_num.max(),  # maior valor
    'Amplitude': df_num.max() - df_num.min(),  # diferença entre máximo e mínimo
    'Assimetria': df_num.apply(skew),  # inclinação da distribuição
    'Curtose': df_num.apply(kurtosis)  # achatamento da distribuição
})
print(" Estatísticas:\n", estatisticas)

# ================================
#  Correlação de Pearson
# ================================
correlacao = df_num.corr()  # calcula correlação entre variáveis numéricas
print("\n Correlação de Pearson:\n", correlacao)

# Heatmap para visualizar correlações
plt.figure(figsize=(12, 8))
sns.heatmap(correlacao, annot=True, cmap='coolwarm')
plt.title('Mapa de Correlação de Pearson')
plt.show()

# ================================
#  Regressão Linear Simples
# ================================
X_lin = df[['Area Population']]  # variável independente
y = df['Price']  # variável dependente
modelo_lin = LinearRegression()  # cria o modelo
modelo_lin.fit(X_lin, y)  # treina o modelo
y_pred_lin = modelo_lin.predict(X_lin)  # faz previsões
print("\n Regressão Linear → R²:", r2_score(y, y_pred_lin))  # avalia o modelo

# ================================
#  Regressão Polinomial + Curvas de Aprendizado
```

```
# ================================
poly = PolynomialFeatures(degree=2)  # transforma para grau 2
X_poly = poly.fit_transform(X_lin)  # aplica transformação
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2,␣
 ↪random_state=42)
modelo_poly = LinearRegression()
modelo_poly.fit(X_train, y_train)
y_pred_poly = modelo_poly.predict(X_test)
print("\n  Regressão Polinomial → R²:", r2_score(y_test, y_pred_poly))

# Curvas de aprendizado
train_sizes, train_scores, test_scores = learning_curve(
    modelo_poly, X_poly, y, cv=5, scoring='neg_mean_squared_error',
    train_sizes=np.linspace(0.1, 1.0, 10)
)
train_scores_mean = -np.mean(train_scores, axis=1)
test_scores_mean = -np.mean(test_scores, axis=1)

# Gráfico das curvas
plt.figure(figsize=(8, 5))
plt.plot(train_sizes, train_scores_mean, 'o-', label='Erro de treino')
plt.plot(train_sizes, test_scores_mean, 'o-', label='Erro de teste')
plt.title('Curvas de Aprendizado - Regressão Polinomial')
plt.xlabel('Tamanho do treino')
plt.ylabel('Erro quadrático médio')
plt.legend()
plt.grid(True)
plt.show()

# ================================
#  Modelos Regularizados (Ridge, Lasso, ElasticNet)
# ================================
X_multi = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of␣
 ↪Rooms',
              'Avg. Area Number of Bedrooms', 'Area Population']]
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X_multi, y, test_size=0.2,␣
 ↪random_state=42)

# Dicionário com os modelos
modelos = {
    'Ridge': Ridge(alpha=1.0),
    'Lasso': Lasso(alpha=0.1),
    'ElasticNet': ElasticNet(alpha=0.1, l1_ratio=0.5)
}

# Treinamento e avaliação
```

```
for nome, modelo in modelos.items():
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    print(f"{nome} → MSE: {mean_squared_error(y_test, y_pred):.2f}, R²:␣
 ↪{r2_score(y_test, y_pred):.4f}")

# ================================
#  Árvore de Decisão - Regressão
# ================================
modelo_tree_reg = DecisionTreeRegressor(max_depth=5, min_samples_split=10,␣
 ↪min_samples_leaf=5, random_state=42)
modelo_tree_reg.fit(X_train, y_train)
y_pred_tree = modelo_tree_reg.predict(X_test)
print("\n Árvore de Regressão → R²:", r2_score(y_test, y_pred_tree))

# Visualização da árvore
plt.figure(figsize=(24, 14))
plot_tree(modelo_tree_reg, feature_names=X_multi.columns, filled=True)
plt.title('Árvore de Decisão - Regressão')
plt.show()

# ================================
#  Árvore de Decisão - Classificação
# ================================
# Cria uma variável binária: 1 se Price > média, 0 caso contrário
df['Classe'] = (df['Price'] > df['Price'].mean()).astype(int)
X_class = X_multi
y_class = df['Classe']
X_train, X_test, y_train, y_test = train_test_split(X_class, y_class,␣
 ↪test_size=0.2, random_state=42)

# Treina o classificador
modelo_tree_clf = DecisionTreeClassifier(max_depth=4, random_state=42)
modelo_tree_clf.fit(X_train, y_train)
y_pred_clf = modelo_tree_clf.predict(X_test)

# Avaliação do modelo de classificação
print("\n Árvore de Classificação:\n", classification_report(y_test,␣
 ↪y_pred_clf))

# Visualização da árvore de classificação
plt.figure(figsize=(24, 14))
plot_tree(modelo_tree_clf, feature_names=X_class.columns,␣
 ↪class_names=['Barata', 'Cara'], filled=True)
plt.title('Árvore de Decisão - Classificação')
plt.show()
```

Estatísticas:

	Média	Mediana	Desvio Padrão
Avg. Area Income	6.858311e+04	6.880429e+04	10657.991214
Avg. Area House Age	5.977222e+00	5.970429e+00	0.991456
Avg. Area Number of Rooms	6.987792e+00	7.002902e+00	1.005833
Avg. Area Number of Bedrooms	3.981330e+00	4.050000e+00	1.234137
Area Population	3.616352e+04	3.619941e+04	9925.650114
Price	1.232073e+06	1.232669e+06	353117.626584
Classe	5.006000e-01	1.000000e+00	0.500050

	Variância	Mínimo	Máximo
Avg. Area Income	1.135928e+08	17796.631190	1.077017e+05
Avg. Area House Age	9.829854e-01	2.644304	9.519088e+00
Avg. Area Number of Rooms	1.011700e+00	3.236194	1.075959e+01
Avg. Area Number of Bedrooms	1.523095e+00	2.000000	6.500000e+00
Area Population	9.851853e+07	172.610686	6.962171e+04
Price	1.246921e+11	15938.657920	2.469066e+06
Classe	2.500496e-01	0.000000	1.000000e+00

	Amplitude	Assimetria	Curtose
Avg. Area Income	8.990512e+04	-0.033710	0.044329
Avg. Area House Age	6.874784e+00	-0.007212	-0.084554
Avg. Area Number of Rooms	7.523394e+00	-0.040984	-0.075777
Avg. Area Number of Bedrooms	4.500000e+00	0.376128	-0.702064
Area Population	6.944910e+04	0.050634	-0.007926
Price	2.453127e+06	-0.002717	-0.056063
Classe	1.000000e+00	-0.002400	-1.999994

Correlação de Pearson:

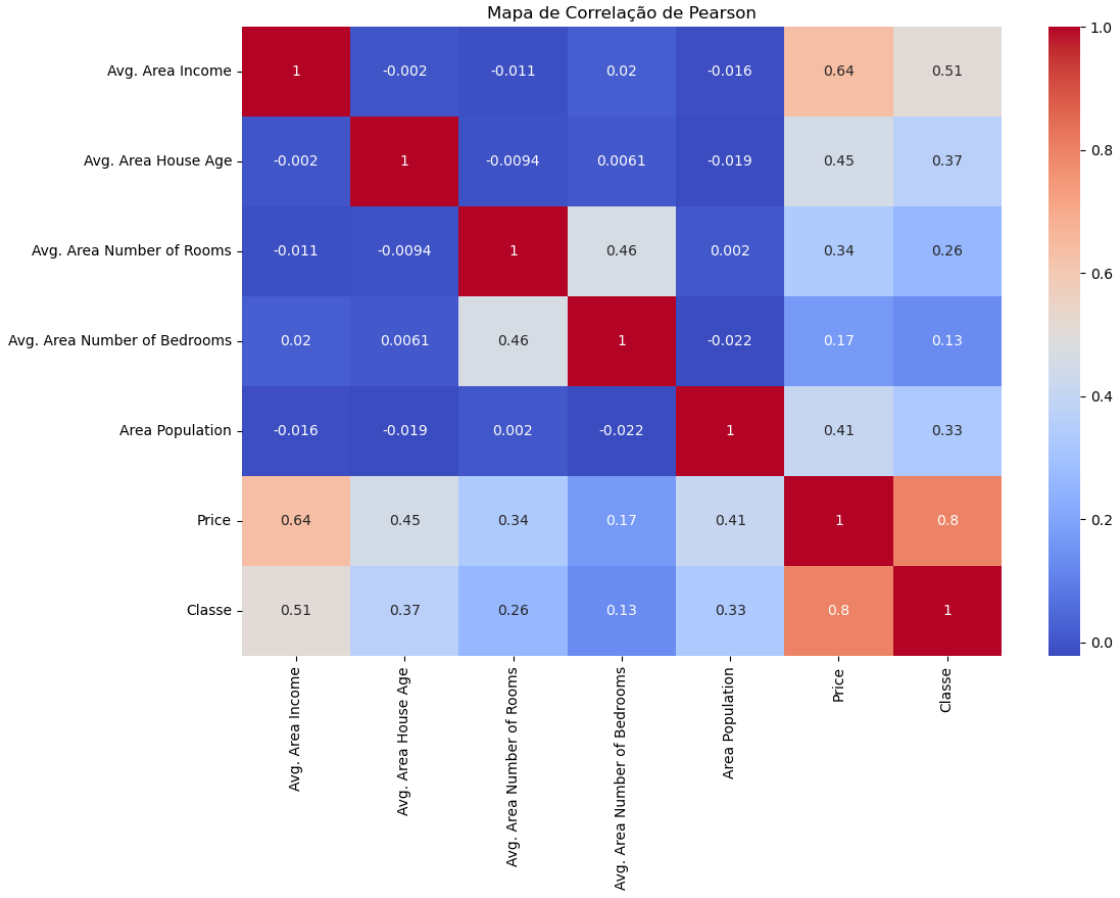
	Avg. Area Income	Avg. Area House Age
Avg. Area Income	1.000000	-0.002007
Avg. Area House Age	-0.002007	1.000000
Avg. Area Number of Rooms	-0.011032	-0.009428
Avg. Area Number of Bedrooms	0.019788	0.006149
Area Population	-0.016234	-0.018743
Price	0.639734	0.452543
Classe	0.505707	0.365857

	Avg. Area Number of Rooms
Avg. Area Income	-0.011032
Avg. Area House Age	-0.009428
Avg. Area Number of Rooms	1.000000
Avg. Area Number of Bedrooms	0.462695
Area Population	0.002040
Price	0.335664
Classe	0.262750

	Avg. Area Number of Bedrooms	Area Population

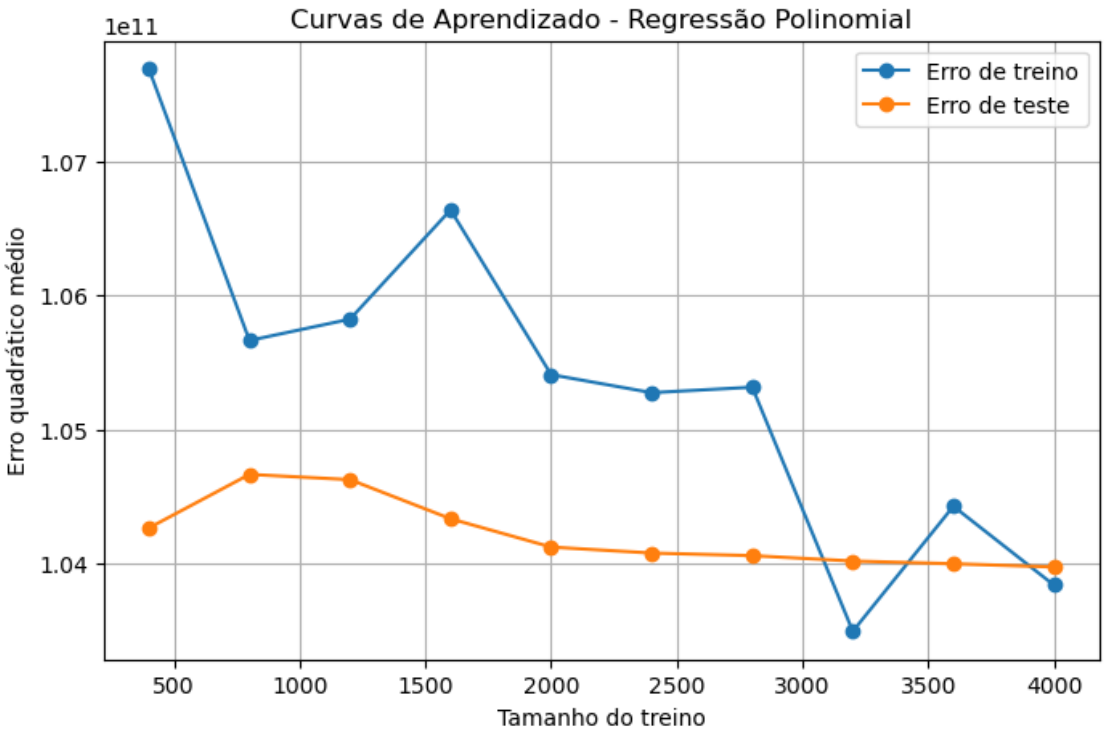
Avg. Area Income	0.019788	-0.016234
Avg. Area House Age	0.006149	-0.018743
Avg. Area Number of Rooms	0.462695	0.002040
Avg. Area Number of Bedrooms	1.000000	-0.022168
Area Population	-0.022168	1.000000
Price	0.171071	0.408556
Classe	0.129665	0.328662

	Price	Classe
Avg. Area Income	0.639734	0.505707
Avg. Area House Age	0.452543	0.365857
Avg. Area Number of Rooms	0.335664	0.262750
Avg. Area Number of Bedrooms	0.171071	0.129665
Area Population	0.408556	0.328662
Price	1.000000	0.799461
Classe	0.799461	1.000000



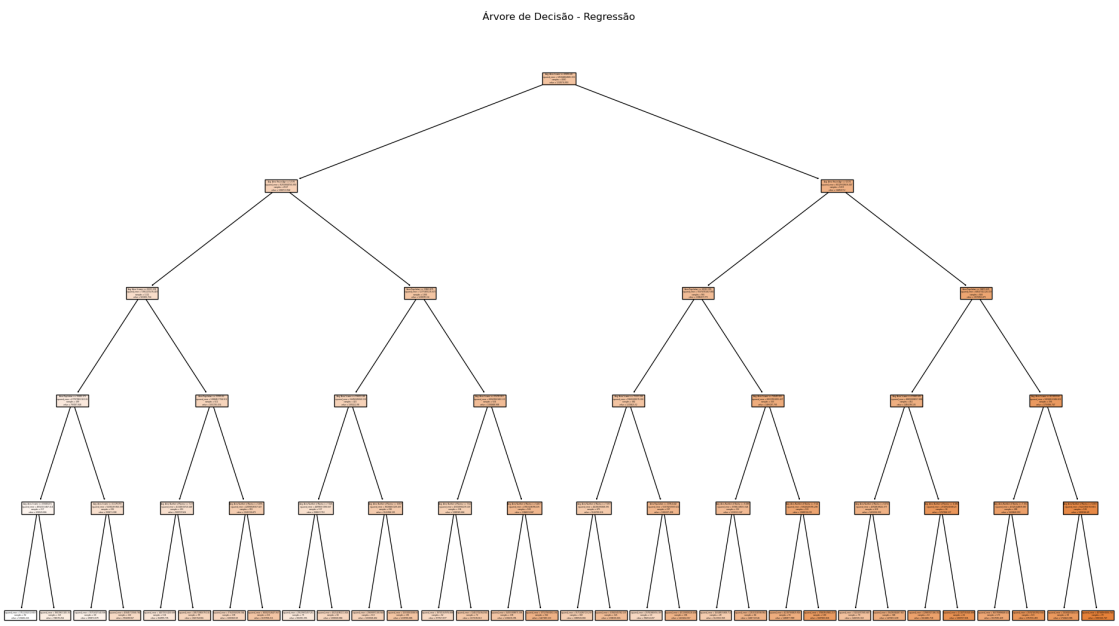
Regressão Linear → R^2: 0.16691790652769944

Regressão Polinomial → R^2: 0.1714864341215081



Ridge → MSE: 10089716571.16, R^2: 0.9180
Lasso → MSE: 10089010509.60, R^2: 0.9180
ElasticNet → MSE: 10323866841.71, R^2: 0.9161

Árvore de Regressão → R^2: 0.6546816913096776



Árvore de Classificação:

	precision	recall	f1-score	support
0	0.79	0.75	0.77	478
1	0.78	0.82	0.80	522
accuracy			0.79	1000
macro avg	0.79	0.79	0.79	1000
weighted avg	0.79	0.79	0.79	1000

Árvore de Decisão - Classificação