Documento AC 2

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler


dados = pd.read csv("/content/PRECOS_CASAS.csv")
```

R: Esse código resume a preparação do ambiente de trabalho e a importação das bibliotecas necessárias para análise de dados e modelagem de machine learning, bem como o carregamento de dados de um arquivo CSV para um DataFrame do Pandas.

```
dados.isnull().sum()
dados.dropna(inplace=True)
```

R: Esse código resume a identificação e remoção de linhas com valores nulos do DataFrame ‘`dados`’. Ele primeiro calcula o número de valores nulos em cada coluna e, em seguida, remove as linhas que contêm pelo menos um valor nulo do DataFrame.

```
Casa        0
Preco       0
Area        0
Quartos     0
Banheiros   0
Ofertas     0
Tijolo      0
Bairro      0
dtype: int64
```

```
scaler = StandardScaler()



dados["Area"]  =  scaler.fit_transform(dados["Area"].values.reshape(-1, 1))

dados["Quartos"]                                                    = scaler.fit_transform(dados["Quartos"].values.reshape(-1, 1))

dados["Banheiros"]                                                  = scaler.fit_transform(dados["Banheiros"].values.reshape(-1, 1))
```

R:Esse código aplica a padronização (standardization) às colunas "Area", "Quartos" e "Banheiros" do DataFrame `dados`. Ele cria um objeto `StandardScaler`, que é usado para ajustar e transformar os dados, garantindo que tenham média zero e desvio padrão unitário. As transformações são aplicadas às colunas individualmente usando o método `fit_transform()` do `StandardScaler`, resultando em valores padronizados para essas colunas.

```
X = dados[["Area", "Quartos", "Banheiros"]]
y = dados["Preco"]

X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y, test_size=0.2, random_state=42)
```

R: Esse código utiliza a classe `StandardScaler` do scikit-learn para padronizar os dados das colunas "Area", "Quartos" e "Banheiros" do DataFrame `dados`. A padronização é feita para garantir que essas colunas tenham média zero e desvio padrão unitário. Cada coluna é tratada separadamente, com o scaler sendo ajustado aos dados e em seguida aplicando a transformação de padronização. Isso é útil

para preparar os dados antes de alimentá-los em modelos de machine learning, garantindo que eles estejam na mesma escala.

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

R: Essas linhas de código criam e treinam um modelo de regressão linear.

1. `modelo = LinearRegression()`: Cria uma instância do modelo de regressão linear.
2. `modelo.fit(X_train, y_train)`: Ajusta o modelo aos dados de treinamento (`X_train` contém os recursos e `y_train` contém os rótulos), permitindo que o modelo aprenda os padrões nos dados.

```
LinearRegression
LinearRegression()
```

```
y_pred = modelo.predict(X_test)

rmse = mean_squared_error(y_test, y_pred)
print(f"RMSE: {rmse:.2f}")

r2 = r2_score(y_test, y_pred)
print(f"R²: {r2:.2f}")
```

R: Essas linhas de código calculam e imprimem duas métricas de avaliação do desempenho do modelo de regressão linear nos dados de teste:

1. RMSE (Root Mean Squared Error): Fornece uma medida da dispersão dos erros do modelo, indicando o quão bem as previsões correspondem aos valores reais dos dados de teste. Quanto menor o RMSE, mais preciso é o modelo.
2. R² (Coefficient of Determination): Indica a qualidade do ajuste do modelo aos dados de teste, representando a proporção da variabilidade nos rótulos que é explicada pelo modelo. Um valor de R² mais próximo de 1 indica um bom ajuste do modelo aos dados.

```
RMSE: 320149938.23
R²: 0.46
```

```
print(f"Coeficientes: {modelo.coef_}")
```

R: Essa linha de código imprime os coeficientes estimados pelo modelo de regressão linear. Os coeficientes representam a mudança média na variável de resposta para uma unidade de mudança na variável de entrada, mantendo todas as outras variáveis constantes.

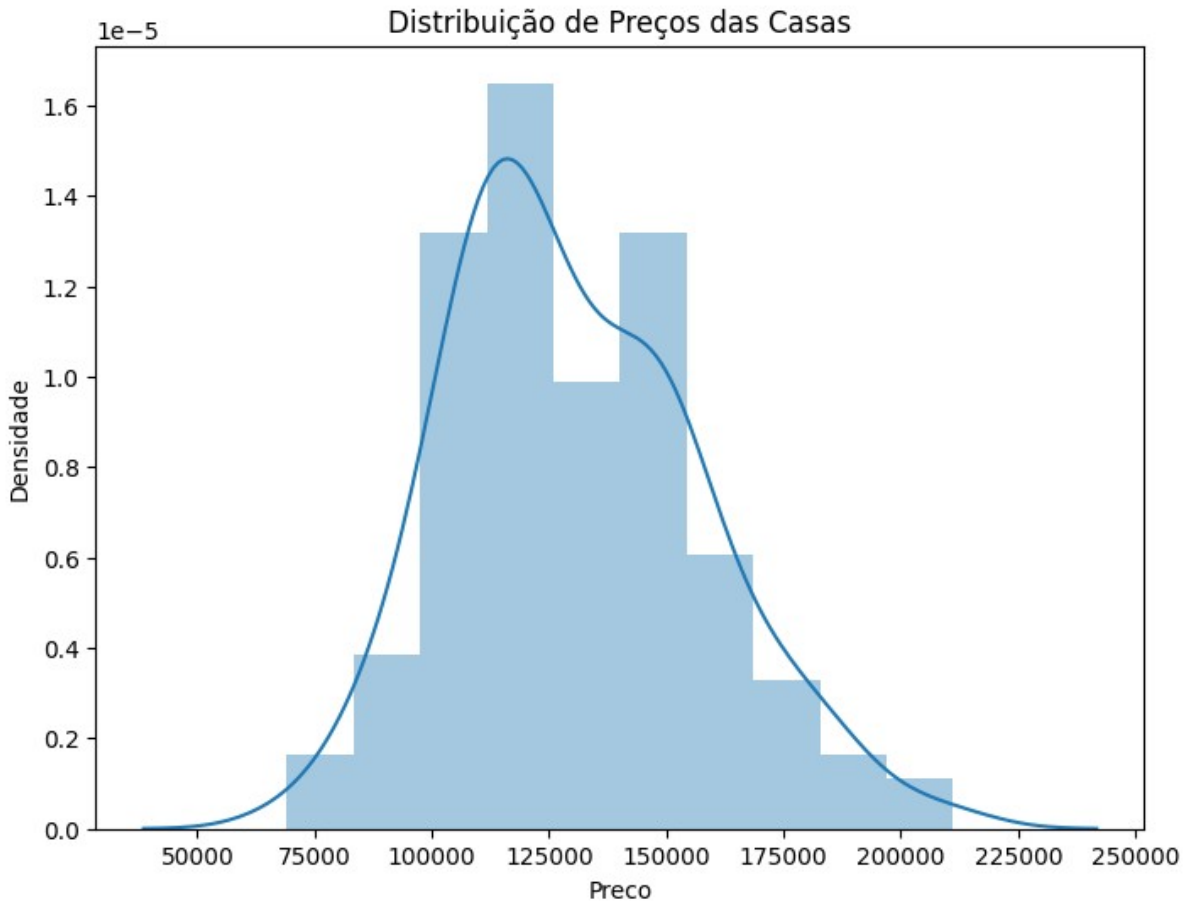
```
Coeficientes: [8734.83761539 7432.13195417 6708.54485934]
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

R: Essas linhas de código importam as bibliotecas Matplotlib e Seaborn, usadas para visualização de dados em Python. A Matplotlib é amplamente utilizada para criar gráficos, enquanto o Seaborn fornece uma interface simplificada para criar gráficos estatísticos mais atraentes e informativos, baseados no Matplotlib. Essas bibliotecas são comumente utilizadas durante a análise exploratória de dados e na criação de visualizações para comunicar resultados e insights.

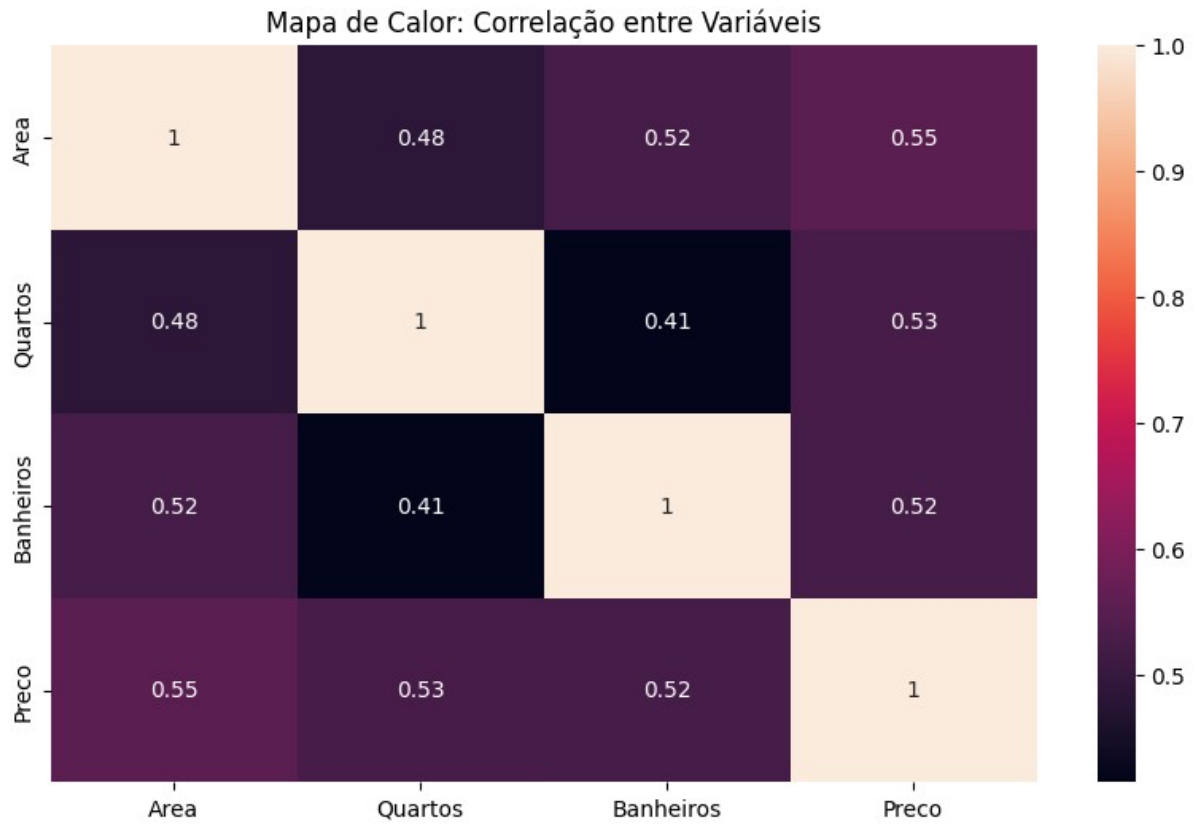
```
plt.figure(figsize=(8, 6))
sns.distplot(dados["Preco"])
plt.xlabel("Preco")
plt.ylabel("Densidade")
plt.title("Distribuição de Preços das Casas")
plt.show()
```

R: Essas linhas de código criam e exibem um histograma da distribuição dos preços das casas. O histograma é gerado usando a biblioteca Seaborn para plotagem e a biblioteca Matplotlib para configuração do tamanho da figura. O eixo x é rotulado como "Preço", o eixo y como "Densidade", e o título do gráfico é definido como "Distribuição de Preços das Casas".




```
plt.figure(figsize=(10, 6))
sns.heatmap(dados[["Area",  "Quartos",  "Banheiros",  "Preco"]].corr(),
annot=True)
plt.title("Mapa de Calor: Correlação entre Variáveis")
plt.show()
```

R: O código gera um mapa de calor que mostra a correlação entre as variáveis "Area", "Quartos", "Banheiros" e "Preco" do DataFrame `dados`. O tamanho da figura é definido como 10 unidades de largura por 6 unidades de altura. O mapa de calor exibe valores de correlação entre -1 e 1, onde cores mais claras indicam correlações mais fortes e cores mais escuras indicam correlações mais fracas. O parâmetro `annot=True` adiciona os valores de correlação nas células do mapa de calor.


Mapa de Calor: Correlação entre Variáveis
Area
Quartos
Banheiros
Preco
1
0.48
0.52
0.55
0.48
1
0.41
0.53
0.52
0.41
1
0.52
0.55
0.53
0.52
1
Area
Quartos
Banheiros
Preco
1.0
0.9
0.8
0.7
0.6
0.5