MINERAÇÃO DE PREÇOS DE CASAS

Relatório

1-

```
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")
```

Explicação:
Neste trecho de código, estamos importando as bibliotecas essenciais para a análise de dados e construção do modelo de regressão linear:

- pandas: para manipulação e análise de dados.
- numpy: para operações matemáticas.
- train_test_split: para dividir o conjunto de dados em treinamento e teste.
- LinearRegression: para criar e treinar o modelo de regressão linear.
- mean_squared_error e r2_score: para avaliar o desempenho do modelo.
- StandardScaler: para padronização dos dados.

2-

```
dados.isnull().sum()
```

Resposta:

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

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

Explicação:

- dados.isnull().sum(): Conta o número de valores nulos em cada coluna do DataFrame.
- dados.dropna(inplace=True): Remove todas as linhas que contêm valores nulos do DataFrame.

3-
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))

Explicação:

Explicação:

Neste trecho, estamos padronizando as colunas Area, Quartos e Banheiros usando StandardScaler, o que transforma os dados para que tenham média 0 e desvio padrão 1. Isso ajuda a normalizar os dados, melhorando o desempenho dos algoritmos de machine learning.

4-

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)
```

Explicação:

- X contém as variáveis independentes (Área, Quartos, Banheiros).
- y contém a variável dependente (Preço).
- train_test_split: Divide os dados em conjuntos de treinamento (80%) e teste (20%).

5-

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

Resposta:
LinearRegression
LinearRegression()

Explicação:

- modelo = LinearRegression(): Cria uma instância do modelo de regressão linear.

- modelo.fit(X_train, y_train): Treina o modelo usando os dados de treinamento.

6-

```
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}")
```

Resposta:
RMSE: 320149938.23
R²: 0.46

Explicação:

- y_pred = modelo.predict(X_test): Faz previsões usando os dados de teste.
- mean_squared_error(y_test, y_pred): Calcula o erro quadrático médio (RMSE), que indica a média das diferenças quadradas entre valores previstos e observados.
- r2_score(y_test, y_pred): Calcula o coeficiente de determinação (R²), que indica a proporção da variabilidade dos dados explicada pelo modelo.

8-

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

Resposta:
Coeficientes: [8734.83761539 7432.13195417 6708.54485934]

Explicação:

- modelo.coef_: Exibe os coeficientes das variáveis independentes do modelo, indicando a importância de cada variável na previsão do preço.

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

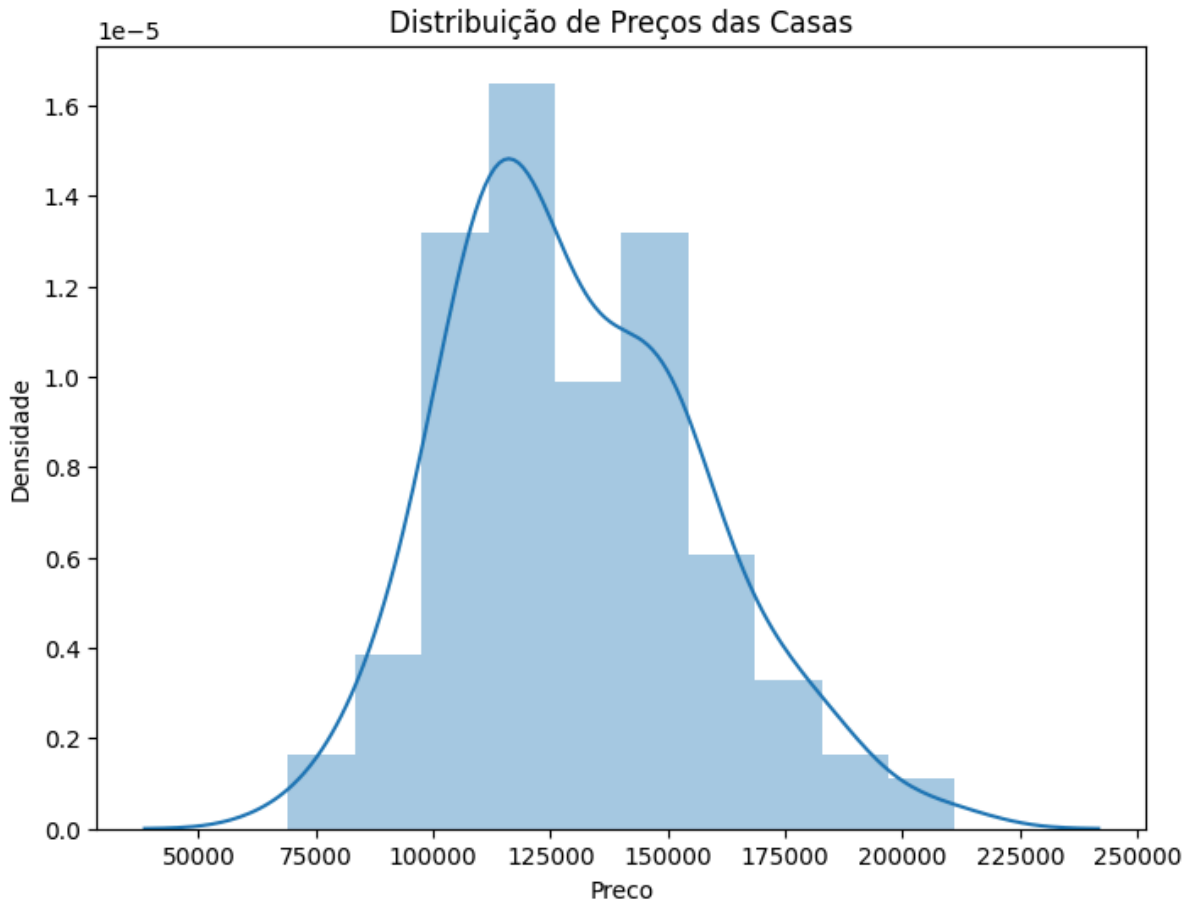
Explicação: Estamos importando matplotlib e seaborn para visualização dos dados.

Essas bibliotecas ajudam a criar gráficos e plots de alta qualidade.

9-

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

Explicação:



- Cria uma figura para o gráfico.
- sns.distplot(dados["Preco"]): Plota a distribuição dos preços das casas.
- Adiciona rótulos aos eixos e título ao gráfico.
- plt.show(): Exibe o gráfico.

Gráfico Gerado:

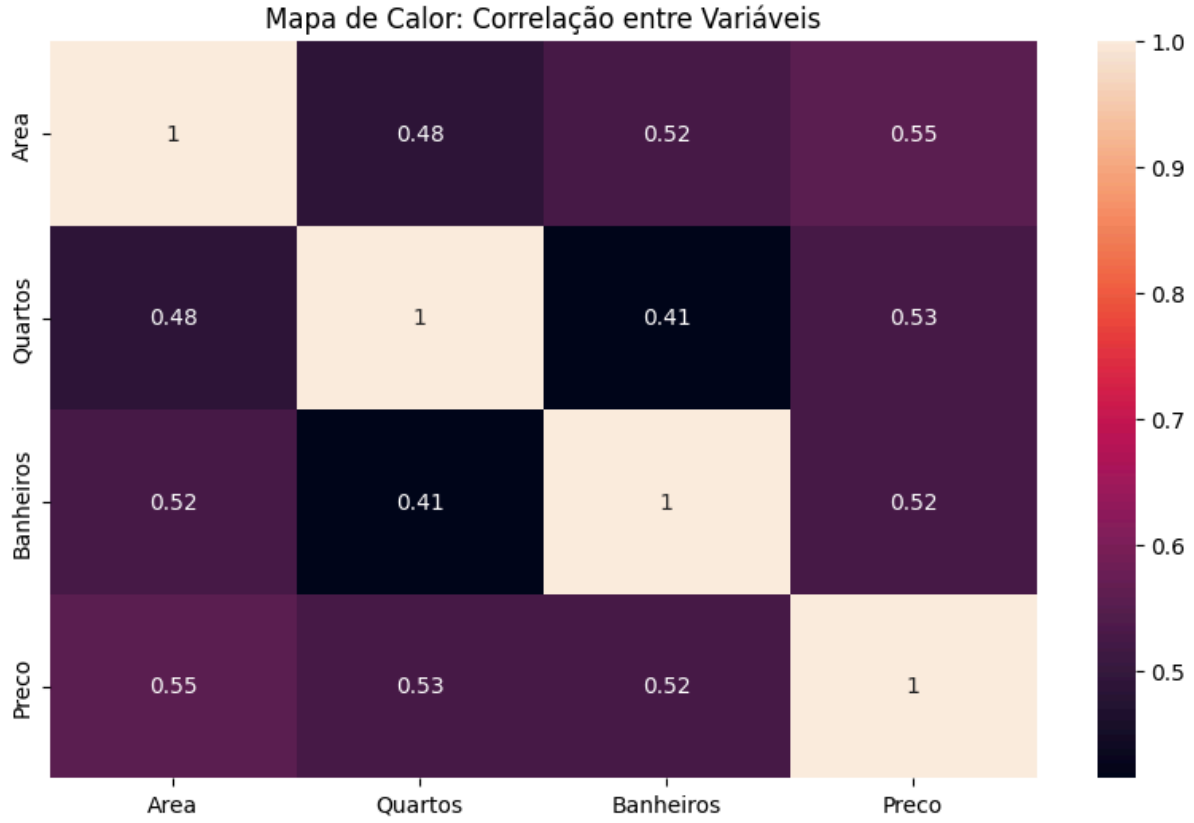
- O gráfico será um histograma com uma curva de densidade mostrando como os preços das casas estão distribuídos.

Explicação do Gráfico de Distribuição:

- Um gráfico de distribuição mostra a frequência de diferentes valores de uma variável. No caso dos preços das casas, ele mostra como os preços estão distribuídos, indicando a concentração de casas em diferentes faixas de preço.

10-

```
# Correlação entre Variáveis
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()
```



Explicação:

- Cria uma figura para o gráfico.
- `sns.heatmap(dados[["Area", "Quartos", "Banheiros", "Preco"]].corr(), annot=True)`: Plota um mapa de calor das correlações entre as variáveis, onde `annot=True` adiciona os valores de correlação no gráfico.
- Adiciona título ao gráfico.
- `plt.show()`: Exibe o gráfico.

Explicação do Mapa de Calor:O mapa de calor mostra a correlação entre diferentes variáveis. Correlações próximas de 1 ou -1 indicam uma forte relação

linear positiva ou negativa, respectivamente, enquanto correlações próximas de 0 indicam pouca ou nenhuma relação linear.