RELATÓRIO – MINERAÇÃO DE PREÇOS DE CASAS

Grupo:

Julio Cesar Ribeiro - RA: 37765
João Victor Santos Carmo - RA: 38195
Leandro Miranda Freitas - RA: 37003
Pedro Da Silva Valença - RA: 40236
Thyago Vinicius Félix Santana - RA: 38451

CÓDIGO 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")
```

Esse código em Python é utilizado para realizar análise e modelagem de dados de preços de casas. Ele começa importando as bibliotecas necessárias, incluindo pandas para manipulação de dados, numpy para operações numéricas, e diversas ferramentas do pacote scikit-learn (`sklearn`) para modelagem de machine learning, como divisão dos dados em conjuntos de treinamento e teste, regressão linear, e métricas de avaliação de modelo. Em seguida, o código carrega os dados de um arquivo CSV chamado "PRECOS_CASAS.csv" e armazena-os em um DataFrame chamado `dados`.

CÓDIGO 2:

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

Este código verifica a quantidade de valores nulos em cada coluna do DataFrame chamado dados. Em seguida, remove todas as linhas que contenham valores nulos desse DataFrame.

CÓDIGO 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))
```

Esse código utiliza o `StandardScaler` para padronizar as variáveis numéricas do DataFrame `dados`, incluindo "Area", "Quartos" e "Banheiros". Ele ajusta o scaler aos dados e, em seguida, transforma essas colunas para terem média zero e desvio padrão unitário.

CÓDIGO 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)
```

Esse código divide os dados em conjuntos de treinamento e teste para serem usados em um modelo de machine learning. Ele seleciona as variáveis preditoras (features) do DataFrame `dados`, que são "Area", "Quartos" e "Banheiros", e as armazena em uma variável chamada `X`. Em seguida, seleciona a variável alvo (target), que é "Preco", e a armazena em uma variável chamada `y`.

Depois, utiliza a função `train_test_split` para dividir os dados em conjuntos de treinamento e teste, onde `X_train` e `y_train` representam as features e target do conjunto de treinamento, respectivamente, e `X_test` e `y_test` representam as features e target do conjunto de teste, respectivamente. O parâmetro `test_size=0.2` indica que 20% dos dados serão reservados para o conjunto de teste, enquanto o restante será utilizado para treinamento. O parâmetro `random_state=42` define uma semente para garantir a reprodutibilidade dos resultados.

CÓDIGO 5:

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

Este código cria um modelo de regressão linear utilizando a classe `LinearRegression` do pacote scikit-learn (`sklearn`). Em seguida, ajusta o modelo aos dados de treinamento (`X_train` e `y_train`) utilizando o método `fit()`. Ou seja, o modelo está sendo treinado para aprender os padrões nos dados de treinamento, de forma a realizar previsões precisas sobre a variável alvo (`y_train`) com base nas variáveis preditoras (`X_train`).

CÓDIGO 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}")
```

Este código utiliza o modelo de regressão linear treinado para fazer previsões sobre os dados de teste (`X_test`) e armazena essas previsões na variável `y_pred`. Em seguida, calcula a raiz do erro quadrático médio (RMSE) entre as previsões (`y_pred`) e os valores reais dos dados de teste (`y_test`). O RMSE é uma medida da diferença entre os valores previstos e os valores reais, indicando a precisão do modelo. Depois, calcula o coeficiente de determinação (R^2), que é uma medida da qualidade

das previsões do modelo em relação à variabilidade dos dados. Ambas as métricas são impressas na tela para avaliar o desempenho do modelo.

Resposta:

```
RMSE: 320149938.23
R²: 0.46
```

CÓDIGO 7:

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

Este código imprime os coeficientes do modelo de regressão linear treinado. Os coeficientes representam o peso atribuído a cada variável preditora no modelo para prever a variável alvo. A expressão `{modelo.coef_}` acessa os coeficientes do modelo `modelo`, e a função `print()` exibe esses coeficientes na tela, precedidos pelo texto "Coeficientes:".

Resposta:

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

CÓDIGO 8:

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

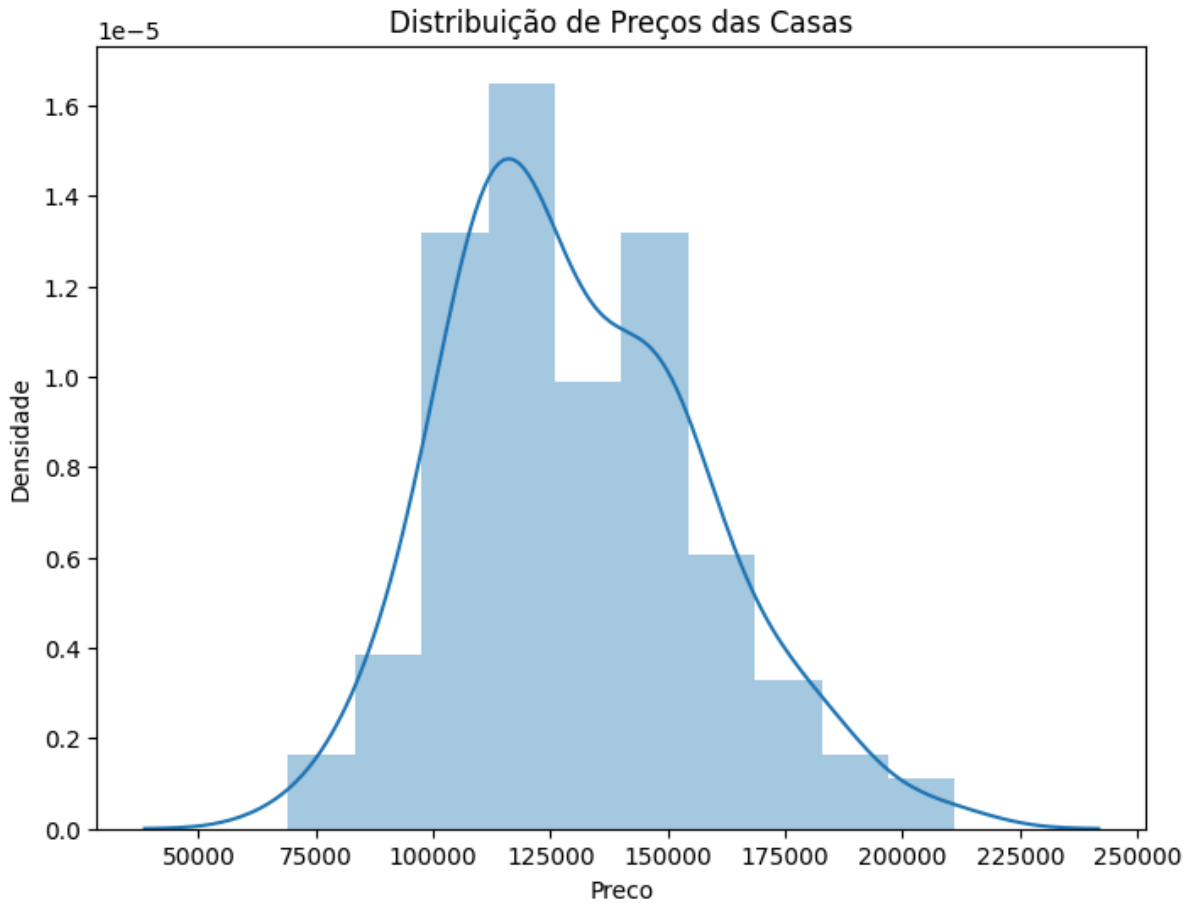
Este código importa as bibliotecas `matplotlib.pyplot` e `seaborn` para visualização de dados em Python. Essas bibliotecas são frequentemente utilizadas para criar gráficos e visualizações estatísticas de maneira eficiente e esteticamente agradável.

CÓDIGO 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()
```

Este código cria um histograma e uma estimativa de densidade de kernel para visualizar a distribuição dos preços das casas. Ele utiliza as bibliotecas `matplotlib.pyplot` (abreviada como `plt`) e `seaborn` (abreviada como `sns`) para gerar o gráfico. O tamanho da figura é definido como 8x6 polegadas. O eixo x é rotulado como "Preço", o eixo y é rotulado como "Densidade" e o título do gráfico é "Distribuição de Preços das Casas". Finalmente, o gráfico é exibido com `plt.show()`.

Resposta:



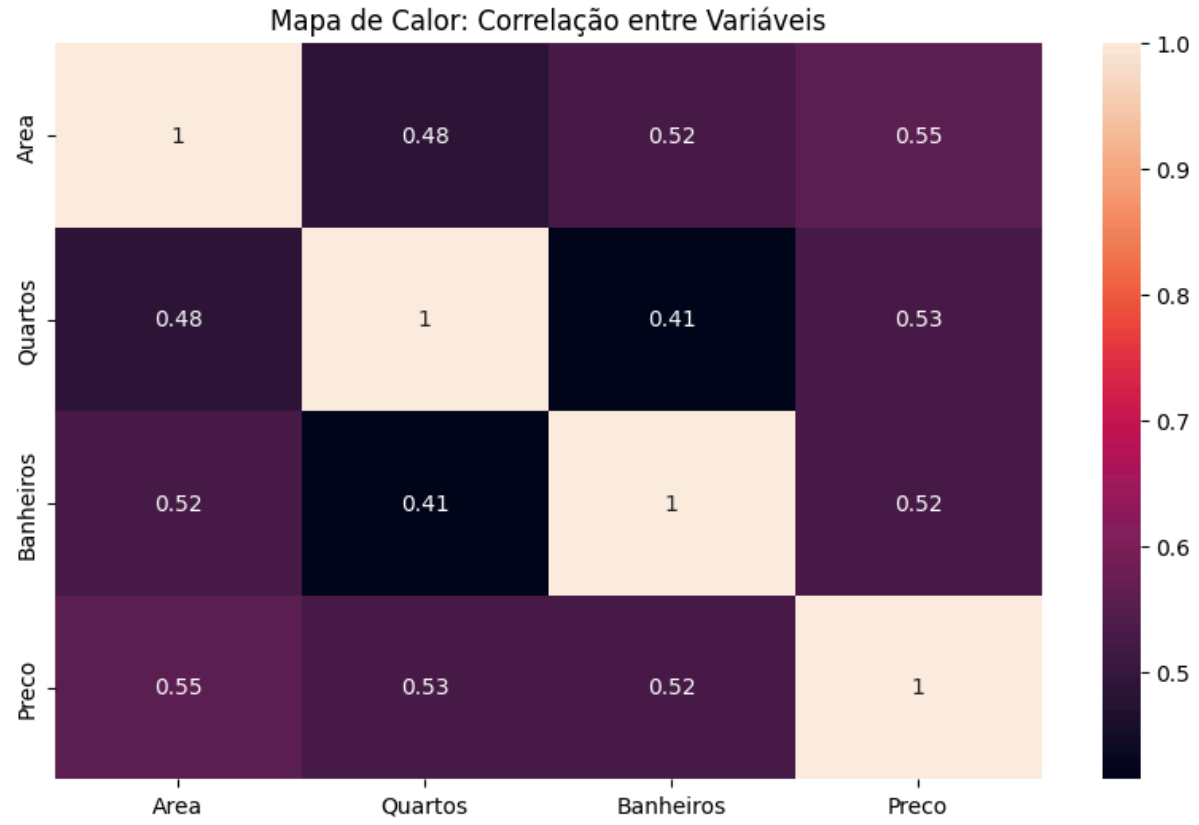
CÓDIGO 10:

```
# 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()
```

Este código cria um mapa de calor para visualizar a correlação entre as variáveis "Área", "Quartos", "Banheiros" e "Preço" do DataFrame `dados`. Ele utiliza as bibliotecas `matplotlib.pyplot` (abreviada como `plt`) e `seaborn` (abreviada como `sns`) para gerar o gráfico. O tamanho da figura é definido como 10x6 polegadas. O mapa de calor exibe os coeficientes de correlação entre as variáveis, onde valores mais altos indicam uma correlação mais forte. Os valores numéricos das correlações são exibidos no gráfico (`annot=True`). O título do gráfico

é "Mapa de Calor: Correlação entre Variáveis". Finalmente, o gráfico é exibido com `plt.show()`.

Resposta:



Relatório sobre a Importância da Utilização de Ciência de Dados nas Empresas

A Ciência de Dados é crucial para as empresas modernas, oferecendo insights valiosos a partir de dados para informar decisões estratégicas e operacionais. Neste relatório, destacamos a importância da Ciência de Dados, o papel do Aprendizado de Máquina e da Mineração de Dados, além da relevância do Web Scraping no contexto de Big Data.

Importância da Ciência de Dados:

Baseada em dados: Permite decisões mais informadas e eficazes, compreendendo o mercado, os clientes e os processos internos.

Identificação de padrões: Ajuda a detectar tendências de mercado e comportamentos dos consumidores, impulsionando estratégias de negócio.

Otimização de processos: Identifica eficiências operacionais e reduz custos ao eliminar gargalos e desperdícios.

Experiência do cliente: Personaliza produtos e serviços para oferecer uma experiência mais relevante e satisfatória.

Gestão de riscos: Detecta fraudes e gerencia riscos de forma proativa.

Aprendizado de Máquina na Ciência de Dados:

O Aprendizado de Máquina desenvolve modelos que aprendem com dados históricos para fazer previsões e tomar decisões automatizadas. Seu objetivo é criar modelos preditivos e descritivos para aplicação em diversas áreas, como previsões de vendas e recomendações personalizadas.

Mineração de Dados:

A Mineração de Dados descobre padrões e conhecimentos úteis em grandes conjuntos de dados. Utiliza técnicas como classificação, regressão, agrupamento, associação e análise de anomalias para extrair informações valiosas para tomada de decisões.

Web Scraping e Big Data:

O Web Scraping automatiza a coleta de dados de websites, enriquecendo conjuntos de dados e fornecendo informações adicionais para análise. É particularmente útil em ambientes de Big Data, onde múltiplas fontes de dados são combinadas para obter insights abrangentes.

Conclusão:

A Ciência de Dados é essencial para empresas que buscam vantagem competitiva. Investir em técnicas como Aprendizado de Máquina, Mineração de Dados e Web Scraping permite aproveitar ao máximo os dados disponíveis, gerando insights valiosos que impulsionam o crescimento e o sucesso organizacional.