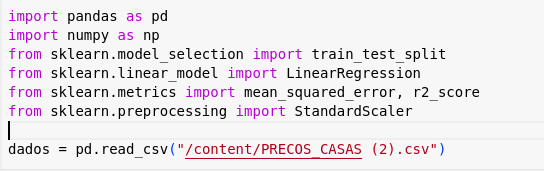
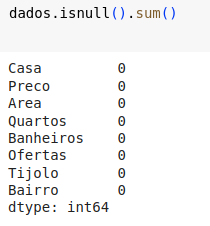
**RELATÓRIO - MINERAÇÃO DE PREÇOS DE CASAS**

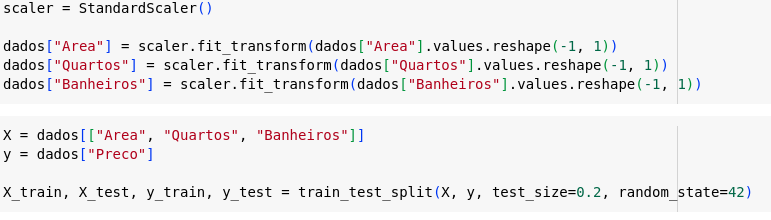


Esse trecho de código em Python está realizando várias operações relacionadas à análise de dados e modelagem preditiva usando a biblioteca pandas e ferramentas da biblioteca scikit-learn (sklearn). Vamos analisar cada linha:

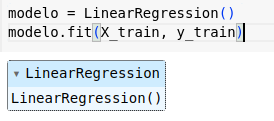
1. import pandas as pd: Esta linha importa a biblioteca pandas com o alias pd, que é comumente usado como convenção.
2. import numpy as np: Esta linha importa a biblioteca NumPy com o alias np. NumPy é frequentemente usado para operações numéricas e manipulação de arrays em Python.
3. from sklearn.model\_selection import train\_test\_split: Esta linha importa a função train\_test\_split do módulo model\_selection do scikit-learn. Esta função é usada para dividir um conjunto de dados em conjuntos de treinamento e teste, o que é essencial para avaliar o desempenho de modelos de aprendizado de máquina.
4. from sklearn.linear\_model import LinearRegression: Esta linha importa a classe LinearRegression do módulo linear\_model do scikit-learn. Essa classe é usada para realizar regressão linear, um método comum de aprendizado supervisionado para modelar a relação entre uma variável dependente e uma ou mais variáveis independentes.
5. from sklearn.metrics import mean\_squared\_error, r2\_score: Esta linha importa as funções mean\_squared\_error e r2\_score do módulo metrics do scikit-learn. Essas funções são usadas para avaliar a qualidade das previsões feitas por modelos de regressão, fornecendo métricas como o erro quadrático médio (MSE) e o coeficiente de determinação (R²).
6. from sklearn.preprocessing import StandardScaler: Esta linha importa a classe StandardScaler do módulo preprocessing do scikit-learn. Esta classe é usada para padronizar recursos, o que é comumente usado em muitos algoritmos de aprendizado de máquina para melhorar o desempenho e a convergência do modelo.
7. dados = pd.read\_csv("/content/PRECOS\_CASAS (2).csv"): Esta linha lê um arquivo CSV chamado "PRECOS\_CASAS (2).csv" e armazena os dados em um DataFrame do pandas chamado dados. Presumivelmente, este arquivo CSV contém dados sobre preços de casas.



Este comando dados.isnull().sum() calcula a contagem de valores nulos em cada coluna do DataFrame dados e retorna essas contagens como uma série pandas.

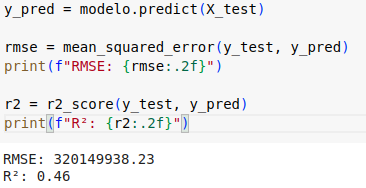
Esse trecho de código está realizando pré-processamento nos dados antes de treinar um modelo de regressão linear. Aqui está o que cada linha faz:

1. scaler = StandardScaler(): Cria uma instância do objeto StandardScaler, que será usada para padronizar os dados.
2. dados["Area"] = scaler.fit\_transform(dados["Area"].values.reshape(-1, 1)): Padroniza a coluna "Area" dos dados, aplicando a transformação de padronização usando o método fit\_transform do objeto StandardScaler. O método reshape(-1, 1) é usado para garantir que os dados tenham a forma correta para serem transformados.
3. dados["Quartos"] = scaler.fit\_transform(dados["Quartos"].values.reshape(-1, 1)): Padroniza a coluna "Quartos" dos dados da mesma maneira que a coluna "Area".
4. dados["Banheiros"] = scaler.fit\_transform(dados["Banheiros"].values.reshape(-1, 1)): Padroniza a coluna "Banheiros" dos dados da mesma maneira que as colunas "Area" e "Quartos".
5. X = dados[["Area", "Quartos", "Banheiros"]]: Cria uma matriz de características X que contém as colunas padronizadas "Area", "Quartos" e "Banheiros".
6. y = dados["Preco"]: Cria um vetor de destino y que contém os valores da coluna "Preco", que é a variável alvo.
7. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42): Divide os dados em conjuntos de treinamento (X\_train e y\_train) e teste (X\_test e y\_test) usando a função train\_test\_split do scikit-learn. O parâmetro test\_size=0.2 indica que 20% dos dados serão usados como conjunto de teste, enquanto o restante será usado como conjunto de treinamento. O parâmetro random\_state=42 define a semente aleatória para garantir que a divisão seja reproduzível.



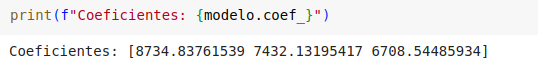
Este trecho de código cria uma instância do modelo de regressão linear (LinearRegression), ajusta o modelo aos dados de treinamento e o treina. Aqui está o que cada linha faz:

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. A função fit() é usada para treinar o modelo, onde X\_train são as características de treinamento e y\_train são os valores-alvo correspondentes. O modelo tentará aprender uma relação entre as características e os valores-alvo durante este processo de treinamento.



Este trecho de código realiza previsões usando o modelo treinado e calcula duas métricas comuns para avaliar o desempenho do modelo. Aqui está o que cada linha faz:

1. y\_pred = modelo.predict(X\_test): Esta linha gera previsões para as características de teste (X\_test) usando o modelo treinado. A função predict() é usada para fazer previsões com base nas características de entrada fornecidas.
2. rmse = mean\_squared\_error(y\_test, y\_pred): Calcula o erro quadrático médio (RMSE) entre os valores reais (y\_test) e as previsões feitas pelo modelo (y\_pred). A função mean\_squared\_error() do scikit-learn é usada para calcular o RMSE.
3. print(f"RMSE: {rmse:.2f}"): Imprime o valor do RMSE com duas casas decimais.
4. r2 = r2\_score(y\_test, y\_pred): Calcula o coeficiente de determinação (R²) entre os valores reais (y\_test) e as previsões feitas pelo modelo (y\_pred). A função r2\_score() do scikit-learn é usada para calcular o R².
5. print(f"R²: {r2:.2f}"): Imprime o valor do R² com duas casas decimais.

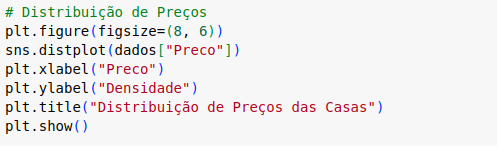


Essa linha de código imprime os coeficientes (pesos) aprendidos pelo modelo de regressão linear para cada uma das características (variáveis independentes). Aqui está o que a linha faz:

1. modelo.coef\_: Atributo coef\_ do modelo de regressão linear contém os coeficientes aprendidos para cada característica. Esses coeficientes representam o efeito de cada característica na variável de destino.
2. print(f"Coeficientes: {modelo.coef\_}"): Esta linha imprime os coeficientes na forma de uma lista. Cada coeficiente corresponde a uma característica na mesma ordem em que as características foram fornecidas durante o treinamento do modelo.

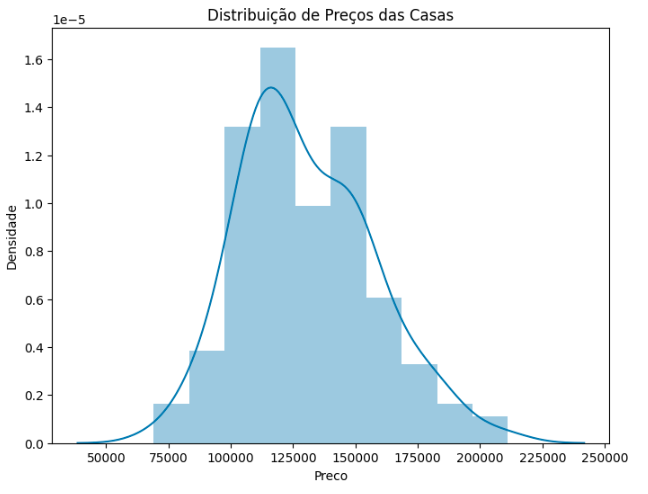


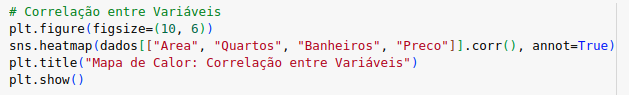
O matplotlib.pyplot é uma biblioteca popular para visualização de dados em Python, enquanto o seaborn é uma biblioteca baseada no matplotlib que oferece uma interface de alto nível para gráficos estatísticos atrativos.



Este trecho de código cria um gráfico de distribuição (histograma) dos preços das casas. Aqui está o que cada linha faz:

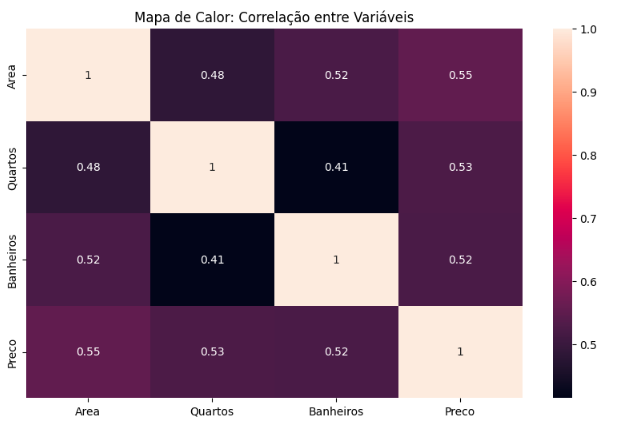
1. plt.figure(figsize=(8, 6)): Esta linha cria uma nova figura para o gráfico com uma determinada largura e altura (definida pelo parâmetro figsize). Neste caso, a figura terá 8 polegadas de largura por 6 polegadas de altura.
2. sns.distplot(dados["Preco"]): Esta linha cria o histograma dos preços das casas usando a função distplot do Seaborn. O histograma mostra a distribuição dos preços das casas, ou seja, quantas casas estão em cada intervalo de preço.
3. plt.xlabel("Preco"): Define o rótulo do eixo x como "Preco", indicando que a variável representada no eixo x é o preço das casas.
4. plt.ylabel("Densidade"): Define o rótulo do eixo y como "Densidade", que representa a densidade da distribuição dos preços das casas.
5. plt.title("Distribuição de Preços das Casas"): Define o título do gráfico como "Distribuição de Preços das Casas".
6. plt.show(): Exibe o gráfico.





Este trecho de código cria um mapa de calor (heatmap) para visualizar a correlação entre as variáveis "Area", "Quartos", "Banheiros" e "Preco". Aqui está o que cada linha faz:

1. plt.figure(figsize=(10, 6)): Esta linha cria uma nova figura para o gráfico com uma largura de 10 polegadas e uma altura de 6 polegadas.
2. sns.heatmap(dados[["Area", "Quartos", "Banheiros", "Preco"]].corr(), annot=True): Esta linha cria o mapa de calor usando a função heatmap do Seaborn. O argumento dados[["Area", "Quartos", "Banheiros", "Preco"]] seleciona apenas as colunas relevantes do DataFrame dados, enquanto .corr() calcula a matriz de correlação entre essas colunas. O argumento annot=True adiciona valores numéricos aos quadrados do mapa de calor para mostrar a magnitude da correlação.
3. plt.title("Mapa de Calor: Correlação entre Variáveis"): Define o título do gráfico como "Mapa de Calor: Correlação entre Variáveis".
4. plt.show(): Exibe o mapa de calor.



**CONCLUSÃO DO RELATÓRIO**

* Qual a importância para uma empresa usar Ciência de Dados

Tomada de decisão informada: Baseia as decisões em dados, não apenas em intuição.

Identificação de oportunidades: Descobre oportunidades de negócio e tendências de mercado.

Otimização de processos: Aumenta a eficiência operacional e reduz custos.

Personalização e segmentação: Permite oferecer produtos e serviços adaptados às necessidades individuais dos clientes.

Previsão e planejamento: Ajuda a antecipar tendências futuras e a se preparar para elas.

Vantagem competitiva: Empresas que utilizam Ciência de Dados têm uma vantagem competitiva significativa.

A Ciência de Dados capacita as empresas a entenderem melhor seus dados, clientes e operações, resultando em melhores decisões, eficiência aprimorada e uma posição mais forte no mercado.

* Qual o objetivo de Aprendizado de Máquina na Ciência de Dados

O objetivo do Aprendizado de Máquina (Machine Learning) na Ciência de Dados é desenvolver algoritmos e modelos que possam aprender padrões nos dados e fazer previsões ou tomar decisões com base nesses padrões. Em resumo, o Aprendizado de Máquina é utilizado na Ciência de Dados para:

Previsão: Criar modelos que possam prever resultados futuros com base em dados históricos. Por exemplo, prever vendas futuras com base em dados de vendas passadas.

Classificação: Classificar dados em categorias ou grupos com base em características. Por exemplo, classificar emails como spam ou não spam.

Agrupamento: Agrupar dados semelhantes em clusters ou grupos. Por exemplo, identificar grupos de clientes com comportamentos de compra semelhantes.

Recomendação: Fornecer recomendações personalizadas com base nos interesses e histórico de um usuário. Por exemplo, recomendar produtos ou filmes com base nas preferências anteriores do usuário.

Detecção de anomalias: Identificar padrões incomuns ou outliers nos dados. Por exemplo, detectar atividades fraudulentas em transações financeiras.

Otimização: Encontrar a melhor solução para um problema com base em critérios específicos. Por exemplo, otimizar rotas de entrega para minimizar o tempo de viagem.

* Qual o objetivo e mecanismos existentes de Mineração de Dados

O objetivo da Mineração de Dados é descobrir informações valiosas e conhecimento útil a partir de grandes conjuntos de dados. Isso é alcançado por meio de diversos mecanismos e técnicas. Aqui estão alguns dos objetivos e mecanismos comuns da Mineração de Dados:

Descoberta de Padrões: Identificar padrões, tendências e relações interessantes nos dados que podem não ser óbvios à primeira vista. Isso pode incluir associações entre itens em conjuntos de transações, sequências temporais, relações hierárquicas, entre outros.

Classificação e Previsão: Classificar dados em categorias ou prever valores futuros com base em padrões encontrados nos dados históricos. Isso é feito usando algoritmos de aprendizado supervisionado, como árvores de decisão, regressão logística, etc.

Agrupamento: Agrupar dados semelhantes em clusters ou segmentos. Esses agrupamentos podem revelar informações sobre a estrutura subjacente dos dados e são úteis para segmentação de mercado, detecção de outliers, entre outros.

Detecção de Anomalias: Identificar padrões incomuns ou excepcionais nos dados que podem indicar comportamento anormal, fraudes, falhas, entre outros. Algoritmos de detecção de anomalias podem ser baseados em estatísticas, distâncias ou modelos de aprendizado de máquina.

Recomendação: Fornecer recomendações personalizadas para usuários com base em seus comportamentos, preferências e histórico de interações. Isso é comumente usado em sistemas de recomendação em e-commerce, streaming de vídeo, redes sociais, etc.

Redução de Dimensionalidade: Reduzir a complexidade dos dados ao extrair características relevantes ou transformar os dados em um espaço de menor dimensionalidade. Isso pode ajudar a melhorar o desempenho dos modelos, reduzir o tempo de processamento e facilitar a visualização dos dados.

Extração de Conhecimento: Extrair conhecimento compreensível e interpretações dos padrões descobertos nos dados. Isso pode incluir regras de associação, árvores de decisão, modelos preditivos, entre outros.

Esses são apenas alguns dos objetivos e mecanismos comuns da Mineração de Dados. A escolha do método depende dos dados específicos, do problema em questão e dos objetivos de análise.

* O que é Web Scraping o quanto ele ajuda no contexto de Big Data

Web scraping é o processo de coletar automaticamente informações de páginas da web, extrair dados estruturados e transformá-los em um formato adequado para análise. Geralmente, isso é feito por meio de scripts ou programas que acessam páginas da web, extraem os dados desejados e os armazenam para uso posterior.

No contexto de Big Data, o web scraping pode ser extremamente útil por várias razões:

Ampliação do Conjunto de Dados: O web scraping permite coletar dados de uma variedade de fontes online, o que pode enriquecer significativamente o conjunto de dados disponíveis para análise.

Atualização em Tempo Real: Ao automatizar o processo de coleta de dados da web, as informações podem ser atualizadas regularmente e em tempo real, garantindo que as análises sejam baseadas nos dados mais recentes.

Diversidade de Fontes: O scraping da web pode ajudar a obter dados de diversas fontes, incluindo redes sociais, fóruns online, sites de notícias, entre outros, o que pode proporcionar uma visão mais abrangente e completa do contexto em que os dados estão inseridos.

Descoberta de Insights: O acesso a uma variedade de fontes de dados pode levar à descoberta de insights únicos e oportunidades de negócio que podem não ser encontrados em conjuntos de dados tradicionais.

Personalização e Recomendação: Os dados obtidos por meio de web scraping podem ser usados para personalizar recomendações de produtos, conteúdo ou serviços com base nos interesses e comportamentos dos usuários.

Monitoramento da Concorrência: O scraping da web pode ser usado para monitorar as atividades e estratégias da concorrência, fornecendo informações valiosas para ajustar as próprias estratégias de negócios.

No entanto, é importante observar que o web scraping deve ser feito com responsabilidade e em conformidade com os termos de serviço dos sites, para evitar problemas legais ou éticos. Além disso, a escalabilidade e a eficiência do scraping da web em contextos de Big Data podem ser desafiadoras devido à necessidade de lidar com grandes volumes de dados e à complexidade de processar e armazenar esses dados de forma eficaz.