Tipos de dados

Ao iniciar sua jornada na análise de dados, é crucial compreender os tipos de dados disponíveis e suas origens. Quanto à classificação, os dados podem ser divididos em três categorias principais: estruturados, não estruturados e semiestruturados.
Os dados estruturados possuem uma estrutura definida, apresentando-se em formato tabular e podendo ser armazenados e manipulados em um banco de dados relacional. Por outro lado, os dados não estruturados não seguem um formato tabular. Exemplos incluem arquivos de áudio mp3, imagens e textos.
Já os dados semiestruturados não possuem formato tabular, mas são estruturados por meio de *tags* que estabelecem uma hierarquia. Exemplos incluem arquivos XML e JSON.

Pandas

A primeira etapa de um projeto de análise de dados é a obtenção dos dados, e as principais fontes costumam ser arquivos, bancos de dados ou APIs web.
Utilizaremos a biblioteca *pandas* como ferramenta principal para a extração dos dados. O *pandas* é uma das principais ferramentas para trabalhar com dados estruturados em *Python* e oferece uma variedade de funções para obter dados de diversas fontes.
Uma dessa fontes é o arquivo de texto separado por vírgulas (CSV). Trata-se de um formato de arquivo que armazena dados tabulares em texto simples, em que cada linha representa uma linha da tabela e os valores das colunas são separados por vírgulas.
Para ler dados de um arquivo desse tipo, utilizamos a função *read_csv()*. Em projetos reais, é comum encontrar dados "sujos" que precisam de tratamento. Para lidar com possíveis problemas nos dados, a função *read_csv* possui uma série de parâmetros opcionais que podem ser utilizados.
Arquivos do Excel são outra fonte comum de dados e podem ser lidos pela função *read_excel* do *pandas*. Assim como *read_csv*, esta função possui parâmetros opcionais que podem ser úteis em casos específicos.
Após a transformação dos dados usando o *pandas*, é possível escrevê-los em diferentes formatos usando as funções da biblioteca. Para salvar os dados em um arquivo CSV ou Excel, basta usar as funções *to_csv* e *to_excel*, respectivamente.
Outra forma de armazenar os dados é no formato *pickle*. De acordo com Mackinney (2018, [s. p.]), "uma das maneiras mais eficientes de armazenar dados de forma binária é usando a serialização pickle nativa do Python". A classe *DataFrame* do *pandas* possui o método *to_pickle* que pode ser usado para escrever os dados nesse formato.
O formato pickle é recomendado para arquivos que você não quer armazenar por muito tempo. Este formato pode não ser estável a longo prazo, pois um dado serializado hoje pode não ser lido por uma versão futura do *Python*.

Bancos de dados

Uma outra fonte de dados muito comum em empresas são os bancos de dados relacionais, em que os dados são organizados em tabelas. Uma tabela, em termos de banco de dados relacionais, é uma estrutura que organiza dados em linhas e colunas. Cada linha representa uma entrada de dados, também conhecida como

registro ou tupla, e cada coluna representa um atributo ou campo específico dos dados. As tabelas são utilizadas para armazenar e organizar informações de forma estruturada, facilitando a consulta e manipulação dos dados por meio de consultas no formato de *Structured Query Language* (SQL), ou Linguagem de Consulta Estruturada.
A linguagem SQL é usada para manipular dados em bancos de dados relacionais, e o resultado de uma consulta SQL é uma tabela (BEAULIEU, 2020). As consultas são feitas com o comando SELECT, que pode ser combinado com outros comandos para refinar a busca.

```
SELECT id_transacao, id_produto, quantidade
FROM venda
WHERE id_cliente = 1;
```

Neste exemplo, o *SELECT* seleciona as colunas a serem retornadas, o *FROM* identifica a tabela e o *WHERE* cria uma condição para retornar apenas as transações do cliente 1.
Consultas mais complexas podem ser feitas adicionando mais comandos SQL.

```
SELECT id_transacao, id_produto, SUM(quantidade)
FROM venda
GROUP BY id_cliente, id_produto;
```

No código apresentado no Quadro 2, usamos a função *SUM* dentro do *SELECT* para somar a coluna quantidade e o comando *GROUP BY* para agregar as quantidades por cliente e produto.
Quando abordamos os tipos de dados semiestruturados, o formato *JavaScript Object Notation* (JSON) foi apresentado como exemplo. O JSON é um formato de dados leve e legível por humanos, usado para transmitir dados estruturados entre um servidor e um cliente, comumente usados em serviços web. Ele é baseado em pares de chave-valor e suporta *arrays* (listas) e objetos (dicionários).

Tratamento de dados

Geralmente, os dados não estão disponíveis no formato ideal para análise. Grande parte do tempo de um analista (ou cientista) de dados é dedicado ao preparo dos dados (DALLEMULE, 2017), realizando operações como extração, combinação, transformação e agregação das informações.
Ao lidar com dados estruturados, uma parte significativa do trabalho envolve a manipulação de *DataFrame*s *pandas*. A seguir, veremos algumas maneiras de acessar parte dos dados em um *DataFrame*, o que é chamado de filtragem de dados. A indexação em um *DataFrame* pode ser usada para selecionar uma ou mais linhas ou colunas.

Para selecionar uma coluna de um *DataFrame*, utilizamos a sintaxe *dataframe['nome_da_coluna']*. Se quisermos acessar mais de uma coluna, podemos usar uma lista e a sintaxe *dataframe[['nome_da_coluna1', 'nome_da_coluna2']]*. Uma sintaxe semelhante pode ser usada para selecionar linhas com base em seus índices. A notação *dataframe[]* é a mesma para seleção de colunas, mas dentro dos colchetes devemos usar o mesmo padrão de fatiamento de listas em *Python*. Portanto, a sintaxe para acessar as linhas de um *DataFrame* será *dataframe[indice_inicial:indice_final]*.

Para selecionar um subconjunto de linhas e colunas de um *DataFrame*, existem os operadores especiais de indexação *loc* e *iloc*.

- *loc*: por meio do operador *loc*, é possível selecionar as linhas pelos rótulos de índice, utilizando fatiamento ou um *array* de booleanos. As colunas podem ser selecionadas pelos seus rótulos.

- iloc: o operador iloc seleciona linhas e colunas pela sua posição, utilizando números inteiros.

Tratamento de nulo

Durante a limpeza e tratamento dos dados, é comum lidar com dados faltantes. A biblioteca *pandas* oferece funcionalidades para encontrar e tratar esses valores nulos.

- *isna*: para verificar a presença de dados nulos em um *DataFrame* ou *Series*, podemos usar o método *isna()*. Ele verifica cada elemento e retorna *True* se o valor for nulo. Em *Python*, valores booleanos podem ser interpretados como números inteiros, onde *True* é equivalente a 1 e *False* a 0. Podemos usar o método de agregação *sum* junto com *isna()* para contar a quantidade de valores nulos em um *DataFrame*.
- *fillna*: o método *fillna()* substitui os valores nulos por um valor escolhido pelo usuário.

Agregações

Aqui, é crucial deixar as informações prontas para análise, já que os dados brutos raramente fornecem insights relevantes. Uma maneira de extrair informações úteis é por meio de agregações. O *pandas* oferece diversos métodos de agregação para uso em *DataFrames* e *Series*.
As agregações também podem ser feitas por grupos de dados usando o método *groupby()*, que divide o *DataFrame* em partes usando uma ou mais chaves. Após dividir o *DataFrame*, podemos aplicar métodos de agregação, transformação ou outras manipulações de dados (MACKINNEY, 2018).
Quando utilizamos o método *groupby*, podemos aplicar o método *mean* para calcular a média dos valores agrupados. O método *groupby* também pode ser combinado com o método *agg* para dar ainda mais flexibilidade nas possibilidades de agregações possíveis.
Podemos combinar estruturas de dados do *pandas* de duas maneiras: usando o método *merge()*, que combina linhas de dois *DataFrame*s com base em uma ou mais chaves, e usando o método *concat()*, que "empilha" dois ou mais objetos *pandas*.

Por padrão, o método *merge()* usará todas as colunas em comum nos dois *DataFrame*s. No entanto, podemos explicitamente indicar quais colunas devem ser usadas como chave utilizando o parâmetro *on*.

Feature engineering

Feature engineering é um conjunto de técnicas para preparar dados para modelos de *machine learning*, incluindo seleção e criação de características relevantes. Isso envolve transformações matemáticas, tratamento de valores ausentes e manipulação de *outliers*. Além disso, o processo abrange a criação de novas características. É crucial considerar a área de negócio para aplicar essas técnicas com eficácia, pois a qualidade dos dados impacta diretamente nos resultados dos modelos de *machine learning*.
Praticamente todos os algoritmos de *machine learning* possuem entradas e saídas. As entradas são colunas de dados estruturados, chamadas de *features*, e as saídas são variáveis dependentes ou classes que estamos tentando prever. O *Feature Engineering* envolve análise de dados, aplicação de regras práticas, bom senso e testes. Ele é geralmente realizado manualmente, o que pode ser um gargalo no processo de construção de um modelo de *machine learning*.

Algumas técnicas comuns de *Feature Engineering* incluem:

- Criação de novas *features* a partir das existentes, especialmente útil quando há poucos dados para evitar *overfitting*.
- Transformações de *features*:
    - Tratamento de valores ausentes: valores ausentes na base de dados podem ocorrer por vários motivos, como questões de permissões, erros humanos, erros de código, etc. A maioria dos algoritmos de aprendizado de máquina não aceitam conjuntos de dados com valores ausentes. Há um consenso de que, se uma *feature* tiver mais que 20% dos dados faltantes em sua coluna, é melhor não a utilizar e entender por qual motivo esses valores estão ausentes. Caso uma *feature* tenha até 20% dos valores em falta, deve-se preencher esses dados com a média ou a mediana da variável. Quando se tratar de um valor categórico, preencher com o valor categórico médio. E quando menos de 2% dos dados de uma *feature* estiverem ausentes, o mais indicado é deletar esses registros, pois sendo uma quantidade pequena, a redução da base de dados será mínima, além de evitar possíveis problemas (CORRÊA, 2021).
    - Manipulação de *outliers*: *outliers* são valores fora do "padrão" encontrados nos dados. Um valor é considerado um *outlier* quando a diferença dele para a média é maior que *x* vezes o desvio padrão, ou quando os valores estão abaixo do primeiro quartil (Q1) ou acima do terceiro quartil (Q3) da distribuição de dados. Ao identificar um *outlier,* é sempre importante identificar sua causa, pois ele pode ser uma boa oportunidade de negócio. No entanto, *outliers* também podem indicar erros no conjunto de dados (CORRÊA, 2021).
    - *Binning*: divide dados numéricos em intervalos.
    - *One-hot encoding*: transforma variáveis categóricas em colunas binárias.
    - *Grouping*: realiza agrupamentos de variáveis categóricas e numéricas.

- Seleção de *features*: após a transformação das *features*, é importante selecionar quais delas serão utilizadas no modelo, pois um grande número de *features* pode prejudicar o desempenho do modelo e aumentar o tempo de treinamento. A escolha das *features* adequadas requer testar várias combinações para medir a acurácia do modelo preditivo.

O processo de *Feature Engineering* é crucial para o sucesso de um projeto de *machine learning*, pois dados de entrada de alta qualidade resultam em melhores modelos. Embora existam estudos para automatizar esse processo, atualmente ele é realizado manualmente e demanda tempo e conhecimento da área de negócio.

Gráficos

O tipo de visualização deve ser compatível com as informações que serão visualizadas. Há vários tipos de visualização para apresentar os dados de forma eficaz e interessante: gráficos, tabelas, diagramas, mapas, infográficos e painéis. Destes, um tipo muito utilizado para visualização de dados é a utilização de gráficos Os gráficos são representações que facilitam a análise de dados, os quais costumam ser dispostos em tabelas. Eles oferecem mais praticidade, principalmente quando os dados não são discretos, ou seja, quando envolvem números consideravelmente grandes. Além disso, os gráficos também apresentam de maneira evidente os dados em seu aspecto temporal.

Principais tipos de gráficos

- Histograma: ideal para visualizar a distribuição de uma variável numérica. É útil para entender a forma da distribuição e identificar padrões, como simetria e assimetria.
- Gráfico de Barras: usado para comparar a frequência ou proporção de categorias em uma variável categórica. É útil para visualizar diferenças entre grupos.
- Gráfico de Linhas: ótimo para mostrar tendências ao longo do tempo ou em uma sequência ordenada. É útil para identificar padrões de crescimento ou declínio.
- *Scatterplot*: utilizado para mostrar a relação entre duas variáveis numéricas. É útil para identificar padrões de correlação ou dispersão entre as variáveis.
- *Boxplot*: ideal para visualizar a distribuição, dispersão e identificação de *outliers* em uma variável numérica.
- *Heatmap*: útil para visualizar padrões em uma matriz de dados, destacando áreas de alta ou baixa intensidade.

Matplotlib

A biblioteca *Matplotlib* é um dos pilares da visualização de dados em *Python*, e inúmeras outras bibliotecas a aproveitam como base para construir novas funcionalidades

Seaborn

O *Seaborn* utiliza muitos elementos do *Matplotlib* e nos fornece uma API simples para construir gráficos visualmente bonitos.

Usando classes para reutilizar gráficos

Para facilitar a reutilização e a organização dos gráficos, podemos criar classes em *Python* que encapsulam a lógica de plotagem. Isso nos permite criar gráficos personalizados com facilidade, reutilizando o código e mantendo a consistência visual em nossas análises.
Ao usar classes para reutilizar gráficos, podemos economizar tempo e manter nossas análises mais organizadas e consistentes.

Data Storytelling

Data storytelling é a prática de usar dados para contar histórias e envolver pessoas. Empresas têm usado esse método como uma forma de se conectar com o público, aproveitando a crescente importância dos números nos últimos anos. Assim como as pessoas mostram interesse em dados como o número de seguidores em redes sociais, elas também se interessam por dados que tragam informações impactantes, vendo-os como uma prova de sucesso.
Essa abordagem visa criar credibilidade ao demonstrar que os números obtidos refletem sucesso e engajamento. Além disso, o *data storytelling* é importante no uso interno das empresas, especialmente em apresentações para equipes e *stakeholders*. Dominar essa técnica ajuda os profissionais a se comunicarem melhor com outras áreas, já que uma narrativa bem elaborada facilita a compreensão dos dados e contribui para o alcance de objetivos comuns.
As marcas estão adotando o público como foco principal em suas estratégias de *data storytelling*, usando os dados gerados por ele para criar narrativas em que os consumidores são os protagonistas. Um exemplo é a campanha *Year in Search* do *Google*, que mostra as principais tendências do ano com gamificação. Os usuários podem interagir com diferentes categorias de conteúdo e descobrir mais sobre as tendências e seu volume de buscas. Esse fluxo contínuo de interações mantém o engajamento e facilita o compartilhamento. A campanha é simples, mas eficaz, aproximando os usuários da marca. Os dados são fornecidos pelo *Google Trends*, uma plataforma importante para o planejamento de estratégias de conteúdo.

Passos para o Storytelling com Dados

1. Reúna os principais insights: comece com uma análise de dados para gerar perspectivas e ideias valiosas. Permita que a equipe tenha liberdade criativa para pensar em diferentes possibilidades.
2. Pense na audiência: considere que tipo de informações serão relevantes para o público-alvo. Os dados devem ser estratégicos e atrativos para despertar interesse.

3. Escolha o melhor formato: adapte a campanha de *storytelling* com dados para diferentes plataformas e mídias. Experimente formatos interativos, como infográficos e vídeos, de acordo com a audiência e o objetivo da campanha.
4. Defina uma chamada para ação (CTA): Utilize CTAs para direcionar a ação do público, seja compartilhar a experiência, fornecer insights ou promover produtos e serviços.
5. Crie contexto: contextualize os dados apresentados para que façam sentido dentro da narrativa, reforçando o produto ou serviço em questão.
6. Escolha os argumentos: utilize os dados como base para sustentar argumentos convincentes, seja para vender um produto ou serviço, captar investimentos ou convencer *stakeholders*.
7. Defina um fluxo para a história: organize as ideias e a apresentação dos dados em um fluxo temporal que mantenha a narrativa coesa e cativante para o público-alvo.