| FT03 |
|---|
| **Curso**: UFCD **10809** |
| **UFCD/Módulo/Temática:** UFCD 10793 - **Fundamentos de Python** |
| **Ação: 10809_1L Visualização de dados em Python** |
| **Formador/a**: Sandra Liliana Meira de Oliveira |
| **Data**: |
| **Nome do Formando/a**: |

# 1. Analisar dados – Shoping Trends

## Descrição Detalhada do Dataset

O conjunto de dados fornecido contém informações relacionadas a compras realizadas por clientes, onde cada linha representa uma transação individual. O dataset inclui várias variáveis que descrevem as características dos clientes e das suas compras. A seguir, são descritas as colunas presentes no ficheiro:

1. **Customer ID**: Identificador único do cliente. Cada cliente tem um número distinto.
2. **Age**: Idade do cliente, uma variável numérica que representa a idade do comprador no momento da compra.
3. **Gender**: Género do cliente, categorizado como "Male" (Masculino) ou "Female" (Feminino).
4. **Item Purchased**: Nome do item comprado, que pode ser uma peça de roupa, calçado ou outros produtos.
5. **Category**: Categoria do produto comprado, como "Clothing" (Roupas), "Footwear" (Calçado), etc.
6. **Purchase Amount (USD)**: Valor gasto na compra, em dólares americanos (USD).
7. **Location**: Localização do cliente, identificada pelo estado ou região onde o cliente reside.
8. **Size**: Tamanho do artigo comprado, com categorias como "S" (pequeno), "M" (médio), "L" (grande).
9. **Color**: Cor do artigo comprado, por exemplo, "Gray", "Maroon", "Turquoise".
10. **Season**: Estação do ano em que a compra foi realizada, como "Winter" (Inverno) ou "Spring" (Primavera).
11. **Review Rating**: Avaliação dada pelo cliente ao produto, numa escala numérica (por exemplo, de 1 a 5).
12. **Subscription Status**: Status de subscrição do cliente, que pode ser "Yes" (Sim) ou "No" (Não).
13. **Payment Method**: Método de pagamento utilizado para a compra, como "Credit Card" (Cartão de Crédito), "Bank Transfer" (Transferência Bancária), "Cash" (Dinheiro), entre outros.

14. **Shipping Type**: Tipo de envio escolhido pelo cliente, por exemplo, "Express", "Free Shipping" (Envio Grátis), "Next Day Air" (Envio no Dia Seguinte).
15. **Discount Applied**: Indica se foi aplicado um desconto à compra ("Yes" ou "No").
16. **Promo Code Used**: Indica se foi utilizado um código promocional para a compra ("Yes" ou "No").
17. **Previous Purchases**: Número de compras anteriores realizadas pelo cliente.
18. **Preferred Payment Method**: Método de pagamento preferido pelo cliente (por exemplo, "Credit Card" ou "PayPal").
19. **Frequency of Purchases**: Frequência com que o cliente faz compras, com opções como "Weekly" (Semanal), "Fortnightly" (Quinzenal), "Annually" (Anualmente).

Com base nestas variáveis, podemos proceder com a análise, tratamento e visualização dos dados.

### Passo 1: Limpeza de Dados

Antes de realizar qualquer análise, é essencial garantir que o dataset está limpo e pronto para processamento. A limpeza de dados pode envolver as seguintes etapas:

1. **Verificação de valores ausentes**: Detectar e tratar dados faltantes.
2. **Verificação de duplicados**: Identificar e remover linhas duplicadas, se existirem.
3. **Correção de tipos de dados**: Garantir que cada coluna está com o tipo de dado correto (por exemplo, numérico, categórico).
4. **Verificação de valores inconsistentes**: Identificar dados que possam estar incorretos ou fora de contexto.

```
# Verificando valores ausentes
missing_values = shopping_trends_df.isnull().sum()

# Verificando duplicados
duplicates = shopping_trends_df.duplicated().sum()

# Exibindo tipos de dados das colunas
data_types = shopping_trends_df.dtypes
```

*Interpretação:*

- **Valores ausentes**: A verificação de valores ausentes permitirá identificar colunas com dados faltantes. Se existirem valores ausentes, podemos optar por preenchê-los com valores médios, modas, ou até mesmo eliminar as linhas afetadas.
- **Duplicados**: Linhas duplicadas podem distorcer os resultados da análise, portanto, é importante removê-las.
- **Tipos de dados**: As colunas devem estar no tipo adequado para garantir a correta análise (por exemplo, "Age" deve ser numérica, "Gender" deve ser categórica).

### Passo 2: Tratamento de Dados

Uma vez limpos os dados, podemos proceder com o tratamento de valores e transformar variáveis conforme necessário. Aqui estão algumas operações comuns de tratamento:

1. **Conversão de tipos de dados**: Converter colunas para o tipo adequado, como inteiros ou categorias.
2. **Criação de variáveis derivadas**: Por exemplo, criar uma nova coluna indicando a faixa etária (jovem, adulto, idoso) com base na idade.

```
# Conversão de variáveis categóricas
shopping_trends_df['Gender'] =
shopping_trends_df['Gender'].astype('category')

# Criação de faixa etária
bins = [0, 18, 40, 60, 100]
labels = ['Jovem', 'Adulto', 'Meia-Idade', 'Idoso']
shopping_trends_df['Age Group'] = pd.cut(shopping_trends_df['Age'],
bins=bins, labels=labels)
```

### Passo 3: Análise Exploratória de Dados (AED)

A análise exploratória de dados (AED) tem como objetivo entender melhor a distribuição dos dados e identificar padrões ou tendências interessantes. Algumas análises importantes incluem:

1. **Distribuição de variáveis numéricas**: Verificar a distribuição da idade, quantidade de compras, valor das compras.
2. **Distribuição de variáveis categóricas**: Verificar a distribuição do género, categoria de item comprado, método de pagamento.
3. **Análise de correlações**: Verificar se existem relações significativas entre variáveis numéricas, como entre a idade e o valor da compra.

```
# Estatísticas descritivas para variáveis numéricas
numerical_summary = shopping_trends_df.describe()

# Contagem de variáveis categóricas
gender_distribution = shopping_trends_df['Gender'].value_counts()
category_distribution = shopping_trends_df['Category'].value_counts()

# Correlações entre variáveis numéricas
correlations = shopping_trends_df.corr()
```

### Passo 4: Monitorização

A monitorização dos dados é uma etapa contínua em que se observa o comportamento das variáveis ao longo do tempo. Para isso, podemos analisar a frequência de compras dos clientes e o impacto de promoções ou descontos.

```
# Contagem das frequências de compra
purchase_frequency = shopping_trends_df['Frequency of
Purchases'].value_counts()

# Análise de compras com e sem desconto
```

```
discount_purchase_analysis = shopping_trends_df.groupby('Discount
Applied')['Purchase Amount (USD)'].mean()
```

### Passo 5: Processamento e Agrupamento

O processamento dos dados pode envolver a agregação de variáveis para obter insights adicionais, como o gasto médio por cliente, por categoria de produto, ou por localização.

```
# Gasto médio por cliente
avg_spend_per_customer = shopping_trends_df.groupby('Customer
ID')['Purchase Amount (USD)'].mean()

# Gasto médio por categoria
avg_spend_per_category = shopping_trends_df.groupby('Category')['Purchase
Amount (USD)'].mean()
```

### Passo 6: Visualizações Gráficas

A visualização gráfica é essencial para comunicar as descobertas de forma eficaz. Usaremos gráficos simples utilizando o `pandas` para ilustrar as distribuições e correlações.

```
import matplotlib.pyplot as plt

# Distribuição de idades
shopping_trends_df['Age'].hist(bins=10)
plt.title('Distribuição de Idades dos Clientes')
plt.xlabel('Idade')
plt.ylabel('Frequência')
plt.show()

# Gasto médio por categoria
avg_spend_per_category.plot(kind='bar')
plt.title('Gasto Médio por Categoria de Produto')
plt.xlabel('Categoria')
plt.ylabel('Gasto Médio (USD)')
plt.show()

# Correlação entre variáveis numéricas
correlations['Purchase Amount (USD)'].plot(kind='bar')
plt.title('Correlação com o Gasto Médio')
plt.xlabel('Variáveis')
plt.ylabel('Correlação')
plt.show()
```

### Passo 7: Conclusões

Após a análise e visualização dos dados, podemos tirar algumas conclusões, como:

- Identificar a faixa etária predominante dos clientes.
- Verificar quais categorias de produtos são mais populares.
- Analisar o impacto de promoções e descontos nas compras.
- Observar se o género influencia o valor da compra ou a frequência de compras.

Esses passos são essenciais para realizar uma análise detalhada e obter insights valiosos sobre os comportamentos de compra dos clientes.

A limpeza de dados é uma etapa crítica para garantir a qualidade e a confiabilidade das análises subsequentes. Existem várias abordagens e técnicas que podem ser aplicadas para melhorar a limpeza de dados. Abaixo, explico algumas das melhores práticas que podem ser implementadas para uma limpeza de dados mais eficaz:

**Podemos Ainda Acrescentar:**

## 1. Identificação e Tratamento de Valores Ausentes

Valores ausentes (ou `NaN` - Not a Number) são comuns em muitos conjuntos de dados e podem afetar a análise. Existem diferentes maneiras de tratar valores ausentes:

- **Verificar a quantidade de dados ausentes por coluna**: Antes de tomar uma decisão sobre como tratar os valores ausentes, é importante entender qual a quantidade de dados ausentes em cada coluna.

```
# Verificar os valores ausentes em cada coluna
missing_values = shopping_trends_df.isnull().sum()

# Verificar a porcentagem de dados ausentes
missing_percentage = (shopping_trends_df.isnull().mean() * 100).round(2)
```

- **Preenchimento de valores ausentes**: Se o número de valores ausentes for baixo, pode-se preencher esses valores com uma estratégia apropriada. Aqui estão algumas opções:
  - **Preenchimento com média ou mediana (para variáveis numéricas)**: Preencher os valores ausentes com a média ou mediana da coluna pode ser uma boa solução, especialmente se os valores ausentes forem em pequena quantidade.

```
# Preencher valores ausentes em uma coluna numérica com a média
shopping_trends_df['Age'].fillna(shopping_trends_df['Age'].mean(),
inplace=True)
```

- **Preenchimento com moda (para variáveis categóricas)**: Para variáveis categóricas, podemos preencher os valores ausentes com a moda (o valor mais frequente da coluna).

```
# Preencher valores ausentes em uma coluna categórica com a moda
shopping_trends_df['Gender'].fillna(shopping_trends_df['Gender'].mode()[0],
inplace=True)
```

- **Remoção de linhas com valores ausentes**: Em casos onde os dados ausentes são numerosos ou não podem ser inferidos adequadamente, podemos eliminar as linhas com valores ausentes.

```
# Eliminar linhas com valores ausentes
shopping_trends_df.dropna(inplace=True)
```

## 2. Tratamento de Valores Duplicados

Linhas duplicadas podem distorcer a análise, e devem ser removidas para garantir que os dados estejam limpos. Identificar e remover duplicados é uma parte importante da limpeza de dados.

```
# Verificar duplicados no dataset
duplicates = shopping_trends_df.duplicated().sum()

# Remover duplicados
shopping_trends_df.drop_duplicates(inplace=True)
```

## 3. Correção de Tipos de Dados

Verificar e corrigir os tipos de dados é fundamental para evitar problemas durante a análise. Algumas colunas podem estar com o tipo incorreto (por exemplo, valores numéricos armazenados como texto). Podemos ajustar os tipos conforme necessário.

- **Converter variáveis numéricas para o tipo adequado**:

```
# Garantir que a coluna 'Purchase Amount (USD)' seja numérica
shopping_trends_df['Purchase Amount (USD)'] =
pd.to_numeric(shopping_trends_df['Purchase Amount (USD)'], errors='coerce')
```

- **Converter variáveis categóricas**:

```
# Garantir que as colunas 'Gender' e 'Subscription Status' sejam
categóricas
shopping_trends_df['Gender'] =
shopping_trends_df['Gender'].astype('category')
shopping_trends_df['Subscription Status'] =
shopping_trends_df['Subscription Status'].astype('category')
```

## 4. Detecção e Correção de Outliers

Outliers são valores que estão muito distantes da distribuição normal dos dados. Detectá-los e tratá-los pode melhorar a qualidade das análises. Existem várias maneiras de identificar e tratar outliers:

- **Uso de percentis**: Uma maneira comum de detectar outliers é utilizar o intervalo interquartil (IQR) para identificar valores fora do intervalo esperado.

```
# Calcular o intervalo interquartil (IQR)
Q1 = shopping_trends_df['Purchase Amount (USD)'].quantile(0.25)
Q3 = shopping_trends_df['Purchase Amount (USD)'].quantile(0.75)
IQR = Q3 - Q1

# Definir os limites inferior e superior para detectar outliers
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

# Remover outliers
shopping_trends_df = shopping_trends_df[(shopping_trends_df['Purchase
Amount (USD)'] >= lower_limit) &
```

```
                                        (shopping_trends_df['Purchase
Amount (USD)'] <= upper_limit)]
```

### 5. Correção de Inconsistências e Erros Tipográficos

Em conjuntos de dados com texto, como a coluna **Location** ou **Item Purchased**, é comum encontrar inconsistências de grafia. Para garantir a integridade dos dados, pode ser necessário padronizar os valores.

- **Uniformizar categorias de texto**: Se houver valores inconsistentes, podemos padronizá-los (ex: "Red" e "red" devem ser tratados como iguais).

```
# Padronizar valores de texto para evitar inconsistências
shopping_trends_df['Color'] = shopping_trends_df['Color'].str.lower()
shopping_trends_df['Location'] =
shopping_trends_df['Location'].str.strip().str.lower()
```

### 6. Tratamento de Variáveis Categóricas

Em variáveis categóricas, é importante verificar se todas as categorias são válidas e bem definidas. Algumas opções incluem:

- **Verificar e substituir valores inválidos em variáveis categóricas**:

```
# Substituir valores inválidos ou inesperados na coluna 'Gender'
shopping_trends_df['Gender'] = shopping_trends_df['Gender'].replace({'M':
'Male', 'F': 'Female'})
```

- **Criar novas variáveis a partir de variáveis categóricas**: Podemos criar novas variáveis a partir de variáveis existentes, como a faixa etária com base na idade, ou transformar o status de subscrição em variável binária (Sim/Não).

```
# Criar variável binária para status de subscrição
shopping_trends_df['Is Subscribed'] = shopping_trends_df['Subscription
Status'].apply(lambda x: 1 if x == 'Yes' else 0)
```

### 7. Verificação de Consistência nas Colunas Numéricas

É importante garantir que os valores numéricos estejam dentro de um intervalo razoável. Por exemplo, verificar se o valor gasto na compra é maior que zero e se a idade do cliente está dentro de uma faixa plausível.

```
# Remover valores de compra menores que zero
shopping_trends_df = shopping_trends_df[shopping_trends_df['Purchase Amount
(USD)'] > 0]

# Garantir que a idade esteja dentro de um intervalo plausível (18 a 100
anos)
shopping_trends_df = shopping_trends_df[(shopping_trends_df['Age'] >= 18) &
(shopping_trends_df['Age'] <= 100)]
```

Melhorar a limpeza de dados é um processo contínuo que envolve a identificação de valores ausentes, duplicados, outliers e inconsistências. Uma boa prática é garantir que todos os tipos de dados estejam corretos, e que as variáveis estejam bem tratadas e consistentes. Com esses cuidados, é possível garantir que a análise e os modelos subsequentes sejam baseados em dados confiáveis e de qualidade.

A **Análise Exploratória de Dados (EAD)** é uma fase crucial em qualquer processo de análise de dados, permitindo que os analistas entendam melhor os dados antes de aplicar técnicas mais complexas de modelagem ou análise. As principais etapas de EAD são descritas a seguir de forma detalhada, considerando que o objetivo é identificar padrões, tendências e insights nos dados, além de detectar problemas como valores ausentes, outliers e inconsistências.

### 1. Definição dos Objetivos da Análise

Antes de iniciar qualquer exploração dos dados, é essencial definir claramente os objetivos da análise. O que se deseja descobrir ou resolver com os dados disponíveis? Alguns objetivos comuns podem incluir:

- **Identificar tendências de consumo**: Como os clientes compram os produtos? Quais são as categorias de produtos mais populares?
- **Verificar relações entre variáveis**: Existe alguma correlação entre idade e valor da compra? Ou entre o tipo de pagamento e a frequência das compras?
- **Detecção de anomalias**: Encontrar clientes com comportamentos de compra atípicos, como um gasto muito elevado ou um número excessivo de compras.

### 2. Importação e Preparação dos Dados

A segunda etapa envolve carregar os dados e prepará-los para a análise. Isso pode envolver:

- **Carregar os dados em um ambiente de análise**: Importação de dados de arquivos CSV, Excel, banco de dados, entre outros.
- **Verificação de tipos de dados**: Garantir que as variáveis estão corretamente tipadas (numéricas, categóricas, etc.).
- **Limpeza preliminar**: Realizar a limpeza inicial, como remover duplicados, tratar valores ausentes, corrigir erros de digitação, entre outros.

```
# Carregar dados
import pandas as pd
df = pd.read_csv(shopping_trends.csv')

# Verificar os tipos de dados
df.dtypes
```

### 3. Análise Descritiva

A análise descritiva visa resumir e entender a distribuição dos dados de uma maneira geral, para fornecer uma visão inicial dos mesmos. Algumas técnicas incluem:

- **Estatísticas descritivas**: Cálculo de medidas como média, mediana, desvio padrão, mínimos e máximos.

```
# Estatísticas descritivas
df.describe()
```

- **Distribuição das variáveis**: Verificar a distribuição das variáveis numéricas (histogramas, boxplots) e categóricas (contagem de valores únicos).

```
# Visualizar a distribuição de uma variável numérica
df['Purchase Amount (USD)'].hist()

# Visualizar a distribuição de uma variável categórica
df['Category'].value_counts()
```

- **Verificação de valores ausentes**: Determinar quais variáveis têm valores ausentes e em que proporção.

```
# Verificar valores ausentes
df.isnull().sum()
```

### 4. Identificação de Outliers

Outliers são valores que estão fora do padrão esperado e podem distorcer as análises. É importante identificá-los e decidir se devem ser removidos ou tratados. As principais abordagens incluem:

- **Uso de boxplots**: Para detectar outliers em variáveis numéricas.

```
# Boxplot para detectar outliers
import seaborn as sns
sns.boxplot(x=df['Purchase Amount (USD)'])
```

- **Análise com o intervalo interquartil (IQR)**: Detectar outliers usando a diferença entre os quartis.

```
# Calcular IQR
Q1 = df['Purchase Amount (USD)'].quantile(0.25)
Q3 = df['Purchase Amount (USD)'].quantile(0.75)
IQR = Q3 - Q1
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR

# Filtrar outliers
df = df[(df['Purchase Amount (USD)'] >= lower_limit) & (df['Purchase Amount
(USD)'] <= upper_limit)]
```

### 5. Detecção de Correlações

Após a análise descritiva, é essencial entender como as variáveis estão relacionadas entre si. A análise de correlação ajuda a identificar relações lineares entre variáveis numéricas. As correlações podem ser visualizadas por meio de:

- **Matriz de correlação**: Exibição das correlações entre todas as variáveis numéricas.

```
# Matriz de correlação
correlation_matrix = df.corr()
print(correlation_matrix)
```

- **Heatmap**: Um gráfico de calor pode ser usado para visualizar as correlações de forma mais intuitiva.

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

# Heatmap de correlação
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```

### 6. Análise de Variáveis Categóricas

Além das variáveis numéricas, é importante analisar as variáveis categóricas para entender a distribuição de classes e as possíveis relações com outras variáveis. Algumas técnicas incluem:

- **Contagem de frequências**: Para variáveis como gênero, categoria do produto, status de subscrição, etc.

```
# Frequência de uma variável categórica
df['Gender'].value_counts()
```

- **Gráficos de barras**: Para visualização das distribuições das variáveis categóricas.

```
# Gráfico de barras para uma variável categórica
df['Category'].value_counts().plot(kind='bar')
```

- **Tabela de contingência**: Para verificar a relação entre duas variáveis categóricas.

```
# Tabela de contingência entre 'Gender' e 'Subscription Status'
pd.crosstab(df['Gender'], df['Subscription Status'])
```

### 7. Transformações de Dados

Durante a EAD, é comum aplicar transformações nos dados para facilitar a análise e melhorar a performance dos modelos futuros:

- **Normalização e padronização**: Caso haja variáveis com escalas muito diferentes (por exemplo, idade e valor da compra), pode ser necessário normalizar ou padronizar essas variáveis.

```
from sklearn.preprocessing import StandardScaler

# Padronizar a coluna 'Purchase Amount (USD)'
```

```
scaler = StandardScaler()
df['Purchase Amount (USD)'] = scaler.fit_transform(df[['Purchase Amount
(USD)']])
```

- **Criação de novas variáveis**: A criação de variáveis derivadas pode ser útil para a análise. Por exemplo, a criação de uma coluna que classifique os clientes em diferentes faixas de idade.

```
# Criação de faixas etárias
bins = [0, 18, 40, 60, 100]
labels = ['Jovem', 'Adulto', 'Meia-Idade', 'Idoso']
df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels)
```

### 8. Visualizações Avançadas

Além de visualizar distribuições e correlações, a visualização avançada pode ser usada para identificar padrões mais complexos, como:

- **Gráficos de dispersão**: Para entender a relação entre duas variáveis numéricas.

```
# Gráfico de dispersão entre 'Age' e 'Purchase Amount (USD)'
df.plot(kind='scatter', x='Age', y='Purchase Amount (USD)')
plt.show()
```

- **Gráficos de linhas ou séries temporais**: Se os dados incluírem uma variável temporal, podemos observar as tendências ao longo do tempo.

```
# Gráfico de série temporal (exemplo com data)
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
df['Purchase Amount (USD)'].resample('M').sum().plot()
plt.show()
```

### 9. Conclusões Preliminares

Após a análise exploratória de dados, é importante resumir as principais descobertas e insights. Isso inclui:

- **Padrões encontrados**: Como as variáveis estão distribuídas? Existe alguma tendência óbvia, como aumento no gasto durante uma determinada estação?
- **Correlação entre variáveis**: Quais variáveis estão fortemente correlacionadas? Quais não apresentam relação significativa?
- **Outliers e anomalias**: Quais dados podem estar distorcendo as análises?
- **Transformações necessárias**: Quais variáveis precisam ser transformadas ou criadas para melhorar a análise?

## Exercício – Concretiza os passos anteriores no VS Code num ficheiro .py