DATA SCIENCE & ANALYTICS

MBA

PYTHON

00000001

Table of Contents

1. Machine Learning 3
   1.1. NÃO SUPERVISIONADO ou EXPLORATÓRIO: 3
   1.2. Modelos supervisionados, ou confirmatórios 3
2. Banco de Dados 4
3. Tipos de Variáveis 4
   3.1. Variáveis Qualitativas 4
   3.2. Variáveis Quantitativas 4
   3.3. Detalhamento das Variáveis 5
4. PACOTES PYTHON 6
   4.1. pandas 6
      4.1.1. Aplicabilidade do Pandas 6
      4.1.2. Principais Estruturas de Dados 6
      4.1.3. Principais Comandos 7
   4.2. NumPy 11
      4.2.1. Aplicabilidade do NumPy 11
      4.2.2. Estruturas de Dados do NumPy 11
      4.2.3. Principais Comandos 12
   4.3. Matplotlib 15
      4.3.1. Aplicabilidade do Matplotlib 15
      4.3.2. Principais Comandos 15
   4.4. Seaborn 19
      4.4.1. Aplicabilidade do Seaborn 19
      4.4.2. Principais Comandos 19
   4.5. Plotly 22
      4.5.1. Aplicabilidade do Plotly 22
      4.5.2. Principais Comandos 22

1. MACHINE LEARNING

Machine learning, ou aprendizado de máquina, pode ser definido como a utilização de dados e algoritmos, incluindo métodos estatísticos, para produzir informações relevantes para a tomada de decisão. Este campo abrange diversas definições, mas essencialmente se concentra em criar modelos preditivos ou analisar a interdependência entre dados. Por exemplo, pode-se usar machine learning para prever tendências de mercado ou compreender as relações entre variáveis em um conjunto de dados.

1.1. NÃO SUPERVISIONADO OU EXPLORATÓRIO:

A escolha da técnica adequada de machine learning depende da natureza das variáveis e do objetivo da análise. Modelos não supervisionados, ou exploratórios, investigam a relação entre variáveis sem criar modelos confirmatórios e sem inferência para observações fora da amostra. Seus objetivos incluem a redução de dados, classificação, agrupamento de observações e correlação entre variáveis.

1.2. MODELOS SUPERVISIONADOS OU CONFIRMATÓRIOS

Modelos supervisionados, ou confirmatórios, têm como objetivo estimar modelos para elaborar previsões, permitindo inferência para outras observações fora da amostra. A escolha entre modelos supervisionados e não supervisionados depende do propósito da análise e do tipo de inferências desejadas.

2. BANCO DE DADOS

- O banco de dados é o objeto onde se armazenam informações de interesse para análise ou estudo.
- A composição de um banco de dados inclui variáveis e observações. As variáveis representam características ou atributos observados, medidos ou categorizados, enquanto as observações são as unidades cujas características e atributos são medidos.
- Estruturalmente, os bancos de dados geralmente organizam variáveis em colunas e observações em linhas, formando uma estrutura tabular.

3. TIPOS DE VARIÁVEIS

As variáveis podem ser classificadas em métricas e não métricas. Variáveis métricas são quantitativas e podem ser mensuradas ou contadas. Variáveis não métricas são qualitativas e representam características que não podem ser medidas diretamente, sendo frequentemente categóricas. Identificar o tipo de variável é crucial para escolher o método estatístico apropriado para a análise dos dados.

3.1. VARIÁVEIS QUALITATIVAS

As variáveis qualitativas são representadas por tabelas de distribuição de frequências ou gráficos. A moda pode ser calculada para estas variáveis, identificando a categoria que mais se repete. No entanto, medidas como média ou desvio padrão não são aplicáveis a variáveis qualitativas. Exemplos incluem escalas Likert, nacionalidade, cor do veículo e profissão.

3.2. VARIÁVEIS QUANTITATIVAS

Variáveis quantitativas podem ser representadas por gráficos e medidas de posição, dispersão e forma. Medidas de posição incluem média, mediana, percentis e quartis. Medidas de dispersão abrangem variância e desvio padrão, enquanto medidas de forma incluem assimetria e curtose. Exemplos de variáveis quantitativas são idade, renda em

Reais, número de habitantes, distância em metros e rentabilidade percentual de uma empresa.

3.3. Detalhamento das Variáveis

Variáveis qualitativas podem ser dicotômicas (duas categorias) ou policotômicas (mais de duas categorias), e podem ser nominais (sem ordem) ou ordinais (com ordem). Variáveis quantitativas podem ser discretas, com um conjunto finito e numerável de valores, geralmente derivados de contagens, ou contínuas, que assumem valores em um intervalo de números reais.

4. PYTHON

principais

4.1. PANDAS

Pandas é uma biblioteca de código aberto em Python, amplamente utilizada para manipulação e análise de dados. Oferece estruturas de dados e ferramentas de análise de alto desempenho, projetadas para tornar o trabalho com dados estruturados ou de tabela simples e intuitivo. Abaixo, detalhamos os principais comandos e a aplicabilidade do pacote Pandas, juntamente com exemplos práticos.

4.1.1. APLICABILIDADE DO PANDAS

Pandas é extremamente versátil e pode ser utilizado em várias etapas do ciclo de análise de dados:

- Limpeza de Dados: Remover ou corrigir dados incorretos, duplicados ou faltantes.
- Transformação de Dados: Converter formatos de dados, normalizar e criar novas variáveis.
- Análise Exploratória de Dados (EDA): Resumir as principais características dos dados com estatísticas descritivas e visualizações.
- Preparação de Dados para Modelagem: Pré-processar os dados para algoritmos de machine learning.
- Relatórios e Dashboards: Gerar relatórios de dados e dashboards para visualização e tomada de decisão.

4.1.2. PRINCIPAIS ESTRUTURAS DE DADOS

- Series: Uma série é um array unidimensional que pode armazenar qualquer tipo de dados (inteiros, strings, floats, etc.). Cada elemento tem um rótulo de índice associado.

```
import pandas as pd
s = pd.Series([1, 3, 5, 7, 9])
print(s)
```

- DataFrame: Um DataFrame é uma estrutura de dados bidimensional, como uma tabela, onde cada coluna pode ter um tipo de dado diferente (números, strings, etc.).

```
data = {
    'Nome': ['Ana', 'Bruno', 'Carlos'],
    'Idade': [22, 35, 58],
    'Cidade': ['São Paulo', 'Rio de Janeiro', 'Belo Horizonte']
}
df = pd.DataFrame(data)
print(df)
```

4.1.3. Principais Comandos

- .drop(columns=[ ]): Remove uma ou mais colunas de um DataFrame.

- .drop(index=[ ]): Remove uma ou mais linhas de um DataFrame.

- .info( ): Fornece um resumo informativo do DataFrame.

- .head( ): Visualiza as primeiras n linhas do DataFrame.

- Leitura de Dados: Pandas pode ler dados de diversos formatos, como CSV, Excel, SQL, JSON, entre outros.

```python
# Leitura de um arquivo CSV
df = pd.read_csv('dados.csv')

# Leitura de um arquivo Excel
df = pd.read_excel('dados.xlsx')
```

- Visualização de Dados: Comandos para visualizar o início, final e amostra dos dados.

```python
# Mostrar as primeiras 5 linhas
print(df.head())

# Mostrar as últimas 5 linhas
print(df.tail())

# Mostrar uma amostra aleatória de 5 linhas
print(df.sample(5))
```

- Informação dos Dados: Comandos para obter informações sobre o DataFrame.

```python
# Informações gerais do DataFrame
print(df.info())

# Descrição estatística dos dados numéricos
print(df.describe())
```

- Seleção e Filtro de Dados: Selecionar colunas e filtrar linhas com base em condições.

```python
# Selecionar uma coluna
print(df['Nome'])

# Selecionar múltiplas colunas
print(df[['Nome', 'Idade']])

# Filtrar linhas com base em uma condição
print(df[df['Idade'] > 30])
```

- Manipulação de Dados: Adicionar, remover ou modificar colunas e linhas.

```python
# Adicionar uma nova coluna
df['Salário'] = [5000, 6000, 7000]

# Remover uma coluna
df.drop('Cidade', axis=1, inplace=True)

# Modificar valores de uma coluna
df['Idade'] = df['Idade'] + 1
```

- Agrupamento e Agregação: Agrupar dados e aplicar funções agregadas.

```python
# Agrupar dados pela coluna 'Idade' e calcular a média do salário
df_grouped = df.groupby('Idade').mean()
print(df_grouped)
```

- Operações com Dados Faltantes: Identificar, remover ou substituir dados faltantes.

```python
# Identificar dados faltantes
print(df.isnull().sum())

# Remover linhas com dados faltantes
df.dropna(inplace=True)
```

4.2. NumPy

NumPy é uma biblioteca fundamental para computação científica em Python, oferecendo suporte para arrays e matrizes multidimensionais, além de uma coleção abrangente de funções matemáticas para operações com esses arrays. A seguir, apresentamos os principais comandos e a aplicabilidade do pacote NumPy, juntamente com exemplos práticos.

4.2.1. Aplicabilidade do NumPy

NumPy é amplamente utilizado em diversas áreas da ciência de dados e engenharia, devido à sua eficiência e versatilidade:

- Análise de Dados: Manipulação e análise eficiente de grandes conjuntos de dados.
- Computação Científica: Resolução de problemas científicos e de engenharia que requerem cálculos numéricos intensivos.
- Machine Learning: Pré-processamento de dados e implementação de algoritmos de aprendizado.
- Processamento de Imagens: Manipulação de matrizes que representam imagens para operações de transformação e análise.

4.2.2. Estruturas de Dados do NumPy

- Arrays: A estrutura principal do NumPy é o ndarray, que representa um array N-dimensional. Arrays são mais eficientes em termos de memória e performance comparados às listas padrão do Python.

```python
import numpy as np

# Criar um array unidimensional
arr1 = np.array([1, 2, 3, 4, 5])

# Criar um array bidimensional
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
```

4.2.3. Principais Comandos

- Criação de Arrays: NumPy oferece diversas funções para criar arrays de diferentes maneiras.

```
# Array de zeros
zeros = np.zeros((3, 3))

# Array de uns
ones = np.ones((2, 4))

# Array de valores aleatórios
random = np.random.random((2, 2))

# Array com valores espaçados uniformemente
linspace = np.linspace(0, 10, 5)  # De 0 a 10, com 5 elementos
```

- Operações Matemáticas Básicas: NumPy permite realizar operações matemáticas básicas em arrays.

```
a = np.array([10, 20, 30, 40])
b = np.array([1, 2, 3, 4])

# Adição
print(a + b)

# Subtração
print(a - b)

# Multiplicação
print(a * b)

# Divisão
print(a / b)
```

- Operações Estatísticas: Funções para calcular estatísticas básicas.

```python
data = np.array([1, 2, 3, 4, 5])

# Média
print(np.mean(data))

# Mediana
print(np.median(data))

# Desvio padrão
print(np.std(data))

# Variância
print(np.var(data))
```

- Indexação e Fatiamento: Similar às listas em Python, mas com mais funcionalidades.

```python
arr = np.array([1, 2, 3, 4, 5])

# Acessar elemento
print(arr[2])

# Fatiar array
print(arr[1:4])

# Acessar com passos
print(arr[::2])
```

- Manipulação de Arrays: Modificar a forma e a estrutura dos arrays.

```
arr = np.array([[1, 2, 3], [4, 5, 6]])

# Transpor array
print(arr.T)

# Redimensionar array
reshaped = arr.reshape((3, 2))

# Concatenar arrays
arr1 = np.array([1, 2])
arr2 = np.array([3, 4])
print(np.concatenate((arr1, arr2)))
```

4.3. Matplotlib

Matplotlib é uma biblioteca amplamente utilizada em Python para a criação de gráficos e visualizações de dados. É uma ferramenta essencial para cientistas de dados e analistas, permitindo a criação de uma ampla variedade de gráficos com uma interface fácil de usar. A seguir, apresentamos um resumo do Matplotlib, destacando os principais comandos e sua aplicabilidade, juntamente com exemplos práticos.

4.3.1. Aplicabilidade do Matplotlib

Matplotlib é utilizado em diversas áreas para visualização de dados, incluindo:

- Análise de Dados: Facilita a exploração e compreensão de conjuntos de dados através de visualizações claras e intuitivas.
- Ciência de Dados: Utilizado para visualizar resultados de análises e modelos de machine learning.
- Financeiro: Criação de gráficos de séries temporais para análise de tendências de mercado.
- Engenharia e Pesquisa Científica: Visualização de dados experimentais e resultados de simulações.

4.3.2. Principais Comandos

- Importação da Biblioteca:

Antes de usar o Matplotlib, é necessário importá-lo. Normalmente, importa-se o módulo pyplot com o alias plt.

```
import matplotlib.pyplot as plt
```

- Gráficos de Linha:

```
# Gráfico de Linha
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
plt.plot(x, y)
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Line Plot')
plt.show()
```

- Gráficos de Dispersão:

```
# Gráfico de Dispersão
plt.scatter(x, y)
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Scatter Plot')
plt.show()
```

- Gráficos de Barras:

```
# Gráfico de Barras
categorias = ['A', 'B', 'C', 'D']
valores = [5, 7, 3, 8]
plt.bar(categorias, valores)
plt.xlabel('Categorias')
plt.ylabel('Valores')
plt.title('Bar Plot')
plt.show()
```

- Histogramas:

```
# Histograma
dados = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
plt.hist(dados, bins=5, edgecolor='black')
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.title('Histogram')
plt.show()
```

- Gráficos de Pizza:

```
# Gráfico de Pizza
labels = ['A', 'B', 'C', 'D']
sizes = [15, 30, 45, 10]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Pie Chart')
plt.show()
```

- Customização de Gráficos: Matplotlib oferece diversas opções para personalizar gráficos, como cores, estilos de linha, e anotações.

```
# Customização de Gráfico de Linha
plt.plot(x, y, color='green', linestyle='dashed', marker='o', markerfacecolor='blue')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Customized Line Plot')
plt.grid(True)
plt.show()
```

- Subplots: É possível criar múltiplos gráficos em uma única figura usando subplots.

```
# Subplots
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(x, y)
plt.title('Subplot 1')

plt.subplot(1, 2, 2)
plt.bar(categorias, valores)
plt.title('Subplot 2')

plt.tight_layout()
plt.show()
```

4.4. Seaborn

Seaborn é uma biblioteca de visualização de dados baseada no Matplotlib, que fornece uma interface de alto nível para a criação de gráficos estatísticos atraentes e informativos. Ele é particularmente útil para explorar e entender melhor os dados. A seguir, apresentamos um resumo do Seaborn, destacando os principais comandos e sua aplicabilidade, juntamente com exemplos práticos.

4.4.1. Aplicabilidade do Seaborn

Seaborn é amplamente utilizado em várias etapas da análise de dados devido à sua capacidade de criar visualizações estatísticas complexas com simplicidade:

- Análise Exploratória de Dados (EDA): Facilita a compreensão das distribuições e relações entre variáveis.
- Visualização de Dados Estatísticos: Criação de gráficos estatísticos que ajudam a destacar tendências e padrões nos dados.
- Comparação entre Variáveis: Visualizações que permitem a comparação entre diferentes grupos e categorias de dados.
- Correlação e Relações: Visualização de correlações e interdependências entre múltiplas variáveis.

4.4.2. Principais Comandos

- Importação da Biblioteca:

Antes de usar o Seaborn, é necessário importá-lo.

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

- Gráficos de Dispersão (Scatter Plots): Utilizados para examinar a relação entre duas variáveis.

```
# Dataset de exemplo
data = sns.load_dataset('tips')

# Gráfico de dispersão
sns.scatterplot(x='total_bill', y='tip', data=data)
plt.title('Scatter Plot of Total Bill vs Tip')
plt.show()
```

- Gráficos de Barras (Bar Plots): Usados para mostrar a média de uma variável em diferentes categorias.

```
# Gráfico de barras
sns.barplot(x='day', y='total_bill', data=data)
plt.title('Bar Plot of Total Bill by Day')
plt.show()
```

- Histogramas e Densidade (Histograms and KDE): Utilizados para mostrar a distribuição de uma variável.

```
# Histograma
sns.histplot(data['total_bill'], bins=20, kde=True)
plt.title('Histogram and KDE of Total Bill')
plt.show()
```

- Box Plots: Utilizados para mostrar a distribuição de uma variável através de quartis.

```
# Box plot
sns.boxplot(x='day', y='total_bill', data=data)
plt.title('Box Plot of Total Bill by Day')
plt.show()
```

- Heatmaps: Utilizados para visualizar matrizes de dados, como tabelas de correlação.

```
# Heatmap de correlação
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Heatmap of Correlation Matrix')
plt.show()
```

- Pair Plots: Utilizados para criar uma matriz de gráficos de dispersão entre múltiplas variáveis.

```
# Pair plot
sns.pairplot(data)
plt.suptitle('Pair Plot of the Dataset', y=1.02)
plt.show()
```

- Facet Grids: Utilizados para criar múltiplos gráficos categorizados por uma ou mais variáveis.

```
# Facet grid
g = sns.FacetGrid(data, col='sex', row='time')
g.map(sns.histplot, 'total_bill')
g.add_legend()
plt.show()
```

4.5. Plotly

Plotly é uma biblioteca de gráficos interativos em Python que permite a criação de visualizações dinâmicas e interativas. É amplamente utilizada para criar dashboards e gráficos complexos que podem ser incorporados em aplicações web. Plotly suporta uma vasta gama de tipos de gráficos, desde gráficos simples de linha e dispersão até gráficos 3D e mapas geoespaciais. A seguir, apresentamos um resumo do Plotly, destacando os principais comandos e sua aplicabilidade, juntamente com exemplos práticos.

4.5.1. Aplicabilidade do Plotly

Plotly é amplamente utilizado em várias áreas devido à sua capacidade de criar gráficos interativos e visualmente atraentes:

- Análise de Dados: Facilita a exploração e compreensão de grandes conjuntos de dados através de gráficos interativos.
- Desenvolvimento de Dashboards: Criação de painéis interativos que podem ser incorporados em aplicações web.
- Visualização Científica: Criação de visualizações complexas para comunicar resultados científicos.
- Business Intelligence: Desenvolvimento de ferramentas interativas para a análise de dados empresariais e tomada de decisão.

4.5.2. Principais Comandos

- Importação da Biblioteca: Antes de usar o Plotly, é necessário importá-lo.

```
import plotly.express as px
import plotly.graph_objects as go
```

- Gráficos de Linha (Line Charts): Utilizados para mostrar tendências ao longo do tempo.

```
# Dados de exemplo
df = px.data.gapminder().query("country == 'Canada'")

# Gráfico de linha
fig = px.line(df, x='year', y='gdpPercap', title='GDP Per Capita over Time')
fig.show()
```

- Gráficos de Dispersão (Scatter Plots): Utilizados para examinar a relação entre duas variáveis.

```
# Dados de exemplo
df = px.data.iris()

# Gráfico de dispersão
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='species', title='Sepal
fig.show()
```

- Gráficos de Barras (Bar Charts): Usados para mostrar comparações entre categorias.

```
# Dados de exemplo
df = px.data.gapminder().query("year == 2007")

# Gráfico de barras
fig = px.bar(df, x='continent', y='pop', color='continent', title='Population by Cont
fig.show()
```

- Histogramas (Histograms): Utilizados para mostrar a distribuição de uma variável.

```
# Dados de exemplo
df = px.data.tips()

# Histograma
fig = px.histogram(df, x='total_bill', nbins=20, title='Total Bill Distribution')
fig.show()
```

- Box Plots: Utilizados para mostrar a distribuição de uma variável através de quartis.

```
# Dados de exemplo
df = px.data.tips()

# Box plot
fig = px.box(df, x='day', y='total_bill', title='Total Bill by Day')
fig.show()
```

- Heatmaps: Utilizados para visualizar matrizes de dados.

```
# Dados de exemplo
z = [[1, 20, 30], [20, 1, 60], [30, 60, 1]]

# Heatmap
fig = go.Figure(data=go.Heatmap(z=z, x=['X1', 'X2', 'X3'], y=['Y1', 'Y2', 'Y3']))
fig.update_layout(title='Heatmap Example')
fig.show()
```

- Mapas Geoespaciais: Utilizados para visualizações geográficas.

```
# Dados de exemplo
df = px.data.gapminder().query("year == 2007")

# Mapa de dispersão geoespacial
fig = px.scatter_geo(df, locations='iso_alpha', color='continent', hover_name='country
fig.show()
```