Tutorial de Pandas

Neste tutorial, vamos explorar os conceitos essenciais da biblioteca Pandas, que é amplamente utilizada para análise de dados e manipulação de estruturas de dados tabulares. Vamos abordar os principais componentes do Pandas, incluindo Series, DataFrame e as operações básicas de análise exploratória e pré-processamento de dados. Vamos começar!

1. Introdução ao Pandas

O Pandas é uma biblioteca de código aberto que fornece estruturas de dados de alto desempenho e ferramentas de análise de dados fáceis de usar. Duas estruturas principais do Pandas são a Series e o DataFrame.

1.1 Series

Uma Series é uma estrutura unidimensional que pode armazenar qualquer tipo de dado. É semelhante a uma coluna em uma planilha ou uma coluna em uma tabela de banco de dados.

```
import pandas as pd

# Criando uma Series
s = pd.Series([1, 3, 5, np.nan, 6, 8])

# Exibindo a Series
print(s)
```

1.2 DataFrame

Um DataFrame é uma estrutura bidimensional que pode armazenar dados de diferentes tipos. É semelhante a uma planilha ou uma tabela de banco de dados, com linhas e colunas rotuladas.

```
import pandas as pd

# Criando um DataFrame a partir de um dicionário
data = {'Nome': ['João', 'Maria', 'Ana', 'Pedro'],
        'Idade': [25, 30, 28, 35],
        'Cidade': ['São Paulo', 'Rio de Janeiro', 'Belo Horizonte', 'Porto
Alegre']}
df = pd.DataFrame(data)

# Exibindo o DataFrame
print(df)
```

2. Carregando Dados

O Pandas oferece várias maneiras de carregar dados em um DataFrame. Os formatos suportados incluem CSV, Excel, SQL, entre outros.

```
import pandas as pd

# Carregando dados de um arquivo CSV
df = pd.read_csv('dados.csv')

# Exibindo as primeiras linhas do DataFrame
print(df.head())
```

3. Análise Exploratória de Dados

A análise exploratória de dados é uma etapa crucial para entender e resumir as características dos dados. Vamos explorar algumas operações básicas.

3.1 Informações Gerais

```
# Verificando a quantidade de linhas e colunas
print(df.shape)

# Exibindo os nomes das colunas
print(df.columns)

# Verificando os tipos de dados das colunas
print(df.dtypes)

# Exibindo informações gerais do DataFrame
print(df.info())
```

3.2 Estatísticas Descritivas

```
# Calculando estatísticas descritivas básicas
print(df.describe())

# Calculando a média de uma coluna
print(df['Coluna'].mean())

# Calculando a contagem de valores únicos em uma coluna
print(df['Coluna'].value_counts())
```

4. Pré-processamento de Dados

O pré-processamento de dados é uma etapa importante para tratar valores ausentes, realizar filtragens e transformações nos dados.

4.1 Valores Ausentes

```
# Verificando valores

 ausentes
print(df.isnull().sum())

# Preenchendo valores ausentes com a média da coluna
df['Coluna'].fillna(df['Coluna'].mean(), inplace=True)

# Removendo linhas com valores ausentes
df.dropna(inplace=True)
```

4.2 Filtragem de Dados

```
# Filtrando dados com base em uma condição
filtro = df['Coluna'] > 10
df_filtrado = df[filtro]
```

4.3 Ordenação de Dados

```
# Ordenando o DataFrame com base em uma coluna
df_ordenado = df.sort_values(by='Coluna', ascending=False)
```

5. Exportando Dados

Após realizar as análises e pré-processamento, é possível exportar os dados para uso futuro.

```
# Exportando DataFrame para um arquivo CSV
df.to_csv('dados_processados.csv', index=False)
```

6. Manipulação de Dados

Além das operações básicas de análise exploratória e pré-processamento, o Pandas oferece várias funcionalidades para manipular e transformar os dados em um DataFrame.

6.1 Adicionar Colunas

```
# Adicionando uma nova coluna com base em cálculos
df['Nova Coluna'] = df['Coluna1'] + df['Coluna2']
```

```
# Adicionando uma nova coluna com base em condições
df['Nova Coluna'] = np.where(df['Coluna'] > 10, 'Maior', 'Menor')
```

6.2 Selecionar Colunas e Linhas

```
# Selecionando uma coluna específica
coluna = df['Coluna']

# Selecionando múltiplas colunas
colunas = df[['Coluna1', 'Coluna2']]

# Selecionando linhas com base em uma condição
linhas_filtradas = df[df['Coluna'] > 10]

# Selecionando linhas com base em múltiplas condições
linhas_filtradas = df[(df['Coluna1'] > 5) & (df['Coluna2'] < 3)]
```

6.3 Agrupamento de Dados

```
# Agrupando dados com base em uma coluna e calculando a média de outra
coluna
agrupamento = df.groupby('Coluna1')['Coluna2'].mean()

# Agrupando dados com base em múltiplas colunas e calculando estatísticas
descritivas
agrupamento = df.groupby(['Coluna1', 'Coluna2']).agg({'Coluna3': 'mean',
'Coluna4': 'sum'})
```

6.4 Pivot Tables

```
# Criando uma tabela dinâmica para resumir os dados
tabela_dinamica = pd.pivot_table(df, values='Valor', index='Coluna1',
columns='Coluna2', aggfunc=np.mean)
```

7. Visualização de Dados

O Pandas também possui integração com outras bibliotecas de visualização de dados, como o Matplotlib e o Seaborn, permitindo a criação de gráficos e visualizações atraentes dos dados.

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

# Criando um gráfico de barras
sns.barplot(x='Coluna1', y='Coluna2', data=df)
```

```
plt.show()

# Criando um gráfico de dispersão
sns.scatterplot(x='Coluna1', y='Coluna2', data=df)
plt.show()
```

8. Lidando com Dados Ausentes

Em muitos conjuntos de dados reais, é comum encontrar valores ausentes ou faltantes. O Pandas fornece métodos para lidar com esses dados ausentes de maneira eficiente.

8.1 Identificando Valores Ausentes

```
# Verificando se há valores ausentes em cada coluna
print(df.isnull().sum())

# Verificando se há valores ausentes em todo o DataFrame
print(df.isnull().any().any())
```

8.2 Tratando Valores Ausentes

Existem várias estratégias para lidar com valores ausentes, e a escolha depende do contexto e da natureza dos dados. Algumas opções comuns são:

8.2.1 Remoção de Valores Ausentes

```
# Removendo linhas com valores ausentes
df.dropna()

# Removendo colunas com valores ausentes
df.dropna(axis=1)

# Removendo linhas com valores ausentes em colunas específicas
df.dropna(subset=['Coluna1', 'Coluna2'])
```

8.2.2 Preenchimento de Valores Ausentes

```
# Preenchendo valores ausentes com um valor específico
df.fillna(value)

# Preenchendo valores ausentes com a média da coluna
df['Coluna'].fillna(df['Coluna'].mean(), inplace=True)

# Preenchendo valores ausentes com a mediana da coluna
df['Coluna'].fillna(df['Coluna'].median(), inplace=True)
```

8.3 Interpolação de Valores Ausentes

A interpolação é outra técnica útil para preencher valores ausentes com base em valores existentes próximos.

```
# Preenchendo valores ausentes usando interpolação linear
df['Coluna'].interpolate(method='linear', inplace=True)
```

9. Trabalhando com Datas e Horários

O Pandas possui recursos poderosos para trabalhar com dados de datas e horários, permitindo realizar análises temporais de maneira eficiente.

9.1 Conversão de Colunas para Tipos de Datas

```
# Convertendo uma coluna para o tipo de data
df['Data'] = pd.to_datetime(df['Data'])

# Convertendo uma coluna para o tipo de data e hora
df['DataHora'] = pd.to_datetime(df['DataHora'])
```

9.2 Extração de Componentes de Datas

```
# Extraindo o ano de uma coluna de datas
df['Ano'] = df['Data'].dt.year

# Extraindo o mês de uma coluna de datas
df['Mês'] = df['Data'].dt.month

# Extraindo o dia da semana de uma coluna de datas
df['DiaSemana'] = df['Data'].dt.dayofweek
```

9.3 Resample de Dados Temporais

```
# Agrupando e resampleando dados temporais por dia
df.resample('D').mean()

# Agrupando e resampleando dados temporais por mês
df.resample('M').sum()
```

Conclusão

Neste tutorial, exploramos técnicas importantes para lidar com dados ausentes, além de mostrar como trabalhar com dados de datas e horários. Essas habilidades serão valiosas na

análise de dados reais, onde é comum encontrar valores ausentes e dados temporais. Continue explorando a documentação do Pandas para obter mais informações sobre esses tópicos e amplie suas habilidades em manipulação e análise de dados.

Trabalho 2:Análise Exploratória e Pré-processamento de Dados com Pandas

Agora vamos explorar conceitos mais avançados, incluindo os comandos `groupby` e `pivot_table`, além de técnicas de visualização de dados.

Trabalho Avançado de Análise Exploratória e Pré-processamento de Dados com Pandas

Olá, alunos! Este é o segundo trabalho de análise exploratória e pré-processamento de dados utilizando a biblioteca Pandas. Desta vez, vamos explorar conceitos mais avançados, incluindo os métodos `groupby` e `pivot_table`, que são ferramentas poderosas para análise de dados. Vocês já receberam um dataset com dados reais de COVID. Agora, vamos aos exercícios!

Exercícios utilizando o método `groupby`:

Exercício 1:
a) Agrupe os dados por mês e calcule a soma dos casos confirmados.
b) Ordene o resultado em ordem crescente com base na soma dos casos confirmados.
c) Exiba as primeiras cinco linhas do resultado.

Exercício 2:
a) Agrupe os dados por mês e calcule a média dos óbitos.
b) Ordene o resultado em ordem decrescente com base na média dos óbitos.
c) Exiba as últimas cinco linhas do resultado.

Exercício 3:
a) Agrupe os dados por idade e calcule a contagem dos casos recuperados.
b) Ordene o resultado em ordem decrescente com base na contagem dos casos recuperados.
c) Exiba as primeiras 10 linhas do resultado.

Exercício 4:
a) Agrupe os dados por idade e calcule a soma dos casos ativos.
b) Ordene o resultado em ordem crescente com base na soma dos casos ativos.
c) Exiba as últimas 10 linhas do resultado.

Exercício 5:
a) Agrupe os dados por idade e calcule a média da taxa de letalidade.
b) Ordene o resultado em ordem decrescente com base na média da taxa de letalidade.
c) Exiba as primeiras cinco linhas do resultado.

Exercício 6:
a) Agrupe os dados por mês e calcule a soma dos casos confirmados, óbitos e recuperados.
b) Exiba o resultado do agrupamento em uma tabela organizada.

Exercício 7:
a) Agrupe os dados por mês e calcule a média dos casos confirmados.
b) Crie um gráfico de linha para visualizar a evolução dos casos confirmados ao longo do tempo.

Exercício 8:
a) Agrupe os dados por mês e encontre a data com o maior número de casos confirmados.
b) Exiba o resultado do agrupamento mostrando a data com o maior número de casos confirmados.

Exercícios utilizando o método `pivot_table`:

Exercício 9:
a) Crie uma tabela dinâmica que resuma os dados de casos confirmados e óbitos por idade e mês.
b) Exiba a tabela dinâmica criada.

Exercício 10:
a) Crie uma tabela dinâmica que resuma os dados de casos confirmados e óbitos por idade e gênero.
b) Exiba a tabela dinâmica criada.

Exercício 11:
a) Crie uma tabela dinâmica que resuma os dados de casos confirmados e óbitos por mês e gênero.
b) Exiba a tabela dinâmica criada.

Exercício 12:
a) Crie uma tabela dinâmica que resuma os dados de casos confirmados, óbitos e recuperados por idade e gênero.
b) Exiba a tabela dinâmica criada.

Exercício 13:
a) Crie uma tabela dinâmica que resuma os dados de casos confirmados, óbitos e recuperados por mês e gênero.
b) Exiba a tabela dinâmica criada.

Exercício 14:
a) Crie uma tabela dinâmica que resuma os dados de casos confirmados, óbitos e recuperados por mês e faixa etária.
b) Exiba a tabela dinâmica criada.

Exercício 15:
a) Crie um gráfico de barras para visualizar a quantidade total de casos confirmados por idade.
b) Personalize o gráfico adicionando título, rótulos de eixo, etc.

Exercício 16:
a) Crie um gráfico de dispersão para visualizar a relação entre casos confirmados e óbitos por idade.
b) Adicione marcadores diferenciados para cada faixa etária.

Exercício 17:
a) Crie um gráfico de pizza para visualizar a distribuição dos casos confirmados por gênero.
b) Adicione uma legenda e destaque o gênero com maior número de casos.

Exercício 18:
a) Crie um gráfico de linha para visualizar a evolução dos casos confirmados, óbitos e recuperados ao longo do tempo para uma faixa etária específica.
b) Adicione legendas e rótulos adequados para o gráfico.

Lembre-se de documentar cada etapa do trabalho e fornecer explicações claras para cada exercício. Utilizem as funções do Pandas e do Matplotlib para realizar as análises e visualizações. Boa sorte e aproveitem a prática com o Pandas!