AULA- 6 CÓDIGOS

terça-feira, 19 de setembro de 2023 20:56

Análise dos quartis de uma série de dados:

```
import pandas as pd
# Conjunto de dados
dados = pd.Series([42, 53, 36, 28, 26, 41, 37, 40, 48, 45,
                   19, 38, 36, 56, 43, 34, 52, 38, 50, 43])
# Determinação dos três quartis
quartis = dados.quantile([0.25, 0.5, 0.75], interpolation='linear')
print('Quartis em um Panda Series:')
display(quartis) # É um objeto panda Series
# Mostrando os quartis ao usuário
print('Desmembrando os quartis:')
lquartis = list(quartis)
for i in range(len(lquartis)):
  print(f'Q{str(i+1)} = {lquartis[i]}')
# Calculando a amplitude interquartil
interquartil = quartis[0.75] - quartis[0.25]
print(f'Amplitude interquartil: {interquartil}')
# Limites para detecção de outliers
lim_min = quartis[0.25] - 1.5*interquartil
lim_max = quartis[0.75] + 1.5*interquartil
print('Limites de detecção para outliers:')
print(f'Inferior = {lim_min}')
print(f'Superior = {lim_max}')
# Procurando por outliers
out_inf = dados.loc[dados <= lim_min]
print('Outlier inferior:')
display(out_inf)
out_sup = dados.loc[dados >= lim_max]
print('Outlier superior:')
display(out_sup)
```

import pandas as pd

- Aqui, você está importando a biblioteca Pandas para trabalhar com estruturas de dados de séries temporais e dados tabulares.

# Conjunto de dados
dados = pd.Series([42, 53, 36, 28, 26, 41, 37, 40, 48, 45,
19, 38, 36, 56, 43, 34, 52, 38, 50, 43])

- Neste trecho, você está criando uma série de dados chamada `dados` usando Pandas. Esta série contém uma lista de valores numéricos.

# Determinação dos três quartis
quartis = dados.quantile([0.25, 0.5, 0.75], interpolation='linear')

- Aqui, você está calculando os três quartis (primeiro quartil, mediana e terceiro quartil) dos dados usando o método `quantile()`. A opção `interpolation='linear'` indica que o método deve usar interpolação linear ao calcular os quartis.

print('Quartis em um Panda Series:')

```
display(quartis) # É um objeto panda Series
```

- Este trecho exibe os quartis calculados na saída. A função `display()` é usada para mostrar o objeto Pandas que contém os quartis.

```
# Mostrando os quartis ao usuário
print('Desmembrando os quartis:')
lquartis = list(quartis)
for i in range(len(lquartis)):
  print(f'Q{str(i+1)} = {lquartis[i]}')
```

- Aqui, você está exibindo os quartis individualmente ao usuário. O objeto `quartis` é convertido em uma lista, e um loop é usado para mostrar cada quartil com um rótulo.

```
# Calculando a amplitude interquartil
interquartil = quartis[0.75] - quartis[0.25]
print(f'Amplitude interquartil: {interquartil}')
```

- Este trecho calcula a amplitude interquartil, que é a diferença entre o terceiro quartil (0.75) e o primeiro quartil (0.25). A amplitude interquartil é uma medida de dispersão que indica a variação no meio dos dados.

```
# Limites para detecção de outliers
lim_min = quartis[0.25] - 1.5 * interquartil
lim_max = quartis[0.75] + 1.5 * interquartil
print('Limites de detecção para outliers:')
print(f'Inferior = {lim_min}')
print(f'Superior = {lim_max}')
```

- Aqui, você está calculando os limites para detecção de outliers com base na regra do "1,5 vezes o IQR" (amplitude interquartil). São calculados limites inferior e superior para identificar valores que são considerados outliers.

```
# Procurando por outliers
out_inf = dados.loc[dados <= lim_min]
print('Outlier inferior:')
display(out_inf)
out_sup = dados.loc[dados >= lim_max]
print('Outlier superior:')
display(out_sup)
```

- Este trecho procura por outliers nos dados. Os valores que estão abaixo do limite inferior (`lim_min`) ou acima do limite superior (`lim_max`) são identificados como outliers e são exibidos na saída.

No geral, o código realiza uma análise estatística básica dos dados, calculando quartis, amplitude interquartil e identificando outliers com base nos limites estabelecidos.

Customizando as separatrizes:

```
# Determinação dos percentis customizados
percentis = dados.quantile([0.1, 0.35, 0.68, 0.95], interpolation='linear')
print('Percentis em um Panda Series:')
display(percentis) # É um objeto panda Series
```

# Determinação dos percentis customizados
percentis = dados.quantile([0.1, 0.35, 0.68, 0.95], interpolation='linear')

- Neste trecho, você está calculando percentis customizados dos dados usando o método `quantile()`. Você especificou uma lista de percentis desejados `[0.1, 0.35, 0.68, 0.95]` para serem calculados. A opção `interpolation='linear'` indica que o método deve usar interpolação linear ao calcular os percentis.

print('Percentis em um Panda Series:')
display(percentis) # É um objeto panda Series

- Este trecho exibe os percentis calculados na saída. A função `display()` é usada para mostrar o objeto Pandas que contém os percentis.

Os percentis são medidas estatísticas que indicam o valor abaixo do qual uma porcentagem específica dos dados está localizada. Neste caso, você calculou os percentis personalizados de 10%, 35%, 68% e 95% dos seus dados, o que pode ser útil para entender a distribuição dos valores em diferentes pontos ao longo da série de dados.

Análise exploratória inicial:

```
# Análise exploratória
dados.describe()
```

O código `dados.describe()` realiza uma análise exploratória básica dos dados contidos na série `dados`. Vou explicar cada trecho deste código:

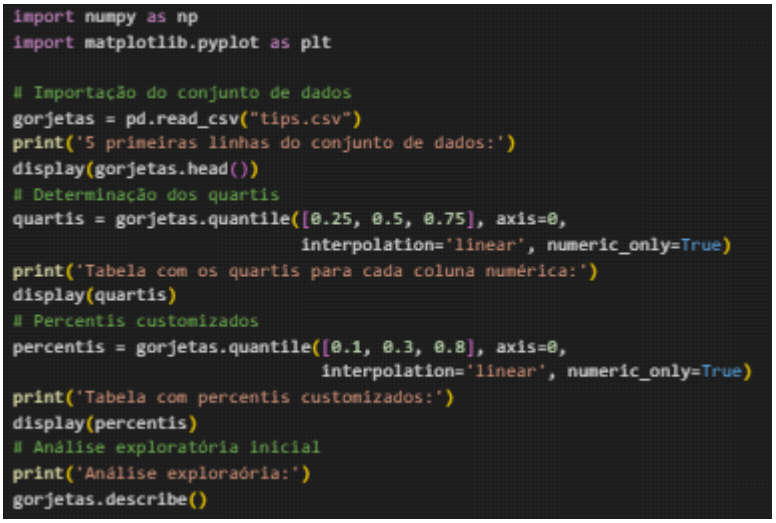
dados.describe()

- Este comando utiliza o método `describe()` da série Pandas `dados`. O método `describe()` é uma função de resumo estatístico que calcula várias estatísticas descritivas para os dados, incluindo:

- Contagem: O número de observações válidas.
- Média: A média aritmética dos valores.
- Desvio padrão: Uma medida de dispersão que indica a variação dos valores em relação à média.
- Mínimo: O valor mínimo nos dados.
- 25% (Primeiro quartil): O valor que separa os 25% inferiores dos dados.
- 50% (Mediana): O valor que separa os 50% inferiores dos dados, que é equivalente ao segundo quartil.
- 75% (Terceiro quartil): O valor que separa os 25% superiores dos dados.
- Máximo: O valor máximo nos dados.

O resultado desta função será uma tabela que apresenta todas essas estatísticas para a série `dados`. Isso é útil para obter uma visão geral rápida da distribuição e tendência central dos dados, bem como para identificar valores extremos (outliers) com base nos quartis.

Análise dos quartis de um painel de dados:



```
import numpy as np
import matplotlib.pyplot as plt

# Importação do conjunto de dados
gorjetas = pd.read_csv("tips.csv")
print('5 primeiras linhas do conjunto de dados:')
display(gorjetas.head())
# Determinação dos quartis
quartis = gorjetas.quantile([0.25, 0.5, 0.75], axis=0,
                            interpolation='linear', numeric_only=True)
print('Tabela com os quartis para cada coluna numérica:')
display(quartis)
# Percentis customizados
percentis = gorjetas.quantile([0.1, 0.3, 0.8], axis=0,
                              interpolation='linear', numeric_only=True)
print('Tabela com percentis customizados:')
display(percentis)
# Análise exploratória inicial
print('Análise exploraória:')
gorjetas.describe()
```

```
# Importação do conjunto de dados
gorjetas = pd.read_csv("tips.csv")
```

- Neste trecho, você está importando um conjunto de dados de um arquivo CSV chamado "tips.csv" usando a função `pd.read_csv()` do Pandas. Os dados são carregados em um DataFrame chamado `gorjetas`.

```
print('5 primeiras linhas do conjunto de dados:')
display(gorjetas.head())
```

- Aqui, você está exibindo as primeiras cinco linhas do conjunto de dados usando `gorjetas.head()`. O comando `display()` é usado para mostrar o DataFrame na saída.

```
# Determinação dos quartis
quartis = gorjetas.quantile([0.25, 0.5, 0.75], axis=0,
                            interpolation='linear', numeric_only=True)
```

- Neste trecho, você está calculando os quartis (primeiro quartil, mediana e terceiro quartil) para cada coluna numérica do DataFrame `gorjetas` usando a função `quantile()`. A opção `axis=0` indica que você deseja calcular os quartis ao longo das colunas. A interpolação linear (`interpolation='linear'`) é especificada, e `numeric_only=True` garante que apenas as colunas numéricas sejam consideradas para o cálculo.

```
print('Tabela com os quartis para cada coluna numérica:')
display(quartis)
```

- Aqui, você está exibindo uma tabela que contém os quartis calculados para cada coluna numérica. Novamente, a função `display()` é usada para mostrar o DataFrame na saída.

```
# Percentis customizados
percentis = gorjetas.quantile([0.1, 0.3, 0.8], axis=0,
                              interpolation='linear', numeric_only=True)
```

- Neste trecho, você está calculando percentis personalizados (10%, 30% e 80%) para cada coluna numérica do DataFrame `gorjetas` usando a função `quantile()`. As mesmas opções de interpolação e `numeric_only` são aplicadas aqui.

```
print('Tabela com percentis customizados:')
display(percentis)
```

- Aqui, você está exibindo uma tabela que contém os percentis personalizados calculados para cada coluna numérica.

```
# Análise exploratória inicial
print('Análise exploraória:')
gorjetas.describe()
```

- Por fim, você está realizando uma análise exploratória inicial dos dados usando `gorjetas.describe()`. Esta função fornece estatísticas descritivas, como contagem, média, desvio padrão e quartis para todas as colunas numéricas do DataFrame `gorjetas`. Essa análise oferece uma visão geral da distribuição e tendência central dos dados.

Boxplot - Série de dados:

```
# É preciso transformar o Pandas Series em um Pandas DataFrame
# Não há o método boxplot para Pandas Series
df = pd.DataFrame(dados)
df.boxplot(color = 'red', vert=True, whis=1.5, showmeans=True)
plt.title('Distribuição dos dados')
```

```
# É preciso transformar o Pandas Series em um Pandas DataFrame
df = pd.DataFrame(dados)
```

- Neste trecho, você está convertendo o Pandas Series chamado `dados` em um Pandas DataFrame chamado `df`. Isso é necessário porque a função `boxplot()` que você pretende usar posteriormente é aplicada a DataFrames, não a séries.

```
df.boxplot(color='red', vert=True, whis=1.5, showmeans=True)
```

- Aqui, você está criando um gráfico de caixa (boxplot) dos dados contidos no DataFrame `df`. Vou explicar os parâmetros usados na função `boxplot()`:

- `color='red'`: Define a cor das caixas do boxplot como vermelha.
- `vert=True`: Indica que você deseja que o boxplot seja orientado verticalmente.
- `whis=1.5`: Define o fator de alcance interquartil (IQR) para determinar os limites dos outliers. Um valor de 1.5 é comum nesse contexto.
- `showmeans=True`: Mostra uma linha representando a média dos dados no boxplot.

plt.title('Distribuição dos dados')

- Este trecho adiciona um título ao gráfico de caixa, que é definido como 'Distribuição dos dados'. O `plt.title()` é usado para definir o título do gráfico.

Em resumo, o código transforma um Pandas Series em um Pandas DataFrame e cria um gráfico de caixa para visualizar a distribuição dos dados contidos no DataFrame. O gráfico de caixa mostra informações sobre a mediana, quartis, possíveis outliers e a média dos dados.

Boxplot - Painéis de dados:

```
# Relembrando o formato dos dados
display(gorjetas.head())
gorjetas.boxplot(by='day', column=['total_bill'], color='red', showmeans=True)
plt.title('Distribuição do valor das contas')
plt.suptitle('')
```

Neste trecho de código, você está relembrando o formato dos dados e, em seguida, criando um gráfico de caixa (boxplot) condicional com base na coluna 'day' para visualizar a distribuição do valor das contas. Vou explicar cada parte:

# Relembrando o formato dos dados
display(gorjetas.head())

- Aqui, você está usando `display()` para exibir as primeiras linhas do DataFrame `gorjetas`, relembrando o formato dos dados. Isso é útil para verificar as primeiras observações e colunas do DataFrame.

gorjetas.boxplot(by='day', column=['total_bill'], color='red', showmeans=True)

- Este trecho cria o gráfico de caixa condicional. Vamos explicar os parâmetros utilizados:

- 'by='day'`: Você está agrupando os dados pelo valor da coluna 'day', o que significa que cada caixa do boxplot será criada para um dia específico da semana.
- 'column=['total_bill']`: Aqui, você está especificando que deseja criar o boxplot para a coluna 'total_bill', que representa o valor total da conta.
- 'color='red'`: Define a cor das caixas do boxplot como vermelha.
- 'showmeans=True`: Mostra uma linha representando a média dos valores de 'total_bill' para cada dia da semana no gráfico.

plt.title('Distribuição do valor das contas')
plt.suptitle('')

- Estes trechos definem o título do gráfico principal como 'Distribuição do valor das contas' usando `plt.title()`. O `plt.suptitle('')` é usado para remover o título superior gerado automaticamente pelo Pandas, deixando apenas o título principal.

O resultado é um gráfico de caixa que mostra a distribuição do valor das contas para cada dia da semana, permitindo visualizar a mediana, quartis, possíveis outliers e a média para cada categoria de dia. Essa visualização é útil para entender como os valores das contas variam ao longo dos dias da semana.

```
gorjetas.boxplot(by='day', column=['tip'], color='green', showmeans=True)
plt.title('Distribuição do valor das gorjetas')
plt.suptitle('')
```

gorjetas.boxplot(by='day', column=['tip'], color='green', showmeans=True)

- Neste trecho, você está criando um gráfico de caixa (boxplot) condicional usando o método `boxplot()` do Pandas:

  - `by='day'`: Você está agrupando os dados pelo valor da coluna 'day', o que significa que cada caixa do boxplot será criada para um dia específico da semana.
  - `column=['tip']`: Especifica que deseja criar o boxplot para a coluna 'tip', que representa o valor das gorjetas.
  - `color='green'`: Define a cor das caixas do boxplot como verde.
  - `showmeans=True`: Mostra uma linha representando a média dos valores de gorjeta ('tip') para cada dia da semana no gráfico.

plt.title('Distribuição do valor das gorjetas')

- Este trecho define o título do gráfico principal como 'Distribuição do valor das gorjetas' usando `plt.title()`.

plt.suptitle('')

- O `plt.suptitle('')` é usado para remover o título superior gerado automaticamente pelo Pandas, deixando apenas o título principal.

O resultado é um gráfico de caixa que mostra a distribuição do valor das gorjetas para cada dia da semana. Esse tipo de gráfico é útil para visualizar a variação nos valores das gorjetas, incluindo a mediana, quartis, possíveis outliers e a média, para cada categoria de dia.

```
gorjetas.boxplot(by=['day', 'sex'], column=['tip'],
                 color='blue', figsize=(12,6), showmeans=True)
plt.title('Distribuição do valor das gorjetas')
plt.suptitle('')
```

gorjetas.boxplot(by='time', column=['tip'], color='blue', showmeans=True)

- Neste trecho, você está criando um gráfico de caixa (boxplot) condicional usando o método `boxplot()` do Pandas:

  - `by='time'`: Você está agrupando os dados pelo valor da coluna 'time', o que significa que cada caixa do boxplot será criada para um período de tempo específico.
  - `column=['tip']`: Especifica que deseja criar o boxplot para a coluna 'tip', que representa o valor das gorjetas.
  - `color='blue'`: Define a cor das caixas do boxplot como azul.
  - `showmeans=True`: Mostra uma linha representando a média dos valores de gorjeta ('tip') para cada período de tempo no gráfico.

plt.title('Distribuição do valor das gorjetas')

- Este trecho define o título do gráfico principal como 'Distribuição do valor das gorjetas' usando `plt.title()`.

plt.suptitle('')

- O `plt.suptitle('')` é usado para remover o título superior gerado automaticamente pelo Pandas, deixando apenas o título principal.

O resultado é um gráfico de caixa que mostra a distribuição do valor das gorjetas para cada período de tempo (provavelmente, almoço e jantar). Esse tipo de gráfico é útil para visualizar a variação nos valores das gorjetas, incluindo a mediana, quartis, possíveis outliers e a média, para cada categoria de período de tempo.

```
gorjetas.boxplot(by=['day', 'sex'], column=['tip'],
                 color='blue', figsize=(12,6), showmeans=True)
plt.title('Distribuição do valor das gorjetas')
plt.suptitle('')
```

gorjetas.boxplot(by=['day', 'sex'], column=['tip'],
color='blue', figsize=(12,6), showmeans=True)

Neste trecho, você está criando um gráfico de caixa (boxplot) condicional com base nas colunas 'day' (dia da semana) e 'sex' (sexo) para visualizar a distribuição do valor das gorjetas ('tip'). Aqui estão os detalhes:

- `by=['day', 'sex']`: Você está agrupando os dados por duas colunas, 'day' e 'sex', o que significa que o boxplot será criado para cada combinação de dia da semana e sexo.
- `column=['tip']`: Especifica que deseja criar o boxplot para a coluna 'tip', que representa o valor das gorjetas.
- `color='blue'`: Define a cor das caixas do boxplot como azul.
- `figsize=(12,6)`: Define o tamanho da figura do gráfico como (12, 6), o que controla as dimensões do gráfico na tela.
- `showmeans=True`: Mostra uma linha representando a média dos valores de gorjeta ('tip') para cada combinação de dia da semana e sexo no gráfico.

plt.title('Distribuição do valor das gorjetas')

- Este trecho define o título do gráfico principal como 'Distribuição do valor das gorjetas' usando `plt.title()`.

plt.suptitle('')

- O `plt.suptitle('')` é usado para remover o título superior gerado automaticamente pelo Pandas, deixando apenas o título principal.

O resultado é um gráfico de caixa que mostra a distribuição do valor das gorjetas para cada combinação de dia da semana e sexo. Isso permite visualizar a variação nos valores das gorjetas, incluindo a mediana, quartis, possíveis outliers e a média, para diferentes grupos de clientes com base no dia da semana e sexo. O gráfico tem um tamanho específico de 12x6 para facilitar a visualização.

Calculando escores-z:

```
import scipy.stats as stats
# Criando o conjunto de dados
dados = pd.DataFrame({'Nome': ['Nik', 'Kate', 'Joe', 'Mitch', 'Alana'],
                      'Idade': [32, 30, 67, 34, 20],
                      'Renda': [80000, 90000, 45000, 23000, 12000]})
print('Tabela original:')
display(dados)
# Calculando os escores-z das colunas numéricas
tabela_z = dados.select_dtypes(include='number').apply(stats.zscore)
print('Tabela com colunas substituídas pelos escores-z:')
display(tabela_z)
# Agregando novas colunas com os escores-z
dados['Escore-z Idade'] = stats.zscore(dados['Idade'])
dados['Escore-z Renda'] = stats.zscore(dados['Renda'])
print('Tabela orignal + escores-z:')
display(dados)
```

# Criando o conjunto de dados
dados = pd.DataFrame({'Nome': ['Nik', 'Kate', 'Joe', 'Mitch', 'Alana'],
'Idade': [32, 30, 67, 34, 20],
'Renda': [80000, 90000, 45000, 23000, 12000]})

- Neste trecho, você está criando um DataFrame chamado `dados` contendo informações sobre cinco pessoas, incluindo seus nomes, idades e rendas.

print('Tabela original:')
display(dados)

- Aqui, você está exibindo o DataFrame `dados` para mostrar a tabela original na saída.

# Calculando os escores-z das colunas numéricas
tabela_z = dados.select_dtypes(include='number').apply(stats.zscore)

- Neste trecho, você está calculando os escores-z (z-scores) para as colunas numéricas do DataFrame

`dados`. A função `select_dtypes(include='number')` é usada para selecionar apenas as colunas numéricas, e `apply(stats.zscore)` é aplicada a essas colunas para calcular os escores-z. Os escores-z são uma medida de quantos desvios padrão um valor está longe da média.

```
print('Tabela com colunas substituídas pelos escores-z:')
display(tabela_z)
```

- Aqui, você está exibindo o DataFrame `tabela_z`, que contém as colunas originais substituídas pelos escores-z. Isso permite ver como os valores originais se comparam à média e ao desvio padrão.

```
# Agregando novas colunas com os escores-z
dados['Escore-z Idade'] = stats.zscore(dados['Idade'])
dados['Escore-z Renda'] = stats.zscore(dados['Renda'])
```

- Neste trecho, você está adicionando duas novas colunas ao DataFrame `dados`. A primeira coluna, 'Escore-z Idade', contém os escores-z calculados para a coluna 'Idade'. A segunda coluna, 'Escore-z Renda', contém os escores-z calculados para a coluna 'Renda'. Isso permite que você tenha os escores-z disponíveis no DataFrame original.

```
print('Tabela original + escores-z:')
display(dados)
```

- Finalmente, você exibe o DataFrame `dados` com as colunas originais e as colunas de escores-z adicionadas. Isso mostra a tabela original com as informações de escores-z das colunas 'Idade' e 'Renda'.

O resultado final é um DataFrame que inclui tanto os dados originais como os escores-z calculados para algumas das colunas, permitindo uma análise estatística das variáveis em relação à média e ao desvio padrão.