CÓDIGOS

terça-feira, 19 de setembro de 2023 21:59

Distribuição Discreta de Probabilidade:

```
# Conjunto de dados
tabela = pd.DataFrame({'Vendas/dia': list(range(8)),
                       'Num_dias': [16, 19, 15, 21, 9, 10, 8, 2]})
display(tabela)
```

Nesse trecho de código, você está criando um DataFrame do pandas chamado "tabela" com duas colunas: "Vendas/dia" e "Num_dias". Vou explicar cada parte do código:

- tabela = pd.DataFrame(...)`: Isso cria um novo DataFrame usando o construtor `pd.DataFrame()`. Um DataFrame é uma estrutura de dados tabular semelhante a uma planilha, onde você pode armazenar e manipular dados de maneira organizada.

-`{'Vendas/dia': list(range(8)), 'Num_dias': [16, 19, 15, 21, 9, 10, 8, 2]}`: Este é o dicionário de dados que você passa para o construtor do DataFrame. Ele possui duas chaves: "Vendas/dia" e "Num_dias". O valor associado à chave "Vendas/dia" é uma lista gerada pela função `list(range(8))`, que cria uma lista com os números de 0 a 7. Isso cria a primeira coluna chamada "Vendas/dia" com os valores de 0 a 7. O valor associado à chave "Num_dias" é a lista `[16, 19, 15, 21, 9, 10, 8, 2]`, que cria a segunda coluna chamada "Num_dias" com esses valores.

- display(tabela)`: Este comando exibe o DataFrame "tabela" na saída, permitindo que você veja os dados tabulares de forma clara e organizada.

No geral, este trecho de código cria e exibe um DataFrame com informações sobre vendas por dia e o número de dias correspondentes. Isso é útil para análises posteriores ou para representar esses dados de maneira tabular.

```
# Cálculo das probabilidades
total_vendas = tabela['Num_dias'].sum()
tabela['P(x)'] = tabela['Num_dias']/total_vendas
display(tabela)
```

Nesse trecho de código, você está calculando e adicionando uma nova coluna chamada "P(x)" ao DataFrame "tabela" que representa as probabilidades de cada valor da coluna "Vendas/dia". Vou explicar cada parte do código:

-`total_vendas = tabela['Num_dias'].sum()`: Isso calcula a soma de todos os valores na coluna "Num_dias" do DataFrame "tabela". Essa soma representa o total de dias.

- tabela['P(x)'] = tabela['Num_dias']/total_vendas`: Aqui, você está calculando as probabilidades individuais (P(x)) para cada valor na coluna "Vendas/dia". Para fazer isso, você está dividindo o número de dias correspondentes a cada valor na coluna "Num_dias" pelo total de dias calculado anteriormente.

- display(tabela)`: Este comando exibe o DataFrame atualizado na saída, agora com a coluna "P(x)" adicionada, mostrando as probabilidades associadas a cada valor na coluna "Vendas/dia".

Portanto, após a execução deste código, o DataFrame "tabela" terá uma nova coluna "P(x)" que

representa as probabilidades de cada valor na coluna "Vendas/dia" com base no número de dias correspondentes a esses valores em relação ao total de dias. Isso é útil para análises estatísticas ou de probabilidade associadas a esses dados.

```
# Cálculo da probabilidade acumulada
tabela['P_ac(x)'] = tabela['P(x)'].cumsum()
display(tabela)
```

Nesse trecho de código, você está calculando e adicionando uma nova coluna chamada "P_ac(x)" ao DataFrame "tabela" que representa a probabilidade acumulada. Vou explicar cada parte do código:

- tabela['P_ac(x)']`: Isso cria uma nova coluna chamada "P_ac(x)" no DataFrame "tabela" onde você armazenará as probabilidades acumuladas.

-`tabela['P(x)'].cumsum()`: Aqui, você está calculando as somas acumuladas das probabilidades da coluna "P(x)". A função `.cumsum()` é usada para calcular a soma cumulativa dos valores em "P(x)".

- display(tabela)`: Este comando exibe o DataFrame atualizado na saída, agora com a coluna "P_ac(x)" adicionada, mostrando as probabilidades acumuladas.

Portanto, após a execução deste código, o DataFrame "tabela" terá uma nova coluna "P_ac(x)" que representa as probabilidades acumuladas. Essas probabilidades acumuladas indicam a probabilidade de obter um valor igual ou menor ao valor correspondente na coluna "Vendas/dia". Isso é útil para análises estatísticas e visualizações, como gráficos de probabilidade acumulada.

Representação gráfica da distribuição discreta de probabilidades:

```
# Gráfico de segmentos verticais (caules)
plt.stem(tabela['Vendas/dia'], tabela['P(x)'], linefmt='b-',
         label='P(efetuar x vendas ao dia)')
plt.xlabel('Vendas por dia')
plt.ylabel('Probabilidades')
plt.title('Distribuição discreta de probabilidades')
plt.grid()
plt.legend()
plt.show()
```

Neste trecho de código, você está criando um gráfico de segmentos verticais (um gráfico de caules) para visualizar a distribuição discreta de probabilidades. Vou explicar cada parte do código:

- plt.stem(tabela['Vendas/dia'], tabela['P(x)'], linefmt='b-', label='P(efetuar x vendas ao dia)')`: Aqui, você está usando a função `stem` do matplotlib para criar o gráfico de segmentos verticais. Os argumentos são:
  - tabela['Vendas/dia']`: Isso representa o eixo x do gráfico e contém os valores de "Vendas por dia" do DataFrame "tabela".
  - tabela['P(x)']`: Isso representa o eixo y do gráfico e contém as probabilidades do DataFrame "tabela".
  - linefmt='b-'`: Define o formato da linha, onde 'b-' significa uma linha azul sólida. Isso controla a aparência dos segmentos verticais.
  - label='P(efetuar x vendas ao dia)'`: É um rótulo para a legenda do gráfico.

- plt.xlabel('Vendas por dia')`: Define o rótulo do eixo x como "Vendas por dia".

-`plt.ylabel('Probabilidades')`: Define o rótulo do eixo y como "Probabilidades".

- plt.title('Distribuição discreta de probabilidades')`: Define o título do gráfico como "Distribuição discreta de probabilidades".

- plt.grid()`: Adiciona linhas de grade ao gráfico para facilitar a leitura.

- plt.legend()`: Mostra a legenda no gráfico com base no rótulo definido anteriormente.

- plt.show()`: Exibe o gráfico na saída.

Portanto, este código cria um gráfico de caules que representa a distribuição discreta de probabilidades das vendas por dia, ajudando a visualizar como as probabilidades estão distribuídas para diferentes valores de vendas por dia

```
# Gráfico de barras verticais
plt.bar(tabela['Vendas/dia'], tabela['P(x)'],
        label='P(efetuar x vendas ao dia)')
plt.xlabel('Vendas por dia')
plt.ylabel('Probabilidades')
plt.title('Distribuição discreta de probabilidades')
plt.grid(False)
plt.legend()
plt.show()
```

Neste trecho de código, você está criando um gráfico de barras verticais para visualizar a distribuição discreta de probabilidades. Vou explicar cada parte do código:

- plt.bar(tabela['Vendas/dia'], tabela['P(x)'], label='P(efetuar x vendas ao dia)')`: Aqui, você está usando a função `bar` do matplotlib para criar o gráfico de barras verticais. Os argumentos são:
  - tabela['Vendas/dia']`: Isso representa o eixo x do gráfico e contém os valores de "Vendas por dia" do DataFrame "tabela".
  - tabela['P(x)']`: Isso representa o eixo y do gráfico e contém as probabilidades do DataFrame "tabela".
  - label='P(efetuar x vendas ao dia)'`: É um rótulo para a legenda do gráfico.

- plt.xlabel('Vendas por dia')`: Define o rótulo do eixo x como "Vendas por dia".

- plt.ylabel('Probabilidades')`: Define o rótulo do eixo y como "Probabilidades".

- plt.title('Distribuição discreta de probabilidades')`: Define o título do gráfico como "Distribuição discreta de probabilidades".

- plt.grid(False)`: Remove as linhas de grade do gráfico.

- plt.legend()`: Mostra a legenda no gráfico com base no rótulo definido anteriormente.

- plt.show()`: Exibe o gráfico na saída.

Portanto, este código cria um gráfico de barras verticais que representa a distribuição discreta de probabilidades das vendas por dia, permitindo visualizar facilmente as probabilidades associadas a cada valor de vendas.

```
# Representação gráfica da probabilidade discreta acumulada
plt.plot(tabela['Vendas/dia'], tabela['P_ac(x)'], 'g-o',
         label='P(efetuar x vendas ao dia) [acumulada)]')
plt.xlabel('Vendas por dia')
plt.ylabel('Probabilidades')
plt.title('Probabilidade discreta acumulada')
plt.grid()
plt.legend()
plt.show()
```

Neste trecho de código, você está criando um gráfico de linha para representar a probabilidade discreta acumulada das vendas por dia. Vou explicar cada parte do código:

- plt.plot(tabela['Vendas/dia'], tabela['P_ac(x)'], 'g-o', label='P(efetuar x vendas ao dia) [acumulada)]')`: Aqui, você está usando a função `plot` do matplotlib para criar o gráfico de linha. Os argumentos são:
  - tabela['Vendas/dia']`: Isso representa o eixo x do gráfico e contém os valores de "Vendas por dia" do DataFrame "tabela".
  - tabela['P_ac(x)']`: Isso representa o eixo y do gráfico e contém as probabilidades acumuladas do DataFrame "tabela".
  - `g-o'`: Isso define o estilo da linha no gráfico. "g" significa verde (green), e "o" significa que os pontos de dados serão marcados com círculos.
  - label='P(efetuar x vendas ao dia) [acumulada)]'`: É um rótulo para a legenda do gráfico.

- plt.xlabel('Vendas por dia')`: Define o rótulo do eixo x como "Vendas por dia".

- plt.ylabel('Probabilidades')`: Define o rótulo do eixo y como "Probabilidades".

- plt.title('Probabilidade discreta acumulada')`: Define o título do gráfico como "Probabilidade discreta acumulada".

- plt.grid()`: Adiciona as linhas de grade ao gráfico.

- plt.legend()`: Mostra a legenda no gráfico com base no rótulo definido anteriormente.

- plt.show()`: Exibe o gráfico na saída.

Portanto, este código cria um gráfico de linha que representa a probabilidade discreta acumulada das vendas por dia, permitindo visualizar como as probabilidades acumulam à medida que o número de vendas aumenta.

Média de uma distribuição discreta de probabilidades:

```
# Inserção da coluna Vendas*Probabilidades
tabela['xP(x)'] = tabela['Vendas/dia']*tabela['P(x)']
display(tabela)
```

Neste trecho de código, você está calculando e inserindo uma nova coluna chamada 'xP(x)' no DataFrame "tabela". Vou explicar cada parte do código:

- tabela['xP(x)']`: Isso cria uma nova coluna chamada 'xP(x)' no DataFrame "tabela" para armazenar os valores resultantes da multiplicação entre as colunas 'Vendas/dia' e 'P(x)'.

- tabela['Vendas/dia']`: Isso acessa a coluna 'Vendas/dia' no DataFrame "tabela", que contém os valores das vendas por dia.

- tabela['P(x)']`: Isso acessa a coluna 'P(x)' no DataFrame "tabela", que contém as probabilidades

correspondentes às vendas por dia.

- tabela['Vendas/dia']*tabela['P(x)']`: Essa operação calcula o produto entre os valores das colunas 'Vendas/dia' e 'P(x)' para cada linha do DataFrame. Ou seja, para cada valor de vendas por dia, é multiplicado pela probabilidade correspondente.

- display(tabela)`: Isso exibe o DataFrame "tabela" com a nova coluna 'xP(x)' adicionada, mostrando os resultados da multiplicação.

Essa nova coluna 'xP(x)' representa a contribuição de cada valor de vendas por dia para o valor esperado ou média ponderada das vendas, uma medida importante em estatísticas e probabilidade. É comumente usada para calcular valores esperados em distribuições discretas.

```
# Cálculo da média da distribuição
media = tabela['xP(x)'].sum()
print(f'A média da distribuição é de {media:.1f} vendas/dia.')
```

Neste trecho de código, você está calculando a média da distribuição discreta. Vou explicar cada parte do código:

- tabela['xP(x)']`: Isso acessa a coluna 'xP(x)' no DataFrame "tabela", que representa o produto entre as vendas por dia ('Vendas/dia') e as probabilidades ('P(x)') correspondentes a cada valor de vendas.

- .sum()`: Este método é aplicado à coluna 'xP(x)' e calcula a soma de todos os valores contidos nessa coluna. Essa soma representa a média ponderada das vendas por dia.

- media = tabela['xP(x)'].sum()`: Aqui, o resultado da soma é atribuído à variável 'media', que conterá o valor da média ponderada das vendas por dia.

- print(f'A média da distribuição é de {media:.1f} vendas/dia.')`: Esta linha imprime a média calculada no formato de uma string formatada. O valor da média é exibido com uma casa decimal.

Portanto, o código calcula a média ponderada das vendas por dia, levando em consideração as probabilidades associadas a cada valor de vendas, e exibe essa média como resultado. A média é uma medida central que representa o valor esperado das vendas por dia nessa distribuição discreta.

Variância e desvio padrão de uma distribuição discreta de probabilidades:

```
# Inserção da coluna Vendas^2*Probabilidades
tabela['x^2P(x)'] = tabela['Vendas/dia']**2 * tabela['P(x)']
display(tabela)
```

Neste trecho de código, você está calculando a coluna 'x^2P(x)' no DataFrame "tabela", que envolve o quadrado das vendas por dia ('Vendas/dia') multiplicado pelas probabilidades ('P(x)') correspondentes a cada valor de vendas. Vou explicar cada parte do código:

- tabela['Vendas/dia']**2`: Isso calcula o quadrado das vendas por dia para cada linha no DataFrame. `**2` é a operação de exponenciação ao quadrado.

- tabela['x^2P(x)']`: Isso acessa a coluna 'x^2P(x)' no DataFrame "tabela", que será preenchida com o resultado do cálculo acima.

- tabela['Vendas/dia']**2 * tabela['P(x)']`: Isso multiplica o quadrado das vendas por dia ('Vendas/dia' elevado ao quadrado) pelas probabilidades ('P(x)') correspondentes a cada valor de vendas.

- display(tabela)`: Isso exibe o DataFrame "tabela" após a adição da coluna 'x^2P(x)'.

Essa coluna 'x^2P(x)' representa o resultado da multiplicação das vendas por dia ao quadrado pelas probabilidades correspondentes, o que é útil para o cálculo de momentos estatísticos, como a variância. Essa é uma etapa comum na análise estatística de distribuições discretas quando se deseja calcular momentos de ordem superior.

```
# Cálculo da variância da distribuição
variancia = tabela['x^2P(x)'].sum() - media**2
print(f'A variância da distribuição é de {variancia:.1f} (vendas/dia)^2.')
```

Neste trecho de código, você está calculando a variância da distribuição discreta. Vou explicar cada parte do código:

- variancia: É a variável que armazenará o valor da variância.

- tabela['x^2P(x)'].sum()`: Aqui, você está somando todos os valores da coluna 'x^2P(x)' no DataFrame "tabela". Isso corresponde ao somatório de x^2P(x) para cada valor de vendas por dia.

- media**2: Isso calcula o quadrado da média da distribuição, que já foi calculada anteriormente.

- variancia = tabela['x^2P(x)'].sum() - media**2`: Finalmente, você calcula a variância subtraindo o quadrado da média do somatório de x^2P(x). Essa é a fórmula básica para calcular a variância de uma distribuição discreta.

- print(f'A variância da distribuição é de {variancia:.1f} (vendas/dia)^2.')`: Isso imprime a variância calculada com uma formatação adequada para exibir uma casa decimal.

A variância é uma medida de dispersão que indica o quão espalhados estão os valores da distribuição em relação à média. Quanto maior a variância, mais dispersos os valores; quanto menor, mais próximos eles estão da média.

Calculando algumas probabilidades:

```
# Cálculo do desvio padrão da distribuição
desvio = variancia**(1/2)
print(f'O desvio padrão da distribuição é de {desvio:.1f} vendas/dia.')
```

Neste trecho de código, você está calculando o desvio padrão da distribuição discreta. Vou explicar cada parte do código:

- desvio`: É a variável que armazenará o valor do desvio padrão.

- variancia**(1/2)`: Aqui, você calcula a raiz quadrada da variância previamente calculada. O desvio padrão é a raiz quadrada da variância, e essa é a fórmula básica para obtê-lo.

- print(f'O desvio padrão da distribuição é de {desvio:.1f} vendas/dia.')`: Isso imprime o desvio padrão calculado com uma formatação adequada para exibir uma casa decimal.

O desvio padrão é uma medida de dispersão que indica o quanto os valores de uma distribuição estão, em média, afastados da média. Quanto maior o desvio padrão, maior é a dispersão dos valores em relação à média, e vice-versa. Portanto, o desvio padrão é uma medida importante para entender a variabilidade em uma distribuição de dados.

```
# P(4 vendas em um único dia)
tab_filtrada = tabela['P(x)'].loc[tabela['Vendas/dia'] == 4]
print(tab_filtrada)
prob = tab_filtrada.values[0]
print(f'P(4 vendas em um único dia) = {prob*100:.1f}%')
```

Neste trecho de código, você está calculando a probabilidade de ocorrerem exatamente 4 vendas em um único dia. Vou explicar cada parte do código:

- tab_filtrada`: Essa variável armazena a coluna de probabilidades `P(x)` da tabela `tabela` (que representa a distribuição discreta de vendas) apenas para os casos em que a coluna `Vendas/dia` é igual a 4. Isso significa que você está filtrando a probabilidade correspondente a exatamente 4 vendas em um único dia.

- print(tab_filtrada)`: Isso exibe a probabilidade filtrada, que deve conter apenas um valor (porque estamos procurando um número específico de vendas, neste caso, 4).

- prob`: Essa variável obtém o valor da probabilidade filtrada. Para fazer isso, você acessa os valores da coluna `tab_filtrada` usando `tab_filtrada.values` e pega o primeiro valor (o único valor na coluna, neste caso).

- print(f'P(4 vendas em um único dia) = {prob*100:.1f}%')`: Isso imprime a probabilidade de ocorrerem exatamente 4 vendas em um único dia em formato percentual com uma casa decimal.

Portanto, este trecho de código está extraindo e calculando a probabilidade de um evento específico (4 vendas em um único dia) a partir da distribuição discreta de vendas representada na tabela.

```
# P(menos de 4 vendas em um único dia)
tab_filtrada = tabela['P(x)'].loc[tabela['Vendas/dia'] < 4]
print(tab_filtrada)
prob = tab_filtrada.sum()
print(f'P(menos de 4 vendas em um único dia) = {prob*100:.1f}%')
```

Neste trecho de código, você está calculando a probabilidade de ocorrerem menos de 4 vendas em um único dia. Vou explicar cada parte do código:

- tab_filtrada`: Essa variável armazena a coluna de probabilidades `P(x)` da tabela `tabela` (que representa a distribuição discreta de vendas) apenas para os casos em que a coluna `Vendas/dia` é menor que 4. Isso significa que você está filtrando as probabilidades correspondentes a menos de 4 vendas em um único dia.

- print(tab_filtrada)`: Isso exibe as probabilidades filtradas, que correspondem a menos de 4 vendas em um único dia.

- prob`: Essa variável calcula a soma das probabilidades filtradas. Como você está interessado na probabilidade de ocorrerem "menos de 4 vendas", basta somar todas as probabilidades correspondentes a 0, 1, 2 e 3 vendas (que são as probabilidades filtradas).

- print(f'P(menos de 4 vendas em um único dia) = {prob*100:.1f}%')`: Isso imprime a probabilidade de ocorrerem menos de 4 vendas em um único dia em formato percentual com uma casa decimal.

Portanto, este trecho de código está calculando a probabilidade acumulada de ocorrerem menos de 4 vendas em um único dia com base na distribuição discreta de vendas representada na tabela.

```
# P(mais de 4 vendas em um único dia)
tab_filtrada = tabela['P(x)'].loc[tabela['Vendas/dia'] > 4]
print(tab_filtrada)
prob = tab_filtrada.sum()
print(f'P(mais de 4 vendas em um único dia) = {prob*100:.1f}%')
```

Neste trecho de código, você está calculando a probabilidade de ocorrerem mais de 4 vendas em um único dia. Vou explicar cada parte do código:

- tab_filtrada`: Essa variável armazena a coluna de probabilidades `P(x)` da tabela `tabela` (que representa a distribuição discreta de vendas) apenas para os casos em que a coluna `Vendas/dia` é maior que 4. Isso significa que você está filtrando as probabilidades correspondentes a mais de 4 vendas em um único dia.

- print(tab_filtrada)`: Isso exibe as probabilidades filtradas, que correspondem a mais de 4 vendas em um único dia.

- prob: Essa variável calcula a soma das probabilidades filtradas. Como você está interessado na probabilidade de ocorrerem "mais de 4 vendas", basta somar todas as probabilidades correspondentes a 5, 6 e 7 vendas (que são as probabilidades filtradas).

- print(f'P(mais de 4 vendas em um único dia) = {prob*100:.1f}%')`: Isso imprime a probabilidade de ocorrerem mais de 4 vendas em um único dia em formato percentual com uma casa decimal.

Portanto, este trecho de código está calculando a probabilidade acumulada de ocorrerem mais de 4 vendas em um único dia com base na distribuição discreta de vendas representada na tabela.