**Métodos Quantitativos I**

**Camila Guedes de Farias**

2ª Lista de Exercícios, 7/4/2025

*Descrição:*

*Exercícios dos capítulos 4, 5 e 6 do livro Estatística Aplicada a Administração e Economia.*

*Cap. 4: 10, 20, 28, 33, 38, 42*

*Cap. 5: 3, 8, 18, 27, 34*

*Cap. 6: 2, 10, 16, 22, 24, 40*

***Cap. 4:10***

1. *Probabilidade de um voo da Delta Air Lines chegar no horário: 0.865*
2. *Probabilidade de escolher uma companhia com menos de dois relatórios de bagagem extraviada: 0.3*
3. *Probabilidade de escolher uma companhia com mais de uma reclamação de cliente: 0.5*
4. *Probabilidade de um voo da AirTran Airways não chegar no horário: 0.129*

***Cap. 4:20***

1. *Probabilidade para cada categoria de idade:*

*16 a 20: 0.2023 ou 20.23%*

*21 a 24: 0.4947 ou 49.47%*

*25 a 27: 0.2585 ou 25.85%*

*28 ou mais: 0.0445 ou 4.45%*

1. *Probabilidade de ser financeiramente independente antes dos 25 anos: 0.6970 ou 69.70%*
2. *Probabilidade de ser financeiramente independente após os 24 anos: 0.3030 ou 30.30%*
3. *Análise sobre o realismo das expectativas:*

*A idade média aproximada em que os adolescentes acreditam que serão independentes: 22.83 anos. A maioria dos adolescentes (aproximadamente 49.47%) espera ser financeiramente independente entre 21 e 24 anos. Isso pode ser considerado otimista, dado que muitos jovens nessa faixa etária ainda estão na faculdade ou iniciando suas carreiras, com salários iniciais mais baixos.*

***Cap. 4:28***

1. *Probabilidade de um assinante ter alugado um carro por razões comerciais OU pessoais: 0.6980 ou 69.80%*
2. *Probabilidade de um assinante NÃO ter alugado um carro: 0.3020 ou 30.20%*

***Cap. 4:33***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Status Matrícula | Administração | Engenharia | Outros | Totais |
| Período Integral | 0,269732 | 0,150958 | 0,192337 | 0,613027 |
| Meio Período | 0,114943 | 0,123372 | 0,148659 | 0,386973 |
| Totais | 0,384674 | 0,274330 | 0,340996 | 1,0 |

1. *Probabilidades marginais por setor de graduação:*

*Probabilidade de ser graduado em Administração: 0.3847 ou 38.47%*

*Probabilidade de ser graduado em Engenharia: 0.2743 ou 27.43%*

*Probabilidade de ser graduado em Outros cursos: 0.3410 ou 34.10%*

*O setor de Administração tem o maior potencial de estudantes de MBA.*

1. *Probabilidade de ser graduado em Engenharia e frequentar MBA em tempo integral: 0.1510 ou 15.10%*
2. *Probabilidade de frequentar meio período sendo da Administração: 0.2988 ou 29.88%*
3. *Verificação de independência dos eventos:*

*P(Tempo Integral) = 0.6130*

*P(Administração) = 0.3847*

*P(Tempo Integral e Administração) = 0.2697*

*P(Tempo Integral) \* P(Administração) = 0.2358*

*Os eventos NÃO são independentes, pois P(A e B) ≠ P(A) \* P(B)*

*Há uma dependência positiva: ser da Administração aumenta a chance de frequentar em tempo integral*

***Cap. 4:38***

1. Probabilidade de um aluno ter recebido um diploma universitário: 0.4200 ou 42.00%
2. Probabilidade de um aluno não ter recebido um diploma universitário: 0.5800 ou 58.00%
3. Probabilidade de ter empréstimo inadimplente dado que recebeu diploma: 0.3810 ou 38.10%
4. Probabilidade de ter empréstimo inadimplente dado que não recebeu diploma: 0.5862 ou 58.62%
5. Impacto de abandonar a faculdade sem diploma para alunos com empréstimo:

Probabilidade de inadimplência com diploma: 0.3810 ou 38.10%

Probabilidade de inadimplência sem diploma: 0.5862 ou 58.62%

Diferença absoluta: 0.2053 ou 20.53 pontos percentuais

Aumento relativo: 53.88%

Conclusão: Não obter um diploma universitário aumenta a probabilidade de inadimplência em 20.53 pontos percentuais, o que representa um aumento de 53.88% na chance de inadimplência.

***Cap. 4:42***

1. Probabilidade a posteriori de inadimplência dado que deixou de efetuar pagamento: P(inadimplente | deixou pagamento) = 0.2083 ou 20.83%
2. Probabilidade a posteriori de inadimplência: 0.2083 ou 20.83%

Limite estabelecido pelo banco para cancelamento: 0.20 ou 20%

Sim, o banco deveria cancelar o cartão visto que a probabilidade a posteriori de inadimplência (20.83%) é maior que o limite de 20%.

***Cap. 5: 3***

1. Resultados experimentais possíveis:

0 ofertas: nenhum estudante recebe oferta

1 oferta: apenas um estudante recebe oferta

2 ofertas: dois estudantes recebem ofertas

1. ofertas: todos os três estudantes recebem ofertas
2. Definição da variável aleatória:

X = número de ofertas recebidas pelos três estudantes

X pode assumir os valores: 0, 1, 2 ou 3

A variável aleatória é discreta, pois assume um número finito de valores

1. Valores da variável aleatória para cada resultado experimental:

Se nenhum estudante recebe oferta: X = 0

Se apenas um estudante recebe oferta: X = 1

Se dois estudantes recebem ofertas: X = 2

Se todos os três estudantes recebem ofertas: X = 3

Demonstração da simulação:

Simulação 1: 2 ofertas recebidas

Simulação 2: 2 ofertas recebidas

Simulação 3: 1 oferta recebida

Simulação 4: 0 ofertas recebida

Simulação 5: 0 oferta recebida

***Cap. 5: 8***

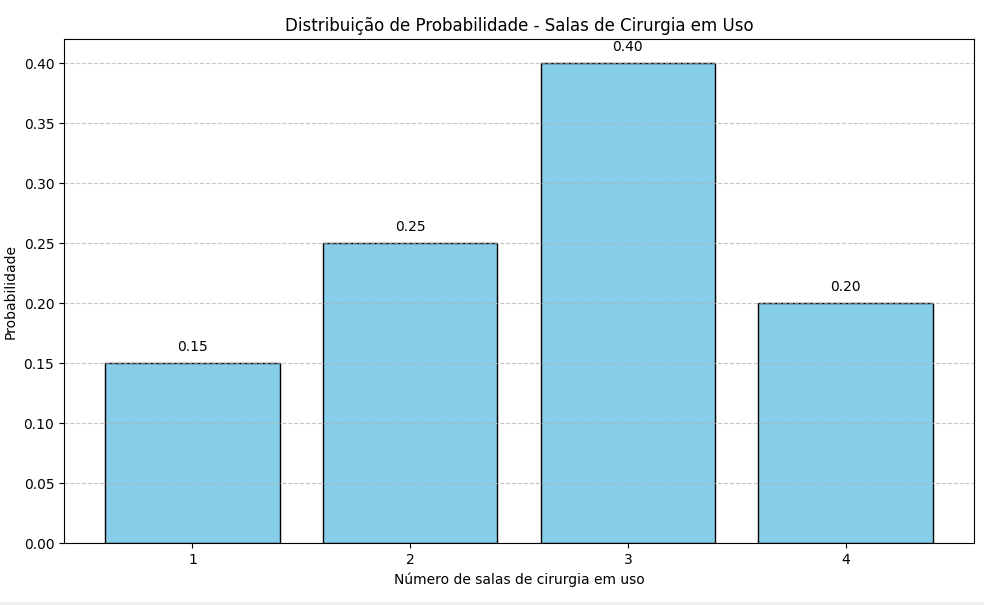
1. Distribuição de probabilidade empírica:

P(X = 1) = 0.15 ou 15%

P(X = 2) = 0.25 ou 25%

P(X = 3) = 0.40 ou 40%

P(X = 4) = 0.20 ou 20%



|  |  |  |  |
| --- | --- | --- | --- |
| Verificação das Condições para uma Distribuição Discreta Válida | | | |
|  | | | |
| Condição | Nº | Descrição | Verificação |
| 1 | 1 | Todas as probabilidades são não-negativas | True |
| 2 | 2 | A soma de todas as probabilidades é igual a 1 | True |
| 3 | 3 | Soma das probabilidades | 1.0000000000 |
| Resultado Final | A distribuição satisfaz as condições necessárias para ser uma distribuição discreta de probabilidade válida. | | |
|  | | | |
| \*\*Estatísticas Adicionais\*\* | | | |
| Estatística | | Valor | |
| Valor Esperado (média) | | 2,65 salas | |
| Variância | | 0,93 | |
| Desvio Padrão | | 0,96 | |

***Cap. 5: 18***

1. P(x = 0) = 0.2188 ou 21.88%

P(x = 1) = 0.5484 ou 54.84%

P(x = 2) = 0.1241 ou 12.41%

P(x = 3) = 0.0489 ou 4.89%

P(x = 4 ou mais) = 0.0598 ou 5.98%

1. O valor esperado (média) é calculado como:

E(x) = 0×P(x=0) + 1×P(x=1) + 2×P(x=2) + 3×P(x=3) + 4×P(x=4 ou mais)

A variância é calculada como:

Var(x) = (0-E(x))²×P(x=0) + (1-E(x))²×P(x=1) + ... + (4-E(x))²×P(x=4 ou mais)

Para proprietários:

Valor esperado de x ≈ 0,8523

Variância de x ≈ 0,9669

1. Variável aleatória y = número de vezes que casas ocupadas por locatários tiveram interrupção no fornecimento de água com duração de 6 horas ou mais nos últimos 3 meses.

Distribuição de probabilidade para y:

P(y = 0) = 394/1578 ≈ 0,2497 ou 24,97%

P(y = 1) = 760/1578 ≈ 0,4816 ou 48,16%

P(y = 2) = 221/1578 ≈ 0,1400 ou 14,00%

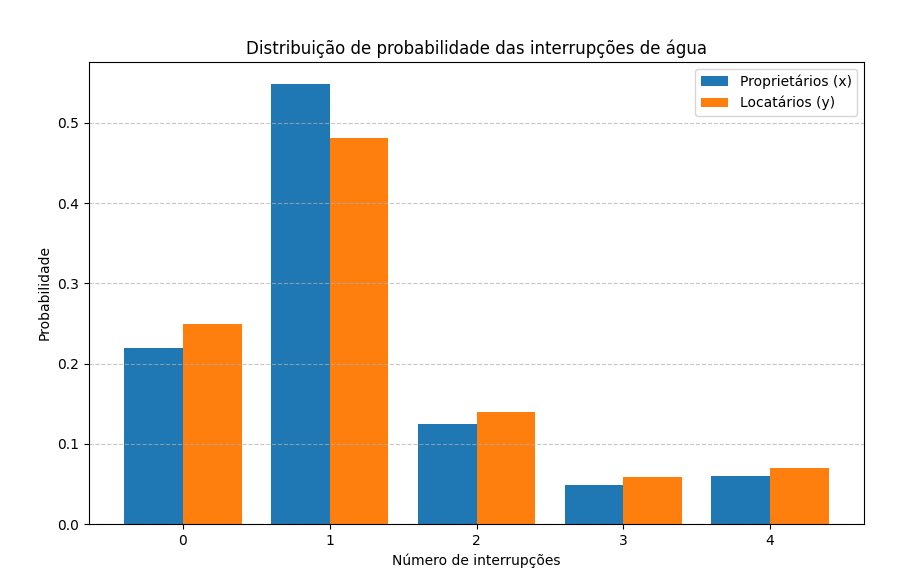
P(y = 3) = 92/1578 ≈ 0,0583 ou 5,83%

P(y = 4 ou mais) = 111/1578 ≈ 0,0703 ou 7,03%

|  |  |
| --- | --- |
| Estatísticas para Casas de Locatários | |
| | **Estatística** | **Valor** | | --- | --- | | Valor esperado (média) de interrupções | 1,2180 | | Variância das interrupções | 1,2085 | | |
|  | |
| Ocupação | **Média de Interrupções** | **Variância** |
| Proprietários | 1,1825 | 1,0435 |
| Locatários | 1,2180 | 1,2085 |

1. Observações da comparação entre proprietários e locatários:

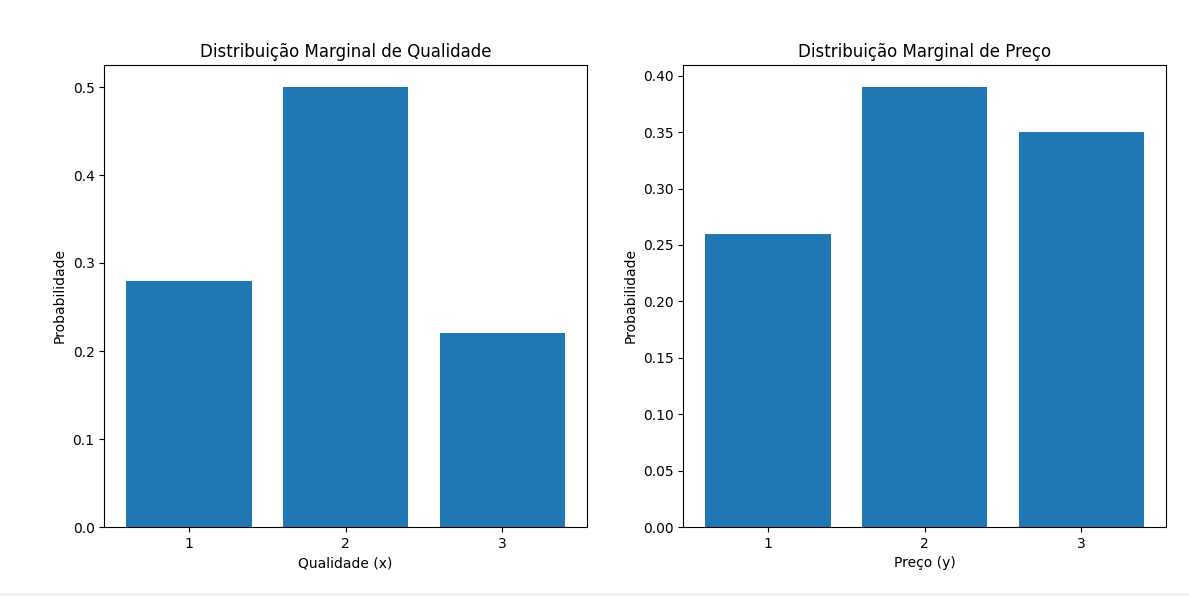
|  |  |  |  |
| --- | --- | --- | --- |
| Critério | Média Proprietários | Média Locatários | Observação |
| Média de interrupções | 1,1825 | 1,2180 | Locatários têm, em média, mais interrupções. |
| Variância | 1,0435 | 1,2085 | Variabilidade maior entre locatários. |
| % sem interrupções | 21,9% | 25,0% | Mais locatários não sofreram interrupções. |
| % com 3 ou mais interrupções | 10,9% | 12,9% | Locatários enfrentam mais casos de interrupções. |



***Cap. 5:27***

1. Distribuição de Probabilidade Bivariada P(X=x, Y=y)

|  |  |  |  |
| --- | --- | --- | --- |
| Qualidade (x) | Preço (y) = 1 | Preço (y) = 2 | Preço (y) = 3 |
| 1 | 0,14 | 0,13 | 0,01 |
| 2 | 0,11 | 0,21 | 0,18 |
| 3 | 0,01 | 0,05 | 0,16 |



1. Distribuição Marginal de Qualidade P(X = x)

|  |  |
| --- | --- |
| ***Qualidade (x)*** | ***Probabilidade*** |
| 1 | 0,28 |
| 2 | 0,50 |
| 3 | 0,22 |
| **Estatísticas:** | |
| Valor Esperado | E[X] = 1,9400 |
| Variância | Var[X] = 0,4964 |

1. Distribuição Marginal de Preço P(Y = y)

|  |  |
| --- | --- |
| **Preço (y)** | **Probabilidade** |
| 1 | 0,26 |
| 2 | 0,39 |
| 3 | 0,35 |
| **Estatísticas:** | |
| Valor Esperado | E[Y] = 2,0900 |
| Variância | Var[Y] = 0,6019 |

1. Covariância

E[XY] = 4,3400

Covariância entre X e Y = 0,2854

Verificação da variância da soma:

O enunciado afirma que Var(X + Y) = 1,6691

Calculamos:

Var(X + Y) = Var(X) + Var(Y) + 2 × Cov(X,Y)

= 0,4964 + 0,6019 + 2 × 0,2854

= 1,6691

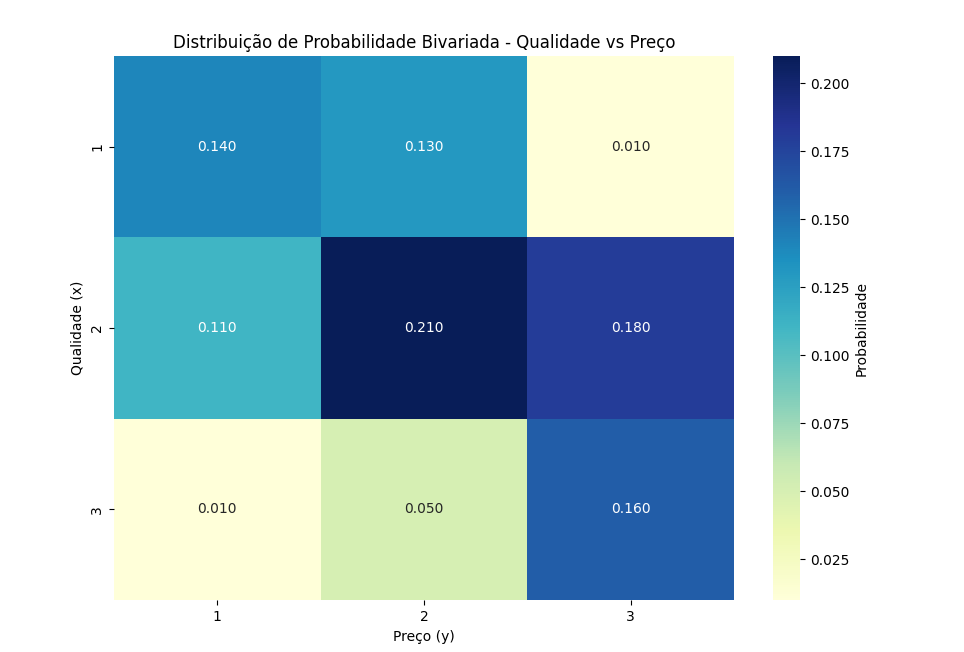
1. Coeficiente de Correlação

|  |  |
| --- | --- |
| Análise | Resultado |
| Correlação entre qualidade e preço | 0,5221 (moderada/positiva) |
| Prob. qualidade alta dado preço baixo | 0,0385 (3,85%) |
| Conclusão | Improvável encontrar restaurante barato com alta qualidade |

Correlação entre X e Y = 0,5221. A correlação entre qualidade e preço é moderada e positiva (0,5221). Isso indica que restaurantes com melhor qualidade tendem a ter preços mais altos.

Probabilidade condicional:

Probabilidade de encontrar qualidade alta (3) dado preço baixo (1) = 0,0385. Isso equivale a 3,85% dos restaurantes de preço baixo. Dessa forma é improvável encontrar um restaurante de baixo custo com alta qualidade nesta cidade.



***Cap. 5: 34***

1. n = 10 número de adolescentes na amostra aleatória

p\_adolescentes\_pandora = 0.39 39% dos adolescentes usam o Pandora Media

É um experimento binomial, sim, pois temos:

Número fixo de ensaios (n=10)

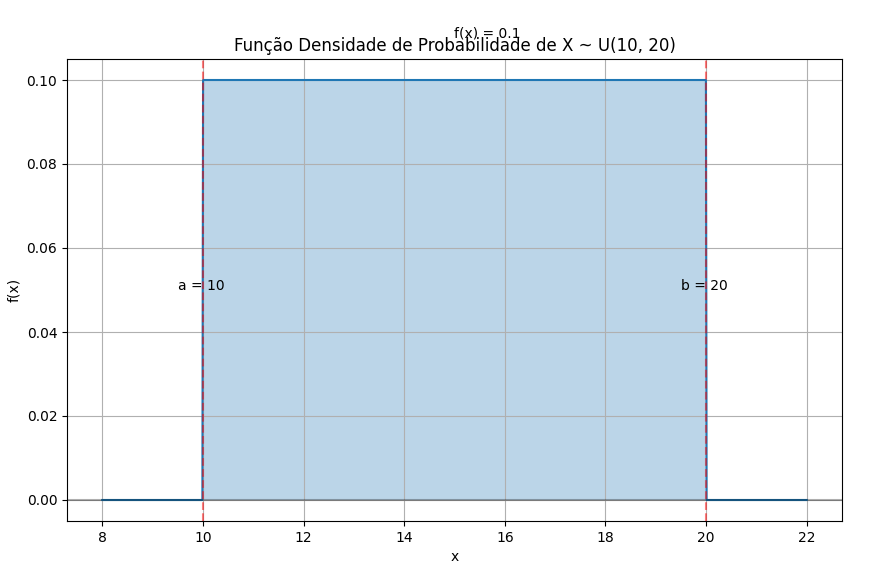
Cada ensaio tem apenas dois resultados possíveis (usa ou não usa o Pandora)

A probabilidade de sucesso é constante (p=0.39)

Os ensaios são independentes (assume-se que os adolescentes são selecionados aleatoriamente)

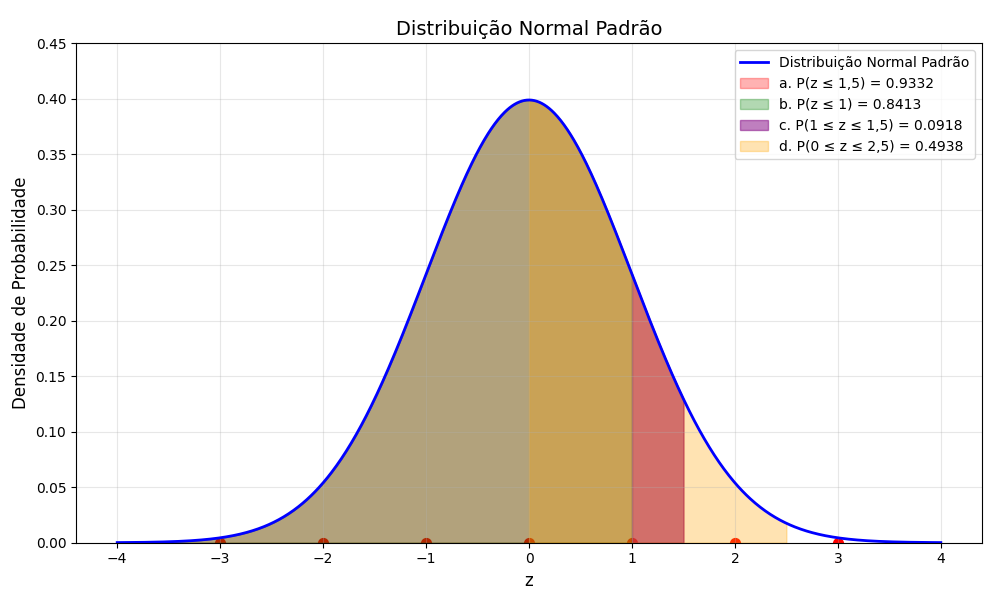
1. Probabilidade de que nenhum dos 10 adolescentes use o serviço: 0.007133
2. Probabilidade de exatamente 4 usarem o Pandora: 0.250298
3. Probabilidade de pelo menos 2 usarem o Pandora: 0.947259

***Cap. 6: 2***



1. P(X < 15) = 0.5
2. P(12 ≤ X ≤ 18) = 0.6
3. E(X) = 15.0
4. Var(X) = 8.333333333333334

***Cap. 6:10***



1. P(z ≤ 1,5) = 0.9332
2. P(z ≤ 1) = 0.8413
3. P(1 ≤ z ≤ 1,5) = 0.0918
4. P(0 ≤ z ≤ 2,5) = 0.4938

***Cap. 6:16***

1. Área à direita de z é 0.01: z = 2.3263
2. Área à direita de z é 0.025: z = 1.9600
3. Área à direita de z é 0.05: z = 1.6449
4. Área à direita de z é 0.1: z = 1.2816

***Cap. 6:22***

1. Probabilidade de assistir entre 5 e 10 horas: 0.6553 ou 65.53%
2. Horas para estar entre os 3% que mais assistem: 13.05 horas
3. Probabilidade de assistir mais de 3 horas: 0.9838 ou 98.38%

***Cap. 6: 24***

1. Probabilidade de gastos inferiores a US$ 400: 0.0604 ou 6.04%
2. Probabilidade de gastos de US$ 800 ou mais: 0.4103 ou 41.03%
3. Probabilidade de gastos entre US$ 500 e US$ 1.000: 0.7335 ou 73.35%
4. Gastos para os 5% das famílias com planos mais caros: US$ 1119.09 ou mais

***Cap. 6: 40***

1. *Valor para os 10% de bolsas com menor valor: US$ 16308.74*
2. *Porcentagem de bolsas de US$ 22.000 ou mais: 0.0766 ou 7.66%*
3. *Valor para os 3% de bolsas mais valiosas: US$ 22949.67 ou mais*

***Apêndice – Código em Python***

*# lista 2*

*'''*

*Descrição:*

*Exercícios dos capítulos 4, 5 e 6 do livro Estatística Aplicada a Administração e Economia.*

*Cap. 4: 10, 20, 28, 33, 38, 42*

*Cap. 5: 3, 8, 18, 27, 34*

*Cap. 6: 2, 10, 16, 22, 24, 40*

*'''*

*#Cap4:10*

*# Importando bibliotecas necessárias*

*import pandas as pd*

*import numpy as np*

*# Criando o DataFrame com os dados da tabela*

*data = {*

*'Companhia': ['Virgin America', 'JetBlue', 'AirTran Airways', 'Delta Air Lines',*

*'Alaska Airlines', 'Frontier Airlines', 'Southwest Airlines',*

*'US Airways', 'American Airlines', 'United Airlines'],*

*'ChegadasHorario': [83.5, 79.1, 87.1, 86.5, 87.5, 77.9, 83.1, 85.9, 76.9, 77.4],*

*'BagagensExtraviadas': [0.87, 1.88, 1.58, 2.10, 2.93, 2.22, 3.08, 2.14, 2.92, 3.87],*

*'ReclamacoesClientes': [1.50, 0.79, 0.91, 0.73, 0.51, 1.05, 0.25, 1.74, 1.30, 4.24]*

*}*

*df = pd.DataFrame(data)*

*# a. Probabilidade de um voo da Delta Air Lines chegar no horário*

*def questao\_a():*

*prob\_delta = df[df['Companhia'] == 'Delta Air Lines']['ChegadasHorario'].values[0] / 100*

*return prob\_delta*

*# b. Probabilidade de escolher uma companhia com menos de dois relatórios de bagagem extraviada*

*def questao\_b():*

*companhias\_menos\_2\_bagagens = df[df['BagagensExtraviadas'] < 2]*

*prob = len(companhias\_menos\_2\_bagagens) / len(df)*

*return prob*

*# c. Probabilidade de escolher uma companhia com mais de uma reclamação de cliente*

*def questao\_c():*

*companhias\_mais\_1\_reclamacao = df[df['ReclamacoesClientes'] > 1]*

*prob = len(companhias\_mais\_1\_reclamacao) / len(df)*

*return prob*

*# d. Probabilidade de um voo da AirTran Airways não chegar no horário*

*def questao\_d():*

*prob\_airtran\_no\_horario = df[df['Companhia'] == 'AirTran Airways']['ChegadasHorario'].values[0] / 100*

*prob\_airtran\_nao\_horario = 1 - prob\_airtran\_no\_horario*

*return prob\_airtran\_nao\_horario*

*# Executando e mostrando os resultados*

*print("a. Probabilidade de um voo da Delta Air Lines chegar no horário:", questao\_a())*

*print("b. Probabilidade de escolher uma companhia com menos de dois relatórios de bagagem extraviada:", questao\_b())*

*print("c. Probabilidade de escolher uma companhia com mais de uma reclamação de cliente:", questao\_c())*

*print("d. Probabilidade de um voo da AirTran Airways não chegar no horário:", questao\_d())*

*#Cap4:20*

*# Importando bibliotecas necessárias*

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Criando o DataFrame com os dados da tabela*

*data = {*

*'Idade': ['16 a 20', '21 a 24', '25 a 27', '28 ou mais'],*

*'Numero\_Respostas': [191, 467, 244, 42]*

*}*

*df = pd.DataFrame(data)*

*# Calculando o total de respostas*

*total\_respostas = df['Numero\_Respostas'].sum()*

*print(f"Total de respostas: {total\_respostas}")*

*# a. Probabilidade de ser financeiramente independente para cada categoria de idade*

*def questao\_a():*

*df['Probabilidade'] = df['Numero\_Respostas'] / total\_respostas*

*print("\na. Probabilidade para cada categoria de idade:")*

*for i, row in df.iterrows():*

*print(f"   {row['Idade']}: {row['Probabilidade']:.4f} ou {row['Probabilidade']\*100:.2f}%")*

*return df['Probabilidade'].tolist()*

*# b. Probabilidade de ser financeiramente independente antes dos 25 anos*

*def questao\_b():*

*antes\_25 = df[df['Idade'].isin(['16 a 20', '21 a 24'])]['Numero\_Respostas'].sum()*

*prob\_antes\_25 = antes\_25 / total\_respostas*

*print(f"\nb. Probabilidade de ser financeiramente independente antes dos 25 anos: {prob\_antes\_25:.4f} ou {prob\_antes\_25\*100:.2f}%")*

*return prob\_antes\_25*

*# c. Probabilidade de ser financeiramente independente após os 24 anos*

*def questao\_c():*

*apos\_24 = df[df['Idade'].isin(['25 a 27', '28 ou mais'])]['Numero\_Respostas'].sum()*

*prob\_apos\_24 = apos\_24 / total\_respostas*

*print(f"\nc. Probabilidade de ser financeiramente independente após os 24 anos: {prob\_apos\_24:.4f} ou {prob\_apos\_24\*100:.2f}%")*

*return prob\_apos\_24*

*# d. Análise sobre o realismo das expectativas*

*def questao\_d():*

*print("\nd. Análise sobre o realismo das expectativas:")*

*# Calculando a idade média estimada (aproximadamente)*

*idades\_medias = {*

*'16 a 20': 18,  # média aproximada da faixa*

*'21 a 24': 22.5,*

*'25 a 27': 26,*

*'28 ou mais': 30  # valor arbitrário para representar essa faixa*

*}*

*df['Idade\_Media'] = df['Idade'].map(idades\_medias)*

*idade\_media\_ponderada = sum(df['Idade\_Media'] \* df['Numero\_Respostas']) / total\_respostas*

*print(f"   A idade média aproximada em que os adolescentes acreditam que serão independentes: {idade\_media\_ponderada:.2f} anos")*

*print("   Análise: A maioria dos adolescentes (aproximadamente {:.2f}%) espera ser financeiramente independente entre 21 e 24 anos.".format(df.loc[1, 'Probabilidade']\*100))*

*print("   Isso pode ser considerado otimista, dado que muitos jovens nessa faixa etária")*

*print("   ainda estão na faculdade ou iniciando suas carreiras, com salários iniciais mais baixos.")*

*# Criando um gráfico para visualizar as respostas*

*plt.figure(figsize=(10, 6))*

*bars = plt.bar(df['Idade'], df['Numero\_Respostas'], color='skyblue')*

*plt.title('Expectativa de Idade para Independência Financeira')*

*plt.xlabel('Faixa Etária')*

*plt.ylabel('Número de Respostas')*

*# Adicionar valores em cima das barras*

*for bar in bars:*

*height = bar.get\_height()*

*plt.text(bar.get\_x() + bar.get\_width()/2., height,*

*f'{height}', ha='center', va='bottom')*

*# Exibir o gráfico (comentado para não interromper a execução)*

*# plt.show()*

*return idade\_media\_ponderada*

*# Executando todas as funções*

*questao\_a()*

*questao\_b()*

*questao\_c()*

*questao\_d()*

*#cap4:28*

*# Importando bibliotecas necessárias*

*import numpy as np*

*# Dados do problema*

*comercial = 0.458  # 45,8% alugaram por razões comerciais*

*pessoal = 0.54  # 54% alugaram por razões pessoais*

*ambos = 0.30  # 30% alugaram por ambas as razões*

*# a. Probabilidade de um assinante ter alugado um carro por razões comerciais OU pessoais*

*def questao\_a():*

*# Probabilidade da união (A ou B) = P(A) + P(B) - P(A e B)*

*# onde A = alugou por razões comerciais e B = alugou por razões pessoais*

*probabilidade = comercial + pessoal - ambos*

*return probabilidade*

*# b. Probabilidade de um assinante NÃO ter alugado um carro por razões comerciais ou pessoais*

*def questao\_b():*

*# Primeiro calculamos a probabilidade de ter alugado (já feito na questão a)*

*prob\_alugou = questao\_a()*

*# Probabilidade de NÃO ter alugado = 1 - probabilidade de ter alugado*

*probabilidade = 1 - prob\_alugou*

*return probabilidade*

*# Calculando e exibindo os resultados*

*print("Dados do problema:")*

*print(f"- Assinantes que alugaram carro por razões comerciais: {comercial\*100:.1f}%")*

*print(f"- Assinantes que alugaram carro por razões pessoais: {pessoal\*100:.1f}%")*

*print(f"- Assinantes que alugaram carro por ambas as razões: {ambos\*100:.1f}%")*

*print()*

*# Resultados das questões*

*prob\_a = questao\_a()*

*prob\_b = questao\_b()*

*print(f"a. Probabilidade de um assinante ter alugado um carro por razões comerciais OU pessoais: {prob\_a:.4f} ou {prob\_a\*100:.2f}%")*

*print(f"b. Probabilidade de um assinante NÃO ter alugado um carro: {prob\_b:.4f} ou {prob\_b\*100:.2f}%")*

*# Verificação adicional*

*print("\nVerificação:")*

*print(f"Soma das probabilidades (alugou + não alugou): {prob\_a + prob\_b}")*

*#Cap.4:33*

*# Importando bibliotecas necessárias*

*import pandas as pd*

*import numpy as np*

*# Criando o DataFrame com os dados da tabela*

*data = {*

*'Status Matrícula': ['Período Integral', 'Meio Período', 'Totais'],*

*'Administração': [352, 150, 502],*

*'Engenharia': [197, 161, 358],*

*'Outros': [251, 194, 445],*

*'Totais': [800, 505, 1305]*

*}*

*df = pd.DataFrame(data)*

*# Exibindo a tabela*

*print("Tabela original:")*

*print(df)*

*print("\n")*

*# a. Desenvolver tabela de probabilidades conjuntas*

*def questao\_a():*

*print("a. Tabela de probabilidades conjuntas:")*

*prob\_df = pd.DataFrame(index=df['Status Matrícula'], columns=df.columns[1:])*

*total\_geral = df.loc[2, 'Totais']  # Total de estudantes (1305)*

*for i in range(3):  # Linhas*

*for j in range(1, 5):  # Colunas*

*prob\_df.iloc[i, j-1] = df.iloc[i, j] / total\_geral*

*print(prob\_df)*

*return prob\_df*

*# b. Probabilidades marginais por setor de graduação*

*def questao\_b():*

*print("\nb. Probabilidades marginais por setor de graduação:")*

*total\_geral = df.loc[2, 'Totais']  # Total de estudantes (1305)*

*prob\_adm = df.loc[2, 'Administração'] / total\_geral*

*prob\_eng = df.loc[2, 'Engenharia'] / total\_geral*

*prob\_outros = df.loc[2, 'Outros'] / total\_geral*

*print(f"Probabilidade de ser graduado em Administração: {prob\_adm:.4f} ou {prob\_adm\*100:.2f}%")*

*print(f"Probabilidade de ser graduado em Engenharia: {prob\_eng:.4f} ou {prob\_eng\*100:.2f}%")*

*print(f"Probabilidade de ser graduado em Outros cursos: {prob\_outros:.4f} ou {prob\_outros\*100:.2f}%")*

*# Verificando qual setor tem maior potencial*

*if prob\_adm > prob\_eng and prob\_adm > prob\_outros:*

*print("O setor de Administração tem o maior potencial de estudantes de MBA.")*

*elif prob\_eng > prob\_adm and prob\_eng > prob\_outros:*

*print("O setor de Engenharia tem o maior potencial de estudantes de MBA.")*

*else:*

*print("O setor de Outros cursos tem o maior potencial de estudantes de MBA.")*

*return prob\_adm, prob\_eng, prob\_outros*

*# c. Probabilidade de ser graduado em Engenharia e frequentar MBA em tempo integral*

*def questao\_c():*

*total\_geral = df.loc[2, 'Totais']  # Total de estudantes (1305)*

*eng\_tempo\_integral = df.loc[0, 'Engenharia']  # Engenharia em tempo integral (197)*

*prob = eng\_tempo\_integral / total\_geral*

*print(f"\nc. Probabilidade de ser graduado em Engenharia e frequentar MBA em tempo integral: {prob:.4f} ou {prob\*100:.2f}%")*

*return prob*

*# d. Probabilidade de frequentar meio período sendo da Administração*

*def questao\_d():*

*total\_adm = df.loc[2, 'Administração']  # Total de estudantes de Administração (502)*

*adm\_meio\_periodo = df.loc[1, 'Administração']  # Administração em meio período (150)*

*prob = adm\_meio\_periodo / total\_adm*

*print(f"\nd. Probabilidade de frequentar meio período sendo da Administração: {prob:.4f} ou {prob\*100:.2f}%")*

*return prob*

*# e. Verificar independência dos eventos*

*def questao\_e():*

*print("\ne. Verificação de independência dos eventos:")*

*total\_geral = df.loc[2, 'Totais']  # Total de estudantes (1305)*

*# Probabilidade de frequentar em tempo integral*

*prob\_integral = df.loc[0, 'Totais'] / total\_geral*

*# Probabilidade de ser graduado em Administração*

*prob\_adm = df.loc[2, 'Administração'] / total\_geral*

*# Probabilidade de frequentar em tempo integral E ser graduado em Administração*

*prob\_integral\_adm = df.loc[0, 'Administração'] / total\_geral*

*# Se os eventos são independentes, P(A e B) = P(A) \* P(B)*

*prob\_se\_independente = prob\_integral \* prob\_adm*

*print(f"P(Tempo Integral) = {prob\_integral:.4f}")*

*print(f"P(Administração) = {prob\_adm:.4f}")*

*print(f"P(Tempo Integral e Administração) = {prob\_integral\_adm:.4f}")*

*print(f"P(Tempo Integral) \* P(Administração) = {prob\_se\_independente:.4f}")*

*if abs(prob\_integral\_adm - prob\_se\_independente) < 0.001:*

*print("Os eventos são independentes, pois P(A e B) ≈ P(A) \* P(B)")*

*else:*

*print("Os eventos NÃO são independentes, pois P(A e B) ≠ P(A) \* P(B)")*

*# Verificar se são positivamente ou negativamente dependentes*

*if prob\_integral\_adm > prob\_se\_independente:*

*print("Há uma dependência positiva: ser da Administração aumenta a chance de frequentar em tempo integral")*

*else:*

*print("Há uma dependência negativa: ser da Administração diminui a chance de frequentar em tempo integral")*

*return prob\_integral\_adm, prob\_se\_independente*

*# Executando todas as funções*

*tabela\_prob = questao\_a()*

*questao\_b()*

*questao\_c()*

*questao\_d()*

*questao\_e()*

*#Cap.4:38*

*# Importando bibliotecas necessárias*

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*# Criando a tabela de probabilidades conjuntas*

*data = {*

*'Status': ['Satisfatório', 'Inadimplente', 'Total'],*

*'Diploma\_Sim': [0.26, 0.16, 0.42],*

*'Diploma\_Nao': [0.24, 0.34, 0.58],*

*'Total': [0.50, 0.50, 1.00]*

*}*

*df = pd.DataFrame(data)*

*# Exibir a tabela de probabilidades*

*print("Tabela de Probabilidades Conjuntas:")*

*print(df)*

*print("\n")*

*# a. Probabilidade de um aluno ter recebido um diploma universitário*

*def questao\_a():*

*prob\_diploma = df.loc[2, 'Diploma\_Sim']  # Total da coluna Diploma\_Sim*

*print(f"a. Probabilidade de um aluno ter recebido um diploma universitário: {prob\_diploma:.4f} ou {prob\_diploma\*100:.2f}%")*

*return prob\_diploma*

*# b. Probabilidade de um aluno não ter recebido um diploma universitário*

*def questao\_b():*

*prob\_sem\_diploma = df.loc[2, 'Diploma\_Nao']  # Total da coluna Diploma\_Nao*

*print(f"b. Probabilidade de um aluno não ter recebido um diploma universitário: {prob\_sem\_diploma:.4f} ou {prob\_sem\_diploma\*100:.2f}%")*

*return prob\_sem\_diploma*

*# c. Probabilidade de ter empréstimo inadimplente dado que recebeu diploma*

*def questao\_c():*

*prob\_inadimplente\_dado\_diploma = df.loc[1, 'Diploma\_Sim'] / df.loc[2, 'Diploma\_Sim']*

*print(f"c. Probabilidade de ter empréstimo inadimplente dado que recebeu diploma: {prob\_inadimplente\_dado\_diploma:.4f} ou {prob\_inadimplente\_dado\_diploma\*100:.2f}%")*

*return prob\_inadimplente\_dado\_diploma*

*# d. Probabilidade de ter empréstimo inadimplente dado que não recebeu diploma*

*def questao\_d():*

*prob\_inadimplente\_dado\_sem\_diploma = df.loc[1, 'Diploma\_Nao'] / df.loc[2, 'Diploma\_Nao']*

*print(f"d. Probabilidade de ter empréstimo inadimplente dado que não recebeu diploma: {prob\_inadimplente\_dado\_sem\_diploma:.4f} ou {prob\_inadimplente\_dado\_sem\_diploma\*100:.2f}%")*

*return prob\_inadimplente\_dado\_sem\_diploma*

*# e. Impacto de abandonar a faculdade sem diploma para alunos com empréstimo*

*def questao\_e():*

*# Probabilidade condicional de inadimplência dado diploma*

*prob\_inadimplente\_dado\_diploma = df.loc[1, 'Diploma\_Sim'] / df.loc[2, 'Diploma\_Sim']*

*# Probabilidade condicional de inadimplência dado sem diploma*

*prob\_inadimplente\_dado\_sem\_diploma = df.loc[1, 'Diploma\_Nao'] / df.loc[2, 'Diploma\_Nao']*

*# Diferença entre as probabilidades*

*diferenca = prob\_inadimplente\_dado\_sem\_diploma - prob\_inadimplente\_dado\_diploma*

*# Aumento percentual na chance de inadimplência*

*aumento\_percentual = (diferenca / prob\_inadimplente\_dado\_diploma) \* 100*

*print("\ne. Impacto de abandonar a faculdade sem diploma para alunos com empréstimo:")*

*print(f"   Probabilidade de inadimplência com diploma: {prob\_inadimplente\_dado\_diploma:.4f} ou {prob\_inadimplente\_dado\_diploma\*100:.2f}%")*

*print(f"   Probabilidade de inadimplência sem diploma: {prob\_inadimplente\_dado\_sem\_diploma:.4f} ou {prob\_inadimplente\_dado\_sem\_diploma\*100:.2f}%")*

*print(f"   Diferença absoluta: {diferenca:.4f} ou {diferenca\*100:.2f} pontos percentuais")*

*print(f"   Aumento relativo: {aumento\_percentual:.2f}%")*

*print("\n   Conclusão: Não obter um diploma universitário aumenta a probabilidade")*

*print(f"   de inadimplência em {diferenca\*100:.2f} pontos percentuais, o que representa")*

*print(f"   um aumento de {aumento\_percentual:.2f}% na chance de inadimplência.")*

*# Visualização das probabilidades condicionais*

*labels = ['Com Diploma', 'Sem Diploma']*

*satisfatorio = [df.loc[0, 'Diploma\_Sim'] / df.loc[2, 'Diploma\_Sim'],*

*df.loc[0, 'Diploma\_Nao'] / df.loc[2, 'Diploma\_Nao']]*

*inadimplente = [prob\_inadimplente\_dado\_diploma, prob\_inadimplente\_dado\_sem\_diploma]*

*# Criando figura para visualização*

*plt.figure(figsize=(10, 6))*

*x = np.arange(len(labels))*

*width = 0.35*

*# Comentando a plotagem para não interromper a execução*

*# rects1 = plt.bar(x - width/2, satisfatorio, width, label='Pagamento Satisfatório')*

*# rects2 = plt.bar(x + width/2, inadimplente, width, label='Inadimplente')*

*# plt.ylabel('Probabilidade')*

*# plt.title('Status do Empréstimo por Situação do Diploma')*

*# plt.xticks(x, labels)*

*# plt.ylim(0, 1)*

*# plt.legend()*

*# plt.tight\_layout()*

*# plt.show()*

*return diferenca, aumento\_percentual*

*# Executando todas as funções*

*questao\_a()*

*questao\_b()*

*questao\_c()*

*questao\_d()*

*questao\_e()*

*#Cap. 4:42*

*# Importando bibliotecas necessárias*

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Dados do problema*

*p\_priori\_inadimplencia = 0.05  # Probabilidade a priori de inadimplência (5%)*

*p\_deixar\_pagamento\_dado\_nao\_inadimplente = 0.20  # P(deixar pagamento | não inadimplente)*

*p\_deixar\_pagamento\_dado\_inadimplente = 1.0  # P(deixar pagamento | inadimplente)*

*# a. Calcular a probabilidade a posteriori de inadimplência dado que o cliente deixou de efetuar pagamentos*

*def calcular\_posteriori\_inadimplencia():*

*# Aplicando o Teorema de Bayes:*

*# P(inadimplente | deixou pagamento) = P(deixou pagamento | inadimplente) \* P(inadimplente) / P(deixou pagamento)*

*# Primeiro calculamos P(deixou pagamento)*

*# P(deixou pagamento) = P(deixou pagamento | inadimplente) \* P(inadimplente) +*

*#                       P(deixou pagamento | não inadimplente) \* P(não inadimplente)*

*p\_nao\_inadimplente = 1 - p\_priori\_inadimplencia*

*p\_deixar\_pagamento = (p\_deixar\_pagamento\_dado\_inadimplente \* p\_priori\_inadimplencia +*

*p\_deixar\_pagamento\_dado\_nao\_inadimplente \* p\_nao\_inadimplente)*

*# Agora aplicamos o Teorema de Bayes*

*p\_inadimplente\_dado\_deixar\_pagamento = (p\_deixar\_pagamento\_dado\_inadimplente \**

*p\_priori\_inadimplencia / p\_deixar\_pagamento)*

*return p\_inadimplente\_dado\_deixar\_pagamento*

*# b. Analisar se o banco deveria cancelar o cartão se o cliente deixar de fazer um pagamento*

*def analisar\_decisao\_cancelamento():*

*p\_posteriori = calcular\_posteriori\_inadimplencia()*

*# Verificar se a probabilidade a posteriori é maior que 0.20*

*deveria\_cancelar = p\_posteriori > 0.20*

*return p\_posteriori, deveria\_cancelar*

*# Executando as funções*

*p\_posteriori = calcular\_posteriori\_inadimplencia()*

*decisao\_resultado = analisar\_decisao\_cancelamento()*

*# Exibindo os resultados*

*print(f"Probabilidade a priori de inadimplência: {p\_priori\_inadimplencia:.2f} ou {p\_priori\_inadimplencia\*100:.1f}%")*

*print(f"Probabilidade de clientes não inadimplentes deixarem de efetuar pagamento: {p\_deixar\_pagamento\_dado\_nao\_inadimplente:.2f} ou {p\_deixar\_pagamento\_dado\_nao\_inadimplente\*100:.1f}%")*

*print(f"Probabilidade de clientes inadimplentes deixarem de efetuar pagamento: {p\_deixar\_pagamento\_dado\_inadimplente:.2f} ou {p\_deixar\_pagamento\_dado\_inadimplente\*100:.1f}%")*

*print("\na. Probabilidade a posteriori de inadimplência dado que deixou de efetuar pagamento:")*

*print(f"   P(inadimplente | deixou pagamento) = {p\_posteriori:.4f} ou {p\_posteriori\*100:.2f}%")*

*print("\nb. Análise sobre cancelamento do cartão:")*

*print(f"   Probabilidade a posteriori de inadimplência: {p\_posteriori:.4f} ou {p\_posteriori\*100:.2f}%")*

*print(f"   Limite estabelecido pelo banco para cancelamento: 0.20 ou 20%")*

*if decisao\_resultado[1]:*

*print("   Decisão: SIM, o banco deveria cancelar o cartão.")*

*print("   Justificativa: A probabilidade a posteriori de inadimplência ({:.2f}%) é maior que o limite de 20%.".format(p\_posteriori\*100))*

*else:*

*print("   Decisão: NÃO, o banco não deveria cancelar o cartão.")*

*print("   Justificativa: A probabilidade a posteriori de inadimplência ({:.2f}%) é menor que o limite de 20%.".format(p\_posteriori\*100))*

*# Visualização da alteração na probabilidade de inadimplência (antes e depois)*

*# Comentado para não interromper a execução*

*'''*

*labels = ['Probabilidade a priori', 'Probabilidade a posteriori']*

*valores = [p\_priori\_inadimplencia, p\_posteriori]*

*plt.figure(figsize=(10, 6))*

*plt.bar(labels, valores, color=['blue', 'red'])*

*plt.title('Comparação entre probabilidade a priori e a posteriori de inadimplência')*

*plt.ylabel('Probabilidade')*

*plt.axhline(y=0.20, color='green', linestyle='--', label='Limite para cancelamento (20%)')*

*plt.ylim(0, max(valores) \* 1.2)*

*plt.legend()*

*plt.show()*

*'''*

*#Cap.5:3*

*import random*

*def simular\_entrevistas():*

*"""*

*Simula o experimento de três estudantes em entrevistas, onde cada um pode*

*receber ou não uma oferta de emprego.*

*Retorna: O número total de ofertas recebidas pelos três estudantes.*

*"""*

*# Simula o resultado de cada entrevista (True = oferta recebida, False = sem oferta)*

*resultados = [random.choice([True, False]) for \_ in range(3)]*

*# Conta o número total de ofertas*

*num\_ofertas = sum(resultados)*

*return num\_ofertas*

*# a) Enumerar os resultados experimentais*

*print("a) Resultados experimentais possíveis:")*

*print("0 ofertas: nenhum estudante recebe oferta")*

*print("1 oferta: apenas um estudante recebe oferta")*

*print("2 ofertas: dois estudantes recebem ofertas")*

*print("3 ofertas: todos os três estudantes recebem ofertas")*

*# b) Definir uma variável aleatória*

*print("\nb) Definição da variável aleatória:")*

*print("X = número de ofertas recebidas pelos três estudantes")*

*print("X pode assumir os valores: 0, 1, 2 ou 3")*

*print("A variável aleatória é discreta, pois assume um número finito de valores")*

*# c) Mostrar o valor da variável aleatória para cada resultado experimental*

*print("\nc) Valores da variável aleatória para cada resultado experimental:")*

*print("Se nenhum estudante recebe oferta: X = 0")*

*print("Se apenas um estudante recebe oferta: X = 1")*

*print("Se dois estudantes recebem ofertas: X = 2")*

*print("Se todos os três estudantes recebem ofertas: X = 3")*

*# Demonstração da simulação*

*print("\nDemonstração da simulação:")*

*for i in range(5):*

*ofertas = simular\_entrevistas()*

*print(f"Simulação {i+1}: {ofertas} oferta(s) recebida(s)")*

*#Cap. 5:8*

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Dados do problema*

*num\_dias = 20*

*salas\_por\_dia = [1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4]*

*# a) Distribuição discreta de probabilidade empírica*

*def calcular\_distribuicao(dados):*

*# Contagem de ocorrências*

*valores\_unicos = sorted(set(dados))*

*contagem = {valor: dados.count(valor) for valor in valores\_unicos}*

*# Cálculo da frequência relativa (probabilidade empírica)*

*total = len(dados)*

*probabilidade = {valor: contagem[valor] / total for valor in valores\_unicos}*

*return probabilidade*

*# Calcular a distribuição de probabilidade*

*distribuicao = calcular\_distribuicao(salas\_por\_dia)*

*print("a) Distribuição de probabilidade empírica:")*

*for salas, prob in distribuicao.items():*

*print(f"P(X = {salas}) = {prob:.2f} ou {prob\*100:.0f}%")*

*# b) Desenhar o gráfico da distribuição*

*def plotar\_distribuicao(distribuicao):*

*valores = list(distribuicao.keys())*

*probabilidades = list(distribuicao.values())*

*plt.figure(figsize=(10, 6))*

*plt.bar(valores, probabilidades, color='skyblue', edgecolor='black')*

*plt.xlabel('Número de salas de cirurgia em uso')*

*plt.ylabel('Probabilidade')*

*plt.title('Distribuição de Probabilidade - Salas de Cirurgia em Uso')*

*plt.xticks(valores)*

*plt.grid(axis='y', linestyle='--', alpha=0.7)*

*# Adicionar os valores de probabilidade acima das barras*

*for i, v in enumerate(probabilidades):*

*plt.text(valores[i], v + 0.01, f'{v:.2f}', ha='center')*

*# Para visualização em notebook ou janela separada*

*plt.tight\_layout()*

*plt.show()*

*# Imprimir as informações para quem não pode visualizar o gráfico*

*print("\nb) Gráfico da distribuição (representação textual):")*

*for salas, prob in distribuicao.items():*

*barra = '#' \* int(prob \* 50)*

*print(f"{salas} sala(s): {barra} {prob:.2f}")*

*plotar\_distribuicao(distribuicao)*

*# c) Verificar se satisfaz as condições de uma distribuição discreta válida*

*def verificar\_distribuicao\_valida(distribuicao):*

*# Condição 1: Todos os valores de probabilidade devem ser não-negativos*

*condicao1 = all(p >= 0 for p in distribuicao.values())*

*# Condição 2: A soma de todas as probabilidades deve ser igual a 1*

*condicao2 = abs(sum(distribuicao.values()) - 1.0) < 1e-10*

*return condicao1, condicao2*

*condicao1, condicao2 = verificar\_distribuicao\_valida(distribuicao)*

*print("\nc) Verificação das condições para uma distribuição discreta válida:")*

*print(f"1. Todas as probabilidades são não-negativas: {condicao1}")*

*print(f"2. A soma de todas as probabilidades é 1: {condicao2}")*

*print(f"3. Soma das probabilidades = {sum(distribuicao.values()):.10f}")*

*if condicao1 and condicao2:*

*print("A distribuição satisfaz as condições necessárias para ser uma distribuição discreta de probabilidade válida.")*

*else:*

*print("A distribuição NÃO satisfaz todas as condições necessárias.")*

*# Estatísticas adicionais*

*valores = np.array(list(distribuicao.keys()))*

*probabilidades = np.array(list(distribuicao.values()))*

*media = sum(valores \* probabilidades)*

*variancia = sum((valores - media)\*\*2 \* probabilidades)*

*print("\nEstatísticas adicionais:")*

*print(f"Valor esperado (média): {media:.2f} salas")*

*print(f"Variância: {variancia:.2f}")*

*print(f"Desvio padrão: {np.sqrt(variancia):.2f}")*

*#Cap. 5: 18*

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Dados da tabela (em milhares de casas)*

*dados = {*

*'vezes': [0, 1, 2, 3, 4],  # 4 representa "4 vezes ou mais"*

*'proprietarios': [439, 1100, 249, 98, 120],*

*'locatarios': [394, 760, 221, 92, 111]*

*}*

*# Convertendo para arrays numpy*

*vezes = np.array(dados['vezes'])*

*proprietarios = np.array(dados['proprietarios'])*

*locatarios = np.array(dados['locatarios'])*

*# Calculando o total de casas para cada categoria*

*total\_proprietarios = sum(proprietarios)*

*total\_locatarios = sum(locatarios)*

*# a) Variável aleatória x e distribuição de probabilidade para proprietários*

*def calcular\_distribuicao(valores, total):*

*"""Calcula a distribuição de probabilidade."""*

*return valores / total*

*# Distribuição de probabilidade para proprietários*

*prob\_proprietarios = calcular\_distribuicao(proprietarios, total\_proprietarios)*

*print("a) Variável aleatória x = número de vezes que casas ocupadas por proprietários")*

*print("   tiveram interrupção no fornecimento de água com duração de 6 horas ou mais")*

*print("   nos últimos 3 meses.")*

*print("\nDistribuição de probabilidade para x:")*

*for i, prob in enumerate(prob\_proprietarios):*

*valor\_x = "4 ou mais" if i == 4 else str(i)*

*print(f"P(x = {valor\_x}) = {prob:.4f} ou {prob\*100:.2f}%")*

*# b) Valor esperado e variância para x (proprietários)*

*def calcular\_valor\_esperado(valores, probabilidades):*

*"""Calcula o valor esperado (média) de uma distribuição de probabilidade."""*

*return sum(valores \* probabilidades)*

*def calcular\_variancia(valores, probabilidades, valor\_esperado):*

*"""Calcula a variância de uma distribuição de probabilidade."""*

*return sum((valores - valor\_esperado)\*\*2 \* probabilidades)*

*# Para o cálculo correto do valor esperado e variância quando temos "4 ou mais",*

*# vamos considerar como exatamente 4 para simplificar*

*valor\_esperado\_x = calcular\_valor\_esperado(vezes, prob\_proprietarios)*

*variancia\_x = calcular\_variancia(vezes, prob\_proprietarios, valor\_esperado\_x)*

*print("\nb) Para casas ocupadas por proprietários:")*

*print(f"   Valor esperado de x = {valor\_esperado\_x:.4f}")*

*print(f"   Variância de x = {variancia\_x:.4f}")*

*# c) Variável aleatória y e distribuição de probabilidade para locatários*

*prob\_locatarios = calcular\_distribuicao(locatarios, total\_locatarios)*

*print("\nc) Variável aleatória y = número de vezes que casas ocupadas por locatários")*

*print("   tiveram interrupção no fornecimento de água com duração de 6 horas ou mais")*

*print("   nos últimos 3 meses.")*

*print("\nDistribuição de probabilidade para y:")*

*for i, prob in enumerate(prob\_locatarios):*

*valor\_y = "4 ou mais" if i == 4 else str(i)*

*print(f"P(y = {valor\_y}) = {prob:.4f} ou {prob\*100:.2f}%")*

*# d) Valor esperado e variância para y (locatários)*

*valor\_esperado\_y = calcular\_valor\_esperado(vezes, prob\_locatarios)*

*variancia\_y = calcular\_variancia(vezes, prob\_locatarios, valor\_esperado\_y)*

*print("\nd) Para casas ocupadas por locatários:")*

*print(f"   Valor esperado de y = {valor\_esperado\_y:.4f}")*

*print(f"   Variância de y = {variancia\_y:.4f}")*

*# Visualização das distribuições (opcional)*

*def plotar\_comparacao():*

*"""Plota um gráfico comparando as duas distribuições."""*

*plt.figure(figsize=(10, 6))*

*bar\_width = 0.35*

*posicoes = np.arange(len(vezes))*

*plt.bar(posicoes - bar\_width/2, prob\_proprietarios, bar\_width,*

*label='Proprietários', color='skyblue', edgecolor='black')*

*plt.bar(posicoes + bar\_width/2, prob\_locatarios, bar\_width,*

*label='Locatários', color='lightgreen', edgecolor='black')*

*# Adicionando rótulos e título*

*plt.xlabel('Número de interrupções')*

*plt.ylabel('Probabilidade')*

*plt.title('Distribuição de Probabilidade de Interrupções no Fornecimento de Água')*

*plt.xticks(posicoes, ['0', '1', '2', '3', '4 ou mais'])*

*plt.legend()*

*plt.grid(axis='y', linestyle='--', alpha=0.7)*

*plt.tight\_layout()*

*plt.show()*

*# Análise comparativa*

*print("\nAnálise Comparativa:")*

*print(f"Média de interrupções em casas de proprietários: {valor\_esperado\_x:.4f}")*

*print(f"Média de interrupções em casas de locatários: {valor\_esperado\_y:.4f}")*

*if valor\_esperado\_x > valor\_esperado\_y:*

*print("Em média, casas ocupadas por proprietários sofrem mais interrupções.")*

*elif valor\_esperado\_x < valor\_esperado\_y:*

*print("Em média, casas ocupadas por locatários sofrem mais interrupções.")*

*else:*

*print("Em média, ambos os tipos de ocupação sofrem o mesmo número de interrupções.")*

*print(f"\nVariabilidade para proprietários: {variancia\_x:.4f}")*

*print(f"Variabilidade para locatários: {variancia\_y:.4f}")*

*# Apenas para representação textual do gráfico (quando não é possível visualizar)*

*print("\nRepresentação textual das distribuições:")*

*print("\nProprietários:")*

*for i, prob in enumerate(prob\_proprietarios):*

*barra = "#" \* int(prob \* 50)*

*valor = "4+" if i == 4 else str(i)*

*print(f"{valor}: {barra} {prob:.4f}")*

*print("\nLocatários:")*

*for i, prob in enumerate(prob\_locatarios):*

*barra = "#" \* int(prob \* 50)*

*valor = "4+" if i == 4 else str(i)*

*print(f"{valor}: {barra} {prob:.4f}")*

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*# Dados da tabela*

*vezes\_interrupcao = [0, 1, 2, 3, 4]  # 4 representa "4 vezes ou mais"*

*casas\_proprietarios = [439, 1100, 249, 98, 120]  # em milhares*

*casas\_locatarios = [394, 760, 221, 92, 111]  # em milhares*

*# a. Variável aleatória x = número de vezes para casas ocupadas por proprietários*

*# Calcular as probabilidades para x*

*total\_proprietarios = sum(casas\_proprietarios)*

*prob\_x = [valor / total\_proprietarios for valor in casas\_proprietarios]*

*# Calcular valor esperado E[x] e variância para x*

*E\_x = sum(x \* p for x, p in zip(vezes\_interrupcao, prob\_x))*

*E\_x2 = sum(x\*\*2 \* p for x, p in zip(vezes\_interrupcao, prob\_x))*

*Var\_x = E\_x2 - E\_x\*\*2*

*# c. Variável aleatória y = número de vezes para casas ocupadas por locatários*

*# Calcular as probabilidades para y*

*total\_locatarios = sum(casas\_locatarios)*

*prob\_y = [valor / total\_locatarios for valor in casas\_locatarios]*

*# Calcular valor esperado E[y] e variância para y*

*E\_y = sum(y \* p for y, p in zip(vezes\_interrupcao, prob\_y))*

*E\_y2 = sum(y\*\*2 \* p for y, p in zip(vezes\_interrupcao, prob\_y))*

*Var\_y = E\_y2 - E\_y\*\*2*

*# Criando um DataFrame para mostrar a distribuição de probabilidade de x*

*df\_x = pd.DataFrame({*

*'Número de vezes': vezes\_interrupcao,*

*'Casas de proprietários (milhares)': casas\_proprietarios,*

*'Probabilidade': prob\_x*

*})*

*# Criando um DataFrame para mostrar a distribuição de probabilidade de y*

*df\_y = pd.DataFrame({*

*'Número de vezes': vezes\_interrupcao,*

*'Casas de locatários (milhares)': casas\_locatarios,*

*'Probabilidade': prob\_y*

*})*

*# Exibir resultados*

*print("a. Distribuição de probabilidade para x (proprietários):")*

*print(df\_x)*

*print(f"\nValor esperado E[x] = {E\_x:.4f}")*

*print(f"Variância de x = {Var\_x:.4f}")*

*print("\nc. Distribuição de probabilidade para y (locatários):")*

*print(df\_y)*

*print(f"\nValor esperado E[y] = {E\_y:.4f}")*

*print(f"Variância de y = {Var\_y:.4f}")*

*# Comparação visual das distribuições de probabilidade*

*plt.figure(figsize=(10, 6))*

*plt.bar([x - 0.2 for x in vezes\_interrupcao], prob\_x, width=0.4, label='Proprietários (x)')*

*plt.bar([x + 0.2 for x in vezes\_interrupcao], prob\_y, width=0.4, label='Locatários (y)')*

*plt.xlabel('Número de interrupções')*

*plt.ylabel('Probabilidade')*

*plt.title('Distribuição de probabilidade das interrupções de água')*

*plt.xticks(vezes\_interrupcao)*

*plt.legend()*

*plt.grid(axis='y', linestyle='--', alpha=0.7)*

*# Observações para o item e*

*print("\ne. Observações da comparação entre proprietários e locatários:")*

*if E\_x > E\_y:*

*print(f"- Proprietários têm, em média, mais interrupções ({E\_x:.4f} vs {E\_y:.4f}).")*

*elif E\_x < E\_y:*

*print(f"- Locatários têm, em média, mais interrupções ({E\_y:.4f} vs {E\_x:.4f}).")*

*else:*

*print("- Ambos os grupos têm, em média, o mesmo número de interrupções.")*

*if Var\_x > Var\_y:*

*print(f"- A variabilidade é maior entre proprietários (variância {Var\_x:.4f} vs {Var\_y:.4f}).")*

*elif Var\_x < Var\_y:*

*print(f"- A variabilidade é maior entre locatários (variância {Var\_y:.4f} vs {Var\_x:.4f}).")*

*else:*

*print("- Ambos os grupos têm a mesma variabilidade.")*

*# Calcular percentuais de casas sem interrupções*

*pct\_sem\_interrupcao\_prop = prob\_x[0] \* 100*

*pct\_sem\_interrupcao\_loc = prob\_y[0] \* 100*

*print(f"- {pct\_sem\_interrupcao\_prop:.1f}% dos proprietários não tiveram interrupções, contra {pct\_sem\_interrupcao\_loc:.1f}% dos locatários.")*

*# Calculando percentuais para 3 ou mais interrupções*

*pct\_3mais\_prop = sum(prob\_x[3:]) \* 100*

*pct\_3mais\_loc = sum(prob\_y[3:]) \* 100*

*print(f"- {pct\_3mais\_prop:.1f}% dos proprietários tiveram 3 ou mais interrupções, contra {pct\_3mais\_loc:.1f}% dos locatários.")*

*#Cap5:27*

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*from scipy import stats*

*# Dados da tabela*

*data = np.array([*

*[42, 39, 3],  # Qualidade 1, Preços 1, 2, 3*

*[33, 63, 54],  # Qualidade 2, Preços 1, 2, 3*

*[3, 15, 48]   # Qualidade 3, Preços 1, 2, 3*

*])*

*# Total de restaurantes*

*n\_total = np.sum(data)*

*# a. Desenvolver a distribuição de probabilidade bivariada*

*# Calculando a matriz de probabilidades P(X=x, Y=y)*

*prob\_bivariada = data / n\_total*

*# Criando um DataFrame para melhor visualização*

*qualidade = [1, 2, 3]*

*preco = [1, 2, 3]*

*df\_prob = pd.DataFrame(prob\_bivariada, index=qualidade, columns=preco)*

*df\_prob.index.name = 'Qualidade (x)'*

*df\_prob.columns.name = 'Preço (y)'*

*# b. Calcular o valor esperado e a variância para a classificação de qualidade, x*

*# Calculando as probabilidades marginais P(X=x)*

*prob\_x = np.sum(prob\_bivariada, axis=1)*

*df\_prob\_x = pd.DataFrame({'Probabilidade': prob\_x}, index=qualidade)*

*df\_prob\_x.index.name = 'Qualidade (x)'*

*# Valor esperado de X*

*E\_X = np.sum(qualidade \* prob\_x)*

*# Variância de X*

*Var\_X = np.sum((np.array(qualidade) \*\* 2) \* prob\_x) - E\_X \*\* 2*

*# c. Calcular o valor esperado e a variância para o preço da refeição, y*

*# Calculando as probabilidades marginais P(Y=y)*

*prob\_y = np.sum(prob\_bivariada, axis=0)*

*df\_prob\_y = pd.DataFrame({'Probabilidade': prob\_y}, index=preco)*

*df\_prob\_y.index.name = 'Preço (y)'*

*# Valor esperado de Y*

*E\_Y = np.sum(preco \* prob\_y)*

*# Variância de Y*

*Var\_Y = np.sum((np.array(preco) \*\* 2) \* prob\_y) - E\_Y \*\* 2*

*# d. Calcular a covariância entre x e y*

*# E[XY] - cálculo do valor esperado do produto*

*E\_XY = 0*

*for i in range(len(qualidade)):*

*for j in range(len(preco)):*

*E\_XY += qualidade[i] \* preco[j] \* prob\_bivariada[i, j]*

*# Covariância*

*Cov\_XY = E\_XY - E\_X \* E\_Y*

*# e. Calcular o coeficiente de correlação*

*Corr\_XY = Cov\_XY / np.sqrt(Var\_X \* Var\_Y)*

*# Criando heatmap para visualizar a distribuição bivariada*

*plt.figure(figsize=(10, 8))*

*sns.heatmap(df\_prob, annot=True, cmap="YlGnBu", fmt=".3f", cbar\_kws={'label': 'Probabilidade'})*

*plt.title('Distribuição de Probabilidade Bivariada - Qualidade vs Preço')*

*# Criando gráfico de barras para as distribuições marginais*

*fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))*

*ax1.bar(qualidade, prob\_x)*

*ax1.set\_xlabel('Qualidade (x)')*

*ax1.set\_ylabel('Probabilidade')*

*ax1.set\_title('Distribuição Marginal de Qualidade')*

*ax1.set\_xticks(qualidade)*

*ax2.bar(preco, prob\_y)*

*ax2.set\_xlabel('Preço (y)')*

*ax2.set\_ylabel('Probabilidade')*

*ax2.set\_title('Distribuição Marginal de Preço')*

*ax2.set\_xticks(preco)*

*# Exibindo resultados*

*print("a. Distribuição de Probabilidade Bivariada P(X=x, Y=y):")*

*print(df\_prob)*

*print("\n")*

*print("b. Distribuição Marginal de Qualidade P(X=x):")*

*print(df\_prob\_x)*

*print(f"Valor Esperado E[X] = {E\_X:.4f}")*

*print(f"Variância Var[X] = {Var\_X:.4f}")*

*print("\n")*

*print("c. Distribuição Marginal de Preço P(Y=y):")*

*print(df\_prob\_y)*

*print(f"Valor Esperado E[Y] = {E\_Y:.4f}")*

*print(f"Variância Var[Y] = {Var\_Y:.4f}")*

*print("\n")*

*print("d. Covariância:")*

*print(f"E[XY] = {E\_XY:.4f}")*

*print(f"Covariância entre X e Y = {Cov\_XY:.4f}")*

*enunciado\_d = 1.6691*

*print(f"O enunciado afirma que Var(X + Y) = 1.6691")*

*var\_x\_plus\_y = Var\_X + Var\_Y + 2\*Cov\_XY*

*print(f"Calculamos Var(X + Y) = Var(X) + Var(Y) + 2\*Cov(X,Y) = {var\_x\_plus\_y:.4f}")*

*print("\n")*

*print("e. Coeficiente de Correlação:")*

*print(f"Correlação entre X e Y = {Corr\_XY:.4f}")*

*# Análise da relação entre qualidade e preço*

*print("\nAnálise da relação entre qualidade e preço:")*

*if Corr\_XY > 0.7:*

*forca = "forte"*

*elif Corr\_XY > 0.3:*

*forca = "moderada"*

*else:*

*forca = "fraca"*

*print(f"A correlação entre qualidade e preço é {forca} e positiva ({Corr\_XY:.4f}).")*

*print(f"Isso indica que restaurantes com melhor qualidade tendem a ter preços mais altos.")*

*# Probabilidade condicional de alta qualidade dado baixo preço*

*p\_alta\_dado\_baixo = prob\_bivariada[2, 0] / prob\_y[0]*

*print(f"\nProbabilidade de encontrar qualidade alta (3) dado preço baixo (1) = {p\_alta\_dado\_baixo:.4f}")*

*print(f"Isso equivale a {p\_alta\_dado\_baixo\*100:.2f}% dos restaurantes de preço baixo.")*

*if p\_alta\_dado\_baixo < 0.1:*

*resposta = "Improvável"*

*else:*

*resposta = "Possível, mas não comum"*

*print(f"Conclusão: É {resposta} encontrar um restaurante de baixo custo com alta qualidade nesta cidade.")*

*#Cap.5:34*

*import numpy as np*

*from scipy import stats*

*# Dados do problema*

*p\_adolescentes\_pandora = 0.39  # 39% dos adolescentes usam o Pandora Media*

*n = 10  # número de adolescentes na amostra aleatória*

*# a. É um experimento binomial?*

*# Sim, pois temos:*

*# 1. Número fixo de ensaios (n=10)*

*# 2. Cada ensaio tem apenas dois resultados possíveis (usa ou não usa o Pandora)*

*# 3. A probabilidade de sucesso é constante (p=0.39)*

*# 4. Os ensaios são independentes (assume-se que os adolescentes são selecionados aleatoriamente)*

*# b. Probabilidade de que nenhum dos 10 adolescentes use o serviço*

*prob\_nenhum = stats.binom.pmf(k=0, n=n, p=p\_adolescentes\_pandora)*

*print(f"b. Probabilidade de nenhum usar o Pandora: {prob\_nenhum:.6f}")*

*# c. Probabilidade de que 4 dos 10 adolescentes usem o serviço*

*prob\_quatro = stats.binom.pmf(k=4, n=n, p=p\_adolescentes\_pandora)*

*print(f"c. Probabilidade de exatamente 4 usarem o Pandora: {prob\_quatro:.6f}")*

*# d. Probabilidade de que pelo menos 2 dos 10 adolescentes usem o serviço*

*# Isto é igual a 1 menos a probabilidade de que 0 ou 1 usem*

*prob\_menos\_que\_dois = stats.binom.pmf(k=0, n=n, p=p\_adolescentes\_pandora) + stats.binom.pmf(k=1, n=n, p=p\_adolescentes\_pandora)*

*prob\_pelo\_menos\_dois = 1 - prob\_menos\_que\_dois*

*print(f"d. Probabilidade de pelo menos 2 usarem o Pandora: {prob\_pelo\_menos\_dois:.6f}")*

*# Alternativamente para o item d, podemos usar a função de distribuição cumulativa*

*# prob\_pelo\_menos\_dois = 1 - stats.binom.cdf(k=1, n=n, p=p\_adolescentes\_pandora)*

*#Cap.6:2*

*import numpy as np*

*import matplotlib.pyplot as plt*

*from scipy import stats*

*# Dados do problema: X ~ U(10, 20)*

*a = 10  # limite inferior*

*b = 20  # limite superior*

*# Função densidade de probabilidade para X ~ U(a, b)*

*def pdf(x):*

*if a <= x <= b:*

*return 1/(b-a)*

*else:*

*return 0*

*# a. Gráfico da função densidade de probabilidade*

*x\_values = np.linspace(8, 22, 1000)*

*y\_values = [pdf(x) for x in x\_values]*

*plt.figure(figsize=(10, 6))*

*plt.plot(x\_values, y\_values)*

*plt.fill\_between(x\_values, y\_values, where=[(x >= a and x <= b) for x in x\_values], alpha=0.3)*

*plt.grid(True)*

*plt.title('Função Densidade de Probabilidade de X ~ U(10, 20)')*

*plt.xlabel('x')*

*plt.ylabel('f(x)')*

*plt.axhline(y=0, color='k', linestyle='-', alpha=0.3)*

*plt.axvline(x=a, color='r', linestyle='--', alpha=0.5)*

*plt.axvline(x=b, color='r', linestyle='--', alpha=0.5)*

*plt.text(a-0.5, 0.05, f'a = {a}')*

*plt.text(b-0.5, 0.05, f'b = {b}')*

*plt.text((a+b)/2, 1/(b-a)+0.01, f'f(x) = {1/(b-a)}')*

*plt.show()*

*# b. Calcule P(x < 15)*

*prob\_less\_than\_15 = (15 - a)/(b - a)*

*print(f"b. P(X < 15) = {prob\_less\_than\_15}")*

*# c. Calcule P(12 ≤ x ≤ 18)*

*prob\_between\_12\_and\_18 = (18 - 12)/(b - a)*

*print(f"c. P(12 ≤ X ≤ 18) = {prob\_between\_12\_and\_18}")*

*# d. Calcule E(X)*

*# Para uma distribuição uniforme, E(X) = (a + b)/2*

*expected\_value = (a + b)/2*

*print(f"d. E(X) = {expected\_value}")*

*# e. Calcule Var(X)*

*# Para uma distribuição uniforme, Var(X) = (b - a)²/12*

*variance = (b - a)\*\*2/12*

*print(f"e. Var(X) = {variance}")*

*#Cap.6:10*

*import numpy as np*

*import matplotlib.pyplot as plt*

*from scipy import stats*

*import matplotlib.patches as patches*

*# Definir o intervalo de valores para z*

*z = np.linspace(-4, 4, 1000)*

*# Calcular a função de densidade de probabilidade da distribuição normal padrão*

*pdf = stats.norm.pdf(z)*

*# Criar o gráfico*

*plt.figure(figsize=(10, 6))*

*plt.plot(z, pdf, 'b-', lw=2, label='Distribuição Normal Padrão')*

*# Marcar os pontos específicos no eixo x*

*pontos\_x = [-3, -2, -1, 0, 1, 2, 3]*

*pontos\_y = [0] \* len(pontos\_x)*

*plt.scatter(pontos\_x, pontos\_y, color='red', s=50)*

*# Adicionar rótulos aos pontos*

*for i, (x, y) in enumerate(zip(pontos\_x, pontos\_y)):*

*plt.annotate(str(x), (x, y-0.01), ha='center', fontsize=12)*

*# Adicionar rótulos e título*

*plt.title('Distribuição Normal Padrão', fontsize=14)*

*plt.xlabel('z', fontsize=12)*

*plt.ylabel('Densidade de Probabilidade', fontsize=12)*

*# Configurar os limites do eixo y*

*plt.ylim(0, 0.45)*

*# Calcular as probabilidades solicitadas usando scipy*

*prob\_a = stats.norm.cdf(1.5)*

*prob\_b = stats.norm.cdf(1)*

*prob\_c = stats.norm.cdf(1.5) - stats.norm.cdf(1)*

*prob\_d = stats.norm.cdf(2.5) - stats.norm.cdf(0)*

*# Adicionar as áreas sombreadas para cada probabilidade*

*# a. P(z ≤ 1,5)*

*x\_a = np.linspace(-4, 1.5, 500)*

*y\_a = stats.norm.pdf(x\_a)*

*plt.fill\_between(x\_a, y\_a, alpha=0.3, color='red', label=f'a. P(z ≤ 1,5) = {prob\_a:.4f}')*

*# b. P(z ≤ 1)*

*x\_b = np.linspace(-4, 1, 500)*

*y\_b = stats.norm.pdf(x\_b)*

*plt.fill\_between(x\_b, y\_b, alpha=0.3, color='green', label=f'b. P(z ≤ 1) = {prob\_b:.4f}')*

*# c. P(1 ≤ z ≤ 1,5)*

*x\_c = np.linspace(1, 1.5, 500)*

*y\_c = stats.norm.pdf(x\_c)*

*plt.fill\_between(x\_c, y\_c, alpha=0.5, color='purple', label=f'c. P(1 ≤ z ≤ 1,5) = {prob\_c:.4f}')*

*# d. P(0 ≤ z ≤ 2,5)*

*x\_d = np.linspace(0, 2.5, 500)*

*y\_d = stats.norm.pdf(x\_d)*

*plt.fill\_between(x\_d, y\_d, alpha=0.3, color='orange', label=f'd. P(0 ≤ z ≤ 2,5) = {prob\_d:.4f}')*

*# Adicionar legenda*

*plt.legend(loc='upper right', fontsize=10)*

*# Resultados*

*print("Resultados das probabilidades:")*

*print(f"a. P(z ≤ 1,5) = {prob\_a:.4f}")*

*print(f"b. P(z ≤ 1) = {prob\_b:.4f}")*

*print(f"c. P(1 ≤ z ≤ 1,5) = {prob\_c:.4f}")*

*print(f"d. P(0 ≤ z ≤ 2,5) = {prob\_d:.4f}")*

*# Mostrar o gráfico*

*plt.grid(True, alpha=0.3)*

*plt.tight\_layout()*

*plt.show()*

*# Cap. 6:16*

*import numpy as np*

*from scipy import stats*

*# Função para encontrar z tal que P(Z > z) = área*

*def encontrar\_z\_area\_direita(area):*

*# A função ppf calcula o quantil, mas para área à esquerda*

*# Para área à direita, usamos 1-área*

*return stats.norm.ppf(1 - area)*

*# Situações para calcular*

*areas = [0.01, 0.025, 0.05, 0.10]*

*letras = ['a', 'b', 'c', 'd']*

*# Calcular e exibir os resultados*

*print("Resultados:")*

*for letra, area in zip(letras, areas):*

*z = encontrar\_z\_area\_direita(area)*

*print(f"{letra}. Área à direita de z é {area}: z = {z:.4f}")*

*#Cap. 6:22*

*import numpy as np*

*from scipy import stats*

*# Parâmetros da distribuição normal*

*media = 8.35  # tempo médio em horas*

*desvio\_padrao = 2.5  # desvio padrão em horas*

*# Criar a distribuição normal*

*distribuicao = stats.norm(loc=media, scale=desvio\_padrao)*

*# a. Qual é a probabilidade de que um espectador assista à TV durante 5 a 10 horas por dia?*

*prob\_entre\_5\_e\_10 = distribuicao.cdf(10) - distribuicao.cdf(5)*

*print(f"a. Probabilidade de assistir entre 5 e 10 horas: {prob\_entre\_5\_e\_10:.4f} ou {prob\_entre\_5\_e\_10\*100:.2f}%")*

*# b. Por quantas horas um espectador deve assistir à TV para estar entre os 3% que mais assistem TV?*

*# Os 3% que mais assistem correspondem ao percentil 97*

*horas\_top\_3\_percent = distribuicao.ppf(0.97)*

*print(f"b. Horas para estar entre os 3% que mais assistem: {horas\_top\_3\_percent:.2f} horas")*

*# c. Qual é a probabilidade de que um telespectador assista à TV por mais de 3 horas por dia?*

*prob\_mais\_de\_3 = 1 - distribuicao.cdf(3)*

*print(f"c. Probabilidade de assistir mais de 3 horas: {prob\_mais\_de\_3:.4f} ou {prob\_mais\_de\_3\*100:.2f}%")*

*#Cap.6:24*

*import numpy as np*

*from scipy import stats*

*# Parâmetros da distribuição normal*

*media = 749  # gasto médio em US$*

*desvio\_padrao = 225  # desvio padrão em US$*

*# Criar a distribuição normal*

*distribuicao = stats.norm(loc=media, scale=desvio\_padrao)*

*# a. Qual é a probabilidade de as despesas familiares do fim de semana serem inferiores a US$ 400?*

*prob\_menos\_400 = distribuicao.cdf(400)*

*print(f"a. Probabilidade de gastos inferiores a US$ 400: {prob\_menos\_400:.4f} ou {prob\_menos\_400\*100:.2f}%")*

*# b. Qual é a probabilidade de as despesas familiares do fim de semana serem de US$ 800 ou mais?*

*prob\_800\_ou\_mais = 1 - distribuicao.cdf(800)*

*print(f"b. Probabilidade de gastos de US$ 800 ou mais: {prob\_800\_ou\_mais:.4f} ou {prob\_800\_ou\_mais\*100:.2f}%")*

*# c. Qual é a probabilidade de as despesas familiares do final de semana ficarem entre US$ 500 e US$ 1.000?*

*prob\_entre\_500\_1000 = distribuicao.cdf(1000) - distribuicao.cdf(500)*

*print(f"c. Probabilidade de gastos entre US$ 500 e US$ 1.000: {prob\_entre\_500\_1000:.4f} ou {prob\_entre\_500\_1000\*100:.2f}%")*

*# d. Quais seriam as despesas do fim de semana do Dia do Trabalho para os 5% das famílias com planos de viagem mais caros?*

*# Os 5% mais caros correspondem ao percentil 95*

*gastos\_top\_5\_percent = distribuicao.ppf(0.95)*

*print(f"d. Gastos para os 5% das famílias com planos mais caros: US$ {gastos\_top\_5\_percent:.2f} ou mais")*

*#Cap.6:40*

*import numpy as np*

*from scipy import stats*

*# Parâmetros da distribuição normal*

*media = 19000  # valor médio da bolsa em US$*

*desvio\_padrao = 2100  # desvio padrão em US$*

*# Criar a distribuição normal*

*distribuicao = stats.norm(loc=media, scale=desvio\_padrao)*

*# a. Para os 10% de bolsas de estudos com menor valor para atletas quanto elas valem?*

*# Isso corresponde ao percentil 10*

*valor\_percentil\_10 = distribuicao.ppf(0.10)*

*print(f"a. Valor para os 10% de bolsas com menor valor: US$ {valor\_percentil\_10:.2f}")*

*# b. Qual a porcentagem de bolsas de estudos avaliadas em US$ 22.000 ou mais?*

*prob\_22000\_ou\_mais = 1 - distribuicao.cdf(22000)*

*print(f"b. Porcentagem de bolsas de US$ 22.000 ou mais: {prob\_22000\_ou\_mais:.4f} ou {prob\_22000\_ou\_mais\*100:.2f}%")*

*# c. Para os 3% de bolsas de estudos que são mais valiosas quanto elas valem?*

*# Isso corresponde ao percentil 97*

*valor\_percentil\_97 = distribuicao.ppf(0.97)*

*print(f"c. Valor para os 3% de bolsas mais valiosas: US$ {valor\_percentil\_97:.2f} ou mais")*