Questão 1 A)

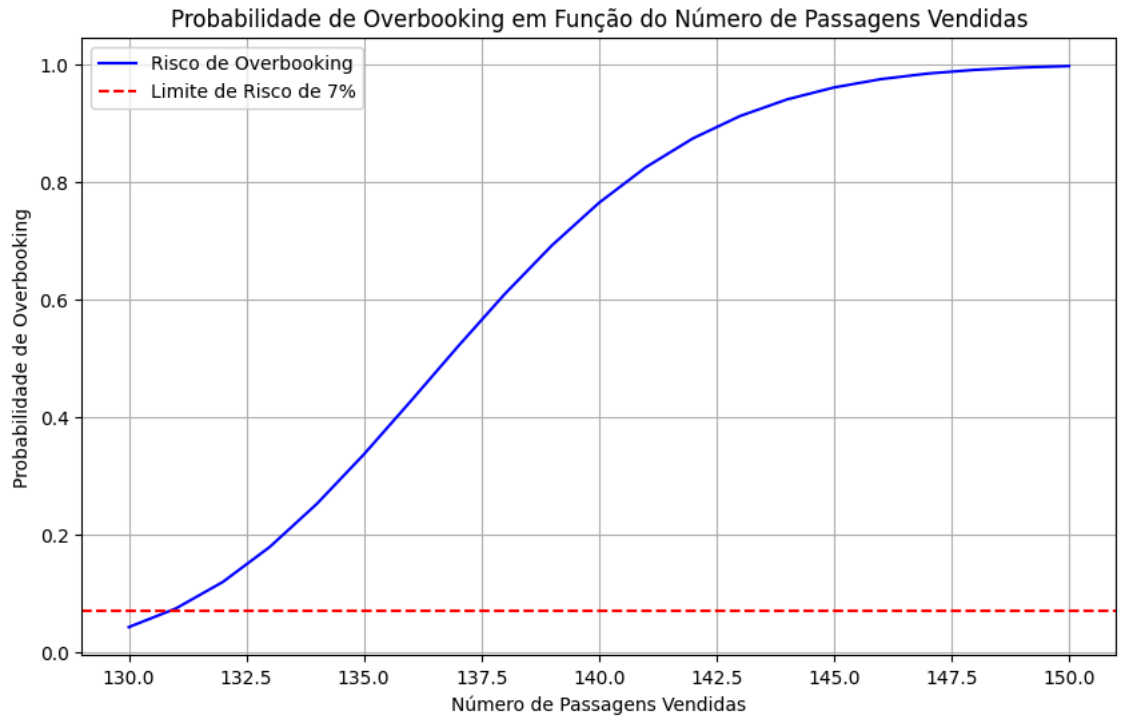
```
from scipy.stats import binom


n = 130
p = 0.88
capacidade = 120


prob_overbooking = binom.sf(capacidade, n, p)


print(f"A probabilidade de overbooking é: {prob_overbooking:.4f} ou {prob_overbooking:.2%}")
```

```
A probabilidade de overbooking é: 0.0426 ou 4.26%
```

Questão 1 B e C)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom


capacidade_voo = 120
probabilidade_comparecimento = 0.88
num_passagens_iniciais = 130
num_passageiros_vendidos_maximo = 150

# overbooking
def calcular_probabilidade_overbooking(passagens_vendidas):
    # Calcular a probabilidade de ter mais do que 120 passageiros no voo
    probabilidade_overbooking = 1 - binom.cdf(capacidade_voo, passagens_vendidas, probabilidade_comparecimento)
    return probabilidade_overbooking

# passagens vendidas
passagens_vendidas = np.arange(130, num_passageiros_vendidos_maximo + 1)
risco_overbooking = [calcular_probabilidade_overbooking(pv) for pv in passagens_vendidas]

# gráfico
plt.figure(figsize=(10, 6))
plt.plot(passagens_vendidas, risco_overbooking, label='Risco de Overbooking', color='b')
plt.axhline(y=0.07, color='r', linestyle='--', label='Limite de Risco de 7%')
plt.xlabel('Número de Passagens Vendidas')
plt.ylabel('Probabilidade de Overbooking')
plt.title('Probabilidade de Overbooking em Função do Número de Passagens Vendidas')
plt.legend()
plt.grid(True)
plt.show()

# Identificar o número máximo de passagens que pode ser vendido mantendo o risco de overbooking abaixo de 7%
limite_risco = 0.07
passagens_maximas = passagens_vendidas[np.where(np.array(risco_overbooking) <= limite_risco)[0][0]]

print(f'O número máximo de passagens que pode ser vendido mantendo o risco de overbooking abaixo de 7% é: {passagens_maximas}')

# cálculos financeiros (viabilidade de vender +10 passagens)
valor_indenizacao_por_passageiro = 500  # Valor estimado da indenização por passageiro não embarcado
preco_medio_passagem = 1000  # Preço médio de cada passagem
passagens_extra = 10  # Número de passagens extras vendidas além da capacidade

# custo de indenização com base no número de passageiros vendidos
def calcular_custo_indenizacao(passagens_vendidas):
    # Calcular a probabilidade de overbooking (número de passageiros que não cabem no voo)
    probabilidade_overbooking = 1 - binom.cdf(capacidade_voo, passagens_vendidas, probabilidade_comparecimento)

    # Estimativa do número de passageiros afetados (que não poderão embarcar)
    passageiros_afetados = int(np.ceil(probabilidade_overbooking * passagens_vendidas))

    # Calculando o custo total das indenizações
    custo_total_indenizacoes = passageiros_afetados * valor_indenizacao_por_passageiro
    return custo_total_indenizacoes, passageiros_afetados, probabilidade_overbooking

# Calculando o custo de indenização para 130 passagens vendidas
custo_indenizacao_130, passageiros_afetados_130, probabilidade_overbooking_130 = calcular_custo_indenizacao(130)

# Calculando o aumento na receita pela venda de 10 passagens extras
aumento_na_receita = preco_medio_passagem * passagens_extra
```

```
# Comparando a receita extra com o custo de indenizações
print(f"\nCusto total das indenizações para 130 passagens vendidas: R${custo_indenizacao_130}")
print(f"Probabilidade de overbooking para 130 passagens: {probabilidade_overbooking_130:.2f}")
print(f"Número estimado de passageiros afetados: {passageiros_afetados_130}")
print(f"Aumento na receita pela venda de 10 passagens extras: R${aumento_na_receita}")

# viabilidade financeira
if aumento_na_receita > custo_indenizacao_130:
    print("\nFinanceiramente, vender 10 passagens extras é viável, pois o aumento na receita supera o custo das indenizações.")
else:
    print("\nFinanceiramente, vender 10 passagens extras não é viável, pois o custo das indenizações é maior do que o aumento na receita.


print("\nConsiderações sobre a imagem de marca:")
print("Vender mais passagens do que a capacidade pode prejudicar a imagem da empresa, principalmente se houverem muitos passageiros afeta
print("Além disso, o custo indireto de imagem e perda de lealdade de clientes pode ser mais alto do que o valor das indenizações.")
```



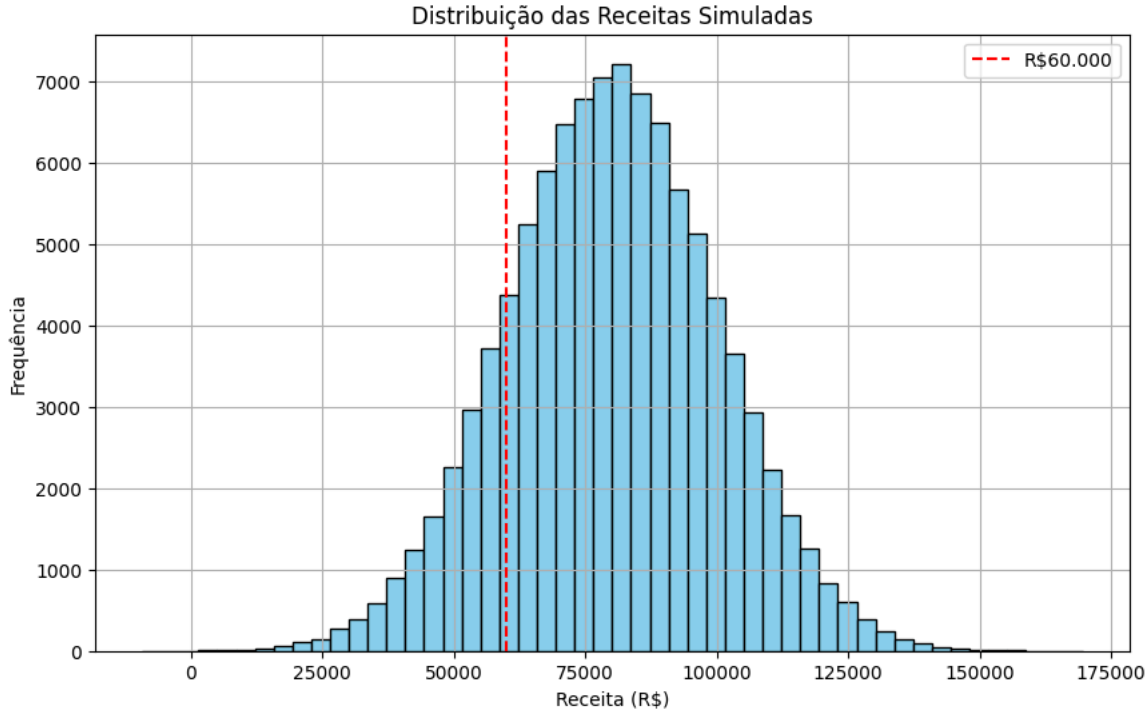

```
O número máximo de passagens que pode ser vendido mantendo o risco de overbooking abaixo de 7% é: 130

Custo total das indenizações para 130 passagens vendidas: R$3000
Probabilidade de overbooking para 130 passagens: 0.04
Número estimado de passageiros afetados: 6
Aumento na receita pela venda de 10 passagens extras: R$10000

Financeiramente, vender 10 passagens extras é viável, pois o aumento na receita supera o custo das indenizações.

Considerações sobre a imagem de marca:
Vender mais passagens do que a capacidade pode prejudicar a imagem da empresa, principalmente se houverem muitos passageiros afetadc
```

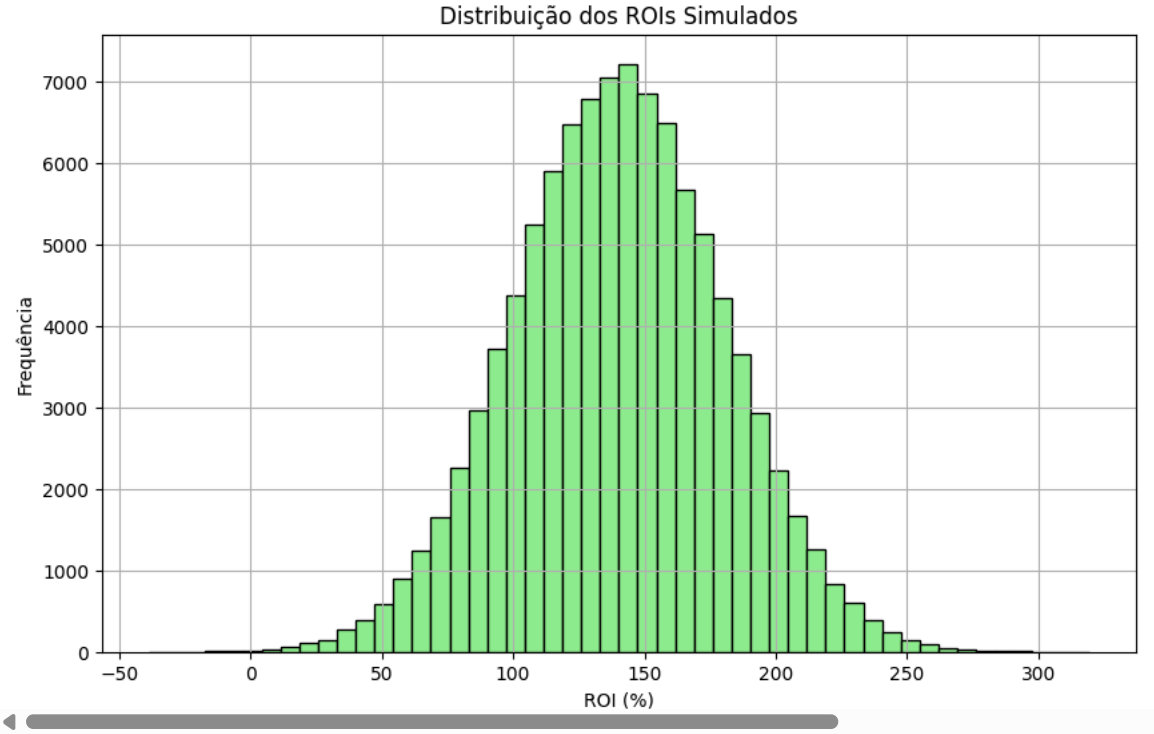
Questão 2 B)

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


np.random.seed(42)

investimento_inicial = 50000
custo_operacional_anual = 10000
receita_esperada = 80000
desvio_padrao_receita = 20000
n_simulacoes = 100000

# Simulação de receitas
receitas_simuladas = np.random.normal(loc=receita_esperada, scale=desvio_padrao_receita, size=n_simulacoes)

# Cálculo do Lucro e ROI
lucros = receitas_simuladas - custo_operacional_anual
rois = (lucros / investimento_inicial) * 100
```

```
# probabilidades
prob_receita_abaixo_60000 = np.mean(receitas_simuladas < 60000) * 100

# Tabela
cenarios = {
    'Cenário': ['Otimista', 'Realista', 'Pessimista'],
    'Receita (R$)': [100000, 80000, 50000],
    'Lucro (R$)': [100000-10000, 80000-10000, 50000-10000],
    'ROI (%)': [(100000-10000)/50000*100, (80000-10000)/50000*100, (50000-10000)/50000*100]
}
df_cenarios = pd.DataFrame(cenarios)


print(f"Probabilidade de Receita < R$60.000: {prob_receita_abaixo_60000:.2f}%")
print("\nTabela de Cenários:")
print(df_cenarios)

# Gráficos

# Receita
plt.figure(figsize=(10,6))
plt.hist(receitas_simuladas, bins=50, color='skyblue', edgecolor='black')
plt.axvline(x=60000, color='red', linestyle='--', label='R$60.000')
plt.title('Distribuição das Receitas Simuladas')
plt.xlabel('Receita (R$)')
plt.ylabel('Frequência')
plt.legend()
plt.grid(True)
plt.show()

# ROI
plt.figure(figsize=(10,6))
plt.hist(rois, bins=50, color='lightgreen', edgecolor='black')
plt.title('Distribuição dos ROIs Simulados')
plt.xlabel('ROI (%)')
plt.ylabel('Frequência')
plt.grid(True)
plt.show()
```

```
Probabilidade de Receita < R$60.000: 15.85%

Tabela de Cenários:
      Cenário  Receita (R$)  Lucro (R$)  ROI (%)
0    Otimista        100000       90000    180.0
1    Realista         80000       70000    140.0
2  Pessimista         50000       40000     80.0
```



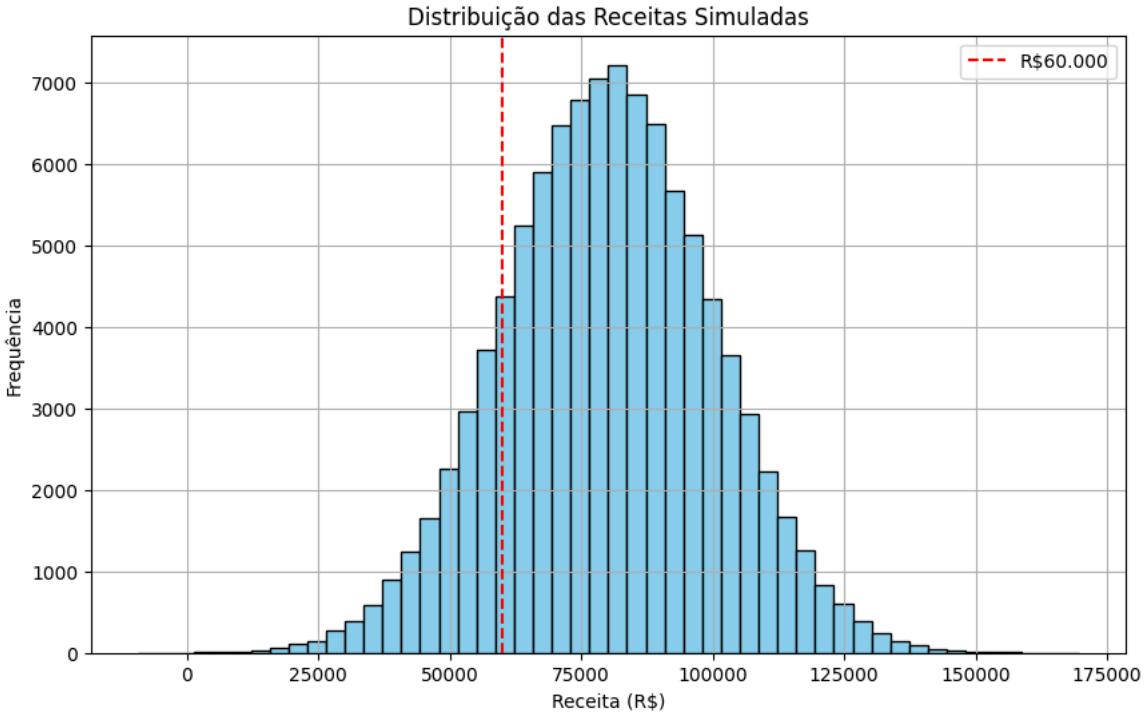


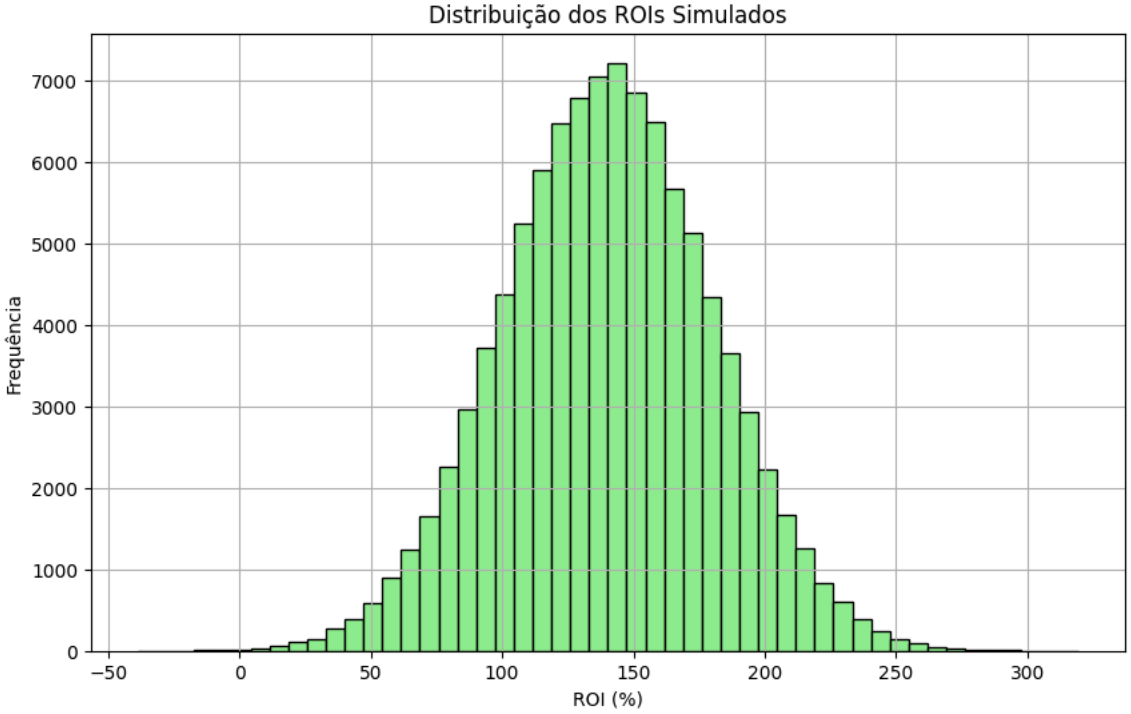
Questão 2 B) (sem os valores)

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

# Entrada dos dados pelo usuário
ii = float(input("Digite o valor do investimento inicial (R$): "))
coa = float(input("Digite o custo operacional anual (R$): "))
re = float(input("Digite a receita esperada (R$): "))
dpr = float(input("Digite o desvio padrão da receita (R$): "))
n_simulacoes = int(input("Digite o número de simulações: "))

# Semente para simulações reproduzíveis
np.random.seed(42)
```

```
# Simulação de receitas
receitas_simuladas = np.random.normal(loc=re, scale=dpr, size=n_simulacoes)

# Cálculo do Lucro e ROI
lucros = receitas_simuladas - coa
rois = (lucros / ii) * 100

# Cálculo da probabilidade de receita abaixo de R$ 60.000
prob_receita_abaixo_60000 = np.mean(receitas_simuladas < 60000) * 100

# Tabela de cenários fixos
cenarios = {
    'Cenário': ['Otimista', 'Realista', 'Pessimista'],
    'Receita (R$)': [100000, 80000, 50000],
    'Lucro (R$)': [100000 - coa,
                   80000 - coa,
                   50000 - coa],
    'ROI (%)': [(100000 - coa) / ii * 100,
                (80000 - coa) / ii * 100,
                (50000 - coa) / ii * 100]
}
df_cenarios = pd.DataFrame(cenarios)

# Exibição dos resultados
print(f"\nProbabilidade de Receita < R$60.000: {prob_receita_abaixo_60000:.2f}%")
print("\nTabela de Cenários:")
print(df_cenarios)

# Gráficos

# Gráfico da Receita Simulada
plt.figure(figsize=(10,6))
plt.hist(receitas_simuladas, bins=50, color='skyblue', edgecolor='black')
plt.axvline(x=60000, color='red', linestyle='--', label='R$60.000')
plt.title('Distribuição das Receitas Simuladas')
plt.xlabel('Receita (R$)')
plt.ylabel('Frequência')
plt.legend()
plt.grid(True)
plt.show()

# Gráfico do ROI Simulado
plt.figure(figsize=(10,6))
plt.hist(rois, bins=50, color='lightgreen', edgecolor='black')
plt.title('Distribuição dos ROIs Simulados')
plt.xlabel('ROI (%)')
plt.ylabel('Frequência')
plt.grid(True)
plt.show()
```

```
Digite o custo operacional anual (R$): 10000
Digite a receita esperada (R$): 80000
Digite o desvio padrão da receita (R$): 20000
Digite o número de simulações: 100000

Probabilidade de Receita < R$60.000: 15.85%

Tabela de Cenários:
      Cenário  Receita (R$)  Lucro (R$)  ROI (%)
0    Otimista        100000     90000.0    180.0
1    Realista         80000     70000.0    140.0
2  Pessimista         50000     40000.0     80.0
```





Questão extra

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QFormLayout, QLabel, QLineEdit, QPushButton, QHBoxLayout, QFrame
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPalette, QColor
from scipy.stats import binom
```

```
    class App(QWidget):
        def __init__(self):
            super().__init__()

            self.setWindowTitle('Simulador de Overbooking e ROI')
            self.setGeometry(100, 100, 600, 500)

            # Estilização da interface
            self.setStyleSheet("""
                QWidget {
                    background-color: #f5d0cb; /* Rosa claro */
                    color: #4b0082; /* Texto roxo escuro */
                    font-family: 'Arial', sans-serif;
                }
                QLabel {
                    font-size: 14px;
                    font-weight: bold;
                }
                QLineEdit {
                    background-color: #ffffff;
                    border: 2px solid #e5a1b7;
                    border-radius: 5px;
                    padding: 5px;
                    font-size: 12px;
                }
                QPushButton {
                    background-color: #f48fb1;
                    color: white;
                    border: 2px solid #f06292;
                    border-radius: 5px;
                    padding: 10px;
                    font-size: 14px;
                }
                QPushButton:hover {
                    background-color: #f06292;
                }
                QPushButton:pressed {
                    background-color: #e91e63;
                }
                QFrame {
                    border: 1px solid #e5a1b7;
                    margin: 20px 0;
                }
            """)

            self.layout = QVBoxLayout()

            # Criando o layout do ROI
            self.roi_section = QVBoxLayout()
            self.roi_form = QFormLayout()
            self.ii_input = QLineEdit(self)
            self.coa_input = QLineEdit(self)
            self.re_input = QLineEdit(self)
            self.dpr_input = QLineEdit(self)
            self.n_simulacoes_input = QLineEdit(self)

            self.roi_form.addRow('Investimento Inicial (R$):', self.ii_input)
            self.roi_form.addRow('Custo Operacional Anual (R$):', self.coa_input)
            self.roi_form.addRow('Receita Esperada (R$):', self.re_input)
            self.roi_form.addRow('Desvio Padrão da Receita (R$):', self.dpr_input)
            self.roi_form.addRow('Número de Simulações:', self.n_simulacoes_input)

            # Criando o botão para calcular o ROI
            self.roi_button = QPushButton('Calcular ROI', self)
            self.roi_button.clicked.connect(self.calcular_roi)

            self.roi_section.addLayout(self.roi_form)
            self.roi_section.addWidget(self.roi_button)

            # Criando uma linha divisória entre ROI e Overbooking
            self.separator = QFrame(self)
            self.separator.setFrameShape(QFrame.HLine)
            self.separator.setFrameShadow(QFrame.Sunken)

            # Criando o layout do Overbooking
            self.overbooking_section = QVBoxLayout()
            self.overbooking_form = QFormLayout()
            self.n_input = QLineEdit(self)
            self.p_input = QLineEdit(self)
            self.k_input = QLineEdit(self)

            self.overbooking_form.addRow('Número de Passagens Vendidas:', self.n_input)
            self.overbooking_form.addRow('Probabilidade de Comparecimento:', self.p_input)
```

```
        self.overbooking_form.addRow('Capacidade da Aeronave:', self.k_input)

        # Criando o botão para calcular o Overbooking
        self.overbooking_button = QPushButton('Calcular Overbooking', self)
        self.overbooking_button.clicked.connect(self.calcular_overbooking)

        self.overbooking_section.addLayout(self.overbooking_form)
        self.overbooking_section.addWidget(self.overbooking_button)

        # Layout para os botões de cálculo
        self.buttons_layout = QHBoxLayout()
        self.buttons_layout.addWidget(self.roi_button)
        self.buttons_layout.addWidget(self.overbooking_button)

        # Resultado do ROI e Overbooking
        self.result_label = QLabel('Resultados serão exibidos aqui', self)

        # Adicionando os layouts ao widget principal
        self.layout.addLayout(self.roi_section)
        self.layout.addWidget(self.separator)
        self.layout.addLayout(self.overbooking_section)
        self.layout.addWidget(self.result_label)

        self.setLayout(self.layout)

    def calcular_roi(self):
        try:
            # Pegar os valores inseridos pelo usuário
            ii = float(self.ii_input.text())
            coa = float(self.coa_input.text())
            re = float(self.re_input.text())
            dpr = float(self.dpr_input.text())
            n_simulacoes = int(self.n_simulacoes_input.text())
        except ValueError as e:
            self.result_label.setText(f"Erro: Por favor, insira valores válidos. Detalhes: {str(e)}")
            return

        # Semente para simulações reprodutíveis
        np.random.seed(42)

        # Simulação de receitas
        receitas_simuladas = np.random.normal(loc=re, scale=dpr, size=n_simulacoes)

        # Cálculo do Lucro e ROI
        lucros = receitas_simuladas - coa
        rois = (lucros / ii) * 100

        # Cálculo da probabilidade de receita abaixo de R$ 60.000
        prob_receita_abaixo_60000 = np.mean(receitas_simuladas < 60000) * 100

        # Exibindo resultados no layout
        result_text = f"Probabilidade de Receita < R$60.000: {prob_receita_abaixo_60000:.2f}%"
        self.result_label.setText(result_text)

        # Gerar gráfico da distribuição das receitas simuladas
        self.plot_receitas(receitas_simuladas)

        # Gerar gráfico do ROI
        self.plot_roi(rois)

    def plot_receitas(self, receitas_simuladas):
        try:
            plt.figure(figsize=(10,6))
            plt.hist(receitas_simuladas, bins=50, color='lightblue', edgecolor='black')
            plt.title('Distribuição das Receitas Simuladas')
            plt.xlabel('Receita (R$)')
            plt.ylabel('Frequência')
            plt.grid(True)
            plt.show()
        except Exception as e:
            print(f"Erro ao plotar gráfico de receitas: {str(e)}")

    def plot_roi(self, rois):
        try:
            plt.figure(figsize=(10,6))
            plt.hist(rois, bins=50, color='lightgreen', edgecolor='black')
            plt.title('Distribuição dos ROIs Simulados')
            plt.xlabel('ROI (%)')
            plt.ylabel('Frequência')
            plt.grid(True)
            plt.show()
        except Exception as e:
```