

Quantum Machine Learning

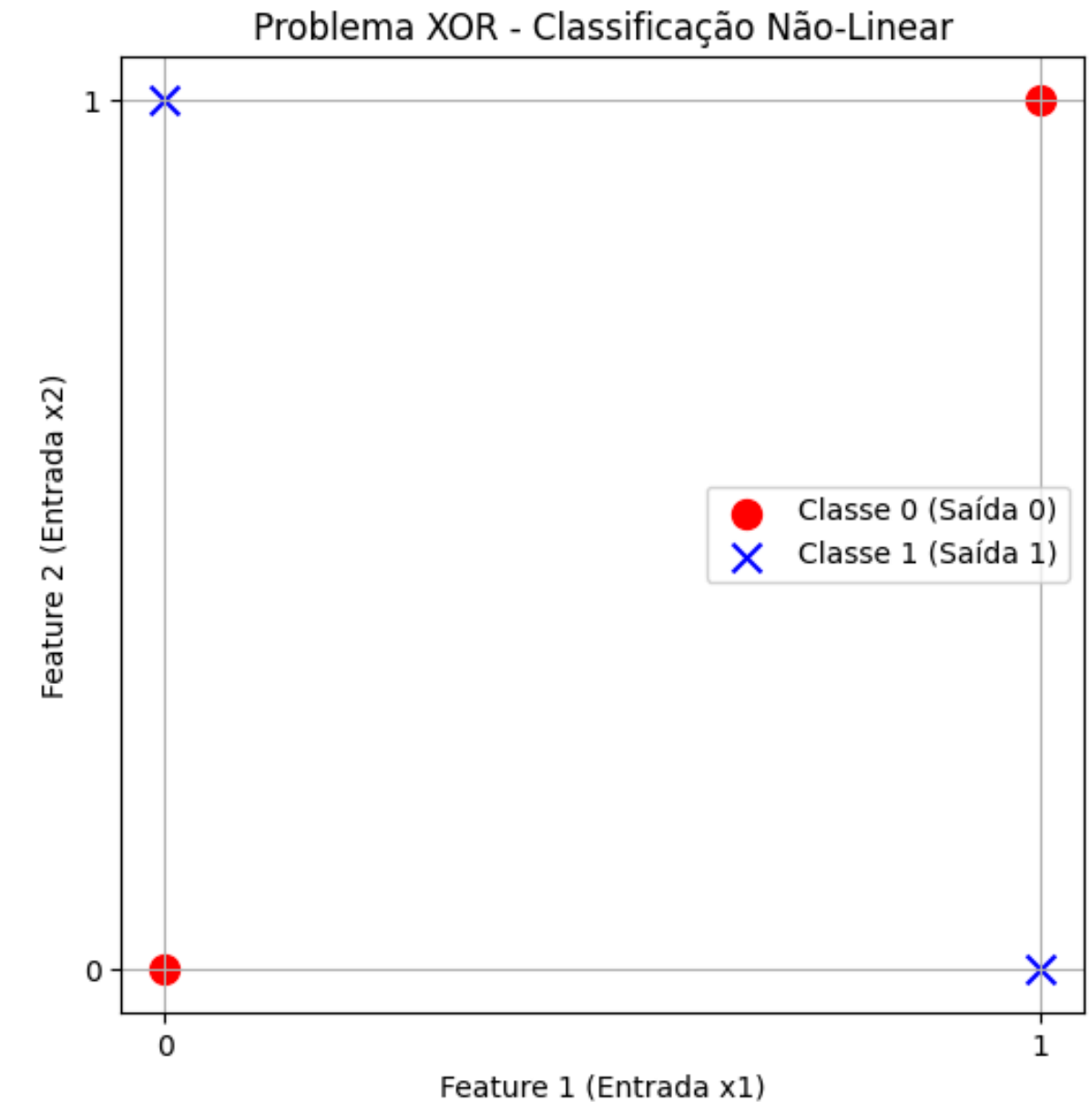
Computação Quântica - CIN0039

Gabriel Azevedo Amorim Vieira Belo
Gabriel Marques de Albuquerque
Henrique César Higino Holanda Cordeiro
Vinícius de Sousa Rodrigues

Fundamentos e Motivação

1. Por que usar Quantum Machine Learning?
 - a. Problemas não-lineares e Classificação complexa
 - b. Circuitos Quânticos Parametrizados
 - c. Espaço de Hilbert

2. Embedding de Dados clássicos:
 - a. Conceito e Funcionamento
 - b. Angle Embedding

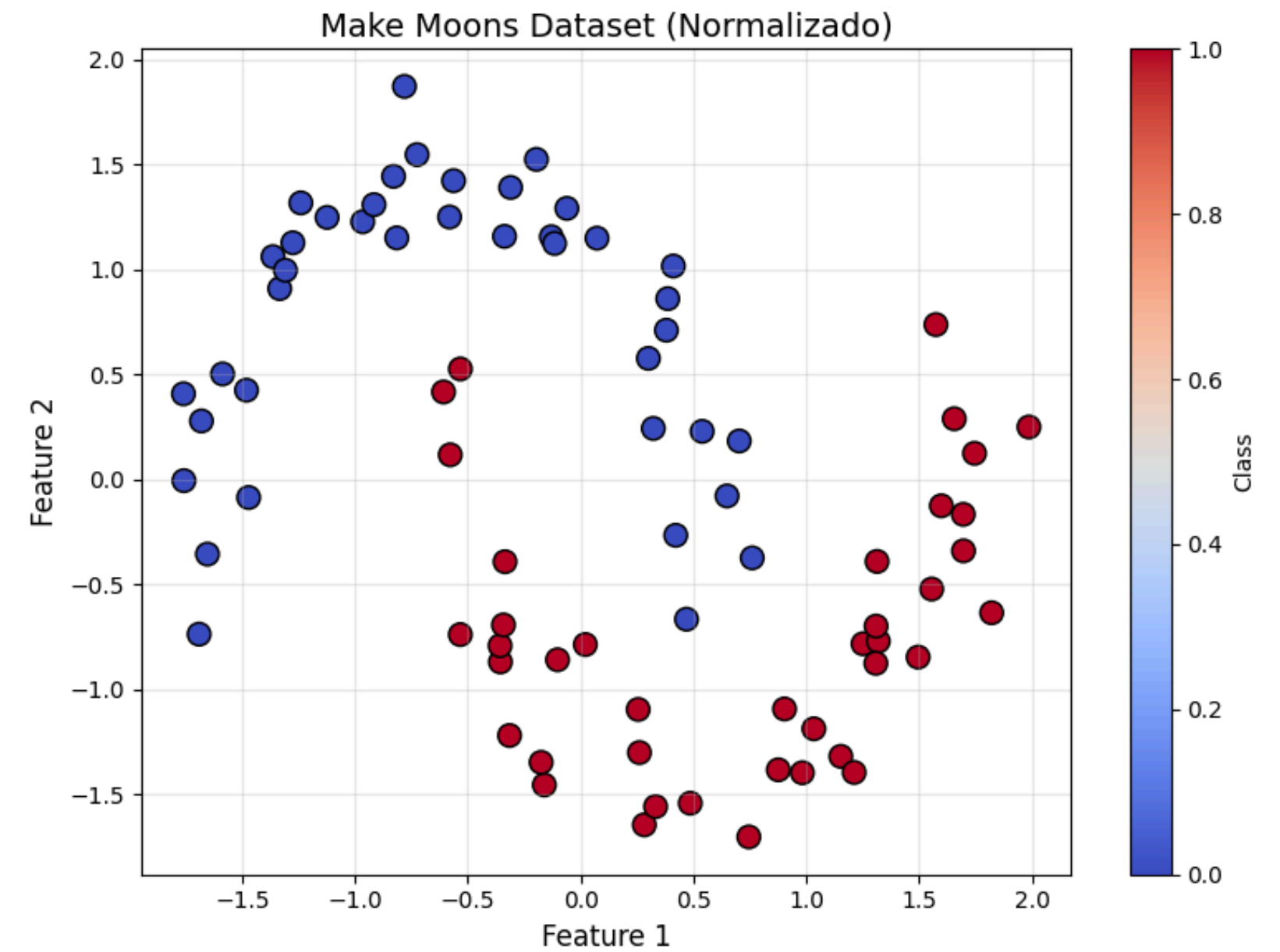
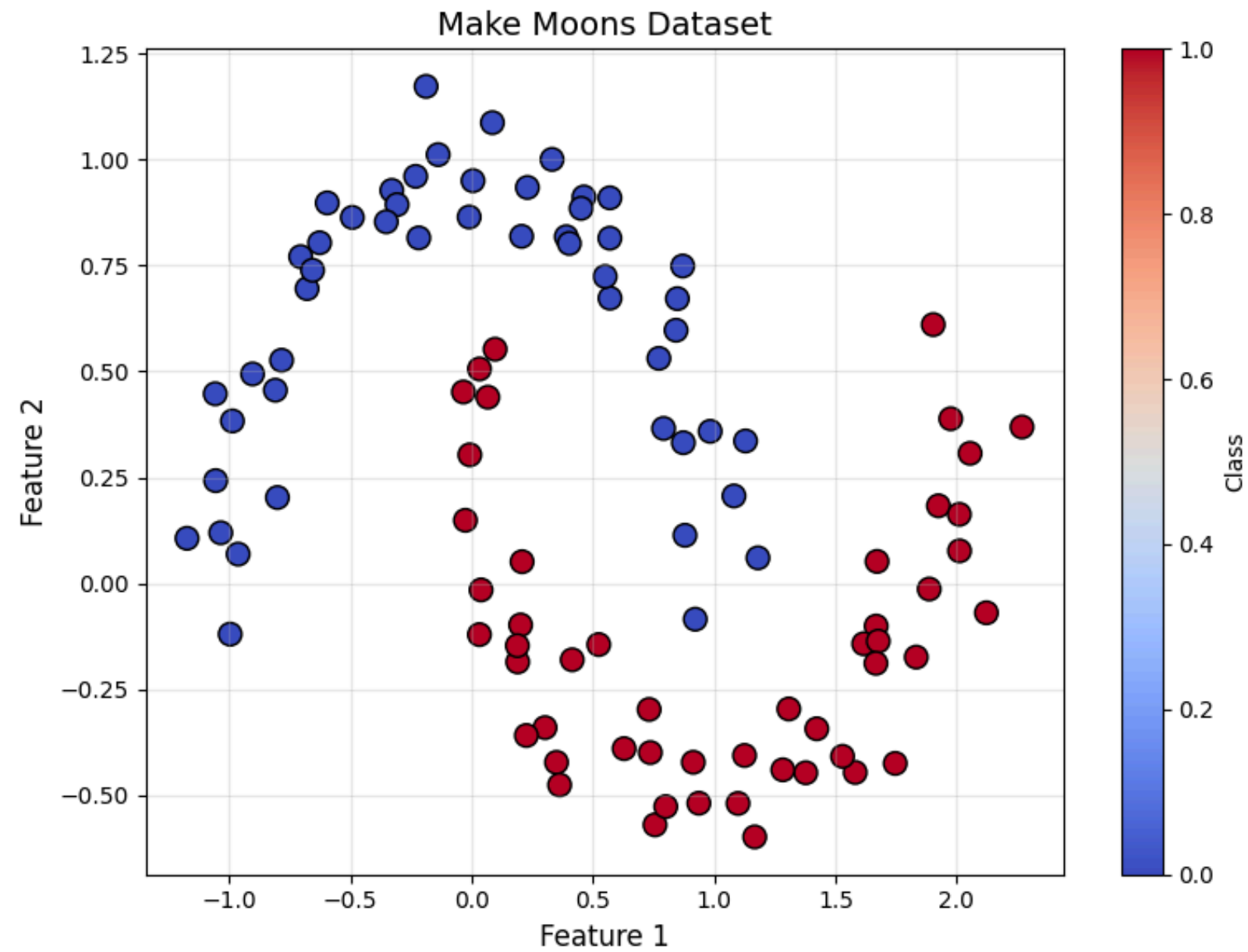


Fundamentos e Motivação

- 3. Modelos Variacionais (MQV):
 - a. Circuito Quântico Parametrizado
 - b. Algoritmo Híbrido
 - c. Analogia ao Clássico e Importância

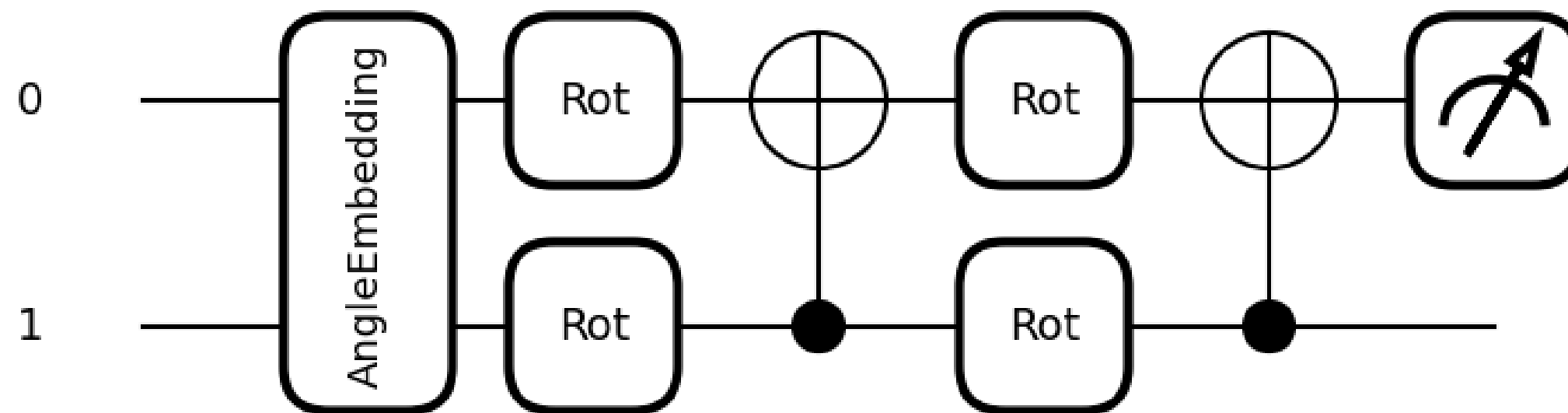
- 4. Características do Modelo Quântico:
 - a. Parâmetros Ajustáveis
 - b. Quantum Gradients: Parameter Shift-Rule
 - c. Extra: Dataset Make Moons

DATASET



EMBEDDING E ARQUITETURA

```
def Angle_embedding(x, n_qubits):  
  
    qml.AngleEmbedding(x, wires=range(n_qubits), rotation='Y')  
    return qml.state()
```



```
return qml.expval(qml.PauliZ(0))
```

TREINAMENTO: CONFIGURAÇÃO

1. Criação do circuito (QNode)

```
N_QUBITS = 2  
N_LAYERS = 2  
Q_CIRCUIT_GD = create_variational_circuit(N_QUBITS, n_layers=N_LAYERS, diff_method='parameter-shift')
```

2. Hiperparâmetros

```
n_epochs_gd = 60  
learning_rate = 0.1
```

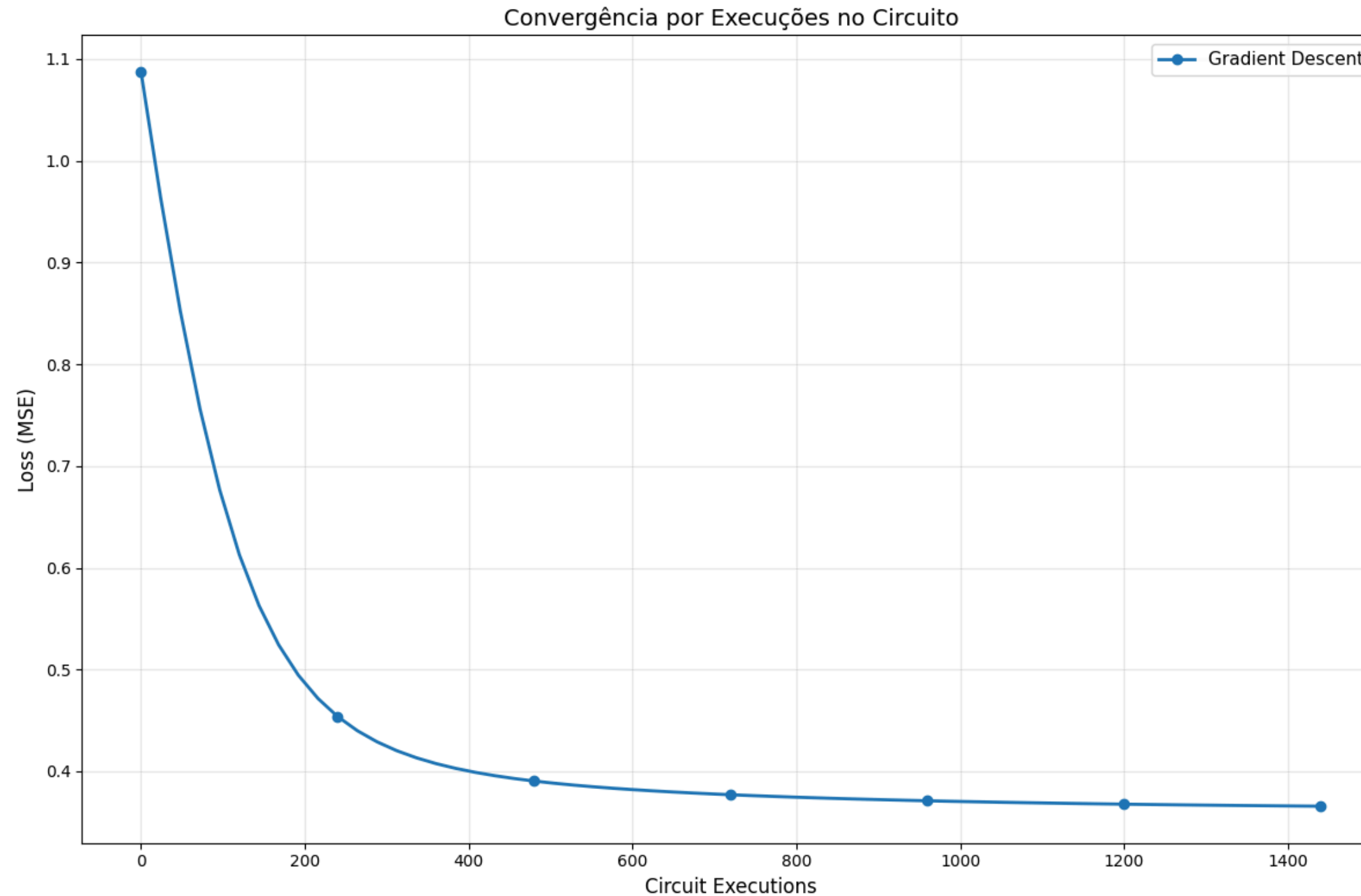
3. Função de Custo Usada no Otimizador

```
cost_fn_gd = create_loss_function(Q_CIRCUIT_GD, X_train, y_train_quantum)
```

4. Inicialização de Parâmetros Treináveis e Otimizador

```
params_init = pnp.random.random((N_LAYERS, N_QUBITS, 3), requires_grad=True)  
opt_gd_bp = qml.GradientDescentOptimizer(stepsize=learning_rate)
```

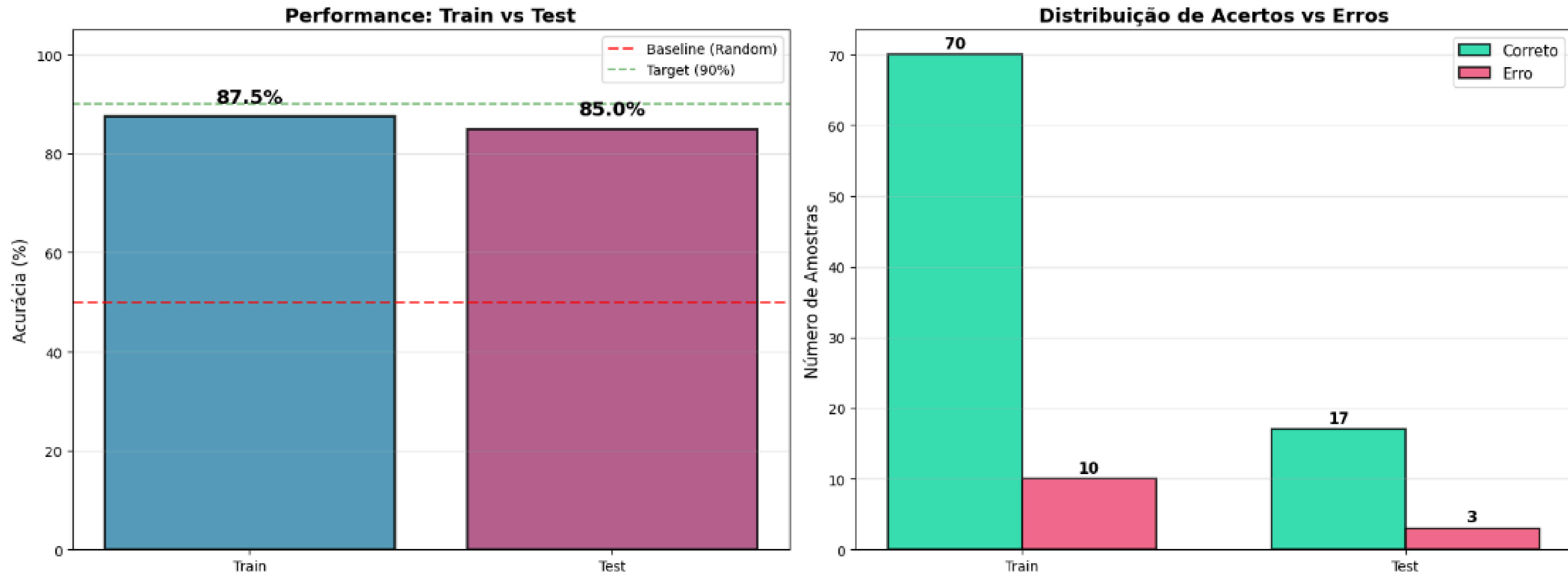
TREINAMENTO: LOOP e RESULTADO



```
loss_history_gd, exec_history_gd, params_gd = train_model(  
    optimizer=opt_gd_bp,  
    cost_function=cost_fn_gd,  
    init_params=params_init * 1.0, # Cópia dos parâmetros  
    num_steps=n_epochs_gd,  
    print_interval=10,  
    execs_per_step=execs_per_step_gd_ps # Parameter-shift: 2*num_params  
)
```

Resultados e análises - Circuit GD

Critério de desempenho: Acurácia

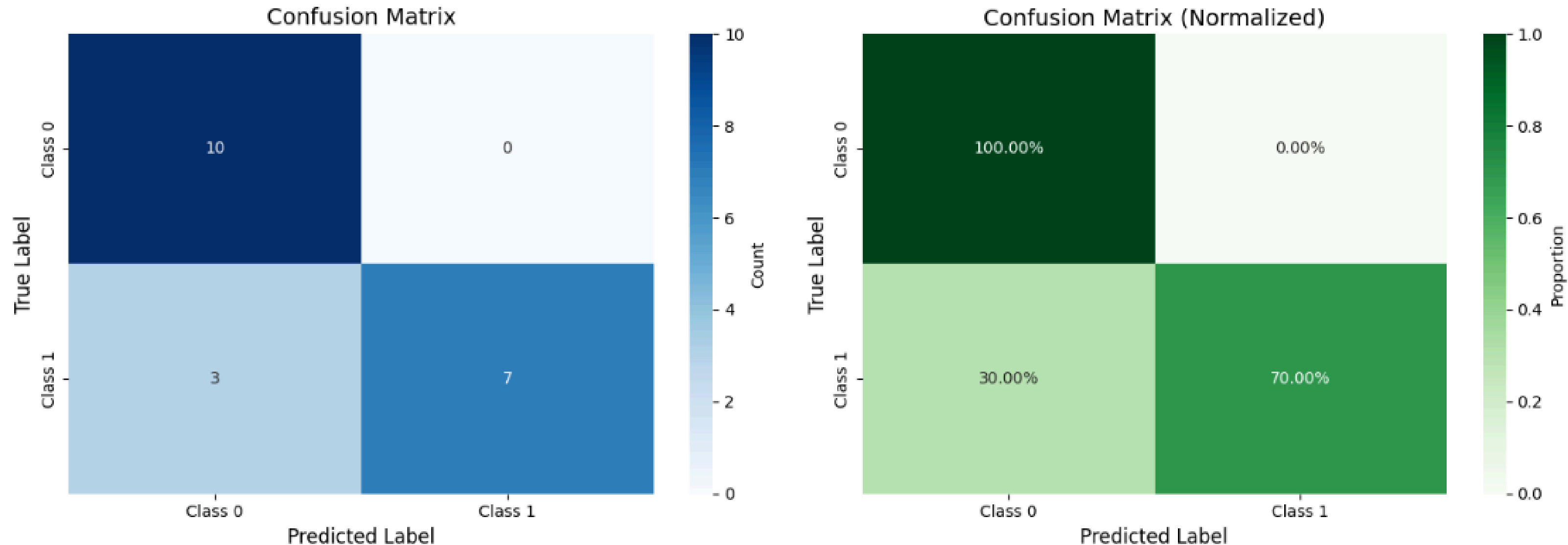


Insights:

- Alta generalização e baixo overfitting
 - Aprendizado de padrões reais em vez de apenas memorizar os dados de treino
- Proximidade da meta (target)

Resultados e análises - Circuit GD

Critério de desempenho: Matriz de confusão



Insights:

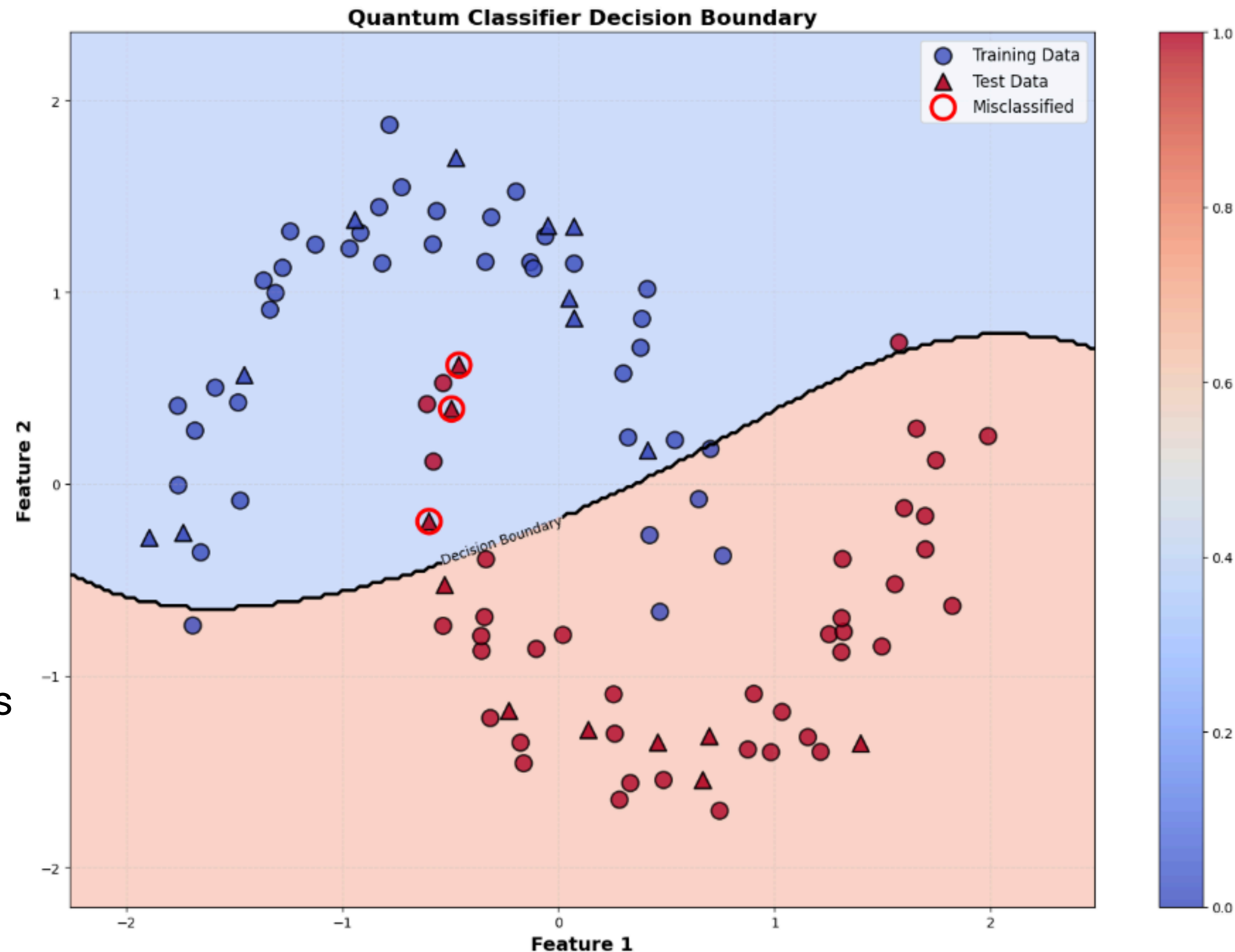
- Recall de 100% na Classe 0
- Zero Falsos Positivos para a Classe 1
- Erro de falso negativo (30%) para Classe 1
- Sensibilidade (recall) da classe 1 (70%)

Resultados e análises - Circuit GD

Critério de desempenho: Fronteira de Decisão

Insights:

- Natureza não-linear da fronteira
 - Consegue capturar relações não-lineares entre a "Feature 1" e a "Feature 2" sem a necessidade de kernels clássicos complexos
- A transição entre as zonas azul (Classe 0) e vermelha (Classe 1) é bem definida, indicando uma separação clara na maior parte do domínio
- Os erros de classificação não estão espalhados aleatoriamente, mas concentrados em uma região específica
- O modelo lida bem com os dados de teste (triângulos), seguindo o mesmo padrão dos dados de treino (círculos)



Resultados e análises - Circuit GD

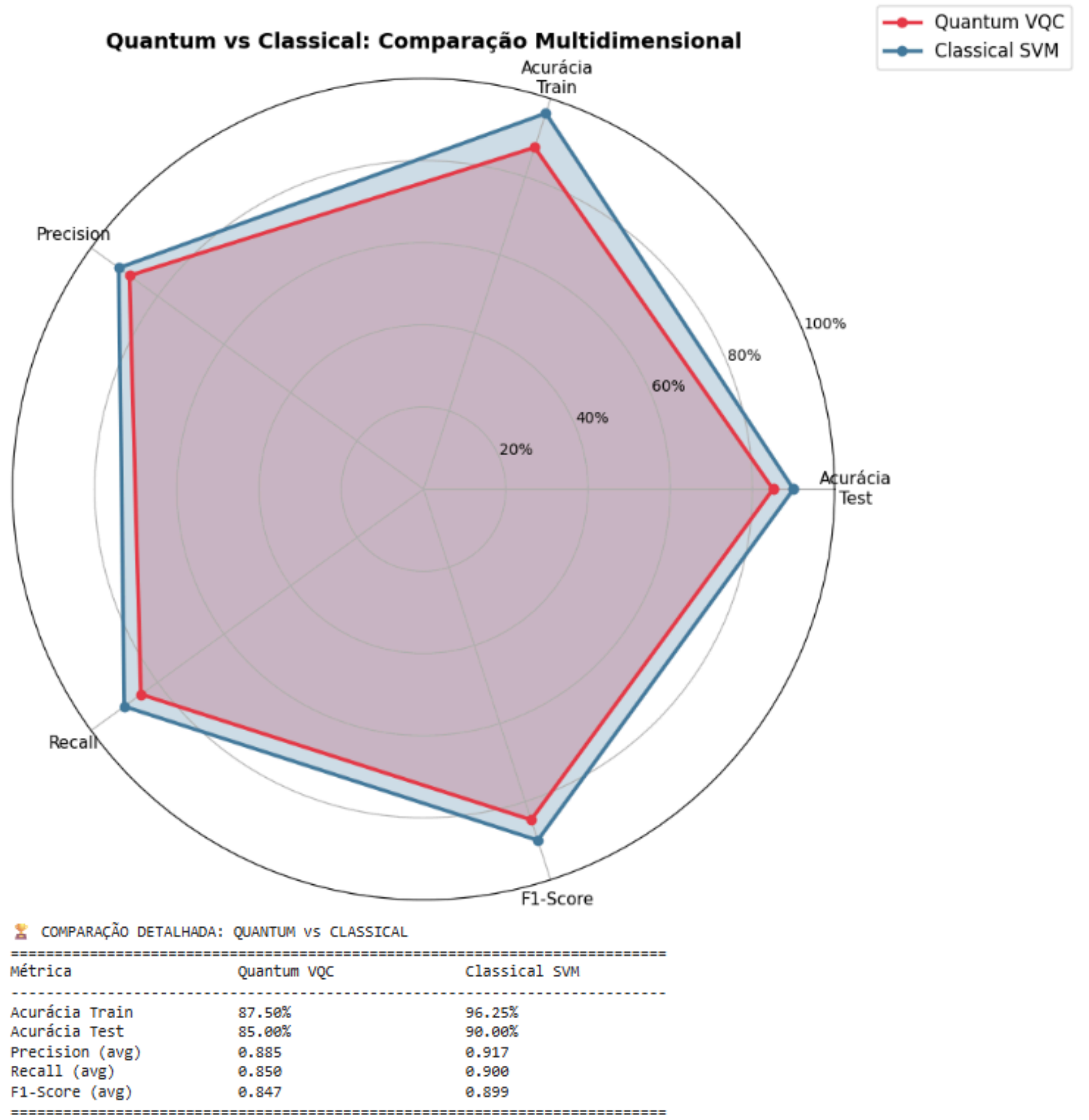
Critério de desempenho: Acurácia

Insights:

- Superioridade clássica em performance bruta
 - Acurácia e F1-Score um pouco superior
 - Harmonia ligeiramente melhor entre precisão e sensibilidade
- Estabilidade e Generalização (overfitting)
 - O modelo quântico demonstra uma maior estabilidade
 - SVM: Cai de 96,25% (treino) para 90,00% (teste)
 - GD: Cai de 87,50% (treino) para 85,00% (teste)

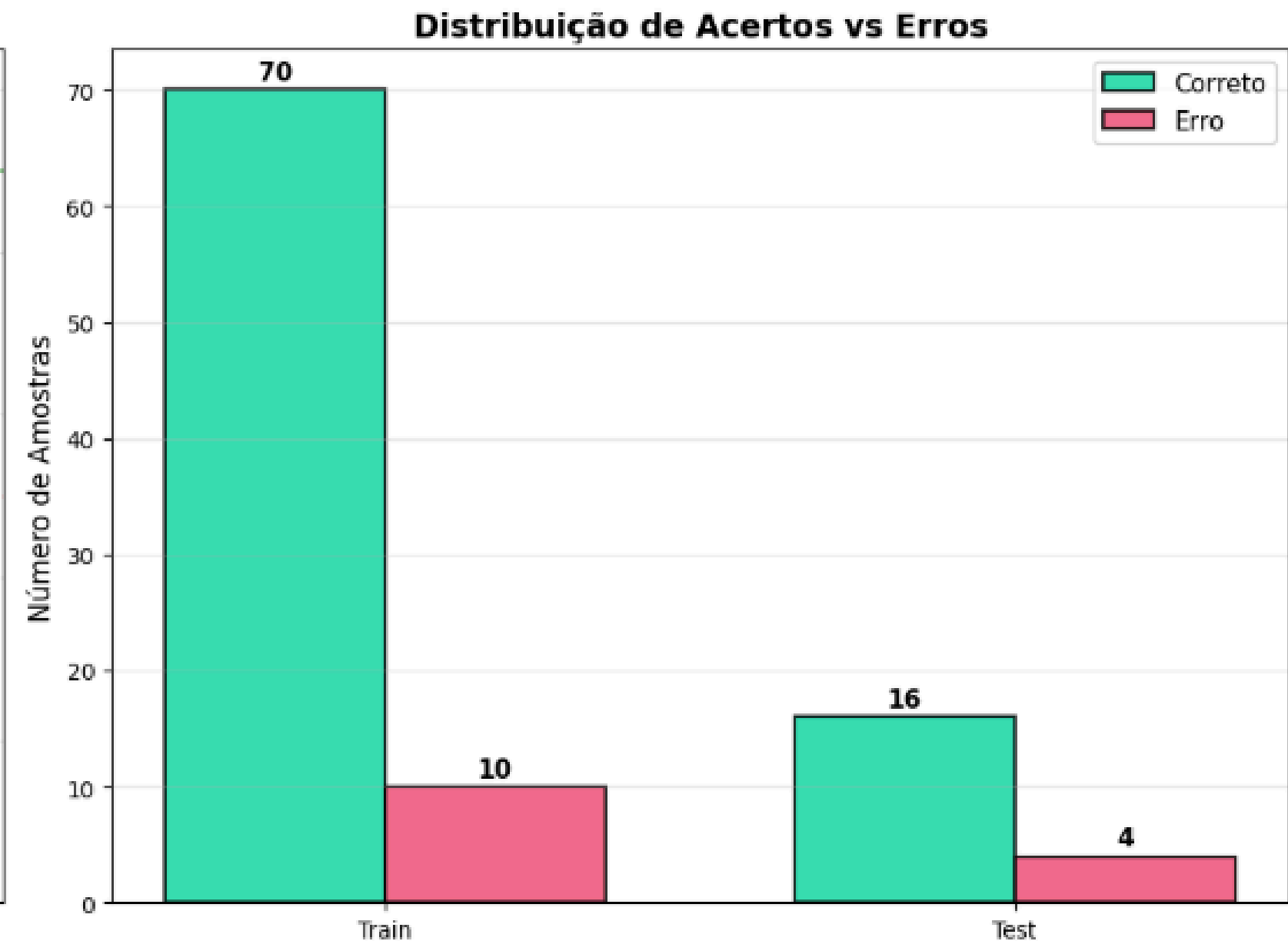
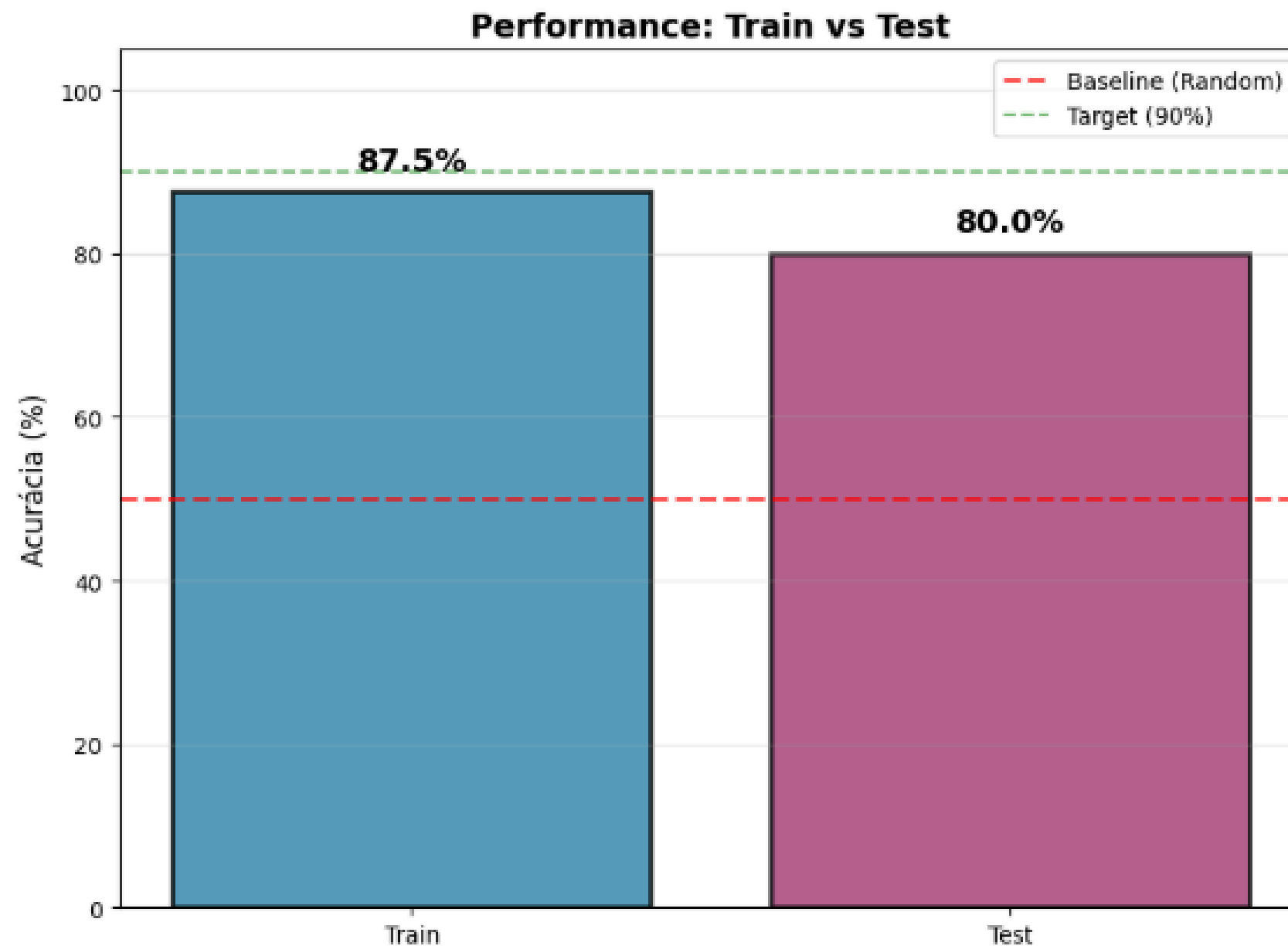
Conclusão:

- O SVM Clássico é a escolha mais eficiente para predições imediatas devido à sua maior acurácia.
- Quantum GD mostra-se promissor por sua resiliência ao overfitting



Resultados e análises - Circuit SPSA

Critério de desempenho: Acurácia

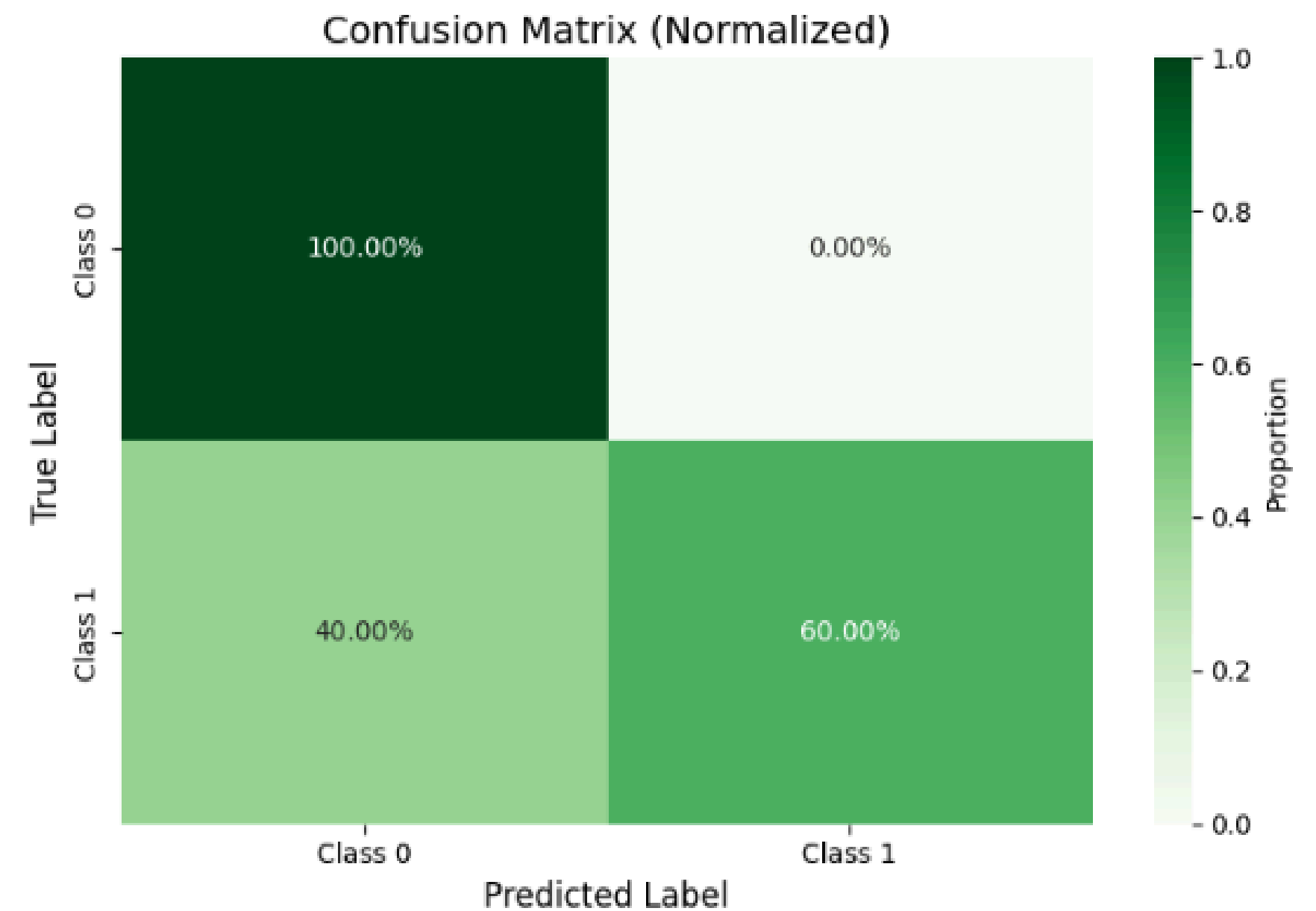
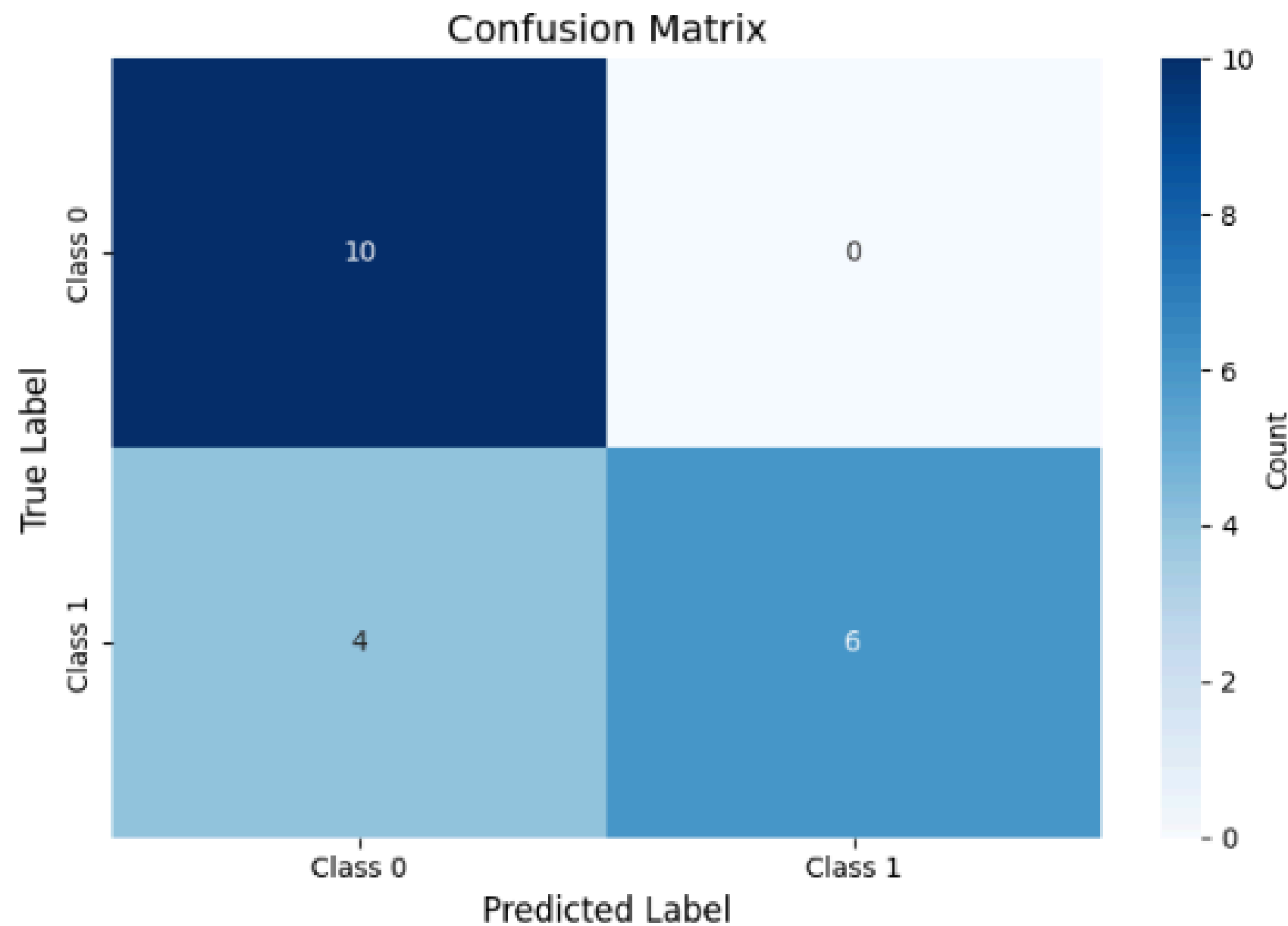


Insights:

- Alta generalização e relativamente baixo overfitting
 - Circuit GD desempenhou melhor
- Proximidade da meta (target)

Resultados e análises - Circuit SPSA

Critério de desempenho: Matriz de confusão



Insights:

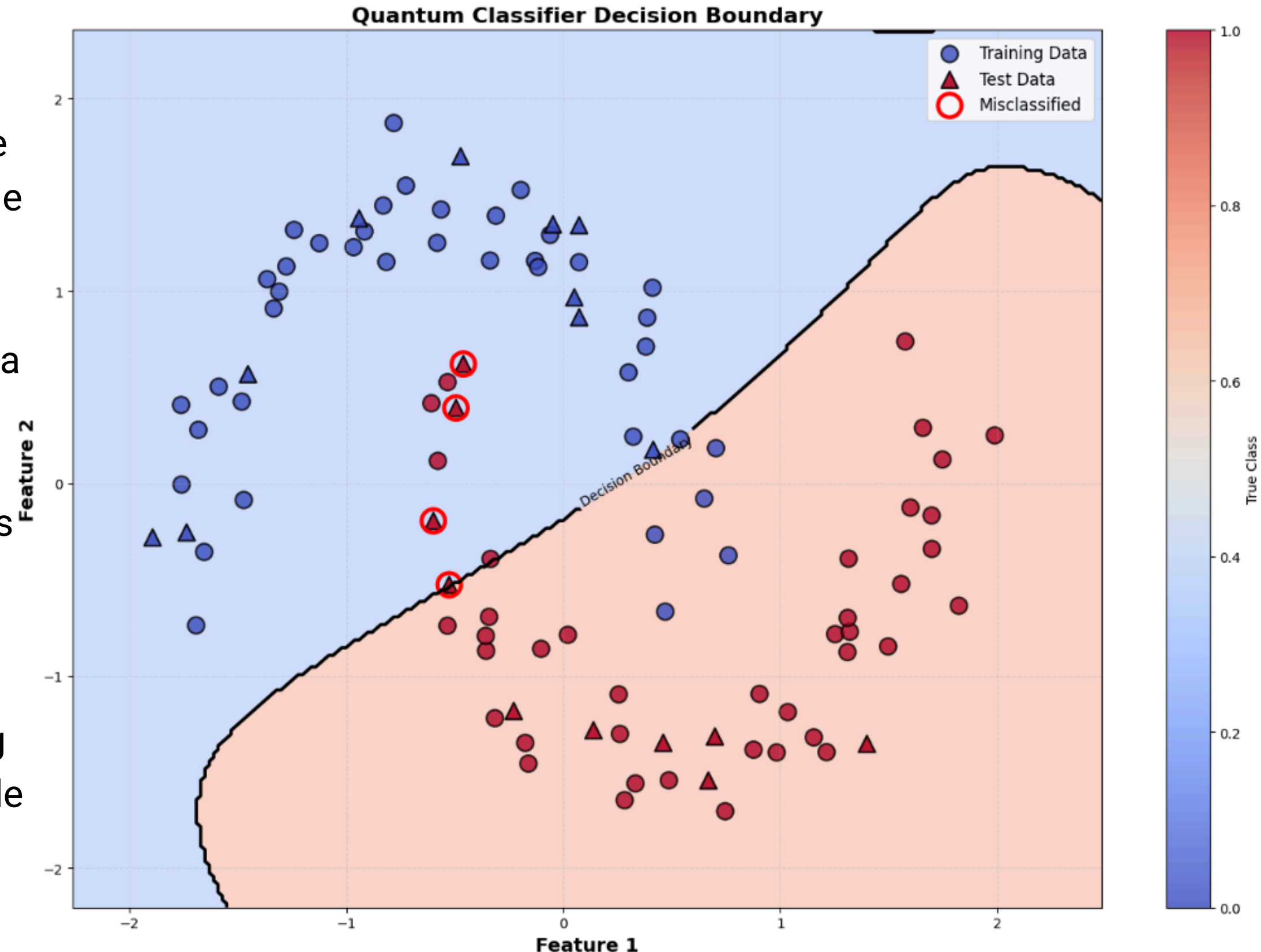
- Recall de 100% na Classe 0
- Zero Falsos Positivos para a Classe 1
- Erro de falso negativo (40%) para Classe 1
- Sensibilidade (recall) da classe 1 (60%)

Resultados e análises - Circuit SPSA

Critério de desempenho: Fronteira de Decisão

Insights:

- Natureza não-linear da fronteira
 - Consegue capturar relações não-lineares entre a "Feature 1" e a "Feature 2" sem a necessidade de kernels clássicos complexos
- A transição entre as zonas azul (Classe 0) e vermelha (Classe 1) é bem definida, indicando uma separação clara na maior parte do domínio
- O modelo tem dificuldade em uma "zona de sobreposição" específica. Esses erros são comuns em otimizações SPSA, que podem convergir para um mínimo local ou ser limitados pelo ruído inerente ao processo estocástico.
- O modelo não parece estar sofrendo de overfitting significativo, mas sim de uma leve incapacidade de capturar a fronteira exata naquela zona central densa



Resultados e análises - Circuit SPSA

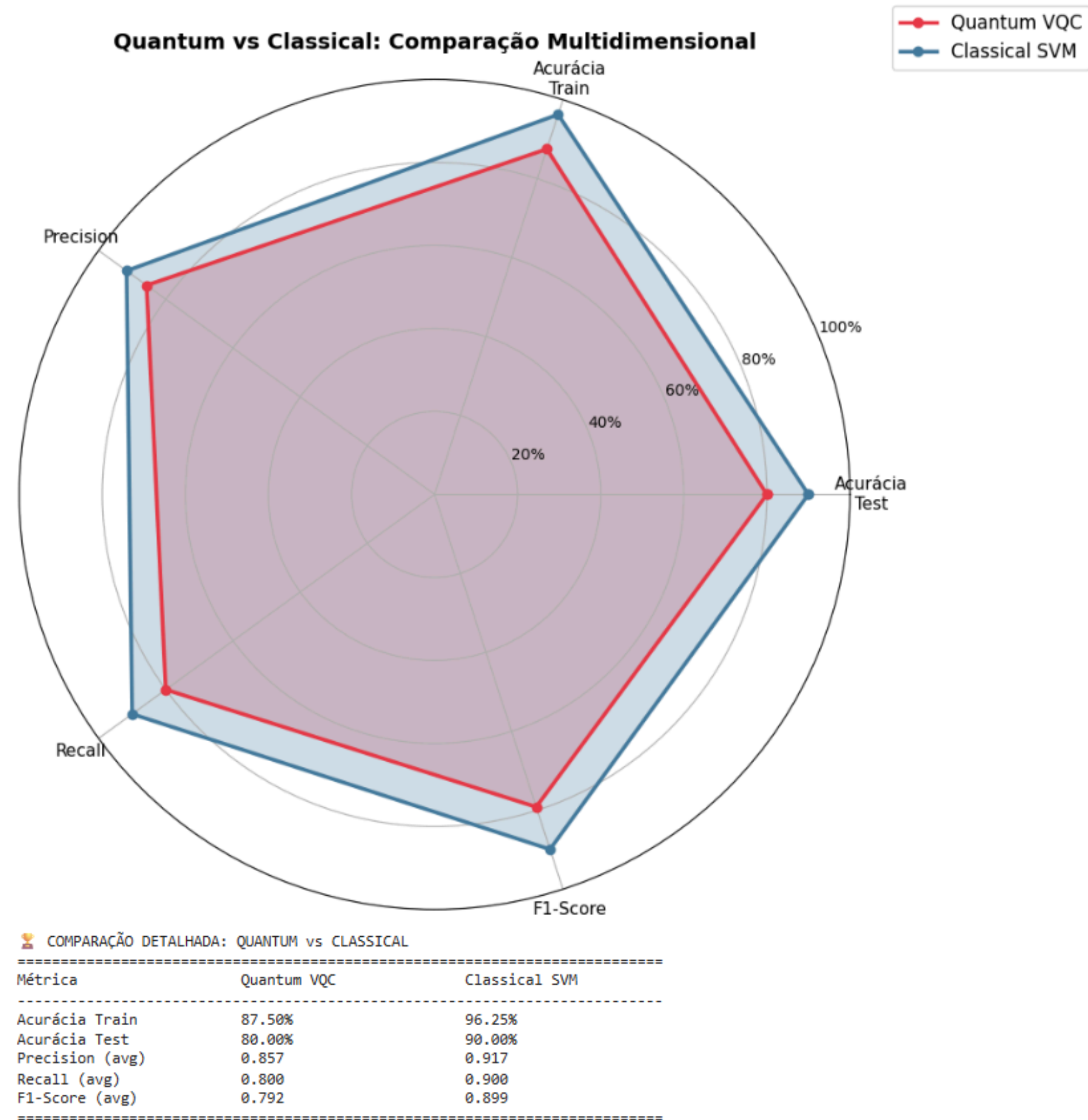
Critério de desempenho: Acurácia

Insights:

- Superioridade clássica em performance bruta
 - Acurácia e F1-Score um pouco superior
 - Harmonia ligeiramente melhor entre precisão e sensibilidade
- Generalização e overfitting
 - O modelo quântico está generalizando quase tão bem quanto o clássico, mas o SVM ainda continua sendo superior nesse aspecto também

Conclusão:

- O SVM Clássico é a escolha mais eficiente para predições imediatas devido à sua maior acurácia.
- Quantum SPSA mostra-se promissor por sua resiliência ao overfitting



Conclusões

O que funcionou:

- **Prova de conceito bem-sucedida**
 - Conseguiu aprender padrão não-linear do make_moons
 - Acurácia de 80-90% demonstra capacidade de aprendizado
- **Framework matematicamente sólido**
 - PennyLane fornece ferramentas robustas para QML
- **Aprendizado Técnico**
 - Compreensão profunda de circuitos variacionais
 - Experiência com otimização quântica
 - Conhecimento de limitações práticas

Limitações:

- **NÃO há vantagem quântica demonstrada**
 - Make_moons é um problema clássico simples
 - Overhead quântico é alto
 - Custo e múltiplas execuções
 - Dataset é pequeno
- **Escalabilidade Fundamental**
 - Limitado a problemas com poucos features (<10)
 - Hardware atual insuficiente para aplicações reais
 - Embedding é o gargalo principal
 - Sensibilidade a ruído

Valor principal do QML: Exploração científica!

Obrigado!