Relatório: Modelos SVM e Decision Tree

João Victor de Souza Albuquerque

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará - Campus Maracanaú (IFCE)
Av.Parque Central, 1315 - Distrito Industrial I, Maracanaú - CE, 61939-140

Resumo. *Este relatório documenta o experimento da lista 5 de reconhecimento de padrões.*

1. Base Teórica

1.1. SVM

É um modelo que na sua formulação original foi pensado para encontrar uma reta de separação dos dados perfeita, assim não aceitando que ocorra erros de predição com os dados de treino, produzindo assim erro zero para os dados de treino, ele chegava ao vetor de W desejado e Bias através das formulas:

$$\mathbf{w}_* = \sum_{i \in S} \alpha_i x_i y_i, \quad b_* = \frac{1}{N_S} \sum_{i \in S} \left(y_i - \mathbf{w}_*^\top x_i \right).$$

A predição desse modelo é feita através das equações:

O segundo modelo, Naive Bayes Gaussiano, foi modelo seguindo a seguinte função de matemática:

$$\begin{aligned} \hat{y}_j &= \text{sign}\left(w_*^\top x_j + b_* \right) \\ \hat{y}_j &= \text{sign}\left(\left(\sum_{i \in S} \alpha_i y_i x_i^\top \right) x_j + b_* \right) \\ &= \text{sign}\left(b_* + \sum_{i \in S} \alpha_i y_i x_i^\top x_j \right). \end{aligned}$$

Como todos a reta de separação dos dados tem que está perfeitamente ajustada para os dados de treino, o modelo está muito suscetível ao overfitting. Para minimizar esse problema, o modelo foi ajustado para que ele tenha uma margem de erro aceitável para os dados de treino, chamada de soft margin. Outra questão que surgiu seria como modificar o modelo para que ele seja capaz de lidar com problemas não-lineares, a solução encontrada foi adicionar uma função de kernel que adiciona não linearidade aos dados de treino, permitindo o modelo a lidar com problemas não-lineares. Aplicando as modificações necessárias ao modelo obtemos as novas equações:

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j x_i^\top x_j$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \forall i, \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

$$\hat{y}_j = \text{sign}\left(b_* + \sum_{i \in S_*} \alpha_i y_i x_i^\top x_j \right).$$

1.2. Decision Tree

São modelos de simples compreensão, sendo estruturado a partir de uma raiz que contem um limiar em relação a uma feature, e contem duas saídas, um para quando a amostra tem seu valor na feature maior que o limiar e uma saída para quando não tem, essas saídas levam a folhas que tem features alvo próprias e limiares próprios e com duas saídas como a raiz, e assim a arvore vai crescendo ate que se alcance uma altura que tenha dividido todas as classes das amostras em folhas únicas, ou seja ate que tenha folhas compostas por elementos de uma única classe. Para se descobrir o melhor limiar com a melhor feature para cada folha e raiz se testa todas as combinações possíveis de feature e limiar na separação dos dados e ver qual combinação tem o menor valor no índice gini, que se refere a pureza dos dados. Como o modelo busca parar quando se alcança folhas puras, isso pode ocasionar overfit, para evitar isso pode ser adicionado um nível de gini, aceitável, uma altura máxima para o modelo ou numero mínimo de amostras em folhas.

1.3. Normalização

As vezes os dados que queremos que o modelo se adapte estão em escalas muito diferentes o que pode acarretar em dificuldade do modelo em se adaptar a eles, por isso é recomendado deixar os dados numa escala comum, esse processo se chama de normalização, que pode ser colocar os dados em uma escala de [0,1] ou [-1,1] ou mesmo forção uma média e variância. Esse método de preprocessamento nos da maior controle dos valores dos parâmetros e facilita o ajuste dos hiperparâmetros

No trabalho foi utilizado a normalização dos dados na escala de [0,1], essa regularização seguir os seguintes passos:

Primeiro se calcula o valor máximo do vetor Y e matriz X, coluna a coluna:

$$y_{max} = \max(y), \quad [x_{max}]_d = \max([X]_d), \forall d.$$

Depois disso, calcule o menor valor do vetor Y e da matiz X, coluna a coluna:

$$y_{min} = \min(y), \quad [x_{min}]_d = \min([X]_d), \forall d.$$

E por fim se aplica a formula a seguir para finalizar a regularização:

$$y \leftarrow \frac{y - y_{min}}{y_{max} - y_{min}}, \quad [X]_d \leftarrow \frac{[X]_d - [x_{min}]_d}{[x_{max}]_d - [x_{min}]_d}, \forall d.$$

2. Metodologia

2.1. Dados

Importei os dados contidos no bostonbin.csv para um dataframe do pandas.

2.2. Criação do modelo

Para a realização dessa lista foi necessário fazer a arvore de decisão do zero seguindo as instruções matemáticas e logicas contidas nos slides das aulas, todas as formulas foram adaptadas para código python utilizando a biblioteca do numpy como apoio para manipulação matemática.

2.3. Normalização

Para Implenetar a classe de normalização, preciso desenvolver 3 funções, a função fit(), normalize e desnormalize. A função fit salva os minemos e máximos do X e Y internamente na classe. A função de normalize pega os valores de máximos e minemos salvos na classe e utiliza na função de normalização já mostrada e que foi traduzida para código, e no final devolve os X e Y normalizados. A função desnormalize pega os valores mínimo e máximo salvos internamente na classe e as utiliza na inversa da função de normalização.

2.4. Questão 1

2.4.1. SVM

Primeiro treinei o modelo do sklearn utilizando hiperparâmetros aleatórios e fiz as estatísticas de acurácia, revocação, precisão e f1-score para que eu tenha valores bases, para comparar com os valores que serão obtidas em cada otimização, como também plotei a curva ROC e precision-recall para comparações futuras. A primeira otimização feita foi o grid search, os hiperparmetros passados foram C = [2^{-5} até 2^{15}] e gamma =[2^{-15} até 2^{3}], como kernel foi utilizado o rbf. O grid foi cofigurado para utilizar 10 folds na buscas e utilizar a acurácia como métrica de otimização. Após a finalização do grid search foi refeita a as metricas de acurácia, revocação, precisão e f1-score para o modelo ajustado e o plot da curva ROC e precision-recall. Após isso foi feita o mesmo processo de otimização utilizando o opturna, foram usados os mesmos intervalos e configurações do grid search.

2.4.2. Decision Tree

Após a montagem do modelo para seguir os padrões do sklearn, os passos a passos foram o mesmo do svm. Primeiro foi feito o modelo foi treinado com hiperparametros aleatórios para se ter uma base de comparação, foram obtidas as métricas acurácia, revocação, precisão e f1-score e plotei a curva ROC e precision-recall. Após isso, Utilizei o grid search com os hiperparemetros max_deep = [1 até 10] e min_samples = [1 até 5], configurei o grid search para usar 10 folds e a acurácia como métrica de otimização, com o modelo ajustado eu peguei as métricas acurácia, revocação, precisão e f1-score e plotei os graficos curva ROC e precision-recall. Em seguida, fiz o mesmo processo utilizando o optuna, utilizando os mesmo hiperparametros e configuração.

3. Resultado

3.1. Questão 1

3.1.1. SVM

Com o modelo não ajudado foram obtidas as seguintes métricas:

Métrica	Valor
Acurácia	0.8486842105263158
Revocação	0.8518518518518519
Precisão	0.8625
F1-score	0.8571428571428572

Table 1. Métricas do modelo svm não ajusatdo

Ao ajustar o modelo com grid search foi encontrado como melhores hiperparametros C = 32 e gamma = 0.125, utilizando esses hiperparametros no modelo obtive as seguintes métricas:

Métrica	Valor
Acurácia	0.868421052631579
Revocação	0.8395061728395061
Precisão	0.9066666666666666
F1-score	0.8717948717948718

Table 2. Métricas do modelo svm com grid-search

Ao ajustar o modelo com o Optuna foi encontrado como melhores hiperparametros C = 4184.261347935534 e gamma = 0.00023268431283865178, utilizando esses hiperparametros no modelo obtive as seguintes métricas:

Métrica	Valor
Acurácia	0.8355263157894737
Revocação	0.8395061728395061
Precisão	0.85
F1-score	0.84472049689441

Table 3. Métricas do modelo svm com Optuna

3.1.2. Decision Tree

Com o modelo não ajudado foram obtidas as seguintes métricas:

Ao ajustar o modelo com grid search foi encontrado como melhores hiperparametros max_deep = 1 e min_samples = 1, utilizando esses hiperparametros no modelo obtive as seguintes métricas:

Métrica	Valor
Acurácia	0.7960526315789473
Revocação	0.7530864197530864
Precisão	0.8472222222222222
F1-score	0.7973856209150327

Table 4. Métricas do modelo decision tree não ajusatdo

Métrica	Valor
Acurácia	0.8552631578947368
Revocação	0.9506172839506173
Precisão	0.8105263157894737
F1-score	0.8749999999999999

Table 5. Métricas do modelo modelo decision tree com grid-search

Ao ajustar o modelo com o Optuna foi encontrado como melhores hiperparametros max_deep = 10 e min_samples = 5, utilizando esses hiperparametros no modelo obtive as seguintes métricas:

Métrica	Valor
Acurácia	0.8157894736842105
Revocação	0.8024691358024691
Precisão	0.8441558441558441
F1-score	0.8227848101265822

Table 6. Métricas do modelo decision tree com Optuna

3.1.3. Graficos ROC

3.1.4. Graficos Precision-Recall

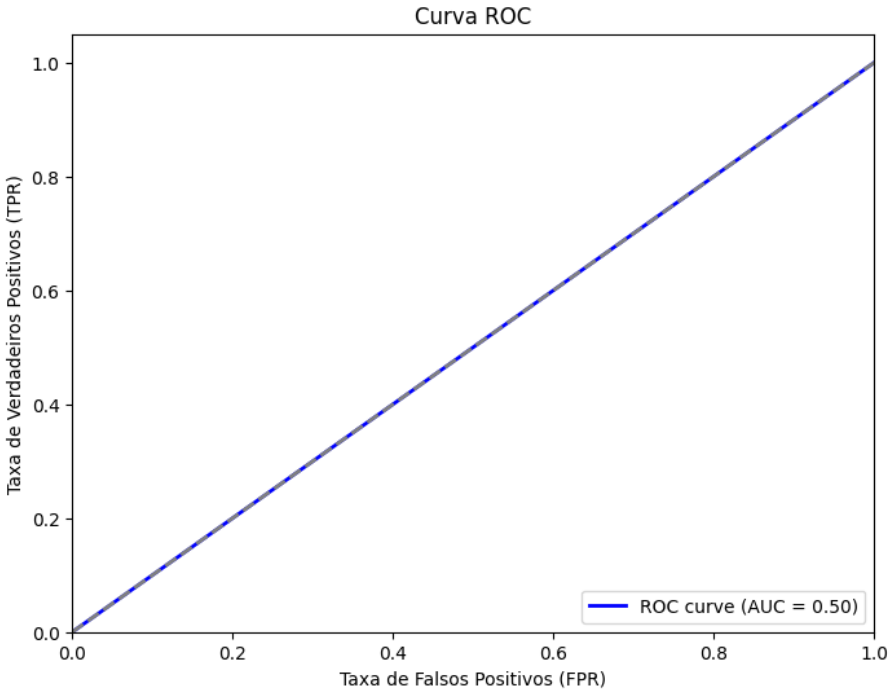

Figure 1. SVM não ajusatdo

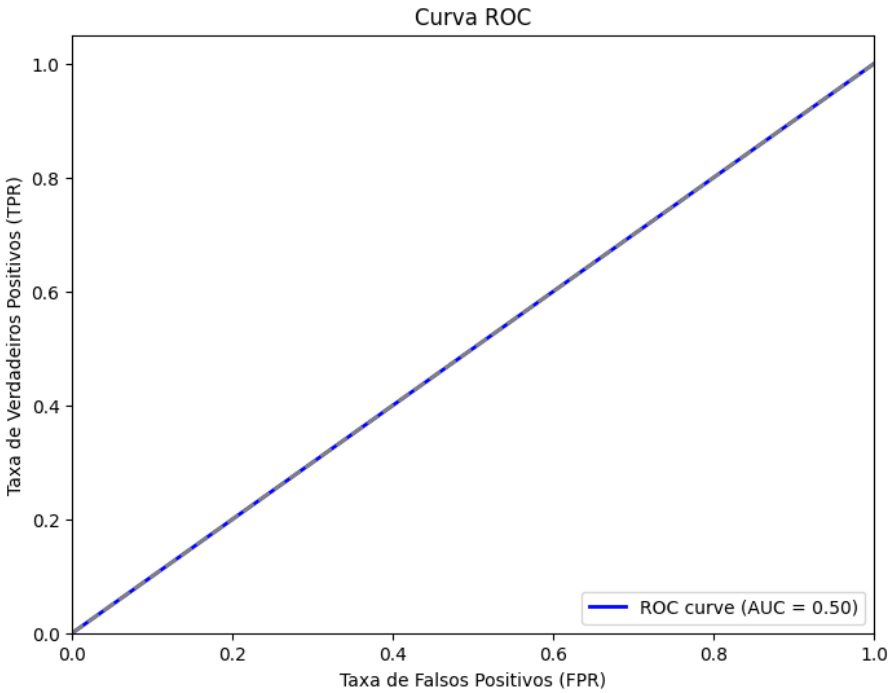

Figure 2. SVM com grid search

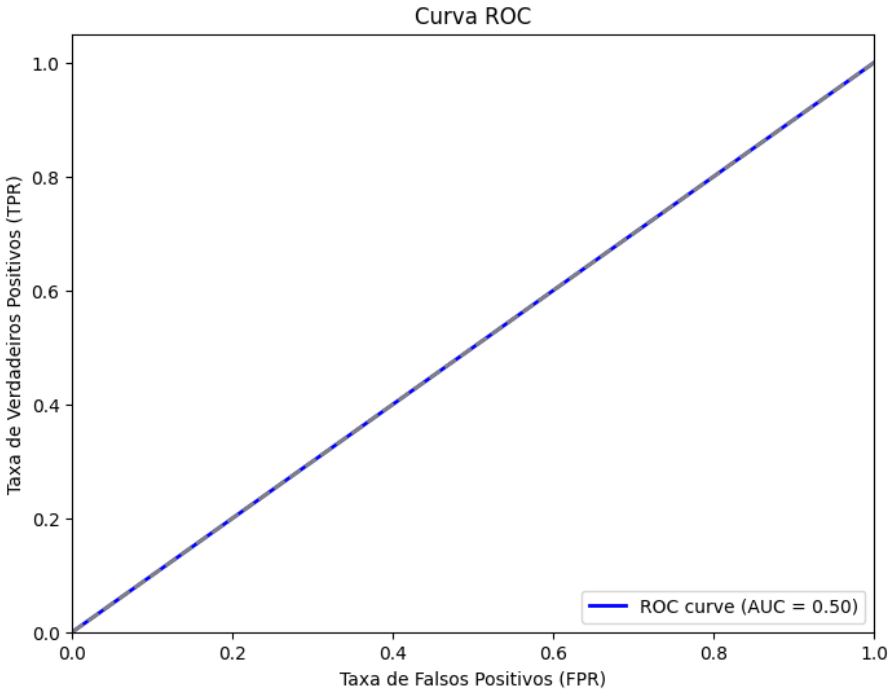

Figure 3. SVM com Optuna

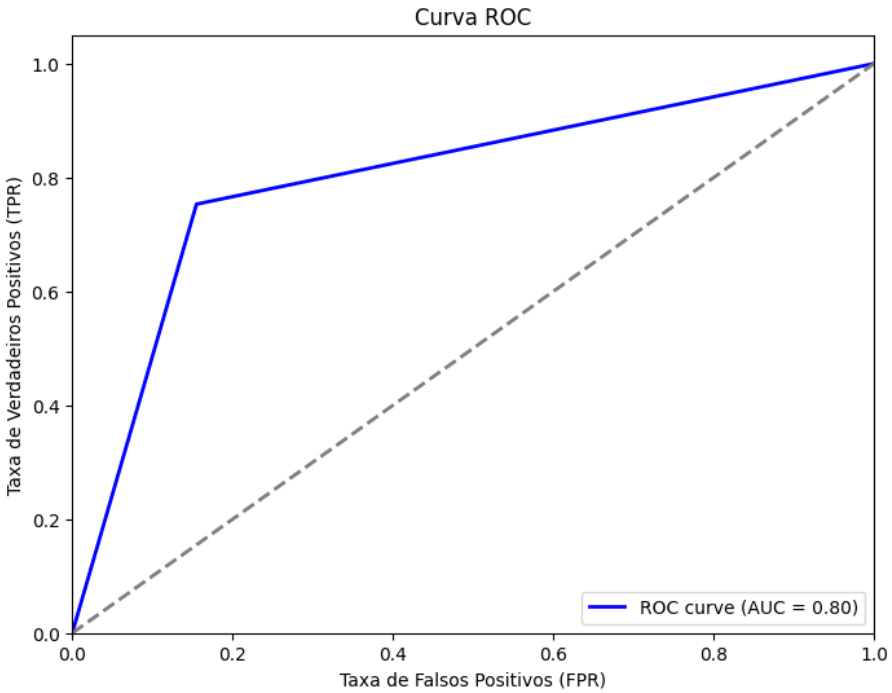

Figure 4. Decision Tree não ajustada

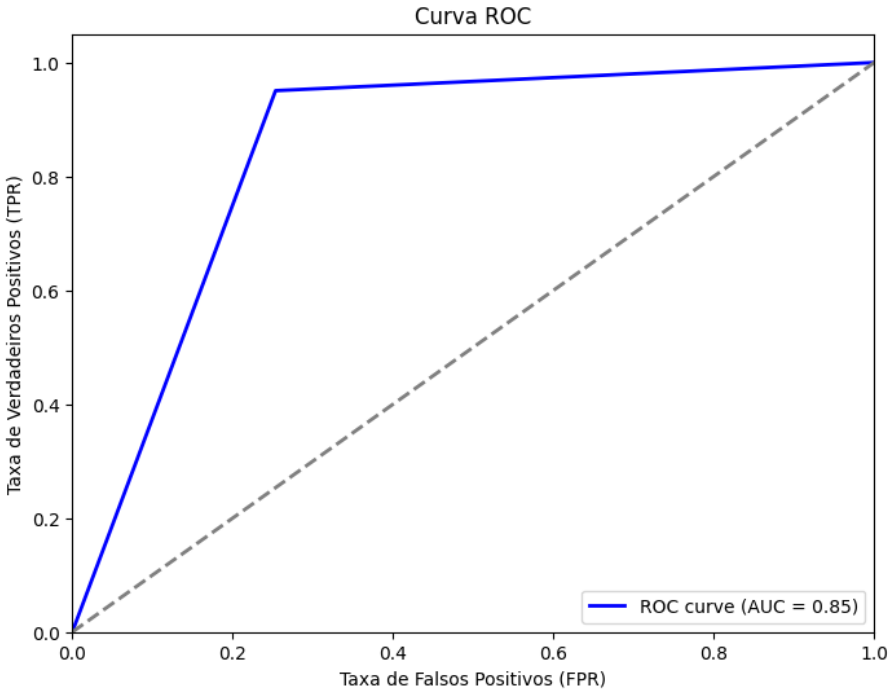

Figure 5. Decision Tree com grid search

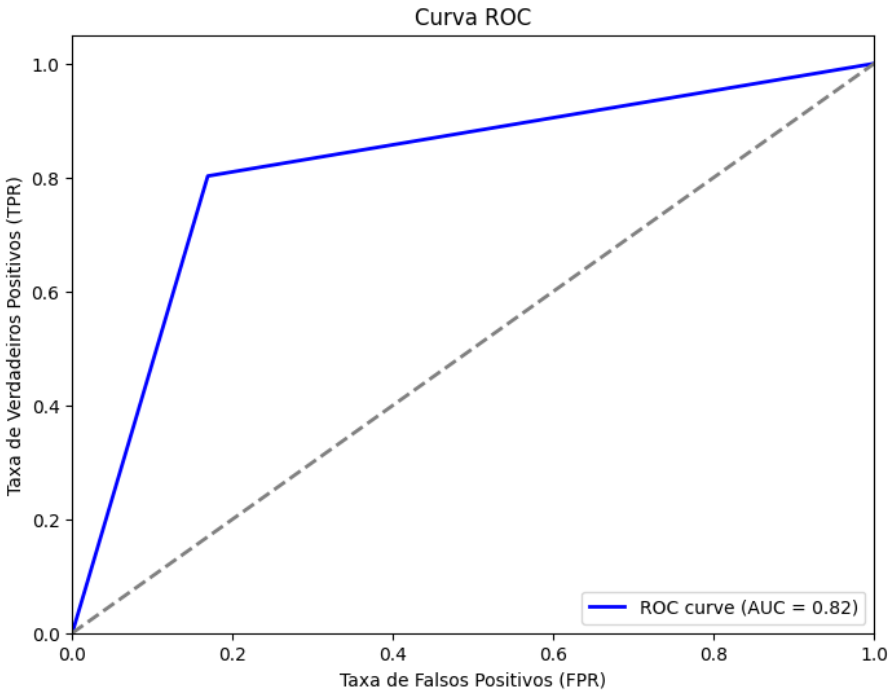

Figure 6. Decision Tree com Optuna

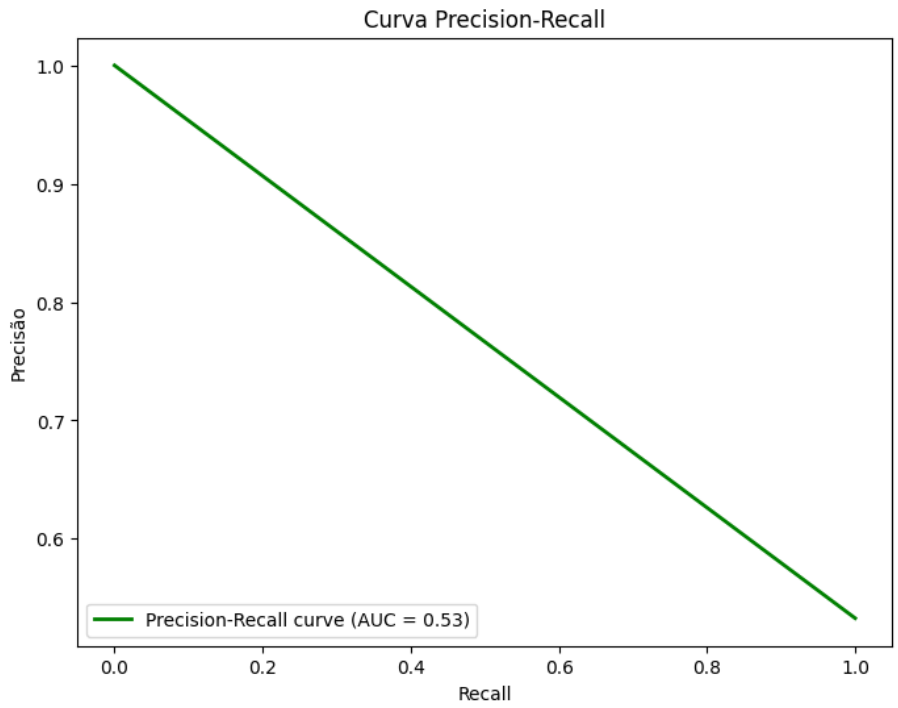

Figure 7. SVM não ajusatdo

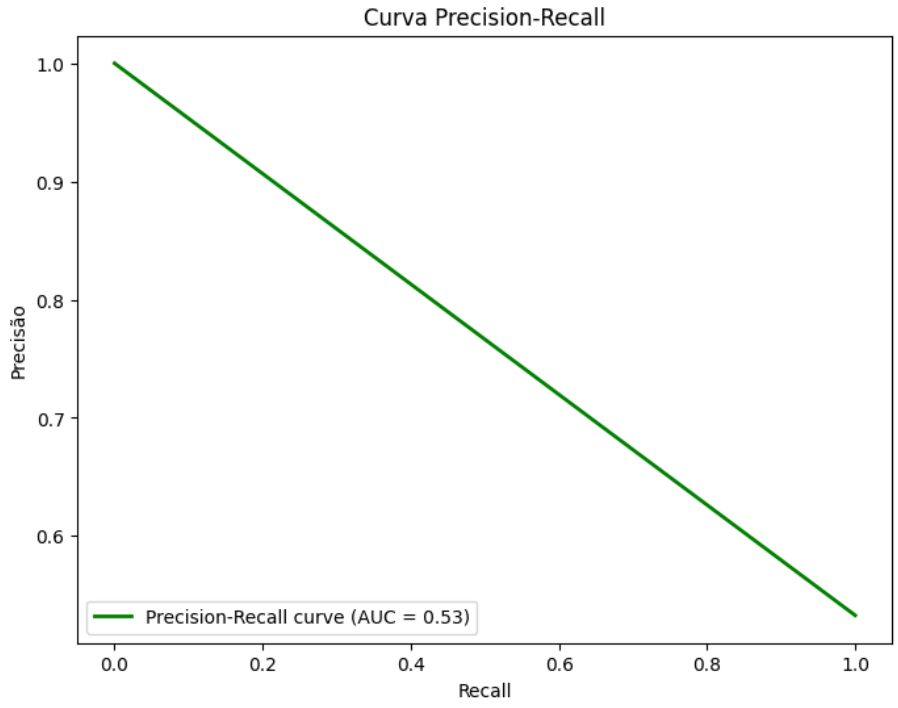

Figure 8. SVM com grid search

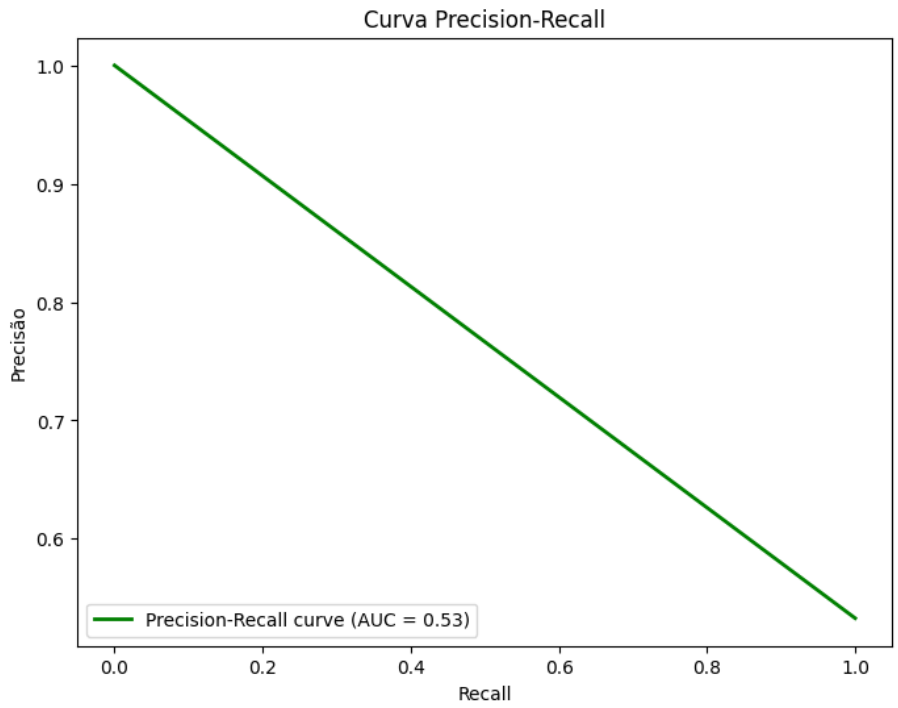

Figure 9. SVM com Optuna

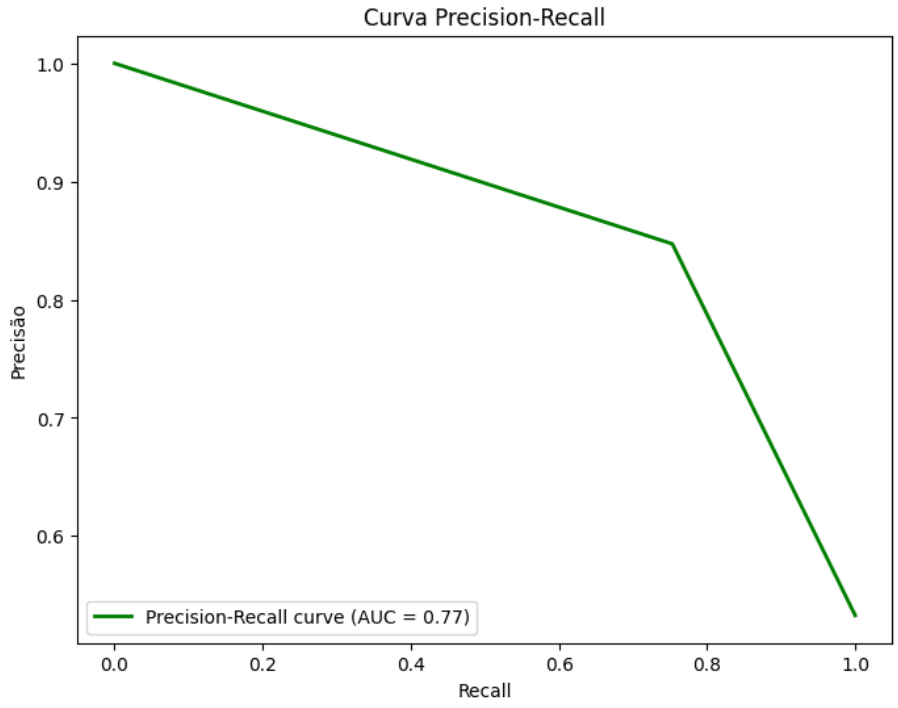

Figure 10. Decision Tree não ajustada

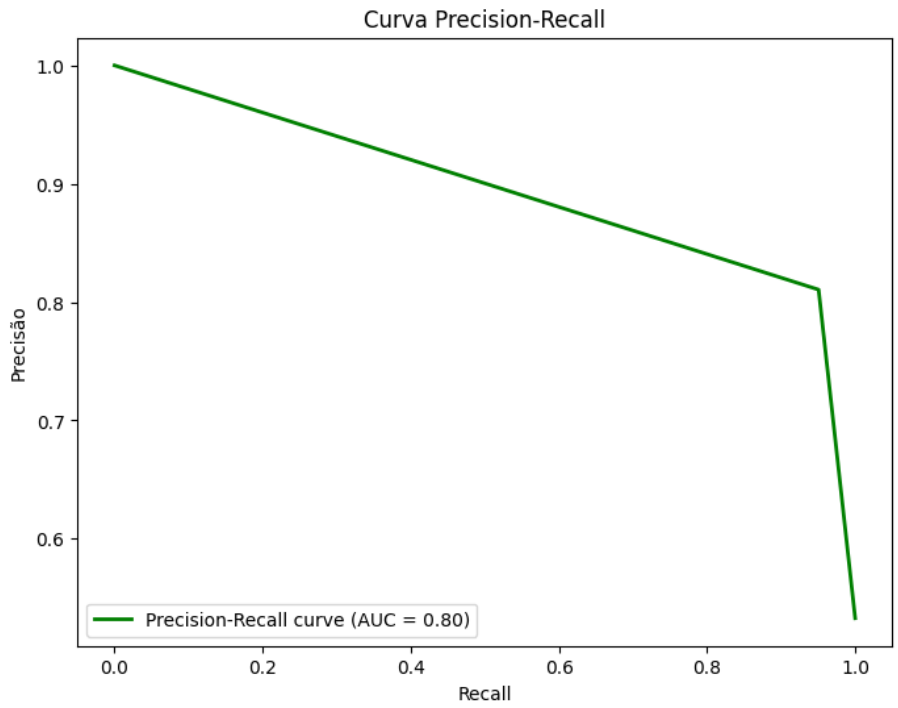

Figure 11. Decision Tree com grid search

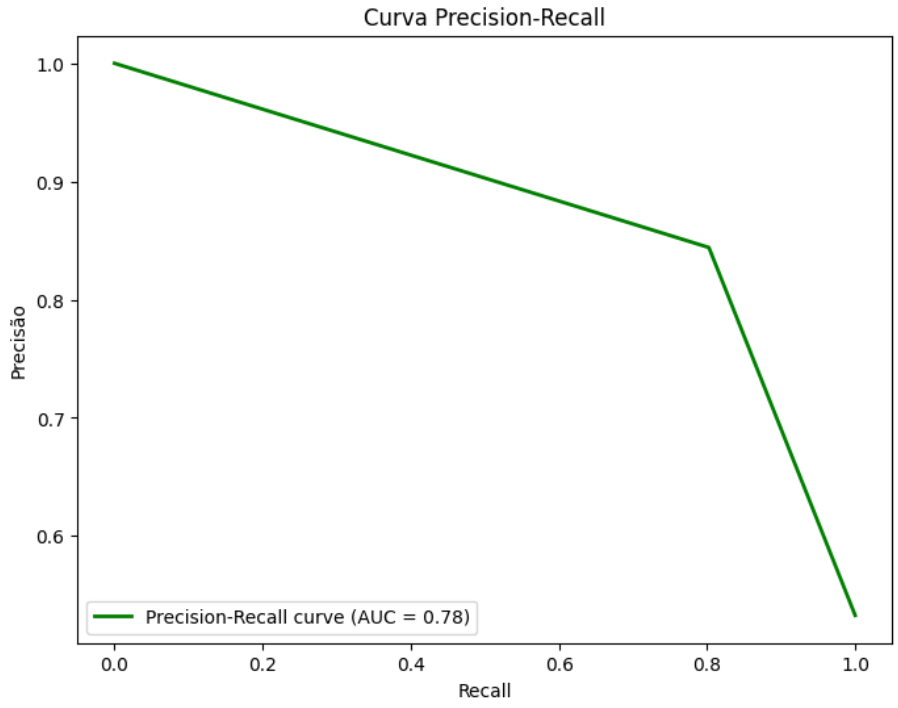

Figure 12. Decision Tree com Optuna