

Aprendizagem Computacional I

Adriana Oliveira - up202307136


Carolina Leite - up202307856

Lara Gonçalves - up202307857



Table of contents



- 01 Normalizar os dados dos Datasets**
valores nulos, colunas com elevada correlação, etc
 - 02 Implementar Random Forest Padrão**
E avaliar o desempenho
 - 03 Implementar Random Forest Modificado**
E avaliar o desempenho
 - 04 Comparar os algoritmos**
E teste de wilcoxon
 - 05 Resultados**
 - 06 Complicações e Conclusões**
- 




Random Forest

Vantagens

- Alta performance e precisão
- Menor risco de overfitting
- Robustez a dados ruidosos ou incompletos
- Trabalha bem com dados de alta dimensão

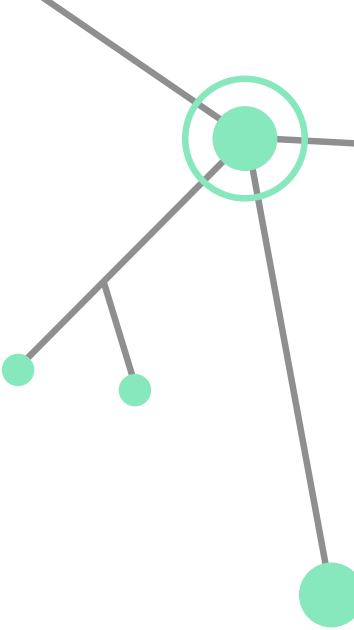
Desvantagens

- Menos interpretável
 - Mais lento e pesado
 - Pode ser enviesado em dados desbalanceados
- 



Porquê o Random Forest?

Implementação Random Forest



Class BaseEstimator

Validação e preparação dos dados de entrada (X e y).

Class Tree

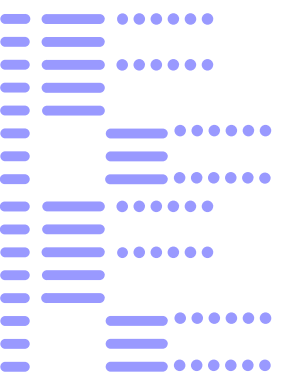
Implementa Uma árvore de decisão recursiva para classificação ou regressão, com busca gulosa pelo melhor split.

Class Random Forest

Implementa uma floresta de árvores de decisão, onde várias árvores são treinadas em paralelo com subconjuntos dos dados

Class RandomForestClassifier

Implementa a versão de classificação do RandomForest, com voto majoritário (média de probabilidades) entre árvores.





Class Imbalance in Binary Classification





Random Forest Vs. Class Imbalance

- **Tendência para a classe majoritária:** o modelo aprende mais facilmente a prever a classe com mais exemplos, ignorando as minoritárias.
- **Baixa atenção/sensibilidade às classes raras:** o recall para as classes com menos amostras tende a ser baixo, ou seja, o modelo falha em identificá-las corretamente.
- **Importância falsa nas features:** como a maioria dos dados pertence à classe dominante, o cálculo da importância das variáveis pode ficar enviesado.

Implementação Random Forest Modificado

weighted_focal_entropy

Calcula uma entropia ponderada e focada, usada para medir a impureza de um conjunto de rótulos (y), com mais penalização para erros em classes minoritárias.

calcular_class_weights

Calcula pesos automáticos para cada classe com base no nível de desbalanceamento.

escolher_focal_gamma

Escolhe um valor adequado para o parâmetro gamma do focal loss, com base no desbalanceamento.

Implementação Random Forest Modificado

escolher_parametros

Sugere hiperparâmetros da árvore (max_depth e min_samples_split) com base no desbalanceamento dos dados.

encontrar_melhor_thresfold

Encontra o melhor limiar de decisão por classe (threshold) com base na curva de precisão-recall.

Implementação Random Forest Modificado

Class Tree2

- Utiliza entropia focal ponderada
- Incentiva divisões (splits) que ajudam a classe minoritária (min_class)
- Calcula o melhor split com base no ganho de entropia ponderado
- Faz previsões enquanto percorre os nós das árvores (predict_row e predict)

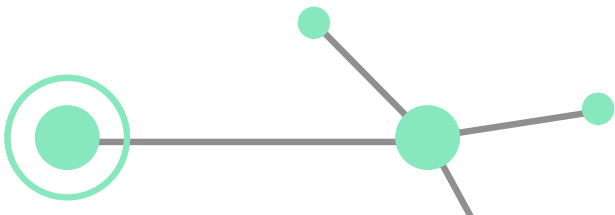
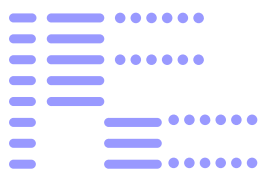
Class RandomForestClassifier

- Treina várias árvores (Tree2) em paralelo com amostragem bootstrap (fit)
- Usa entropia focal e pesos de classe para favorecer a classe minoritária
- Cada árvore recebe um peso baseado no desempenho da minoria (f1-score e recall)
- A predição final é feita por votação ponderada das árvores



Hiperparâmetros: Decision Tree

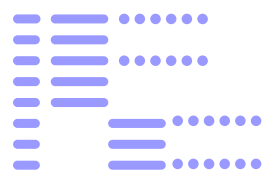
Hiperparâmetros	Antes: Tree	Depois: Tree2
<i>criterion</i>	<i>entropy</i>	Depende do dataset
<i>max_depth</i>	<i>None</i>	Depende do dataset
<i>min_sample_split</i>	10	Depende do dataset
<i>gamma</i>	0	Depende do dataset





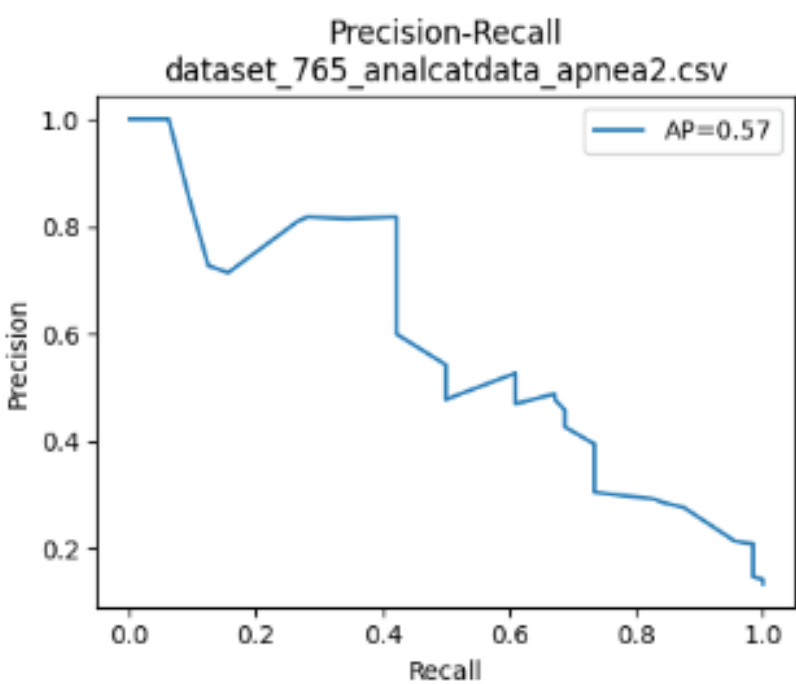
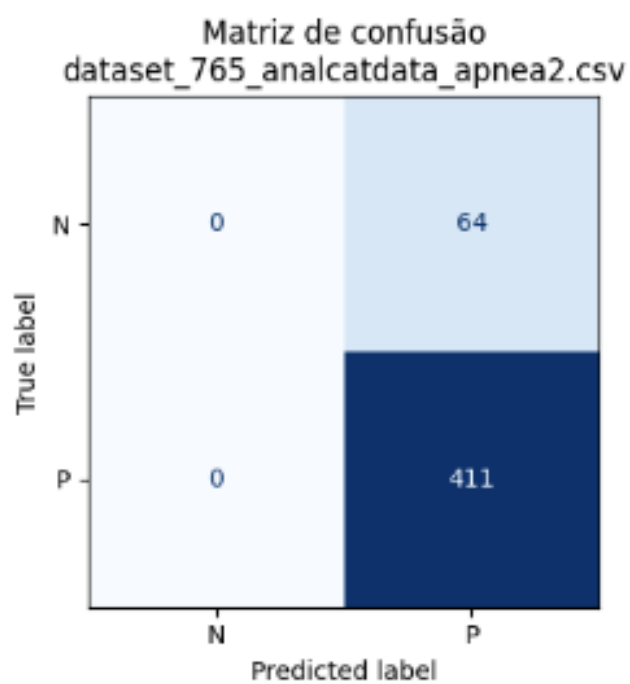
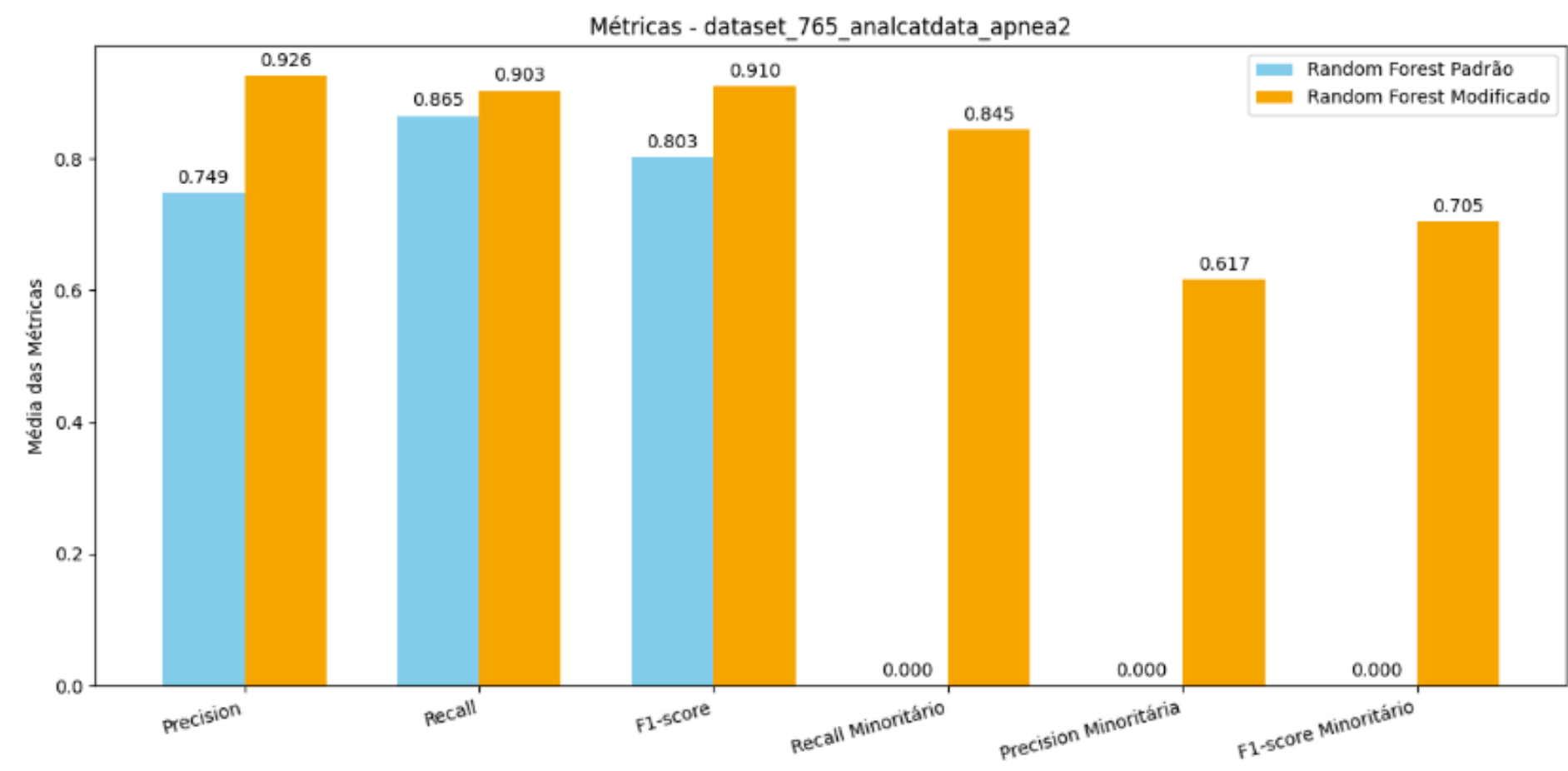
Hiperparâmetros: Random Forest

Hiperparâmetros	Antes	Depois
<i>n_estimators</i>	10	50
<i>criterion</i>	<i>entropy</i>	Depende do dataset
<i>max_depth</i>	<i>None</i>	Depende do dataset
<i>min_sample_split</i>	10	Depende do dataset
<i>gamma</i>	0	Depende do dataset

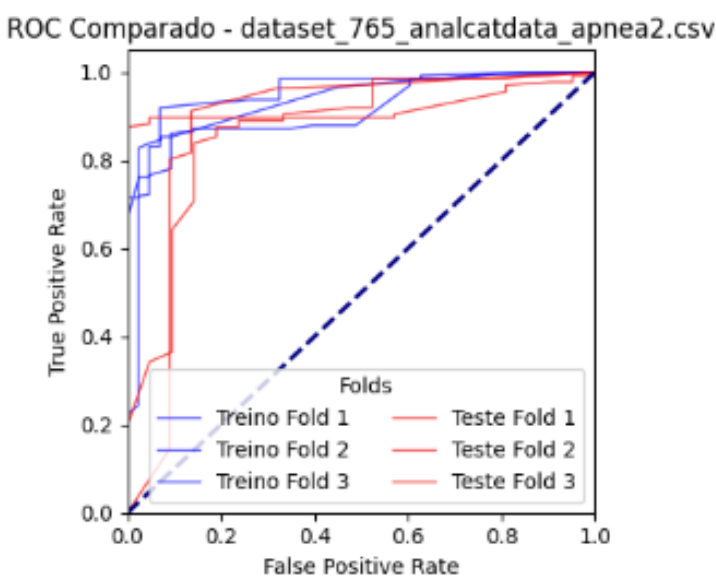


Desempenho dos Algoritmos

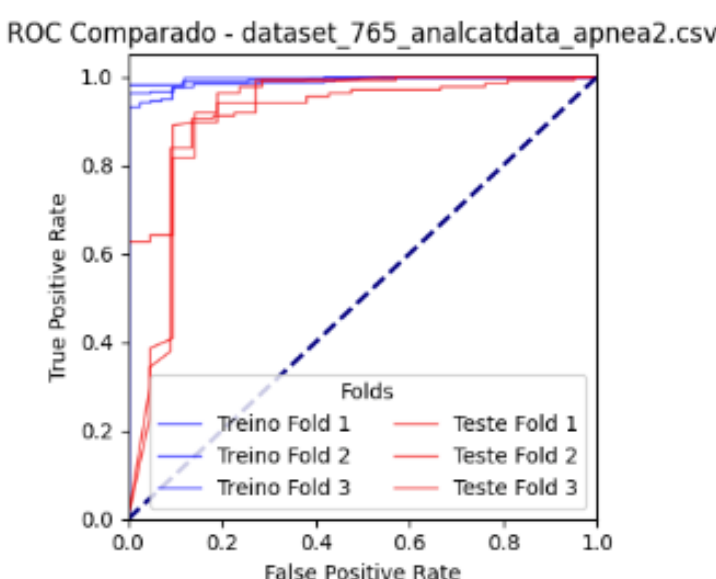
Antes



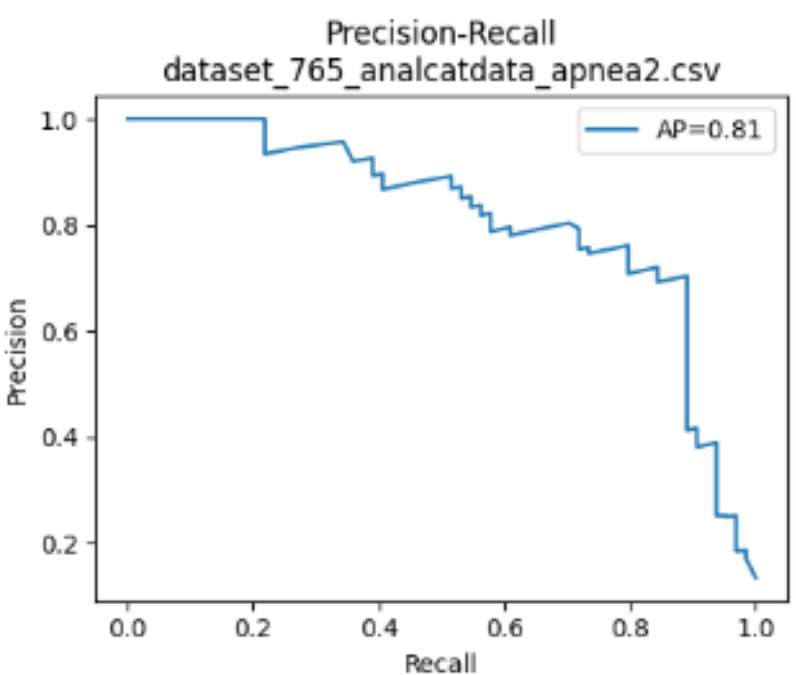
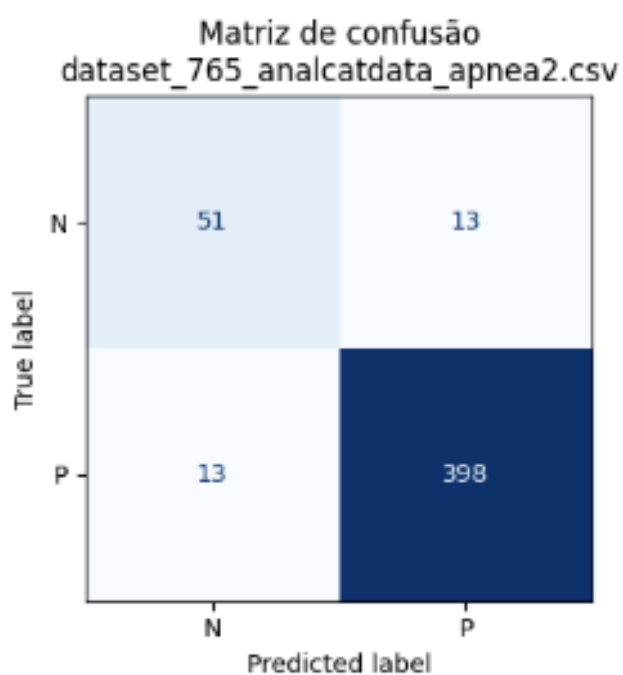
Antes



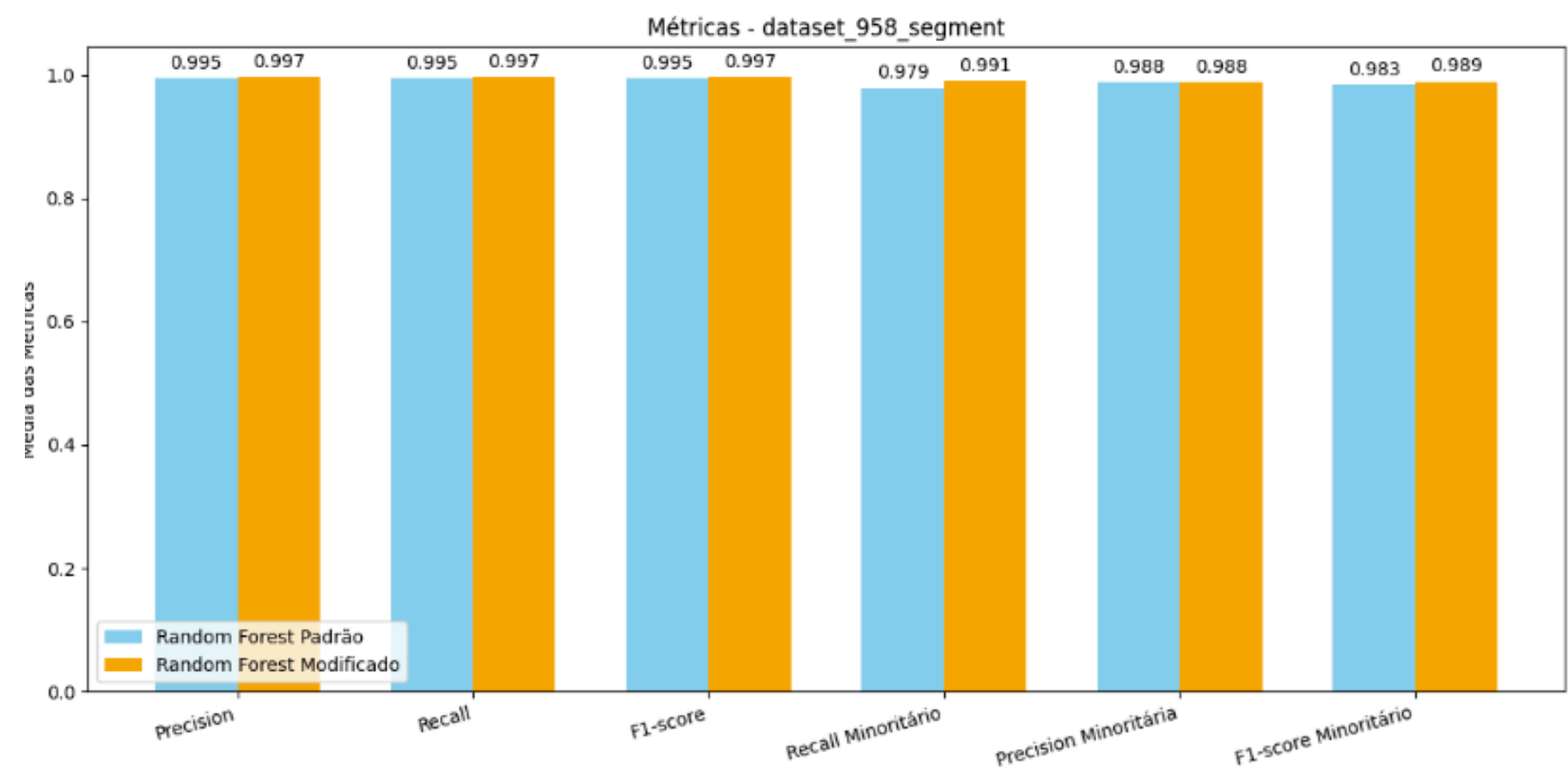
Depois



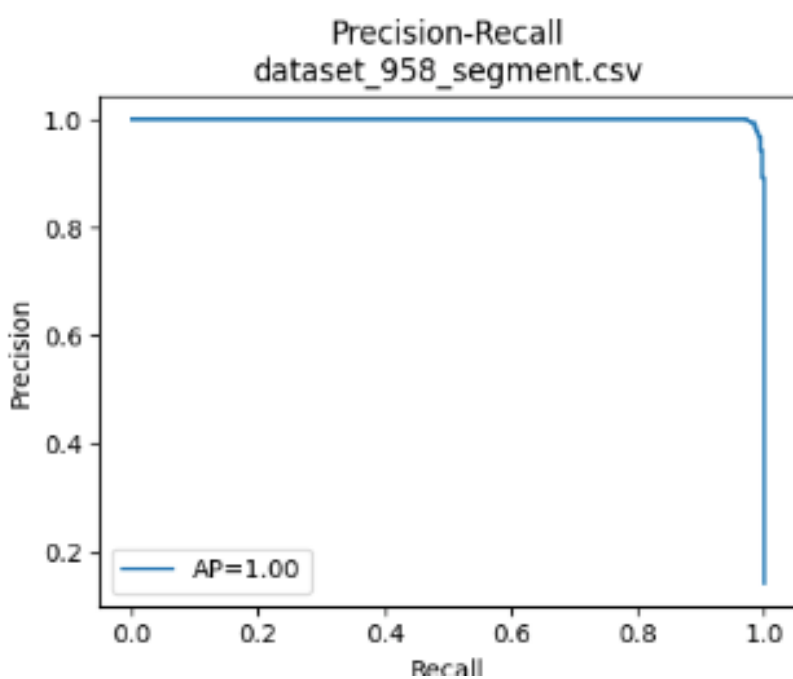
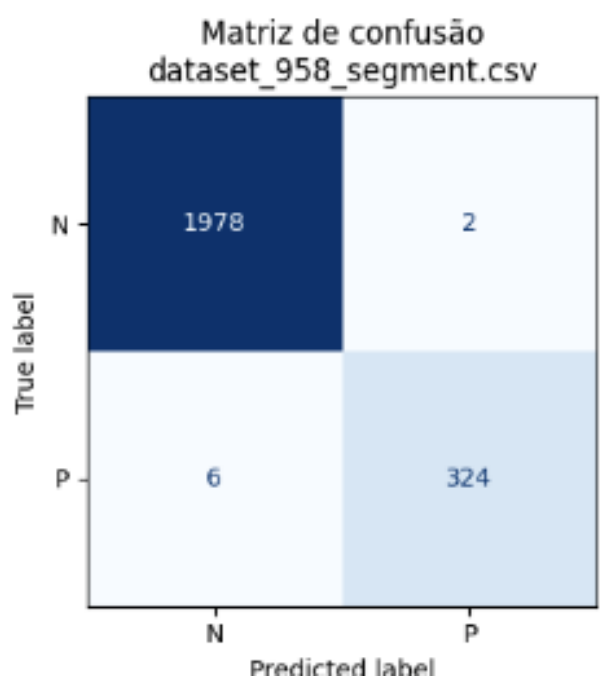
Depois



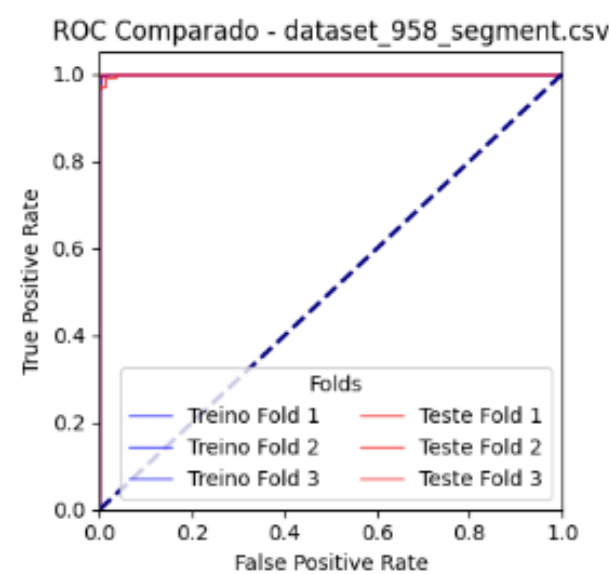
Desempenho dos Algoritmos



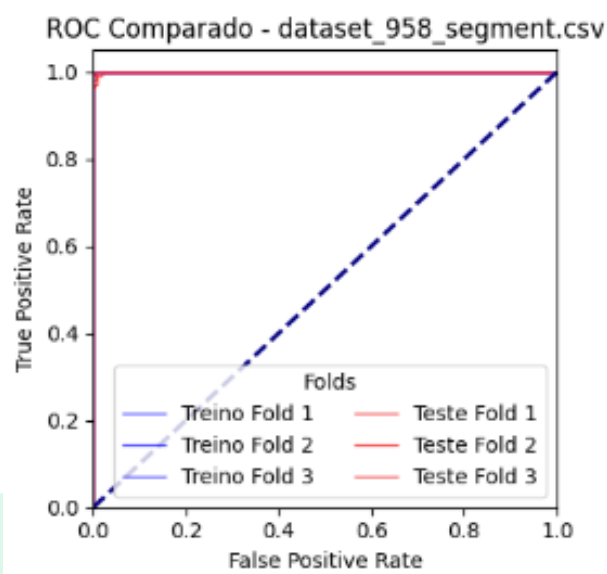
Antes



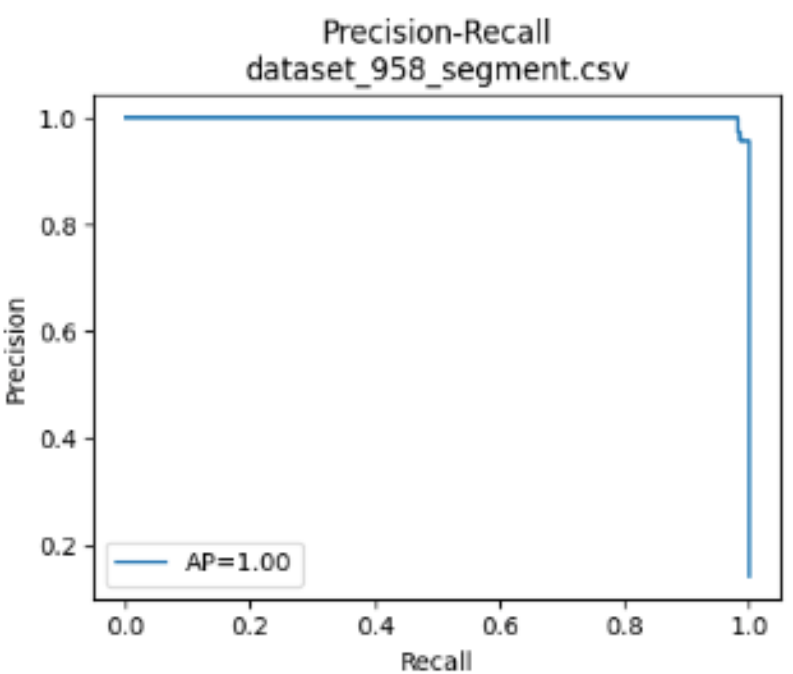
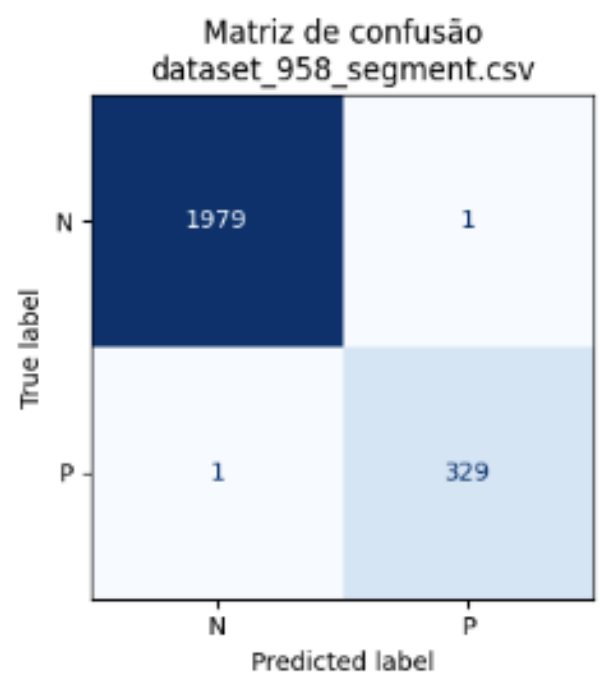
Antes



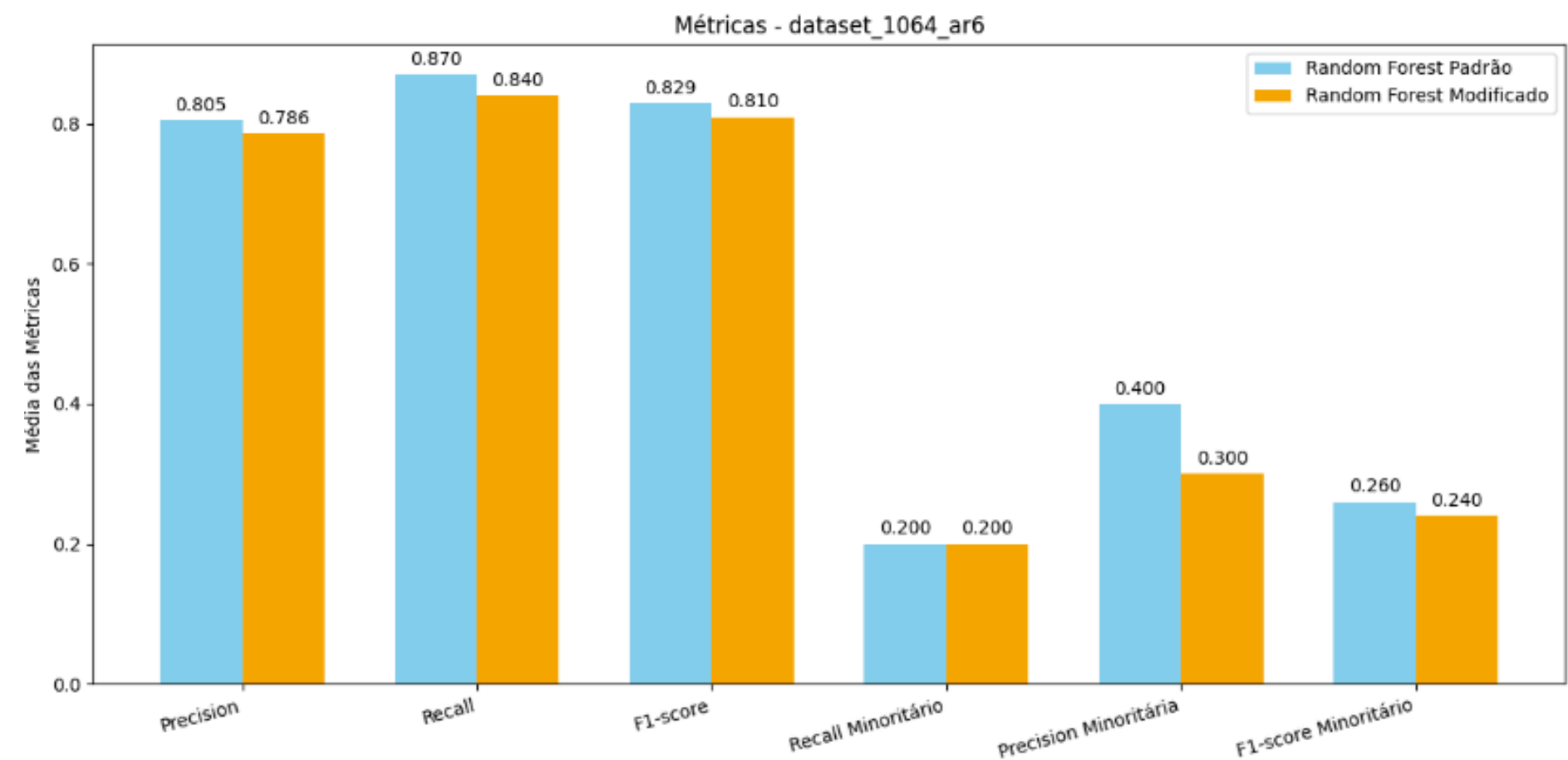
Depois



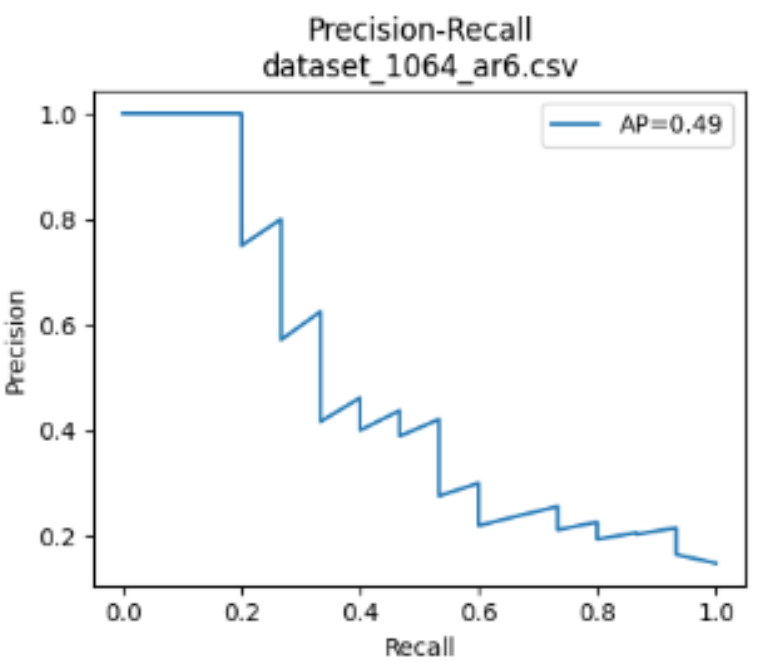
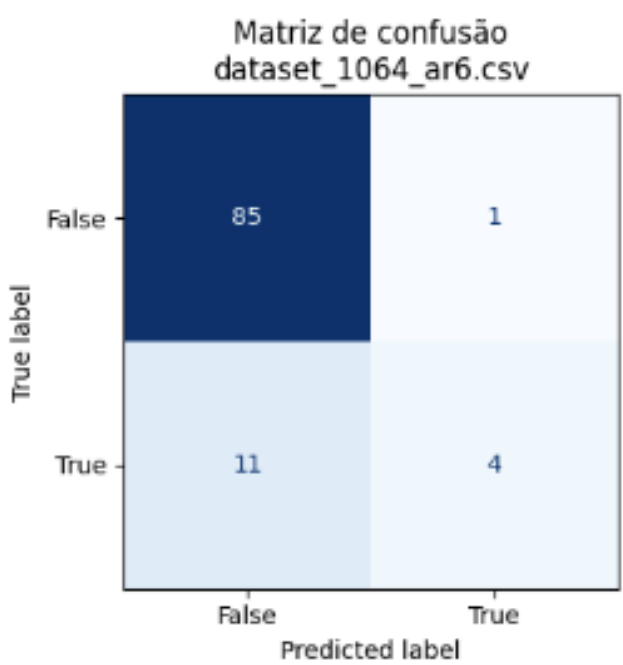
Depois



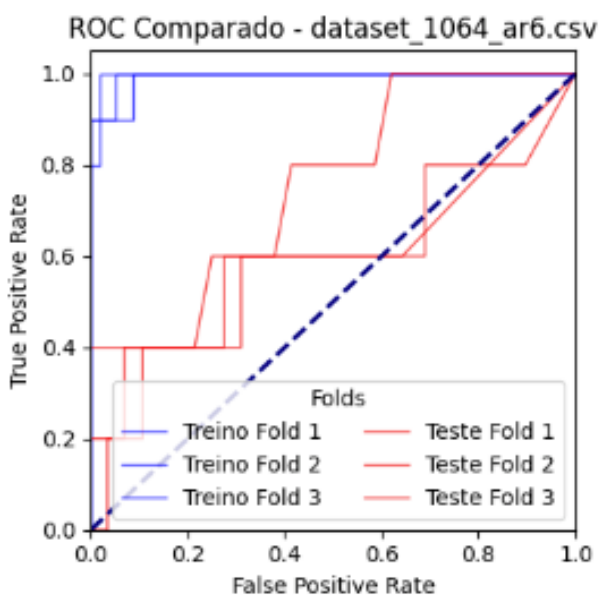
Desempenho dos Algoritmos



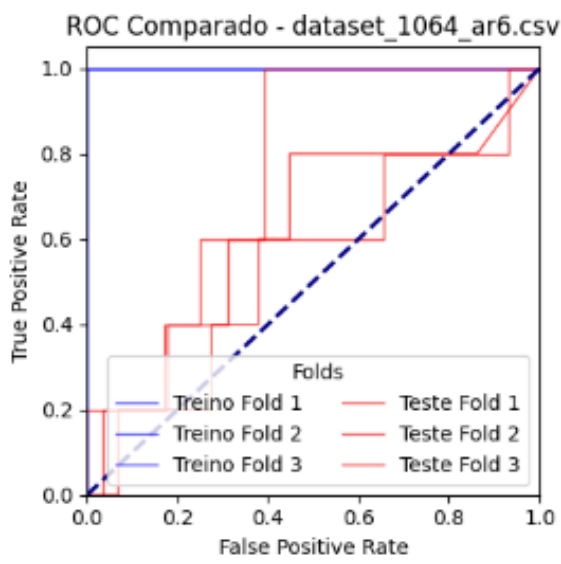
Antes



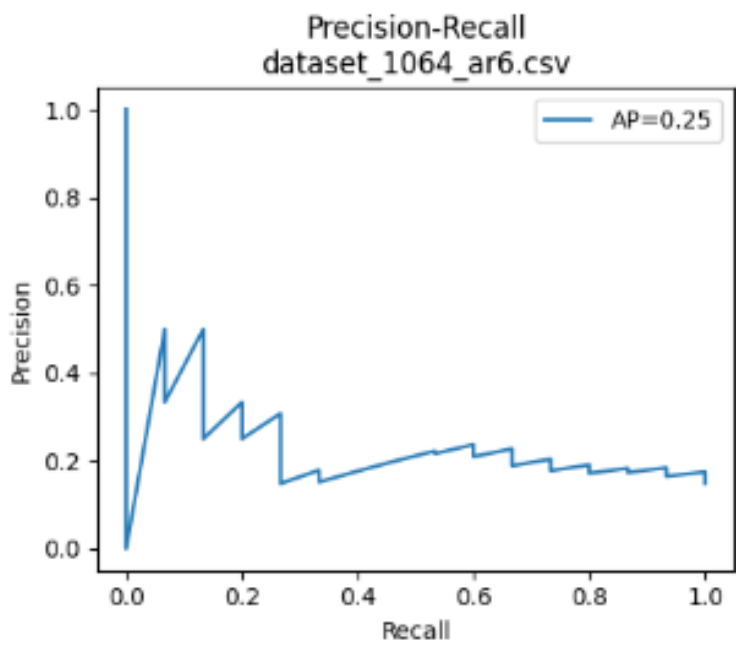
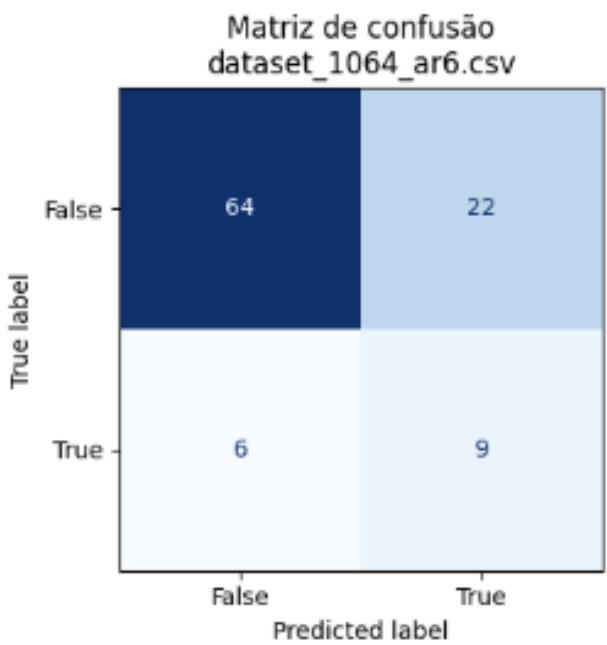
Antes



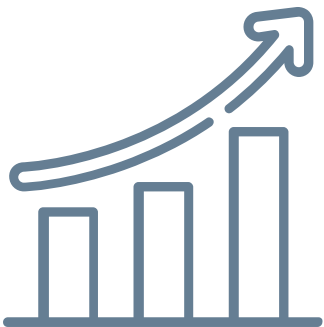
Depois



Depois



Desempenho Dos Algoritmos - Teste Estatístico



Teste de Wilcoxon

Métricas	Estatística	p_valor	Confiança
<i>precision</i>	238.00	0.00033	99.97%
<i>recall</i>	349.00	0.01423	98.58%
<i>f1-score</i>	445.00	0.14246	85.75%
recall minoritária	0.00	3.52107 e-9	100.00%
<i>precision minoritária</i>	333.00	0.03729	96.27%
f1-score minoritária	98.00	5.01494 e-7	100.00%

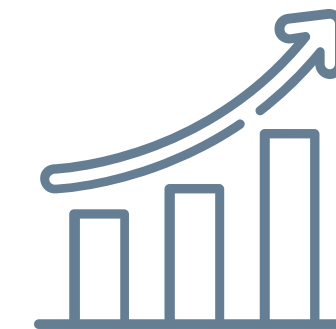
Desempenho Dos Algoritmos - Teste Estatístico



Teste de Wilcoxon

Métricas	Estatística	p_valor	Confiança
Métricas Combinadas	9823.00	9.12250 e-14	100.00%

Desempenho Dos Algoritmos - Melhoria Percentual

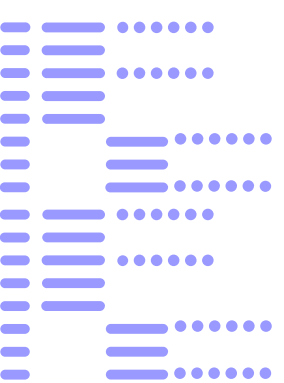


Métricas	Média Padrão	Média Modificado	Melhoria
<i>precision</i>	0.8971	0.9177	+ 2.30 % ▲
<i>recall</i>	0.9325	0.8992	-3.57 % ▼
<i>f1-score</i>	0.9097	0.9028	-0.75 % ▼
recall minoritária	0.3679	0.6773	+ 84.09 % ▲
<i>precision minoritária</i>	0.5124	0.6074	+ 18.54 % ▲
f1-score minoritária	0.4076	0.6098	+ 49.62 % ▲

Melhoria % Global (média do RF Modificado sobre o Padrão) → + 25.04% ▲

Complicações durante o Desenvolvimento do Projeto

- Desbalanceamento extremo de classes
- Trade-offs entre métricas
- Diferenças marginais de desempenho
- Limitações computacional





Conclusão Final e Trabalho Futuro

- **Ciclos Iterativos com Interpretabilidade**

- Técnicas explicativas como SHAP ou LIME podem ser usadas não apenas para interpretar decisões do modelo, mas também como ferramentas para detectar sinais de overfitting, por exemplo, quando certas variáveis dominam excessivamente as decisões nas árvores.
- **Resultado:** seria possível construir um ciclo de feedback onde o modelo é ajustado iterativamente, com foco em generalizar melhor sem comprometer a identificação da classe minoritária.

- **Robustez Adversarial Focada na Classe Minoritária**

- Uma proposta inovadora seria aplicar testes adversariais direcionados à classe minoritária, de forma a avaliar o impacto que pequenas perturbações nos exemplos dessa classe causam na estabilidade do modelo.
- **Resultado:** Esse tipo de teste revelaria o quanto o modelo realmente aprendeu os padrões da classe minoritária, e não apenas memorizações — ajudando a combater o overfitting de forma mais inteligente e focada.