Gabriel Henrique Vasconcelos

Professor Leandro Maciel Almeida

Aprendizagem de Máquina

31 de Março de 2025

Relatório do Projeto Aprendizado de Máquina

ENTENDIMENTO DO NEGÓCIO

Contextualização

O mercado imobiliário é altamente competitivo e vem crescendo cada vez mais nos últimos anos, trazendo com isso a necessidade de se refinarem as estratégias para se manter nele. A atribuição de preço a casas e apartamentos é uma prática que apesar de se debruçar sobre uma base objetiva dos dados referentes àquele imóvel, ainda se utilizade uma subjetividade humana, além da sazonalidade e oscilações do contexto do momento. Desde área, quantidade de cômodos, até a localização, vizinhança, ou a conjuntura econômica são atributos que influenciam no seu valor, e alguns desses atributos podem variar de acordo com a subjetividade de quem compra. Dessa forma, identificando a natureza multifatorial deste problema podemos inferir que trata-se de uma situação que pode se beneficiar de modelos de aprendizagem de máquina para tratar essa grande quantidade de atributos diferentes que o compõem e assim se tornar uma ferramenta efetiva para o problema.

Objetivos do negócio

Para entender as metas estabelecidas é preciso saber que o dataset utilizado tem um target binário, ou seja, trata-se de um problema de classificação. Então, o objetivo do negócio é criar modelos de aprendizagem capazes de forma satisfatória, afim de demonstrar que as ferramentas de inteligência artificial podem fazer parte do ecossistema imobiliário.

Desafios e limitações

Por se tratar de um problema multifatorial e discretizado a partir de uma base de dados, não temos as descrições de cada atributo. Portanto, o entendimento dos resultados surgirá a partir da avaliação do desempenho dos modelos, e não de análises dos dados. Dado que a base de dados é grande e com muitos atributos, e também, como descrita na documentação, possivelmente com alta variância e baixa correlação entre atributos, é possível que isso exija uma complexidade maior do modelo aplicado, o que pode resultar em uma baixa performance caso essa complexidade não seja atingida.

Critérios de sucesso do negócio

Atingir uma acurácia de 90% na classificação dos imóveis. Esse valor demonstra uma performance suficientemente boa para auxiliar usuários na identificação de imóveis caros, o que pode trazer maior segurança nos negócios.

Inventário de recursos

O projeto contará com uma base de dados imobiliária do census governamental dos Estados Unidos, a qual possui 16 atributos e um rótulo binário criado a partir do preço. No quesito

computacional, o desenvolvimento se beneficiará do uso da linguagem de programação Python e suas bibliotecas para manipulação de dados e criação de modelos de inteligência artificial, como pandas, Scikitearn e PyTorch. Além disso, a utilização do ambiente de desenvolvimento colaborativo Google Colab para trabalho em equipe, implementação dos modelos e treinamento utilizando gpu.

Requisitos

Implementação de um modelo de classificação binária para precificação de imóveis e sua avaliação por métricas como acurácia, f1-score, precision, recall.

Pressuposições. O problema é resolvível por modelos de aprendizado de máquina ou não excede a complexidade que o poder computacional disponível nos permite utilizar. Os dados são representativos da realidade.

Restrições

Atributos do dataset foram discretizados, dessa forma perdemos um entendimento do que cada atributo significa na realidade. Possível baixa correlação entre variáveis.

Riscos

Baixa performance do modelo como reflexo de uma complexidade alta no problema. Baixa capacidade de generalização por conta do dataset não ser representativo da realidade. Baixa capacidade de explicação dos resultados por conta dos atributos estarem discretos.

Objetivos da ciência de dados

O objetivo da ciência de dados neste projeto é ser capaz de identificar padrões nos dados que nos permitam classificar entradas em uma das duas classes do problema, através da criação, treinamento e testes de modelos de aprendizagem de máquina. Dessa forma, proporcionando uma ferramenta computacional capaz de diminuir a subjetividade na precificação no mercado imobiliário.

Critérios de sucesso da ciência de dados

Acurácia do Modelo – O modelo deve atingir pelo menos 90% de acurácia na classificação dos imóveis, conforme definido nos critérios de sucesso do negócio.

Equilíbrio entre Precisão e Recall – Métricas como F1-score devem ser avaliadas para garantir que a classificação não esteja enviesada para uma das classes.

Generalização – O modelo deve ser capaz de fazer previsões consistentes em diferentes subconjuntos dos dados, evitando overfitting.

Eficiência Computacional – O tempo de treinamento e inferência do modelo deve ser viável para implementação prática.

Plano de projeto

O projeto será desenvolvido seguindo as etapas:

Planejamento e análise do problema: Fase inicial do projeto que visa entender o contexto, listar as restrições e possibilidades, conhecer as ferramentas e técnicas disponíveis e estabelecer metas.

Análise dos dados. Fase na qual se realiza um mergulho no dataset do problema visando entender melhor a realidade do que será trabalhado. Nesta fase já iremos realizar as manipulações necessárias para que os dados estejam no formato ideal para o treinamento dos modelos.

Modelagem e treinamento. Fase onde escolhemos, modelamos e treinamos o modelo. Métodos de escolha de hiperparâmetros e o uso de métricas para acompanhamento do desenvolvimento dos modelos serão utilizados para acompanhamento.

Avaliação e documentação. Fase onde analisamos o desempenho final obtido no conjunto de teste. Além disso será realizada a documentação da fase de desenvolvimento afim de entender os erros e acertos no processo.

Ferramentas e técnicas previstas

Ferramentas: Python (linguagem principal), Pandas, NumPy, Scikit-learn, PyTorch, Matplotlib e Seaborn. Técnicas: Regularização, cross-validation, gridsearch..

ANÁLISE EXPLORATÓRIA DOS DADOS

Relatório inicial da coleta de dados

Os dados foram obtidos a partir do site OpenML disponibilizados por Joaquin Vanschoren sendo a segunda versão de outro dataset. Os dados originais foram retirados do US Census Bureau de uma coleta que fez parte do censo de 1990 dos Estados Unidos. A versão 2 deste dataset, difere da versão anterior apenas em relação ao atributo “Price”, preço, o qual foi transformado em uma classe que divide em positivo e negativo a partir de um valor arbitrário

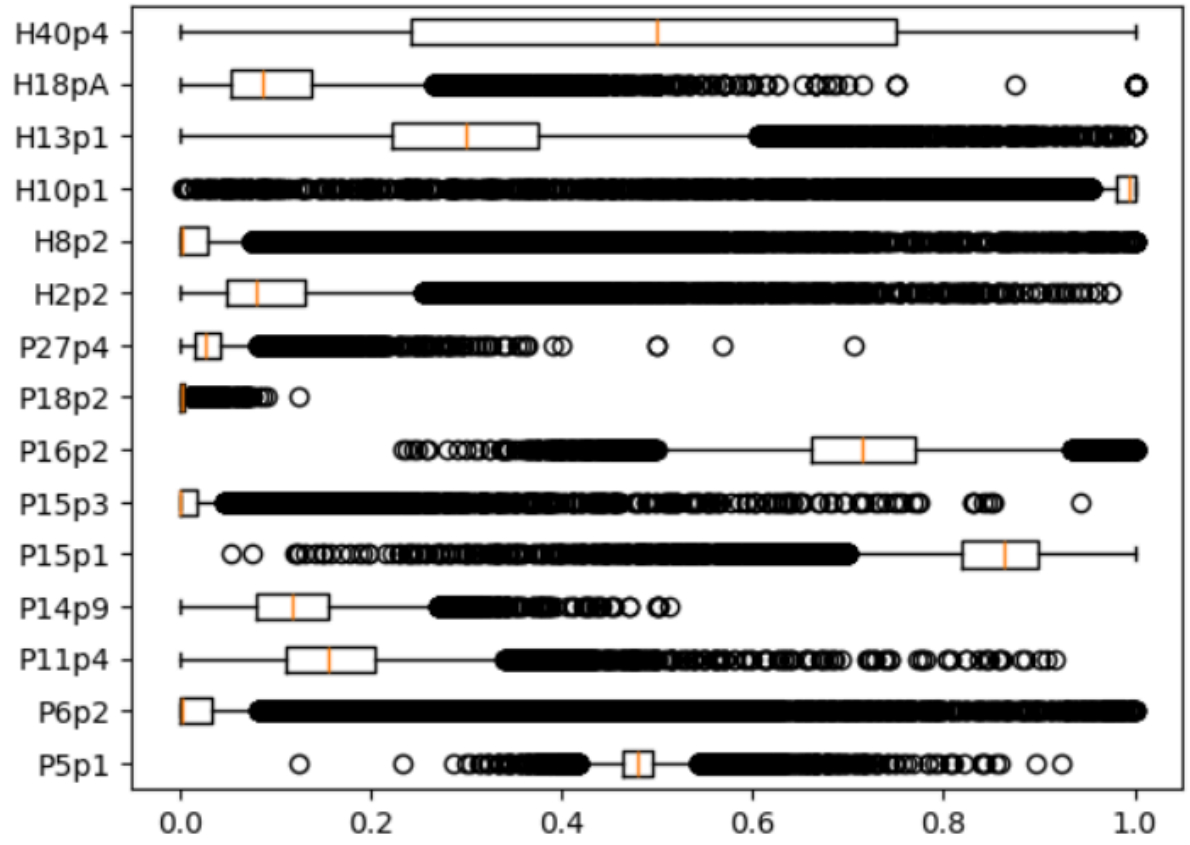
escolhido pelo desenvolvedor dos dados. Todas os atributos estão discretizados e não foram fornecidas descrições sobre eles na documentação.

Descrição dos dados

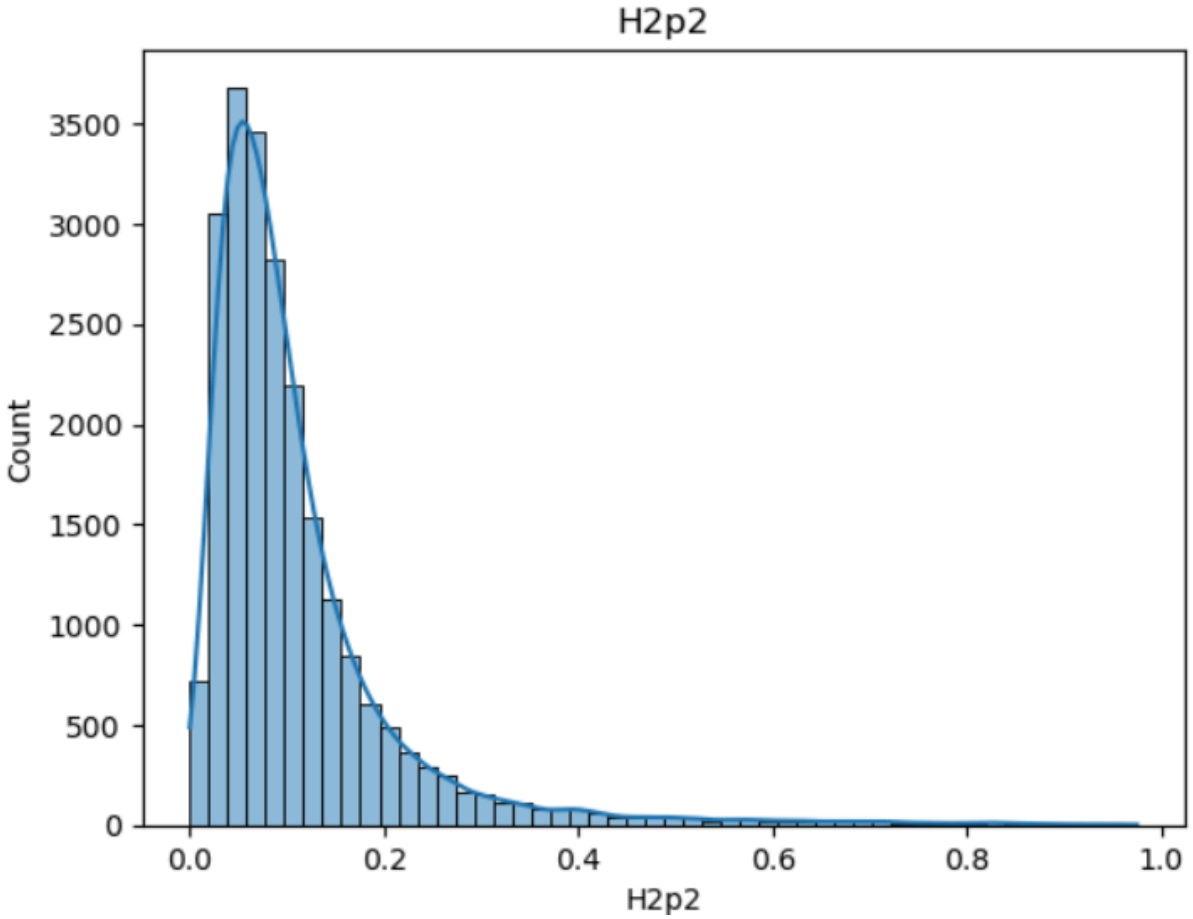
O conjunto conta com 17 colunas, sendo 16 features e 1 target baseado em preço, e possui 22784 exemplos. Não há valores ausentes na tabela. Todas as features são valores de ponto flutuante e o target é binário. As informações originais foram transformadas em proporções adequadas para este dataset, portanto encontram-se num alcance geral de 0 a 1 aproximadamente, com exceção da coluna “P1” que possui valores entre 2 e 7 milhões.

Exploração dos dados

Utilizamos a plotagem em boxplot para visualizar a distribuição de cada atributo e identificamos que a maior parte das colunas possuía uma grande quantidade de outliers.

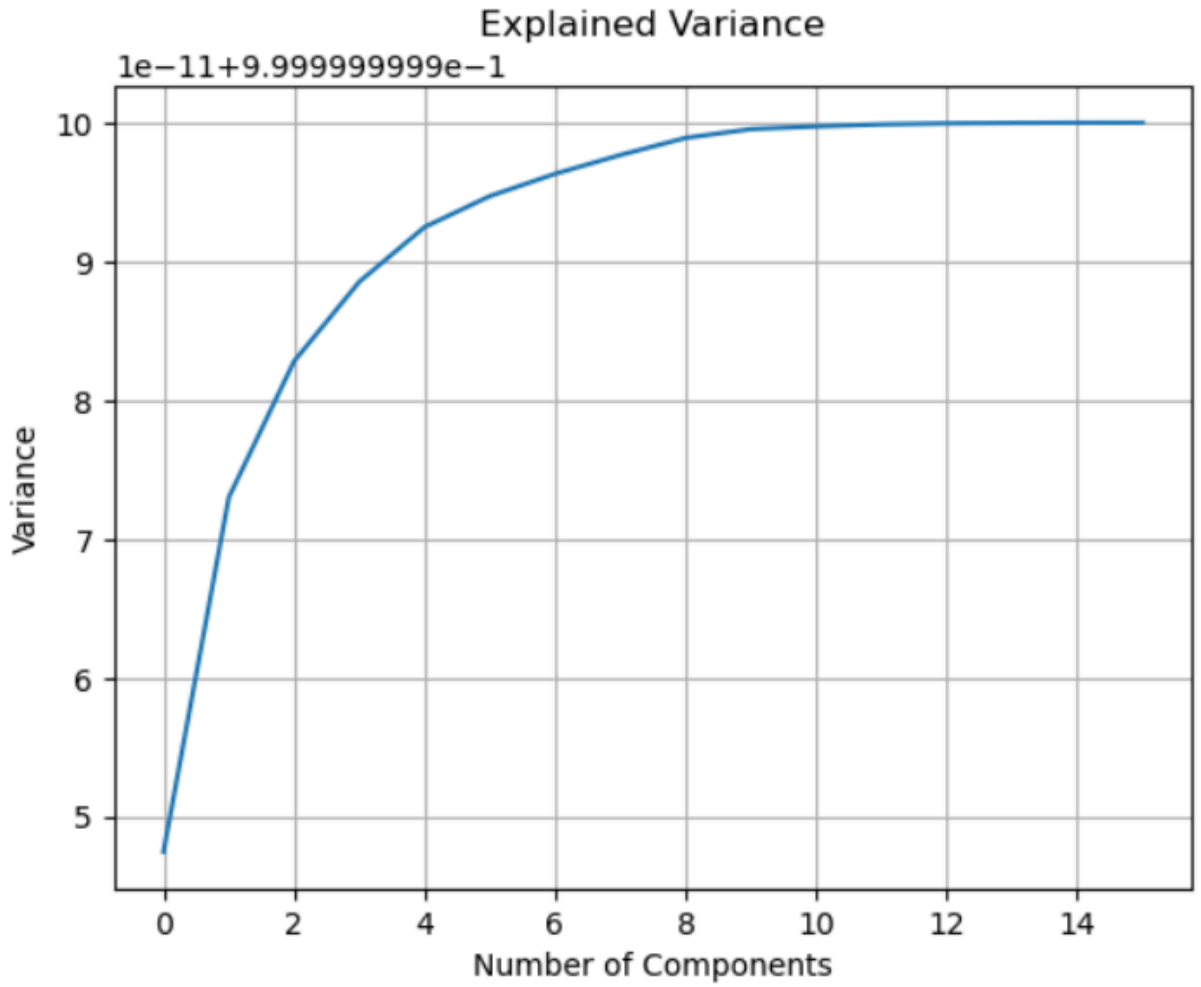


Isso evidencia que as colunas têm uma alta densidade próximo a determinado valor porém ainda existem valores que divergem dessa parte mais densa dos dados, ou seja a calda da distribuição é longa e o pico é alto. A alta presença de outliers, aliada ao fato de que sabemos que os dados originais eram contagens e foram transformados em proporções nos indica que a remoção desses valores não é a abordagem ideal.



Pegando a coluna “H2p2” como exemplo, um histograma ilustra de forma clara como há a concentração da maior parte dos valores em um alcance pequeno de 0 a 0.2, e ainda mantém-se tendo valores até 1. Essa mesma dinâmica foi observada na grande maioria das colunas do dataset. Ao realizar uma Análise de Componentes Principais (PCA), observamos que entre 8 e 10 componentes explicam aproximadamente 100% da variância dos dados. Isso indica que é possível reduzir a dimensionalidade do problema para um conjunto de 8 a 10 variáveis sem perda significativa de informação. Essa redução pode beneficiar a fase de

criação e teste de modelos de classificação, pois diminui a complexidade dos dados, reduzindo o risco de sobreajuste (overfitting) e melhorando a eficiência computacional, sem comprometer a performance preditiva.



A partir de uma análise da versão dos dados utilizadas no projeto v2, e a versão anterior, conseguimos identificar o limiar utilizado para separar as classes no problema, o que não foi explicado na documentação do conjunto. A classe target foi dividida da seguinte forma, valores menores ou iguais a 50000 foram mapeados para a classe positiva “P”, enquanto que valores maiores se tornaram a classe negativa “N”. Esta é a conclusão mais provável, entretanto não podemos afirmar com certeza, dado que ao realizar a análise encontramos que uma das classe tinha o valor máximo 50000 e a outra o valor mínimo de 50100, estando o limiar dentro desse alcance supomos 50000 por conveniência.

Qualidade dos dados

A ausência de dados faltantes é um fator positivo observado no dataset, uma vez que a etapa de tratamento desse tipo de problema não precisou ser implementada. A documentação deixou claro que foram realizadas alterações nos dados, entretanto não informou exatamente que tipo foi executado, apenas alegou ter transformado contagens em proporções adequadas. A falta de descrições na documentação sobre o significado de cada atributo é um fator que pode atrapalhar o processo de treinamento e aprendizagem do modelo, visto que ter informações sobre as colunas ajudaria em ter uma ideia do que influencia o desempenho do modelo e assim ter o entendimento e capacidade de explicação do problema na realidade. Apesar de afirmar ter realizado alterações, acreditamos que uma das colunas, a "P1", não foi alvo delas, mas a conclusão dessa suposição não pode ser feita dado que não sabemos o que a coluna representa. A grande presença de outliers não é um problema, e a partir da análise que fizemos, acreditamos que o sucesso do projeto depende em grande parte da forma como iremos lidar com eles. O dataset carece de uma descrição detalhada e diverge um pouco das afirmações feitas sobre ele na documentação. Apesar disso, acreditamos que ele tem qualidade o suficiente para que o projeto seja realizado e que as metas definidas sejam atingidas de forma satisfatória.

APLICAÇÃO DOS MODELOS

Escolha dos hiperparâmetros e alcances

Utilizamos o RandomizedSearchCV para a busca dos hiperparâmetros dado que a busca em grid para alguns modelos poderia resultar em altos tempos de treinamento. O algoritmo funciona escolhendo aleatoriamente os valores dentro do espaço definido na busca para construir o modelo. Fizemos o treinamento de cada um dos 20 modelos resultantes da busca

no conjunto de treinamento e guardamos o desempenho f1-score deles avaliado no conjunto de validação. O modelo final escolhido foi o que obteve a melhor performance nessa avaliação.

KNN

A escolha dos hiperparâmetros do modelo K-Nearest Neighbors (KNN) foi baseada na exploração de um espaço de busca abrangente, dado que o modelo é simples e tem um treinamento rápido. Para o KNN, realizamos 20 iterações do da busca, o que resulta em 20 modelos finais para comparação da métrica escolhida. E a busca, foi realizada com n_iter com valor 10, que significa que cada iteração da busca testou 10 combinações diferentes dos hiperparâmetros e retornou a que desempenhou melhor. A velocidade de treinamento do modelo foi o que proporcionou essa amplitude de busca. Os hiperparâmetros ajustados foram:

Número de vizinhos (n_neighbors): Definimos um intervalo entre 1 e 50 para capturar a transição entre um modelo altamente sensível ao ruído (overfitting em valores baixos) e um modelo mais estável, porém potencialmente menos responsivo a padrões locais (underfitting em valores altos).

Peso das amostras (weights):

Uniform: Todos os vizinhos contribuem igualmente para a predição.

Distance: Os vizinhos mais próximos têm maior influência, útil para conjuntos de dados com variações na densidade.

Métrica de distância (metric): Diferentes métricas foram testadas para avaliar o impacto da distância na performance do modelo:

Euclidean: Distância padrão para dados contínuos.

Manhattan: Melhor para dados onde as variáveis seguem um formato de grade ou esparsidade elevada.

Minkowski: Generalização das métricas anteriores, ajustável conforme o parâmetro p.

Algoritmo de busca (algorithm): Como a eficiência computacional é um fator importante, avaliamos diferentes abordagens para encontrar os vizinhos mais próximos:

Auto: Seleção automática do melhor algoritmo com base nos dados.

Ball Tree e KD Tree: Estruturas eficientes para buscas em alta dimensionalidade.

Brute: Busca exaustiva, útil como referência para comparação.

O processo de busca levou 2 minutos e 37 segundos para ser concluído. O modelo final escolhido foi aquele que obteve o melhor f1-score no conjunto de validação.

Os hiperparâmetros vencedores foram:

Número de vizinhos: 14

Peso da amostras: uniform

Métrica de distância: manhattan

Algoritmo de busca: ball_tree

SVC

A escolha dos hiperparâmetros do modelo Support Vector Classifier (SVC) foi baseada na exploração de um espaço de busca abrangente, garantindo um equilíbrio entre desempenho e viabilidade computacional. Como o treinamento do SVC pode ser mais custoso, a utilização

do RandomizedSearchCV permitiu testar diferentes combinações de hiperparâmetros sem necessidade de uma busca exaustiva. É importante pontuar que o modelo utilizado foi o LinearSVC o qual é mais simples e tem menor capacidade de generalização para problemas com mais dimensões. Realizamos 20 iterações da busca e o número de combinações testadas por iteração foi definido com n_iter = 10. Os hiperparâmetros ajustados foram:

Parâmetro de regularização (C): Definimos um conjunto de valores variando de 0.001 a 50 para controlar o equilíbrio entre margem e erro de classificação. Valores mais baixos incentivam um modelo mais simples e generalizável, enquanto valores mais altos permitem ajustes mais específicos aos dados de treinamento.

Função de perda (loss):

Hinge: Perda padrão do SVM, adequada para classificadores de margem máxima.

Squared Hinge: Variante que penaliza desvios quadraticamente, resultando em uma solução mais estável.

Penalidade (penalty):

L1: Favorece soluções esparsas, eliminando alguns coeficientes.

L2: Penaliza grandes coeficientes, incentivando uma solução mais regularizada.

Forma da otimização (dual):

True: Utiliza a formulação dual do problema, mais eficiente quando o número de amostras for maior que o número de features.

False: Utiliza a formulação primária, indicada para casos com muitas features e menos amostras.

Número máximo de iterações (max_iter): Foram testados valores de 1000, 5000 e 10000 para garantir que o modelo tivesse tempo suficiente para convergir sem comprometer o tempo de execução.

O processo de busca levou 20 minutos e 20 segundos para ser concluído. O modelo final escolhido foi aquele que obteve o melhor f1-score no conjunto de validação.

Os hiperparâmetros vencedores foram:

Penalidade: L2

Número máximo de iterações: 10000

Função de perda: Hinge

Forma da otimização: True

Parâmetro de regularização: 20

MLP

A escolha dos hiperparâmetros do modelo Multi-Layer Perceptron (MLP) foi baseada na exploração de um espaço de busca abrangente, considerando a complexidade do modelo e a necessidade de balancear desempenho e tempo de treinamento. Como redes neurais podem ser sensíveis à escolha dos hiperparâmetros, utilizamos RandomizedSearchCV para testar diferentes configurações sem necessidade de uma busca exaustiva.

Realizamos 20 iterações da busca, resultando em 20 modelos distintos, cada um avaliado com base no f1-score no conjunto de validação. O número de combinações testadas por iteração foi definido com n_iter = 5, garantindo que cada execução testasse 5 combinações aleatórias de hiperparâmetros e retornasse a melhor. Os hiperparâmetros ajustados foram:

Tamanho das camadas ocultas (hidden_layer_sizes): Foram testadas arquiteturas com (125,), (100,), (50,), (25,) e (10,) neurônios para avaliar o impacto da complexidade do modelo na capacidade de generalização.

Função de ativação (activation): Diferentes funções de ativação foram exploradas para verificar qual se adaptava melhor à estrutura dos dados:

Identity: Ativação linear.

Logistic: Função sigmoide, útil para problemas de classificação binária.

Tanh: Variante da sigmoide, porém centrada em zero.

ReLU: Ativação amplamente utilizada por sua eficiência computacional e mitigação do problema do gradiente desaparecendo.

Otimizador (solver): Foram avaliados diferentes métodos para a atualização dos pesos da rede:

LBFGS: Algoritmo de segunda ordem eficiente para conjuntos menores.

SGD: Gradiente descendente estocástico, eficiente para grandes volumes de dados.

Adam: Método adaptativo amplamente utilizado devido à sua robustez.

Tamanho do lote (batch_size): Foram testados valores de 16, 32, 64 e 128, avaliando o impacto na estabilidade do treinamento e no desempenho da rede.

Taxa de aprendizado (learning_rate):

Constant: Mantém uma taxa de aprendizado fixa.

Invscaling: Ajusta a taxa de aprendizado de forma inversamente proporcional ao número de iterações.

Adaptive: Ajusta dinamicamente a taxa de aprendizado com base no desempenho do modelo.

O processo de busca levou 15minutos e 56 segundos para ser concluído. A complexidade do modelo exigiu uma abordagem equilibrada para garantir um bom desempenho sem aumentar excessivamente o tempo de treinamento. O modelo final escolhido foi aquele que obteve o melhor f1-score no conjunto de validação.

Os hiperparâmetros vencedores foram:

Otimizador: LBFGS

Taxa de aprendizado: Adaptive

Tamanho da(s) camada(s) oculta(s): (50,)

Tamanho do lote: 32

Função de ativação: Logistic

Árvore de decisão

Foram explorados diferentes hiperparâmetros para avaliar a influência de cada configuração na performance do modelo. O uso do RandomizedSearchCV permitiu testar uma variedade de combinações de forma eficiente. Os hiperparâmetros ajustados foram:

Critério de divisão (criterion): Define a métrica usada para avaliar a qualidade da separação:

Gini: Mede a impureza dos nós.

Entropy: Baseada na teoria da informação, favorece divisões mais informativas.

Log Loss: Variante que utiliza a perda logística para classificações probabilísticas.

Estratégia de divisão (splitter):

Best: Escolhe a melhor divisão para cada nó.

Random: Realiza divisões aleatórias, útil para introduzir variação no modelo.

Profundidade máxima da árvore (max_depth): Variando entre 1 e 16, para evitar sobreajuste e manter um bom equilíbrio entre generalização e expressividade do modelo.

Número mínimo de amostras para divisão (min_samples_split): Testamos valores entre 2 e 19, ajustando o tamanho mínimo necessário para que um nó seja dividido.

Número mínimo de amostras por folha (min_samples_leaf): Definido entre 1 e 19, garantindo que cada folha tenha amostras suficientes para evitar sobreajuste.

Número máximo de features consideradas por nó (max_features): Variando entre 1 e 16, permitindo testar diferentes quantidades de atributos na decisão de cada divisão.

Número máximo de folhas (max_leaf_nodes): Testamos entre 2 e 19 para controlar o crescimento da árvore e evitar divisões excessivas.

Peso das classes (class_weight):

- Balanced: Ajusta automaticamente os pesos das classes inversamente proporcionais às suas frequências no conjunto de treinamento.
- None: Mantém pesos iguais para todas as classes.

O modelo final foi selecionado com base no f1-score no conjunto de validação.

Os hiperparâmetros vencedores foram:

- Critério de divisão: Gini
- Estratégia de divisão: Best
- Número mínimo de amostras para divisão: 15
- Número mínimo de amostras por folha: 3
- Número máximo de folhas: 13
- Número máximo de features por nó: 14
- Profundidade máxima da árvore: 8
- Peso das classes: None

Random Forest

A técnica de Random Forest combina múltiplas árvores de decisão para reduzir o sobreajuste e melhorar a capacidade de generalização. A busca pelos hiperparâmetros ideais foi realizada por RandomizedSearchCV, testando diferentes configurações para equilibrar desempenho e eficiência computacional. Os hiperparâmetros ajustados foram:

Número de árvores (n_estimators): Variando entre 10 e 150, com incrementos de 10, para avaliar o impacto da quantidade de estimadores na estabilidade do modelo.

Critério de divisão (criterion): Define a métrica para avaliar a qualidade da separação, considerando:

Gini: Mede a impureza dos nós.

Entropy: Baseada na teoria da informação.

Log Loss: Variante que utiliza perda logística para classificação probabilística.

Uso de amostragem com reposição (bootstrap):

True: Cada árvore recebe uma amostra aleatória com reposição dos dados.

False: Cada árvore recebe uma amostra sem reposição, aumentando a diversidade.

Profundidade máxima da árvore (max_depth): Testamos valores entre 1 e 16, equilibrando expressividade e generalização.

Número mínimo de amostras para divisão (min_samples_split): Variando entre 2 e 19, garantindo que os nós só sejam divididos se houver amostras suficientes.

Número mínimo de amostras por folha (min_samples_leaf): Definido entre 1 e 19, evitando folhas com poucos dados, o que poderia levar ao sobreajuste.

Número máximo de features por nó (max_features): Entre 1 e 16, determinando quantos atributos são considerados para cada divisão.

Número máximo de folhas (max_leaf_nodes): Variando entre 2 e 19, limitando o número de nós terminais da árvore.

Peso das classes (class_weight):

Balanced: Ajusta pesos automaticamente com base na distribuição das classes.

None: Mantém pesos iguais para todas as classes.

Após a busca, o modelo com melhor f1-score foi escolhido para compor o conjunto final. O processo levou 10 minutos e 22 segundos para ser concluído.

Os hiperparâmetros vencedores foram:

Número de estimadores: 130

Critério de divisão: Gini

Uso de amostragem com reposição: False

Profundidade máxima da árvore: 14

Número mínimo de amostras para divisão: 17

Número mínimo de amostras por folha: 13

Número máximo de folhas: 16

Número máximo de features por nó: 7

Peso das classes: None

XGBoost

O XGBoost é um dos algoritmos mais eficientes para problemas de aprendizado supervisionado, combinando velocidade e desempenho. Para ajustar seus hiperparâmetros, utilizamos RandomizedSearchCV, explorando diferentes configurações para otimizar o equilíbrio entre viés e variância. Os hiperparâmetros ajustados foram:

Número de estimadores (n_estimators): Testamos valores entre 5 e 100, buscando encontrar um número adequado de árvores para evitar overfitting ou underfitting.

Número máximo de folhas (max_leaves): Variamos entre 0 e 100, permitindo a criação de estruturas mais complexas para capturar padrões nos dados.

Critério de divisão (criterion):

- Gini: Mede a impureza dos nós.
- Entropy: Baseada na teoria da informação.
- Log Loss: Variante que utiliza perda logística para classificação probabilística.

Política de crescimento da árvore (grow_policy):

- Depthwise: Cresce as árvores até uma profundidade máxima fixa.
- Lossguide: Expande os nós com base na redução da perda, criando árvores mais balanceadas.

Tipo de booster (booster):

- gbtree: Modelo baseado em árvores, adequado para a maioria dos problemas.
- gblinear: Utiliza uma abordagem linear para aprendizado mais simples e rápido.

dart: Variante com dropout para evitar overfitting.

Profundidade máxima da árvore (max_depth): Valores testados entre 1 e 16, equilibrando expressividade e generalização.

Após a busca, o modelo com melhor f1-score foi escolhido para compor o conjunto final. O processo levou 3 minutos e 17 segundos para ser concluído.

Os hiperparâmetros vencedores foram:

Número de estimadores: 43

Número máximo de folhas: 15

Profundidade máxima da árvore: 6

Política de crescimento: lossguide

Critério de divisão: entropy

Tipo de booster: dart

LightGBM

O LightGBM é uma implementação eficiente de gradient boosting, projetada para lidar com grandes volumes de dados e alta dimensionalidade, mantendo alta precisão. Utilizamos RandomizedSearchCV para otimizar os hiperparâmetros, buscando a melhor configuração para nosso modelo. Os hiperparâmetros ajustados foram:

Tipo de boosting (boosting_type):

gbdt: O método tradicional de gradient boosting, que se mostrou eficaz em muitos problemas.

dart: Variante com dropout para reduzir overfitting, similar ao XGBoost.

rf: Baseado em florestas aleatórias, adequado para problemas onde árvores individuais não são suficientes.

Número de estimadores (n_estimators): Variamos entre 50 e 250, com incrementos de 25, para avaliar o impacto do número de árvores no desempenho do modelo.

Taxa de aprendizado (learning_rate): Valores variando de 0.01 a 0.15, para ajustar a velocidade com que o modelo se adapta aos dados.

Número de folhas (num_leaves): Testamos valores entre 5 e 100, controlando a complexidade da árvore para evitar overfitting.

Profundidade máxima da árvore (max_depth): Variando entre 1 e 9, para limitar a complexidade das árvores e melhorar a generalização.

Após a busca, o modelo com melhor f1-score foi escolhido para compor o conjunto final. O processo levou 3 minutos e 49 segundos para ser concluído.

Os hiperparâmetros vencedores foram:

Número de folhas: 75

Número de estimadores: 225

Profundidade máxima da árvore: 7

Taxa de aprendizado: 0.1267

Tipo de boosting: gbdt

Comitê Heterogêneo (Stacking)

O Comitê Heterogêneo foi formado utilizando uma combinação de 4 modelos distintos: Árvore de Decisão, Random Forest, XGBoost e LightGBM. Esses modelos foram os melhores obtidos em buscas de hiperparâmetros anteriores e foram combinados de maneira a explorar a diversidade entre diferentes abordagens de aprendizado.

A busca pelos melhores parâmetros para o comitê foi realizada utilizando RandomizedSearchCV. A estratégia de busca envolveu a combinação de 2 a 4 modelos diferentes, testando diferentes formas de agregação de suas predições com a técnica de votação:

- Votação suave (soft): A predição final é determinada pela média das probabilidades de cada classe atribuída pelos modelos individuais.
- Votação dura (hard): A predição final é determinada pela maioria das predições feitas pelos modelos.

O RandomizedSearchCV foi configurado para rodar 10 vezes, cada uma com n_iter=2 para testar as diferentes combinações e os tipos de votação para cada conjunto de modelos gerados. O processo levou 2 minutos e 45 segundos para ser concluído.

Após a busca, o melhor comitê foi selecionado com base no desempenho f1-score, combinando os melhores modelos e o tipo de votação que mais contribuiu para a melhoria da precisão final.

O comitê heterogêneo selecionado foi formado pelos seguintes modelos:

- Árvore de Decisão
- Random Forest

XGBoost

Método de votação: Soft

Comitê de Redes Neurais

O Comitê de Redes Neurais foi formado a partir de uma lista de 10 MLPs (Multi-Layer Perceptron) geradas aleatoriamente, com a definição dos parâmetros baseados em um intervalo de valores para cada hiperparâmetro. Utilizamos RandomizedSearchCV para otimizar a combinação de diferentes redes neurais e suas configurações. Os parâmetros selecionados para cada MLP foram:

hidden_layer_sizes: Definimos diferentes configurações de camadas ocultas usando random.choice() para selecionar entre:

(50,)

(100,)

(50, 50)

activation: Função de ativação escolhida aleatoriamente entre:

ReLU (Rectified Linear Unit)

Tanh (Tangente hiperbólica)

alpha: Valor da regularização L2, escolhido aleatoriamente entre:

0.0001

0.001

0.01

solver: Método de otimização escolhido aleatoriamente entre:

Adam

SGD (Stochastic Gradient Descent)

Após a geração dessas 10 MLPs, realizamos uma busca testando combinações de 2 a 10 redes neurais, variando entre os tipos de votação: soft e hard. O RandomizedSearchCV rodou por 7 minutos e 56 segundos para determinar a melhor configuração do comitê.

As MLPs selecionadas pela busca para o comitê foram:

mlp_0:

Activation: tanh

Alpha: 0.001

Hidden Layers: (50, 50)

Solver: Adam

Early Stopping: Sim

Random State: 4

mlp_1:

Activation: tanh

Alpha: 0.01

Hidden Layers: (50, 50)

Solver: SGD

Early Stopping: Sim

Random State: 2

Método de votação: Soft

O processo envolveu testar diferentes combinações de MLPs e o tipo de votação, e o modelo final foi escolhido com base na melhor pontuação f1-score.

Avaliação do desempenho individual

Após a busca pelos melhores hiperparâmetros, avaliamos o desempenho dos modelos utilizando diversas métricas estatísticas. A escolha dessas métricas visa garantir uma análise completa da capacidade preditiva dos modelos, identificando tanto sua performance geral quanto possíveis desequilíbrios na classificação.

1. Média e Desvio Padrão da Validação Cruzada

Durante o processo de busca de hiperparâmetros com RandomizedSearchCV, utilizamos a validação cruzada estratificada KFold. Esse método divide o conjunto de dados em K partes (folds), garantindo que cada fold tenha proporções semelhantes das classes, preservando a distribuição original dos dados.

Média do f1-score na validação: Representa o desempenho médio do modelo nos diferentes folds da validação cruzada.

Desvio padrão do f1-score: Mede a variação do desempenho entre os folds, indicando consistência ou instabilidade do modelo.

Um desvio padrão baixo indica que o modelo generaliza bem.

Um desvio padrão alto sugere que o modelo pode estar sensível a pequenas mudanças nos dados.

2. Curva de Precisão e Recall & Área sob a Curva (AUC-PR)

A Curva de Precisão-Recall (Precision-Recall Curve) é uma forma alternativa à curva ROC para avaliar modelos, principalmente quando há desequilíbrio nas classes.

Precisão (Precision): Mede a proporção de previsões positivas corretas em relação a todas as previsões positivas feitas pelo modelo.

Recall (Sensibilidade): Mede a proporção de exemplos positivos corretamente identificados pelo modelo.

Área sob a Curva de Precisão-Recall (AUC-PR): Quanto maior essa área, melhor a separação entre classes.

3. Métricas em Conjunto de Treino e Teste

Avaliamos os modelos usando diferentes métricas tanto no conjunto de treino quanto no conjunto de teste. Isso permite identificar overfitting (quando o modelo memoriza os dados de treino, mas generaliza mal para novos dados).

As métricas utilizadas incluem:

Acurácia: Mede a proporção total de classificações corretas. Funciona bem para classes balanceadas, mas pode ser enganosa quando há desequilíbrio.

Precisão (Precision): Mostra a qualidade das previsões positivas do modelo.

Recall (Sensibilidade): Mede a capacidade do modelo de capturar os exemplos positivos.

F1-score: Média harmônica entre Precisão e Recall, útil para conjuntos desbalanceados.

Área sob a Curva ROC (AUC-ROC): Mede a capacidade do modelo em distinguir entre classes. Quanto maior, melhor a separação.

Comparar essas métricas entre treino e teste ajuda a verificar se o modelo está superajustado (overfitting) ou subajustado (underfitting).

4. Matriz de Confusão

A matriz de confusão apresenta os erros e acertos do modelo, dividindo as previsões em quatro categorias, verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. A partir dessa matriz, conseguimos entender onde o modelo está errando e como esses erros afetam a performance.

Muitos Falsos Positivos (FP): Modelo pode ser muito permissivo.

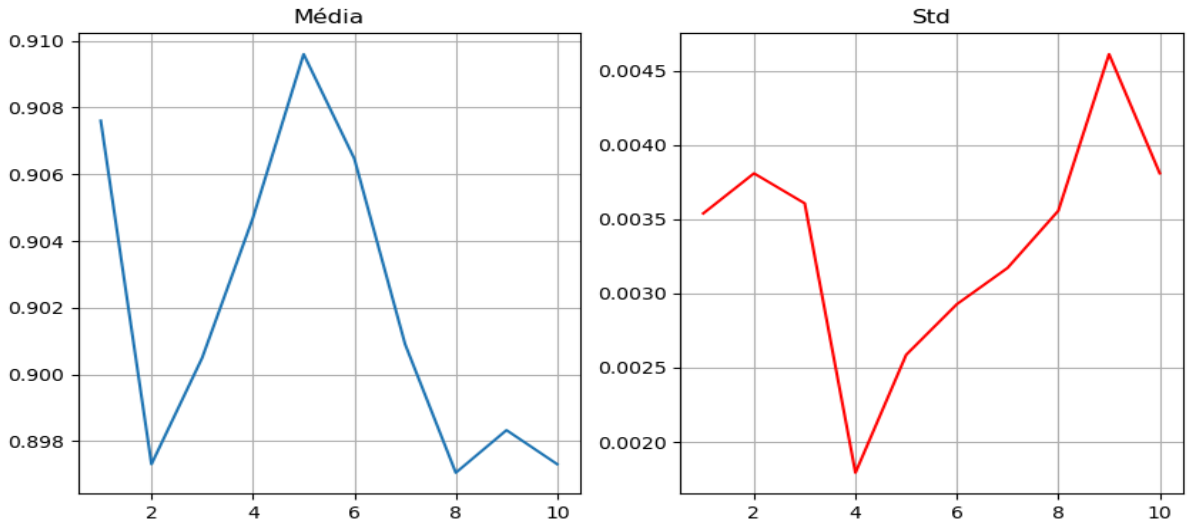
Muitos Falsos Negativos (FN): Modelo pode estar perdendo exemplos importantes.

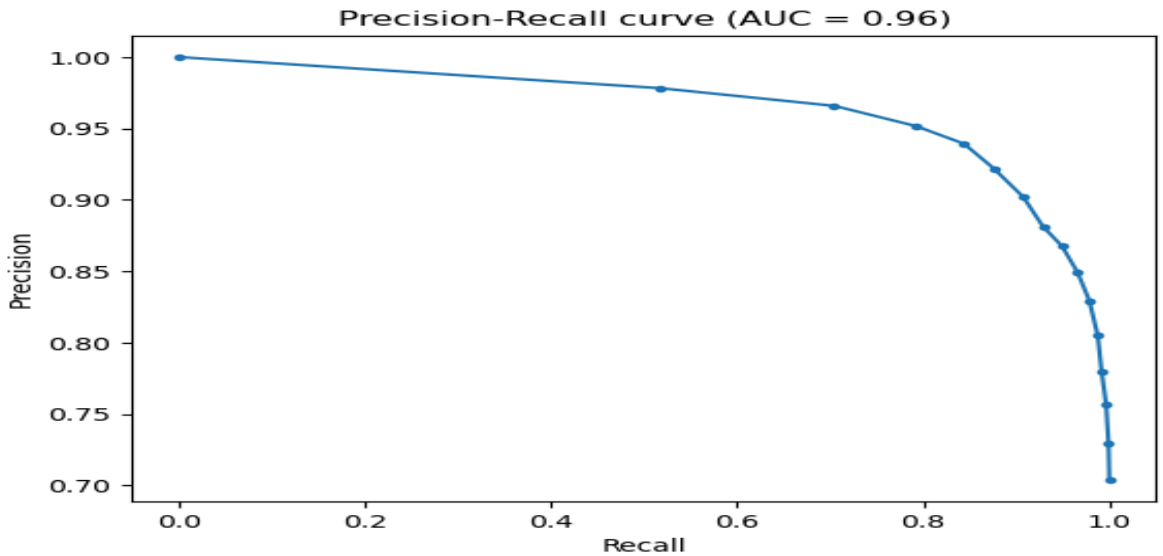
A matriz de confusão é essencial para diagnosticar padrões de erro que as métricas agregadas podem mascarar.

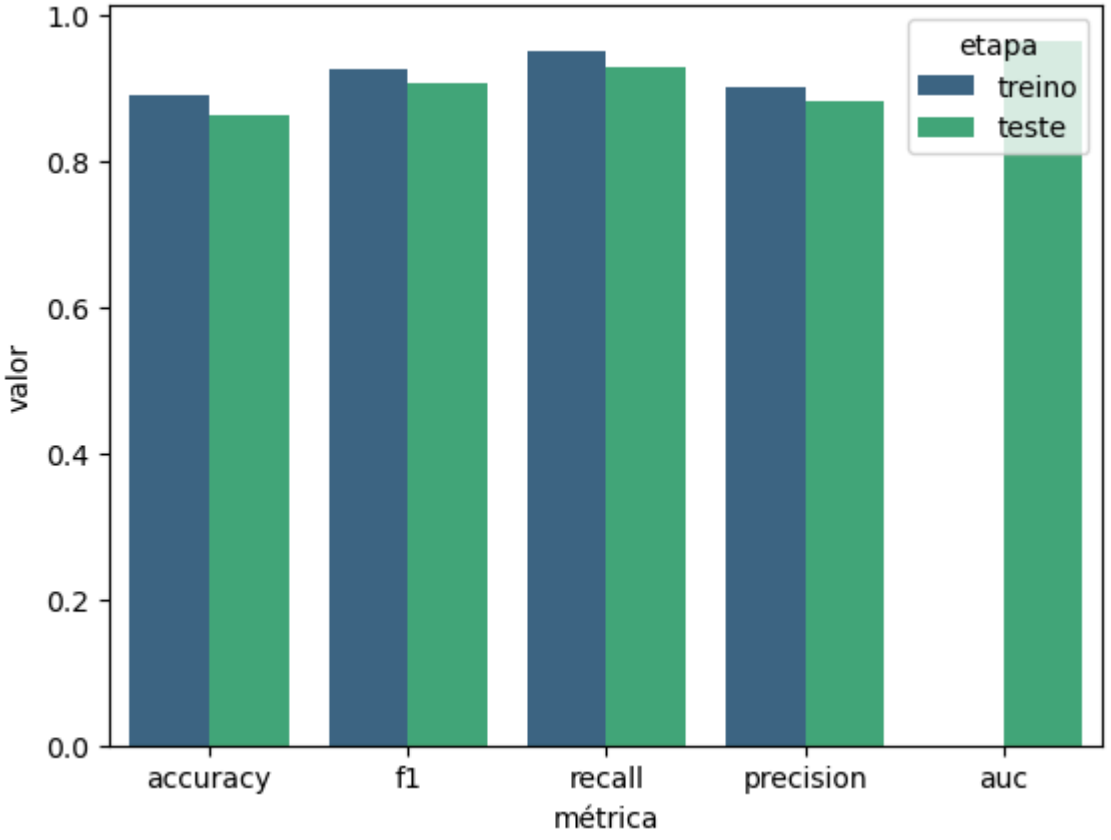
A combinação dessas análises permite avaliar a qualidade geral do modelo, identificar possíveis problemas como overfitting e definir ajustes necessários para melhorar a performance na classificação. Essas quatro etapas compõem o pipeline de avaliação que aplicamos a todos os modelos implementados.

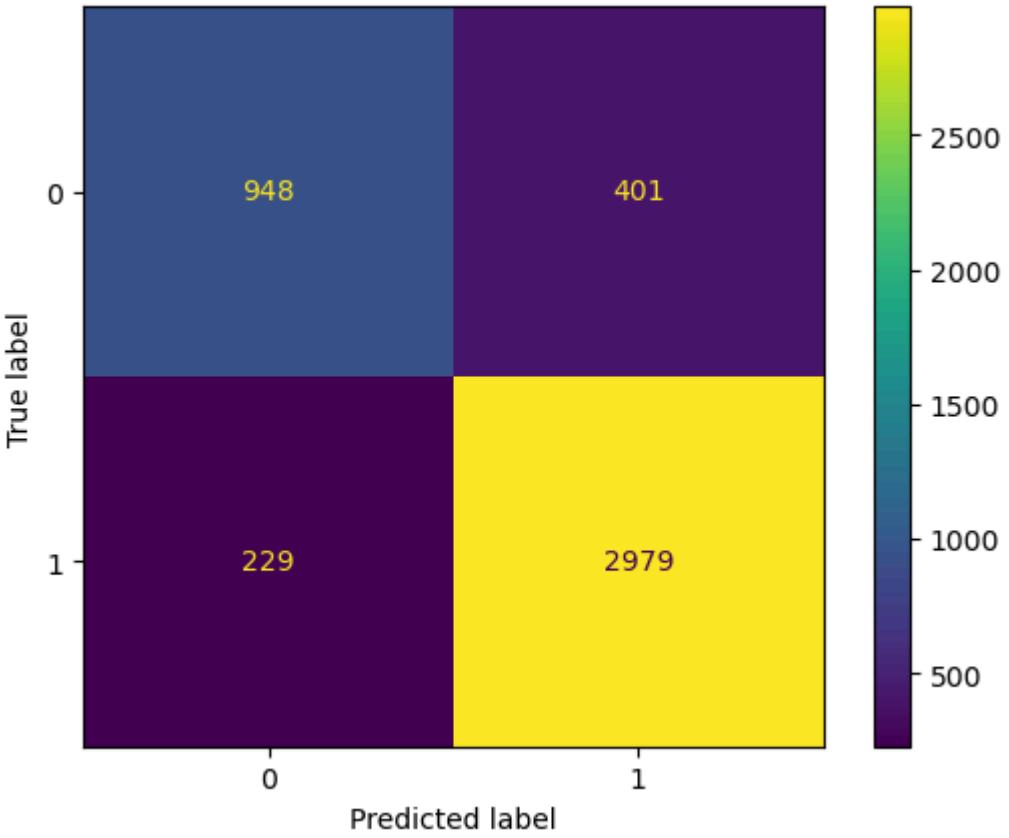
KNN

O KNN apresentou uma acurácia de 88,94% no treino e 86,17% no teste, indicando boa generalização. O f1-score foi de 92,36% no treino e 90,44% no teste, equilibrando bem precisão e recall. O recall atingiu 94,86% no treino e 92,86% no teste, mostrando alta capacidade de recuperar os casos positivos. Já a precisão foi de 89,98% no treino e 88,13% no teste. O AUC em teste ficou em 96,39%, sugerindo um bom poder discriminativo.



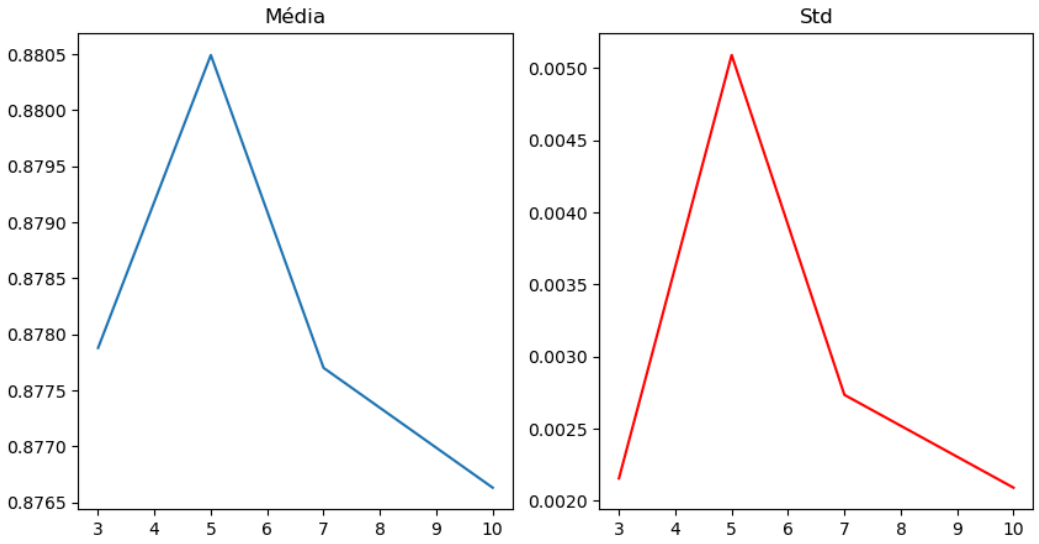


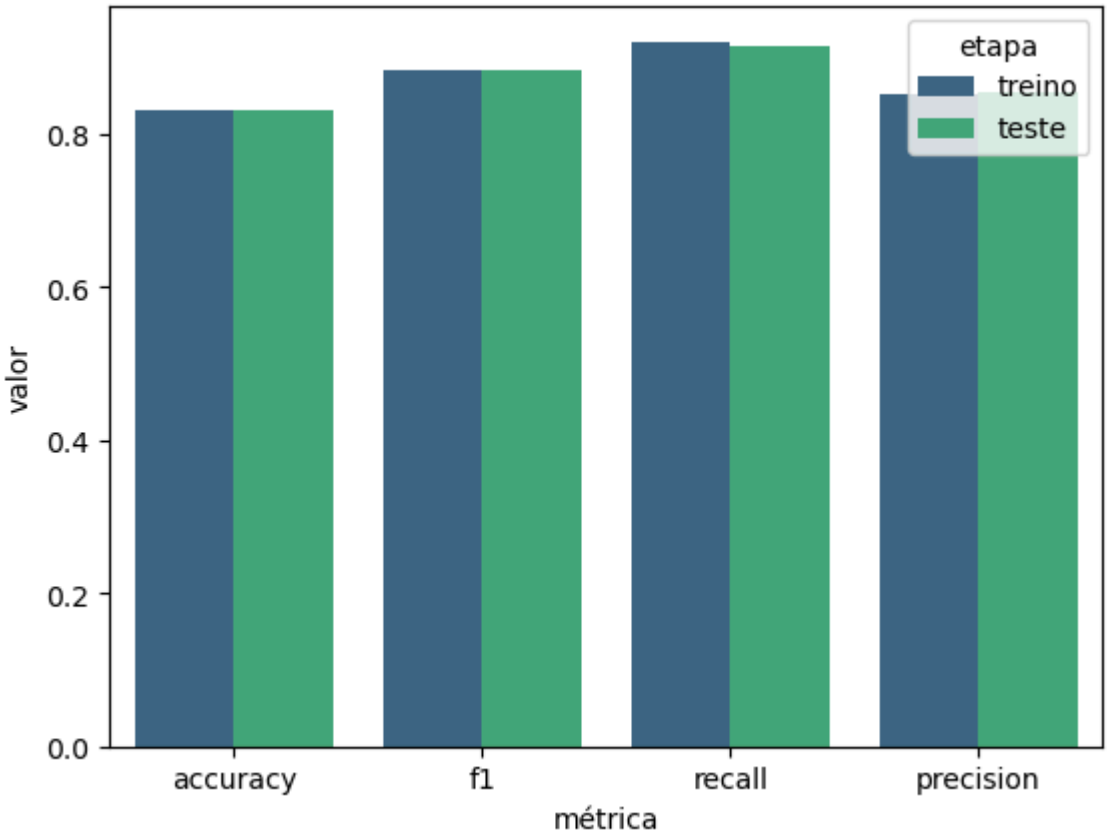

1.0
0.8
0.6
0.4
0.2
0.0
valor
etapa
treino
teste
accuracy
f1
recall
precision
auc
métrica

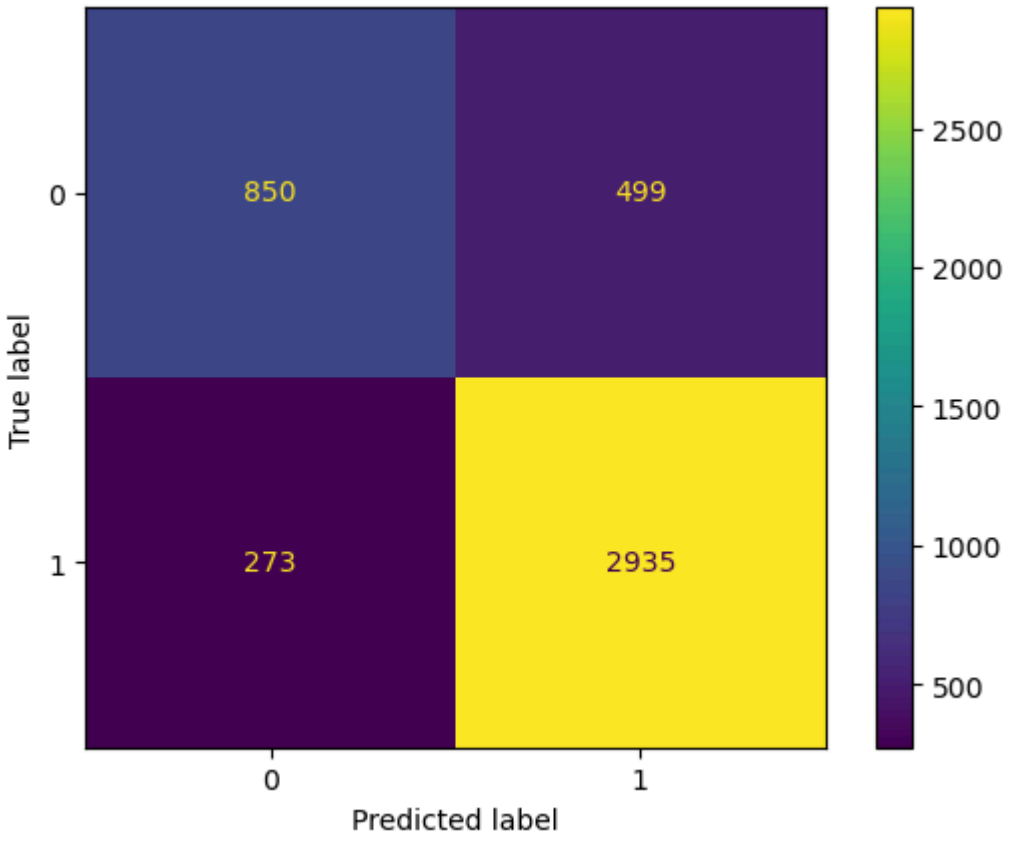

948
401
229
2979
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

SVC

O SVC teve uma acurácia de 82,99% no treino e 83,06% no teste, mantendo estabilidade entre as fases. O f1-score foi de 88,39% no treino e 88,38% no teste, com um recall de 91,99% no treino e 91,49% no teste. A precisão foi de 85,06% no treino e 85,47% no teste, mostrando um modelo confiável sem sinais de overfitting.

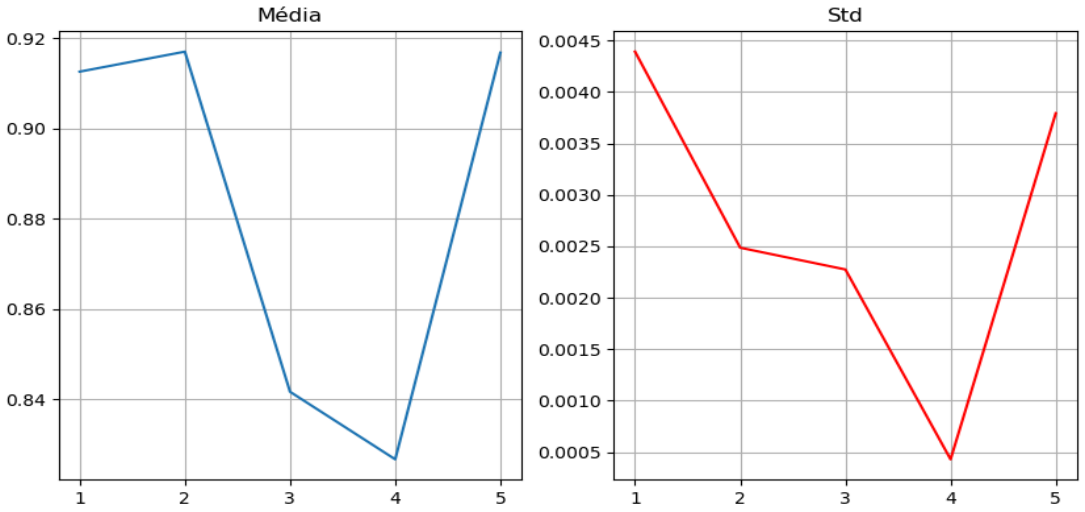


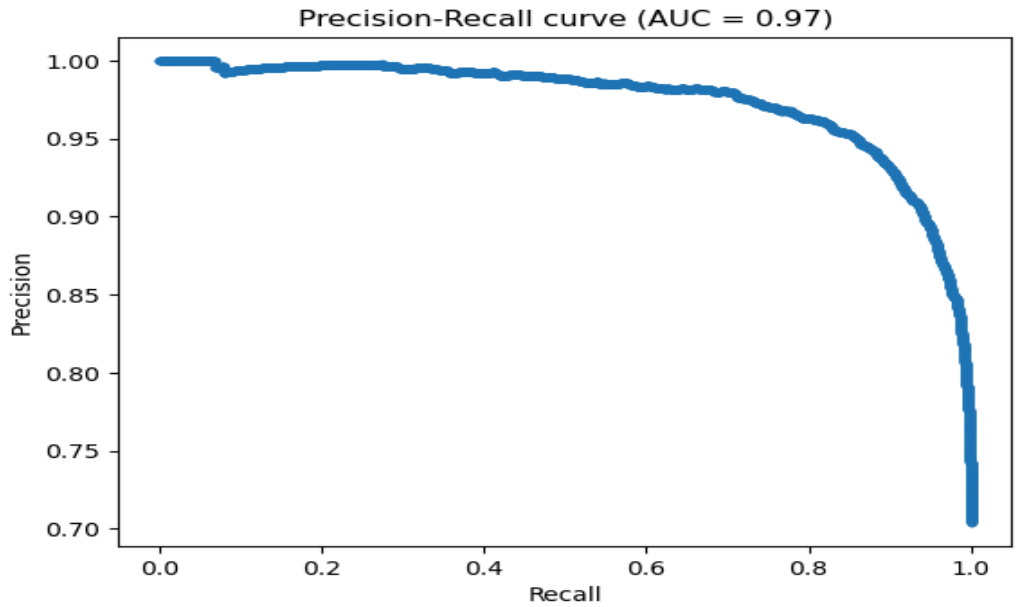

etapa
treino
teste
0.0
0.2
0.4
0.6
0.8
valor
accuracy
f1
recall
precision
métrica

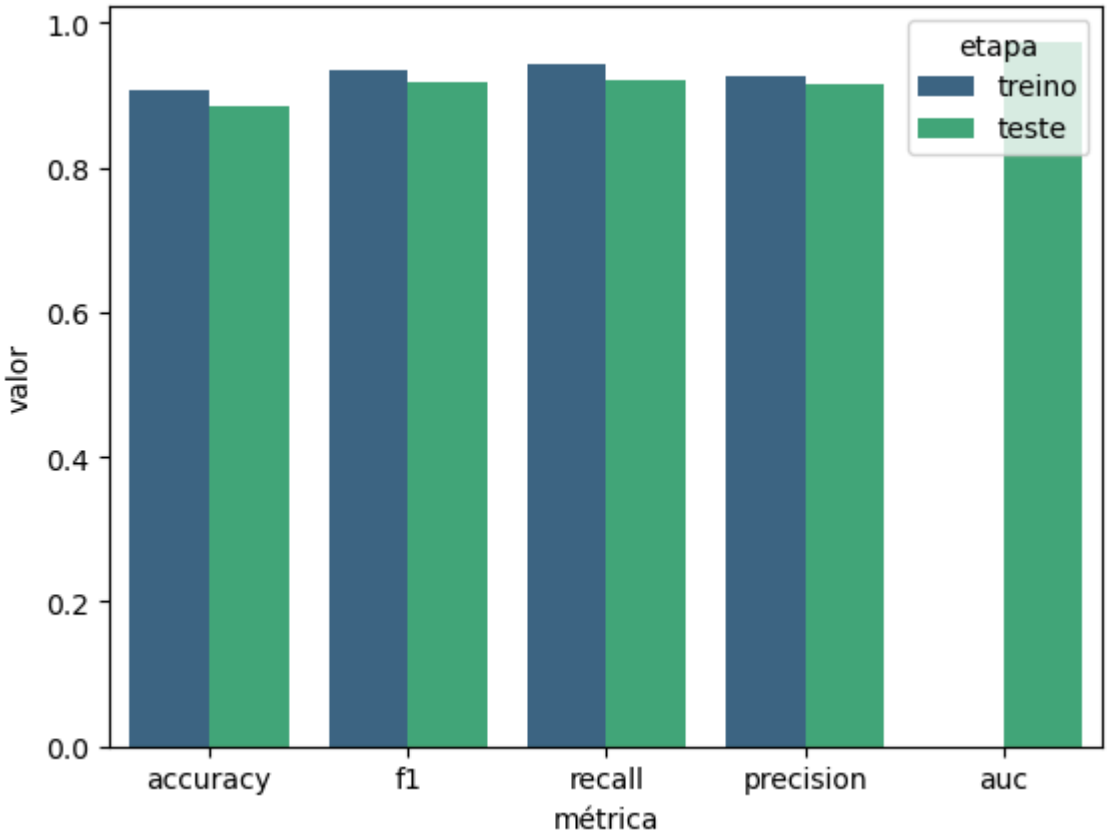

850
499
273
2935
0
1
True label
0
1
Predicted label
500
1000
1500
2000
2500

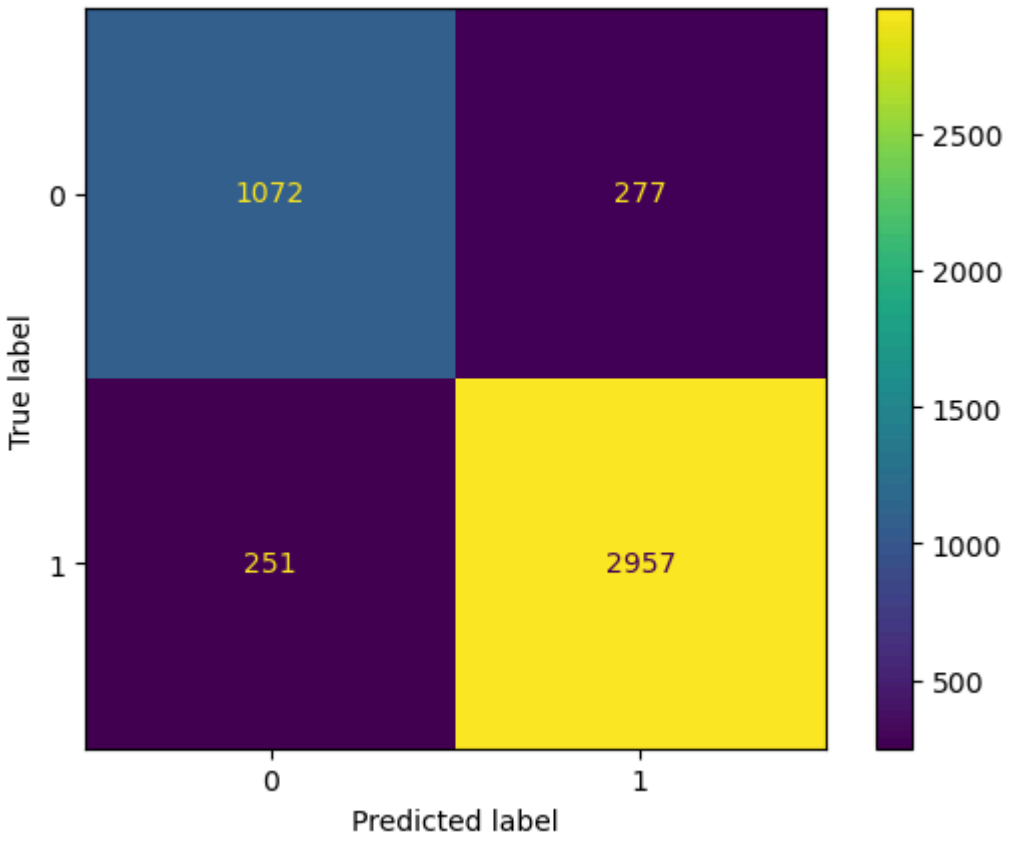
MLP

O MLP apresentou um desempenho forte, com acurácia de 90,68% no treino e 88,41% no teste. O f1-score foi de 93,44% no treino e 91,80% no teste. O recall foi de 94,31% no treino e 92,18% no teste, enquanto a precisão atingiu 92,59% no treino e 91,43% no teste. O AUC foi de 97,44% no teste, reforçando sua boa capacidade preditiva.



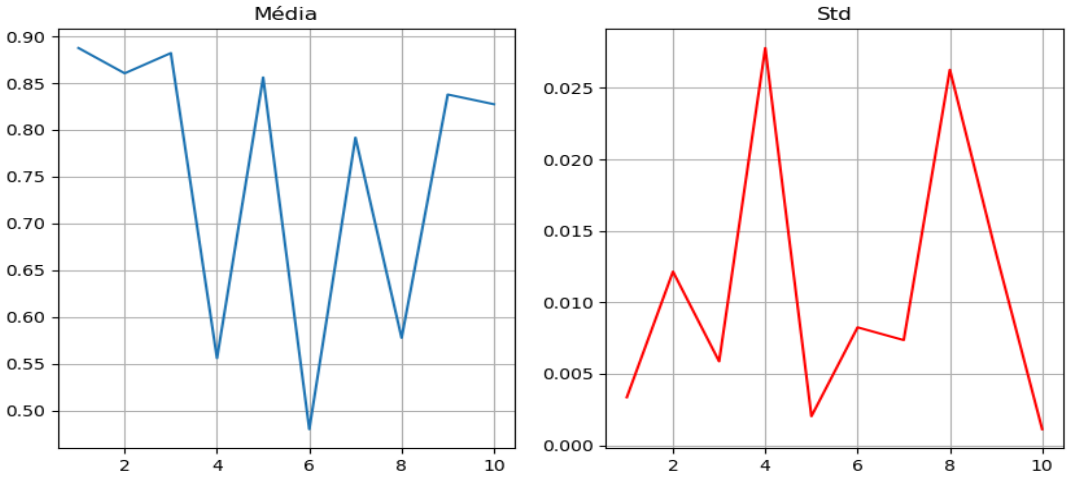


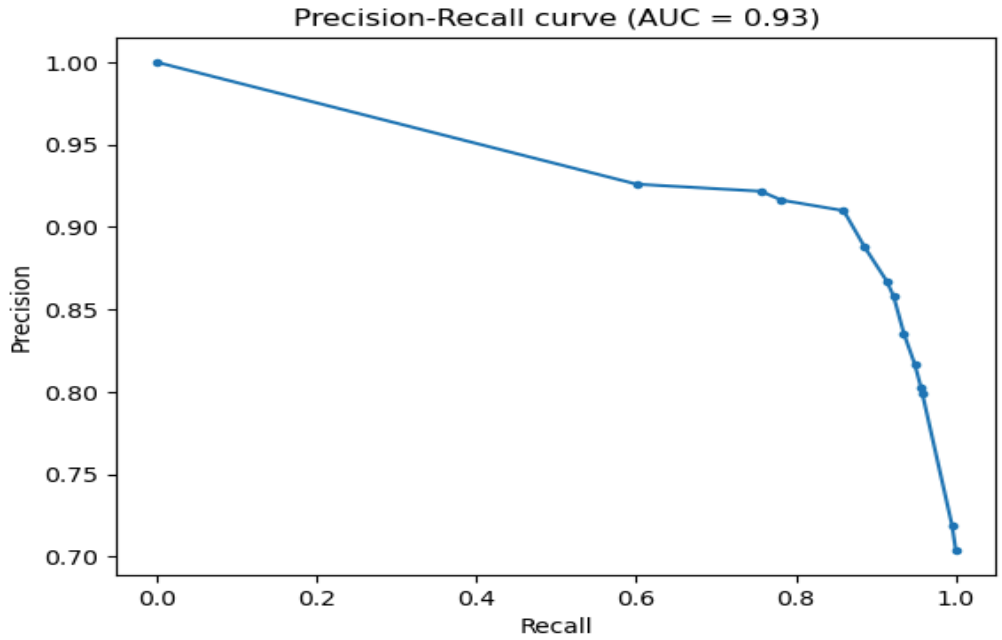

1.0
0.8
0.6
0.4
0.2
0.0
valor
etapa
treino
teste
accuracy
f1
recall
precision
auc
métrica

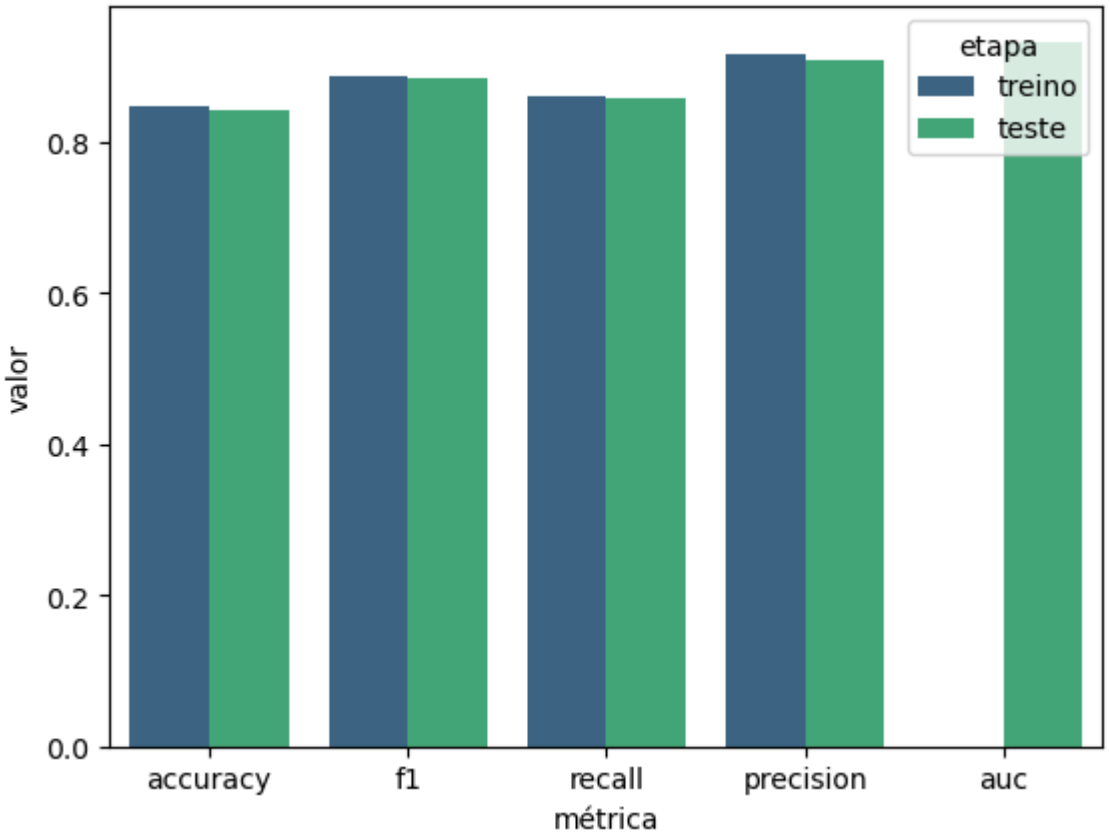

1072
277
251
2957
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

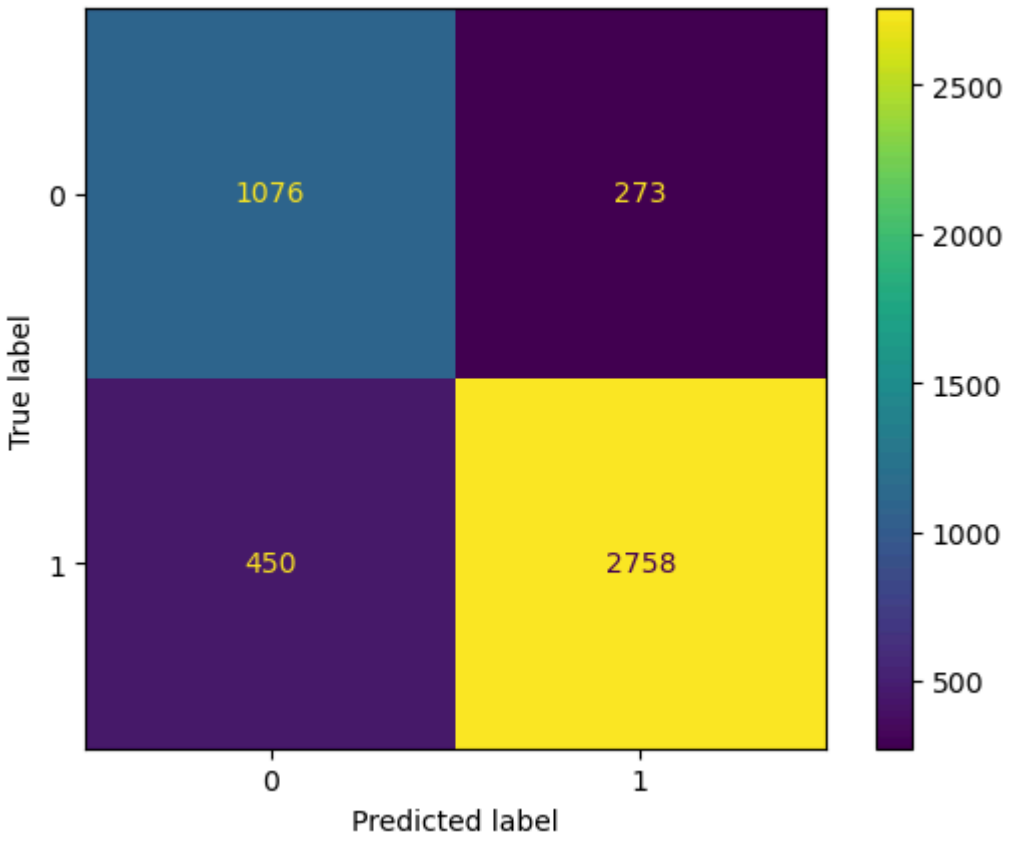
Árvore de decisão

A Árvore de Decisão teve acurácia de 84,75% no treino e 84,13% no teste, com f1-score de 88,84% no treino e 88,41% no teste. O recall ficou em 86,20% no treino e 85,97% no teste, enquanto a precisão foi de 91,64% no treino e 90,99% no teste. O AUC em teste foi de 93,34%.



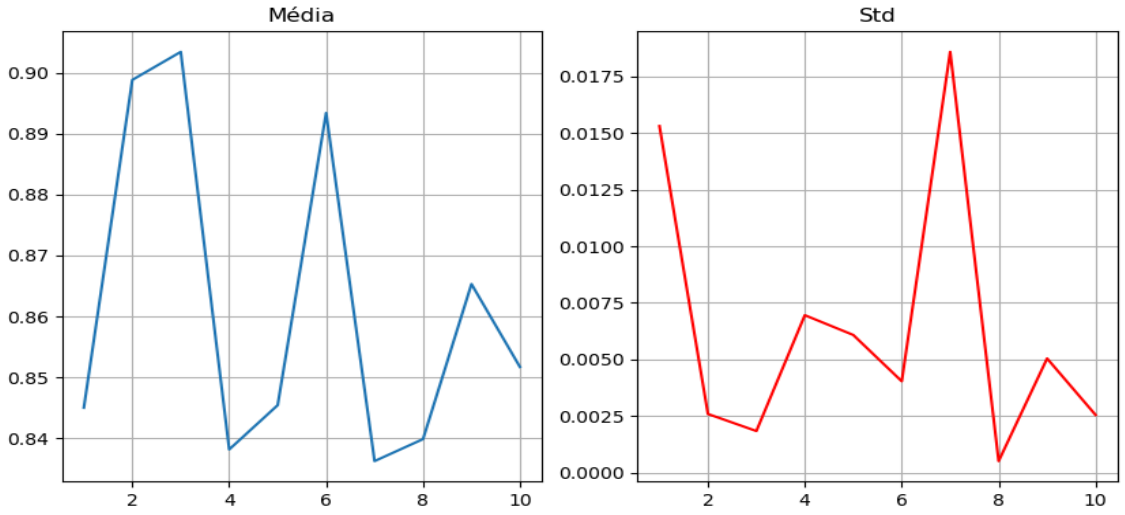


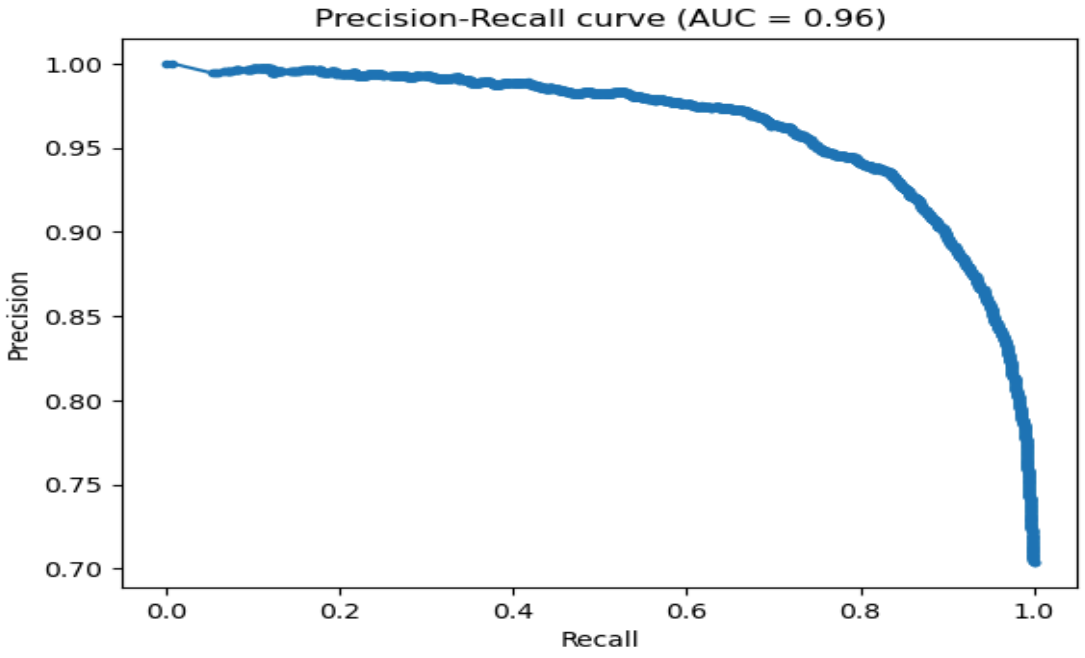

etapa
treino
teste
valor
0.0
0.2
0.4
0.6
0.8
accuracy
f1
recall
precision
auc
métrica

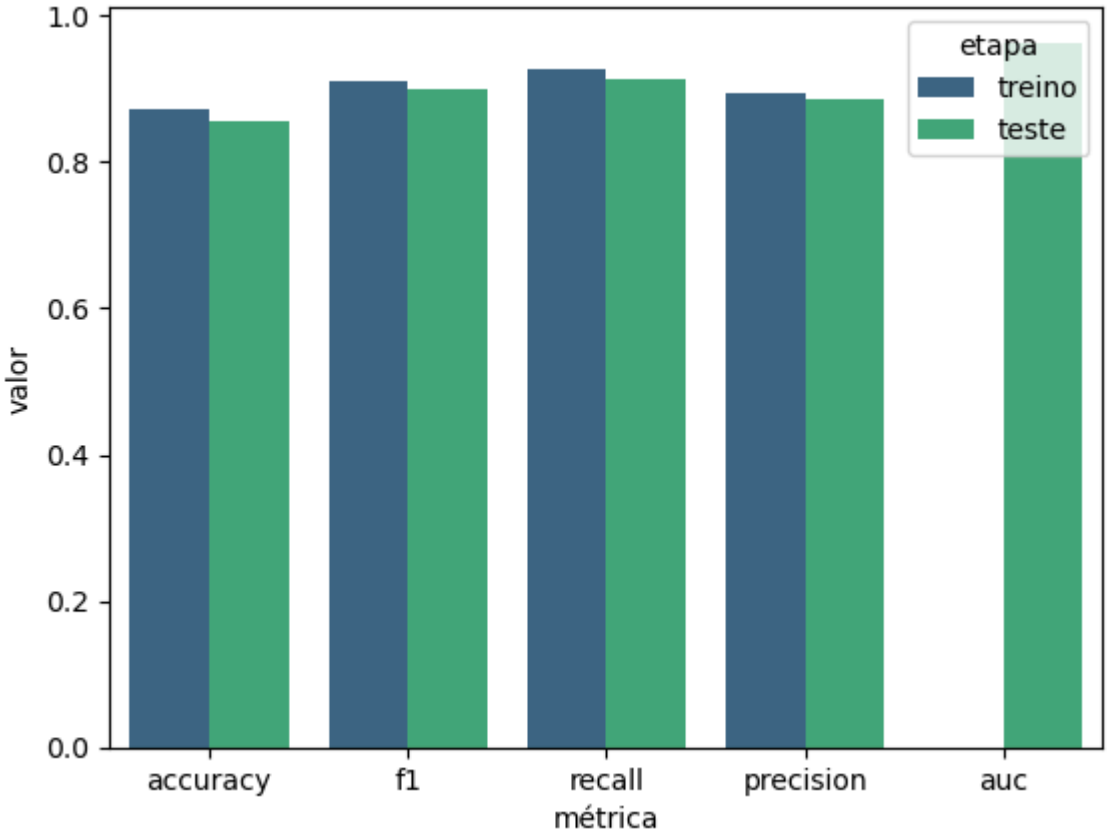

1076
273
450
2758
True label
0
1
Predicted label
0
1
2500
2000
1500
1000
500

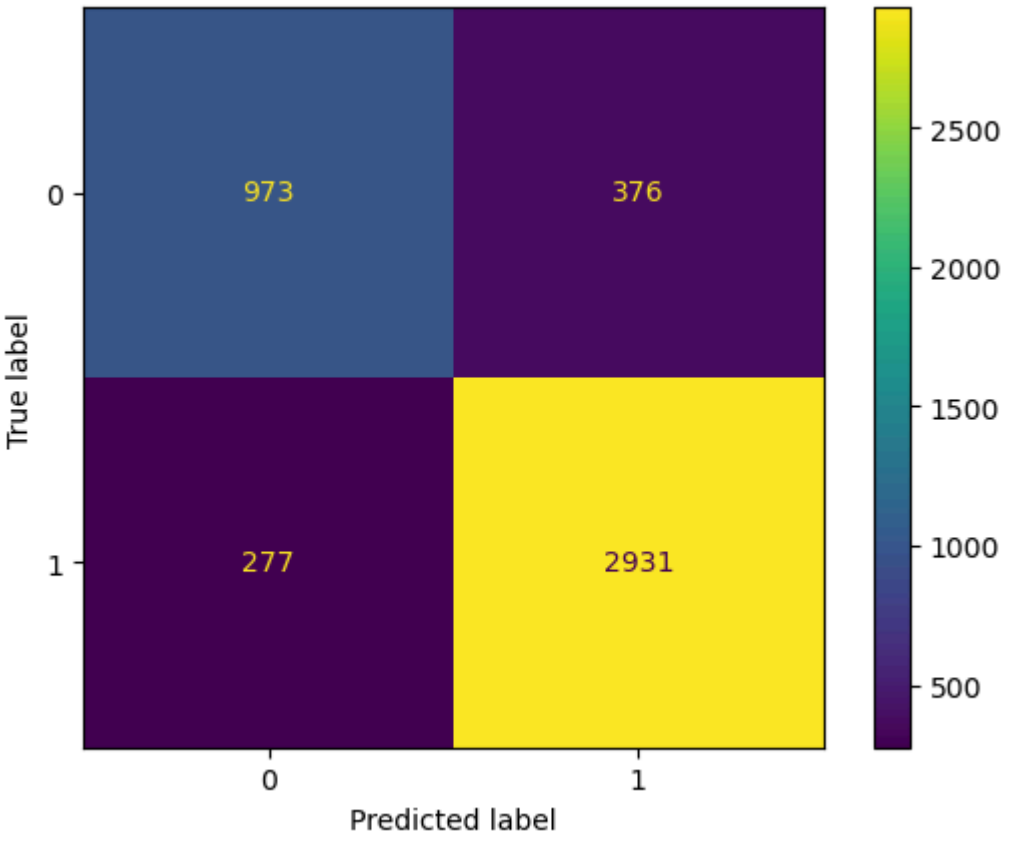
Random Forest

O Random Forest obteve acurácia de 87,13% no treino e 85,67% no teste. O f1-score foi de 91,03% no treino e 89,98% no teste. O recall atingiu 92,69% no treino e 91,37% no teste, enquanto a precisão foi de 89,42% no treino e 88,63% no teste. O AUC ficou em 96,31% no teste.



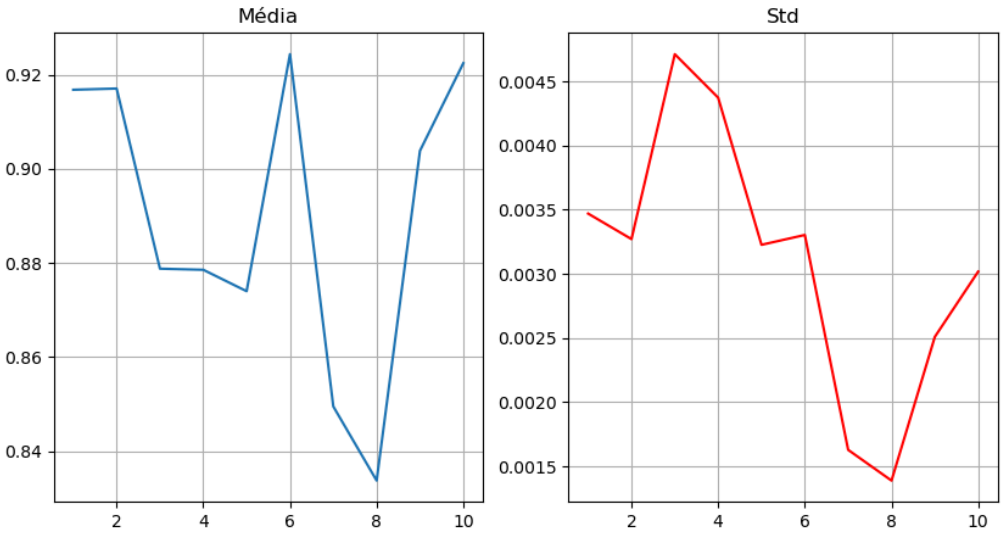


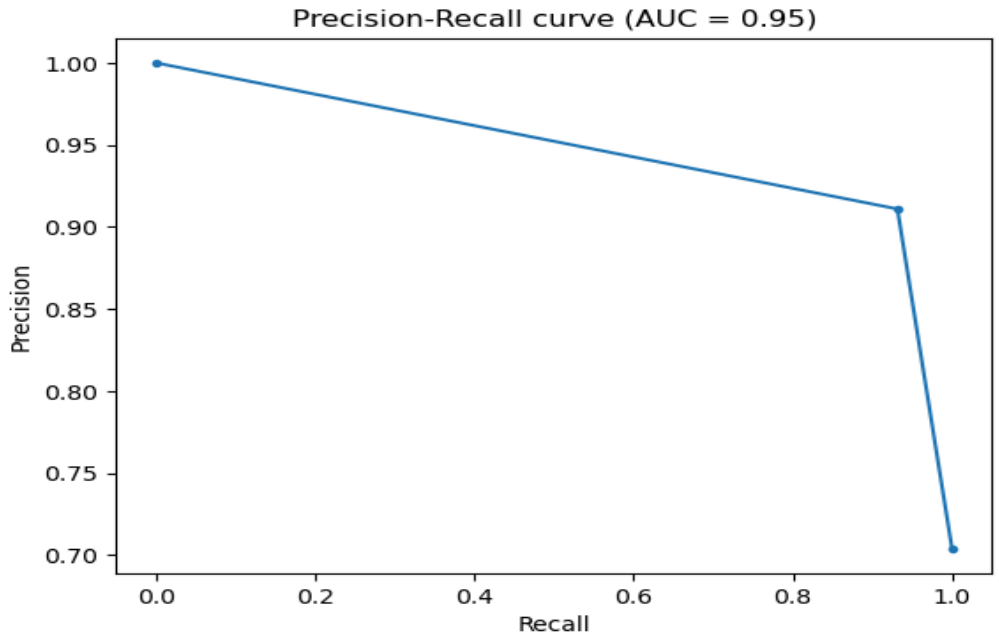

1.0
0.8
0.6
0.4
0.2
0.0
valor
etapa
treino
teste
accuracy
f1
recall
precision
auc
métrica

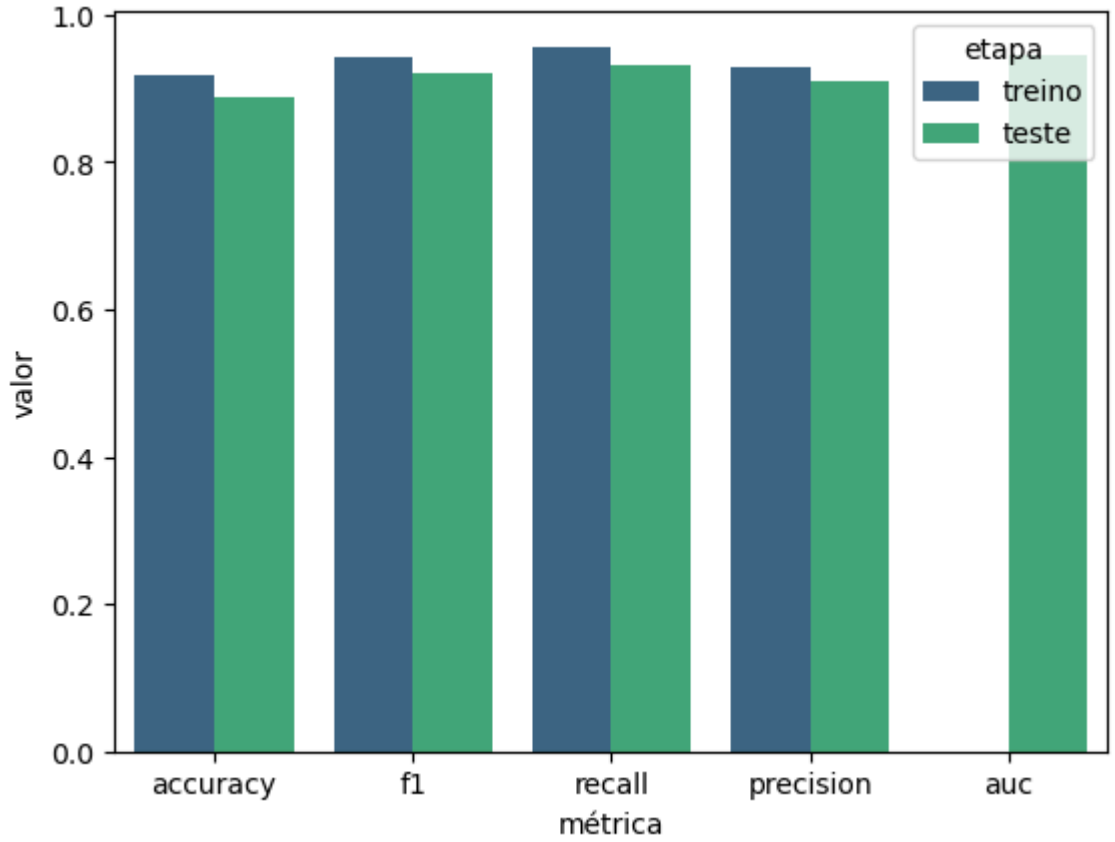

973
376
277
2931
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

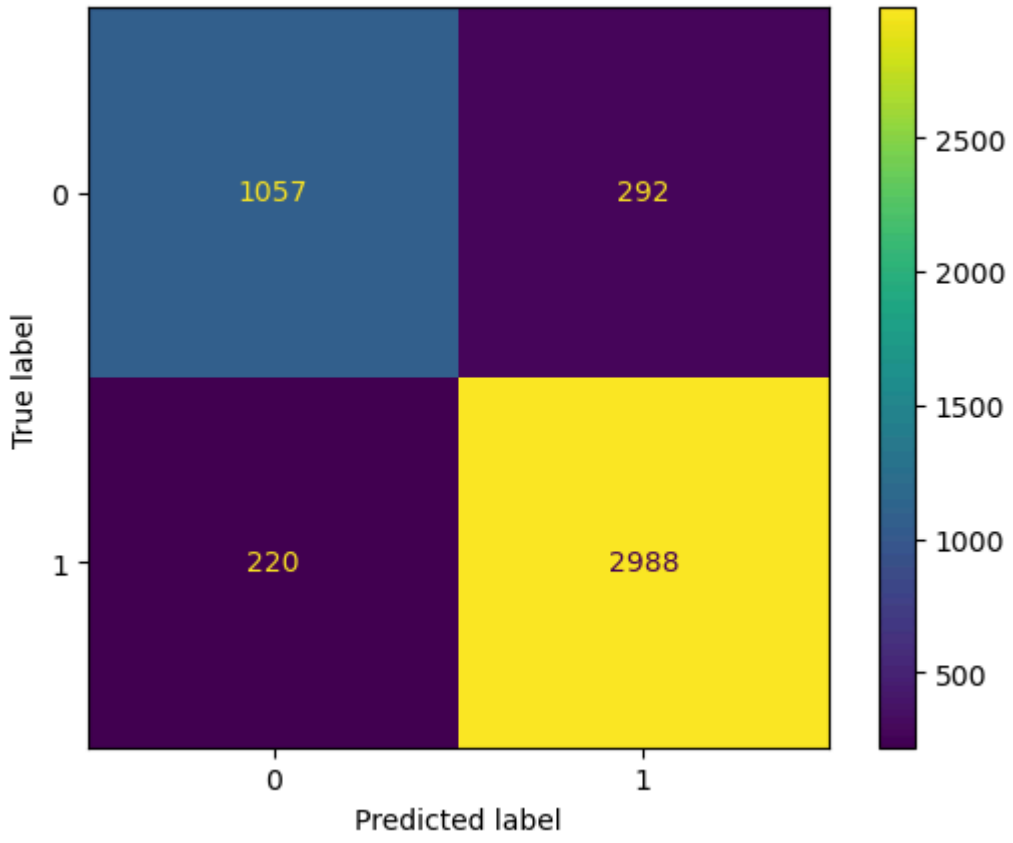
XGBoost

O XGBoost teve acurácia de 91,69% no treino e 88,76% no teste, mostrando alto desempenho. O f1-score foi de 94,19% no treino e 92,11% no teste. O recall atingiu 95,67% no treino e 93,14% no teste, enquanto a precisão foi de 92,76% no treino e 91,10% no teste. O AUC foi de 94,53% no teste.



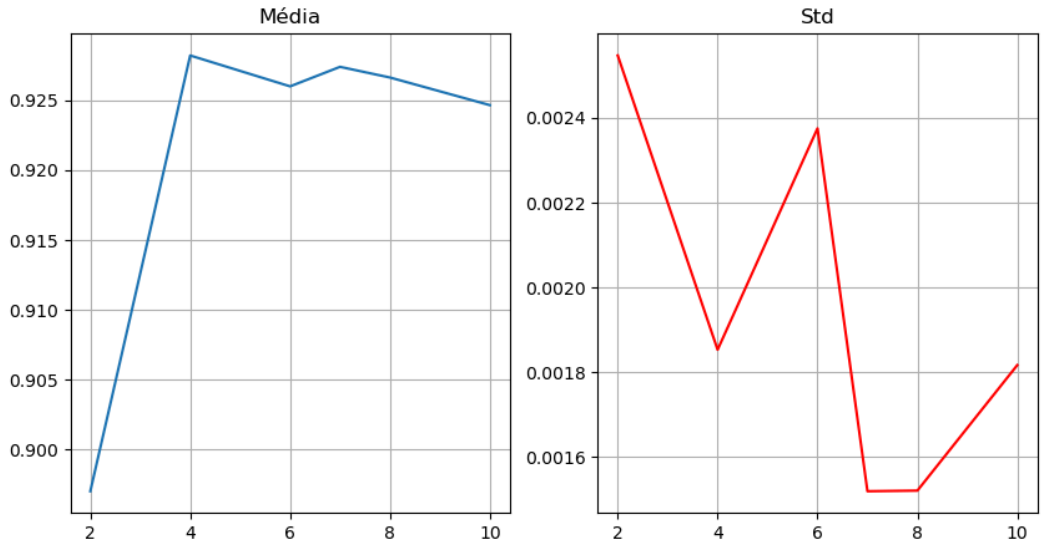


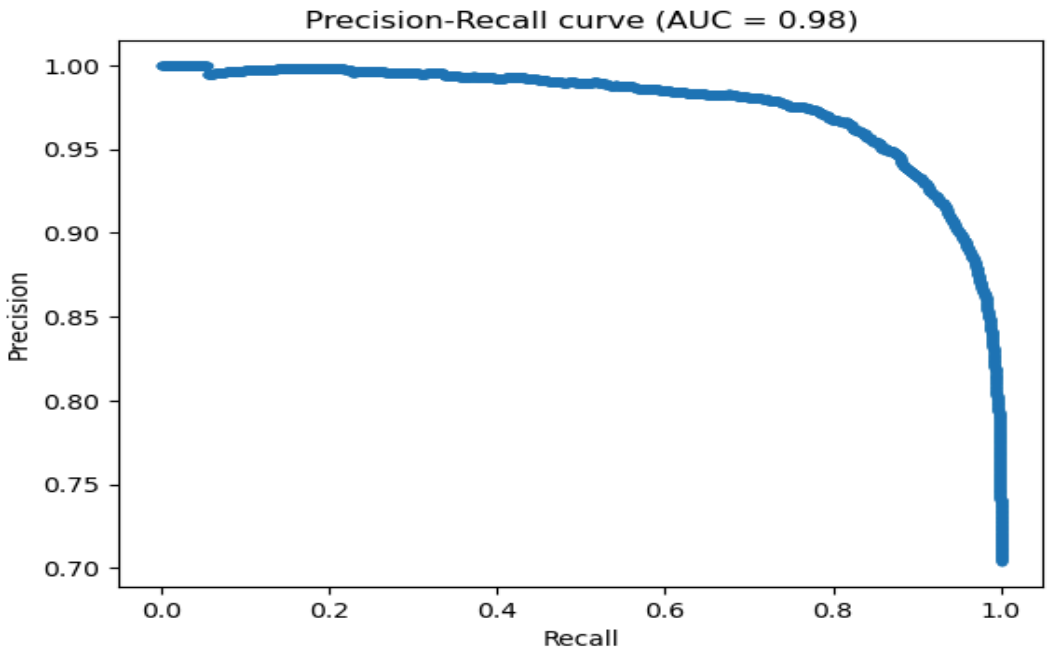

1.0
0.8
0.6
0.4
0.2
0.0
valor
etapa
treino
teste
accuracy
f1
recall
precision
auc
métrica

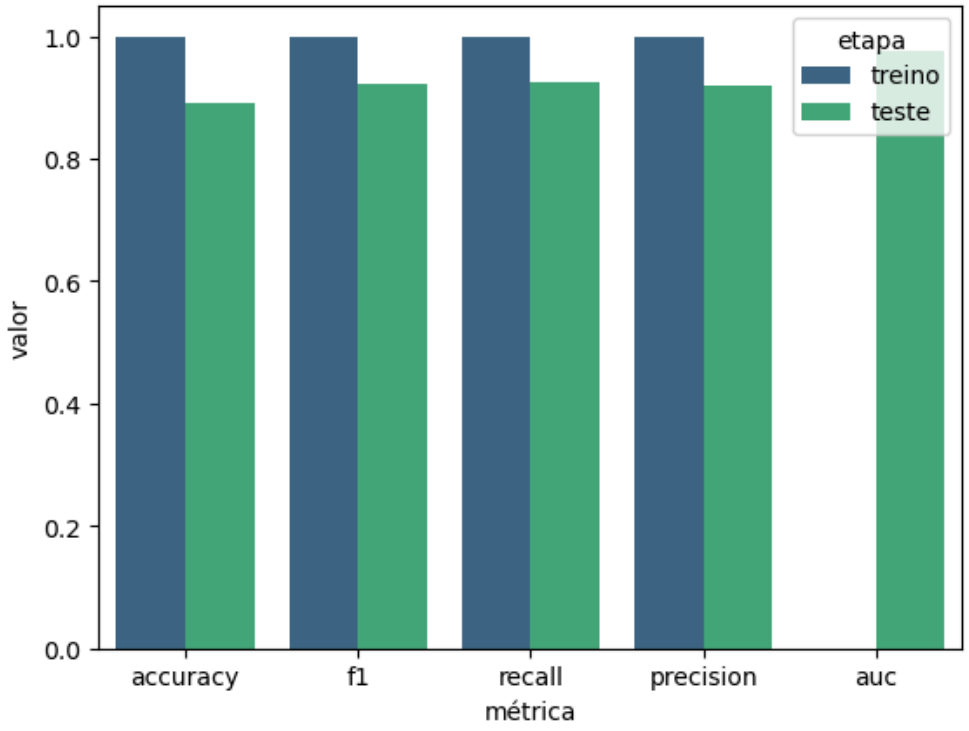

1057
292
220
2988
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

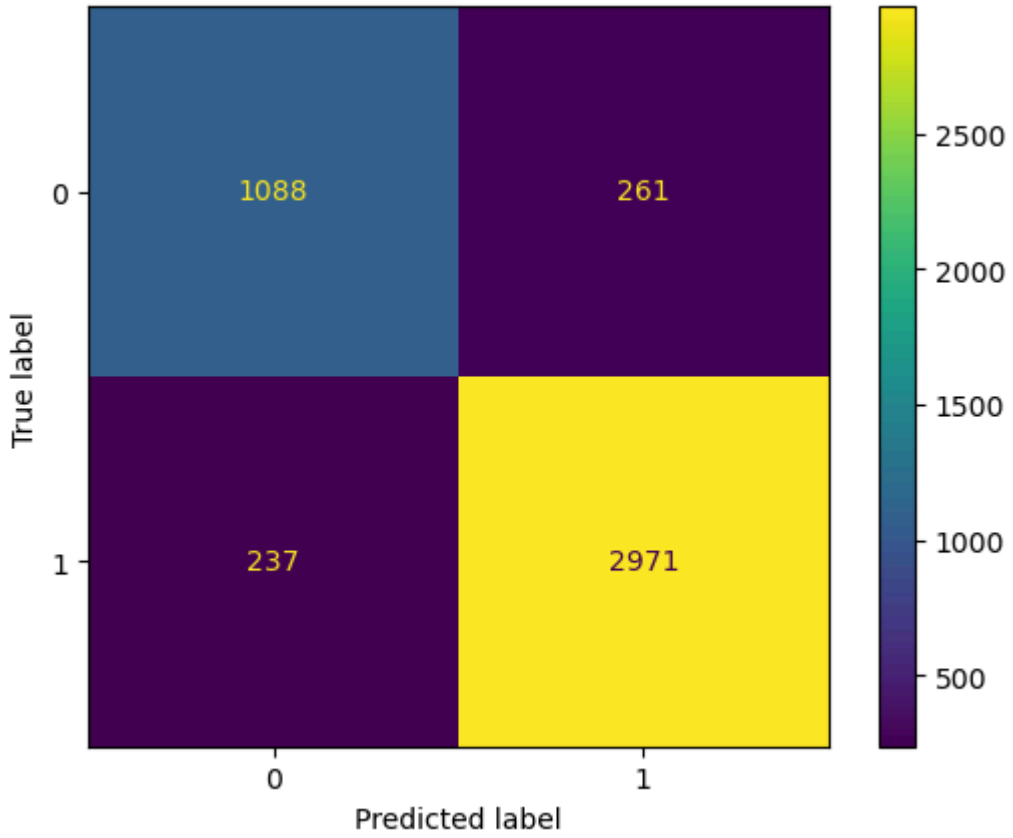
LightGBM

O LightGBM teve a maior acurácia no treino (99,87%), mas caiu para 89,07% no teste, sugerindo um possível overfitting. O f1-score foi de 99,91% no treino e 92,27% no teste. O recall atingiu 99,95% no treino e 92,61% no teste, enquanto a precisão foi de 99,86% no treino e 91,92% no teste. O AUC foi de 97,68% no teste.



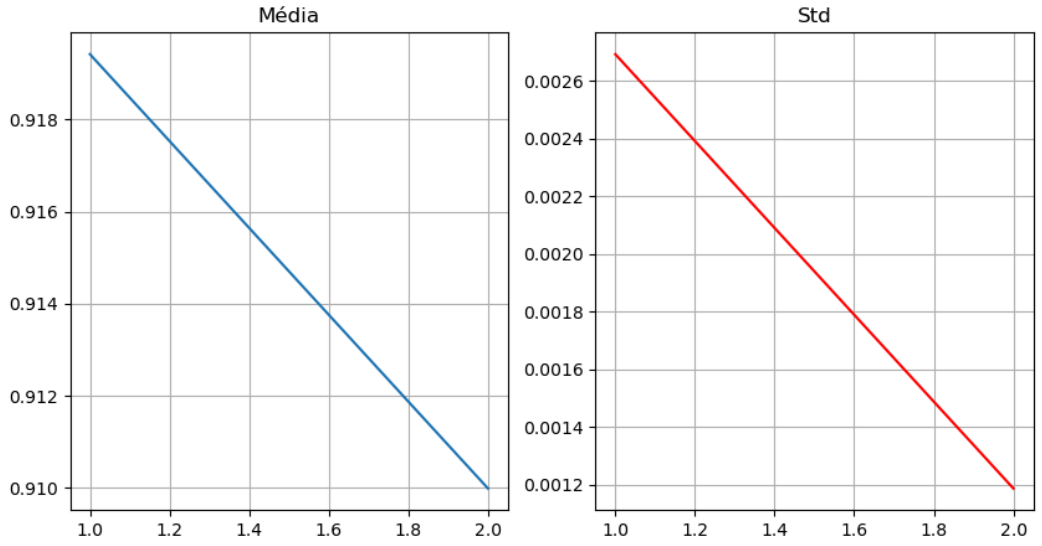


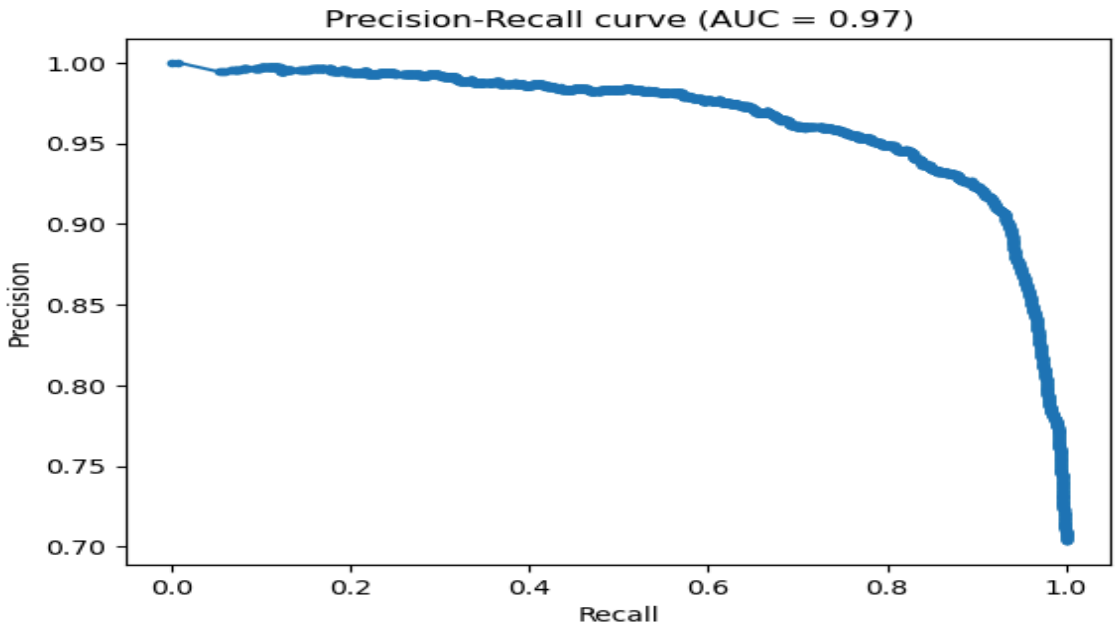

etapa
treino
teste
1.0
0.8
0.6
0.4
0.2
0.0
valor
accuracy
f1
recall
precision
auc
métrica

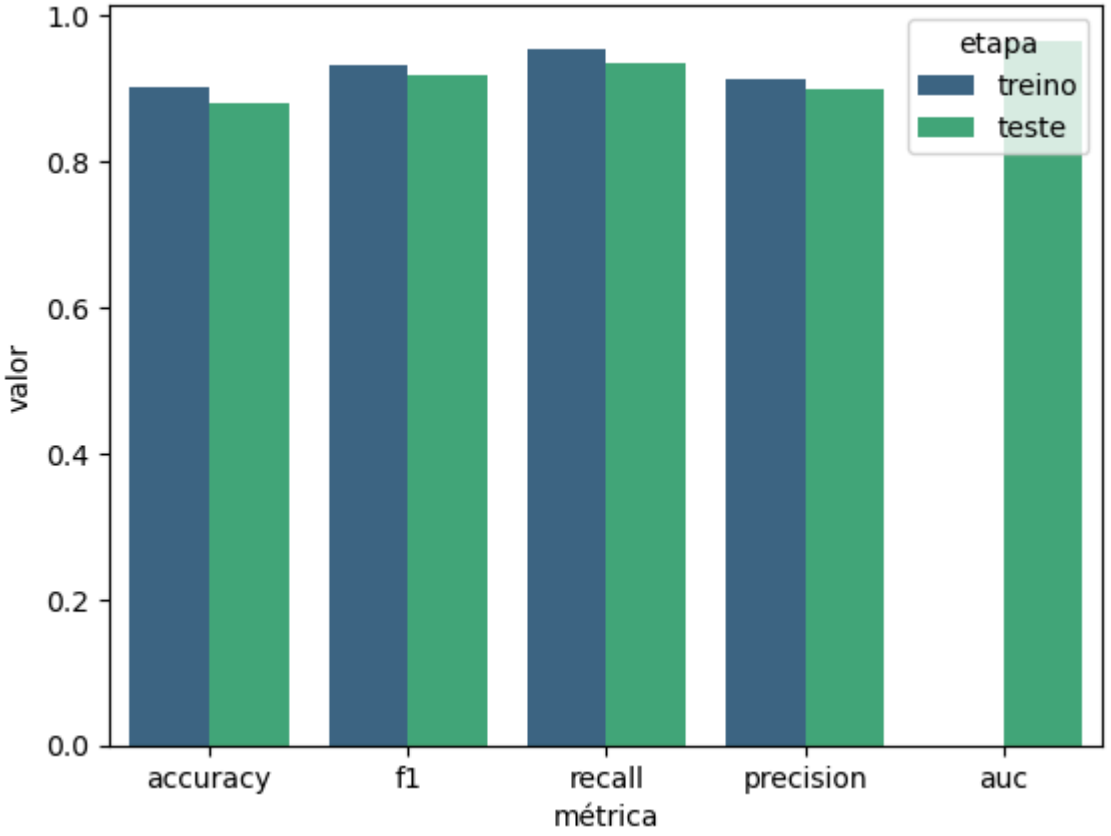

1088
261
237
2971
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

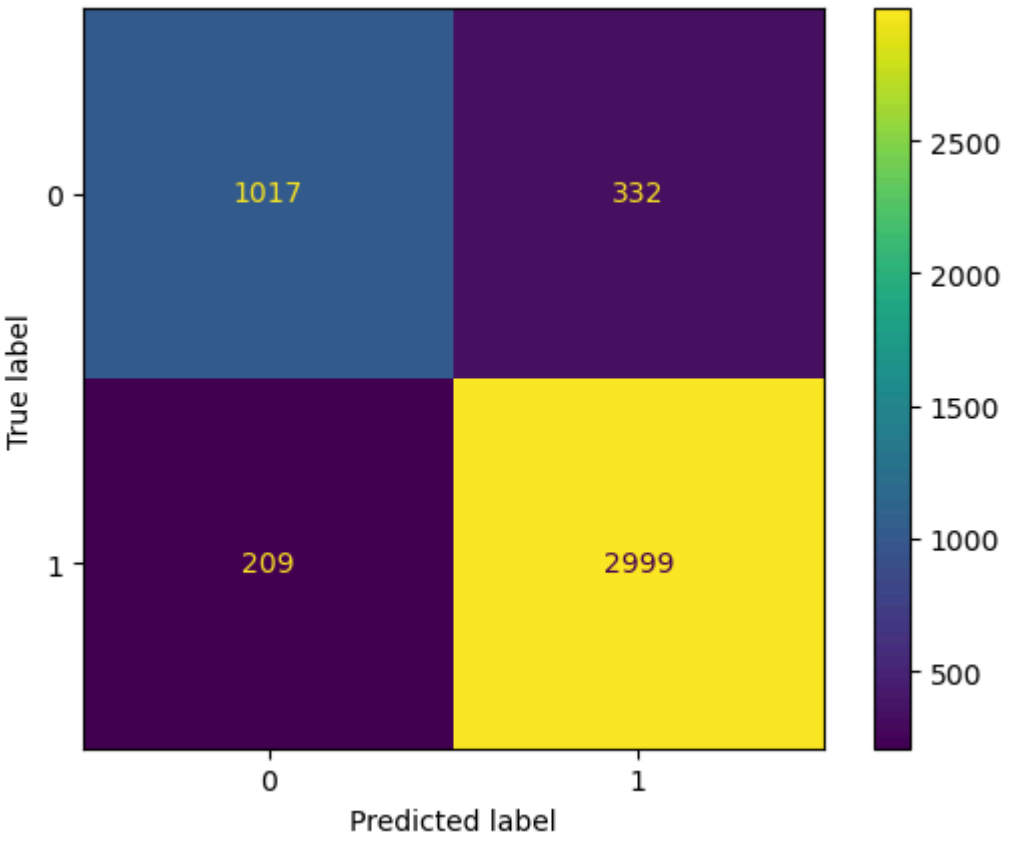
Comitê Heterogêneo (Stacking)

O Comitê Heterogêneo teve um bom equilíbrio entre desempenho e estabilidade, com acurácia de 90,29% no treino e 88,12% no teste. O f1-score foi de 93,26% no treino e 91,73% no teste. O recall atingiu 95,39% no treino e 93,49% no teste, enquanto a precisão foi de 91,22% no treino e 90,03% no teste. O AUC foi de 96,59% no teste.



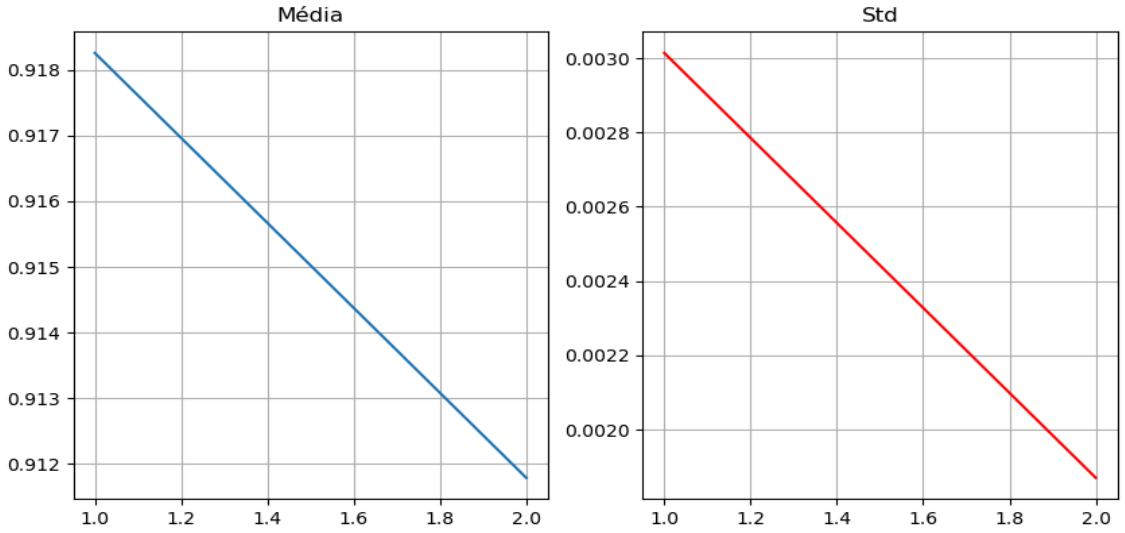


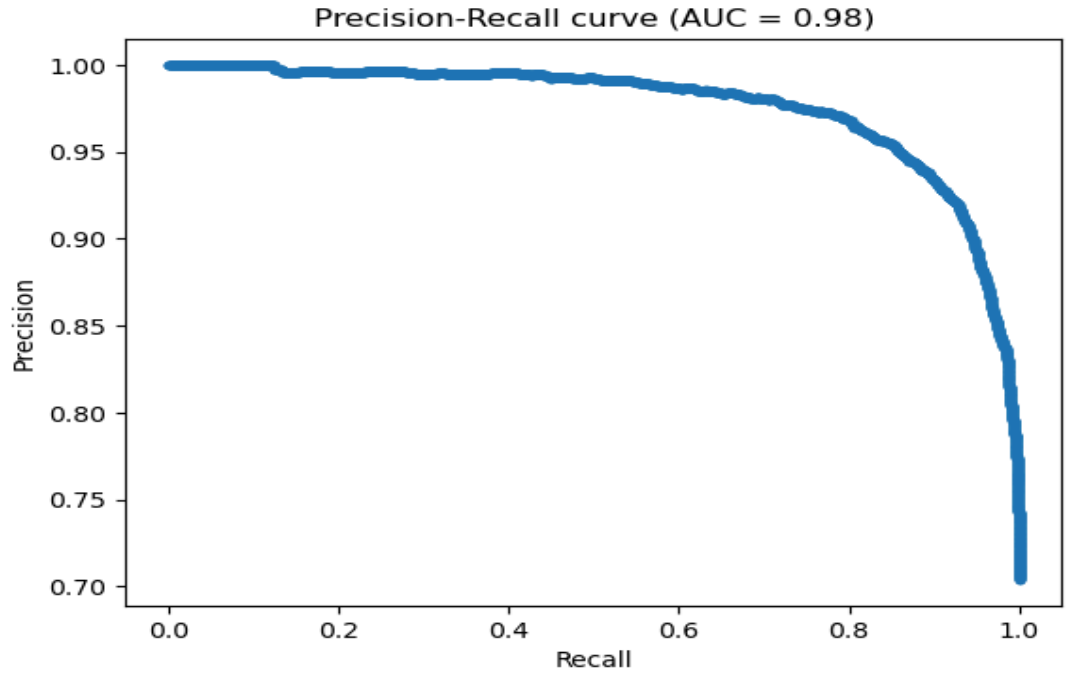

1.0
0.8
0.6
0.4
0.2
0.0
valor
etapa
treino
teste
accuracy
f1
recall
precision
auc
métrica

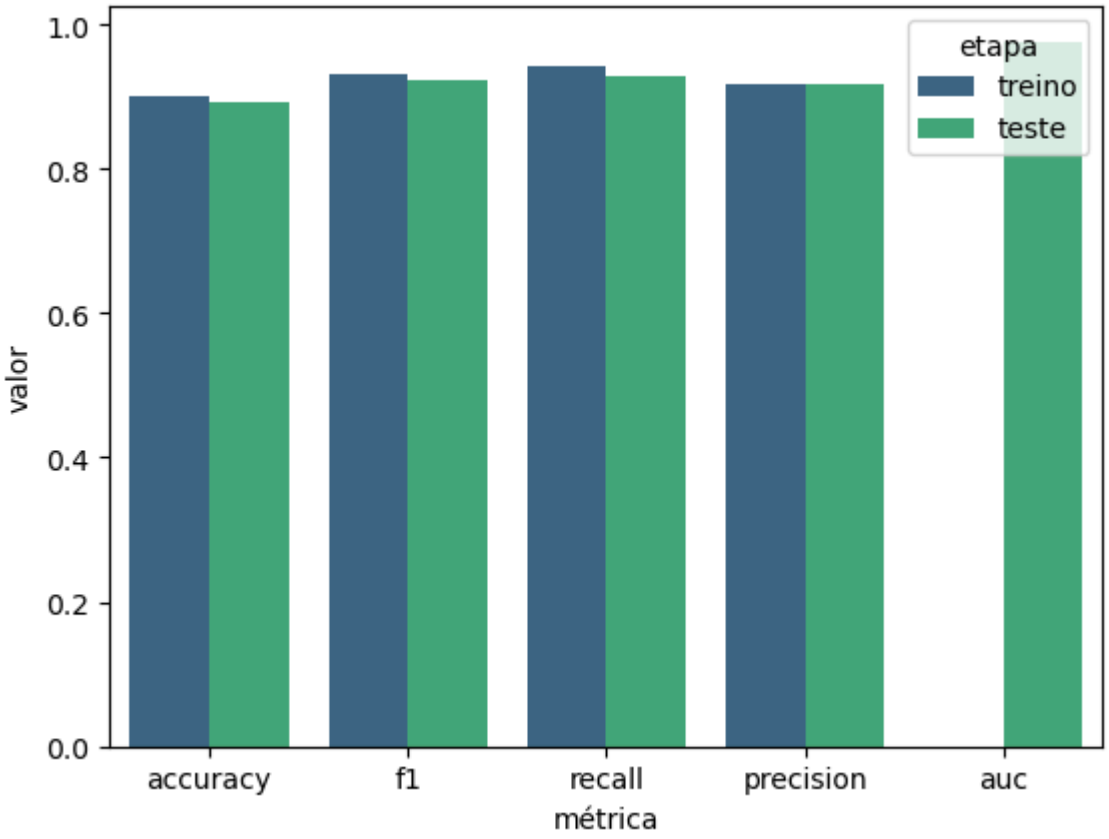

1017
332
209
2999
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

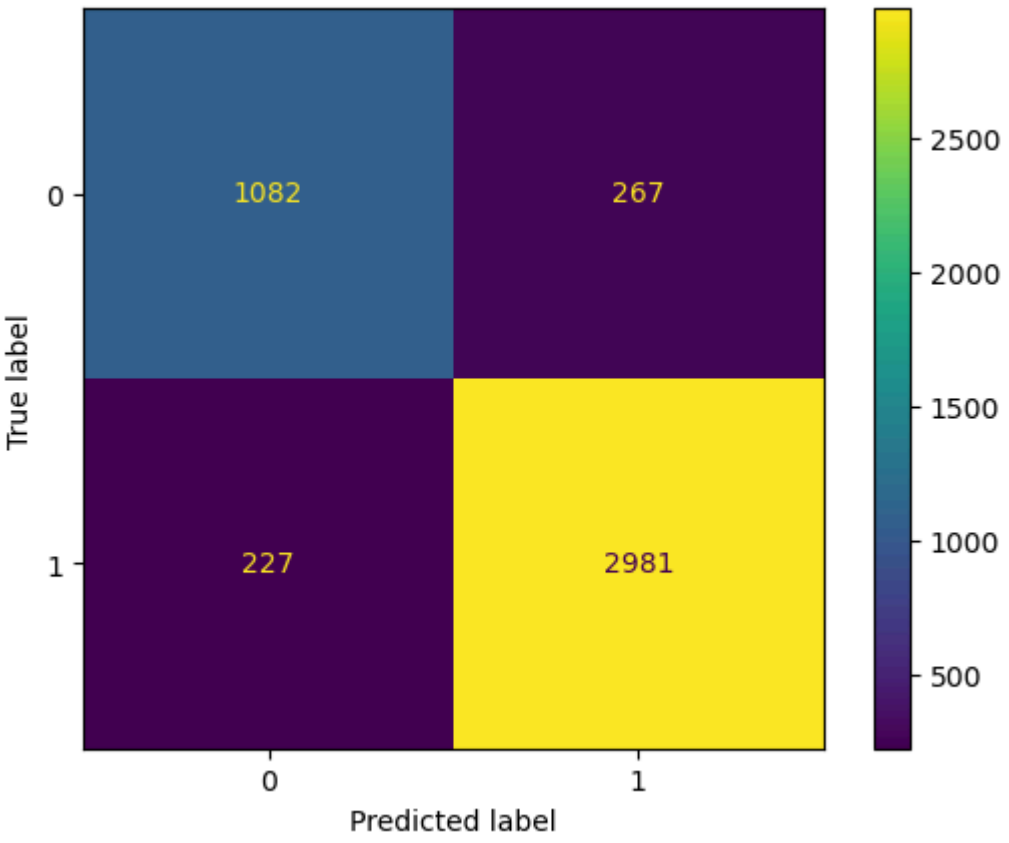
Comitê de Redes Neurais

O Comitê de MLPs se destacou com acurácia de 90,01% no treino e 89,16% no teste. O f1-score foi de 92,99% no treino e 92,35% no teste. O recall foi de 94,19% no treino e 92,92% no teste, enquanto a precisão ficou em 91,83% no treino e 91,78% no teste. O AUC atingiu 97,62% no teste.



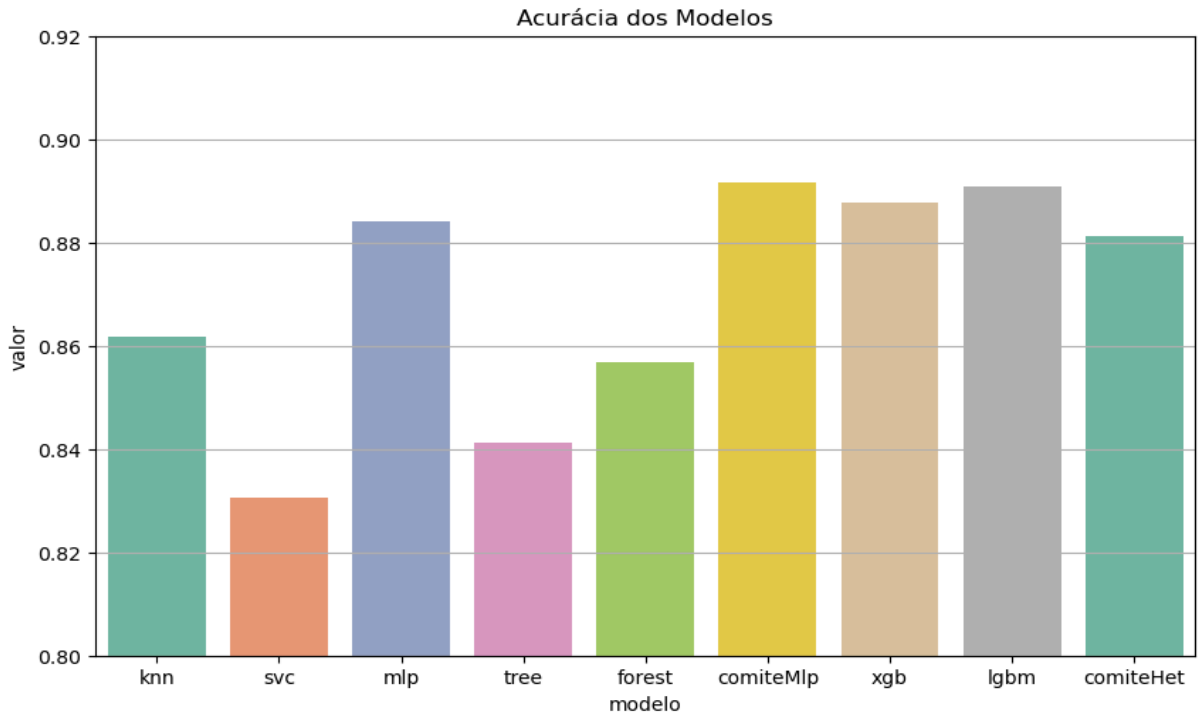


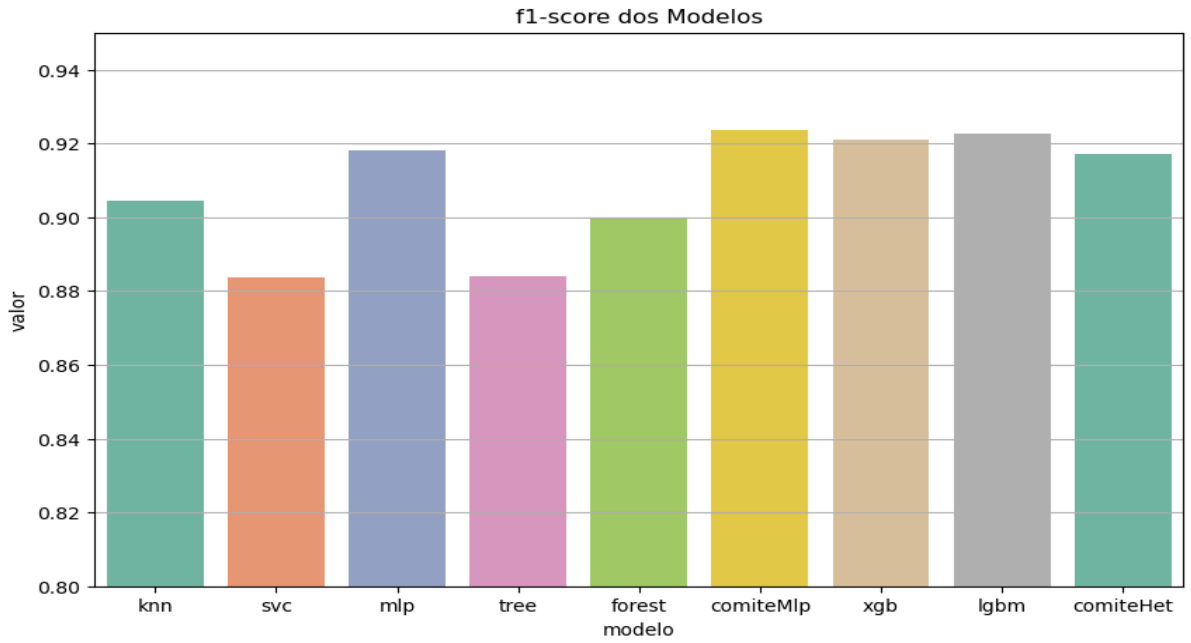

1.0
0.8
0.6
0.4
0.2
0.0
valor
etapa
treino
teste
accuracy
f1
recall
precision
auc
métrica

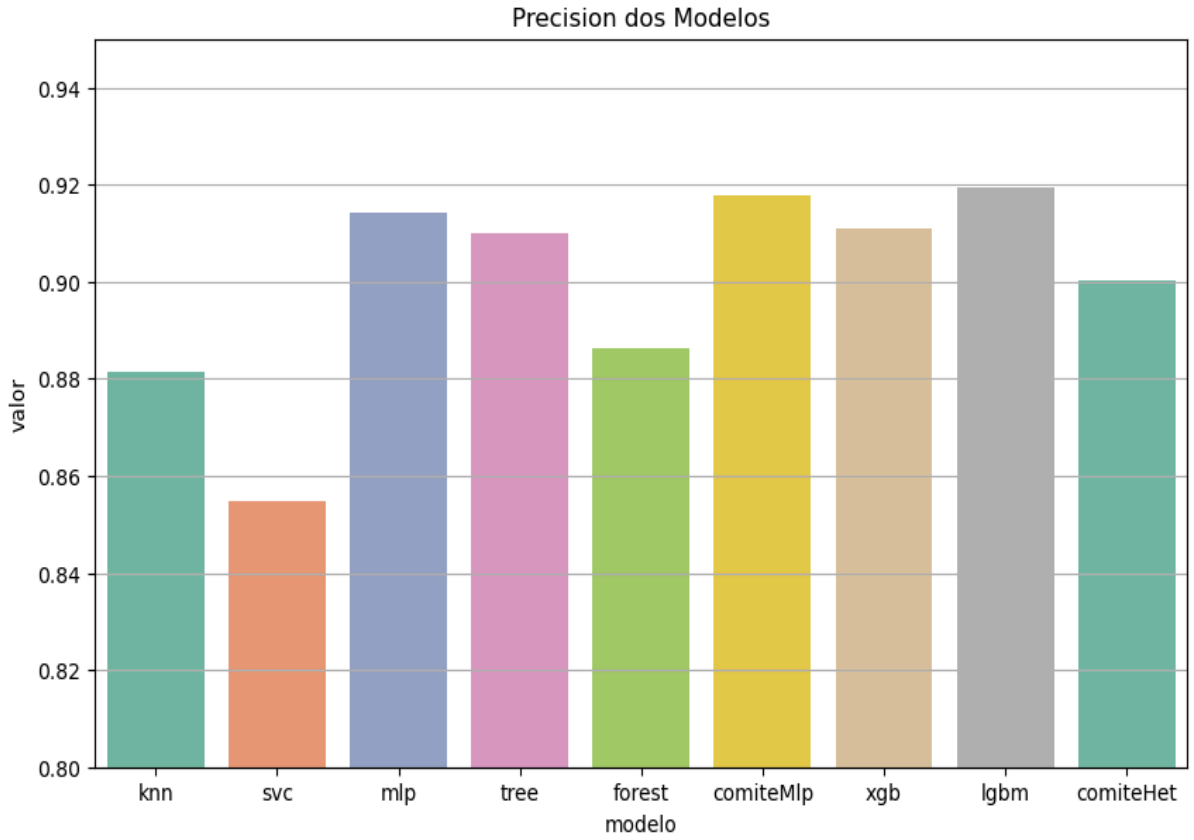

1082
267
227
2981
0
1
True label
0
1
Predicted label
2500
2000
1500
1000
500

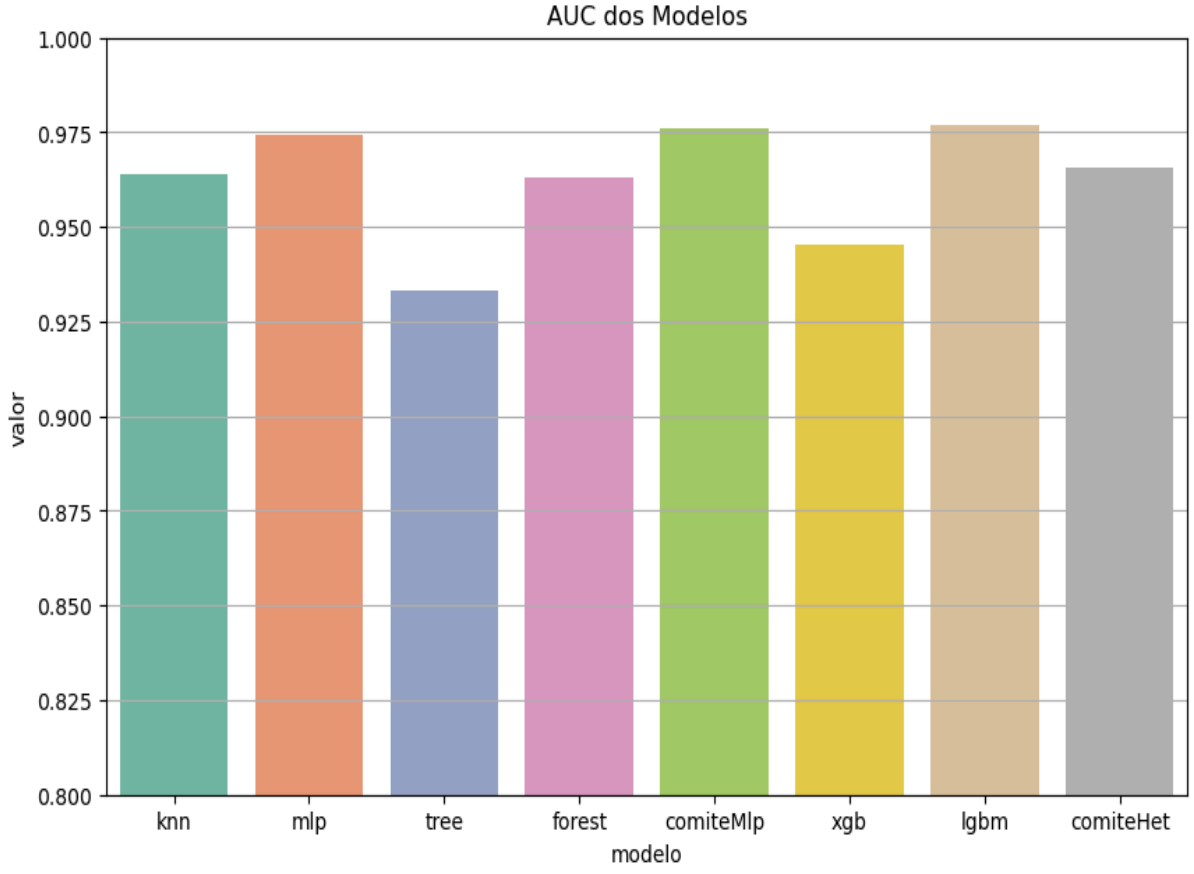
Avaliação Comparativa

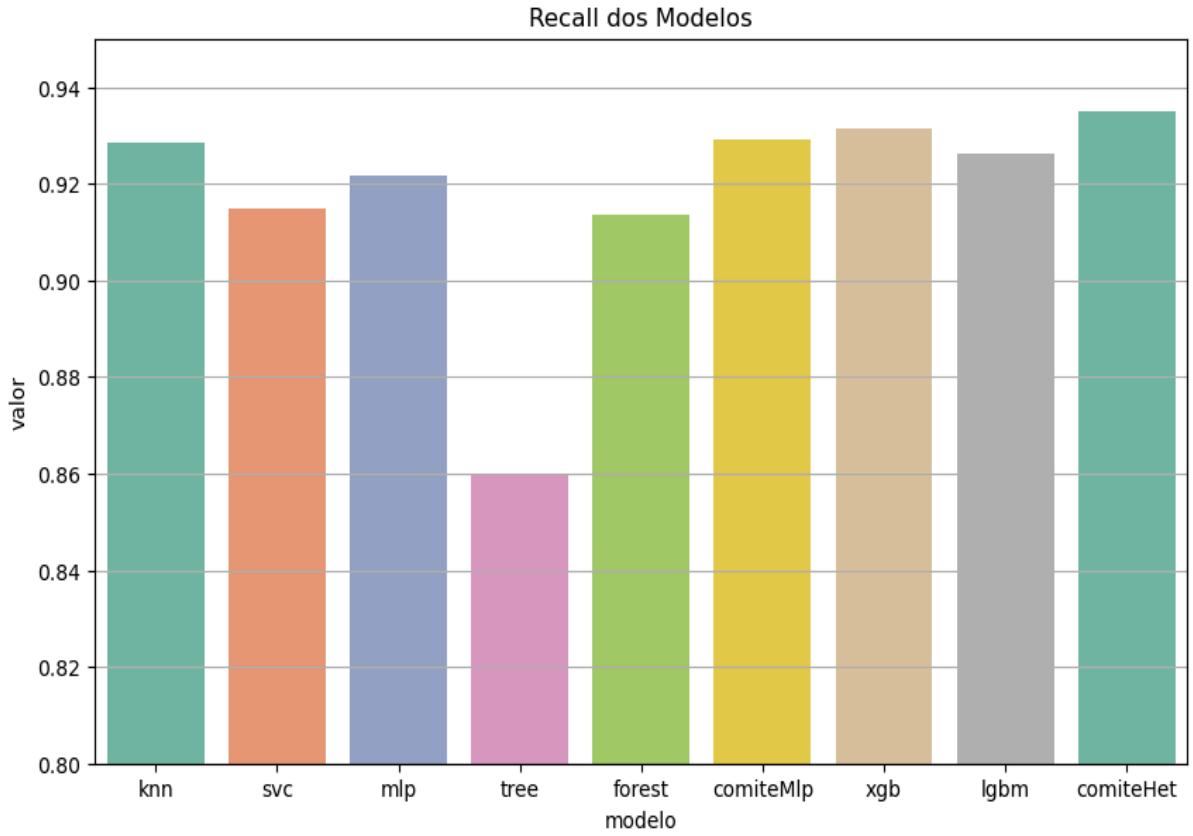
Para compreender melhor o desempenho dos modelos selecionados, foram gerados gráficos comparativos exibindo os valores das principais métricas no conjunto de teste. Esses gráficos de barras permitem uma visualização clara das diferenças entre os modelos finais, facilitando a identificação de pontos fortes e fracos de cada um. Através dessa análise visual, podemos avaliar como cada modelo equilibra acurácia, f1-score, recall, precisão e AUC, além de entender quais técnicas se mostraram mais eficazes na tarefa proposta.

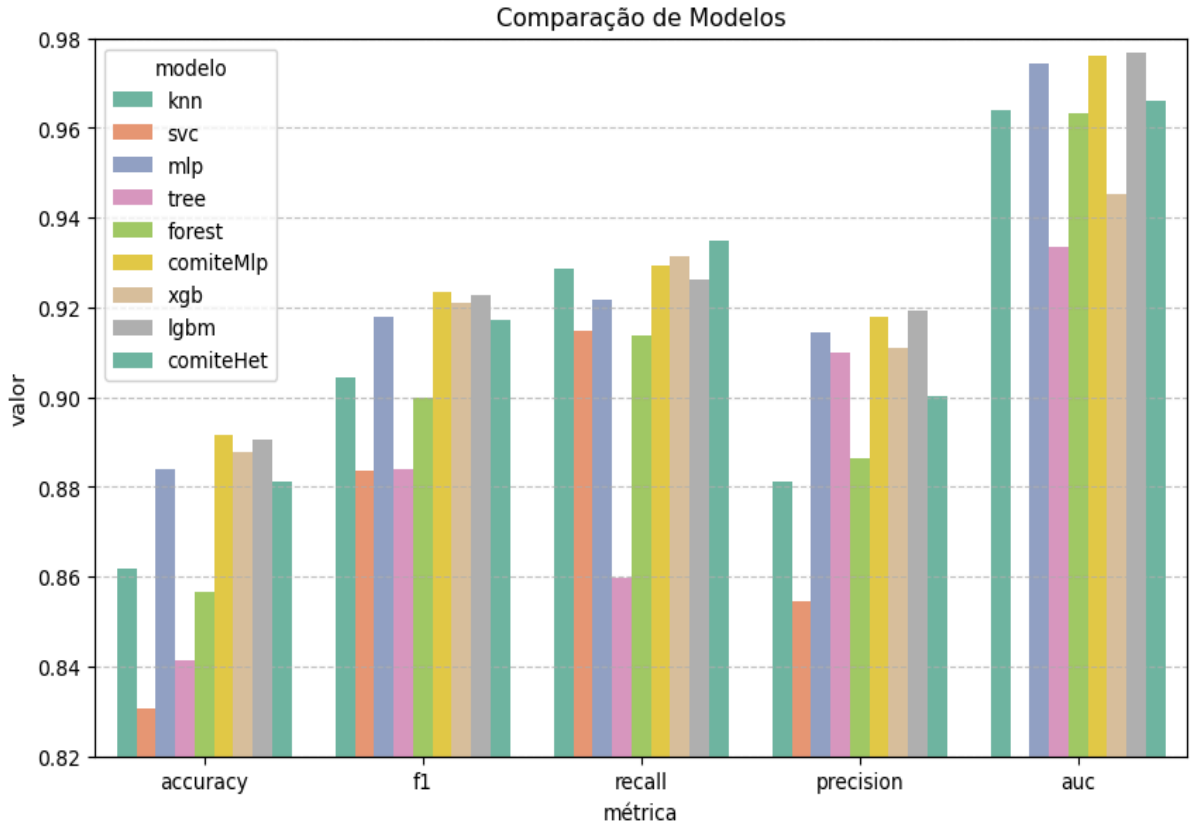



f1-score dos Modelos
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
valor
knn
svc
mlp
tree
forest
comiteMlp
xgb
lgbm
comiteHet
modelo


Precision dos Modelos
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
valor
knn
svc
mlp
tree
forest
comiteMlp
xgb
lgbm
comiteHet
modelo


AUC dos Modelos
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
valor
knn
mlp
tree
forest
comiteMlp
xgb
lgbm
comiteHet
modelo


Recall dos Modelos
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
valor
knn
svc
mlp
tree
forest
comiteMlp
xgb
lgbm
comiteHet
modelo


Comparação de Modelos
modelo
knn
svc
mlp
tree
forest
comiteMlp
xgb
lgbm
comiteHet
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
valor
accuracy
f1
recall
precision
auc
métrica

Os modelos KNN, Árvore de Decisão, Random Forest e SVC apresentaram desempenho inferior em comparação aos outros devido a diferentes limitações inerentes às suas abordagens:

KNN: Apesar de sua simplicidade e eficácia em certos cenários, o KNN tende a sofrer com alta variância e sensibilidade a ruído, especialmente em conjuntos de dados maiores e mais complexos. Além disso, seu desempenho pode ser prejudicado se os dados não possuírem uma separação bem definida no espaço vetorial.

Árvore de Decisão: Embora interpretable, esse modelo frequentemente sofre de overfitting, aprendendo demasiadamente os padrões do conjunto de treino e falhando em generalizar bem para novos dados.

Random Forest: Como um conjunto de múltiplas árvores de decisão, esse modelo é mais robusto que uma única árvore, mas ainda assim pode ser limitado em cenários onde a complexidade dos padrões exige modelos mais sofisticados. Além disso, pode apresentar menor capacidade de captura de relações complexas nos dados em comparação com modelos mais avançados, como redes neurais e boosting.

SVC (Support Vector Classifier): O SVC pode ser muito eficaz em espaços de baixa dimensionalidade, mas pode ter dificuldades em conjuntos de dados grandes, especialmente quando os dados não são perfeitamente separáveis. Seu desempenho também depende fortemente da escolha do kernel e da escala dos dados, o que pode torná-lo menos eficiente em comparação a modelos como XGBoost e LightGBM, que conseguem capturar padrões complexos com maior flexibilidade.

No geral, os modelos baseados em boosting e comitês apresentaram um melhor equilíbrio entre generalização e robustez, o que explica sua superioridade nos resultados finais.

Seleção dos 4 melhores modelos

Para selecionar os quatro melhores modelos, foi aplicada uma média ponderada considerando diferentes pesos para as métricas de avaliação: acurácia (1), f1-score (2.5), recall (2), precisão (1.5) e AUC (2). Essa abordagem valoriza mais o f1-score, seguido do recall e AUC, refletindo a importância de modelos que não apenas acertam a classificação, mas também garantem um bom equilíbrio entre precisão e recall, além de entender a natureza dos dados e seu desbalanceamento entre classes.

Os quatro melhores modelos selecionados com essa estratégia foram:

LGBM – Média ponderada mais alta, com destaque para recall (0.9261), f1-score (0.9227) e AUC (0.9768), indicando alta capacidade preditiva e boa separação entre classes.

Comitê de MLPs – Desempenho muito próximo ao LGBM, com f1-score (0.9235) e recall (0.9292) elevados, além de um AUC de 0.9762.

MLP – Fica ligeiramente abaixo do Comitê de MLPs, mantendo um equilíbrio entre todas as métricas, especialmente recall (0.9218) e AUC (0.9744).

Comitê Heterogêneo – Apesar de ter uma acurácia um pouco menor (0.8813), o modelo se destacou pelo recall (0.9349) e um AUC competitivo (0.9659), garantindo boa separação de classes.

Comparando esses resultados, os modelos LGBM e Comitê de MLPs são os mais fortes, possivelmente devido à combinação de técnicas de aprendizado mais sofisticadas. O MLP isolado também apresentou um desempenho robusto, o que indica que modelos de redes

neurais foram bem ajustados. Já o Comitê Heterogêneo se saiu bem em recall, garantindo boa recuperação de instâncias positivas.

A análise gráfica desses modelos permite uma comparação visual dos desempenhos ponderados, destacando suas diferenças em cada métrica e confirmando quais se saíram melhor em aspectos específicos.