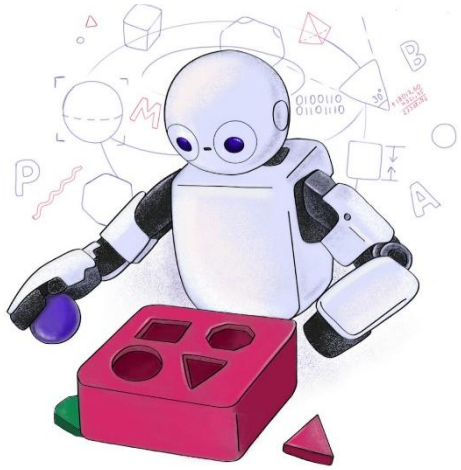
TP558 - Tópicos avançados em Machine Learning:

Optuna

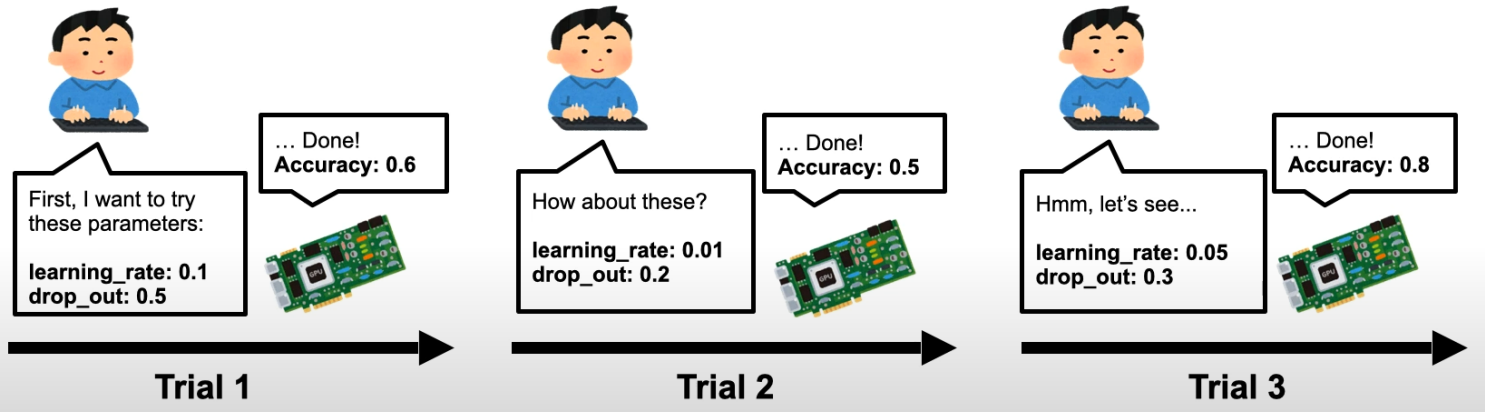


Inatel

Alessandra Carolina Domiciano
alessandra.carolina@mtel.inatel.br

Otimização de hiperparâmetros: o que é e por quê?

- A otimização de hiperparâmetros é o processo de encontrar os **melhores valores** para os hiperparâmetros de um modelo de aprendizado de máquina.
- Esse processo permite **melhorar o desempenho do modelo** ao encontrar uma configuração mais eficiente, além de aumentar a capacidade de generalização e reduzir o consumo de tempo e recursos computacionais.
- Como resultado, a otimização de hiperparâmetros pode levar a **vantagens de negócio**:
  - Ex: decisões mais precisas, automação mais eficaz e redução de custos operacionais, etc.



O que são hiperparâmetros?

- Em aprendizado de máquina, hiperparâmetros são parâmetros cujos valores são definidos **antes do início do processo de treinamento** do modelo. Eles possuem a função de controlar o processo de aprendizagem.
- Já os valores dos outros parâmetros são definidos ao longo do treinamento do modelo.

Exemplos de hiperparâmetros:

- Passo de aprendizagem
- Número de épocas de treinamento
- Fatores de regularização
- Função de ativação
- Número de estimadores (ensemble)

Evolução dos otimizadores

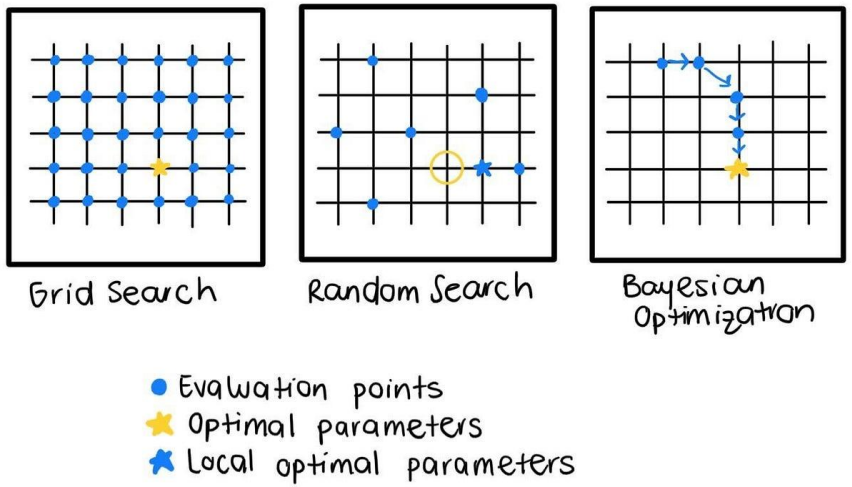
- Historicamente, a otimização de hiperparâmetros começou com métodos simples, como **Grid Search** e **Random Search**.

- Posteriormente, surgiram abordagens mais inteligentes como a **Otimização Bayesiana**, inicialmente com o modelo Gaussian Process (GP) e, depois, com algoritmos mais escaláveis como o Tree-structured Parzen Estimator (TPE).

- Atualmente, existem frameworks modernos que se baseiam na Otimização Bayesiana e agregam diversos recursos e funcionalidades, tornando-se ferramentas avançadas para otimização de hiperparâmetros, como o **Optuna**.

Evolução dos otimizadores

- **Grid Search:** Testa **todas as combinações** possíveis dentro de uma grade definida.
  - Vantagem: simplicidade
  - Desvantagem: alto custo computacional
- **Random Search:** Escolhe **combinações aleatórias** dentro dos intervalos da grade.
  - Vantagem: mais eficiente que o GS
  - Desvantagem: depende de “sorte”
- **Bayesian Search:** Constrói um **modelo probabilístico da função objetivo**, tipicamente com Processo Gaussiano, e o utiliza para orientar a próxima combinação a se testar.
  - Vantagem: inteligente, mais eficiente
  - Desvantagem: escala mal com muitos hiperparâmetros, dificuldade com espaços condicionais (se A, então testar B)



Evolução dos otimizadores – Optuna!

- O Optuna é uma biblioteca moderna para otimização de hiperparâmetros.
- Baseada em Otimização Bayesiana, com suporte a múltiplos métodos de amostragem.
- Suporta pruning (interrupção de tentativas pouco promissoras).
- Oferece otimização multiobjetivo, ideal para problemas com múltiplas métricas.
- Permite integração com outras bibliotecas de ML (scikit-learn, PyTorch, XGBoost, LightGBM, Keras).
- Possui um dashboard interativo, com visualização em tempo real de métricas, progresso e gráficos.
- Suporta execução paralela e distribuída e possui recursos robustos para persistência.

OPTUNA

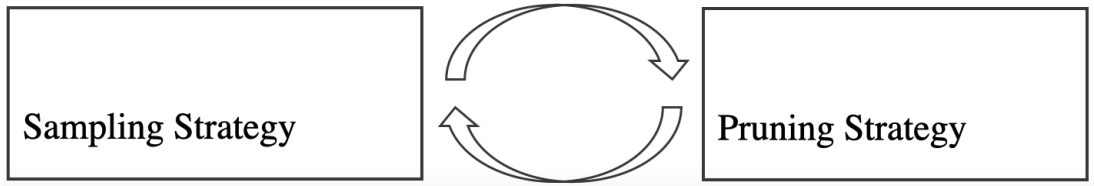
Função objetivo

- O processo de otimização de hiperparâmetros no **Optuna** é guiado por uma **função objetivo**, que define a **métrica a ser otimizada**.
- Essa função é especificada pelo usuário e é chamada em cada trial (teste de hiperparâmetros).
- A métrica retornada pela função está diretamente relacionada ao **modelo utilizado**:
  - Em modelos de **regressão**, ela pode avaliar métricas como o **erro quadrático médio (RMSE)**.
  - Em modelos de **classificação**, podem ser usadas métricas como **acurácia**, **F1-score** ou **AUC**.
- O objetivo é encontrar a combinação de hiperparâmetros que **maximiza ou minimiza** essa métrica, conforme o problema.
- Ou seja, a função objetivo é o elo entre o modelo e o processo de otimização. É ela quem informa ao Optuna se uma certa combinação de hiperparâmetros produziu um bom ou mau resultado.

Processo de busca de hiperparâmetros

- No Optuna, a definição do espaço de busca ocorre **durante a execução** da função objetivo. Essa abordagem é chamada de **Define-by-Run** e permite que:
  - Hiperparâmetros sejam definidos **de forma dinâmica e até condicional**;
  - O espaço de busca seja **mais flexível**;
  - O processo de busca seja mais eficiente, guiado por **algoritmos inteligentes**;
  - Diferentes modelos ou arquiteturas sejam explorados no mesmo estudo.

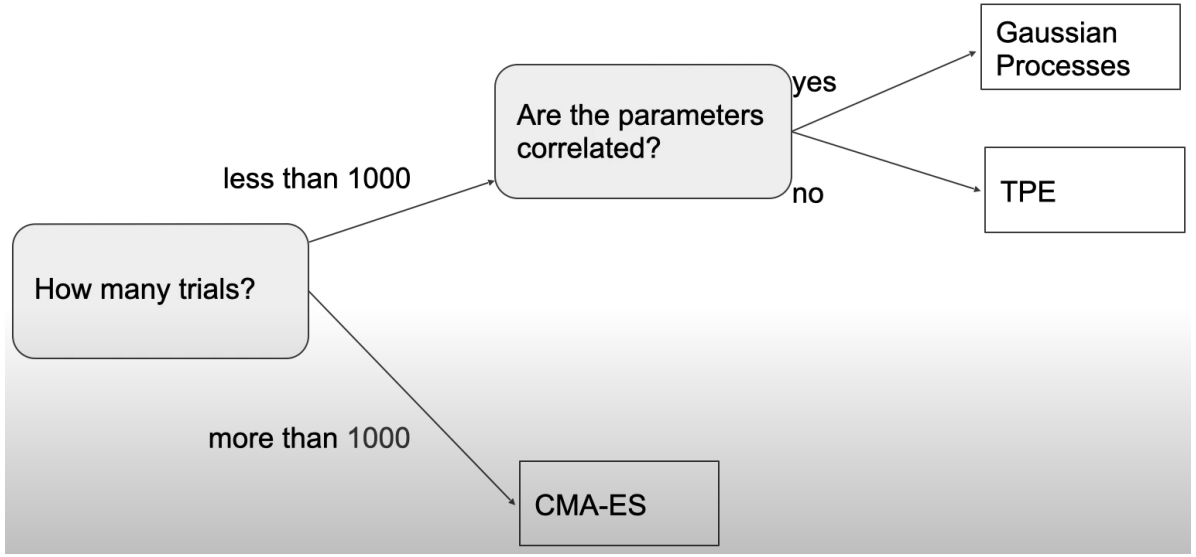
- De forma geral, o processo de busca de hiperparâmetros é dividido em duas partes:
  - A **estratégia de amostragem**, que define o algoritmo para escolha das combinações de hiperparâmetros a serem testadas.
  - A **estratégia de poda**, que permite que experimentos ruins sejam encerrados mais cedo, aumentando a eficiência.



Amostragem (sampling)

O Optuna possui vários métodos para amostragem de hiperparâmetros, sendo os principais:

- **TPE (Tree-structured Parzen Estimator)**
- **GP (Gaussian Processes)**
- **CMA-ES (Covariance Matrix Adaptation Evolution Strategy)**



TPE (Tree-structured Parzen Estimator)

Tipo: Bayesiano

Funcionamento básico:

- Executa testes com n amostras aleatórias na função objetivo.
- Modela uma distribuição de probabilidade para valores com bons resultados (l(x)).
- Modela outra distribuição de probabilidade para os demais valores (g(x)).
- Escolhe novos pontos onde **a razão l(x) / g(x) é alta** → regiões promissoras.

Vantagens:

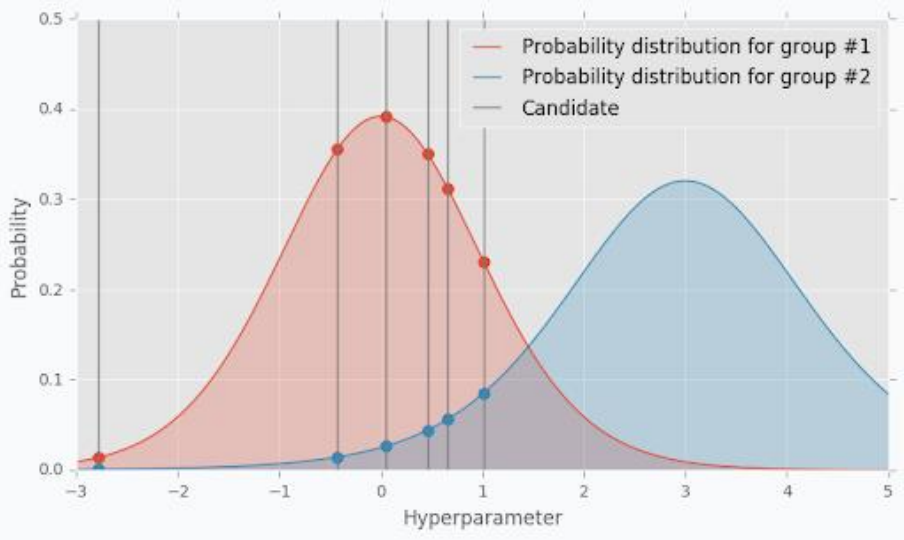
- Eficiente em alta dimensionalidade
- Baixo custo computacional por iteração
- Suporta variáveis categóricas

Desvantagem:

- Modelo probabilístico simplificado,

não captura correlações entre os hiperparâmetros

- Pode ter baixo desempenho em espaços pequenos



GP (Gaussian Processes)

Tipo: Bayesiano

Funcionamento básico:

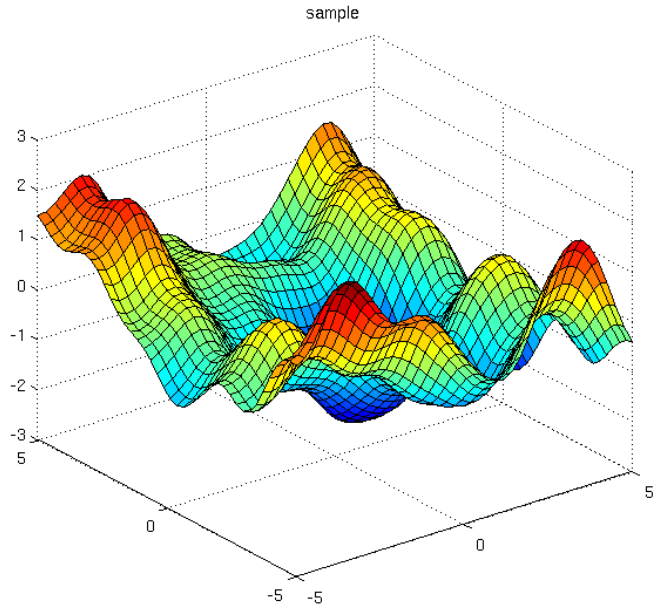
- Executa testes com n amostras aleatórias na função objetivo.
- Constrói um modelo probabilístico para a função objetivo, utilizando um processo gaussiano para modelar os resultados com:
  - média μ(x): predição de desempenho
  - desvio padrão σ(x): incerteza associada
- Utiliza uma função de aquisição para sugerir uma nova combinação de hiperparâmetros com base em:
  - média μ(x): onde o desempenho é alto
  - desvio padrão σ(x): onde a incerteza é alta
- Testa os novos valores e repete o processo, atualizando o modelo de predição.

Vantagens:

- Captura correlações entre hiperparâmetros
- Alta precisão em espaços pequenos

Desvantagens:

- Baixa eficiência em alta dimensionalidade
- Alto custo computacional por iteração
- Não suporta variáveis categóricas



CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

Tipo: Evolutivo

Funcionamento básico:

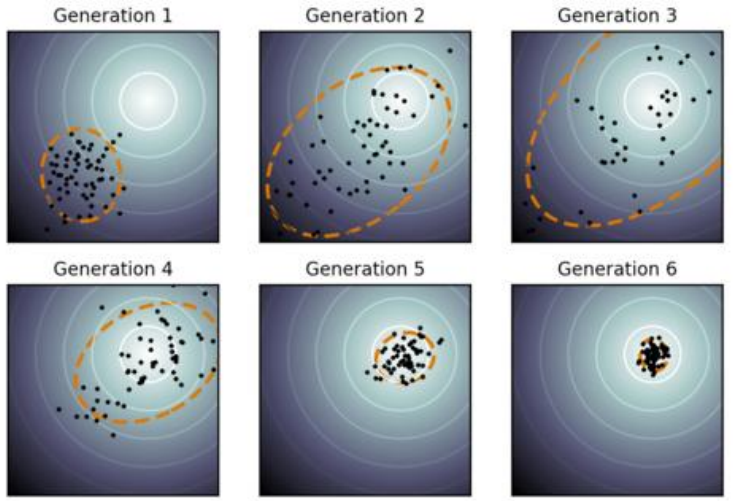
- Gera n amostras aleatórias (população inicial) a partir de uma distribuição gaussiana e executa os testes na função objetivo.
  - A população inicial possui uma média $\mu_{t=0}$ e uma matriz de covariância $C_{t=0}$.
- Seleciona os melhores candidatos.
- Ajusta a média e a matriz de covariância da distribuição da população (μ_{t+1}, C_{t+1}).
- Gera novas amostras (nova geração) a partir da nova distribuição e repete o processo.

Vantagens:

- Captura correlações entre hiperparâmetros
- Bom desempenho em funções não lineares e ruidosas

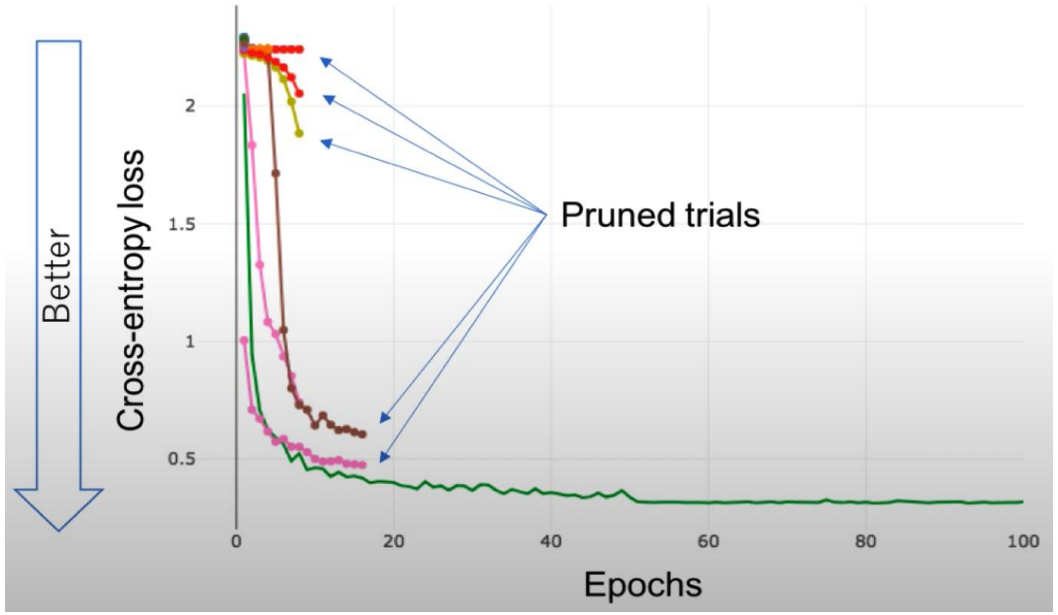
Desvantagens:

- Baixa eficiência em alta dimensionalidade
- Alto custo computacional por iteração
- Não suporta variáveis categóricas



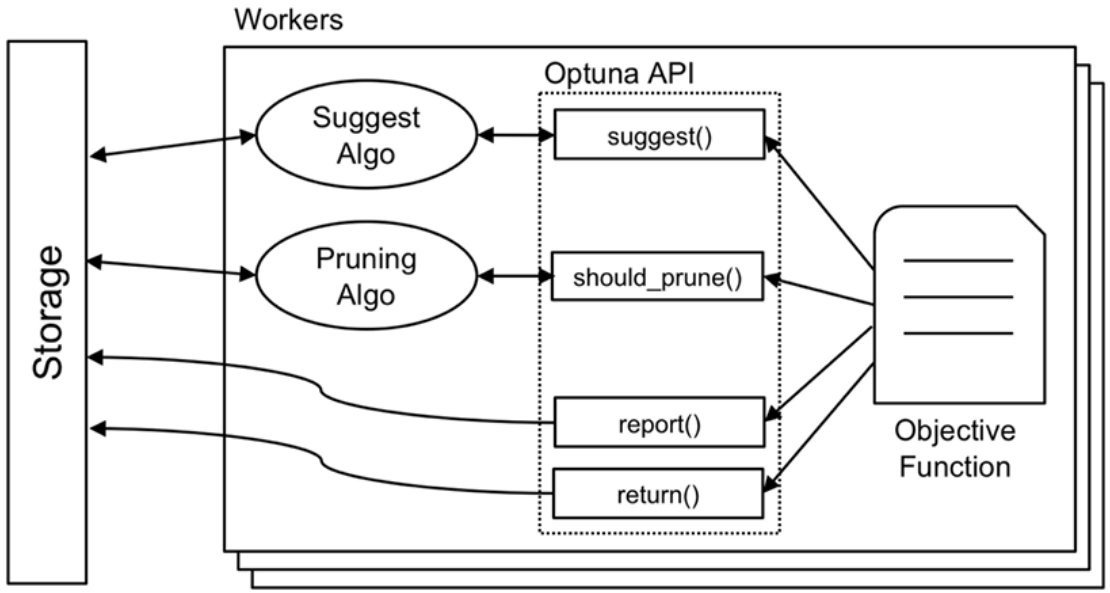
Poda (Pruning)

- **Pruning** é uma técnica usada pelo Optuna para **encerrar tentativas com baixo desempenho antes que terminem**.
- Em modelos cujo treinamento ocorre em **múltiplas etapas** (épocas, iterações), o Optuna pode ser configurado para interromper antecipadamente um teste com uma combinação de hiperparâmetros se ele tiver desempenho ruim nas primeiras etapas.
- As vantagens do pruning são:
  - Economia de tempo
  - Maior número de testes
  - Foco nos melhores candidatos



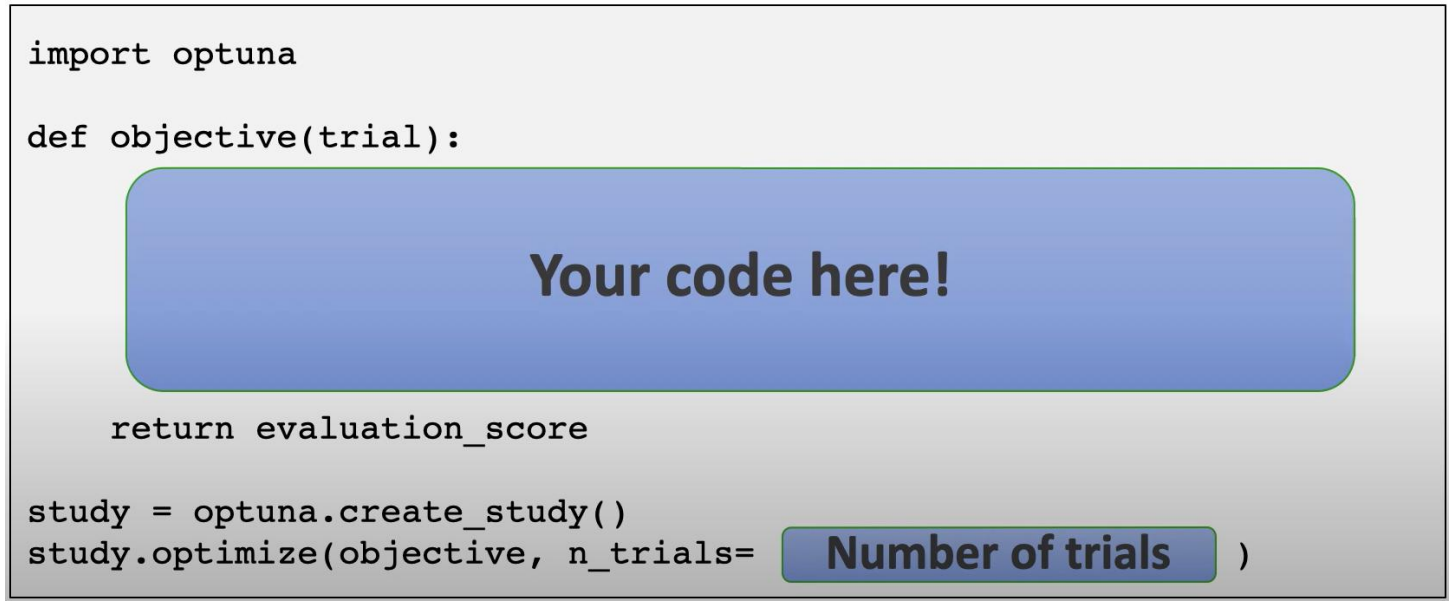
Optuna – Arquitetura

- Uma sessão de otimização é chamado de estudo.
- Cada worker executa uma instância da função objetivo de um estudo e compartilha o progresso do estudo atual por meio do armazenamento.
- A função objetivo executa seu teste usando as APIs do Optuna e os dados dos estudos anteriores, quando necessário, que são armazenadas.
- O Optuna permite configurar um banco de dados local ou externo, como SQLite e PostgreSQL.



Implementação básica do Optuna

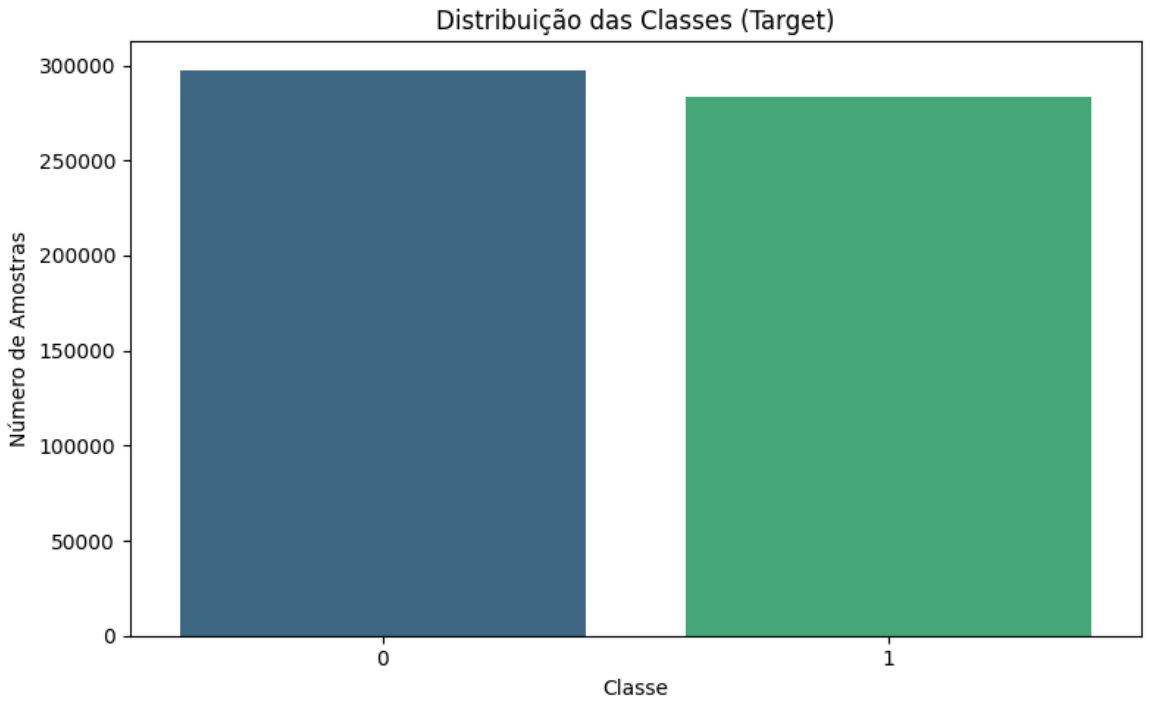
1. Definir a função objetivo para minimizar/maximizar durante o processo de otimização dos hiperparâmetros.
2. Obter os hiperparâmetros de teste a cada trial.
3. Executar a busca com study.optimize().



Exemplo – Classificação binária

Dataset: Covertype

- Amostras sobre tipo de cobertura do solo (vegetação/floresta) com base em características cartográficas.
- Número de classes: 7, foi adaptado para binário (classe 2 contra o resto)
- Número de amostras: 581.012
- Número de features: 53 + alvo
- Conjunto dividido em treino (80%) e teste (20%)

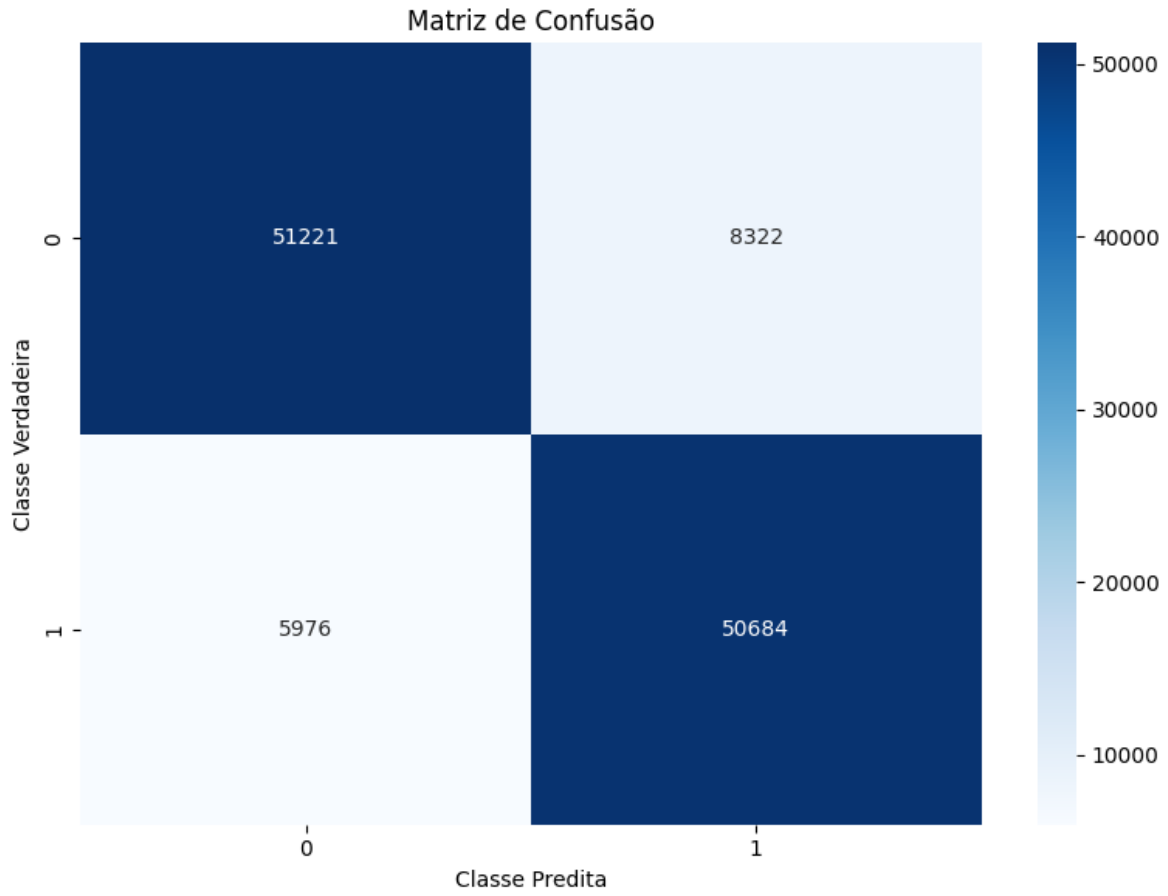


Modelo testado: XGBoost

Exemplo – Classificação binária

Modelo XGBoost padrão:

- Acurácia: 88%
- Precisão: 88%
- Recall: 88%
- F1-score: 88%



Exemplo – Classificação binária

```python
def objective(trial):
    params = {
        "objective": "binary:logistic",
        "n_estimators": trial.suggest_int("n_estimators", 50, 200),
        "reg_lambda": trial.suggest_float("reg_lambda", 1e-4, 1.0, log=True),
        "reg_alpha": trial.suggest_float("reg_alpha", 1e-4, 1.0, log=True),
        "max_depth": trial.suggest_int("max_depth", 4, 10),
        "learning_rate": trial.suggest_float("learning_rate", 1e-3, 0.3, log=True),
        "gamma": trial.suggest_float("gamma", 0, 10),
        "seed": 42,
        "verbosity": 0,
    }

    model = xgb.XGBClassifier(**params)

    model.fit(X_train, y_train, verbose=False)

    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    return acc
```

- Função de perda: logloss
- Número de árvores no ensemble: 50 a 200
- Regularização L2: 0,0001 a 1
- Regularização L1: 0,0001 a 1
- Profundidade máxima das árvores: 4 a 10
- Passo de aprendizagem: 0,001 a 0,3
- Fator de poda: 0 a 10

- Métrica objetivo: acurácia

Exemplo – Classificação binária

```
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
```

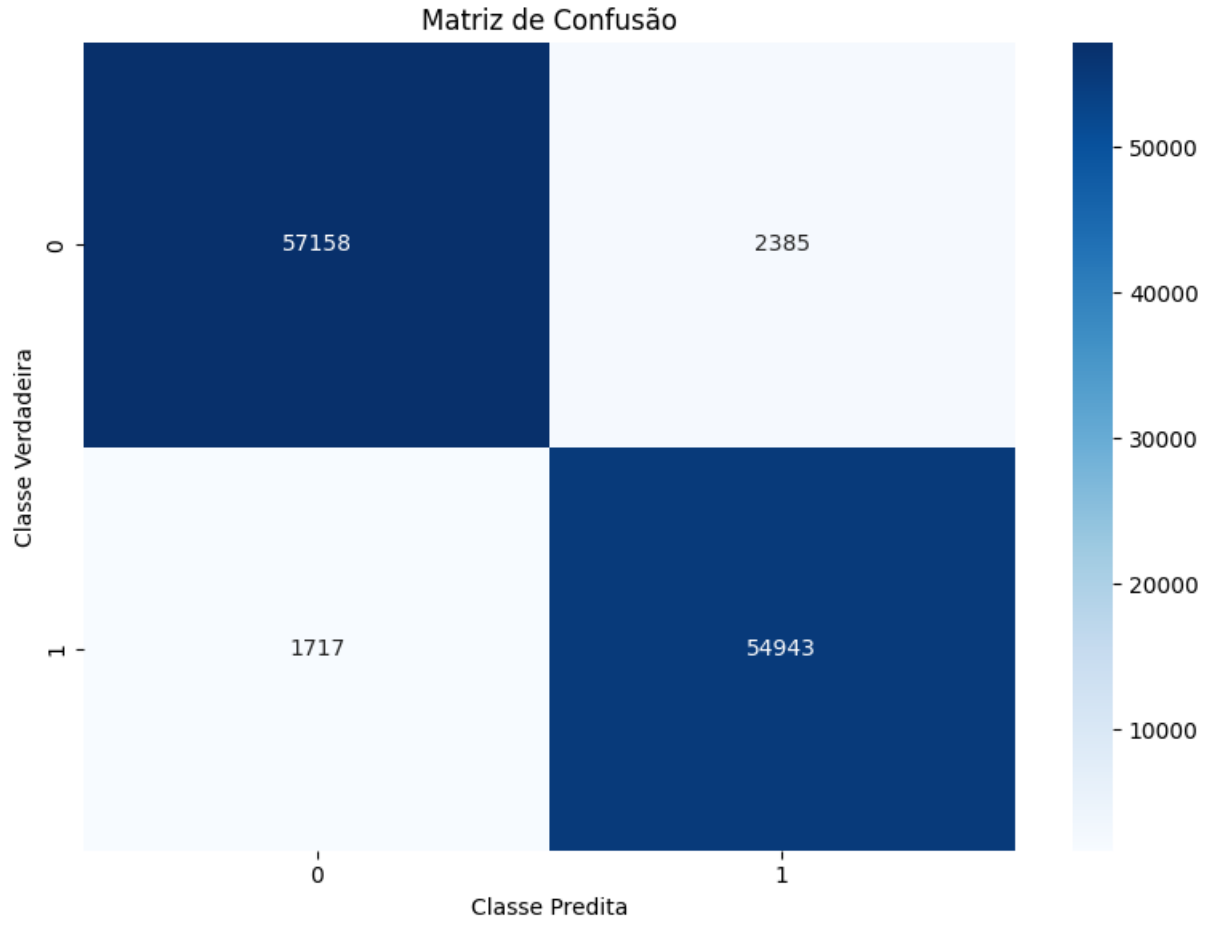
- A otimização é feita no sentido de maximizar a função objetivo (acurácia).
- O processo executa 100 trials.
- Tempo total: 1h30min

- Melhores hiperparâmetros:
  - Função de perda: logloss
  - Número de árvores no ensemble: 175
  - Regularização L2: 0,00021
  - Regularização L1: 0,08989
  - Profundidade máxima das árvores: 10
  - Passo de aprendizagem: 0,25
  - Fator de poda: 0,2832

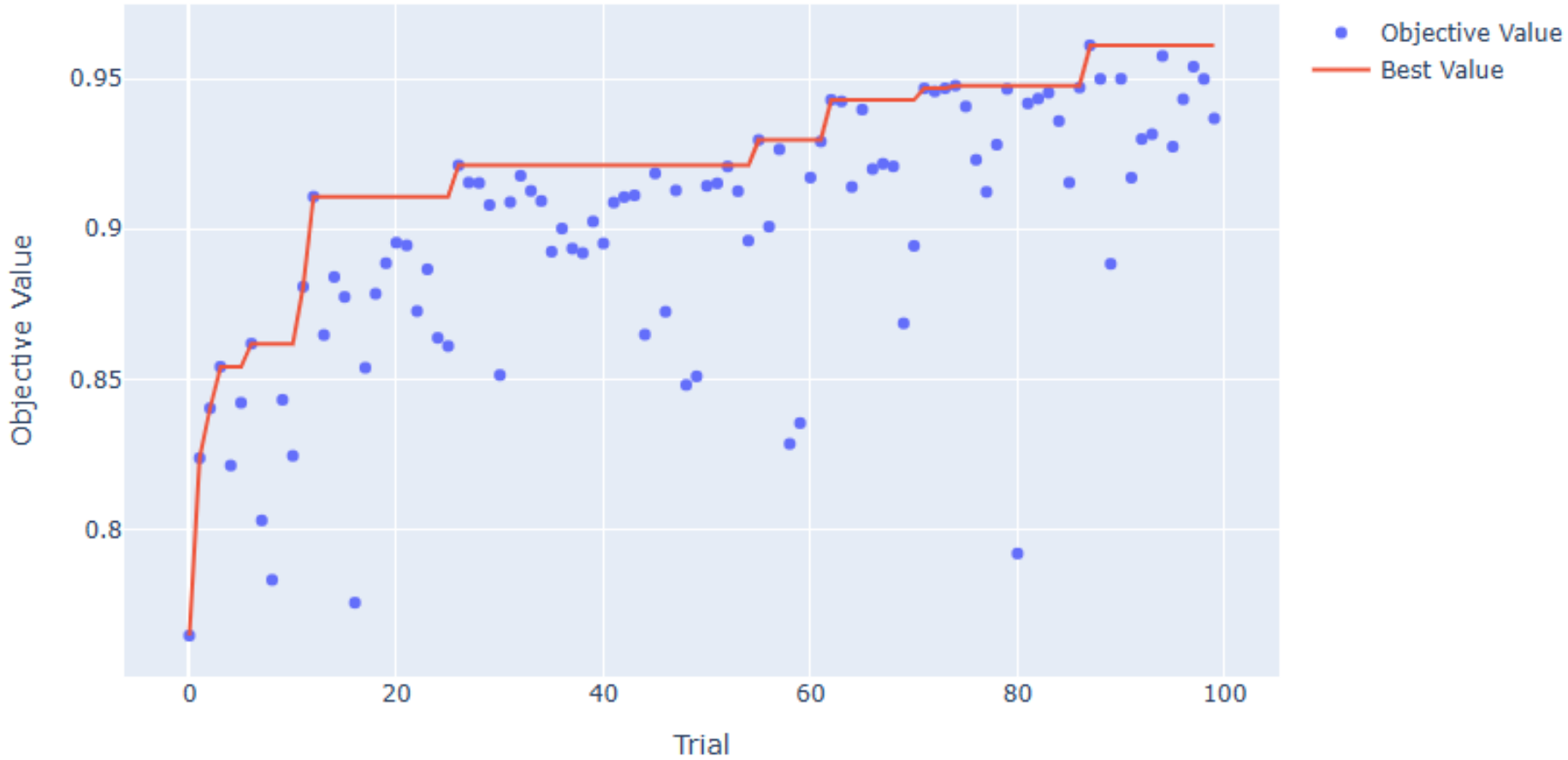
Exemplo – Classificação binária

Modelo XGBoost otimizado:

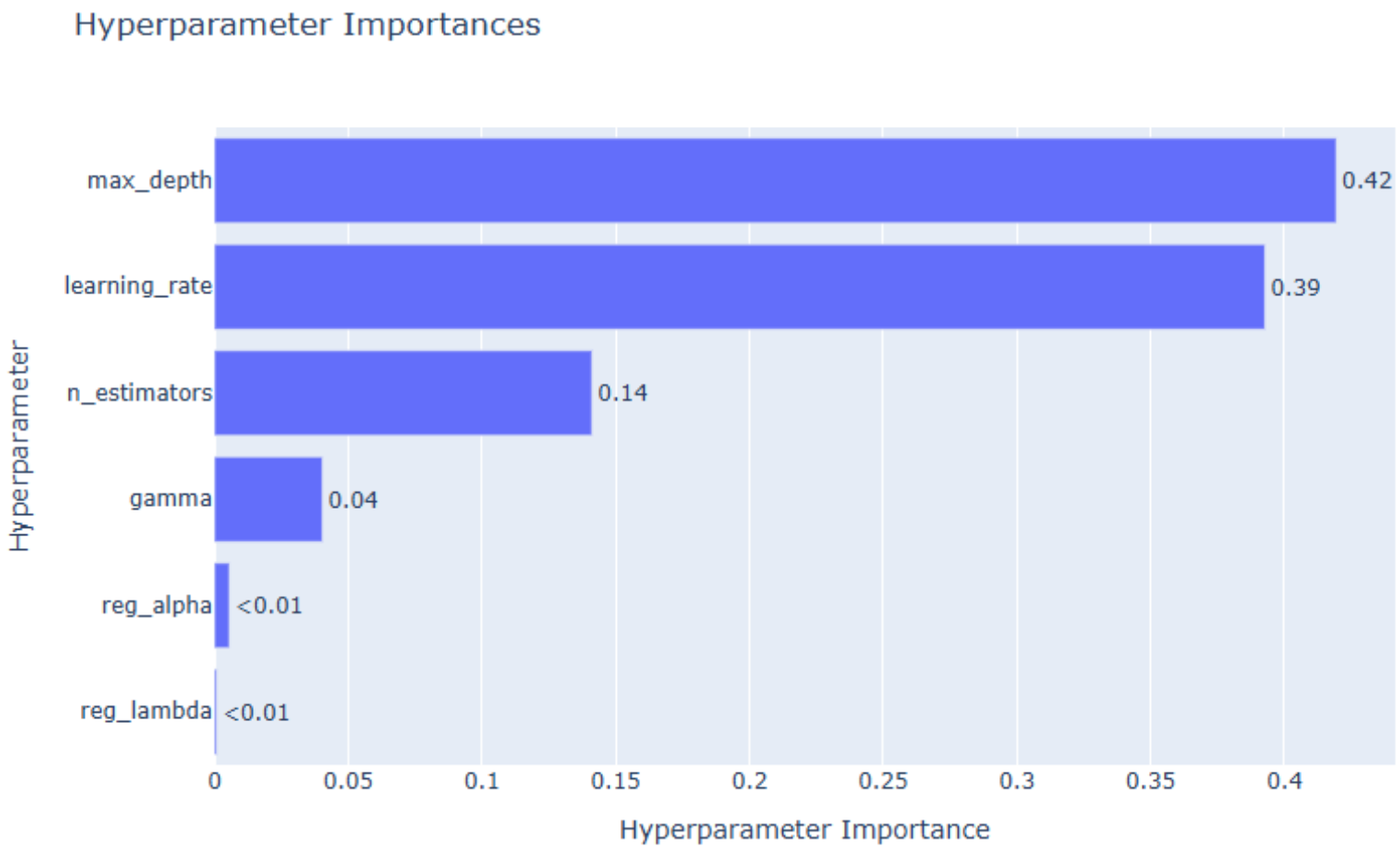
- Acurácia: 96%
- Precisão: 96%
- Recall: 96%
- F1-score: 96%



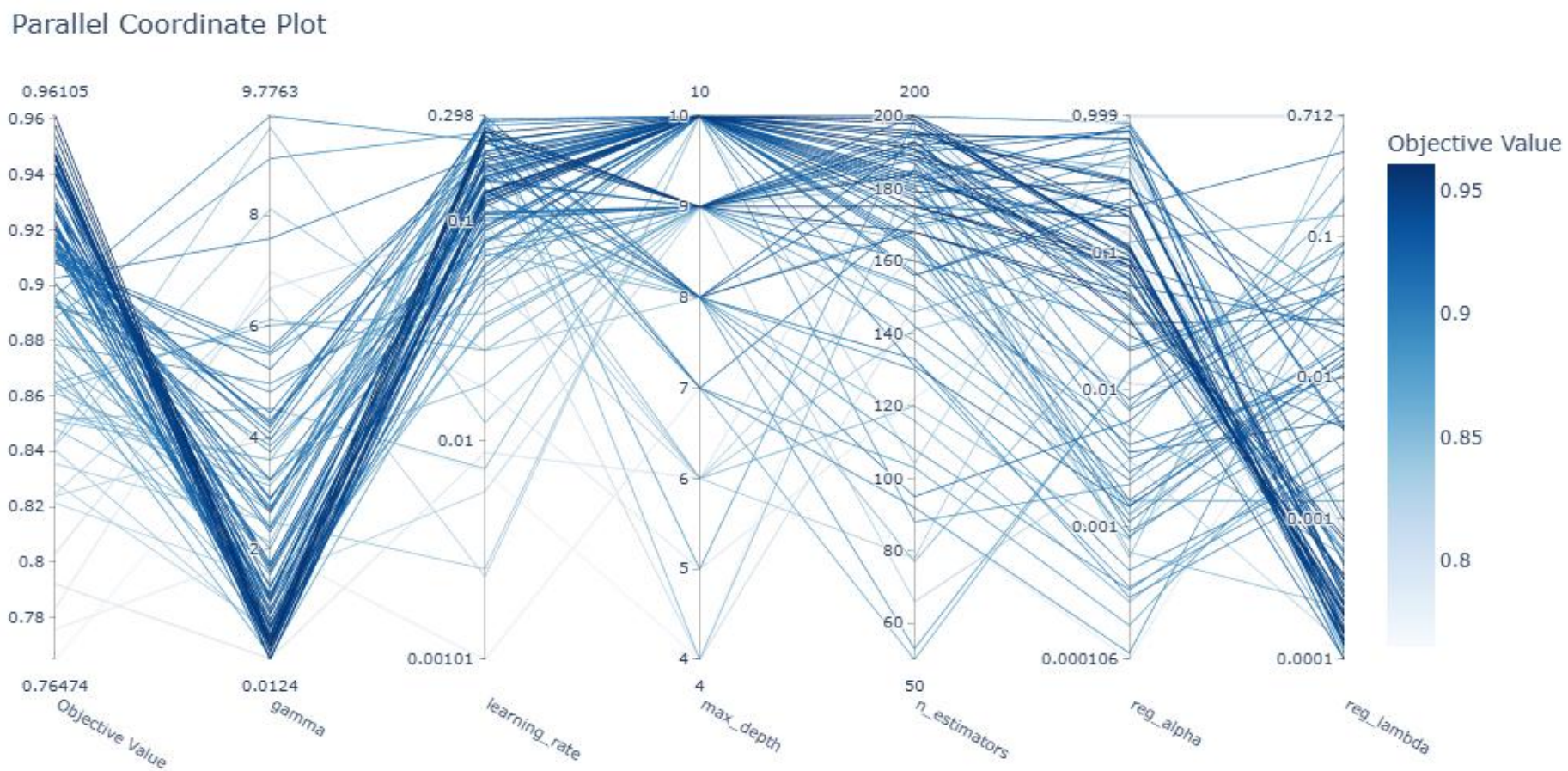
Exemplo – Classificação binária



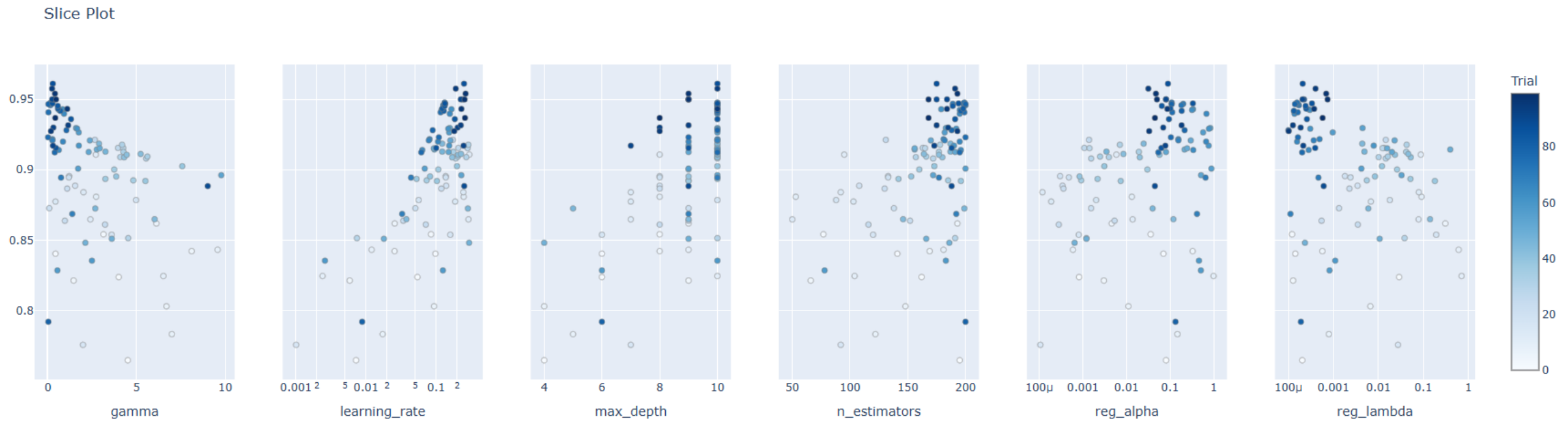
Exemplo – Classificação binária



Exemplo – Classificação binária

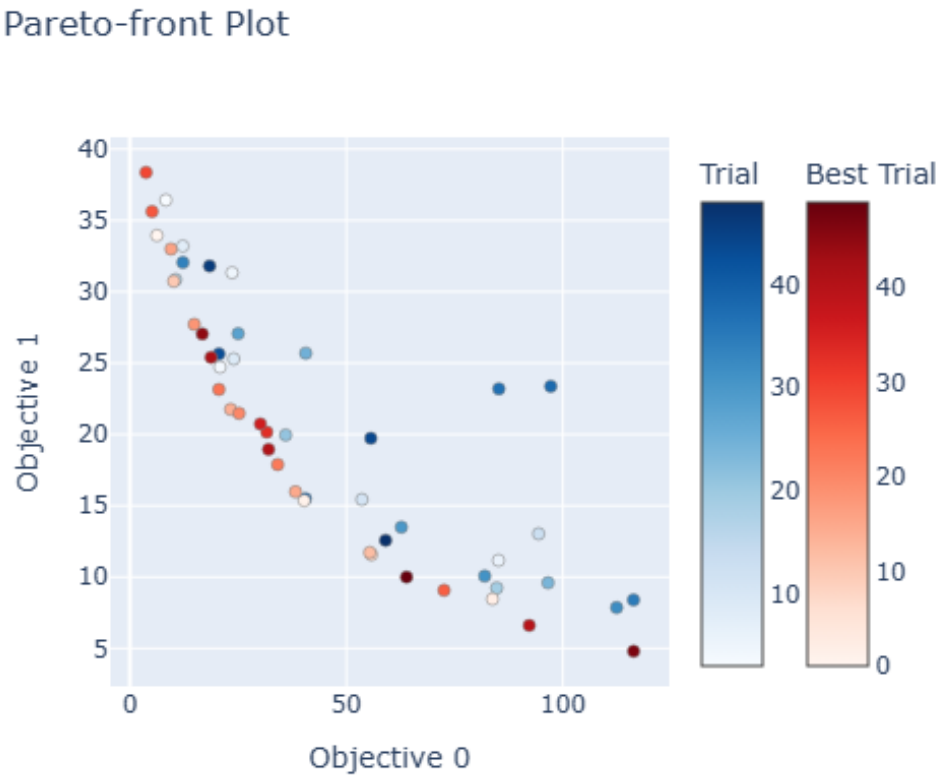


Exemplo – Classificação binária



Otimização multiobjetivo

- Desde a versão 2.0, o Optuna suporta otimização multiobjetivo.
- Ela permite que os hiperparâmetros sejam otimizados considerando mais de uma métrica simultaneamente, por exemplo, acurácia e tempo de inferência.
- A otimização multiobjetivo é baseada no conceito de fronteira de Pareto, que representa o conjunto das melhores soluções não dominadas por outras.
- Na imagem, as bolinhas vermelhas representam a fronteira de Pareto em um exemplo onde o objetivo é minimizar ambas as métricas.



Paralelismo

- O Optuna suporta paralelismo dos tipos:
  - Multi-threading: Mesma máquina, vários threads no mesmo processo Python
  - Multi-processo: Mesma máquina, vários processos Python paralelos
  - Multi-máquina (distribuído): várias máquinas em cluster

- No paralelismo, vários Workers são instanciados e executados ao mesmo tempo, o que possibilita realizar vários trials simultaneamente.

Persistência

- O Optuna permite salvar os resultados dos seus estudos/trials de forma que seja possível:
  - Retomar a otimização mais tarde.
  - Rodar trials em paralelo (vários processos acessando o mesmo estudo).
  - Analisar os resultados depois (relatórios, visualizações).
  - Garantir que os dados não sejam perdidos se o script for interrompido.

- Para isso, é necessário ter configurado um banco de dados a parte.

Perguntas?

Link para o quiz

https://forms.gle/oZfSZA71RD5UbYHj9

Referências

- AKIBA, Takuya; SANO, Shotaro; YANASE, Toshihiko; OHTA, Takeru; KOYAMA, Masanori. Optuna: A Next-generation Hyperparameter Optimization Framework. 2019. Preprint (arXiv:1907.10902) — Disponível em: https://arxiv.org/abs/1907.10902. Acesso em: 19 set. 2025.
- OPTUNA Contributors. *Optuna: A hyperparameter optimization framework* [Versão 4.5.0]. Disponível em: https://optuna.readthedocs.io/en/stable/index.html. Acesso em: 19 set. 2025.
- SCIPY. *Optuna: A Define by Run Hyperparameter Optimization Framework* [vídeo]. YouTube, 2019. Disponível em: https://www.youtube.com/watch?v=J_aymk4YXhg. Acesso em: 19 set. 2025.
- AUTO-TUNING HYPERPARAMETERS WITH OPTUNA AND PYTORCH [vídeo]. YouTube. Disponível em: https://www.youtube.com/watch?v=P6NwZVl8ttc. Acesso em: 19 set. 2025.

Obrigado!