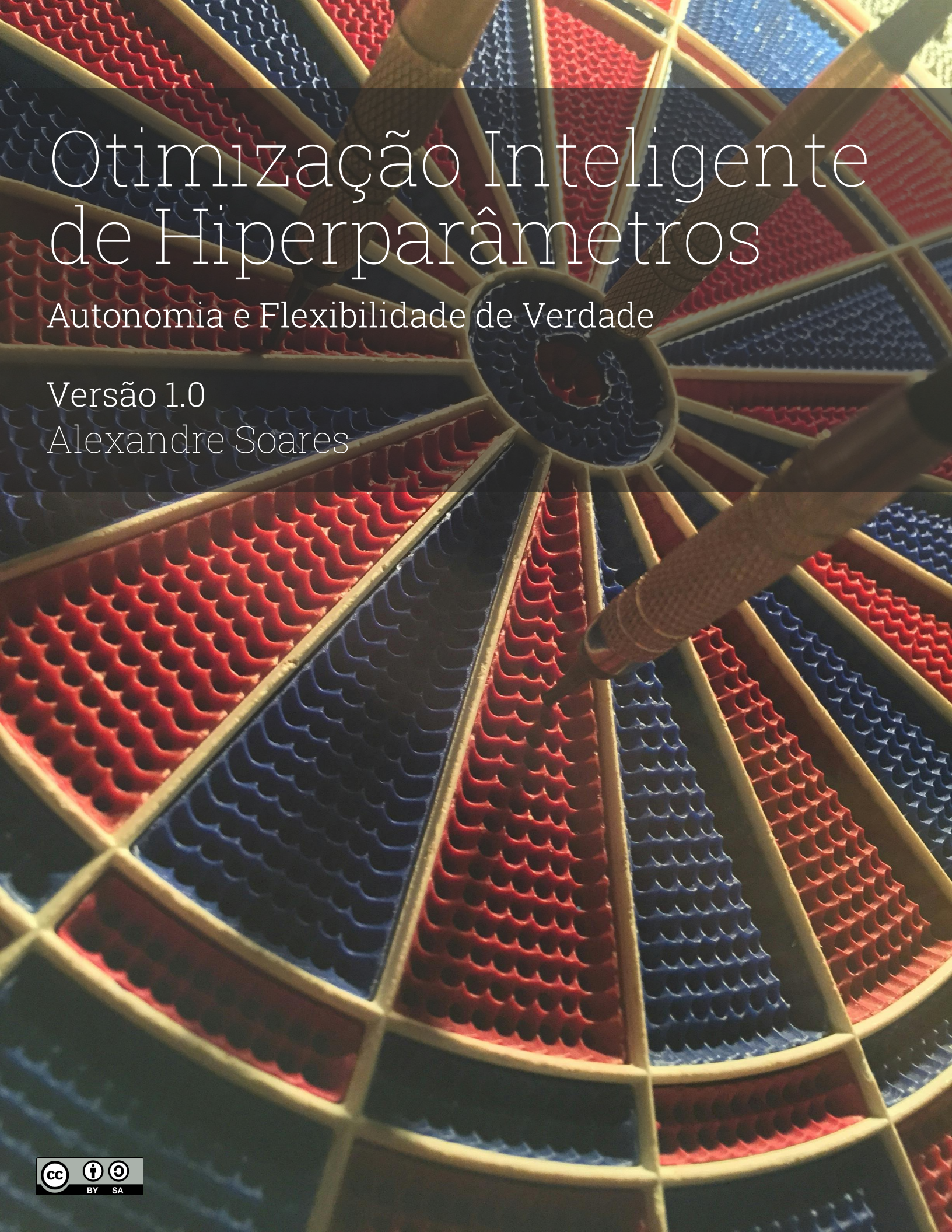

Otimização Inteligente de Hiperparâmetros

Autonomia e Flexibilidade de Verdade

Versão 1.0

Alexandre Soares

Otimização Inteligente de Hiperparâmetros

Autonomia e Flexibilidade de Verdade

por

Alexandre Soares

Notas Sobre o Estudo

Todos os programas fonte e dados utilizados para a elaboração deste estudo, bem como o PDF deste material, estão disponíveis para download no repositório GitHub, no endereço: https://github.com/Alxsoa/Artigos.

Conteúdo

Lista de Figuras

Lista de Tabelas

1

Motivação do Trabalho

1.1. Objetivo do Estudo

Este estudo tem como finalidade apresentar a superioridade do framework Optuna em comparação às técnicas convencionais de otimização de hiperparâmetros disponibilizadas pelo scikit-learn, mostrando de que maneira a utilização de ferramentas especializadas pode converter um processo historicamente ineficaz em uma solução robusta, flexível e eficiente em termos computacionais. O estudo realiza uma análise crítica que contrasta as restrições das metodologias tradicionais do scikit-learn com as funcionalidades aprimoradas do Optuna, fornecendo provas substanciais de que a otimização bayesiana constitui um avanço qualitativo relevante em ciência de dados.

A ineficácia das ferramentas convencionais do scikit-learn torna-se evidente, principalmente, nas implementações de GridSearchCV e RandomizedSearchCV, as quais possuem restrições essenciais. O GridSearchCV apresenta uma explosão combinatorial, o que torna inviável sua utilização em contextos reais que envolvem múltiplos hiperparâmetros, obrigando os profissionais a limitar artificialmente o espaço de busca. O RandomizedSearchCV, por sua vez, adota uma metodologia estritamente aleatória que consome recursos em áreas não promissoras, sem considerar as experiências passadas ou elaborar estratégias inteligentes de exploração.

Em comparação, o Optuna configura uma evolução paradigmática ao integrar a otimização bayesiana por meio de algoritmos avançados, que desenvolvem modelos probabilísticos a respeito da função objetivo, visando antecipar áreas promissoras dentro do espaço dos hiperparâmetros. A versatilidade do framework ultrapassa significativamente as restrições do scikit-learn, proporcionando um suporte nativo para otimização multiobjetivo, e implementa a poda adaptativa que suspende configurações com baixo potencial.

Os resultados deste estudo demonstram que a transição para o Optuna não se configura apenas como uma melhora incremental, mas sim como uma alteração substancial na metodologia de otimização de hiperparâmetros. Profissionais que utilizam o Optuna vivenciam não apenas melhorias quantificáveis no desempenho dos modelos, mas também uma diminuição considerável no tempo exigido para a otimização, aumento da satisfação profissional em razão da eliminação de atividades repetitivas e manuais, além de uma capacidade aprimorada para explorar espaços complexos que se mostrariam inviáveis com ferramentas convencionais.

2

Principais Desafios na Área de Machine Learning

2.1. A Importância dos Hiperparâmetros

Na área de ciência de dados, os hiperparâmetros exercem uma função crucial na eficácia, robustez e na habilidade de generalização dos modelos de aprendizado de máquina. Diferentemente dos parâmetros que são adquiridos de forma automática ao longo do treinamento, os hiperparâmetros são estipulados antecipadamente e regulam elementos essenciais do processo de modelagem, tais como a taxa de aprendizado, a quantidade de camadas em redes neurais, o tamanho do lote, a quantidade de árvores em métodos de ensemble e os coeficientes de regularização.

Esses fatores determinam o funcionamento do algoritmo, afetando sua rapidez de convergência, sua complexidade e sua habilidade de se ajustar aos dados. A seleção inadequada de hiperparâmetros pode resultar em underfitting ou overfitting, prejudicando a habilidade do modelo de realizar previsões eficazes em dados não observados. Em contrapartida, a escolha apropriada pode converter um modelo insatisfatório em uma solução extremamente eficiente.

A otimização de hiperparâmetros representa um dos principais desafios na área de machine learning, em virtude da amplitude e da complexidade do espaço de pesquisa. Técnicas como busca em grade, busca aleatória, algoritmos genéticos e enxame de partículas são frequentemente empregadas para investigar diversas combinações de maneira eficaz.

Recentemente, abordagens como o agendamento da taxa de aprendizado, a interrupção antecipada e os passos de aquecimento têm-se destacado, especialmente nos modelos de aprendizado profundo, nos quais pequenas alterações em hiperparâmetros podem influenciar de maneira significativa os resultados.

Além de impactar o desempenho, os hiperparâmetros exercem influência sobre a eficiência computacional e a interpretabilidade do modelo. Em contextos de produção, é fundamental equilibrar a precisão, a duração do treinamento, o consumo de recursos e a estabilidade.

A validação cruzada é fundamental para a avaliação robusta de combinações de hiperparâmetros, especialmente em conjuntos de dados reduzidos. Além do mais, a aplicação de visualizações, como gráficos de eixos paralelos, pode contribuir para a compreensão de como diversas configurações influenciam o desempenho.

A competência na seleção e ajuste de hiperparâmetros não constitui uma atividade isolada, mas é um componente essencial do fluxo de trabalho em ciência de dados. Trata-se de uma habilidade estratégica que demonstra uma compreensão abrangente de como os algoritmos aprendem e de como se pode extrair valor dos dados.

Conforme as ferramentas de automação se desenvolvem, o cientista de dados permanecerá essencial na elaboração de hipóteses, na interpretação de resultados e na tomada de decisões fundamentadas em evidências.

2.2. Porque é Chato Ajustar os Hiperparâmetros de Maneira Manual

O ajuste manual de hiperparâmetros é amplamente reconhecido como uma tarefa muito árdua, ineficiente e frustrante no desenvolvimento de modelos de machine learning. Essa prática exige que o cientista de dados percorra inúmeras combinações de valores por tentativa e erro, repetindo ciclos de treinamento e avaliação, o que consome um tempo considerável sem garantia de alcançar a configuração ideal. A Tabela-2.1, apresenta de modo estruturado as limitações e suas principais características.

Limitação	Descrição
Natureza repetitiva e exaustiva	O processo força o profissional a realizar tarefas operacionais mecânicas, como editar arquivos de configuração e registrar manualmente resultados, desviando o foco da análise estratégica e da interpretação dos dados.
Alta propensão a erros humanos	Erros como falhas de digitação, execuções com parâmetros incorretos ou confusão sobre combinações já testadas são comuns e comprometem a confiabilidade dos resultados.
Custo computacional elevado	Cada nova configuração exige o treinamento completo do modelo, o que se torna inviável em cenários com prazos curtos ou modelos complexos.
Dificuldade de reprodutibilidade e consistência	Pequenas variações não controladas entre execuções (como sementes aleatórias ou divisões de dados) dificultam a comparação entre resultados e reduzem a confiabilidade das conclusões.
Desorganização da informação	A ausência de ferramentas estruturadas leva ao acúmulo de anotações dispersas e perda de configurações promissoras, comprometendo a análise sistemática e a memória do projeto.
Exploração ineficiente do espaço de busca	A falta de uma estratégia formal resulta em ajustes baseados em intuição, frequentemente negligenciando parâmetros importantes.
Impacto psicológico negativo	O cansaço mental e a frustração com resultados pouco satisfatórios afetam a capacidade de análise e tomada de decisões.
Baixa escalabilidade	O gerenciamento manual de execuções paralelas é complexo, levando à subutilização de recursos computacionais e tempos de execução prolongados.
Isolamento e baixa colaboração	A centralização do processo em um único indivíduo dificulta o compartilhamento do progresso, distribuição de tarefas e uso de conhecimento coletivo.
Falta de aprendizado acumulado	Sem ferramentas para visualização e análise sistemática, o processo manual não contribui significativamente para o desenvolvimento de intuição reutilizável em projetos futuros.

Tabela 2.1: Principais limitações do ajuste manual de hiperparâmetros

Embora o ajuste manual possa proporcionar certo entendimento sobre o comportamento dos hiperparâmetros, ele é marcado por ineficiência, propensão a erros, dificuldade de reprodutibilidade e baixa escalabilidade. Em virtude disso, abordagens automatizadas e ferramentas específicas vêm sendo cada vez mais adotadas para tornar esse processo mais eficaz, confiável e sustentável no contexto da ciência de dados moderna.

2.3. Desafios e Estratégias na Otimização de Hiperparâmetros

A busca pelos hiperparâmetros ideais é um dos maiores desafios na prática da ciência de dados. Essa dificuldade decorre principalmente da alta dimensionalidade do espaço de busca, da interdependência entre os hiperparâmetros, da variabilidade estocástica dos algoritmos e da não transferibilidade das melhores configurações entre diferentes problemas ou conjuntos de dados.

O espaço de busca exponencialmente grande torna impraticável a avaliação exaustiva de todas as combinações possíveis, especialmente em modelos complexos como redes neurais profundas. Cada nova configuração exige o treinamento completo do modelo, o que gera custos computacionais elevados.

Além disso, pequenas alterações em um hiperparâmetro podem influenciar o efeito de outros de forma não linear, exigindo uma abordagem conjunta e estratégica para sua otimização. A Tabela-2.2 mostra as diversas técnicas têm sido desenvolvidas para lidar com esses desafios:

Método	Descrição
Grid Search	Abordagem sistemática e determinística que testa todas as combinações de uma grade predefinida. Embora garanta cobertura completa, é ineficiente para espaços grandes.
Random Search	Escolhe combinações aleatórias, frequentemente obtendo melhores resultados que o grid search, especialmente quando poucos hiperparâmetros influenciam significativamente o desempenho.
Otimização Bayesiana	Usa modelos probabilísticos para prever quais regiões do espaço de busca são mais promissoras com base em resultados anteriores. Ferramentas como Hyperopt, Scikit-Optimize e Ray Tune adotam essa abordagem.
Algoritmos Genéticos e Evolutivos	Inspirados na evolução biológica, exploram o espaço por meio de cruzamento, mutação e seleção. São eficazes em cenários altamente não lineares ou com múltiplos mínimos locais.
Hyperband e BOHB	Utilizam estratégias baseadas em alocação adaptativa de recursos e interrupção precoce (early stopping), concentrando esforços em configurações promissoras e aumentando a eficiência.
AutoML	Ferramentas como Auto-sklearn, TPOT e H2O AutoML automatizam não apenas o ajuste de hiperparâmetros, mas também a seleção de algoritmos e engenharia de features, democratizando o acesso a técnicas avançadas.
Paralelização e Computação Distribuída	Frameworks como Ray Tune, Dask eApache Spark possibilitam a execução simultânea de múltiplas tentativas, acelerando consideravelmente o processo.

Tabela 2.2: Principais estratégias de otimização de hiperparâmetros em ciência de dados

Apesar dessas inovações, o ajuste de hiperparâmetros continua sendo uma etapa delicada e essencial nos projetos de ciência de dados. A otimização ainda depende de experimentação cuidadosa, análise crítica e conhecimento especializado, uma vez que não há valores universais ótimos.

Além disso, aspectos como a interpretabilidade dos resultados, trade-offs entre desempenho e custo computacional, e a dificuldade de generalização entre domínios reforçam a importância de abordagens inteligentes, adaptativas e alinhadas ao contexto do problema.

2.4. Vantagens e Desvantagens das Principais Técnicas

O ajuste manual de hiperparâmetros é amplamente reconhecido como uma tarefa árdua, ineficiente e frustrante no desenvolvimento de modelos de machine learning. Essa prática exige que o cientista de dados percorra inúmeras combinações de valores por tentativa e erro, repetindo ciclos de treinamento e avaliação, o que consome um tempo considerável sem garantia de alcançar a configuração ideal.

Método	Vantagens	Desvantagens
Grid Search	Exaustivo, garantindo que todas as opções sejam avaliadas. Simples de implementar (disponível no scikit-learn via GridSearchCV).	Rapidamente se torna computacionalmente inviável à medida que o número de hiperparâmetros cresce. Pode desperdiçar tempo testando valores claramente subótimos.
Random Search	Mais eficiente que o Grid Search em espaços de alta dimensão. Pode encontrar boas soluções com menos iterações.	Não garante a melhor combinação, apenas uma combinação razoável. Pode perder regiões promissoras do espaço de busca.
Otimização Bayesiana	Muito mais eficiente que buscas aleatórias ou em grade. Aprende com iterações anteriores para focar em áreas promissoras.	Pode ficar preso em máximos locais se mal configurado. Requer mais conhecimento estatístico para ajustar adequadamente.
Algoritmos Evolucionários	Boa para espaços de busca complexos e não diferenciáveis. Pode escapar de máximos locais melhor que métodos bayesianos.	Pode ser lento devido ao grande número de avaliações necessárias. Mais difícil de controlar a convergência.

Tabela 2.3: Comparativo entre métodos de otimização de hiperparâmetros

A tabela-2.3 apresenta de maneira estruturada as vantagens e desvantagens dos principais métodos.

3

Otimização Automática dos Hiperparâmetros

3.1. Sobre o Optuna

O Optuna é uma biblioteca de código aberto para otimização automática de hiperparâmetros, amplamente utilizada em projetos de aprendizado de máquina e aprendizado profundo. Seu principal objetivo é encontrar, de forma eficiente, os melhores conjuntos de hiperparâmetros para modelos complexos, como redes neurais, modelos de árvore de decisão e até mesmo pipelines de processamento de dados.

A principal vantagem do Optuna em relação a outras abordagens de otimização (como grid search ou random search) está em sua estratégia de busca baseada em algoritmos de otimização bayesiana e Tree-structured Parzen Estimator (TPE), que permitem uma exploração mais inteligente e eficaz do espaço de busca.

O fluxo básico de uso do Optuna envolve a definição de uma função objetivo (chamada ‘objective‘) que recebe como entrada um conjunto de hiperparâmetros propostos pela biblioteca, treina o modelo e retorna uma métrica de desempenho (como acurácia, erro quadrático médio, AUC, etc). A função ‘study.optimize(objective, n_trials=...)‘ é então chamada para realizar a otimização, onde ‘n_trials‘ define o número de execuções experimentais. A cada trial, o Optuna atualiza seu conhecimento sobre o espaço de busca e sugere novas configurações com base nos resultados anteriores.

Além de sua eficiência, o Optuna possui recursos poderosos como: Pruning automático: interrompe experimentos pouco promissores precocemente para economizar tempo computacional; Visualizações integradas: permite gerar gráficos de importância de hiperparâmetros, evolução da otimização e distribuição de valores testados; Integração com frameworks populares: como PyTorch, TensorFlow, Scikit-Learn, LightGBM, XGBoost e outros;

Assim, o Optuna fornece uma estrutura robusta, escalável e inteligente para experimentação e busca automatizada de hiperparâmetros, contribuindo significativamente para a melhoria do desempenho de modelos de machine learning de forma prática e reprodutível.

3.2. Funcionamento Geral do Optuna

O funcionamento geral do Optuna é apresentado na Figura-3.1 este esquema funciona como um sistema inteligente de otimização de hiperparâmetros que automatiza o processo de busca pelos melhores valores para maximizar ou minimizar uma função objetivo. Seu principal objetivo é encontrar a melhor combinação de parâmetros para algoritmos de aprendizado de máquina de forma eficiente, flexível e automatizada. Ele se destaca por empregar uma abordagem moderna chamada “define-by-run”, que permite que a busca de hiperparâmetros seja adaptada dinamicamente durante a execução do código, sem exigir que o espaço de busca seja totalmente especificado antecipadamente.

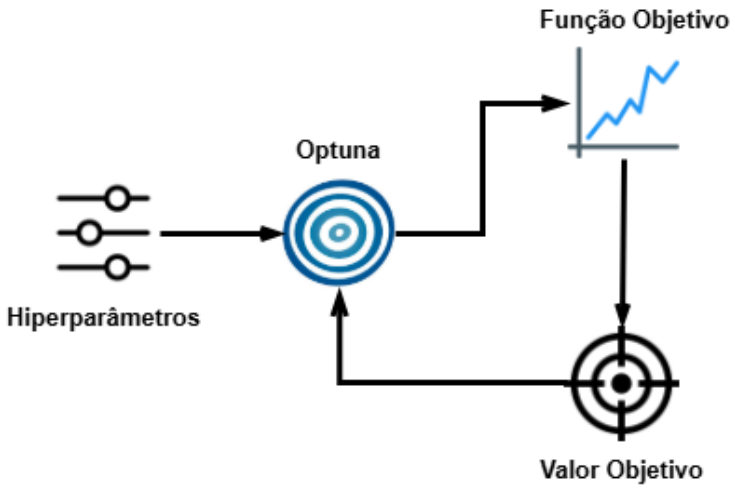

Figura 3.1: Funcionamento Geral do Optuna

O funcionamento geral do Optuna pode ser dividido em três componentes principais: o study, o trial e a função objective. O study é o objeto principal que gerencia o processo de otimização — ele registra os resultados das tentativas anteriores e decide quais configurações tentar a seguir. A função objetivo (objective function) define o problema de otimização e retorna um valor escalar (como um erro, perda ou métrica de avaliação negativa) que o Optuna tentará minimizar ou maximizar. Cada execução dessa função, com um conjunto diferente de hiperparâmetros sugerido pelo Optuna, é chamada de trial.

Durante o processo, o Optuna utiliza algoritmos de otimização baseados em técnicas como Tree-structured Parzen Estimator (TPE) ou simulated annealing, que são eficientes para explorar espaços de busca grandes e complexos. Além disso, o Optuna possui recursos como pruning automático de experimentos ruins, integração com frameworks populares (PyTorch, TensorFlow, XGBoost etc.), e suporte nativo à otimização distribuída, o que o torna altamente escalável.

Em resumo, o funcionamento do Optuna pode ser entendido como um laço de otimização inteligente onde, a cada tentativa, são testados novos valores de hiperparâmetros, com base nos resultados anteriores, buscando sempre encontrar a melhor configuração possível para o problema proposto. Isso o torna uma ferramenta poderosa e amplamente adotada em projetos de ciência de dados, machine learning e deep learning.

3.3. Utilizando o Optuna

O código desenvolvido tem como objetivo identificar a melhor combinação de hiperparâmetros para um modelo XGBoost aplicado a um problema de classificação multiclasse. Para isso, realiza testes com diferentes quantidades de tentativas, analisando como o número de iterações influencia a qualidade do modelo final.

Os resultados obtidos são armazenados para posterior análise, incluindo: os melhores hiperparâmetros encontrados, o histórico completo do processo de otimização e a comparação de desempenho entre diferentes quantidades de tentativas.

Importa as bibliotecas essenciais para realizar a otimização de hiperparâmetros, a construção do modelo, a geração e manipulação do conjunto de dados, além da visualização dos resultados.

```
import optuna
from optuna.visualization import plot_optimization_history
import xgboost as xgb
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
import numpy as np
import matplotlib.pyplot as plt
import warnings
import pandas as pd
```

Programa 3.1: Importando as Bibliotecas

Gera um conjunto de dados sintético contendo 10.000 amostras, 20 variáveis preditoras (sendo 15 informativas e 3 redundantes) e três classes, incorporando 10% de ruído nos rótulos.

```
X, y = make_classification(
        n_samples=10000,
        n_features=20,
        n_informative=15,
        n_redundant=3,
        n_classes=3,
        flip_y=0.1,
        random_state=42
    )
```

Programa 3.2: Criação do Dataset Sintético

Define os hiperparâmetros a serem otimizados com base nas sugestões fornecidas pelo Optuna. Utiliza validação cruzada estratificada com 5 divisões para avaliar o desempenho do modelo, retornando a acurácia média como métrica a ser maximizada durante o processo de otimização.

```
def objective(trial):
    params = {
        'objective': 'multi:softmax',
        'eval_metric': 'mlogloss',
        'booster': trial.suggest_categorical('booster', ['gbtree', 'dart']),
        'lambda': trial.suggest_float('lambda', 1e-8, 1.0, log=True),
        'alpha': trial.suggest_float('alpha', 1e-8, 1.0, log=True),
        'max_depth': trial.suggest_int('max_depth', 3, 12),
        'eta': trial.suggest_float('eta', 0.01, 0.3),
        'gamma': trial.suggest_float('gamma', 1e-8, 1.0),
        'grow_policy': trial.suggest_categorical('grow_policy', ['depthwise', '
            lossguide']),
        'subsample': trial.suggest_float('subsample', 0.5, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 1.0),
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
        'num_class': 3,
        'n_estimators': 200
    }

    if params['booster'] == 'dart':
        params['sample_type'] = trial.suggest_categorical('sample_type', ['uniform',
            'weighted'])
        params['normalize_type'] = trial.suggest_categorical('normalize_type', ['tree
            ', 'forest'])
        params['rate_drop'] = trial.suggest_float('rate_drop', 1e-8, 1.0)
        params['skip_drop'] = trial.suggest_float('skip_drop', 1e-8, 1.0)

    # Validação cruzada sem early stopping
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    model = xgb.XGBClassifier(**params)
    scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy', n_jobs=-1)

    return np.mean(scores)
```

Programa 3.3: Função Objetivo

Inicializa um estudo com o Optuna visando maximizar a acurácia do modelo. Realiza o processo de otimização com o número de tentativas definido e armazena os melhores resultados obtidos, assim como o histórico completo da otimização.

```
def run_optimization(n_trials):
    study = optuna.create_study(direction='maximize')
    study.optimize(objective, n_trials=n_trials)

    results = {
        'best_accuracy': study.best_value,
        'best_params': study.best_params,
        'all_trials': study.trials_dataframe(),
        'optimization_history': plot_optimization_history(study)
    }
    return results
```

Programa 3.4: Execução da Otimização

Seleciona a melhor execução entre todas as realizadas e exibe os hiperparâmetros que obtiveram o melhor desempenho.

```
best_overall = max(final_results.items(), key=lambda x: x[1]['best_accuracy'])
```

Programa 3.5: Análise Final

Cria um gráfico mostrando como a acurácia melhora com o aumento do número de tentativas.

```
plt.figure(figsize=(10, 6))
plt.plot(
    list(final_results.keys()),
    [res['best_accuracy'] for res in final_results.values()],
    marker='o',
    linestyle='--'
)
```

Programa 3.6: Visualização

3.4. Visualização dos Parâmetros da Otimização

Com base nas figuras apresentadas, que mostram o histórico de otimização com 100 e 150 tentativas utilizando o Optuna, é possível observar o comportamento do processo de busca pelos melhores hiperparâmetros ao longo dos experimentos.

Na Figura-3.2, mostra o resultado com 100 tentativas, percebe-se uma rápida elevação da métrica de desempenho nas primeiras execuções, seguida por uma estabilização em torno do valor máximo de acurácia. A linha vermelha, que representa o melhor valor encontrado até cada ponto, mostra que, após aproximadamente 30 tentativas, o ganho marginal com novas explorações se torna muito pequeno, indicando convergência prematura da otimização.

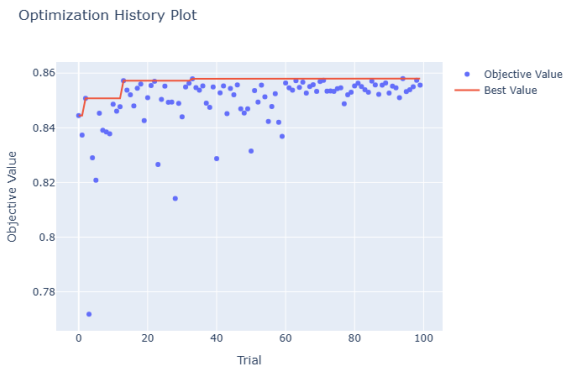

Figura 3.2: Otimização com 100 Tentativas

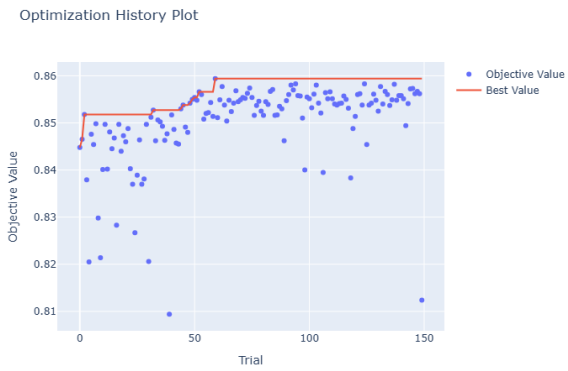

Figura 3.3: Otimização com 150 Tentativas

Na Figura-3.3, mostra o resultado com 150 tentativas, o comportamento inicial é semelhante, com melhora rápida nas primeiras execuções. No entanto, o maior número de tentativas permitiu explorar mais regiões do espaço de hiperparâmetros, o que resultou em uma leve melhoria do valor ótimo encontrado. Apesar disso, observa-se que o ritmo de melhoria diminui significativamente após a 60ª tentativa, sugerindo que o ganho adicional proporcionado por 50 execuções extras é modesto em relação ao custo computacional envolvido.

Essas visualizações reforçam a importância de definir um número adequado de tentativas no processo de otimização: poucas iterações podem resultar em soluções subótimas, enquanto execuções excessivas oferecem ganhos marginais com alto custo. O ponto de equilíbrio deve considerar os recursos disponíveis e a complexidade do problema.

3.5. Melhoria Geral do Modelo

A Figura-3.4 ilustra a evolução da melhor acurácia obtida pelo modelo à medida que se aumenta o número de tentativas (trials) no processo de otimização de hiperparâmetros.

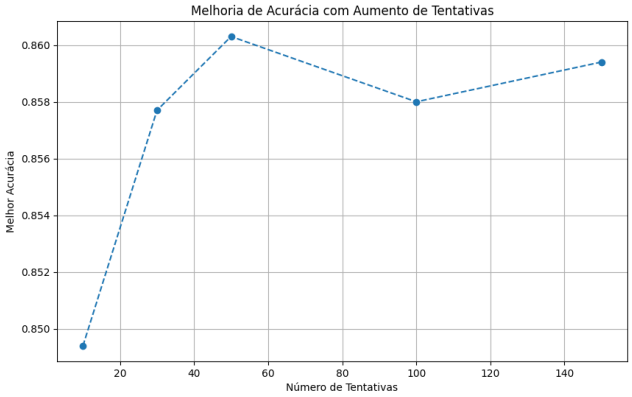

Figura 3.4: Evolução da Acurácia

É possível observar um ganho acentuado de desempenho entre 10 e 50 tentativas, indicando que as primeiras iterações são altamente eficazes para identificar boas regiões no espaço de busca. A acurácia sobe rapidamente de aproximadamente 0.849 para mais de 0.860 sugerindo que, nesse intervalo, a otimização é mais produtiva.

Entretanto, a partir da 50ª tentativa, os ganhos começam a se estabilizar. Notavelmente, com 100 tentativas ocorre uma leve queda na melhor acurácia, possivelmente devido à exploração de regiões menos promissoras do espaço. Com 150 tentativas a acurácia volta a subir levemente, embora sem superar o pico obtido com 50 trials.

Esse comportamento evidencia o fenômeno dos ganhos decrescentes após certo ponto, aumentos no número de tentativas tendem a gerar apenas pequenas melhorias — ou até oscilações — no desempenho final. Assim, para problemas com restrições de tempo ou recursos, limitar a otimização a algo entre 30 e 50 tentativas pode oferecer uma boa relação entre custo computacional e benefício obtido.