



# **Escrever título (escolher no final)**

**Universidade Federal da Paraíba - CCEN**

Gabriel de Jesus Pereira

7 de julho de 2024

# Índice

<b>1</b>	<b>Resumo</b>	<b>4</b>
<b>2</b>	<b>Capítulo 1</b>	<b>5</b>
2.1	Introdução . . . . .	5
2.2	Objetivos . . . . .	5
2.2.1	Objetivo Geral . . . . .	5
2.2.2	Objetivos Específicos . . . . .	5
2.3	Organização do Trabalho . . . . .	5
<b>3</b>	<b>Capítulo 2</b>	<b>6</b>
3.1	Recursos Computacionais . . . . .	6
3.1.1	Linguagem de Programação R . . . . .	6
3.1.2	Linguagem de Programação Python . . . . .	6
3.1.3	Quarto . . . . .	6
3.1.4	Linguagem de Programação Python . . . . .	6
3.1.5	Web Scraping . . . . .	6
<b>4</b>	<b>Métodos baseados em árvore</b>	<b>7</b>
4.1	Árvores de decisão . . . . .	7
<b>5</b>	<b>Metodologia</b>	<b>10</b>
5.1	Os dados e o procedimento adotado para sua obtenção . . . . .	10
5.2	Descritiva dos dados . . . . .	12
5.3	Aprendizado Supervisionado e não supervisionado . . . . .	12
5.3.1	Aprendizado supervisionado . . . . .	12
5.3.2	Aprendizado não supervisionado . . . . .	13
5.4	Reamostragem para avaliação de performance . . . . .	13
5.5	Métricas de avaliação . . . . .	13
5.6	Tunagem de hiperparâmetros . . . . .	13
5.6.1	Otimização Bayesiana . . . . .	14
5.6.2	Tree-Structured Parzen Estimator . . . . .	16
<b>6</b>	<b>Capítulo 4</b>	<b>18</b>
6.1	Resultados . . . . .	18

<b>7</b>	<b>Conclusão</b>	<b>19</b>
<b>8</b>	<b>Referências</b>	<b>20</b>

# 1 Resumo

## **2 Capítulo 1**

– Fazer antes da conclusão –

### **2.1 Introdução**

### **2.2 Objetivos**

#### **2.2.1 Objetivo Geral**

#### **2.2.2 Objetivos Específicos**

### **2.3 Organização do Trabalho**

## **3 Capítulo 2**

— Fazer depois dos modelos baseados em árvore —

### **3.1 Recursos Computacionais**

#### **3.1.1 Linguagem de Programação R**

#### **3.1.2 Linguagem de Programação Python**

#### **3.1.3 Quarto**

#### **3.1.4 Linguagem de Programação Python**

#### **3.1.5 Web Scraping**

## 4 Métodos baseados em árvore

Neste capítulo, serão descritos os métodos baseados em árvore, que fundamentam os algoritmos de aprendizado de máquina utilizados neste trabalho e que podem ser aplicados tanto para regressão quanto para classificação. Os métodos baseados em árvore envolvem a estratificação ou segmentação do espaço dos preditores<sup>1</sup> em várias regiões simples. Além disso, esses métodos são bastante simples, de fácil interpretação e, apesar de sua simplicidade, são poderosos. Alguns dos algoritmos de aprendizagem de máquinas mais conhecidos e que estão contidos nos métodos baseados em árvore e que foram utilizados nesse trabalho, é a Random Forest, Gradient Boosting e Light Gradient Boosting Machine. No entanto, para fundamentar os métodos baseados em árvore, começaremos pela definição da árvore de decisão.

### 4.1 Árvores de decisão

Árvores de decisão são métodos de aprendizado supervisionado não paramétrico utilizados tanto para regressão quanto para classificação. Elas servem de base para muitos dos modelos baseados em árvores empregados neste trabalho, uma vez que esses modelos geram múltiplas árvores de decisão. IZBICKI; SANTOS (2020) define o processo de construção de uma árvore como o particionamento recursivo no espaço das covariáveis, em que cada particionamento recebe o nome de nós e o resultado final recebe o nome de folha. Em cada nó é definida uma condição e, caso essa condição seja satisfeita, ter-se-á como resultado uma das folhas desse nó. Não obstante, caso o resultado seja contrário, seguirá para o próximo nó e verificará a próxima condição, podendo gerar uma folha ou a condição de outro nó. Veja um exemplo na Figura 4.1:

Descrevendo formalmente o processo de construção de uma árvore de regressão<sup>2</sup>, a sua execução é composta por dois passos. No primeiro passo, dividimos o espaço dos preditores em  $J$  regiões distintas e disjuntas denotadas por  $R_1, R_2, \dots, R_J$ . No segundo passo, para cada observação que pertence a região  $R_j$ , a previsão será a mesma. Essa previsão será simplesmente a média dos valores da variável dependente das observações de treinamento que estão dentro da região  $R_j$  (JAMES *et al.*, 2013). Dessa forma, dado que se tenha duas regiões  $R_1$  e  $R_2$ , e a média dos valores da variável resposta tenha sido

---

<sup>1</sup>O espaço dos preditores é o conjunto de todos os valores possíveis para as  $p$  variáveis  $X_1, X_2, \dots, X_p$ .

<sup>2</sup>Uma árvore de regressão é um caso específico da árvore de decisão, mas para regressão.

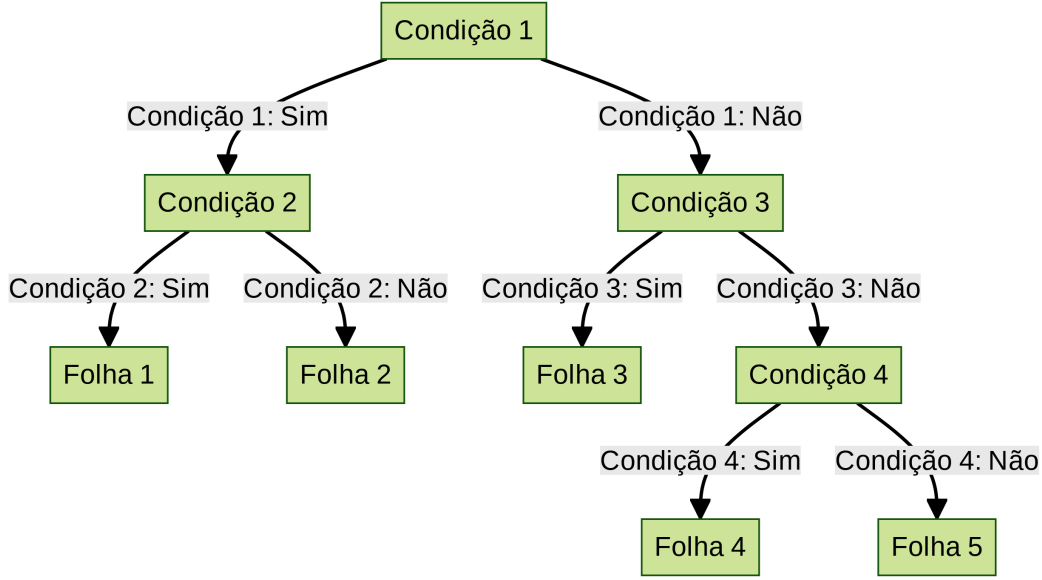


Figura 4.1: Exemplo de estrutura de árvore de regressão. A árvore tem cinco folhas e quatro nós internos.

10 e 20, respectivamente. Então, Para uma observação  $X = x, x \in R_1$ , o valor previsto será 10 e, caso contrário, se  $x \in R_2$ , o valor previsto será 20.

As regiões  $R_1, R_2, \dots, R_J$  são construídas em formato de caixa de forma a minimizar a soma dos quadrados. Dessa forma, podemos modelar a variável resposta como uma constante  $c_m$  em cada região  $R_j$ . Assim, definimos a resposta:

$$f(x) = \sum_{j=1}^J c_j I(x \in R_j)$$

Agora, utilizando o critério de minimizar a soma dos quadrados, deve-se minimizar  $\sum_{x_i \in R_j} [y_i - f(x_i)]^2$  para encontrar um estimador para o parâmetro  $c_j$ . No entanto, percebe-se que  $f(x_i)$  está sendo avaliado somente em um ponto específico  $x_i$ , o que reduzirá  $f(x_i)$  para uma constante  $c_j$ . É fácil de se chegar ao resultado se observarmos a definição da função indicadora  $I(x \in R_j)$ :

$$I_{R_j}(x_i) = \begin{cases} 1, & \text{se } x_i \in R_j \\ 0, & \text{se } x_i \notin R_j \end{cases}$$

e, como um ponto  $x_i$  não pode estar ao mesmo tempo em duas regiões  $R_j$ , pois as regiões são disjuntas, temos que apenas um dos casos a função indicadora será diferente de 0.



Portanto,  $f(x_i) = c_j$ . Assim, derivando  $\sum_{x_i \in R_j} (y_i - c_j)^2$

$$\frac{\partial \sum_{x_i \in R_j} (y_i - c_j)^2}{\partial c_j} = -2 \sum_{x_i \in R_j} (y_i - c_j) \quad (4.1)$$

e igualando Equação 4.1 a 0, temos a seguinte equação

$$\sum_{x_i \in R_j} (y_i - \hat{c}_j) = 0$$

que se abrirmos, o somatório e dividirmos pelo número total de pontos  $N_{R_j}$  na região  $R_j$ , teremos que o estimador de  $c_j$  será somente a média dos  $y_i$  na região  $R_j$ :

$$\sum_{x_i \in R_j} y_i - c_j \sum_{x_i \in R_j} 1 = 0 \implies \hat{c}_j = \frac{1}{N_{R_j}} \sum_{x_i \in R_j} y_i \quad (4.2)$$

## 5 Metodologia

### 5.1 Os dados e o procedimento adotado para sua obtenção

— AINDA SERÁ MODIFICADO —

O Web scraping, também conhecido como extração de dados da web, é uma técnica utilizada para o processo de coleta de dados estruturados da web de maneira automatizada. É um processo que vem sendo constantemente utilizado por instituições públicas e privadas para a construção de produtos que utilizam algoritmos de aprendizagem de máquinas, observa ofertas e descontos, faz análise de mercado ou monitoração de marcas.

Neste projeto, para fins de estudo e análise do mercado imobiliário, os dados foram coletados por meio de extração de dados do site do Zap Imóveis. O Zap Imóveis é um site do Grupo OLX que reúne ofertas do mercado imobiliário e que funciona como uma plataforma dinâmica para facilitar a conexão entre quem deseja alugar, comprar ou vender um imóvel; podendo servir também para corretores ou outros profissionais do setor de imóveis. Este projeto foi possível graças as informações que foram coletadas do site do Zap Imóveis em dois diferentes períodos do ano de 2023. O primeiro deles, as informações foram coletadas utilizando variados pacotes para raspagem de dados e proxies rotativas da linguagem de programação R, a fim de evitar ser bloqueado pelos mecanismos de segurança do site. Na segunda etapa, os dados foram coletados empregando a linguagem de programação Python com as bibliotecas Scrapy e Playwrite, que serve para web crawling e web scraping, e o Playwrite que serve para testes em aplicativos da web, mas que neste caso foi utilizado para manejar páginas dinâmicas.

Desta forma, com a ideia de modelar o valor do imóvel e analisar o mercado imobiliário, foram coletados aqueles variáveis que estavam disponíveis no site do Zap Imóveis e que poderiam de alguma forma ser significativas ao tentar explicar o valor do imóvel durante a sua modelagem. Assim, no total foram coletadas 23 variáveis, das quais 8 são quantitativas e 15 qualitativas nominais, sendo 13 de caráter dicotômico. No entanto, nem todas essas variáveis foram coletadas diretamente do Zap Imóveis, a latitude e longitude foram obtidas pela geocodificação do endereço utilizando o pacote tidygeocoder da linguagem de programação R. Portanto, temos as seguintes variáveis:

- Valor do imóvel: esta é a variável dependente, aquela que será modelada e será o principal objeto de estudo deste trabalho;
- Área: área do imóvel em  $m^2$ ;
- Condomínio: valor pago pelo condomínio;
- IPTU: imposto cobrado de quem tem um imóvel urbano;
- Banheiro: quantidade de banheiros presentes na propriedade;
- Vaga de estacionamento: quantidade total de vagas de estacionamento;
- Quarto: quantidade de quartos no imóvel;
- Latitude: posição horizontal medida em frações decimais de graus;
- Longitude: posição vertical que, assim como a latitude, é medida em frações decimais de graus;
- Tipo do imóvel: foram obtidos 7 tipos de imóveis, apartamentos, casas, casas comerciais, casas de condomínio, casas de vila, coberturas, lotes comerciais e de condomínio;
- Endereço: nome do endereço do imóvel;
- Variáveis dicotômicas que indicam se o imóvel tem ou não aquela característica (representado como 1 ou 0, respectivamente): área de serviço, academia, elevador, espaço gourmet, piscina, playground, portaria 24 horas, quadra de esporte, salão de festa, sauna, spa e varanda gourmet.

No entanto, devido a observações feitas durante o estudo, nem todas essas variáveis foram utilizadas para a modelagem do valor dos imóveis, seja por conter muitos valores ausentes ou por não ter se mostrado significativo para o que se desejava explicar. Ainda, como a coleta destes dados foram feitas em dois momentos distintos, temos dois bancos de dados, um com 29712 observações e o outro com 14956. Por fim, essas duas bases de dados foram unidas e, para não correr o risco de conter imóveis repetidos, aqueles que tinham o mesmo número de identificação foram removidos.

## 5.2 Descritiva dos dados

A análise exploratória de dados marca uma das primeiras etapas de qualquer estudo que utiliza a estatística como uma de suas principais ferramentas, pois permite encontrar padrões de comportamento no dados, descobrir relações entre as variáveis estudadas. Dessa forma, a primeira etapa desse estudo, após a coleta e organização dos dados obtidos do Zap Imóveis, foi fazer uma descritiva dos dados. Essa etapa permitiu encontrar padrões nos diferentes tipos de imóveis bem como o seu tipo pode influenciar na características do imóvel, o que, por consequência, pode afetar o seu valor. Assim, para identificar esses diferentes comportamentos, foram criados gráficos e tabelas a fim de caracterizar as relações das variáveis independentes com a dependente.

## 5.3 Aprendizado Supervisionado e não supervisionado

Na aprendizagem de máquinas, uma das etapas mais importantes é saber qual técnica será utilizada para resolver um problema que se enquadra em diferentes formas de aprendizado. Para isso, existem mais de uma forma em que um algoritmo consegue utilizar os dados e explicar o que está sendo modelado a partir deles. No entanto, a maioria dos problemas de aprendizado de máquinas recais em dois casos mais conhecidos: aprendizado supervisionado e não supervisionado.

### 5.3.1 Aprendizado supervisionado

Suponha uma regressão logística. Sabemos que na regressão logística temos um modelo com a seguinte forma  $Y_i = f(X) + \epsilon$ , em que  $Y_i$  assume 0 ou 1 para classificar o que está sendo modelado e representa a variável dependente,  $f(X)$  representa as variáveis independentes que serão utilizadas para a modelagem e  $\epsilon$  representa o erro da regressão. Dessa forma, podemos considerar o caso em que a regressão logística tenta classificar pacientes que podem ou não estar com diabetes. Para isso, utilizaríamos variáveis significativas para a classificação do estado de cada paciente. Esse exemplo é conhecido como aprendizagem supervisionada. Na aprendizagem supervisionada, busca-se aprender  $Y_i$  através de um exemplo. Nesse caso, as variáveis dependentes podem ser interpretadas como o exemplo, as informações de relações de pacientes que podem ter ou não diabetes, e o estado do paciente pode ser interpretado como o que se deseja aprender. Este processo é entendido como *aprendizado por exemplo*, HASTIE *et al.* (2009). O aprendizado supervisionado pode aparecer em casos de regressão linear, regressão logística, ou até mesmo em métodos mais modernos, como GAM, boosting e máquina de vetores de suporte, JAMES *et al.* (2013).

### 5.3.2 Aprendizado não supervisionado

Por outro lado, o aprendizado não supervisionado aparece em situações mais desafiadoras, pois não há um exemplo para explicar aquilo que se pretende explicar. Este processo é conhecido como *aprendizado sem exemplo*, HASTIE *et al.* (2009). Dessa forma, no aprendizado não supervisionado, tem-se uma amostra com  $N$  observações  $(x_1, \dots, x_N)$  de um vetor aleatório  $X$  com densidade conjunta  $f(x)$  em que o objetivo é inferir propriedades da densidade sem ajuda de exemplos para cada observação. Assim, como há uma falta de uma variável resposta  $y_i$  para supervisionar a análise, pode-se procurar entender a relação entre as variáveis ou as observações, JAMES *et al.* (2013). Por exemplo, uma das técnicas mais aplicadas em problemas que envolvem o aprendizado supervisionado é a análise de cluster, em que o objetivo é determinar, com base em  $x_1, \dots, x_n$ , se as observações são caracterizadas em grupos distintos. Esse é um dos métodos que poderiam ser aplicados, por exemplo, na análise de crédito de clientes de um cartão de crédito, tornando possível analisar o seu perfil e classificá-lo em diferentes grupos para recomendar produtos específicos adequados ao seu perfil.

## 5.4 Reamostragem para avaliação de performance

## 5.5 Métricas de avaliação

## 5.6 Tunagem de hiperparâmetros

Na aprendizagem de máquina, uma das etapas fundamentais é a tunagem dos hiperparâmetros dos algoritmos de aprendizagem. Essa etapa consiste em encontrar a melhor combinação de hiperparâmetros e, conseqüentemente, resultando em uma configuração algoritmo que proporciona melhor performance e capacidade de generalização. No entanto, essa configuração não é trivial.

Os algoritmos de otimização de hiperparâmetros procuram pelo melhor ajuste de hiperparâmetros  $\lambda \in \tilde{\Lambda}$  para um algoritmo de aprendizagem  $I_\lambda$ . O espaço de procura

$\tilde{\Lambda} \in \Lambda$  contém todas as possíveis configurações de hiperparâmetros consideradas para otimização. Dessa forma, temos que:

$$\tilde{\Lambda} = \tilde{\Lambda}_1 \cup \tilde{\Lambda}_2 \cup \dots \cup \tilde{\Lambda}_l \quad (5.1)$$

em que  $l$  são todas as possíveis configurações de hiperparâmetros. Além disso,  $\tilde{\Lambda}_i$  ( $i = 1, 2, \dots, l$ ) representam o subconjunto de limites dos domínios do  $i$ -ésimo hiperparâmetro  $\Lambda_i$ , podendo ser contínuo, discreto ou categórico (BISCHL et al., 2023).

A busca pela melhor combinação de hiperparâmetros  $\lambda$  é feito de forma iterativa, utilizando métodos de reamostragem para dividir o conjunto de dados entre teste e treinamento. Portanto, para validar e encontrar os hiperparâmetros, o algoritmo de aprendizagem é validado através da divisão do conjunto de dados. Assim, tem-se a seguinte estratégia de separação do conjunto de dados:

$$\mathcal{J} = ((J_{treino,1}, J_{teste,1}), \dots, (J_{treino,B}, J_{teste,B})) \quad (5.2)$$

onde  $J_{treino,i}$ ,  $J_{teste,i}$  representam os vetores de índices ( $i = 1, 2, \dots, B$ ) para os conjuntos de treino e teste, respectivamente, e  $B$  representa o número total de divisões.

A motivação para fazer a divisão do conjunto de dados entre treino e teste é ajustar o algoritmo em dados reais e avaliar a sua performance em dados ainda não vistos, processo representado pelo banco de treino e teste, respectivamente. Para fazer a avaliação dessa performance é necessário estimar uma métrica de performance em cada divisão. Assim, para um dado algoritmo  $I_\lambda$  é calculado uma métrica de performance  $\rho$  para cada  $J_{teste,i}$  após o ajuste do algoritmo no  $J_{treino,i}$ . Por fim, pode ser calculado uma média amostral da métrica de cada uma das divisões. Dessa forma, o problema de otimizar os hiperparâmetros pode ser definida da seguinte forma:

$$\lambda^* \in \underset{\lambda \in \tilde{\Lambda}}{\operatorname{argmin}} c(\lambda)$$

onde  $\lambda^*$  denota a melhor combinação possível de hiperparâmetros,  $c(\lambda)$  representa a generalização do erro, podendo ser representado também por  $\widehat{GE}(I, \mathcal{J}, \rho, \lambda)$ , quando  $I$ ,  $\mathcal{J}$ ,  $\rho$  são fixados e  $I$  denota um algoritmo de aprendizagem de máquina. O erro de generalização é estimado e otimizado a fim de evitar um ajuste excessivo (overfitting), podendo prejudicar quando o algoritmo fizer estimativas em dados ainda não vistos. Assim, o processo pode ser visualizado pela figura Figura 5.1:

### 5.6.1 Otimização Bayesiana

Existem diversas técnicas para otimização de hiperparâmetros utilizadas em aprendizagem de máquina. Uma das técnicas mais comuns é o GridSearch. BISCHL

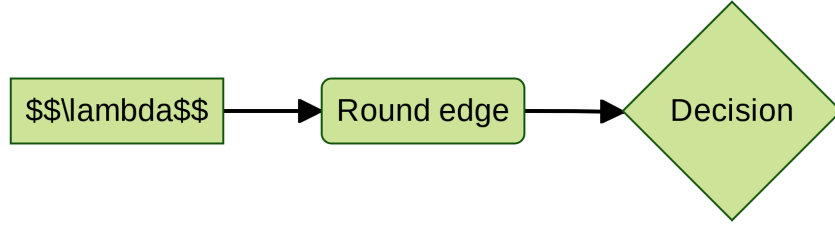


Figura 5.1: Processo de otimização de hiperparâmetros

*et al.* (2023) definem GridSearch como um processo que divide o intervalo contínuo de valores possíveis de cada hiperparâmetro em um conjunto de valores específicos e avalia exaustivamente o algoritmo para todas as combinações possíveis. No entanto, como todas as combinações possíveis aumentam exponencialmente com a quantidade necessária para avaliação do algoritmo, o GridSearch tem um custo computacional bastante elevado. Assim, existem algoritmos de otimização mais sofisticados que entregam melhores performances, como a otimização bayesiana, que foi utilizada neste trabalho.

A otimização bayesiana não se refere a um tipo específico de algoritmo de otimização, mas sim a uma filosofia de otimização baseada em inferência bayesiana, a qual contém uma extensa família de algoritmos de otimização (GARNETT, 2023). Não obstante, a otimização bayesiana tem obtido benchmarks melhores que outros algoritmos em inúmeros problemas complexos de otimização de hiperparâmetros (SNOEK; LAROCHELLE; ADAMS, 2012).

Diferente de outros algoritmos de otimização de hiperparâmetros, a otimização bayesiana determina as futuras tentativas de avaliação com base em resultados obtidos previamente (YANG; SHAMI, 2020). Para a definição dos pontos futuros, é utilizada uma função probabilística  $P(\rho|\lambda)$  (BERGSTRA; YAMINS; COX, 2013). Assim, após o ajuste da função probabilística, tem-se como resultado para cada  $\lambda$  uma estimativa da performance  $\hat{c}(\lambda)$  e da predição da incerteza  $\hat{\sigma}(\lambda)$ , além de obter também a distribuição preditiva da função probabilística. Com a distribuição obtida, uma função de aquisição determina o trade-off entre exploitation e exploration<sup>1</sup>. Dessa forma, os algoritmos de otimização bayesiana são definidos segundo a lei  $\lambda \rightarrow c(\lambda)$  e procuram um equilíbrio entre o processo de exploitation-exploration para detectar as regiões ótimas mais prováveis e não perder melhores configurações em áreas ainda não exploradas.

---

<sup>1</sup>exploitation pode ser entendido como procurar próximo a boas observações e

### 5.6.2 Tree-Structured Parzen Estimator

Existem diversas funções probabilísticas para uso na otimização bayesiana, algumas delas é o Processo Gaussiano, Random Forest ou Tree-Structured Parzen Estimator (TPE). Nesse trabalho, foi utilizado o Tree-Structured Parzen Estimator, utilizando a biblioteca Optuna (AKIBA *et al.*, 2019) para sua aplicação.

O TPE define duas funções,  $l(x)$  e  $g(x)$ , que são usadas para modelar a distribuição das variáveis do domínio (YANG; SHAMI, 2020). Utilizando as duas densidades, o TPE procura modelar a probabilidade de se observar um hiperparâmetro  $x$  dado uma métrica de performance  $\rho$ . Dessa forma, tem-se a seguinte definição:

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases} \quad (5.3)$$

em que  $l(x)$  é definido como a densidade em que a função perda é menor que um limiar  $y^*$  e  $g(x)$  representa a densidade em que a função perda<sup>2</sup> tem valores acima do limiar  $y^*$  (BERGSTRA *et al.*, 2011). O limite  $y^*$  é escolhido através de um hiperparâmetro  $\gamma$ , onde  $\gamma$  representa o percentil dos valores observados de  $y$ , de modo que  $p(y < y^*) = \gamma$ .

Por padrão, o tree-structured parzen estimator tem como função de aquisição o Expected Improvement ( $EI$ ), que pode ser otimizado para o TPE da seguinte forma:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy \quad (5.4)$$

Ainda, para encontrar a probabilidade marginal de  $x$ , temos a seguinte integral  $p(x) = \int_{\mathbb{R}} p(x|y) p(y) dy$ . Particionando o domínio de  $y$ , chega-se em:

$$p(x) = \int_{-\infty}^{y^*} p(x|y) p(y) dy + \int_{y^*}^{\infty} p(x|y) p(y) dy = \gamma l(x) + (1 - \gamma) g(x)$$

Assim, utilizando o Teorema de Bayes e fazendo as substituições na integral da Equação 5.4:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y) p(y)}{p(x)} dy = \left( \gamma y^* - \int_{-\infty}^{y^*} p(y) dy \right) \left( \gamma + (1 - \gamma) \frac{g(x)}{l(x)} \right)^{-1}$$

em que a segunda expressão do produto mostra que para maximizar o Expected Improvement é necessário pontos de  $x$  com maior probabilidade em  $l(x)$  e com baixa probabilidade em  $g(x)$ . Não obstante, no TPE, maximizar o  $EI$  é equivalente a

---

<sup>2</sup>definição de função perda



maximizar a razão entre as duas distribuições  $r(x) = \frac{l(x)}{g(x)}$  (COWEN-RIVERS *et al.*, 2022).

## **6 Capítulo 4**

### **6.1 Resultados**

## 7 Conclusão

## 8 Referências

AKIBA, T. *et al.* Optuna: A Next-generation Hyperparameter Optimization Framework. [S.l.]: [s.n.], 2019.

BERGSTRA, J. *et al.* Algorithms for hyper-parameter optimization. **Advances in neural information processing systems**, 2011. v. 24.

\_\_\_\_\_; YAMINS, D.; COX, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. [S.l.]: PMLR, 2013. p. 115–123.

BISCHL, B. *et al.* Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, 2023. v. 13, n. 2, p. e1484.

COWEN-RIVERS, A. I. *et al.* Hebo: Pushing the limits of sample-efficient hyperparameter optimisation. **Journal of Artificial Intelligence Research**, 2022. v. 74, p. 1269–1349.

GARNETT, R. **Bayesian optimization**. [S.l.]: Cambridge University Press, 2023.

HASTIE, T. *et al.* **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. V. 2.

IZBICKI, R.; SANTOS, T. M. Dos. **Aprendizado de máquina: uma abordagem estatística**. [S.l.]: Rafael Izbicki, 2020.

JAMES, G. *et al.* **An introduction to statistical learning**. [S.l.]: Springer, 2013. V. 112.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. **Advances in neural information processing systems**, 2012. v. 25.

YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms:

Theory and practice. **Neurocomputing**, 2020. v. 415, p. 295–316.