UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS DOIS VIZINHOS
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

DANIEL DE OLIVEIRA SILVA

OTIMIZAÇÃO DE HIPER-PARÂMETROS DE ALGORITMOS DE MACHINE LEARNING APLICADO NO CONTEXTO DE ANÁLISE DE RISCO DE CRÉDITO

TRABALHO DE CONCLUSÃO DE CURSO DE ESPECIALIZAÇÃO

DOIS VIZINHOS
2022

DANIEL DE OLIVEIRA SILVA

OTIMIZAÇÃO DE HIPER-PARÂMETROS DE ALGORITMOS DE MACHINE LEARNING APLICADO NO CONTEXTO DE ANÁLISE DE RISCO DE CRÉDITO

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Orientador: Prof. Dr. Francisco Carlos Monteiro Souza
Coorientador: Prof. Dr. Rafael Gomes Mantovani

DOIS VIZINHOS
2022

DANIEL DE OLIVEIRA SILVA

OTIMIZAÇÃO DE HIPER-PARÂMETROS DE ALGORITMOS DE MACHINE LEARNING APLICADO NO CONTEXTO DE ANÁLISE DE RISCO DE CRÉDITO

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Data de aprovação: 08/novembro/2022

Francisco Carlos Monteiro Souza
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos

André Roberto Ortoncelli
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos

Rodolfo Adamshuk Silva
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos

DOIS VIZINHOS
2022

AGRADECIMENTOS

Aos professores Francisco Carlos e Rafael pela orientação em todas as etapas deste trabalho, principalmente por sempre estarem disponíveis para sanar dúvidas e orientar durante o processo de desenvolvimento da aplicação e escrita do trabalho. Também a minha esposa, que sempre acreditou e esteve ao meu lado durante todo o curso de especialização.

RESUMO

A atividade de análise de risco de crédito é importantíssima em diversos setores em que se oferece um produto creditado. Uma área explorada há muito tempo, a análise de risco de crédito visa classificar certeiramente um cliente como um bom ou mau pagador, a fim de realizar bons negócios e evitar prejuízos. Diversas técnicas e métodos são aplicadas para realizar a análise de crédito, sendo estes métodos tradicionais utilizando modelos estatísticos ou a utilização de técnicas de aprendizado de máquina. Visando cada vez mais aumentar a margem de acerto, a utilização de algoritmos de aprendizado de máquina tem melhorado cada vez mais os resultados das análises. Entretanto, a otimização destes algoritmos pode ser um fator importantíssimo para se obter um melhor resultado nas predições. Com isto, o presente trabalho visa em realizar uma análise do desempenho de diferentes tipos de algoritmos aplicados no contexto de análise de risco de crédito, utilizando 3 diferentes técnicas de otimização de hiper-parâmetros. Para a realização do mesmo, foram selecionados os algoritmos: regressão logística, árvore de decisão, MLP (*Multilayer Perceptron*), KNN(k-Nearest Neighbors), *SVM (Support Vector Machines)*, *Naive Bayes*, *Random Forest* e *XGBoost* aplicados em quatro bases de dados voltadas para análise de risco de crédito. Para realizar a otimização de hiper-parâmetros, foram selecionados as técnicas: *Bayesearch, Randomsearch* e algoritmo genético. A divisão dos dados entre treino e teste foi realizada com a técnica de validação cruzada aninhada, sendo 10 *folds* para verificar o desempenho dos algoritmos e 2 *folds* para realizar a otimização de hiper-parâmetros. Para todos os processos, foi utilizado a métrica Curva ROC e AUC. Ao realizar os testes, o algoritmo *Random Forest* obteve os melhores resultados comparado aos demais algoritmos em todas as bases de dados, onde este resultado somente foi possível devido à otimização de seus hiper-parâmetros, destacando-se a técnica de otimização por algoritmo genético. Com isto, é possível concluir que a otimização de hiper-parâmetros é uma técnica extremamente válida e importantíssima ao aplicar um modelo de aprendizado de máquina no contexto de análise de risco de crédito, tendo em vista que é uma área onde uma pequena melhora no desempenho do modelo, resulta em um grande aumento na assertividade de novas predições.

Palavras-chave: Análise de risco de crédito; otimização; algoritmos; aprendizado de máquina.

ABSTRACT

The activity of credit risk analysis is very important in several sectors in which a credited product is offered. An area explored for a long time, credit risk analysis aims to accurately classify a customer as a good or bad payer, in order to do good business and avoid losses. Several techniques and methods are applied to perform credit analysis, being these traditional methods using statistical models or the use of machine learning techniques. Aiming to increasingly increase the margin of success, the use of machine learning algorithms has increasingly increased the results of analysis. However, the optimization of these algorithms can be a very important factor to obtain a better result in the predictions. With this, the present work aims to carry out an analysis of the performance of different types of algorithms applied in the context of credit risk analysis, using 3 different hyper-parameter optimization techniques. The following algorithms were selected: logistic regression, decision tree, MLP (*Multilayer Perceptron*), KNN(k-Nearest Neighbors), *SVM (Support Vector Machines)*, *Naive Bayes*, *Random Forest* and *XGBoost* applied to four databases focused on credit risk analysis. To perform the hyper-parameter optimization, the following techniques were selected: *Bayesearch, Randomsearch* and genetic algorithm. Data division between training and testing was performed using the nested cross-validation technique, with 10 *folds* to verify the performance of the algorithms and 2 *folds* to perform the hyper-parameter optimization. For all processes, the metric Curve ROC and AUC was used. When performing the tests, the algorithm *Random Forest* obtained the best results compared to the other algorithms in all databases, where this result was only possible due to the optimization of its hyper-parameters, highlighting the technique of optimization by genetic algorithm. With this, it is possible to conclude that hyper-parameter optimization is an extremely valid and very important technique when applying a machine learning model in the context of credit risk analysis, given that it is an area where a small improvement in performance of the model, results in a large increase in the assertiveness of new predictions.

Keywords: Credit risk analysis; optimization; algorithms; machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da economia, bem como a facilidade em processar e armazenar informações, o termo *big data* tem se popularizado no setor de finanças como uma ferramenta para tomada de decisões. Com o grande volume de dados gerados diariamente, a aplicação de métodos de ciência de dados tem crescido exponencialmente. Observando rotinas das instituições financeiras em específico, a de direcionar um melhor produto ao cliente ou selecionar a melhor forma de tratativa de empréstimo conforme o seu perfil, a aplicação e utilização de técnicas e algoritmos de aprendizado de máquina tem substituído técnicas e métodos manuais utilizadas anteriormente, as quais não possuíam tanta assertividade comparado com os modelos de IA (inteligência artificial).

Uma das questões mais abordadas em instituições financeiras é a análise e aprovação de crédito. A prática da realização de empréstimos e aplicação de juros é algo antigo registrado na Babilônia no ano de 2000 antes de Cristo (LEWIS, 1992). Com o avanço da globalização, bem como o aumento das necessidades pessoais, a partir do ano de 1800, a análise de crédito passou a ser uma atividade comum nas empresas e instituições, onde o aumento do número de financiamentos cresceu significativamente, principalmente devido à crescente popularização de automóveis no ano de 1920 (THOMAS; CROOK; EDELMAN, 2017).

Com a popularização da prática de adquirir crédito, surgiu-se a necessidade de desenvolver métodos para identificar riscos e certificar-se de que é seguro realizar o empréstimo a determinada pessoa. Essa análise tornou-se algo imprescindível para as instituições financeiras e qualquer empresa que ofereça um produto creditado. Devido ao não pagamento de empréstimos, pequenas e médias empresas terão um grande deficit em seu orçamento, consequentemente afetando o preço de seus produtos e serviços ((EXPERIAN, 2020b); (EXPERIAN, 2020a)).

Atualmente, um dos modelos mais utilizados pelas instituições bancárias em relação à mensuração quantitativa do risco de crédito é conhecido como metodologias clássicas de padrões lineares, destacando-se o método de regressão logística (CHEN; CHENG, 2013). Com tamanho aumento na geração de dados e o avanço nos estudos de inteligência artificial, bem como o desenvolvimento e aperfeiçoamento de algoritmos de aprendizado de máquina, o setor financeiro tem olhado para este segmento como um aliado em realizar melhores predições em relação aos métodos anteriormente utilizados. Os métodos e modelos de aprendizado de máquina possuem maior precisão ao analisar grande volume de dados, uma questão que hoje, principalmente para instituições financeiras, é algo produzido diariamente e com facilidade devido ao aumento de processamento e larga capacidade de armazenamento. Com isto, pesquisas sobre aplicação de algoritmos de aprendizado de máquina tem sido o foco de maior interesse (BURRELL, 2016).

Ao realizar uma breve pesquisa na literatura, encontram-se facilmente diversos trabalhos relacionados a aplicação de algoritmos de *machine learning* no contexto de análise e aprovação de risco de crédito, propondo a utilização dos mesmos ao invés de técnicas anteriormente

utilizadas. Entretanto, muitos destes algoritmos não atinge a melhor performance devido a não otimização de seus hiper-parâmetros, consequentemente não alcançando seu desempenho máximo. Observando tal cenário, este trabalho propõe um estudo visando analisar a performance de algoritmos de aprendizado de máquina no contexto de aprovação de crédito, aplicando a otimização de hiper-parâmetros e comparando seus resultados.

Para realização deste trabalho, foram definidas as seguintes etapas: i) Verificar o desempenho dos algoritmos: regressão logística, árvore de decisão, *SVM (support vector machine)*, *KNN (k-nearest neighbors)*, *MLP (Multilayer perceptron)*, *Naive Bayes* e os métodos ensemble *Random Forest* e *XGBoost*, de maneira não otimizada, em quatro bases de dados relacionadas a aprovação de crédito; ii) Aplicar as técnicas de otimização de hiper-parâmetros *Bayesearch, Randomsearch* e algoritmo genético ajustando os hiper-parâmetros e aplicando os algoritmos novamente as bases; iii) Comparar o desempenho de ambos os casos e verificar qual algoritmo e técnica mais se destacou no processo de análise.

O restante do trabalho divide-se nos seguintes capítulos: no Capítulo 2, primeiramente será apresentado as técnicas e modelos utilizados neste trabalho e posteriormente a descrição de diversos trabalhos que aplicaram algoritmos de *machine learning* no contexto de análise de risco de crédito e que motivaram a elaboração deste estudo. No Capítulo 3, será explicado quais técnicas, modelos, ferramentas e as bases de dados que foram utilizadas para o desenvolvimento do trabalho. No Capítulo 4 será abordado os resultados do trabalho, onde serão apresentados e comparados o resultado obtido com cada técnica e algoritmo utilizado e por fim, no Capítulo 5 serão apresentadas as considerações finais e sugestões para trabalhos futuros.

2 Fundamentação Teórica

Na primeira seção deste Capítulo, será abordado a definição de pontuação de crédito e uma breve revisão de métodos utilizados anteriormente. Logo após, será descrito o conceito de aprendizado de máquina, algoritmos, métricas e técnicas de otimização de hiper-parâmetros utilizados no trabalho. Na segunda seção, serão abordados trabalhos relacionados ao tema e que motivaram a elaboração deste.

2.1 Análise de risco de crédito

Cada vez mais, empresas e instituições financeiras tem voltado a atenção para o desenvolvimento de novas técnicas e modelos que aumentam a assertividade quanto ao processo de realização de empréstimos. A classificação errônea de um mau pagador traz consequentemente diversos prejuízos financeiros ao credor sendo dificilmente recuperáveis, da mesma forma que classificar um bom pagador como ruim, perdendo a oportunidade de negócio. Portanto, o aumento de assertividade precisa estar relacionado em tomar decisões de forma rápida e dinâmica, um dos critérios almejados em um bom sistema de classificação de análise de crédito. Para se ter um aumento na assertividade, é necessário analisar as informações quantitativas de maneira rápida, assim como usufruir da quantidade e qualidade das informações, para assim formar a pontuação do tomador (*scores*) de forma clara e assertiva.

Credit scoring é definido como sendo um processo responsável por converter informações do tomador em pontuação (*score*), a fim de classificá-lo como um bom ou mau pagador (LEWIS, 1992). A utilização de tal técnica é essencial para as empresas e instituições, pois é a partir deste processo que são definidos e sustentadas as possibilidades de ganhos e perdas. Entretanto, apesar de representar um processo estatístico, isto não inibe a possibilidade de classificar erroneamente um bom pagador ou oferecer crédito a um mau pagador.

Observando o histórico das técnicas utilizadas para análise de crédito, a partir da década de 70 a regressão logística e a análise discriminante foram utilizadas predominantemente. Na década de 90, com o surgimento das redes neurais, começaram a ser utilizadas técnicas de aprendizado de máquina, uma subárea da inteligência artificial, com o objetivo de melhorar o desempenho das técnicas estatísticas já utilizadas (ALTMAN; SAUNDERS, 1997).

Anteriormente, a tarefa de análise de crédito era realizada unicamente por profissionais especialistas, tendo uma alta demanda de trabalho. Com a utilização de algoritmos e técnicas de aprendizado de máquina, esta tarefa tem sido distribuída, reduzindo assim a demanda de tempo e trabalho (MUNKHDALAI et al., 2019).

Com a aplicação de métodos estatísticos e a utilização de algoritmos de aprendizado de máquina, as análises de crédito tornaram-se cada vez mais assertivas e dinâmicas, garantindo assim que a instituição que possui um bom sistema de avaliação de crédito aumente o

gerenciamento cada vez mais de sua carteira de clientes, consequentemente evitando prejuízos e adquirindo cada vez mais bons negócios (COELHO et al., 2021).

2.2 Técnicas de aprendizado de Máquina

O aprendizado de máquina é uma subárea da inteligência artificial, onde são aplicados métodos e técnicas de aprendizado sob determinado problema, a fim de extrair conceitos com base em um modelo de informações. Um sistema de aprendizado toma decisões com base em experiências obtidas por meio de soluções bem sucedidas realizadas anteriormente (MONARD; BARANAUSKAS, 2003).

A criação de cada modelo de aprendizado de máquina está relacionado em duas categorias, podendo ser classificado como supervisionado ou não-supervisionado. Os modelos supervisionados fundamentam-se no aprendizado com dados rotulados, onde já se tem o conhecimento da classificação dos dados, gerando tarefas de regressão ou classificação. Já os não-supervisionados trabalham com dados não rotulados, onde desconhece sua categorização. Para este tipo de modelo, é utilizado técnicas de agrupamento de dados (*clustering*), onde a tarefa é organizar/agrupar dados semelhantes de acordo com suas características.

2.2.1 Regressão logística

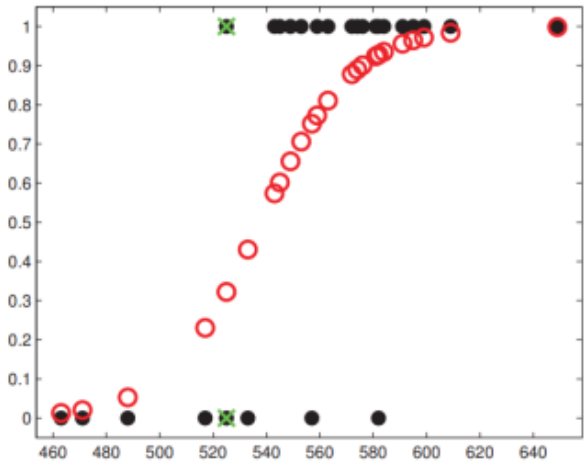
A regressão logística é um método ainda muito utilizado para realizar a análise de crédito. A partir da década de 1980, sua utilização foi cada vez mais frequente e está presente até os dias de hoje como uma das principais ferramentas em avaliar crédito no mercado (THOMAS, 2009). Devido à característica de analisar dados binomiais, o algoritmo apresenta ótimos resultados ao desempenhar tarefas como a análise de *credit scoring*, onde a variável dependente é a ocorrência de inadimplência e as variáveis dependentes como sendo seus fatores explicativos (COELHO et al., 2021).

Semelhante à regressão linear, na qual é responsável por ser um parâmetro de predição onde é representado por uma reta que atravessa dados em um diagrama de dispersão (NORVIG; INTELLIGENCE, 2002), o método de regressão logística é bastante semelhante, porém ao invés de uma reta, é gerado uma curva chamada de função logística plotada em formato de "S", onde determina a probabilidade de um dado ser classificado em alguma categoria, como demonstrado na Figura 1.

A Equação 1 representa como é formada a função logística, onde e é o número de Euler, $x0$ é o valor de x no ponto médio da curva segmóide, k representa a declividade da curva e L é o valor máximo da curva.

$$f(x) = \frac{L}{1 + e^{-k(x-x0)}} \tag{1}$$

Figura 1 – Predição com Regressão Logística



Fonte: (LOPES, 2022)

2.2.2 Árvores de decisão

As árvores de decisão são algoritmos de aprendizado de máquina supervisionado, onde podem ser utilizados para classificação e regressão. Tem seu funcionamento baseado em árvores binárias e possui uma estrutura muito comum a um fluxograma. O algoritmo possui um nó inicial chamado de raiz, onde a busca é realizada percorrendo os nós filhos até encontrar um nó folha. Cada nó, com exceção dos nós folha, possui um condicional relacionado, podendo gerar subárvores ou um nó folha, onde podem ser classificadas em uma categoria ou aproximação numérica, representando os resultados das decisões (MURPHY, 2012).

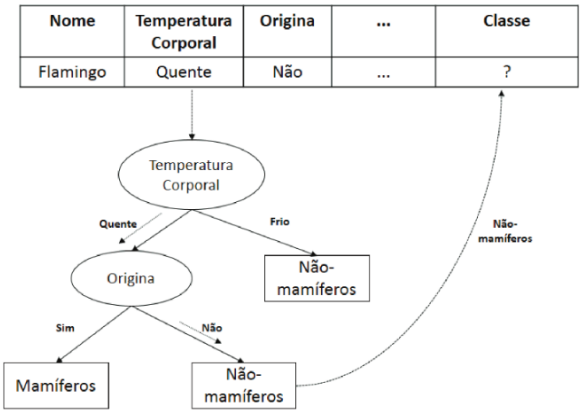
Um dos problemas está em montar a árvore em uma ordem correta. Para isto, são utilizadas algumas técnicas como ganho de informação, entropia e impureza Gini (Gini Index - G1), onde é necessário realizar cálculos que avaliam cada nó, definindo sua posição na árvore.

A procura na árvore é baseada em regras. A condição inicia-se no nó raiz e vai percorrendo até um de seus nó folhas. Em cada registro, aplica-se a condição de teste onde será direcionado a determinada aresta dependendo do resultado, podendo ser um nó folha ou outro nó interno. Caso seja um nó interno, será realizado novamente o teste e direcionado à outra ramificação. Caso seja um nó folha, o registro receberá o rótulo da classe associada ao nó folha. A Figura 2 refere-se ao caminho percorrido na árvore de decisão, que prevê o rótulo da classe de um flamingo.

2.2.3 KNN

O algoritmo KNN (do inglês *K-Nearest Neighbors*) classifica o dado conforme a distância de seus k vizinhos mais próximos, a fim de associá-lo à categoria mais frequente entre os dados adjacentes (BISHOP, 2006). Ao implementar o algoritmo, um dos hiper-parâmetros mais importantes é o k, representando o número de vizinhos que se deseja comparar ao dado.

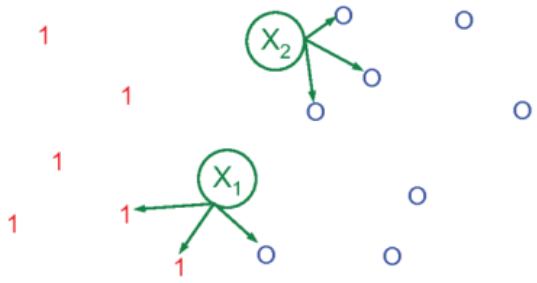
Figura 2 – Árvore de decisão



Fonte: TAN, Kumar e Steinbach (2009)

Geralmente, este número escolhido é um possível número inteiro ímpar, a fim de não haver empates nas comparações. A Figura 3 exemplifica o funcionamento do algoritmo, onde k = 3 e X1 e X2 são os dados a serem classificados. Para X1, os três mais próximos foram classificados respectivamente como 1,1,0. Sendo assim, X1 será classificado como 1. Para X2, os três vizinhos mais próximos estão classificados como 0, sendo assim sua classificação também será.

Figura 3 – Algoritmo KNN - predição com k = 3



Fonte: Lopes (2022)

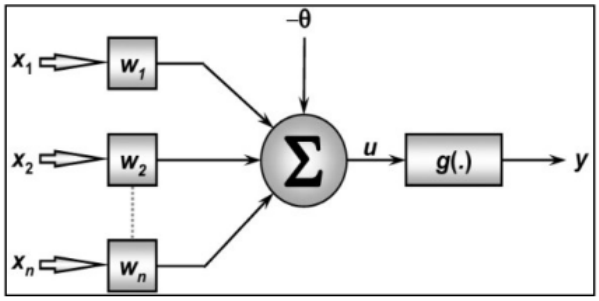
Uma das tarefas importantes é a escolha do K. Caso este seja muito pequeno, a classificação pode não ser assertiva devido o pequeno número de amostras, podendo esta conter dados *outliers*, ou seja, valores discrepantes em relação aos demais. Caso K seja muito grande, fica inviável a classificação para dados mais numerosos, não podendo estar tão próximo do dado a ser classificado.

2.2.4 MLP

Neste estudo, foi utilizado o algoritmo MLP, mas antes de explicar seu funcionamento, será descrito brevemente o funcionamento de um algoritmo de redes neurais. O algoritmo de redes neurais é baseado em modelos biológicos, na qual possui diversos nós conectados

semelhante aos neurônios e as conexões inspiradas nas sinapses, simulando sinapses artificiais estruturadas. Com isto, as RNAs (redes neurais artificiais) possuem a capacidade de aprendizagem por meio do treinamento, regeneração e tolerância a falhas (MARTINS; METTE; MACEDO, 2008). Sua arquitetura é baseada em três camadas: entrada, oculta ou intermediária e saída. Um neurônio pode ser visualizado conforme a Figura 4 abaixo, onde Xi são os sinais de entrada, Wj são os pesos sinápticos, os quais são adaptados conforme os sinais de entrada. Após esta etapa, é feito a soma ponderada dos sinais de entrada, realizada pelo combinador linear $\sum$, onde sua saída será subtraída pela variável limiar de ativação u. A saída dos neurônios artificiais passam pela função $g()$, a qual se trata de uma função de ativação, responsável por delimitar a saída do neurônio no intervalo capaz de ser assumido pela rede, gerando o sinal y (SILVA; SPOTTI; FLAUZINO, 2010).

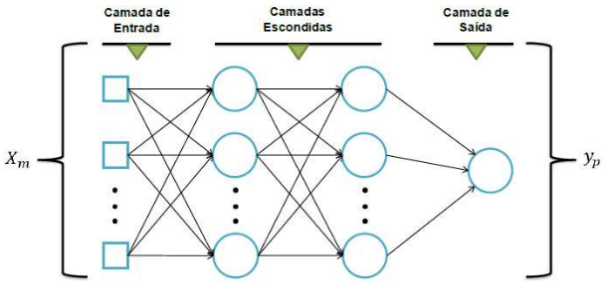
Figura 4 – Neurônio artificial



Fonte: Silva, Spotti e Flauzino (2010)

A rede neural *Multilayer perceptron* é caracterizada por possuir pelo menos uma camada escondida (intermediária) entre as camadas de entrada e saída. A Figura 5 mostra uma rede neural *multlayer* composta por duas camadas ocultas.

Figura 5 – Arquitetura de uma MLP com duas camadas escondidas



Fonte: Silva, Spotti e Flauzino (2010)

Tratando-se da arquitetura, o número de camadas ocultas bem como de neurônios são definidos de forma empírica, ou seja, por meio da realização de experimentos. Um dos algoritmos mais utilizados para treinar a rede é o *backpropagation* e seu funcionamento é realizado por meio do aprendizado por correção de erro, onde é executado em dois processos

entre as camadas: propagação e retro-propagação. A propagação refere-se a um passo a frente e a retro-propagação um passo atrás (DAMETTO, 2018).

2.2.5 Naive Bayes

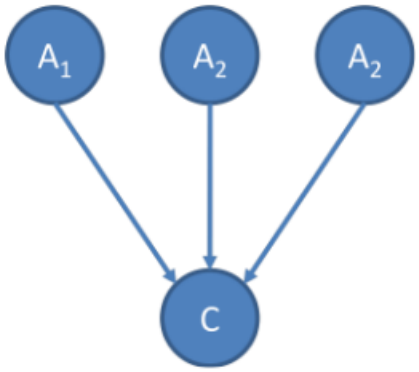
Os algoritmos basados no teorema de *bayes* realizam predições com base em modelos estatísticos, os quais utilizam informações do problema com novos dados a fim de determinar a probabilidade final (MITCHELL; MITCHELL, 1997).

A classe de algoritmos *Naive Bayes* é fundamentada no teorema de *bayes*, desenvolvido por Thomas Bayes no seculo XVIII. O algoritmo fundamenta-se no seguinte conceito: dado um conjunto de dados separados por seus atributos e características, é possível predizer a qual grupo pertence um novo dado. A Equação 2 demostra este conceito, onde se define como: dado um conjunto de evidências B, qual a probabilidade da hipótese A estar contida em B (SCHMITT, 2013).

$$P(A,B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

O termo *"naive"*, que quer dizer ingênuo, significa ser um algoritmo fundamentado no princípio que todas as características do problema são independentes uma das outras, algo que não ocorre com tanta frequência em problemas reais. A Figura 6 ilustra o modelo do algoritmo, onde todos os termos Ai são independentes entre si, porém todos influenciam a classe C. Por mais que seja baseado nesta premissa, *Naive Bayes* é um dos mais efetivos algoritmos de aprendizado de máquina, se destacando principalmente em áreas de classificação de texto (ZHANG, 2004).

Figura 6 – Modelo do algoritmo Naive Bayes



Fonte: Schmitt (2013)

O algortimo Naive Bayes é classificado como um algoritmo supervisionado, isto é, trabalha com conjunto de dados rotulados, utiliza o paradigma estatístico onde aplica fórmulas

e cálculos estatísticos a fim de realizar as predições. O algoritmo faz a leitura dos dados somente uma vez para todas as predições, o que o torna rápido e eficiente. Além disto, é caracterizado pela facilidade em manipular o conjunto de dados de forma fácil e rápida, onde torna simples a inserção de novos dados. Em algumas aplicações o algoritmo possui seu desempenho equiparado aos algoritmos de redes neurais e árvores de decisão (MITCHELL; MITCHELL, 1997).

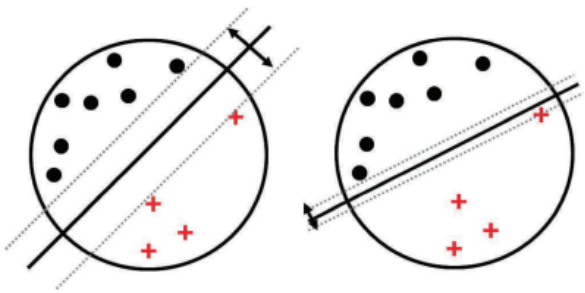
Os algoritmos *Neive Bayes* podem ser diferenciados a partir de qual distribuição será utilizada, ou seja, como P(xi|y) é calculado, podendo ser classificados em Multinomial, Gaussiana ou Bernoulli (METSIS; ANDROUTSOPOULOS; PALIOURAS, 2006).

2.2.6 SVM

O algoritmo SVM (*Suport Vector Machines*) foi desenvolvido em meados dos anos 90 por Cortes e Vapnik (1995). Visando realizar classificações binárias, ao longo do tempo o algoritmo foi cada vez mais ganhando notoriedade por ter um excelente desempenho, sendo uma das principais referências em aprendizado de máquina. Devido ser de fácil implementação e muito utilizado, foram desenvolvidas outras extensões, possibilitando sua utilização em problemas de regressão e em classificações onde possuíam mais de uma classe (SILVA, 2018). O algoritmo SVM visa separar dados em grupos distintos através de um limiar, com o objetivo de maximizar a margem entre os dados dos grupos e a reta limiar, a fim de predizer a qual grupo pertence um novo dado (BISHOP, 2006). O limiar é dado por um hiperplano, onde são utilizados algoritmos de otimização a fim de refinar a posição deste limiar e encontrar um valor adequado para divisão de novos dados.

Uma das vantagens do SVM é a facilidade em manipular dados robustos e *outliers*. Tal característica é possível devido à forma com que o algoritmo separa os dados, em que se utiliza uma margem de erro entre o limiar e os dados. Na Figura 6, é demonstrado a aplicação do algoritmo ao separar esferas pretas e X vermelhos, onde é possível visualizar que a margem do exemplo a esquerda é maior em relação ao da direita (BISHOP, 2006).

Figura 7 – Margem do algoritmo SVM

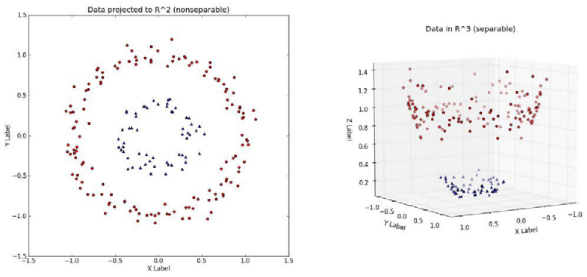


Fonte: Murphy (2012)

Entretanto, existem situações onde os dados não são lineares, não sendo possível estabelecer um hiperplano entre classes. Para solucionar este problema, os dados podem ser

distribuídos em um espaço de maior dimensão, como mostrado na Figura 7 (KIM, 2013).

Figura 8 – Transformação para um espaço de maior dimensão



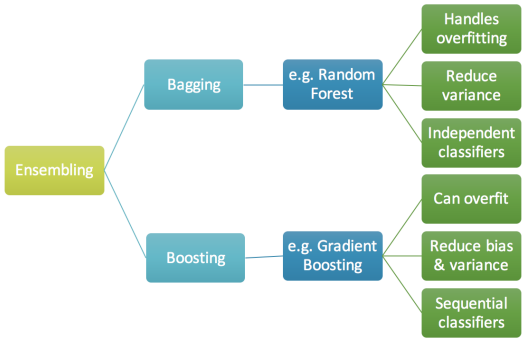
Fonte: Kim (2013)

Após a inserção dos dados em um espaço maior, torna-se possível ser projetado na dimensão anterior e aplicar um hiperplano divisor. Entretanto, ainda assim ficaria muito custoso computacionalmente, e para isto, é imprescindível a utilização do *Kernel Trick*. As funções do *Kernel* são utilizadas para calcular o produto escalar de dois vetores, onde é aplicável no contexto de calcular hiperplanos de forma muito barata computacionalmente. As funções mais utilizadas são: Linear, Polinomial, Radial Basis Function e Segmoidal (SILVA, 2018).

2.2.7 Ensembles

Métodos ensembles podem ser classificados como um agrupamento de classificadores individuais que reportam suas decisões. Seu resultado é obtido com base em todas as decisões (por meio de média ponderada ou não) para assim ter um novo resultado e classificar novos registros. O objetivo dos métodos ensembles é utilizar esta combinação para otimizar o desempenho obtido de qualquer classificador individual (DIETTERICH, 2000). Os métodos *emsembles* podem ser classificados em duas abordagens:*Bagging e Boosting*. As principais diferenças entre ambos são apresentadas na Figura 6.

Figura 9 – Diferença entre métodos Ensembles



Fonte: Grover (2017)

A diferença entre os algoritmos está na forma com que é gerado o resultado. Nos modelos *Bagging*, os algoritmos são treinados de forma individual e posteriormente agregados por meio de um método de agregação, como a média, por exemplo. Já os algoritmos *Boosting*, que também são treinados individualmente, são posteriormente agregados por meio de uma ponderação do resultado de cada modelo. Os algoritmos mais conhecidos dentre os dois tipos são *Random Forest* para os algoritmos *Bagging* e *Gradient Boosting* para *Boosting* (COELHO et al., 2021).

2.2.8 Random Forest

O algoritmo *Random Forest* é um *ensemble* baseado em árvores de decisão. Seu funcionamento é estruturado em 3 etapas principais (MURPHY, 2012). A primeira etapa é responsável por manipular a amostragem de dados, onde é realizado uma seleção de dados de forma aleatória. Mesmo que a amostragem contenha menos dados que o total presente no *dataset*, esta manipulação repete estes mesmos dados até que esteja em tamanho original a base (EFRON; TIBSHIRANI, 1994). A segunda etapa é responsável pela criação das árvores de decisão, onde o atributo responsável pela separação de dados presente na condicional de cada nó da árvore é selecionado sorteadamente, com base em um conjunto de características. A estratégia em realizar um sorteio para o atributo visa aumentar a diversificação das árvores presentes no algoritmo ensemble. Por último, a terceira parte é responsável por gerar o resultado, onde será calculado por meio de voto da maioria ou por meio da média dos resultados das árvores de decisão (LOPES, 2022). A vantagem do algorítimo *Random Forest* está em reduzir os erros de correlação gerados nas árvores de decisão do método *Bagging*. Tal vantagem é possível devido à seleção aleatória das variáveis de entrada, consequentemente sem aumentar a variância (HASTIE; TIBSHIRANI; FRIEDMAN, 2001).

2.2.9 Extreme Gradient Boosting

O algoritmo *Gradient Boosting* é uma generalização do algoritmo *AdaBoosting* proposto por Freund, Schapire et al. (1996). O modelo ensemble utiliza como base as árvores de decisão, mas outros tipos de algoritmos também podem ser aplicados ao modelo.

A principal função do *Gradient Boosting*, assim como todos os algoritmos baseados em *Boosting*, é minimizar a função perda em cada árvore de decisão. Seu objetivo é reduzir a função objetivo ao máximo a fim de aumentar a predição do algoritmo. Esta ação é realizada por meio da utilização de múltiplas árvores de decisão aleatórias estimadas, e o processo é repetido até se atingir o mínimo da função objetivo.

O algoritmo *Extreme gradient Boosting* foi criado visando na otimização da função objetivo, de maneira a auxiliar na regularização a fim de controlar o ajuste (*overfitting*) para aumentar a precisão dos dados, resultando em um modelo mais robusto. Um dos pontos negativos é que com isto, seu desempenho é reduzido ao realizar a predição de novos dados.

Seu funcionamento ocorre da seguinte forma: inicialmente, é criado um modelo $F0$ a fim de predizer valores da variável alvo Y. Em seguida, cria-se um novo modelo $h1$ para complementar o valor residual do primeiro modelo $(Y - F0(x))$, sendo Y os valores reais e $F0(x)$ os valores preditos. Então, os dois modelos são unificados formando uma nova versão do algoritmo dado por $F1(x)$, apresentado pela Equação 3.

$$F1(x) = F0(x) + h1(x) \tag{3}$$

Em comparação aos modelos, o erro quadrático médio de $F1$ é menor que $F2$, oque indica que o modelo possui maior assertividade. Para otimizar o modelo $F1$, o processo se repete a fim de gerar um novo modelo $F2$, e assim sucessivamente até que o valor residual seja reduzido ao máximo (COELHO et al., 2021).

2.2.10 Hiper-parâmetros

Os hiper-parâmetros de algoritmos de aprendizagem de máquina são parâmetros que precisam ser definidos antes mesmo da execução do modelo, sendo importantíssimos, pois definem o processo de aprendizagem do mesmo. Quando não declarados, os hiper-parâmetros possuem valores padrão, onde na maioria das vezes acaba não sendo os melhores para a execução do algoritmo. Como exemplo, o algoritmo *Random Forest* possui como um de seus hiper-parâmetros a quantidade de árvores de decisão que serão aplicadas ao modelo em sua execução, onde a alteração deste valor influencia diretamente no desempenho do algoritmo.

2.2.11 Técnicas de reamostragem

Ao aplicar um modelo de aprendizagem de máquina em um determinado problema, um dos pontos mais importantes está na fase de treinamento e teste do algoritmo, o qual define o quão bom seu modelo está em realizar as predições dos dados a partir das métricas de avaliação. Para tal, existem diversas técnicas utilizadas e uma das mais predominantes é a *holdout*. Esta técnica é fundamentada no conceito de dividir os dados de treinamento e teste em certas proporções. Como exemplo, a base de dados é divida em 70% para treino e 30% para teste, a fim de que o modelo tenha um bom desempenho e realize predições de forma ótima (AALST et al., 2010).

Entretanto, um dos riscos em utilizar a técnica *holdout* esta na forma com que o modelo é treinado. Tais problemas podem ser:

Overfitting: O modelo de dados fica muito sensível à base de treinamento, onde em fase de teste pode apresentar bons resultados, porém poderá ficar "viciado" na base de dados treinada que, ao receber novos dados para serem preditos, o modelo pode não apresentar resultados tão eficientes devido não estar familiarizado. Este tipo de problema é muito comum ao utilizar grandes bases de dados.

Underfitting: neste caso, o modelo não possui muitos dados para realizar seu treinamento e teste, resultando em realizar predições ruins por se tornar um modelo fraco, ou sem muitas experiências.

O desafio em criar um ótimo modelo de aprendizado de máquina está no equilíbrio entre estas duas vertentes, ou seja, os dados entre treino e teste precisam estar balanceados de forma que, ao realizar predições, o modelo esteja muito bem treinado a ponto obter resultados eficientes, independente dos dados de entrada (MÜLLER; GUIDO, 2016).

Uma das técnicas presente na literatura é a Validação Cruzada (do inglês *Cross Validation*). Tal técnica fundamenta-se no conceito de dividir toda base de dados em partições(chamado de *k-fold*), sendo k o número de partições. Para cada iteração o modelo utiliza k-1 para treino e a partição separada para teste. Este processo é iterativo, sendo que a cada iteração os conjuntos de treino e teste são alternados. Dessa forma, o desempenho do algoritmo é calculado com base na média de cada iteração realizada (SILVA, 2018).

Desta forma, o modelo não ficará "viciado" somente em um subconjunto de dados e também não será tão simples a ponto de não conhecer toda a base de dados.

Uma variação da validação cruzada é a técnica validação cruzada aninhada. Esta técnica é comumente utilizada ao realizar o ajuste de hiper-parâmetros de algoritmos. Consiste em realizar a separação de dados entre treinamento e teste em dois laços, sendo o primeiro para realizar a escolha dos melhores hiper-parâmetros e o segundo para verificar o desempenho destes. Esta técnica pode ser utilizada tanto por meio da separação por *holdout*, validação cruzada ou a junção de ambas. Para exemplificar, ao realizar o ajuste de hiper-parâmetros de um algoritmo, a partição de dados é dividia, por exemplo, em *10-folds*. Selecionando os melhores hiper-parâmetros, é necessário testar seu desempenho, e para isto o modelo de dados é particionado novamente com *2-folds*, por exemplo. Esta técnica é muito eficiente, pois valida o desempenho do modelo de forma confiável e assertiva.

2.2.12 Métricas de avaliação

Para a realização deste trabalho, foi utilizada a métrica de avaliação curva ROC (do inglês, *Receiver Operating Characteristic Curve*). A curva ROC é uma métrica que avalia e ilustra o desempenho de um classificador binário à medida que o seu limiar de discriminação é alterado. Possui dois parâmetros: a taxa verdadeiros positivos, representado pela métrica de sensibilidade e a taxa falsos positivos, representado pela métrica de especificidade. A curva ROC é plotada em um plano unitário, onde o eixo x representa a especificidade e o eixo y a sensibilidade.

A fim de encontrar um valor único que apresente claramente o desempenho alcançado pelo algoritmo, a AUC (do inglês, *area under curve*) resume a curva ROC em um único valor, agregando todos os limiares da ROC, calculando a "area sob a curva". O valor da AUC pode variar em um intervalo entre 0 e 1 e o limiar entre a classe é 0,5. Isso quer dizer que, acima de 0,5 o dado pertence a uma classe e abaixo, outra classe. Em relação à assertividade, um

modelo cujas previsões estão 100% erradas possui uma AUC de 0, enquanto um modelo com nível de 100% de assertividade possui uma AUC de 1 (SILVA; FRANÇA,).

2.3 Otimização de hiper-parâmetros

A otimização de hiper-parâmetros é uma técnica utilizada para otimizar o desempenho dos algoritmos e modelos de aprendizado de máquina. Por meio de um intervalo definido, realiza de forma automática a aplicação de todas as possíveis combinações de hiper-parâmetros, a fim de encontrar os que obtiveram melhor desempenho. Um algoritmo com hiper-parâmetros ótimos é possível aumentar consideravelmente seu desempenho, evitando problemas como: sobre-ajuste, baixa capacidade de desempenho e outros. Neste trabalho, foram utilizados as técnicas de otimização *BayeSearch, RandomSearch* e algoritmo genético.

2.3.1 Busca Bayesiana

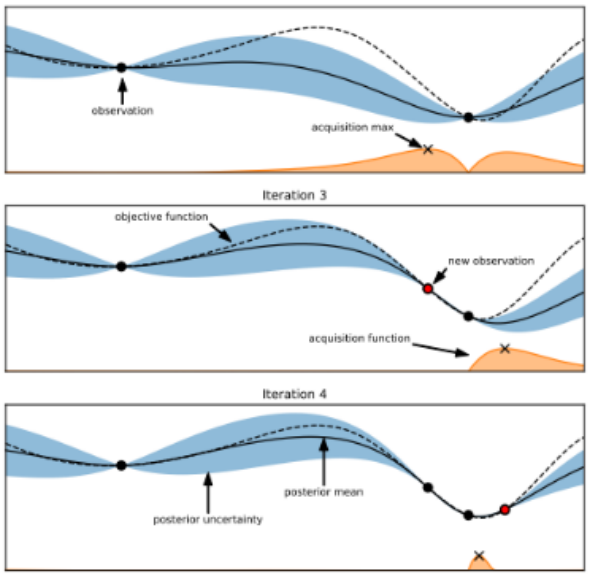
Segundo Snoek, Larochelle e Adams (2012), a busca bayesiana é um dos métodos mais bem sucedidos quando se trata da otimização de hiper-parâmetros. É uma técnica livre de derivação e uma das que possui melhores desempenhos em problemas complexos de otimização em caixas-pretas. A busca Bayesiana é fundamentada na teoria de probabilidade condicional de Bayes, onde possui como componentes principais as curvas de *surrogates* e a maximização da esperança (SNOEK; LAROCHELLE; ADAMS, 2012). As curvas de *Surrogates* são um componente importantíssimo que mede a probabilidade do quão promissor um ponto X está para uma função f(x). Para conseguir uma evolução, as curvas utilizam o histórico de pontos anteriores para atualizar, a fim de que ao longo das iterações, a curva seja o mais similar possível a f(x). A Figura 9 ilustra a iteração da curva, semelhante ao cálculo de uma probabilidade condicional, onde a cada iteração é selecionado novos valores (pontos) com base na maximização da esperança, onde se percebe que o objetivo está em aproximar cada vez mais a curva da função objetivo, o que torna-se nítido ao longo das iterações.

Para determinar o formato da curva, um dos métodos mais utilizados são os processos gaussianos. Estes, verificam pontos utilizados anteriormente a fim de criar novas estimativas em relação a f(x), como demonstrado na figura acima. Os processos Gaussianos possuem grandes vantagens, pois permitem o uso de diversos tipos de parâmetros, podendo ser numéricos ou categóricos. Entretanto, possui dificuldades em analisar dados de alta dimensionalidade, pois não consegue separar claramente os vetores de alta dimensionalidade, sendo uma das motivações em encontrar melhores métodos.

O funcionamento do algoritmo está em separar os pontos mais promissores, a fim de encontrar os melhores em relação a f(x), economizando tempo e processamento. Dessa forma, é possível repetir o processo iterativo com pontos mais promissores a fim de que a curva de *Surrogates* se aproxime mais do comportamento de f(x) (PELLICER, 2020).

A Figura 10 ilustra o funcionamento do processo gaussiano, onde por meio de funções

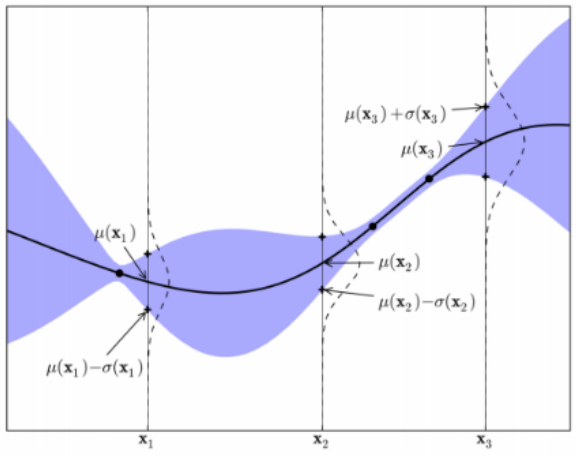
Figura 10 – Iterações - Curva de surrogate



Fonte: Hutter, Kotthoff e Vanschoren (2019)

estatísticas, utiliza uma "aprendizagem preguiçosa"com a medida de similaridade entre os pontos, a fim de realizar novas predições, determinando assim o formato da curva.

Figura 11 – Processo gaussiano



Fonte: Kim e Chung (2019)

2.3.2 Busca Aleatória

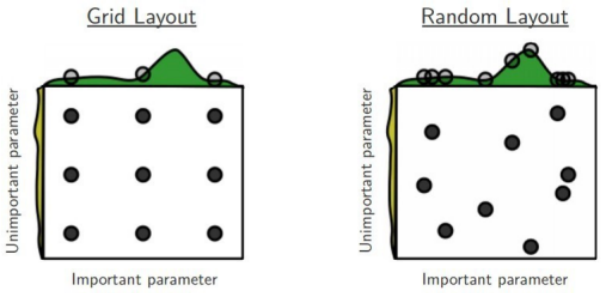
Uma das primeiras técnicas de otimização de hiper-parâmetros é conhecida como *Grid Search*. Atualmente, ainda é utilizada em grande escala devido ser de fácil entendimento e implementação. Basicamente, a técnica explora todas as possíveis combinações de valores de X, sendo conhecido como uma técnica exaustiva.

Entretanto, quando se trata de dados em maiores dimensões, a técnica não apresenta resultados eficientes. Devido à estratégia em testar todos os valores, a cada novo valor testado ou nova dimensão incluída em X, resulta em um aumento exponencial no número de combinações possíveis, tornando-se uma técnica de alto custo computacional (PELLICER, 2020).

Para resolver este problema, foi implementada a busca aleatória (*Random Search*). Ao invés de testar todos os valores possíveis em X, os valores são sorteados e testados de forma aleatória, resolvendo assim os problemas de alta dimensionalidade, sendo menos custoso que o algoritmo *Grid Search*.

A busca aleatória pode apresentar melhor desempenho comparado ao *Grid Search*. Esta melhora é justificada devido ao processo realizado, onde as funções de métricas de desempenho dos modelos de aprendizado de máquina não possuem regiões bem comportadas, pequenas de pontos promissores. O *GridSearch* muitas vezes não explora estas regiões ao espalhar os pontos testados, ao contrário da busca aleatória, onde apresenta uma probabilidade em explorar tais regiões, podendo apresentar melhores resultados. A Figura 12 demonstra a forma com que estes algoritmos exploram as regiões, onde a busca em grade (*Grid Layout*) realiza a busca por meio de valores fixos, obedecendo à estrutura em grade como um modelo engessado. Já a busca aleatória (*Random Layout*), realiza a busca de pontos em um formato não específico, não possuindo uma estrutura fixa, sendo mais variável na seleção de pontos, atingindo qualquer área no campo de seleção (BERGSTRA; BENGIO, 2012).

Figura 12 – Comparativo entre a busca exaustiva e busca aleatória



Fonte: Bergstra e Bengio (2012)

2.3.3 Algoritmos Genéticos

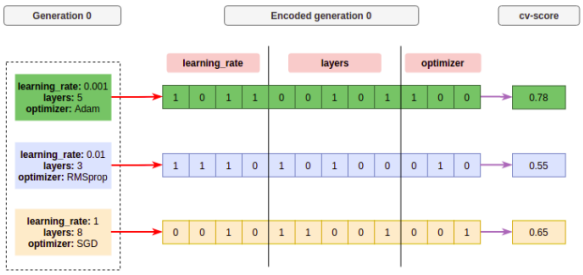
Os algoritmos genéticos são baseados na teoria da evolução, onde por meio de diversas iterações, tem-se o objetivo de gerar indivíduos cada vez mais fortes. No conceito da otimização de hiper-parâmetros de algoritmos de aprendizado de máquina, os algoritmos genéticos têm como finalidade testar diversos valores estabelecidos e aplicar operadores genéticos a fim de encontrar o melhor conjunto de hiper-parâmetros para o algoritmo.

As etapas padrões aplicadas pelo algoritmo genético são: i) Gerar uma população inicial de forma aleatória (diferentes conjuntos de hiper-parâmetros); ii) Avaliar por meio da

validação cruzada qual o desempenho de cada conjunto; iii) Criar uma geração a partir da utilização de operadores genéticos; iv) Repetir as etapas ii e iii até se ter um critério de parada (ARENAS, 2021).

Detalhadamente, a primeira etapa consiste na elaboração da primeira geração da população a qual o algoritmo irá trabalhar. Para isso, é selecionado um conjunto de hiper-parâmetros aleatórios sendo definido como a primeira geração.

Figura 13 – Definindo a população inicial



Fonte: Arenas (2021)

A segunda etapa consiste em avaliar o desempenho deste conjunto de hiper-parâmetros selecionados de forma aleatória, a fim de verificar o desempenho de cada um. A partir daí, serão aplicados os operadores genéticos para a criação de novas gerações.

A etapa iii é responsável por criar uma geração, onde é utilizado alguns operadores genéticos. A seguir, serão descritos os mais comuns:

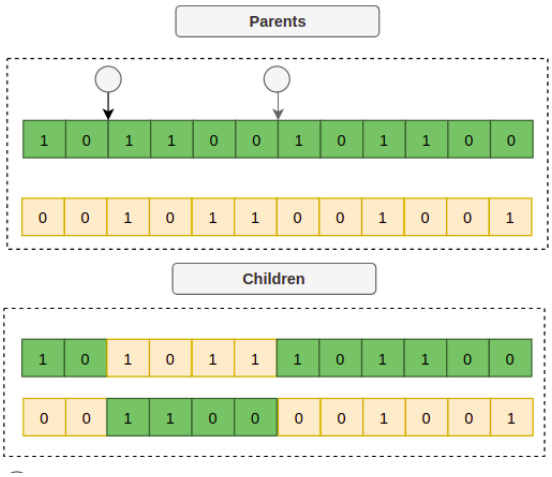
Cruzamento: O cruzamento é o operador responsável por criar indivíduos. Primeiro, é realizado a seleção dos pais, podendo ser por meio de uma função de probabilidade. Após, é realizado o cruzamento e gerado o novo gene, representando assim um novo conjunto de hiper-parâmetros, como mostra a figura abaixo.

Mutação: Após diversas iterações realizadas, a geração de novos genes pode ser muito semelhante. Para resolver este tipo de problema, é utilizado o operador Mutação. Este operador altera de forma suficientemente baixa um hiper-parâmetro a fim de criar conjuntos diversos, como demonstrado na figura abaixo.

Elitismo: este operador visa manter os genes/hiper-parâmetros que obtiveram os melhores resultados ao longo das iterações, a fim de preservar as melhores características em novas gerações. A Figura 16 ilustra o operador Elitismo.

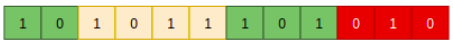
Após percorrer todas as etapas, é necessário estabelecer um critério de parada, podendo ser um número máximo de gerações, um tempo limite para a execução do algoritmo e por fim quando não existem mais melhorias a serem realizadas (ARENAS, 2021).

Figura 14 – Operador Cruzamento



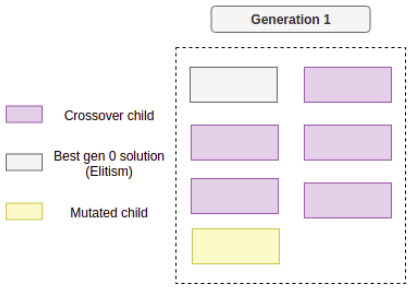
Fonte: Arenas (2021)

Figura 15 – Operador de mutação



Fonte: Arenas (2021)

Figura 16 – Operador Elitismo



Fonte: Arenas (2021)

3 Trabalhos Relacionados

O presente capítulo tem como objetivo a análise de trabalhos relacionados ao tema da pesquisa. Para a seleção dos trabalhos relacionados, foi pesquisado nas bases ACM Digital Library, IEEE Xplore e Google Scholar, priorizando publicações mais recentes. Os estudos foram pesquisados através de buscas realizadas utilizando as palavras-chave: algoritmos de *machine learning*, análise de risco de crédito, utilização de algoritmos por instituições financeiras e otimização de hiper-parâmetros em algoritmos de *machine learning*, com o objetivo de encontrar trabalhos que fossem relacionados ao tema de pesquisa. Concluído a pesquisa, foi realizado a seleção por meio de uma breve análise dos títulos e resumos, e, posteriormente, a partir da leitura de suas introduções e conclusões, para entender o seu contexto e então lido na íntegra para compor esta monografia. Assim, neste capítulo são apresentados os seguintes estudos. A Tabela 1 apresenta o título e o ano de publicação de cada trabalho;

Tabela 1 – Trabalhos relacionados

Título	**Ano de publicação**
Métodos de otimização hiperparamétrica: um estudo comparativo utilizando árvores de decisão e florestas aleatórias na classificação binária.	2018
Estudo comparativo entre metodologias de aprendizado de máquina e híbridas aplicadas a risco de crédito.	2019
Modelo preditivo para avaliação de crédito em empréstimos pessoais.	2019
Analisando métodos de machine learning e avaliação do risco de crédito.	2021

Fonte: elaborada pelo autor.

No estudo de Júnior (2018) foi realizado um comparativo de técnicas de otimização de hiper-parâmetros, sendo: Busca em Grade, Busca Aleatória e Otimização Bayesiana (com o uso do Processo Gaussiano) aplicado aos modelos Árvore de Decisão e Floresta Aleatória. Para este estudo, foram utilizados diferentes conjuntos de dados em problemas de classificação binária, sendo aplicado a técnica de validação cruzada para treino e teste dos modelos. Para avaliação dos resultados, foram utilizados as métricas Acurácia, entropia cruzada, sensibilidade, especificidade e AUC. Ao fim dos testes, a técnica de otimização Bayesiana apresentou melhores resultados que os outros métodos testados.

Em CASTRO et al. (2019) foi apresentado um estudo aplicado a risco de crédito onde compara o desempenho da técnica de regressão logística, utilizadas anteriormente para análise, com modelos de *machine learning*. Para realizar o estudo, foi utilizado dados de clientes, pessoa física e jurídica, provenientes da Serasa Experian, sendo um diferencial por se tratar de uma

fonte diversificada de informações, onde não somente porta dados de uma instituição financeira específica, mas de diversas empresas. Foram utilizadas amostras mensais referente ao período de janeiro a dezembro de 2017. A seleção de dados foi realizada de maneira que contivesse somente dados de empresas ativas na receita federal e pessoas físicas maiores que 18 anos, considerados clientes inadimplentes aqueles que possuem suas dívidas vencidas e não pagas um ano após a data de referência. Para treinamento e teste, a base foi dividida de forma que os dados relacionados aos meses de janeiro a setembro fossem separados para treino e os demais para teste. Para comparação com o método de regressão logística, foram utilizados os métodos ensemble:*Random Forest, Gradient Boosting* e modelos híbridos. Como métrica de avaliação, foi utilizado AUC, KS (*Kolmogorov-Smirnov*) e acurácia, sendo que o modelo que mais obteve melhores resultados que os demais foi o *Gradient Boosting* para classificar dados de pessoas físicas e o modelo híbrido árvore de decisão com *Gradient Boosting* para pessoas jurídicas.

Já no estudo de Nascimento et al. (2019), foi desenvolvido uma arquitetura para análise de créditos pessoais aplicando em dados de uma instituição financeira. O processo de aprovação de crédito é dividido em duas etapas, sendo que anteriormente a aplicação deste trabalho, a primeira etapa analisava 52% das propostas utilizando um sistema com valores previamente definidos e a segunda etapa os outros 48% das propostas, os quais era realizado pelos colaboradores da empresa. Como proposta do trabalho, os autores estruturaram dois modelos de classificação automática utilizando o algoritmo *Random Forest*, substituindo o sistema utilizado anteriormente. A partir disto, foi definido valores de limiares a fim de classificar propostas em que era necessária a análise manual, não deixando somente a encargos do algoritmo realizar todas as análises. A classificação funciona da seguinte forma: Primeiro a proposta de crédito é submetida ao modelo 1, o qual consiste em realizar uma análise bruta, sem a utilização de consultas a dados externos. Caso seja rejeitada, passará para o modelo 2, o qual utiliza a mesma premissa de classificação do modelo 1, porém realiza algumas consultas externas. Caso não se enquadre aos limiares, a proposta será encaminhada a análise manual.

No desenvolvimento os autores realizaram um pré-processamento dos dados, onde foram removidos dados discrepantes (*outliers*) e propostas realizadas por menores de 18 anos. Logo em seguida, foram analisadas 192 propostas de pedidos de crédito limitadas a um intervalo de 14 meses. Prosseguindo, os dados foram divididos em 50% para treino, 50% para teste e posteriormente padronizados. Dos dados separados para o treinamento, 75% foram reservados para treinamento e 25% para serem validados. A métrica de avaliação selecionada foi a acurácia. Na primeira etapa, foi obtido um aumento de 52% para 86,56% de análises classificadas de forma automática com a aplicação da nova arquitetura. Na segunda etapa, mais de 4,04% das propostas foram classificadas automaticamente, resultando um total de 97% de acurácia em classificações automáticas.

Por fim, o trabalho de Coelho et al. (2021) realizou uma análise algoritmos de *machine learning* aplicados a avaliação do risco de crédito em uma base de dados cedida por uma empresa do setor de locações de automóveis. Os dados selecionados foram do período de

Tabela 2 – Resultado dos trabalhos relacionados.

Referência	**Objetivo**	**Algoritmo**	**Desempenho**
Júnior (2018)	Analisar o desempenho de técnicas de otimizacão de hiper-parâmetros.	Otimização Bayesiana	AUC: 93.7%
CASTRO et al. (2019)	Aplicar método de regressão logística e algoritmos de AM no contexto de análise de crédito.	PF: Gradient Boosting PJ: Árvore de decisão+GB	PF(AUC): 82.4% PJ(AUC): 80.8%
Nascimento et al. (2019)	Utilizar algoritmos de AM para realizar análise de propostas de crédito.	Random Forest	Propostas classif. Modelo1: 86.56% Modelo2: 4.04% Ac. modelos: 97%
Coelho et al. (2021)	Utilizar métodos de AM para avaliação do risco de Crédito.	Regressão Logística Random Forest XGBoost	AUC RL:63.02% RF: 66.81% XG: 72.04%

Fonte: elaborada pelo autor.

janeiro de 2018 a abril de 2019. A amostra inicial possui 17.800 clientes de todo o território nacional. Posteriormente, esta amostra foi balanceada aleatoriamente em relação ao número de clientes classificados como Bons e Ruins, resultando em uma amostra final contendo 3.844 clientes, onde 1.492 eram classificados como Bons e 1492 eram classificados como Ruins. Para comparações, foi utilizada a regressão logística clássica e dois métodos de *machine learnig* para *credit scoring*, sendo *Random Forest* e *XGBoost*. Como métricas de avaliação, foram utilizados acurácia, estatística *Kolmogorov-Smirnov* e a curva ROC. Ao fim do experimento, o algoritmo de *machine learning* apresenta maior capacidade preditiva comparado com a regressão logística. O método *XGBoost* obteve melhor desempenho comparado aos demais.

Como resumo apresentado de forma simplificada, a Tabela 2 mostra o resultado obtido em cada trabalho relacionado, apresentando o melhor método e algoritmo, juntamente com o desempenho alcançado de acordo com a métrica de avaliação proposta.

4 Metodologia Experimental

Este capítulo visa apresentar a parte prática do projeto e descrever todas as etapas de desenvolvimento. Na seção de materiais será abordado todas as bibliotecas e *softwares* utilizados e na seção seguinte, serão abordados todas as etapas de desenvolvimento do trabalho.

4.1 Ferramentas

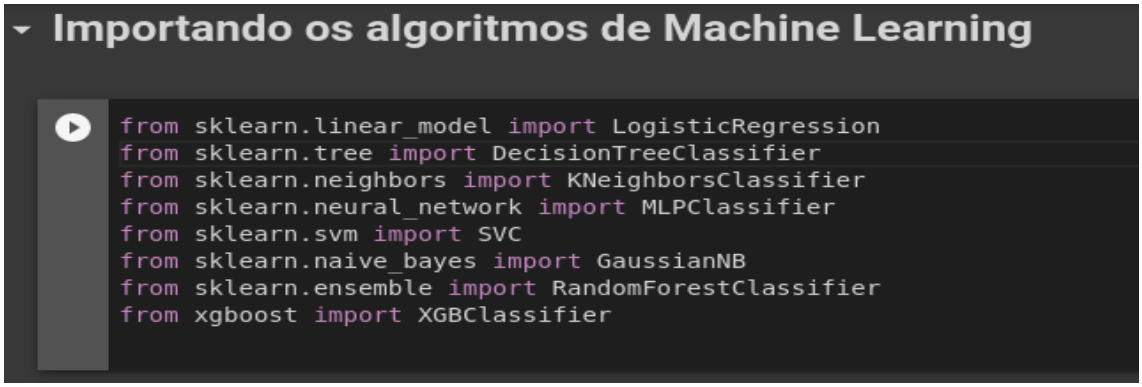
4.1.1 Scikit-learn

A biblioteca *Scikit-learn* foi desenvolvida de forma *open source* para linguagem de programação *Python* de forma que toda a comunidade possa cooperar com seu desenvolvimento, recebendo atualizações constantemente (BUITINCK et al., 2013).

A principal utilização da biblioteca é a facilidade em importar diversos recursos presentes na linguagem *python*, desde a importação de algoritmos supervisionados e não-supervisionados como a aplicação de ferramentas transformadoras de dados, como, por exemplo, *Principal Components analysis*. Também conta com a importação de recursos como algoritmos otimizadores de hiper-parâmetros e *Pipeline* (PEDREGOSA et al., 2011).

Todas as ferramentas presentes na biblioteca podem ser utilizadas por meio da invocação da classe, assim como a instanciação de seus métodos. A utilização de algoritmos, por exemplo, é feita de forma fácil e prática, onde todos possuem métodos para treinamento e teste, assim como a modificação de seus hiper-parâmetros. A biblioteca também comporta técnicas como *cross validation e holdout* para separação de dados. A Figura 17 ilustra a instanciação dos algoritmos utilizados neste trabalho por meio desta biblioteca.

Figura 17 – importação de algoritmos utilizando Sklearn



Fonte: elaborada pelo autor

4.1.2 Numpy

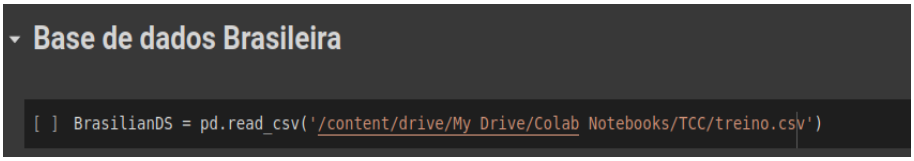
A biblioteca *Numpy* foi desenvolvida visando dar suporte principalmente a toda parte numérica, em especial na linguagem científica. Disponível para ser utilizada na linguagem *python*, a biblioteca é integrante do conjunto *Scipy*, e seu desenvolvimento fundamentado na biblioteca *Numeric*. Desenvolvida em 2005, desde então vem sendo atualizada constantemente devido sua grande importância e utilização (OLIPHANT, 2006). Uma de suas grandes vantagens está na elaboração de vetores de N dimensões (denominados *Numpy Array*), a facilidade no manuseio de vetores, suporte a funções matemáticas como transformada de *Fourier*, álgebra linear dentre outras. Além disto, possui integração com as linguagens C++ e *Fortran*, bem como também é utilizada como base para formação de diversas outras bibliotecas como, por exemplo, *scikit* (SILVA, 2018).

4.1.3 Pandas

Pandas é uma biblioteca essencial para manipulação de dados estruturados em *Python*. Contendo métodos estatísticos, a biblioteca é bastante robusta, podendo ser utilizada em diversas aplicações com grande quantidade de dados, como, por exemplo, em análise de dados no mercado financeiro. A biblioteca possui suporte para criação de vetores uni-dimensionais (Séries) e vetores bi-dimensionais (*DataFrames*). Por meio da utilização do *DataFrame*, é possível criar *subsets*, importar dados de diferentes formatos, tais como arquivos CSV (*Comma Separated Values*), tabelas do Excel, banco de dados e arquivos estruturados em páginas HTML. Além disso, a biblioteca oferece recursos para tratar dados nulos, assim como é possível realizar diferentes filtros semelhante à consulta realizadas em sistemas gerenciamento de banco de dados(SGBD) como, por exemplo, *filter, unior, merge, join, groupby* dentre outros.

A Figura 18 demonstra a importação de uma das bases utilizada neste trabalho por meio da biblioteca pandas.

Figura 18 – Importação da base de dados com pandas



Fonte: elaborada pelo autor

4.1.4 Matplotlib

A *Python Matplotlib* é uma biblioteca feita exclusivamente para a criação de gráficos 2D na linguagem *Python*. Foi criada por John Hunter em 2003 visando ser uma biblioteca *open source* semelhante ao MatLab, *software* muito utilizado por cientistas e engenheiros para elaboração de gráficos. Visando ser uma ferramenta versátil e de fácil manipulação, a biblioteca

permite elaborar e customizar gráficos de diferentes tipos, podendo ser estáticos, animados e iterativos (HUNTER, 2007). Possui diversos tipos de gráficos tais como: gráficos de barras, linhas, dispersão, dentre outros, os quais podem ser facilmente alterados e customizados. Neste trabalho, todos os gráficos foram gerados com suporte desta biblioteca.

4.1.5 Google Colaboratory

O *Google Colaboratory* é uma ferramenta disponibilizada pela Google de forma *online* a fim de incentivar o desenvolvimento de aplicações na área de inteligência Artificial (LALL, 2018). Disponível gratuitamente e também com planos pagos, a ferramenta é disponibilizada na *web*, onde é utilizada com o propósito de rodar aplicações que exigem um *hardware* com maior desempenho, incentivando o desenvolvimento de projetos onde o programador talvez não possua uma estrutura que comporte a execução da aplicação. A ferramenta conta com uma interface bastante simples e intuitiva, onde os códigos podem ser separados por meio de blocos, permitindo assim a execução individual. Também é possível realizar comentários e criar blocos de texto, deixando a aplicação mais legível e organizada.

4.2 Base de dados

Para realizar este trabalho, foram utilizadas 4 bases de dados, sendo 3 destas: *Australian Credit Approval, German Credit Data e Default of Credit Card Clients Data Set* disponibilizada pelo repositório da *UCI Machine Learning* (ASUNCION; NEWMAN, 2007). Já a quarta base de dados foi extraída a partir do repositório da plataforma *Kaggle* (LUPUM, 2020). Todas as bases de dados possuem características semelhantes voltadas a análise de crédito, contendo dados com informações pessoais e a classificação do cliente como inadimplente ou não. A base de dados *Australian Credit Aproval* refere-se a dados de clientes do país da Austrália, contendo 690 casos distribuídos em 15 atributos, sendo estes codificados a fim de preservar a identidade dos dados. A Tabela 3 apresenta o dicionário de dados desta base. O conjunto de dados *German Credit Data* refere-se a versão *German Credit (UPDATE) Dataset*, contendo dados de clientes do país da Alemanha, sendo 1000 casos distribuídos em 21 atributos, os quais podem ser consultados na Tabela 4.

A base de dados *Default of Credit Card Clients Dataset* representa dados de clientes taiwaneses e possui 30.000 casos distribuídos em 24 atributos, os quais podem ser consultados na Tabela 5. Para realização deste trabalho, foram selecionados 10.000 casos deste *dataset*, a fim de possuir uma quantidade semelhante aos demais.

O quarto *dataset* refere-se a dados de clientes brasileiros, onde foram selecionados 10.000 casos distribuídos em 11 atributos, os quais podem ser consultados na na Tabela 6.

As três bases de dados disponibilizadas pela UCI são muito famosas em aplicações de análise de crédito utilizando aprendizado de máquina. Já o quarto conjunto de dados foi selecionado de forma que fosse semelhante as demais bases, a fim ser mais um objeto de estudo

Tabela 3 – Dicionário de dados da base australiana.

Atributo	**Tipo**
A1	Categórico
A2	Numérico
A3	Numérico
A4	Categórico
A5	Categórico
A6	Categórico
A7	Numérico
A8	Categórico
A9	Categórico
A10	Numérico
A11	Categórico
A12	Categórico
A13	Numérico
A14	Numérico
A15	Classe

Fonte: elaborada pelo autor.

deste trabalho.

4.3 Análise exploratória dos dados

Conforme dito anteriormente, o objetivo deste trabalho é realizar um estudo aplicando algoritmos de *machine learning* em quatro bases de dados voltadas à análise de risco de crédito, realizando a otimização de seus hiper-parâmetros e verificar seu desempenho, apresentando assim os melhores algoritmos e técnicas de otimização que obtiveram destaque. Com as bases de dados selecionadas, foi realizado a análise exploratória a fim de preparar os dados para serem testados pelos algoritmos. Primeiramente, foi realizado a procura de valores nulos nas bases de dados, onde foi utilizado a técnica de imputação de dados por média através do método *SimpleImputer* da biblioteca *SKlearn*. A escolha desta técnica foi motivada devido possuir muitos valores nulos, principalmente na base de dados brasileira. A fim de preservar os dados, esta técnica realiza a imputação com base na média calculada de todos os valores presentes no *dataset* deste atributo. Após esta etapa, foi analisado os atributos de cada base de dados, onde foi verificado grande disparidade em relação à grandeza dos valores. A técnica de padronização dos dados visa em manter na mesma escala, mas de ordem proporcional (MURPHY, 2012). Com isto, foi utilizado a técnica *z-score* para padronizar todos os dados a fim de que o algoritmo não fique enviesado para atributos com maior ordem de grandeza. A Equação 4 representa a técnica *z-score*, onde a média é 0 e o desvio padrão (σ) possui valor 1 (LOPES, 2022).

Tabela 4 – Dicionário de dados da base alemã.

Atributo	**Tipo**
Status	Categórico
Duração do mês	Numérico
Histórico de crédito	Categórico
Motivo do empréstimo	Categórico
Montante requerido	Numérico
Poupança / títulos	Categórico
Empregado desde	Categórico
Taxa de parcelamento em porcentagem da renda disponível	Numérico
Estado civil de sexo	Categórico
Fiadores	Categórico
Residência atual desde	Categórico
Propriedade	Categórico
Idade	Numérico
Outros planos de parcelamento	Categórico
Moradia	Categórico
Total de empréstimo existente neste banco	Categórico
Emprego	Categórico
Número de dependentes	Categórico
Número telefônico	Categórico
Trabalhador estrangeiro	Categórico
Classificação do cliente	Classe

Fonte: elaborada pelo autor.

$$x' = \frac{(x - \mu)}{\sigma} \tag{4}$$

4.3.1 Importação dos algoritmos de aprendizado de máquina

Os algoritmos escolhidos para serem aplicados neste trabalho foram: regressão logística, Árvore de decisão, *KNN, MLP, SVM, Gaussian Neive Bayes* e os métodos ensembles: *Random Forest* e *XGBoost*. Para importação do mesmo, foi utilizado as bibliotecas *SKlearn* e *XGBoost*. De primeiro momento, os algoritmos foram instanciados e os hiper-parâmetros foram selecionados de forma aleatória, a fim de comparar seu desempenho com as futuras otimizações. A Figura 19 mostra a instanciação dos algoritmos.

Tabela 5 – Dicionário de dados da base taiwanesa.

Atributo	**Tipo**
Quantidade de crédito cedido ao indivíduo e sua familia (em dólares)	Numérico
Sexo	Categórico
Escolaridade	Categórico
Estado civil	Categórico
Idade	Numérico
Histórico de pagamentos anteriores. Registros de pagamentos mensais anteriores (de abril a setembro de 2005) (PAY_0 a PAY_6)	Numérico
Valor da fatura (em dólares) De abril à setembro de 2005 BILL_AMT1 a BILL_AMT6	Numérico
Quantidade do pagamento anterior (em dólares). De abril à setembro de 2005 PAY_AMT1 a PAY_AMT6	Numérico
Inadimplência	Classe

Fonte: elaborada pelo autor.

Figura 19 – Instanciação de algoritmos

```
logistic_regression= LogisticRegression()

tree = DecisionTreeClassifier(min_samples_split=5, min_samples_leaf=3, random_state=seed)

mlp = MLPClassifier(hidden_layer_sizes=(10,) max_iter=10000, random_state=seed)

knn = KNeighborsClassifier(n_neighbors=3)

svm = SVC(kernel='linear')

gnb = GaussianNB()

RandomForest = RandomForestClassifier(n_estimators=10, random_state=seed)

XGBoosting = XGBClassifier(learning_rate=0.01, max_depth=10, n_estimators=650)
```

Fonte: elaborada pelo autor

4.3.2 Treinamento e Avaliação

A fim de treinar e testar os algoritmos e realizar a otimização de hiper-parâmetros, foi utilizado a técnica de validação cruzada e validação cruzada aninhada. Foi aplicado os métodos *cross validate* e *KFold* da biblioteca *sklearn.modelselection*. Para obter o desempenho

Tabela 6 – Dicionário de dados - base brasileira.

Atributo	**Tipo**
Inadimplente	Numérico
útil linhas inseguras	Numérico
Idade	Numérico
Número de vezes que passou de 30 e 59 dias inadimplente	Numérico
Razão débito	Numérico
Salário mensal	Numérico
Número de linhas de crédito em aberto	Numérico
Número de vezes que passou de 90 dias inadimplente	Numérico
Número de empréstimos imobiliários	Numérico
Número de vezes que passou de 60 e 89 dias inadimplente	Numérico
Número de dependentes	Numérico

Fonte: elaborada pelo autor.

dos algoritmos não otimizados, foi aplicado em cada base de dados a técnica de validação cruzada com *10folds*. Com o resultado do desempenho de cada algoritmo, a próxima etapa refere-se a otimização de hiper-parâmetros, onde será utilizada a técnica de validação cruzada aninhada, sendo *2folds* para a otimização e *10folds* para testar o desempenho dos melhores hiper-parâmetros. Para avaliar os modelos, foi selecionado a métrica de avaliação curva ROC. A escolha desta métrica foi definida devido sua característica em avaliar o desempenho do modelo ao categorizar dados dicotômicos, que resumindo em outras palavras, avalia o quão bom um modelo pode ser em classificar o cliente como adimplente ou inadimplente certeiramente. Além disto, a métrica não se baseia somente ao total de acertos/total da amostra, mas calcula cada predição em relação à taxa de verdadeiros positivos e a taxa de falsos positivos, formando a curva ROC e definindo o melhor valor obtido por meio da AUC (do inglês *area under curve*).

4.3.3 Otimização de hiper-parâmetros

Assim como descrito no capítulo anterior, a otimização visa testar diversos hiper-parâmetros a fim de encontrar o que melhor performa no algoritmo em certa métrica de avaliação. No presente trabalho, foram selecionadas 3 diferentes formas para realizar a otimização, sendo estas: *BayeSearch, RandomSearch* e Algoritmo Genético, onde seu conceito e funcionamento podem ser consultados na seção anterior. A otimização foi realizada com base da métrica curva ROC.

Em cada método de otimização, foi utilizado a técnica de validação cruzada com

2folds. Também foi definido para cada algoritmo utilizado nos métodos de otimização, o número máximo de 100 iterações. Para implementação, foi utilizado a biblioteca *scikit-optimize* e o método *BayesSearchCV* da biblioteca *skopt* para otimização *BayeSearch*. Para a implementação da otimização *RandomSearch*, foi utilizado o método *RandomizedSearchCV* da biblioteca *sklearn.modelselection*. Já para Algoritmo genético, foi utilizado o método *GASearchCV* da biblioteca *sklearngenetic*.

5 RESULTADOS

Neste capítulo são descritos os resultados obtidos com o desenvolvimento do trabalho, com o foco em apresentar o desempenho dos algoritmos em cada base de dados, bem como o desempenho de cada otimização de hiper-parâmetro. Ao final, realizar uma breve análise e verificar qual algoritmo/método de otimização apresentou melhores resultados para a classificação de clientes inadimplentes.

Primeiramente, foi analisado o desempenho de cada algoritmo de forma não otimizada em todas as bases de dados, no qual estão apresentados: Figura 20 - base australiana; Figura 21 - base alemã; Figura 22 - base taiwanesa e Figura 23 - base brasileira.

Figura 20 – Desempenho dos algoritmos na base australiana

	roc_auc
Regressão logística	0.930413
Árvore de decisao	0.843406
MLP	0.923778
KNN	0.879279
SVM	0.919617
Gaussian Naive Bayes	0.897501
Random Forest	0.916609
XGBoost	0.928283

Fonte: Elaborada pelo autor

Como é possível observar, na Figura 20, representando a base australiana, o algoritmo de Regressão Logística foi o melhor algoritmo comparado aos demais, obtendo 93%.

A Figura 21 apresenta o desempenho dos algoritmos na base Alemã, onde o algoritmo *Support Vector Machines* (SVM) obteve melhor desempenho com 91%. Enquanto a Figura 22 apresenta o desempenho dos algoritmos na base de dados taiwanesa, nesta base, o algoritmo que se destacou foi o *ensemble XGBoost* com o desempenho de 77.7%.

A Figura 23 apresenta o desempenho dos algoritmos na base brasileira. Ao realizar a análise, o algoritmo que apresentou melhor desempenho foi o *ensemble Random Forest* com 92.6%.

Pode-se observar que, durante as análises realizadas em todas as bases de dados, não foi somente um único algoritmo que apresentou os melhores resultados, mas em cada base de dados, sempre um algoritmo destacava-se dos demais. Como todos os algoritmos foram iniciados com parâmetros aleatórios, isto poderia ter contribuído para o desempenho diferente em cada base,

Figura 21 – Desempenho dos algoritmos na base alemã

	roc_auc
Regressão logística	0.715095
Árvore de decisao	0.668262
MLP	0.745571
KNN	0.673214
SVM	0.919617
Gaussian Naive Bayes	0.897501
Random Forest	0.748619
XGBoost	0.789905

Fonte: Elaborada pelo autor

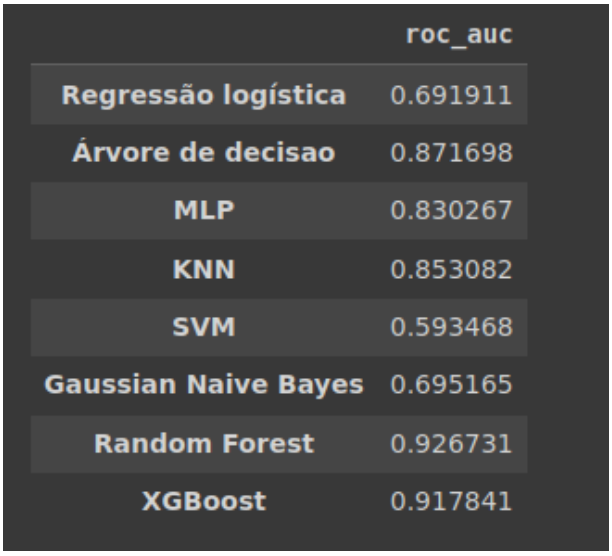
Figura 22 – Desempenho dos algoritmos na base taiwanesa

	roc_auc
Regressão logística	0.722116
Árvore de decisao	0.644555
MLP	0.772526
KNN	0.676447
SVM	0.698388
Gaussian Naive Bayes	0.736367
Random Forest	0.732348
XGBoost	0.777465

Fonte: Elaborada pelo autor

devido não estarem otimizados, e por mais que as bases de dados fossem semelhantes, existiam algumas divergências em relação aos atributos, trazendo diversidade no desempenho ao ser analisada por cada classificador. Os que apresentaram melhores desempenhos foram: regressão logística, SVM, *XGBoost* e *Random Forest*. Na próxima Seção é apresentado o desempenho dos algoritmos com seus hiper-parâmetros otimizados a fim de analisar melhorias ou não em seu desempenho e encontrar os que apresentaram os melhores resultados na classificação.

Figura 23 – Desempenho dos algoritmos na base brasileira



	roc_auc
Regressão logística	0.691911
Árvore de decisao	0.871698
MLP	0.830267
KNN	0.853082
SVM	0.593468
Gaussian Naive Bayes	0.695165
Random Forest	0.926731
XGBoost	0.917841

Fonte: Elaborada pelo autor

Tabela 7 – Hiper-parâmetros otimizados.

Algoritmo	**Hiper-parâmetros**
Regressão logística	Solver: newton-cg, lbfgs, liblinear, sag, saga. C: valores entre 0.01 e 100
Árvore de decisão	Random State: valores entre 1 e 10 Max depth: valores entre 1 e 50
MLP	Activation: identity, logistic, tanh, relu. Random state: valores entre 1 e 10.
KNN	Algorithm: auto, ball tree, kd tree e brute. neighbors: valores entre 1 e 50.
SVM	Gamma:valores entre 0.001 e 1. Kernel: Linear, rbf, sigmoid.
Naive Bayes	var smoothing: valores entre 0.001 e 10. priors: None.
Random Forest	Random State: valores entre 1 e 50. Max depth: valores entre 1 e 50.
XGBoost	Learning rate: valores entre 1 e 50. Max depth: valores entre 1 e 50.

Fonte: elaborada pelo autor.

5.0.1 Otimização de hiper-parâmetros

Para realizar a otimização, foram selecionados 2 hiper-parâmetros de cada algoritmo e aplicadas as técnicas de otimização: *BayeSearch, RandomSearch* e Algoritmo Genético. A Tabela 7 apresenta a relação dos algoritmos e seus hiper-parâmetros, com o intervalo de valores.

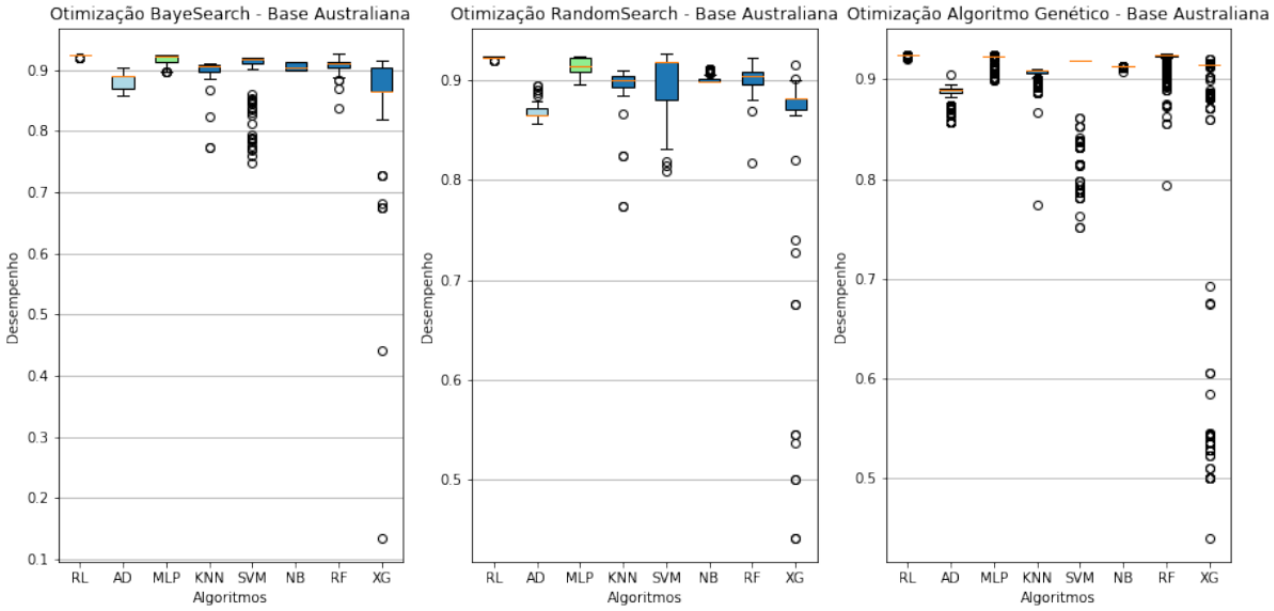
A otimização foi realizada utilizando a técnica de validação cruzada com *2folds* e avaliada pela métrica curva ROC. Cada método de otimização foi executado 2 vezes e

cada algoritmo de otimização (Bayesiana, Busca Aleatória e Algoritmo Genético) realizou 100 iterações em cada vez. Nas próximas subseções, serão apresentados o desempenho dos algoritmos nas respectivas bases de dados.

5.0.2 Base de dados Australiana

Após a realização das otimizações de hiper-parâmetros, foram gerados os gráficos *box plot*, onde apresentam o desempenho da otimização de cada algoritmo, uma tabela contendo o tempo gasto para que cada otimização fosse realizada e por fim, o gráfico com o resultado dos algoritmos otimizados. A Figura 24 ilustra o gráfico *box plot* representando cada tipo de otimização aplicado em cada algoritmo. É possível verificar a variação de resultados de cada método de otimização utilizado, onde é detalhado os valores mínimos, médios, máximos e valores *outliers*. Observa-se que o algoritmo *Random Forest*, aplicado à otimização bayesiana, alcançou os maiores resultados comparado aos demais.

Figura 24 – Gráfico *box plot*: otimização de hiper-parâmetros - base australiana



Fonte: Elaborada pelo autor

A Tabela 8 apresenta o tempo gasto para realizar a otimização de cada algoritmo e por fim, a Figura 25 apresenta o resultado das classificações com os algoritmos otimizados. Ao realizar a análise dos dados apresentados, o algoritmo que obteve maior destaque foi o *Random Forest* utilizando a otimização *bayesearch* obtendo um desempenho de 94.4%, onde os melhores hiper-parâmetros selecionados foram: *random state=50 e max depth=6*. O algoritmo que apresentou melhor tempo de otimização foi o *knn*, com 6 segundos para realizar a otimização *RandomSearch*. E por fim, o algoritmo que apresentou maior ganho ao ser otimizado foi a Árvore de decisão, onde teve um aumento de 84.3% para 92.5% em seu resultado.

Tabela 8 – Tempo de otimização - base australiana.

Algoritmo	**Otimização**	**Tempo (minutos)**
Regressão logística	BayeSearch	4m
Árvore de decisão	BayeSearch	4m54s
MLP	BayeSearch	5m55s
KNN	BayeSearch	4m3s
SVM	BayeSearch	4m17s
Naive Bayes	BayeSearch	4m8s
Random Forest	BayeSearch	4m12s
XGBoost	BayeSearch	5m24s
Regressão logística	RandomSearch	0.6s
Árvore de decisão	RandomSearch	0.2s
MLP	RandomSearch	3m37
KNN	RandomSearch	0.27s
SVM	RandomSearch	0.27s
Naive Bayes	RandomSearch	0.14s
Random Forest	RandomSearch	46s
XGBoost	RandomSearch	46s
Regressão logística	Algoritmo genético	1m14s
Árvore de decisão	Algoritmo genético	49s
MLP	Algoritmo genético	32m
KNN	Algoritmo genético	1m23s
SVM	Algoritmo genético	1m15
Naive Bayes	Algoritmo genético	58s
Random Forest	Algoritmo genético	1m48s
XGBoost	Algoritmo genético	2m20s

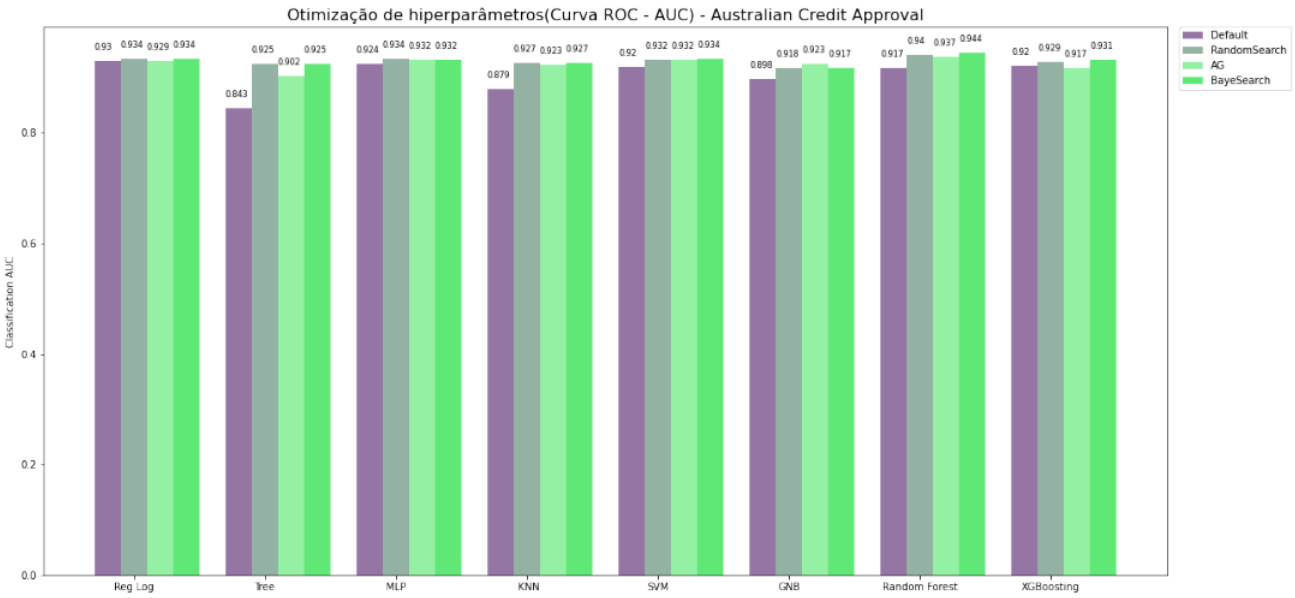
Fonte: elaborada pelo autor.

5.0.3 Base de dados alemã

Para a base de dados alemã, a Figura 26 apresenta a otimização dos hiper-parâmetros por meio do gráfico *box plot*, onde mostra com detalhes a variação de desempenho obtidos durante a otimização, podendo observar a média, valores mínimos e máximos, bem como *outliers* e a Tabela 9 apresenta o tempo utilizado para realizar a otimização em cada algoritmo. Após selecionados os melhores hiper-parâmetros, foram realizados novamente os testes de desempenho na base, conforme mostrado na Figura 27.

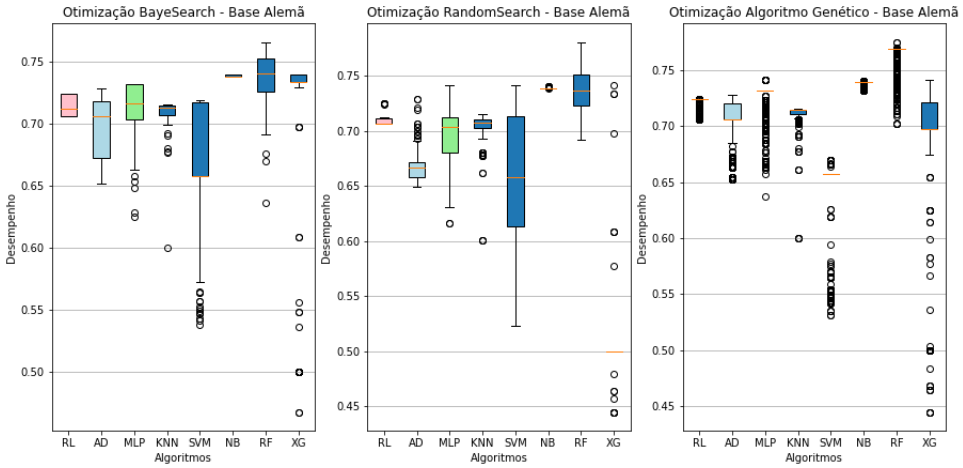
Como pode-se observar, o algoritmo que apresentou melhores resultados em termos de desempenho foi o *Random Forest* obtendo 79.3%, utilizando o método de otimização Algoritmo genético, onde os melhores hiper-parâmetros selecionados foram: *random state=24 e max depth=11*. O algoritmo que obteve melhor tempo de otimização foi o *Random Forest*, onde foi utilizado 23s para realizar a otimização por *RandomSearch*; e o algoritmo que obteve maior ganho ao realizar a otimização, comparado ao seu resultado anterior foi o KNN, onde obteve um aumento de 67.3% para 73.5% com a otimização *Randomsearch*.

Figura 25 – Desempenho dos algoritmos otimizados na base australiana



Fonte: Elaborada pelo autor

Figura 26 – Gráfico *box plot*: otimização de hiper-parâmetros - base alemã



Fonte: Elaborada pelo autor

5.0.4 Base de dados taiwanesa

A seguir, será apresentado os resultados obtidos com a otimização de hiper-parâmetros na base taiwanesa. A Figura 28 mostra o gráfico *box plot*, representando o desempenho alcançado nas iterações de cada método de otimização de hiper-parâmetros de cada algoritmo e a Tabela 10 apresenta o tempo utilizado para a realização de cada otimização. Por fim, a Figura 29 apresenta o desempenho dos algoritmos otimizados.

O algoritmo que obteve melhor resultado foi o ensemble *Random Forest* obtendo

Tabela 9 – Tempo de otimização - base alemã.

Algoritmo	**Otimização**	**Tempo (minutos)**
Regressão logística	BayeSearch	4m38s
Árvore de decisão	BayeSearch	3m47s
MLP	BayeSearch	9m34s
KNN	BayeSearch	4m28s
SVM	BayeSearch	4m32s
Naive Bayes	BayeSearch	3m17s
Random Forest	BayeSearch	4m19s
XGBoost	BayeSearch	4m
Regressão logística	RandomSearch	8s
Árvore de decisão	RandomSearch	1m66s
MLP	RandomSearch	4m35s
KNN	RandomSearch	4s
SVM	RandomSearch	6s
Naive Bayes	RandomSearch	13s
Random Forest	RandomSearch	55s
XGBoost	RandomSearch	69s
Regressão logística	Algoritmo genético	10s
Árvore de decisão	Algoritmo genético	10s
MLP	Algoritmo genético	73m
KNN	Algoritmo genético	116m
SVM	Algoritmo genético	2m23s
Naive Bayes	Algoritmo genético	1m2s
Random Forest	Algoritmo genético	2m3s
XGBoost	Algoritmo genético	3m55

Fonte: elaborada pelo autor.

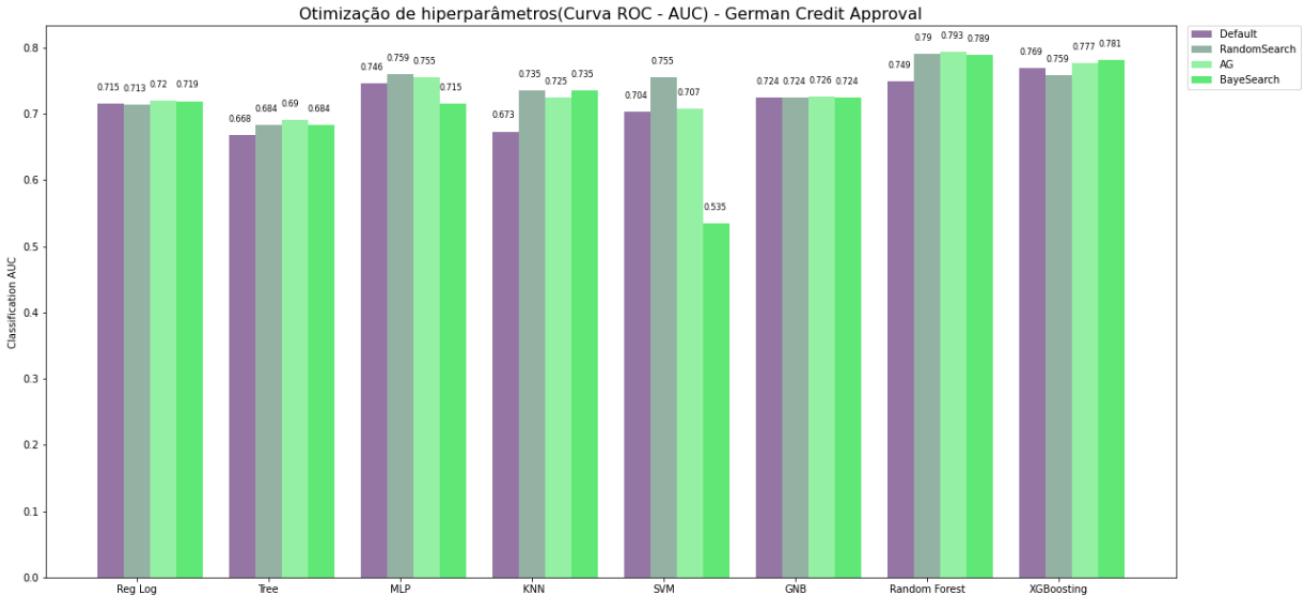
85.1% por meio da otimização *RandomSearch*, onde os melhores hiper-parâmetros selecionados foram: *random state=15 e max depth=9*. O algoritmo que obteve o menor tempo de otimização foi *Random Forest* utilizando 23s para ser otimização via *Randomsearch*. Já o algoritmo que obteve maiores resultados com a otimização foi a Árvore de decisão, onde obteve um ganho de 65% para 79% com a otimização *Randomsearch*.

5.0.5 Base de dados Brasileira

Por fim, será apresentado os resultados das otimizações dos algoritmos na base brasileira. A Figura 30 apresenta as otimizações de cada algoritmo por meio do gráfico *box plot*, onde mostra o desempenho das iterações, os valores máximo, médio e mínimo, bem como a presença de *outliers*. A Tabela 11 apresenta o tempo utilizado para realizar cada otimização e a Figura 31 mostra o desempenho de cada algoritmo otimizado aplicado à base de dados brasileira.

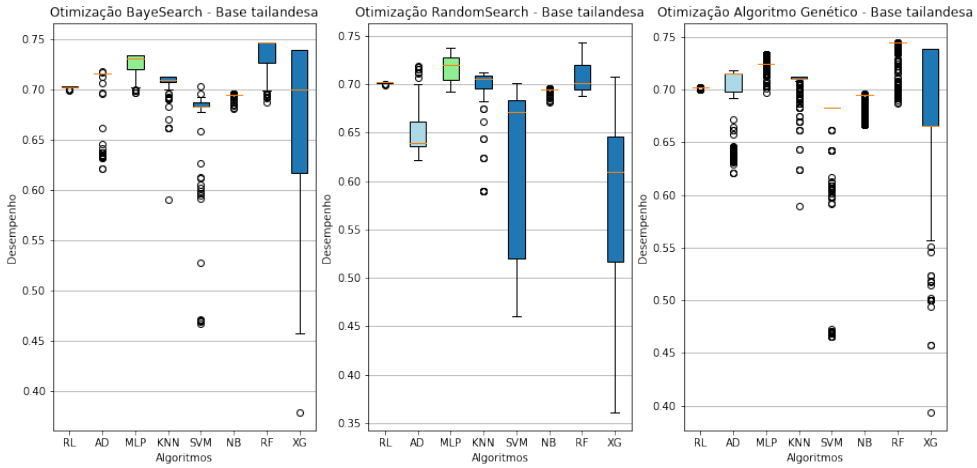
O algoritmo que apresentou melhor desempenho foi o ensemble *Random Forest*

Figura 27 – Desempenho dos algoritmos otimizados na base alemã



Fonte: Elaborada pelo autor

Figura 28 – Gráfico *box plot*: otimização de hiper-parâmetros - base taiwanesa



Fonte: Elaborada pelo autor

obtendo 85.5% por meio da otimização por algoritmo genético. Os melhores hiper-parâmetros selecionados foram: *random state=5 e max depth=5*. O algoritmo que apresentou melhor tempo de otimização foi o KNN, levando 5s para ser otimizado via *Randomsearch*. Já o algoritmo que obteve maior ganho ao ser otimizado foi a árvore de decisão, obtendo um aumento de 65% para 82.2%, via Algoritmo Genético.

Tabela 10 – Tempo de otimização - base taiwanesa.

Algoritmo	**Otimização**	**Tempo (minutos)**
Regressão logística	BayeSearch	6m28s
Árvore de decisão	BayeSearch	4m49s
MLP	BayeSearch	17m26s
KNN	BayeSearch	16m
SVM	BayeSearch	23m44s
Naive Bayes	BayeSearch	3m40s
Random Forest	BayeSearch	4m28s
XGBoost	BayeSearch	8m06s
Regressão logística	RandomSearch	9m34s
Árvore de decisão	RandomSearch	15m6s
MLP	RandomSearch	9m59s
KNN	RandomSearch	2m16s
SVM	RandomSearch	13m30
Naive Bayes	RandomSearch	2m77s
Random Forest	RandomSearch	23s
XGBoost	RandomSearch	2m14
Regressão logística	Algoritmo genético	13m19s
Árvore de decisão	Algoritmo genético	5m18s
MLP	Algoritmo genético	154m
KNN	Algoritmo genético	32m41s
SVM	Algoritmo genético	511m
Naive Bayes	Algoritmo genético	4m45
Random Forest	Algoritmo genético	7m8s
XGBoost	Algoritmo genético	46m41

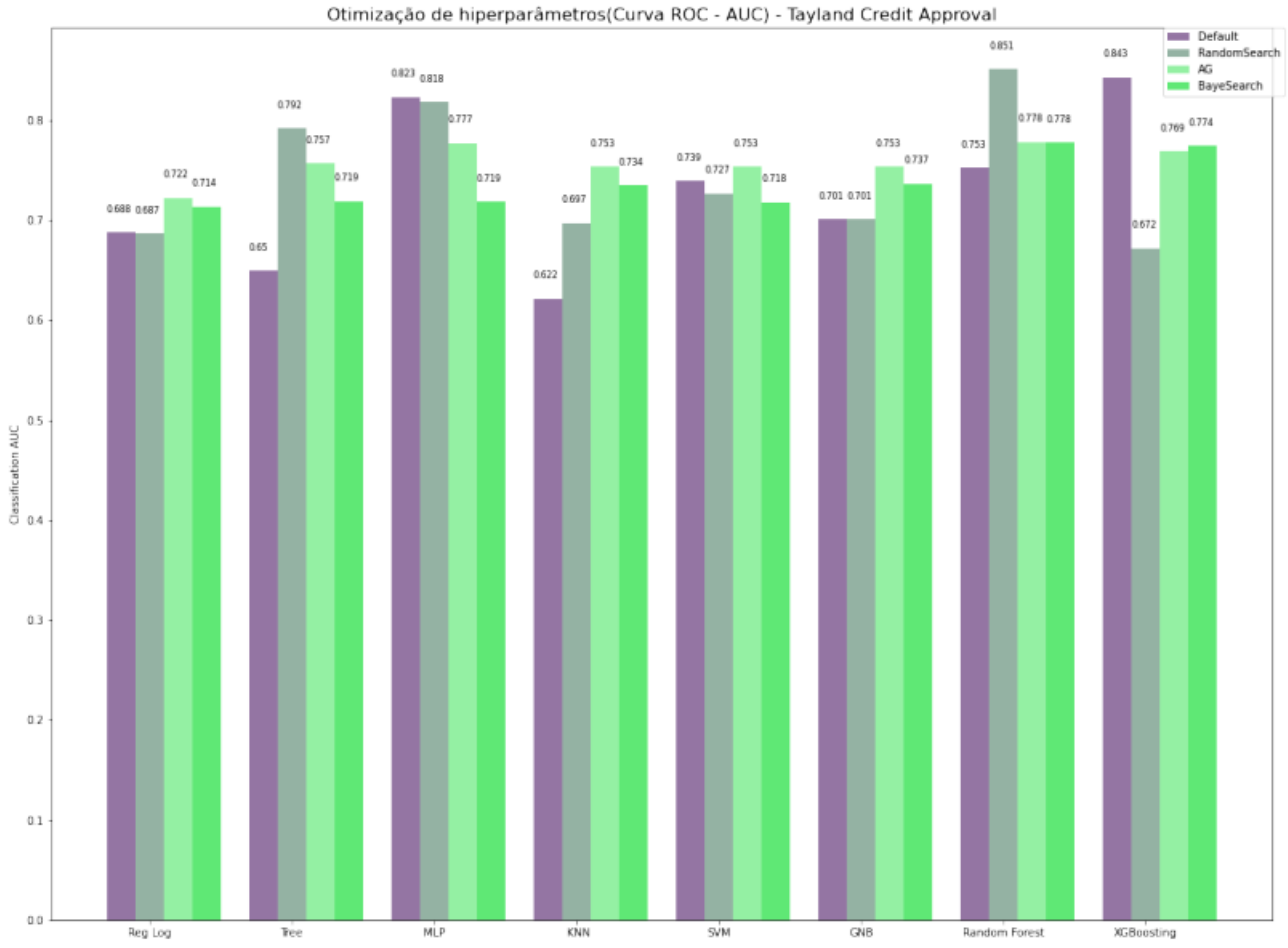
Fonte: elaborada pelo autor.

5.0.6 Análise geral

Realizando uma análise mais isolada no ensemble *Random Forest*, o qual foi o algoritmo que apresentou melhor desempenho comparado aos demais em todos os testes, a Figura 32 apresenta os resultados antes e depois das otimizações realizadas em cada base de dados, na qual apresentou ganhos satisfatórios ao ser otimizado por qualquer método, destacando-se a otimização por algoritmo genético, obtendo melhor resultado nas bases alemã e brasileira.

Também foi feito um estudo observando o tempo que cada método de otimização, que apresentou melhor desempenho, utilizou para realizar a otimização no algoritmo *Random Forest* em cada base de dados estudada. A Figura 33 apresenta detalhadamente esta comparação, na qual a busca em grade foi o método mais rápido ao realizar a otimização, seguindo pelo algoritmo genético. É possível observar que, ao analisar o desempenho do algoritmo *Random Forest* em cada base de dados, diferentes métodos de otimização se destacaram. Tal resultado pode ter sido motivado devido aos diferentes atributos presentes em cada base de dados, bem como a forma com que cada método de otimização funciona. O número de iterações e intervalo

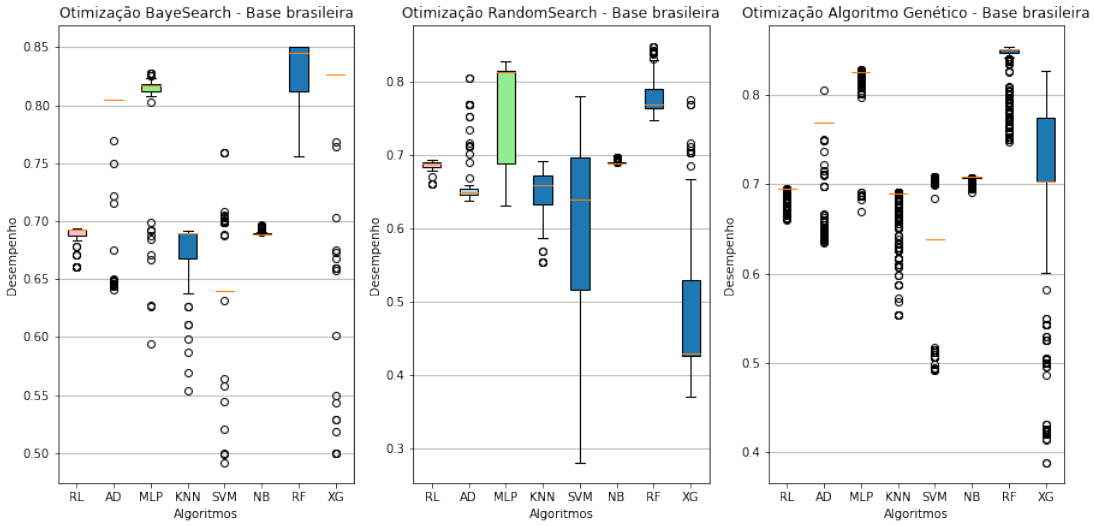
Figura 29 – Desempenho dos algoritmos otimizados na base taiwanesa



Fonte: Elaborada pelo autor

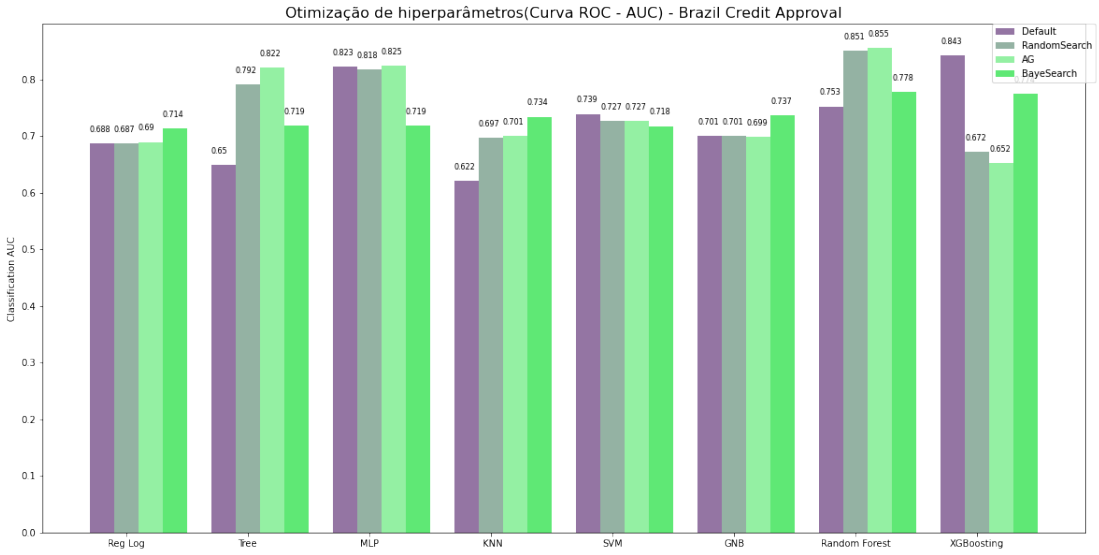
de valores de cada hiper-parâmetro também são variáveis a se considerar.

Figura 30 – Gráfico *box plot*: otimização de hiper-parâmetros - base brasileira



Fonte: Elaborada pelo autor

Figura 31 – Desempenho dos algoritmos otimizados na base brasileira



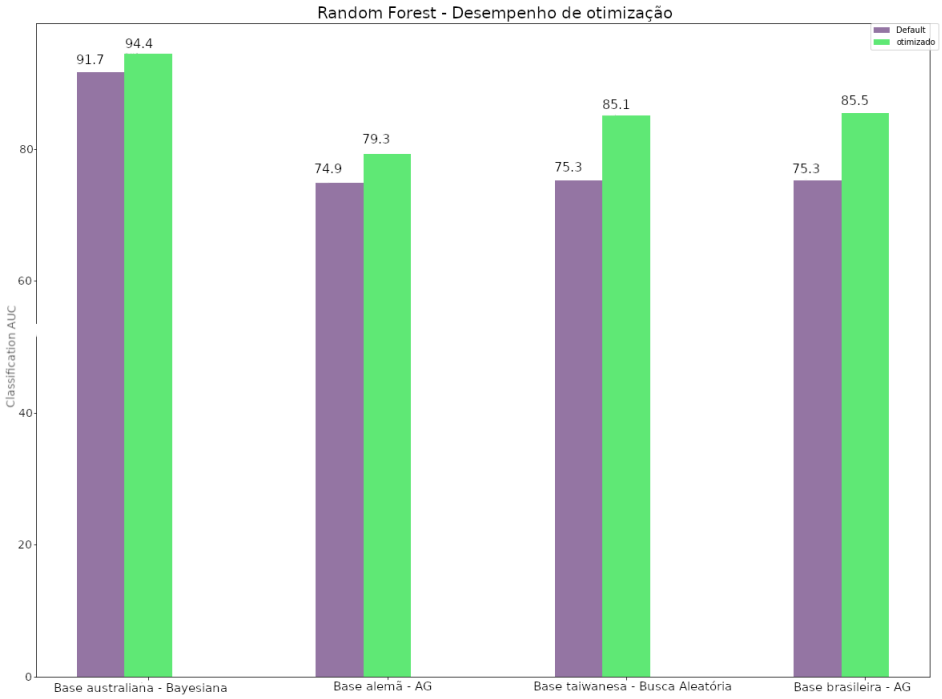
Fonte: Elaborada pelo autor

Tabela 11 – Tempo de otimização - base brasileira.

Algoritmo	**Otimização**	**Tempo (minutos)**
Regressão logística	BayeSearch	4m54s
Árvore de decisão	BayeSearch	4m22s
MLP	BayeSearch	10m5s
KNN	BayeSearch	5m13s
SVM	BayeSearch	10m34s
Naive Bayes	BayeSearch	3m26s
Random Forest	BayeSearch	3m52
XGBoost	BayeSearch	5m2s
Regressão logística	RandomSearch	5m37s
Árvore de decisão	RandomSearch	57s
MLP	RandomSearch	5m4s
KNN	RandomSearch	5s
SVM	RandomSearch	3m
Naive Bayes	RandomSearch	1m9s
Random Forest	RandomSearch	14s
XGBoost	RandomSearch	1m15
Regressão logística	Algoritmo genético	1m4s
Árvore de decisão	Algoritmo genético	1m19
MLP	Algoritmo genético	62m
KNN	Algoritmo genético	29m
SVM	Algoritmo genético	117m
Naive Bayes	Algoritmo genético	1m2s
Random Forest	Algoritmo genético	2m28s
XGBoost	Algoritmo genético	23m

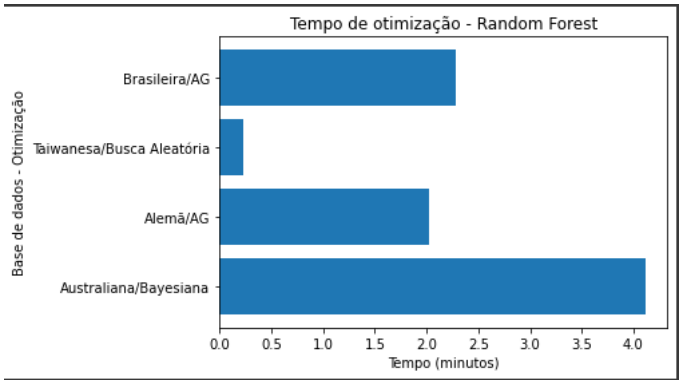
Fonte: elaborada pelo autor.

Figura 32 – Otimização - Random Forest



Fonte: Elaborada pelo autor

Figura 33 – Tempo de otimização - Random Forest



Fonte: Elaborada pelo autor

6 CONCLUSÃO

Este trabalho teve como objetivo realizar um estudo comparando o desempenho de diversos algoritmos de AM, focando na otimização de hiper-parâmetros, a fim de encontrar um classificador que obtivesse melhor destaque no contexto de análise de risco de crédito. Ao analisar o desempenho de cada algoritmo nas quatro bases de dados, é possível concluir que o *ensemble Random Forest* foi aquele que apresentou melhores resultados em todas as análises otimizadas. Este resultado somente foi possível devido a otimização de hiper-parâmetros, onde o tipo de otimização que maior obteve destaque foi por algoritmo genético, elevando o desempenho do classificador nas bases de dados alemã e brasileira. Com isto, é possível concluir que a otimização de hiper-parâmetros é uma etapa fundamental ao aplicar algoritmos de aprendizado de máquina em qualquer contexto, principalmente na análise de risco de crédito, tendo em vista que ao realizar a otimização, percebe-se um aumento satisfatório na assertividade do classificador, na qual torna-se um enorme diferencial. Também é possível concluir que o algoritmo *Random Forest* apresentou ser um classificador de grande potencial na análise de risco de crédito, superando modelos consolidados em realizar esta tarefa, tal como o algoritmo de regressão logística.

6.1 Limitações

O presente trabalho foi desenvolvido utilizando a ferramenta *Google Colab* em sua versão gratuita, onde possui certas limitações de tempo em relação aos recursos computacionais disponíveis para realizar o processamento e otimização dos algoritmos. Devido a isto, foram limitados o número de hiper-parâmetros a serem otimizados, bem como a faixa de valores a serem exploradas. Outra limitação foi o tempo disponível, onde não foi possível realizar mais testes envolvendo outros algoritmos e tipos de otimização de hiper-parâmetros.

6.2 Trabalhos Futuros

Devido a isto, é sugerido como continuação deste trabalho a exploração de outros algoritmos de aprendizado de máquina, bem como analisar outras técnicas de otimização, além de aumentar a quantidade de hiper-parâmetros a serem otimizados. Também fica a sugestão em implementar técnicas de pré-processamento de dados (como análise de componente principal e outras) visando selecionar os atributos da base de dados que possuem maior importância na classificação, consequentemente aumentando o desempenho dos algoritmos.

Referências

AALST, W. M. Van der et al. Process mining: a two-step approach to balance between underfitting and overfitting. **Software & Systems Modeling**, Springer, v. 9, n. 1, p. 87–111, 2010. Citado na página 21.

ALTMAN, E. I.; SAUNDERS, A. Credit risk measurement: Developments over the last 20 years. **Journal of banking & finance**, Elsevier, v. 21, n. 11-12, p. 1721–1742, 1997. Citado na página 12.

ARENAS, R. **Are You Still Using Grid Search for Hyperparameters Optimization?** 2021. Urlhttps://towardsdatascience.com/hyperparameters-tuning-from-grid-search-to-optimization-a09853e4e9b8/. Citado 2 vezes nas páginas 26 e 27.

ASUNCION, A.; NEWMAN, D. **UCI machine learning repository**. [S.l.]: Irvine, CA, USA, 2007. Citado na página 33.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **Journal of machine learning research**, v. 13, n. 2, 2012. Citado na página 25.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006. Citado 2 vezes nas páginas 14 e 18.

BUITINCK, L. et al. Api design for machine learning software: experiences from the scikit-learn project. **arXiv preprint arXiv:1309.0238**, 2013. Citado na página 31.

BURRELL, J. How the machine 'thinks': Understanding opacity in machine learning algorithms. **Big data & society**, Sage Publications Sage UK: London, England, v. 3, n. 1, p. 2053951715622512, 2016. Citado na página 10.

CASTRO, J. S. d. et al. Estudo comparativo entre metodologias de aprendizado de máquina e híbridas aplicadas a risco de crédito. Fundação Escola de Comércio Álvares Penteado, 2019. Citado 2 vezes nas páginas 28 e 30.

CHEN, Y.-S.; CHENG, C.-H. Hybrid models based on rough set classifiers for setting credit rating decision rules in the global banking industry. **Knowledge-Based Systems**, Elsevier, v. 39, p. 224–239, 2013. Citado na página 10.

COELHO, F. F. et al. Analisando métodos de machine learning e avaliação do risco de crédito. **Revista Gestão & Tecnologia**, v. 21, n. 1, p. 89–116, 2021. Citado 5 vezes nas páginas 13, 20, 21, 29 e 30.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 18.

DAMETTO, R. C. Estudo da aplicação de redes neurais artificiais para predição de séries temporais financeiras. Universidade Estadual Paulista (UNESP), 2018. Citado na página 17.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. **International workshop on multiple classifier systems**. [S.l.], 2000. p. 1–15. Citado na página 19.

EFRON, B.; TIBSHIRANI, R. J. **An introduction to the bootstrap**. [S.l.]: CRC press, 1994. Citado na página 20.

EXPERIAN, S. **5 em cada 10 PMEs sofreram com inadimplência de clientes durante a pandemia. 2021**. 2020. Urlhttps://www.serasaexperian.com.br/conteudos/estudos-e-pesquisas/5-em-cada-10-pmes-sofreram-com-inadimplencia-de-clientes-durante-a-pandemia. Citado na página 10.

EXPERIAN, S. **Qual é a melhor estratégia para lidar com clientes inadimplentes?** 2020. Urlhttps://empresas.serasaexperian.com.br/blog/clientes-inadimplentes/. Citado na página 10.

FREUND, Y.; SCHAPIRE, R. E. et al. Experiments with a new boosting algorithm. In: CITESEER. **icml**. [S.l.], 1996. v. 96, p. 148–156. Citado na página 20.

GROVER, P. **Gradient Boosting from scratch**. 2017. Urlhttps://blog.mlreview.com/gradient-boosting-from-scratch-1e317ae4587d. Citado na página 19.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The elements of statistical learning. springer series in statistics. **New York, NY, USA**, 2001. Citado na página 20.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in science & engineering**, IEEE Computer Society, v. 9, n. 03, p. 90–95, 2007. Citado na página 33.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. **Automated machine learning: methods, systems, challenges**. [S.l.]: Springer Nature, 2019. Citado na página 24.

JÚNIOR, W. J. de A. Métodos de otimização hiperparamétrica: um estudo comparativo utilizando árvores de decisão e florestas aleatórias na classificação binária. Universidade Federal de Minas Gerais, 2018. Citado 2 vezes nas páginas 28 e 30.

KIM, E. **Everything You Wanted to Know about the Kernel Trick**. 2013. Urlhttps://www.eric-kim.net/eric-kim-net/posts/1/kernel$_trick.html/.Citadonapgina19$.

KIM, Y.; CHUNG, M. An approach to hyperparameter optimization for the objective function in machine learning. **Electronics**, MDPI, v. 8, n. 11, p. 1267, 2019. Citado na página 24.

LALL, V. Google colab—the beginner's guide. **Lean In Women in Technology, India**, v. 1, 2018. Citado na página 33.

LEWIS, E. **Introduction to credit scoring**. [S.l.]: San Rafael: Athena Press, 1992. Citado 2 vezes nas páginas 10 e 12.

LOPES, R. S. Comparação de métodos de aprendizado de máquina para análise de risco de crédito. Serra, 2022. Citado 4 vezes nas páginas 14, 15, 20 e 34.

LUPUM, C. **Predictive model for credit approval**. 2020. Urlhttps://www.kaggle.com/code/caesarlupum/predictive-model-for-credit-approval/data. Citado na página 33.

MARTINS, M. A. dos S.; METTE, F.; MACEDO, G. R. de. A utilizaçao de redes neurais artificiais para a estimaçao dos preços da petrobrás pn na bovespa. **ConTexto**, v. 8, n. 14, 2008. Citado na página 16.

METSIS, V.; ANDROUTSOPOULOS, I.; PALIOURAS, G. Spam filtering with naive bayes-which naive bayes? In: MOUNTAIN VIEW, CA. **CEAS**. [S.l.], 2006. v. 17, p. 28–69. Citado na página 18.

MITCHELL, T. M.; MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1. Citado 2 vezes nas páginas 17 e 18.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, Manole, v. 1, n. 1, p. 32, 2003. Citado na página 13.

MÜLLER, A. C.; GUIDO, S. **Introduction to machine learning with Python: a guide for data scientists**. [S.l.]: "O'Reilly Media, Inc.", 2016. Citado na página 22.

MUNKHDALAI, L. et al. An empirical comparison of machine-learning methods on bank client credit assessments. **Sustainability**, MDPI, v. 11, n. 3, p. 699, 2019. Citado na página 12.

MURPHY, K. P. **Machine learning: a probabilistic perspective**. [S.l.]: MIT press, 2012. Citado 4 vezes nas páginas 14, 18, 20 e 34.

NASCIMENTO, P. S. et al. Modelo preditivo para avaliação de crédito em empréstimos pessoais. In: SBC. **Anais da VII Escola Regional de Informática de Goiás**. [S.l.], 2019. p. 223–236. Citado 2 vezes nas páginas 29 e 30.

NORVIG, P. R.; INTELLIGENCE, S. A. A modern approach. **Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems**, v. 90, p. 33–48, 2002. Citado na página 13.

OLIPHANT, T. E. **Guide to NumPy**. 2006. Urlhttps://web.mit.edu/dvp/Public/numpybook.pdf. Citado na página 32.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. **the Journal of machine Learning research**, JMLR. org, v. 12, p. 2825–2830, 2011. Citado na página 31.

PELLICER, L. F. A. O. **Otimização de hiperparâmetros de modelos machine learning com BarySearch.** Tese (Doutorado) — Universidade de São Paulo, 2020. Citado 2 vezes nas páginas 23 e 25.

SCHMITT, V. F. Uma análise comparativa de técnicas de aprendizagem de máquina para prever a popularidade de postagens no facebook. 2013. Citado na página 17.

SILVA, I. N.; SPOTTI, D. H.; FLAUZINO, R. A. **Redes neurais artificiais para engenharia e ciências aplicadas**. [S.l.]: Artliber, 2010. Citado na página 16.

SILVA, L. F. B. d. Utilização de técnicas de machine learning para detecção de botnets. Universidade Estadual Paulista (Unesp), 2018. Citado 4 vezes nas páginas 18, 19, 22 e 32.

SILVA, V. H. M. C.; FRANÇA, J. M. S. de. Modelos de machine learning na classificação de pobreza: Uma aplicação para o estado do ceará. Citado na página 23.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. **Advances in neural information processing systems**, v. 25, 2012. Citado na página 23.

TAN, P.; KUMAR, V.; STEINBACH, M. **Introdução ao datamining: mineração de dados**. [S.l.]: Ciência Moderna, 2009. Citado na página 15.

THOMAS, L.; CROOK, J.; EDELMAN, D. **Credit scoring and its applications**. [S.l.]: SIAM, 2017. Citado na página 10.

THOMAS, L. C. **Consumer credit models: Pricing, profit and portfolios**. [S.l.]: OUP Oxford, 2009. Citado na página 13.

ZHANG, H. The optimality of naive bayes. **Aa**, v. 1, n. 2, p. 3, 2004. Citado na página 17.