Universidade Federal de Pernambuco

Centro de Informática

Graduação em Engenharia da Computação

Coloração de Advertência: Desenvolvimento de Uma Técnica de Pré-processamento Baseada em Algoritmos Genéticos.

Víctor Hugo Meirelles Silva

Trabalho de Graduação

Recife, Pernambuco

Março , 2025

Universidade Federal de Pernambuco

Centro de Informática

Víctor Hugo Meirelles Silva

Coloração de Advertência: Desenvolvimento de Uma Técnica de Pré-processamento Baseada em Algoritmos Genéticos.

Trabalho apresentado ao Curso de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de concentração: Inteligência artificial, Algoritmos Evolutivos.

Orientador: Adriano Augusto de Moraes Sarmento

Recife, Pernambuco

Março, 2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Silva, Víctor Hugo Meirelles.
Coloração de advertência: desenvolvimento de uma técnica de pré-processamento baseada em algoritmos genéticos. / Víctor Hugo Meirelles Silva. - Recife, 2025.
70 p.

Orientador(a): Adriano Augusto de Moraes Sarmento
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2025.

1. Machine Learning. 2. Algoritmo Evolutivo. 3. Coloração de Advertência. I. Sarmento, Adriano Augusto de Moraes. (Orientação). II. Título.

000 CDD (22.ed.)

VÍCTOR HUGO MEIRELLES SILVA

COLORAÇÃO DE ADVERTÊNCIA: Desenvolvimento de Uma Técnica de Pré-processamento Baseada em Algoritmos Genéticos.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: 20/08/2025

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Augusto de Moraes Sarmento (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto (Examinador Interno)

Universidade Federal de Pernambuco

Dedico este trabalho a Deus e minha família.

Agradecimentos

Eu gostaria de agradecer à minha família, principalmente meus pais, Geraldo Pereira e Francilene Batista que sempre cuidaram de mim e continuam me ensinando a cada dia, meus irmãos, Lucas Emanuel, meu chapa e João Gabriel, meu mano. Um agradecimento especial para minha tia Antonia Leide que deixa saudades junto de belas memórias.

Também agradeço aos meus amigos José Victor e Andresa Almeida, que me acolheram em Recife e me fizeram sentir como se estivesse sempre em família, sempre juntos nos melhores momentos e nas dificuldades. Agradeço também aos amigos que fiz na residência estudantil que morei, principalmente Lucas Calixto, Gabriel Oliveira, Sophia e Maria Fernanda, vivi vários momentos que marcantes perto dessas pessoas.

Quero agradecer aos meus amigos de Teresina, especialmente à Maria Carolina, que é uma das amigas mais antigas que tenho e me apoiou em diversos momentos da minha vida. Agradecimento aos meus parceiros de robótica do ensino médio Kennyo Fortes e Thiago César, que além de colegas de equipe, são amigos próximos. Agradeço ao meu amigo Vitor Emanuel, que sempre foi um exemplo de pessoa e alguém em quem se inspirar.

Agradeço aos meus amigos de curso, principalmente Matheus Teotonio, Daniel Perazzo, Riei Joaquim, Victor Ximenes e Zenio Angelo. Todos fizeram parte e tiveram grnade importância na minha vida acadêmica e vida pessoal.

Um agradecimento especial a Zilde Souto, que está comigo desde o início do curso, passamos por vários desafios juntos e várias noites acordados estudando. Outro agradecimento a Victor Miguel que sempre nos acompanhava nos trabalhos e nas conversas.

Agradeço também a Pedro Nogueira, José Carlos e Lucas Ambrósio, companheiros da Maratona de programação, pessoas muito esforçadas, talentosas, além de companheiros que me ensinaram lições que eu levarei para a vida.

Aos meus companheiros que além da faculdade e amizade estavam sempre presentes nas situações mais inusitadas, Fernando Macedo e Mateus Elias, agradeço pelas conversas, histórias contadas e vividas e pela amizade.

Agradeço aos amigos da Maratona e aos amigos da equipe Maracatrônics. Em ambos os projetos pude aprender diversas coisas além do curso de engenharia.

Finalmente agradeço também aos professores do Centro de Informática essenciais para minha formação, principalmente ao professor Adriano Sarmento pela orientação.

"Viver é lutar.
A vida é combate,
Que os fracos abate,
Que os fortes, os bravos,
Só pode exaltar."

— Canção do Tamoio – Gonçalves Dias

Resumo

Ao utilizar-se de uma técnica de aprendizagem de máquina com intuito de resolver algum problema, faz-se necessário um conjunto de dados que descreva a situação em questão. A abordagem usada tem a missão de encontrar relações entre os valores presentes no conjunto de treinamento e, a partir dessas relações, ser capaz de classificar ou fazer estimativas sobre novos registros. Esse estudo propõe uma nova técnica chamada de Coloração de Advertência que tem o intuito de evidenciar algumas características, de forma a facilitar o trabalho do algoritmo de inteligência artificial utilizado. A nova técnica se diferencia de outras abordagens ao utilizar de algoritmos genéticos para encontrar transformações a serem feitas na base de dados original, que possibilitam um melhor aprendizado por outros algoritmos.

Palavras-chave: Machine Learning. Algoritmo Evolutivo. Coloração de Advertência.

Abstract

When using a machine learning technique to solve a problem, a set of data that describes the situation in question is necessary. The approach used has the mission of finding relationships between the values present in the training set and, based on these relationships, being able to classify or make estimates about new records. This study proposes a new technique called Coloração de Advertência that aims to highlight some characteristics, in order to facilitate the work of the artificial intelligence algorithm used. The new technique differs from other approaches by using genetic algorithms to find transformations to be made to the original database, which enables better learning by other algorithms.

Keywords: Machine Learning. Evolutionary Algorithm. Aposematic Coloration.

Lista de Tabelas

4.1 Matriz de Confusão sem pré-processamento. Fonte: Elaborado pelo autor (2025). 40
4.2 Matriz de Confusão com a utilização do PolynomialFeatures. Fonte: Elaborado pelo autor (2025). 41
4.3 Matriz de Confusão com a utilização do SplineTransformer. Fonte: Elaborado pelo autor (2025). 41
4.4 Matriz de Confusão utilizando Feature Engeneering com as features selecionadas a partir do LIME. Fonte: Elaborado pelo autor (2025). 41
4.5 Matriz de Confusão com a proposta da Coloração de Advertência. Fonte: Elaborado pelo autor (2025). 42
4.6 Matriz de Confusão sem pré-processamento. Fonte: Elaborado pelo autor (2025). 42
4.7 Matriz de Confusão com a utilização do PolynomialFeatures. Fonte: Elaborado pelo autor (2025). 43
4.8 Matriz de Confusão com a utilização do SplineTransformer. Fonte: Elaborado pelo autor (2025). 43
4.9 Matriz de Confusão utilizando Feature Engeneering com as features selecionadas a partir do LIME. Fonte: Elaborado pelo autor (2025). 44
4.10 Matriz de Confusão com a proposta da Coloração de Advertência. Fonte: Elaborado pelo autor (2025). 44
4.11 Médias de acurácia por Iteração. Fonte: Elaborado pelo autor (2025). 47
4.12 Média de acurácia total. Fonte: Elaborado pelo autor (2025). 47
4.13 Médias de acurácia por Iteração. Fonte: Elaborado pelo autor (2025). 48
4.14 Média de acurácia total. Fonte: Elaborado pelo autor (2025). 49
4.15 Tempo em horas para a realização do Grid Search em cada modelo da base do Censo. Fonte: Elaborado pelo autor (2025). 52

4.16 Tempo em minutos para a realização do Grid Search em cada modelo da base de doenças cardíacas. Fonte: Elaborado pelo autor (2025). 52
4.17 Tempo em horas para a realização da etapa evolutiva da Coloração de Advertência em cada base de dados. Fonte: Elaborado pelo autor (2025). 52
4.18 Valores percentuais obtidos na base de dados do Censo. Fonte: Elaborado pelo autor (2025). 53
4.19 Valores percentuais obtidos na base de doenças cardíacas. Fonte: Elaborado pelo autor (2025). 53

Lista de Figuras

2.1 Três tipos de Machine Learning. Fonte: V. Raschka; Mirjalili (2019) [1]. 4

2.2 Estrutura de um neurônio artificial. Fonte: Kalita (2023) [2]. 5

2.3 Rede com camada de entrada e saída. Fonte: Kalita (2023) [2]. 6

2.4 Representação com camadas ocultas. Fonte: Kalita (2023) [2]. 6

2.5 A figura mostra o Feed Forward nos passos 1 a l. Do l+1 em diante é mostrado o erro da saída sendo propagado do fim ao início atualizando os pesos da rede neural. Fonte: Kalita (2023) [2]. 8

2.6 Hiperplano traçado pelo SVM. Fonte: V. Raschka; Mirjalili (2019) [1]. 10

2.7 Transformação de um espaço bidimensional para tridimensional Fonte: V. Raschka; Mirjalili (2019) [1]. 11

2.8 Fluxograma de um algoritmo evolutivo. Fonte: Elaborada pelo autor (2025). . . 13

2.9 Passo a passo do algoritmo LIME. Fonte: Visani (2020) [3] 15

3.1 Esquema que mostra o fluxo da técnica de Coloração de Advertência. Fonte: Elaborado pelo autor (2025). 18

3.2 Percentual de pessoas que recebem 50 mil ou menos em azul e pessoas que recebem mais de 50 mil em laranja. Fonte: Elaborado pelo autor (2025). 23

3.3 Matriz de correlação entre as variáveis numéricas. Fonte: Elaborado pelo autor (2025). 24

3.4 Percentual de pessoas que possuem doenças cardíacas em azul e que não possuem em laranja. Fonte: Elaborado pelo autor (2025). 26

3.5 Enter Caption 27

3.6 Gráfico relacionando uma variável nova, resultado da aplicação da função de transformação sobre capital-gain e hour-per-week, com a variável dos dados originais hour-per-week. Fonte: Elaborado pelo autor (2025). 30

3.7 Gráfico 3.6 dividido para cada classe. Fonte: Elaborado pelo autor (2025). . . . 30

3.8 Gráfico relacionando duas variáveis novas, capital-gain combinada com hour-per-week e age combinada com education-num. Fonte: Elaborado pelo autor (2025). 31
3.9 Gráfico 3.8 dividido para cada classe. Fonte: Elaborado pelo autor (2025). . . . 31
3.10 Gráfico relacionando duas variáveis originais: capital-gain e hour-per-week. Fonte: Elaborado pelo autor (2025). 32
3.11 Gráfico 3.10 dividido para cada classe. Fonte: Elaborado pelo autor (2025). . . . 32
3.12 Gráfico relacionando uma variável nova, Cholesterol combinada com Oldpeak, com a variável original MaxHR. Fonte: Elaborado pelo autor (2025). 33
3.13 Gráfico 3.12 dividido para cada classe. Fonte: Elaborado pelo autor (2025). . . . 33
3.14 Gráfico relacionando uma variável nova, Cholesterol combinada com MaxHR, com a variável original Oldpeak. Fonte: Elaborado pelo autor (2025). 34
3.15 Gráfico 3.14 dividido para cada classe. Fonte: Elaborado pelo autor (2025). . . . 34
3.16 Gráfico mostrando a distribuição de classes da variável nova resultada da combinação das variáveis MaxHR com Oldpeak. Em laranja os pacientes com problemas cardíacos, saudáveis em azul. Fonte: Elaborado pelo autor (2025). 35
3.17 Gráfico relacionando duas variáveis originais: Oldpeak com MaxHR. Fonte: Elaborado pelo autor (2025). 35
3.18 Gráfico 3.17 dividido para cada classe. Fonte: Elaborado pelo autor (2025). . . . 35

4.1 Curva de perda dos modelos treinados com a base do censo. Fonte: Elaborado pelo autor (2025). 45
4.2 Curva de perda dos modelos treinados com a base de doenças cardíacas. Fonte: Elaborado pelo autor (2025). 46
4.3 Boxplot das médias obtidas na validação cruzada na base de dados do Censo. Fonte: Elaborado pelo autor (2025). 50
4.4 Boxplot das médias obtidas na validação cruzada na base de dados de doenças cardíacas. Fonte: Elaborado pelo autor (2025). 51

Sumário

1 Introdução **1**

2 Revisão Teórica **3**

2.1 Aprendizagem de máquina 3

2.2 Multilayer Perceptron 4

2.2.1 Aprendizado na MLP 6

2.3 K-Nearest Neighbors 8

2.4 Support Vector Machine 9

2.4.1 Métodos de Kernel 10

2.5 Algoritmos Genéticos 11

2.6 Feature Engeneering 13

2.7 LIME (Local Interpretable Model-agnostic Explanations) 14

3 Metodologia **16**

3.1 Desenvolvimento da Coloração de Advertência 17

3.2 Bases de dados 21

3.2.1 Base do censo americano 22

3.2.2 Base de doenças cardíacas 24

3.3 Detalhamento da Coloração de Advertência 27

3.3.1 Função de transformação 27

3.3.2 Indivíduo 28

3.3.3 Algoritmo Genético 28

3.4 Gráficos 29

3.4.1 Base do censo 30

3.4.2 Base de doenças cardíacas 33

3.5 Utilização dos dados na MLP 36
3.6 Modelos de feature engeneering para comparação 37
3.6.1 Método Polynomial Features 37
3.6.2 Método Spline Transformer 37
3.6.3 Utilização do LIME para seleção de features 38

4 Resultados Obtidos **40**
4.1 Matriz de confusão 40
4.1.1 Base de dados do censo 40
4.1.2 Base de dados de doenças cardíacas 42
4.2 Curva de Perda 45
4.3 Validação Cruzada 46
4.3.1 Médias na Base do Censo 47
4.3.2 Médias na Base de Doenças Cardíacas 48
4.3.3 Comparação Entre Modelos Usando Boxplot 49
4.3.4 Tempo Gasto no Grid Search 51
4.4 Melhoria de Acurácia Obtida 52

5 Conclusão e trabalhos futuros **54**

Referências Bibliográficas **56**

Capítulo 1

Introdução

A utilização da inteligência artificial avança cada vez mais nos dias de hoje. Quando se trata de problemas que não possuem uma resposta bem resolvida, muitas vezes os algoritmos de aprendizagem de máquina são capazes de chegar a uma solução com precisão considerável. As IAs superam os seres humanos quando se trata de análise de grandes volumes de dados, jogos, simulações e recomendação de propagandas em conteúdos de streaming. São boas para a automação de tarefas, atendimento ao cliente e diagnóstico médico.

Outra vertente para a solução de problemas na computação é o ramo de computação bioinspirada: técnicas de programação que se inspiram em fenômenos comuns na natureza para criar algoritmos. Um exemplo são os algoritmos genéticos: algoritmos inspirados no processo de evolução biológica da seleção natural. Nessas abordagens, vários indivíduos são representados por possíveis soluções descritas por seus genes. Tais genes são valores que podem se combinar com genes de outros indivíduos a fim de gerar novas soluções.

Existe um fenômeno natural chamado de aposematismo ou coloração de advertência descrito por Poulton (1890) [4], onde animais desenvolvem cores vivas e padrões chamativos para sinalizar perigo para os possíveis predadores. Esse estudo tem a intenção de apresentar o desenvolvimento de uma nova técnica de otimização que, inspirada nesse fenômeno e nos algoritmos genéticos como os descritos por Goldberg (1989) [5], consegue uma melhoria no treinamento de uma rede neural, tanto no tempo de treinamento quanto na precisão final do modelo.

A nova técnica foi chamada de Coloração de Advertência e, para verificar a sua eficiência, os próximos capítulos mostram a aplicação dela em bases de dados. Verifica-se a diferença de resultados entre modelos de classificação com e sem a otimização. Além dos resultados, são analisados as características do modelo de forma a tentar entender quais dados podem ter tido

uma maior contribuição para o sucesso do algoritmo.

Capítulo 2

Revisão Teórica

Os algoritmos evolutivos e a aprendizagem de máquina são áreas bem abrangentes no campo da computação. Esta revisão teórica visa buscar os pontos mais importantes dessas áreas que contribuíram para o desenvolvimento do trabalho e auxiliam no seu entendimento.

2.1 Aprendizagem de máquina

A habilidade de obter conhecimento de dados para fazer previsões é um sub ramo da Inteligência Artificial conhecido como Aprendizagem de Máquina [1]. Ela nos permite criar modelos de situações e fazer com que a máquina crie seus parâmetros de previsão e os melhore gradualmente, tarefa que antes exigia que um ser humano descrevesse regras para a máquina a partir de uma análise manual dos dados.

Existem diferentes tipos de aprendizagem de máquina: supervisionado, não supervisionado e aprendizado por reforço. No aprendizado supervisionado, os dados vem rotulados e, a partir das características desses dados, o modelo tenta aprender o que caracteriza cada rótulo de forma a poder classificar ou prever resultados ao receber uma nova entrada não rotulada. O aprendizado não supervisionado não recebe dados rotulados, ele tenta fazer agrupamentos de acordo com os dados recebidos. Dessa forma ele é capaz de descobrir subgrupos nos dados mesmo sem ter um conhecimento prévio da existência deles. O aprendizado por reforço diferencia-se por interagir com o ambiente em que é submetido. Ao tomar determinada decisão, o agente vai receber um feedback do ambiente, como um sinal de recompensa. A tarefa do agente é tentar buscar um comportamento que maximize as recompensas. Nesse trabalho o foco é o aprendizado supervisionado: o modelo, através de um conjunto de dados classificados, busca ser capaz de

classificar uma nova entrada com um rótulo já existente no conjunto de treino.

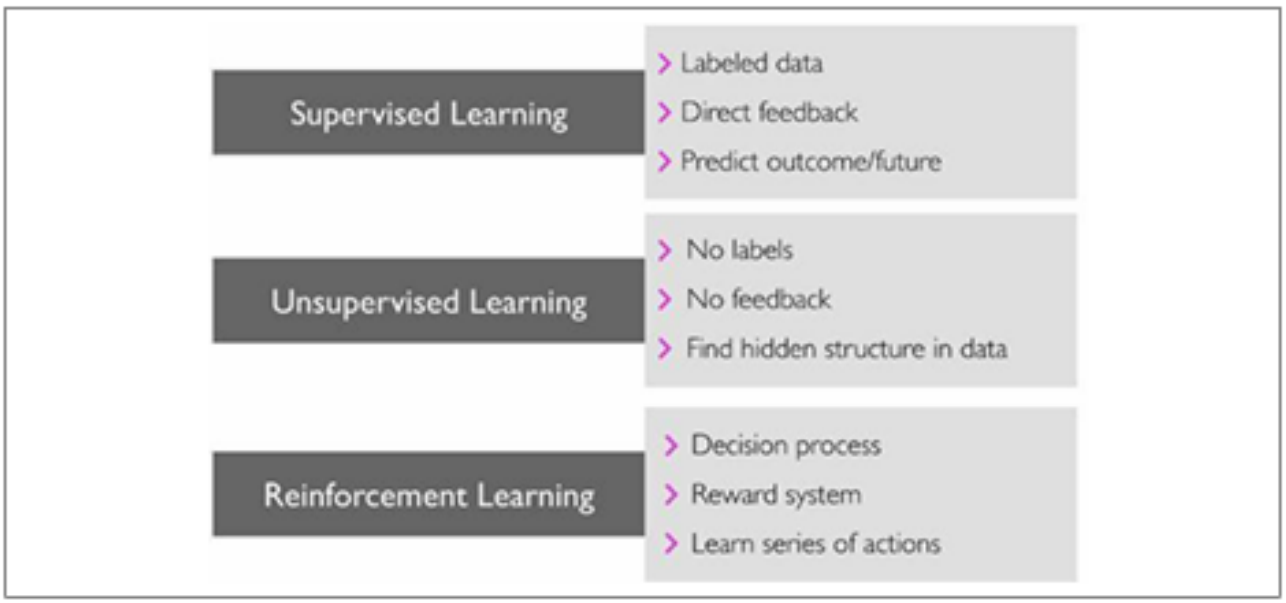


Figura 2.1: Três tipos de Machine Learning. Fonte: V. Raschka; Mirjalili (2019) [1].

2.2 Multilayer Perceptron

Inspirados na ideia dos neurônios biológicos, células do sistema nervoso que podem ser excitados por estímulos eletro-químicos e se transmitem informações entre elas, surge o perceptron: o neurônio artificial usado para construir redes neurais artificiais.[2]

Os perceptrons possuem entradas, que podem ser tanto as entradas de valores que descrevem o problema que a rede tenta resolver, quanto a saída de outros perceptrons. Cada entrada é associada à um peso, que descreve a contribuição daquela entrada no resultado final. A saída do neurônio é dada por uma função de ativação aplicada sobre o somatório dos produtos das entradas por seus respectivos pesos. Dessa forma, considerando as entradas como $a_1, a_2, \ldots, a_n$ e os pesos como $w_1, w_2, \ldots, w_n$, podemos ter uma função de ativação $f(x)$ que nos dá a saída o definida por:

$$o = f(\sum_{i=1}^{n} w_i * a_i) \tag{2.1}$$

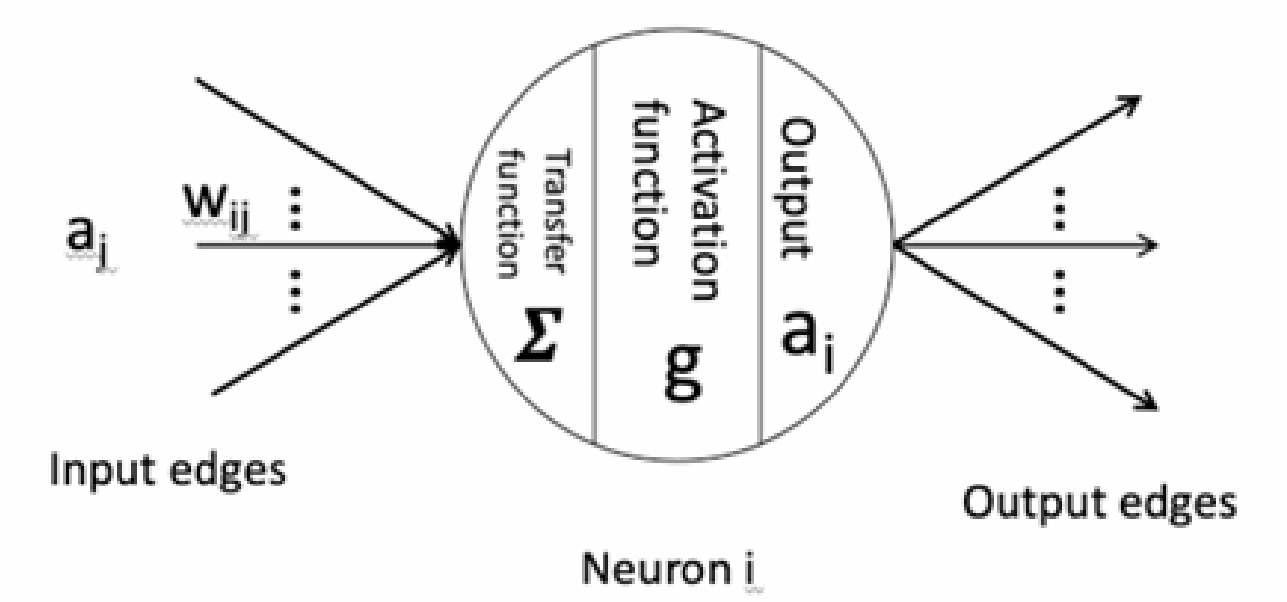


Figura 2.2: Estrutura de um neurônio artificial. Fonte: Kalita (2023) [2].

Os neurônios artificiais servem como base para estruturas mais sofisticadas conhecidas como redes neurais artificiais[2]. Ao invés de passar as informações como entrada para apenas um neurônio, utiliza-se camadas de neurônios interligados que vão passando as informações adiante até chegar na camada de saída. A informação chega inicialmente na camada de entrada que representa as características dos dados. Após a camada de entrada, podem vir camadas ocultas cujas quantidades podem ser definidas previamente, dessa forma, ocorre um maior processamento sobre os dados. Por fim vem a camada de saída cujos resultados podem servir para atribuir uma classe para a entrada inserida (classificação) ou um valor contínuo (regressão). Dessas múltiplas camadas surge a denominação Multilayer Perceptrons.

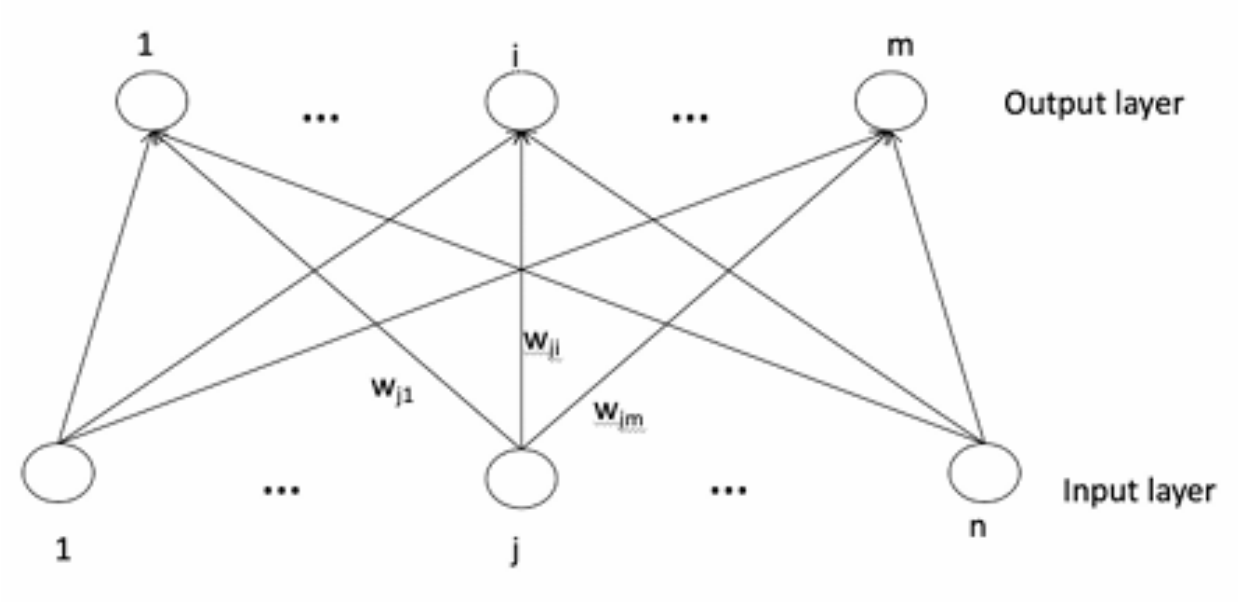


Figura 2.3: Rede com camada de entrada e saída. Fonte: Kalita (2023) [2].

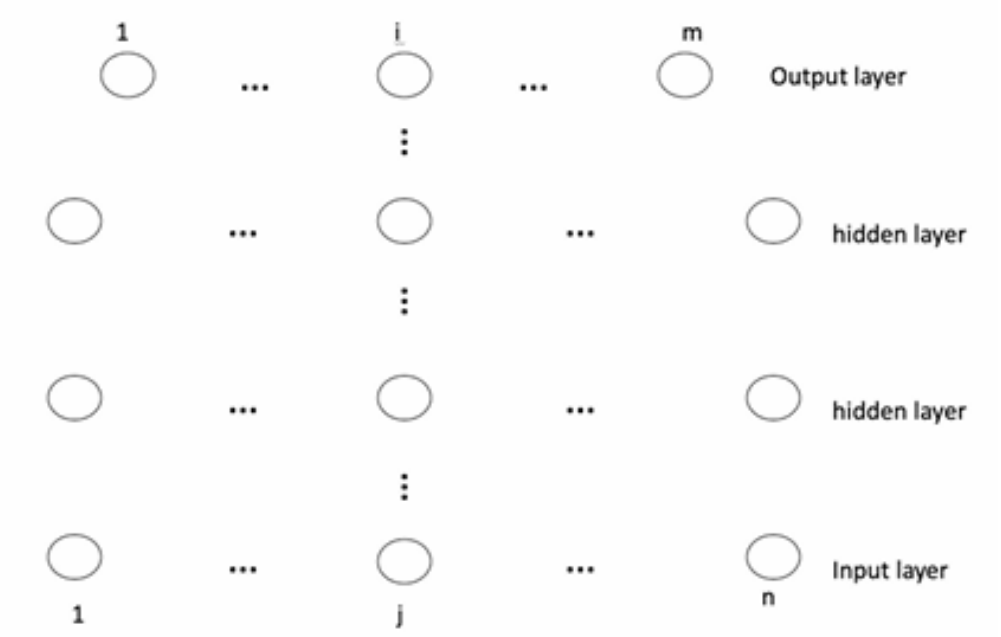


Figura 2.4: Representação com camadas ocultas. Fonte: Kalita (2023) [2].

2.2.1 Aprendizado na MLP

Como dito anteriormente, cada entrada dos neurônios possui um peso para a informação recebida. A função do aprendizado da rede neural é ajustar esses pesos de forma a possibilitar a classificação de novos exemplos. Esse ajuste é feito pelo algoritmo de Backpropagation que realiza os seguintes passos:

2.2.1.1 Feed Forward

A primeira coisa que acontece na MLP é o processo de Feed Forward. Esse nome é dado ao processo de passar a informação adiante, desde a camada de entrada até a de saída. É nele que cada neurônio recebe suas entradas, aplica os pesos em cada entrada, faz a soma e executa a função de ativação. Esse resultado é passado para o neurônio seguinte até chegar no fim da rede.

2.2.1.2 Loss Function

A Loss Function é a função usada para calcular o erro: Cada exemplo que entra na rede neural durante o treinamento possui uma resposta esperada, a loss function retorna um valor que indica o quanto a resposta obtida difere do esperado.

2.2.1.3 Backward Pass

Numa rede neural, tanto a função de ativação quanto a loss function não mudam com o passar do aprendizado e que a entrada vem de cada exemplo, a única coisa que pode ser alterada durante o aprendizado são os pesos das entradas, logo, os pesos são chamados de parâmetros da rede neural.

Visto que conseguir uma solução analítica seria algo muito complexo, pois uma rede neural pode ter milhares de pesos, a ideia é utilizar uma função que leve em conta os parâmetros. A partir dessa função pode-se usar técnicas como a descida do gradiente: baseada na ideia de que os mínimos locais tem a derivada igual a 0, a técnica visa mover os pesos para esses vales buscando uma solução onde o erro é mínimo. Esse processo é repetido desde a resposta na camada de saída, ajustando os pesos de cada neurônio até chegar na camada de entrada, daí vem o nome Backpropagation, ou propagação reversa [2].

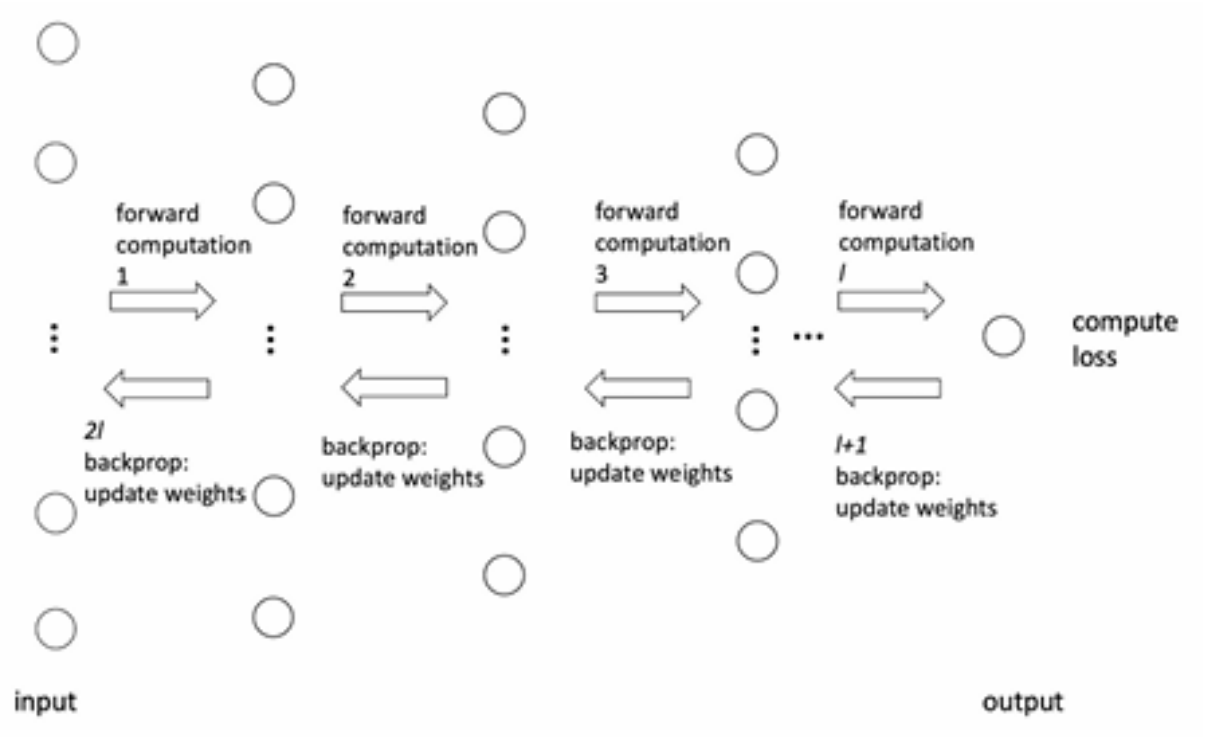


Figura 2.5: A figura mostra o Feed Forward nos passos 1 a l. Do l+1 em diante é mostrado o erro da saída sendo propagado do fim ao início atualizando os pesos da rede neural. Fonte: Kalita (2023) [2].

2.3 K-Nearest Neighbors

O K-Nearest Neighbors ou simplesmente KNN é um algoritmo simples que pode ser usado na tarefa de classificação. Enquanto outros algoritmos como redes neurais possuem um treinamento muito custoso, o KNN possui um método de aprendizagem bem simples e é classificado como uma abordagem de aprendizado preguiçoso [6], isto é, uma abordagem que espera até que a consulta seja feita para observar os dados de treinamento.

O algoritmo funciona encontrando os k elementos no conjunto de treino que mais se aproximam do elemento testado e atribui a ele a classe predominante nessa vizinhança. O valor de k e a forma de calcular a distância entre dois elementos podem ser escolhidos previamente. Sendo dois elementos $\mathbf{x}$ e $\mathbf{y}$ com n atributos, normalmente utiliza-se a distância Euclidiana (2.2) ou a distância de Manhattan (2.3).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.2}$$

$$d(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|} \tag{2.3}$$

Como o algoritmo só precisa salvar o conjunto de treino para fazer o cálculo da distância quando a consulta for requisitada e o tempo da consulta cresce linearmente com o tamanho do conjunto de treino, o KNN possui uma característica interessante para o desenvolvimento desse trabalho: um baixo custo temporal nas etapas de treinamento e de teste.

2.4 Support Vector Machine

O Support Vector Machine (SVM) tem uma idéia semelhante a do perceptron. Dado um elemento **x** (2.5) com **n** atributos, utiliza-se uma combinação linear com um vetor de pesos **w** (2.4) somado com um viés **b** para gerar um hiperplano que separe dados linearmente separáveis em duas classes (2.6).

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \tag{2.4}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.5}$$

$$z = b + \sum_{i=1}^{n} w_i * x_i \tag{2.6}$$

$$z = b + w^T x \tag{2.7}$$

Sobre o resultado de **z** aplica-se uma função $\phi(z)$ que retorna 1 se $z \geq 0$, representando uma classe, e -1 se $z < 0$, representando outra classe [1].

A ideia do SVM é que esse hiperplano gerado tenha sua margem maximizada, sendo essa margem representada pela distância do hiperplano para os pontos mais próximos de cada conjunto, os quais são chamados de vetores de suporte.

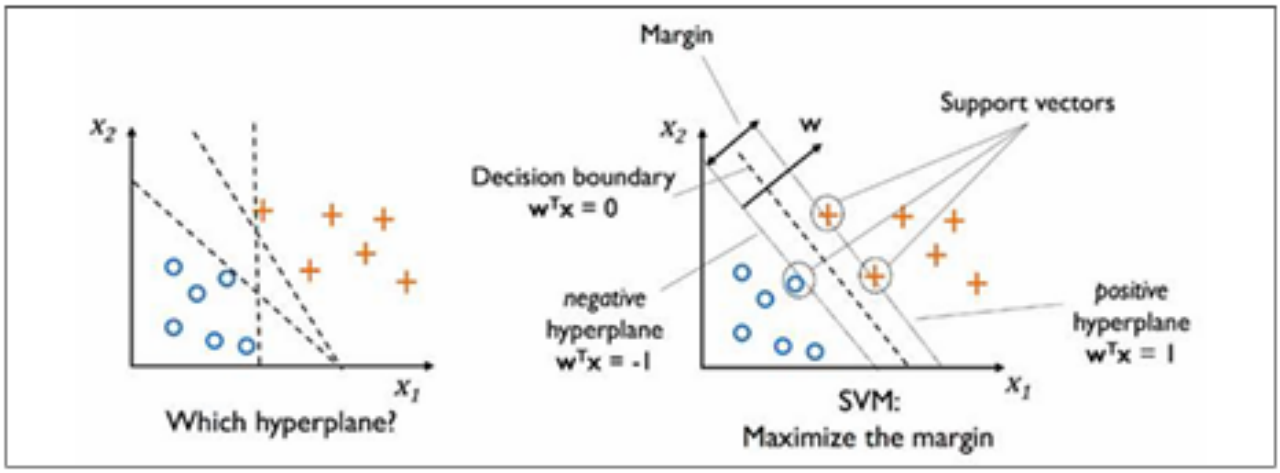


Figura 2.6: Hiperplano traçado pelo SVM. Fonte: V. Raschka; Mirjalili (2019) [1].

2.4.1 Métodos de Kernel

Em alguns casos, os dados estão distribuídos de forma que não é possível traçar um hiperplano que divida os dados em dois grupos. Para tratar esse tipo de caso, tenta-se aumentar a dimensionalidade do conjunto com o uso de um Kernel. A ideia do Kernel baseia-se numa transformação nos dados através de uma combinação não linear das características existentes na qual eles se transformem num conjunto linearmente separável. Apesar de ser possível construir uma nova característica em cima do problema, para evitar um grande custo computacional, o SVM utiliza a técnica de comparar similaridades entre os pontos com o Kernel, dessa forma ele apenas simula o mapeamento para um espaço separável sem construir o mapeamento de fato.

Supondo um dataset bidimensional não linearmente separável, é possível utilizar uma função (2.8) que adiciona uma terceira dimensão resultada da soma dos quadrados das dimensões anteriores. A imagem 2.7 mostra visualmente como a função torna os dados separáveis.

$$\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2) \tag{2.8}$$

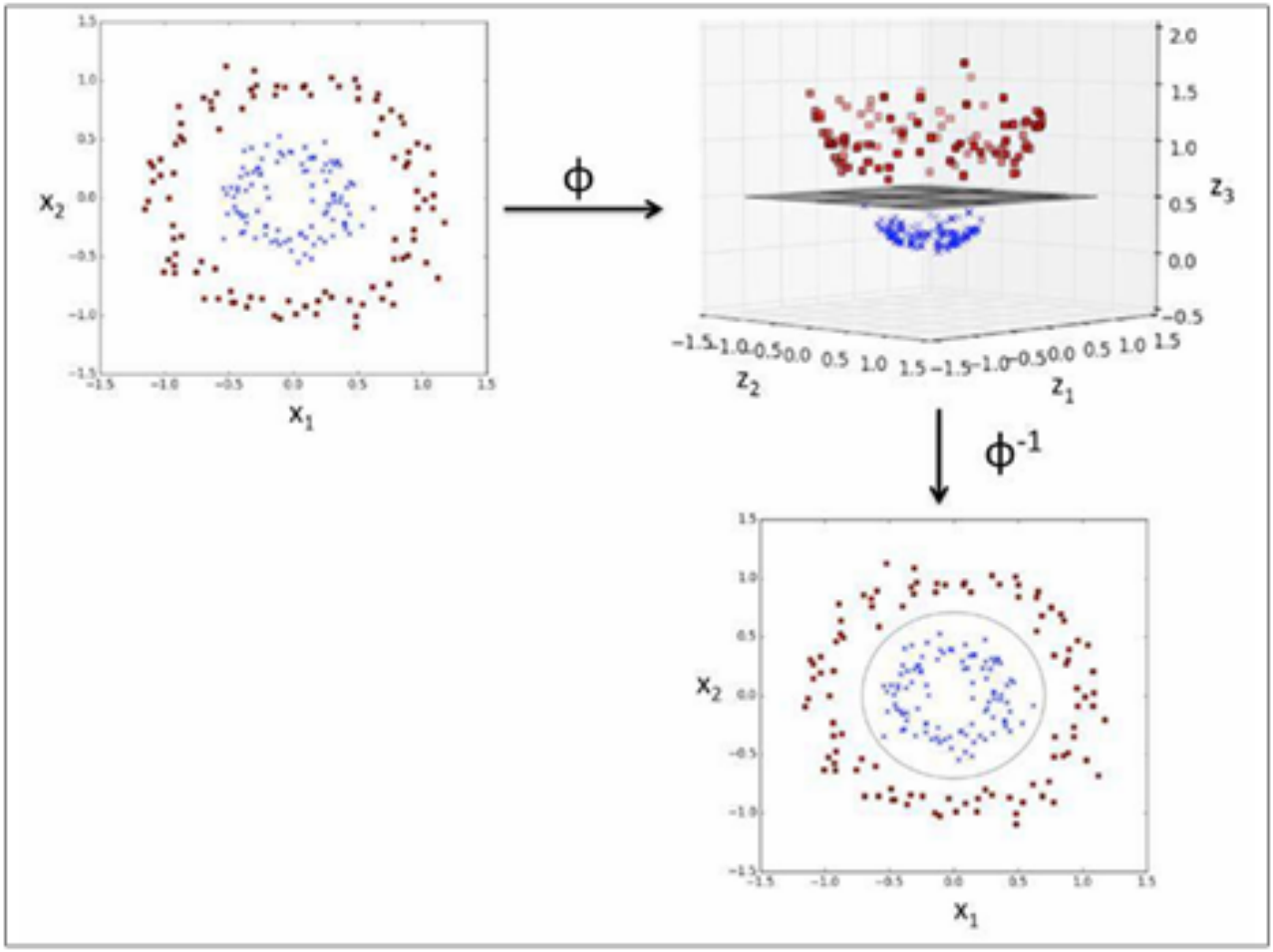


Figura 2.7: Transformação de um espaço bidimensional para tridimensional Fonte: V. Raschka; Mirjalili (2019) [1].

2.5 Algoritmos Genéticos

Baseados na ideia da evolução biológica, os algoritmos genéticos propõem um sistema artificial que busca otimizar uma solução criando um conjunto de indivíduos em que cada um representa uma possível solução para o problema. Esse grupo sofre transformações que tentam imitar processos como o da seleção natural, mutações e recombinações a fim de gerar o melhor indivíduo, ou seja, a solução que apresenta o melhor desempenho no problema [5].

O algoritmo segue os seguintes passos:

- **Gerar População**: É gerado um conjunto de indivíduos, normalmente de forma aleatória, modelados como possíveis soluções para o problema. Podem ser strings binárias, vetores de números reais, variando de acordo com o problema. Cada número do vetor pode ser visto como um gene do indivíduo.

- **Avaliação dos indivíduos**: Utiliza-se uma função de fitness para avaliar o desempenho de cada indivíduo na resolução do problema.

- **Seleção de sobreviventes**: Baseado no fitness de cada indivíduo, utiliza-se um critério de decisão que seleciona quais indivíduos continuarão a fazer parte da população, normalmente os que tem melhor fitness tendem a sobreviver, semelhante a ideia da seleção natural.

- **Recombinação de genes**: Nessa etapa novos indivíduos são gerados a partir da recombinação dos genes da população restante. Uma possível abordagem é escolher pares de indivíduos onde cada par gera um novo a partir da combinação dos genes deles.

- **Mutação**: Ocasionalmente adiciona-se um ruído nos indivíduos da população com o intuito de gerar novas características. Esse ruído pode ser enxergado como a troca aleatória, com uma pequena chance de acontecer, de um gene. Durante o processo evolutivo, é comum que as características do melhor indivíduo sejam propagadas para outros indivíduos da população, dessa forma o algoritmo tende a parar de evoluir, pois todos os genes tendem a ficar iguais. Ao realizar a mutação, torna-se possível alcançar novas soluções que não eram viáveis com o conjunto de genes inicial da população.

- **Substituição da população**: A população antiga é substituída pela nova e o algoritmo volta para o passo de avaliação repetindo o processo até atingir um critério de parada. O critério de parada pode ser o número de repetições, uma convergência ou um indivíduo ter chegado num valor de fitness desejado.

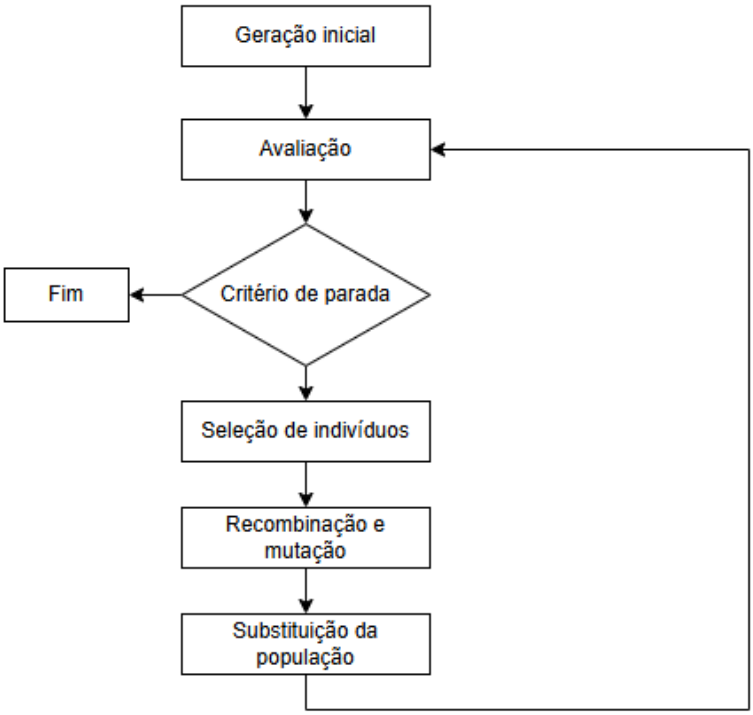


Figura 2.8: Fluxograma de um algoritmo evolutivo. Fonte: Elaborada pelo autor (2025).

A figura 2.8 mostra um fluxograma de um algoritmo genético. É possível fazer alterações no algoritmo, uma possibilidade é mudar a ordem: fazer a seleção de indivíduos após a recombinação. Outra alteração pode ser manter sempre alguns dos melhores indivíduos vivos durante a seleção.

2.6 Feature Engeneering

Para realizar o treinamento de um modelo de inteligência artificial, é muito importante possuir uma base de dados para treinamento de qualidade. Ao possuir uma boa quantidade de exemplos independentes nos dados, cada qual com uma boa correlação com a classe que ele representa, torna-se mais fácil para o modelo reconhecer as diferentes classes. Dependendo da forma que o conjunto de dados é apresentado, ele pode não ser muito amigável ao processo de aprendizado, contudo, pode-se construir novas características, baseadas nas originais, que ajudam nesse processo.

A ideia de gerar novas características chama-se Feature Engeneering. As novas características podem vir de transformações nas variáveis existentes, como no processo de padronização, que subtrai os valores de um campo e os divide pelo desvio padrão daquele campo, fazendo

com que a média dele vá para zero e o desvio padrão unitário vá para um. Ao fazer isso, o campo afetado fica todo na mesma escala, evitando que valores muito altos ou muito baixos atrapalhem a generalização do aprendizado. É possível também escolher entre as características do conjunto de treino, quais são mais importantes, bem como criar novas variáveis combinando as que já existem. Uma forma de fazer isso é gerar uma nova variável que representa a razão entre duas outras.

Utilizar dessa ideia, pode trazer algumas desvantagens: gerar features em excesso pode gerar overfitting, além de aumentar o custo computacional, além de que, durante a geração de novas variáveis, um conhecimento prévio sobre o problema pode fazer uma grande diferença geração de variáveis significativas.

Para este trabalho, utilizou-se métodos de geração de novas features presentes na biblioteca scikit-learn [7] com a ideia de comparar os resultados dessas técnicas com a Coloração de Advertência.

2.7 LIME (Local Interpretable Model-agnostic Explanations)

Os modelos de previsões de Machine Learning, muitas vezes tem um aprendizado complexo e difícil de ser interpretado, como nas redes neurais. Normalmente esses algoritmos funcionam como uma caixa-preta, onde o modelo consegue fazer previsões sobre os dados sem deixar claro uma justificativa de quais características contribuíram mais para a previsão.

O LIME tem a missão de, a partir de uma amostra original dos dados e da previsão do modelo sobre ele, elaborar uma explicação sobre quais características foram mais relevantes para aquela previsão específica. É importante lembrar que essa explicação é válida apenas para aquela amostra.

Para obter a explicação, o algoritmo gera amostras a partir da instância a ser observada alterando campos da amostra original. Em seguida atribui-se pesos a essas amostras através de uma função que calcule a proximidade de cada uma em relação a original. Usa-se então o modelo caixa-preta, isto é, o modelo sobre o qual tenta-se obter a explicação, para classificar cada um dessas amostras artificiais. Com as amostras classificadas, é treinado outro modelo sobre elas, dessa vez um modelo interpretável, como árvore de decisão ou regressão linear. Dessa forma o novo modelo treinado pode ser usado como uma base de explicação para a amostra

original. A figura 2.9 mostra uma imagem do passo a passo do algoritmo.

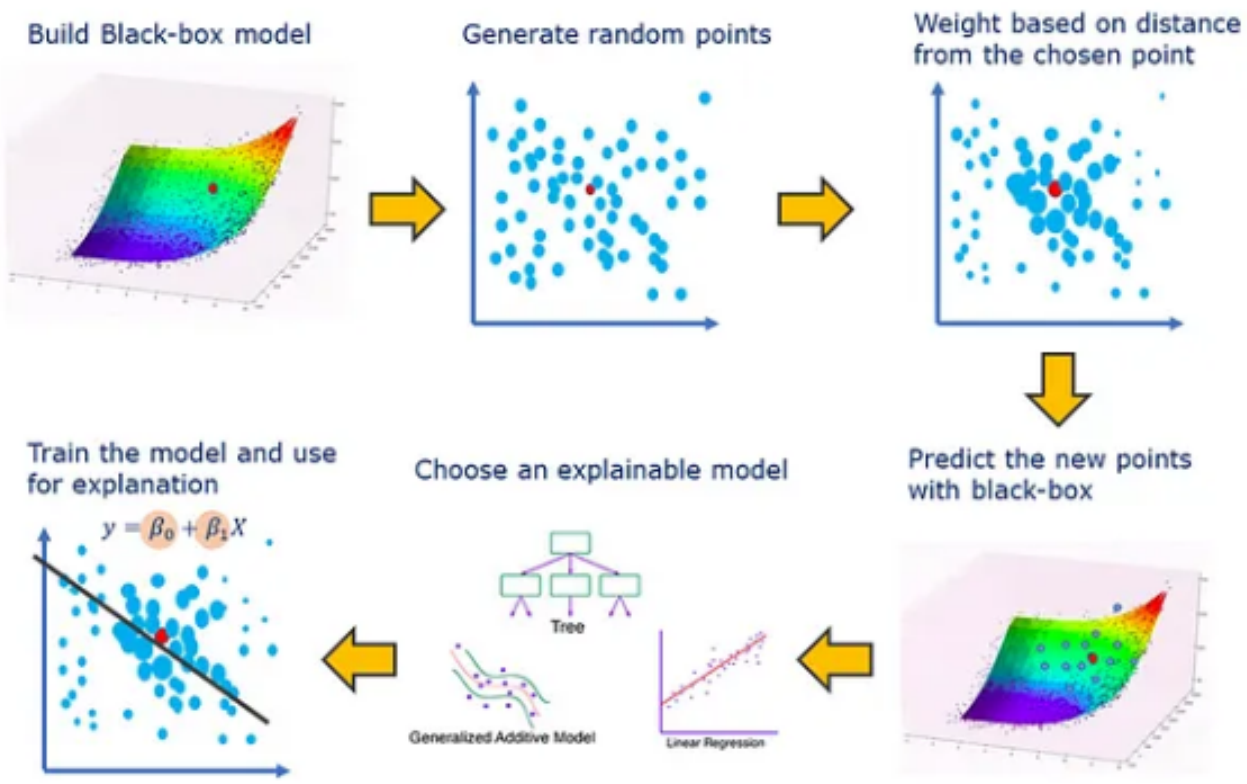


Figura 2.9: Passo a passo do algoritmo LIME. Fonte: Visani (2020) [3]

Com a utilização do LIME é possível extrair quais características do dataset mais influenciaram na previsão daquela amostra. Usando a ideia de que pode-se chegar em uma explicação mais geral observando várias explicações locais, como é defendido no artigo do algoritmo SHAP (SHapley Additive exPlanations) [8], outro algoritmo de explicação de modelos. Para este trabalho, foi escolhido o algoritmo LIME para selecionar as features mais importantes de cada base de dados. Essas features foram combinadas e sofreram transformações para gerar novas características. Essa geração de novas características é uma forma de Feature Engeneering usada neste trabalho para comparar essa abordagem com a abordagem da Coloração de Advertência proposta.

Capítulo 3

Metodologia

O trabalho visa mostrar o desenvolvimento de uma nova técnica de otimização chamada de Coloração de Advertência, além de observar os resultados da sua aplicação. Para verificar seu desempenho, analisou-se um conjunto de dados já existente e observou-se o desempenho do algoritmo de aprendizagem de máquina MLP da biblioteca scikit-learn da linguagem Python antes e depois da otimização. O ambiente usado no trabalho foi o jupyter notebook.

A ideia da técnica baseia-se no truque de kernel utilizado no SVM em conjunto com a ideia de feature engineering de gerar novos campos nos dados de treino e com os algoritmos genéticos. Como mencionado anteriormente, a ideia por trás da kernelização é aumentar a dimensionalidade do problema a partir de transformações envolvendo as características originais dos dados. Visto que essas transformações podem gerar uma característica que era implícita ao problema inicial, o algoritmo proposto tem a missão de fazer algo semelhante aplicando uma transformação relacionando pares de características do problema inicial. Como não é tão simples descobrir de que forma essas características se relacionam ou qual seria a melhor transformação a se aplicar, utiliza-se uma transformação com vários parâmetros que são melhorados através do uso de um algoritmo genético. Em suma, o que a Coloração de Advertência faz é receber uma bases de dados de um problema e devolver uma nova base de dados que seja mais fácil de executar um treinamento de uma inteligência artificial sobre ela.

Para entender melhor como funciona a Coloração de Advertência, faz-se importante ressaltar algumas características do fenômeno natural de mesmo nome. Na natureza, os indivíduos mais adaptados sobrevivem e passam suas características para os descendentes. As espécies que desenvolveram o aposematismo possuem características chamativas que indicam perigo aos predadores, sinalizando que tem um sabor ruim ou são venenosas. Dessa forma, os predadores

evitam esses indivíduos e eles tem mais chance de sobrevivência. Essas características são traduzidos para o desenvolvimento da técnica da seguinte forma:

- A população vai ser formada por indivíduos com as seguintes características: cada um possui um conjunto de parâmetros que serão usados para transformar o dataset original em outro. Esse conjunto de parâmetros representa os genes do indivíduo.
- A transformação do dataset funciona como se fosse as cores geradas na pele do indivíduo pelos seus genes. Se esse dataset transformado ajudar mais no aprendizado que o dataset normal, é como dizer que aquele indivíduo gerou cores mais marcantes e fáceis de identificar.
- Os indivíduos que gerarem piores datasets serão eliminados, como na natureza são eliminados os indivíduos com características menos vantajosas.
- Ao final de tudo, o melhor indivíduo é usado para gerar um novo dataset a partir do original e esse novo dataset será usado para treinar um modelo que resolve o problema em questão.

Vale ressaltar que a Coloração de Advertência não busca uma resolução o problema em si, e sim melhorar o resultado de outros algoritmos. De forma semelhante, as cores vibrantes não são o real perigo na natureza, mas indicam o real perigo para os predadores.

3.1 Desenvolvimento da Coloração de Advertência

Para aplicar a técnica da Coloração de Advertência, é necessário seguir alguns passos, como mostrado na figura 3.1:

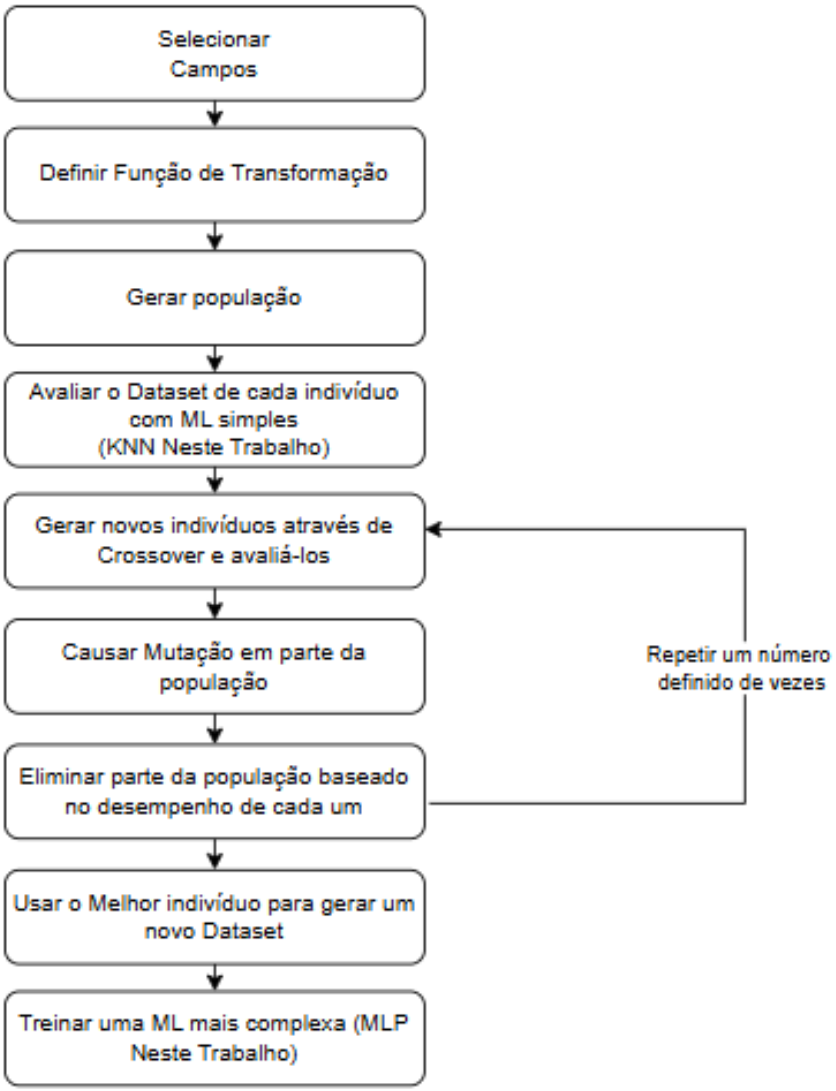


Figura 3.1: Esquema que mostra o fluxo da técnica de Coloração de Advertência. Fonte: Elaborado pelo autor (2025).

- **Escolher campos**: Escolher quais campos da base de dados serão usados para gerar novos campos. Para este trabalho foram escolhidos todos os dados numéricos das bases analisadas.

- **Função de transformação**: Deve-se definir uma função de transformação, que, a partir de dois ou mais campos originais, gere um campo novo com alguma transformação não linear. É importante que essa função tenha parâmetros a serem ajustados, como constantes multiplicativas ou expoentes. Esses parâmetros serão melhorados por um algoritmo evolutivo.

- **Algoritmo Genético**: Na figura 3.1, o algoritmo genético é representado nos passos 3

a 7. Usa-se um algoritmo genético para melhorar os parâmetros. Cada indivíduo indica um conjunto de parâmetros da função de transformação. Esses parâmetros são usados para gerar uma nova base de dados do problema original em cada indivíduo. Constroi-se, para cada indivíduo, um classificador mais simples, como o KNN, que resolve o problema original, mas agora usando seu dataset transformado. O fitness é calculado de acordo com a acurácia desse modelo treinado a partir do novo dataset gerado por aquele indivíduo. A eficiência de cada indivíduo pode ser vista como o destaque que ele causa na base de dados quando a transforma.

- **Gerar nova base de dados**: O melhor indivíduo da evolução é entendido como o indivíduo que mais evidencia as características dos dados, análogo a ter a cor mais chamativa na natureza. Usa-se esse indivíduo para gerar uma nova base de dados com a função de transformação.

- **Treinar modelo**: Com a nova base de dados, um novo classificador é treinado, usando um algoritmo mais complexo do que o utilizado no calculo de fitness da analise evolutiva.

Observa-se que o método da Coloração de Advertência tem os algoritmos genéticos como principal base. O diferencial da técnica está principalmente no fato de que, para transformar a base de dados, escolhe-se a função de transformação que transforma os dados e na avaliação do fitness é feito através de modelos previsores que tentam solucionar o problema original. Dessa forma ao invés de encontrar uma solução para o problema, a Coloração de Advertência procura os melhores parâmetros possíveis para usar na função de transformação e a ideia de cálculo do fitness da um viés para encontrar esses parâmetros.

O algoritmo 1 mostra a função de transformação usada nesse trabalho. Ao implementar a técnica, pode-se escolher outras funções no lugar, desde que possua parâmetros que podem ser melhorados.

Algorithm 1: Função De Transformação (Fonte: elaborado pelo autor (2025))

Input: real: x, real: y, tupla de reais: (i, j, a, b)

Output: Novo campo na base de dados

```
1 dx ← x - i;
2 dy ← y - j;
3 return a*(dx*dx) + b*(dy*dy);
```

4 O algoritmo 2 mostra como foi aplicada a função de transformação para gerar um novo dataset.

Algorithm 2: Aplicar Transformação (Fonte: elaborado pelo autor (2025))

Input: matriz: dataset, vetor de tuplas: indices, vetor de tuplas: genes

Output: Nova base de dados

```
1  //indices são os campos escolhidos do dataset para fazer as combinações e gerar novos
     campos
2  //genes são os parâmetros da função de transformação que vem do indivíduo
3  novoDataset ← dataset;
4  for i ← 0 to dataset.tamanho() - 1 do
5      l ← 0;
6      for j ← 0 to indices.tamanho() - 1 do
7          for k ← j+1 to indices.tamanho() - 1 do
8              p1 ← dataset[i][indices[j]];
9              p2 ← dataset[i][indices[k]];
10             novoDataset[i].append( Função De Transformação(p1, p2, genes[l]));
11             l ← l + 1;
12 return novoDataset;
```

O algoritmo 3 mostra a aplicação do algoritmo genético para mehlorar os parâmetros. Nele é possível notar como o KNN pode ser usado para avaliar a transformação que cada indivíduo faz no dataset.

Algorithm 3: Algoritmo Genético (Fonte: elaborado pelo autor (2025))

Input: matriz: dataset, vetor: resultados, vetor: indices

Output:

```
1  //dataset é o dataset original do problema, resultados são as classificações verdadeiras
     de cada linha do dataset
2  população ← inicializar();
3  //Avaliar população
4  for i ← 0 to população.tamanho() - 1 do
5      novoDataset ← Aplicar Transformação(dataset, indices, população[i].genes);
6      KNN ← treinarKNN(novoDataset, resultados);
7      população[i].fitness ← KNN.acurácia();
8  //Processo Evolutivo
9  for i ← 0 to numero de gerações do
10     pais ← selecionar pais();
11     filhos ← crossover(pais);
12     população.append(filhos);
13     população.mutação();
14     //Reavaliar
15     for i ← 0 to população.tamanho() - 1 do
16         novoDataset ← Aplicar Transformação(dataset, indices, população[i].genes);
17         KNN ← treinarKNN(novoDataset, resultados);
18         população[i].fitness ← KNN.acurácia();
19     população.eliminar();
20 return população.melhorIndivíduo();
```

Nas próximas seções serão dados mais detalhes sobre como variou a implementação para cada base de dados usada.

3.2 Bases de dados

Para o experimento foram utilizadas duas base de dados. Uma é do censo americano [9] que possui informações como idade, trabalho, educação e mostra a arrecadação anual. A outra é uma combinação de bases de dados de vários lugares sobre doenças cardíacas [10] contendo

informações como idade, pressão sanguínea, colesterol, informações sobre dores torácicas e mostra a incidência de doenças cardíacas.

3.2.1 Base do censo americano

O conjunto possui 32561 linhas, cada uma representando uma pessoa e 15 colunas representando as características das pessoas.

- **age**: Idade de cada indivíduo.
- **workclass**: O tipo de trabalho exercido.
- **Final Weight**: Fator que indica quantas pessoas a amostra em questão representa.
- **education**: Nível educacional da pessoa.
- **education-num**: Número de anos de educação completa.
- **marital-status**: Estado civil.
- **occupation**: Trabalho exercido.
- **relationship**: Status de relacionamento.
- **race**: Raça do indivíduo.
- **sex**: Sexo do indivíduo.
- **capital-gain**: Lucro obtido ao vender ativos por um preço superior ao de compra.
- **capital-loss**: Prejuízo obtido ao vender ativos por um preço inferior ao de compra.
- **hours-per-week**: Número de horas trabalhadas por semana.
- **native-country**: País de origem.
- **income**: Renda anual do indivíduo (maior ou menor que 50000).

O atributo income foi escolhido a ser previsto pelo classificador. O income caractriza duas classes: Os que ganham cinquenta mil ou menos por ano e os que recebem mais. Usando a biblioteca matplotlib, foi gerado um gráfico 3.2 com o percentual de cada classe.

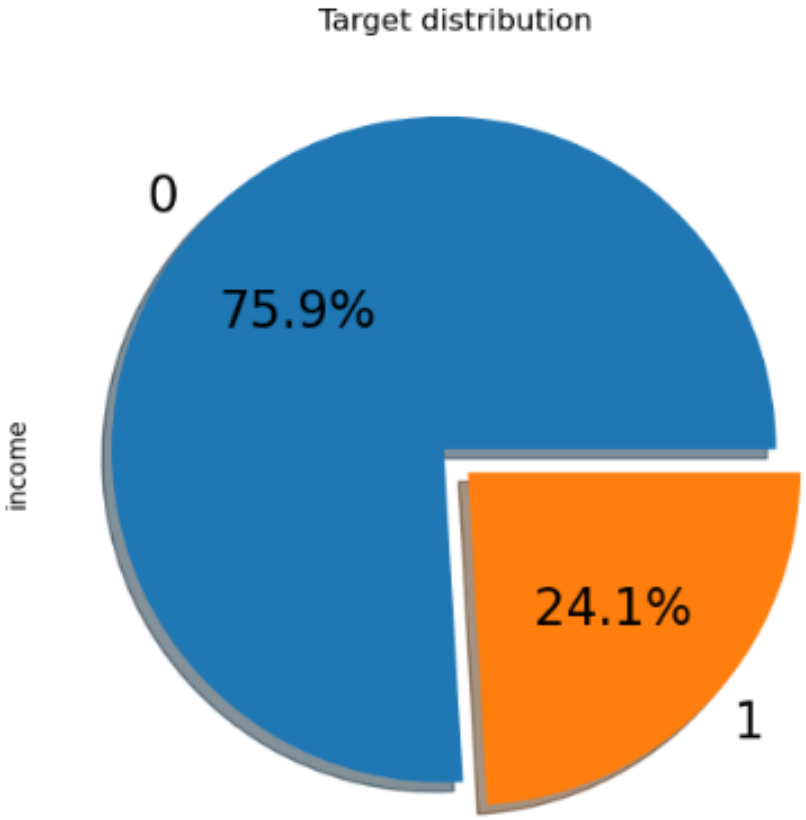


Figura 3.2: Percentual de pessoas que recebem 50 mil ou menos em azul e pessoas que recebem mais de 50 mil em laranja. Fonte: Elaborado pelo autor (2025).

Como algoritmo proposto visa evidenciar características implícitas no dataset, foi verificado a relação existente entre os atributos numéricos a partir de uma matriz de correlação 3.3.

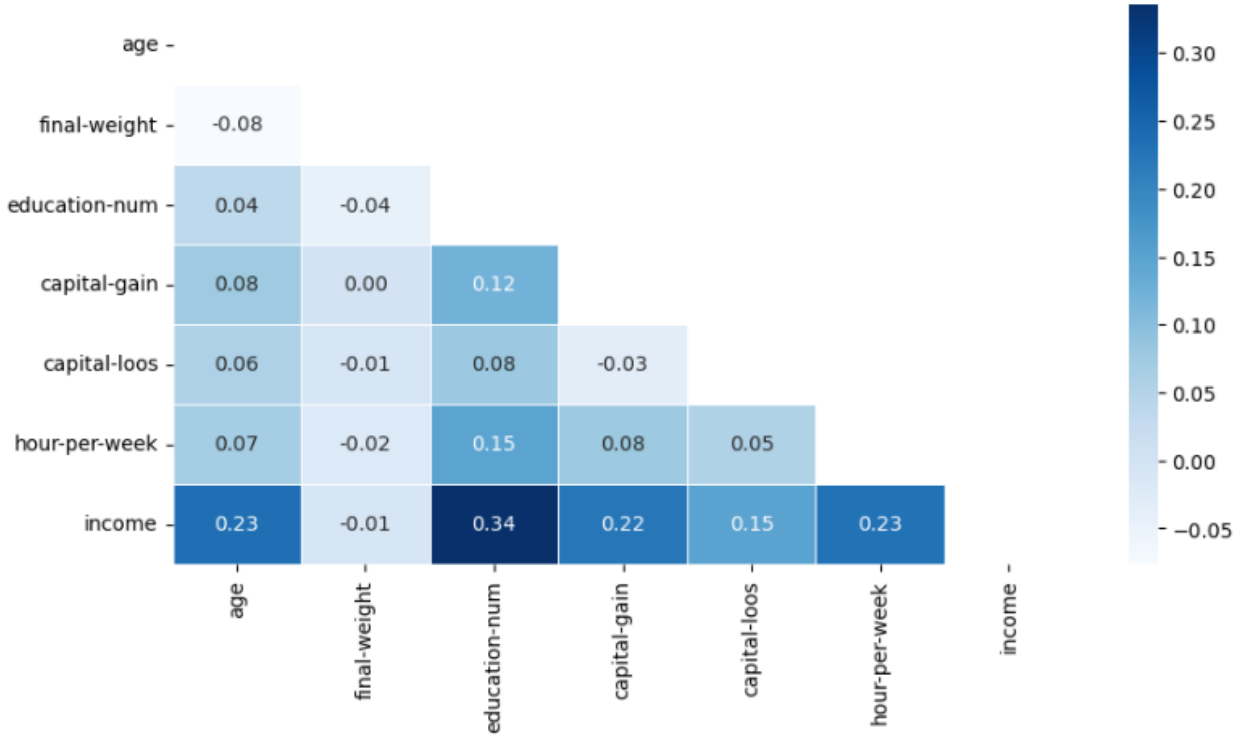


Figura 3.3: Matriz de correlação entre as variáveis numéricas. Fonte: Elaborado pelo autor (2025).

Em seguida, foi realizada a padronização dos dados com a biblioteca sklearn [7] no intuito de deixar os dados na mesma escala para que a diferença de escala não atrapalhe durante o aprendizado. Após a padronização, foi o conjunto foi separado em um grupo de treino equivalente a 80% do tamanho original e um grupo de teste com 20% do tamanho.

3.2.2 Base de doenças cardíacas

O conjunto possui 918 linhas, cada uma representando uma pessoa e 12 colunas representando as características das pessoas.

- **age**: Idade de cada indivíduo.
- **sex**: Sexo do indivíduo.
- **ChestPainType**: O tipo de dor torácica que o indivíduo possui.
- **RestingBP**: Pressão arterial em repouso: pressão exercida nas paredes das artérias enquanto o coração está em repouso entre os batimentos.
- **Cholesterol**: Colesterol presente no sangue.

- **FastingBS**: Nível de glicemia.
- **RestingECG**: Resultado de um eletrocardiograma em repouso.
- **MaxHR**: Maior frequência que o coração pode atingir durante um esforço.
- **ExerciseAngina**: Angina causada por esforço, diz se o indivíduo sente dores no peito ao realizar exercícios.
- **Oldpeak**: Depressão ST induzida por exercício relativamente sossegado. O ST refere-se a um seguimento presente no exame de eletrocardiograma.
- **ST_Slope**: Inclinação da extremidade do segmento ST no exercício.
- **HeartDisease**: Campo que diz se o indivíduo possui ou não doença cardíaca.

O atributo HeartDisease foi escolhido a ser previsto pelo classificador. Sendo o valor 0 usado para representar a ausência de problemas cardíacos e o 1 a existência, foi gerado um gráfico 3.4 com o percentual de cada classe semelhante ao da base do censo.

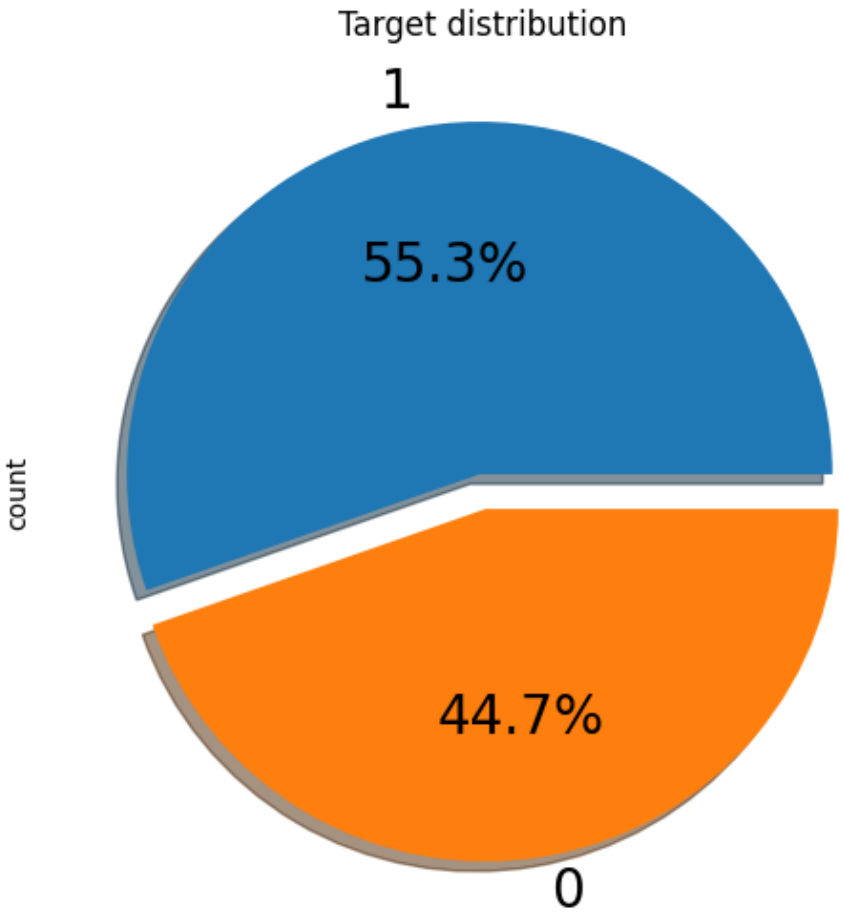


Figura 3.4: Percentual de pessoas que possuem doenças cardíacas em azul e que não possuem em laranja. Fonte: Elaborado pelo autor (2025).

Foi gerado também uma matriz de correlação para observar como os atributos se relacionam nessa base de dados 3.5.

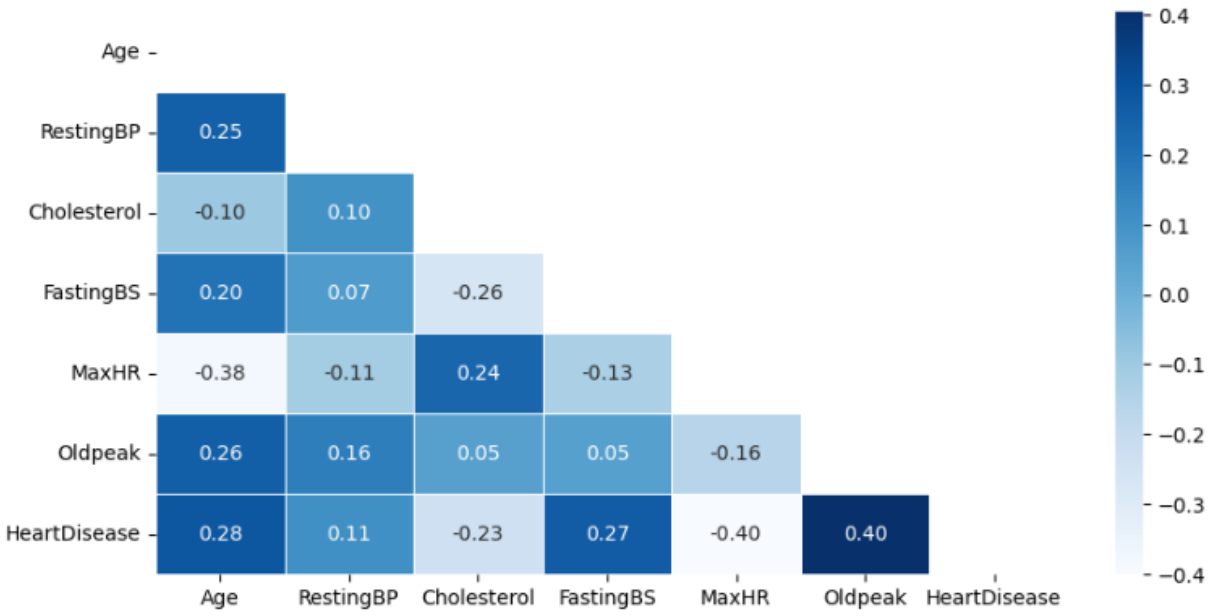


Figura 3.5: Enter Caption

Assim como na base de dados do Censo, também foi feita a padronização dos dados e a separação em grupo de treino de 80% do conjunto original e teste de 20%.

3.3 Detalhamento da Coloração de Advertência

Para a melhor compreensão, adotou-se a palavra **elemento** para referenciar uma linha do conjunto de dados e a palavra **indivíduo** para representar os indivíduos da população gerada pelo Algoritmo Genético.

3.3.1 Função de transformação

A função escolhida para gerar novos valores funciona da seguinte forma:

- **Parâmetros**: Recebe **x**, **y** e **gama**. As variáveis **x** e **y** são as duas características do dataset original que gerarão uma nova. O **gama**: É o gene que vem do indivíduo e parametriza a transformação. Ele é representado por uma tupla (i, j, a, b), sendo que i pode variar entre o menor e o maior valor existente na base de dados da variável **x**, enquanto j pode variar entre o menor e maior valor existente da variável **y**. a e b são valores intensificadores das operações realizadas variando entre 0 e 1.

- **Execução**: Calcula-se $dx = x - i$ e $dy = y - j$. A função retorna $a * dx^2 + b * dy^2$.

As variáveis i e j representam pontos os quais se calcula a distância das características originais até esses pontos. Essas distâncias são elevadas ao quadrado para aumentar o efeito de distâncias maiores e multiplicadas pelos fatores a e b que representam a relevância daquela diferença.

3.3.2 Indivíduo

Cada indivíduo foi modelado para ser um conjunto de genes com a seguinte configuração:

- **genes**:No caso da base de dados do censo, o indivíduo é um vetor de 15 tuplas (i, j, a, b), cada uma representando os parâmetros da função de transformação. Cada tupla é aplicada em uma combinação diferente de 2 dos 6 campos numéricos dos elementos, por exemplo: a primeira tupla é usada para combinar os campos age com final-weight, a segunda para combinar os campos age com education-num e assim por diante até formar todas as 15 combinações possíveis. Na base de dados de doenças cardíacas foi adotada a mesma configuração, porém com apenas 10 tuplas e 5 campos numéricos.

- **Cálculo de fitness**: Aplica-se transformações de acordo com os genes do indivíduo em cada elemento do conjunto gerando novas colunas. Utiliza-se algum método de Inteligência Artificial para construir um classificador sobre o novo conjunto. O resultado do fitness é igual a acurácia desse classificador. Visto que essa operação será calculada para vários indivíduos, foi escolhido o algoritmo KNN. O KNN era apropriado para essa tarefa por não exigir treinamento além de que, pela natureza do KNN, caso as novas características posicionem os dados de um mesmo grupo em regiões próximas, o KNN terá uma melhor precisão.

3.3.3 Algoritmo Genético

O algoritmo foi executado com uma população de **i** indivíduos com **g** gerações, isto é, repetindo o laço do algoritmo **g** vezes. Primeiro foi criado uma população de indivíduos com os genes criados aleatoriamente respeitando os limites de valores que cada elemento da tupla pode assumir. Para a base de dados do censo, o valor de **i** escolhido foi 50 e **g** 150. No caso da base das doenças cardíacas, o valor de **i** também foi 50, no entanto o **g** foi de 1000, devido ao fato de que essa base de dados é menor. A avaliação do fitness de cada indivíduo foi feita

assim que o indivíduo era gerado. Após gerar a população, o algoritmo seguiu a seguinte ordem repetindo-se por **g** iterações:

1. **Seleção de pais**: A população é embaralhada e divida em janelas de 5 indivíduos. Em cada uma dessas janelas, os dois melhores indivíduos eram selecionados para gerar um novo indivíduo.

2. **Recombinação de genes**: Gera-se novos indivíduos a partir das duplas de pais. Cada gene dos novos indivíduos vem de um dos dois pais tendo 50% de chance de vir do primeiro e 50% de chance de vir do segundo.

3. **Mutação**: Cada indivíduo tem 35% de chance de iniciar o processo de mutação. Apesar de ser uma chance alta, ao iniciar o processo de mutação, cada gene tem apenas 12% de chance de ser substituído por outro aleatoriamente. Dessa forma, a probabilidade total de ocorrer uma mutação, para cada gene, é de $0.35 * 0.12 = 0.042$, ou seja 4.2%.

4. **Seleção de sobreviventes**: 20% da nova população é composta pelos melhores indivíduos existentes na população atual, incluindo os que foram gerados na iteração atual. O restante dos sobreviventes são ordenados em ordem decrescente de acordo com o valor de fitness, contudo, é aplicado uma taxa de 5% de chance de ocorrer um evento caótico: ao invés de usar o valor de fitness real para definir a posição do indivíduo, usa-se esse fitness multiplicado por um número real aleatório entre 0 e 1. Esse caos tem o intuito de fazer com que alguns indivíduos que sobreviveriam sejam jogados para as últimas posições e assim dar chance para indivíduos com um baixo fitness, que podem ter alguns genes bons, sobrevivam.

Ao final do Algoritmo Genético, usou-se os genes do melhor indivíduo para modificar o conjunto de dados com a função de transformação.

3.4 Gráficos

Foram gerados gráficos para observar a distribuição das classes de acordo com os pares de características. Observou-se principalmente alguns gráficos onde a diferença das classes era mais notável.

3.4.1 Base do censo

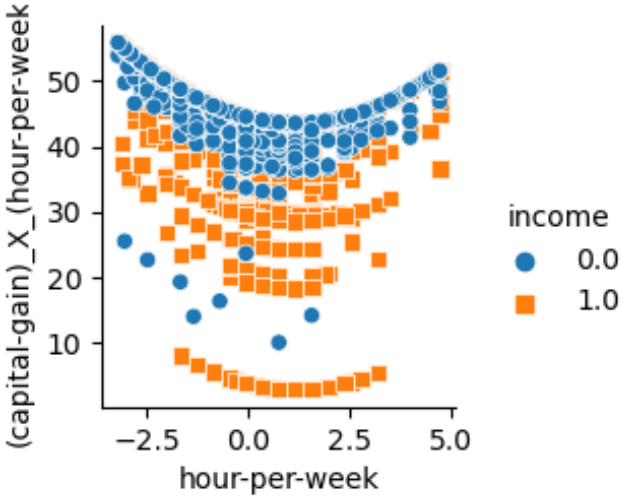
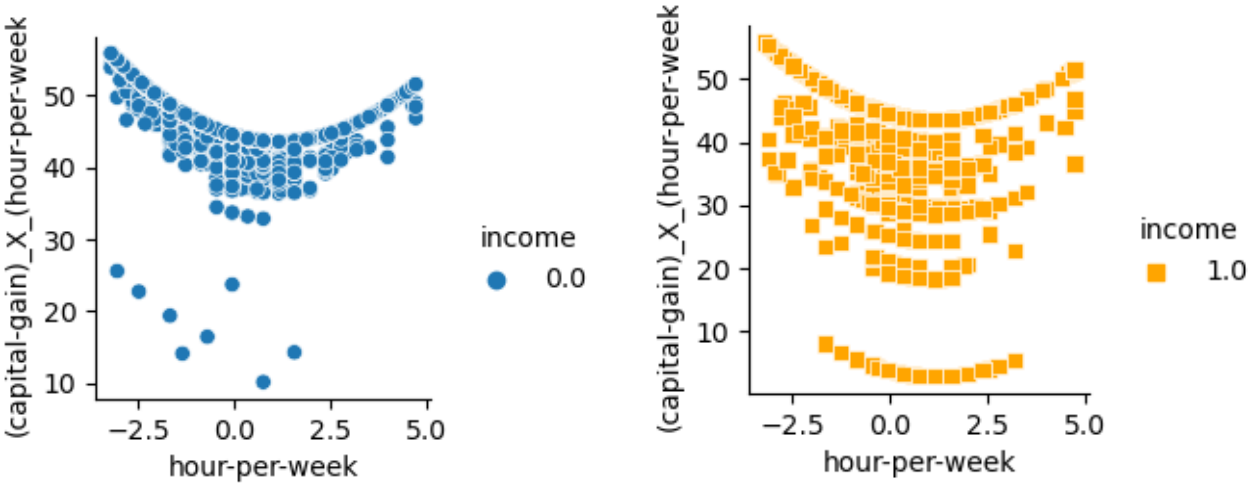


Figura 3.6: Gráfico relacionando uma variável nova, resultado da aplicação da função de transformação sobre capital-gain e hour-per-week, com a variável dos dados originais hour-per-week. Fonte: Elaborado pelo autor (2025).



(a) Classe 0, recebe 50 mil ou menos.

(b) Classe 1, recebe mais de 50 mil.

Figura 3.7: Gráfico 3.6 dividido para cada classe. Fonte: Elaborado pelo autor (2025).

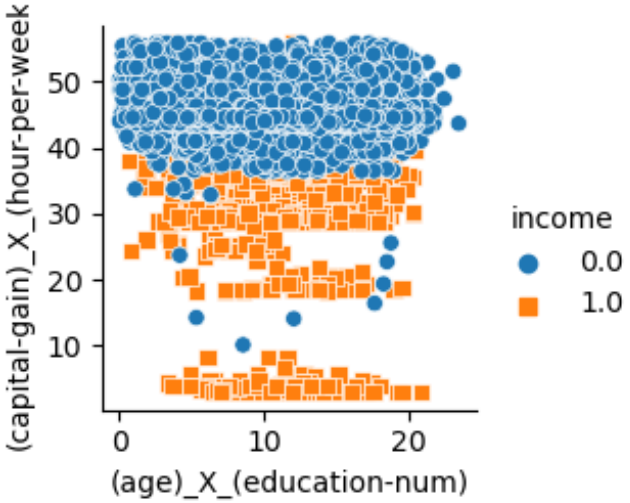
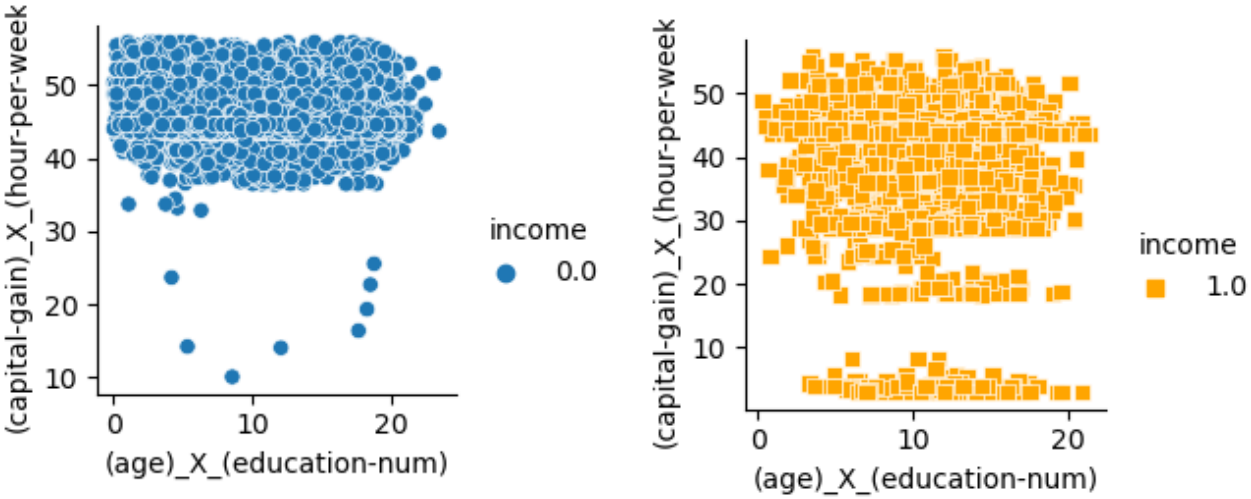


Figura 3.8: Gráfico relacionando duas variáveis novas, capital-gain combinada com hour-per-week e age combinada com education-num. Fonte: Elaborado pelo autor (2025).



(a) Classe 0, recebe 50 mil ou menos.

(b) Classe 1, recebe mais de 50 mil.

Figura 3.9: Gráfico 3.8 dividido para cada classe. Fonte: Elaborado pelo autor (2025).

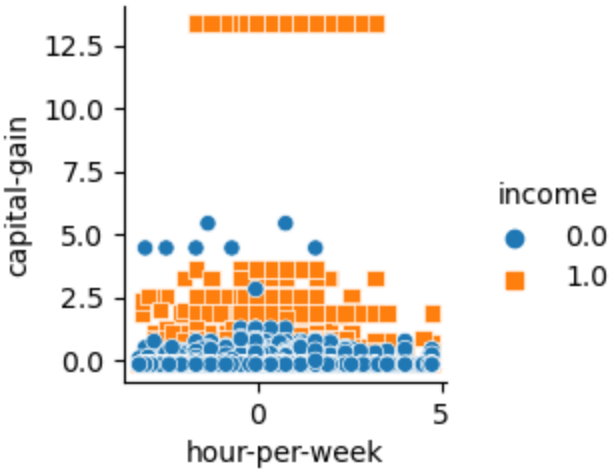
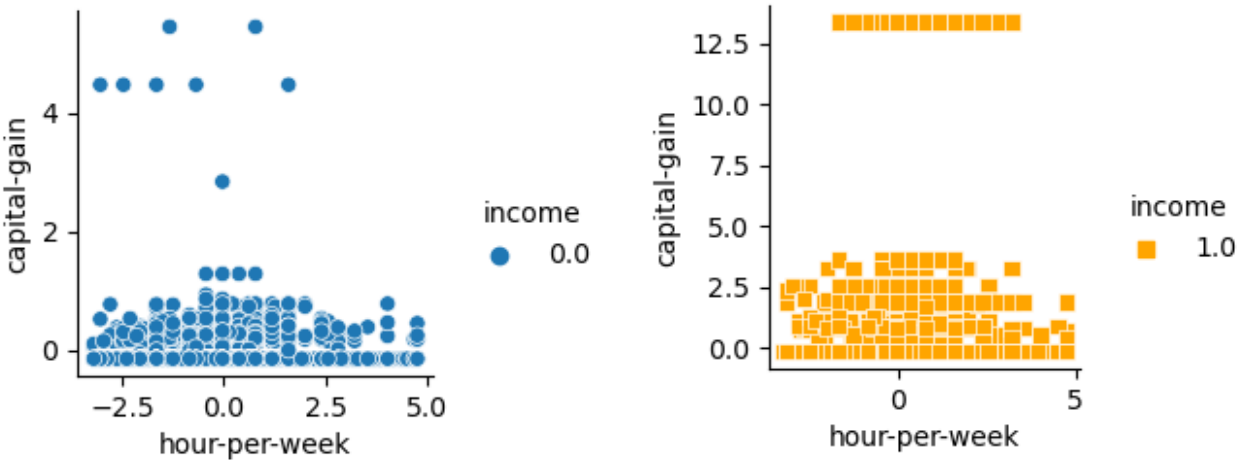


Figura 3.10: Gráfico relacionando duas variáveis originais: capital-gain e hour-per-week. Fonte: Elaborado pelo autor (2025).



(a) Classe 0, recebe 50 mil ou menos.

(b) Classe 1, recebe mais de 50 mil.

Figura 3.11: Gráfico 3.10 dividido para cada classe. Fonte: Elaborado pelo autor (2025).

3.4.2 Base de doenças cardíacas

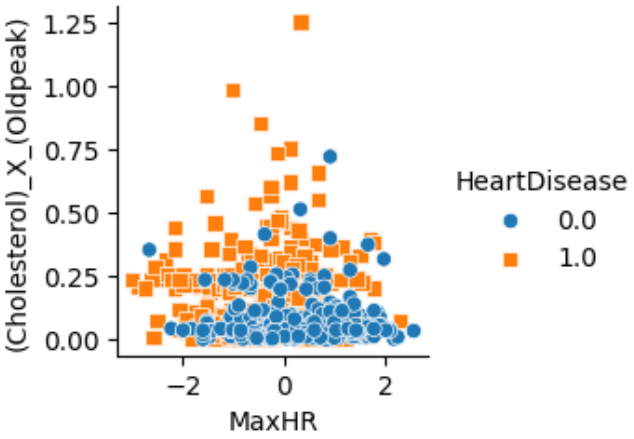
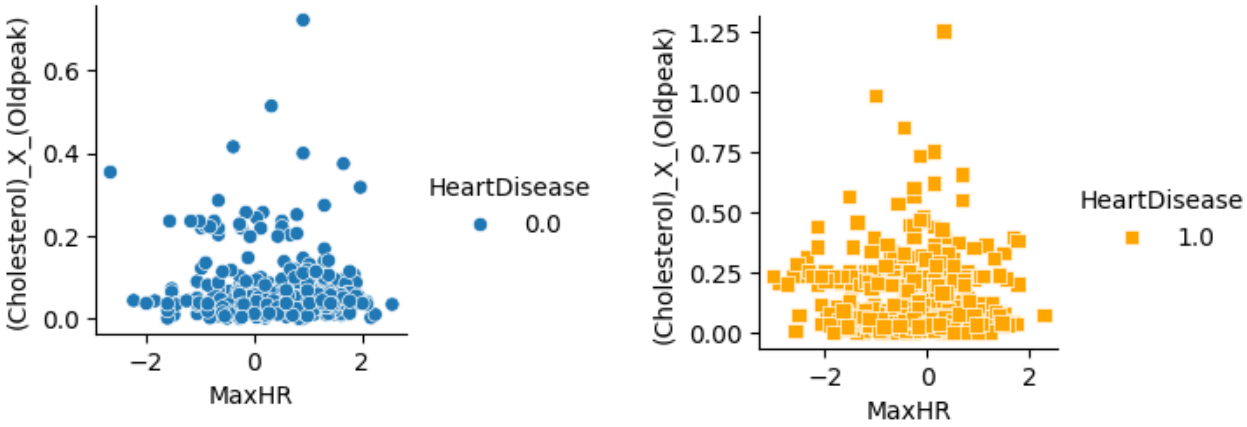


Figura 3.12: Gráfico relacionando uma variável nova, Cholesterol combinada com Oldpeak, com a variável original MaxHR. Fonte: Elaborado pelo autor (2025).



(a) Classe 0, não possui doenças cardíacas.

(b) Classe 1, possui doenças cardíacas.

Figura 3.13: Gráfico 3.12 dividido para cada classe. Fonte: Elaborado pelo autor (2025).

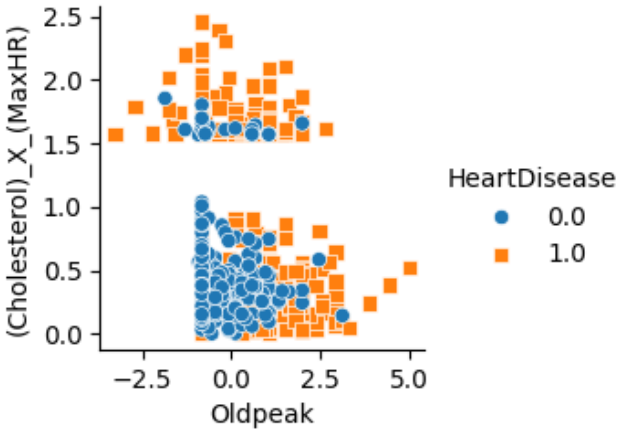
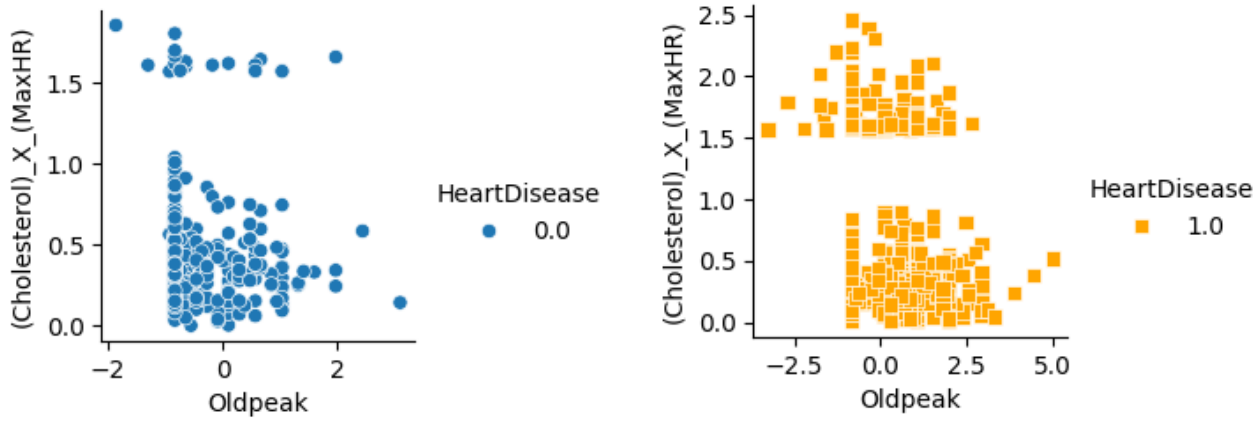


Figura 3.14: Gráfico relacionando uma variável nova, Cholesterol combinada com MaxHR, com a variável original Oldpeak. Fonte: Elaborado pelo autor (2025).



(a) Classe 0, não possui doenças cardíacas.

(b) Classe 1, possui doenças cardíacas.

Figura 3.15: Gráfico 3.14 dividido para cada classe. Fonte: Elaborado pelo autor (2025).

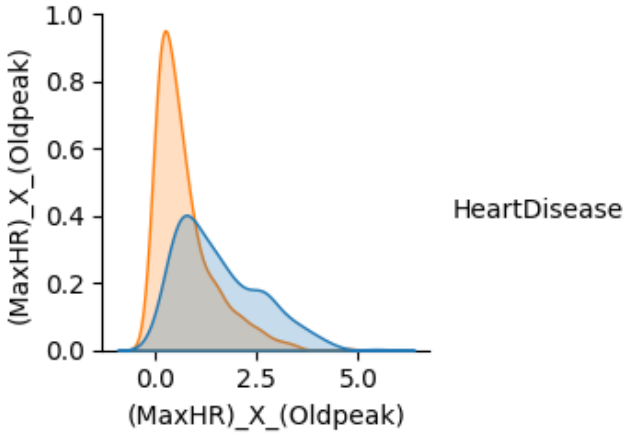


Figura 3.16: Gráfico mostrando a distribuição de classes da variável nova resultada da combinação das variáveis MaxHR com Oldpeak. Em laranja os pacientes com problemas cardíacos, saudáveis em azul. Fonte: Elaborado pelo autor (2025).

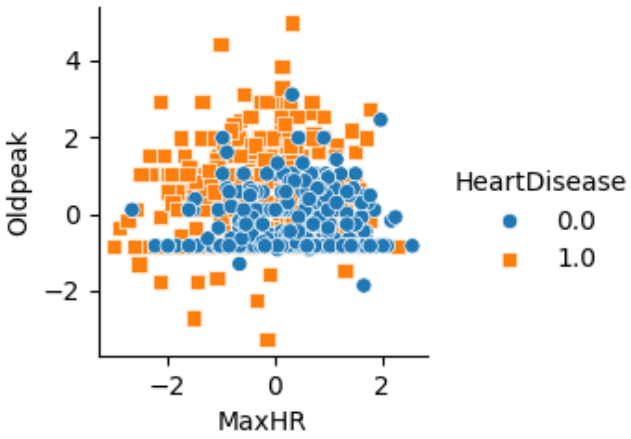
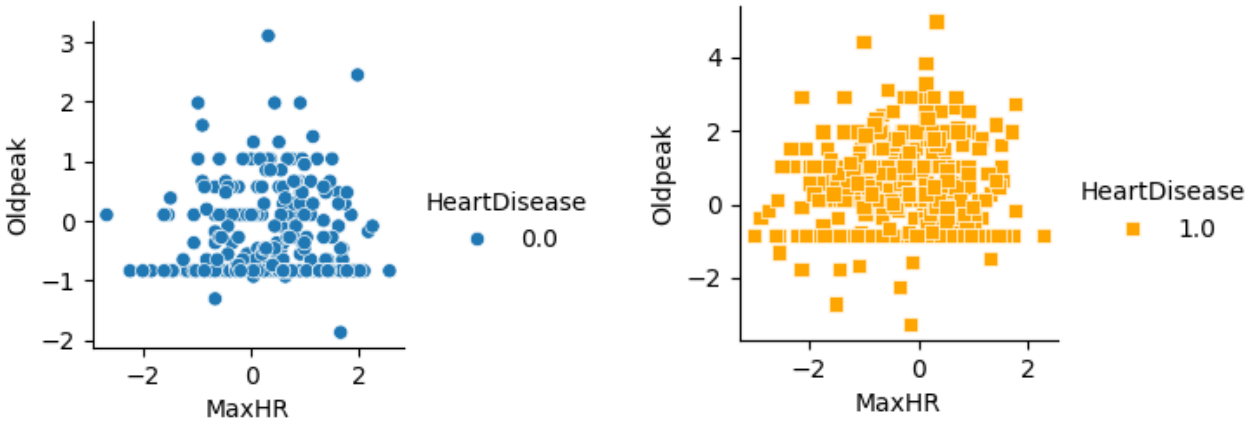


Figura 3.17: Gráfico relacionando duas variáveis originais: Oldpeak com MaxHR. Fonte: Elaborado pelo autor (2025).



(a) Classe 0, não possui doenças cardíacas.

(b) Classe 1, possui doenças cardíacas.

Figura 3.18: Gráfico 3.17 dividido para cada classe. Fonte: Elaborado pelo autor (2025).

3.5 Utilização dos dados na MLP

Em posse do conjunto original e do transformado das duas bases de dados, usou-se os conjuntos para treinar classificadores. Com a biblioteca sklearn [7] usou-se o método de grid search para escolher os melhores parâmetros para cada classificador.

Ao final do grid search, foram adotados as seguintes configurações para cada classificador:
Base de dados do censo:

- **MLP com o conjunto de dados original**: max_iter = 2000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (55,55).
- **MLP com o conjunto de dados transformado**: max_iter = 2000, activation = 'relu', tol = 0.0001, solver = 'adam', batch_size = 10, hidden_layer_sizes = (67,67).

Base de dados de doenças cardíacas:

- **MLP com o conjunto de dados original**: max_iter = 4000, activation = 'logistic', tol = 0.00001, solver = 'sgd', batch_size = 100, hidden_layer_sizes = (30,30).
- **MLP com o conjunto de dados transformado**: max_iter = 2000, activation = 'logistic', tol = 0.00001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (30,30).

Para uma boa avaliação dos modelos, foi aplicado o método de validação cruzada. Esse método divide o conjunto de dados em k partes e faz k treinamentos usando cada parte uma vez como conjunto de teste, enquanto o restante do conjunto é usado para o treino. A acurácia de cada treino foi armazenada num vetor para comparação futura.

Utilizando o método KFold da biblioteca sklearn [7], fez-se 5 ensaios no conjunto do censo e 20 ensaios no conjunto de doenças cardíacas, cada um dividindo o conjunto de dados em 10 partes para as abordagens com e sem a otimização. Em cada ensaio, a variável de estado aleatório foi modificada, possibilitando que o método gerasse sempre 10 conjuntos diferentes, o que garante uma melhor generalização.

Utilizou-se os valores obtidos para fazer testes de Shapiro e verificar a normalidade dos dados, além de plotar uma curva de densidades usando a biblioteca seaborn [11].

3.6 Modelos de feature engeneering para comparação

Com o intuito de observar o desempenho da técnica de coloração de advertência em relação a outros tipos de feature engeneering, aplicou-se técnicas de geração de novas features, seguidos por um treinamento de mlp, para comparar o desempenho dos diferentes modelos.

3.6.1 Método Polynomial Features

Utilizou-se o método **PolynomialFeatures** da biblioteca **sklearn** [7]. O método foi aplicado sobre os valores numéricos de cada dataset. A transformação resultante foi unida aos valores categóricos originais e aplicada no treinamento de um modelo classificador. O PolynomialFeatures combina as features originais e as eleva a diferentes potências 3.1 para criar características polinomiais sobre o conjunto original 3.2.

$$X = \begin{bmatrix} x_1, & x_2 \end{bmatrix} \tag{3.1}$$

$$X_p = \begin{bmatrix} 1, & x_1, & x_2, & x_1^2, & x_1x_2, & x_2^2 \end{bmatrix} \tag{3.2}$$

Após a aplicação da técnica do Polynomial Features, usou-se o grid search para chegar nos seguintes parâmetros:

Base de dados do censo:

- max_iter = 5000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (65,65).

Base de dados de doenças cardíacas:

- max_iter = 5000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (22,22).

3.6.2 Método Spline Transformer

Outro artifício presente na biblioteca da biblioteca **sklearn** [7] é o Spline Transformer. Repetindo o processo usado anteriormente, o Spline Transformer foi aplicado sobre os valores numéricos do dataset e a transformação resultante foi combinada com as variáveis categoricas já existentes. Em seguida foi realizado o treinamento do modelo. Para gerar novas features,

o Spline transformer divide os valores existentes em intervalos, chamados de nós, e, a partir deles, gera funções que serão aplicadas nas variáveis de forma a gerar novas features 3.3.

$$[x] \rightarrow [B_1(x), B_2(x), B_3(x), ..., B_N(x)] \tag{3.3}$$

Após a aplicação do Spline Transformer, usou-se o grid search para chegar nos seguintes parâmetros:

Base de dados do censo:

- max_iter = 5000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (66,66).

Base de dados de doenças cardíacas:

- max_iter = 5000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (23,23).

3.6.3 Utilização do LIME para seleção de features

Existem estudos, como o próprio artigo do LIME [3], que sugerem que o uso de técnicas de explicação local pode revelar padrões mais gerais. Esses padrões, por sua vez, podem servir para a seleção de features. Usou-se então o algoritmo LIME para observar as features que mais tiveram influência sobre a decisão do classificador em 1000 amostras da base de dados do censo e 184 amostras na base de dados das doenças cardíacas. Observou-se que as seguintes características se mostraram mais relevantes:

Base de dados do censo:

- capital-loos
- age
- hour-per-week
- education-num
- capital-gain

Base de dados de doenças cardíacas:

- Cholesterol.

- FastingBS.
- Oldpeak.

As características foram combinadas duas a duas de forma a criar novas colunas resultantes de cada combinação. Dado duas características a e b, utilizou-se da seguinte fórmula para gerar uma nova característica ab:

$$ab = \ln(1 + \sqrt{a^2 + b^2}) \tag{3.4}$$

Com essa função, relaciona-se a magnitude das duas características $\sqrt{a^2 + b^2}$ aplicada no logaritmo natural que é uma função não linear.

Após a criação das novas features, usou-se o grid search para chegar nos seguintes parâmetros:

Base de dados do censo:

- max_iter = 5000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 56, hidden_layer_sizes = (59,59).

Base de dados de doenças cardíacas:

- max_iter = 5000, activation = 'logistic', tol = 0.0001, solver = 'sgd', batch_size = 100, hidden_layer_sizes = (30,30).

Capítulo 4

Resultados Obtidos

Esse capítulo apresenta os resultados obtidos dos classificadores com e sem a melhoria proposta, bem como os resultados das outras abordagens existentes.

4.1 Matriz de confusão

A matriz de confusão mostra o resultado das previsões de cada modelo no conjunto de teste. Através dela é possível observar os acertos e erros de cada modelo. As tabelas a seguir mostram o desempenho de cada modelo.

4.1.1 Base de dados do censo

	Previsto negativo	**Previsto positivo**
Real negativo	TN (4529)	FP (389)
Real positivo	FN (595)	TP (1000)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (50 mil ou menos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (mais de 50 mil).

Tabela 4.1: Matriz de Confusão sem pré-processamento. Fonte: Elaborado pelo autor (2025).

A tabela 4.1 apresenta os resultados das previsões do modelo sem pré-processamento, ela mostra uma acurácia de 84.9%, precision de 72%, recall de 62.7%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (4458)	FP (460)
Real positivo	FN (599)	TP (996)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (50 mil ou menos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (mais de 50 mil).

Tabela 4.2: Matriz de Confusão com a utilização do PolynomialFeatures. Fonte: Elaborado pelo autor (2025).

A tabela 4.2 apresenta os resultados das previsões do modelo com o Polynomial Features, ela mostra uma acurácia de 83.7%, precision de 68.4%, recall de 62.4%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (4513)	FP (405)
Real positivo	FN (630)	TP (965)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (50 mil ou menos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (mais de 50 mil).

Tabela 4.3: Matriz de Confusão com a utilização do SplineTransformer. Fonte: Elaborado pelo autor (2025).

A tabela 4.3 apresenta os resultados das previsões do modelo com o SplineTransformer, ela mostra uma acurácia de 84.1%, precision de 70.4%, recall de 60.5%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (4522)	FP (396)
Real positivo	FN (580)	TP (1015)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (50 mil ou menos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (mais de 50 mil).

Tabela 4.4: Matriz de Confusão utilizando Feature Engeneering com as features selecionadas a partir do LIME. Fonte: Elaborado pelo autor (2025).

A tabela 4.4 apresenta os resultados das previsões do modelo com o Feature Engeneering com as features selecionadas a partir do LIME, ela mostra uma acurácia de 85%, precision de 71.9%, recall de 63.6%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (4565)	FP (353)
Real positivo	FN (572)	TP (1023)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (50 mil ou menos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (mais de 50 mil).

Tabela 4.5: Matriz de Confusão com a proposta da Coloração de Advertência. Fonte: Elaborado pelo autor (2025).

A tabela 4.5 apresenta os resultados das previsões do modelo com a Coloração de Advertência, ela mostra uma acurácia de 85.8%, precision de 74.3%, recall de 64.1%.

Observando os valores de acurácia, precision e recall de cada tabela, observa-se que a Coloração de Advertência conseguia a melhor acurácia, acertando mais resultados, o maior valor na precision demostra que, ao fazer uma previsão positiva, ele tem mais chance de estar certo que os demais modelos e o valor de Recall mostra que ele teve uma capacidade maior de encontrar os reais positivos.

4.1.2 Base de dados de doenças cardíacas

	Previsto negativo	**Previsto positivo**
Real negativo	TN (56)	FP (21)
Real positivo	FN (11)	TP (96)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (sem problemas cardíacos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (com problemas cardíacos).

Tabela 4.6: Matriz de Confusão sem pré-processamento. Fonte: Elaborado pelo autor (2025).

A tabela 4.6 apresenta os resultados das previsões do modelo sem pré-processamento, ela mostra uma acurácia de 82.6%, precision de 82.1%, recall de 89.7%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (59)	FP (18)
Real positivo	FN (10)	TP (97)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (sem problemas cardíacos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (com problemas cardíacos).

Tabela 4.7: Matriz de Confusão com a utilização do PolynomialFeatures. Fonte: Elaborado pelo autor (2025).

A tabela 4.7 apresenta os resultados das previsões do modelo com o Polynomial Features, ela mostra uma acurácia de 84.8%, precision de 84.3%, recall de 90.7%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (58)	FP (19)
Real positivo	FN (17)	TP (90)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (sem problemas cardíacos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (com problemas cardíacos).

Tabela 4.8: Matriz de Confusão com a utilização do SplineTransformer. Fonte: Elaborado pelo autor (2025).

A tabela 4.8 apresenta os resultados das previsões do modelo com o SplineTransformer, ela mostra uma acurácia de 80.4%, precision de 82.6%, recall de 84.1%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (57)	FP (20)
Real positivo	FN (11)	TP (96)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (sem problemas cardíacos).

FP Falso Positivo: Modelo previu "1", esperava-se "0".

FN Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (com problemas cardíacos).

Tabela 4.9: Matriz de Confusão utilizando Feature Engeneering com as features selecionadas a partir do LIME. Fonte: Elaborado pelo autor (2025).

A tabela 4.9 apresenta os resultados das previsões do modelo com Feature Engeneering com as features selecionadas a partir do LIME, ela mostra uma acurácia de 83.2%, precision de 82.8%, recall de 89.7%.

	Previsto negativo	**Previsto positivo**
Real negativo	TN (56)	FP (21)
Real positivo	FN (9)	TP (98)

TN Verdadeiro Negativo: Modelo acertou a classe 0 (sem problemas cardíacos).

FN Falso Positivo: Modelo previu "1", esperava-se "0".

FP Falso Negativo: Modelo previu "0", esperava-se "1".

TP Verdadeiro Positivo: Modelo acertou a classe 1 (com problemas cardíacos).

Tabela 4.10: Matriz de Confusão com a proposta da Coloração de Advertência. Fonte: Elaborado pelo autor (2025).

A tabela 4.10 apresenta os resultados das previsões do modelo com a Coloração de Advertência, ela mostra uma acurácia de 83.7%, precision de 82.4%, recall de 91.6%.

Observando os valores de acurácia, precision e recall de cada tabela, observa-se que a Coloração de Advertência conseguia a uma boa acurácia e um bom valor de precisão, apesar de ficar atrás do método Polynomial Features nesses valores, contudo, obteve o maior Recall entre os modelos mostrando que ele foi o melhor em encontrar os reais positivos.

4.2 Curva de Perda

Foi gerada a curva de perda dos modelos utilizando a biblioteca matplotlib [12]. Comparou-se a curva de perda dos diferentes modelos nas duas bases de dados 4.1 4.2.

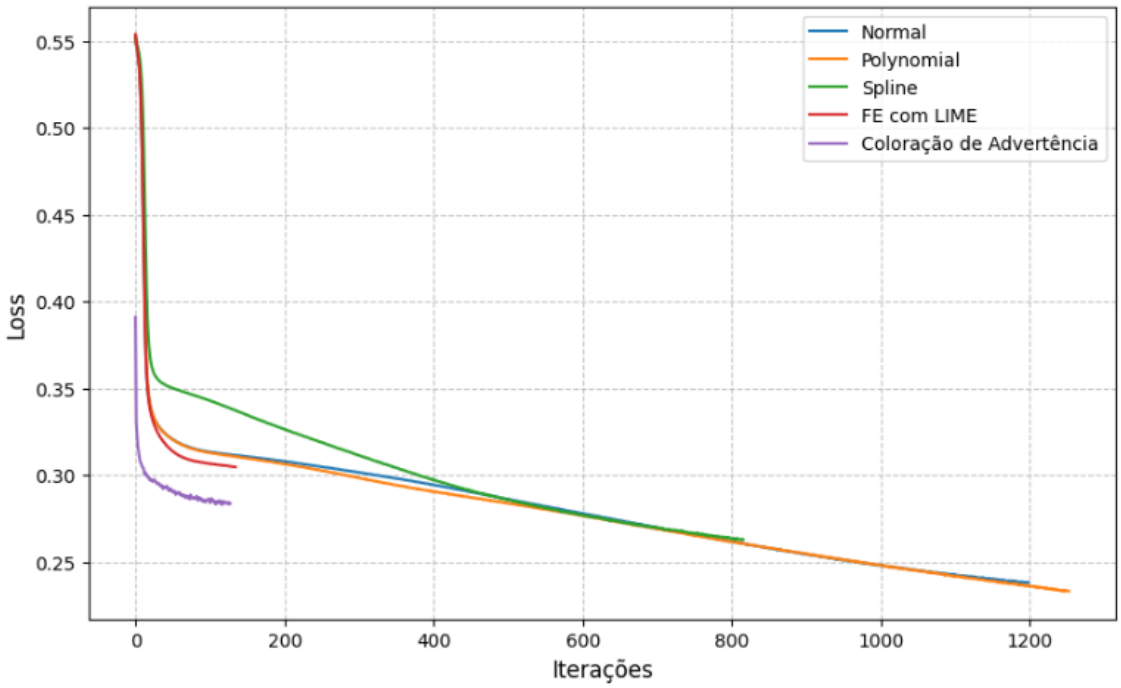


Figura 4.1: Curva de perda dos modelos treinados com a base do censo. Fonte: Elaborado pelo autor (2025).

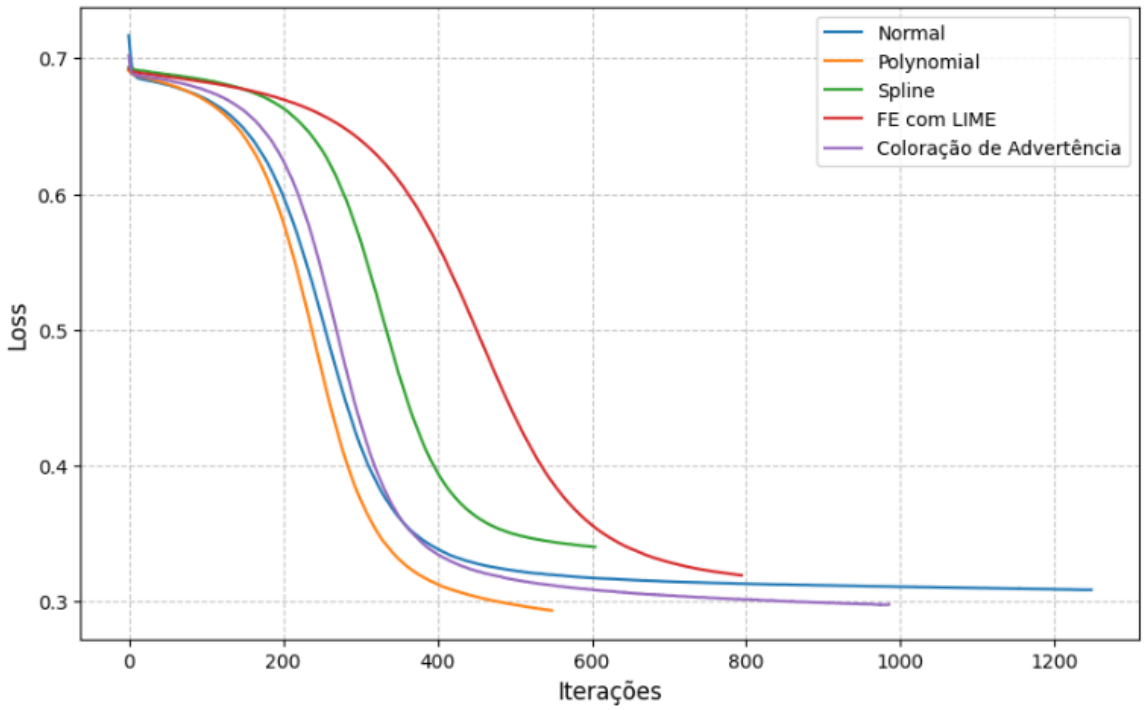


Figura 4.2: Curva de perda dos modelos treinados com a base de doenças cardíacas. Fonte: Elaborado pelo autor (2025).

Na base de dados do censo, as curvas mostram uma convergência mais rápida do modelo com a aplicação da Coloração de Advertência, entretanto, na base de dados de doenças cardíacas, o modelo com a Coloração de Advertência foi o segundo que mais demorou para convergir.

4.3 Validação Cruzada

Ao realizar a validação cruzada, foi obtido as seguintes médias de acurácias em cada modelo Tabela 4.11.

4.3.1 Médias na Base do Censo

Iteração	Médias sem otimização	Médias Polynomial-Features	Médias Spline-Transformer	Médias Feature Engeneering com LIME	Médias com Coloração de Advertência
1	0.8417	0.8285	0.8476	0.8427	0.8588
2	0.8428	0.8333	0.8470	0.8427	0.8593
3	0.8462	0.8324	0.8474	0.8456	0.8581
4	0.8400	0.8309	0.8456	0.8422	0.8604
5	0.8455	0.8369	0.8466	0.8437	0.8589

Tabela 4.11: Médias de acurácia por Iteração. Fonte: Elaborado pelo autor (2025).

Média total sem otimização	Média total Polynomial-Features	Média total SplineTransformer	Média total Feature Engeneering com LIME	Média total com Coloração de Advertência
0.84324	0.8324	0.8468	0.8434	0.8591

Tabela 4.12: Média de acurácia total. Fonte: Elaborado pelo autor (2025).

Os resultados mostram que o modelo com a Coloração de Advertência, além de obter uma média total maior, obteve uma média de acurácia melhor em todos os casos.

4.3.2 Médias na Base de Doenças Cardíacas

Iteração	Médias sem otimização	Médias Polynomial-Features	Médias Spline-Transformer	Médias Feature Engeneering com LIME	Médias com Coloração de Advertência
1	0.8639	0.8736	0.828	0.8671	0.8726
2	0.8605	0.8692	0.8268	0.8648	0.8616
3	0.8617	0.8769	0.8247	0.8606	0.8747
4	0.866	0.8725	0.8562	0.8649	0.8693
5	0.8616	0.8703	0.8551	0.8648	0.8682
6	0.8616	0.8692	0.8562	0.8638	0.8725
7	0.8605	0.8704	0.854	0.8594	0.8791
8	0.8638	0.8682	0.8573	0.8649	0.8703
9	0.8594	0.8671	0.8562	0.8616	0.8715
10	0.8617	0.8671	0.854	0.866	0.8682
11	0.8638	0.8801	0.8268	0.8638	0.866
12	0.865	0.8726	0.8562	0.8693	0.8672
13	0.8594	0.8725	0.8519	0.8573	0.8747
14	0.8616	0.8681	0.854	0.8627	0.867
15	0.8616	0.866	0.7931	0.866	0.8737
16	0.8639	0.8715	0.853	0.8627	0.8715
17	0.8606	0.8353	0.8541	0.8617	0.8705
18	0.867	0.8714	0.8594	0.8649	0.8703
19	0.8693	0.8759	0.8193	0.8671	0.8704
20	0.8649	0.8725	0.8562	0.8594	0.8715

Tabela 4.13: Médias de acurácia por Iteração. Fonte: Elaborado pelo autor (2025).

Média total sem otimização	**Média total Polynomial-Features**	**Média total SplineTransformer**	**Média total Feature Engeneering com LIME**	**Média total com Coloração de Advertência**
0.8629	0.8695	0.8446	0.8636	0.8705

Tabela 4.14: Média de acurácia total. Fonte: Elaborado pelo autor (2025).

Os resultados mostram que o modelo com a Coloração de Advertência obteve médias superiores aos outros modelos em vários casos, contudo algumas vezes foi superado pela abordagem Polynomial Features. Mesmo assim o modelo proposto ainda obteve uma média geral maior, apesar de ser uma diferença pouco significativa.

4.3.3 Comparação Entre Modelos Usando Boxplot

Usou-se o método de boxplot da biblioteca seaborn [13] para observar e comparar a distribuição que cada um teve durante o processo de validação cruzada. A figura 4.3 mostra que, na base de dados do censo, o método de Coloração de Advertência obteve a maior média, além de uma menor variabilidade nos resultados, o que indica que o modelo apresentou mairo consistência.

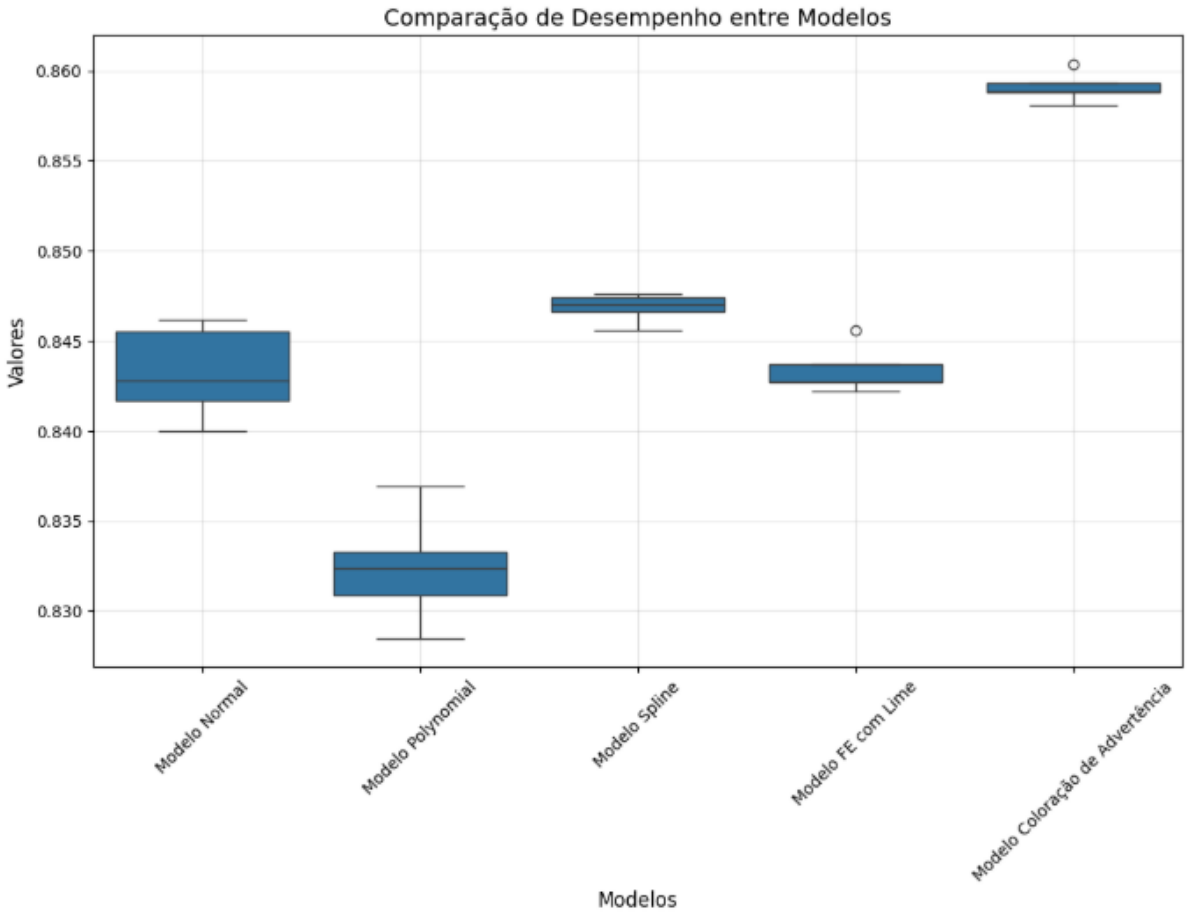


Figura 4.3: Boxplot das médias obtidas na validação cruzada na base de dados do Censo. Fonte: Elaborado pelo autor (2025).

Já na figura 4.4, que se refere à base de dados de doenças cardíacas, mostra que o método de Coloração de Advertência também teve resultados bons e consistentes, contudo, o método de Polynomial Features obteve um resultado semelhante.

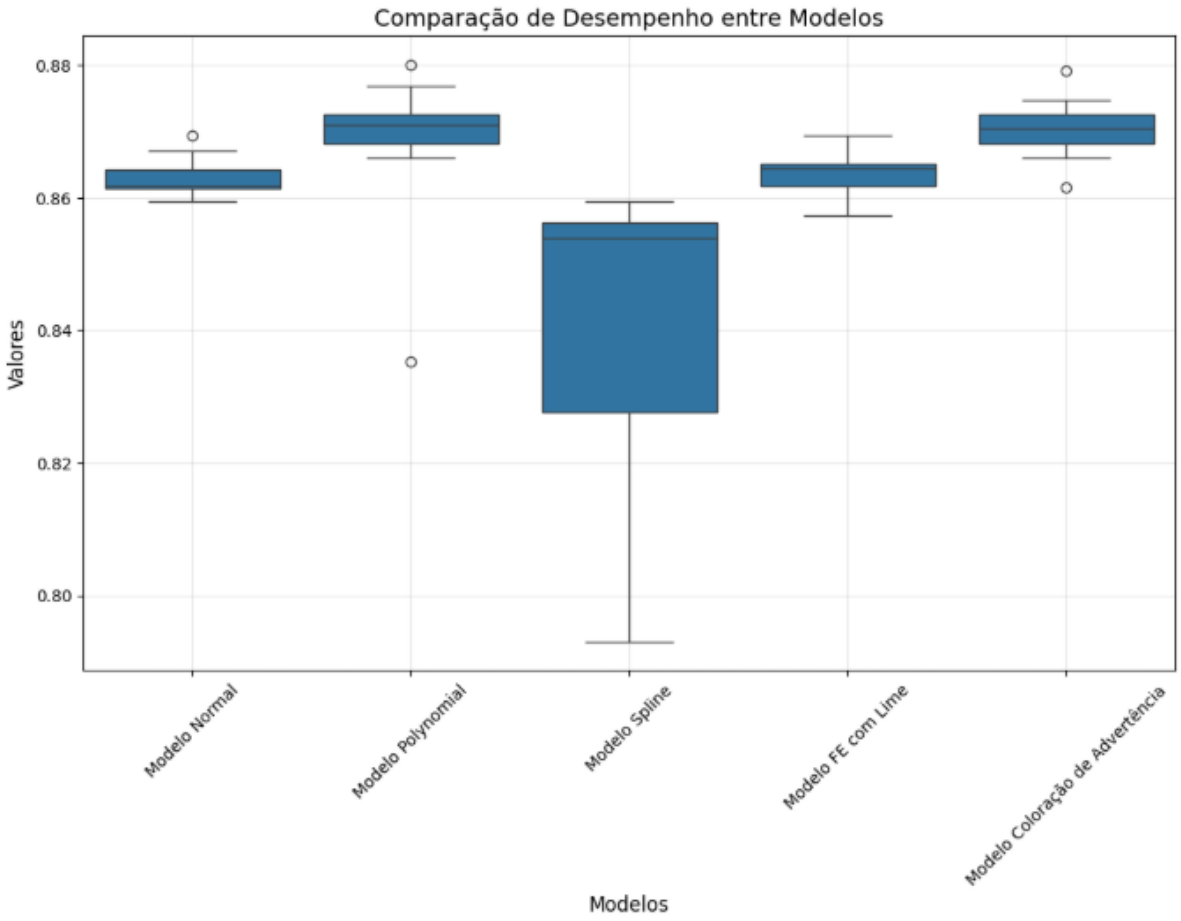


Figura 4.4: Boxplot das médias obtidas na validação cruzada na base de dados de doenças cardíacas. Fonte: Elaborado pelo autor (2025).

4.3.4 Tempo Gasto no Grid Search

As tabelas 4.15 e 4.16 mostram o tempo gasto para realizar o grid search em cada modelo das bases observadas. A tabela 4.17 mostra o tempo gasto para realizar a etapa evolutiva da Coloração de Advertência. Ao observar as tabelas, nota-se que, na base de dados do Censo, a soma do tempo de evolução com o tempo do grid search da Coloração de Advertência fica menor do que o tempo do grid search dos demais modelos, o que mostra que o tempo gasto durante a evolução foi compensado pelo baixo tempo do grid search. Na base de doenças cardíacas, apesar de que a soma do tempo de evolução com o tempo do grid search ficar maior que o tempo do grid search dos demais modelos, o tempo adicional causado pela etapa de evolução não é de uma ordem tão grande a ponto de tornar a técnica inviável.

Tempo total sem otimização (h)	Tempo total PolynomialFeatures (h)	Tempo total SplineTransformer (h)	Tempo total Feature Engeneering com LIME (h)	Tempo total com Coloração de Advertência (h)
30.76	23.27	22.83	27.43	9.92

Tabela 4.15: Tempo em horas para a realização do Grid Search em cada modelo da base do Censo. Fonte: Elaborado pelo autor (2025).

Tempo total sem otimização (min)	Tempo total PolynomialFeatures (min)	Tempo total SplineTransformer (min)	Tempo total Feature Engeneering com LIME (min)	Tempo total com Coloração de Advertência (min)
48.04	59.91	37.86	57.20	46.94

Tabela 4.16: Tempo em minutos para a realização do Grid Search em cada modelo da base de doenças cardíacas. Fonte: Elaborado pelo autor (2025).

Tempo da Coloração de Advertência na base do Censo (h)	Tempo da Coloração de Advertência na base de doenças cardíacas (h)
11.10	2.89

Tabela 4.17: Tempo em horas para a realização da etapa evolutiva da Coloração de Advertência em cada base de dados. Fonte: Elaborado pelo autor (2025).

4.4 Melhoria de Acurácia Obtida

Observou-se uma breve melhoria do modelo proposto quando comparado ao treinamento sem a aplicação da técnica proposta, como retratado nas tabelas 4.18 e 4.19.

Valor sem otimização	**Valor com a Coloração de Advertência**	**Melhoria absoluta**	**Melhoria Relativa**
84.32%	85.91%	1.59%	1.89%

Tabela 4.18: Valores percentuais obtidos na base de dados do Censo. Fonte: Elaborado pelo autor (2025).

Valor sem otimização	**Valor com a Coloração de Advertência**	**Melhoria absoluta**	**Melhoria Relativa**
86.29%	87.05%	0.76%	0.88%

Tabela 4.19: Valores percentuais obtidos na base de doenças cardíacas. Fonte: Elaborado pelo autor (2025).

Capítulo 5

Conclusão e trabalhos futuros

Este trabalho propôs a técnica de otimização chamada de Coloração de Advertência que, através de um tratamento nos dados, busca uma melhor acurácia no treinamento de um algoritmo de inteligência artificial. Após testes de duas bases de dados diferentes utilizando, a Coloração de Advertência obteve sucesso em conseguir um melhor desempenho quando aplicada em uma MLP.

Os testes mostraram que o modelo proposto atingiu uma pequena, porém consistente, melhora em relação aos modelos sem a aplicação da técnica. Apesar de que uma pequena melhora pudesse ser causado apenas pelo fato de uma MLP costumar ter pequenas variações de convergência, os resultados apresentaram uma melhora em quase todos os testes de validação cruzada, sustentando que a técnica proposta realmente conseguiu influenciar no desempenho do aprendizado.

Além da melhoria na acurácia, os gráficos da seção 3.4 sugerem um maior destaque visual para a separação de classes nos modelos em que a Coloração de Advertência foi aplicada.

Na questão temporal, as curvas de perda da seção 4.2 mostraram a influência da técnica apresentada sobre a convergência do treinamento da MLP. Embora não tenha contribuído tanto na base de dados de doenças cardíacas, observa-se que, na base de dados do Censo, o algoritmo convergiu em bem menos iterações quando foi utilizada a Coloração de Advertência, o que acabou refletindo no tempo de treinamento mostrado na tabela 4.15.

Usualmente, um ganho pequeno não costuma ter uma grande importância, contudo, existem situações onde mesmo uma pequena diferença se faz importante. Situações de alto risco, como diagnóstico médico e detecções de fraude podem se beneficiar bastante mesmo que a melhora seja breve.

Em trabalhos futuros, pode-se testar diferentes funções de transformação, além de observar quais tipos de dataset a abordagem se sai melhor. Outra ideia é associar o modelo à outras modificações no dataset, um exemplo seria, antes de tentar fazer a Coloração de Advertência, realizar uma transformada de fourier nos dados e observar se ocorre algum ganho.

Referências Bibliográficas

[1] V. RASCHKA, Sebastian; MIRJALILI, *Python Machine Learning*. Packt Publishing Ltd, 3 ed., 2019.

[2] J. KALITA, *Machine Learning: Theory and Practice*. CRC Press, 1 ed., 2023.

[3] G. VISANI, "Lime: explain machine learning predictions." Disponível em: https://medium.com/data-science/lime-explain-machine-learning-predictions-af8f18189bfe, 2020. Acesso em: 21 abr. 2025.

[4] E. B. Poulton, *The Colours of Animals*. London: Kegan Paul, Trench, Trübner & Co., 1890.

[5] V. GOLDBERG, David E.; MIRJALILI, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Inc, 3 ed., 1989.

[6] V. KUMAR, *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC, 1 ed., 2009.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 4765–4774, 2017. Acesso em: 20 jun. 2025.

[9] T. ELMETWALLY, "Census income dataset." Disponível em: https://www.kaggle.com/datasets/tawfikelmetwally/census-income-dataset, 2024. Acesso em: 2 nov. 2024.

[10] Fedesoriano, "Heart failure prediction dataset." Disponível em: https://www.kaggle.com/fedesoriano/heart-failure-prediction, 2021. Acesso em: 20 abr. 2025.

[11] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] M. L. Waskom, "seaborn: statistical data visualization," 2021.