Universidade Federal de Pernambuco

Centro de Informática

Graduação em Engenharia da Computação

Comparação de técnicas de machine learning na classificação de imagens de fundo de olho para identificar miopia patológica

Fernando do Rego Pessoa de Macedo Neto

Trabalho de Graduação

Recife, Pernambuco

Outubro , 2024

Universidade Federal de Pernambuco
Centro de Informática

Fernando do Rego Pessoa de Macedo Neto

Comparação de técnicas de machine learning na classificação de imagens de fundo de olho para identificar miopia patológica

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: Machine learning
Orientador: Adriano Augusto de Moraes Sarmento
Coorientador: Eronides Felisberto da Silva Neto

Recife, Pernambuco
Outubro, 2024

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

Macedo Neto, Fernando do Rego Pessoa de.
Comparação de técnicas de machine learning na classificação de imagens de fundo de olho para identificar miopia patológica / Fernando do Rego Pessoa de Macedo Neto. - Recife, 2024.
47p. : il., tab.

Orientador(a): Adriano Augusto de Moraes Sarmento
Cooorientador(a): Eronides Felisberto da Silva Neto
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2024.
Inclui referências.

1. Aprendizagem de Máquina. 2. Classificação de imagens. 3. Detecção de Miopia. 4. Comparação de classificadores. 5. Miopia. I. Sarmento, Adriano Augusto de Moraes . (Orientação). II. Silva Neto, Eronides Felisberto da. (Coorientação). IV. Título.

000 CDD (22.ed.)

Fernando do Rego Pessoa de Macedo Neto

Comparação de técnicas de machine learning na classificação de imagens de fundo de olho para identificar miopia patológica

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

Aprovado em: 18/10/2024

BANCA EXAMINADORA

_____

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

Universidade Federal de Pernambuco

_____

Prof. Eronides Felisberto da Silva Neto (Coorientador)

Universidade Federal de Pernambuco

_____

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

Universidade Federal de Pernambuco

Agradecimentos

Agradeço à minha família, por me apoiarem desde o início da minha educação até agora, que certamente ainda não é o final da minha busca por conhecimento, porém é um marco muito importante para mim, o fim da minha graduação. Agradeço ao meu avô Fernando por sempre confiar em mim e por estar sempre me apoiando em tudo que eu faço. Agradeço aos meus pais, por sempre estarem me direcionando a buscar mais e crescer na vida, sempre me tornando uma pessoa melhor, e ao meu irmão, à quem tento servir de exemplo, e está seguindo os mesmos passos, já cursando o mesmo curso e seguindo a mesma jornada, que espero que termine ainda melhor.

Agradeço também aos meus amigos, Mateus Silvestre, Ricardo Jorge e João Pedro Barros, por estarem sempre comigo nessa loucura que é a vida, sempre incentivando uns aos os outros à crescermos juntos e buscar sempre nos tornar pessoas melhores em todos os aspectos possíveis.

Agradeço a todos os amigos que fiz na faculdade, Victor Hugo, Victor Ximenes, José Carlos, Zilde Souto, Pedro Nogueira, Lucas Ambrósio, Daniel Perazzo, Riei Joaquim, Vitor Miguel, Zenio Ângelo, Mateus Teotônio, Mateus Elias, Eneri D'angelis e todos os outros, que passaram vários momentos de sufoco comigo, noites viradas fazendo projetos e estudando para provas, além de todo o período remoto durante a pandemia. Minha graduação teria sido muito mais difícil sem o apoio de vocês e muito mais sem graça sem os momentos de união que passamos juntos, tanto os acadêmicos quanto os outros, que ajudaram bastante a descansar a mente quando era necessário.

Agradeço ao professor Augusto Sarmento pela orientação e acompanhamento no desenvolvimento desse trabalho e ao meu coorientador Eronides Felisberto, que além de ser meu amigo desde pequeno, também é professor da área e me ajudou bastante no desenvolvimento desse projeto.

Agradeço a toda experiência e às pessoas que estiveram comigo durante os projetos que foram além da minha graduação, desde os meus 2 anos estagiando na Motorola pelo projeto CIn/Motorola da FADE, que me mostraram na prática como é o mercado de trabalho dessa área, além de colocar em contato e me fazer conhecer pessoas incríveis. Agradeço também ao professor Vinícius Garcia que me guiou no meu período como monitor, em que ajudei outros alunos. Assim como eu fui ajudado pelos meus monitores em outras disciplinas, a quem também agradeço bastante.

"Aprender é a única coisa que a mente nunca se cansa, nunca tem medo e nunca se arrepende."

— Leonardo da Vinci

Resumo

Atualmente, o aprendizado de máquina está auxiliando as pessoas em cada vez mais áreas. Sempre ajudando, facilitando e simplificando trabalhos que antes dependiam de extenso trabalho humano para se obter resultados. Porém agora, esses trabalhos podem ser feitos automática e imediatamente com um altíssimo grau de precisão por uso destes tipos de programas. Alguns exemplos de tarefas são detecção de erros gramaticais em documentos de texto, detecção de inconsistências em documentos jurídicos, navegação inteligente em mapas e detecção de doenças e condições de saúde.

Diante deste cenário, foi selecionado o problema da detecção de miopia para contribuir na área da saúde, mais especificamente, a detecção de miopia pela detecção de problemas causados por ela, que podem ser observados em imagens de exames de fundo de olho, já que normalmente a miopia só é detectada em exames comuns de vista.

Múltiplos classificadores, então, foram utilizados em conjunto com algumas técnicas de tratamento de dados e extração de características. Além do ajuste de hiperparâmetros pelo uso da Busca de Grade e da Validação Cruzada. Com a finalidade de comparar a acurácia desses modelos, e encontrar não só quais os modelos, mas também quais os hiperparâmetros desses modelos obteriam melhor acurácia, e portanto seriam melhores escolhas para uma possível implementação embarcada do mesmo.

Palavras-chave: Aprendizado de Máquina; Classificadores; Miopia; Máquina de Vetores de Suporte; Regressão Logística; Floresta Aleatória; K-ésimo Vizinho mais Próximo; Classificador Gaussiano Ingênuo de Bayes; Histograma de Gradientes Orientados; Detecção; Olhos; Oftalmologia; Comparação.

Abstract

Nowadays, machine learning is helping people each time in more and more areas. Always facilitating and simplifying jobs that used to rely on extended human work to obtain results. But now, this can be done automatically and immediately with a high degree of precision through the use of these kinds of programs. Some examples of those tasks are grammatical error detection on text documents, detection of inconsistencies in legal documents, smart navigation in maps, and disease and health conditions detection.

In the face of this scenario, the problem of myopia detection was selected to contribute to the health area. Specifically, the detection of myopia through the detection of other problems that are caused by it. Since those problems can be observed in eye fundus exams, and myopia can only be detected through common eye exams.

So multiple classifiers were utilized in addition to some data treatment techniques and feature extraction. Along with hyperparameter tuning using Gridsearch and Cross-Validation. With the intent of comparing the accuracy of those models, and not only finding which models, but also which hyperparameter combination will obtain the highest accuracy, therefore being the best choices for a possible embedded implementation of this classifier.

Keywords: Machine Learning; Classifiers; Myopia; Support Vector Machine; Logistic Regression; Random Forrest; K-nearest Neighbors; Gaussian Naive Bayes; Histogram of Oriented Gradients; Detection; Eyes; Ophthalmology; Comparison.

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

1.1 Motivação

As deficiências oculares estão se tornando cada vez mais normais, principalmente entre jovens e adultos de todo o mundo. Atualmente a miopia afeta mais de dois bilhões de pessoas [6] , o que é o dobro da quantidade de pessoas afetadas há 50 anos. Ela é um dos problemas oculares mais comuns no mundo inteiro, e a OMS projeta que até 2050 metade da população mundial será míope, e que no Brasil a taxa de altos graus de miopia (acima de 6 graus) está aumentando mais que os graus baixos e moderados [7].

Isso se torna ainda mais preocupante pois a alta miopia também pode provocar diversos outros problemas oculares e até cegueira. Portanto, é perceptível que existe uma forte demanda de métodos de detecção mais precisos e automatizados para auxiliar na realização de diagnósticos precoces da miopia. Já que a detecção precoce dessa condição é de extrema importância, principalmente em crianças, pois pode evitar a perda irreversível de visão além dessas outras complicações [8].

Hoje em dia, inúmeras aplicações de inteligência artificial vêm sendo utilizadas em diversas áreas com a finalidade de simplificar e facilitar a vida das pessoas. Segundo a OMS, a inteligência artificial é uma grande promessa para melhorar a prestação de serviços de saúde em todo o mundo [9]. No âmbito da medicina, essas aplicações podem auxiliar tanto os médicos quanto os pacientes na detecção precoce de diversos problemas, como a miopia, com um altíssimo grau de precisão. Apesar de não ser exatamente o resultado definitivo de um especialista, os dados utilizados para o treinamento do algoritmo viriam de diagnósticos feitos pelos especialistas. Assim fazendo com que a aplicação tenha uma resposta bastante precisa, além de imediata. O que resolveria o problema da detecção da miopia.

A criação de uma aplicação de inteligência artificial, nesse contexto, pode ser uma ferramenta poderosa na detecção dos casos de miopia. Essa ferramenta utilizaria para treinamento

múltiplas imagens de fundo de olho obtidas através de exames, com o laudo médico dizendo se a imagem é de um paciente com ou sem miopia. A partir disso, a aplicação aprenderia as características de imagens com e sem miopia, e então aprenderia a classificar novas imagens em "míope"ou "não míope".

Na literatura atual existem exemplos de trabalhos fazendo comparações de modelos de classificadores e até do uso de deep learning na resolução desse problema [10]. Porém alguns desses trabalhos não focavam apenas em miopia [11], usavam diferentes exames [12], não obtiveram uma acurácia final acima de 95% [12], ou não tinham datasets tão extensos [11].

Portanto, a motivação para este trabalho surge da necessidade de não só criar um classificador que possua essa finalidade de detectar pela imagem do exame de fundo de olho se o paciente teria ou não miopia com alta acurácia, mas também de comparar diferentes modelos de classificadores que são amplamente utilizados para problemas desse tipo e descobrir qual deles traria maior resultado dessa métrica na detecção desses casos. Dessa forma, além de criar um processo rápido e eficiente de diagnóstico, ele também seria mais acessível. Pois apesar de um sistema desse jamais substituir a presença de um médico, ele facilitaria a vida de diversas pessoas que vivem em regiões remotas, onde há escassez de equipamentos sofisticados e de profissionais mais experientes.

1.2 Objetivos

O objetivo deste projeto, portanto, é de desenvolver 5 classificadores, selecionados por serem bastante usados na literatura [10][12][11]. Os classificadores escolhidos foram: O Support Vector Machine (SVM), o Random Forrest Classifier, o K-Nearest Neighbors Classifier, o classificador Gaussian Naive Bayes e a Regressão Logística, todos da a biblioteca SciKit-Learn do python. Fazer o treinamento de cada um deles, ajustar os hiperparâmetros desses classificadores para tentar melhorar ainda mais os resultados obtidos por eles. E no final, fazer uma comparação detalhada entre os resultados obtidos. Comparando tanto antes quanto depois da otimização de cada um, usando métricas, como acurácia, f1-score, e a análise da matriz de confusão, de cada um desses casos. Tentando chegar à maior acurácia possível.

Após a análise espera-se obter uma melhor compreensão de quais classificadores se saíram melhor em questão de precisão, quais tiveram melhor acurácia, e quais foram melhores em generalizar o problema e obter melhores resultados. Assim, então, descobrindo quais perfor-

marão melhor em um cenário real. Conseguindo, desta forma, informações importantes para o um possível futuro desenvolvimento e implantação dessa tecnologia no mundo real, de forma embarcada em um dispositivo acoplado à própria máquina que realiza o exame para acelerar o processo de diagnóstico e auxiliar os profissionais da área de saúde e trazer mais conveniência para a vida dos pacientes.

1.3 Estrutura do documento

O resto do documento está organizado da forma a seguir:

Capítulo 2: Revisão teórica

Este capítulo explica um pouco mais sobre alguns conceitos relevantes que serão utilizados ao longo do trabalho e que serão fundamentais para o entendimento do mesmo, como o algoritmo HOG que extrai as features das imagens, os 5 classificadores escolhidos, Support Vector Classifier, Random Forrest Classifier, K Neighbours, Gaussian Naive Bayes e Logistic Regression, além de falar também sobre as métricas utilizadas para comparação de resultados, a precisão, recall, acurácia, f1-score, macro-avg, e matriz de confusão. Além disso este capítulo também irá justificar as escolhas dos algoritmos que foram usados e serão comparados neste estudo.

Capítulo 3: Metodologia

Este capítulo é o que vai explicar a metodologia utilizada na obtenção dos resultados, apresentando o passo a passo de todas as etapas do desenvolvimento da parte prática do projeto, dizendo especificamente o que foi feito e como foi feito, desde o conjunto de dados utilizado, explicando melhor o dataset e as técnicas de tratamento de dados e extração de features utilizadas, até os processos de treinamento, otimização e obtenção de resultados.

Capítulo 4: Resultados Obtidos

Este capítulo tem a função de fazer a exibição detalhada das métricas de cada um dos casos testados utilizando não só os dados obtidos, mas também matrizes de confusão e tabelas comparando estes resultados. Além disso, o capítulo mostrará também a análise dos resultados obtidos pelos métodos usados nos experimentos mencionados no capítulo anterior, e fazer uso das diferentes métricas de forma que a comparação dos classificadores, leve em conta todos os cenários, assim definindo qual deles obteve melhor desempenho no cenário proposto e encontrar o que melhor resolve o problema.

Capítulo 5: Conclusão

O capítulo final do trabalho tem a função de apresentar as conclusões encontradas, sintetizando as principais descobertas, e discutir sobre as implicações dessa conclusão para o cenário proposto de automação no diagnóstico da miopia. Além disso, também será discutido sobre possíveis melhorias e sugestões de projetos futuros para pesquisa nessa área, como outras técnicas que poderiam ter sido utilizadas para melhoria dos classificadores, o uso de redes neurais para a classificação, a expansão da classificação para captar também outros problemas oculares ou até a possibilidade de embarcar essa tecnologia direto na máquina de exame.

2 Revisão Teórica

Neste capítulo serão apresentados os conceitos necessários para entendimento do trabalho, ele será dividido em quatro seções, a primeira falará do método de grayscale, a segunda seção da parte de extração de features e o algoritmo HOG que é o que será utilizado neste trabalho, a terceira seção discutirá sobre os algoritmos de classificação a serem comparados e a quarta explicará as métricas que serão utilizadas para comparação dos resultados.

2.1 Miopia

A miopia é um problema ocular comum que impossibilita que as pessoas que a tem enxerguem objetos distantes com clareza. Os olhos do míope geralmente são mais achatados que o normal, causando a formação da imagem do que ele enxerga um pouco antes da retina, como pode ser visto na Figura 2.1.

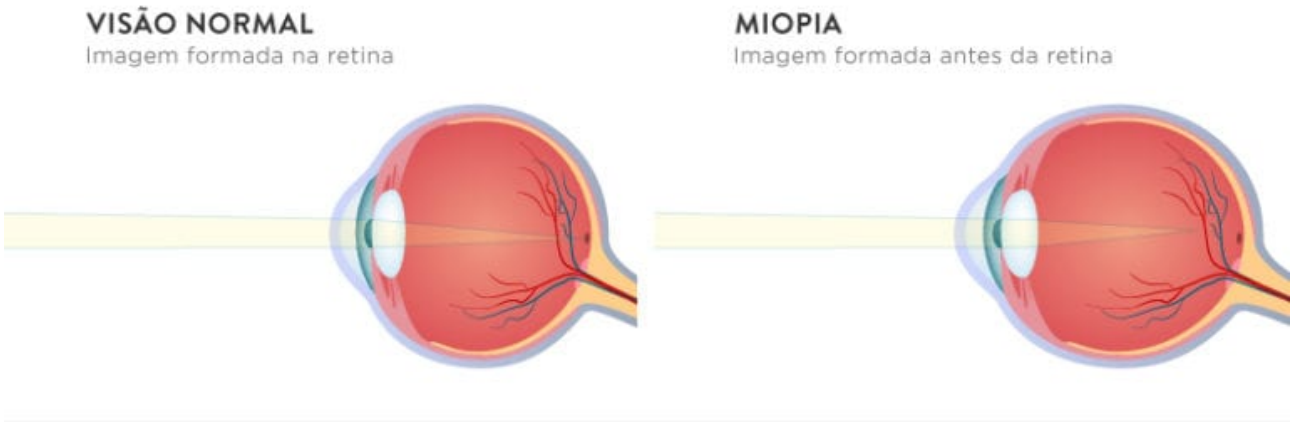


Figura 2.1: Formação de imagem em um olho normal e em um olho míope [1].

A alta miopia, definida quando o grau da miopia é maior que 6 graus, pode causar inúmeros outros problemas oculares como o descolamento da retina, edema na mácula (porção central da retina), catarata, degeneração da retina e glaucoma [13]. Apesar de a miopia não ser detectável

pela imagem do exame de fundo de olho, elas podem ser usadas para treinamento do classificador pois muitos desses problemas causados por ela são visíveis nelas, o que pode ser observado na Figura 2.2. Os problemas mostrados são: Atrofia peripapilar (a), que ocorre próximo ao disco óptico; Retina tesselada (b), com grandes vasos coróides observáveis; Hemorragia macular (c), que ocorre perto do centro da fóvea ou nas suas imediações; Atrofia de retina (d), que é a aglomeração de pigmento dentro e ao redor da lesão devido à migração das células degeneradas do epitélio pigmentado da retina para as camadas internas da retina; Descolamento de retina (e), um problema grave em que a retina é puxada para fora da sua posição normal; Opacidade vítrea (f), em que o vítreo encolhe e forma fios que projetam sombras na retina.

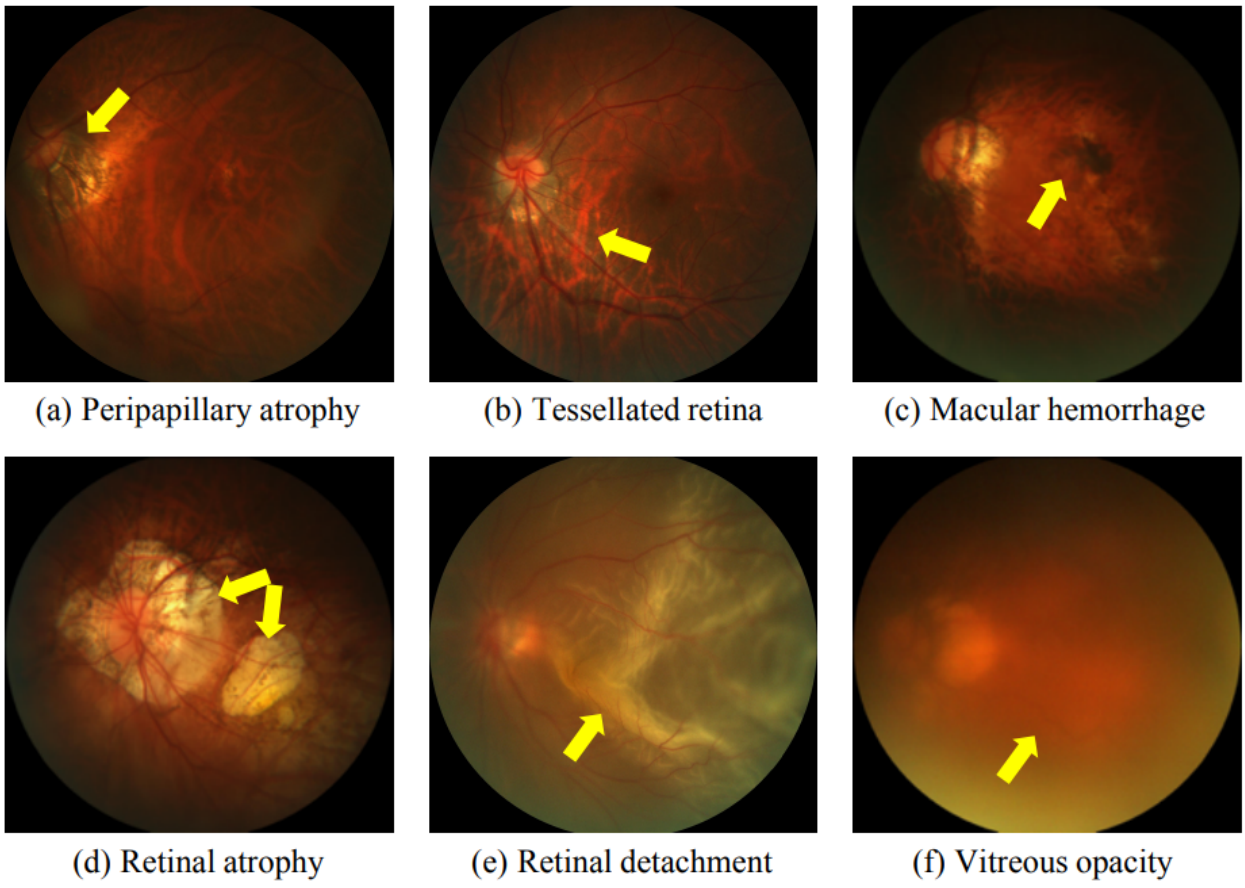


Figura 2.2: Exemplos de lesões na retina comumente associadas à miopia [2].

2.2 Grayscale

A técnica de conversão de uma imagem colorida para escalas de cinza (grayscale [14]) é uma técnica de tratamento de dados utilizada para transformar uma imagem colorida em monocromática. Transformando os valores rgb (red, green and blue), que simbolizam as cores

de cada um dos pixels em valores que variam de 0 (preto) a 1 (branco). Essa técnica é bastante utilizada em modelos de machine learning pois ajuda a reduzir o ruído das variações de cores enquanto foca na textura e contraste das imagens. A técnica ainda ajuda a aumentar a eficiência computacional do modelo, tornando sua execução menos custosa.

Outro motivo para a utilização desta técnica é o algoritmo de extração de features utilizado. Pois ele tende a funcionar melhor em imagens em preto e branco justamente pelos motivos citados acima. Além de que mesmo que ele possa ser utilizado com imagens coloridas, as informações de cor não são relevantes, já que ele depende apenas da intensidade luminosa dos pixels da imagem.

2.3 Extração de Características

As features de uma imagem são características ou padrões que podem ou não se repetir em outras imagens. Essas features são utilizadas pelo classificador para tentar encontrar esses padrões assim, classificando imagens de olhos como míopes ou não míopes.

O algoritmo de extração de features que foi escolhido para ser utilizado neste trabalho foi o Histogram of Oriented Gradients, ou HOG [15], um algoritmo amplamente utilizado em visão computacional e detecção de objetos. Este algoritmo foi selecionado pela sua eficácia na detecção de bordas e padrões, como se pode ver na figura Figura 2.3. Outros algoritmos poderiam ter sido parte da comparação, porém a parte de extração de features só utilizando um algoritmo já foi muito custosa computacionalmente, então esse seria um possível tópico para pesquisa futura.

O algoritmo funciona calculando para cada pixel da imagem um gradiente em relação aos seus vizinhos, indicando a direção e a intensidade da mudança de intensidade de brilho. O intuito é de capturar bordas, curvas e detalhes presentes em cada bloco para que seja feita a comparação com outras imagens.

Então a imagem é dividida em células de tamanho predeterminado, e é criado um bloco contendo algumas destas células, por exemplo um bloco de 2x2 células. A partir de cada célula é extraída um histograma de valores, representando a orientação dos pixels na célula, sendo esses valores as features dessa célula. Finalmente as features de todas as células do bloco são concatenadas, e isso se repete, movimentando o bloco por todas as células da imagem, como se pode observar na Figura 2.4. Então, assim é criado um enorme vetor de features, com as

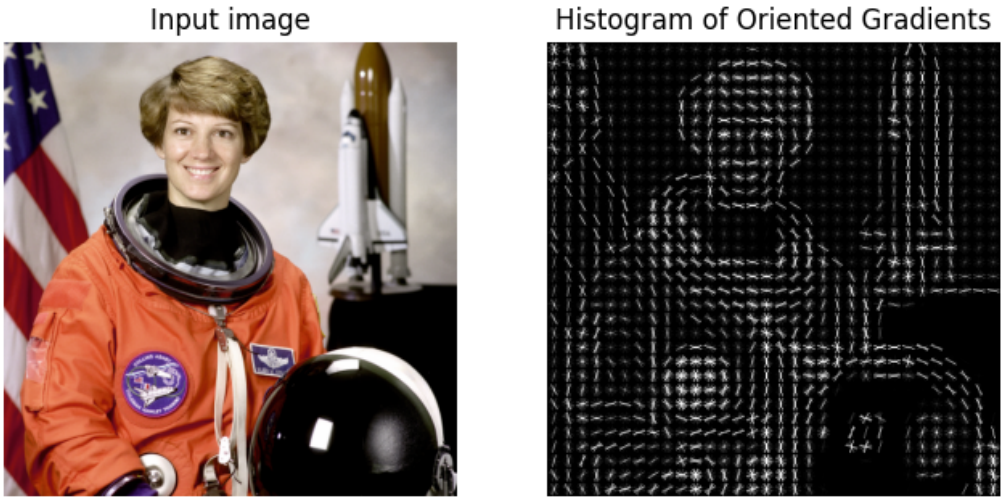


Figura 2.3: Exemplo da detecção de bordas do algoritmo HOG [3].

features de todos os blocos, que são usadas pelo classificador.

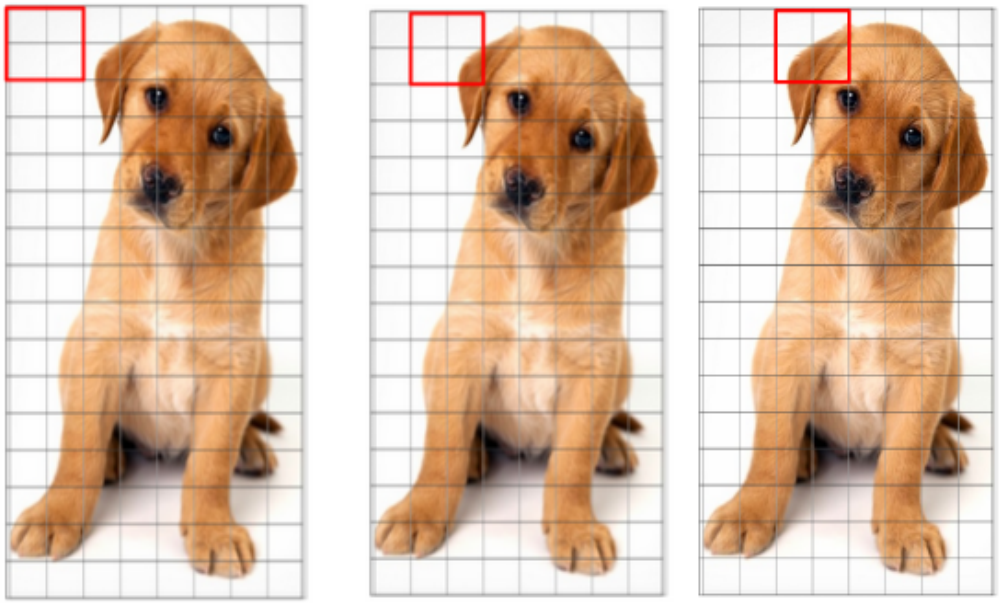


Figura 2.4: Movimentação do bloco pelas células no algoritmo HOG [4].

2.4 Algoritmos de Classificação

Os algoritmos de classificação utilizados foram o Support Vector Classifier (SVC), Gaussian Naive Bayes (GaussianNB), K-Neighbors Classifier (KNN), Random Forest Classifier e o Lo-

gistic Regression, todos da biblioteca do SciKit-Learn. Esses algoritmos foram selecionados por serem bastante utilizados em problemas de classificação desse tipo, além de abrangerem uma grande variedade de modelos, desde modelos mais simples como o classificador de Bayes como modelos mais robustos como o Support Vector Classifier.

2.4.1 Support Vector Classifier

O Support Vector Classifier [16] é um classificador mais robusto que tenta encontrar o hiperplano ótimo para separar os dados de forma mais clara nas diferentes classes. Ele tem uma ótima capacidade de generalização e uma boa eficácia para problemas com dados complexos, o que demonstra ser ótimo para o objetivo desse trabalho, já que envolve usar features extraídas das imagens do fundo do olho, onde poderá ser difícil isolar certos padrões com precisão, e por isso foi escolhido para fazer parte da comparação.

2.4.2 Gaussian Naive Bayes

O GaussianNB [17] é um classificador probabilístico do SciKit-Learn que assume que as features seguem uma distribuição gaussiana. Ele é um classificador simples, rápido e eficiente, e foi escolhido por ser de fácil implementação e rápida execução, além de muitas vezes fornecer resultados próximos aos modelos mais complexos, mesmo quando os dados não estejam distribuídos como o classificador pressupõe. Utilizando este modelo na comparação poderá ser observado se existe de fato uma melhora nos resultados ao utilizar outros modelo mais complexos e, caso exista, quão grande ela seria.

2.4.3 K-Neighbors Classifier

O K-Nearest Neighbors [18] é um classificador baseado na proximidade dos dados, ele classifica novos dados comparando com as "K"amostras que mais se aproximam desse novo dado. O algoritmo também é simples, mas ele pode ser bastante eficaz caso haja uma grande similaridade entre amostras de um mesmo grupo, portanto o algoritmo foi escolhido pois ele poderá ter um bom resultado de detecção se houver proximidade entre as features de olhos com ou sem miopia.

2.4.4 Random Forest Classifier

O classificador Random Forrest [19], é na verdade um classificador que une vários classificadores de arvore de decisão e combina as previsões delas de forma que melhor se adeque ao problema. Este classificador é um modelo robusto que tenta otimizar sua acurácia ao mesmo tempo que tenta reduzir o overfitting, fazendo com que ele tenha uma ótima generalização, e foi escolhido por ser um dos modelos mais robustos e mais amplamente utilizados e por oferecer uma ótima capacidade de lidar com uma maior variabilidade nos dados e estimação de importância das features, o que pode ser bom para distinguir quais delas realmente tem a ver com a miopia.

2.4.5 Logistic Regression

A regressão logística [20] é um modelo estatístico simples utilizado para classificar dados com base na probabilidade de pertencerem a cada uma das classes com base nas suas features. O modelo foi selecionado pois apesar de bem simples, ele é amplamente utilizado em problemas de classificação binária, como é o caso deste estudo, trazendo ótimos resultados além de uma boa eficiência computacional.

2.5 Métricas dos resultados

Para comparação dos resultados obtidos por cada um dos classificadores faz-se necessário o uso de diferentes métricas. É importante utilizar diferentes métricas para avaliar o desempenho do classificador, pois cada métrica avalia um diferente aspecto dele. A seguir são descritas as métricas empregadas nesse trabalho.

2.5.1 Precisão

A métrica de precisão [21] é basicamente a proporção de elementos corretamente categorizados como positivos, ou seja, a capacidade de evitar falsos positivos. Essa é uma métrica importantíssima, pois mesmo um modelo tendo maior acurácia que outro, as vezes um falso positivo pode ser mais custoso que um falso negativo, principalmente no caso de diagnósticos médicos, onde um falso positivo poderá levar um paciente a se preocupar e buscar tratamento desnecessariamente, então é importante ter em mente essa métrica ao analisar os dados. A

precisão é definida como:

$$\text{Precisão} = \frac{TP}{TP + FP}$$

Onde:

- TP = True Positives (Verdadeiros Positivos)
- FP = False Positives (Falsos Positivos)

2.5.2 Recall

O Recall [22] é uma métrica que identifica a capacidade do classificador de evitar falsos negativos, o que seria importante quando a condição buscada é algo bem grave e que não pode ser ignorado, como uma doença grave cuja não identificação dela pode fazer o paciente não buscar um tratamento que teria salvado a vida dele. Portanto comparando com outro classificador que tenha uma acurácia melhor, mas um Recall pior, nesse caso o Recall é uma métrica que não pode ser ignorada. O recall é definido por:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Onde:

- TP = True Positives (Verdadeiros Positivos)
- FN = False Negatives (Falsos Negativos)

2.5.3 F1-Score

O F1-Score [23] é uma média entre as métricas Precision e Recall, ela é utilizada para balancear a relação entre essas duas métricas, especialmente quando as classes estão desbalanceadas. Essa métrica é útil em cenários onde tanto os falsos positivos quanto os falsos negativos são importantes. O F1-Score é uma métrica valiosa para problemas em que os falsos positivos e falsos negativos têm um impacto semelhante, assim buscando um modelo que não seja tão desbalanceado nem para falsos positivos nem para falsos negativos.

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}}$$

Ou:

$$F1 = 2 \times \frac{TP}{2TP + FP + FN}$$

2.5.4 Macro-avg

A métrica de Macro-avg [24] é uma média que leva em conta a quantidade de dados em cada classe justamente para impedir o problema de "bias"que acontece com a acurácia. Ela é um ótimo indicador para mostrar que um classificador está tendencioso para uma das classes e é extremamente útil nos casos em que há um desbalanceamento de amostras. Existe a macro-avg de cada uma das métricas previamente citadas e elas são calculadas da seguinte maneira:

$$\text{Macro-Avg Precisão} = \frac{P_1 + P_2}{2}$$

$$\text{Macro-Avg Recall} = \frac{R_1 + R_2}{2}$$

$$\text{Macro-Avg F1} = \frac{F1_1 + F1_2}{2}$$

Onde:

- P_i = Precisão da classe i
- R_i = Recall da classe i
- $F1_i$ = F1-score da classe i

2.5.5 Acurácia

Essa é, como o nome já diz, a acurácia do modelo [25], e ela mede a proporção de todas as classificações corretas (tanto verdadeiros positivos quanto verdadeiros negativos) em relação ao total de exemplos. Essa é a métrica mais comum apesar de que caso o conjunto de dados esteja desbalanceado ela pode ser enganosa. Fazendo assim com que o classificador se torne um biased classifier (ou majority class classifier), ou seja, ele apenas atribui novos dados à classe com mais exemplares 100% das vezes. E como nesse caso, a maior parte dos exemplos é dessa classe, a acurácia vai estar com um valor alto, dando a entender que o classificador está com uma boa

taxa de acerto, enquanto a outra classe terá 0% de acerto, o que é um problema gravíssimo. A acurácia se dá pela seguinte fórmula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Onde:

- TP = True Positives (Verdadeiros Positivos)
- TN = True Negatives (Verdadeiros Negativos)
- FP = False Positives (Falsos Positivos)
- FN = False Negatives (Falsos Negativos)

2.5.6 Matriz de Confusão

A Matriz de Confusão [26] é uma ferramenta que oferece uma visão detalhada do desempenho do modelo, mostrando a distribuição de verdadeiros positivos (TP), verdadeiros negativos (TN), falsos positivos (FP) e falsos negativos (FN) obtidos pelo resultado da predição do classificador. A matriz de confusão é essencial para avaliar o desempenho de um modelo, pois permite a análise detalhada dos erros e acertos do classificador, o que é especialmente útil para ajudar a identificar onde o classificador mais errou, e consequentemente deve ser melhorado. A matriz de confusão é representada da seguinte forma:

$$\begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix}$$

2.6 GridSearchCV

O Grid Search [27] é uma técnica de fine-tuning (ajuste fino) usada para otimizar os hiperparâmetros dos classificadores. Os hiperparâmetros são parâmetros dos classificadores que devem ser definidos manualmente antes do treinamento e definem como o modelo se comportará, eles afetam diretamente o desempenho do modelo. O GridsearchCV é usado durante o treinamento com uma grade de possíveis valores para os hiperparâmetros do classificador que está sendo utilizado. Essa grade é então testada com o dataset de treino, fazendo todas as

possíveis combinações entre os valores definidos para cada variável. Essa busca é feita com uma validação cruzada, fazendo com que o dataset de treinamento seja dividido em k partes, e cada uma dessas partes é usada uma vez para testes, e k-1 vezes para treinamento, pode-se observar um exemplo dessa divisão com k=5 na Figura 2.5. Após todos os testes, é possível observar qual foi a combinação de hiperparâmetros que obteve o melhor resultado, e ,também, que teve o resultado mais consistente entre as k-folds testadas, assim evitando a chance de ocorrer overfitting [5], já que a subdivisão do dataset de treino ajuda a fazer uma testagem mais generalizada.

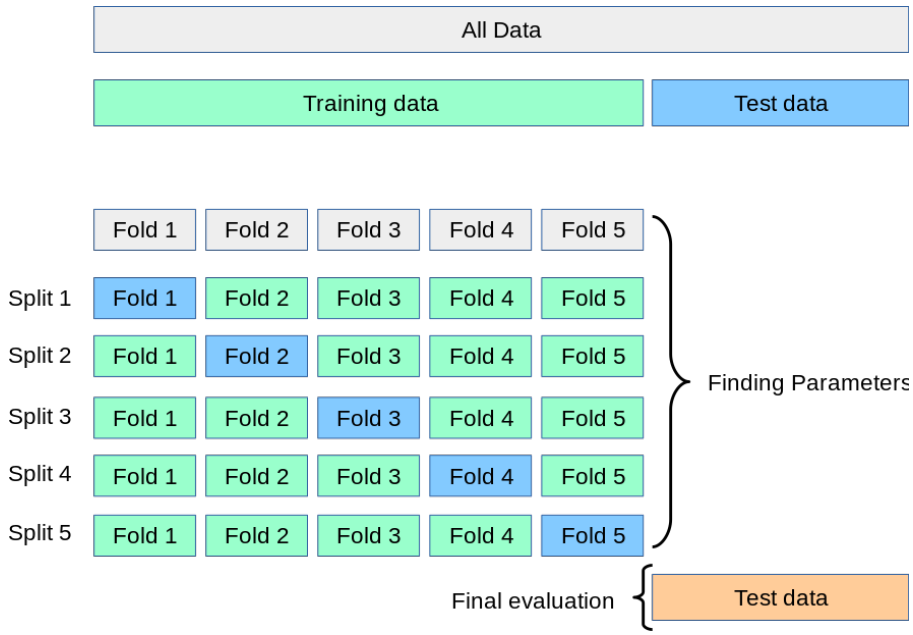


Figura 2.5: Divisão dos dados para uso do GridSearchCV [5]

3 Metodologia

A metodologia utilizada nesse projeto é mostrada na Figura 3.1.

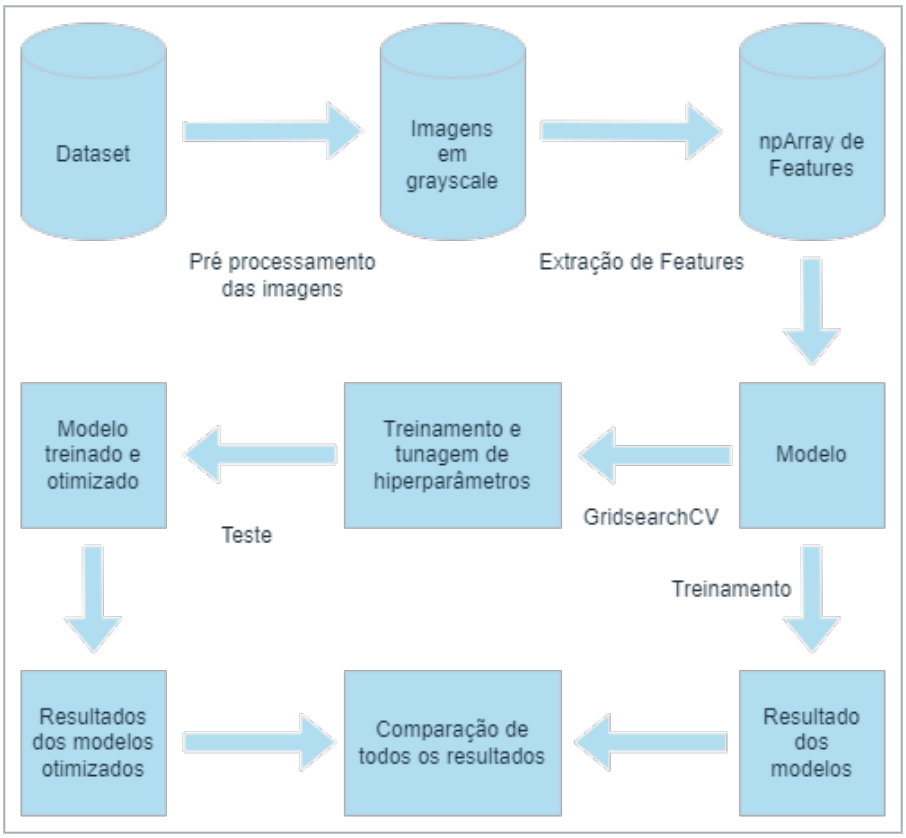


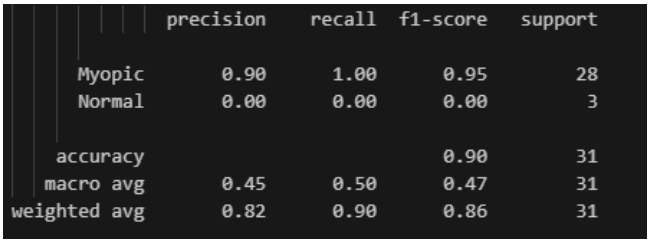
Figura 3.1: Esquema do projeto

3.1 Análise dos dados

Para teste e treinamento do classificador, foram utilizadas imagens retiradas do dataset disponibilizado pelo PALM (PathoLogic Myopia Challenge) [28], um evento de desafio do IEEE ISBI (International Symposium on Biomedical Imaging) [29], que aconteceu no ano de 2019 em Veneza, Itália, e foi disponibilizado online. Nesses dados estavam 1200 imagens de exames de

fundo de olho, já com as labels do diagnóstico de miopia [2] e algumas outras anotações que não foram utilizadas pois iam além do escopo do projeto.

Dentre essas 1200 imagens, 1047 estavam na resolução de 2124x2056px, já as outras 153 estavam na de 1444x1444px. Ao ser feito um teste rápido com as 153 imagens foi observado que a quantidade de imagens estava muito desbalanceada, 140 eram de olhos míopes enquanto apenas 13 eram de olhos sem miopia como mostrado na Tabela 3.1. O resultado do desbalanceamento pode ser observado na métrica macro-avg, mostrado na Figura 3.2. Portanto mesmo com uma boa acurácia percebia-se que essa métrica não significava muito nesse caso. Portanto deveria ser analisado o resto do dataset.



	precision	recall	f1-score	support
Myopic	0.90	1.00	0.95	28
Normal	0.00	0.00	0.00	3
accuracy			0.90	31
macro avg	0.45	0.50	0.47	31
weighted avg	0.82	0.90	0.86	31

Figura 3.2: Resultados da classificação em uma divisão desbalanceada do dataset.

As outras 1047 imagens estavam igualmente distribuídas entre míope e não míope, sendo 523 imagens de olhos míopes e 524 de olhos não míopes, como se pode observar na Tabela 3.1. portanto foi decidido utilizar apenas as imagens em maior quantidade sem resize para não haver perda de qualidade além de estarem mais bem distribuídas entre as duas classes a serem usadas no classificador: "Míope"e "Não míope".

Resolução	**Imagens não míopes**	**Imagens míopes**	**Total de imagens**
2124x2056p	524	523	1047
1444x1444p	13	140	153

Tabela 3.1: Comparação das imagens do dataset.

As imagens do Dataset já estavam separadas em 3 pastas diferentes: Test, train e validation. Com 400 imagens em cada. Porém para um maior grau de aleatoriedade nos testes e um controle maior sobre a quantidade de imagens que seria utilizada em cada etapa, todas as imagens foram unificadas em uma mesma pasta. Assim possibilitando que a divisão do dataset de treino e teste possa ser feita aleatoriamente entre todas as imagens e não apenas usar as imagens que já estavam separadas. Essa unificação também permitiu que a proporção de imagens separadas para teste e treino fosse modificada.

Podem-se visualizar exemplos das imagens do dataset observando a Figura 3.3 e a Figura 3.4. Nelas é possível observar alguns dos problemas causados pela alta miopia, porém é importante mostras as duas comparações para mostrar que não é algo tão simples de se observar em todos os casos.

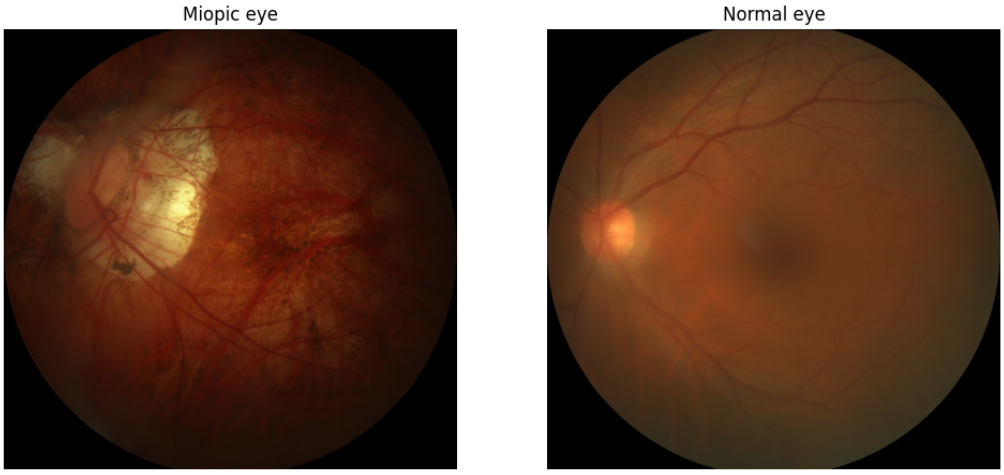


Figura 3.3: Exemplo de imagens de olhos míope e normal retirados do dataset.

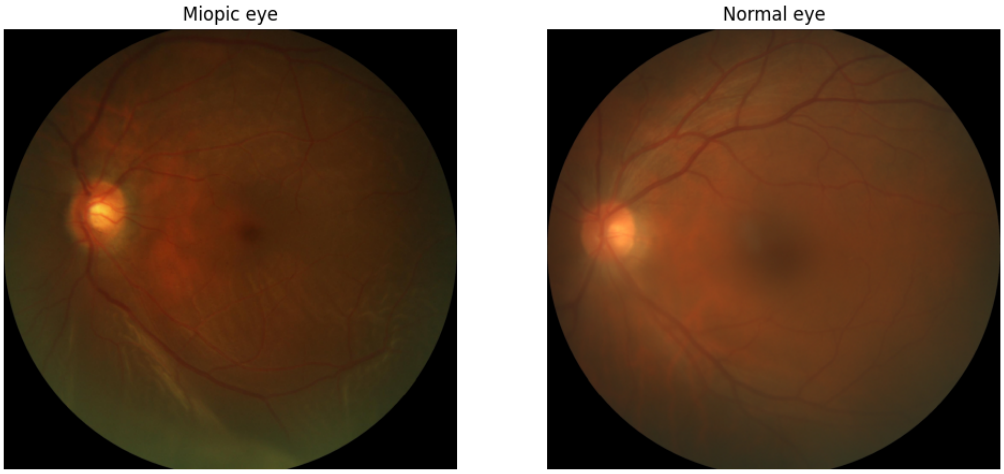


Figura 3.4: Mais um exemplo de imagens de olhos míope e normal retirados do dataset.

3.2 Pré-processamento

Na etapa do pré-processamento foi usada a técnica de greyscaling das imagens para uso em todos os casos no algoritmo de extração de features e nos classificadores, pois era fundamental e necessário para o funcionamento dos mesmos. A Figura 3.5 mostra um exemplo de uma das imagens em greyscale.

Figura 3.5: Comparação de imagem do dataset colorida e grayscale.

3.3 Extração de features

Como neste projeto serão utilizados modelos de classificadores tradicionais, será necessário fazer a extração das features das imagens antes do treinamento para se obter uma melhor precisão ao usá-las de entrada no classificador, já que estes tipos de classificadores mais simples não tem a capacidade de aprender representações complexas durante o treinamento como os modelos de deep learning tem. Então usando o algoritmo HOG já mencionado foi feita a extração das características da imagem, resultando em uma imagem como a da Figura 3.6.

Após extração das features da imagem, elas são salvas em uma lista, enquanto a label (rótulo simbolizando se a imagem é de um olho com ou sem miopia) correspondente à essa imagem é salva em outra lista. Isso é feito para cada uma das imagens do dataset. Após conclusão da extração, essas listas são convertidas em npArrays com a finalidade de otimização, já que

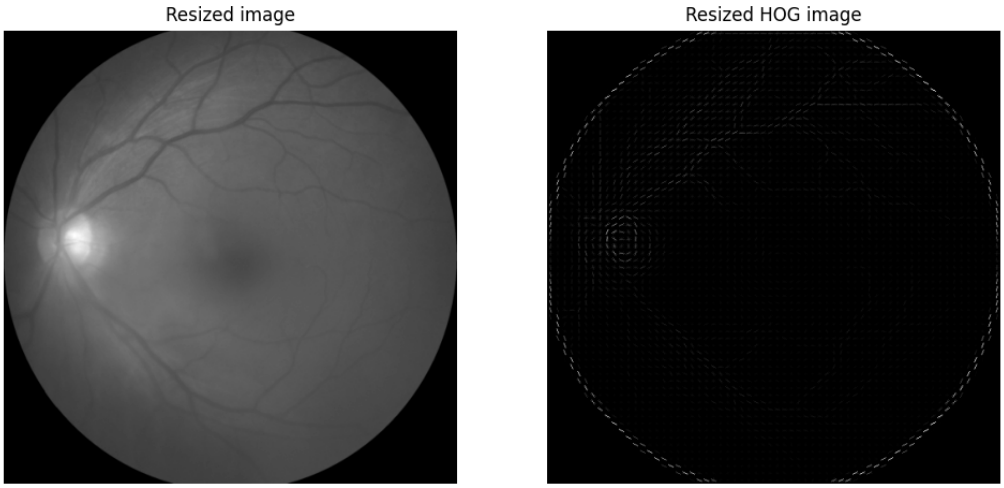


Figura 3.6: Comparação de imagem do dataset vs imagem HOG.

npArrays são otimizados tanto para operações numéricas quanto para dados multidimensionais vindo de imagens [30].

3.4 Treinamento e classificação

Para uma comparação justa, os npArrays com as features e labels foram divididos em treino e teste, 20% para teste e 80% para treino, aleatoriamente pelo uso da função "train_test_split"da biblioteca sklearn com o parâmetro stratify=labels, que faz com que a proporção de imagens das duas classes se mantenha em ambas as divisões. Fazendo, assim, com que tanto para treinamento quanto para teste exista uma quantidade similar de imagens para cada uma das classes, e evitando assim que um caso haja um classificador tendencioso a uma das classes, ele se saia melhor na comparação por conta disso.

Além disso, os npArrays foram salvos para que todos os classificadores sejam testados com os mesmos dados, assim tendo uma comparação mais justa, e impedindo que um tenha sido treinado com um conjunto de dados que acabe sendo mais vantajoso do que o do outro. Então após feita a divisão dos dados, os 5 classificadores foram, um a um, instanciados e treinados com esses mesmos dados.

3.5 Otimização de hiperparâmetros

Após todos os classificadores terem finalizado suas predições, foi feita uma tunagem de hiperparâmetros com cada um deles, de forma a tentar melhorar ainda mais os resultados obtidos pelos mesmos e também comparar os classificadores antes e depois dessa otimização. Para isso foi usado o GridSearchCV dividindo os dados de treino em 5 partes para fazer a validação cruzada. Foi determinada, então, uma grade de hiperparâmetros para cada um dos classificadores, e quando o melhor resultado era o maior ou o menor valor da grade, era feita outra tentativa de otimização com mais valores na grade.

3.5.1 SVC

Para a otimização do Support Vector Classifier foram testados os hiperparâmetros: C, gamma e kernel, e os valores colocados na grade podem ser observados na Tabela 3.6. O parâmetro C é a margem de erro que o classificador pode tolerar, então se for muito pequeno ele pode não se adequar muito ao modelo mas se for grande demais ele pode acabar causando overfitting. O parâmetro gamma é um coeficiente que indica o alcance da influência de um ponto do kernel caso ele não seja linear. E o kernel define o tipo de hiperplano que será usado para dividir as classes.

C	**Gamma**	**Kernel**
00.1	0.001	linear
0.1	0.01	poly
1	0.1	rbf
10	1	sigmoid
100		
1000		

Tabela 3.2: Hiperparâmetros usados na otimização do SVC.

3.5.2 RandomForestClassifier

Para a otimização do Random Forrest Classifier foram variados os hiperparâmetros: n_estimators (n), max_depth (profundidade), min_samples_split (split), min_samples_leaf (folha), max_features (features) e bootstrap, como se pode observar na Tabela 3.4. O primeiro e o segundo parâmetros se referem à quantidade e profundidade máxima, respectivamente, das árvores de decisão a serem criadas na floresta. O parâmetro min_samples_leaf representa o número mínimo de

amostras para que um nó final, ou seja uma folha, seja criado. O min_samples_split representa o número mínimo de amostras para que um nó seja dividido em ramos filhos. O max_features representa o maior número de features consideradas na hora de procurar a melhor divisão de cada nó. E por fim o hiperparâmetro bootstrap define se o modelo usará uma amostragem com ou sem substituição.

n	**Profundidade**	**split**	**folha**	**features**	**bootstrap**
100	None	liblinear	2	1	auto
200	10	saga	5	2	sqrt
300	20	lbfgs	10	4	log2
	30				

Tabela 3.3: Hiperparâmetros usados na otimização do Random Forrest classifier.

3.5.3 KNeighborsClassifier

Para a otimização do K-nearest neighbors Classifier foram variados os hiperparâmetros: N_neighbors (K), weights (peso), metric (métrica) e algorithm (algoritmo), como se pode observar na Tabela 3.4. O primeiro hiperparâmetro é o K, ou seja, a quantidade de vizinhos considerados no cálculo, o segundo, o peso, tem a função de considerar se todos os vizinhos pesam igual no cálculo ou se a distância também será considerada. O hiperparâmetro de métrica define qual métrica de distância será utilizad. E o último hiperparâmetro é o algoritmo que será utilizado para buscar os vizinhos.

K	**peso**	**métrica**	**algoritmo**
3	uniforme	euclidean	auto
5	distance	manhattan	ball_tree
7		minkowski	kd_tree
9			brute

Tabela 3.4: Hiperparâmetros usados na otimização do KNeighborsClassifier.

3.5.4 LogisticRegression

Para a otimização da regressão logística foram variados os hiperparâmetros: C, penalty (pen), solver (sol), max_iter (max), class_weight (peso), tol e l1_ratio (ratio), como se pode observar na Tabela 3.5. Onde C é o inverso da regularização, e similar ao C do support vector machine valores muito baixos podem não se ajustar tão bem aos dados e valores muito altos

podem causar overfitting. Penalty é o tipo de penalidade aplicada. Solver é o algoritmo de otimização que será utilizado. Max_iter é a quantidade máxima de iterações para os algoritmos convergirem. Class_weight é um hiperparâmetro que serve para quando o dataset está desbalanceado, mas foi utilizado para testar se a diferença de uma imagem seria relevante. O hiperparâmetro tol está ligado com a tolerância para convergência do algoritmo e pode ser diminuido caso o algoritmo não esteja convergindo. O l1_ratio é um hiperparâmetro usado apenas quando a penalidade for definida como 'elasticnet', e é usado para escolher valores entre l1 e l2 para a penalidade.

C	**pen**	**solv**	**max**	**peso**	**tol**	**ratio**
0.01	l1	liblinear	10	None	10^{-6}	0.0001
0.1	l2	saga	25	balanced	10^{-5}	0.001
1	elasticnet	lbfgs	50		10^{-4}	0.01
10	none		100		10^{-3}	0.1
100			200		10^{-2}	0.5
1000			500			0.7
10000						0.0

Tabela 3.5: Hiperparâmetros usados na otimização da regressão logística.

3.5.5 GaussianNB

Para a otimização do classificador gaussiano Naive Bayes, por ser mais simples, havia apenas 1 hiperparâmetro, o var_smoothing, os valores testados podem ser vistos na **??**. Esse hiperparâmetro é um valor pequeno que é somado às variâncias das features para suavizar essa variância.

Var_smoothing
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}

Tabela 3.6: Hiperparâmetros usados na otimização do Naive Bayes.

4 Resultados

Nessa sessão serão mostrados os resultados obtidos por cada um dos classificadores tanto antes quanto após a tunagem de hiperparâmetros e será feita uma comparção tanto do antes e depois da otimização quanto entre os diferentes classificadores.

4.1 SVM

A classificação do support vector machine já é muito boa sem nenhuma alteração, mas após a otimização dos hiperparâmetros ele teve um desempenho superior e conseguiu acertar todos os verdadeiros negativos do dataset de teste. Na Tabela 4.1 e na Figura 4.1 pode-se observar as diferenças nos resultados antes e depois do ajuste de hiperparâmetros.

Não só a acurácia melhorou em 5% como aumentou bastante a precisão do modelo, 8% de aumento. Também é importante ressaltar a taxa de 100% recall obtida por esse classificador tanto antes quanto após o tune. Os valores dos hiperparâmetros antes e depois do ajuste podem ser visualizados na Tabela 4.2.

Ajustado	**Acurácia**	**Precisão**	**Recall**	**F1-score**
Sim	96%	93%	100%	96%
Não	91%	85%	100%	92%

Tabela 4.1: Comparação dos resultados do SVM antes e depois da otimização de hiperparâmetros.

Ajustado	**C**	**Gamma**	**Kernel**
sim	1	1	linear
não	1	scale	rbf

Tabela 4.2: Valores dos hiperparâmetros usados antes e depois do ajuste do SVM.

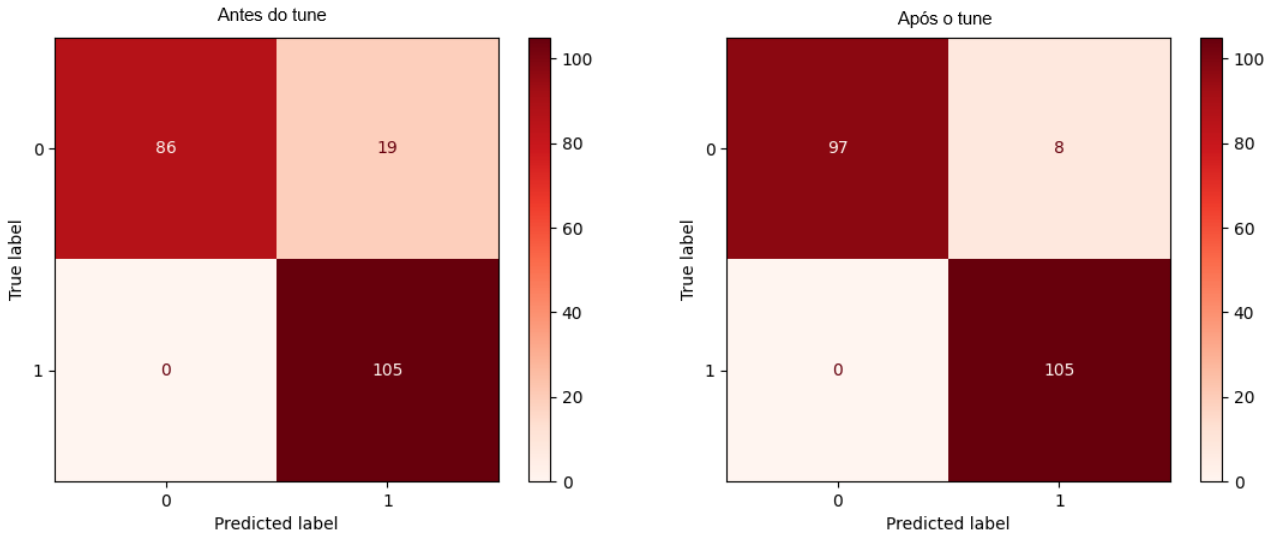


Figura 4.1: Matriz de confusão dos resultados do SVM antes e depois da tunagem de Hiperparâmetros

4.2 Random Forest

A classificação do random forrest foi boa e passou dos 90% de acurácia. Após ajuste dos hiperparâmetros, porém, não houve aumento significativo em nenhuma métrica, a acurácia e a precisão aumentaram apenas 1%. Pode-se observar as diferenças nos resultados antes e depois da otimização de hiperparâmetros na Tabela 4.3 e na Figura 4.2. Os valores dos hiperparâmetros antes e depois do ajuste podem ser visualizados na Tabela 4.4.

Ajustado	**Acurácia**	**Precisão**	**Recall**	**F1-score**
Sim	93%	89%	97%	93%
Não	92%	88%	97%	92%

Tabela 4.3: Comparação dos resultados do RandomForestClassifier antes e depois da otimização de hiperparâmetros.

Ajustado	**n**	**Profundidade**	**split**	**folha**	**features**	**bootstrap**
Sim	300	None	sqrt	5	1	False
Não	100	None	sqrt	2	1	True

Tabela 4.4: Valores dos hiperparâmetros usados antes e depois do ajuste do classificador Random Forrest.

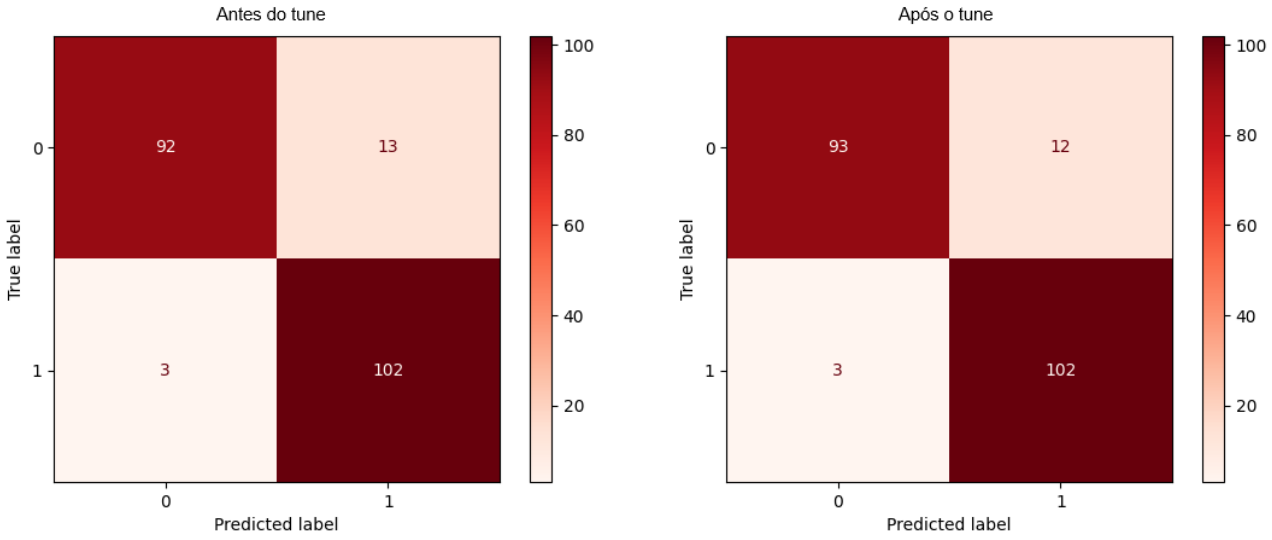


Figura 4.2: Matriz de confusão dos resultados do classificador Random Forest antes e depois da tunagem de Hiperparâmetros

4.3 KNeighborsClassifier

A classificação do KNeighborsClassifier é relativamente boa, não chega nos 90% de acurácia, mas chega bem perto. Porém após a otimização dos hiperparâmetros não houve aumento significativo na acurácia do modelo. Na Tabela 4.5 e na Figura 4.3 é possível observar as diferenças nos resultados antes e depois da otimização de hiperparâmetros.

Após a otimização dos hiperparâmetros houve apenas um caso de verdadeiro negativo a mais, porém um caso de falso negativo a mais, o que mostra que a precisão aumentou, porém o recall diminuiu, o classificador está acertando mais verdadeiros negativos e menos verdadeiros positivos. O algoritmo de gridsearch considerou isso como uma melhora pois o resultado foi mais consistente dentro da validação cruzada, o que pode implicar que os resultados do classificador serão mais consistentes em um cenário real. Os valores dos hiperparâmetros antes e depois do ajuste podem ser visualizados na Tabela 4.6.

Ajustado	**Acurácia**	**Precisão**	**Recall**	**F1-score**
Sim	89%	83%	96%	89%
Não	88%	81%	99%	89%

Tabela 4.5: Comparação dos resultados do KNeighborsClassifier antes e depois da otimização de hiperparâmetros.

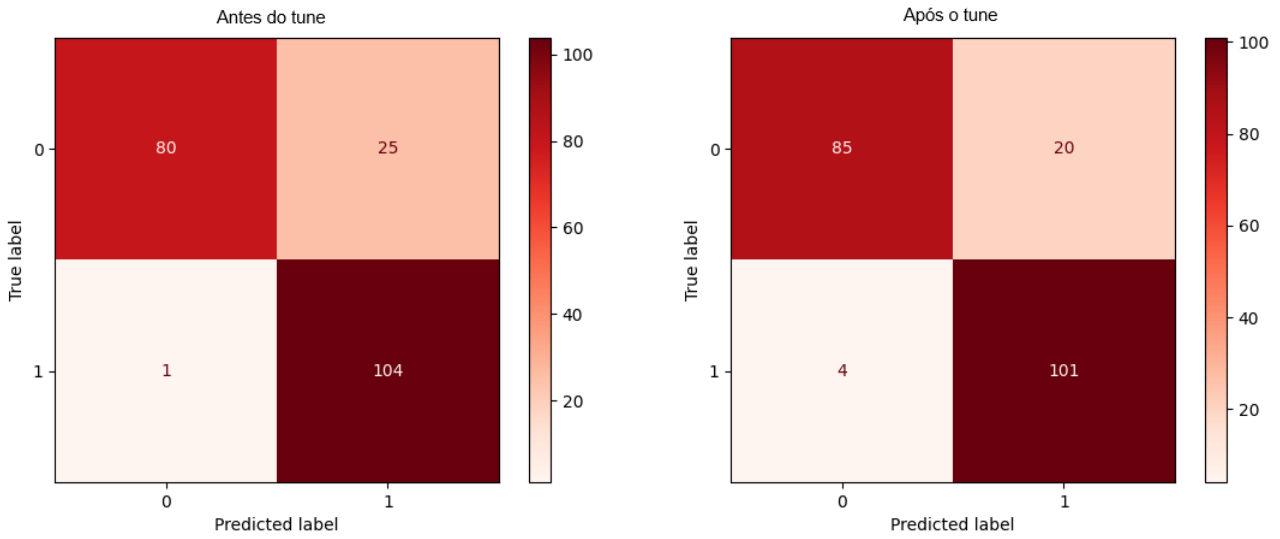


Figura 4.3: Matriz de confusão dos resultados do classificador KNeighborsClassifier antes e depois da tunagem de Hiperparâmetros

Ajustado	**K**	**peso**	**métrica**	**algoritmo**
Sim	3	uniform	euclidean	auto
Não	5	uniform	minkowski	auto

Tabela 4.6: Valores dos hiperparâmetros usados antes e depois do ajuste do classificador K-Nearest Neighbours.

4.4 LogisticRegression

A classificação da regressão logística obteve um ótimo resultado, acima de 95% de acurácia. Porém após a otimização dos hiperparâmetros não houve aumento significativo nas métricas do modelo. Na Tabela 4.7 e na Figura 4.4 pode-se observar as diferenças nos resultados antes e depois da otimização de hiperparâmetros.

Após a otimização dos hiperparâmetros houve mais uma vez um caso de verdadeiro negativo a mais e um caso de falso negativo a mais, aumentando a precisão e diminuindo o recall, porém não foi uma mudança muito significativa. Mais uma vez os parâmetros ajustados foram considerados melhor por trazerem uma capacidade maior de generalização do modelo de acordo com a validação cruzada. Os valores dos hiperparâmetros antes e depois do ajuste podem ser visualizados na Tabela 4.8.

Ajustado	Acurácia	Precisão	Recall	F1-score
Sim	96%	94%	99%	96%
Não	96%	93%	100%	96%

Tabela 4.7: Comparação dos resultados do LogisticRegression antes e depois da otimização de hiperparâmetros.

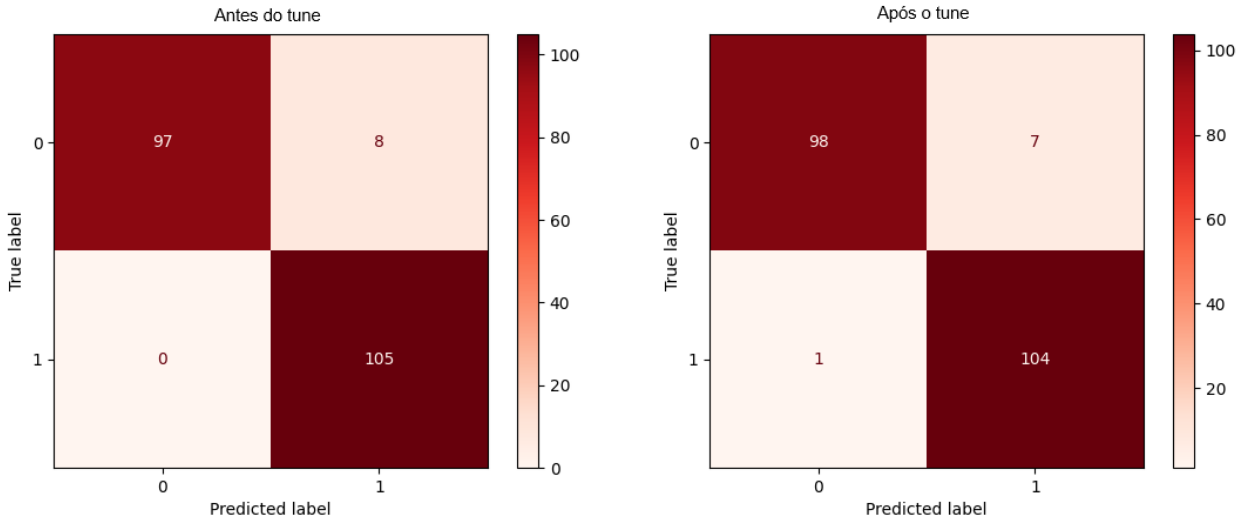


Figura 4.4: Matriz de confusão dos resultados do classificador LogisticRegression antes e depois da tunagem de Hiperparâmetros

Ajustado	C	pen	solv	max	peso	tol
Sim	100	l2	lbfgs	100	None	10^{-5}
Não	1	l2	lbfgs	100	None	10^{-4}

Tabela 4.8: Valores dos hiperparâmetros usados antes e depois do ajuste da Regressão Logística.

4.5 GaussianNB

A classificação do GaussianNB obteve um resultado que a pesar de não ser o melhor, quando otimizado, conseguiu superar os 90% de acurácia. Após o ajuste do hiperparâmetro houve um aumento em todas as métricas do modelo. A precisão aumentou 3%, a acurácia 5% e o recall 9%. Isso pode ser observado na Tabela 4.9 e na Figura 4.5. O valore do hiperparâmetro antes e depois do ajuste pode ser visualizado na Tabela 4.10.

Ajustado	Acurácia	Precisão	Recall	F1-score
Sim	92%	86%	100%	93%
Não	87%	83%	91%	87%

Tabela 4.9: Comparação dos resultados do GaussianNB antes e depois da otimização de hiperparâmetros.

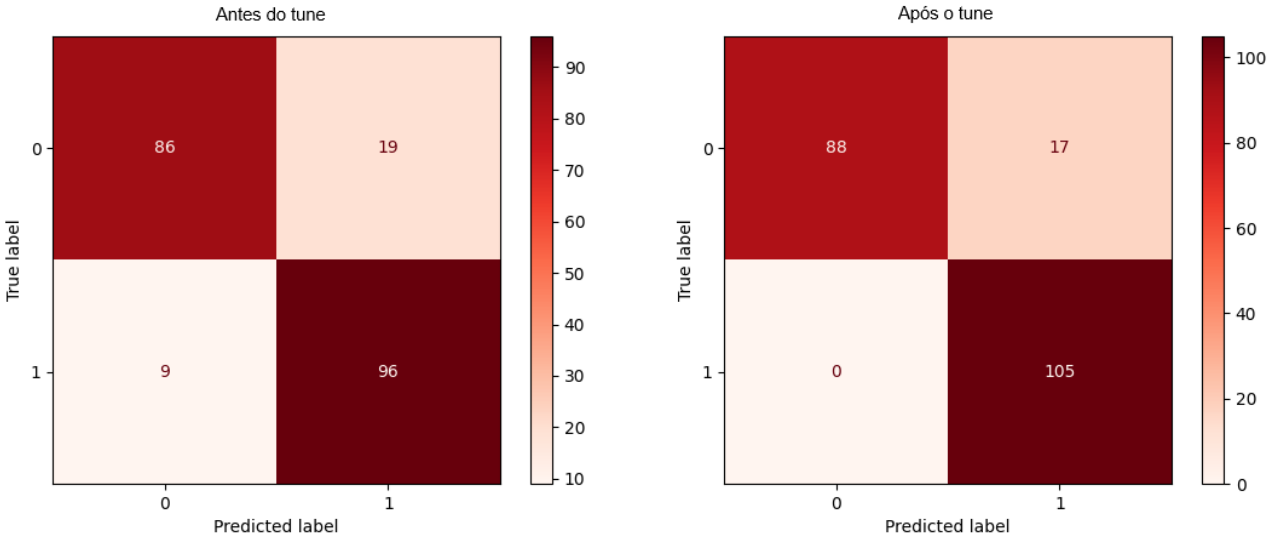


Figura 4.5: Matriz de confusão dos resultados do classificador GaussianNB antes e depois da tunagem de Hiperparâmetros

Ajustado	**Var_smoothing**
Sim	10^{-4}
Não	10^{-9}

Tabela 4.10: Valores dos hiperparâmetros usados antes e depois do ajuste do classificador Naive Bayes.

4.6 Comparação final

Observando juntamente todos os resultados na Tabela 4.11 é possível ver que os melhores resultados foram do support vector machine e da regressão logística. E apesar dos dois resultados estarem muito próximos o SVC demonstrou melhor capacidade de otimização, e por ser um classificador mais robusto deve ter um certo potencial para melhorar ainda mais ao usar outros métodos. Porém a regressão logística se mostrou uma ótima opção computacionalmente menos custosa, principalmente considerando que o algoritmo demorou quase 1/5 do tempo para treinamento e predição. Mas apesar disso para ser usado em um cenário real dentro de um hospital o SVC ainda seria o melhor modelo.

Classificador	**Ajustado**	**Acurácia**	**f1**	**Precisão**	**Recall**
SVM	Sim	96%	96%	93%	100%
LR	Sim	96%	96%	94%	99%
LR	Não	96%	96%	93%	100%
RF	Sim	93%	93%	89%	97%
NB	Sim	92%	93%	86%	100%
RF	Não	92%	92%	88%	97%
SVM	Não	91%	92%	85%	100%
KN	Sim	89%	89%	83%	96%
KN	Não	88%	89%	81%	99%
NB	Não	87%	87%	83%	91%

Tabela 4.11: Comparação de todos os resultados ordenados por acurácia e depois por f1-score.

5 Conclusão e trabalhos futuros

5.1 Conclusão

Neste trabalho foi utilizado um dataset de imagens de fundo de olho disponibilizado pelo Simpósio Internacional de Processamento Biomédico de Imagens da IEEE. As imagens desse dataset foram pré-processadas e carácterísticas dessas imagens foram extraídas, usando o algorítmo HOG, para serem usadas de entrada em diferentes modelos de classificadores.

Foram usados 5 modelos de classificadores do SKlearn: O Random Forrest, o Support Vector Machine, a Regressão Logística, o K-Nearest Neighbours e o Gaussian Naive Bayes. Na classificação de imagens de fundo de olho a fim de identificar olhos míopes pela detecção de certos problemas causados por ela. Cada um desses classificadores teve também seus hiperparâmetros ajustados, usando as técnicas de Gridsearch e Validação Cruzada para encontrar os valores que maximizaram a acurácia do modelo. E por fim os resultados de todos os modelos tanto antes quanto após a otimização foram comparados lado a lado.

Após feitas todas as comparações, foram encontrados ótimos resultados, alguns acima de 95% de acurácia. Conclui-se então que, com base nos resultados obtidos, o modelo do classificador Support vector machine com os hiperparâmetros ajustados e o modelo de Regressão Logística seriam as melhores escolhas para serem utilizados em uma aplicação real. Graças à sua alta precisão e à sua grande capacidade de generalização. Fazendo assim com que se mantenha sua alta taxa de acerto mesmo quando usado com novos dados.

5.2 Trabalhos futuros

Baseando-se neste trabalho, é possível notar que ficaram abertas algumas possibilidades de trabalhos que possam ser futuras contribuições:

- Modificar variáveis que não foram testadas, como outros algoritmos de extração de fea-

tures, modificação dos parâmetros do HOG ou o uso de imagens coloridas.

- Comparar os resultados destes classificadores com uma rede neural.
- Expandir estes classificadores para fazer a detecção de outros problemas oculares em conjunto com a miopia.
- Embarcar este projeto em algum aparelho acoplado à máquina de exames para fazer com que uma prévia do resultado já saia imediatamente quando o exame for feito.

Referências Bibliográficas

[1] Lenscope, "As principais causas da miopia que você não sabia," 2020. Acessado em: 10 de outubro de 2024, disponível em https://lenscope.com.br/blog/causas-da-miopia/.

[2] H. F. Huazhu Fu *et al.*, "Palm: Open fundus photograph dataset with pathologic myopia recognition and anatomical structure annotation," 2019. Acessado em: 2 de outubro de 2024, disponível em https://arxiv.org/pdf/2305.07816.

[3] Scikit-image, "Histogram of oriented gradients," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html.

[4] A. Singh, "Feature engineering for images: A valuable introduction to the hog feature descriptor," 2024. Acessado em: 2 de outubro de 2024, disponível em https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor.

[5] Scikit-learn, "Cross-validation: evaluating estimator performance," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/cross_validation.html.

[6] L. Merkl, "Halting nearsightedness epidemic goal of uh vision scientist," 2017. Acessado em: 2 de outubro de 2024, disponível em https://www.uh.edu/news-events/stories/2017/March/031017SmithNIHmyopia-kids.php.

[7] T. Ferreira, "Brasil terá quase o dobro de pessoas com alta miopia em 2040," 2022. Acessado em: 2 de outubro de 2024, disponível em https://olhardigital.com.br/2022/11/16/medicina-e-saude/brasil-tera-quase-o-dobro-de-pessoas-com-alta-miopia-em-2040/.

[8] M. Alvarez *et al.*, "Early detection of refractive errors by photorefraction at school age," 2022. Acessado em: 10 de outubro de 2024, disponível em https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9739333/.

[9] E. Coelho, "Inteligência artificial na medicina: perspectivas, benefícios e desafios," 2024. Acessado em: 2 de outubro de 2024, disponível em https://med.estrategia.com/portal/atualidades/inteligencia-artificial-na-medicina-perspectivas-beneficios-e-desafios/.

[10] R. L. Yahan Yang *et al.*, "Automatic identification of myopia based on ocular appearance images using deep learning," 2020. Acessado em: 20 de outubro de 2024, disponível em https://pmc.ncbi.nlm.nih.gov/articles/PMC7327333/.

[11] M. C. Danilo Leite *et al.*, "Machine learning automatic assessment for glaucoma and myopia based on corvis st data," 2021. Acessado em: 20 de outubro de 2024, disponível em https://www.sciencedirect.com/science/article/pii/S1877050921022596.

[12] D. J. C. Yong Chan Kim *et al.*, "Machine learning prediction of pathologic myopia using tomographic elevation of the posterior sclera," 2021. Acessado em: 20 de outubro de 2024, disponível em https://www.nature.com/articles/s41598-021-85699-0.

[13] D. M. Jampaulo, "Alta miopia: conheça os riscos e os cuidados necessários," 2021. Acessado em: 10 de outubro de 2024, disponível em https://vivaoftalmologia.com.br/alta-miopia-conheca-os-riscos-e-os-cuidados-necessarios/.

[14] Scikit-image, "Rgb to grayscale," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_rgb_to_gray.html.

[15] Wikipedia, "Histogram of oriented gradients," 2024. Acessado em: 2 de outubro de 2024.

[16] Scikit-learn, "Svc," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[17] Scikit-learn, "Gaussiannb," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html.

[18] Scikit-learn, "Kneighborsclassifier," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.

[19] Scikit-learn, "Randomforestclassifier," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[20] Scikit-learn, "Logisticregression," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[21] Scikit-learn, "precision_score," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score.

[22] Scikit-learn, "recall_score," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score.

[23] Scikit-learn, "f1_score," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score.

[24] Scikit-learn, "Metrics and scoring: quantifying the quality of predictions," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/model_evaluation.html.

[25] Scikit-learn, "accuracy_score," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score.

[26] Scikit-learn, "confusion_matrix," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix.

[27] Scikit-learn, "Gridsearchcv," 2024. Acessado em: 2 de outubro de 2024, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[28] H. F. Huazhu Fu *et al.*, "Palm: Pathologic myopia challenge," 2019. Acessado em: 2 de outubro de 2024, disponível em https://ieee-dataport.org/documents/palm-pathologic-myopia-challenge.

[29] IEEE, "Thank you for attending isbi 2019!," 2019. Acessado em: 2 de outubro de 2024, disponível em https://biomedicalimaging.org/2019/.

[30] G. for Geeks, "Python lists vs numpy arrays," 2023. Acessado em: 2 de outubro de 2024.