Classificação de Raças de Gatos e Cães com Aprendizado Supervisionado

Luiz Felipe de Souza Silva[1]

[1] Departamento de Engenharia da Computação (DCA) – Universidade Federal do Rio Grande do Norte (UFRN)

lf06092004@gmail.com.br, luiz.felipe.souza.082@ufrn.edu.br

Abstract. *This report presents the development of a breed classifier for cats and dogs as part of the practical project for the Machine Learning course (IMD1101). The project applied supervised learning techniques to classify the following breeds: Egyptian Mau, Pomeranian, Ragdoll, and Shiba Inu. Algorithms such as k-Nearest Neighbors, Decision Tree, Naive Bayes, and Artificial Neural Networks were implemented, along with the construction of classifier ensembles. The goal is to analyze the individual and combined performance of the methods, as well as the contribution of preprocessing and statistical tests to the robustness of the results.*

Resumo. *Este relatório apresenta o desenvolvimento de um classificador de raças de gatos e cães como parte do trabalho prático da disciplina de Aprendizado de Máquina (IMD1101). O projeto aplicou técnicas de aprendizado supervisionado para a classificação das seguintes raças: Egyptian Mau, Pomeranian, Ragdoll e Shiba Inu. Foram implementados algoritmos como k-NN, Árvore de Decisão, Naive Bayes e Redes Neurais Artificiais, além da formação de comitês de classificadores. O objetivo é analisar o desempenho individual e combinado dos métodos, bem como a contribuição do pré-processamento e dos testes estatísticos para a robustez dos resultados.*

1. Introdução

Aprendizado de Máquina é um dos tópicos mais relevantes da Inteligência Artificial atualmente, sendo amplamente aplicado em diversas áreas devido à crescente demanda por automatização de tarefas e apoio à tomada de decisão [Sarker 2021]. Nesse contexto, a disciplina de Aprendizado de Máquina (IMD1101) propõe um projeto prático final com o intuito de aplicar os conhecimentos e algoritmos estudados ao longo do curso.

Este relatório tem como objetivo apresentar os resultados obtidos com a aplicação de diferentes técnicas supervisionadas na construção de um classificador de raças de gatos e cães. A classificação automática de raças pode ter aplicações úteis em diferentes domínios, como: recomendações específicas para cuidados veterinários, escolha de métodos adequados de higiene em pet shops, ou ainda na identificação automatizada em sistemas de monitoramento.

A tarefa de classificar animais por meio de imagens apresenta desafios típicos da área de visão computacional, como variações de iluminação, ângulo de captura, poses e semelhanças visuais entre espécies, o que torna a extração e classificação de características mais complexas para os algoritmos de aprendizado de máquina.

A abordagem adotada neste trabalho envolve etapas de pré-processamento das imagens, extração de características, aplicação de algoritmos supervisionados — como k-Nearest Neighbors (k-NN), Naive Bayes, Árvore de Decisão e Redes Neurais Artificiais — e a formação de comitês de classificadores. Além disso, são realizados testes estatísticos para verificar a significância dos resultados obtidos.

Este relatório está organizado da seguinte forma: a Seção 2 apresenta a base de dados utilizada; a Seção 3 descreve o processo de pré-processamento; a Seção 4 detalha os algoritmos supervisionados aplicados; a Seção 5 aborda os comitês de classificadores; a Seção 6 apresenta os testes estatísticos realizados; e, por fim, a Seção 7 discute as conclusões obtidas a partir da análise dos resultados. .

2. Base de Dados

A base de dados utilizada neste trabalho é pública e foi disponibilizada pelo Oxford Visual Geometry Group (VGG) por meio do dataset Oxford-IIIT Pet[1]. O conjunto completo contém milhares de imagens de cães e gatos, com 37 raças rotuladas, e apresenta variações significativas de iluminação, angulação e pose, sendo amplamente utilizado em tarefas de classificação de imagens na área de visão computacional.

(a) Egyptian Mau (b) Ragdoll

(c) Pomeranian (d) Shiba Inu

Figura 1. Exemplos das raças utilizadas no projeto.

[1] https://www.robots.ox.ac.uk/~vgg/data/pets/

Para este projeto, foi selecionado um subconjunto com 4 raças distintas: *Egyptian Mau* e *Ragdoll* (gatos), *Pomeranian* e *Shiba Inu* (cães). Com isso, formulou-se um problema de classificação multiclasse com quatro categorias.

A base de dados já apresenta uma distribuição equilibrada para as raças escolhidas, contendo 200 imagens por classe, totalizando 800 instâncias. Essa estratificação automática garante um conjunto balanceado para o treinamento e avaliação dos algoritmos supervisionados, reduzindo viés no aprendizado.

Visualmente, as raças selecionadas apresentam diferenças claras: os gatos possuem características bem distintas entre si, enquanto os cães, apesar de apresentarem coloração semelhante, diferenciam-se por altura, porte, formato dos olhos e tipo de pelagem. A Figura 1 ilustra exemplos dessas quatro raças.

3. Pré-processamento e Extração de Características

Na etapa de pré-processamento, foram aplicadas técnicas para construir bases de dados mais limpas e adequadas ao aprendizado de máquina. Durante a análise inicial, observou-se que 5 imagens da classe *Egyptian Mau* estavam corrompidas, sem apresentar qualquer informação de pixel. Essas imagens foram descartadas, resultando em um total de 795 instâncias no conjunto final: 195 de *Egyptian Mau* e 200 de cada uma das demais raças (*Ragdoll*, *Pomeranian* e *Shiba Inu*).

Modelos de aprendizado de máquina tradicionais, como os utilizados neste trabalho, não operam diretamente com imagens brutas. Por isso, é necessário realizar a extração de características por meio de extratores. Por isso, é necessário realizar a extração de características por meio de extratores. Neste trabalho, foram utilizados dois métodos distintos: redes convolucionais (CNN) pré-treinadas e histogramas de gradientes orientados (HOG). Antes da extração, aplicaram-se técnicas como redimensionamento das imagens e normalização dos pixels.

As imagens foram redimensionadas para dois formatos: 128×128 e 256×256 pixels, com o objetivo de avaliar o impacto do tamanho padronizado na qualidade da classificação. Para a aplicação do HOG, as imagens também foram convertidas para escala de cinza, a fim de otimizar a extração de contornos e bordas. Já no caso das CNNs, foram utilizadas arquiteturas pré-treinadas, especificamente a *VGG16* e *VGG19*, conhecidas por sua eficácia em tarefas de classificação de imagens.

As bases extraídas foram salvas em um padrão de nome para facilitar a visualização das entradas que foram utilizadas na extração de características pelo HOG e CNNs, note na Tabela 1 e Tabela 2.

Tabela 1. Padrão de nomenclatura das bases extraídas com HOG

Nome da Base	Tamanho da Imagem	Pixels por Célula
HOG_128_20x20	128x128	20x20
HOG_128_16x16	128x128	16x16
HOG_256_20x20	256x256	20x20
HOG_256_16x16	256x256	16x16

Nesta mesma etapa, os conjuntos de dados extraídos (via HOG e CNNs) foram ini-

Tabela 2. Padrão de nomenclatura das bases extraídas com CNNs

Nome da Base	Modelo	Tamanho da Imagem	Pooling
CNN_VGG16_256_max	VGG16	256x256	Max Pooling
CNN_VGG16_256_avg	VGG16	256x256	Average Pooling
CNN_VGG16_128_max	VGG16	128x128	Max Pooling
CNN_VGG16_128_avg	VGG16	128x128	Average Pooling
CNN_VGG19_256_max	VGG19	256x256	Max Pooling
CNN_VGG19_256_avg	VGG19	256x256	Average Pooling
CNN_VGG19_128_max	VGG19	128x128	Max Pooling
CNN_VGG19_128_avg	VGG19	128x128	Average Pooling

cialmente avaliados por meio do classificador *k-Nearest Neighbors* (k-NN), com o intuito de identificar o melhor extrator. Indiscutivelmente, as CNNs pré-treinadas demonstraram desempenho superior ao HOG, com maior capacidade de generalização frente a variações de iluminação, ângulo e coloração semelhante.

A Figura 2 apresenta a comparação das acurácias entre os dois extratores quando utilizados como entrada para o classificador *k-NN*, o gráfico mostra acurácia média da melhor base de HOG em comparação com a pior base CNN.

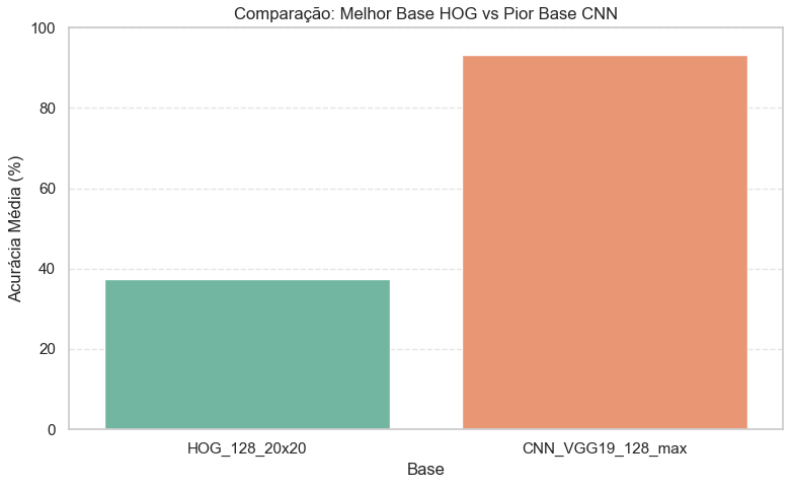

Figura 2. Acurácia da melhor base de HOG com a pior base de CNN

Ademais, após o treinamento inicial com k-NN, foi aplicado o redutor de dimensionalidade Principal Component Analysis (PCA) nas bases de dados, a fim de avaliar sua relevância para o projeto. Buscou-se responder a questões como: a técnica reduziria o custo computacional? Uma base compactada manteria ou superaria o desempenho da base completa? Verificou-se que a redução de dimensionalidade foi benéfica, validando a sua importância. Nas 6 melhores bases extraídas por CNN, o uso do PCA afetou positivamente o treinamento, aumentando em alguns pontos percentuais a acurácia. É possível visualizar a mudança após da acurácia após aplicação do PCA na Tabela 3. A base que obteve o maior ganho percentual foi à base que utilizou a CNN pré treinada VGG 16 com imagens padronizadas em 256x256 pixels com entradas de pooling “max”, 1,09

Tabela 3. Comparação da acurácia média antes e depois da aplicação do PCA

Base	Sem PCA	Com PCA	Diferença (%)
CNN_VGG16_256_max	98,05%	99,14%	+1,09
CNN_VGG16_256_avg	98,37%	98,80%	+0,43
CNN_VGG19_128_max	93,17%	93,85%	+0,68
CNN_VGG19_128_avg	93,82%	94,29%	+0,47
CNN_VGG19_256_max	98,49%	98,91%	+0,42
CNN_VGG19_256_avg	97,22%	98,26%	+1,04

Para visualizar o impacto positivo da aplicação do PCA no desempenho, foi construída uma matriz de dispersão (gráfico de pares). Esse tipo de gráfico permite avaliar a separabilidade das classes no espaço de características reduzido. A boa separação visual entre as classes, mesmo que em apenas dois ou três componentes, é um forte indicativo de que o classificador terá bom desempenho. Conforme a Figura 3, os componentes, *pca_1* e *pca_2* demonstram ser bons indicadores para a separação das classes. Essa análise foi realizada apenas para o modelo que apresentou o maior ganho percentual de desempenho após a aplicação do PCA, utilizando o classificador *k*-NN: o modelo *PCA_CNN_VGG16_256_max*.

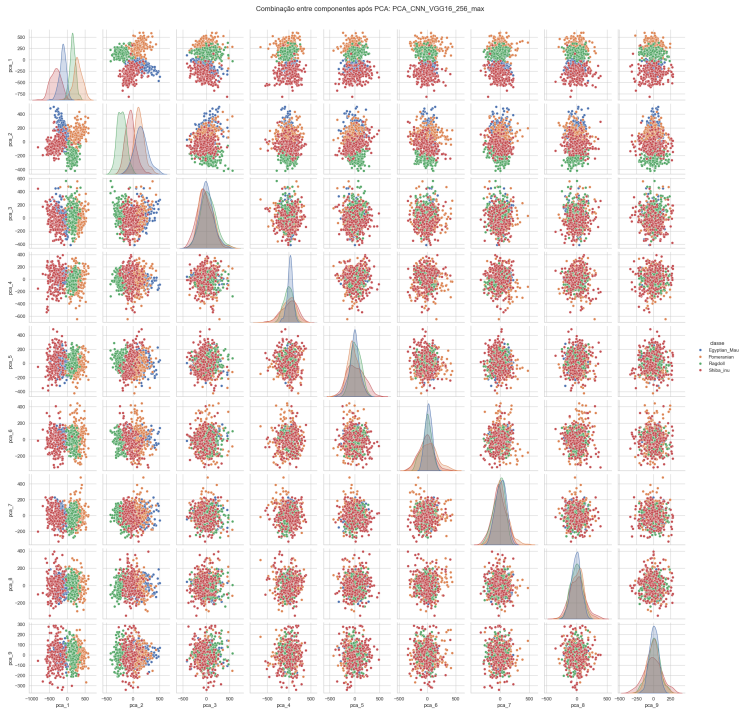

Figura 3. Matriz de dispersão: PCA_CNN_VGG16_256_max

4. Aprendizado Supervisionado

Para realizar o aprendizado supervisionado de classificação, foram utilizados quatro algoritmos amplamente consolidados quanto à sua eficácia e desempenho: *k-Nearest Neighbors* (*k*-NN), Árvore de Decisão, *Naive Bayes* e Redes Neurais Artificiais.

Cada modelo foi avaliado com base na métrica de acurácia e na matriz de confusão. Para o treinamento, foram adotadas duas estratégias clássicas de validação de modelos: *Holdout* e *k-Fold Cross Validation*. A técnica *Holdout* consiste em realizar uma única divisão do conjunto de dados, adotando a proporção de 70% para treinamento e 30% para teste. No entanto, por se tratar de uma divisão única e aleatória, essa abordagem pode introduzir viés, favorecendo ou desfavorecendo o modelo dependendo da distribuição dos dados.

Para mitigar essa limitação, também foi utilizada a validação cruzada *k-Fold*, que consiste em dividir o conjunto de dados em k subconjuntos de tamanhos iguais. Em cada uma das k iterações, um subconjunto é utilizado como conjunto de teste, enquanto os demais são utilizados para o treinamento. Ao final do processo, todos os dados são utilizados tanto para treino quanto para validação, promovendo uma avaliação mais robusta e confiável do desempenho dos modelos. Neste trabalho, foi adotado o valor de $k = 10$, caracterizando uma validação cruzada 10-fold. A Figura 4 demonstra as bases treinadas com *k*-NN bases que tiveram melhor resultado com holdout ou com k-fold.

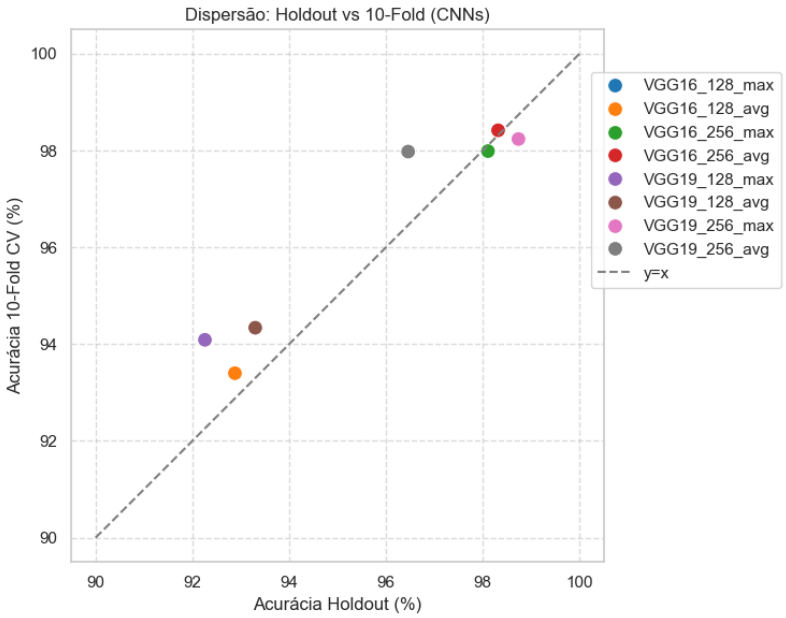

Figura 4. Dispersão Holdout x 10-fold

É importante destacar que, como os modelos serão comparados entre si, a comparação de desempenho deve ser feita entre modelos avaliados sob a mesma estratégia de validação. Ou seja, os modelos treinados com *Holdout* devem ser comparados entre si, e o mesmo vale para os modelos avaliados com *k-Fold*. Comparações cruzadas entre as duas estratégias (por exemplo, *Holdout* versus *k-Fold*) não são apropriadas, pois envolvem critérios de avaliação distintos.

4.1. k-Nearest Neighbors (*k*-NN)

O *k*-NN é um modelo de aprendizado de máquina supervisionado proposto em 1975, que realiza previsões e classificações com base nos vizinhos mais próximos da instância a ser classificada [Fukunage and Narendra 1975]. O algoritmo calcula a distância entre a

nova instância e as instâncias previamente rotuladas; em seguida, identifica os k vizinhos mais próximos e determina a classe com base em votação majoritária. Na Figura 5, é possível observar que, para o exemplo com $k = 5$, a nova instância seria classificada como vermelha (3 votos vermelhos e 2 votos azuis).

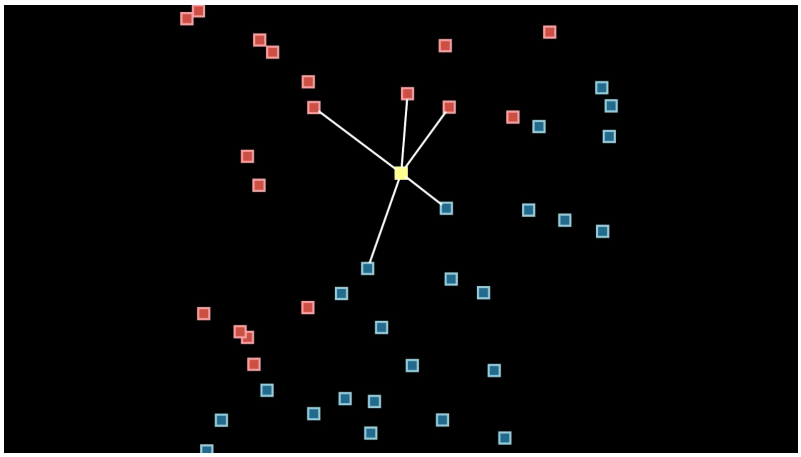

Figura 5. Classificação de uma nova instância por *k*-NN com $k = 5$

O algoritmo k-NN foi o primeiro aplicado às bases de dados apresentadas nas Tabelas 1 e 2. Nesta etapa, a análise concentrou-se no impacto do parâmetro k sobre o desempenho do modelo, bem como nas configurações adotadas na construção das bases de dados. A padronização do tamanho das imagens mostrou-se um fator crucial para a otimização do desempenho: as bases com imagens redimensionadas para 128x128 pixels apresentaram desempenho inferior quando comparadas às com 256x256 pixels. Essa diferença pode ser observada claramente no mapa de calor da Figura 6.

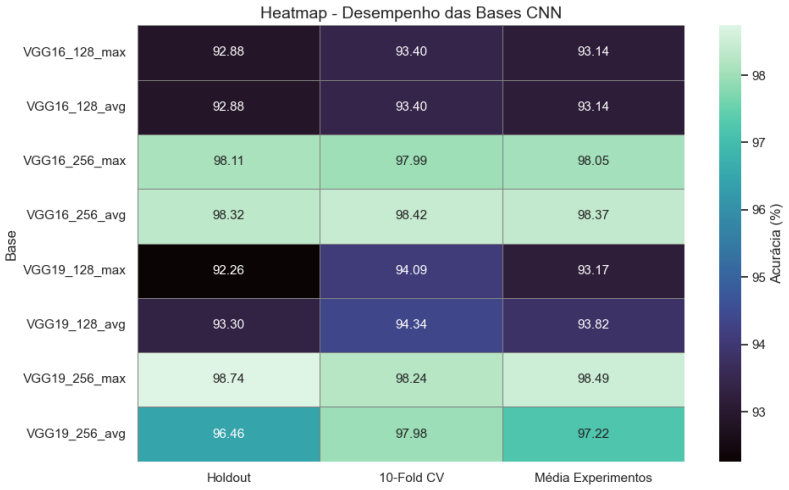

Figura 6. Comparação de desempenho entre imagens 128x128 e 256x256

Quanto ao comportamento em função do número de vizinhos, nota-se um padrão não linear, com oscilações de desempenho conforme o valor de k. Nesta análise, optou-se por desconsiderar o valor de $k = 1$, por ser considerado baixo demais para uma predição confiável, podendo resultar em classificações instáveis.

Ao analisar a média geral das acurácias em função de k, observa-se que o intervalo ideal se encontra entre $k = 4$ e $k = 8$, onde os modelos atingem seus maiores desempenhos médios — tanto antes quanto após a aplicação da técnica de redução de dimensionalidade PCA. Valores de k acima desse intervalo resultam em queda gradual de desempenho, o que indica que considerar muitos vizinhos pode introduzir ruído, agregando instâncias de outras classes e reduzindo a precisão. Por outro lado, valores de k muito baixos dificultam a generalização do modelo.

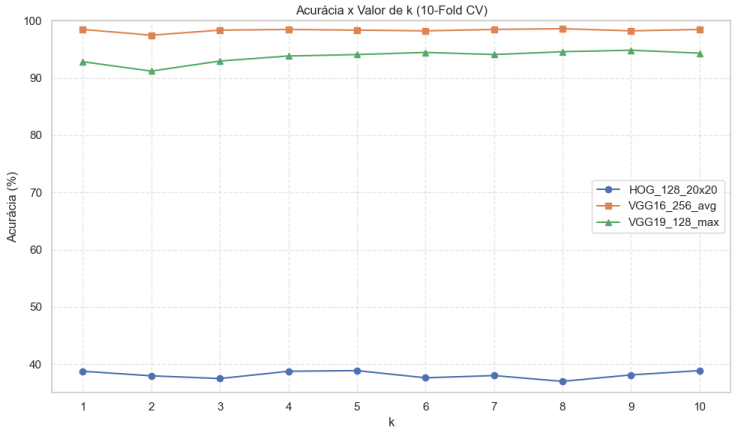

Figura 7. Comportamento da acurácia média em função de k (10-Fold)

O gráfico da Figura 7 reforça essa análise, evidenciando uma estabilidade nas acurácias dentro do intervalo mencionado, o que sugere que esse intervalo de valores é o mais adequado para a tarefa de classificação em questão.

4.2. Decision Tree (Árvore de Decisão)

A árvore de decisão é um algoritmo de aprendizado supervisionado utilizado tanto para tarefas de classificação quanto de regressão. Ela organiza decisões de forma hierárquica, semelhante a um fluxograma, onde cada nó representa uma decisão baseada em um atributo dos dados, e cada folha representa um resultado possível [IBM 2025]

Estrutura da Árvore:

- Nó raiz: ponto inicial, representa o atributo mais relevante para a divisão dos dados.
- Nós de decisão: pontos intermediários onde são feitas novas divisões com base em outros atributos.
- Ramos: caminhos que conectam os nós, indicando as escolhas feitas em cada etapa.
- Nós folha: resultados finais, que podem ser uma classe (classificação) ou um valor (regressão)

Este modelo é amplamente utilizado, principalmente por ser a base para algoritmos de comitê, como Random Forest e Gradient Boosting. Ele é suscetível ao overfitting; no entanto, destaca-se pela fácil construção e interpretabilidade. Para esse algoritmo foram analisados o comportamento da acurácia com relação a profundidade da árvore e a padronização do tamanho das imagens. Novamente as bases com padronização de tamanho de 256x256 pixels demonstrou eficácia em termos de acurácia. Para evidenciar esse

detalhe a Figura 8 mostra a matriz de confusão das bases com mesma entrada de dados mudando apenas o tamanho da imagem.

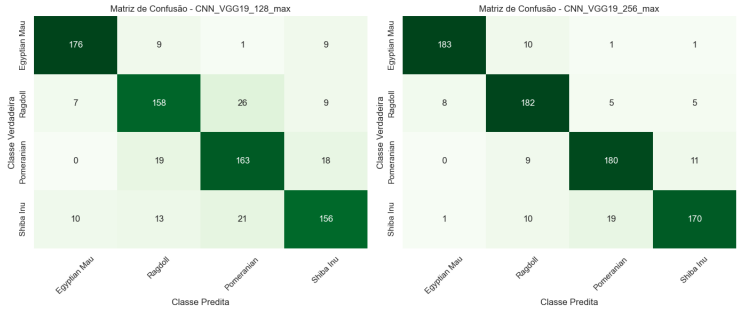

Figura 8. Matriz de Confusão 128x128 e 256x256

A análise da matriz de confusão revela que o modelo treinado com imagens de 128x128 pixels classificou incorretamente 151 instâncias (18,99%), enquanto o modelo com imagens de 256x256 pixels errou apenas 80 instâncias (10,0%). Isso reforça que a padronização de imagens com maior resolução foi crucial para a eficácia do classificador neste problema..

Quanto ao desempenho do modelo de acordo com sua profundidade, nota-se um desempenho elevado entre $md = 4$ e $md = 7$, nesse intervalo para a maioria das bases a acurácia atinge um pontos altissimos e estariam menos propenso a overfitting, além de menos custoso computacionalmente. A faixa de melhor desempenho é relativa, pois a escolha da profundidade ideal depende do critério de otimização (ex: custo-benefício ou desempenho máximo). Por exemplo, na Figura 9, a base PCA + CNN_VGG19_128_max apresenta seu melhor desempenho com $md = 9$. No entanto, um modelo com essa profundidade é computacionalmente mais custoso, o que poderia ser um fator limitante em aplicações práticas. É possível evidenciar algumas acurácias de algumas bases que foram treinadas com árvore de decisão na Figura 9.

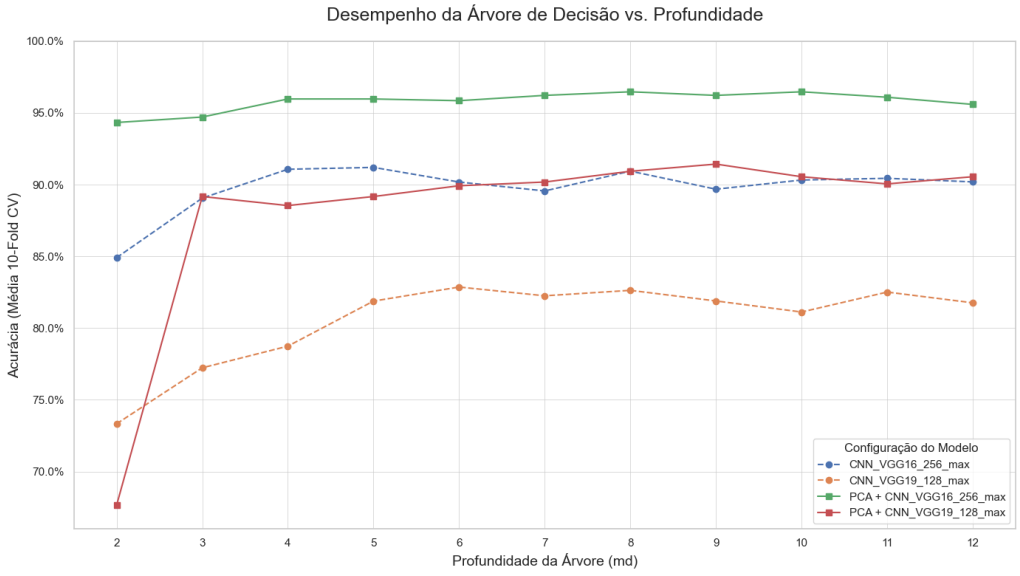

Figura 9. Acurácia de algumas bases de AD

4.3. Naive Bayes

O Naive Bayes é um algoritmo de aprendizado supervisionado baseado no Teorema de Bayes, que realiza classificações assumindo que as características são independentes entre si, uma suposição chamada de ”ingênua”(*naive*). Ele calcula a probabilidade de uma instância pertencer a uma classe com base nas probabilidades condicionais das suas características.

Neste trabalho, foram avaliadas três variantes do algoritmo: GaussianNB, que assume que as características seguem uma distribuição normal; MultinomialNB, projetado para dados de contagem; e ComplementNB, uma adaptação do multinomial para classes desbalanceadas. Os resultados completos são apresentados na Tabela 4.

Tabela 4. Acurácia dos classificadores Naive Bayes nas diferentes bases

Bases	Treino/teste	GaussianNB	MultinomialNB	ComplementNB	Média
CNN_VGG16_256_max (512)	70/30	97,49%	98,32%	97,07%	97,70%
	10-Fold CV	97,22%	98,23%	97,10%	97,52%
CNN_VGG16_256_avg (512)	70/30	95,81%	98,32%	98,74%	97,62%
	10-Fold CV	96,21%	98,74%	98,23%	97,73%
CNN_VGG19_128_max (512)	70/30	92,88%	94,97%	93,30%	93,72%
	10-Fold CV	94,08%	96,10%	93,45%	94,54%
CNN_VGG19_256_max (512)	70/30	97,49%	97,90%	98,32%	97,90%
	10-Fold CV	97,48%	98,48%	97,48%	97,81%
CNN_VGG19_128_avg (512)	70/30	89,54%	96,65%	92,46%	92,88%
	10-Fold CV	92,95%	95,97%	94,09%	94,34%
CNN_VGG19_256_avg (512)	70/30	95,81%	99,16%	97,90%	97,62%
	10-Fold CV	96,85%	98,86%	97,86%	97,86%
PCA_CNN_VGG16_256_max (10)	70/30	98,74%	95,39%	96,23%	96,79%
	10-Fold CV	98,99%	94,84%	95,46%	96,43%
PCA_CNN_VGG16_256_avg (10)	70/30	98,32%	94,56%	94,56%	95,81%
	10-Fold CV	98,61%	94,84%	95,85%	96,43%
PCA_CNN_VGG19_128_max (10)	70/30	92,05%	77,82%	90,37%	86,75%
	10-Fold CV	93,58%	86,93%	90,82%	90,44%
PCA_CNN_VGG19_256_max (10)	70/30	97,90%	95,39%	95,39%	96,23%
	10-Fold CV	98,61%	98,73%	95,09%	97,48%
PCA_CNN_VGG19_128_avg (10)	70/30	93,30%	83,68%	92,05%	89,68%
	10-Fold CV	94,21%	88,93%	92,95%	92,03%
PCA_CNN_VGG19_256_avg (10)	70/30	97,90%	88,28%	94,14%	93,44%
	10-Fold CV	98,23%	93,83%	94,09%	95,38%
Média		**95,95%**	**94,37%**	**95,13%**	**95,38%**
Desvio Padrão		**0,0256**	**0,0533**	**0,0237**	—

Analisando o desempenho médio, o GaussianNB (95,95%) se destaca como a variante mais robusta e com o baixo desvio padrão (0,0256), indicando alta estabilidade entre as diferentes bases. Em contrapartida, o MultinomialNB, apesar de atingir picos de performance em bases específicas, apresentou a maior instabilidade (desvio padrão de 0,0533), com quedas drásticas de desempenho em certos cenários, especialmente após a aplicação do PCA.

O desempenho superior do GaussianNB entre as variantes testadas é um forte

indício de que a distribuição das características extraídas se aproxima de uma função gaussiana contínua. Essa hipótese é reforçada pela instabilidade do MultinomialNB, que pressupõe dados de outra natureza.

Um dos resultados mais notáveis foi o impacto positivo da redução de dimensionalidade. Após a aplicação do PCA, a acurácia não apenas foi mantida, como em alguns casos aumentou significativamente. O exemplo mais expressivo é o da base PCA_CNN_VGG16_256_max, que atingiu a impressionante acurácia de 98,99% com GaussianNB e validação 10-Fold, um ganho de quase 1,8 pontos percentuais em relação à sua contraparte com 512 dimensões.

Este ganho pode ser diretamente relacionado à visualização da Figura 3. O sucesso do GaussianNB com os dados pós-PCA demonstra sua capacidade de capitalizar sobre os componentes mais discriminativos. Mesmo que apenas alguns componentes separem as classes de forma clara as classes. Como o algoritmo modela corretamente a natureza contínua de todos os componentes, a evidência dos mais informativos se torna um fator determinante, justificando a alta performance do classificador.

4.4. Redes Neurais Artificiais - MLP

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados na estrutura e funcionamento do cérebro humano, sendo uma das áreas mais proeminentes da Inteligência Artificial. Elas se destacam pela capacidade de aprender padrões complexos e não lineares diretamente dos dados [MBA USP ESALQ 2024]. Neste trabalho, foi utilizada a arquitetura Multi-Layer Perceptron (MLP), um tipo clássico de rede neural feedforward composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. O aprendizado no MLP ocorre através do ajuste dos pesos das conexões entre os neurônios, um processo otimizado por algoritmos como o backpropagation [MarketMuse 2024].

Para avaliar a eficácia do MLP na classificação das raças, foi conduzida uma análise sistemática de seus principais hiperparâmetros, utilizando a implementação MLPClassifier da biblioteca Scikit-learn. O objetivo foi encontrar a configuração ótima através de um processo iterativo, ajustando um parâmetro de cada vez e fixando o de melhor desempenho para a etapa seguinte. Foram investigados os seguintes parâmetros: função de ativação, otimizador (solver), número de neurônios na camada oculta, taxa de aprendizagem e número máximo de iterações. Ao final, os resultados do ajuste manual foram comparados com os obtidos pela busca automatizada do GridSearchCV para validar a eficácia da metodologia.

4.4.1. Função de Ativação

A primeira etapa consistiu em determinar a função de ativação mais eficaz para as camadas ocultas. Foram avaliadas as quatro opções disponíveis na biblioteca Scikit-learn: relu, identity, logistic e tanh.

Durante os testes iniciais com o número padrão de 200 iterações, observou-se que o otimizador frequentemente não atingia a convergência. Para garantir uma comparação justa e robusta, o número máximo de iterações foi fixado em 1800 para todos os testes

subsequentes nesta seção. Mesmo sob essa nova condição, a função logistic (também conhecida como sigmoide) se destacou, alcançando a maior acurácia média (97,12%) e um desvio padrão competitivo (0,0255), conforme detalhado na Tabela 5. Portanto, a função logistic foi selecionada para as próximas etapas de otimização do MLP.

Tabela 5. Acurácia obtida com diferentes funções de ativação

Bases	ReLU	Tanh	Identity	Logistic
CNN_VGG16_256_max (512)	97,49%	98,32%	96,65%	99,58%
CNN_VGG16_256_avg (512)	97,49%	98,74%	97,07%	99,16%
CNN_VGG19_128_max (512)	91,63%	91,63%	92,46%	94,56%
CNN_VGG19_256_max (512)	95,81%	97,90%	96,65%	98,74%
CNN_VGG19_128_avg (512)	92,46%	94,97%	89,54%	93,30%
CNN_VGG19_256_avg (512)	96,65%	98,74%	97,07%	99,58%
PCA_CNN_VGG16_256_max (10)	97,49%	98,74%	99,16%	99,58%
PCA_CNN_VGG16_256_avg (10)	99,16%	98,74%	99,16%	99,16%
PCA_CNN_VGG19_128_max (10)	89,12%	92,46%	92,05%	92,46%
PCA_CNN_VGG19_128_avg (10)	92,05%	94,56%	92,88%	95,81%
PCA_CNN_VGG19_256_max (10)	97,07%	97,90%	97,90%	98,74%
PCA_CNN_VGG19_256_avg (10)	94,49%	98,74%	97,90%	98,32%
Média	**94,81%**	**96,86%**	**95,71%**	**97,12%**
Desvio Padrão	**0,0299**	**0,0220**	**0,0324**	**0,0255**

4.4.2. Seleção de otimizador - Solver

A escolha do otimizador (solver) é um passo crucial no treinamento de uma MLP, pois ele define o método utilizado para ajustar os pesos da rede e minimizar a função de erro. A biblioteca Scikit-learn oferece principalmente as opções adam, sgd (Gradiente Descendente Estocástico) e lbfgs.

Para este projeto, o otimizador adam foi selecionado como padrão para todos os experimentos subsequentes. A justificativa para esta escolha se baseia em suas propriedades teóricas e eficácia prática, amplamente documentada na literatura. O adam é um algoritmo de otimização de gradiente estocástico que se destaca por manter uma taxa de aprendizado adaptativa para cada parâmetro da rede. Essa capacidade de ajuste automático o torna, em geral, mais rápido para convergir e menos sensível à escolha da taxa de aprendizado inicial em comparação com o sgd tradicional.

Sua robustez em uma vasta gama de problemas e o fato de ser o otimizador padrão da implementação MLPClassifier reforçaram a decisão de adotá-lo, visando obter um treinamento eficiente e estável para a rede neural.

4.4.3. Hidden Layer Sizes

Para determinar a complexidade ideal da rede, o número de neurônios na camada oculta foi variado. Utilizou-se como ponto de partida a heurística definida pela Equação 1, que sugeriu um valor inicial de 258 neurônios. A partir deste valor, foram testados pontos em um amplo espectro (1, 25, 258, 512, 1000) para identificar as regiões de underfitting e overfitting do modelo.

$$\text{Nº de neurônios} = \frac{2 \times \text{Nº de atributos} + \text{Nº de classes}}{2} \tag{1}$$

Os resultados, exemplificados para duas bases de dados distintas nas Figuras 10 e 11, demonstram um comportamento clássico de redes neurais.

```
Nº neurônios: 1 | Acurácia Treino: 0.5144 | Acurácia Teste: 0.4017
Nº neurônios: 25 | Acurácia Treino: 0.9946 | Acurácia Teste: 0.9498
Nº neurônios: 258 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9665
Nº neurônios: 512 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9623
Nº neurônios: 1000 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9540
```

Figura 10. Acurácia de teste e treino da CNN_VGG19_128_max

```
Nº neurônios: 1 | Acurácia Treino: 0.5144 | Acurácia Teste: 0.4100
Nº neurônios: 25 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9498
Nº neurônios: 258 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9498
Nº neurônios: 512 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9498
Nº neurônios: 1000 | Acurácia Treino: 1.0000 | Acurácia Teste: 0.9456
```

Figura 11. Acurácia de teste e treino da CNN_VGG19_128_avg

Com apenas 1 neurônio, o modelo exibe um claro underfitting: tanto a acurácia de treino quanto a de teste são extremamente baixas (próximas a 51% e 41%, respectivamente), indicando que a rede não possui capacidade suficiente para aprender os padrões dos dados.

O fenômeno de overfitting é ilustrado de forma exemplar na Figura 10. A acurácia de teste atinge seu pico com 258 neurônios (96,65%). Após esse ponto, embora a acurácia de treino permaneça em 100% (indicando que o modelo ”decorou”os dados de treino), a acurácia de teste começa a cair progressivamente para 96,23% (512 neurônios) e 95,40% (1000 neurônios). Essa divergência entre as curvas de treino e teste é a principal característica do overfitting, onde o modelo perde sua capacidade de generalização para dados não vistos.

A Figura 11 mostra um comportamento um pouco diferente, onde o modelo atinge um platô de desempenho. A acurácia de teste máxima (94,98%) é alcançada com apenas 25 neurônios e não melhora significativamente ao adicionar mais complexidade. Uma queda muito sutil no desempenho só é observada com 1000 neurônios, reforçando que aumentar a complexidade da rede nem sempre resulta em melhor performance e pode ser computacionalmente ineficiente.

Com base nesta análise, conclui-se que a faixa ótima de neurônios para este problema se encontra entre 25 e 258, dependendo da base de dados. Para as etapas seguintes,

foi escolhido o valor de 258 neurônios, por representar o ponto de melhor desempenho médio antes do início do overfitting.

4.4.4. Taxa de aprendizagem - rate learning

As taxas de aprendizagem foram testadas nos seguintes valores: 0.001, 0.01 e 0.1, com o objetivo de entender como esse hiperparâmetro influencia o processo de ajuste da rede neural. A taxa de aprendizagem tem relação direta com o tamanho dos passos que o modelo dá durante a atualização dos pesos a cada iteração. Dessa forma, taxas muito altas podem fazer o modelo não convergir para uma solução ideal, ou até mesmo divergir. Por outro lado, taxas muito baixas podem resultar em um aprendizado lento, impedindo que o modelo alcance uma boa capacidade de generalização.

Tabela 6. Acurácia obtida para diferentes taxas de aprendizado com HLS = 258 nas 12 bases de dados

Base de Dados	learning rate = 0,001	learning rate = 0,01	learning rate = 0,1
CNN_VGG16_256_max (512)	99,16% 98,99%	99,16% 98,99%	96,23% 72,21%
CNN_VGG16_256_avg (512)	99,16% 99,37%	99,16% 99,12%	98,74% 98,73%
CNN_VGG19_128_max (512)	96,65% 96,60%	95,40% 95,97%	84,52% 87,92%
CNN_VGG19_256_max (512)	99,16% 99,11%	98,74% 99,49%	71,97% 79,88%
CNN_VGG19_128_avg (512)	94,98% 97,23%	95,82% 96,98%	93,72% 94,21%
CNN_VGG19_256_avg (512)	99,58% 99,62%	99,16% 99,37%	98,33% 98,61%
PCA_CNN_VGG16_256_max (10)	99,58% 99,24%	99,58% 99,74%	99,58% 99,49%
PCA_CNN_VGG16_256_avg (10)	99,16% 99,24%	99,58% 99,24%	98,33% 98,74%
PCA_CNN_VGG19_128_max (10)	92,47% 95,22%	92,05% 95,85%	92,89% 93,33%
PCA_CNN_VGG19_128_avg (10)	98,74% 96,10%	98,74% 95,97%	98,74% 95,47%
PCA_CNN_VGG19_256_max (10)	98,33% 99,11%	98,33% 98,86%	96,23% 98,46%
PCA_CNN_VGG19_256_avg (10)	94,56% 98,90%	94,98% 98,99%	94,14% 98,49%
Média	97,93%	97,89%	93,29%

Desvio padrão: learning rate = 0,001: **0,0196**, learning rate = 0,01: **0,0196**, learning rate = 0,1: **0,0801**

Ao analisar os resultados, foi possível observar os seguinte comportamentos:

- A taxa de 0.1 apresentou, em alguns casos, problemas de não convergência, com acurácias bem abaixo do esperado. Isso caracterizou situações de underfitting, onde tanto a acurácia de treino quanto a de teste foram baixas.
- Já as taxas de 0.001 e 0.01 mostraram-se mais estáveis, resultando em bons níveis de acurácia e sendo recomendadas como faixas iniciais adequadas para testes em MLPs. Mas ainda assim a taxa 0.001 se mostrou como mais estável para o aprendizado da MLP.

Por fim, com base nos experimentos, foi possível concluir que a faixa de underfitting tende a ocorrer em valores mais altos da taxa de aprendizagem (acima de 0.1). Já a ocorrência de overfitting esteve mais associada ao número de iterações e à complexidade da rede (como o número de neurônios) do que propriamente à taxa de aprendizagem.

4.4.5. Número de interações

A fim de verificar como o número de interações influencia na construção da rede, foram testados diferentes configurações desse hiperparâmetro. Os valores testados foram: 200, 500, 1200, 2000, 5000, 8000 e 10000. Nesse tipo de parâmetro, o comportamento esperado é o seguinte: para valores baixos a rede deve apresentar underfitting, ou seja, à quantidade de interação não foi o suficiente para o aprendizado da MLP e para valores altos de interação a MLP se ajusta demais aos meus dados apresentando overfitting. O número 200 é a quantidade de interação padrão da documentação do Scikit-Learn. Para essa quantidade em alguns casos à rede não tinha convergido diretamente, o que sugere que um intervalo de 1-200 de underfitting. Para interações de 500, 1200, 2000 à rede mostrou-se tá bem ajustada e generalizando muito bem, inclusive o valor de acurácias para essas 3 interações foram exatamentes o mesmo. No entanto para valores acima de 5000 a acurácia de treino continua bem alta, mas a acurácia de teste começa à ter decaída demonstrando problema de overfitting.

4.5. GridSearchCV

Por questão de comparação, foi utilizado a função Grid Search para verificar o quão distante os valores encontrados manualmente ficaram distantes dos oferecidos pela função utilizada. A busca foi realizada em um espaço de parâmetros contendo diferentes quantidades de neurônios, funções de ativação, taxas de aprendizagem com um número fixo de interações, 500 interações. É possível notar os resultados na Tabela 7 A função “logistic” ou função sigmoidal se apresentou como melhor função de ativação na maioria das bases, o que conferiu com uma análise manual da MLP. Em outras poucas bases a melhor função de ativação se apresentou como a Relu e Tangente Hiperbólica. No entanto, é possível concluir que a função sigmoide se apresenta como melhor função de ativação para as bases utilizadas nesse projeto.

Tabela 7. Resultados dos experimentos com diferentes configurações de MLP

Base	Função de Ativação	Nº de Neurônios	Learning Rate	Acurácia Média CV	Acurácia Holdout
cnn_VGG16_256_max	logistic	258	0.001	0.9928	0.9916
cnn_VGG16_256_avg	logistic	100	0.001	0.9946	0.9916
cnn_VGG19_128_max	tanh	400	0.001	0.9712	0.9163
cnn_VGG19_256_max	logistic	500	0.001	0.9928	0.9916
cnn_VGG19_128_avg	logistic	100	0.01	0.9748	0.9540
cnn_VGG19_256_avg	logistic	100	0.001	0.9946	0.9958
PCA_cnn_VGG16_256_max	relu	500	0.001	0.9964	0.9916
PCA_cnn_VGG16_256_avg	tanh	400	0.001	0.9946	0.9916
PCA_cnn_VGG19_128_max	logistic	200	0.001	0.9712	0.9247
PCA_cnn_VGG19_256_max	logistic	100	0.01	0.9928	0.9874
PCA_cnn_VGG19_128_avg	logistic	100	0.01	0.9676	0.9623
PCA_cnn_VGG19_256_avg	logistic	258	0.01	0.9964	0.9833

Na taxa de aprendizagem demonstrou as expectativas analisadas manualmente, onde 0.1 era uma taxa muito alta e a rede não convergia para uma solução adequada. E o 0.001 e a 0.01 demonstraram resultados mais estáveis e seria um bom pontapé inicial para construção da MLP. Com o Grid Search demonstrou-se claramente essas duas taxas como as melhores para as MLP treinadas nas bases desse projeto. A diferença de acurácia foi pequena, entre 1 a 2 pontos percentuais, mas a função Grid Search conseguiu encontrar melhores configurações para redes que inicialmente tiveram um resultado abaixo na busca manual, como as bases padronizadas com imagens de 128x128, demonstrando à importância de funções automatizadas para construção de redes neurais. A quantidade de neurônios apresentou muita variação com relação ao que foi considerado melhor na análise manual, 258 neurônios. No entanto, com análise da função outras quantidades demonstraram melhores resultados: 100, 400, 500, demonstrando uma diferença de 158, 142 e 252 neurônios, respectivamente. Mas ainda sim, em algumas bases à quantidade de partida demonstrou como a melhor quantidade de neurônios, mostrando que é um bom ponto de partida.

Listing 1. Definição do grid de hiperparâmetros

```
# Definindo o grid de hiperparmetros
param_grid = {
    'hidden_layer_sizes': [(100,), (200,), (258,), (400,),
       (500,)],
    'activation': ['logistic', 'relu', 'tanh'],
    'learning_rate_init': [0.001, 0.01, 0.1],
}
```

De modo geral as redes neurais apresentaram acurácias acima de 98,3%, novamente, apenas bases de dados com imagens de 128x128 pixels demonstraram eficácia e acurácia abaixo das demais, em um intervalo de 94% - 96%. Mas ainda assim, é muito melhor que os outros classificadores tradicionais usados nesse trabalho. A Tabela 8 mostra os resultados obtidos nessa classificação.A Tabela será mostrada em paisagem devido seu tamanho.

Tabela 8. Resultados dos experimentos de otimização de hiperparâmetros para o MLP.

		Configurações do MLP (Acurácia)										
Bases	**Treino/teste**	HLS=1, 200it	HLS=25, 200it	HLS=258, 200it	HLS=512, 200it	HLS=1000, 200it	HLS=258, LR=0.001	HLS=258, LR=0.01	HLS=258, LR=0.1	HLS=258, LR=0.001, 500it	HLS=258, LR=0.001, 1200it	HLS=258, LR=0.001, 2000it
CNN_VGG16_256_max	70/30	44,77%	98,33%	99,16%	99,58%	99,58%	99,16%	99,16%	96,23%	99,16%	99,16%	99,16%
	10-Fold CV	45,92%	98,99%	98,99%	99,49%	99,49%	98,99%	98,99%	72,21%	98,99%	98,99%	98,99%
CNN_VGG16_256_avg	70/30	45,19%	98,74%	99,16%	99,16%	99,16%	99,16%	99,16%	98,74%	99,16%	99,16%	99,16%
	10-Fold CV	46,80%	99,37%	99,37%	99,49%	99,49%	99,37%	99,12%	98,73%	99,37%	99,37%	99,37%
CNN_VGG19_128_max	70/30	40,17%	94,98%	96,65%	96,23%	95,40%	96,65%	95,40%	84,52%	96,65%	96,65%	96,65%
	10-Fold CV	43,66%	95,35%	96,60%	96,10%	95,72%	96,60%	95,97%	87,92%	96,60%	96,60%	96,60%
CNN_VGG19_256_max	70/30	44,77%	98,33%	99,16%	98,71%	98,74%	99,16%	98,74%	71,97%	99,16%	99,16%	99,16%
	10-Fold CV	45,79%	98,36%	99,11%	99,49%	99,24%	99,11%	99,49%	79,88%	99,11%	99,11%	99,11%
CNN_VGG19_128_avg	70/30	41,00%	94,98%	94,98%	94,98%	94,56%	94,98%	95,82%	93,72%	94,98%	94,98%	94,98%
	10-Fold CV	44,53%	96,47%	97,23%	96,85%	96,98%	97,23%	96,98%	94,21%	97,23%	97,23%	97,23%
CNN_VGG19_256_avg	70/30	45,19%	99,16%	99,58%	99,58%	99,58%	99,58%	99,16%	98,33%	99,58%	99,58%	99,58%
	10-Fold CV	47,18%	99,62%	99,62%	99,74%	99,74%	99,62%	99,37%	98,61%	99,62%	99,62%	99,62%
PCA_CNN_VGG16_256_max	70/30	51,08%	99,16%	99,58%	99,58%	99,58%	99,58%	99,58%	99,58%	99,58%	99,58%	99,58%
	10-Fold CV	48,93%	99,37%	99,24%	99,62%	99,87%	99,24%	99,74%	99,49%	99,24%	99,24%	99,24%
PCA_CNN_VGG16_256_avg	70/30	48,95%	98,74%	99,16%	99,16%	99,16%	99,16%	99,58%	98,33%	99,16%	99,16%	99,16%
	10-Fold CV	47,17%	99,37%	99,24%	99,24%	99,37%	99,24%	99,24%	98,74%	99,24%	99,24%	99,24%
PCA_CNN_VGG19_128_max	70/30	49,37%	94,14%	92,42%	92,05%	92,89%	92,47%	92,05%	92,89%	92,47%	92,47%	92,47%
	10-Fold CV	45,54%	95,47%	95,22%	95,72%	95,35%	95,22%	95,85%	93,33%	95,22%	95,22%	95,22%
PCA_CNN_VGG19_128_avg	70/30	46,06%	94,14%	94,56%	93,72%	93,31%	98,74%	98,74%	98,74%	94,56%	94,56%	94,56%
	10-Fold CV	44,79%	96,22%	96,10%	96,10%	95,85%	96,10%	95,97%	95,47%	96,10%	96,10%	96,10%
PCA_CNN_VGG19_256_max	70/30	51,05%	98,33%	98,74%	98,74%	98,74%	98,33%	98,33%	96,23%	98,74%	98,74%	98,74%
	10-Fold CV	49,82%	98,74%	99,11%	99,11%	99,11%	99,11%	98,86%	98,46%	99,11%	99,11%	99,11%
PCA_CNN_VGG19_256_avg	70/30	49,79%	99,16%	98,33%	98,33%	98,33%	94,56%	94,98%	94,14%	98,33%	98,33%	98,33%
	10-Fold CV	45,41%	99,24%	98,99%	98,99%	98,87%	98,90%	98,99%	98,49%	98,99%	98,99%	98,99%
MÉDIA		46,37%	97,70%	97,93%	97,91%	97,84%	97,93%	97,89%	93,29%	97,93%	97,93%	97,93%
DESVIO PADRÃO		0,0275	0,0185	0,0194	0,0211	0,0216	0,0194	0,0196	0,0801	0,0194	0,0194	0,0194

5. Comitê de Classificadores

Comitês de classificadores, conhecidos como métodos ensemble, são técnicas de aprendizado supervisionado que combinam a previsão de vários modelos para um desempenho preditivo superior a um único modelo isolado.O princípio central é que diferentes modelos podem cometer erros distintos, e a combinação de suas decisões compensa os erros individuais, reduzindo o erro geral do sistema [GeeksforGeeks 2023]. Dessa forma, a diversidade nas respostas desses modelos é de extrema importância. Neste trabalho foram usados os 4 comitês apresentados na Tabela 9.

Tabela 9. Exemplos de Comitês de Classificadores

Comitê	Descrição
Bagging	(Bootstrap Aggregating) Treina múltiplos modelos independentes em subconjuntos aleatórios dos dados e combina suas previsões por votação (classificação) ou média (regressão).
Boosting	Cria uma sequência de modelos onde cada novo modelo corrige os erros dos anteriores, atribuindo maior peso às amostras mal classificadas.
Random Forest	Variante do bagging que utiliza árvores de decisão e ainda adiciona aleatoriedade na escolha dos atributos em cada divisão da árvore.
Stacking	Combina diferentes tipos de modelos (nível-0) cujas previsões alimentam um meta-modelo (nível-1), que aprende a melhor maneira de combiná-los.

Para a construção dos comitês Bagging, foram utilizados como modelos base os quatro algoritmos supervisionados: k-NN, Árvore de Decisão (AD), Naive Bayes e MLP. A análise foi dividida em duas etapas: primeiro, variando o número de estimadores do Bagging padrão e, em seguida, avaliando o impacto da seleção de atributos.

5.0.1. Impacto do Número de Estimadores.

Inicialmente, o número de estimadores (n_estimators) foi variado entre 10, 20 e 30, utilizando todas as características disponíveis (max_features=1.0). Conforme ilustrado na Figura 12, o desempenho do comitê tende a aumentar ao passar de 10 para 20 estimadores. No entanto, o ganho de performance de 20 para 30 estimadores torna-se marginal ou até inexistente, indicando que o modelo atinge um platô de estabilidade. Isso ocorre porque, a partir de um certo ponto, adicionar mais modelos ao comitê oferece retornos decrescentes na redução da variância do ensemble. Esse comportamento é evidenciado na Figura 12.

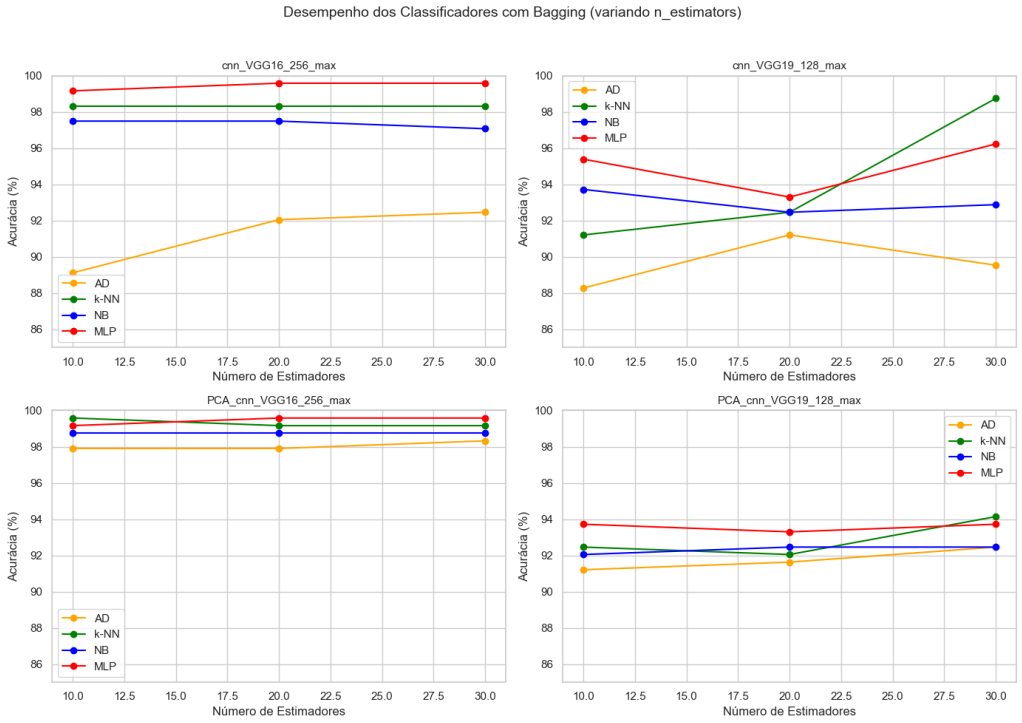

Figura 12. Acurácia de algumas bases variando os estimadores

5.0.2. Impacto da Seleção de Atributos em Bases de Alta Dimensionalidade (CNN).

A segunda análise investigou o efeito da seleção de atributos, transformando o Bagging em uma técnica similar ao Random Forest. Conforme detalhado na Tabela 10, ao reduzir o parâmetro max_features para valores como 0.5 ou 0.3, o desempenho do comitê com Árvores de Decisão geralmente melhorou. Isso ocorre porque a Árvore de Decisão é um modelo de alta variância (suscetível a overfitting), e forçar cada árvore do comitê a usar um subconjunto aleatório de características aumenta a diversidade e reduz a correlação entre elas, resultando em um modelo final mais robusto e generalista.

5.0.3. Impacto da Seleção de Atributos em Bases de Baixa Dimensionalidade (PCA).

O comportamento, no entanto, foi diferente para as bases de dados pós-PCA, que possuem apenas 10 características altamente informativas (Tabela 11). Nesses casos, a seleção de um subconjunto muito pequeno de atributos (3 componentes com max_features=0.3) frequentemente levou a uma queda no desempenho. A hipótese para este resultado é que, com poucas características importantes, a seleção aleatória corre um risco maior de omitir os componentes mais discriminativos, fornecendo uma visão incompleta dos dados para os modelos base. Portanto, para estas bases de baixa dimensão, a utilização de uma maior porcentagem de características se mostrou mais eficaz. A Figura 13 mostra esses comportamentos.

Tabela 10. Configuração do parâmetro `max_features` para as bases CNN

`max_features`	Nº de Atributos por Estimador
`1.0` (padrão)	512
`0.3`	153
`0.5`	256
`0.8`	410

Tabela 11. Configuração do parâmetro `max_features` para as bases PCA

`max_features`	Nº de Atributos por Estimador
`1.0` (padrão)	10
`0.3`	3
`0.5`	5
`0.8`	8

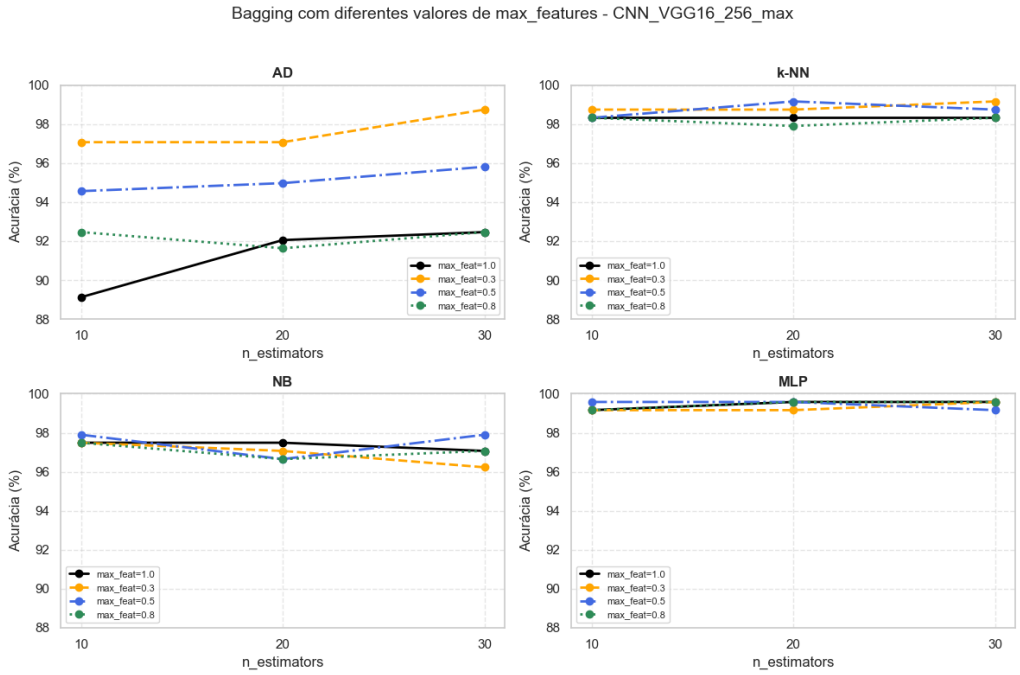

Figura 13. Variação do Bagging com seleção de features

5.0.4. Boosting

Diferente do Bagging, que treina modelos de forma paralela, o Boosting constrói o comitê de forma sequencial, onde cada novo modelo é treinado para corrigir os erros do anterior. Neste trabalho, foi utilizada a implementação GradientBoostClassifier do scikitLearn com Árvore de Decisão e Naive Bayes como modelos base.

A Tabela 12 e a Figura 14 apresentam a comparação direta de desempenho com o Bagging. Observa-se que, para a Árvore de Decisão, o Bagging consistentemente superou o Boosting em ambas as bases de dados. Uma hipótese para este resultado é a sensibilidade do AdaBoost a ruídos: ao focar intensamente nos erros, o modelo pode ter se ajustado a outliers, prejudicando sua capacidade de generalização. Para o Naive Bayes, um modelo já estável, ambas as técnicas de comitê apresentaram resultados muito similares entre si e ao modelo base, indicando que não houve ganho significativo com a aplicação de comitês.

Tabela 12. Comparativo de Acurácia: Bagging vs. Boosting (AdaBoost)

	Bagging Padrão			Boosting (AdaBoost)		
Classificador Base	**n=10**	**n=20**	**n=30**	**n=10**	**n=20**	**n=30**
Árvore de Decisão (AD)	89,12%	92,05%	92,46%	87,44%	88,28%	88,28%
Naive Bayes (NB)	97,49%	97,49%	97,07%	97,07%	97,07%	97,49%
Árvore de Decisão (AD)	89,54%	91,63%	89,95%	81,59%	82,24%	80,33%
Naive Bayes (NB)	91,63%	89,95%	89,95%	89,12%	89,12%	89,12%

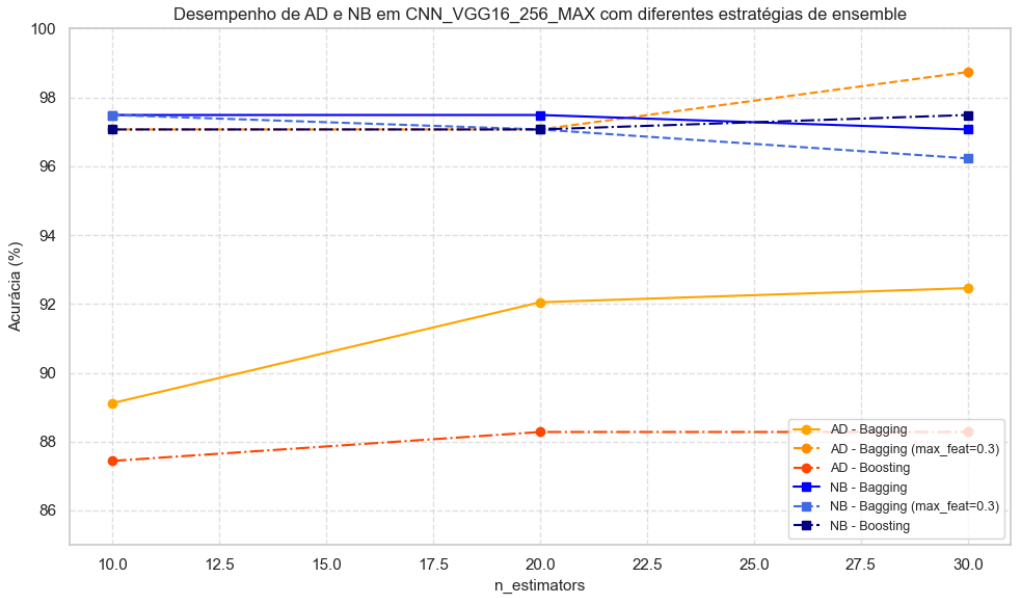

Figura 14. Análise de Bagging x Boosting

5.0.5. Random Forest

O Random Forest é uma técnica de comitê que pode ser vista como uma evolução do Bagging aplicado a Árvores de Decisão. Ele utiliza a mesma base do Bagging (amostragem de dados com reposição), mas adiciona um segundo nível de aleatoriedade: em cada nó da árvore, apenas um subconjunto aleatório de características é considerado para a divisão. Isso força as árvores a serem mais diversas entre si, o que geralmente leva a um modelo final mais robusto.

A Figura 15 compara o desempenho do Random Forest com o Bagging (utilizando a média de todos os modelos treinados). Os resultados mostram que as duas técnicas possuem acurácias muito competitivas e similares, com uma ou outra se destacando a depender da base de dados. Isso confirma que ambas as abordagens de redução de variância são altamente eficazes.

Uma comparação adicional foi feita entre o Random Forest e um comitê Bagging utilizando o classificador MLP na sua melhor configuração (Figura 16). Neste cenário, o Bagging com MLP demonstrou uma clara superioridade em acurácia. No entanto, é fundamental notar que o custo computacional de treinar um comitê de Redes Neurais é ordens de magnitude maior que o de um Random Forest, tornando o segundo uma opção muito mais viável e com excelente custo-benefício em aplicações práticas.

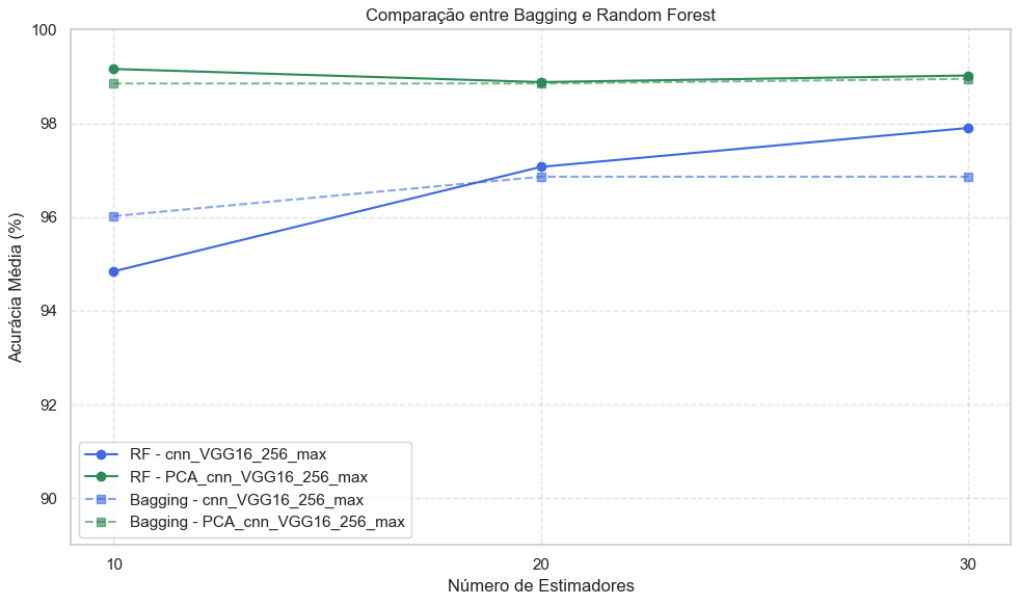

Figura 15. Análise de Bagging x Random Forest

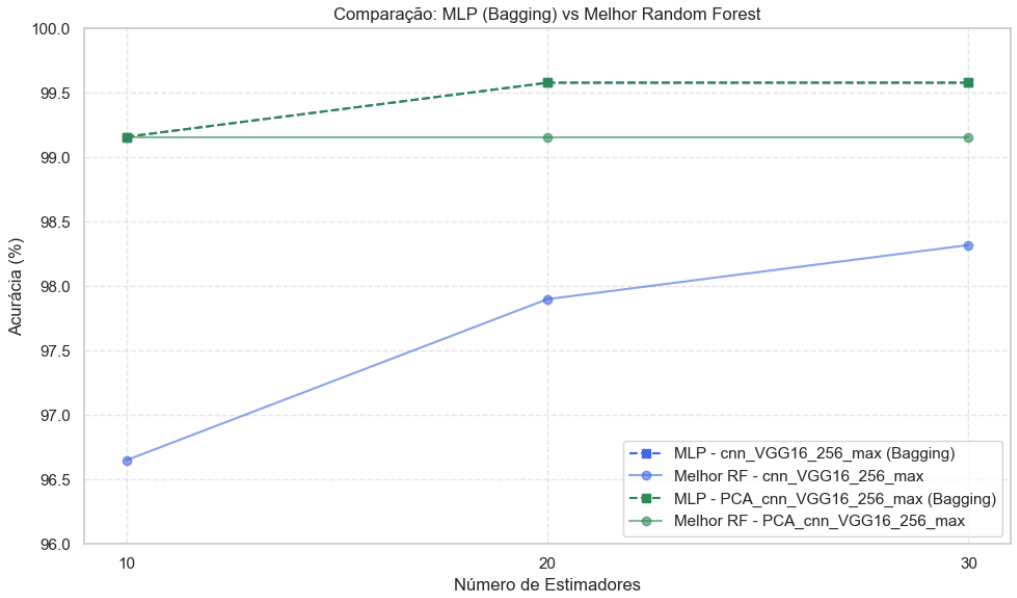

Figura 16. Análise de Bagging + MLP x Random Forest

5.0.6. Stacking

O Stacking (ou empilhamento) é uma técnica de comitê que utiliza múltiplos modelos, muitas vezes de tipos diferentes (heterogêneos), para gerar previsões que servem de entrada para um meta-modelo final. Este, por sua vez, aprende a combinar as previsões base da forma mais eficaz possível.

Nos experimentos, foram testadas configurações com 5, 10, 15 e 20 classificadores base. Observou-se que o comitê com 5 classificadores alcançou o melhor desempenho. Adicionar mais modelos não resultou em melhorias, possivelmente devido a um superajuste (overfitting) no nível do meta-modelo, que pode ter recebido características cada vez mais correlacionadas.

A análise final, ilustrada na Figura 17, compara o desempenho dos modelos individuais, do comitê homogêneo (Bagging geral e apenas MLP) e do comitê heterogêneo (Stacking). Fica evidente que ambas as técnicas de comitês superam os modelos individuais com folga, exceto o modelo de MLP. Comparando os comitês, o Stacking e Bagging + MLP apresentaram as maiores acurácias médias, consolidando-se como as abordagens de melhores desempenho geral entre todas as técnicas avaliadas neste trabalho.

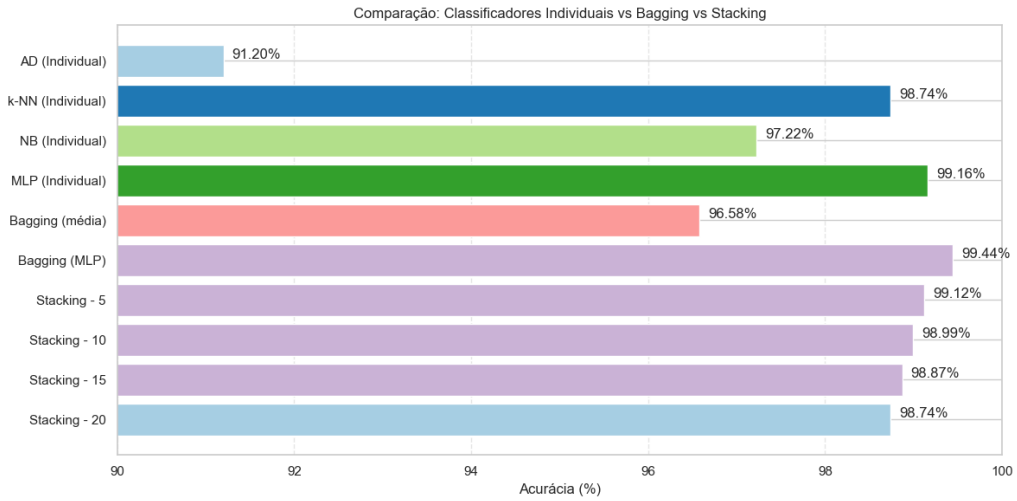

Figura 17. Comparação Bagging x Stacking x Individuais

6. Teste Estatístico

As seções anteriores apresentaram uma análise de desempenho baseada nos valores de acurácia. Contudo, para validar formalmente se as diferenças observadas entre os modelos são estatisticamente relevantes ou fruto do acaso, foi realizada uma análise estatística não paramétrica, conforme recomendado na literatura de Aprendizado de Máquina.

O primeiro passo consistiu em comparar os quatro classificadores base (k-NN, AD, NB e MLP) utilizando o Teste de Friedman. Esse teste avalia se os desempenhos médios dos classificadores provêm da mesma distribuição. Para isso, foram definidas as seguintes hipóteses:

- Hipótese Nula (H): Todos os classificadores apresentam, em média, o mesmo desempenho.
- Hipótese Alternativa (H): Pelo menos um dos classificadores apresenta desempenho médio significativamente diferente dos demais.

Os dados de acurácia da Validação Cruzada 10-Fold de cada classificador, obtidos nas 12 bases de dados, foram utilizados como entrada para o teste, conforme exemplificado no Código 3.

Listing 2. Execução do Teste de Friedman para os classificadores base.

```
from scipy.stats import friedmanchisquare

# Acur cias para cada classificador nas 12 bases
knn = [97.99, 98.42, 94.09, 94.34, 98.24, 97.98, 99.11, 98.86,
   95.03, 95.28, 98.86, 98.61]
ad  = [91.20, 94.84, 81.88, 90.57, 86.16, 94.46, 95.97, 97.73,
   91.43, 92.07, 97.98, 98.24]
nb  = [97.22, 96.21, 94.08, 97.48, 92.95, 96.85, 98.99, 98.61,
   93.58, 94.21, 98.61, 98.23]
mlp = [98.99, 99.37, 96.60, 99.11, 97.23, 99.62, 99.24, 99.24,
   95.22, 96.10, 99.11, 98.99]
```

```
# Aplicando o teste de Friedman
stat, p = friedmanchisquare(knn, ad, nb, mlp)

print(f'Estatistica de Friedman: {stat:.4f}')
print(f'p-valor: {p:.8f}')

# Saida esperada:
# Estatistica de Friedman: 32.5000
# p-valor: 0.00000041
```

O teste resultou em um p-valor de 0,00000041. Como este valor é muito inferior ao nível de significância padrão ($\alpha = 0{,}05$), rejeita-se a hipótese nula. Assim, pode-se concluir, com alta confiança estatística, que há diferença real de desempenho entre os quatro classificadores.

Uma vez estabelecida a diferença global, foi realizado um teste post-hoc para identificar quais pares de modelos diferem estatisticamente. O teste escolhido foi o teste de Nemenyi, apropriado para comparações múltiplas após o teste de Friedman. Os resultados estão apresentados na Tabela 13 e visualmente representados no Diagrama de Diferença Crítica (CD), na Figura 19.

Tabela 13. Matriz de p-valores do Teste de Nemenyi entre classificadores.

	k-NN	**AD**	**NB**	**MLP**
k-NN	-	0.001574	0.229104	0.303273
AD	0.001574	-	0.303273	0.000000
NB	0.229104	0.303273	-	0.001574
MLP	0.303273	0.000000	0.001574	-

A análise da matriz mostra os pares de classificadores com diferenças estatisticamente significativas:

- MLP vs. AD (p ¡ 0.0001): Diferença significativa; MLP é estatisticamente superior à Árvore de Decisão.
- MLP vs. NB (p = 0.0016): Diferença significativa; MLP é estatisticamente superior ao Naive Bayes.
- k-NN vs. AD (p = 0.0016): Diferença significativa; k-NN é estatisticamente superior à Árvore de Decisão.
- MLP vs. k-NN (p = 0.3033): Sem diferença significativa; MLP e k-NN são considerados estatisticamente equivalentes.
- k-NN vs. NB (p = 0.2291): Sem diferença significativa; k-NN e Naive Bayes são estatisticamente equivalentes.
- NB vs. AD (p = 0.3033): Sem diferença significativa; Naive Bayes e Árvore de Decisão são estatisticamente equivalentes.

Esses resultados confirmam que o MLP obteve o melhor desempenho estatisticamente significativo sobre dois dos três classificadores concorrentes (AD e NB), en-

quanto se manteve estatisticamente equivalente ao k-NN. Abaixo o diagrama de CD para visualização melhor do teste.

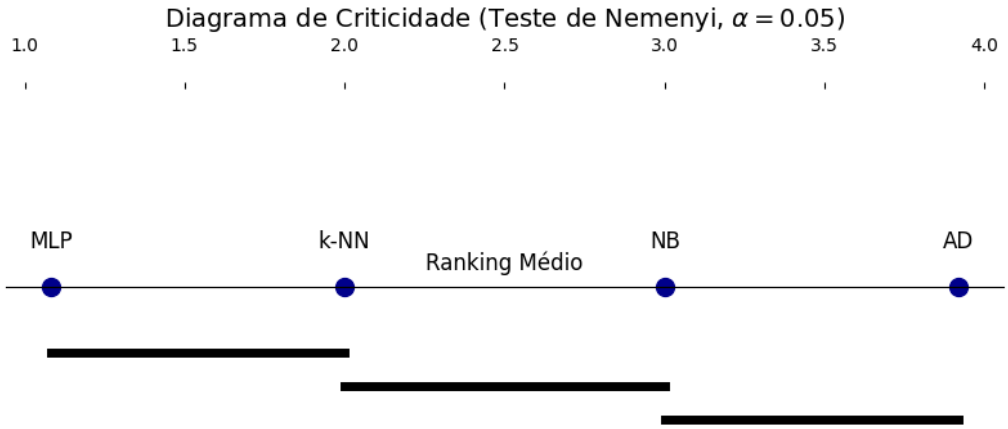

Figura 18. Diagrama CD - Modelos tradicionais

Da mesma forma foram analisados os comitês de classificadores via teste estátisco. As duas hipoteses descritas também foram equivalente as anteriores:

- Hipótese Nula (H): Todos os comitês apresentam, em média, o mesmo desempenho.
- Hipótese Alternativa (H): Pelo menos um dos comitês apresenta desempenho médio significativamente diferente dos demais.

Após a ação do teste de Fredman foi evidenciado que existem diferença entre os comitês de classificadores, o p-value apresentado foi de 0.00000098 derrubando a hipotese nula.

Listing 3. Execução do Teste de Friedman para os classificadores base.

```
from scipy.stats import friedmanchisquare

# Acur cias para cada classificador nas 12 bases
bagging = [96.58, 97.28, 92.95, 95.95, 92.71, 95.43, 98.88,
   98.46, 92.64, 97.97, 93.44, 97.73]
boosting = [97.21, 95.81, 93.16, 91.63, 89.12, 94.56, 98.04,
   97.76, 91.77, 96.79, 94.14, 96.93]
random_forest = [98.74, 99.16, 94.14, 98.74, 95.81, 98.32,
   99.16, 97.90, 94.56, 98.74, 94.97, 97.90]
stacking = [99.12, 98.87, 96.23, 98.87, 95.97, 99.55, 99.25,
   98.87, 95.72, 99.12, 96.10, 98.99]

# Aplicando o teste de Friedman
stat, p = friedmanchisquare(bagging, boosting, random_forest,
   stacking)

print(f'Estat stica de Friedman: {stat:.4f}')
```

```
print(f'p-valor: {p:.8f}')

Estatística de Friedman: 30.7000
p-valor: 0.00000098
```

Uma vez estabelecida a diferença foi novamente realizado o teste post-hoc para onde ocorre essa diferença. A Tabela 14 mostra o resultado do teste Nimenyi.

Tabela 14. Matriz de p-valores do Teste de Nemenyi entre os métodos de ensemble.

	Bagging	**Boosting**	**Random Forest**	**Stacking**
Bagging	-	0.685342	0.119495	0.000449
Boosting	0.685342	-	0.004980	0.000003
Random Forest	0.119495	0.004980	-	0.303273
Stacking	0.000449	0.000003	0.303273	-

A comparação estatística entre as abordagens de comitê revelou uma hierarquia de desempenho interessante. As técnicas Stacking e Random Forest formaram o grupo de elite, sem apresentar diferença estatística significativa entre si (p = 0.3033).

O Boosting, em contrapartida, demonstrou ser a abordagem menos eficaz neste cenário, com um desempenho estatisticamente inferior tanto ao Stacking (p ¡ 0.0001) quanto ao Random Forest (p = 0.0050). O Bagging ocupou uma posição intermediária, sendo estatisticamente equivalente ao Boosting (p = 0.6853) e ao Random Forest (p = 0.1195), mas inferior ao Stacking (p = 0.0004). Estes resultados sugerem que as estratégias que promovem maior diversidade entre modelos heterogêneos (Stacking) ou entre árvores decorrelacionadas (Random Forest) foram as mais bem-sucedidas. O diagrama CD abaixo, evidencia bem esse comportamento.

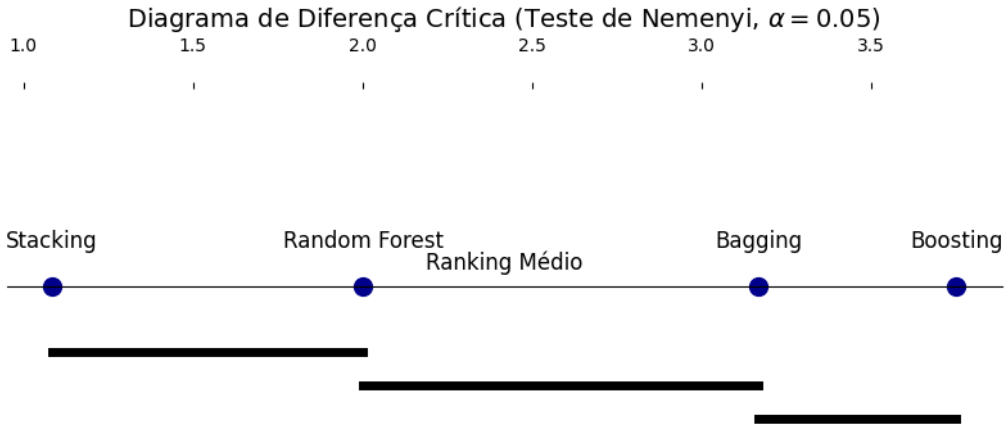

Figura 19. Diagrama CD - Comitês de Classificadores

7. Conclusões

Referências

Fukunage, K. and Narendra, P. M. (1975). A branch and bound algorithm for computing k-nearest neighbors. *IEEE Trans. Comput.*, 24(7):750–753.

GeeksforGeeks (2023). A comprehensive guide to ensemble learning. Acesso em: 08 jul. 2025.

IBM (2025). Decision trees: o que são e como funcionam. `https://www.ibm.com/br-pt/think/topics/decision-trees`. Acesso em: 8 jul. 2025.

MarketMuse (2024). Multilayer perceptron – definition. Acesso em: 9 jul. 2025.

MBA USP ESALQ (2024). Algoritmos de redes neurais: como funcionam e quais os principais. Acesso em: 9 jul. 2025.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):160.