Análise do comportamento de técnicas supervisionadas aplicadas a classificação de imagens

Carlos Eduardo Dos Santos Garabito[1], Davi Dutra Ferreira[1], Felipe Augusto Cortez Muniz da Silva[1]

[1]Instituto Metrópole Digital – Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal 1524 – 59078-970– Natal – RN – Brazil

{eduardo.garabito21@gmail.com, davidutrajprn@hotmail.com,
felipe.silva.088@ufrn.edu.br

Abstract. *This study evaluates the effectiveness of various supervised learning algorithms for image classification, focusing on differentiating between cat and dog images. We experimented with different feature extraction techniques (HOG and CNN) and classification methods (k-NN, Decision Trees, Naive Bayes, MLP, and ensemble methods). Our findings indicate that while Random Forest achieved the highest average accuracy, the overall performance of the models was influenced by factors such as preprocessing techniques and parameter tuning. Statistical tests revealed no significant differences between many of the algorithms, suggesting that the choice of model may be less critical than careful feature engineering and hyperparameter optimization. These results highlight the importance of tailored approaches to image classification problems and emphasize the need for further research into optimizing deep learning models for specific datasets.*

Resumo. *Este estudo avalia a eficácia de diversos algoritmos de aprendizado supervisionado para a classificação de imagens, com foco na distinção entre imagens de gatos e cães. Experimentamos diferentes técnicas de extração de características (HOG e CNN) e métodos de classificação (k-NN, Árvores de Decisão, Naive Bayes, MLP e métodos ensemble). Nossos resultados indicam que, embora o Random Forest tenha atingido a maior acurácia média, o desempenho geral dos modelos foi influenciado por fatores como técnicas de pré-processamento e ajuste de hiperparâmetros. Testes estatísticos revelaram que não houve diferenças significativas entre muitos dos algoritmos, sugerindo que a escolha do modelo pode ser menos crítica do que a engenharia de características cuidadosa e a otimização de hiperparâmetros. Esses resultados destacam a importância de abordagens personalizadas para problemas de classificação de imagens e enfatizam a necessidade de pesquisas futuras para otimizar modelos de aprendizado profundo para conjuntos de dados específicos.*

1. Introdução

Este relatório apresenta as atividades desenvolvidas na disciplina de Aprendizado de Máquina, com foco na elaboração de modelos de classificação aplicados à identificação de imagens de cães e gatos. A visão computacional é uma das áreas mais promissoras

dentro do campo de aprendizado de máquina, devido à sua capacidade de identificar padrões, comportamentos e agrupamentos em imagens. Essa tecnologia tem se consolidado como uma ferramenta indispensável, com aplicações que vão desde a detecção de tumores em exames de ultrassom até tarefas mais simples, como as abordadas neste trabalho.

O principal objetivo deste projeto foi demonstrar o conhecimento adquirido ao longo do curso por meio da construção, avaliação e comparação de modelos de classificação. Além disso, realizamos testes estatísticos para avaliar se algum dos modelos desenvolvidos apresenta desempenho significativamente superior aos demais.

2. Base de Dados

A base de dados utilizada neste trabalho foi um repositório público disponibilizado pela Universidade de Oxford, contendo diversas imagens de cães e gatos de diferentes raças. Para este projeto, nosso grupo foi responsável por trabalhar com as imagens das raças abissínio e sphynx, no caso dos gatos, e basset hound e chihuahua, no caso dos cães. No total, foram analisadas 800 imagens, sendo 400 pertencentes à classe "gatos" e 400 à classe "cães".

3. Pré-processamento

Antes de iniciar a modelagem, foi necessário passar as imagens por modelos de redes profundas capazes de processá-las e transformá-las em dados compreensíveis pela máquina. Esse processo permitiu que os modelos de classificação aprendessem os padrões que distinguem as duas espécies (cães e gatos). Para isso, utilizamos duas abordagens principais de visão computacional: HOG e CNN, aplicadas com diferentes configurações de parâmetros, o que resultou em 16 bases de dados distintas.

3.1. HOG

O Histogram of Oriented Gradients (HOG) foi uma das técnicas utilizadas para o processamento das imagens. Essa técnica, amplamente aplicada em visão computacional, extrai características identificando a distribuição das direções dos gradientes de intensidade em uma imagem, que representa bordas e formas.

O HOG divide a imagem em pequenas regiões e calcula, para cada pixel, a direção e a intensidade das mudanças de brilho (gradientes). Essas informações são organizadas em histogramas, que indicam quais direções de borda são mais comuns em cada região. Os dados são então normalizados para reduzir o impacto de variações de iluminação e contraste, e os histogramas de todas as regiões são combinados em um único vetor que representa a imagem.

Para o processamento, foram configuradas 4 combinações de parâmetros:

- Tamanho da imagem: 128x128 e 256x256 pixels.
- Quantidade de pixels por célula: 16x16 e 20x20 pixels.

3.2. CNN

As Redes Neurais Convolucionais (CNNs) foram a outra abordagem utilizada para o processamento. As CNNs são redes projetadas para processar dados estruturados em grade, como imagens, e funcionam extraindo características automaticamente por meio

de convoluções. Esses filtros destacam bordas, texturas e formas, passando por diversas camadas para identificar informações cada vez mais complexas. Além disso, etapas como pooling ajudam a reduzir a dimensionalidade, simplificando os dados sem perder informações relevantes.

As CNNs ajustam seus filtros durante o treinamento, aprendendo quais características são mais importantes para resolver o problema, o que as torna extremamente eficazes em tarefas de visão computacional.

Para este trabalho, utilizamos 8 configurações de parâmetros diferentes, baseadas nas seguintes variáveis:

- Tamanho da imagem: 128x128 e 256x256 pixels.
- Arquiteturas:
    - VGG16 (13 camadas convolucionais + 3 camadas totalmente conectadas).
    - VGG19 (16 camadas convolucionais + 3 camadas totalmente conectadas).
- Tipo de pooling:
    - MAX: Seleciona o maior valor dentro da janela.
    - AVG: Calcula a média dos valores na janela.

Essas configurações permitiram comparar o impacto de diferentes combinações de parâmetros na qualidade das características extraídas, formando um conjunto robusto de bases de dados para análise.

4. Metodologia dos Experimentos

Após o processamento das bases de dados, o processo de modelagem foi dividido em cinco sessões: k-NN, Árvore de Decisão, Naive Bayes, MLP e Comitê de Classificadores. Para cada uma das bases, avaliamos dois cenários diferentes de separação dos dados para treinamento e teste: holdout (70/30) e k-fold com k=10. Na primeira sessão, foi aplicado o PCA nas seis melhores bases, e essas bases foram substituídas nas seis piores. Esse mesmo procedimento foi seguido para todas as outras sessões de experimentos.

Ao final, foram realizados testes estatísticos de Friedman e Nemenyi para verificar se existia algum modelo ou definição que se destacasse em relação aos demais. O primeiro teste foi realizado entre os modelos e, em seguida, entre as melhores definições de cada modelo. A métrica utilizada para todas as etapas foi a acurácia, que representa a proporção de previsões corretas em relação ao total de previsões feitas por um modelo.

4.1. k-NN

O k-NN (k-Nearest Neighbors) é um algoritmo de aprendizado supervisionado utilizado para tarefas de classificação e regressão. Sua abordagem se baseia na ideia de que dados semelhantes estão localizados próximos uns dos outros no espaço das características. Para classificar uma nova instância, o algoritmo identifica os k vizinhos mais próximos no conjunto de treinamento e atribui à instância o rótulo mais comum entre esses vizinhos. A escolha do valor de k é crucial: valores pequenos tornam o modelo sensível a ruídos,

enquanto valores grandes podem suavizar demais a decisão.

Para este estudo, foram realizadas 10 configurações diferentes, variando o valor de k de 1 a 10.

4.1.1 PCA

Após a coleta das acurácias, selecionamos as seis melhores bases e aplicamos o PCA nelas, substituindo as seis piores bases. As seis melhores bases incluíram duas com HOG e quatro com CNN. Em seguida, as acurácias dessas novas bases foram coletadas, e essa definição de base de dados foi mantida para todas as etapas subsequentes.

O PCA (Principal Component Analysis) é uma técnica de redução de dimensionalidade que transforma um conjunto de variáveis correlacionadas em um conjunto de variáveis não correlacionadas, denominadas componentes principais, preservando a maior parte da variabilidade dos dados.

4.2. Decision Tree

A Árvore de Decisão é um algoritmo de aprendizado supervisionado utilizado em tarefas de classificação e regressão. Ela constrói um modelo em formato de árvore, onde cada nó interno representa uma decisão baseada em uma característica dos dados, e cada ramo corresponde a uma possível resposta ou resultado dessa decisão. As folhas da árvore indicam a classificação ou o valor final. O algoritmo constrói a árvore por meio de uma divisão recursiva dos dados, utilizando critérios que maximizam a separação das classes ou minimizam o erro, como o índice de Gini ou a entropia. Apesar de sua facilidade de interpretação, as Árvores de Decisão podem sofrer de overfitting, especialmente quando são excessivamente profundas.

Para melhorar o desempenho da árvore e reduzir o overfitting, foi aplicada a técnica de pós-poda, que consiste em remover ramos da árvore que têm pouca ou nenhuma contribuição para o aumento da acurácia do modelo. A poda ocorre após a construção inicial da árvore e visa reduzir o ajuste excessivo aos dados de treinamento, melhorando assim a capacidade de generalização do modelo para novos dados. Esse processo é realizado com base em critérios como a minimização do erro de validação ou a complexidade do modelo, resultando em uma árvore mais simples e eficiente.

No experimento, foram testados 9 modelos diferentes, variando o parâmetro da profundidade da árvore entre 2 e 10.

4.3. Naive Bayes

O Naive Bayes é um algoritmo de aprendizado supervisionado amplamente utilizado em

tarefas de classificação. Ele se baseia no teorema de Bayes, que calcula a probabilidade de uma classe dada as características observadas dos dados. O termo "naive" (ingênuo) refere-se à suposição simplificadora de que as características dos dados são independentes entre si, o que torna os cálculos mais eficientes, apesar de nem sempre refletir a realidade.

Foram utilizados 3 diferentes modelos: GussianNB, MultinomialNB e Complement NB.

4.4. MLP

O Perceptron Multicamadas (MLP) é um tipo de rede neural composto por várias camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio de uma camada recebe sinais de entrada, realiza um cálculo baseado em uma função de ativação e transmite o resultado para a camada seguinte. Durante o treinamento, os pesos das conexões entre os neurônios são ajustados através de um processo chamado retropropagação, que visa minimizar a diferença entre a saída prevista e a saída real. Esse processo de ajuste gradativo dos pesos permite que o modelo aprenda de forma eficaz, melhorando sua acurácia ao longo do tempo.

As redes neurais são sistemas computacionais inspirados no funcionamento do cérebro humano, projetados para reconhecer padrões e aprender a partir de exemplos. Compostas por camadas de neurônios interconectados, as redes neurais são capazes de representar relações complexas e não lineares nos dados, o que as torna particularmente eficazes em tarefas como classificação e regressão. O MLP, como um tipo de rede neural, é amplamente utilizado em diversas áreas, como reconhecimento de imagens, processamento de linguagem natural e análise de dados complexos.

A definição dos modelos e suas configurações ficou a cargo do grupo. Para garantir que os modelos convergissem de forma eficiente em qualquer banco de dados, estabelecemos um limite de 5000 iterações e utilizamos o GridSearch, uma técnica que busca as melhores configurações de modelos. A escolha foi feita na base com o maior número de colunas, visando definir a quantidade de neurônios de forma consistente, considerando o cenário mais desafiador. Após a busca pelos melhores parâmetros, selecionamos as cinco melhores configurações e avaliamos o desempenho de todas as bases nesses modelos. Estes foram os modelos utilizados:

```
best_models = [
    MLPClassifier(activation= 'relu', hidden_layer_sizes= (442, 442), learning_rate_init= 0.01, max_iter= 5000, solver= 'sgd'),
    MLPClassifier(activation= 'tanh', hidden_layer_sizes= (442, 442), learning_rate_init= 0.1, max_iter= 5000, solver= 'adam'),
    MLPClassifier(activation= 'relu', hidden_layer_sizes= (442, 442), learning_rate_init= 0.001, max_iter= 5000, solver= 'sgd'),
    MLPClassifier(activation= 'logistic', hidden_layer_sizes= (1767,), learning_rate_init= 0.1, max_iter= 5000, solver= 'adam'),
    MLPClassifier(activation= 'relu', hidden_layer_sizes= (884, 883), learning_rate_init= 0.001, max_iter= 5000, solver= 'sgd')
]
```

4.5. Comitê de Classificadores

Um comitê de classificadores é uma abordagem que combina as previsões de múltiplos modelos de classificação para melhorar a acurácia geral e reduzir o risco

de erros. A ideia é que, ao reunir diferentes modelos, o comitê pode aproveitar as forças de cada um e corrigir as falhas dos outros. Essa combinação pode ser feita de diferentes formas, como por votação, onde a classe mais votada pelos modelos é a escolhida, ou por média ponderada, dependendo do tipo de problema e dos modelos utilizados. A utilização de um comitê de classificadores é útil especialmente quando os modelos individuais têm desempenho semelhante, mas suas falhas são complementares, o que melhora a robustez e a generalização do sistema.

Para o experimento, foram utilizadas 5 técnicas diferentes de comitês: Bagging, Boosting, Random Forest, Stacking e Voting.

4.5.1 Bagging

O Bagging, ou Bootstrap Aggregating, é uma técnica de aprendizado de máquina projetada para melhorar a precisão e a estabilidade de modelos preditivos, especialmente em algoritmos que podem ser altamente sensíveis às variações nos dados de treinamento, como as Árvores de Decisão. O processo começa com a criação de múltiplos subconjuntos de dados a partir do conjunto original, utilizando amostragem com reposição (bootstrap), o que significa que alguns dados podem ser repetidos enquanto outros ficam de fora. Para cada subconjunto gerado, um modelo é treinado, e as previsões de todos os modelos são combinadas, geralmente por votação no caso de classificação ou por média para regressão. Essa abordagem reduz a variabilidade e melhora a capacidade de generalização do modelo final, tornando-o mais robusto e menos propenso ao overfitting.

No experimento, utilizou-se o Bagging com os classificadores k-NN, Árvores de Decisão, Naive Bayes e MLP, empregando as configurações de modelo que apresentaram maior acurácia nos testes anteriores. Para cada configuração, o número de estimadores foi variado entre 10 (padrão), 20 e 30. Em seguida, o experimento foi repetido utilizando o parâmetro *max_features=0.5* no Bagging, a fim de avaliar o impacto dessa alteração na acurácia do modelo e comparar os resultados com a configuração anterior.

4.5.2. Boosting

O Boosting é uma técnica de aprendizado de máquina que combina múltiplos modelos fracos (modelos que têm desempenho ligeiramente melhor que o acaso) para formar um modelo forte e mais preciso. Diferente do Bagging, que treina modelos de forma independente, o Boosting treina os modelos sequencialmente, com cada novo modelo corrigindo os erros do anterior. A ideia central é focar nos exemplos que foram mal classificados pelos modelos anteriores, ajustando o peso desses exemplos para garantir que o próximo modelo os considere de forma mais cuidadosa. O Boosting é eficaz em melhorar a acurácia e reduzir o erro, sendo especialmente útil para problemas complexos, mas pode ser sensível ao overfitting se não for bem regularizado. Para o experimento, foi utilizado o AdaBoost.

Utilizou-se o Boosting com os classificadores Árvores de Decisão e Naive Bayes, empregando as configurações de modelo que apresentaram maior acurácia nos testes anteriores. Para cada configuração, o número de estimadores foi variado entre 10 (padrão), 20 e 30.

4.5.3. Random Forest

O Random Forest é um algoritmo de aprendizado de máquina que utiliza um conjunto de Árvores de Decisão para realizar tarefas de classificação e regressão. Cada árvore é treinada em um subconjunto aleatório de dados e características, o que permite que o modelo combine os resultados de várias árvores para obter previsões mais robustas e precisas. Durante a construção das árvores, utiliza-se a amostragem com reposição (bootstrap) para gerar diferentes subconjuntos de dados, e, em cada divisão, uma seleção aleatória de um subconjunto de características é realizada, o que aumenta a diversidade entre as árvores e minimiza o risco de overfitting.

O Random Forest é altamente eficaz, sendo capaz de lidar com uma ampla gama de problemas e resistente a ruídos nos dados. Sua principal vantagem é a capacidade de generalização, uma vez que a combinação de diversas árvores reduz a variação e melhora a precisão das previsões.

No experimento, foram testadas três configurações diferentes, variando os critérios de avaliação entre Gini, Entropy e Log Loss. Para cada uma dessas configurações, também foram ajustados o número de árvores, com quantidades de 10, 20, 30 e 100 árvores, para verificar como o desempenho do modelo é afetado por essas mudanças.

4.5.4. Stacking

O Stacking é uma técnica de aprendizado de máquina que visa combinar múltiplos modelos, podendo ser de tipos diferentes ou semelhantes, com o objetivo de melhorar a precisão das previsões. A estratégia consiste em treinar diversos modelos base (de nível 0) no conjunto de dados e, em seguida, utilizar suas previsões como entradas para um modelo final (de nível 1), que é responsável por realizar a decisão final. O modelo de nível 1 aprende a combinar de forma otimizada as saídas dos modelos base, ajustando seus próprios parâmetros para maximizar a performance geral. O Stacking é eficaz porque tira proveito das forças de diferentes modelos, compensando suas fraquezas, o que geralmente resulta em um desempenho superior ao de qualquer modelo individual.

No experimento, optamos por utilizar os mesmos métodos empregados nas etapas de Bagging e Boosting, com a adição de uma configuração simplificada do MLP, a fim de reduzir o tempo de processamento. Foram realizadas quatro configurações diferentes, variando o número de classificadores em 5, 10, 15 e 20, para analisar o impacto dessa variação no desempenho do modelo final.

4.5.5. Voting

Voting é uma técnica de aprendizado de máquina em que múltiplos modelos são combinados para fazer uma previsão final com base nas previsões individuais de cada modelo. Em um modelo de votação, as previsões dos classificadores são combinadas de duas formas principais: votação majoritária (para classificação) ou média (para regressão). No caso da votação majoritária, a classe que receber o maior número de votos entre os modelos é escolhida como a previsão final. A principal vantagem dessa técnica é que ela melhora a robustez do modelo, pois combina a diversidade dos classificadores, o

que tende a reduzir o risco de erro e aumentar a precisão geral.

Para o experimento, as configurações utilizadas no modelo de Voting foram as mesmas aplicadas no Stacking, ou seja, foi feita uma configuração simples do MLP e variado o número de classificadores em 5, 10, 15 e 20, com o objetivo de comparar os resultados obtidos entre as duas abordagens.

4.6. Teste Estatístico

Testes estatísticos são métodos utilizados para analisar e tirar conclusões sobre um conjunto de dados com base em evidências numéricas. Eles ajudam a verificar hipóteses e determinar se os resultados observados são significativos ou se podem ter ocorrido por acaso. Em aprendizado de máquina, os testes estatísticos são frequentemente usados para comparar a performance de diferentes modelos ou abordagens, garantindo que as diferenças observadas não sejam fruto de flutuações nos dados, mas sim resultados reais.

Nos experimentos foram utilizados os testes de Friedman e o teste de Nemenyi. O teste de Friedman é utilizado para comparar a performance de múltiplos modelos em diferentes conjuntos de dados, sendo particularmente útil quando há mais de dois grupos a serem comparados. Já o teste de Nemenyi é uma extensão do teste de Friedman e é usado para realizar comparações post-hoc entre os modelos quando o teste de Friedman indica diferenças significativas, ajudando a identificar quais modelos são estatisticamente superiores uns aos outros. Esses testes garantem que as conclusões obtidas sobre os modelos são robustas e não apenas acidentais.

Inicialmente, os testes foram aplicados para comparar as configurações de modelos utilizados entre os métodos. Após essa análise, os melhores modelos de cada método foram selecionados, e os testes foram repetidos para comparar essas seleções e determinar, com base em evidências estatísticas, se era possível afirmar a superioridade de algum modelo em relação aos outros.

5. Resultados

Os métodos avaliados não apresentaram diferenças significativas em termos de desempenho. De maneira geral, as acurácias obtidas variaram entre 55% e 70%. Considerando que se tratava de um problema de classificação binária balanceada em visão computacional, esses resultados ficaram abaixo do esperado, o que pode ser considerado uma performance desapontante dado o potencial da área e a natureza do problema.

De todos os modelos, aquele com a maior média foi o Random Forest, com 64,68%.

5.1. Testes Estatísticos (Entre Parâmetros)

5.1.1. k-NN

Embora o teste de Friedman tenha apresentado um p-value de 0%, indicando diferenças estatisticamente significativas entre os métodos, o teste de Nemenyi não revelou nenhuma

configuração claramente superior ou inferior em relação às outras, do ponto de vista estatístico. Como resultado, foi escolhida a configuração com k=8, pois essa apresentou a maior média de acurácia entre as diferentes abordagens.

5.1.2. Decision Tree

O teste de Friedman obteve um p-value de 70%, o que indica que há uma diferença muito pequena entre as configurações avaliadas. Diante disso, a configuração com profundidade máxima igual a 8 foi selecionada, pois apresentou a maior média de acurácia entre todas as opções testadas.

5.1.3. Naive Bayes

O teste de Friedman para o Naive Bayes obteve um p-value de 0%, indicando uma diferenciação significativa entre as configurações. Ao realizar o teste de Nemenyi, observou-se que havia, de fato, mudanças substanciais entre os modelos, com p-values baixos entre as configurações. No entanto, mesmo com essas diferenças, não foi possível afirmar com 95% de confiança que um modelo fosse superior aos outros. Dessa forma, os resultados do modelo com a maior média de acurácia, que foi o ComplementNB, foram selecionados para a análise final.

5.1.4. MLP

O teste de Friedman revelou um p-valor de 0%, indicando diferenças significativas entre os modelos analisados. Após a aplicação do teste de Nemenyi, constatou-se que o modelo 5 apresentou o melhor desempenho geral. Com base nos resultados, é possível afirmar, com 95% de confiança, que o modelo 5 é superior aos modelos 2 e 4. No entanto, para os demais modelos, não foi possível obter evidências estatísticas suficientes (p-valor < 5%) para estabelecer uma diferença significativa. Apesar disso, o modelo 5 foi selecionado para testes adicionais com outros métodos posteriormente.

5.1.5. Bagging

Para o Bagging, selecionamos os modelos configurados com *max_features = 0.5*. Inicialmente, avaliamos os diferentes métodos de classificação, selecionando o melhor entre eles. Em seguida, realizamos testes variando o número de estimadores (*n_estimators*) para o método selecionado.

No caso dos métodos, o teste de Friedman apresentou um p-valor de 44%, indicando pouca diferença significativa entre eles. Ainda assim, optamos pelo Bagging combinado com k-NN, que obteve o melhor desempenho geral. A partir dessa combinação, realizamos uma nova análise sobre o impacto do número de estimadores. Nesse caso, o teste de Friedman apontou um p-valor superior a 90%, sugerindo que a alteração desse parâmetro tem pouco efeito no desempenho do modelo. Como procedimento padrão, selecionamos o modelo configurado com 30 estimadores para os testes subsequentes.

5.1.6. Boosting

Para o Boosting, seguimos os mesmos procedimentos utilizados no Bagging. Avaliamos os diferentes métodos de classificação, selecionando o melhor entre eles, e posteriormente realizamos testes variando o número de estimadores (*n_estimators*) para o método escolhido.

Como a comparação inicial envolveu apenas dois métodos, o teste de Friedman não foi aplicado, pois exige pelo menos três amostras para sua execução. Assim, utilizamos o teste de Nemenyi, que indicou, com 95% de confiança, que o Boosting com Árvore de Decisão apresentou desempenho superior ao Boosting com Naive Bayes.

Com base no melhor método selecionado (Boosting com Árvore de Decisão), realizamos testes para avaliar o impacto do número de estimadores. O teste de Friedman indicou um p-valor de 0%, revelando diferenças significativas entre as três configurações avaliadas. Com 95% de confiança, podemos afirmar que o Boosting com 10 estimadores foi o pior entre as configurações. Já o modelo com 30 estimadores quase se destacou como o melhor, com um p-valor de 7% em relação ao modelo com 20 estimadores, o que não foi suficiente para confirmar uma diferença estatisticamente significativa. De qualquer forma, os resultados do modelo com 30 estimadores foram selecionados para os testes futuros.

5.1.7. Random Forest

Para o Random Forest, os testes foram realizados inicialmente para avaliar os critérios de divisão, selecionando o melhor modelo. Em seguida, repetimos os testes variando o número de árvores utilizadas.

Quanto aos critérios de divisão, o teste de Friedman apresentou um p-valor de 67%, indicando pouca diferença significativa entre eles. Apesar disso, selecionamos o critério Entropy, que obteve a maior média de desempenho.

Na análise do número de árvores utilizando o critério selecionado, o teste de Friedman revelou diferenças significativas, com um p-valor de 0%. Após aplicar o teste de Nemenyi, foi possível afirmar que o modelo com 10 árvores é estatisticamente o pior, pois, além de apresentar a pior média, mostrou diferenças significativas em relação a todos os outros modelos. Entre os demais modelos (com 20, 50 e 100 árvores), não foi possível identificar diferenças estatísticas significativas, uma vez que nenhum dos testes atingiu o critério de p-valor < 5%. Como procedimento padrão, selecionamos o modelo com a maior média de desempenho, que foi o modelo configurado com 100 árvores.

5.1.8. Stacking e Voting

Com os métodos Stacking e Voting, ocorreu um fato curioso: o número de classificadores não teve qualquer impacto na acurácia. Em todas as configurações testadas, o desempenho foi exatamente o mesmo, independentemente do modelo ou do número de classificadores utilizados.

Dessa forma, não foi necessário realizar testes estatísticos para comparar as configurações, já que as alterações nos parâmetros não influenciaram os resultados. Assim, optamos por salvar os resultados de uma única configuração de cada método (Stacking e Voting) para utilizá-los nos testes comparativos entre métodos.

5.2. Testes Estatísticos (entre métodos)

Com a etapa anterior concluída, selecionamos 9 modelos, correspondentes aos 9 métodos diferentes, sendo cada modelo aquele com o melhor desempenho médio. Realizamos o teste de Friedman para comparar esses modelos, que resultou em um p-valor de 0%, indicando diferenças estatisticamente significativas entre eles.

Em seguida, aplicamos o teste de Nemenyi para identificar as diferenças específicas entre os modelos. Os resultados deste teste nos permitem afirmar que:

- Não há um modelo que seja estatisticamente o melhor ou o pior em relação a todos os outros, embora as médias das acurácias variem entre os métodos.
- O Random Forest, que obteve os melhores resultados médios, é estatisticamente superior aos métodos DecisionTree, NaiveBayes, MLP e Voting.
- O MLP, que apresentou os piores resultados médios, é estatisticamente inferior aos métodos Random Forest e Stacking.

6. Discussões

Um fator que pode ter influenciado os resultados foi o pré-processamento das imagens. Acreditamos que seria possível melhorar o desempenho realizando um fine-tuning dos parâmetros do método CNN, já que, em todas as suas bases, cerca de 90% dos valores resultantes eram completamente zeros, indicando um possível problema na extração de características.

Em relação aos métodos, a maior dificuldade foi observada com o MLP. Além do tempo de processamento, que demandou praticamente um dia inteiro para processar todos os dados, o método apresentou o pior desempenho médio em termos de acurácia. Uma sugestão seria realizar um fine-tuning dos hiper parâmetros mais sensíveis para ajustar o modelo a cada base de dados de forma personalizada. Trabalhar com um único modelo de rede neural para bases com formatos completamente diferentes é pouco realista, e modelos específicos para cada base poderiam trazer resultados mais consistentes.

Em suma, quando se trabalha com redes neurais, é essencial ter mais cuidado no ajuste dos hiper parâmetros e no design do modelo. Esses métodos são poderosos, mas altamente sensíveis a fatores como a qualidade dos dados, o pré-processamento e a configuração dos parâmetros. Uma abordagem criteriosa pode fazer uma grande diferença nos resultados.

7. Conclusão

Em resumo, os experimentos realizados com diferentes métodos de classificação para a

identificação de imagens de cães e gatos mostraram que, embora os métodos mais tradicionais como o Random Forest e o k-NN tenham obtido os melhores resultados médios, não houve uma diferença significativa entre muitos dos modelos analisados. A análise estatística, incluindo os testes de Friedman e Nemenyi, revelou que, em muitos casos, a escolha de parâmetros não teve um impacto expressivo no desempenho final dos modelos, especialmente no caso do Stacking e do Voting, onde o número de classificadores não alterou os resultados. Contudo, o Random Forest destacou-se como o modelo com melhor desempenho médio, enquanto o MLP apresentou os piores resultados, reforçando a necessidade de ajustes mais precisos, especialmente em redes neurais.

Uma das principais dificuldades encontradas foi relacionada ao pré-processamento das imagens, onde a técnica de CNN necessitou de ajustes nos parâmetros devido à excessiva quantidade de zeros nos valores extraídos. Além disso, o MLP enfrentou problemas de tempo de processamento, que prejudicaram o desempenho geral, o que sugere que uma abordagem mais focada na personalização de modelos para cada base poderia trazer resultados mais consistentes. O fine-tuning dos hiperparâmetros e a personalização do modelo para cada conjunto de dados são essenciais para o sucesso de redes neurais, que, apesar de seu potencial, são altamente sensíveis à qualidade do pré-processamento e à configuração dos parâmetros.

Por fim, apesar dos desafios enfrentados e da performance abaixo do esperado, os resultados reforçam a importância de um processo criterioso de seleção de métodos e ajuste de parâmetros em tarefas de classificação. O aprendizado obtido com esses experimentos pode ser valioso para futuras investigações, sugerindo que a combinação de técnicas de pré-processamento mais sofisticadas e a personalização de modelos são caminhos promissores para melhorar o desempenho em problemas de classificação de imagens, particularmente em cenários com características variadas.

8. Referências

1. ***Hastie, T., Tibshirani, R., & Friedman, J. (2009).*** *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.*
2. ***Bishop, C. M. (2006).*** *Pattern Recognition and Machine Learning. Springer.*
3. ***Shalev-Shwartz, S., & Ben-David, S. (2014).*** *Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.*