Relatório do Projeto de Aprendizado de Máquina

Cauã R. Brasil[1] , Dinorah F. Chagas[1], Victor A. Gomes [1]

[1]Instituto Metrópole Digital – Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal 1524 – 59078-900 – Natal – RN – Brazil

caua.brasil.017@ufrn.edu.br, dinorah.farias.080@ufrn.edu.br,
victor.gomes.090@ufrn.edu.br

Abstract. This meta-article describes and analyzes the final project of the Aprendizado de Máquina discipline through analysis of accuracies and statistical methods. A project that aims to use machine learning models to identify whether the image received of a dog of the same breed. Pug or American Bulldog, or a Bengal or a Ragdoll cat.

Resumo. Este meta-artigo descreve e analisa o projeto final da disciplina de Aprendizado de Máquina por meio de análise de acurácias e métodos estatísticos. Projeto esse que possui o objetivo de utilizar modelos de machine learning para identificar se a imagem recebida é de um cachorro da raça Pug ou American Bulldog, ou um gato da raça Ragdoll ou Bengal.

1. Introdução

Esse projeto busca resolver o problema de classificação de imagens, com o objetivo de identificar a espécie e a raça de animais, especificamente gatos e cachorros. A relevância desse problema reside na aplicação potencial em sistemas de reconhecimento de animais para áreas como controle de animais domésticos, monitoramento de raças para fins de pesquisa genética, ou até mesmo em plataformas de adoção, onde é crucial identificar corretamente o tipo e a raça dos animais. Caso esse problema seja resolvido, seria possível automatizar a categorização de animais com grande precisão, facilitando a interação com sistemas de imagem e melhorando processos em diversas áreas.

O problema aparece com frequência em contextos como bancos de imagens, sistemas de monitoramento, e-commerce de pets, entre outros. Para abordar a solução, foi utilizada uma base de dados composta por imagens de gatos e cachorros, com suas respectivas raças: Bengal e Ragdoll para gatos, e Pug e American Bulldog para cachorros. Dados adicionais incluem arquivos CSV gerados a partir dessas imagens, contendo informações relevantes para a modelagem. O conhecimento de domínio em machine learning e o contexto também se beneficiam de exemplos resolvidos de classificação de imagens em tarefas semelhantes.

2. Descrição do Problema e Base de Dados

2.1. Descrição do Problema

O foco principal desse projeto é determinar qual dos modelos de machine learning utilizados possui maior eficácia em a partir da base de dados oferecida de imagens, identificar se os animais são cachorros ou gatos, e qual a raça deles entre as raças supracitadas. Assim, tornando necessário lidar com 4 classes para análise.

2.2. Base de Dados

As bases de dados usadas para aplicação dos modelos, foram geradas a partir de uma pasta contendo 200 imagens de gatos Bengal, 200 gatos Ragdoll, 200 cachorros Pug e 200 cachorros American Bulldog, totalizando 400 imagens. A fim de transformar essas imagens em bases para o teste dos modelos, foram utilizados 2 tipos de transformadores, Histogram of Oriented Gradients (HOG) e o Convolutional Neural Network (CNN).

Ao todo foram criadas 18 bases de dados diferentes, 4 utilizando configurações diferentes de HOG, 8 com diferentes parâmetros de CNN e aplicando Principal Component Analysis (PCA) com 10 componentes principais em 6 bases de dados selecionadas. Dessas

18 bases, os experimentos dos modelos foram realizados com somente 12, 6 geradas pelo PCA e as 6 melhores entre as criadas pelo HOG e CNN.

3. Pré Processamento

3.1. Pré Processamento das imagens

Antes de aplicar os transformadores, foi necessário o redimensionamento das imagens em dois tamanhos diferentes, 128x128 e 256x256. Pois padronizar as imagens ajuda a manter a consistência dos resultados, otimiza os algoritmos e generaliza mais o modelo. Além disso, foi criada uma última coluna com o nome “y”, responsável por guardar as labels.

3.2. Pré Processamento KNN

Para a utilização do KNN é necessário que a label que contém as classes para treinamento possua variáveis numéricas, trazendo a necessidade de realizar essa mudança nas bases de dados, pois as label que possui as classes é categórica.

3.3. Pré Processamento Naive Bayes

Para obter resultados melhores no Gaussian Naive Bayes, a base de dados precisa possuir uma distribuição normal, ou gaussiana, portanto, para melhorar a eficácia do modelo, foi criada uma função para normalizar a base utilizando da StandartScaler da biblioteca Sklearn.

Os modelos Complement e Multinomial do Naive Bayes exigem que os dados estejam em uma distribuição discreta, tornando necessário discretizar as bases usadas com a KBinsDiscretizer da biblioteca Sklearn através de uma função.

3.4. Pré processamento MLP e Comitês

Para melhorar a eficácia do Multilayer Perceptrons (MLP), foi essencial realizar um pré-processamento rigoroso dos dados. Inicialmente, as bases utilizadas foram normalizadas aplicando o MinMaxScaler, garantindo que todas as variáveis ficassem dentro de uma escala uniforme (tipicamente entre 0 e 1). Além disso, para mitigar possíveis limitações nos dados disponíveis, como desbalanceamento ou falta de variabilidade, foi aplicada a técnica de data augmentation. Essa abordagem gerou versões aumentadas das bases originais, introduzindo ruído controlado e variações nas amostras (como rotações, espelhamentos e ajustes de brilho). Essa estratégia teve como objetivos principal aumentar a variabilidade, proporcionando ao modelo dados mais diversificados, o que melhora sua capacidade de generalização e reduz o risco de overfitting.

4. Metodologia dos Experimentos

4.1. Metodologia de Teste

Para avaliar a eficácia dos modelos K-Nearest Neighbors (KNN), Decision Tree (DT), Naive Bayes (NB) e Multilayer Perceptron (MLP), foram aplicados dois tipos principais de testes de avaliação:

1. 10-Fold Cross-Validation: O conjunto de dados foi dividido em 10 partes (folds), e o modelo foi treinado e avaliado 10 vezes, utilizando 9 folds para treinamento e 1 fold para teste em cada iteração. Esse método oferece uma avaliação mais robusta da performance dos modelos, reduzindo o viés da divisão aleatória do dataset.

2. Split-Percentage 70/30: Neste teste, o conjunto de dados foi dividido em 70% para treinamento e 30% para teste. Essa divisão foi repetida 10 vezes para fornecer uma medida mais confiável da performance do modelo em diferentes subconjuntos do dataset.

Os parâmetros principais utilizados para avaliar os modelos foram:

- Acurácia: A medida principal para avaliar o desempenho, indicando a proporção de previsões corretas feitas pelos modelos.
- Desvio padrão: Usado para mensurar a variabilidade das acurácias nas diferentes iterações dos testes, fornecendo uma noção da estabilidade do modelo em diferentes subdivisões dos dados.

Para aplicar esses métodos avaliativos, foram utilizadas accuracy_score para o cálculo da acurácia, o cross_val_score e o train_test_split para o 10-fold e split-percentage respectivamente.

Essas abordagens permitem uma análise detalhada da capacidade de generalização e robustez de cada modelo avaliado.

4.2. Metodologia K-NN

O K-Nearest Neighbors (KNN) foi implementado com a avaliação do desempenho para diferentes valores de k, variando de 1 a 10, para verificar a sensibilidade do modelo à quantidade de vizinhos.

Devido às especificações necessárias para o uso do KNN, houve a necessidade de usar o LabelEnconder para transformar os dados categóricos da label com as categorias em dados numéricos.

Essa abordagem permitiu analisar o impacto de diferentes valores de k sobre a performance do KNN, ajudando a identificar o número ideal de vizinhos para o modelo.

Os testes do modelo K-Nearest Neighbors ocorreram com os parâmetros padrões, pois, variando somente o número de vizinhos k de 1 a 10 para cada uma das 12 bases formadas pelos transformadores HOG e CNN.

Após a obtenção dos resultados, foram selecionadas as 6 bases com maior média de acurácia para aplicar o Principal Component Analysis com 10 componentes, trocando as 6 com menor média de acurácia pelas 6 geradas pelo PCA.

4.3. Metodologia DT

Nesta análise, foi avaliado o impacto da profundidade máxima da árvore de decisão (*max_depth*), para isso foi utilizado um loop para aplicar o DT para diferentes níveis de profundidade, além do uso dos parâmetros default do modelo, de maneira semelhante ao KNN.

Os resultados foram apresentados com a média e o desvio padrão da acurácia para cada valor de *max_depth*, comparando as abordagens de divisão treino/teste e validação cruzada.

4.4. Metodologia NB

4.4.1. Gaussian Naive Bayes

Para avaliar o desempenho do modelo Gaussian Naive Bayes, foram implementadas duas funções principais: uma para normalizar o dataset e outra para treinar e validar o modelo.

A função padronizar_dados foi utilizada para padronizar as variáveis preditoras do dataset, garantindo que todas as features tivessem média 0 e desvio padrão 1. Isso foi realizado com a biblioteca StandardScaler do scikit-learn, essencial para modelos que assumem dados normalmente distribuídos.

Para a aplicação do modelo, foram utilizados os hiperparâmetros default e a GaussianNB, importada da biblioteca Sklearn.

4.4.2. Multinomial Naive Bayes

Para o modelo Multinomial Naive Bayes foi criada uma função responsável por discretizar os dados utilizados no modelo, para tal, houve o uso da KBinsDiscretizer. Além disso, foi utilizado a MultinomialNB da biblioteca Sklearn para utilizar a versão Multinomial do NB.

4.4.3. Complement Naive Bayes

O modelo Complement Naive Bayes , de maneira semelhante a versão Multinomial, necessitou da criação de uma função discretizadora para melhor eficácia do modelo. Para a utilização do da versão Complement do NB foi importado o Complement Naive Bayes para o uso do ComplementNB.

Essa abordagem demonstrou a eficácia do Complement Naive Bayes em dados discretizados, especialmente para problemas com classes desbalanceadas.

4.5. Metodologia MLP

Os experimentos realizados com redes neurais do tipo Multilayer Perceptron (MLP) buscaram avaliar diferentes configurações de hiperparâmetros para otimizar o desempenho do modelo. O objetivo foi identificar a melhor arquitetura capaz de generalizar o problema e alcançar altas taxas de acurácia. Foram explorados os seguintes aspectos:

Arquiteturas de Redes (Hidden Layers):

Testamos diferentes arquiteturas de camadas ocultas, variando a profundidade e o número de neurônios. O tamanho das camadas variou de uma única camada com 300 neurônios até cinco camadas com configurações como (300, 200, 150, 100, 50).

Solver e Otimização:

O otimizador Adam foi utilizado em todas as configurações por sua robustez e capacidade de ajustar a taxa de aprendizado dinamicamente.

Taxa de Aprendizado (Learning Rate):

Foram testados valores de 0.001, 0.0001, e 5e-05, visando observar o impacto em estabilidade e convergência do modelo.

Número Máximo de Iterações (Max Iter):

Para garantir a convergência, variamos o número de iterações máximas entre 1000, 1500, e 2000.

Validação Cruzada e Divisão de Dados:

A avaliação foi conduzida utilizando duas abordagens:

Split 70/30:

Divisão de 70% para treinamento e 30% para teste, repetida 10 vezes.

Validação Cruzada 10-Fold:

O modelo foi avaliado em 10 folds para garantir maior confiabilidade nos resultados.

Configuração 1: Hidden Layers: (300, 200, 150, 100, 50), Solver: adam, Learning Rate: 0.001, Max Iter: 1000

Configuração 2: Hidden Layers: (300, 200, 150, 100, 50), Solver: adam, Learning Rate: 0.001, Max Iter: 2000

Configuração 3: Hidden Layers: (300, 200), Solver: adam, Learning Rate: 0.001, Max Iter: 1000

Configuração 4: Hidden Layers: (300, 200), Solver: adam, Learning Rate: 0.001, Max Iter: 1500

Configuração 5: Hidden Layers: (300, 200, 100), Solver: adam, Learning Rate: 0.0001, Max Iter: 2000

Configuração 6: Hidden Layers: (300, 200, 100), Solver: adam, Learning Rate: 0.0001, Max Iter: 1500

Configuração 7: Hidden Layers: (300, 200, 100), Solver: adam, Learning Rate: 5e-05, Max Iter: 1500

Configuração 8: Hidden Layers: (300,), Solver: adam, Learning Rate: 0.0001, Max Iter: 2000

Configuração 9: Hidden Layers: (300,), Solver: adam, Learning Rate: 0.001, Max Iter: 1000

Configuração 10: Hidden Layers: (300,), Solver: adam, Learning Rate: 5e-05, Max Iter: 2000

4.5.11. Resumo Geral

As configurações variam em profundidade, taxa de aprendizado e número de iterações, equilibrando complexidade e estabilidade. As arquiteturas mais profundas tendem a capturar mais padrões, mas exigem taxas de aprendizado menores e mais iterações para convergir. Configurações simples, como (300,), são mais rápidas, mas podem perder detalhes importantes.

4.6. Metodologia Comitês

4.6.1. Bagging (Bootstrap Aggregating)

O Bagging usa múltiplos classificadores treinados em subconjuntos aleatórios dos dados. Utilizamos BaggingClassifier com DecisionTreeClassifier e variamos o número de estimadores (10, 20 e 30). Para a segunda tabela, aplicamos feature selection com max_features=0.5 para observar seu impacto na acurácia, que pode reduzir o overfitting mas também limitar a capacidade do modelo.

4.6.2. Boosting (AdaBoost)

O Boosting treina modelos sequencialmente, corrigindo os erros dos modelos anteriores. Usamos o AdaBoostClassifier com DecisionTreeClassifier e variamos o número de estimadores (10, 20 e 30). O Boosting tende a melhorar a acurácia ao focar nos exemplos difíceis, mas pode ser mais suscetível ao overfitting em dados ruidosos.

4.6.3 Random Forest

O Random Forest treina múltiplas árvores de decisão com amostras aleatórias e utiliza critérios de divisão como gini, entropy e log_loss. Variamos o número de estimadores (10, 20, 30 e 100). Ele é robusto e resistente ao overfitting, e tende a ter um bom desempenho com dados complexos.

4.6.4 Stacking

O Stacking combina múltiplos classificadores cujas previsões alimentam um modelo final. Usamos MLP, k-NN, DT e NB como modelos base e variamos o número de classificadores (5, 10, 15 e 20). O Stacking é eficaz ao combinar diferentes tipos de modelos, podendo melhorar a acurácia ao capturar diferentes aspectos dos dados.

4.6.5 Voting

O Voting Classifier combina os resultados de múltiplos modelos base. Usamos MLP, k-NN, DT e NB, variando o número de classificadores (5, 10, 15 e 20). O Voting pode melhorar a robustez do modelo, pois considera a votação de diferentes classificadores.

5. Resultados Experimentais

5.1. Resultados Experimentais KNN

Bases	Treino/Teste	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
HOG_128_16x16	**70/30**	**0.4166**	**0.3784**	**0.4062**	**0.3813**	**0.4029**	**0.3768**	**0.3992**	**0.3826**	**0.3913**	**0.3934**
	10-fold CV	**0.4258**	**0.3820**	**0.4070**	**0.3845**	**0.3958**	**0.3946**	**0.3921**	**0.3883**	**0.4058**	**0.3933**
HOG_128_20x20	70/30	0.4224	0.3909	0.4249	0.4195	0.4237	0.4220	0.4324	0.4398	0.4340	0.4274
	10-fold CV	0.4370	0.4145	0.4432	0.4444	0.4320	0.4470	0.4344	0.4420	0.4445	0.4457
HOG_256_16x16	**70/30**	**0.3627**	**0.3241**	**0.3241**	**0.3270**	**0.3390**	**0.3344**	**0.3133**	**0.3062**	**0.3295**	**0.3129**
	10-fold CV	**0.3546**	**0.3358**	**0.3371**	**0.3546**	**0.3496**	**0.3396**	**0.3284**	**0.3246**	**0.3334**	**0.3271**
CNN_VGG16_256_avg	70/30	0.4357	0.4307	0.4407	0.4373	0.4639	0.4544	0.4365	0.4697	0.4622	0.4664
	10-fold CV	0.4295	0.4296	0.4408	0.4495	0.4746	0.4708	0.4733	0.4795	0.4945	0.4907
CNN_VGG19_128_avg	**70/30**	**0.3589**	**0.3506**	**0.3884**	**0.3780**	**0.3830**	**0.3801**	**0.3747**	**0.3938**	**0.3988**	**0.3900**
	10-fold CV	**0.3758**	**0.3621**	**0.3821**	**0.3770**	**0.3946**	**0.4058**	**0.3983**	**0.3908**	**0.3970**	**0.4044**
CNN_VGG19_256_avg	70/30	0.4452	0.4162	0.4643	0.4668	0.4763	0.4751	0.4755	0.4697	0.4635	0.4772
	10-fold CV	0.4544	0.4157	0.4619	0.4719	0.4807	0.4894	0.5056	0.4919	0.5144	0.4919
PCA_HOG_128_16x16	**70/30**	**0.4075**	**0.3992**	**0.4224**	**0.4299**	**0.4432**	**0.4373**	**0.4598**	**0.4373**	**0.4469**	**0.4278**
	10-fold CV	**0.4019**	**0.4057**	**0.4181**	**0.4306**	**0.4581**	**0.4357**	**0.4519**	**0.4531**	**0.4481**	**0.4506**
PCA_HOG_128_20x20	70/30	0.4365	0.4207	0.4328	0.4461	0.4560	0.4631	0.4539	0.4751	0.4909	0.4867
	10-fold CV	0.4344	0.4083	0.4295	0.4470	0.4695	0.4694	0.4744	0.4732	0.4794	0.5032
PCA_HOG_256_16x16	**70/30**	**0.4075**	**0.3909**	**0.3925**	**0.4245**	**0.4245**	**0.4149**	**0.4349**	**0.4245**	**0.4490**	**0.4477**
	10-fold CV	**0.4131**	**0.3944**	**0.4031**	**0.4106**	**0.4206**	**0.4206**	**0.4156**	**0.4257**	**0.4381**	**0.4505**
PCA_CNN_VGG16_256_avg	70/30	0.4158	0.3851	0.4170	0.4324	0.4373	0.4299	0.4456	0.4407	0.4656	0.4602
	10-fold CV	0.3982	0.3896	0.4306	0.4381	0.4269	0.4469	0.4569	0.4545	0.4632	0.4557
PCA_CNN_VGG19_128_avg	**70/30**	**0.3307**	**0.3398**	**0.3506**	**0.3568**	**0.3697**	**0.3685**	**0.3776**	**0.3838**	**0.3971**	**0.4025**
	10-fold CV	**0.3470**	**0.3420**	**0.3533**	**0.3770**	**0.3683**	**0.3720**	**0.3908**	**0.3920**	**0.3920**	**0.3995**
PCA_CNN_VGG19_256_avg	70/30	0.4079	0.3971	0.4187	0.4120	0.4274	0.4149	0.4212	0.4378	0.4216	0.4307
	10-fold CV	0.4194	0.3795	0.4194	0.4294	0.4494	0.4193	0.4331	0.4219	0.4120	0.4232

Os hiperparâmetros escolhidos foram os default, pois ao realizar diferentes testes, foi notado que não havia diferenças relevantes para necessitar alguma mudança no KNN. Suas baixas acurácias são justificadas pela dimensionalidade dos dados, pois, mesmo utilizando do HOG e CNN para extrair as características das imagens, a dimensionalidade se mantém alta e o KNN não lida bem com isso.

5.2. Resultados Experimentais DT

Bases	Treino/Teste	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
DT_HOG_128_16x16	70/30	0.2743	0.3270	0.3365	0.3527	0.3346	0.3411	0.3199	0.3237	0.3270	0.3336
	10-fold CV	0.3021	0.3159	0.3508	0.3346	0.3221	0.3308	0.3421	0.3096	0.3283	0.3371
DT_HOG_128_20x20	70/30	0.2963	0.3332	0.3278	0.3402	0.3485	0.3606	0.3382	0.3358	0.3282	0.3427
	10-fold CV	0.3008	0.3557	0.3408	0.3408	0.3409	0.3246	0.3358	0.3146	0.3246	0.3246
DT_HOG_256_16x16	70/30	0.2544	0.2884	0.3183	0.3104	0.3237	0.3183	0.3054	0.2959	0.3091	0.3162
	10-fold CV	0.2747	0.2996	0.3533	0.3671	0.3558	0.3433	0.3433	0.3234	0.3359	0.3283
DT_CNN_VGG16_256_avg	70/30	0.3320	0.3585	0.3647	0.3793	0.3842	0.3639	0.3710	0.3610	0.3622	0.3502
	10-fold CV	0.3496	0.4045	0.4208	0.4057	0.4032	0.3732	0.3707	0.3746	0.3795	0.3658
DT_CNN_VGG19_128_avg	70/30	0.3207	0.3784	0.3975	0.3759	0.3809	0.3830	0.3867	0.3631	0.3651	0.3415
	10-fold CV	0.3283	0.3645	0.4082	0.4144	0.3845	0.3683	0.3383	0.3582	0.3345	0.3420
DT_CNN_VGG19_256_avg	70/30	0.4137	0.4237	0.4477	0.4237	0.4261	0.4344	0.4207	0.4332	0.4037	0.4232
	10-fold CV	0.4157	0.4482	0.4557	0.4407	0.4294	0.4357	0.4332	0.4132	0.4257	0.4145
DT_PCA_HOG_128_16x16	70/30	0.3050	0.3710	0.3672	0.3598	0.3859	0.3739	0.3851	0.3697	0.3701	0.3589
	10-fold CV	0.3233	0.3746	0.3558	0.3721	0.3982	0.3920	0.3895	0.3845	0.3895	0.3757
DT_PCA_HOG_128_20x20	70/30	0.2934	0.3834	0.3602	0.3838	0.4012	0.3834	0.3888	0.3846	0.3772	0.3909
	10-fold CV	0.3334	0.3858	0.3833	0.3971	0.3983	0.4120	0.4144	0.4220	0.4082	0.4170
DT_PCA_HOG_256_16x16	70/30	0.3174	0.3618	0.3780	0.3647	0.3614	0.3639	0.3589	0.3780	0.3402	0.3427
	10-fold CV	0.3533	0.3870	0.3871	0.3771	0.3807	0.3745	0.3507	0.3620	0.3557	0.3546
DT_PCA_CNN_VGG16_256_avg	70/30	0.3448	0.3867	0.4154	0.4203	0.4286	0.3917	0.4062	0.3909	0.3871	0.3751
	10-fold CV	0.3633	0.4146	0.4183	0.4221	0.4046	0.4108	0.4208	0.3883	0.3833	0.3871
DT_PCA_CNN_VGG19_128_avg	70/30	0.2888	0.2929	0.3149	0.3531	0.3423	0.3137	0.3415	0.3257	0.3324	0.3324
	10-fold CV	0.3333	0.3121	0.3421	0.3520	0.3457	0.3508	0.3532	0.3482	0.3445	0.3545
DT_PCA_CNN_VGG19_256_avg	70/30	0.3066	0.3365	0.3382	0.3813	0.3726	0.3606	0.3739	0.3747	0.3664	0.3519
	10-fold CV	0.3296	0.3470	0.3832	0.3869	0.3970	0.4033	0.3908	0.3658	0.4020	0.3958

As baixas acurácias do método Decision Tree se justificam com a dificuldade que o modelo possui de lidar com distribuições mais complexas e com maiores dimensionalidades.

Os hiperparâmetros escolhidos para esse modelo foram os default, variando somente a quantidade de níveis de profundidade.

5.3. Resultados Experimentais NB

Bases	Treino/Teste	Gaussian	Multinomial	Complement
NB_HOG_128_16x16	**70/30**	**0.5199**	**0.5270**	**0.5145**
	10-fold CV	**0.5554**	**0.5660**	**0.5448**
NB_HOG_128_20x20	70/30	0.5324	0.4813	0.4938
	10-fold CV	0.5468	0.5499	0.5411
NB_HOG_256_16x16	**70/30**	**0.4917**	**0.4398**	**0.4564**
	10-fold CV	**0.5193**	**0.4927**	**0.5027**
NB_CNN_VGG16_256_avg	70/30	0.3564	0.4523	0.4315
	10-fold CV	0.3645	0.4428	0.4453
NB_CNN_VGG19_128_avg	**70/30**	**0.3826**	**0.4689**	**0.4523**
	10-fold CV	**0.3894**	**0.4252**	**0.4153**
NB_CNN_VGG19_256_avg	70/30	0.3452	0.4689	0.4689
	10-fold CV	0.3508	0.4501	0.4314
NB_PCA_HOG_128_16x16	**70/30**	**0.4651**	**0.4398**	**0.4232**
	10-fold CV	**0.4906**	**0.4737**	**0.4638**
NB_PCA_HOG_128_20x20	70/30	0.4954	0.4274	0.4315
	10-fold CV	0.4931	0.4588	0.4575
NB_PCA_HOG_256_16x16	**70/30**	**0.4722**	**0.3942**	**0.4025**
	10-fold CV	**0.4644**	**0.4228**	**0.4215**
NB_PCA_CNN_VGG16_256_avg	70/30	0.3797	0.4357	0.4357
	10-fold CV	0.3869	0.4488	0.4414
NB_PCA_CNN_VGG19_128_avg	**70/30**	**0.3838**	**0.4315**	**0.4274**
	10-fold CV	**0.4044**	**0.3754**	**0.3779**
NB_PCA_CNN_VGG19_256_avg	70/30	0.3660	0.3900	0.4108
	10-fold CV	0.3533	0.4153	0.4128

O modelo de gaussian do NB possui acurácias mais altas que os demais modelos devido a alta dimensionalidade das bases usadas, o modelo gaussiano lida melhor com dados com muitas características.

De maneira semelhante, os modelos Multinomial e Complement são bem robustos para bases com diversas características, o que os diferem dos modelos anteriores.

Os hiperparâmetros utilizados nos modelos de NB foram os default, no caso do gaussian, é o melhor para dados contínuos, para o Complement, é melhor para dados mais desbalanceados e o Multinomial apesar de ser recomendado para textos, se os dados foram discretizados e houver classes bem definidas, ele funciona bem.

5.4. Resultados Experimentais MLP

Bases	Treino/Teste	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6	Config 7	Config 8	Config 9	Config 10
hog_128_16_augmented	70/30	0.6870	0.6798	0.7119	0.6881	0.6352	0.6715	0.6777	0.6756	0.6902	0.6601
	10-fold CV	0.5899	0.5700	0.5725	0.5706	0.5672	0.5737	0.5725	0.5610	0.5818	0.5622
hog_128_20_augmented	70/30	0.7047	0.6891	0.6663	0.6736	0.6560	0.6767	0.6767	0.6632	0.6580	0.6601
	10-fold CV	0.5846	0.5908	0.5659	0.5650	0.5609	0.5796	0.5796	0.5557	0.5715	0.5696
HOG_256_16_augmented	70/30	0.7270	0.7191	0.7356	0.6936	0.6680	0.6873	0.6863	0.6814	0.6981	0.6904
	10-fold CV	0.5876	0.6003	0.5988	0.5714	0.5809	0.5782	0.5788	0.5814	0.5915	0.5993
vgg16_128_augmented	70/30	0.9627	0.9627	0.9679	0.9679	0.9627	0.9627	0.9482	0.9648	0.9731	0.9627
	10-fold CV	0.9154	0.9154	0.9117	0.9117	0.9111	0.9111	0.9089	0.9148	0.9154	0.9095
vgg19_128_augmented	70/30	0.9689	0.9689	0.9658	0.9658	0.9596	0.9596	0.9554	0.9617	0.9710	0.9585
	10-fold CV	0.8983	0.8983	0.9064	0.9064	0.9011	0.9011	0.8993	0.9033	0.9049	0.8930
vgg19_256_augmented	70/30	0.9969	0.9969	0.9948	0.9948	0.9948	0.9948	0.9907	0.9979	0.9990	0.9886
	10-fold CV	0.9804	0.9804	0.9832	0.9832	0.9764	0.9764	0.9758	0.9804	0.9829	0.9754
pca10_hog_128_16_augmented	70/30	0.5378	0.5378	0.5378	0.5482	0.5264	0.5078	0.5192	0.5005	0.4839	0.4798
	10-fold CV	0.4885	0.4885	0.4885	0.5015	0.4997	0.5074	0.4944	0.4686	0.4879	0.4502
pca10_hog_128_20_augmented	70/30	0.5202	0.5202	0.5202	0.5886	0.5098	0.4974	0.4902	0.4756	0.4725	0.4528
	10-fold CV	0.4726	0.4726	0.4726	0.4739	0.4870	0.4813	0.4863	0.4782	0.4860	0.4393
pca10_hog_256_16_augmented	70/30	0.5472	0.5472	0.5472	0.4943	0.5098	0.4819	0.4881	0.4829	0.4756	0.4363
	10-fold CV	0.4677	0.4677	0.4677	0.4484	0.4720	0.4670	0.4658	0.4450	0.4565	0.4453
pca10_vgg16_128_augmented	70/30	0.9098	0.9098	0.9098	0.9026	0.9036	0.8891	0.9016	0.8674	0.8829	0.8705
	10-fold CV	0.8744	0.8744	0.8744	0.8788	0.8784	0.8778	0.8706	0.8629	0.8682	0.8489
pca10_vgg19_128_augmented	70/30	0.3595	0.3595	0.3595	0.4256	0.4752	0.4504	0.3884	0.2851	0.3264	0.2893
	10-fold CV	0.4329	0.4329	0.4329	0.4291	0.4441	0.4255	0.4005	0.3784	0.3956	0.3334
pca10_vgg19_256_augmented	70/30	0.9855	0.9855	0.9855	0.9865	0.9865	0.9845	0.9855	0.9865	0.9865	0.9855
	10-fold CV	0.9739	0.9739	0.9739	0.9786	0.9764	0.9754	0.9742	0.9778	0.9753	0.9782

Melhor Configuração: A Configuração 1 (cinco camadas ocultas) foi a mais eficiente, alcançando 99.89% (70/30) e 98.04% (10-fold CV) nos datasets de VGG, mostrando-se robusta para problemas complexos.

Impacto do PCA: O PCA reduziu a acurácia em todos os casos, indicando que eliminou informações relevantes, especialmente para datasets de HOG.

Taxa de Aprendizado: A taxa 0.001 mostrou-se mais estável e eficaz, enquanto taxas menores, como 5e-05, prejudicaram a convergência.

Bases de Dados: Features de VGG tiveram os melhores resultados, enquanto HOG apresentou menor expressividade.

A Configuração 1 (Hidden Layers: (300, 200, 150, 100, 50)) é a mais eficaz, equilibrando profundidade e convergência. Bases com features de VGG se mostram mais adequadas para o problema, enquanto o PCA, apesar de reduzir a dimensionalidade, comprometeu o desempenho, sugerindo que deve ser usado com cautela. Para futuras otimizações, explorar arquiteturas mais simples em bases com VGG pode ser uma alternativa eficiente.

5.5. Resultados Experimentais Comitês

5.5.1. Bagging

Bagging

Estratégia	10	20	30	Media(Acc TAM
AD	0.9585	0.9626	0.9678	0.9629
K-NN	0.9823	0.9834	0.9803	0.9820
NB	0.9678	0.9637	0.9678	0.9664
MLP	0.9855	0.9855	0.9844	0.9851
Media (Acc Classificadores)	0.9735	0.9738	0.9751	

Bagging max_features=0.5

Estratégia	10	20	30	Media(Acc TAM)
AD	0.8505	0.8868	0.9013	0.8795
K-NN	0.8941	0.9065	0.9200	0.9069
NB	0.9086	0.9221	0.9107	0.9138
MLP	0.9242	0.9252	0.9294	0.9263
Media (Acc Classificadores)	0.8944	0.9102	0.9154	

As altas acurácias do baggin se justificam por meio de como o baggin age, ele reduz a variância e torna os modelos utilizados mais robustos, além disso, seus hiperparâmetros default são mais eficientes, alternando somente o número de estimadores entre 10, 20 e 30.

5.5.2. Boosting

Boosting

Estratégia	10	20	30	Media(Acc TAM)
AD	0.9169	0.9211	0.9148	0.9176
NB	0.9626	0.9616	0.9647	0.9630
Media (Acc Classificadores)	0.9397	0.9414	0.9398	

5.5.3. Random Forest

Random forest

Estratégia	10	20	30	100	Media(Acc TAM)
Gini	0.9595	0.9688	0.9751	0.9782	0.9704
Entropy	0.9678	0.9792	0.9740	0.9782	0.9748
Log-loss	0.9709	0.9761	0.9792	0.9782	0.9761
Media (Acc Classificadores)	0.9661	0.9747	0.9761	0.9762	

5.5.4. Stacking

Stacking NB

Estratégia	Media
5 classificadores	0.9543
10 classificadores	0.9533
15 classificadores	0.9533
20 classificadores	0.9522

Stacking MLP

Estratégia	Media
5 classificadores	0.9834
10 classificadores	0.9844
15 classificadores	0.9834
20 classificadores	0.9834

Stacking K-NN

Estratégia	Media
5 classificadores	0.9792
10 classificadores	0.9792
15 classificadores	0.9792
20 classificadores	0.9792

Stacking DT

Estratégia	Media
5 classificadores	0.9221
10 classificadores	0.9221
15 classificadores	0.9221
20 classificadores	0.9221

5.5.5 Voting

Voting NB

Estratégia	Media
5 classificadores	0.9657
10 classificadores	0.9657
15 classificadores	0.9657
20 classificadores	0.9657

Voting MLP

Estratégia	Media
5 classificadores	0.9855
10 classificadores	0.9875
15 classificadores	0.9865
20 classificadores	0.9855

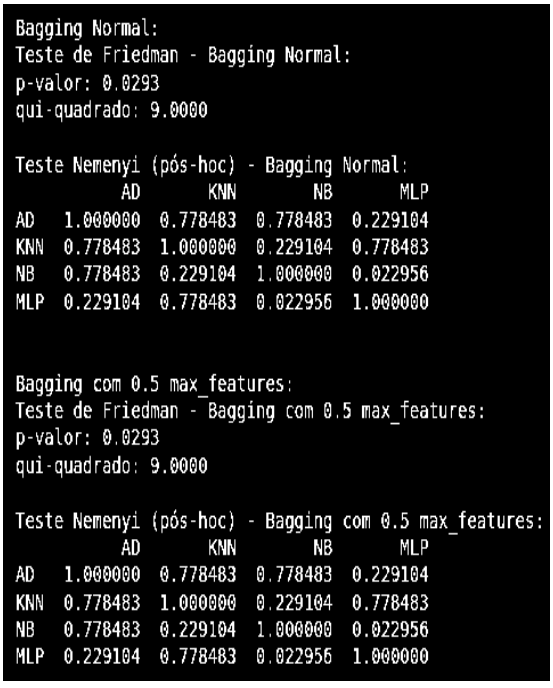
Voting K-NN

Estratégia	Media
5 classificadores	0.9855
10 classificadores	0.9823
15 classificadores	0.9823
20 classificadores	0.9813

DT

Estratégia	Media
5 classificadores	0.9169
10 classificadores	0.9200
15 classificadores	0.9180
20 classificadores	0.9200

5.6. Análise Estatísticas

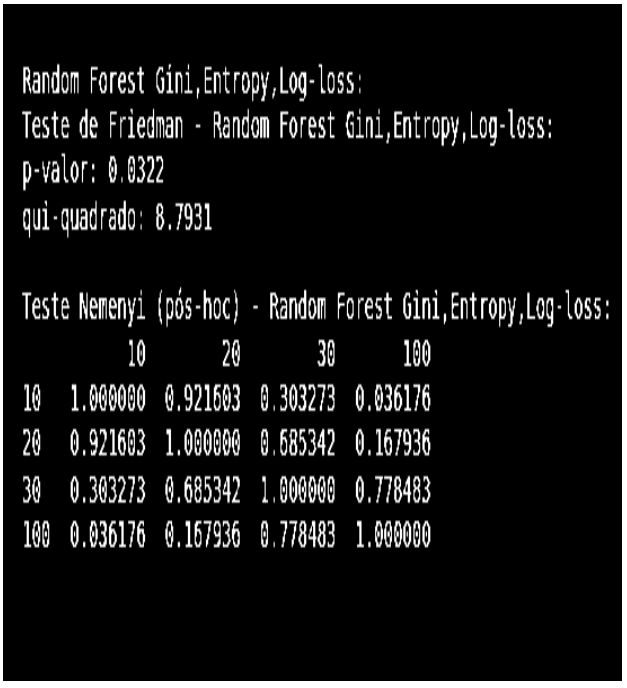


```
Bagging Normal:
Teste de Friedman - Bagging Normal:
p-valor: 0.0293
qui-quadrado: 9.0000

Teste Nemenyi (pós-hoc) - Bagging Normal:
          AD       KNN        NB       MLP
AD   1.000000  0.778483  0.778483  0.229104
KNN  0.778483  1.000000  0.229104  0.778483
NB   0.778483  0.229104  1.000000  0.022956
MLP  0.229104  0.778483  0.022956  1.000000


Bagging com 0.5 max_features:
Teste de Friedman - Bagging com 0.5 max_features:
p-valor: 0.0293
qui-quadrado: 9.0000

Teste Nemenyi (pós-hoc) - Bagging com 0.5 max_features:
          AD       KNN        NB       MLP
AD   1.000000  0.778483  0.778483  0.229104
KNN  0.778483  1.000000  0.229104  0.778483
NB   0.778483  0.229104  1.000000  0.022956
MLP  0.229104  0.778483  0.022956  1.000000
```



```
Random Forest Gini,Entropy,Log-loss:
Teste de Friedman - Random Forest Gini,Entropy,Log-loss:
p-valor: 0.0322
qui-quadrado: 8.7931

Teste Nemenyi (pós-hoc) - Random Forest Gini,Entropy,Log-loss:
          10        20        30        100
10   1.000000  0.921603  0.303273  0.036176
20   0.921603  1.000000  0.685342  0.167936
30   0.303273  0.685342  1.000000  0.778483
100  0.036176  0.167936  0.778483  1.000000
```

A partir desses resultados, é possível inferir duas análises:

Bagging Normal e Bagging com 0.5 max_features:

Resultados idênticos: Os testes de Friedman e Nemenyi para ambos os cenários apresentaram exatamente os mesmos resultados. Isso sugere que a redução do número máximo de features para 0.5 não teve um impacto significativo na performance dos algoritmos. O p-valor baixo nos testes de Friedman indica que pelo menos um dos algoritmos (AD, KNN, NB ou MLP) se destaca dos demais. O teste de Nemenyi detalha quais algoritmos são significativamente diferentes. Por exemplo, o MLP parece ser significativamente diferente dos outros algoritmos.

Random Forest Gini, Entropy, Log-loss:

Novamente, o p-valor baixo no teste de Friedman indica que a escolha do número de árvores (10, 20, 30 ou 100) influencia significativamente a performance do modelo. O teste de Nemenyi sugere que, em geral, modelos com mais árvores tendem a apresentar melhores resultados. No entanto, a diferença entre 30 e 100 árvores não parece ser tão significativa.

6. Dificuldades Encontradas

Houveram diversos problemas e dificuldades na preparação e transformação das imagens para as bases utilizadas no projeto, pois os modelos são sensíveis a qualidade das bases de dados, qualidade essa que está diretamente ligada aos transformadores. Logo, quaisquer erros ou configurações nos transformadores, depredaria a precisão dos modelos utilizados, principalmente no MLP. Além disso, a preparação das bases foi de extrema importância, pois cada modelo tem sua especificação e recomendação de uso, gerando dificuldades no momento de usar as bases.

7. Conclusão

Os resultados deste projeto demonstram o desempenho variável dos modelos de machine learning na tarefa de classificação de espécies e raças de animais a partir de imagens, previamente transformadas por meio de técnicas como CNN e HOG. Dentre os modelos avaliados — KNN, Decision Tree (DT), Naive Bayes (NB), Multi-Layer Perceptron (MLP) e comitês de modelos —, os melhores desempenhos foram alcançados pelo MLP e pelos comitês, que se destacaram pela sua eficácia.

A utilização do HOG (Histogram of Oriented Gradients) e da CNN (Convolutional Neural Networks) na transformação das imagens foi crucial para a extração de características relevantes, permitindo que os modelos realizassem classificações precisas. O MLP demonstrou grande capacidade de aprendizado, especialmente em padrões complexos, enquanto os comitês se beneficiaram da combinação de diferentes modelos, proporcionando maior estabilidade e acurácia nos resultados.

Esses resultados reforçam a relevância de estratégias robustas de pré-processamento e seleção de modelos no contexto de classificação de imagens, evidenciando a eficácia das abordagens adotadas neste projeto.

8. Referências

https://colab.research.google.com/drive/10iOFipp_r1Cc_SR0Tf_XXhvelmD_lPaP#scrollTo=S2dC9-6QIW1Z

https://docs.google.com/spreadsheets/d/1N4w8PSWY1yMkMpXPfFlbcR3yI8QsuH-1/edit?s=&gid=1885511017#gid=1885511017

https://docs.google.com/spreadsheets/d/1gG8li59Y59qS2QycsraMT8VNRE1iFuDzkkFKnZ0tZNM/edit?usp=sharing