Trabalho Prático de Classificação: The Oxford-IIIT Pet Dataset

Geovana Silva de Oliveira[1]

[1] Departamento de Computação e Sistemas (DECSI) – Universidade Federal de Ouro Preto
João Monlevade – MG, Brasil.

geovana.so@aluno.ufop.edu.br

1. Introdução

Este trabalho apresenta os experimentos realizados com técnicas clássicas de Aprendizado de Máquina no dataset Oxford-IIIT Pet. O foco foi utilizar a extração manual de características para verificar se essas técnicas, combinadas com balanceamento de dados e redução de dimensionalidade, são capazes de classificar corretamente as raças de cães e gatos.

2. Classificação Binária (Cães vs. Gatos)

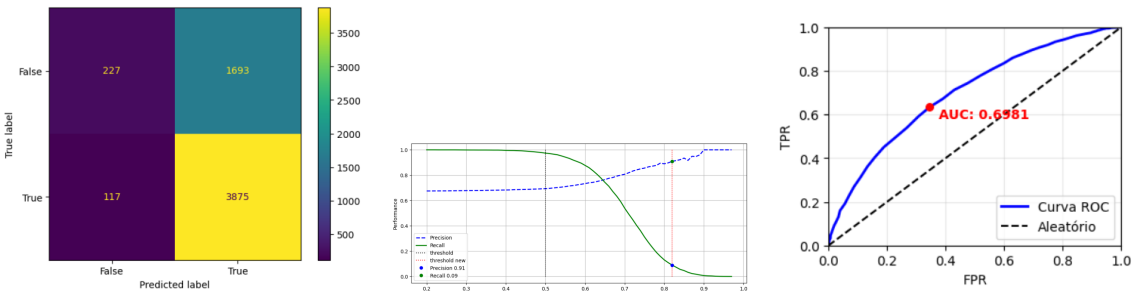
A primeira etapa consistiu na classificação binária. Essa tarefa serviu como ”teste de sanidade”para validar o pipeline. Se o modelo não for capaz de classificar entre cão e gato, não conseguirá classificar entre as 37 classes. A classe ”Cão”foi definida como positiva.

2.1. Carregamento e Pré-processamento

As imagens foram redimensionadas de 224x224 para 128x128 pixels para garantir a execução em tempo hábil, mas preservando as informações visuais essenciais.

2.1.1. Baseline: Random Forest (Pixels Brutos)

Como linha de base (*baseline*), treinou-se um Random Forest utilizando os valores de pixels brutos. O modelo foi configurado com pesos de classe balanceados (*class_weight="balanced"*) para mitigar o desequilíbrio natural do *dataset*, mas sem estratégias ativas para tal.



(a) Matriz de confusão **(b) Curva PR** **(c) Curva ROC**

Figura 1. Baseline: Random Forest com pixels brutos.

Como mostra a Figura 1, o desempenho foi ruim. A matriz de confusão (a) mostra que o modelo errou quase todos os gatos (1.693 erros). A análise dos gráficos confirma a fraqueza do modelo:

- **Curva Precision-Recall (b):** A curva despenca rapidamente. Isso significa que, para tentar recuperar mais gatos (aumentar o Recall), a precisão do modelo cai drasticamente. Ele não consegue manter a confiança nas predições.
- **Curva ROC (c):** A área sob a curva (AUC) de 0.69 indica que o classificador é pouco melhor que um chute aleatório (0.5), falhando em separar bem as classes.

2.1.2. Extração de Características de Forma (HOG)

Para tentat capturar a forma dos animais, foi utilizado o descritor Histograma de Gradientes Orientados (HOG), da biblioteca *scikit-image*. Esse método descreve a imagem com base nas direções das bordas e dos contornos, destacando a silhueta geral do animal. Dessa forma, o HOG permite capturar informações estruturais importantes, como o formato do corpo e da cabeça, sendo pouco sensível a variações de iluminação. A Figura 2 apresenta exemplos das imagens após a aplicação do descritor HOG.

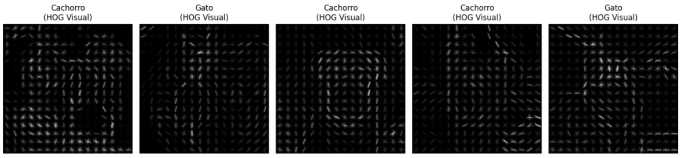
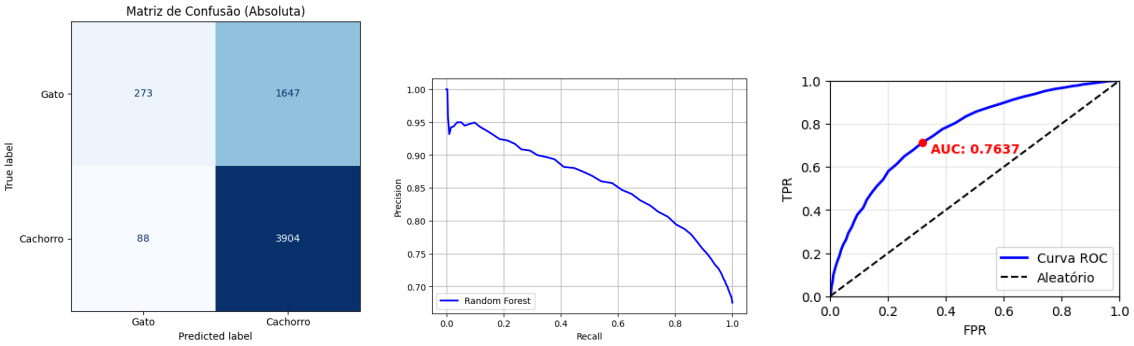


Figura 2. Exemplo da aplicação do HOG.



(a) Matriz de confusão **(b) Curva PR** **(c) Curva ROC**

Figura 3. Random Forest + HOG.

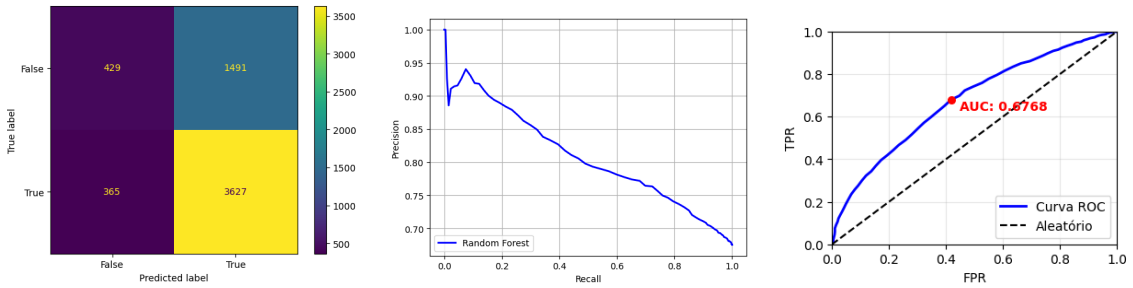
Com o HOG, houve uma melhora perceptível nos gráficos:

- **Curva PR (b):** Diferente do baseline, esta curva consegue se manter mais alta (acima de 0.8 de precisão) por mais tempo antes de cair. Isso mostra que as bordas (HOG) são características mais confiáveis que os pixels.
- **Curva ROC (c):** A AUC subiu para 0.76, indicando uma capacidade de separação global superior.

Apesar disso, a matriz (a) ainda mostra muitos erros na classe Gato devido ao desbalanceamento.

2.1.3. Redução de Dimensionalidade (LLE)

Buscando compactar a representação e remover redundâncias, aplicou-se o *Locally Linear Embedding* (LLE), uma técnica de aprendizado de variedades não-linear.



(a) Matriz de confusão **(b) Curva PR** **(c) Curva ROC**

Figura 4. Random Forest + HOG + LLE.

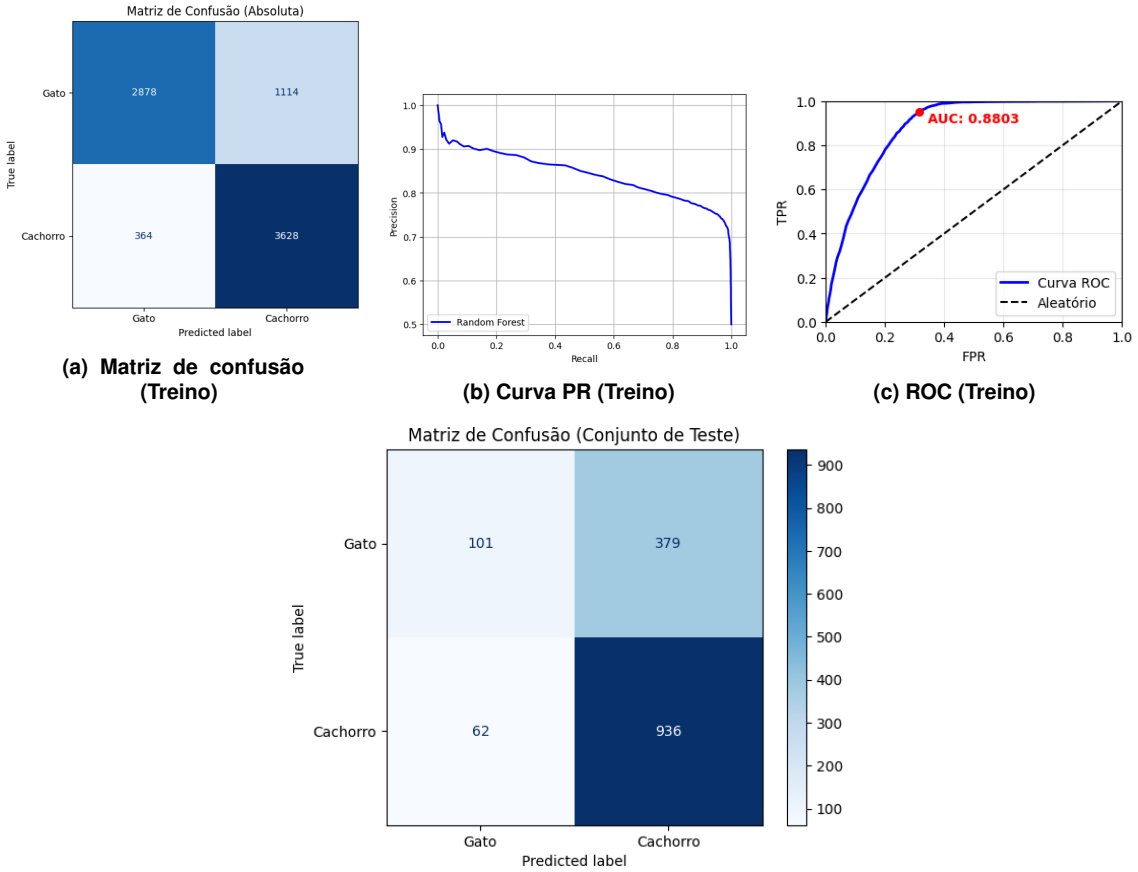
O LLE prejudicou a estabilidade do modelo, o que fica claro nos gráficos:

- **Curva PR (b):** A curva apresenta um decaimento quase linear, muito pior que a do HOG puro. Isso sinaliza que o modelo perdeu a capacidade de ordenar corretamente as predições mais confiáveis.
- **Curva ROC (c):** A AUC caiu para 0.67, inferior até mesmo ao baseline de pixels em alguns pontos. A redução de dimensionalidade parece ter misturado as classes no espaço latente.

Os resultados apresentados na Figura 4 indicam que o LLE prejudicou a distinção entre as classes. Embora tenha aumentado a sensibilidade para a classe de gatos, a precisão geral do modelo caiu . Isso sugere que, ao tentar preservar relações locais entre as amostras no espaço, o LLE acabou dificultando a diferenciação entre as espécies.

2.1.4. Estratégia Final: Upsampling + HOG + LLE

Para corrigir o viés de classe, foi aplicado o *Stratified Upsampling* no conjunto de treinamento. Nesse processo, cada raça de gato foi selecionada na mesma proporção durante o aumento das amostras, evitando que o modelo aprendesse padrões de raças específicas, como poderia ocorrer em uma seleção aleatória. Além disso, todas as imagens aumentadas foram rotacionadas aleatoriamente entre -20 e 20 graus.



(a) Matriz de confusão (Treino)

(b) Curva PR (Treino)

(c) ROC (Treino)

(d) Matriz (Teste)

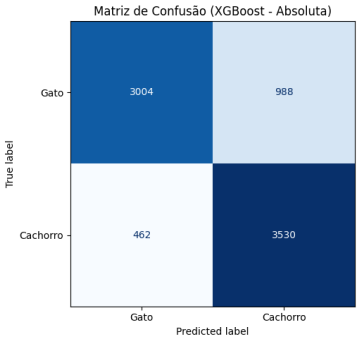
Figura 5. Random Forest com Upsampling, HOG e LLE.

Aqui observamos um fenômeno de *overfitting* clássico ao comparar os gráficos de treino com a matriz de teste:

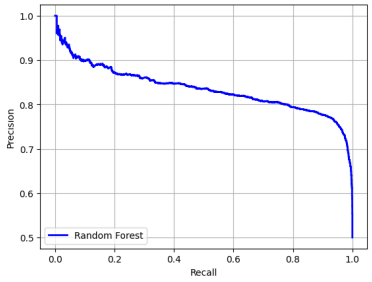
- **No Treino:** A Curva PR (b) ficou excelente, mantendo precisão alta em quase toda a faixa de recall. A ROC (c) atingiu 0.88, o melhor resultado até agora.
- **No Teste:** A matriz (d) revela a realidade. O Recall da classe Gato despencou para 21%. O modelo ”decorou”os dados repetidos do upsampling (o que explica os gráficos perfeitos no treino), mas falhou em generalizar para dados novos.

2.1.5. Comparativo: *XGBoost*

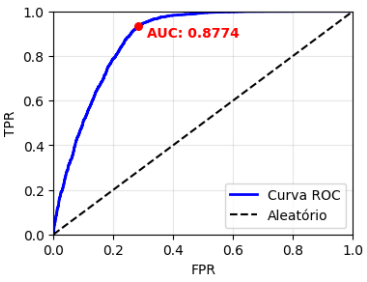
Para fins de comparação, testamos as mesmas configurações no modelo *XGBoost*. As configurações detalhadas de todos os modelos estão disponíveis nos *notebooks* no GitHub (5).



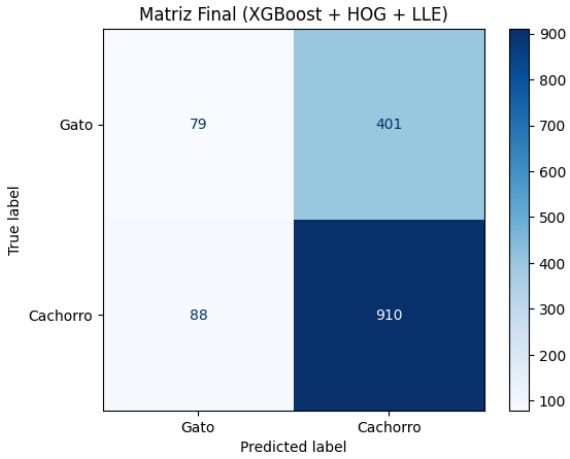
(a) Matriz de confusão(Treino)



(b) Curva PR (Treino)



(c) ROC (Treino)



(d) Matriz (Teste)

Figura 6. XGBoost com Upsampling, HOG e LLE.

O XGBoost apresentou a melhor Curva Precision-Recall de treino (b) de todo o estudo, com uma área bem preenchida no canto superior direito. Isso mostra que o algoritmo conseguiu aprender os exemplos difíceis do treino. Porém, a matriz de teste (d) confirmou que ele sofreu ainda mais com o *overfitting* (Recall de 16.4%), sendo ”enganado”pela repetição dos dados sintéticos.

3. Classificação Multiclasse (37 Raças)

A etapa multiclasse apresentou um desafio muito maior: distinguir entre 37 raças, muitas com características quase idênticas.

3.1. Desempenho com LLE (HOG + LLE)

Como a estratégia de HOG + LLE apresentou o melhor desempenho na classificação binária, decidiu-se mantê-la também para o problema multiclasse.

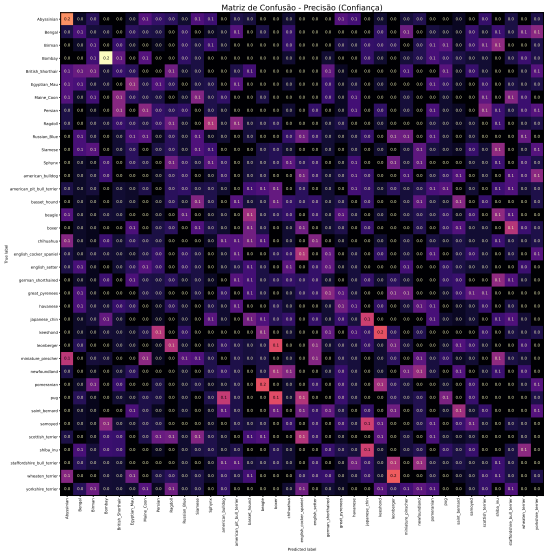


Figura 7. Precisão: RF + HOG + LLE

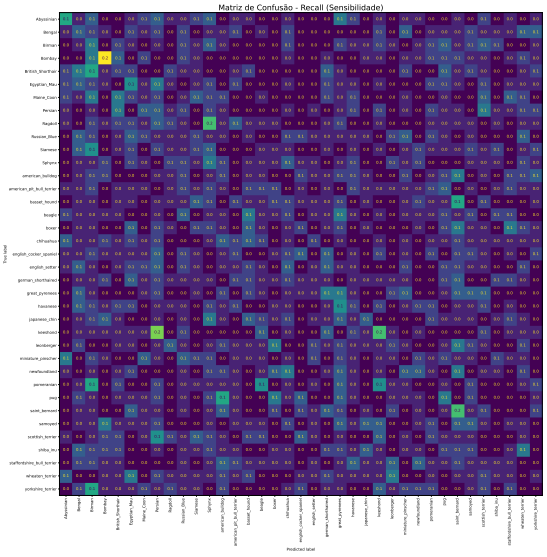


Figura 8. Recall: RF + HOG + LLE

O resultado foi catastrófico. As matrizes mostram uma confusão generalizada (sem diagonal definida). Isso acontece porque o LLE projeta os dados com base na vizinhança local. Em um problema com 37 classes muito próximas, a projeção para baixa dimensão acabou apagando as diferenças sutis necessárias para distinguir, por exemplo, um *Beagle* de um *Basset Hound*.

3.2. Melhora sem o LLE (Random Forest + HOG Bruto)

A hipótese de que o LLE estava descartando informação vital foi testada removendo-o e utilizando os vetores HOG brutos (alta dimensionalidade).

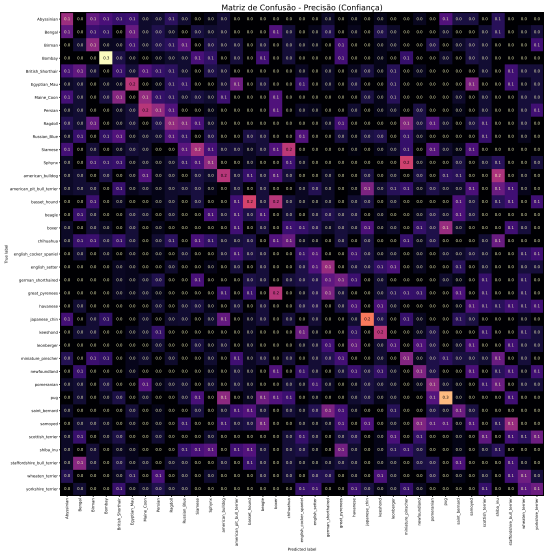


Figura 9. Precisão: RF + HOG

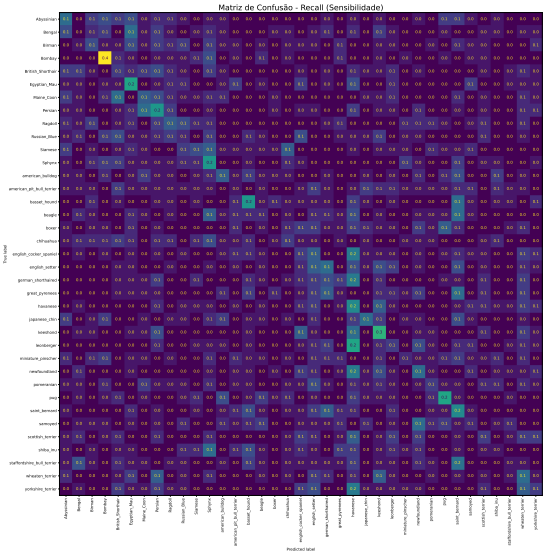


Figura 10. Recall: RF + HOG

A acurácia subiu para **12,18%**. As matrizes de confusão começaram a apre-

sentar uma diagonal visível (valores entre 0.1 e 0.3 de recall para algumas raças), provando que o HOG bruto contém informações importantes que o LLE estava descartando.

3.3. Uso do XGBoost (XGBoost + HOG)

Para tentar melhorar as fronteiras de decisão, aplicou-se o XGBoost aos vetores HOG originais.

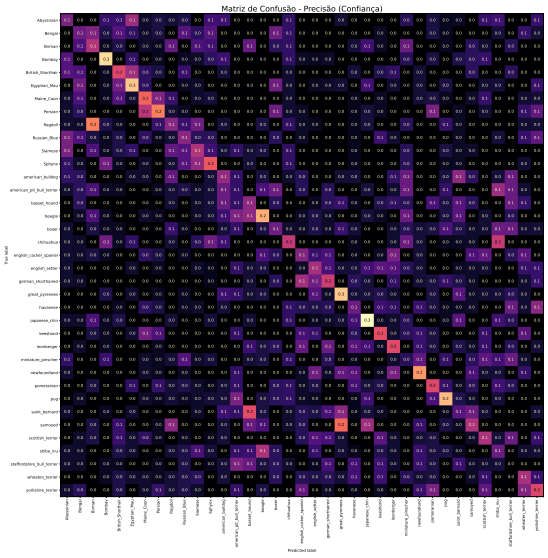


Figura 11. Precisão: XGBoost + HOG

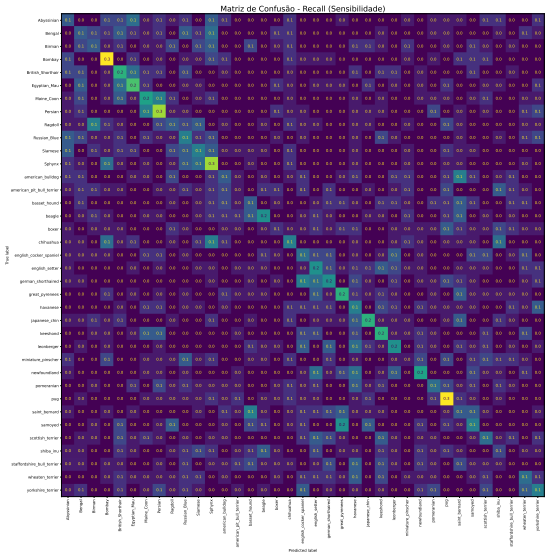


Figura 12. Recall: XGBoost + HOG

As matrizes mostram uma diagonal mais consistente que a do Random Forest. O XGBoost conseguiu extrair melhor as características de borda para diferenciar as raças e a acurácia global alcançou 15,76%. Esse aumento ocorre porque o *Boosting* reduz o viés: enquanto o Random Forest busca diminuir a variância, o XGBoost corrige os erros do modelo de forma sequencial, permitindo capturar melhor pequenas diferenças entre raças parecidas que o Random Forest considerava ruído.

3.4. Adição de Textura (HOG + LBP)

Na última tentativa, adicionou-se informação de textura usando o LBP (Local Binary Patterns), um descritor que captura padrões de textura local em uma imagem. O LBP compara cada pixel com seus vizinhos e gera um código binário que representa pequenas variações de intensidade, permitindo distinguir superfícies com texturas diferentes, como pelagens de raças semelhantes.

3.4.1. Random Forest + HOG + LBP

A acurácia permaneceu em 12,31%, com um ganho de apenas 0,13%. O Random Forest não conseguiu aproveitar as novas informações de textura, tratando o vetor LBP estendido basicamente como ruído.

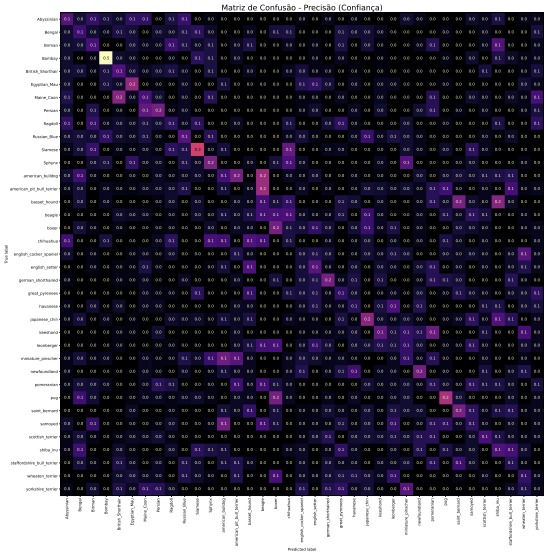


Figura 13. Precisão: RF + HOG + LBP

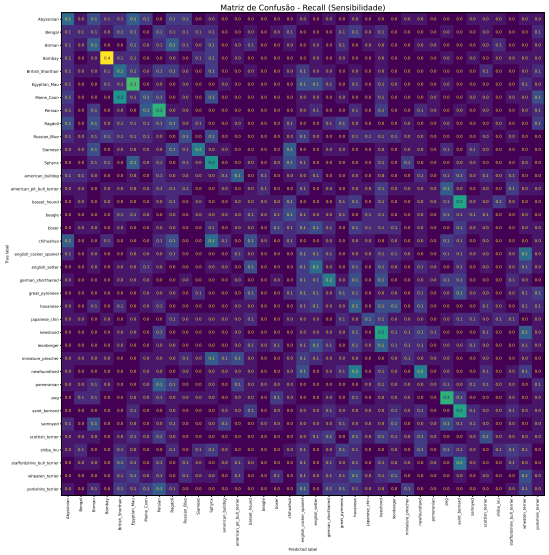


Figura 14. Recall: RF + HOG + LBP

3.4.2. XGBoost + HOG + LBP

Também foi testado a combinação HOG + LBP com o XGBoost. O melhor resultado global foi de 16,04%. Apesar de ser o maior valor obtido no estudo, o ganho em relação ao uso apenas do HOG (+0,28%) é bem pequeno. Uma hipótese é de que o LBP foi prejudicado pelo fundo das imagens (grama, sofá), já que não houve segmentação. O modelo pode ter confundido a textura do fundo com a do animal.

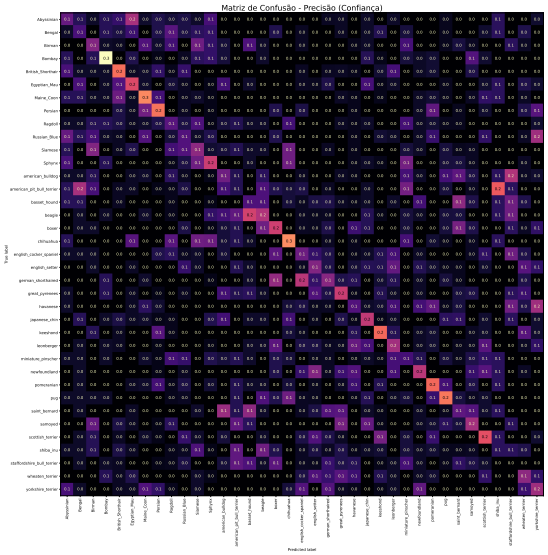


Figura 15. Precisão: XGBoost + HOG + LBP

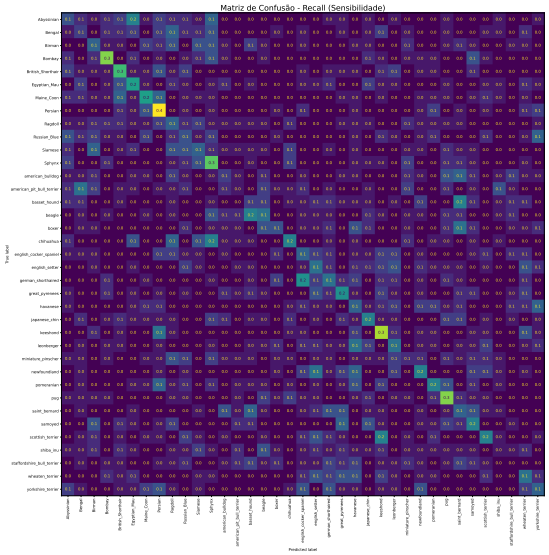


Figura 16. Recall: XGBoost + HOG + LBP

4. Conclusões

4.1. Sobre a Classificação Binária

Conclui-se que o uso de descritores manuais (HOG) funciona bem para separar Gato de Cachorro (AUC 0.88), pois as classes possuem formatos diferentes. O principal problema encontrado foi o *overfitting* causado pelo *upsampling*, que prejudicou o desempenho nos testes. A estratégia de *downsampling* na classe cachorro, porém, faria com que praticamente metade das instâncias fossem perdidas, mesmo que isso diminuísse o *overfitting*. Provavelmente o modelo não teria um bom desempenho nessa classe devido à perda de variância dos dados.

4.2. Sobre a Classificação Multiclasse

O melhor resultado obtido foi 16% de acurácia global.

- O LLE prejudicou o desempenho ao descartar informações.
- O HOG foi o descritor mais promissor, mas insuficiente isoladamente.
- O LBP não trouxe melhorias significativas devido ao ruído do fundo.

Superar esse limite exigiria testar novas estratégias, como o uso de bibliotecas de realce de contraste, correção de cor ou filtros de textura (por exemplo, Haralick). No entanto, todas essas abordagens têm desvantagens e podem levar à perda de informações importantes. Por exemplo, aplicar remoção de fundo poderia cortar partes essenciais do animal para sua identificação. Uma combinação cuidadosa dessas técnicas poderia trazer melhorias, mas a solução mais promissora seria o uso de Redes Neurais Convolucionais (CNNs), capazes de aprender representações automáticas e focar na atenção ao objeto de interesse.

4.3. Sobre o uso de OvR e OvO

Em relação ao uso de estratégias *One-versus-One* (OvO) ou *One-versus-Rest* (OvR):

1. **Complexidade Computacional (OvO):** A estratégia OvO exige o treinamento de $\frac{N(N-1)}{2}$ classificadores. Para $N = 37$, seriam necessários 666 modelos distintos. Considerando que treinar o XGBoost com vetores densos (HOG+LBP) já consome muitos recursos, multiplicar esse custo por 666 tornaria a abordagem inviável dentro do tempo disponível, sem garantia de aumento de acurácia que justificasse o esforço.
2. **Desbalanceamento Induzido (OvR):** A estratégia OvR treinaria 37 classificadores. Apesar de ser computacionalmente mais leve que OvO, cada classificador enfrentaria um desbalanceamento extremo (1 classe positiva contra 36 negativas). Modelos como Random Forest são sensíveis a esse desequilíbrio, e técnicas de balanceamento precisariam ser aplicadas individualmente 37 vezes, aumentando a complexidade do pipeline.
3. **Capacidade Nativa:**

Além disso, tanto *Random Forest* quanto *XGBoost* possuem implementações nativas eficientes para problemas multiclasse, com árvores que particionam o espaço em K regiões. Por isso, o uso de OvR ou OvO costuma ser redundante e menos eficiente.

Tabela 1. Resumo dos Resultados (Multiclasse)

Modelo	Features	Configuração	Acurácia	Observação
RF	HOG	Com LLE	6.02%	Perda excessiva de informação.
RF	HOG	Bruto	12.18%	Melhora significativa sem LLE.
RF	HOG + LBP	Bruto	12.31%	Textura não ajudou (ruído).
XGBoost	HOG	Bruto	15.76%	Boosting superou Bagging.
XGBoost	**HOG + LBP**	**Bruto**	**16.04%**	**Melhor resultado.**

5. Código

O código utilizado encontra-se disponível no repositório: `https://github.com/gseovana/classificacao-pet-dataset`