

# Comparação de indutores para modelos de predição de classificação de imagens

Lucas Mrowskovsy Paim\*

2019

## Resumo

Este trabalho tem por objetivo comparar os indutores apresentados durante a disciplina de Machine Learning do programa de especialização em Inteligência Artificial Aplicada da Pontifícia Universidade Católica do Paraná, para predição de classificação de imagens, utilizando principalmente a biblioteca scikit-learn e a linguagem Python.

Tem por objetivo também a comparação dos resultados utilizando técnicas de extração de características manuais, handcrafted, e CNN deep-learning.

Para a extração de características de imagens foram utilizados as bibliotecas: opencv-python, skimage e keras

A base de características extraídas via Deep Learning alcançou margens melhores com todos os indutores, que dentre eles se destacou o SVM, que detectou erro de classificação na base de dados, ou seja, detectou o erro humano e classificou a imagem corretamente.

**Palavras-chave:** Naive Bayes. Árvores de decisão. Redes Neurais. k-Vizinhos Mais Próximos. Máquina de Vetores de Suporte. SVM.

## Abstract

This paper aims to compare the inductors presented during the Machine Learning discipline of the Pontifical Catholic University of Paraná specialization program in Applied Artificial Intelligence, to predict image classification using mainly the scikit-learn library and the Python language.

It also aims to compare the results using handcrafted and CNN deep-learning feature extraction techniques.

---

\*Programa de Pós Graduação em Inteligência Artificial, Curitiba - PR, 80215-901; Especializando em Inteligência Artificial Aplicada na PUCPR; Tecnólogo em Sistemas para Internet pela Universidade Positivo; E-mail: lucasmpaim1@gmail.com

For the extraction of image characteristics we used the libraries: opencv-python, skimage and keras.

The base of features extracted via Deep Learning achieved better margins with all inductors, among which stood out SVM, which detected misclassification in the database, that is, detected human error and classified the image correctly.

**Keywords:** Naive Bayes. Decision Tree. Neural Network. k-Nearest Neighbors. Support Vector Machine. SVM.

## 1 Introdução

A utilização de classificadores de imagens é utilizada nas mais diversas áreas do conhecimento, que vão desde o auxílio de diagnóstico de câncer de pele [Soares \(2008\)](#), sistemas de segmentação e classificação de imagens via satélite [Erhal et al. \(1991\)](#), OCR's (Optical Character Recognition) para reconhecimento de textos e dígitos [Ribas et al. \(2012\)](#) etc.

Os mesmos indutores também podem ser utilizados para outras finalidades que não sejam classificação de imagem, como: detecção de emoções em processamento de textos [Dosciatti, Ferreira e Paraiso \(2013\)](#), regressão linear etc.

Segundo [Coppin \(2004\)](#), fornecer a capacidade de visão a sistemas computacionais, assim como a robôs e agentes computacionais é algo, sem dúvida, extremamente desejável.

As técnicas de extração de características de imagens handcrafted que foram utilizadas são:

- LBP: Local Binary Pattern: Segundo, [Rosebrock \(2015\)](#), esta é uma técnica de extração de texturas de uma imagem, em que cada pixel é comparado com seus vizinhos a partir de uma máscara e com isso é gerado uma sequência binária que toma o lugar daquele pixel.
- HOG: Histogram of Oriented Gradients: Segundo [Mallick \(2016\)](#) é uma técnica de extrai a distribuição (Histogramas) de direções de gradientes.
- Histograma de Cores RGB.

Visto o exposto o estudo desse trabalho se torna importante para a área de classificação de imagens, podendo ser aplicado em diversas áreas, tendo por objetivo criar modelos de predição para classificação de imagens comparando vários indutores e analisando seus resultados.

## 2 Método

A base utilizada para este trabalho, conta com apenas mil imagens, separadas em dez classes com cem imagens cada, sendo assim, é uma base completamente balanceada.

Todos os indutores, foram treinados utilizando a mesma base de imagens, em um primeiro momento ela é separada em 70% em treinamento e 30% em testes, técnica esta chamada de holdout (percentage-split), após esta separação ainda foi separado mais 30% da base de treinamento e criada uma base de validação, para evitar assim o chamado

overfitting, que segundo [Sarle \(2016\)](#), é um dos maiores problemas da aprendizagem de máquina, isto indica que o indutor está perdendo sua capacidade de generalização, isto é, segundo [Lima, Pinheiro e Santos \(2014\)](#) a capacidade do indutor responder adequadamente a dados que não fazem parte do conjunto de treinamento.

Devido ao não suporte dos métodos de overfitting para alguns indicadores, este trabalho utiliza, quando disponível, o método:

```
.partial_fit(X, y, [classes])
```

Este método segundo [learn \(2019\)](#), é ideal para treinamento incremental, sendo assim, é possível estender a lógica do early stopping para alguns indicadores que não suportam tal método nativamente, como o Naive Bayes.

Como o método incremental, tende a adicionar ruído na base de dados, foi utilizado o método de IIR (Infinite Impulse Response), que segundo [Grout \(2008\)](#) é um filtro recursivo em que a saída do filtro é calculada usando as entradas atuais e anteriores e as saídas anteriores, fazendo assim com que o ruído seja minimizado, este filtro foi aplicado no tratamento de erro tanto da base de validação quanto da base de teste.

Sua fórmula consiste em:

$$y_t = \sum_{i=0}^M b_i x(t-i) - \sum_{j=1}^L \alpha_j y(t-j) \quad (1)$$

O resultado da aplicação desta técnica pode ser consultada na [Figura 1](#), onde é possível observar que o gráfico foi suavizado, de forma que, quando ambas as bases forem confrontadas será possível detectar mais facilmente onde está ocorrendo o overfitting.

Sendo assim podemos definir a taxa de aprendizado como sendo:

$$r = y_{t-1} - y_t \quad (2)$$

Onde  $y_{t-1}$  se refere ao erro na base de validação na iteração anterior e  $y_t$  se refere a iteração atual de treinamento, desta forma quando obtemos um valor para  $r < 10^{-3}$  que é a taxa mínima de aprendizagem e isso ocorre por  $N$  vezes seguidas, significa que não há um ganho no treinamento e este pode ser interrompido.

## 2.1 Naive Bayes

Segundo, [Britto \(2019a\)](#), o teorema de Bayes consiste em uma abordagem probabilística para aprendizagem, calculando assim a probabilidade de uma característica  $X$  de um vetor de prever determinada classe.

A técnica de Naive Bayes, assume que todas as características no vetor são independentes, por isso, é chamado de Naive (do inglês Ingênuo), pois determinadas características podem estar relacionadas entre si, o que faz com que essa técnica tenha um resultado inferior a técnica de Redes Bayesianas que consideram que os atributos podem ou não ser relacionados, isto é, a característica  $X$  pode ou não estar diretamente relacionada a característica  $Y$ .

### 2.1.1 Resultados do Treinamento

A matriz de confusão para o esta técnica usando houldout e a base de características usando deeplearning pode ser consultada em [Figura 3](#), assim como sua taxa de aprendizagem [Figura 2](#)

Acurácia Naive Bayes Validação : 0.8928571428571429  
Acurácia Naive Bayes Teste : 0.9266666666666666

Para o método usando as características extraídas de forma hand crafting, a curva de erro na aprendizagem pode ser consultado na [Figura 4](#), assim como sua matriz de confusão na [Figura 5](#).

Acurácia Naive Bayes Validação : 0.7785714285714286  
Acurácia Naive Bayes Teste : 0.8533333333333334

## 2.2 Árvores de decisão

A técnica da árvore de decisão utiliza a teoria do ganho de informação (entropia) , para decidir seus novos vértices até que consiga realizar a classificação das instâncias.

A entropia pode ser definida pela fórmula:

$$S = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad (3)$$

Onde  $p_+$  é a proporção de elementos positivos em  $S$  e  $p_-$  a de elementos negativos.

As árvores geradas para a este trabalho podem ser consultadas em: [Figura 6](#), [Figura 8](#)

### 2.2.1 Resultados do Treinamento

DEEP

Acurácia Decision Tree Teste : 0.8466666666666667

HAND-CRAFT

Acurácia Decision Tree Teste : 0.67

## 2.3 Redes Neurais

### 2.3.1 Resultados do Treinamento

As matrizes de confusão desta técnica podem ser consultadas em: [Figura 12](#) e [Figura 13](#), assim como suas curvas de aprendizado podem ser consultadas em [Figura 14](#) e [Figura 17](#)

DEEP

Acurácia Neural Network - adam Validação : 0.9357142857142857  
Acurácia Neural Network - adam Teste : 0.92

## HAND-CRAFT

Acurácia Neural Network - adam Validação: 0.10714285714285714

Acurácia Neural Network - adam Teste : 0.1

### 2.4 KNN

A técnica de k-nearest neighbors, k-vizinhos mais próximos, consiste em agrupar instâncias similares, e quando novas instâncias são inseridas para a classificação, elas são classificadas pelos  $k$ -vizinhos mais próximos.

Uma grande desvantagem desta abordagem que dispensa treinamento é que ela é mais lenta, uma vez que realiza todos os cálculos necessários a cada nova classificação

#### 2.4.1 Resultados

As matrizes de confusão podem ser consultadas na [Figura 15](#) e [Figura 16](#)

DEEP

Acurácia KNN Teste : 0.96

## HAND-CRAFT

Acurácia KNN Teste : 0.3966666666666667

### 2.5 SVM

A técnica de SVM consiste em "encontrar" novas dimensões para classificar os atributos

#### 2.5.1 Resultado do Treinamento

As matrizes de confusão podem ser consultadas na [Figura 10](#) [Figura 11](#)

DEEP

Acurácia SVM Teste : 0.9833333333333333

## HAND-CRAFT

Acurácia SVM Teste : 0.7533333333333333

## 3 Resultados

Os indutores tiveram resultados mais consistentes utilizando a base com características extraídas via deep-learning, que por mais que tenha menos características que a base que utilizou hand-crafting, estas características representam melhor o problema.

Utilizando a técnica t-SNE, que segundo [Maaten \(2019\)](#), é uma técnica de redução de dimensionalidade estocástica, ou seja, utilizando esta técnica é possível reduzir um vetor de tamanho  $N$  para um de tamanho  $X$ , foi realizada a redução das bases de características

e apresentadas graficamente, para assim analisar o quão bem distribuídos estão as classes, estes gráficos podem ser conferidos na [Figura 18](#) e [Figura 19](#)

É possível notar que os pontos na [Figura 19](#) estão extremamente misturados, o que dificulta a classificação, com isso seria necessário um trabalho de limpeza de características ou a extração de novas características que representem melhor o problema.

A técnica de SVM usando a base de Deep Learning, trouxe os melhores resultados para esta base de treinamento, sua margem de acerto foi de 98% e foram poucos as imagens que foram classificadas erradas, como é possível ver em sua respectiva matriz de confusão na [Figura 10](#) e [Figura 11](#), podemos notar também que, as imagens em que este método errou, é com uma pequena taxa de confiança:

```
Confundi "base/humanos/72.jpg" com comida, proba: [0.31407948]
Confundi "base/humanos/79.jpg" com praia, proba: [0.49498053]
Confundi "base/obras/225.jpg" com montanhas, proba: [0.40270914]
Confundi "base/humanos/73.jpg" com praia, proba: [0.45795092]
Confundi "base/humanos/90.jpg" com praia, proba: [0.49488543]
```

Estas imagens, se deram devido a classificação humana da base ter sido feita erroneamente, pois a imagem "humanos/72.png" não há humanos na foto e sim algo muito parecido com comida, na "humanos/79.png" é uma foto de um lago, também sem nenhum humano presente, na imagem "obras/225.jpg" também existem montanhas na foto, e as fotos "humanos/73.jpg" e "humanos/90.jpg" são fotos de praias e não de humanos.

## 4 Considerações finais

Em geral os indutores tiveram melhor utilizando a base de características usando Deep Learning.

O indutor que houve melhor desempenho nesta base é o SVM e que nas poucas imagens em que ele errou da base teste a classificação das imagens estavam classificadas erroneamente na própria base, sendo assim, a técnica encontrou erros de classificação na base e a efetuou a classificação corretamente.

## Referências

- BRITTO, A. *Aprendizagem Bayesiana*. 2019. Ppgia. Disponível em: <<http://www.ppgia.pucpr.br/~alceu/am/4%20-%20Naive%20Bayes%20and%20Bayesian%20Networks/NaiveBayes.pdf>>. Acesso em: 25 set 2019. Citado na página 3.
- BRITTO, A. *Conceitos Básicos*. 2019. Ppgia. Disponível em: <<http://www.ppgia.pucpr.br/~alceu/am/1%20-%20Introduction%20to%20ML/Conceitos%20B%e1sicos.pdf>>. Acesso em: 22 set 2019. Nenhuma citação no texto.
- COPPIN, B. *Inteligência Artificial*. 1. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A., 2004. Acesso em: 22 set 2019. Citado na página 2.

DOSCIATTI, M. M.; FERREIRA, L. P. C.; PARAISO, E. C. *Identificando Emoções em Textos em Português do Brasil usando Máquina de Vetores de Suporte em Solução Multiclasse*. 2013. PPGIA. Disponível em: <<http://www.ppgia.pucpr.br/~paraiso/mineracaodeemocoes/recursos/emocoesENIAC2013.pdf>>. Acesso em: 21 set 2019. Citado na página 2.

ERHAL, G. J. et al. *Um Sistema de Segmentação e Classificação de Imagens De Satélite*. 1991. INPE. Disponível em: <<http://sibgrapi.sid.inpe.br/col/sid.inpe.br/sibgrapi/2012/09.28.17.13/doc/Um%20sistema%20de%20segmentacao%20e%20classificacao.pdf>>. Acesso em: 21 set 2019. Citado na página 2.

GROUT, I. *Learn more about Infinite Impulse Response*. 2008. Sciencedirect. Disponível em: <<https://www.sciencedirect.com/topics/computer-science/infinite-impulse-response>>. Acesso em: 24 set 2019. Citado na página 3.

LEARN scikit. *Incremental learning*. 2019. Scikit-learn. Disponível em: <[https://scikit-learn.org/0.18/modules/scaling\\_strategies.html#incremental-learning](https://scikit-learn.org/0.18/modules/scaling_strategies.html#incremental-learning)>. Acesso em: 24 set 2019. Citado na página 3.

LIMA, I.; PINHEIRO, C. A. M.; SANTOS, F. A. O. *Inteligência Artificial*. 1. ed. Rio de Janeiro: CAMPUS, 2014. Acesso em: 22 set 2019. Citado na página 2.

MAATEN, L. van der. *Aprendizagem Bayesiana*. 2019. Lvdmaaten. Disponível em: <<https://lvdmaaten.github.io/tsne/>>. Acesso em: 25 set 2019. Citado na página 5.

MALLICK, S. *Histogram of Oriented Gradients*. 2016. Learnopencv. Disponível em: <<https://www.learnopencv.com/histogram-of-oriented-gradients/>>. Acesso em: 22 set 2019. Citado na página 2.

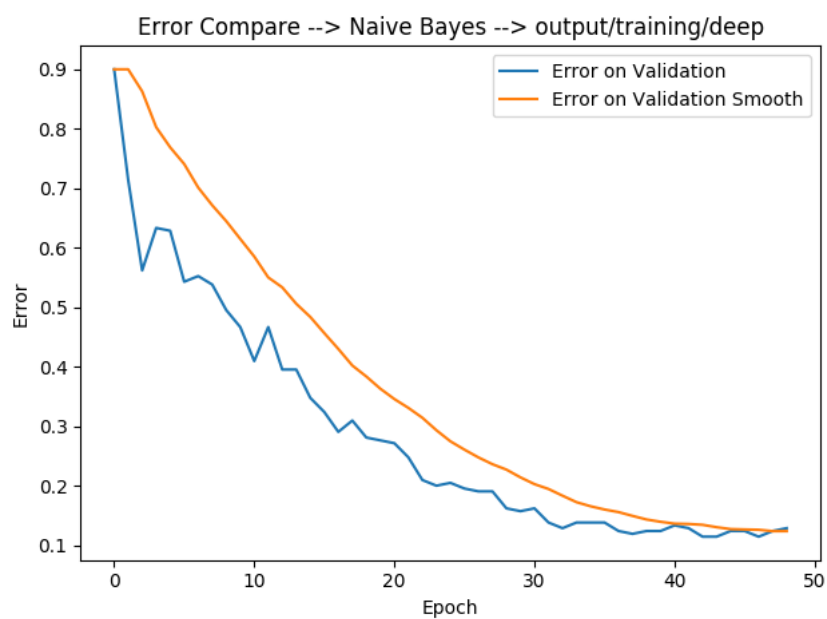
RIBAS, F. C. et al. *Handwritten digit segmentation: a comparative study*. 2012. UFPR. Disponível em: <<http://www.inf.ufpr.br/lesoliveira/download/IJDAR2012.pdf>>. Acesso em: 22 set 2019. Citado na página 2.

ROSEBROCK, A. *Local Binary Patterns with Python & OpenCV*. 2015. Pyimagesearch. Disponível em: <<https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>>. Acesso em: 22 set 2019. Citado na página 2.

SARLE, W. S. *Stopped Training and Other Remedies for Overfitting*. 2016. Overfitting. Disponível em: <<https://s3.amazonaws.com/academia.edu.documents/31153602/1329.pdf>>. Acesso em: 22 set 2019. Citado na página 2.

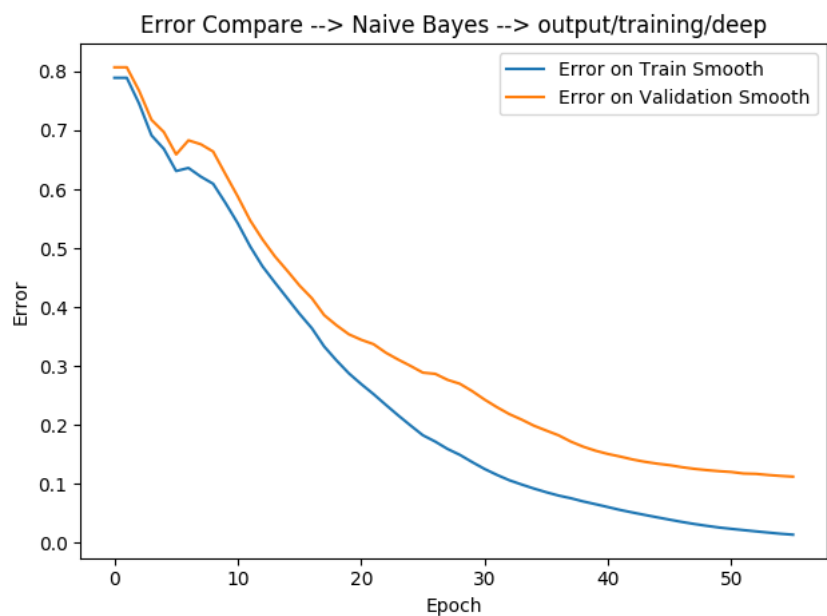
SOARES, H. B. *Análise e Classificação de Imagens de lesões da pele por atributos de cor, forma e textura utilizando máquina de vetor de suporte*. 2008. UFRN. Disponível em: <[https://repositorio.ufrn.br/jspui/bitstream/123456789/15118/1/HelianaBS\\_TESE.pdf](https://repositorio.ufrn.br/jspui/bitstream/123456789/15118/1/HelianaBS_TESE.pdf)>. Acesso em: 21 set 2019. Citado na página 2.

Figura 1 – Comparação do Erro com e sem IIR, usando Holdout



Fonte: O Autor

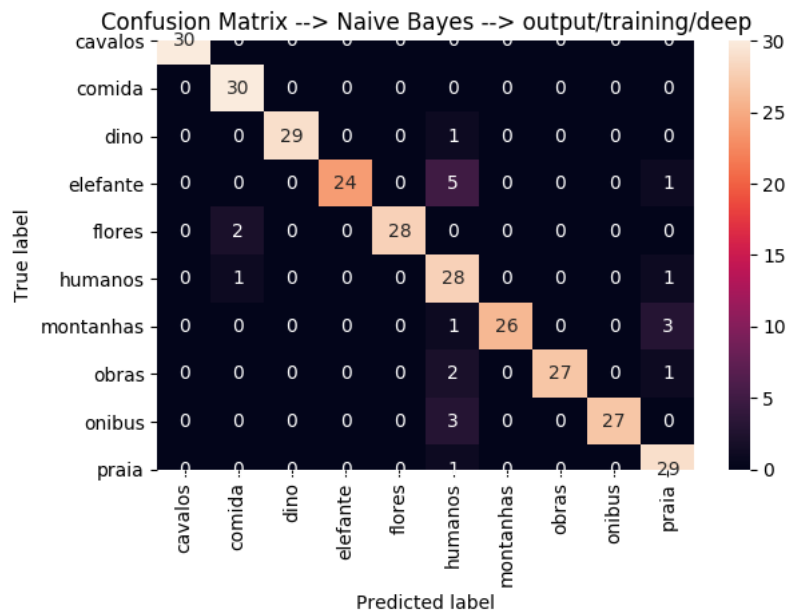
Figura 2 – Gráfico de Margem de erro usando Holdout



Fonte: O Autor

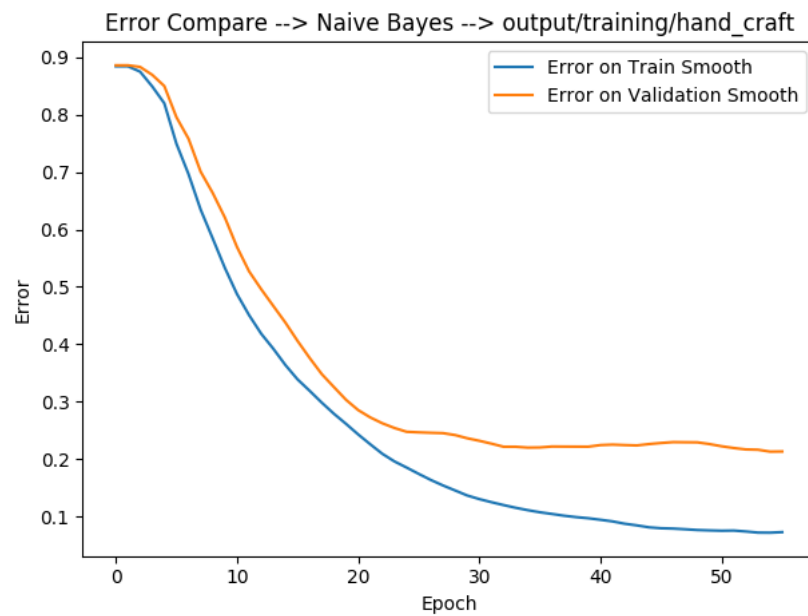


Figura 3 – Matrix de confusão Naive Bayes, usando Holdout



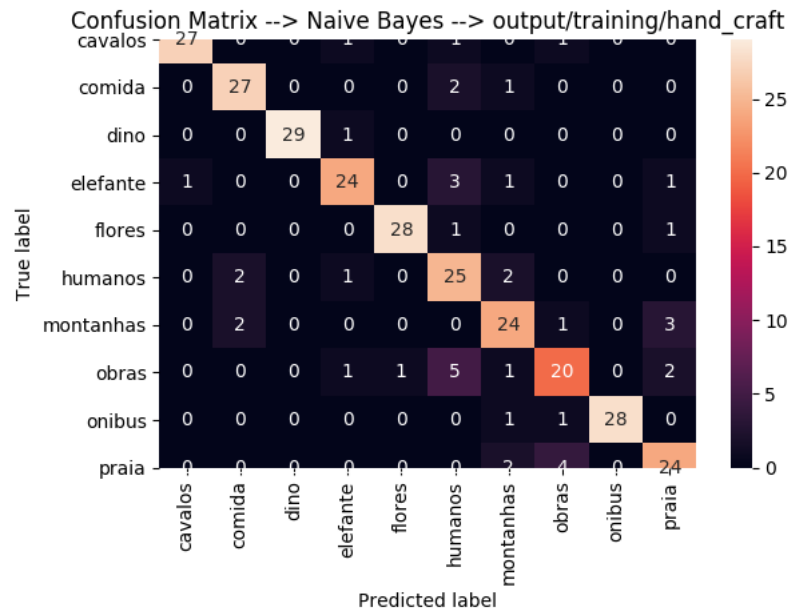
Fonte: O Autor

Figura 4 – Gráfico de treinamento Naive Bayes, usando Holdout



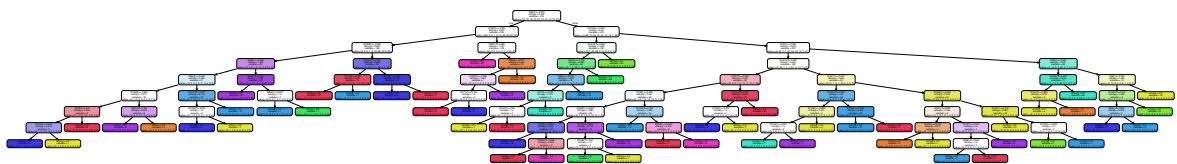
Fonte: O Autor

Figura 5 – Matrix de confusão Naive Bayes, usando Holdout



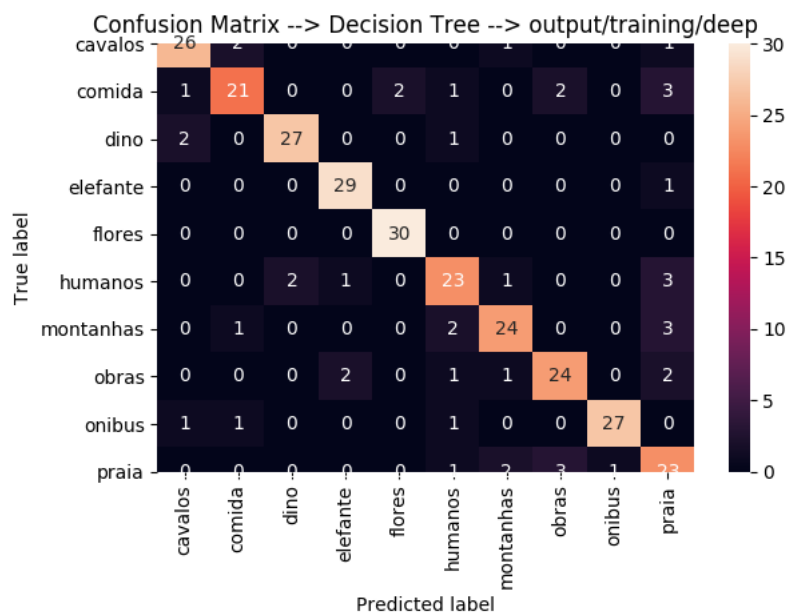
Fonte: O Autor

Figura 6 – Decision Tree, usando Holdout e Deep Learning



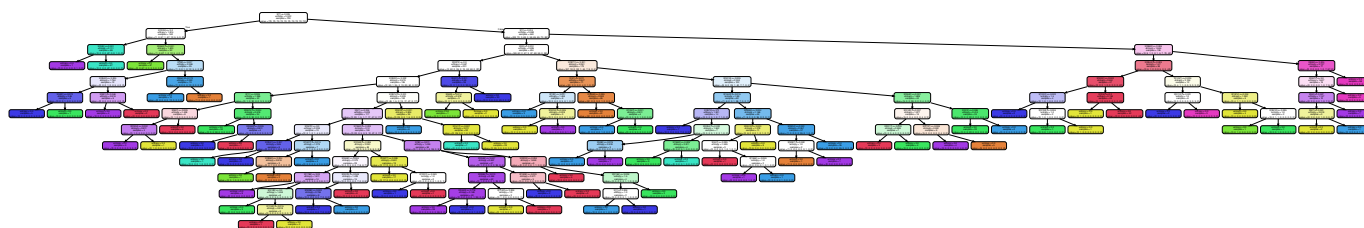
Fonte: O Autor

Figura 7 – Matrix de confusão Decision Tree, usando Holdout e Deep Learning



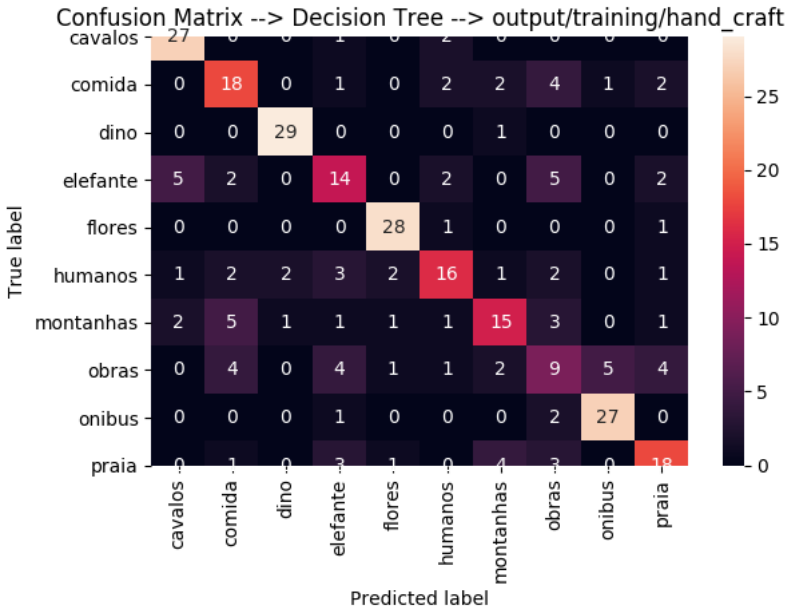
Fonte: O Autor

Figura 8 – Árvore de Designação, usando Holdout e Hand Craft



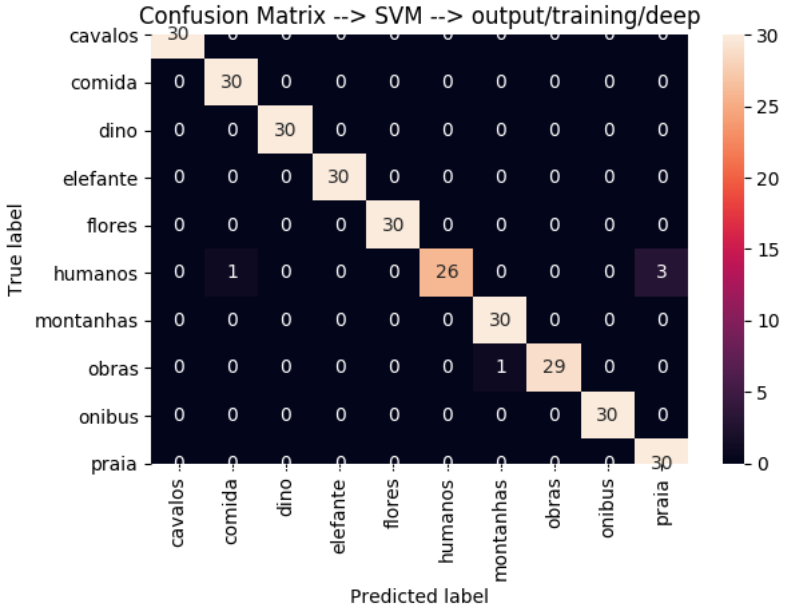
Fonte: O Autor

Figura 9 – Matrix de confusão Decision Tree, usando Holdout e Hand Craft



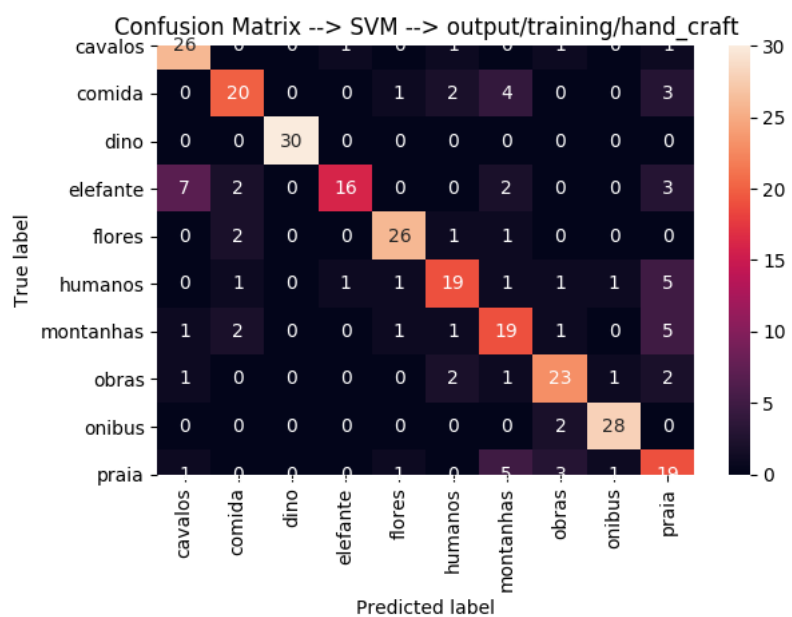
Fonte: O Autor

Figura 10 – Matrix de confusão SVM, usando Holdout e Deep Learning



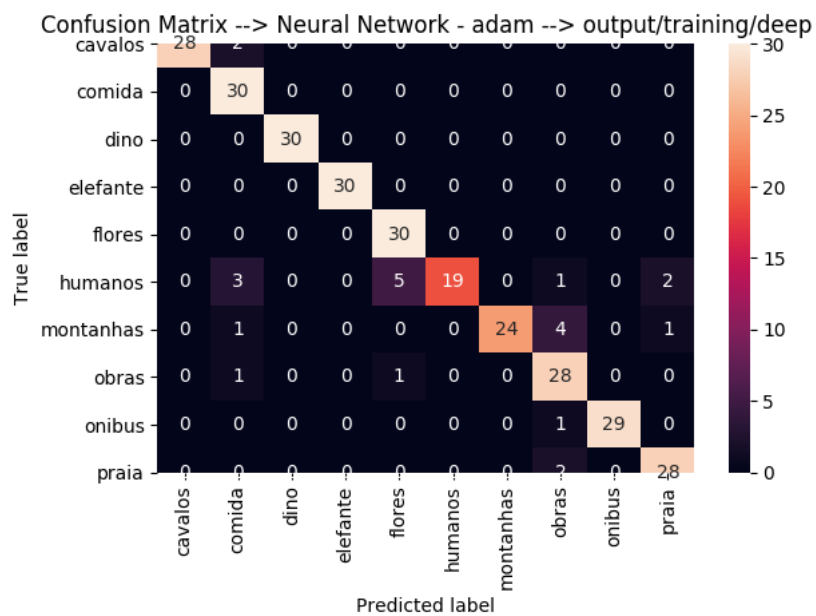
Fonte: O Autor

Figura 11 – Matrix de confusão SVM, usando Holdout e Hand Craft



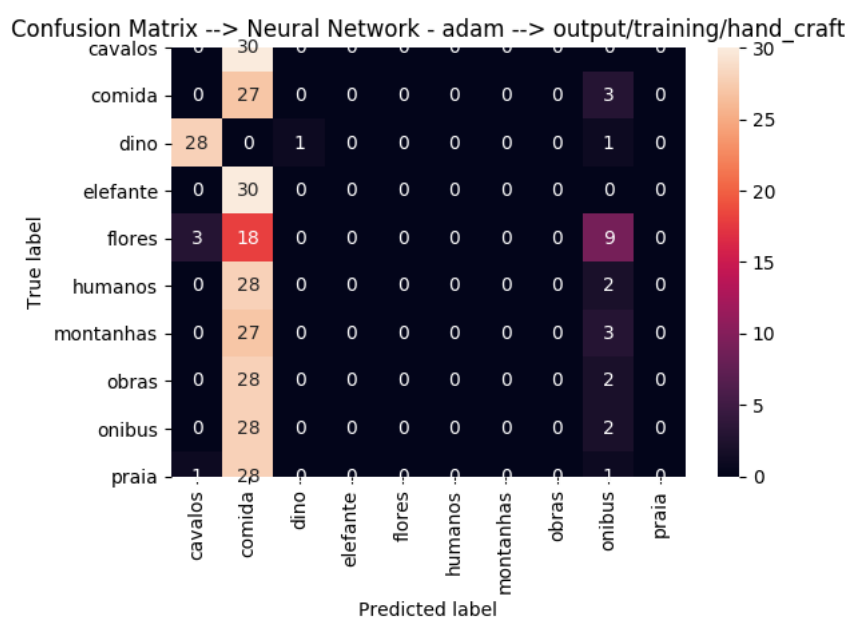
Fonte: O Autor

Figura 12 – Matrix de confusão Redes Neurais, usando Holdout e Deep Learning



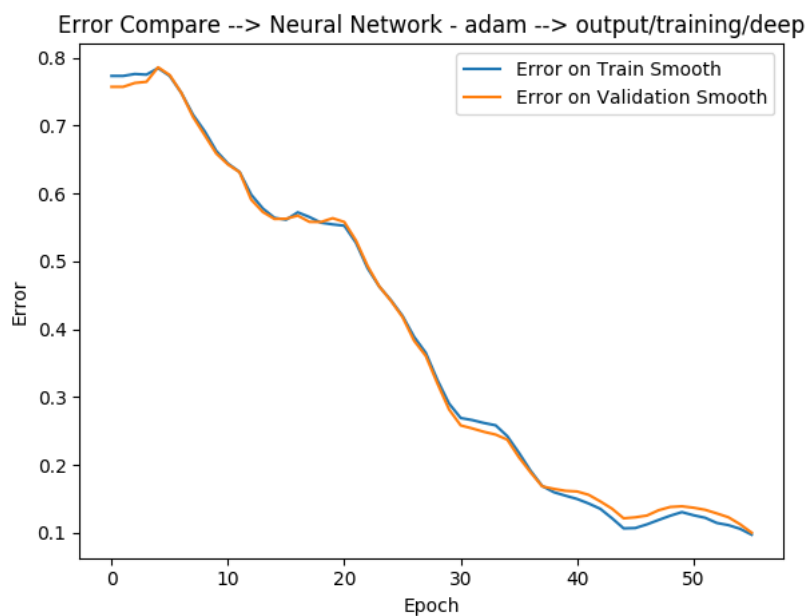
Fonte: O Autor

Figura 13 – Matrix de confusão Redes Neurais, usando Holdout e Hand Craft



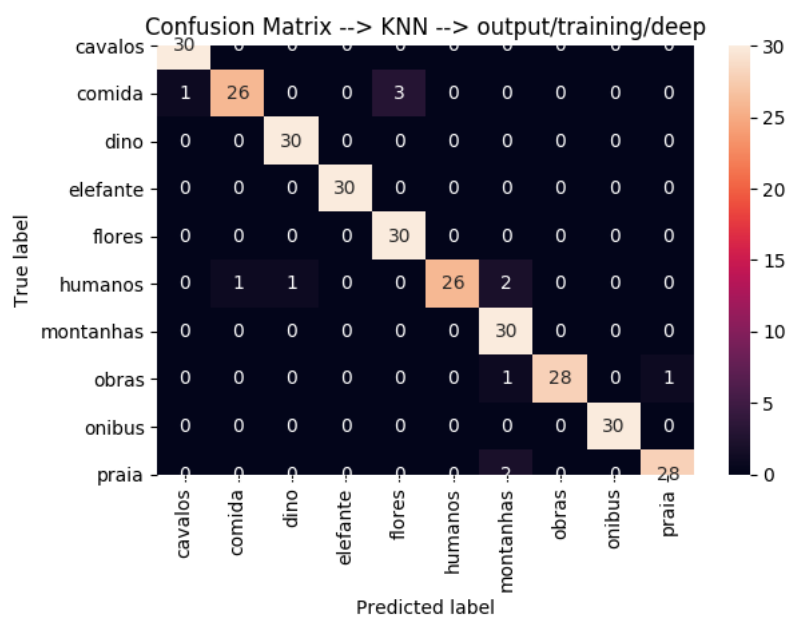
Fonte: O Autor

Figura 14 – Gráfico da Margem de Erro utilizando Holdout e Deep Learning



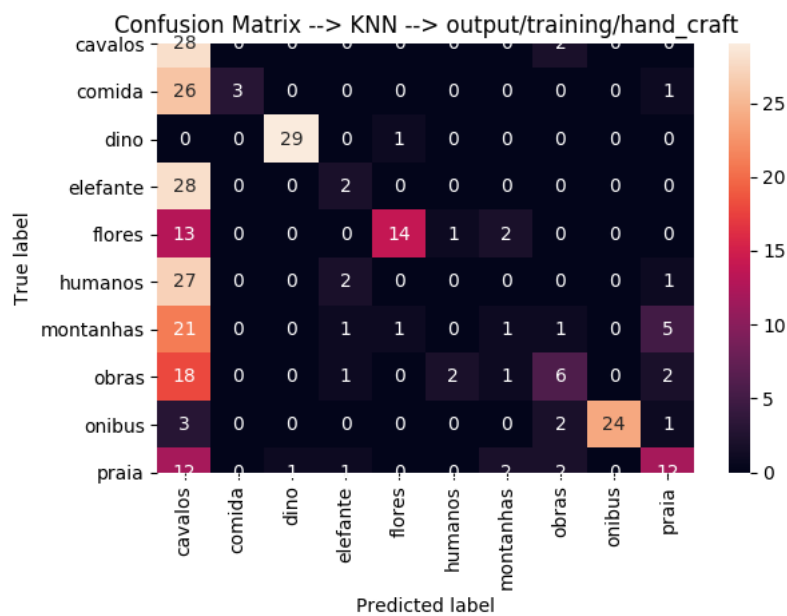
Fonte: O Autor

Figura 15 – Matrix de confusão KNN, usando Holdout e Deep Learning



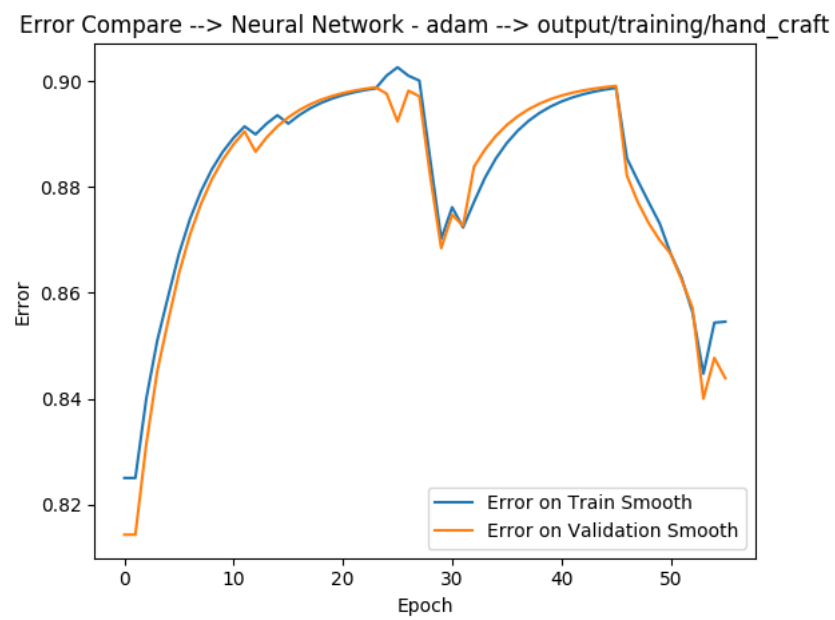
Fonte: O Autor

Figura 16 – Matrix de confusão KNN, usando Holdout e Hand Craft



Fonte: O Autor

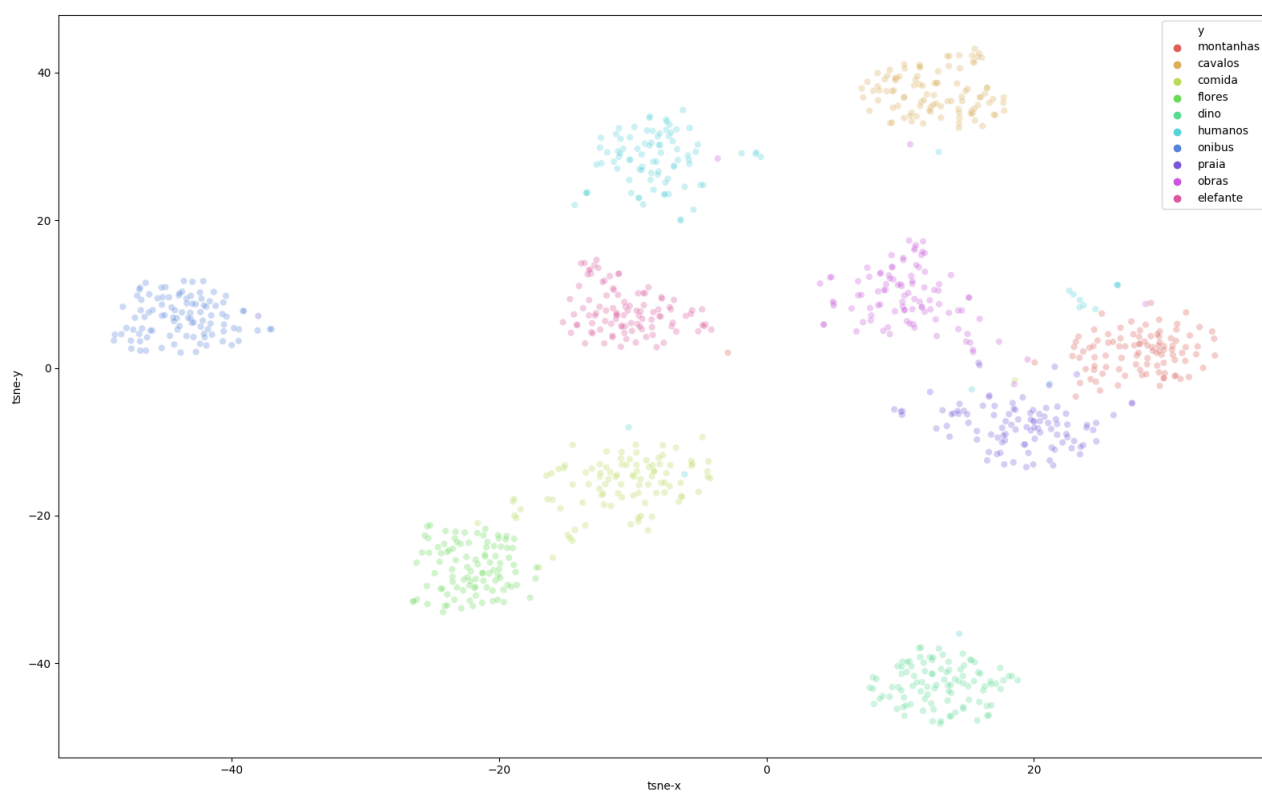
Figura 17 – Gráfico da Margem de Erro utilizando Holdout e Hand Craft



Fonte: O Autor

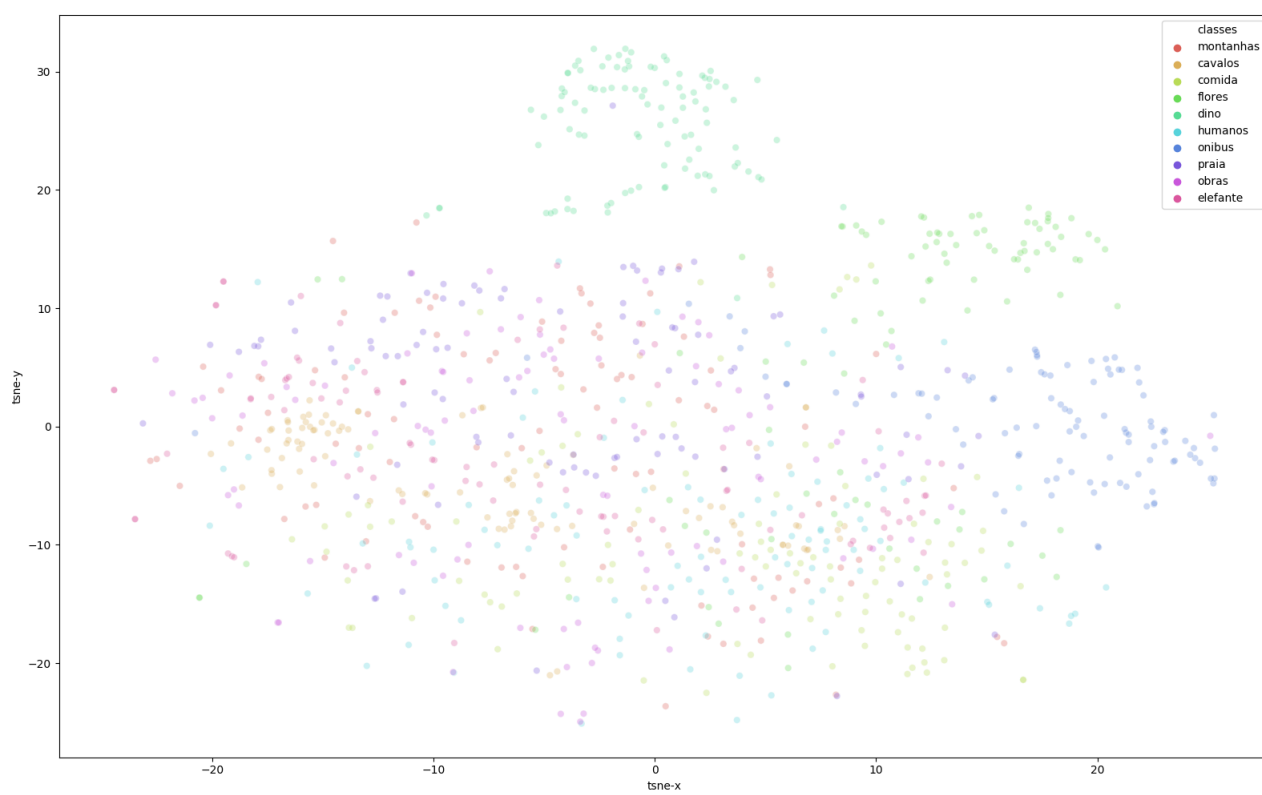


Figura 18 – TSNE Base Deep Learning



Fonte: O Autor

Figura 19 – TSNE Base Hand Crafting



Fonte: O Autor