

# Estudo e Aplicações de Máquinas de Suporte Vetorial.

**Levy Souto Sousa (Voluntário)<sup>1</sup>**

**José Fontebasso Neto (Orientador)<sup>2</sup>**

**Universidade Católica de Santos**

**Curso: Ciência da Computação**

<sup>1</sup> levys@unisantos.br; <sup>2</sup> jfneto@unisantos.br

## RESUMO:

Nesta pesquisa, foram empregados algoritmos de *Machine Learning* para realizar processamento de imagens de tomografia e ressonâncias magnéticas em três dimensões do depósito MRI, como cérebros, pulmões, fígado, entre outros, algumas delas contendo tumores ou edemas. Baseando o estudo em modelos de Classificação, foram usadas Máquinas de Suporte Vetorial (SVM).

Com o processamento das diversas imagens, será possível separá-las em treino, *labels*(etiquetas) e teste, assim o algoritmo deverá ser capaz de prever em quais imagens de teste há um tumor ou não, e o nível de acurácia baseado nas *labels*.

Em outros algoritmos de classificação, os resultados foram obtidos de forma rápida, em apenas alguns segundos, porém o SVM não se mostrou vantajoso em relação aos demais métodos para trabalhar com o *Dataset* escolhido, devido ao grande tempo e esforço computacional.

**Palavras-chave:** SVM, *Machine Learning*, imagens.

## 1 INTRODUÇÃO

Nesta Seção, serão expostos os atributos utilizados para a realização deste Estudo, as quais serão: O Algoritmo de aprendizado de Máquina Empregado; A base de dados; as ferramentas tecnológicas e por fim, o Objetivo do Estudo.

### 1.1 Sobre Máquinas de Suporte Vetorial.

Máquina de Suporte Vetorial (SVM) é um algoritmo supervisionado de *Machine Learning* utilizado para classificação e regressão de dados, separando-os em classes separadas linearmente ou não por um hiperplano. É dividido em módulos

SVC para classificação e SVR para regressão. Na Figura 1.0 há um exemplo de classificação para dados rotulados utilizando SVC.

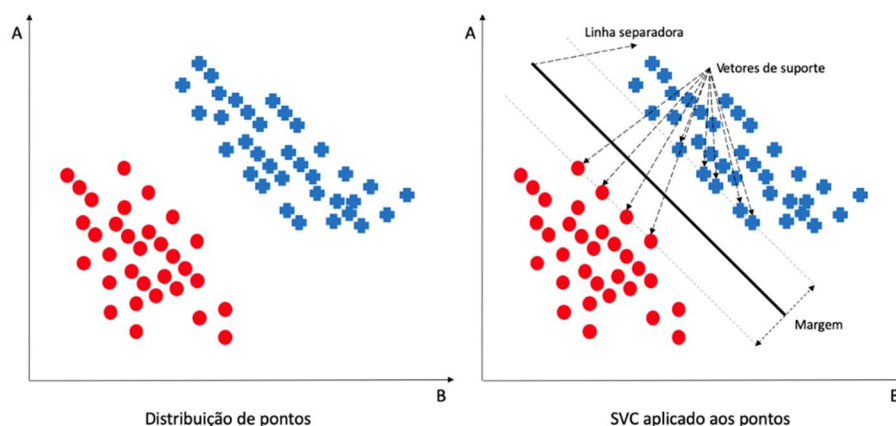


Figura 1.0

Fonte: Digital House

O SVM recebe dois parâmetros: *gamma* e *C* (*Scikit-Learn Documentation*); o parâmetro *gamma* é responsável por definir a influência de uma amostra no treinamento, quanto menor seu valor, pontos mais distantes são considerados no cálculo do hiperplano; Já o parâmetro *C* controla erros de margem na classificação e aplica penalidades afetando o hiperplano, quanto mais alto o valor do parâmetro *C*, mais clara será a separação entre classes, porém custa mais tempo de treinamento pela complexidade da formação de margens.

## 1.2 Sobre o *Dataset* de Neuro-Imagens.

Foram utilizadas tomografias de cérebros com tumores do Repositório MRI, como a representação abaixo, onde são exploradas camadas para verificar existência de tumores (PAL, Arghya; RATHI, Yogesh). As imagens com extensão .nii.gz podem ser manipuladas pela biblioteca Nibabel, são divididas em 4 capturas, cada uma com 3 dimensões: altura, largura e profundidade (240X240X155).

O *Dataset* contém 484 imagens para treino do algoritmo, assim como suas respectivas *labels*, além de mais 250 para testes de predição.

## 1.3 Ferramentas e Bibliotecas.

A principal ferramenta utilizada foi a linguagem de programação Python, devido à sua gama de funcionalidades para manipulação de dados. Dentro desse escopo foram manuseadas as seguintes ferramentas: *Scikit-Learn*, *framework open-source* para suporte à algoritmos de Inteligência Artificial; Matplotlib, biblioteca para criação gráfica; Numpy para funções matemáticas, arranjos e matrizes; Nibabel, para manipular Neuro-Imagens.nii.gz; Nilearn, para trabalhar com Machine Learning em Neuro-Imagens, e Pickle, que permite guardar variáveis em arquivos e utilizá-las

posteriormente no programa. As interfaces de desenvolvimento para interpretar o código-fonte foram Google Collab e Jupyter Notebook.

#### 1.4 Objetivo da Pesquisa.

O estudo tem como propósito, averiguar a aplicabilidade das Máquinas de Suporte Vetorial, bem como, entender seu processo de funcionamento.

Para corroborar com esta tarefa, serão utilizadas Neuro-Imagens de cérebros com e sem tumores, estes dados processados e submetidos ao Algoritmo, para verificarmos se o mesmo terá a capacidade de classificar se existem tumores e quais os tipos apresentados, por meio da taxa de acerto.

## 2. PROCEDIMENTOS DE PESQUISA

### 2.1 Compreendendo o objeto de estudos pela análise individual das imagens.

Inicialmente é feito o carregamento do arquivo pelo Google Drive, a imagem é processada, selecionada e transformada em vetor numpy, reduz-se a dimensionalidade para 3D, a partir disso podemos criar as visualizações por camadas e eixos com Matplotlib, em algumas é possível notar o tumor.

Após estes processos, a biblioteca Nilearn é empregada com o intuito de aplicar a Máquina de Suporte Vetorial nos três eixos da imagem, a função *view\_img* permite interagir com o hiperplano e explorar as camadas, separando a visualização do tumor do resto do cérebro.

### 2.2 Separando dados para treino e teste.

Em seguida são carregadas aproximadamente apenas 5 imagens de treino, *labels* e 2 para teste devido à limitação da Memória RAM, não é possível transferir todos os arquivos. A biblioteca Pickle permitiu guardar as variáveis em arquivos e carregá-las posteriormente com propósito de reutilizá-las sem grande esforço computacional.

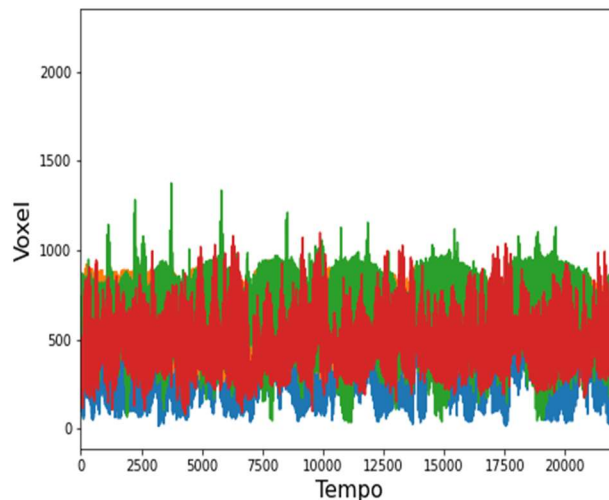
Nas imagens de treino e testes, foram aplicadas máscaras booleanas para retirar o plano de fundo, e outras informações que não fossem o cérebro em si, após este processo, as variáveis de treino, *labels* e teste foram redimensionadas para respectivamente 2 dimensões (amostras, características), 1 dimensão(amostras) e 2 dimensões (amostras, características).

Finalmente é possível a aplicação do SVM a partir da documentação do *Scikit-Learn* em parte do *Dataset* para predição e classificação, o algoritmo deverá ser capaz de identificar se há tumores nas imagens de teste ou não, e revelar a precisão dos acertos (*Scikit-Learn Documentation*).

### 3. RESULTADOS E DISCUSSÃO

A imagem tem sua estrutura formada por *voxels* (grade regular tridimensional, equivalente a pixels em figuras 2d), podemos contar quantos *voxels* há em cada série temporal por imagem na Figura 2.0.

**Figura 2.0: contagem de voxels**



Fonte: Autor

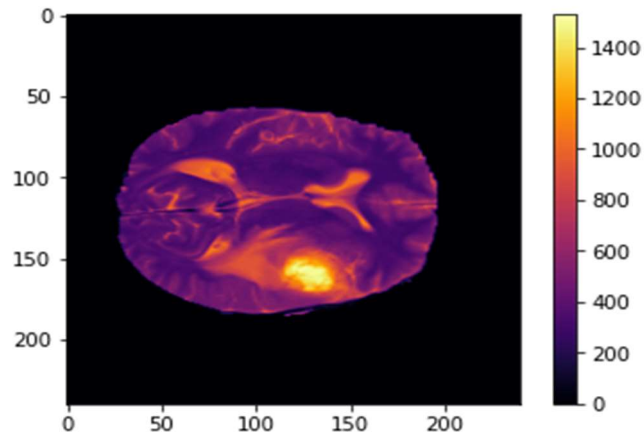
A Figura 2.1 é o código para criar a visualização da Figura 2.2 com a biblioteca Matplotlib, a função `para_3d` reduz a dimensionalidade da série temporal para 3d e retorna em vetor numpy. Na Figura 2.2 é possível observar o tumor.

**Figura 2.1: Função para reduzir dimensionalidade**



Fonte: Autor

**Figura 2.2: Visualização Gráfica**



Fonte: Autor

A Figura 2.3 apresenta o código para a exibição das camadas no eixo Z(cima), através de um índice de repetição que seleciona a partir das 60, dez camadas até a 110.

**Figura 2.3: Código para visualização por camadas**

```
fig, axes = plt.subplots(nrows=1,ncols=6)

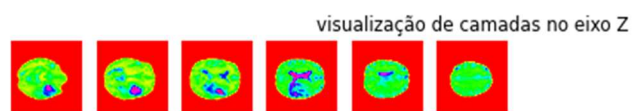
indices =[60,70,80,90,100,110]
for i in range(6):

    cerebro = img_3d[:, :, indices[i]]
    axes[i].imshow(cerebro,cmap='hsv')
    axes[i].axis('off')

plt.title('visualização de camadas no eixo Z')
plt.show()
```

Fonte: Autor

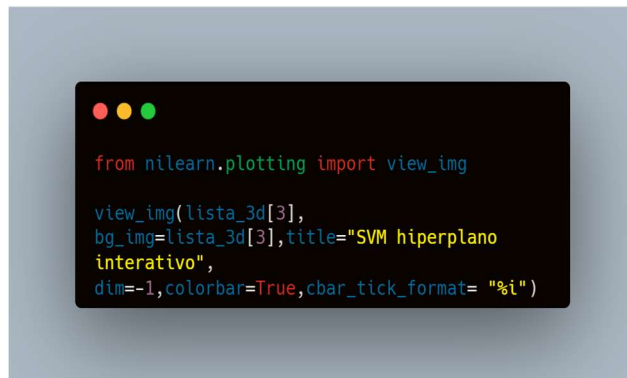
**Figura 2.4: Visualização por camadas**



Fonte: Autor

As Figuras 2.5 e 2.6 apresentam respectivamente o código para aplicação do SVM na imagem, é uma função interativa, sendo possível arrastar o hiperplano em qualquer eixo, o qual exibe a área a qual o tumor abrange.

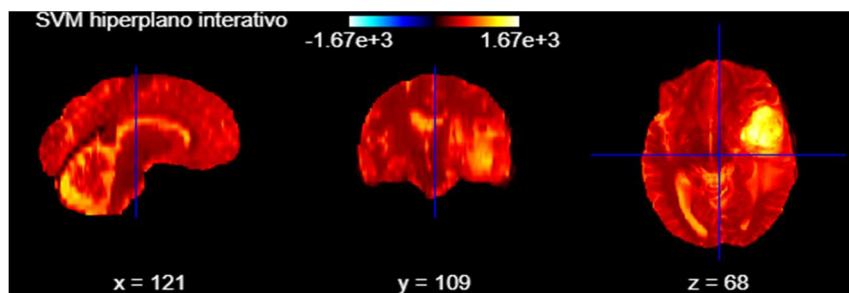
**Figura 2.5: Código para implementação do Hiperplano do SVM**



```
from nilearn.plotting import view_img  
  
view_img(lista_3d[3],  
bg_img=lista_3d[3],title="SVM hiperplano  
interativo",  
dim=-1,colorbar=True,cbar_tick_format= "%i")
```

Fonte: Autor

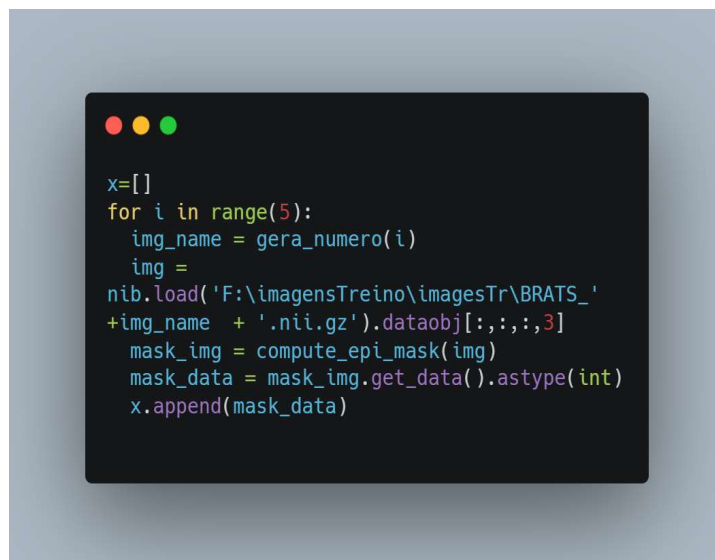
**Figura 2.6: Visualização com Hiperplano Interativo**



Fonte: Autor

Agora, são separados treino, teste e *labels*, aplicando máscaras para remover o plano de fundo e guardando seus valores em listas, seguindo o mesmo processo da Figura 2.7.

**Figura 2.7: Separando Treino e aplicando máscara**



Fonte: Autor

Após os dados lidos, são transformados em *arrays* numpy e formatados para uma dimensão. Como mostra a Figura 2.8

**Figura 2.8: Trabalhando com *arrays* numpy e redimensionamento**

```
x = np.array(x)
y=np.array(y)
t= np.array(t)
x,t,y =
x.reshape(5,240*240*155),t.reshape(2,240*240*1
55),y.reshape(5,240*240*155)#dados com 2
dimensões
y = y.reshape(5*240*240*155)
x=x.reshape(5*240*240*155)#normalizando x e y
para uma dimensão
x = x.reshape(-1,1)
```

Fonte: Autor

Finalmente, é importado o módulo SVC do *Scikit-Learn*, os dados são mais uma vez separados com *train\_test\_split*, é realizado o fit e calculado a taxa de acerto. Como mostra a Figura 2.9.

**Figura 2.9: Código para aplicar dados treino e teste no classificador**

```
from sklearn.svm import SVC
clf = svm.SVC(kernel = 'rbf', gamma = 'auto')
from sklearn.model_selection import
train_test_split

clf =
OneVsRestClassifier(BaggingClassifier(SVC(kern
el='rbf', probability=True,
class_weight='balanced'),max_samples=1.0/n_est
imators,n_estimators=n_estimators) )

clf.fit(x_train,y_train)
clf.predict(t)

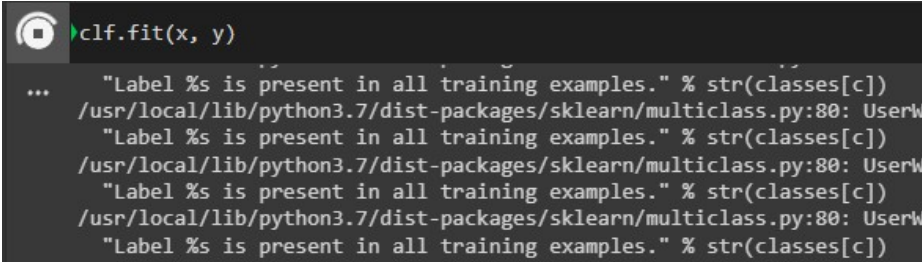
acerto = clf.score(x_test,y_test) * 100
print("taxa de acerto %d%%" %acerto)
```

Fonte: Autor

A Figura 3.0 mostra o código *fit* em execução, depois de algumas horas processando devido ao cálculo quadrático do SVM temos o resultado abaixo.



**Figura 3.0: Código em Execução**

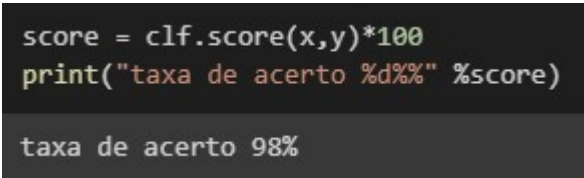


```
clf.fit(x, y)
...
"Label %s is present in all training examples." % str(classes[c])
/usr/local/lib/python3.7/dist-packages/sklearn/multiclass.py:80: UserWarning
"Label %s is present in all training examples." % str(classes[c])
/usr/local/lib/python3.7/dist-packages/sklearn/multiclass.py:80: UserWarning
"Label %s is present in all training examples." % str(classes[c])
/usr/local/lib/python3.7/dist-packages/sklearn/multiclass.py:80: UserWarning
"Label %s is present in all training examples." % str(classes[c])
```

Fonte: Autor

Por fim, a Figura 3.1 mostra a taxa de acerto algoritmo, e apesar de alta, não se pode dizer que é satisfatória, devido ao baixo número de dados de treino, em sistemas computacionais mais avançados, há a possibilidade de treinar o algoritmo com mais dados e mais *labels*.

**Figura 3.1: Demonstração da Taxa de Acerto**



```
score = clf.score(x,y)*100
print("taxa de acerto %d%%" %score)

taxa de acerto 98%
```

Fonte: Autor

## 4 CONSIDERAÇÕES FINAIS

A partir do que foi exposto, conclui-se que a Máquina de Suporte Vetorial é eficiente quando se trata de previsões por classificação, tanto em circunstâncias mais simples como venda de automóveis distintos por ano, quanto em circunstâncias complexas como a detecção de tumores cerebrais. Embora não tenha sido possível treinar e testar com o *Dataset* completo devido à limitação de hardware e esforço computacional, obtém-se uma visão positiva da capacidade do algoritmo em ambientes com maior potencial de processamento.

Diante disso, podemos inferir que a Máquina de Suporte Vetorial é eficiente e pode ser aplicada em ambientes comerciais e até mesmo hospitalares, supervisionada e

treinada por um profissional, pode ser de grande utilidade para o meio a qual esta tecnologia estiver inserida.

## 5 REFERÊNCIAS

Osuna E., Freund R., and Girosi F., "Support Vector Machines: Training and Applications", A.I. Memo No. 1602, Artificial Intelligence Laboratory, MIT, 1997.

MORETTIN, Pedro A.; SINGER, Julio M. **Introdução à ciência de dados fundamentos e aplicações**. São Paulo-SP: Departamento de Estatística da Universidade de São Paulo, IMEUSP, 2020. 354 p.

EVGENIU, Theodorus; PONTIL, Massimiliano. **Support Vector Machines: Theory and Applications** DOI: [10.1007/3-540-44673-7](https://doi.org/10.1007/3-540-44673-7) 12. Disponível em: [https://www.researchgate.net/publication/221621494\\_Support\\_Vector\\_Machines\\_Theory\\_and\\_Applications](https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Applications), janeiro,2001.

DOBILAS, Saul. Classificador SVM e kernel RBF.

Disponível em:

<https://ichi.pro/pt/classificador-svm-e-kernel-rbf-como-fazer-melhoresmodelos-em-python-127248222170971>.

Michael Joseph, Jerrold Jeyachandra, and Erin Dickie (eds):

Data Carpentry: **Introduction to MRI Data Analysis**." Version 2019.11, November 2019,

Disponível em: <https://github.com/carpentries-incubator/SDC-BIDS-IntroMRI>

Bailey, Sthephen, "Exploring 3D images with matplotlib". Dezembro de 2017.

Disponível em: <https://www.youtube.com/watch?v=5jQVQE6yfio>

Scarpance, L. et al. Radiology Data from The Cancer Genome Atlas Glioblastoma Multiforme [TCGA-GBM] collection [Data set]. The Cancer Imaging Archive, 2016.Disponível em: <https://doi.org/10.7937/K9/TCIA.2016.RNYFUYE9>.

**SVM RBF parameters**, Scikit-Learn documentation disponível em: [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html#rbf-svm-parameters](https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#rbf-svm-parameters). Acesso em setembro de 2021.

PAL, Arghya; RATHI, Yogesh, A review of deep learning methods for MRI reconstruction.

PDF Disponível em: <https://arxiv.org/abs/2109.08618>. Cornwell University. Acesso em set 2021