

Análise de algoritmos de reconhecimento de padrões

Antônio Adelino da S. Neto^a, Armstrong Lohãns de M. G. Quintino^b

Garanhuns, Brasil

^a*antonio.asn03@gmail.com*

^b*lohansdemelo1108@gmail.com*

Abstract

O objetivo principal desse trabalho foi a elaboração e o estudo de dois algoritmos para sistemas de aprendizado de máquina, os quais trabalhariam na classificação de textos em linguagem natural. Tais programas foram o algoritmo da Árvore de Decisão e o algoritmo de Naive Bayes. Inicialmente são mostrados os conceitos básicos sobre cada algoritmo de decisão acima citado. Depois haverá uma apresentação das técnicas usadas para a análise e as devidas conclusões.

Keywords: Reconhecimento de Padrões, Árvore de Decisão, Naive Bayes

1. Introdução

Os seres humanos e alguns outros animais possuem, entre outras habilidades, a aptidão no reconhecimento de padrões. O ser humano, especificamente, possui essa capacidade muito bem desenvolvida e tem uma enorme facilidade no reconhecimento formas, dando a elas significado e valor. Dentre elas pode-se citar a fisionomia de outros seres humanos, formas animais e vegetais, características pessoais e afins.

Essa habilidade sempre foi muito importante, pois foi por meio dela que a espécie humana conseguiu desenvolver-se com mais facilidade ao longo do tempo, uma vez que ela permite a assimilação e inferência de características em formas aparentemente semelhantes. Partindo dessa premissa, é possível notar a relevância dessa aptidão em reconhecimento para o ser humano, em especial o reconhecimento de padrões, visto que é por meio dela que consegue-se inferir em formas desconhecidas julgamentos prévios a partir de conhecimentos anteriores.

Com isso, afirma-se então que toda e qualquer forma de reconhecimento de padrões, por indivíduos, dá-se a partir de uma experiência passada. Dessa maneira é possível perceber que a destreza, ou não, no reconhecimento de padrões está

6 de outubro de 2019

18 diretamente vinculada aos estímulos que cada indivíduo foi submetido ao longo
19 de sua vida [1].

20 Partindo dessas afirmativas, o presente artigo expõe um estudo que busca a
21 análise comparativa de dois algoritmos. O algoritmo da Árvore de Decisão e o
22 algoritmo de Naive Bayes, ambos voltados a classificação de dados baseando-se
23 nos princípios de aprendizagem de máquina.

24 A análise desses algoritmos dá-se por meio da classificação de textos em lin-
25 guagem natural, onde os classificadores recebem os textos e inferem a eles o sen-
26 tido do que está escrito de acordo com classes anteriormente estabelecidas.

27 **2. Referencial Teórico**

28 *2.1. Algoritmo da Árvore de Decisão*

29 As Árvores de Decisão são técnicas muito populares de aprendizado de má-
30 quina, são aplicadas às tarefas de classificação e regressão. Esta técnica é carac-
31 terizada pelo seu modelo resultante, o qual é codificado como uma estrutura em
32 árvore [2].

33 As árvores de decisão são algoritmos que buscam a classificação dos dados a
34 partir da estruturação em árvore. O algoritmo divide um conjunto de dados em
35 subconjuntos menores. Sabendo que o código estrutura-se em árvore, cada nó
36 folha representa uma decisão.

37 Para chegar em uma decisão, o algoritmo comporta-se da seguinte maneira,
38 com base nos valores dos recursos das instâncias, as árvores de decisão classificam
39 os dados. Cada nó representa um recurso em uma instância da árvore de decisão
40 que deve ser classificada, e cada ramo representa um valor [3].

41 Sabendo disso, percebe-se que cada dado, para ser classificado, passa por um
42 conjunto finito de nós, tal conjunto é definido como as regras de classificação, pois
43 a partir desse conjunto é possível saber o passo a passo do algoritmo, mostrando
44 assim todas as regras que levaram a classificação daquela única instância. A prin-
45 cipal vantagem do uso das árvores de decisão está justamente na capacidade do
46 retorno dos passos para a decisão e não unicamente no resultado da classificação.

47 *2.2. Algoritmo de Naive Bayes*

48 Além das árvores de decisão, pode-se também fazer uso de outros tipos de
49 classificadores, entre eles destaca-se o Naive Bayes, o qual possui uma análise
50 dos dados a partir de conceitos probabilísticos, diferenciando-se das árvores de
51 decisão.

52 O algoritmo de Naive Bayes é um classificador probabilísticos simples (ba-
53 seado no Teorema de Bayes), tem como base em uma suposição comum de que
54 todos os recursos são independentes um do outro [4]. A partir disso, ele descon-
55 sidera completamente a correlação entre todas as variáveis, tratando cada variável
56 de forma independente, esse algoritmo é frequentemente aplicado em processa-
57 mento de linguagem natural.

58 Uma das principais vantagens do classificador Naive Bayes é que ele requer
59 apenas uma pequena quantidade de dados iniciais de treinamento para poder esti-
60 mar as médias e variações das variáveis necessárias para classificação [5].

61 **3. Ferramentas usadas**

62 *3.1. Linguagem Python*

63 Antes de iniciar a implementação optamos por usar a linguagem de programa-
64 ção Python. Essa linguagem, além de ser uma das mais populares do mundo, é
65 vastamente usada na produção de softwares e algoritmos voltados aos conceitos
66 de aprendizagem de máquina, tanto no meio acadêmico quanto na indústria.

67 Pode-se afirmar também que a linguagem de programação Python é uma lin-
68 guagem de fácil manipulação e de grande eficiência produtiva o que facilita todo
69 o processo de codificação.

70 *3.2. Módulo Sickit-Learn*

71 Partindo dessa escolha inicial, foi possível utilizar o módulo Sickit-Learn para
72 Python. Tal módulo integra em si uma grande quantidade de algoritmos de apren-
73 dizagem de máquina que são voltados para problemas supervisionados e não su-
74 pervisionados [6].

75 Além disso, essa biblioteca Python tem como característica a simplificação
76 do uso de algoritmos de aprendizagem de máquina buscando a sua popularização.
77 Essa difusão dá-se por meio da grande facilidade de uso, do bom desempenho e da
78 sua documentação detalhada. Esse módulo ainda procura incentivar o seu uso, por
79 meio de dependências mínimas e licença simplificada, em ambientes acadêmicos
80 e comerciais de todo o mundo [6].

81 Diante dessas características encontradas na linguagem de programação Python
82 e na biblioteca Sickit-Learn a implementação dos algoritmos, que servem de ex-
83 perimento para o presente artigo, tornou-se mais rápida e mais objetiva.

84 3.3. *Wisdom Quotes (Base de dados)*

85 Para a criação da base de dados inicial dos classificadores foi necessário a
86 utilização do site <http://wisdomquotes.com> a fim de escolher textos aleató-
87 rios que fossem divididos por classificadores. O Wisdom Quotes conta com uma
88 grade variedade de citações separadas por temas, permitindo a relação intuitiva
89 citação-tema, o que vem a facilitar a criação e manipulação da base de dados dos
90 classificadores.

91 4. Algoritmos

92 4.1. *Criação da base de dados*

93 Antes de iniciar a implementação dos algoritmos brutos de aprendizado de
94 máquina (Árvore de Decisão e Naive Bayes) tivemos que coletar as citações do
95 site Wisdom Quotes. Para isso implementamos um script simples em Python
96 (Crawler) que teve como finalidade apenas a coleta das citações do endereço URL
97 que passamos e separar em arquivos de texto.

98 Com o Crawler o processo de coleta de dados tornou-se muito mais rápido e
99 eficaz. A coleta foi feita em três páginas do mesmo site, cada página continha cita-
100 ções de uma única categoria, as três categorias foram escolhidas randomicamente
101 e são elas:

- 102 1. Peace (do inglês, "Paz")
- 103 2. Success (do inglês, "Sucesso")
- 104 3. Silence (do inglês, "Silêncio")

105 Com as citações salvas em arquivos de texto, tivemos que escolher aleatori-
106 amente cento e cinquenta de cada classificador, a fim de deixar a base de dados
107 uniforme, totalizando quatrocentos e cinquenta frases.

108 A partir disso, armazenamos essas frases em dois vetores, o vetor de treino
109 e o vetor de teste. Essas listas continham respectivamente sessenta por cento e
110 quarenta por cento das citações, escolhidas aleatoriamente entre si. Essas porcen-
111 tagens foram estabelecidas visando um melhor desempenho dos classificadores,
112 uma vez que a quantidade de dados para treino é um pouco maior quando compa-
113 rada a quantidade de dados para testes.

114 4.2. Árvore de Decisão

115 Com a base de dados inicial pronta, implementamos a árvore de decisão. Para
116 usar a função de criação de uma árvore de decisão no Sickit-Learn é necessá-
117 rio inicialmente vetorizar as frases de teste, a função (também do Sickit-Learn)
118 `.fit_transform()` do `CountVextorizer()` vetoriza as frases em forma de matriz, além
119 de contar a repetição de cada palavra na frase, trecho ilustrado na Figura 1.

```
x = CountVectorizer()  
treinoFrase = x.fit_transform(treinoFrase).toarray()
```

Figura 1: Vetorizando frases de teste.

120 Com as frases devidamente vetorizadas, criamos a árvore de decisão a partir
121 da função `tree.DecisionTreeClassifier()` e a treinamos com o comando `.fit(parm1,`
122 `parm2)`, como mostrado na Figura 2, passando como parâmetro as frases vetori-
123 zadas e a lista de classificações. Cada frase vetorizada está localizada na posição
124 de índice correspondente ao seu classificador que está na lista de classificadores.

```
arvore = tree.DecisionTreeClassifier()  
arvore.fit(treinoFrase, treinoClass)
```

Figura 2: Instanciando e treinando árvore de decisão.

125 Após a criação da árvore, colocamos a base de testes para validar as classifi-
126 cações, a função `.score(parm1, parm2)` retorna o percentual de acerto dos testes
127 passados no segundo parâmetro, no primeiro parâmetro está o conjunto contendo
128 todas as palavras conhecidas pela árvore (*Bag of Words*), veja na Figura 3.

```
arvore.score(bagOfWords, testeClass)
```

Figura 3: Função que retorna a porcentagem de acertos da estrutura.

129 Tendo a árvore de decisão montada também foi possível obter as regras de
130 classificação citações passadas, para isso usamos a função, também do do Sickit-
131 learn, `.decision_path()`, conforme mostra a Figura 4.

132 Com isso, terminamos a implementação do algoritmo da árvore de decisão e
133 suas funcionalidades principais, dando destaque a possibilidade de listar o cami-
134 nho de decisão de uma citação.

```
arvore.decision_path(frase)
```

Figura 4: Função que retorna a as regras de decisão da árvore.

135 4.3. Naive Bayes

136 Para implementamos o classificador Naive Bayes no Sickit-Learn, foi neces-
137 sário realizar o mesmo processo de vetorização das frases de teste, ilustrado na
138 Figura 1. Tal processo teve como finalidade a vetorização das frases em forma
139 de matriz enumerando a repetição de cada palavra da citação recebida, como já
140 detalhado na subseção anterior.

141 A partir disso, criamos a estrutura do Naive Bayes por meio da função *Gausi-*
142 *anNB()* e a treinamos com o mesmo comando *.fit(parm1, parm2)* usado no treino
143 da árvore de decisão, como mostrado na Figura 5, passando como parâmetro as
144 frases vetorizadas e a lista de classificações, semelhante ao que foi feito na árvore
145 de decisão.

```
gnb = GaussianNB()  
nb = gnb.fit(treinoFrase, treinoClass)
```

Figura 5: Instanciando e treinando o algoritmo de Naive Bayes.

146 Depois de estruturar o Naive Bayes, colocamos a base de testes para vali-
147 dar as classificações, a função que faz isso é a mesma ilustrada na Figura 3
148 (*.score(parm1, parm2)*), a qual tem como retorno o percentual de acerto dos testes
149 passados.

150 Com o algoritmo de Naive Bayes da biblioteca Sickit-Learn foi possível aces-
151 sar as probabilidades de cada classe a partir do atributo *gnb.class_prior_*, esse
152 atributo é composto por uma lista com todas as probabilidades das classes, temos
153 o acesso a essas classes a partir do atributo *gnb.classes_* que retorna uma lista com
154 todas as classes, ambas as listas são organizadas de maneira que cada índice de
155 uma delas esteja diretamente associado ao índice da lista seguinte, veja na Figura
156 6.

157 Dessa maneira terminamos a implementação do algoritmo de classificação
158 Naive Bayes e suas principais funções, dando relevância a possibilidade de lis-
159 tar as probabilidades a partir de cada classe.

```
classes = gnb.classes_  
aPosteriori = gnb.class_prior_
```

Figura 6: Acesso aos atributos de classe e probabilidade do Naive Bayes.

160 4.4. Interação com o usuário

161 Após implementar os dois algoritmos de decisão acima descritos, elaboramos
162 um pequeno algoritmo que permite ao usuário colocar em um campo de texto uma
163 entrada e a partir dessa entrada o algoritmo retorna a sua classificação de acordo
164 com os classificadores (*Peace*, *Success* e *Silence*).

165 A classificação é feita a partir de uma comparação de maior taxa de acertos na
166 base de testes entre a árvore de decisão e o Naive Bayes, o que apresentar maior
167 taxa de acerto terá a sua classificação mostrada ao usuário, Figura 7.

```
if (nb.score(bagOfWords, testeClass) > arvore.score(bagOfWords, testeClass)):  
    print("A melhor classificação é: ",str(gnb.predict(fraseMatriz)[0]))  
else:  
    print("A melhor classificação é: ",str(arvore.predict(fraseMatriz)[0]))
```

Figura 7: Trecho de código para a tomada de decisão.

168 Além disso, o algoritmo retorna as regras de decisão da árvore e a probabili-
169 dade da classe, esse último a partir do Naive Bayes.

170 Esse algoritmo ainda permite ser executado na própria base de testes retor-
171 nando, com essa execução, a melhor classificação, as regras de decisão da árvore
172 e a probabilidade da classe.

173 5. Análises e testes

174 5.1. Árvore de Decisão

175 O primeiro teste realizado na árvore de decisão foi a validação dos dados de
176 teste, os quais representavam quarenta por cento de toda a base de dados, como
177 foi detalhado anteriormente. Para capturar as taxas de acerto usamos a função
178 `.score(parm1, parm2)`, também já descrita.

179 Os resultados obtidos a partir desse teste na árvore de decisão oscilaram por
180 volta de quarenta por cento de taxa de acerto.

181 Após essa primeira avaliação, testamos outra maneira de dividir e avaliar os
182 dados, para isso entramos novamente no site <http://wisdomquotes.com> e co-
183 letamos o máximo de citações possíveis sobre os três classificadores escolhidos
184 (*Peace*, *Success* e *Silence*) e separamos em arquivos, usando o sript Crawler.

185 Com esses novos dados fizemos uma validação cruzada usando uma fun-
186 ção da biblioteca Sickit-Learn, a função `cross_val_score(parm1, parm2, parm3,`
187 `parm4))`, o qual recebe no primeiro parâmetro a estrutura do classificador, no se-
188 gundo parâmetro recebe a base de dados, no terceiro parâmetro recebe os classi-
189 ficadores e no quarto parâmetro recebe a quantidade de subconjuntos que serão
190 divididos os dados para treino e teste, veja na Figura 8.

```
cross_val_score(estrutura, dados, classificadores, cv=5)
```

Figura 8: Função de validação cruzada.

191 Para validarmos o classificador da árvore de decisão, adotamos o valor de
192 cinco subconjuntos para fazer o teste. Com a validação cruzada dos cinco subcon-
193 juntos de dados a árvore de decisão trouxe uma média aproximada de sessenta por
194 cento de acerto, um acréscimo de vinte por cento quando comparado ao primeiro
195 teste.

196 Testamos também mudança nas quantidades de subconjuntos para a análise,
197 a partir de três subconjuntos a taxa média de acerto continuava por volta de ses-
198 senta por cento, entretanto notamos que quanto maior a quantidade de subcon-
199 juntos maior era a taxa de acerto, porém essa variação positiva foi extremamente
200 pequena, para duzentos subconjuntos a porcentagem média foi de cerca de sessenta
201 e quatro por cento, enquanto com três subconjuntos essa média foi de cerca de
202 sessenta e um por cento.

203 5.2. Naive Bayes

204 Semelhante ao que foi realizado na árvore de decisão, a validação dos dados de
205 teste, os quais também representavam quarenta por cento de toda a base de dados,
206 usamos a função `.score(parm1, parm2)`, também já descrita e ilustrada na Figura
207 3. Essa função retorna a taxa de acerto do algoritmo quando ele é executado na
208 base de testes.

209 Os resultados obtidos por meio desse teste no classificador Naive Bayes osci-
210 laram em torno de trinta e cinco por cento.

211 Depois dessa avaliação inicial e tendo novos dados coletados do site <http://wisdomquotes.com> separados pelos classificadores *Peace*, *Success* e *Silence*
212

213 fizemos uma validação cruzada, assim como feito na árvore de decisão, usando
 214 uma função *cross_val_score(parm1, parm2, parm3, parm4)*, a qual já foi descrita
 215 e detalhada na subseção anterior (Figura 8).

216 Adotamos o valor inicial de cinco subconjuntos para fazer o teste, baseando-
 217 se no experimento anterior da árvore de decisão. Com a validação cruzada dos
 218 cinco subconjuntos de dados o classificador Naive Bayes trouxe uma média de
 219 aproximadamente quarenta e três por cento de acerto, um acréscimo de quase dez
 220 por cento quando comparado ao primeiro teste.

221 Entretanto realizamos também mudança nas quantidades de subconjuntos para
 222 a análise, e notamos que não houveram grandes diferenças quando modificamos a
 223 quantidade de subconjuntos no algoritmo de Naive Bayes, uma vez que os valores
 224 de acerto ficaram em torno de quarenta e cinco por cento.

225 5.3. Erro real

226 Um cálculo importante de se fazer é o cálculo do erro real, Fórmula 1. A partir
 227 desse erro damos um intervalo de confiança ao nosso estimador. Determinamos
 228 um intervalo de confiança de 95%, com isso calculamos esse erro para os dois
 229 classificadores acima, tendo como referencia a taxa de acerto do primeiro teste.

$$erro_D(h) = erro_S(h) \pm z_N \sqrt{\frac{erro_S(h)(1 - erro_S(h))}{n}} \quad (1)$$

230 • Árvore de Decisão

$$erro_D(h) = 0,56 \pm 1,96 \sqrt{\frac{0,56(1 - 0,56)}{180}} \Rightarrow erro_D(h) \approx 0,56 \pm 0,072 \quad (2)$$

231 Ou seja, o erro real do algoritmo da árvore de decisão com confiança em
 232 95% é de 56% mais ou menos 7,2%.

233 • Naive Bayes

$$erro_D(h) = 0,66 \pm 1,96 \sqrt{\frac{0,66(1 - 0,66)}{180}} \Rightarrow erro_D(h) \approx 0,66 \pm 0,069 \quad (3)$$

234 Ou seja, o erro real do algoritmo de Naive Bayes com confiança em 95% é
 235 de 66% mais ou menos 6,9%.

236 5.4. Comparando classificadores

237 Ao desenvolver os dois classificadores queremos agora saber se eles apresen-
238 tam a mesma performance, para isso iremos fazer uma análise estatística dos dois
239 classificadores.

CLASS.	1	2	3	4	5	6	7	8	9	10
AD	65.1	64.7	64.7	65.8	62.3	59.0	67.4	67.4	57.8	62.6
NB	48.8	50.5	43.5	36.4	40.0	44.5	43.3	43.3	46.9	45.7

Tabela 1: Desempenho dos classificadores - árvore de decisão (AD) e Naive Bayes (NB).

240 Ao analisar essa tabela vemos que a média de acerto do classificador da árvore
241 de decisão é de 63,68% e a média do classificador Naive Bayes é de 44,29%. Ao
242 avaliar esses resultados, intuitivamente, concluímos que o classificador da árvore
243 de decisão é superior ao classificador Naive Bayes, entretanto para validar essa
244 afirmação é necessário realizar um teste de hipótese. Para o teste de hipóteses, é
245 preciso que saber a média das diferenças entre cada classificação, veja na tabela
246 2.

CLASS.	1	2	3	4	5	6	7	8	9	10
AD - NB	16.3	14.2	21.2	29.4	22.3	14.5	24.1	24.1	10.9	16.9
									Méd.:	19.3

Tabela 2: Diferença e média - Desempenho dos classificadores AD e NB.

247 Com os resultados ilustrados na Tabela 1 e 2, a taxa de acerto das classificações
248 a partir do algoritmo da árvores de decisão são maiores, e temos uma média entre
249 as diferenças de 19.39%. Definindo os testes de hipótese de acordo com a Figura
250 9:

$$H_0 : \mu_d = 0$$

$$H_1 : \mu_d \neq 0$$

Figura 9: Definição de hipóteses

251 Considerando que a hipótese H_0 assume que os classificadores são semelhan-
252 tes e H_1 assume que os classificadores são diferentes, então temos, a partir da

253 média das diferenças, que os classificadores são diferentes pois 19.39 é diferente
254 de zero.

255 Entretanto, para encontrar a resposta real, deve-se ter a média e a significância.
256 Iremos adotar o valor da significância de 99%, uma vez que esse é um valor que
257 garante uma melhor aproximação mais concreta da hipótese.

258 Para 99%, temos o valor de alfa igual a 0,01, logo temos:

$$t(\alpha, n - 1) \Rightarrow t(0.01, 10 - 1) = \pm 3, 250 \quad (4)$$

259 A partir disso, devemos calcular o desvio padrão amostral:

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^N (s_i - \mu_d)^2}{N - 1}} \Rightarrow \sigma_d = 0.057 \quad (5)$$

260 Por fim, temos a seguinte equação para calcular a hipótese:

$$t_{n-1} = \frac{\mu_d}{\frac{\sigma_d}{\sqrt{n}}} \Rightarrow t_{n-1} = 10.50 \quad (6)$$

261 Sendo assim, temos que 10,50 é maior que o limite máximo de rejeição, dado
262 pelo ponto 3,250 da distribuição normal. Portanto, rejeitamos a primeira hipótese
263 (H0). Ou seja, para uma validação cruzada com dez subconjuntos e repetições os
264 classificadores Naive Bayes e Árvore de Decisão não tem performances equiva-
265 lentes.

266 6. Discussão dos resultados

267 Com os resultados que foram obtidos pode-se inferir que os algoritmos de
268 Naive Bayes e a Árvore de Decisão, apesar de eficientes, são muito dependentes
269 da organização da base de dados.

270 Com uma organização simples o desempenho dos dois algoritmos mostrou-
271 se mais impreciso, a Árvore de Decisão com uma taxa de acerto por volta de
272 quarenta por cento e o classificador Naive Bayes com uma taxa de acerto ainda
273 menor, cerca de trinta e cinco por cento.

274 Entretanto ao aplicarmos a validação cruzada, ambas as taxas tiveram um au-
275 mento significativo em suas referidas performances. A árvore de decisão teve um
276 aumento de cerca de vinte por cento, enquanto o Naive Bayes teve um crescimento
277 de dez por cento em sua taxa de acerto.

278 Quanto ao tamanho da divisão dos subconjuntos da validação cruzada mostrou-
279 se pouco relevante, o aumento ou diminuição da taxa de acerto ficaram em volta
280 de dois por cento para mais ou para menos.

7. Conclusão

O presente artigo fez uma análise de dois algoritmos de classificação de dados em aprendizagem de máquina, o algoritmo da Árvore de Decisão e o algoritmo de Naive Bayes. Ambos mostraram-se capazes de inferir classificações a dados a partir de uma base de treino.

Entretanto os testes e análises feitos revelaram uma íntima relação entre o poder de classificação e a maneira de estruturação e uso dessa base de dados. De forma que quanto mais tratados fossem os dados de treino e o seu uso, mais eficiente seria o classificador.

Outro ponto importante de se destacar foi o melhor desempenho do algoritmo da árvore de decisão na classificação de linguagem natural, chegando a ter uma disparidade de dez por cento na taxa de acerto da classificação, se comparado ao algoritmo de Naive Bayes.

As dificuldades encontradas no estudo foram decorrentes da adaptação dos dados em texto para que eles fossem aceitos pelas funções e métodos da biblioteca Sickit-Learn.

Por fim, o presente estudo conclui que cada um dos algoritmos tem sua respectiva peculiaridade, na árvore de decisão destacamos a possibilidade de obter as regras de decisão ao fim da classificação, enquanto no Naive Bayes destacamos a oportunidade de conseguir a probabilidade de cada classificação. Assim mostrando que os algoritmos de aprendizagem de máquina são versáteis e plurais, permitindo ao programador desfrutar de características específicas de um ou de outro para moldar a solução do seu problema.

Referências

- [1] P. Prado, A. Monteiro, Pattern recognition algorithms 5 (2008).
- [2] G. Nuti, L. A. J. Rugama, A.-I. Cross, A bayesian decision tree algorithm, stat 1050 (2019) 11.
- [3] R. Pandya, J. Pandya, C5. 0 algorithm to improved decision tree with feature selection and reduced error pruning, International Journal of Computer Applications 117 (2015) 18–21.
- [4] S. Xu, Bayesian naïve bayes classifiers to text classification, Journal of Information Science 44 (2018) 48–59.

- 313 [5] S. Vijayarani, S. Dhayanand, Liver disease prediction using svm and naïve
314 bayes algorithms, International Journal of Science, Engineering and Techno-
315 logy Research (IJSETR) 4 (2015) 816–820.
- 316 [6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel,
317 M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,
318 D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine
319 learning in Python, Journal of Machine Learning Research 12 (2011) 2825–
320 2830.