Implementação Naïve Bayes: Python

Mateus Tarcinalli Machado

Introdução ao Aprendizado de Máquina
Prof. Dr. José Augusto Baranauskas

17 de maio de 2020

Apresentação

Introdução

Trabalho para a disciplina Introdução ao Aprendizado de Máquina:

Implementação Naïve Bayes: Python

Motivação:

ALMANSA, Luciana Farina et al. Information learned from monitoring microblogs during the 2014 seasonal flu vaccination in Brazil. In: 2014 IEEE 10th International Conference on e-Science. IEEE, 2014. p. 65-66.

Módulos Utilizados

```
# carregar dados
import csv
# validação cruzada
import random
# gráficos
import numpy as np
from mlxtend.plotting import plot_decision_regions
# ocultar avisos jupyter
import warnings
warnings.filterwarnings('ignore')
```

Classe

```
class Classificador:
    dados = []
    modelo = {}
    def carregar(self, arquivo, delimiter, quoteChar,
                 ignoraLinha1, ignoraColuna1)
    def carregarTexto(self, arquivo)
    def treinar(self)
    def g(self, x, media, desvio)
    def predizer(self, tupla, debug=False)
    def predict(self, tuplas)
    def graficoFronteira(self)
```

Carregar

```
def carregar(self, arquivo, delimiter, quoteChar,
                    ignoraLinha1, ignoraColuna1):
    self.dados = []
    with open(arquivo) as csv_file:
        csv_reader = csv.reader(csv_file,
                                delimiter=delimiter,
                                quoting=csv.QUOTE_NONNUMERIC)
        if ignoraLinha1:
            next(csv_reader)
        for i, row in enumerate(csv_reader):
            if type(row[-1]) != str:
                row[-1] = str(int(row[-1]))
            if ignoraColuna1:
                self.dados.append(tuple(row[1:]))
            else:
                self.dados.append(tuple(row))
```

Treinar I

```
def treinar(self):
    total = len(self.dados)
    classes = list(set([x[-1] for x in self.dados]))
    combdata = {}
    pdata = {}
    # contagem dos dados
    for row in self.dados:
        classe = (row[-1])
        if classe in combdata:
            combdata[classe] += 1
        else:
            combdata[classe] = 1
```

Treinar II

```
for i, d in enumerate(row[:-1]):
    # variavel continua
    if type(d) != str:
        ind = 'i'+str(i)
        if ind in combdata:
            combdata[ind].append(float(d))
        else:
            combdata[ind] = [ float(d) ]
        key = (ind,classe)
        if key in combdata:
            combdata[key].append(float(d))
        else:
            combdata[key] = [float(d)]
```

Treinar III

```
# variavel nominal
else:
    if d in combdata:
        combdata[d] += 1
    else:
        combdata[d] = 1
    key = (d,classe)
    if key in combdata:
        combdata[key] += 1
    else:
        combdata[key] = 1
```

Treinar IV

```
# calculando probabilidades
self.modelo = {}
for c in combdata:
    # atributo + classe
    if type(c) == tuple:
        # variável continua
        if type(combdata[c]) != list:
            self.modelo[c] = combdata[c] / combdata[c[1]]
        else:
            media =  sum(combdata[c])/len(combdata[c])
            desvio=(sum([(x-media)**2 for x in combdata[c]]
                    )/(len(combdata[c])-1))**0.5
            self.modelo[c]={'media':media,'desvio':desvio}
```

Treinar V

```
        # somente classe
        else:
            if type(combdata[c]) != list:
                self.modelo[c] = combdata[c] / total
            else:
                media =  sum(combdata[c])/len(combdata[c])
                desvio=(sum([(x-media)**2 for x in combdata[c]]
                        )/(len(combdata[c])-1))**0.5
                self.modelo[c] = {
                    'media' : media,
                    'desvio' : desvio
                }
    self.modelo['classes'] = classes
```

```
def g(self, x, media, desvio):
    p1 = 1 / ( (2*3.141592653589793)**0.5 * desvio  )
    p2=2.718281828459045**(-((x-media)**2)/(2*desvio**2))
    return p1 * p2
```

Predizer I

```
def predizer(self, tupla, debug=False):
    probClasse = {}
    valClasse = 0
    maxClasse = ''
    for classe in self.modelo['classes']:
        numerador = 1
        denominador = 1
        for i, v in enumerate(tupla):
```

Predizer II

```
# atributo nominal
if type(v) == str:
    key = (v, classe)
    if key in self.modelo:
        numerador = numerador * self.modelo[key]
    else:
        numerador = numerador * 0.001
    if v in self.modelo:
        denominador = denominador * self.modelo[v]
    else:
        denominador = denominador * 0.001
```

Predizer III

```
        # atributo continuo
        else:
            v = float(v)
            key1 = ('i'+str(i),classe)
            key2 = 'i'+str(i)
            if key1 in self.modelo:
                numerador=numerador*self.g(v,
                                self.modelo[key1]['media'],
                                self.modelo[key1]['desvio'])
            denominador=denominador*self.g(v,
                                self.modelo[key2]['media'],
                                self.modelo[key2]['desvio'])
    numerador = numerador * self.modelo[classe]
    probClasse[classe] = numerador / denominador
```

Predizer IV

```
        if debug:
            print(classe,probClasse[classe])
        if probClasse[classe] > valClasse:
            valClasse = probClasse[classe]
            maxClasse = classe
    return(maxClasse)
```

Gráfico I

```
def predict(self, tuplas):
    res = []
    for t in tuplas:
        res.append(int(self.predizer(t)))
    return(np.array(res))

def graficoFronteira(self):
    X=np.array([[float(x[0]),float(x[1])]
                    for x in self.dados])
    y=np.array([int(float(x[2])) for x in self.dados])
    plot_decision_regions(X, y, clf=self, legend=2)
```

Validação Cruzada I

```
def validacaoCruzada(clf, porcentagem=0.7):
    dados = clf.dados
    random.shuffle(dados)
    dadosTreinamento = dados[:int(len(dados)*porcentagem)]
    dadosTestes = dados[int(len(dados)*porcentagem):]
    clf2 = Classificador()
    clf2.dados = dadosTreinamento
    clf2.treinar()
    ok = 0
    for tupla in dadosTestes:
        tupl = tupla[:-1]
        pred = clf2.predizer(tupl, debug=False)
        if pred == tupla[-1]:
            ok += 1
```

Validação Cruzada II

```
print('Conj Teste:',len(dadosTestes))
print('Acertos   :',ok)
print('Acurácia  :',ok/len(dadosTestes)*100,'%')
```

Dataset Tenis I

Aparência	Temperatura	Umidade	Ventando	Jogar
sol	85	85	falso	não
sol	80	90	verdadeiro	não
nublado	83	86	falso	sim
chuva	70	96	falso	sim
chuva	68	80	falso	sim
chuva	65	70	verdadeiro	não
nublado	64	65	verdadeiro	sim
sol	72	95	falso	não
sol	69	70	falso	sim

Validação Cruzada I

```
tenis2 = Classificador()
tenis2.carregar('data/tenis2.csv')
tenis2.treinar()
tenis2.predizer(['sol', 83, 73, 'verdadeiro'], debug=True)

sim 0.2984141388375451
não 0.6661952433370804

'não'
```

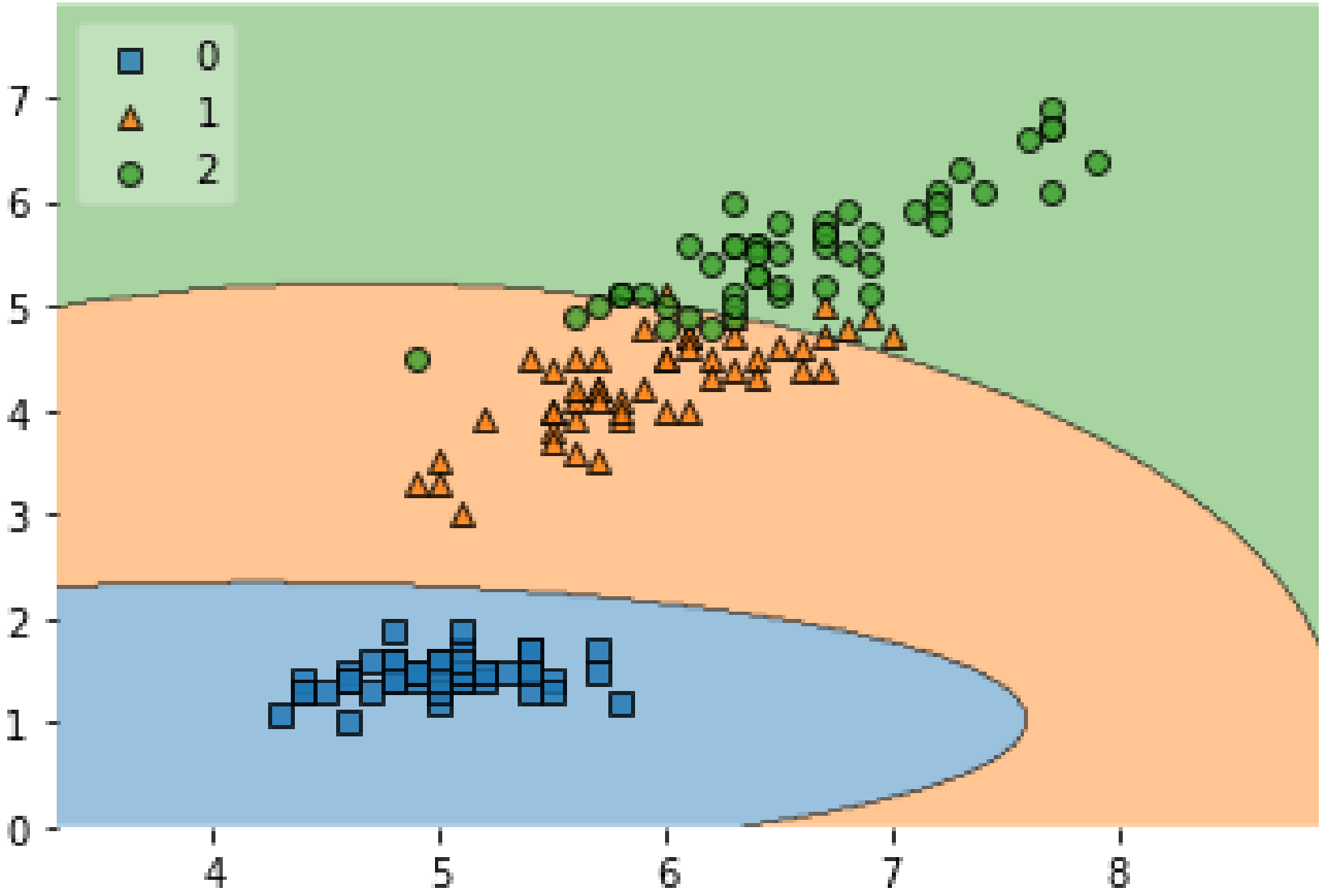
tenis2.modelo I

```
{'não': 0.3571428571428715,
 'sol': 0.3571428571428715,
 ('sol', 'não'): 0.6,
 'i1': {'media': 73.57142857142857,
        'desvio': 6.571667458629 75},
 ('i1', 'não'): {'media': 74.6,
                 'desvio': 7.893034904268446},
 'i2': {'media': 81.64285714285714,
        'desvio': 10.285218242007035},
 ('i2', 'não'): {'media': 86.2,
                 'desvio': 9.731392500562292},
 'falso': 0.5714285714285714,
 ('falso', 'não'): 0.4,
 'verdadeiro': 0.4285714285714285 5,
 ('verdadeiro', 'não'): 0.6,
```

tenis2.modelo II

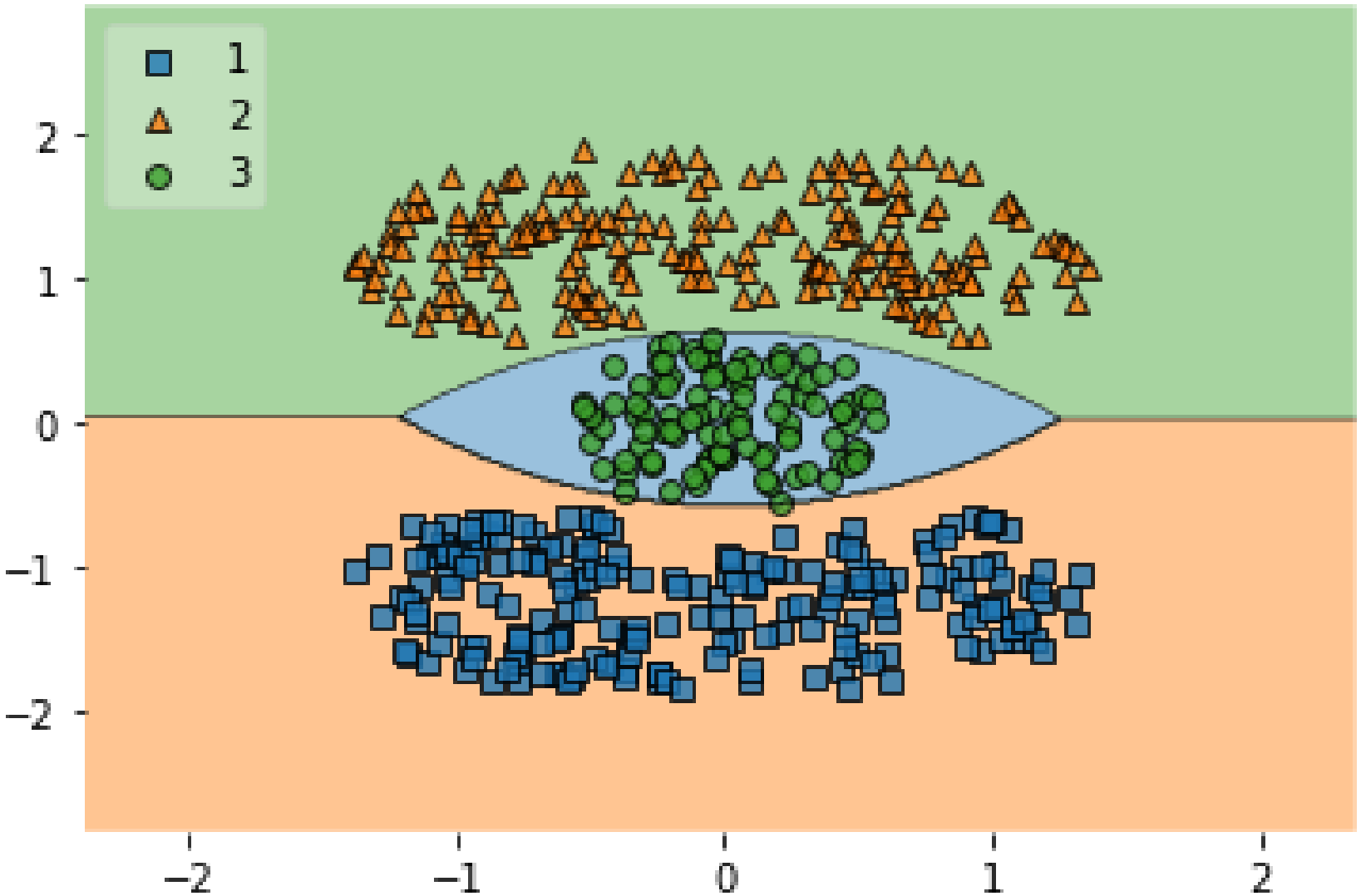
```
'sim': 0.6428571428571429,
'nublado': 0.2857142857142857,
('nublado', 'sim'): 0.4444444444444444,
('i1', 'sim'): {'media': 73.0,
                'desvio': 6.164414002968976},
('i2', 'sim'): {'media': 79.11111111111111,
                'desvio': 10.215728613814635},
('falso', 'sim'): 0.6666666666666666,
'chuva': 0.3571428571428571,
('chuva', 'sim'): 0.3333333333333333,
('chuva', 'não'): 0.4,
('verdadeiro', 'sim'): 0.3333333333333333,
('sol', 'sim'): 0.2222222222222222,
'classes': ['sim', 'não']}
```

iris: 2 dimensões



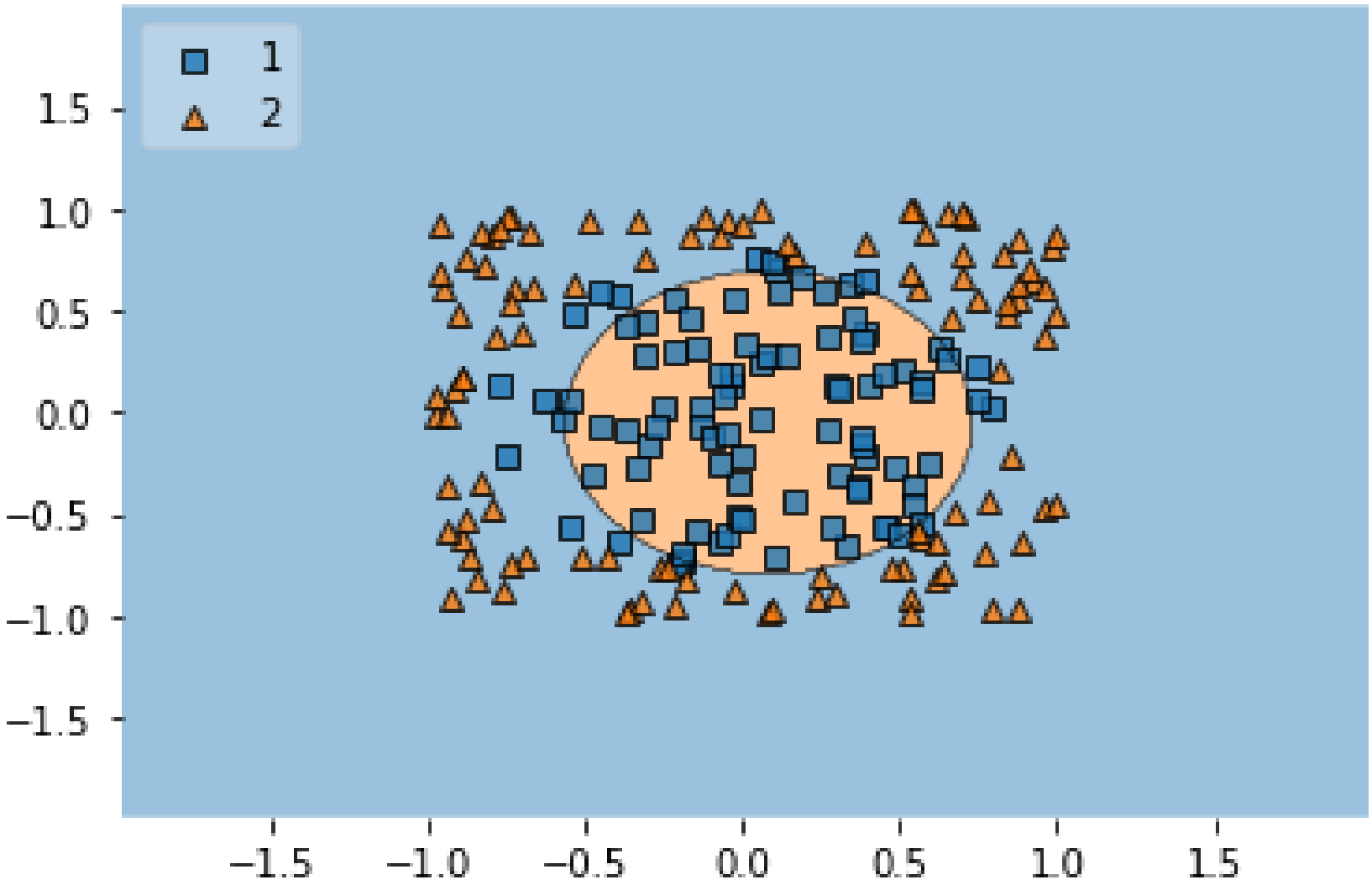
Acurácia : 88.89 %

cassini



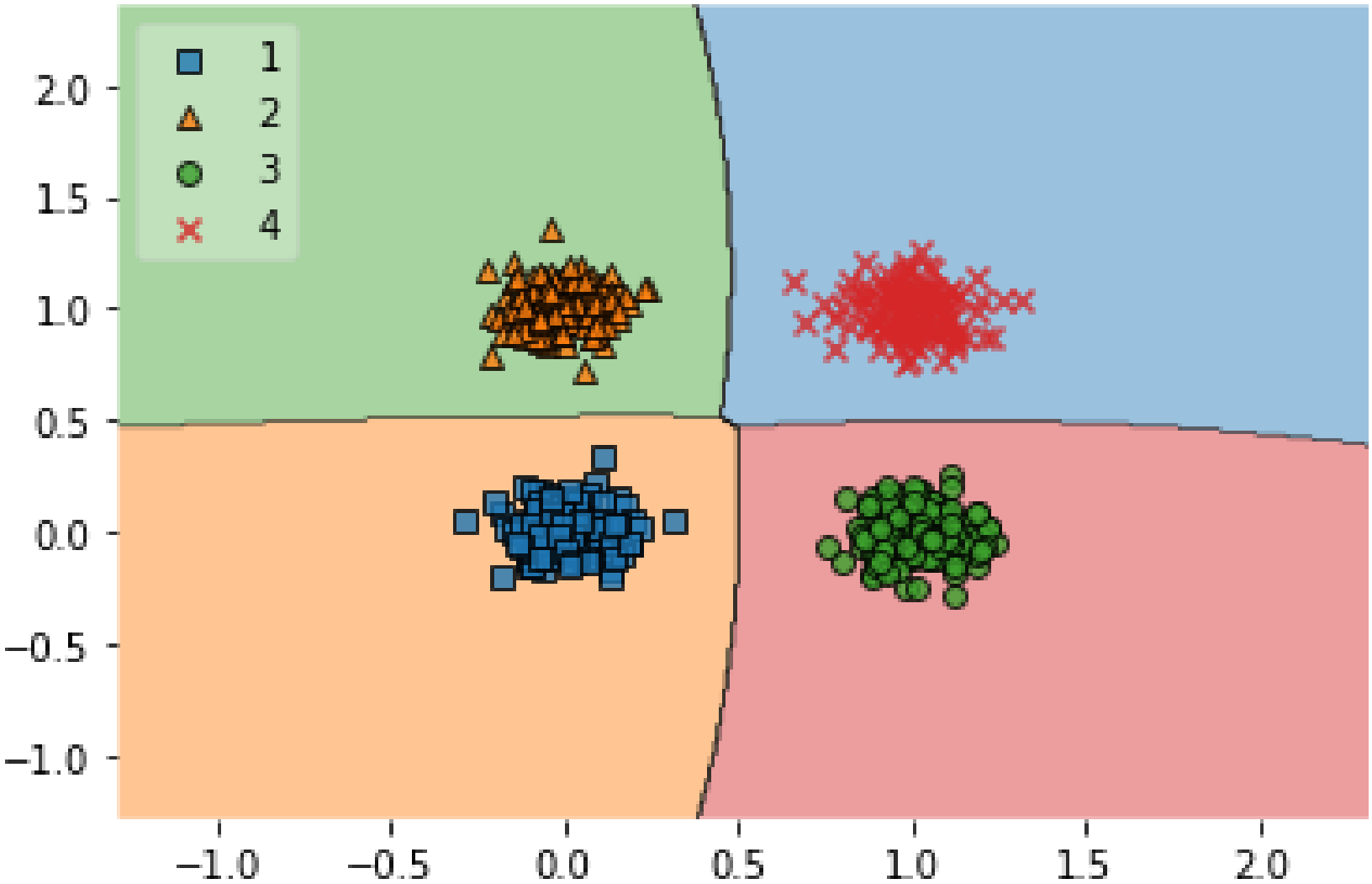
Acurácia : 100.00 %

circle



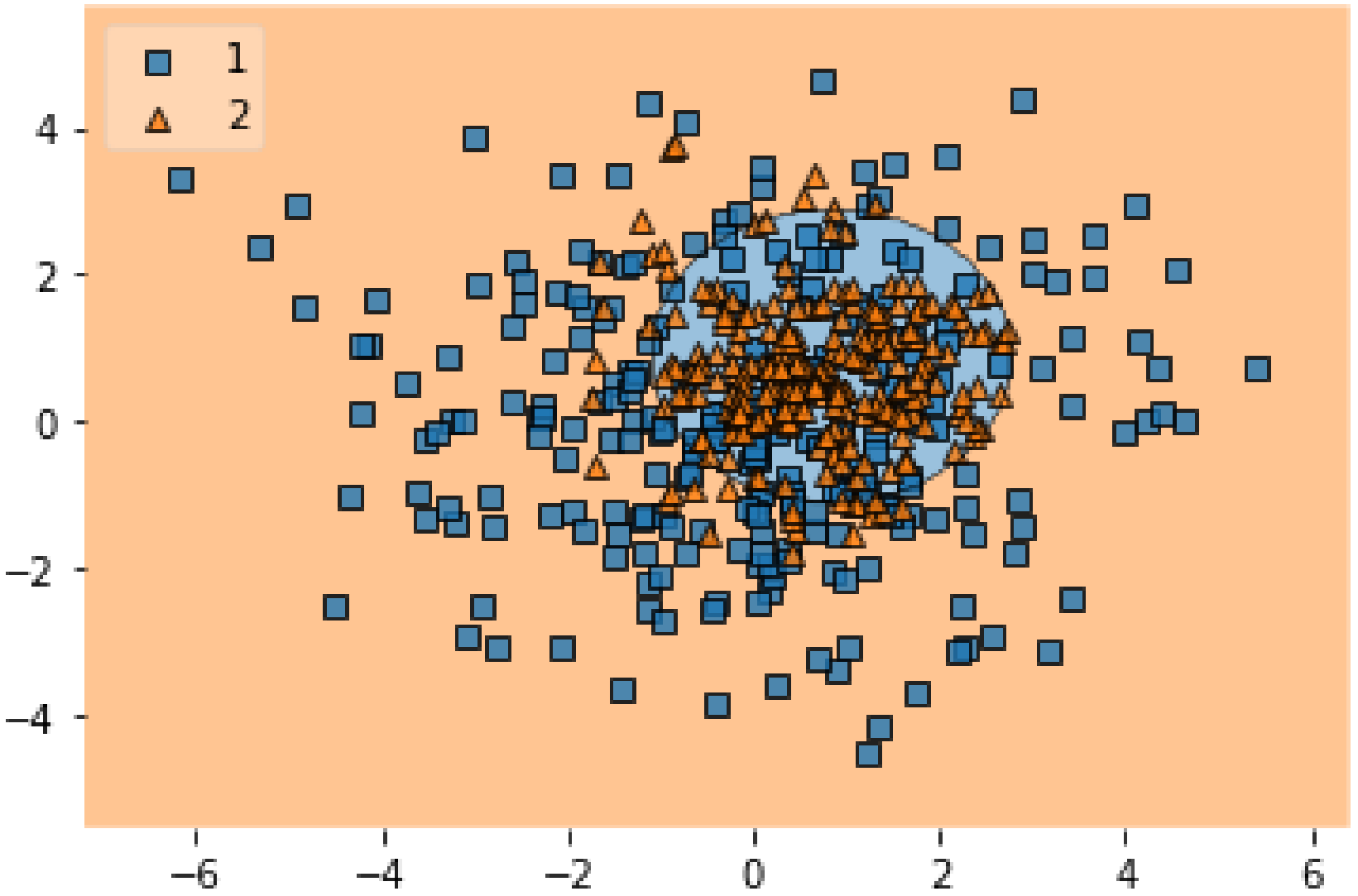
Acurácia : 95.00 %

hypercube



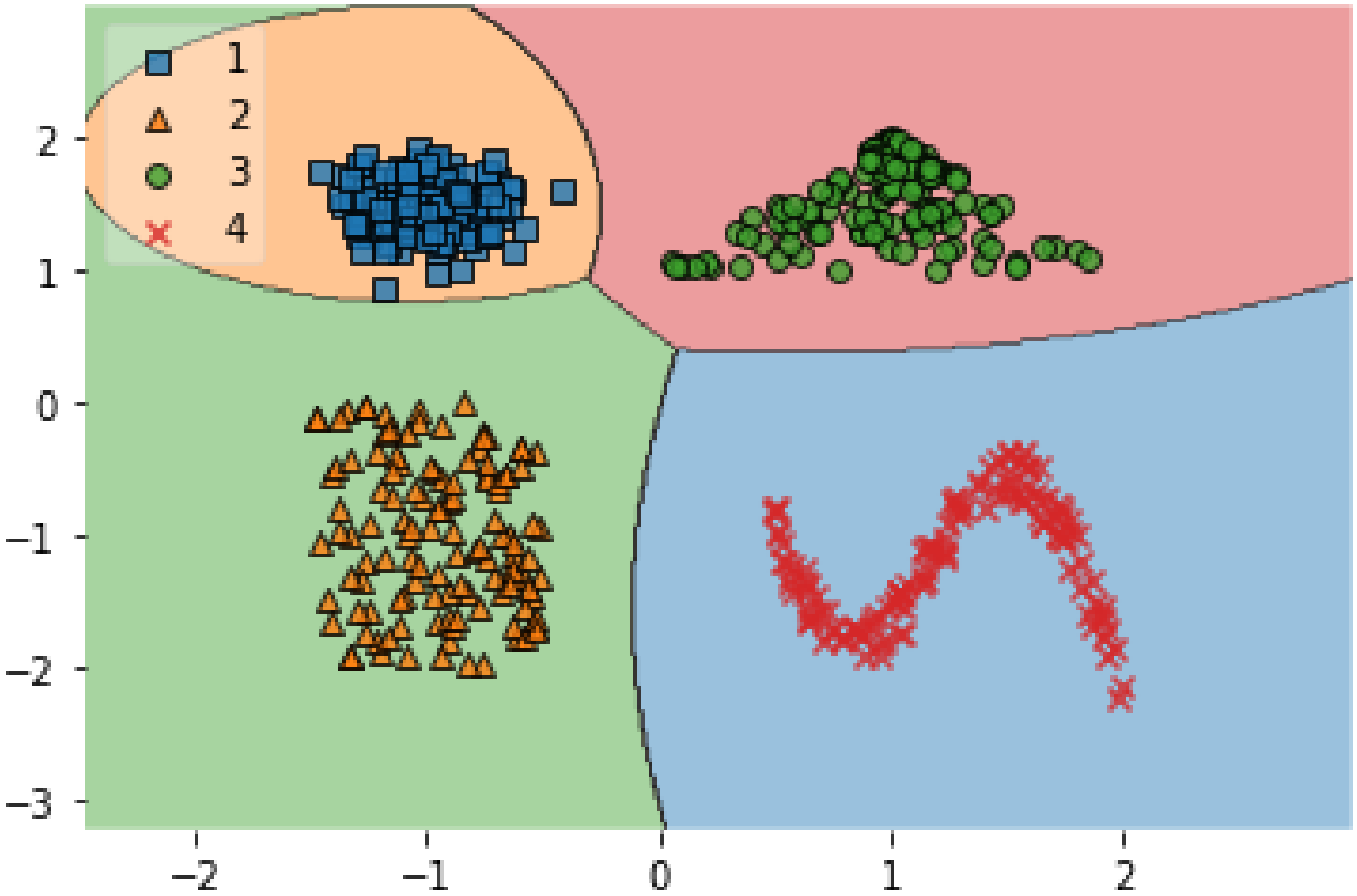
Acurácia : 100.00 %

ringnorm



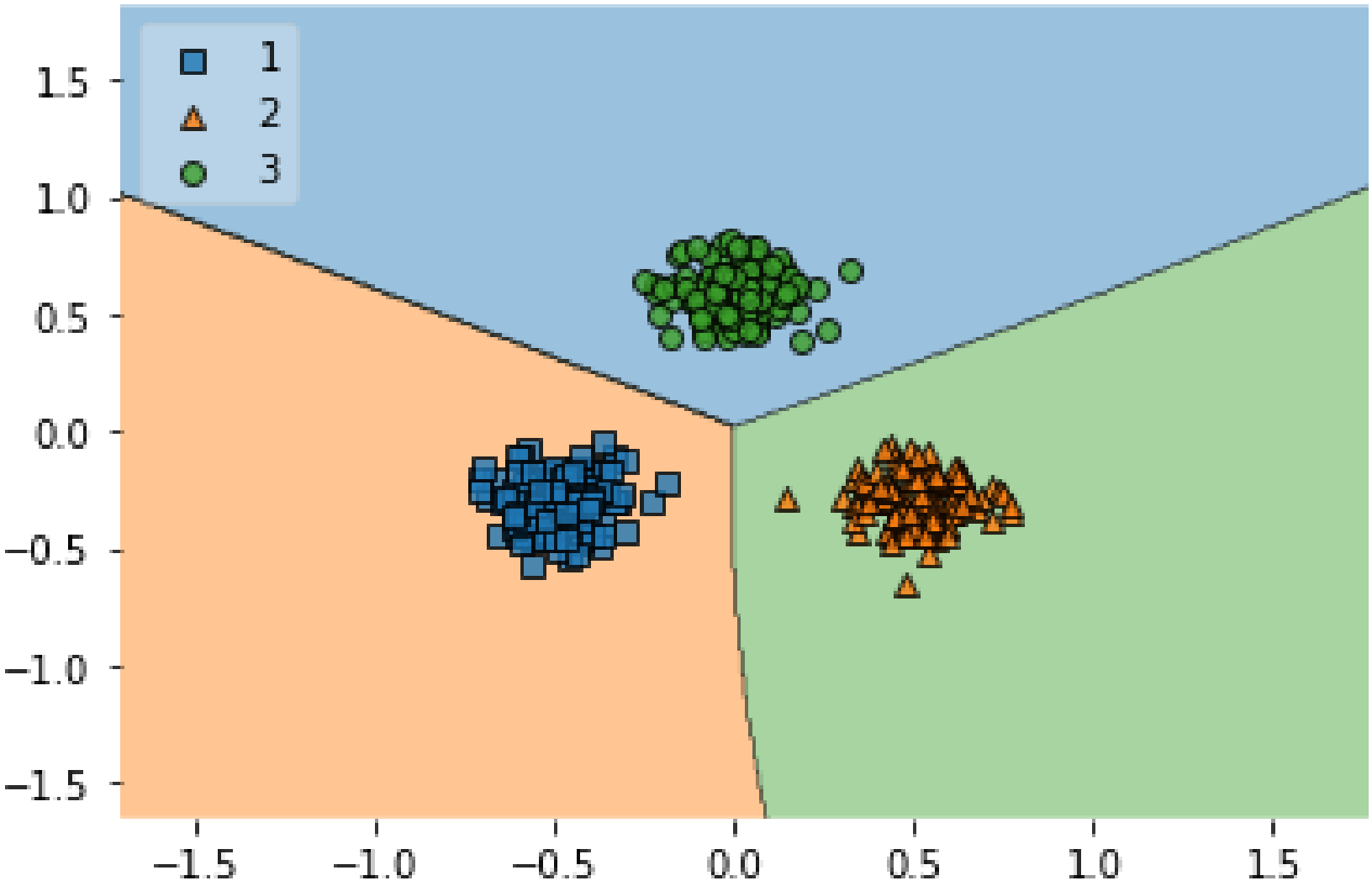
Acurácia : 80.67 %

shapes



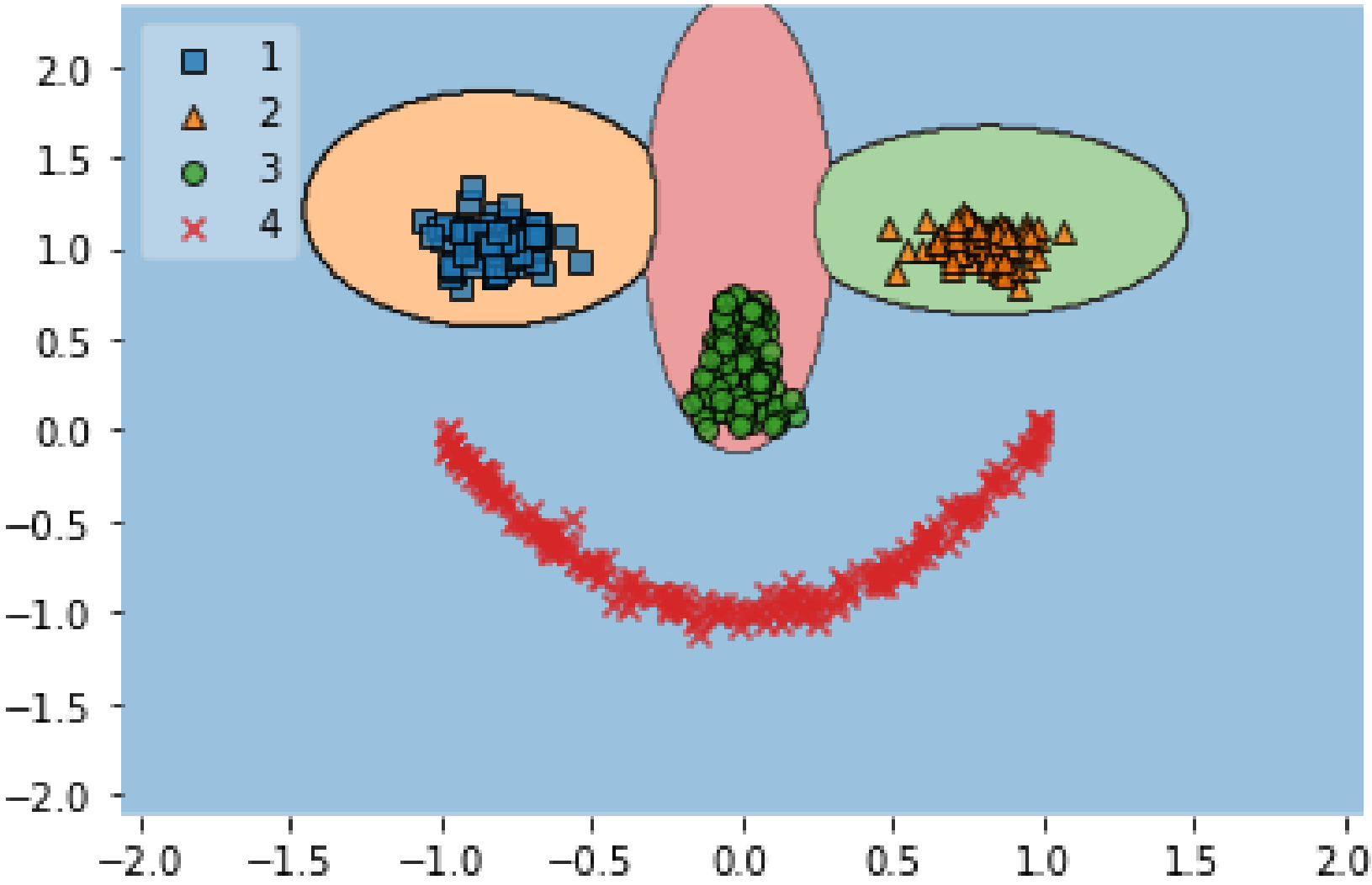
Acurácia : 100.00 %

simplex



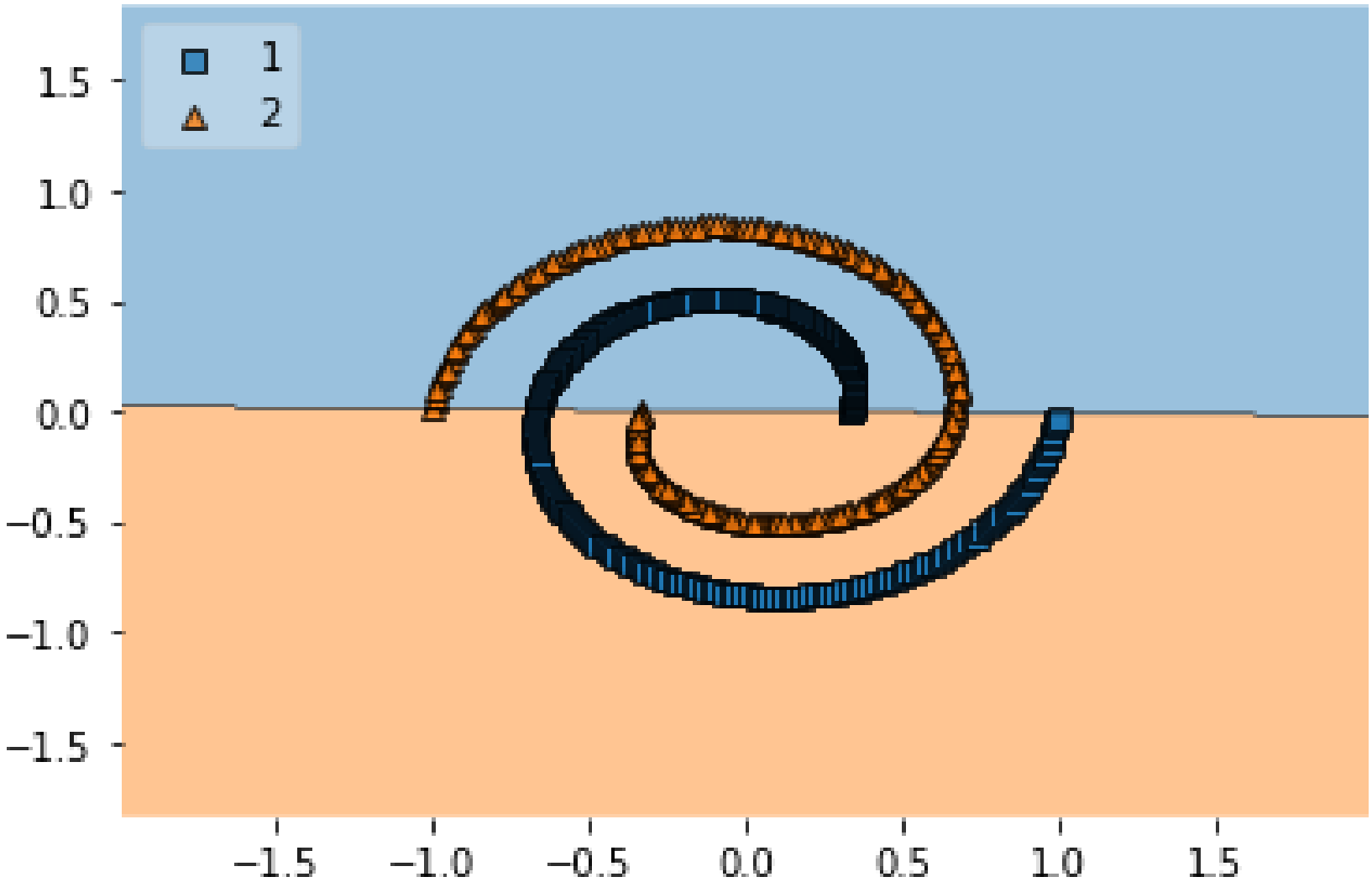
Acurácia : 100.00 %

smiley



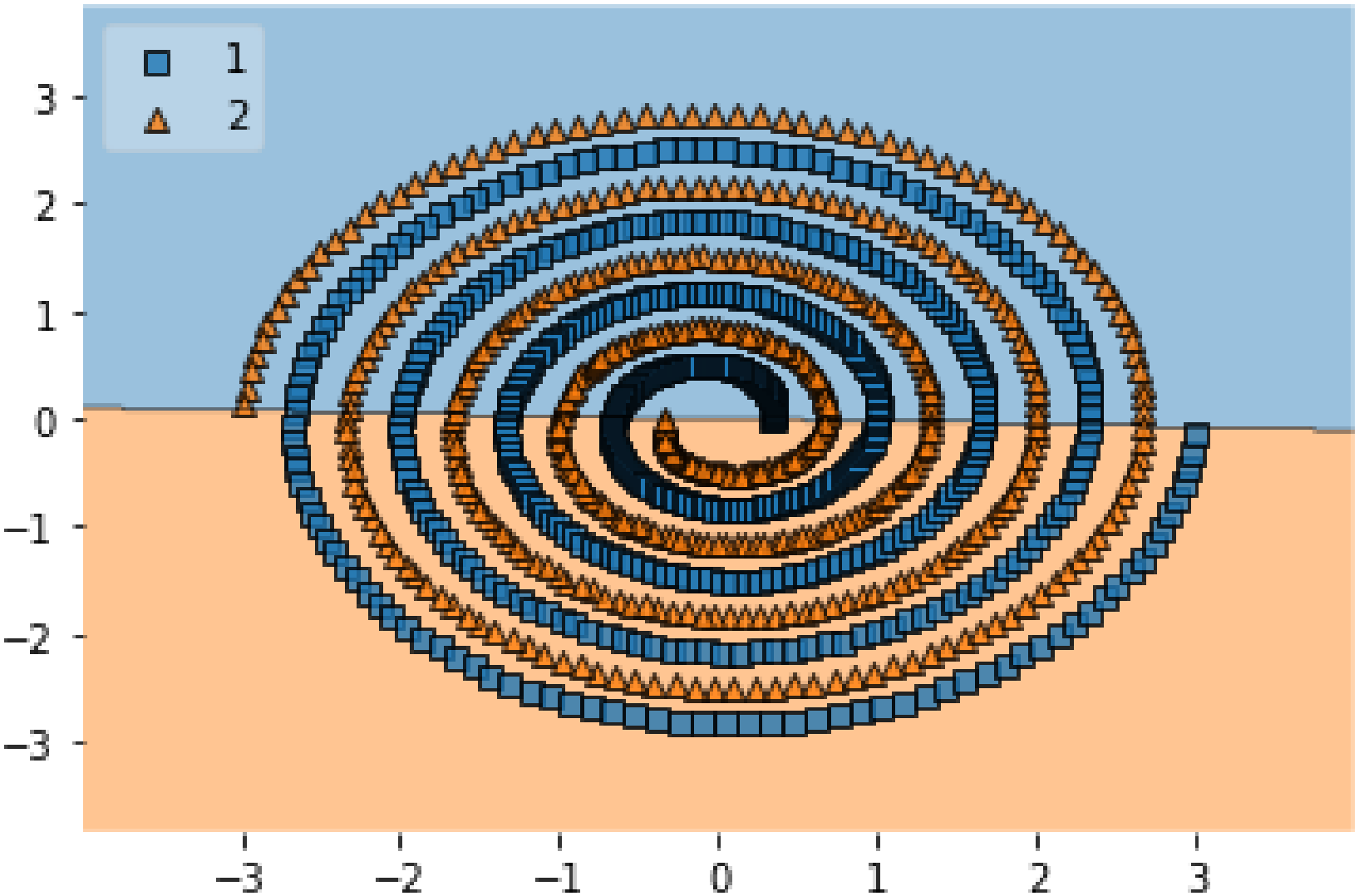
Acurácia : 100.00 %

spirals



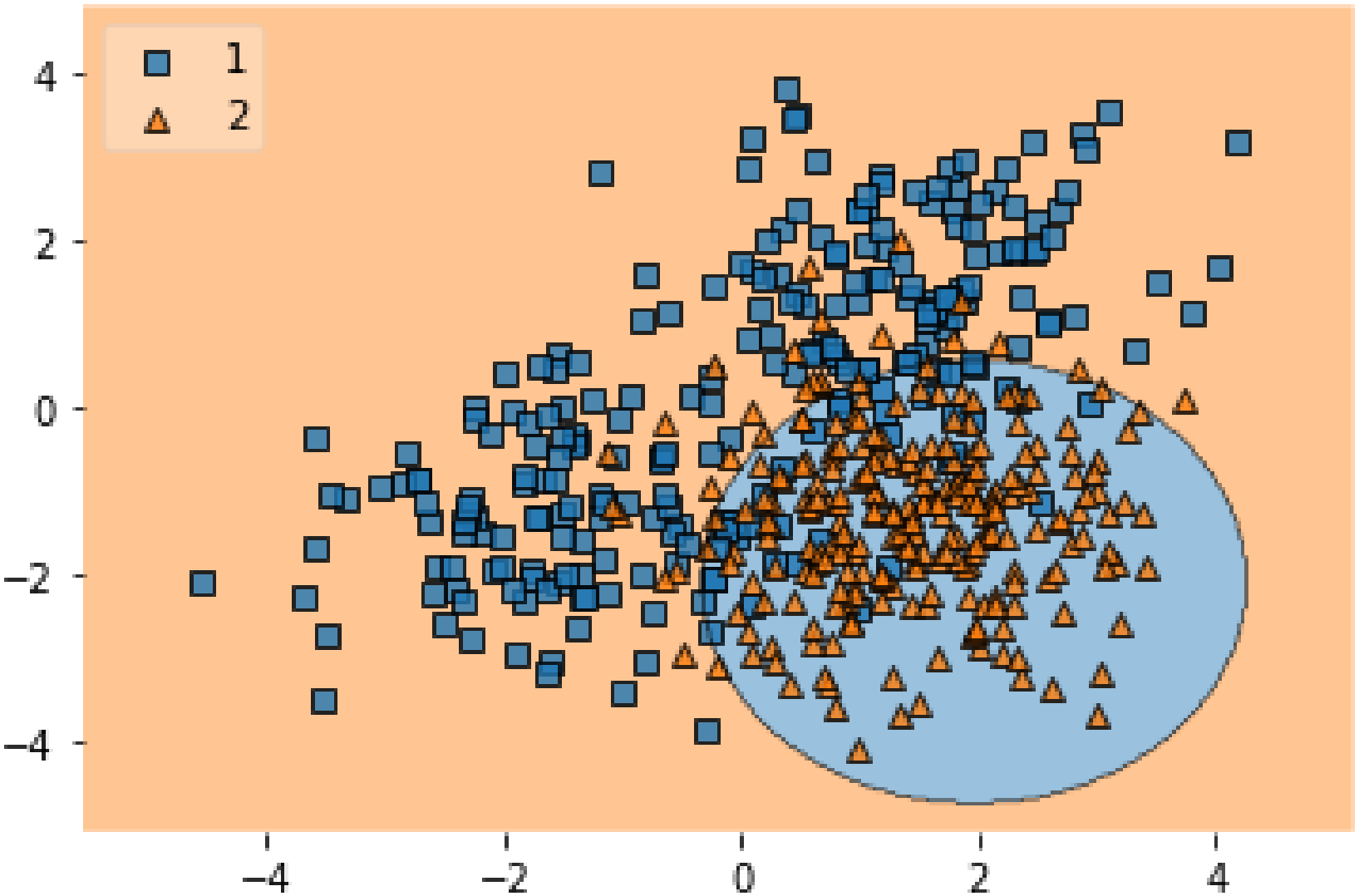
Acurácia : 46.00 %

spirals



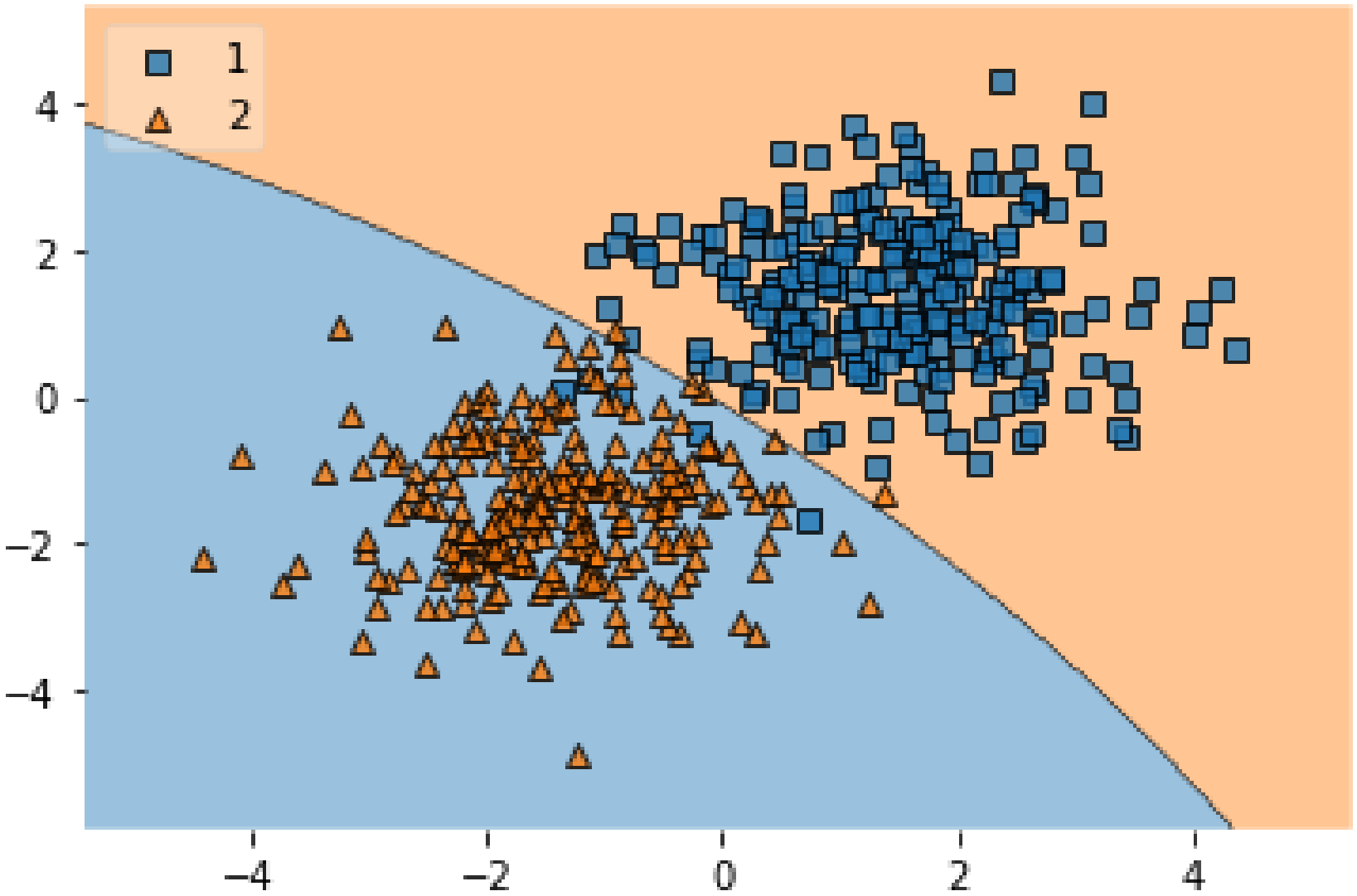
Acurácia : 46.67 %

threenorm



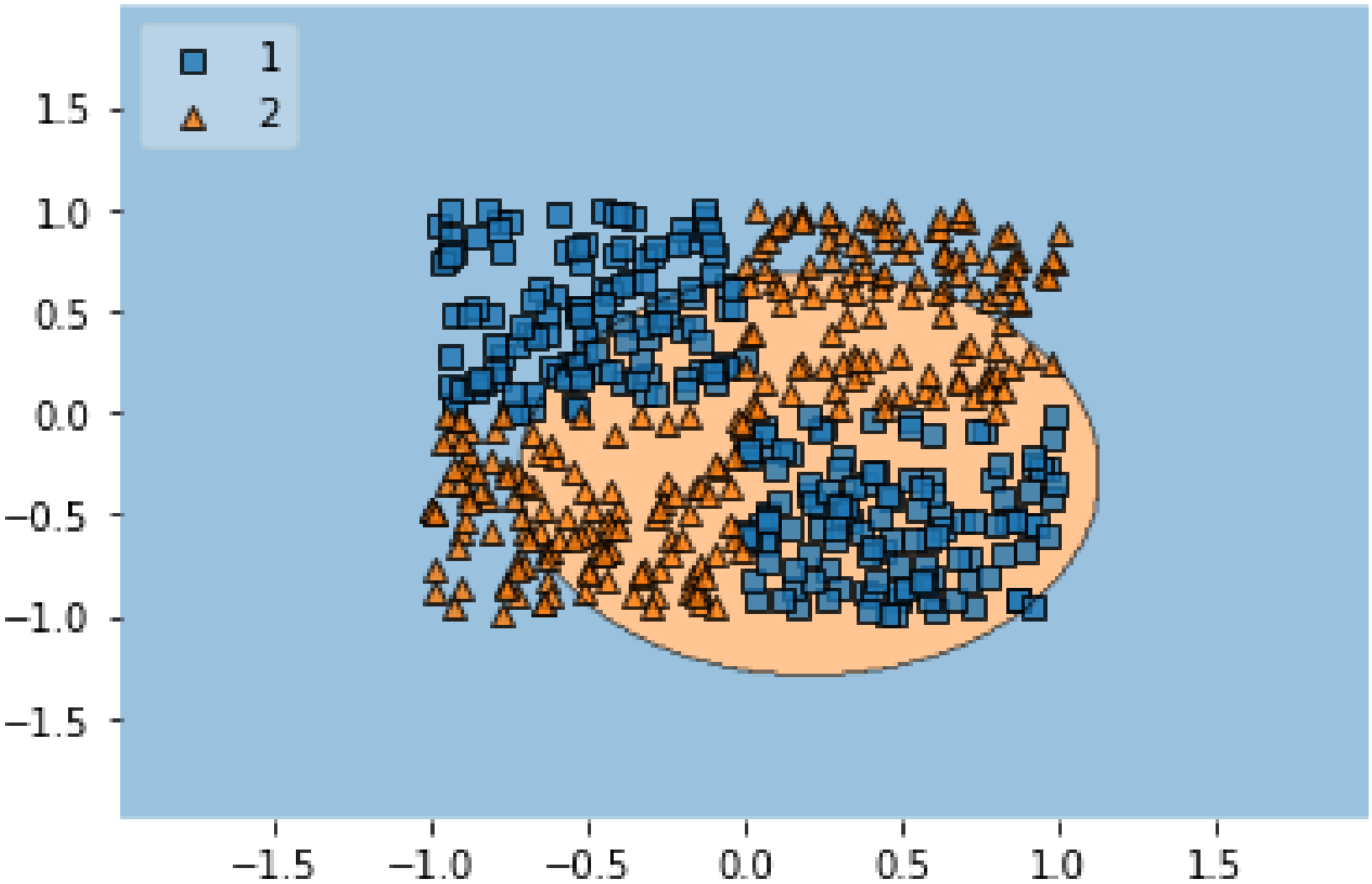
Acurácia : 90.00 %

twonorm



Acurácia : 98.67 %

xor d=2



Acurácia : 52.00 %

Carregar Texto I

```
def carregarTexto(self, arquivo):
    dataset = []
    palavras = []
    with open(arquivo) as file:
        for line in file:
            frase = ''
            for ch in line[:-2].lower():
                if ch.isalpha():
                    frase += ch
                else:
                    frase += ' '
            frase = [ x for x in frase.strip().split(' ')
                        if x != '' and len(x) > 3 ]
            classificacao = int(line[-2])
            dataset.append((frase,classificacao))
```

Carregar Texto II

```
            palavras += frase
    palavras = list(set(palavras))
    dataset2 = []
    for x in dataset:
        aux = []
        for i,p in enumerate(palavras):
            if p in x[0]:
                aux.append('ps'+str(i))
            else:
                aux.append('pn'+str(i))
        aux.append(str(x[1]))
        dataset2.append(aux)
    self.dados = dataset2
```

Dataset Amazon

Frase	Classe
Great for the jawbone.	1
The mic is great.	1
If you are Razr owner...you must have this!	1
What a waste of money and time!.	0

```
print(len(amazon.dados))
# 1000
print(len(amazon.dados[0]))
# 1629
print(amazon.dados[0][:10])
# ['pn0', 'pn1', 'pn2', 'pn3', 'pn4', 'pn5', 'pn6', 'pn7', 'pn8', 'pn9']
```

```
Conj Teste: 300
Acertos : 233
Acurácia : 77.66666666666666 %
```

Dataset IMDB

Frase	Classe
Very little music or anything to speak of.	0
Fans of the genre will be in heaven.	1
Lange had become a great actress.	1
It looked like a wonderful story.	1
Wasted two hours.	0
A bit predictable.	0

Conj Teste: 300
Acertos : 224
Acurácia : 74.66666666666667 %

Dataset Yelp

Frase	Classe
Wow... Loved this place.	1
Crust is not good.	0
Not tasty and the texture was just nasty.	0
Now I am getting angry and I want my damn pho.	0
Honeslty it didn't taste THAT fresh.)	0
The fries were great too.	1
A great touch.	1

Conj Teste: 300
Acertos : 229
Acurácia : 76.33333333333333 %

Perguntas?

Muito Obrigado!