ДОДАТОК А

Вихідний код для експериментальних досліджень

```
from sklearn import svm
from sklearn import datasets
from matplotlib import pyplot as plt
import numpy as np
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.graph_objs as go

init_notebook_mode(connected=True)
from ipywidgets import interact, interactive, fixed
from IPython.core.display import HTML
from IPython.display import display, clear_output
from plotly.widgets import GraphWidget

g = GraphWidget('https://plot.ly/~danya.kosmin/6')
circles = datasets.make_circles(1500, .1, .03)
colors = np.abs(circles[1]-0.2)

full = [(i - 50)/40 for i in range(0, 100)]
'''for gamma in [0.1, 0.2, 0.5, 1, 1.5, 2, 3, 5, 7, 10, 15, 20, 30, 50, 70, 100]:
    c = []
    SVM = svm.SVC(gamma=gamma, probability=True, C=1)
    SVM.fit(circles[0],circles[1])
    for k in full:
        for i in full:
            c.append([k, i])
    c = np.asarray(c)
    pred = SVM.predict(c)
'''
class z_data:
```

```
    def __init__(self):
        self.G = 1
        self.C = 1
        full = [(i - 50)/40 for i in range(0, 100)]
        SVM = svm.SVC( probability=True, kernel='rbf')
        SVM.fit(circles[0], circles[1])
        #pred = SVM.predict(c)
        full = [(i - 50)/30 for i in range(0, 100)]
        c = []
        res = []
        for k in full:
            l =[]
            for i in full:
                c.append([k, i])
                pro = SVM.predict_proba(np.asarray([k, i]).reshape(1, -1))[:,0]
                l.append(pro)


            res.append(l)
        self.z = np.asarray(res).reshape(100,100)


    def replot(self):
            g.restyle({'z': [self.z], 'colorscale': 'Viridis'})


    def on_z_change(self, name):
        full = [(i - 50)/40 for i in range(0, 100)]
        new_value = name['new']
        self.C = new_value
        SVM = svm.SVC(gamma=self.G, C=new_value, probability=True
,kernel='rbf')
        SVM.fit(X, y)
```

```
        c = []
        res = []
        full = [(i - 50)/30 for i in range(0, 100)]
        res = []
        for k in full:
            l =[]
            for i in full:
                c.append([k, i])
                #pro = SVM.predict_proba(np.asarray([k, i]).reshape(1, -1))[:,0]
                pro = SVM.predict(np.asarray([k, i]).reshape(1, -1))
                l.append(pro)

            res.append(l)
        self.z = np.asarray(res).reshape(100,100)
        self.replot()

    def on_z_change2(self, name):
        full = [(i - 50)/40 for i in range(0, 100)]
        new_value = name['new']
        self.G = new_value
        SVM = svm.SVC(gamma=new_value, C=self.C, probability=True
,kernel='rbf')
        SVM.fit(X, y)
        c = []
        res = []
        full = [(i - 50)/30 for i in range(0, 100)]
        res = []
        for k in full:
            l =[]
            for i in full:
```

```
            c.append([k, i])
            #pro = SVM.predict_proba(np.asarray([k, i]).reshape(1, -1))[:,0]
            pro = SVM.predict(np.asarray([k, i]).reshape(1, -1))
            l.append(pro)

        res.append(l)
    self.z = np.asarray(res).reshape(100,100)
    self.replot()



z_slider = widgets.FloatSlider(min=0.00001,max=30,value=0.1,step=0.005,
continuous_update=False)
z_slider.description = 'C'
z_slider.value = 1
z_slider2 = widgets.FloatSlider(min=0.00001,max=30,value=0.1,step=0.005,
continuous_update=False)
z_slider2.description = 'Gamma'
z_slider2.value = 1
z_state = z_data()

z_slider2.observe(z_state.on_z_change2, 'value')
z_slider.observe(z_state.on_z_change, 'value')
g = GraphWidget('https://plot.ly/~danya.kosmin/6')
circles = datasets.make_circles(1500, .1, .03)
colors = np.abs(circles[1]-0.2)
#SVM = svm.SVC(gamma=1, probability=True, C=1)
#SVM.fit(circles[0],circles[1])
#plt.scatter(circles[0][:,0], circles[0][:,1],  c=colors)
```

```
full = [(i - 50)/40 for i in range(0, 100)]
'''for gamma in [0.1, 0.2, 0.5, 1, 1.5, 2, 3, 5, 7, 10, 15, 20, 30, 50, 70, 100]:
    c = []
    SVM = svm.SVC(gamma=gamma, probability=True, C=1)
    SVM.fit(circles[0],circles[1])
    for k in full:
        for i in full:
            c.append([k, i])
    c = np.asarray(c)
    pred = SVM.predict(c)
'''
class z_data:


    def __init__(self):
        full = [(i - 50)/40 for i in range(0, 100)]
        SVM = svm.SVC( probability=True, kernel='rbf')
        SVM.fit(circles[0], circles[1])
        #pred = SVM.predict(c)
        full = [(i - 50)/30 for i in range(0, 100)]
        c = []
        res = []


       for k in full:
            l =[]
            for i in full:
                c.append([k, i])
                pro = SVM.predict_proba(np.asarray([k, i]).reshape(1, -1))[:,0]
                l.append(pro)
            res.append(l)
        self.z = np.asarray(res).reshape(100,100)
```

```
    def replot(self):
            g.restyle({'z': [self.z], 'colorscale': 'Viridis'})

    def on_z_change(self, name):
        full = [(i - 50)/40 for i in range(0, 100)]
        new_value = name['new']
        SVM = svm.SVC(gamma=1, C=new_value, probability=True ,kernel='rbf')
        SVM.fit(X, y)
        c = []
        res = []
        full = [(i - 50)/30 for i in range(0, 100)]
        res = []
        for k in full:
            l =[]
            for i in full:
                c.append([k, i])
                #pro = SVM.predict_proba(np.asarray([k, i]).reshape(1, -1))[:,0]
                pro = SVM.predict(np.asarray([k, i]).reshape(1, -1))
                l.append(pro)

            res.append(l)
        self.z = np.asarray(res).reshape(100,100)
        self.replot()



z_slider = widgets.FloatSlider(min=0.00001,max=30,value=0.1,step=0.005,
continuous_update=False)
z_slider.description = 'C'
```

```
z_slider.value = 1

z_state = z_data()

z_slider.observe(z_state.on_z_change, 'value')

display(z_slider)

display(g)

def twospirals(n_points, noise=.5):
    """
     Returns the two spirals dataset.
    """
    n = np.sqrt(np.random.rand(n_points,1)) * 780 * (2*np.pi)/360
    d1x = -np.cos(n)*n + np.random.rand(n_points,1) * noise
    d1y = np.sin(n)*n + np.random.rand(n_points,1) * noise
    return (np.vstack((np.hstack((d1x,d1y)),np.hstack((-d1x,-d1y)))),
            np.hstack((np.zeros(n_points),np.ones(n_points))))

X, y = twospirals(1000)
X=X/13

trace1 = go.Scatter3d(x=np.power(circles[0][:, 0][circles[1]==1], 2),
                      y=np.power(circles[0][:, 1][circles[1]==1],2),
                      z=np.sqrt(2)*(circles[0][:, 0][circles[1]==1]) * (circles[0][:,
0][circles[1]==1]),
                      marker=dict(color='#B8F7D4'),

                     mode='markers')
```

```
trace2 = go.Scatter3d(x=np.power(circles[0][:, 0][circles[1]==0], 2),
                      y=np.power(circles[0][:, 1][circles[1]==0],2),
                      z=np.sqrt(2)*(circles[0][:, 0][circles[1]==0]) * (circles[0][:,
0][circles[1]==0]),

                      marker=dict(color='#7FA6EE'),
                     mode='markers')
full = [(i - 50)/30 for i in range(0, 100)]
c = []
res = []
for k in full:
   l =[]
   for i in full:
      c.append([k, i])
      pro = SVM.predict_proba(np.asarray([k, i]).reshape(1, -1))[:,0]
      l.append(pro)
   res.append(l)
res = np.asarray(res).reshape(100,100)
#c = np.asarray(c)
#proba = SVM.predict_proba(c)
#print(proba[0])

trace3 = go.Surface(#x=c[:,0],
                      #y=c[:,1],
                      z=res)
                     # marker=dict(color='#835AF1'),
                      #mode='markers')

iplot([trace3])
circles = datasets.make_circles(1500, noise=0.07, factor=.4)
```

```
SVM = svm.SVC(gamma=1, probability=True, C=1, kernel='rbf')
SVM.fit(X, y)
res = SVM.predict(X)
trace1 = go.Scatter(x = X[:,0][res==0]*4,
                    y =  X[:,1][res==0]*4,
                    mode='markers',
                   marker = dict(color='rgba(255, 44, 193, .9)'),
                   name = "1-st Class"
                   )
trace2 = go.Scatter(x = X[:,0][res==1]*4,
                    y =  X[:,1][res==1]*4,
                    mode='markers',
                   marker = dict(color='rgba(0, 120, 22, .9)'),
                   name = "2-nd Class"
                   )
layout = go.Layout(
    autosize=False,
    width=750,
    height=750,
    margin=go.Margin(
        l=50,
        r=50,
        b=100,
        t=100,
        pad=4
    )
)

fig = go.Figure(data=[trace1, trace2], layout=layout)
iplot(fig, filename='size-margins')
```

ДОДАТОК Б

Відомість атестаційної роботи магістра

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	80 с.
	Інші документи	
2.	Презентаційні матеріали	10 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Косьмін Д.С.			Нечітка машина опорних векторів для класифікації	Шифр групи	Код напр./спец.
Перевірив		Узлов Д.Ю.				СШІм-18-2	122
Н.контр.						ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					