پروژه سوم

SVM

ماشین بردار پشتیبان

درس هوش مصنوعی

نگین حقیقی – 99521226

استاد درس: دکتر آرش عبدی

نیم سال اول 1402-1401

سخن آغازین:

در این پروژه قصد داریم از ابزارها و کتابخانه های آماده SVM برای آشنایی با قابلیت های دسته بندی SVM استفاده کنیم.

قصد داریم داده های خود را به استفاده از خط svm کلاس بندی کرده و در شکل گرافیکی نقاط و خط جدا کننده و خط margin را نیز رسم کنیم.

برای مساله دو کلاسه، یک تعدادی نقطه در فضای دو بعدی ایجاد میکنیم و یک تعدادی از این نقاط را عضو کلاس 1 و بعضی دیگر را عضو کلاس 1- در نظر میگیریم.

در مرحله بعد با SVM داده های ساختگی خود را دسته بندی خواهیم کرد. و خط جدا کننده ای که svm یافته است و خط margin را در کنار نقاط آموزشی در یک نمودار رسم میکنیم.

در این پروژه نیز مانند قبلی، داده هایمان را به دو دسته ی آموزشی و آزمایشی تقسیم میکنیم و به اینگونه که 80٪ داده ها را آموزشی در نظر گرفته و 20٪ باقی را آزمایشی میگیریم.

داده ها به صورت (x,Y) هستند که x مختصات هر نقطه است. یعنی آرایه ای دو عضوی که مختصات نقاط در فضای دو بعدی را نگه میدارد و y نیز لیبل و کلاس مربوط به نقطه است.

پیاده سازی و توابع:

ابتدا کلاس SVM را پیاده سازی میکنیم. که در آن یکسری ویژگی های مربوط به آن را تعریف میکنیم برای مثال، بر اساس فرمول های داده شده در اسلایدها میدانیم به یک سری پارامتر مانند w و b و آلفا و لاندا و... نیاز داریم.

```
6   class SVM:
7       def __init__(self, alpha=0.001, landa=0.01, iterationCount=1000):
8           self.lr = alpha #learning rate
9           self.landa = landa
10          self.iterationCount = iterationCount
11          self.w = None
12          self.b = None
13
```

و حالا به تعریف یکسری توابع داخل این کلاس، میپردازیم. و در زیر شرح مختصری از هر تابع را خواهیم گفت.

کد تابع اولیه SVMfit:

```
13
14      def SVMfit(self, X, y):
15          SamplesCount, featuresCount = X.shape
16          y2 = []
17          for l in y:
18              if l <= 0:
19                  y2.append(-1)
20              else:
21                  y2.append(1)
22          self.w = np.zeros(featuresCount)
23          self.b = 0
24          for i in range(self.iterationCount):
25              for idx in range(len(X)):
26                  if y2[idx] * (np.dot(X[idx], self.w) - self.b) >= 1:
27                      self.w -= self.lr * (2 * self.landa * self.w)
28                  else:
29                      self.w -= self.lr * (2 * self.landa * self.w - np.dot(X[idx], y2[idx]))
30                      self.b -= self.lr * y2[idx]
31      def ResultPredict(self, X):
32          a = np.dot(X, self.w) - self.b
33          return np.sign(a)
34
```

در ابتدا ابعاد i و j آرایه ی دوبعدی X را با کمک تابع shape در متغیرهای sampleCount و featureCount میریزیم. میدانیم X آرایه ای از متخصات نقاط است. پس مقدار sampleCount تعداد نقاط را میدهد و featureCount تعداد ابعاد مختصاتی نقاط را که در اینجا 2 بعد دارد.

سپس آرایه y2 را که لیبل های نقاط است و کلاس بندی را انجام میدهد را درست میکنیم و در نتیجه داخل این آرایه تنها اعداد 1 و 1- قرار میگیرد.

حال مقدار w را مشخص میکنیم و در حالت اولیه تمام ابعادش را صفر میدهیم. میدانیم تعداد بعدهایش مانند تعداد بعدهای نقاط است پس این نیز فعلا 2 است. (w یک بردار عمود بر خط جدا کننده است) و مقدار b را نیز در ابتدا 0 میدهیم.

حال به یک تعداد (که قبلا به عنوان تعداد ایتریشن ها مشخص کردیم) باید روی تک تک نقاط forبزنیم و بررسی کنیم که در کدام کلاس است و فرمول مربوط به همان را استفاده کنیم.

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \le -\rho/2 \quad & \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}_i + b \ge \rho/2 \quad & \text{if } y_i = 1 \end{aligned} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \ge \rho/2$$

اگر حاصلضرب y در ضرب داخلی آن نقطه و w منهای b عددی بزرگتر از 1 باشد، از رابطه زیر استفاده میکنیم:

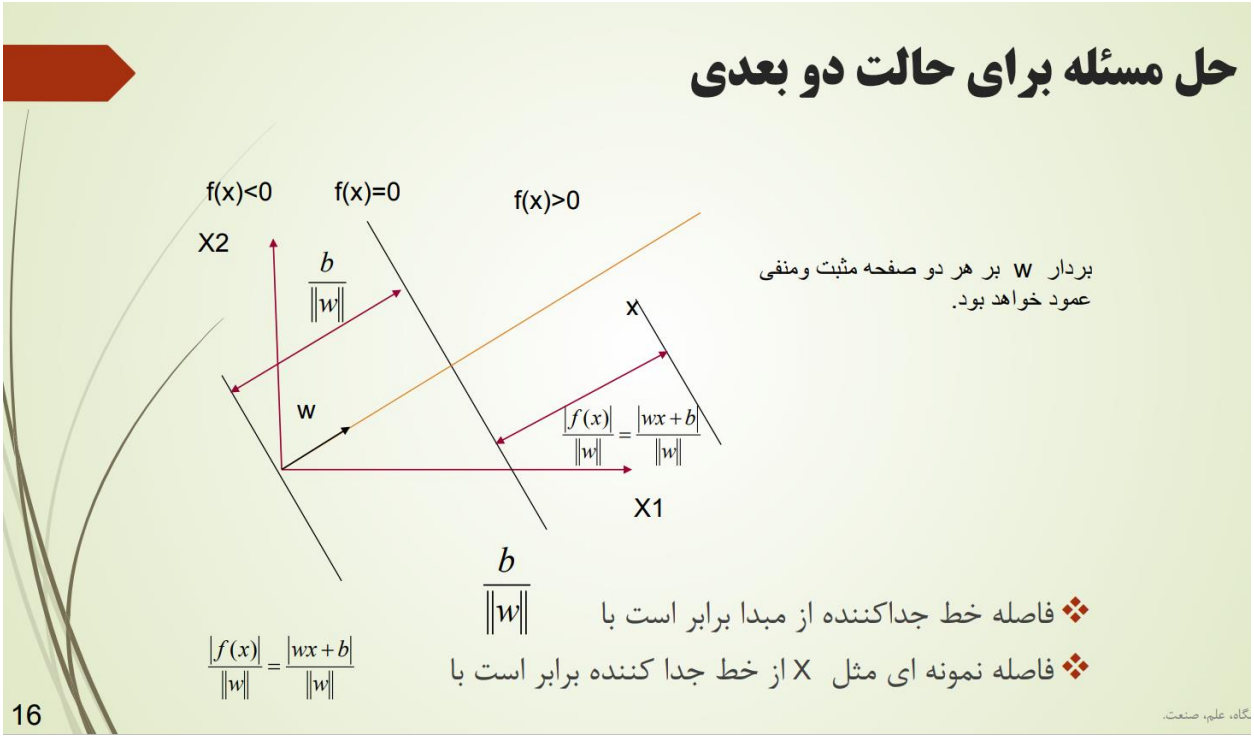
$$w = w - \alpha \cdot dw = w - \alpha \cdot 2\lambda w$$
$$b = b - \alpha \cdot db = b$$

که در خط 27 نشان داده شده است.

و در غیر اینصورت از رابطه زیر که در خط 29 و 30 پیاده سازی شده اند، استفاده میکنیم:

$$w = w - \alpha \cdot dw = w - \alpha \cdot (2\lambda w - y_i \cdot x_i)$$
$$b = b - \alpha \cdot db = b - \alpha \cdot y_i$$

اسلاید درس برای درک بهتر:



تابع پیش بینی ResultPredict:

این تابع برای پیش بینی نتیجه ی داده های آزمایشی هست و در خط 31 تعریف شده است. تابع Sign نیز همانطور که از اسمش پیداست علامت اعداد را پیدا میکند(خروجی 1- و 1+ برای اعداد منفی و مثبت و 0 برای عدد صفر)

تابع drawSVM:

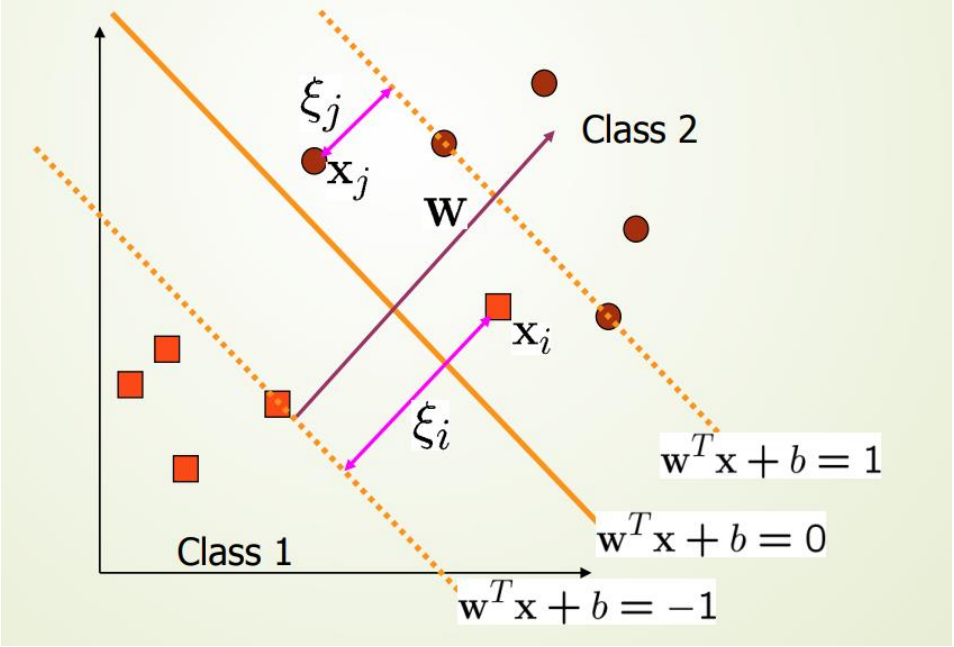
```
34
35  def drawSVM():
36      fg = plt.figure()
37      nemoodar = fg.add_subplot(1, 1, 1)
38      for i in range(len(y)):
39          if y[i] == -1:
40              plt.scatter(X[i, 0], X[i, 1], marker="o", color = "green")
41          else:
42              plt.scatter(X[i, 0], X[i, 1], marker=".", color = "red")
43      Xpoint_1 = np.amin(X[:, 0])
44      Xpoint_2 = np.amax(X[:, 0])
45      Ypoint_min = np.amin(X[:, 1])
46      Ypoint_max = np.amax(X[:, 1])
47      def ClassValueGet(x, w, b, offset):
48          return (-w[0]*x+b+offset)/w[1]
49      Ypoint_1 = ClassValueGet(Xpoint_1, svm.w, svm.b, 0)
50      Ypoint_2 = ClassValueGet(Xpoint_2, svm.w, svm.b, 0)
51      Ypoint_1_m = ClassValueGet(Xpoint_1, svm.w, svm.b, -1)
52      Ypoint_2_m = ClassValueGet(Xpoint_2, svm.w, svm.b, -1)
53      Ypoint_1_p = ClassValueGet(Xpoint_1, svm.w, svm.b, 1)
54      Ypoint_2_p = ClassValueGet(Xpoint_2, svm.w, svm.b, 1)
55
56      nemoodar.plot([Xpoint_1, Xpoint_2], [Ypoint_1, Ypoint_2], "k")
57      nemoodar.plot([Xpoint_1, Xpoint_2], [Ypoint_1_m, Ypoint_2_m], "b--")
58      nemoodar.plot([Xpoint_1, Xpoint_2], [Ypoint_1_p, Ypoint_2_p], "b--")
59      nemoodar.set_ylim([Ypoint_min-1, Ypoint_max+1])
60
```

در اینجا ابتدا به تعیین یکسری مشخصات برای رسم نمودار میپردازیم. در خطهای 36 و 37 ابعاد نمودار و در خطهای 38 تا 42 به رسم نقاط میپردازیم. تک تک نقاط بررسی شدند و شرط زده شده که اگر در کلاس 1- بودند با رنگ سبز و شکل دایره و اگر در کلاس 1 بودند با رنگ قرمز و شکل نقطه مشخص شوند.

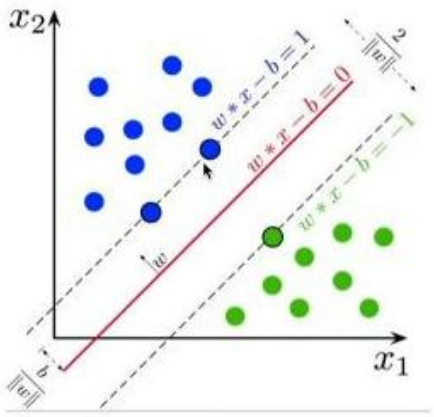
سپس در خطوط بعدی، کوچکترین و بزرگترین X مختصات نقاط و کوچکترین و بزرگترین y مختصات نقاط پیدا شده اند. و با استفاده از تابع ClassValueGet مشخص شده که نقطه در کدام کلاس است. یعنی بهتر است بگویم مشخص شده که نقطه در کدام طرف کدام خط margin است.

نقاطی که w.xi-b شان بزرگتر از 1 باشد در یک سمت خط margin و در کلاس 1 هستند و نقاطی که w.xi-bشان کوچکتر از 1- باشد در یک سمت خط دیگر و در کلاس 1- هستند.

در شکل های زیر کمی واضح تر همین حرف بیان شده است:



$$w \cdot x_i - b \geq 1 \quad \text{if } y_i = 1$$
$$w \cdot x_i - b \leq -1 \quad \text{if } y_i = -1$$



سپس در خط های 56 تا 58 خط جدا کننده و خط های margin مشخص شده اند. و در خط 59 نیز اندازه مختصات محور y را مشخص کردیم تا از 1 واحد بالاتر و پایین تر از نقطه ماکزیمم و مینیمم نشان داده شود و بیشتر از آن نیاز نیست.

بخش اصلی کد(main):

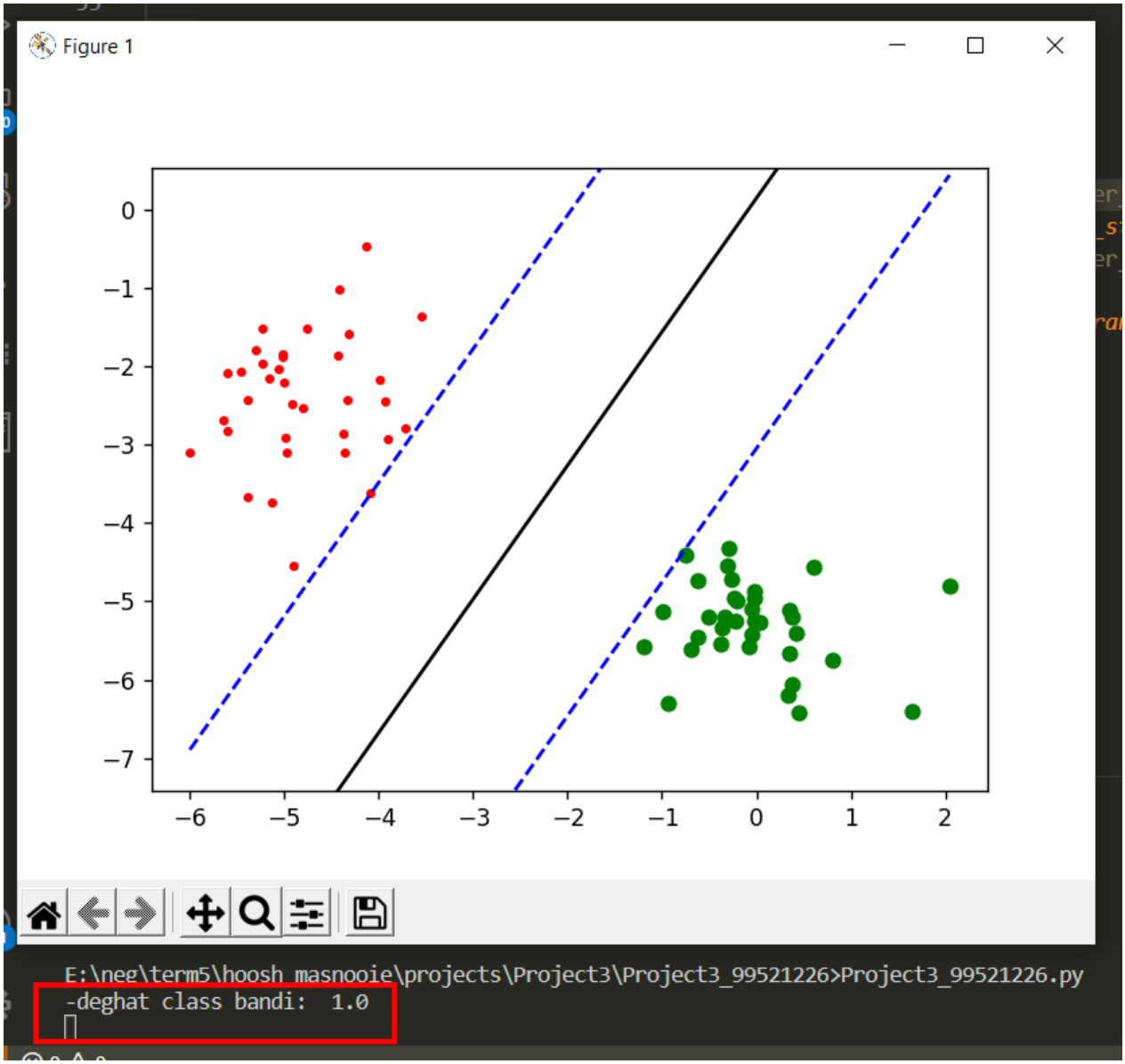
```
# X, y = datasets.make_blobs(n_samples=50, n_features=2, centers=2, cluster_std=1.05, random_state=40)
X, y = datasets.make_blobs(n_samples=70, n_features=2, centers=2, cluster_std=0.65, random_state=50)
# X, y = datasets.make_blobs(n_samples=30, n_features=2, centers=2, cluster_std=0.20, random_state=30)
y = np.where(y == 0, -1, 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
svm = SVM()
svm.SVMfit(X_train, y_train)
res_predictions = svm.ResultPredict(X_test)

deghat = np.sum(y_test == res_predictions) / len(y_test)
print("-deghat class bandi: ", deghat)

drawSVM()
plt.show()
```

ابتدا یکسری نقطه به طور رندوم و دلخواه در صفحه ایجاد کرده و مختصات آنها را در X و لیبل مربوط بهشان را در y ذخیره میکنیم. سپس لیبل هایشان را به دو دسته 1 و 1- تبدیل کرده و داده های آزمایشی و آموزشی را با همان نسبت 20٪ و 80٪ که پیشتر گفتیم، تعیین میکنیم.

و در تابع مربوط به پیش بینی، داده های آزمایشی را چک کرده و خروجی آنها را در یک آرایه نگه داری میکنیم و در آخر چک میکنیم که چه تعداد از آنها درست بوده اند و عدد محاسبه شده مربوط به دقت را چاپ میکنیم و نمودار را نیز نمایش میدهیم.

چند خروجی مختلف:

به ازای نقاط ساخته شده در خط های 61 و 62 و 63 نقاطی ایجاد کرده و خط جدا کننده با SVM آنها و خط margin و دقت هرکدام را بررسی میکنیم



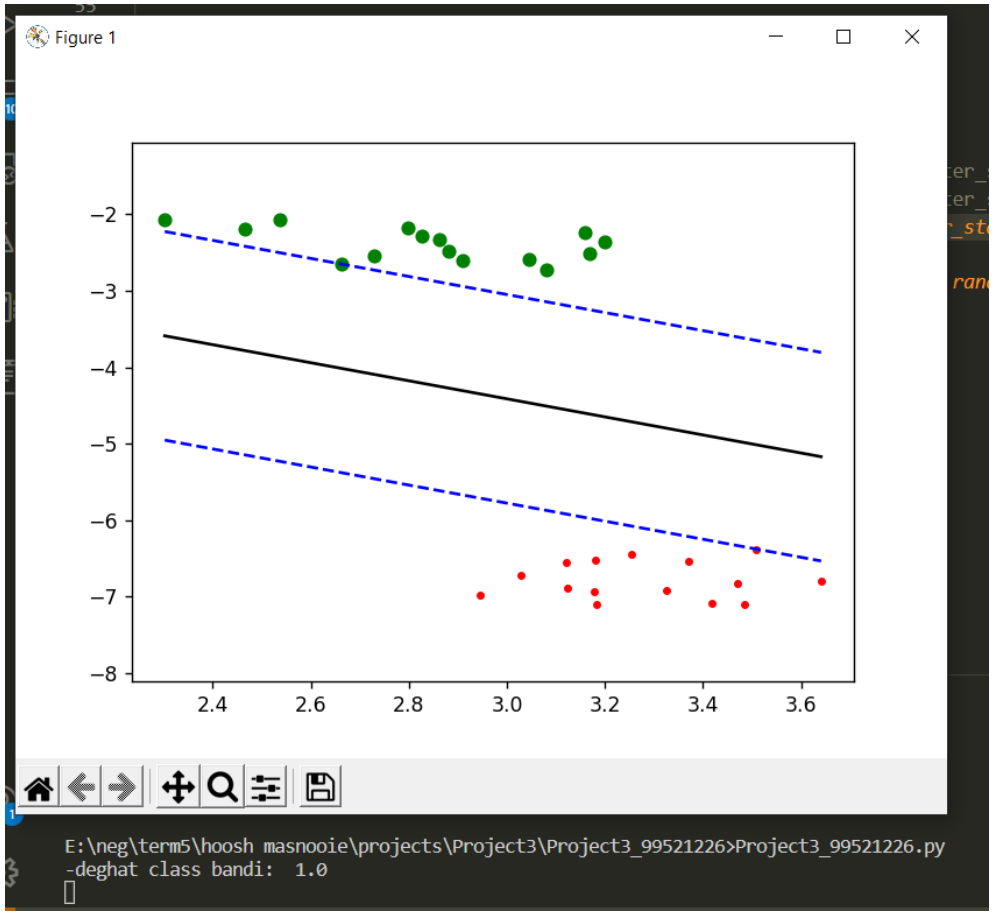


Figure 1
−2
−3
−4
−5
−6
−7
−8
2.4
2.6
2.8
3.0
3.2
3.4
3.6
E:\neg\term5\hoosh masnooie\projects\Project3\Project3_99521226>Project3_99521226.py
-deghat class bandi: 1.0

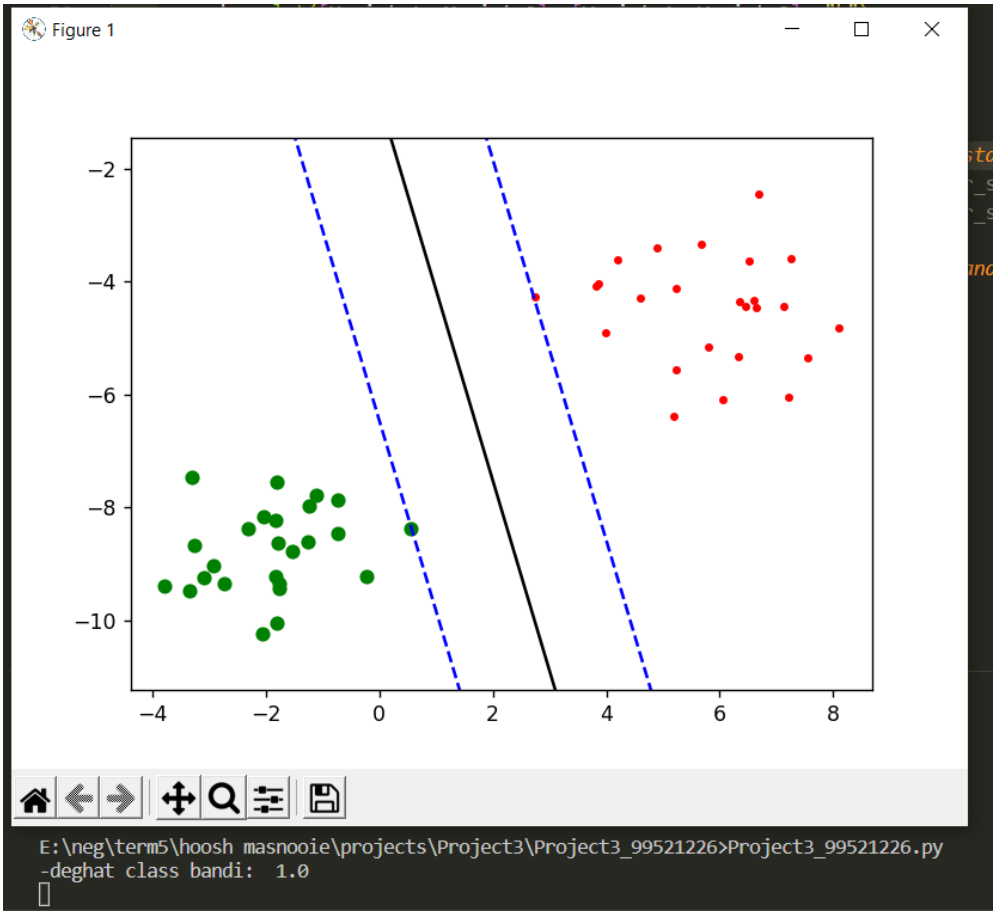


Figure 1
−2
−4
−6
−8
−10
−4
−2
0
2
4
6
8
E:\neg\term5\hoosh masnooie\projects\Project3\Project3_99521226>Project3_99521226.py
-deghat class bandi: 1.0

در طی انجام پروژه با چالش هایی مواجه شدم از جمله نمایش نقاط به صورت نیمه پراکنده و یا انتخاب و پیدا کردن فرمول مناسب برای هر بخش و یا یافتن بهترین خط جدا کننده و پیدا کردن یکسری توابع آماده برای راحتی کار مانند تابع make_blobs و ...

پایان

نگین حقیقی-پاییز1401