

تمرین شماره 4

علیرضا حسینی

شماره دانشجویی : ۸۱۰۱۰۱۱۴۲

یادگیری ماشین

دکتر ابوالقاسمی و دکتر توسلی پور

۱۴۰۲ بهار

فهرست مطالب

7 سوال 1 : سوالاتی در مورد SVM
7 1-1-1-1-هایپرپلین با مرز تصمیم LDF
8 1-1-2-نویز در داده آموزشی SVM
10 1-1-3-مفهوم کلی کرنل و بررسی بزرگی مارجین 2 کرنل
14 2-سوال 2 : بررسی اعتبار کرنل ها
17 3-سوال 3 : تبدیل separable-class به Soft margin SVM
18 4-سوال 4 : در دیتابست iris SVM
19 4-1-مقدمه و توضیح کد ها
32 4-2-نتایج برای داده های 2 کلاسه Sepal
49 4-3-نتایج برای داده های 2 کلاسه Petal
65 5-سوال 5 : سوالاتی در مورد SVM
65 5-1-بخش 1
66 5-2-بخش 2
66 5-3-بخش سوم
67 6-سوال 6 : Ensamble Model
70 7-سوال 7 : تخمین میزان درآمد و تخمین قیمت هتل
70 7-1-تخمین میزان درآمد
74 7-2-تخمین قیمت هتل

فهرست اشکال

12 شکل (۱-۱) مارجین برای ϕ_1
13 شکل (۲-۱) مارجین برای ϕ_2
13 شکل (۳-۱) کد پایتون بررسی مارجین ها.
14 شکل (۴-۱) سوال ۲ بخش ۱
15 شکل (۵-۱) سوال ۲ بخش ۲
16 شکل (۶-۱) سوال ۲ بخش ۳
16 شکل (۷-۱) سوال ۲ بخش ۴
17 شکل (۸-۱) سوال ۲ بخش ۵
19 شکل (۹-۱) کد پایتون لود کردن دیتاست
20 شکل (۱۰-۱) کد نگه داشتن ۲ ویژگی اول
20 شکل (۱۱-۱) کد نگه داشتن ۲ ویژگی دوم
20 شکل (۱۲-۱) کد encode کردن لیل ها
21 شکل (۱۳-۱) کد مقدار دهی ماتریس های X و Y
21 شکل (۱۴-۱) تقسیم داده ها به داده های آموزش و تست
23 شکل (۱۵-۱) تعریف کرnel های مدنظر
24 شکل (۱۶-۱) فیت کردن با کرnel مورد و خروجی گرفتن روی داده تست و گزارش متريک ها
24 شکل (۱۷-۱) رسم مرز جدا شونده به ازای هر کرnel
26 شکل (۱۸-۱) بررسی مقادیر مختلف گاما در مرز تصمیم (شکل از سایت vitalflux)
27 شکل (۱۹-۱) بررسی هایپرپارامتر C
28 شکل (۲۰-۱) کد مقدار دهی های مختلف C و gamm و rbf برای
30 شکل (۲۱-۱) پیاده سازی جستجوی شبکه ای
32 شکل (۲۲-۱) کد بررسی رویکرد های ovo و ovr
33 شکل (۲۳-۱) Classification Report و ماتریس آشفتگی برای کرnel خطی (Sepal)
33 شکل (۲۴-۱) مرز جداسازی با استفاده از کرnel خطی (Sepal)
34 شکل (۲۵-۱) Classification Report و ماتریس آشفتگی برای کرnel poly (Sepal)
34 شکل (۲۶-۱) مرز جداسازی با استفاده از کرnel poly (Sepal)
35 شکل (۲۷-۱) Classification Report و ماتریس آشفتگی برای کرnel rbf (Sepal)
35 شکل (۲۸-۱) مرز جداسازی با استفاده از کرnel rbf (Sepal)
36 شکل (۲۹-۱) ۱
37 شکل (۳۰-۱) ۱
37 شکل (۳۱-۱) ۲
38 شکل (۳۲-۱) ۲

38.....	3 (۳۳-۱) شکل
39.....	3 (۳۴-۱) شکل
39.....	4 (۳۵-۱) شکل
40.....	4 (۳۶-۱) شکل
40.....	5 (۳۷-۱) شکل
41.....	5 (۳۸-۱) شکل
41.....	6 (۳۹-۱) شکل
42.....	6 (۴۰-۱) شکل
42.....	7 (۴۱-۱) شکل
43.....	7 (۴۲-۱) شکل
43.....	8 (۴۳-۱) شکل
44.....	8 (۴۴-۱) شکل
44.....	9 (۴۵-۱) شکل
45.....	9 (۴۶-۱) شکل
46.....	شکل (۴۷-۱) بهترین هایپر پارامتر پس از جستجوی شبکه ای و گزارش classification آن (sepal)
46.....	شکل (۴۸-۱) مرز تصمیم rbf با بهترین پارامتر ها (sepal)
47.....	شکل (۴۹-۱) نتایج رویکرد ovr با کرنل rbf (sepal)
48.....	شکل (۵۰-۱) مرز تصمیم رویکرد ovr با کرنل rbf (sepal)
48.....	شکل (۵۱-۱) نتایج روکرد ovo در کرنل rbf (Sepal)
49.....	شکل (۵۲-۱) Classification Report و ماتریس آشنتگی برای کرنل خطی (petal)
50.....	شکل (۵۳-۱) مرز جداسازی با استفاده از کرنل خطی (petal)
50.....	شکل (۵۴-۱) Classification Report و ماتریس آشنتگی برای کرنل poly (petal)
51.....	شکل (۵۵-۱) مرز جداسازی با استفاده از کرنل poly (petal)
51.....	شکل (۵۶-۱) Classification Report و ماتریس آشنتگی برای کرنل rbf (petal)
52.....	شکل (۵۷-۱) مرز جداسازی با استفاده از کرنل rbf (petal)
53.....	شکل (۵۸-۱) 1
53.....	شکل (۵۹-۱) 1
54.....	شکل (۶۰-۱) 2
54.....	شکل (۶۱-۱) 2
55.....	شکل (۶۲-۱) 3
55.....	شکل (۶۳-۱) 3
56.....	شکل (۶۴-۱) 4
56.....	شکل (۶۵-۱) 4
57.....	شکل (۶۶-۱) 5

57 شکل (۶۷-۱) ۵
58 شکل (۱) ۶ (۶۸-۱)
58 شکل (۱) ۶ (۶۹-۱)
59 شکل (۷) ۷ (۷۰-۱)
59 شکل (۷) ۷ (۷۱-۱)
60 شکل (۸) ۸ (۷۲-۱)
60 شکل (۸) ۸ (۷۳-۱)
61 شکل (۹) ۹ (۷۴-۱)
61 شکل (۹) ۹ (۷۵-۱)
62 شکل (۱) ۹ (۷۶-۱) بهترین هایپر پارامتر پس از جستجوی شبکه ای و گزارش classification آن (petal)
63 شکل (۱) ۹ (۷۷-۱) مرز تصمیم rbf با بهترین پارامتر ها (sepal)
63 شکل (۱) ۹ (۷۸-۱) نتایج رویکرد ovr با کرنل rbf (sepal)
64 شکل (۱) ۹ (۷۹-۱) مرز تصمیم رویکرد ovr با کرنل rbf (sepal)
64 شکل (۱) ۹ (۸۰-۱) نتایج رویکرد ovo در کرنل rbf (Sepal)
65 شکل (۱) ۱ سوال ۵ (۸۱-۱) بخش ۱ سوال ۵
66 شکل (۱) ۱ سوال ۵ (۸۲-۱) بخش ۲ سوال ۵
68 شکل (۱) ۱ پاسخ سوال ۶ (۸۳-۱)
70 شکل (۱) ۱ خواندن دیتاست Q7-part1
70 شکل (۱) ۱ پیش پردازش داده های Q7-part1
71 شکل (۱) ۱ تعریف کرنل های svm
71 شکل (۱) ۱ کد پایتون فیت کردن داده ها روی هر کرنل و گزارش دقت (r^2) روی ترین و تست
72 شکل (۱) ۱ کد پایتون پیشینی داده های تست و رسم همه داده ها و پیش بینی شده آن ها کنار هم
72 شکل (۱) ۱ دقت score r^2 روی داده ترین و تست کرنل rbf
72 شکل (۱) ۱ منحنی داده های واقعی و پیش بینی شده با کرنل rbf
73 شکل (۱) ۱ دقت score r^2 روی داده ترین و تست کرنل linear
73 شکل (۱) ۱ منحنی داده های واقعی و پیش بینی شده با کرنل linear
73 شکل (۱) ۱ دقت score r^2 روی داده ترین و تست کرنل poly
74 شکل (۱) ۱ منحنی داده های واقعی و پیش بینی شده با کرنل poly
75 شکل (۱) ۱ لود دیتاست تست و ترین داده های هتل
75 شکل (۱) ۱ تفکیک ستون ها به X و Y
75 شکل (۱) ۱ نرمال کردن داده های عددی
75 شکل (۱) ۱ داده های Encode categorical
76 شکل (۱) ۱ کردن داده های categorical و چسباندن داده های numerical و categorical به هم
76 شکل (۱) ۱ ADR نرمال سازی داده های

77 شکل (۱۰۱-۱) جای گذاری مقادیر miss شده
77 شکل (۱۰۲-۱) توزیع مقادیر داده های آموزش ADR
78 شکل (۱۰۳-۱) آموزش Support Vector Regression
78 شکل (۱۰۴-۱) گرفتن خروجی و بررسی مقدار خطای MSE
79 شکل (۱۰۵-۱) ذخیره سازی مقادیر پیشینی شده و واقعی و اختلاف آن ها در یک فایل CSV

1-1- سوال 1: سوالاتی در مورد SVM

1-1-1- هایپرپلین با مرز تصمیم : LDF

در یک ابر صفحه که مرز یکتابع تشخیص خطی (LDF) است، جهت و مکان صفحه با ضرایب معادله

LDF تعیین می شود.

معادله LDF معمولاً به شکل زیر است:

$$f(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$$

جایی که:

تابع تفکیک کننده ای است که برچسب کلاس را برای بردار ورودی x مشخص می کند.

x_1, x_2, \dots, x_n ویژگی های بردار ورودی x هستند.

w_1, w_2, \dots, w_n وزن ها یا ضرایب مرتبط با ویژگی ها هستند.

b بایاس است.

جهت ابر صفحه با وزنه های w_1, w_2, \dots, w_n تعریف می شود. هر وزن مربوط به یک ویژگی است و نشان

دهنده اهمیت یا سهم آن ویژگی در فرآیند تصمیم گیری است. مقادیر نسبی و نشانه های وزن ها جهت گیری

هایپرپلین را در فضای ویژگی تعیین می کند. اگر 2 بردار در این صفحه پیدا کرده و ضرب خارجی آن را پیدا

کنیم بردار حاصل بر صفحه عمود است و جهت آن را تعیین میکند. به عبارت دیگر بردار وزن های w خود

برداری میشود که همواره بر ابر صفحه عمود میباشد.

$$W^T X + b = 0 \rightarrow \langle W^T, x_1 x_2 \rangle = 0$$

مکان ابر صفحه در امتداد جهت با عبارت بایاس b تعیین می شود. افست یا فاصله هایپرپلین از مبدا را کنترل

می کند. اصطلاح بایاس، ابر صفحه را به موازات جهت آن تغییر می دهد.

در طول مرحله آموزش یک مدل LDF، وزن‌های w_1, w_2, \dots, w_n و عبارت بایاس b تنظیم می‌شوند تا

خطای طبقه‌بندی را به حداقل برسانند و ابرصفحه بهینه را پیدا کنند که کلاس‌های مختلف را از هم جدا می‌کند.

الگوریتم‌های مختلفی مانند پرسپترون یا ماشین‌های بردار پشتیبان (SVM)، می‌توانند برای یادگیری ضرایب هایپرپلن بر اساس داده‌های آموزشی برچسب‌گذاری شده استفاده شوند.

هنگامی که پارامترهای هایپرپلین تعیین شدند، مرز تصمیم LDF تعیین می‌شود. نقاط یک طرف ابرصفحه در یک کلاس طبقه‌بندی می‌شوند، در حالی که نقاط طرف دیگر در یک کلاس متفاوت طبقه‌بندی می‌شوند.

کلاس خاصی که به یک نقطه اختصاص داده می‌شود بستگی به این دارد که در کدام سمت ابرصفحه قرار بگیرد.

1-1-2- نویز در داده آموزشی SVM

ماشین‌های بردار پشتیبان (SVM) به دلیل توانایی خود در مدیریت نویز در داده‌های آموزشی و ایجاد جداسازی قوی بین کلاس‌ها شناخته شده‌اند. ویژگی‌های ذاتی SVM به آن‌ها اجازه می‌دهد تا به طور موثر داده‌های نویزی را مدیریت کنند و همچنان به عملکرد طبقه‌بندی خوبی دست یابند.

SVM با یافتن ابرصفحه بهینه که حاشیه بین کلاس‌ها را به حداقل می‌رساند و در عین حال خطاهای طبقه‌بندی را به حداقل می‌رساند، به جدایی می‌رسد. حاشیه فاصله بین مرز تصمیم و نزدیکترین نقاط داده از هر کلاس است. با به حداقل رساندن این حاشیه، SVM به دنبال یافتن یک مرز تصمیم است که کمترین تأثیر را از نویز در داده‌ها داشته باشد.

در اینجا نحوه هندل کردن نویز در SVM در طول فرآیند جداسازی آمده است:

• حداکثر کردن حاشیه: هدف SVM یافتن هایپرپلنی است که حاشیه را به حداکثر می رساند، که

منطقه ای است که نقاط داده از هر دو کلاس به خوبی از هم جدا شده اند. با به حداکثر رساندن

حاشیه، SVM در برابر نویز انعطاف پذیرتر است زیرا بر روی قابل اطمینان ترین و جداسازی شده

ترین نمونه ها تمرکز می کند.

بردارهای پشتیبان(SV) : SVM بردارهای پشتیبان را که نزدیک ترین نقاط داده به مرز تصمیم

هستند، شناسایی می کند. این بردارهای پشتیبان نقش مهمی در تعیین ابر صفحه و طبقه بندی دارند.

SVM فقط به زیرمجموعه ای از داده های آموزشی (بردارهای پشتیبانی) متکی است نه کل مجموعه

داده، و تأثیر نقاط داده نویزی را که ممکن است پرت یا دارای برچسب اشتباه باشند، کاهش می

دهد.

• منظم سازی: SVM از یک پارامتر منظم سازی (غلب با C نشان داده می شود) استفاده می کند که

مبادله بین دستیابی به حاشیه بزرگتر و اجازه دادن به برخی طبقه بندی های اشتباه را کنترل می کند.

با تنظیم این پارامتر، SVM می تواند اثر نقاط نویزی یا پرت را متعادل کند، و به طور موثر تأثیر

نقاط داده فردی را بر روی مرز تصمیم کنترل کند.

• جداسازی غیر خطی: SVM می تواند از توابع کرنل برای نگاشت داده ها به یک فضای ویژگی با

بعد بالاتر استفاده کند، جایی که جداسازی غیر خطی ممکن می شود. این انعطاف پذیری به SVM

اجازه می دهد تا مرزهای تصمیم پیچیده ای را پیدا کند که بهتر می تواند نویز و غیرخطی ها را در

داده ها مدیریت کند.

3-1-1- مفهوم کلی کرنل و بردسی بزدگی مارجین 2 کرنل

مفهوم کرنل :

در یادگیری ماشینی، کرنل تابعی است که به ما امکان می‌دهد به طور ضمنی ضرب نقطه‌ای (شماحت) بین نقاط داده را در یک فضای ویژگی با ابعاد بالاتر بدون نگاشت صریح داده‌ها در آن فضا محاسبه کنیم. کرنل‌ها نقش اساسی در الگوریتم‌های مختلف مانند ماشین‌های بردار پشتیبان (SVM)، تجزیه و تحلیل (PCA) و فرآیندهای گاووسی ایفا می‌کنند.

مفهوم کلی کرنل را می‌توان به صورت زیر توضیح داد:

1. فضای ویژگی: در الگوریتم‌های یادگیری ماشین سنتی، ما با داده‌هایی کار می‌کنیم که در یک

فضای ویژگی نمایش داده می‌شوند، جایی که هر نقطه داده با مجموعه‌ای از ویژگی‌ها یا ویژگی‌ها توصیف می‌شود. به عنوان مثال، در یک فضای ویژگی دو بعدی، هر نقطه داده را می‌توان با دو ویژگی، مانند مختصات (x, y) نشان داد.

2. نقشه برداری به فضای بالاتر: کرنل‌ها ما را قادر می‌سازند تا به طور ضمنی نقاط داده را از فضای ویژگی اصلی به یک فضای ویژگی با ابعاد بالاتر نگاشت کنیم. این نگاشت به گونه‌ای انجام می‌شود که ضرب نقطه‌ای (شماحت‌ها) بین نقاط داده را حفظ می‌کند.

3. ضرب نقطه‌ای (dot product) در فضای ویژگی: در فضای ویژگی‌های با ابعاد بالاتر، تابع کرنل به ما اجازه می‌دهد تا ضرب نقطه‌ای را بین دو نقطه داده بدون محاسبه صریح نمایش‌های نگاشت شده آن نقاط محاسبه کنیم. ضرب نقطه نشان دهنده شماحت یا رابطه بین نقاط داده است.

4. تابع کرنل: تابع کرنل اندازه گیری شباهت بین دو نقطه داده در فضای ویژگی اصلی را ارائه می

دهد. بردارهای ویژگی اصلی را به عنوان ورودی می گیرد و مقدار شباهت را محاسبه می کند، که

معمولأً بر اساس حاصل ضرب نقطه ای نمایش های نگاشت شده در فضای ویژگی با ابعاد بالاتر

است. انتخاب تابع کرنل به مقدار و ویژگی های داده بستگی دارد.

5. ترفندهای کرنل: ترفندهای کرنل یک تکنیک ریاضی است که به ما امکان می دهد محاسبات را بر اساس

تابع کرنل بدون محاسبه صریح فضای ویژگی با ابعاد بالاتر انجام دهیم. این ترفندهای طور موثری از

ویژگی های تابع کرنل برای فعال کردن محاسبات کارآمد، حتی در فضاهای ویژگی های بی بعدی

استفاده می کند.

6. کاربردها: کرنل ها به ویژه در الگوریتم هایی مانند SVM مفید هستند، جایی که آنها امکان

جداسازی موثر نقاط داده را در یک فضای ویژگی غیرخطی قابل تفکیک می دهند. با استفاده از

توابع کرنل مناسب، SVM می تواند مرزهای تصمیم گیری پیچیده را پیدا کند و روابط غیر خطی

بین ویژگی ها را مدیریت کند.

سوال در مورد 2 کرنل داده شده :

برای تعیین اینکه کدام کرنل حاشیه بزرگتری تولید می کند، باید مرزهای تصمیم تولید شده توسط دو کرنل

را با هم مقایسه کنیم. حاشیه فاصله بین مرز تصمیم و بردارهای پشتیبانی است.

باید طبقه بندی کننده SVM را با استفاده از یک کرنل چند جمله ای مرتبه دوم با اولین تابع نگاشت

$\phi(x) = [x, x^2]$ در نظر بگیریم. مرز تصمیم منحنی است که نقاط داده را در فضای ویژگی تعریف شده توسط

داده های ورودی تبدیل شده جدا می کند.

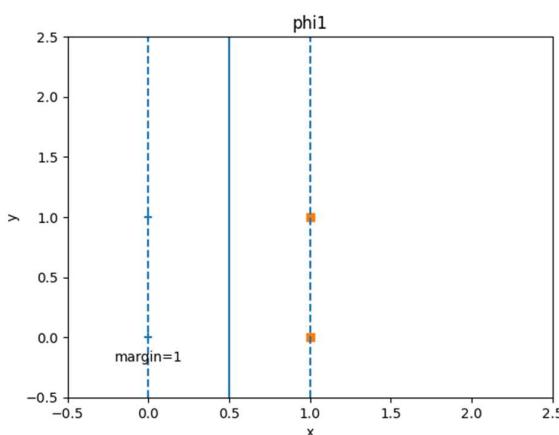
به طور مشابه، برای دومین کرنل چند جمله‌ای با تابع نگاشت $\phi_2(x) = [2x, 2x^2]$ ، مرز تصمیم منحنی دیگری در فضای ویژگی است.

با مقایسه این دو کرنل، مشاهده می‌کنیم که کرنل چند جمله‌ای دوم، $\phi_2(x) = [2x, 2x^2]$ ، حاشیه بزرگ‌تری تولید می‌کند. دلیل این امر این است که کرنل دوم ضریب مقیاس پذیری (2) بیشتری نسبت به کرنل اول است. در نتیجه، کرنل دوم فضای ویژگی را گسترش می‌دهد، فاصله بین نقاط داده و مرز تصمیم را افزایش می‌دهد و منجر به حاشیه بزرگ‌تر می‌شود.

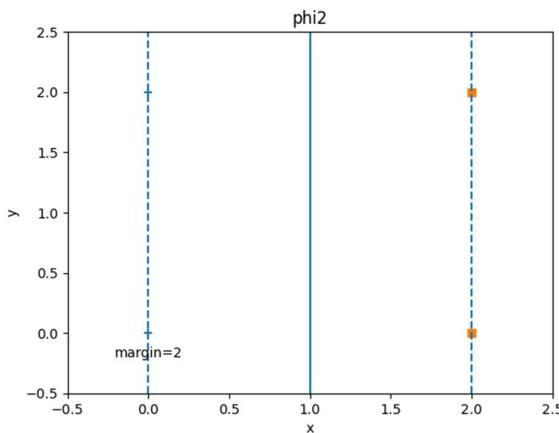
به طور کلی، افزایش ضریب مقیاس در یک کرنل چند جمله‌ای حاشیه را بزرگ‌می‌کند زیرا فضای ویژگی را گسترش می‌دهد و باعث می‌شود که مرز تصمیم کمتر مستعد بیش از حد برآش و تعمیم بهتر به داده‌های جدید باشد.

به عنوان مثال فرض کنیم مجموعه داده‌ای با کلاس مثبت (0.0)، (0.1) و کلاس منفی در (1.0)، (1،1) داریم

به ترتیب مرز‌ها برای ϕ_1 و ϕ_2 به صورت زیر می‌شود:



شکل (۱-۱) مارجین برای ϕ_1



شکل (۱-۲) مارجین برای ϕ_2

کد پایتون استفاده شده برای plot های فوق به صورت زیر میباشد :

```
def plot(D, main, sub):
    plt.plot()
    plt.xlim(-0.5, 2.5)
    plt.ylim(-0.5, 2.5)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title(main)
    plt.text(0, -0.2, sub, ha='center')
    plt.scatter([0, 0], [0, D], marker='+')
    plt.scatter([D, D], [0, D], marker='s')
    plt.axvline(x=0, linestyle='dashed')
    plt.axvline(x=D, linestyle='dashed')
    plt.axvline(x=D/2)
    plt.show()

plot(1, 'phi1', 'margin=1')
plot(2, 'phi2', 'margin=2')
```

شکل (۱-۳) کد پایتون بررسی مارجین ها

1-2- سوال 2: بررسی اعتبار کرنل ها

در عکس های زیر اثبات هر بخش آمده است.

$$\begin{aligned}
 k(x, y) &= \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right) : \text{کرنل RBF} \quad \text{①} \\
 k(x, y) &= \exp\left(-\frac{\|x\|^2 + \|y\|^2 - 2x^T y}{\sigma^2}\right) = \exp\left(-\frac{\|x\|^2}{\sigma^2}\right) \exp\left(-\frac{\|y\|^2}{\sigma^2}\right) \\
 &= g(x) g(y) \exp(-\kappa(x, y)) \\
 &\quad \downarrow \\
 &\text{در بخش ۳ همچنین مذکور است که} \\
 &\text{که این کرنل نیز دویستیک} \\
 &\text{دقیقاً برای این مسئله} \quad \text{۱.} \\
 &\text{یعنی معمولیست که بگوییم} \quad \text{۲.} \\
 e^{\kappa(x, y)} &= 1 + \frac{\kappa(x, y)}{2} + \frac{\kappa(x, y)^2}{3!} + \dots \\
 &\text{که این عبارت باز هم میگوید} \quad \text{۳.} \\
 &\text{و همچنان با توجه به این نتایج} \\
 &\alpha x_1 + \beta x_2 \text{ با خود} \quad \text{۴.} \\
 &\text{معترضی} \quad \text{۵.} \\
 &\text{یعنی داریم} \quad \text{۶.} \\
 &\exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right) \quad \text{۷.} \\
 &\text{یعنی} \quad \text{۸.} \\
 &\exp\left(-\frac{\|\alpha x_1 + \beta x_2 - y\|^2}{\sigma^2}\right) \quad \text{۹.}
 \end{aligned}$$

شکل (۱-۴) سوال 2 بخش 1

$$k(\alpha x + \beta y) = \alpha k(x, y) + \beta k(x, y) \quad (2.1)$$

$$\alpha, \beta > 0 \quad (2.1)$$

Positive scaling of kernel function does not affect validity

جمله در ضرب دلخواه تابع کرنل معتبر است

$$k(\alpha x + \beta y) = k_1(x, y) + k_2(x, y)$$

$$= \Phi_1^T \Phi_1(y) + \Phi_2^T \Phi_2(y)$$

$$= (\underbrace{\Phi_1(x), \Phi_2(x)}^{\Phi(x)^T}) (\underbrace{\Phi_1(y), \Phi_2(y)}_{\Phi(y)}) = \Phi(x)^T \Phi(y)$$

برای دلخواه $\alpha, \beta > 0$ معتبر است

$$\alpha, \beta > 0 \quad (2.2)$$

جمله در ضرب دلخواه $\alpha, \beta > 0$ معتبر است

و دلخواه مثبت می باشد

$$\text{حلن است که } \alpha, \beta > 0 \quad \Leftrightarrow \quad \alpha > 0, \beta > 0 \quad (2.3)$$

$$\alpha, \beta < 0 \quad (2.4)$$

$$k(x,y) = f(x) f(y) \quad (3)$$

$\left[\begin{matrix} x_1, x_2, \dots, x_n \end{matrix} \right]$ ماتریس
 وکتور $\left[\begin{matrix} c_1, c_2, \dots, c_n \end{matrix} \right] \rightarrow$ میکرو
لینیار $\rightarrow \left[\begin{matrix} k_{ij} \end{matrix} \right] = k(x_i, x_j) = f(x_i) f(x_j)$
 $\rightarrow K = F F^T \rightarrow F_i = f(x_i)$
 $v = \left[\begin{matrix} c_1, c_2, \dots, c_n \end{matrix} \right]^T$ کوکس vector
 $v^T K v = v^T F F^T v = (F^T v)^T (F^T v) = \|F^T v\|^2 > 0$ جیوو
positive definite
 حلقی حلقی symmetric
 بین $f(x) f(y) = f(y) f(x)$ حمسراست

شكل (١-٦) سوال 2 بخش 3

$$k(x,y) = h_1(g(x), g(y)) \quad (4)$$

تابع :
 $k_{ij} = h_1(g(x_i), g(x_j)) \rightarrow K = h_1(G, G^T)$
 $v^T K v = v^T h_1(G, G^T) v = (G^T v)^T h_1(G, G^T) (G^T v)$
جیوو حلقی
جیوو حلقی
جیوو حلقی

شكل (١-٧) سوال 2 بخش 4

$$\begin{aligned}
 k(u, y) &= \phi_1(u, y) \cdot \phi_2(y) & (5) \\
 &= \underbrace{\phi_1(u)^T \phi_1(y)}_{\Phi(u)^T} \underbrace{\phi_2(y)^T}_{\Phi(y)} & ; \text{ اذن: } ab^T = ba^T \quad (I) \\
 \stackrel{(I)}{=} \Phi(u)^T \Phi(y) & = \Phi(u)^T \Phi(y) \quad \checkmark
 \end{aligned}$$

مهم: $\Phi(u)^T \Phi(y)$ يُسمى مatrice خطيّة

شکل (1-۸) سوال ۲ بخش ۵

سوال ۳-۱: تبدیل Soft margin SVM به separable-class

برای نشان دادن اینکه تابع هزینه برای Soft Margin SVM می تواند به یک مسئله قابل تفکیک کلاس

تبديل شود، باید آن را بر حسب قابلیت تفکیک کلاس بیان کنیم.

اگر به صورت زیر تعریف شود:

$$\frac{1}{2} \left| \left| w \right| \right|^2 + \frac{c}{2} \sum_{i=1}^N \varepsilon^2$$

۶ پارامتر تنظیم است.

N تعداد نمونه های آموزشی است.

28 نشان دهنده متغیر های slack محدود است.

عبارت اول که مساله اصلی بوده و صراحتاً تفکیک پذیری کلاس ها را در نظر نمی گیرد.

عبارت دوم : نشان دهنده متغیرهای (ی) slack مرتبط با نمونه های طبقه بندی نادرست یا ناقص حاشیه است.

این مدل را برای هر طبقه بندی اشتباه یا نقض حاشیه جریمه می کند. با به حداقل رساندن این عبارت، ما SVM را تشویق می کنیم تا مرز تصمیمی را بیابد که حاشیه را به حداکثر می رساند در حالی که هنوز امکان برخی طبقه بندی های اشتباه را فراهم می کند.

برای تبدیل این تابع هزینه به یک مسئله قابل تفکیک کلاس، می توانیم متغیرهای (ی) slack را با معیار تفکیک پذیری کلاس جایگزین کنیم. یکی از راه های دستیابی به این هدف، معرفی تابع افت لولا است که معمولاً در فرمول های SVM استفاده می شود.

تابع لاغرانژ به صورت زیر می شود:

$$L(W, b) = \sum_{i=1}^N \max(0, 1 - y_i^*(W^T x_i + b))$$

بنابراین تابع cost به صورت زیر می شود :

$$C = 1/2 \|W\|^2 + c * \sum_{i=1}^N \max(0, 1 - y_i^*(W^T x_i + b))$$

به حداقل رساندن این تابع هزینه، SVM را تشویق می کند تا مرز تصمیمی را بیابد که حاشیه را به حداکثر می رساند و کلاس ها را به طور موثر جدا می کند.

۱-۴- سوال 4 : SVM در دیتاست iris

با توجه به اینکه مساله به 2 مساله 2 کلاسه تبدیل می شود و در ادامه به ازای کرنل های مختلف و هایپر پارامتر های مختلف مساله بررسی می شود ، بدین ترتیب پاسخ سوالات و نتایج و مقایسه های بخش 1 و 2 و 3 و 4 و 5 در در هر بخش آمده است و با توجه به یکسان بودن کد ها در هر 2 مورد (استفاده از داده های Sepal Petal یا)

در بخش اول یک مقدمه کلی و مقایسه کلی هایپر پارامتر ها و کرنل ها و توضیح کد ها آمده است و در ادامه نتایج نهایی هر بخش آمده است.

1-4-1- مقدمه و توضیح کد ها

در این بخش یک سری مقدمات و توضیحات و مقایسه ها و توضیحات کلی کد های بخش بعدی آمده است.

1-4-1-1- لود کردن دیتاست و پیش پردازش

ابتدا فایل CSV مربوطه را دانلود کرده و در ادامه به کمک کتابخانه pandas آن را لود میکنیم.

```
# load Dataset
iris_data = pd.read_csv('Iris.csv')
iris_data.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

شکل (۱-۹) کد پایتون لود کردن دیتاست

در ادامه ۲ ویژگی Sepal یا Petal را جدا میکنیم.

```
# Keep first two feature
iris_data_1 = iris_data.drop(['Id', 'PetalLengthCm', "PetalWidthCm"], axis=1)
iris_data_1.head()
```

	SepalLengthCm	SepalWidthCm	Species
0	5.1	3.5	Iris-setosa
1	4.9	3.0	Iris-setosa
2	4.7	3.2	Iris-setosa
3	4.6	3.1	Iris-setosa
4	5.0	3.6	Iris-setosa

شکل (۱-۱۰) کد نگه داشتن ۲ ویژگی اول

مشابه کد فوق میتوان ۲ ویژگی دوم را نیز جدا کرد.

```
#remove first two feature
iris_data_2 = iris_data.drop(['Id', 'SepalLengthCm', "SepalWidthCm"], axis=1)
```

شکل (۱-۱۱) کد نگه داشتن ۲ ویژگی دوم

در ادامه پیش پردازش های لازم را انجام میدهیم :

- انکود کردن لیل ها به ۰ و ۱ و ۲

```
# Encode Label
le = LabelEncoder()
iris_data_1 ['Species'] = le.fit_transform(iris_data['Species'])
```

شکل (۱-۱۲) کد encode کردن لیل ها

- تقسیم داده ها به X و Y

```
# split data to X , Y
X = iris_data_1.drop('Species', axis=1)
y = iris_data_1['Species']
```

شکل (۱-۱۳) کد مقدار دهی ماتریس های X و Y

- نرمال سازی داده ها

یکی از موارد این است که ۲ ویژگی در یک بازه عددی باشند و نسبت به هم برتری ای نداشته

باشند (مثلا رنج عدد های یک فیچر خیلی بزرگ نباشد و تاثیر بیشتری بگذارد)

در اینجا میتوان با `MinMaxScaler` این کار را انجام داد ولی چون هر ۲ فیچر در یک رنج میباشند

اگر نرمالسازی هم انجام نشود باز هم نتایج مثل شرایط با نرمالسازی میشود.

- تقسیم داده ها به داده های آموزش و تست :

به نسبت ۸۰ به ۲۰ داده ها را جدا میکنیم.

```
# Split Data to Train and Test ( 80% train 20% test )
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

شکل (۱-۱۴) تقسیم داده ها به داده های آموزش و تست

۱-۴-۱-۲- توضیح کلی کرنل های خطی و `poly` و `rbf`

در (SVM)، انتخاب کرنل نقش مهمی در فرآیند طبقه‌بندی بازی می‌کند. تابع کرنل داده های ورودی را به

یک فضای ویژگی با ابعاد بالاتر تبدیل می‌کند، جایی که داده ها به صورت خطی قابل تفکیک می‌شوند. سه

تابع کرنل که معمولاً مورد استفاده قرار می‌گیرند عبارتند از: کرنل خطی، کرنل چند جمله‌ای، و کرنل تابع پایه

شعاعی (RBF).

کرنل خطی:

کرنل خطی ساده ترین تابع کرنل است و زمانی که داده ها را بتوان با یک مرز تصمیم خطی جدا کرد، به

خوبی کار می‌کند. حاصل ضرب نقطه‌ای بین ویژگی‌های ورودی را محاسبه می‌کند. این کرنل از نظر محاسباتی

کارآمد است و به ویژه در هنگام برخورد با داده های با ابعاد بالا مفید است.

زمان استفاده: کرنل خطی زمانی موثر است که داده ها به صورت خطی قابل تفکیک باشند، یا زمانی که انتظار می رود مرز تصمیم یک خط مستقیم یا صفحه باشد.

کرنل چند جمله ای (poly):

تابع کرنل چند جمله ای اجازه می دهد تا مرزهای تصمیم گیری غیر خطی را با معرفی عبارت های چند جمله ای ویژگی های اصلی ایجاد کند. با معادله $d = (x * x')^r + \gamma * x'$ ، که در آن γ گاما یک ضریب مقیاس، r یک ثابت و d درجه چند جمله ای است.

زمان استفاده: کرنل چند جمله ای زمانی مناسب است که داده ها روابط غیرخطی پیچیده ای را نشان دهند.

پارامتر درجه، d ، انعطاف پذیری مرز تصمیم را کنترل می کند. مقادیر بالاتر d می تواند منجر به بیش از حد برآش (over fit) شود، در حالی که مقادیر کمتر ممکن است منجر به عدم تناسب شود. تنظیم دقیق پارامتر درجه بر اساس ویژگی های داده خاص مهم است.

کرنل (RBF) Radial Basis Function

کرنل RBF به دلیل توانایی آن در مدل سازی مرزهای تصمیم گیری پیچیده، یک انتخاب محبوب در SVM است. داده ها را در یک فضای بینهای بعدی نگاشت می کند و با معادله زیر تعریف می شود، که در آن گاما پارامتری است که تأثیر را از هر نمونه آموزشی تعیین می کند.

$$K(x, x') = \exp(-\gamma * \|x - x'\|^2)$$

زمان استفاده: کرنل RBF برای طیف گسترده ای از الگوهای داده مناسب است و اغلب انتخاب پیش فرض برای SVM است. در مواردی که داده ها جداسازی خطی یا چند جمله ای واضحی ندارند، عملکرد خوبی دارد.

با این حال، کرنل RBF در مقایسه با کرنل های خطی و چند جمله ای می تواند از نظر محاسباتی سنگین و پیچیده تر باشد و پارامتر گاما باید به دقت تنظیم شود تا از برآش بیش از حد جلوگیری شود.

انتخاب بهترین کرنل:

انتخاب کرنل به ویژگی های خاص داده ها و مشکل موجود بستگی دارد. اگر داده ها به صورت خطی قابل تفکیک هستند، کرنل خطی انتخاب خوبی است. اگر داده ها الگوهای غیر خطی را نشان دهند، می توان کرنل های چند جمله ای یا RBF را در نظر گرفت. کرنل چند جمله ای زمانی موثر است که درجه غیرخطی بودن مشخص باشد یا بتوان آن را تخمین زد. از سوی دیگر، کرنل RBF انعطاف پذیرتر است و می تواند طیف وسیع تری از روابط غیر خطی را مدیریت کند.

در نهایت، آزمایش با کرنل های مختلف و تنظیم پارامترهای آنها با استفاده از تکنیک هایی مانند اعتبار سنجی متقابل برای تعیین بهترین کرنل برای یک مجموعه داده خاص توصیه می شود.

4-1-3- کد فیت شبکه با کرنل های متفاوت

برای جلوگیری از پیچیدگی در یک for خروجی های مورد نظر اعم از رسم مرز و precison و F1 و recall و... را به ازای هر کرنل به کمک کد زیر به دست میاوریم. که بخش ها مختلف کد در زیر توضیح داده شده است.

```
# Define the kernel names and corresponding kernel classes
kernels = ['linear', 'rbf', 'poly']
kernel_classes = [SVC(kernel='linear'), SVC(kernel='rbf'), SVC(kernel='poly')]
```

شکل (1-15) تعریف کرنل های مدنظر

```

# Loop over the kernels
for kernel_name, kernel_class in zip(kernels, kernel_classes):
    # Perform classification using the current kernel
    classifier = kernel_class
    classifier.fit(X_train, y_train)

    # Generate predictions
    y_pred = classifier.predict(X_test)

    # Calculate and print the confusion matrix
    print(f"Confusion Matrix ({kernel_name} kernel):")
    print(confusion_matrix(y_test, y_pred))
    print()

    # Calculate and print the classification report (including precision, recall, and F1-score)
    print(f"Classification Report ({kernel_name} kernel):")
    print(classification_report(y_test, y_pred))
    print()

```

شکل (۱-۱۶) فیت کردن با کرنل مورد و خروجی گرفتن روی داده تست و گزارش متريک ها

```

# Function to plot the decision boundaries
def plot_decision_boundary(X, y, classifier):
    h = 0.02 # step size in the mesh

    # Create a mesh grid of points
    x_min, x_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
    y_min, y_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

    # Make predictions for each point in the mesh grid
    Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot the contour lines and data points
    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=y, cmap=plt.cm.Set1)
    plt.xlabel('SepalLengthCm')
    plt.ylabel('SepalWidthCm')
    plt.title(f'{kernel_name.capitalize()} SVM Decision Boundary')
    plt.show()

# Plot the decision boundaries
plot_decision_boundary(X_test, y_test, classifier)
print('-' * 40)

```

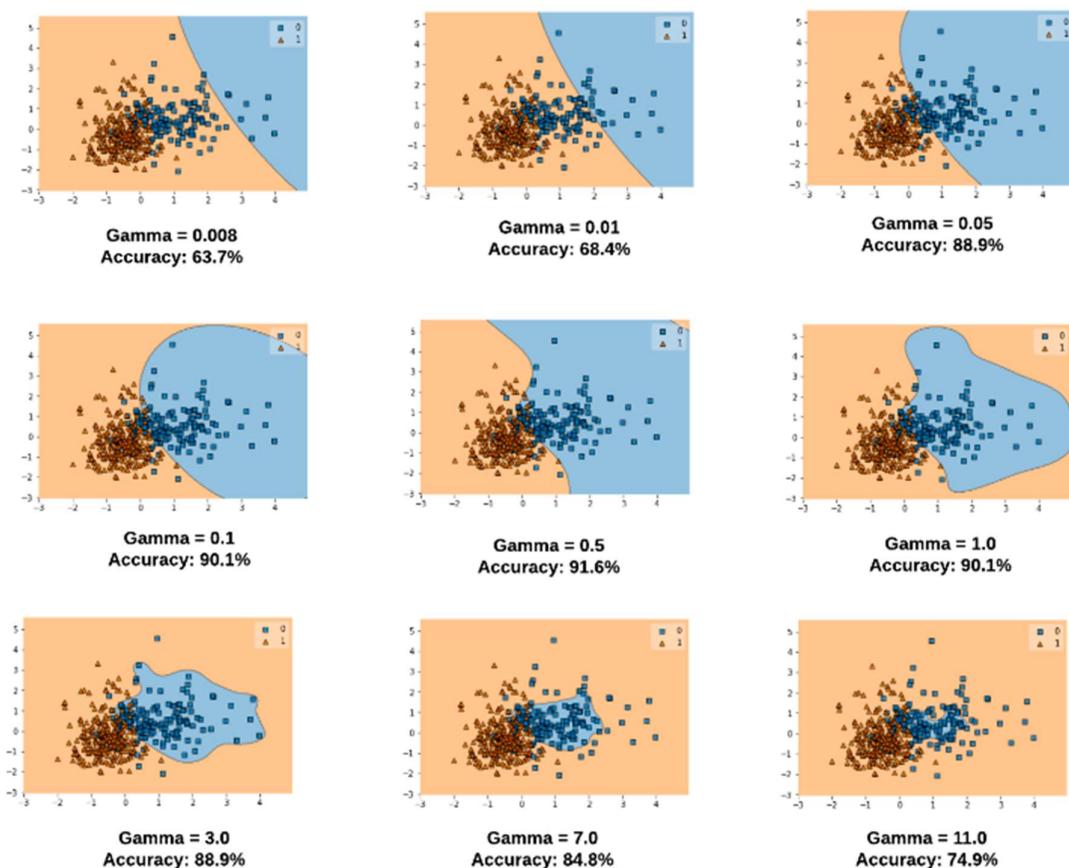
شکل (۱-۱۷) رسم مرز جدا شونده به ازای هر کرنل

1-4-1-4- مقایسه و توضیحات هایپر پارامتر های گاما و C (regularization)

پارامتر گاما :

پارامتر گاما تعیین می کند که تأثیر یک داده آموزشی به چه اندازه می رسد، با مقادیر کم به معنای "دور" و مقادیر بالا به معنای "نزدیک". مقادیر کمتر گاما منجر به مدل هایی با دقت کمتر و همان مقادیر بالاتر گاما می شود. این مقادیر میانی گاما است که مدلی با مرزهای تصمیم خوب ارائه می دهد. همین امر در نمودارهای داده شده در شکل زیر نشان داده شده است.

(توجه شود که با افزایش مقدار گاما، مرزهای تصمیم گیری نقاط را به درستی طبقه بندی می کنند. با این حال، پس از یک نقطه خاص دقت مدل کاهش می یابد. بنابراین می توان فهمید که انتخاب مقادیر مناسب گاما مهم است).



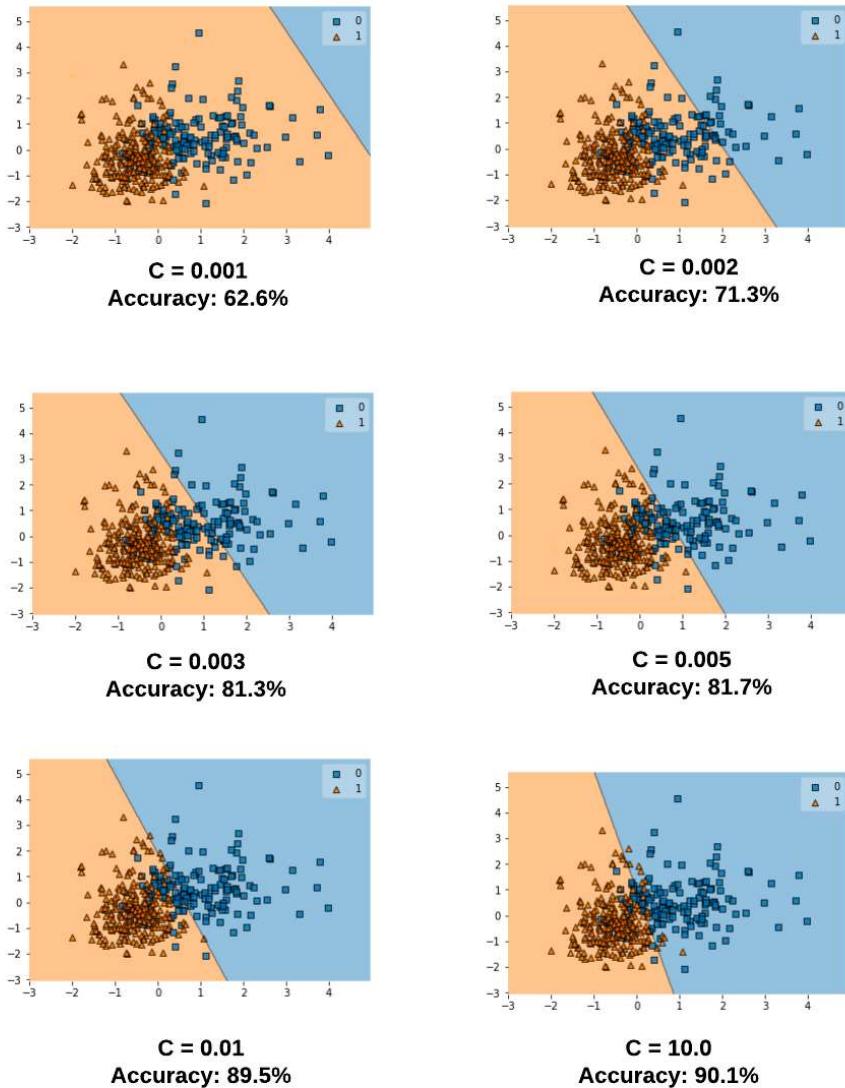
شکل (1-۱۸) بررسی مقادیر مختلف گاما در مرز تصمیم (شکل از سایت vitalflux)

با توجه به شکل فوق میتوان گفت:

- هنگامی که گاما بسیار کوچک است (۰.۰۰۸ یا ۰.۰۱)، مدل بسیار محدود است و نمی تواند پیچیدگی یا "شکل" داده ها را ثبت کند.
- برای مقادیر میانی گاما (۰.۰۵، ۰.۱، ۰.۵)، می توان در نمودار دوم مشاهده کرد که مدل های خوبی را می توان یافت.

پارامتر (C) regularization :

پارامتر C یک پارامتر است که برای تنظیم تلوانس مدل (پارامتر منظم سازی) استفاده می شود تا امکان طبقه‌بندی اشتباه نقاط داده به منظور دستیابی به خطای تعمیم کمتر را فراهم کند.



شکل (۱-۱۹) بررسی هایپرپارامتر C

هر چه مقدار C بالاتر باشد، تحمل کمتر است و آنچه آموزش داده می شود یک طبقه بندی کننده

است. مقدار C کوچکتر، تحمل طبقه بندی اشتباه بزرگتر است و آنچه آموزش داده

می شود یک طبقه بندی soft margin است که بهتر از طبقه بندی کننده hard margin تعیین می یابد.

مقدار C مجازات (پنالتی) طبقه بندی اشتباه را کنترل می کند. مقدار زیاد C منجر به جریمه بالاتر برای

طبقه بندی اشتباه و مقدار کمتر C منجر به جریمه کمتر طبقه بندی اشتباه می شود. با مقدار C بزرگتر،

اگر تابع تصمیم گیری در طبقه بندی صحیح تمام نقاط آموزشی بهتر باشد، حاشیه کمتری پذیرفته می

شود. مدل ممکن است با مجموعه داده آموزشی بیش از حد سازگار شود. C پایین تر، حاشیه بزرگتر را

تشویق می کند، بنابراین یک تابع تصمیم گیری ساده تر داریم.

۱-۴-۱-۵- کد قرار دادن هایپر پارامتر های متفاوت و انتخاب بهترین آن

به کمک کد زیر میتوان مقدار C و γ های متفاوت را به rbf داده و نتایج مختلف را گزارش کنیم.

```
# Define the gamma and C values
gamma_values = [0.1, 1, 10]
C_values = [0.1, 1, 10]

# Perform classification using the rbf kernel with different gamma and C values
for gamma in gamma_values:
    for C in C_values:
        print(f"Gamma: {gamma}, Regularization (C): {C}")

        # Create the classifier with the current gamma and C values
        classifier = SVC(kernel='rbf', gamma=gamma, C=C)
        classifier.fit(X_train, y_train)

        # Generate predictions
        y_pred = classifier.predict(X_test)
```

شکل (۱-۲۰) کد مقدار دهی های مختلف C و γ برای rbf و SVM

باقی بخش های کد جهت نمایش خروجی مثل کد قبلی میباشد.

تنظیم پارامتر های یک مدل SVM با استفاده از کرنل RBF برای دستیابی به عملکرد خوب بسیار مهم است.

روش های مختلفی برای تنظیم این پارامترها از جمله جستجوی شبکه ای (grid) و جستجوی تصادفی وجود

دارد.

جستجوی شبکه (grid) شامل تعریف شبکه‌ای از مقادیر ممکن برای گاما و C و ارزیابی عملکرد مدل برای هر ترکیبی از مقادیر پارامتر است. سپس مقادیر پارامتر بهینه بر اساس معیارهای عملکرد، مانند دقت یا میانگین مربعات خطأ، در یک مجموعه نگهدارنده یا با استفاده از اعتبارسنجی متقابل انتخاب می‌شوند.

جستجوی تصادفی یک رویکرد جایگزین است که در آن مقادیر پارامتر به طور تصادفی از محدوده مشخصی از مقادیر نمونه برداری می‌شود. این روش می‌تواند کارآمدتر از جستجوی شبکه‌ای باشد که فضای پارامتر بزرگ باشد، زیرا نیازی به ارزیابی مدل برای همه ترکیبات ممکن از مقادیر پارامتر ندارد.

هنگام تنظیم پارامترهای یک مدل SVM با استفاده از کرنل RBF، استفاده از محدوده مناسبی از مقادیر پارامتر مهم است. مقدار گاما معمولاً باید در محدوده‌ای از مقادیر که شامل مقادیر کوچک و بزرگ است تغییر کند، در حالی که مقدار C باید در محدوده‌ای از مقادیر که شامل مقادیر کوچک و بزرگ نیز می‌شود، تغییر کند.

در اینجا از انجا که بیشتر پیاده سازی و کد مد نظر است و جهت جلوگیری از زیاد شدن ران تایم و اینکه در ادامه خواهید دید دقت تقریباً 100 است همان 3 تا مقادیر بخش قبل را با جستجوی شبکه‌ای بهترینش را انتخاب می‌کنیم.

کد پایتون این بخش به صورت زیر مبادله که در نهایت در این کد بهترین هایپرپارامتر مشخص می‌شود (یک بخش از کد که کامنت شده توجه شود می‌توان در یک رنج‌های به صورت فوق هم به دنبال هایپرپارامتر های مد نظر گشت)

```

# Define the parameter grid for GridSearchCV
#param_grid = {'gamma': np.arange(0, 10, 0.5), 'C': np.arange(0, 20, 0.1)}
param_grid = {'gamma': [0.1, 1, 10], 'C': [0.1, 1, 10]}
# Create the classifier
classifier = SVC(kernel='rbf')

# Perform grid search to find the best parameters
grid_search = GridSearchCV(classifier, param_grid, cv=3)
grid_search.fit(X_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Print the best parameters
print("Best Parameters:")
print(best_params)
print()

# Generate predictions using the best model
y_pred = best_model.predict(X_test)

```

شکل (۱-۲۱) پیاده سازی جستجوی شبکه ای

۱-۴-۱-۶ ovo و ovr کد ها رویکرد

در SVM، هنگام برخورد با کلاس های متعدد، از دو استراتژی رایج استفاده می شود: یک در مقابل بقیه و یک در مقابل یک. این استراتژی ها به SVM اجازه می دهد تا وظایف طبقه بندی را باشیش از دو کلاس انجام دهد. انتخاب کرنل، مانند خطی، چند جمله ای، یا RBF، می تواند با هر دو استراتژی اعمال شود.

یک در مقابل بقیه (یک در برابر همه):

در استراتژی one-vs-rest، یک طبقه بندی کننده باینری برای هر کلاس آموزش داده می شود که آن را به عنوان کلاس مثبت و کلاس های باقی مانده را به عنوان کلاس منفی در نظر می گیرد. این منجر به K طبقه بندی کننده های باینری می شود، که در آن K تعداد کلاس ها است. در حین پیش بینی، کلاس مرتبط با طبقه بندی کننده با بالاترین امتیاز اطمینان انتخاب می شود.

هنگام استفاده از کرنل های مختلف:

- کرنل خطی: برای هر کلاس، یک طبقه بندی کننده SVM خطی با استفاده از کرنل خطی آموزش داده می شود. کرنل خطی زمانی به خوبی کار می کند که کلاس ها با یک مرز تصمیم خطی به خوبی از هم جدا شوند. مرز تصمیم یک ابر صفحه در فضای ویژگی است.
- کرنل چند جمله ای: هر کلاس با استفاده از کرنل چند جمله ای در برابر بقیه آموزش داده می شود. کرنل چند جمله ای اصطلاحات غیرخطی را برای ثبت روابط پیچیده تر بین ویژگی ها معرفی می کند. درجه چند جمله ای باید به دقت بر اساس داده ها تنظیم شود.
- کرنل RBF: مشابه کرنل چند جمله ای، هر کلاس با استفاده از کرنل RBF در برابر بقیه آموزش داده می شود. کرنل RBF در گرفتن مرزهای تصمیم گیری غیرخطی پیچیده موثر است. پارامتر گاما باید به طور مناسب تنظیم شود تا از برازش بیش از حد جلوگیری شود.

یک در مقابل یک:

در استراتژی یک در مقابل یک، یک طبقه بندی کننده باینری برای هر جفت کلاس آموزش داده می شود. اگر کلاس K وجود داشته باشد، $2 / (K - 1) * K$ طبقه بندی آموزش داده می شود. در حین پیش بینی، هر طبقه بندی کننده یک رای می دهد و کلاسی که بیشترین آرا را داشته باشد به عنوان پیش بینی نهایی انتخاب می شود.

انتخاب بین یک در مقابل بقیه و یک در مقابل یک:

انتخاب بین این استراتژی ها به عوامل مختلفی بستگی دارد. یک در مقابل بقیه معمولاً زمانی ترجیح داده می شود که تعداد کلاس ها زیاد باشد، زیرا فقط شامل آموزش طبقه بندی کننده های باینری K است. یک در مقابل یک می تواند از نظر محاسباتی گران باشد زیرا نیاز به آموزش $2 / (K - 1) * K$ طبقه بندی کننده دارد. با این

حال، زمانی که کلاس‌ها نامتعادل هستند یا زمانی که مرزهای تصمیم‌گیری بین کلاس‌ها پیچیده و با هم تداخل دارند، یک در مقابل یک می‌تواند دقت بهتری ارائه دهد.

آزمایش با استراتژی‌ها و کرنل‌های مختلف برای یافتن مناسب‌ترین رویکرد برای یک مسئله طبقه‌بندی چند طبقه خاص، مهم است. علاوه بر این، تنظیم فراپارامترهای SVM، مانند پارامتر تنظیم (C) و پارامترهای کرنل، می‌تواند به طور قابل توجهی بر عملکرد طبقه‌بندی کننده‌ها تأثیر بگذارد. در کد نیز به صورت زیر می‌توان 2 رویکرد گفته شده را بررسی کرد.

```
# Define the kernels and approaches
kernels = ['rbf', 'linear', 'poly']
approaches = ['ovr', 'ovo']

# Perform classification for each kernel and approach
for kernel in kernels:
    for approach in approaches:
        print(f"Kernel: {kernel}, Approach: {approach}")

    # Create the classifier with the current kernel and approach
    classifier = SVC(kernel=kernel, degree=3, decision_function_shape=approach)
    classifier.fit(X_train, y_train)

    # Generate predictions
    y_pred = classifier.predict(X_test)
```

شکل (۱-۲۲) کد بررسی رویکرد‌های ovr و ovo

1-4-2- نتایج برای داده‌های 2 کلاسه Sepal

با توجه به توضیحات و مقایسه‌ها و تمام کد‌ها که در بالا ذکر شد بدون تکرار اضافات به نتایج هر بخش می‌پردازیم.

ابتدا 2 ویژگی اول را فقط استفاده می‌کنیم.

1-4-2-1- بررسی کرnel های مختلف

** کرnel خطی :

در این حالت Classification Report و ماتریس آشتفتگی و مرز تصمیم و ... به شرح زیر میباشد.

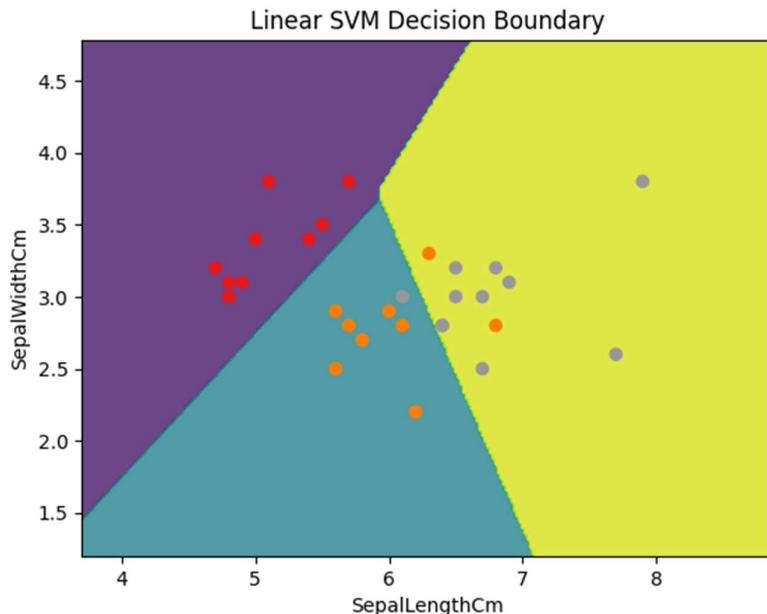
Confusion Matrix (linear kernel):

```
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
```

Classification Report (linear kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شکل (۱-۲۳) ماتریس آشتفتگی برای کرnel خطی (Sepal) Classification Report



شکل (۱-۲۴) مرز جداسازی با استفاده از کرnel خطی (Sepal)

: Poly کرنل **

در این حالت Classification Report و ماتریس آشتفتگی و مرز تصمیم و ... به شرح زیر میباشد.

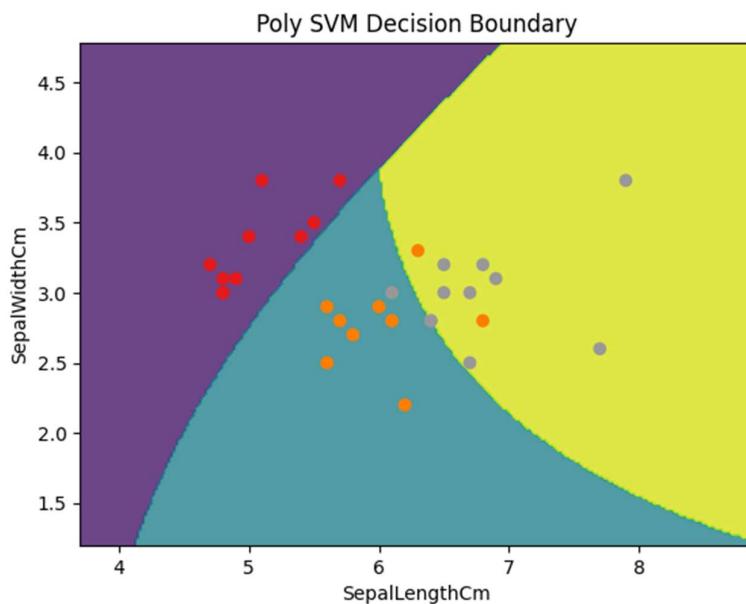
Confusion Matrix (poly kernel):

```
[[10  0  0]
 [ 0  7  2]
 [ 0  3  8]]
```

Classification Report (poly kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.70	0.78	0.74	9
2	0.80	0.73	0.76	11
accuracy			0.83	30
macro avg	0.83	0.84	0.83	30
weighted avg	0.84	0.83	0.83	30

شکل (۱-۲۵) (Sepal) poly و ماتریس آشتفتگی برای کرنل Classification Report



شکل (۱-۲۶) (Sepal) poly مرز جداسازی با استفاده از کرنل

: rbf کرنل **

در این حالت Classification Report و ماتریس آشتفتگی و مرز تصمیم و ... به شرح زیر میباشد.

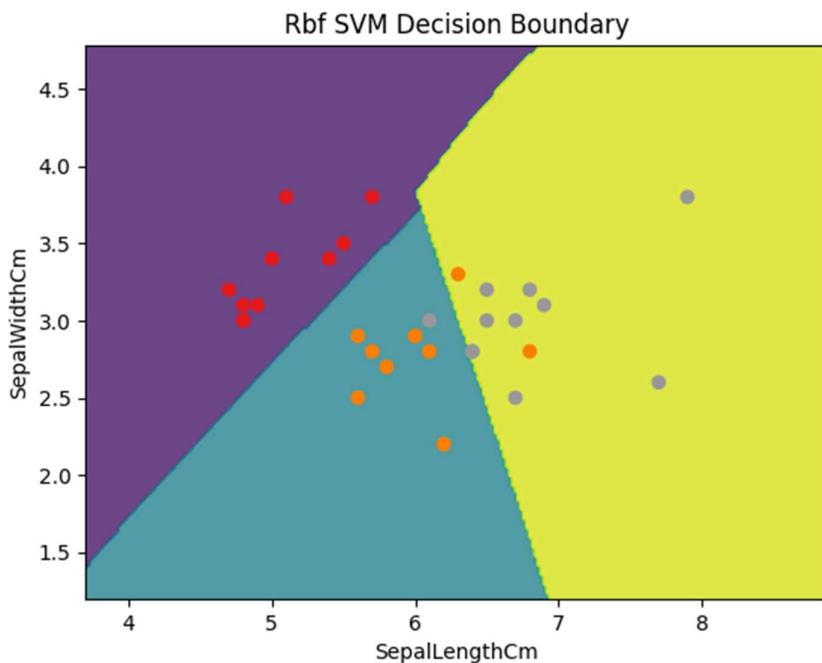
Confusion Matrix (rbf kernel):

```
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
```

Classification Report (rbf kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شکل (۱-۲۷) Classification Report و ماتریس آشتفتگی برای کرنل rbf (Sepal)



شکل (۱-۲۸) مرز جداسازی با استفاده از کرنل rbf (Sepal)

مشاهده میشود که در این حالت کرنل rbf و خطی عملکرد مشابه داشتند و دقت 88 است ولی استفاده از کرنل polynimal به دقت پایین تری رسیده است و دلیل آن است که در اینجا به دلیل توزیع داده ها خط کافیست.

2-4-2-1- بررسی هایپر پارامتر های مختلف

نتایج این بخش به ترتیب در تصاویر زیر آمده است که توضیحات کامل مقادیر هایپر پارامتر ها روی تصاویر

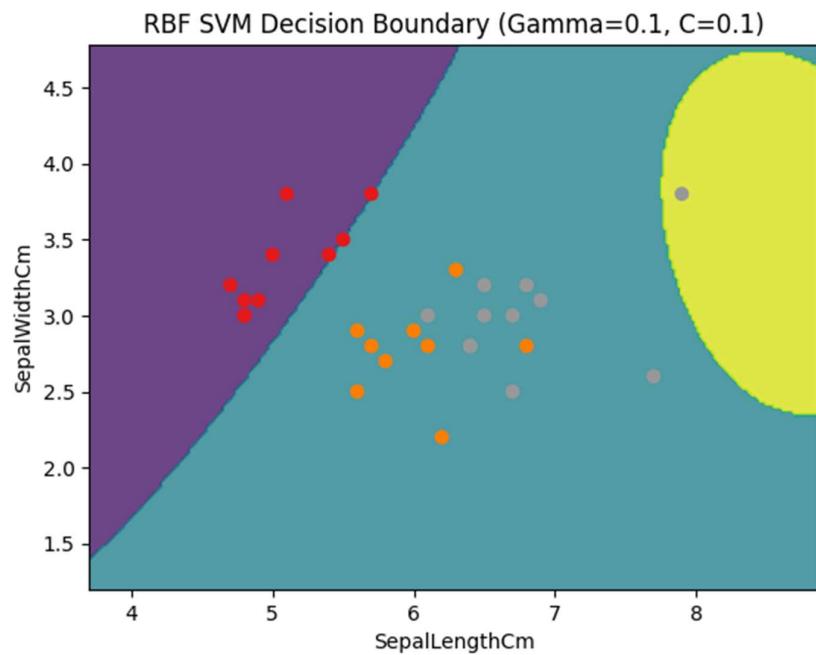
آمده است. (نام تصاویر از 1 تا 9 نامگذاری شده است)

```
Gamma: 0.1, Regularization (C): 0.1
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0 10  1]]

Classification Report:
precision    recall    f1-score   support
          0       1.00      1.00      1.00      10
          1       0.47      1.00      0.64       9
          2       1.00      0.09      0.17      11

accuracy                           0.67      30
macro avg       0.82      0.70      0.60      30
weighted avg    0.84      0.67      0.59      30
```

شکل (1-۲۹) 1



شكل (١-٣٠) ١

Gamma: 0.1, Regularization (C): 1

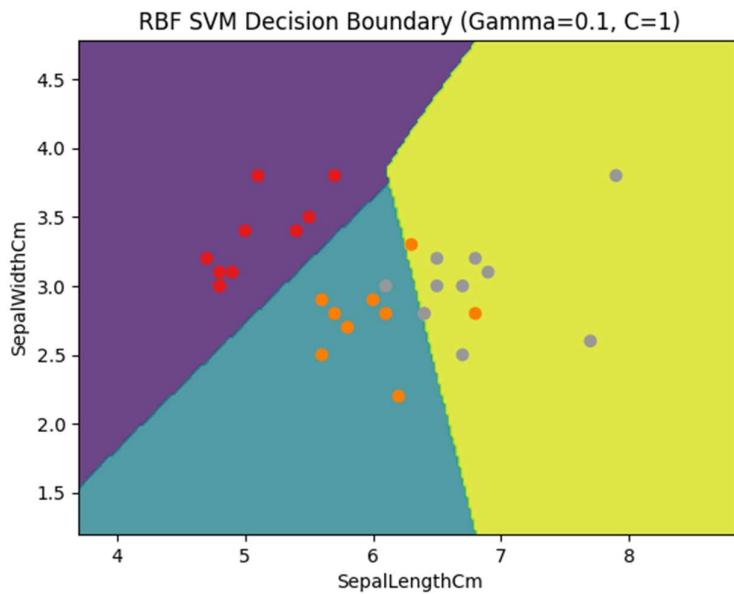
Confusion Matrix:

```
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شكل (١-٣١) ٢



شكل (١-٣٢) ٢

Gamma: 0.1, Regularization (C): 10

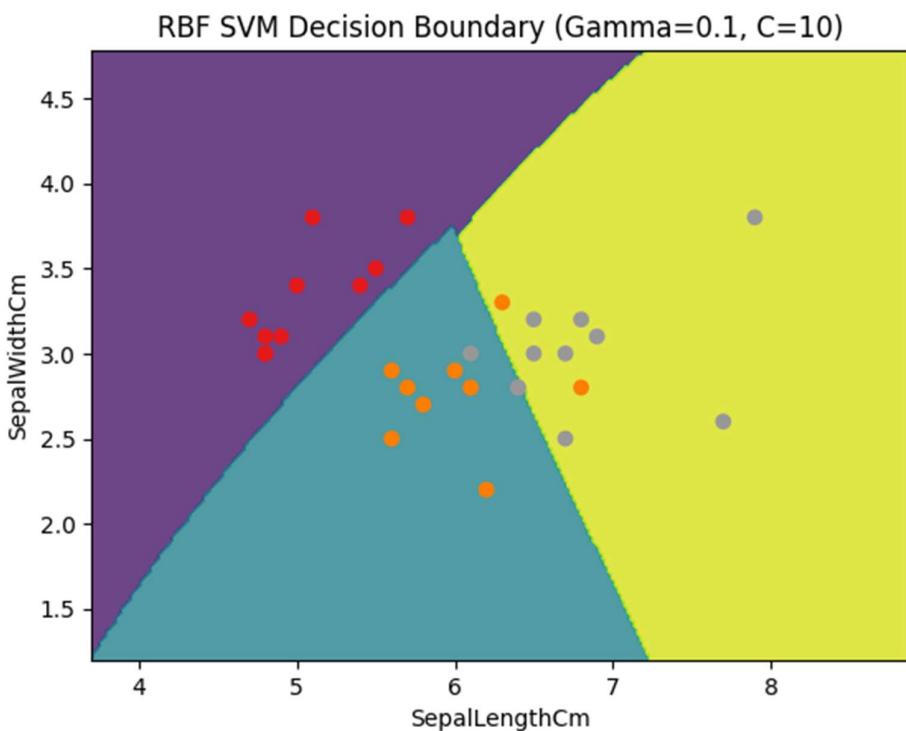
Confusion Matrix:

```
[[10  0  0]
 [ 0  7  2]
 [ 0  3  8]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.70	0.78	0.74	9
2	0.80	0.73	0.76	11
accuracy			0.83	30
macro avg	0.83	0.84	0.83	30
weighted avg	0.84	0.83	0.83	30

شكل (١-٣٣) ٣



شكل (١-٣٤) ٣

Gamma: 1, Regularization (C): 0.1

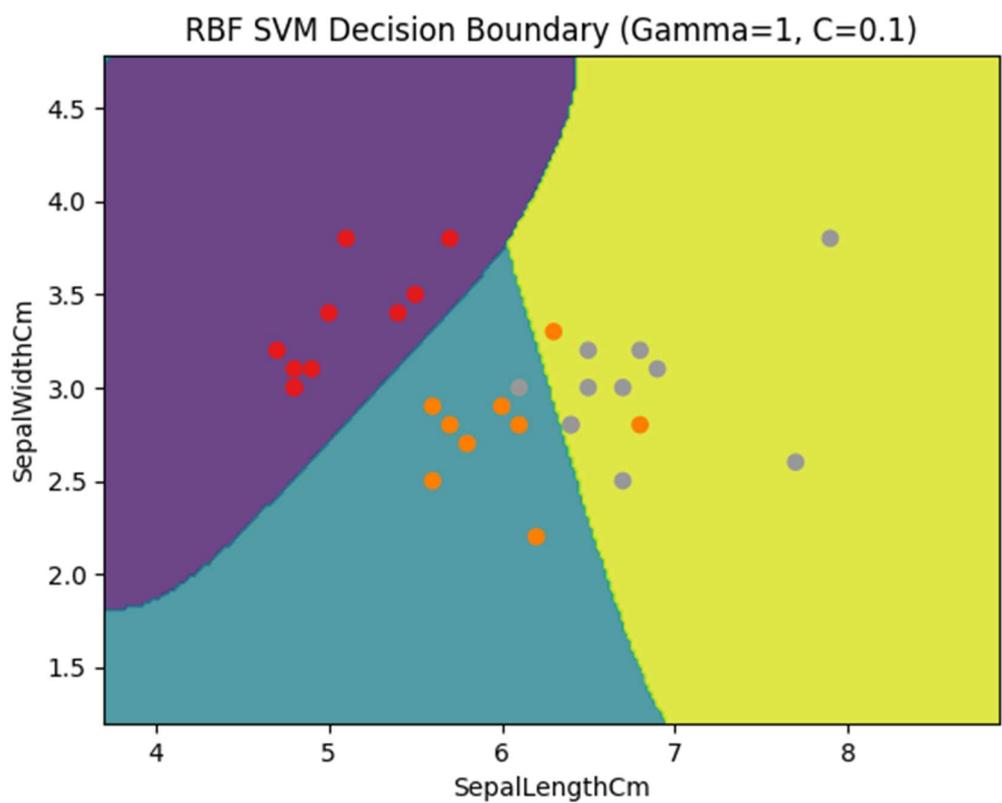
Confusion Matrix:

```
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شكل (١-٣٥) ٤



شكل (٤-٣٦)

Gamma: 1, Regularization (C): 1

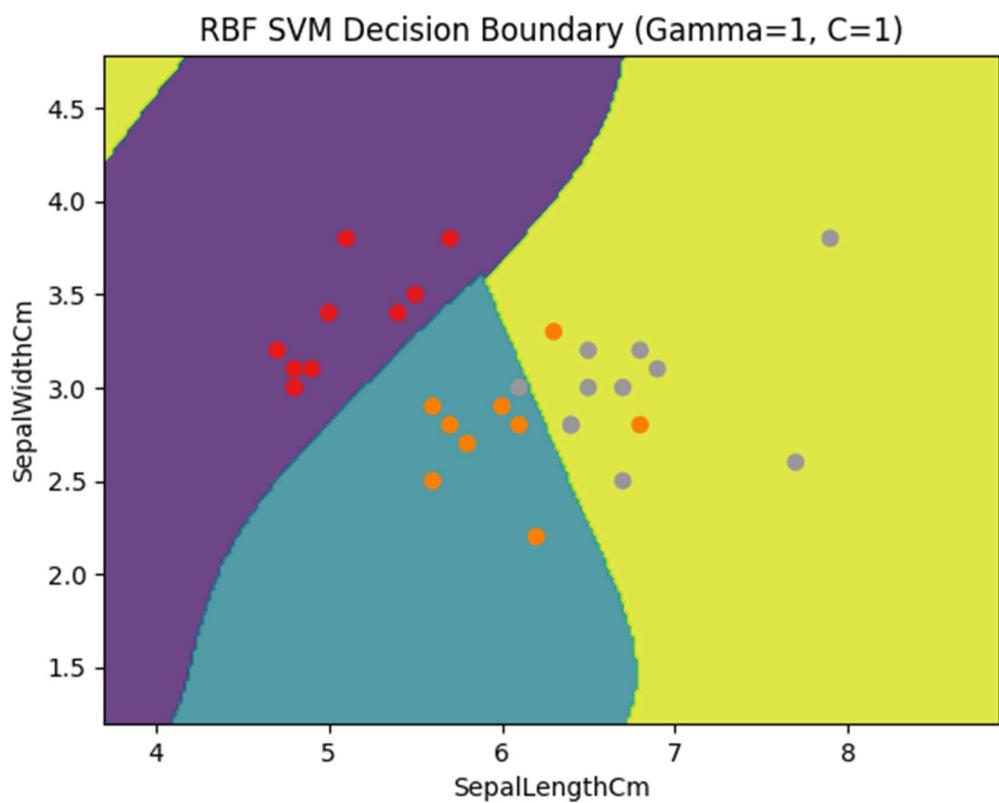
Confusion Matrix:

```
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شكل (٤-٣٧)



شكل (١-٣٨)

Gamma: 1, Regularization (C): 10

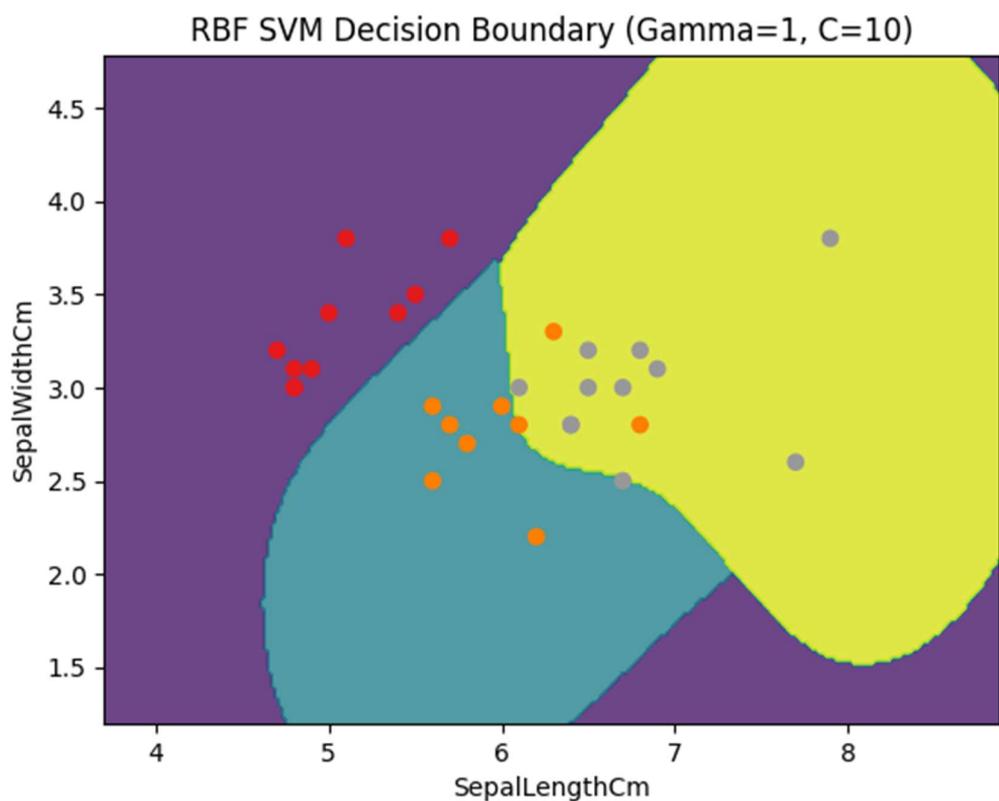
Confusion Matrix:

```
[[10  0  0]
 [ 0  6  3]
 [ 0  1 10]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.86	0.67	0.75	9
2	0.77	0.91	0.83	11
accuracy			0.87	30
macro avg	0.88	0.86	0.86	30
weighted avg	0.87	0.87	0.86	30

شكل (١-٣٩)



شكل (٤٠-٤١)

Gamma: 10, Regularization (C): 0.1

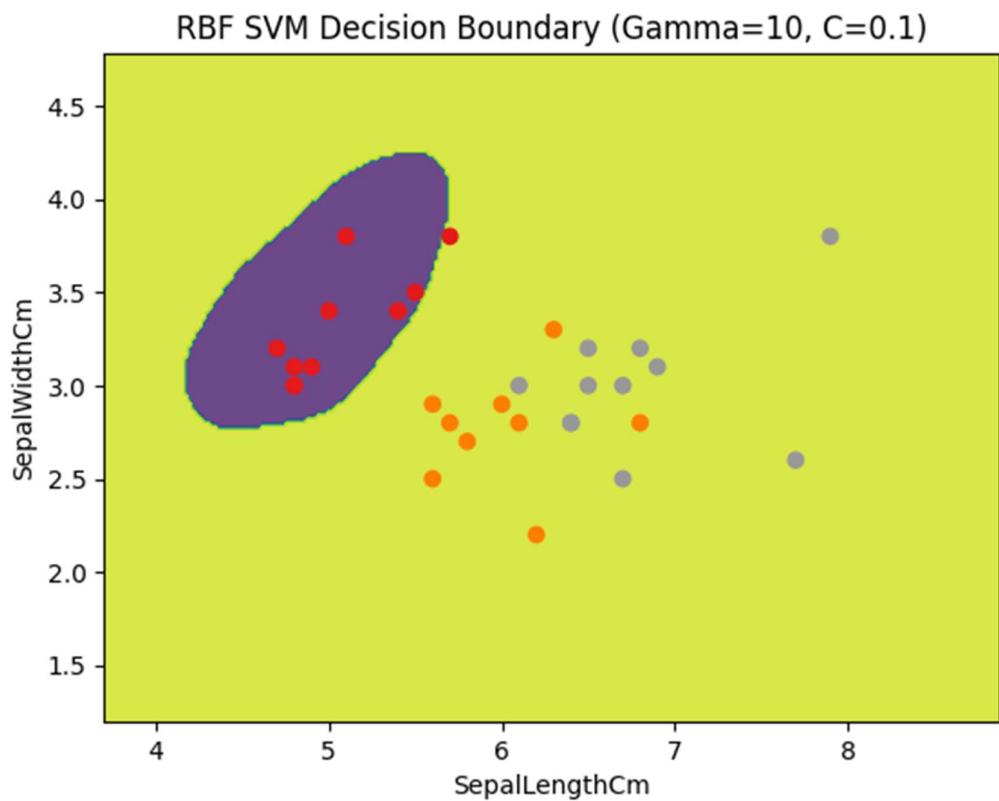
Confusion Matrix:

```
[[ 9  1  0]
 [ 0  9  0]
 [ 0 11  0]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	0.90	0.95	10
1	0.43	1.00	0.60	9
2	0.00	0.00	0.00	11
accuracy			0.60	30
macro avg	0.48	0.63	0.52	30
weighted avg	0.46	0.60	0.50	30

شكل (٤١-٤٢)



شكل (١-٤٢) 7

Gamma: 10, Regularization (C): 1

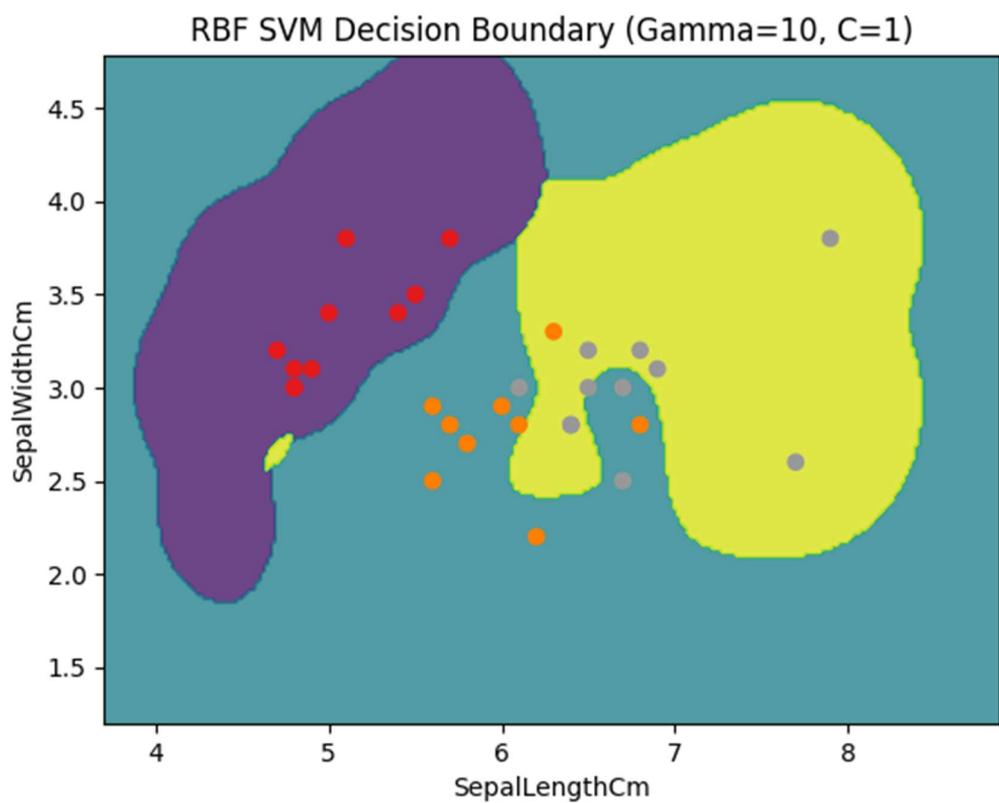
Confusion Matrix:

```
[[10  0  0]
 [ 0  8  1]
 [ 0  4  7]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.67	0.89	0.76	9
2	0.88	0.64	0.74	11
accuracy			0.83	30
macro avg	0.85	0.84	0.83	30
weighted avg	0.85	0.83	0.83	30

شكل (١-٤٣) 8



شكل (١-٤٤) ٨

Gamma: 10, Regularization (C): 10

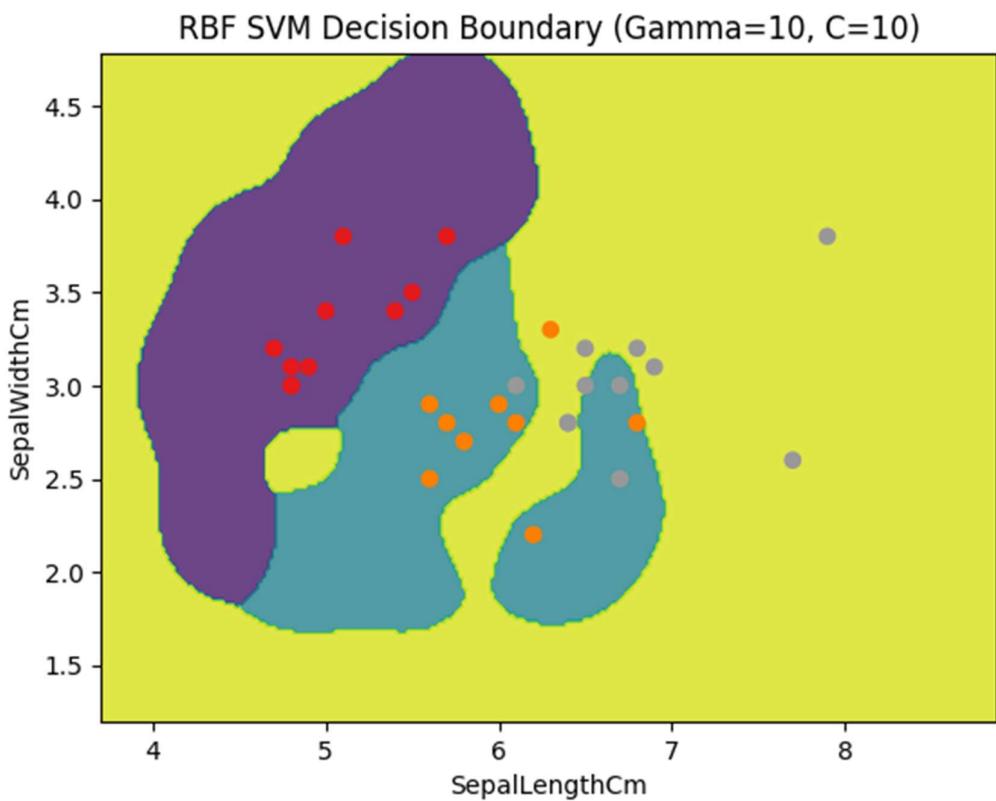
Confusion Matrix:

```
[[10  0  0]
 [ 0  8  1]
 [ 0  3  8]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.73	0.89	0.80	9
2	0.89	0.73	0.80	11
accuracy			0.87	30
macro avg	0.87	0.87	0.87	30
weighted avg	0.88	0.87	0.87	30

شكل (١-٤٥) ٩



شکل (۱-۴۶) ۹

به صورت چشمی هم مواردی که در بخش قبل در توضیخات Γ و C بیان شده بود ملاحظه شد و دیده میشود که Γ برابر ۰.۱ و $C=1$ بهترین نتیجه را دارد.

۱-۴-۲-۳- جستجوی شبکه ای پیدا کردن بهترین هایپرپارامتر

پس از جستجوی شبکه ای به نتایج زیر رسیدیم:

```
Best Parameters:  
{'C': 1, 'gamma': 0.1}
```

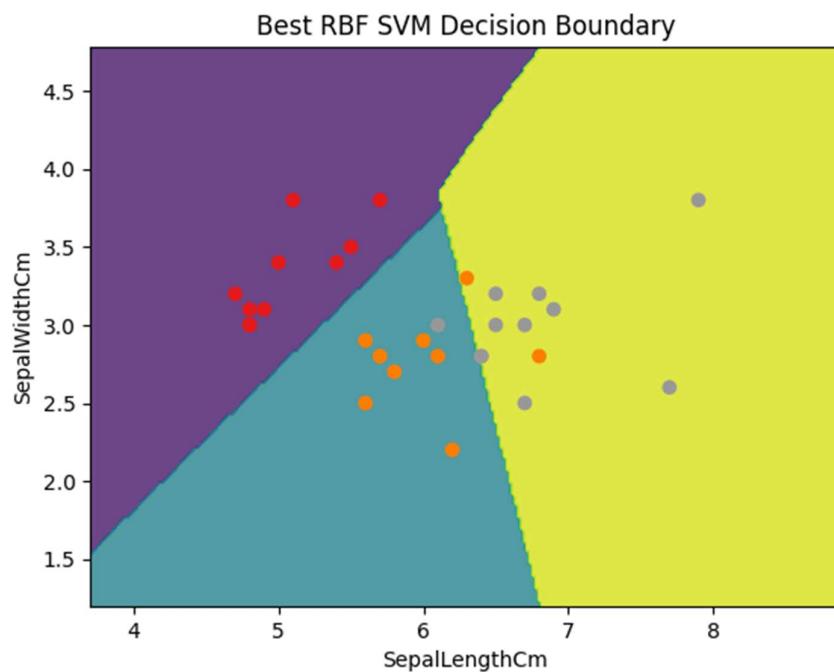
```
Confusion Matrix:
```

```
[[10  0  0]  
 [ 0  7  2]  
 [ 0  1 10]]
```

```
Classification Report:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شکل (۱-۴۷) بهترین هایپر پارامتر پس از جستجوی شبکه ای و گزارش آن (sepal) classification



شکل (۱-۴۸) مرز تصمیم rbf با بهترین پارامتر ها (sepal)

که خروجی جست و جوی شبکه ای با مشاهدات ما تطابق دارد.

بررسی 2 رویکرد ovo و ovr

بررسی رویکرد های مختلف روی کرنل های متفاوت به شرح زیر میباشد.

rbf کرنل **

Kernel: rbf, Approach: ovr

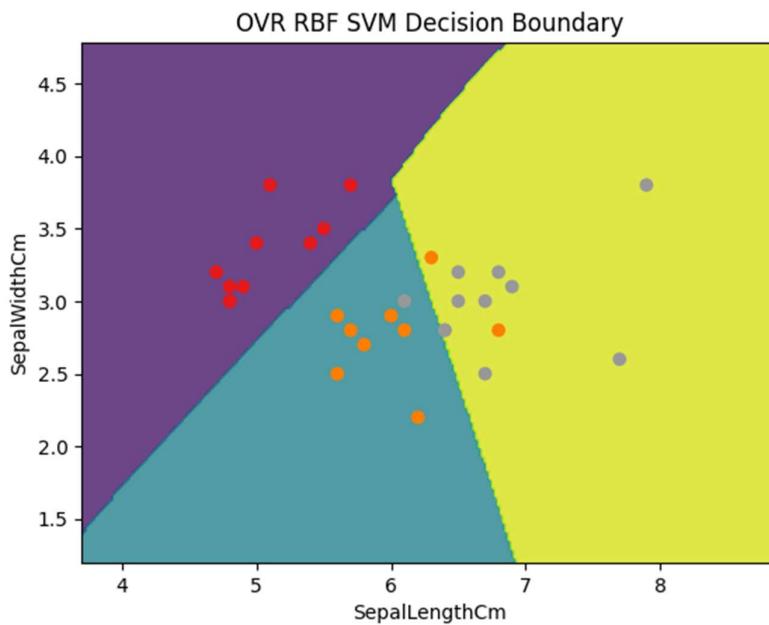
Confusion Matrix:

```
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.88	0.78	0.82	9
2	0.83	0.91	0.87	11
accuracy			0.90	30
macro avg	0.90	0.90	0.90	30
weighted avg	0.90	0.90	0.90	30

شکل (۱-۴۹) نتایج رویکرد ovr با کرنل (sepal) rbf



شکل (۱-۵۰) مرز تصمیم رویکرد ovr با کرnel rbf (sepal)

```

Kernel: rbf, Approach: ovo
Confusion Matrix:
[[10  0  0]
 [ 0  7  2]
 [ 0  1 10]]

Classification Report:
      precision    recall  f1-score   support
0       1.00    1.00    1.00      10
1       0.88    0.78    0.82       9
2       0.83    0.91    0.87      11

accuracy                           0.90      30
macro avg       0.90    0.90    0.90      30
weighted avg    0.90    0.90    0.90      30

```

شکل (۱-۵۱) نتایج رویکرد ovo در کرnel rbf (Sepal)

مشاهده میشود نتیجه 2 رویکرد در این حالت یکی میشود.

به طور کاملا مشابه نتیجه برای این دو رویکرد در linear و poly هم میشود و خروجی ها همان

خروجی های بخش 1 (دقت های به ترتیب 88 و 70 میباشد).

1-4-3- نتایج برای داده های 2 کلاسه Petal

حال 2 ویژگی دوم را فقط استفاده میکنیم.

1-4-3-1- بررسی کرنل های مختلف

** کرنل خطی :

در این حالت Classification Report و ماتریس آشتفتگی و مرز تصمیم و ... به شرح زیر میباشد.

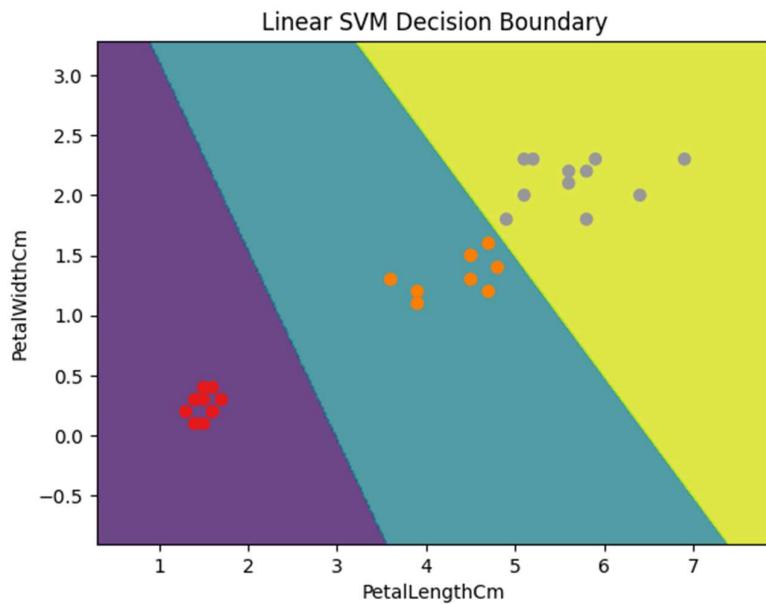
Confusion Matrix (linear kernel):

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report (linear kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شکل (1-۵۲) Classification Report و ماتریس آشتفتگی برای کرنل خطی (petal)



شکل (۱-۵۳) مرز جداسازی با استفاده از کرnel خطی (petal

: Poly ** کرnel

در این حالت Classification Report و ماتریس آشتفتگی و مرز تصمیم و ... به شرح زیر میباشد.

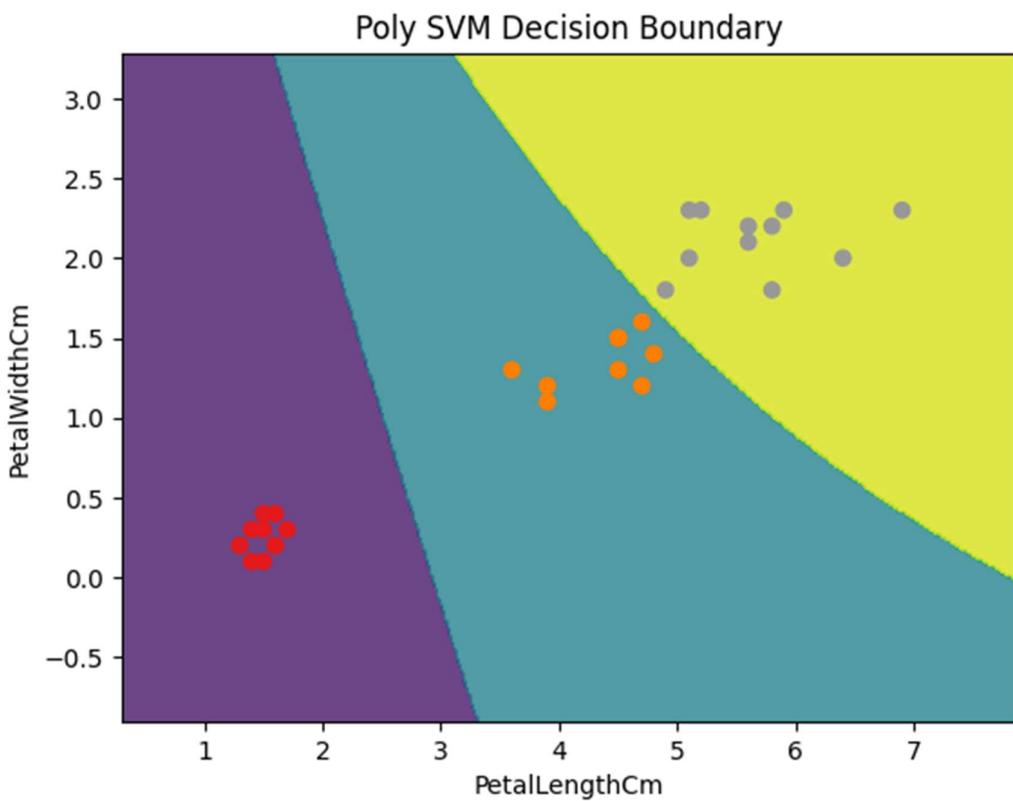
Confusion Matrix (poly kernel):

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report (poly kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شکل (۱-۵۴) Classification Report و ماتریس آشتفتگی برای کرnel poly (petal



شکل (۱-۵۵) مرز جداسازی با استفاده از کرnel (petal) poly

: rbf ** کرnel

در این حالت Classification Report و ماتریس آشتفتگی و مرز تصمیم و ... به شرح زیر میباشد.

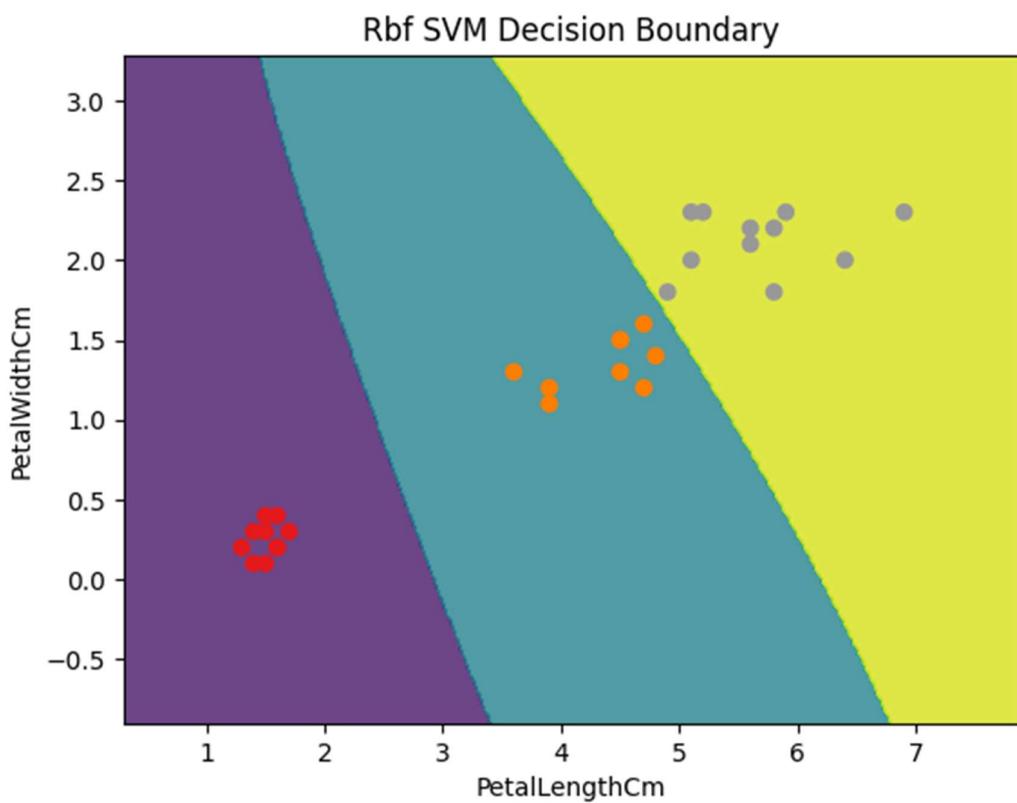
Confusion Matrix (rbf kernel):

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report (rbf kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شکل (۱-۵۶) Classification Report و ماتریس آشتفتگی برای کرnel (petal) rbf



شکل (1-۵۷) مرز جداسازی با استفاده از کرnel (rbf) (petal

مشاهده میشود که در این حالت در تمامی حالات دقت 100 درصد میباشد.

1-4-3-2- بررسی هایپر پارامتر های مختلف

نتایج این بخش به ترتیب در تصاویر زیر آمده است که توضیحات کامل مقادیر هایپر پارامتر ها روی تصاویر

آمده است. (نام تصاویر از 1 تا 9 نامگذاری شده است)

Gamma: 0.1, Regularization (C): 0.1

Confusion Matrix:

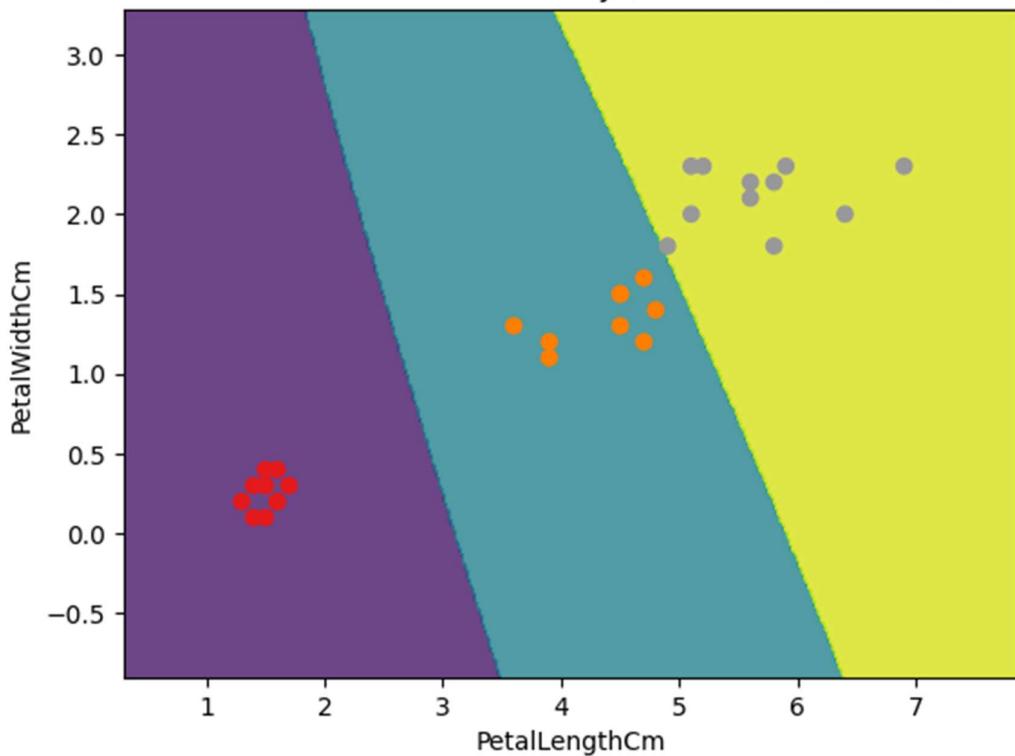
```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شكل (١-٥٨)

RBF SVM Decision Boundary (Gamma=0.1, C=0.1)



شكل (١-٥٩)

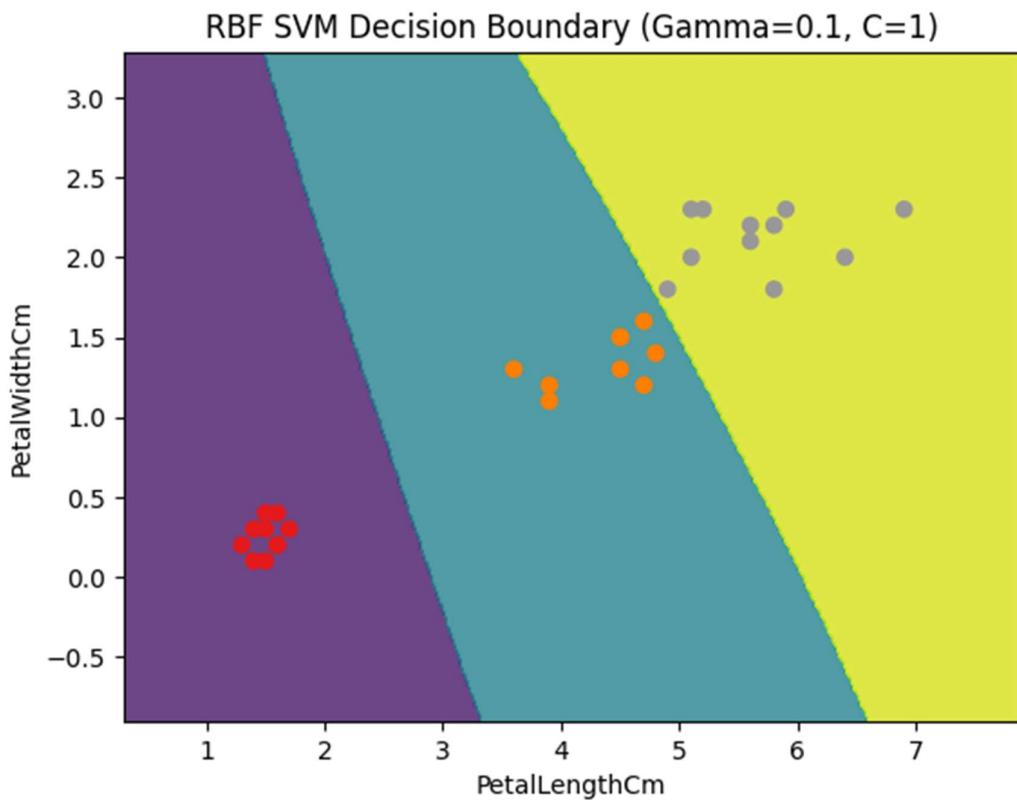
```
Gamma: 0.1, Regularization (C): 1
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
Classification Report:
precision    recall    f1-score   support

          0       1.00      1.00      1.00      10
          1       1.00      1.00      1.00       9
          2       1.00      1.00      1.00      11

   accuracy                           1.00      30
  macro avg       1.00      1.00      1.00      30
weighted avg       1.00      1.00      1.00      30
```

شكل (١-٦٠) ٢



شكل (١-٦١) ٢

Gamma: 0.1, Regularization (C): 10

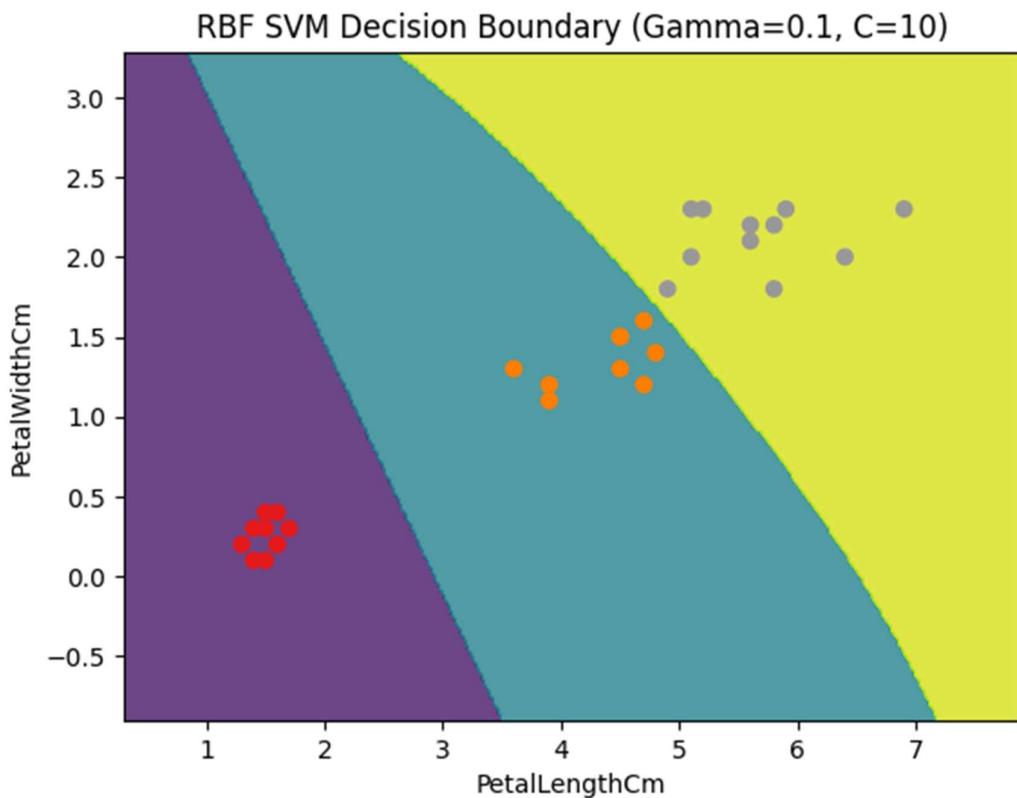
Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شكل (١-٦٢) 3



شكل (١-٦٣) 3

```
Gamma: 1, Regularization (C): 0.1
```

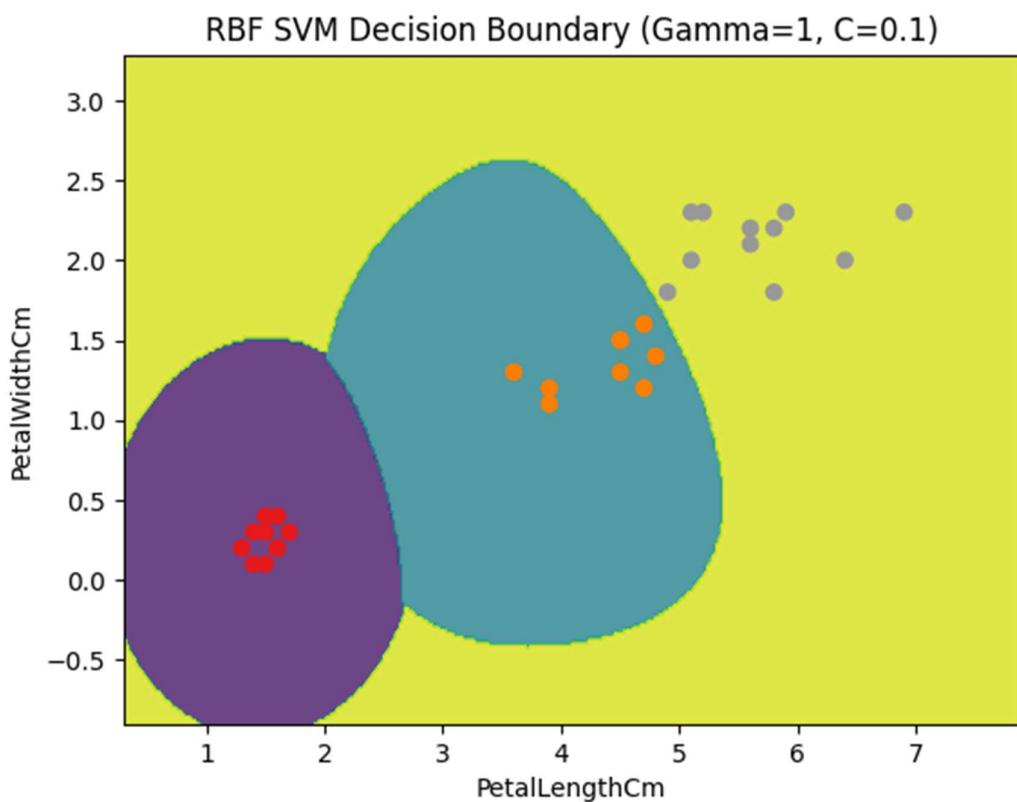
```
Confusion Matrix:
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
Classification Report:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شكل (٤-٦٤)



شكل (٤-٦٥)

Gamma: 1, Regularization (C): 1

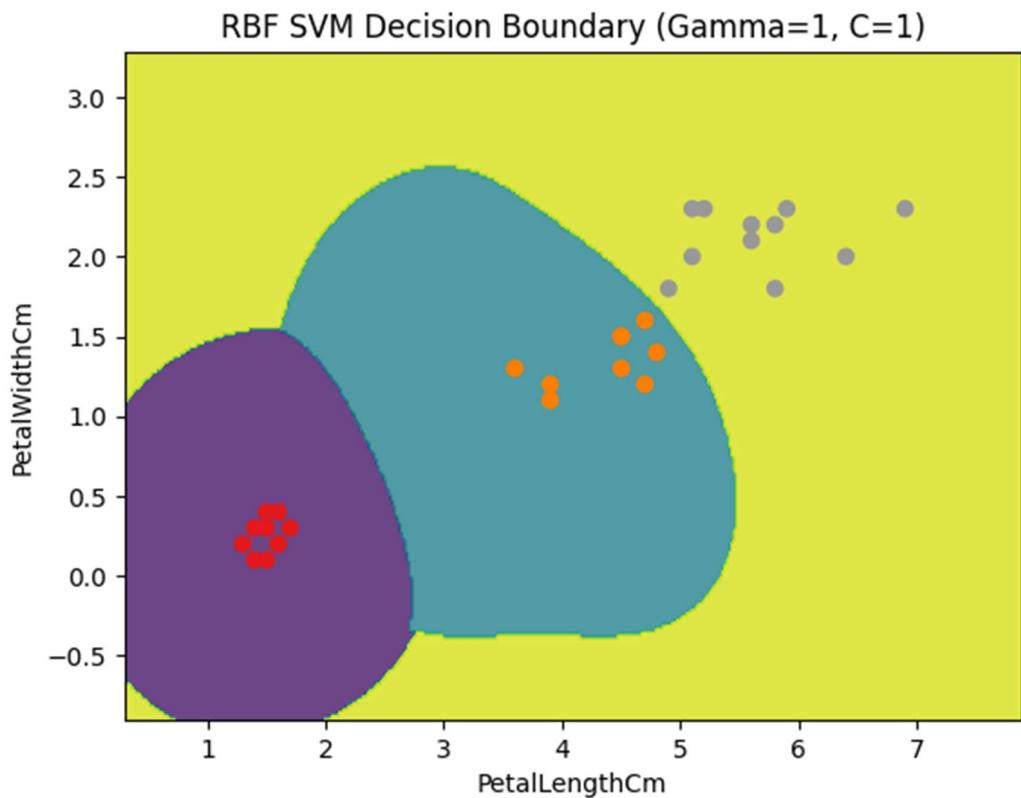
Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شكل (٦٦-١)



شكل (٦٧-١)

Gamma: 1, Regularization (C): 10

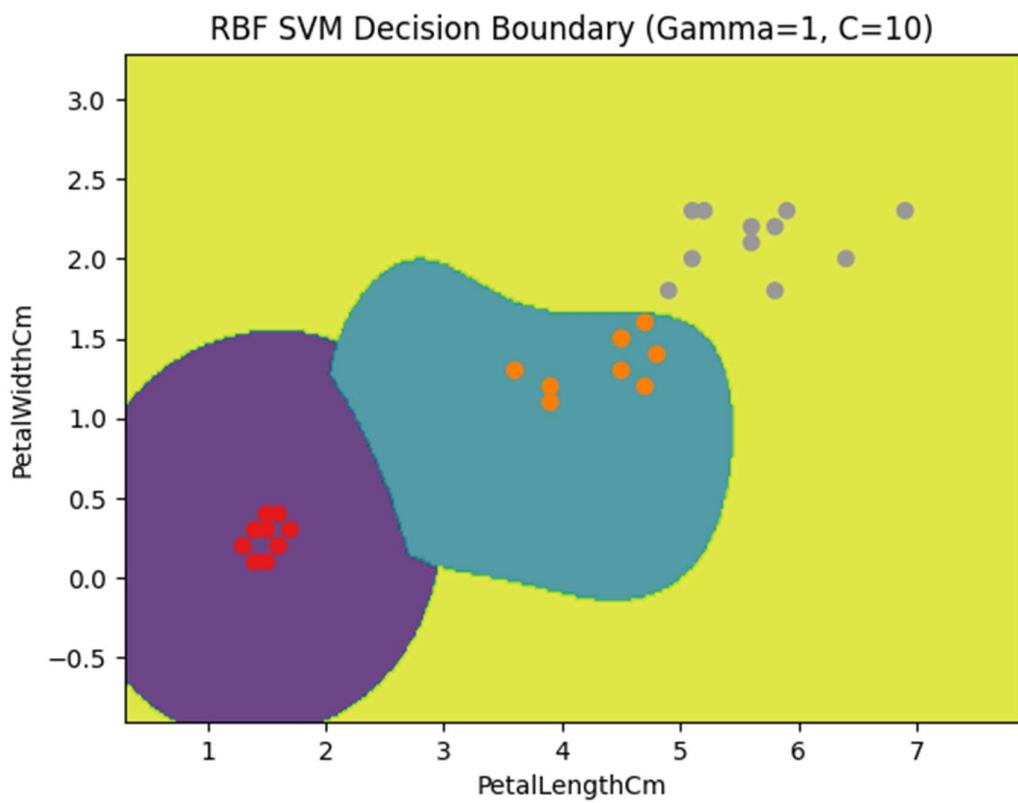
Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شکل (٦-٦٨)



شکل (٦-٦٩)

Gamma: 10, Regularization (C): 0.1

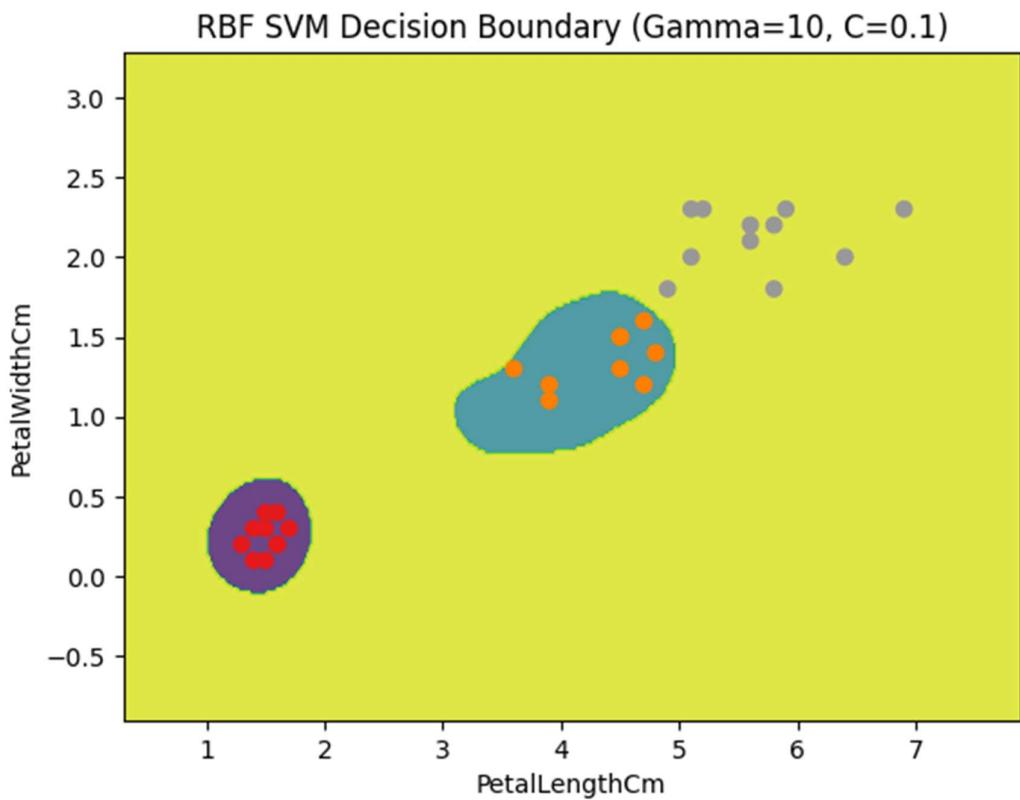
Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شكل (١-٧٠)



شكل (١-٧١)

```
Gamma: 10, Regularization (C): 1
```

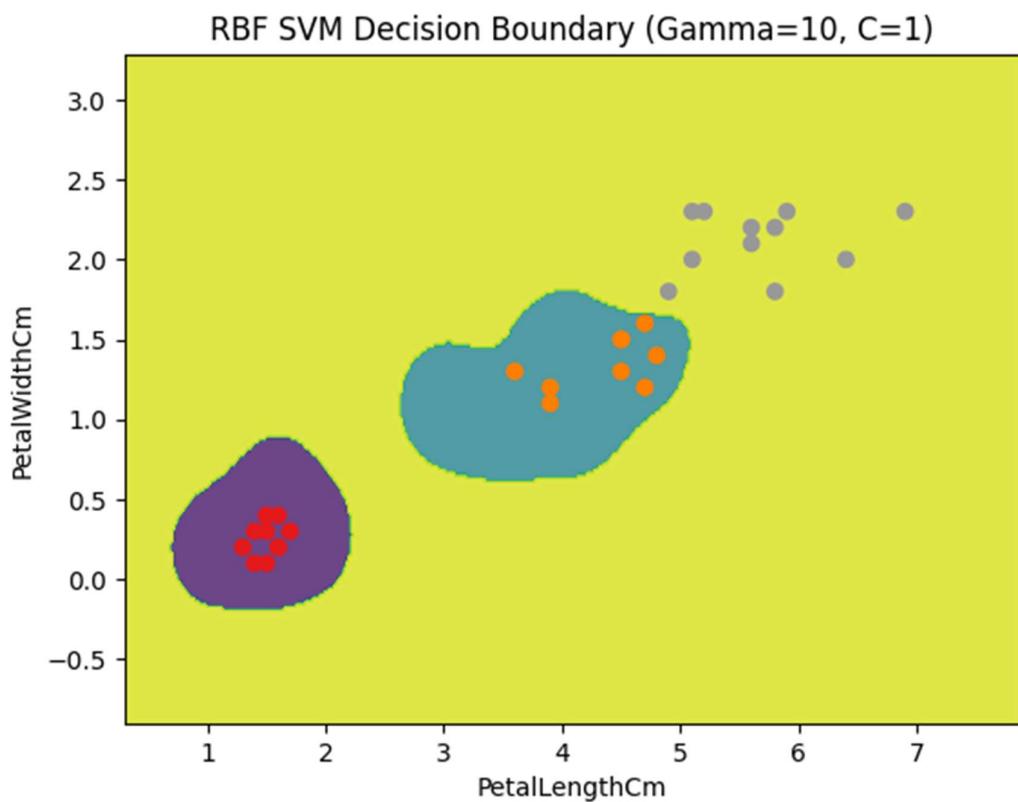
```
Confusion Matrix:
```

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

```
Classification Report:
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شکل (١-٧٢) ٨



شکل (١-٧٣) ٨

Gamma: 10, Regularization (C): 10

Confusion Matrix:

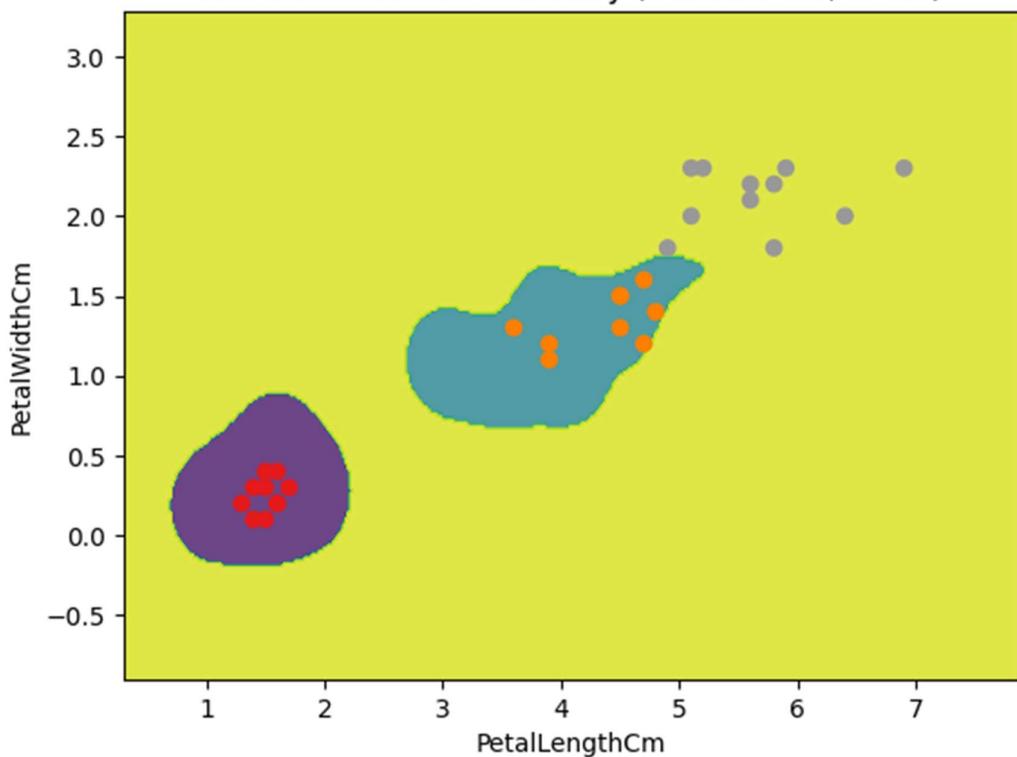
```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

شکل (١-٧٤) ٩

RBF SVM Decision Boundary (Gamma=10, C=10)



شکل (١-٧٥) ٩

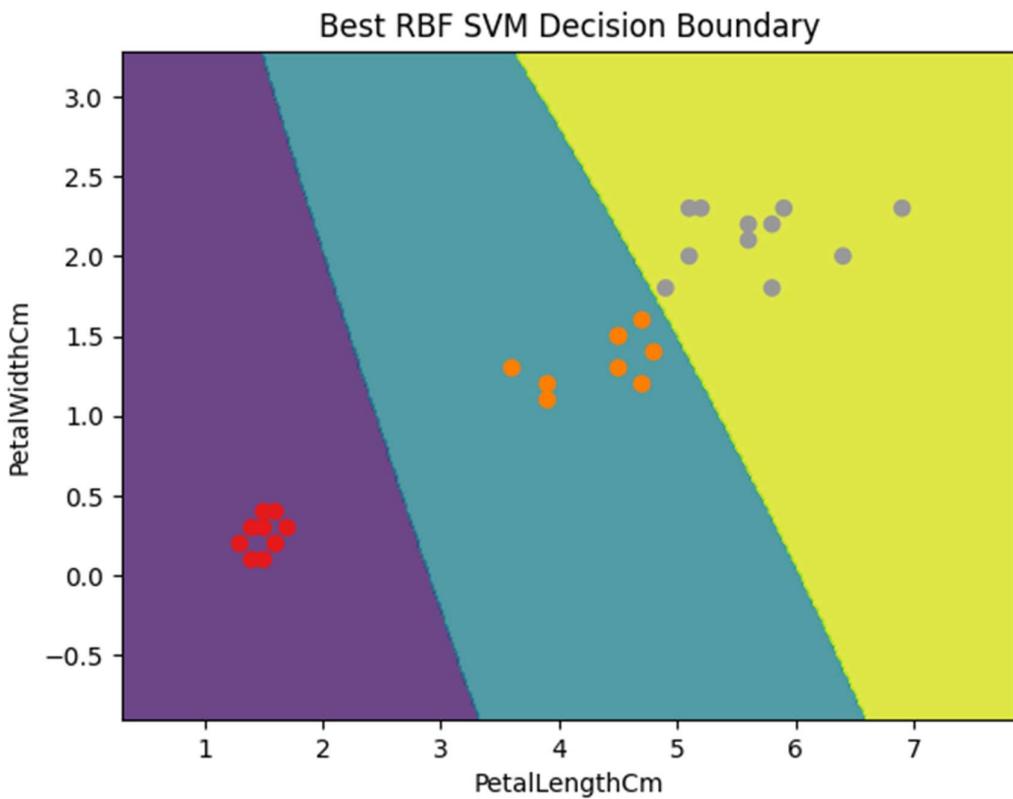
به صورت چشمی هم مواردی که در بخش قبل در توضیخات Gamma و C بیان شده بود ملاحظه شد

1-4-3-3- جستجوی شبکه ای پیدا کردن بهترین هایپر پارامتر

پس از جستجوی شبکه ای به نتایج زیر رسیدیم:

```
Best Parameters:  
{'C': 1, 'gamma': 0.1}  
  
Confusion Matrix:  
[[10  0  0]  
 [ 0  9  0]  
 [ 0  0 11]]  
  
Classification Report:  
 precision    recall    f1-score    support  
  
      0      1.00      1.00      1.00      10  
      1      1.00      1.00      1.00       9  
      2      1.00      1.00      1.00      11  
  
accuracy                           1.00      30  
macro avg       1.00      1.00      1.00      30  
weighted avg    1.00      1.00      1.00      30
```

شکل (1-76) بهترین هایپر پارامتر پس از جستجوی شبکه ای و گزارش آن (petal) classification



شکل (1-۷۷) مرز تصمیم rbf با بهترین پارامتر ها (sepal)

که خروجی جست و جوی شبکه ای با مشاهدات ما تطابق دارد.

بررسی 2 رویکرد ovo و ovr

بررسی رویکرد های مختلف روی کرنل های متفاوت به شرح زیر میباشد.

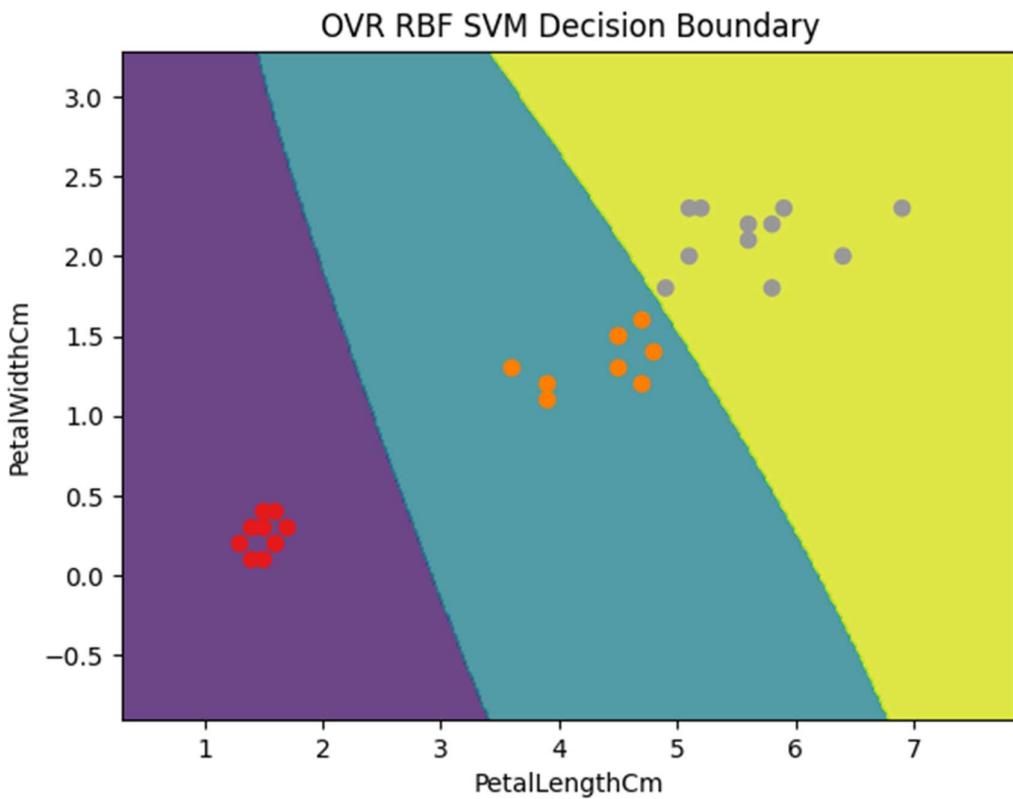
rbf ** کرنل

```

Kernel: rbf, Approach: ovr
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

```

شکل (1-۷۸) نتایج رویکرد ovr با کرنل rbf (sepal)



شکل (۱-۷۹) مرز تصمیم رویکرد ovr با کرnel (sepal) rbf

```

Kernel: rbf, Approach: ovo
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

```

شکل (۱-۸۰) نتایج روکرد ovo در کرnel (Sepal) rbf

مشاهده میشود نتیجه 2 رویکرد در این حالت یکی میشود.

به طور کاملا مشابه نتیجه برای این دو رویکرد در linear و poly هم مانند هم میشود و خروجی ها همان خروجی های بخش 1 و دقت های 100 میشود.

البته تفاوت ها و محل استفاده از هر کدام از این رویکردها در بخش 6-1-4 به طور کامل آمده است.

۱-۵-سوال ۵: سوالاتی در مورد SVM

۱-۵-۱-بخش ۱:

سوال ۵

$$k(x, y) = \exp(x^T y)$$

$$k(x, y) = \exp\left(-\frac{1}{2}\|x - y\|^2\right) \quad \text{پس: } \|x - y\|^2 \leq 2 \quad \text{۱}$$

$$\|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2x^T y$$

$$e^{-\frac{1}{2}(\|x\|^2 + \|y\|^2 - 2x^T y)} = e^{-\frac{\|x\|^2}{2}} e^{-\frac{\|y\|^2}{2}} e^{\frac{2x^T y}{2}} \quad \text{۲}$$

لطفاً دوستی داشت $\|x - y\| \leq \|x\| + \|y\| \quad \text{۳}$

$$\text{۲، ۳} \Rightarrow k(x, y) \leq e^{-\frac{\|x\| + \|y\|}{2} - \frac{\|x - y\|^2}{2}} e^{x^T y}$$

$$k(x, y) = e^{-\frac{1}{2}\|x - y\|^2} \leq 1 \times 1 \times e^{x^T y}$$

$\xrightarrow{\log}$ $-\frac{1}{2}\|x - y\|^2 \leq x^T y$

$\|x - y\|^2 \geq -2\|x\|\|y\|$

* $-2\|x\|\|y\| \leq 2$

$$\Rightarrow \boxed{\|x - y\|^2 \leq 2}$$

شکل (۱-۸۱) بخش ۱ سوال ۵

2-بخش 1-5-2

شکل (۱-۸۲) بخش ۲ سوال ۵

3-5-1-بخش سوم

تعداد بردارهای پشتیبان (SV) در یک SVM خطی hard به ویژگی های داده ها و جدایی بین کلاس ها بستگی دارد. به طور کلی، هدف الگوریتم SVM یافتن هایپرپلنی است که دو کلاس را حداکثر جدا می کند و در عین حال خطای طبقه بندی را به حداقل می رساند.

برای یک مسئله طبقه بندی دو کلاسه در فضای دو بعدی با n نقطه داده، آموزش اولیه یک SVM خطی سخت معمولاً مجموعه ای از SV ها را به دست می دهد که روی یا نزدیک به حاشیه کلاس ها قرار دارند. تعداد دقیق SV های مورد نیاز به پیچیدگی و توزیع داده ها بستگی دارد.

هنگامی که یک نقطه داده جدید به مجموعه داده موجود اضافه می شود و طبقه بندی مجدداً انجام می شود،

تعداد SV های مورد نیاز حداکثر به سناریوهای زیر بستگی دارد:

اگر نقطه داده جدید در منطقه ای قرار گیرد که بر مرز تصمیم گیری یا حاشیه ها تأثیری نداشته باشد، SV

های موجود بدون تغییر باقی می مانند. در این صورت تعداد SV ها ثابت می ماند.

اگر نقطه داده جدید در داخل یا نزدیک به مناطق حاشیه قرار گیرد، ممکن است بر مرز تصمیم تأثیر بگذارد

و به SV های اضافی نیاز داشته باشد. تعداد دقیق SV های اضافی مورد نیاز به موقعیت و ویژگی های نقطه داده

جدید نسبت به SV های موجود و مرزهای حاشیه بستگی دارد.

اگر نقطه داده جدید در داخل یا در سمت اشتباه مرز تصمیم قرار گیرد، ممکن است منجر به تغییر اساسی در

مرز تصمیم گیری شود که به طور بالقوه منجر به تعداد بیشتری SV می شود. تعداد دقیق SV های اضافی مورد

نیاز به پیچیدگی داده ها و میزانی که باید مرز تصمیم گیری برای تطبیق با نقطه جدید تنظیم شود، بستگی دارد.

6-1-سوال 6 : Ensamble Model :

پاسخ سوال در تصویر زیر آمده است.

سؤال ۶

$$N=5 \rightarrow \frac{n+1}{2} = \underline{\underline{3}}$$

$$P(\text{correct} \geq 3) = \sum_{k=3}^5 \binom{5}{k} (0.51)^k (0.49)^{5-k} = \underline{\underline{0.51871\%}}$$

$$N=9 \rightarrow \frac{n+1}{2} = 5$$

$$P(\text{correct} \geq 5) = \sum_{k=5}^9 \binom{9}{k} (0.51)^k (0.49)^{9-k} = \underline{\underline{0.5245\%}}$$

$$N \rightarrow \infty$$

$$\lim_{N \rightarrow \infty} P(\text{correct} \geq \frac{N+1}{2}) = \lim_{N \rightarrow \infty} \left(\sum_{k=\frac{N+1}{2}}^N \binom{N}{k} (0.51)^k (0.49)^{N-k} \right) = 1$$

ذابت در خطا نه است.

در واقعیت نزدیک این برچوں این اتفاق نمایم که N بزرگ باشیم و همچنانکه این مقدار

بزرگ شود (همچنانکه این مقدار ۵ بزرگ باشیم و عدم تاثیر خطا نه است بهم مطابق)

این کار بوده که در واقعیت این اتفاق نمایم.

$$acc=5\%$$

$$P(\text{correct} \geq 3) = \sum_{k=3}^5 \binom{5}{k} (0.5)^k (0.5)^{5-k} = \underline{\underline{5\%}}$$

بخش 4:

اگر هر طبقه‌بندی کننده در مجموعه داده دقت 50٪ (حدس زدن تصادفی) داشته باشد، و گروه از رای اکثربا 5=N استفاده کند، می‌توان نتیجه‌گیری زیر را داشت:

دقت طبقه‌بندی کننده‌های فردی: هر طبقه‌بندی کننده در سطح شانس عمل می‌کند، به این معنی که پیش‌بینی‌های آن به خوبی حدس زدن تصادفی است. دقت 50 درصد نشان می‌دهد که طبقه‌بندی کننده قادر به یادگیری الگوهای معنی دار از داده‌ها نیست.

دقت گروه: در یک گروه داده ای با رای اکثربا، پیش‌بینی نهایی با رای اکثربا طبقه‌بندی کننده‌ها تعیین می‌شود. از آنجایی که دقت هر طبقه‌بندی کننده 50 درصد است، دقت مجموعه نیز نزدیک به 50 درصد خواهد بود. مکانیسم رأی اکثربا در این مورد دقت را بهبود نمی‌بخشد زیرا اساساً حدس‌های تصادفی را ترکیب می‌کند.

عدم بهبود: هدف از استفاده از یک رویکرد گروهی مانند bagging، بهبود دقت کلی با ترکیب چند طبقه‌بندی ضعیف است. با این حال، زمانی که همه طبقه‌بندی کننده‌ها دقت پایینی دارند (50٪)، گروه نمی‌تواند از طبقه‌بندی کننده‌های تکی بهتر عمل کند. در این سناریو، طبقه‌بندی مزیت قابل توجهی نسبت به استفاده از یک طبقه‌بندی کننده ندارد.

۱-۷-سوال ۷ : تخمین میزان درآمد و تخمین قیمت هتل

۱-۷-۱- تخمین میزان درآمد

ابتدا دیتا ست مربوطه را میخوانیم

```
data = pd.read_csv('Q7-Part1.csv')
data.head()
```

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

شکل (۱-۸۴) خواندن دیتاست Q7-part1

در ادامه داده ها را به X و Y تخصیص داده و تمامی داده ها را نرم افزار میکنیم (لازم به ذکر است با توجه به

اینکه level همان position را تعریف میکند ستون position به طور کلی حذف میشود.

```
X = data.iloc[:, 1].values # Job position level
y = data.iloc[:, 2].values # Income

scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X.reshape(-1, 1))
y_scaled = scaler_y.fit_transform(y.reshape(-1, 1))
```

شکل (۱-۸۵) پیش پردازش داده های Q7-part1

در ادامه همانند سوال ۴ در یک loop به ازای هر ۳ کرنل svm را اعمال کرده و خروجی ها را رسم میکنیم.

```

kernels = ['rbf', 'linear', 'poly']
kernel_names = ['RBF', 'Linear', 'Polynomial']

```

شکل (۱-۸۶) تعریف کرnel های svm

```

for kernel, kernel_name in zip(kernels, kernel_names):
    svr = SVR(kernel=kernel)
    svr.fit(X_train, y_train.ravel())

    # Calculate accuracy on train and validation sets
    y_train_pred = svr.predict(X_train)
    y_val_pred = svr.predict(X_test)

    train_accuracy = r2_score(y_train, y_train_pred)
    val_accuracy = r2_score(y_test, y_val_pred)

    print(f'Accuracy (Train) - Kernel: {kernel_name}: {train_accuracy:.2f}')
    print(f'Accuracy (Validation) - Kernel: {kernel_name}: {val_accuracy:.2f}\n')

```

شکل (۱-۸۷) کد پایتون فیت کردن داده ها روی هر کرnel و گزارش دقت (r^2) روی ترین و تست

در ادامه داده های اورجینال و پیشینی شده را کنار هم رسم میکنیم.

توجه شود داده ها را به کمک `imverse` از حالت نرمال شده به حالت اولیه بر میگردانیم.

```

# Print predicted values on test data (in original scale)
y_test_pred = svr.predict(X_test)
y_test_pred = scaler_y.inverse_transform(y_test_pred.reshape(-1, 1))
print(f'Predicted values on test data (Kernel: {kernel_name}): \n{y_test_pred}\n')

# Plot predicted and actual values for the entire dataset (in original scale)
X_actual = scaler_X.inverse_transform(X_scaled)
y_actual = scaler_y.inverse_transform(y_scaled)

X_plot = np.linspace(min(X_actual), max(X_actual), num=100).reshape(-1, 1)
y_pred_plot = svr.predict(scaler_X.transform(X_plot))
y_pred_plot = scaler_y.inverse_transform(y_pred_plot.reshape(-1, 1))

plt.scatter(X_actual, y_actual, color='blue', label='Actual')
plt.plot(X_plot, y_pred_plot, color='red', label='Predicted')
plt.title(f'Actual vs Predicted Income (Kernel: {kernel_name})')
plt.xlabel('Level')
plt.ylabel('Income')
plt.legend()
plt.show()

```

شکل (۱-۸۸) کد پایتون پیشینی داده های تست و رسم همه داده ها و پیش بینی شده آن ها کنار هم

توجه شود با توجه به ۱۰تا بودن داده ها و اینکه ۲۰رصد را برای تست جدا کنیم میشود ۲ تا ، در این بخش تمامی داده ها و پیشینی آن ها را کنار هم رسم کردیم ولی داده تستی که شبکه ندیده داده دوم و هشتم بوده است.

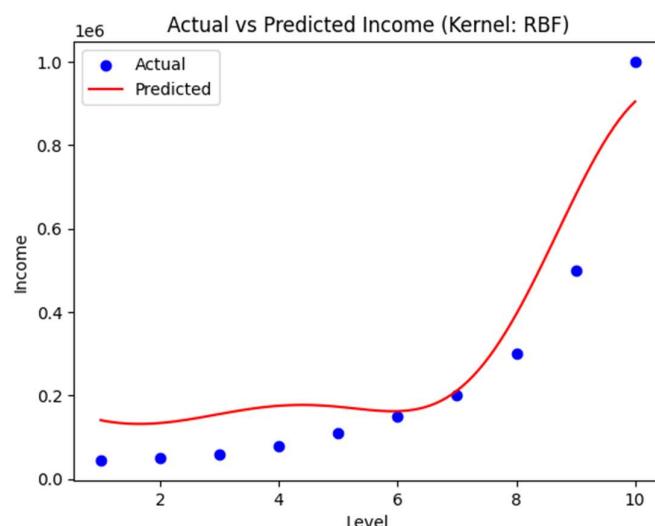
برای وضوح بهتر داده های گسته predicted را به صورت پیوسته رسم میکنیم.

: RBF کرنل **

```
Accuracy (Train) - Kernel: RBF: 0.93
Accuracy (Validation) - Kernel: RBF: 0.61
```

شکل (۱-۸۹) دقت r^2 روی داده ترین و تست کرنل rbf

برای وضوح بهتر داده های گسته predicted را به صورت پیوسته رسم میکنیم.

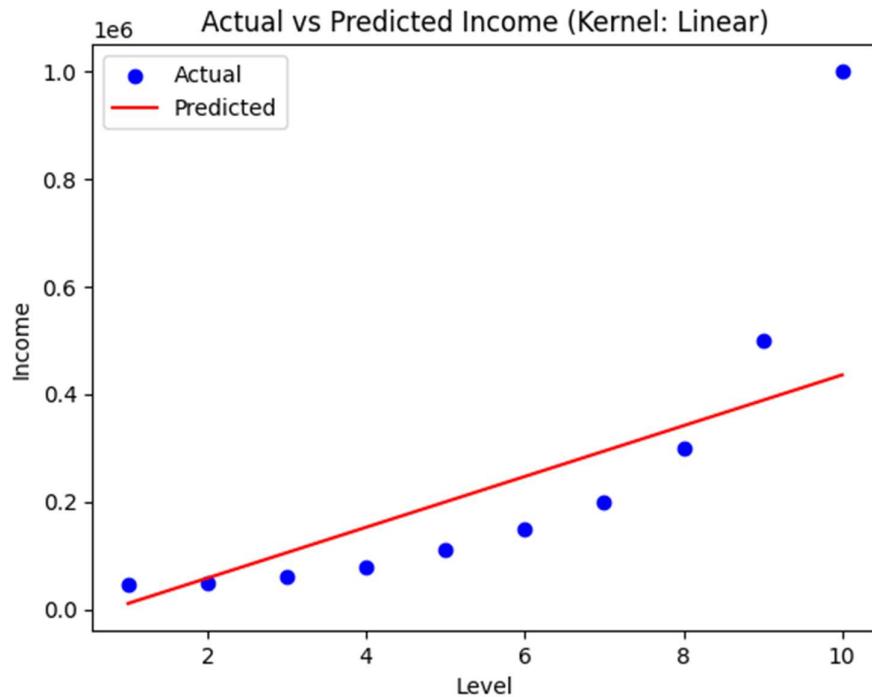


شکل (۱-۹۰) منحنی داده های واقعی و پیش بینی شده با کرنل rbf

: kernel linear **

Accuracy (Train) - Kernel: Linear: 0.50
Accuracy (Validation) - Kernel: Linear: 0.88

شكل (۱-۹۱) دقت r^2 روی داده ترین و تست کرnel linear



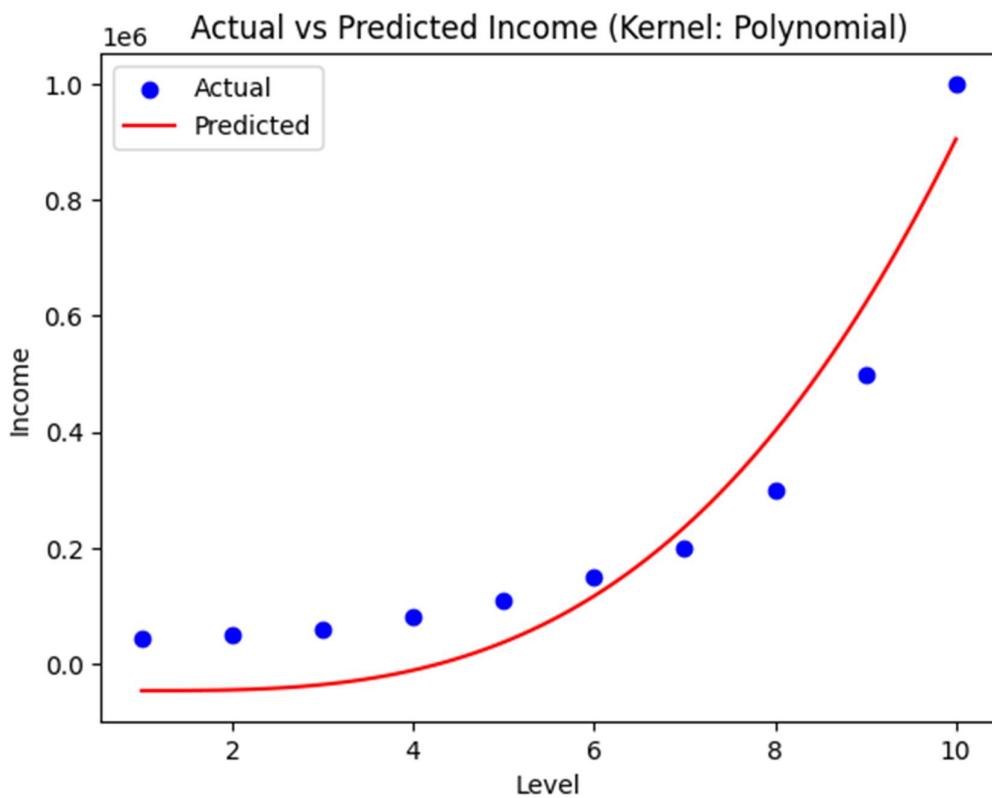
شكل (۱-۹۲) منحنی داده های واقعی و پیش بینی شده با کرnel linear

: kernel Poly **

Accuracy (Train) - Kernel: Polynomial: 0.93
Accuracy (Validation) - Kernel: Polynomial: 0.77

Predicted values on test data (Kernel: Polynomial):
[621633.32357014 -44698.70915763]

شكل (۱-۹۳) دقت r^2 روی داده ترین و تست کرnel poly



شکل (۱-۹۴) منحنی داده های واقعی و پیش بینی شده با کرnel poly

مشاهده میشود بیشترین دقت برای وقتی است که از کرnel polynimal استفاده شده است که پر واضح است که داده ها اگر دقت شود حالت درجه 2 دارند و قطعا استفاده از این کرnel مناسب تر است.

1-7-2- تخمین قیمت هتل

ابتدا فایل دیتاست مربوطه تست و ترین را میخوانیم و در ادامه داده های categorical را one encode و

hot میکنیم و داده های عددی را نیز نرمال میکنیم (ابتدا ستون ADR که تارگت میباشد را جدا میکنیم لازم به

ذکر است این ستون هم نرمال میکنیم).

```
# Load data from H1.csv and H2.csv
train_data = pd.read_csv("H1.csv")
test_data = pd.read_csv("H2.csv")
```

شکل (۱-۹۵) لود دیتاست تست و ترین داده های هتل

```
# Separate features and target variable
X_train = train_data.drop("ADR", axis=1)
y_train = train_data["ADR"]

X_test = test_data.drop("ADR", axis=1)
y_test = test_data["ADR"]
```

شکل (۱-۹۶) تفکیک ستون ها به X و Y

```
# Normalize numerical features
scaler = MinMaxScaler()
numerical_cols = ["LeadTime", "ArrivalDateYear", "ArrivalDateWeekNumber", "ArrivalDateDayOfMonth", "StaysInWeekendNights",
                  "StaysInWeekNights", "Adults", "Children", "Babies", "PreviousCancellations", "PreviousBookingsNotCanceled",
                  "BookingChanges", "DaysInWaitingList", "RequiredCarParkingSpaces", "TotalofSpecialRequests"]
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])
```

شکل (۱-۹۷) نرمال کردن داده های عددی

```
# Encode categorical features
categorical_cols = ["ArrivalDateMonth", "Meal", "Country", "MarketSegment", "DistributionChannel", "ReservedRoomType",
                     "AssignedRoomType", "DepositType", "CustomerType"]

# Convert categorical features using one-hot encoding
encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
X_train_encoded = encoder.fit_transform(X_train[categorical_cols])
X_test_encoded = encoder.transform(X_test[categorical_cols])
```

شکل (۱-۹۸) داده های Encode categorical

```

# Get one-hot encoded feature names
encoded_feature_names = encoder.get_feature_names_out(categorical_cols)

# Convert encoded features to DataFrame
X_train_encoded_df = pd.DataFrame(X_train_encoded, columns=encoded_feature_names)
X_test_encoded_df = pd.DataFrame(X_test_encoded, columns=encoded_feature_names)

# Concatenate encoded features with remaining numerical features
X_train_processed = pd.concat([X_train_encoded_df, X_train[numerical_cols]], axis=1)
X_test_processed = pd.concat([X_test_encoded_df, X_test[numerical_cols]], axis=1)

```

شکل (۱-۹۹) کردن داده های categorical و چسباندن داده های numerical به هم

```

# Normalize target variable
target_scaler = MinMaxScaler()
y_train_normalized = target_scaler.fit_transform(y_train.values.reshape(-1, 1))
y_test_normalized = target_scaler.transform(y_test.values.reshape(-1, 1))

```

شکل (۱-۱۰۰) نرمال سازی داده های ADR

یک مورد دیگه وجود NaN در دیتا است میباشد که مدیریت مقادیر از دست رفته در مجموعه داده به ماهیت داده و الزامات خاص بستگی دارد. چند رویکرد رایج وجود دارد که می توان برای مدیریت مقادیر از دست رفته در نظر گرفت:

- سطرها یا ستون های شامل nan را حذف کینم: اگر مقادیر از دست رفته از نظر تعداد نسبتاً کم هستند

و به طور تصادفی توزیع شده اند، می توان سطرها یا ستون های حاوی مقادیر nan را حذف کرد. این

رویکرد زمانی مناسب است که مقادیر از دست رفته بعد از دست رفته از سوگیری قابل توجهی در تحلیل

و آموزش ایجاد کند.

- Imputation: شامل پر کردن مقادیر با مقادیر تخمینی است. یک رویکرد رایج جایگزینی مقادیر

از دست رفته با میانگین، میانه یا حالت ستون مربوطه است. در اینجا از این روش استفاده میکنیم.

- تکنیک های Imputation پیشرفته: بسته به داده ها و ، ممکن است تکنیک های

پیشرفته تری مانند k-نزدیک ترین همسایگان (KNN) یا Imputation رگرسیونی را

در نظر گرفت، که روابط بین ویژگی ها را برای تخمین مقادیر Nan در نظر می گیرد.

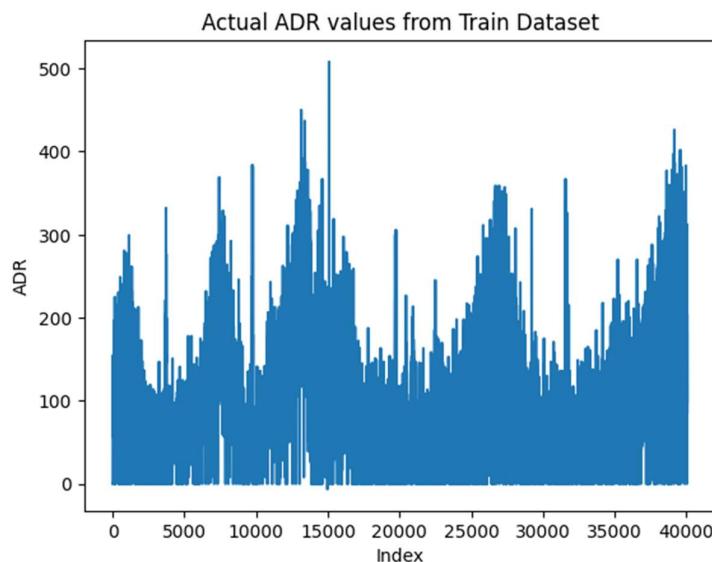
با توجه به توضیحات فوق :

```
# Handle missing values
imputer = SimpleImputer(strategy='mean')
X_train_processed = pd.DataFrame(imputer.fit_transform(X_train_processed), columns=X_train_processed.columns)
X_test_processed = pd.DataFrame(imputer.transform(X_test_processed), columns=X_test_processed.columns)
```

شکل (۱-۱۰۱) جای گذاری مقادیر miss شده

انتخاب کرنل مناسب :

بهتر است نگاهی به توزیع داده های ADR بیاندازیم :



شکل (۱-۱۰۲) توزیع مقادیر داده های آموزش ADR

*** یکی از نکات عجیب در فایل test وجود ADR برابر 5200 میباشد.

مشاهده میشود که خیلی نمیشود به خطی و غیر خطی اعتماد کرد و به همین جهت در این مساله از کرنل

محبوب SVM ها یعنی rbf استفاده میکنیم. (البته استفاده از کدام کرنل یک بحث کاملا تخصصی میباشد و

صرف نگاه کردن توزیع داده ها نمیتواند دلیل قابل قبولی باشد)

```
# SVR model training
svm_model = SVR(kernel='rbf')
svm_model.fit(X_train_processed, y_train_normalized)
```

شکل (۱-۱۰۳) آموزش Support Vector Regression

گرفتن خروجی روی داده های تست و بررسی مقدار خطای MSE

```
# Make predictions on test data
y_pred = svm_model.predict(X_test_processed)

# Evaluation metrics
mse = mean_squared_error(y_test_normalized, y_pred)
print("Mean Squared Error (MSE):", mse)
```

Mean Squared Error (MSE): 0.013236709160265817

شکل (۱-۱۰۴) گرفتن خروجی و بررسی مقدار خطای MSE

مشاهده میشود مقدار خطای قابل قبول میباشد.

در ادامه داده های را numpy کرده و از نرم افزار خارج میکنیم و در نهایت مقدار های واقعی و پیش بینی شده

و تفاوت آن ها را در یک فایل CSV ذخیره میکنیم.

```

# Convert predictions to array
y_pred = np.array(y_pred)
y_test_normalized = np.array(y_test_normalized)

y_pred_actual = target_scaler.inverse_transform(y_pred.reshape(-1, 1))
y_test_actual = target_scaler.inverse_transform(y_test_normalized.reshape(-1, 1))

# Compute the absolute difference
absolute_difference = abs(y_pred_actual.flatten() - y_test_actual.flatten())

# Create a DataFrame with the actual values, absolute difference, and signed difference
df = pd.DataFrame({'y_pred_actual': y_pred_actual.flatten(),
                    'y_test_actual': y_test_actual.flatten(),
                    'absolute_difference': absolute_difference})

# Save the DataFrame to a CSV file
df.to_csv('actual_values_and_absolute_difference.csv', index=False)

```

شکل (۱-۱۰۵) ذخیره سازی مقادیر پیشینی شده و واقعی و اختلاف آن ها در یک فایل CSV