بسمه تعالی

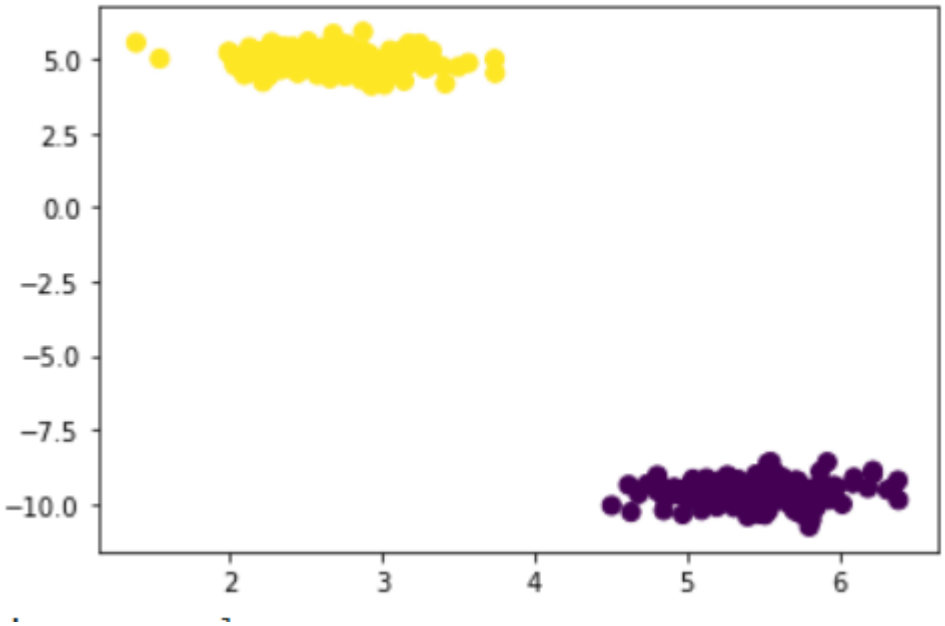
پروژه سوم ماشین بردار پشتیبان

آیسا میاهی نیا

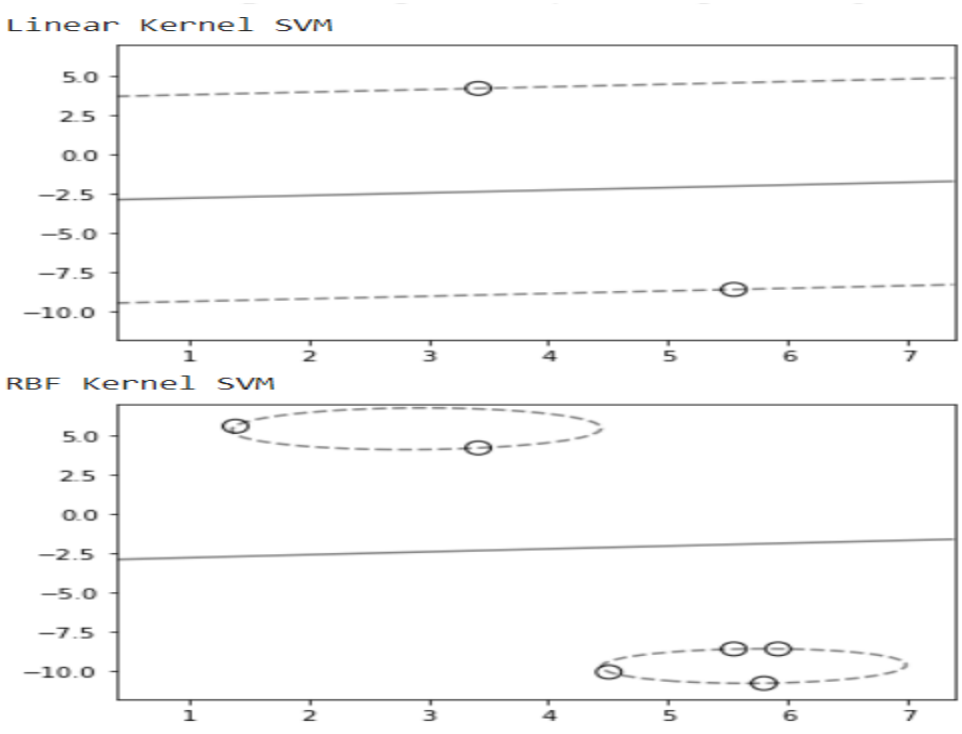
استاد عبدی

برای قسمت اول که ازمون خواسته شده که یک سری داده رندوم را تولید کنیم از تابع make_blobs برای این منظور استفاده می کنیم و با دادن آرگومان های مختلف به این تابع میتوانیم دیتاست هایی با تعداد مختلف و با پیچیدگی مختلف نیز به دست آوریم.

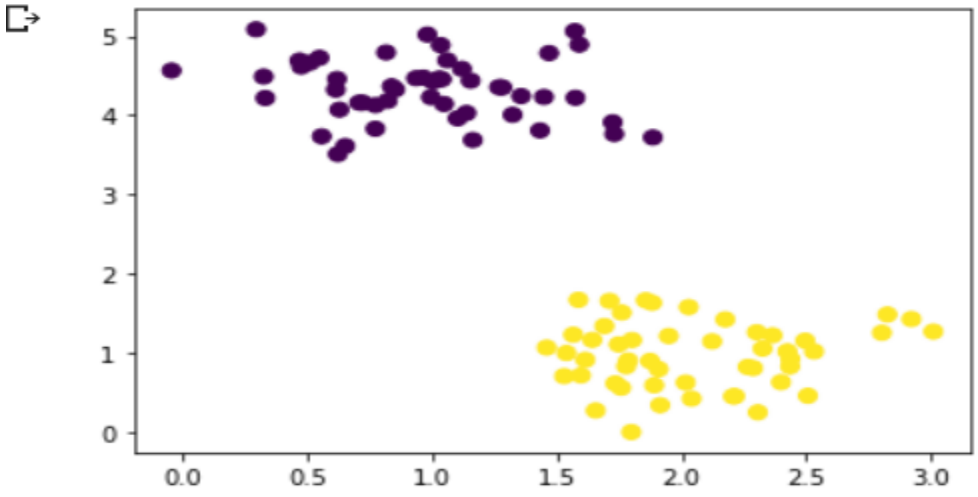
اولین نمونه ای که تولید می کنیم بسیار ساده می باشد با 300 نقطه ابتدا فقط نمودار خود نقاط را نمایش می دهیم که به صورت زیر می باشد.



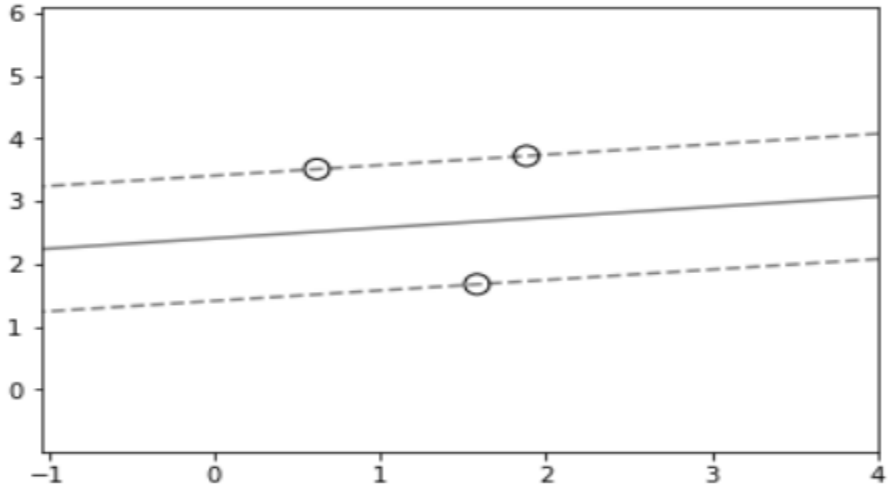
برای استفاده از SVM از دو نوع kernel آن استفاده کردیم که البته واضح است که برای این نمونه استفاده از kernel خطی کافی و مناسب می باشد که میتوان خروجی رو به ازای هم kernel خطی و هم RBF مشاهده کرد. که خط margin آن نیز رسم شده است.



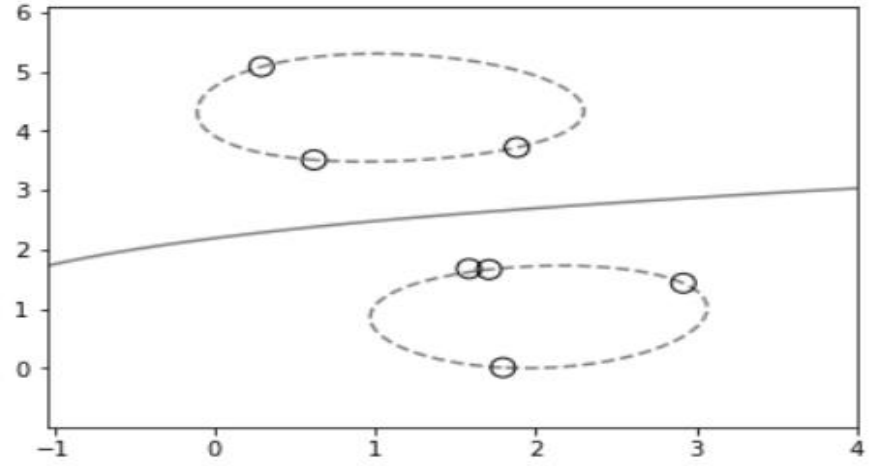
تست کیس دوم ما به صورت زیر می باشد که دارای 100 نقطه و فاصله نقاط از دو گروه مختلف کمتر می باشد. باز هم از دو مدل kernel برای محاسبه SVM آن استفاده کردیم.



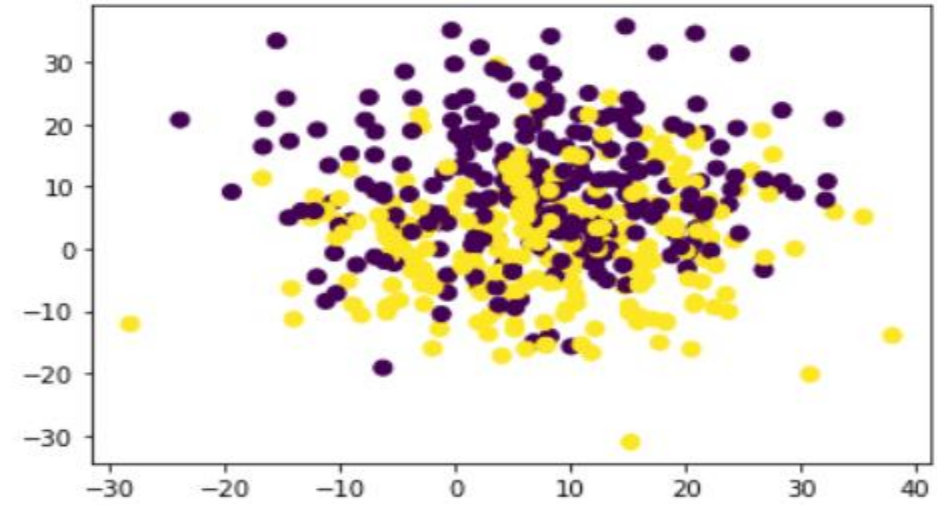
Linear Kernel SVM



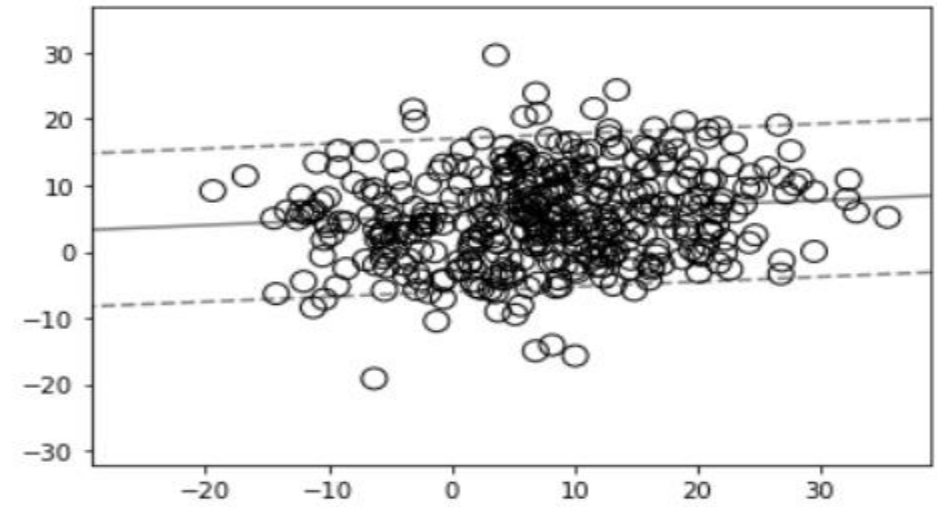
RBF Kernel SVM



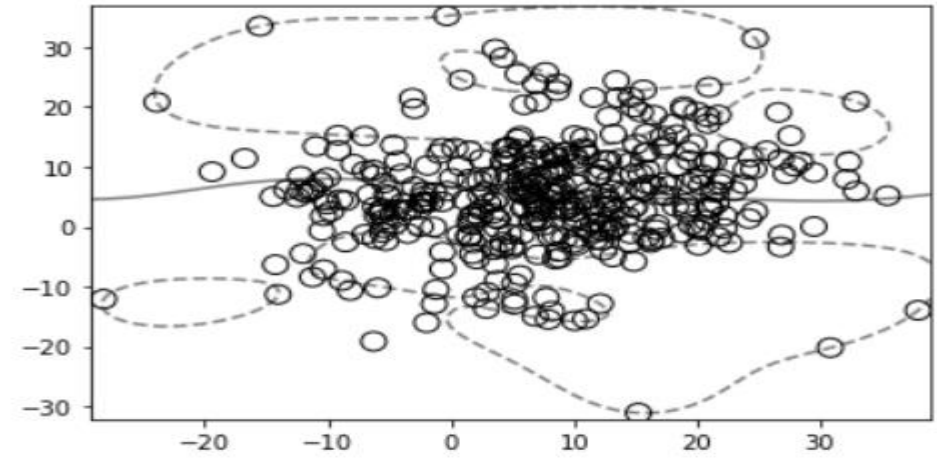
تست کیس سوم، پیچیدگی خیلی بیشتری نسبت به قبلی ها دارد و این پیچدگی با تغییر دادن آرگومان cluster_std به دست آمده است. در این تست کیس می بایست حتما از تابع هسته RBF استفاده کنیم چون همانطور که در تصویر هم قابل مشاهده می باشد استفاده از این تابع هسته داده ها را بسیار بهتر جدا کرده است تا استفاده از تابع هسته خطی.



Linear Kernel SVM



RBF Kernel SVM



تست کیس چهارم هم مشابه تست کیس سوم می باشد فقط نقاط بیشتر داخل هم رفته اند. و باز هم بهتر است از تابع هسته RBF استفاده کرد.

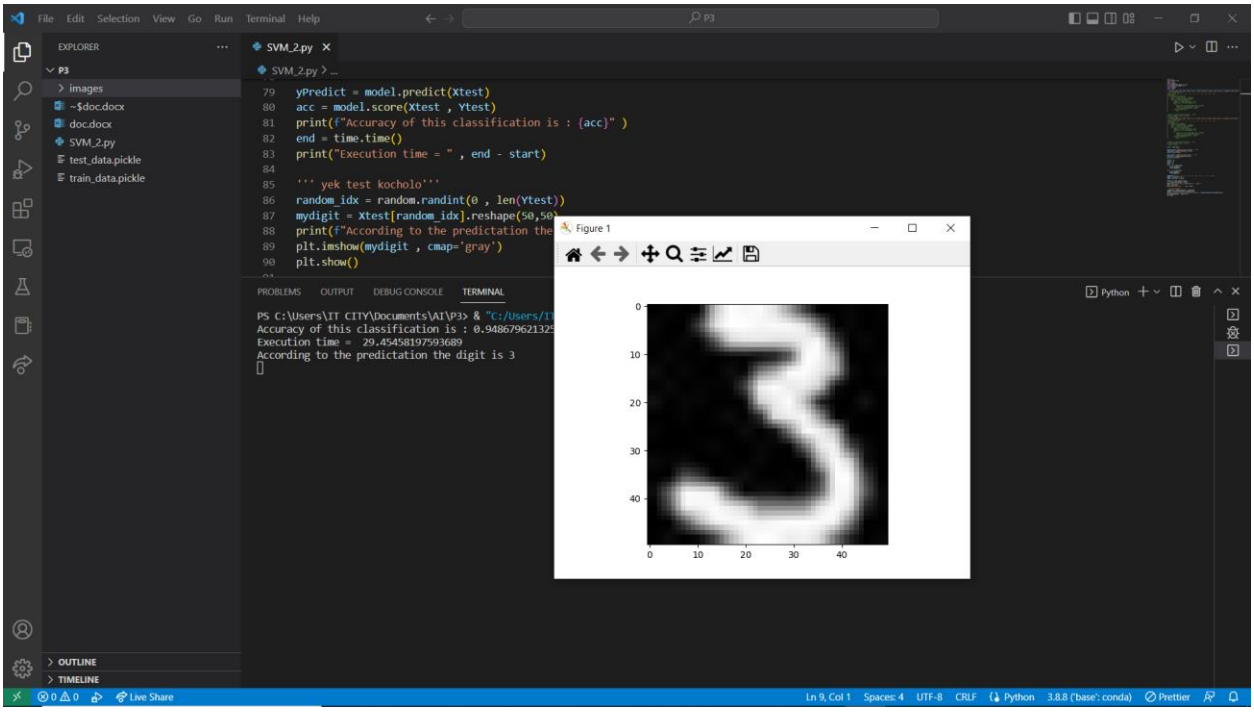
قست دوم پروژه از ما خواسته که SVM را روی یک پایگاه داده پیاده سازی کنیم. ما از پایگاه داده داده شده USPS استفاده می کنیم. در این داده 2007 داده تست داریم و 7291 داده train داریم که آن ها را براساس اینکه چه عددی هستند پوشه بندی میکنیم. سپس این عکس ها را یکبار داده های تست و یکبار هم داده های train را میخوانیم. برای اینکه داده هایی که میخوانیم را به شکل مناسب و درستی برای وروردی دادن به تابع SVM ما باشند، آن ها را resize کرده تا همه دارای سایز های برابری باشند و همچنین آن ها را flatten می کنیم تا آرایه های یک بعدی داشته باشیم و کارمان برای محاسبات بعدی راحت تر باشد.

از آنجایی که خواندن همه این اطلاعات آن هم دوبار یک بار تحت عنوان تست و یکبار هم تحت عنوان train خیلی زمانبر می باشد از کتابخانه pickle استفاده کرده و داده ها را ذخیره می کنیم حال کافیست فقط برای دسترسی به داده های تست یا train فایل pickle مربوطه را load کنیم.

از آنجایی که دیتاست ما از قبل split شده بود به داده های train وtest پس نیازی به اسپلیت کردن نداریم فقط کافیست مقادیر x و y آن ها را جدا کنیم. بعد هم که از تابع آماده SVM استفاده می کنیم و از تابع هسته RBF.

برای بررسی درستی کار، accuracy را محاسبه می کنیم که همانطور که میبینید 0.948 می باشد که مقدار بسیار خوبیست.

برای اینکه عملکرد آن را بهتر مشاهده کنیم یک عکس رندوم از دیتاست test را انتخاب میکنیم و خروجی کد را به ازای آن عکس میبینیم و خود عکس را نیز نمایش می دهیم. برای مثال عکس زیر را مشاهده کنید.



همانطور که در عکس میبینید عدد 3 می باشد، و پیش بینی کد هم 3 می باشد.

یکی از چالش هایی که من داشتم، خواندن عکس ها و طبقه بندی آن ها بود. از طرفی هربار خواندن عکس ها زمان زیادی می گرفت که با ذخیره کردن ان ها در زمان صرفه جویی شد. همچنین اگر داده ها را flatten و resize نمیکردیم دقت کار کاهش پیدا می کرد و شاید پیچیدگی کد هم زیاد می شد.