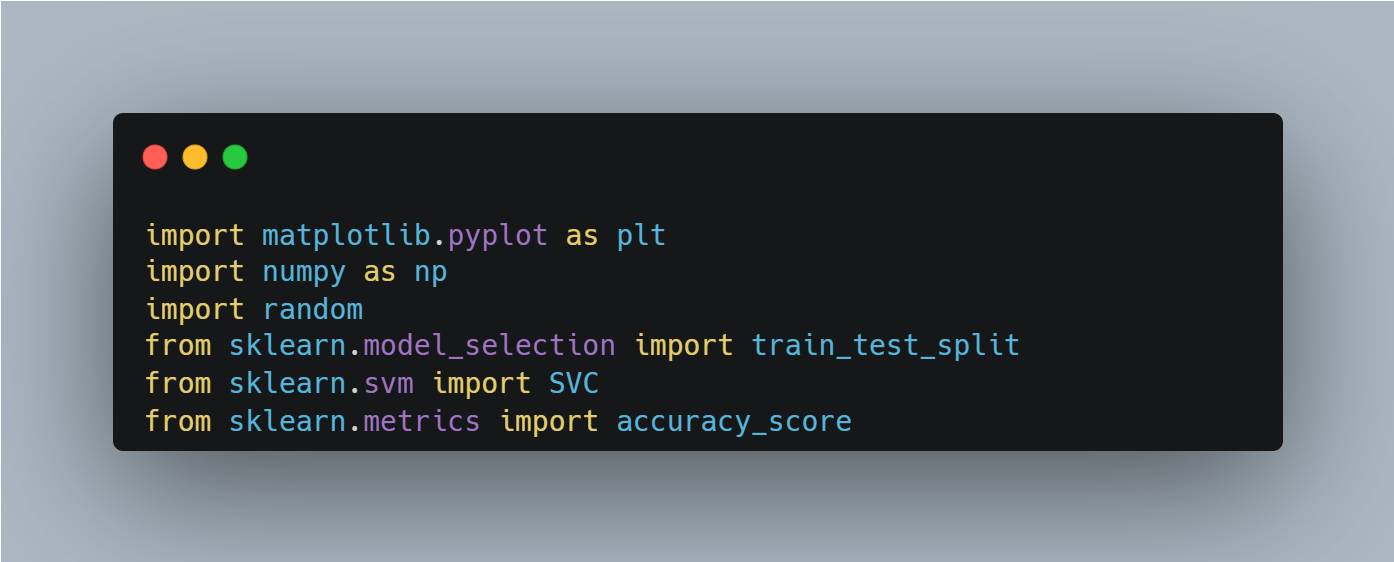
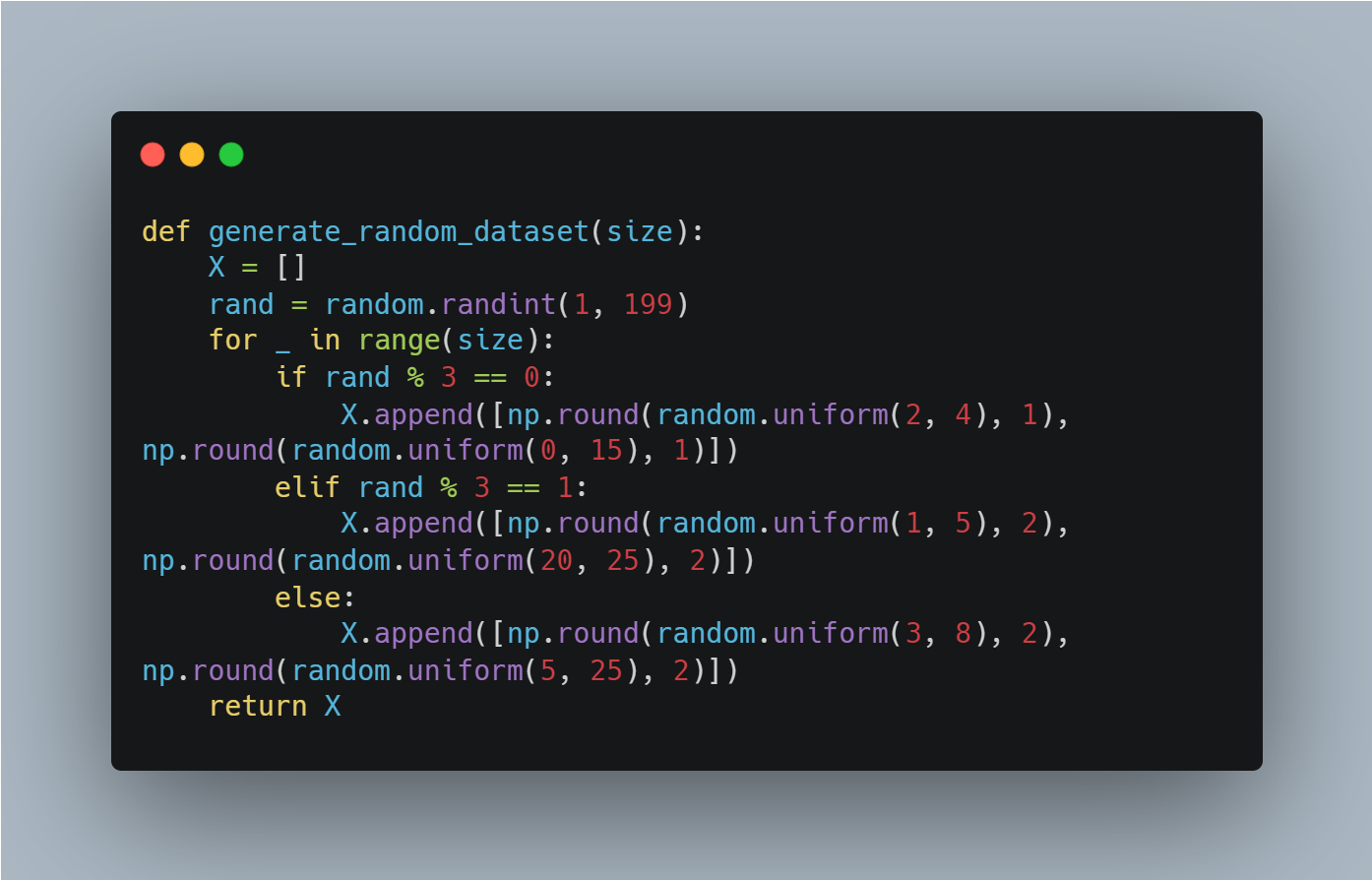
به نام خدا

پروژه چهارم درس هوش مصنوعي – ماشين بردار پشتيبان بهاره كاوسي نژاد – 99431217

**بخش 1:**



ابتدا كتابخانه هاي لازم را import مي‌كنيم.



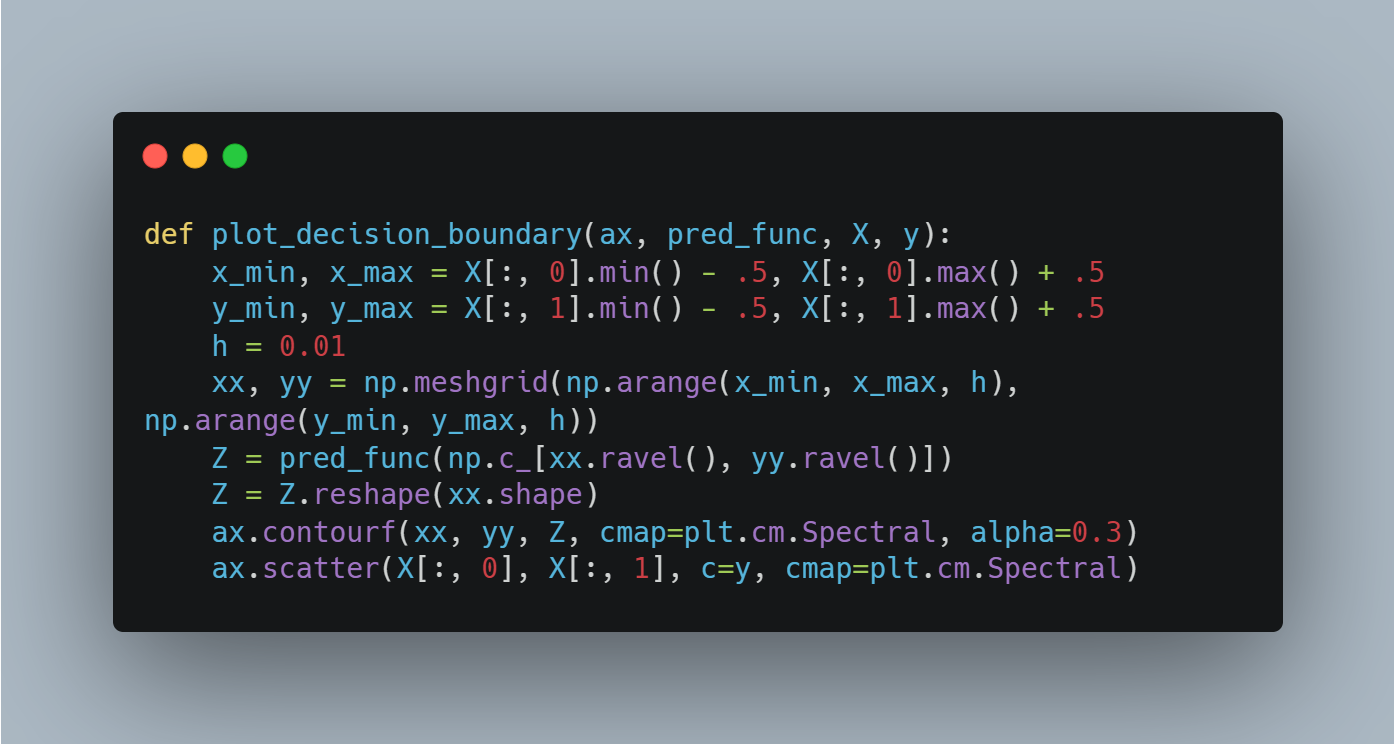
از اين تابع به منظور generate كردن يك dataset تصادفي استفاده مي‌شود.

****

از تابع makePoints براي تعيين برچسب‌هاي dataset توليد شده توسط تابع generate\_random\_dataset استفاده مي‌شود. براي تعيين اين برچسب‌ها از يك threshold استفاده مي‌شود.

****

بر اساس برچسب‌هاي تعيين شده در قسمت قبل، داده‌ها را classify مي‌كنيم و آنها را به دو كلاس تقسيم مي‌كنيم.

****

از تابع plot\_decision\_boundry براي ترسيم ديتاهاي توليد شده و classify شدن آنها استفاده مي‌شود.

****

در اين قسمت در يك حلقه while به توليد 500 ديتاي تصادفي مي‌پردازيم و پس از تعيين برچسب‌هاي آنها با استفاده از تابع makePoints، 80 درصد آن را به عنوان داده train و 20 درصد را به عنوان داده تست قرار مي‌دهيم. با استفاده از تابع ClassifyedPoints آنها را classify مي‌كنيم. اين حلقه تا زماني ادامه مي‌يابد كه مقادير classify شده صفر نباشند. اگر هردو آنها داراي مقدار باشند، حلقه پايان مي‌يابد.

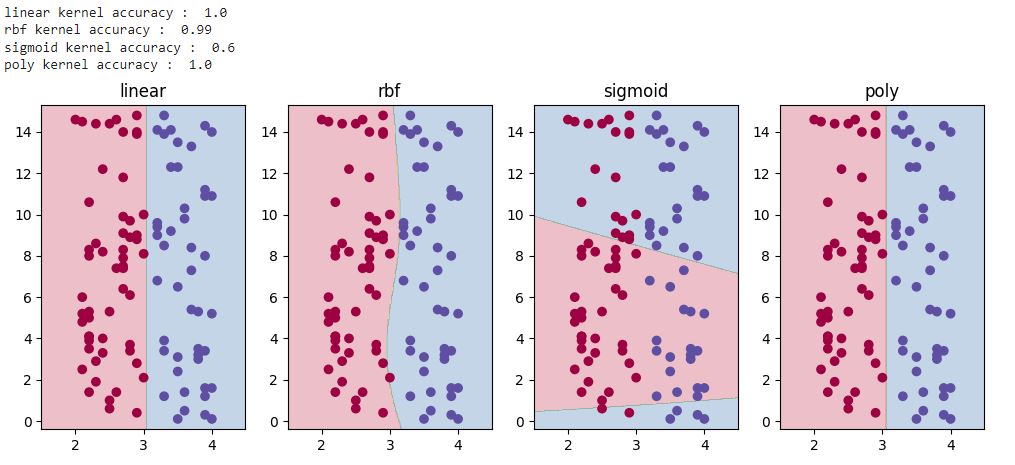
****

در اين قسمت با استفاده از تابع آماده SVC، classifierهايي با چهار kernel متفاوت (linear، rbf، sigmoid و poly) توليد مي‌كنيم. هر classifier روي داده‌هاي train با استفاده از تابع fit، train مي‌شود. پس از train شدن، از classifierها براي پيش‌بيني روي داده‌هاي تست استفاده مي‌شود و دقت با تابع accuracy محاسبه و چاپ مي‌شود.



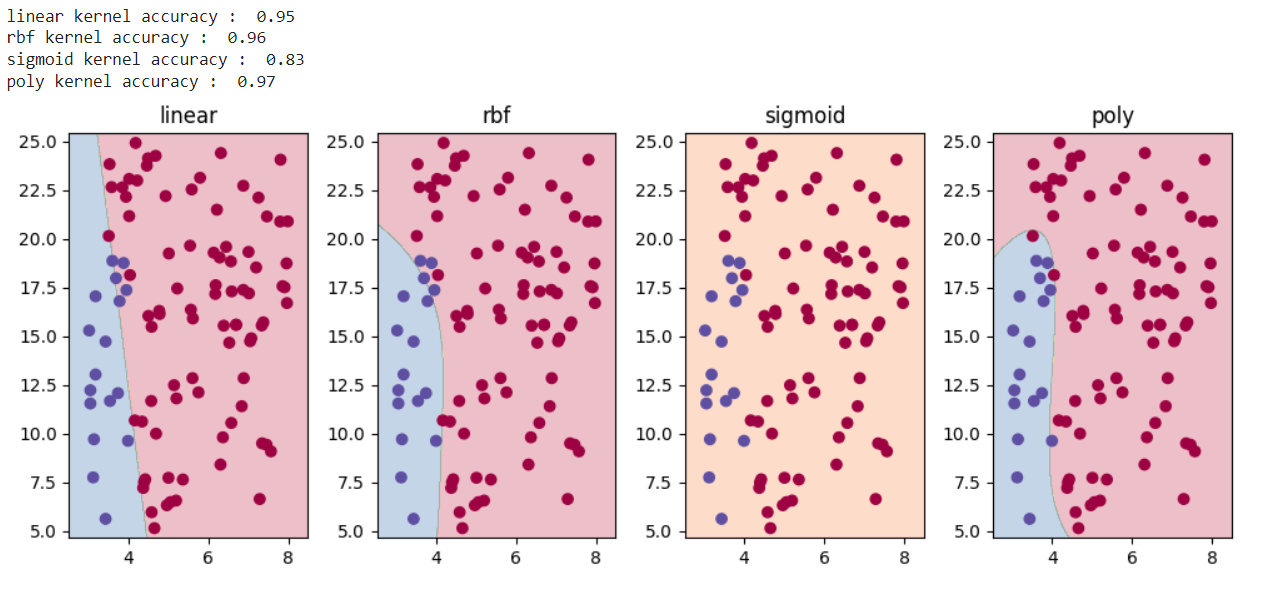
در اين قسمت plot انجام مي‌شود.

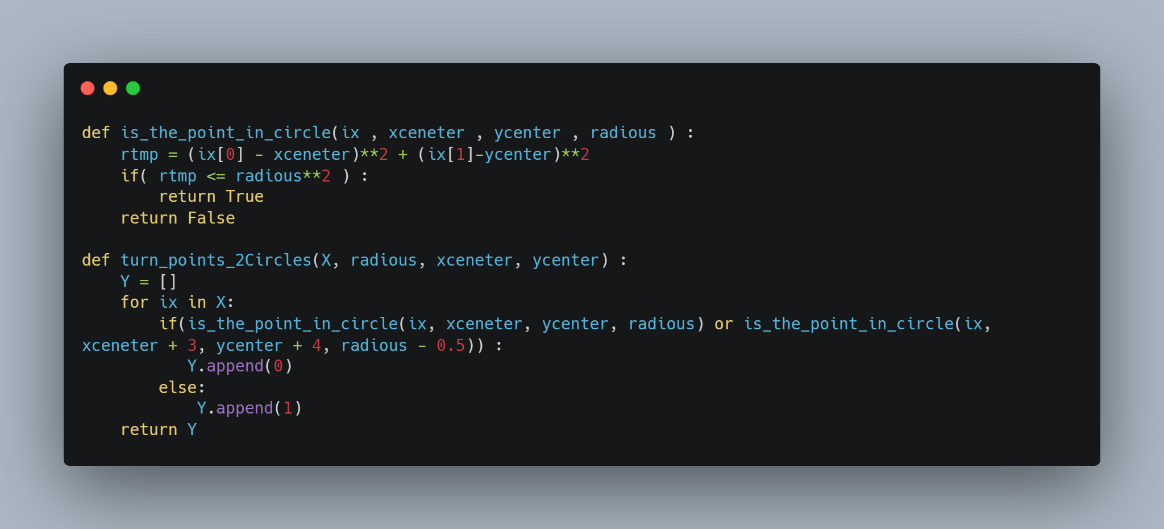
**نتايج:**





یک تابع makePoints دیگر تعریف می کنیم تا نقاط ورودی تولید شده شکل دیگری داشته باشند و مراحل قبلی را طی می کنیم تا train انجام شود. نتایج به صورت زیر است:

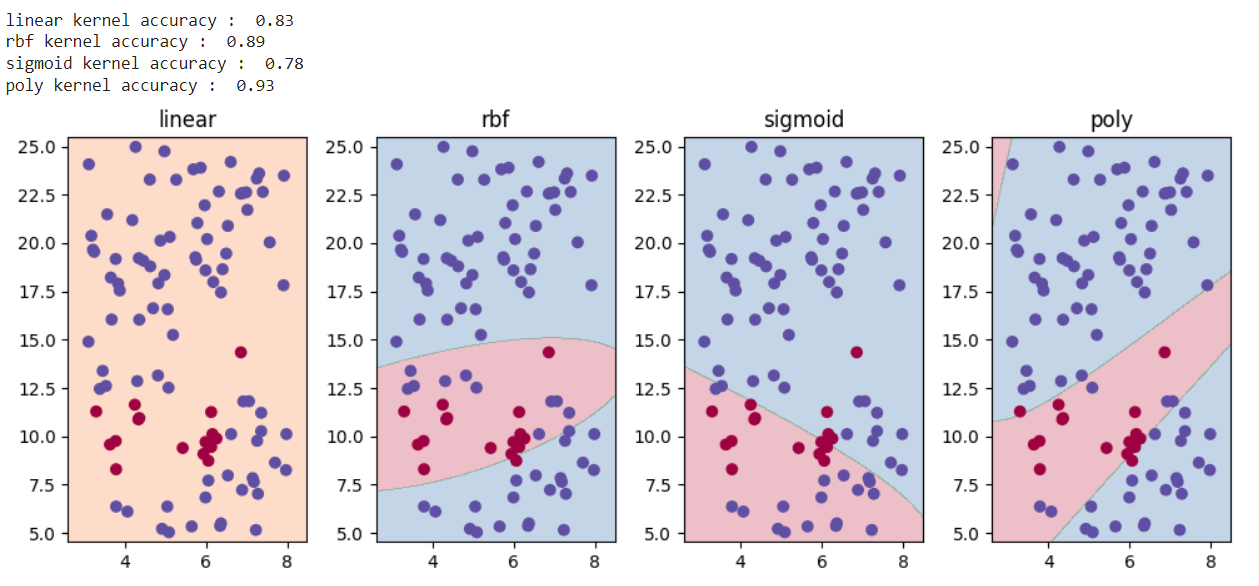


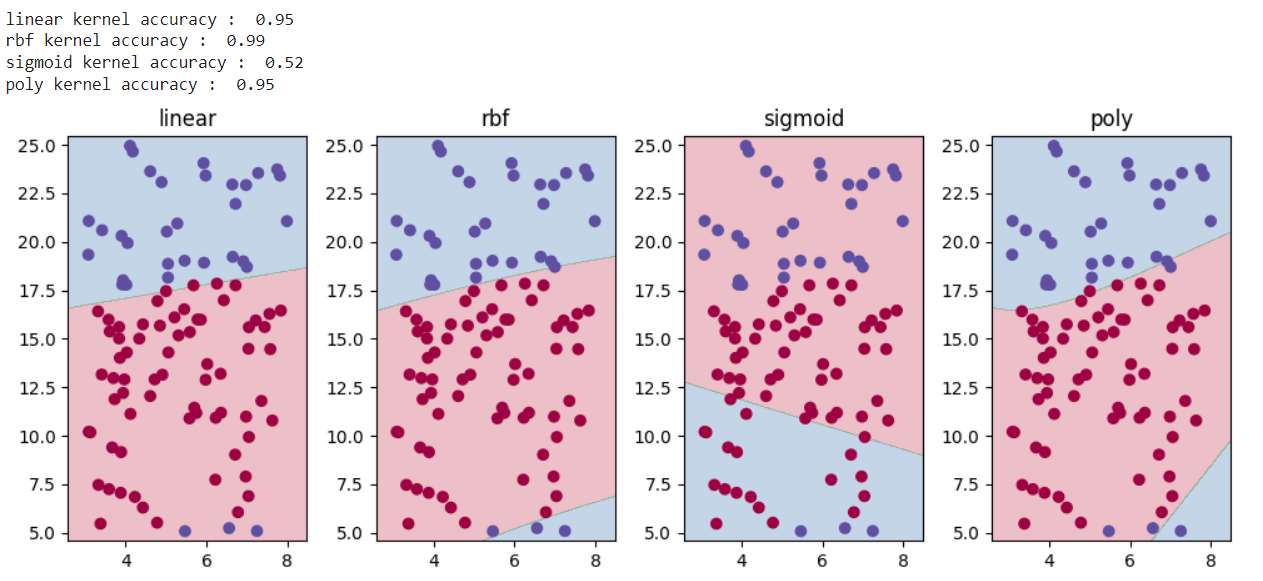


این بار تابع تولید نقاط را به صورت دایره ای در نظر می گیریم؛ صدا کردن تابع به صورت زیر تغییر می کند:

label = turn\_points\_2Circles(dataset, 5, 4, 10)

نتایج به صورت زیر است:

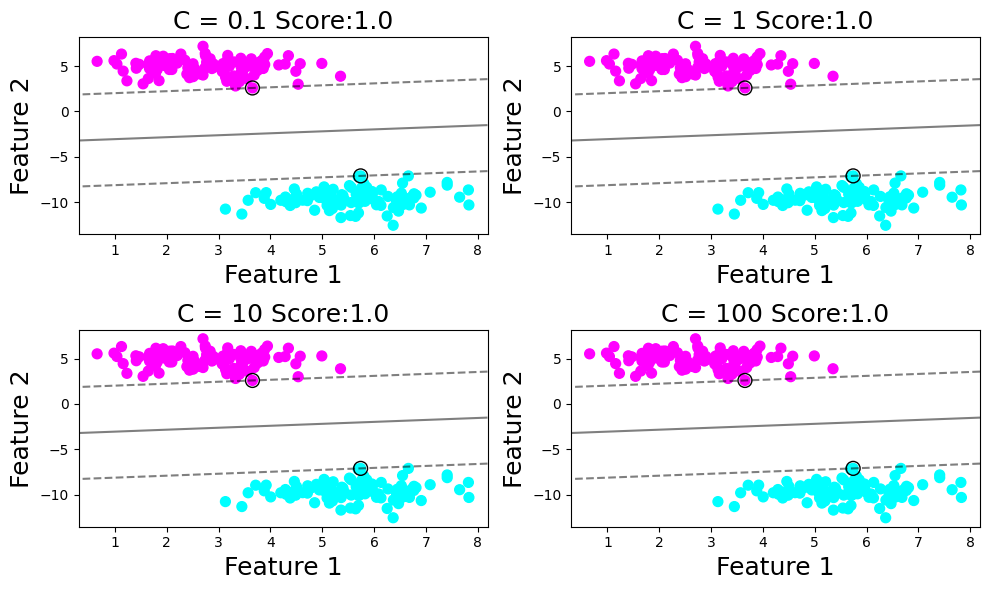






در این قسمت تعدادی تابع جدید تعریف می کنیم تا داده های جدید را تولید، مدل svm را train و نتایج را رسم کنیم. در این قسمت support vector ها نیز رسم می شوند. Train کردن مدل svm با استفاده از یک linear kernel، پارامتر regularization تحت عنوان c، داده ورودی x و برچسب های y انجام می شود.

همچنین در این قسمت یک boundary نیز داریم که توسط تابع plot\_decision\_boundary ترسیم می شود.



در قسمت بعد با استفاده از GridSearchCV بهترین پارامتر برای SVM model را پیدا می کنیم:

* ابتدا با استفاده از تابع generate\_linearly\_separable\_data با دو کلاس 0 و 1 تعدادی داده تولید می کنیم.
* داده های تولید شده را train می کنیم.
* در تابع calculate\_best\_params، یک grid از پارامترها به عنوان ورودی دریافت می شود. ابتدا یک SVM classifier یا clf بدون پارامتر تولید می شود. سپس یک GridSearchCV ساخته می شود که clf و grid پارامترها را به عنوان ورودی دریافت می کند. پارامتر cv تعداد foldهای cross-validation را مشخص می کند (در اینجا 3). سپس GridSearchCV، fit می شود.

**بخش 2:**

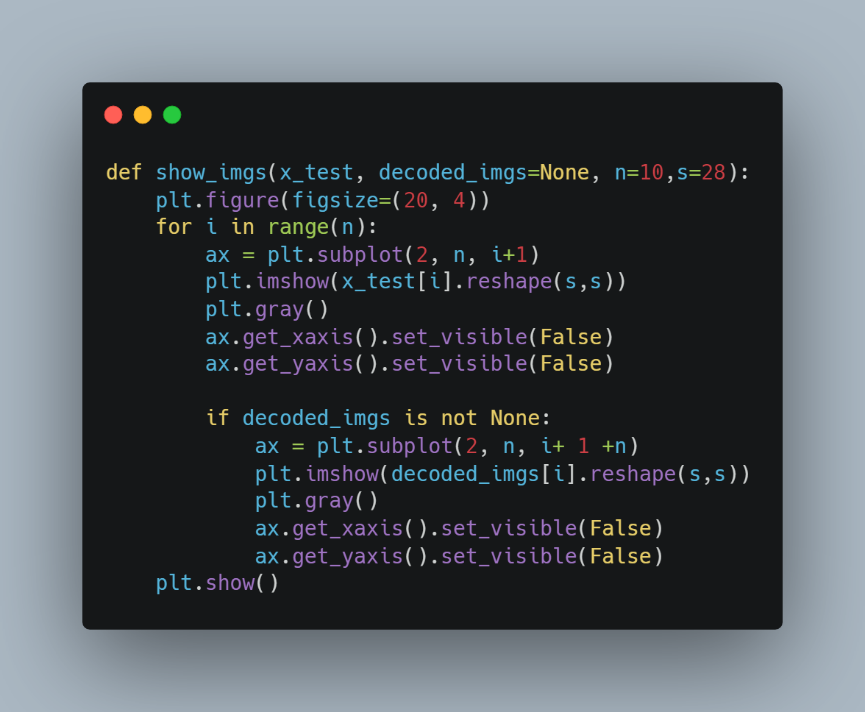
****

ابتدا کتابخانه های لازم را import می کنیم.

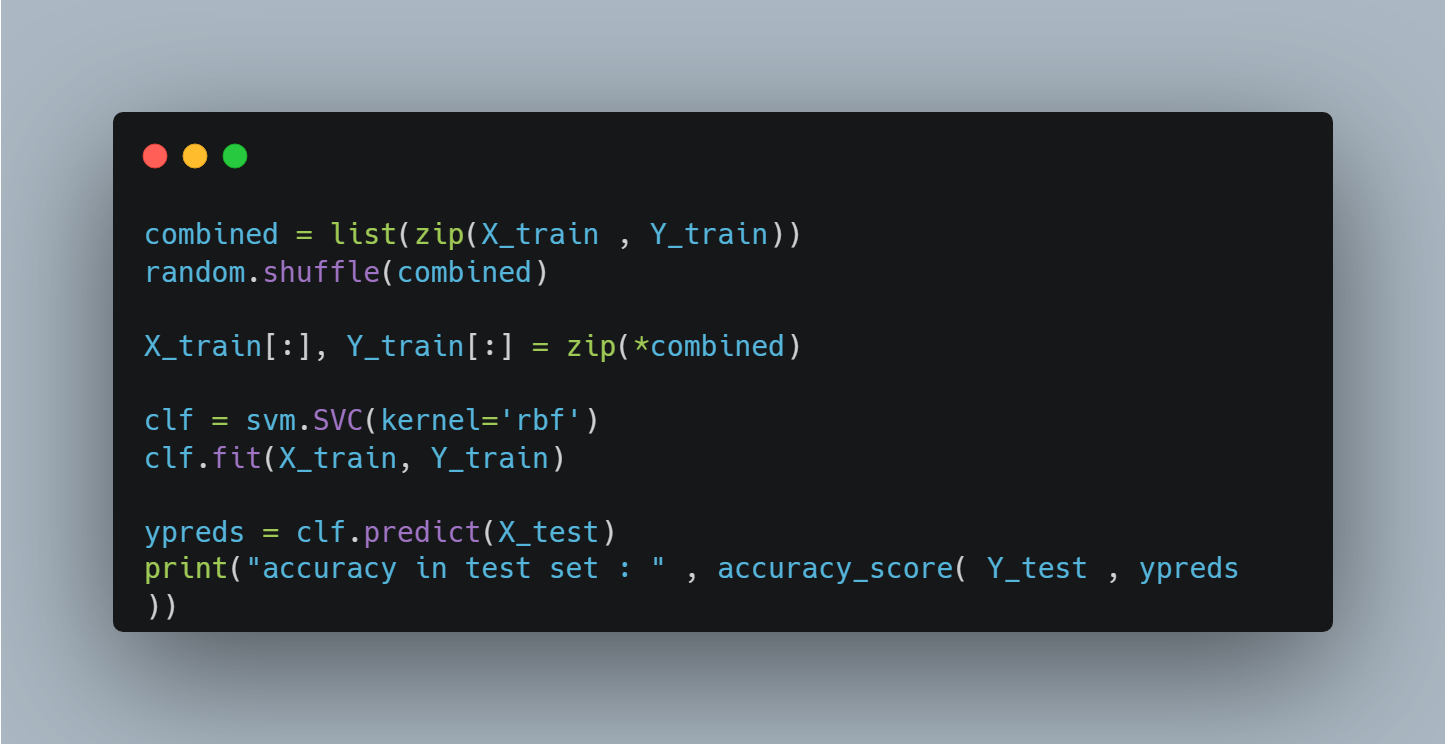
****

دیتاست MNIST را با استفاده از mnist.load\_data()، load کرده و آن را به داده های train و test تقسیم می کنیم.

دیتا را به grayscale تبدیل می کنیم و مقادیر پیکسل ها را نرمال سازی می کنیم. سپس دیتا را reshape می کنیم تا مطابق با ورودی مورد انتظار مدل باشد.

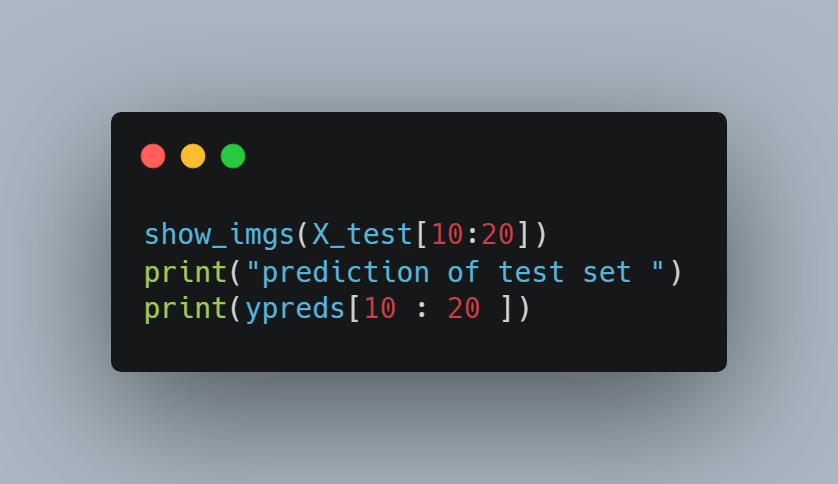
****

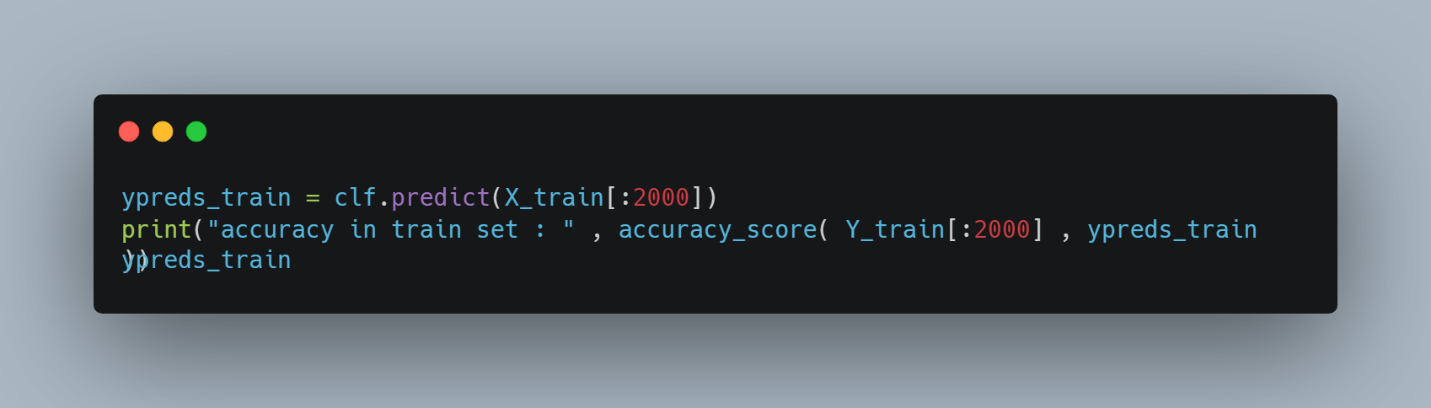
تابعی تحت عنوان show\_imgs تعریف می کنیم و به کمک آن تصاویر دیتاست را نمایش می دهیم.

****

دیتاست را به صورت تصادفی shuffle می کنیم. یک svm classifier یا clf با radial basis function (rbf) به عنوان kernel ساخته و دیتای train را fit می کنیم.

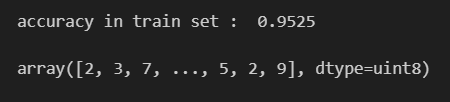
روی دیتای تست پیش بینی انجام می دهیم و دقت را با استفاده از accuracy\_score می سنجیم.

****

****

زیرمجموعه ای از داده های تست را با پیش بینی آن نمایش می دهیم.

نتیجه:

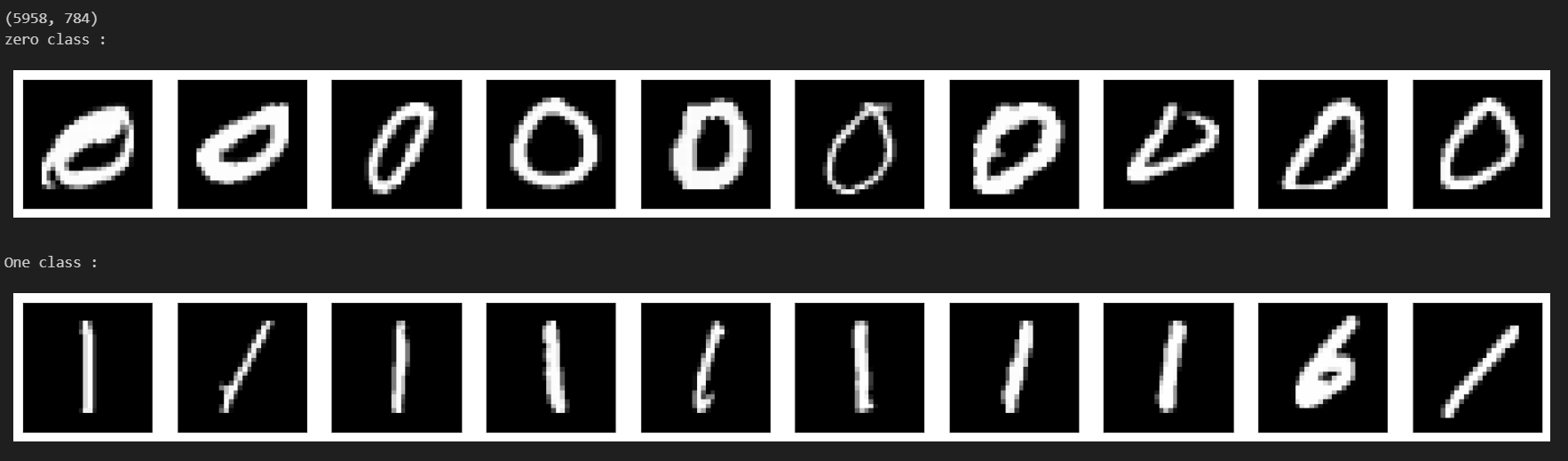




****

یک دیکشنری می سازیم و دیتای ورودی را بر اساس برچسب کلاس هایشان در این دیکشنری قرار می دهیم.

نتیجه:

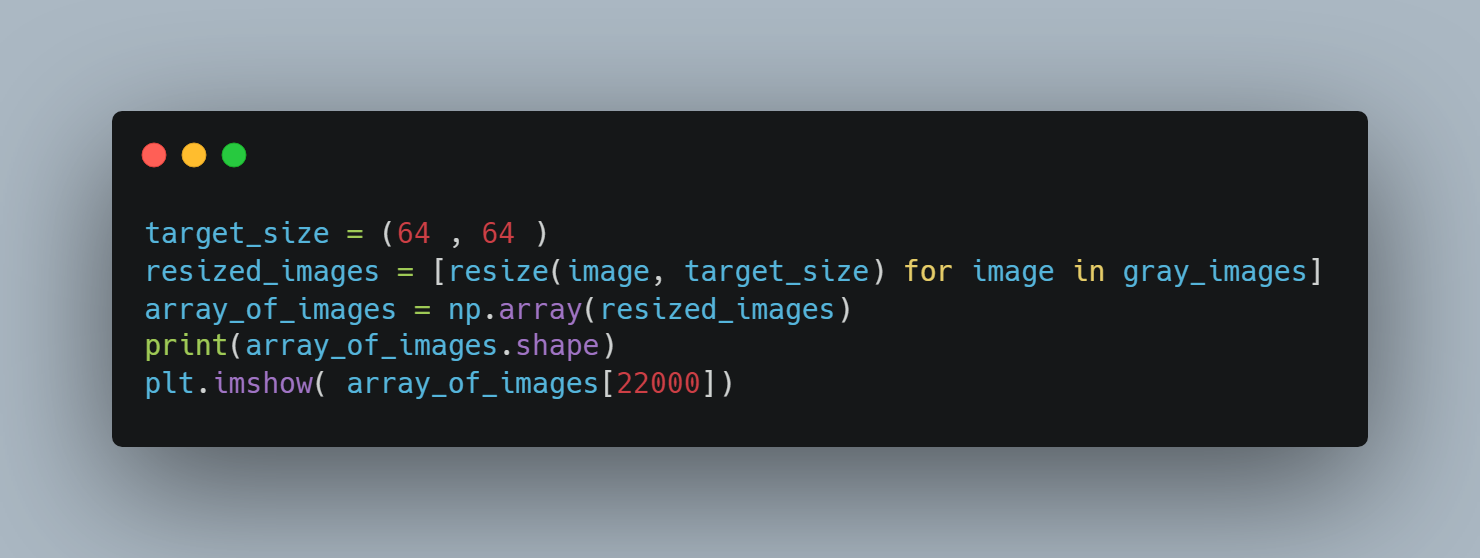


**بخش سوم:**

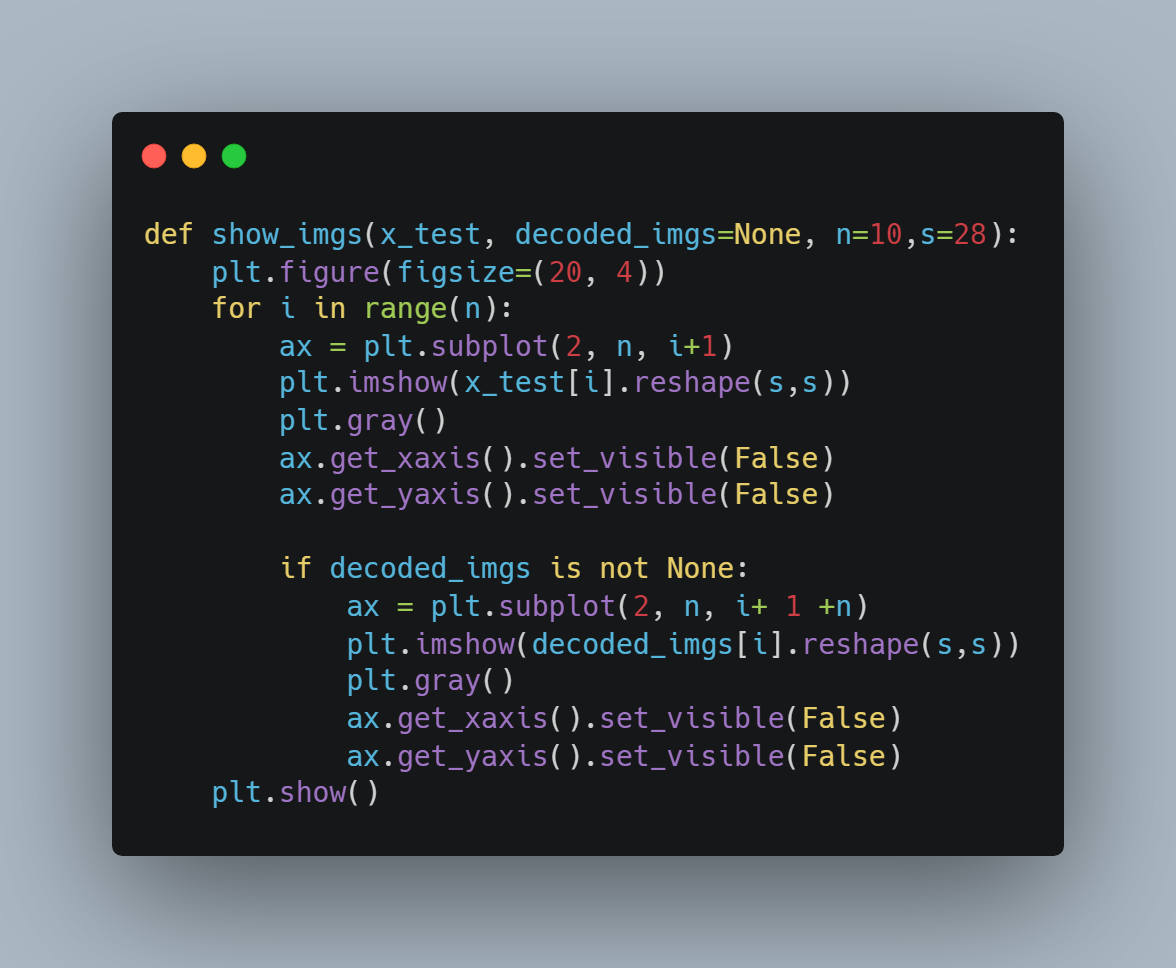




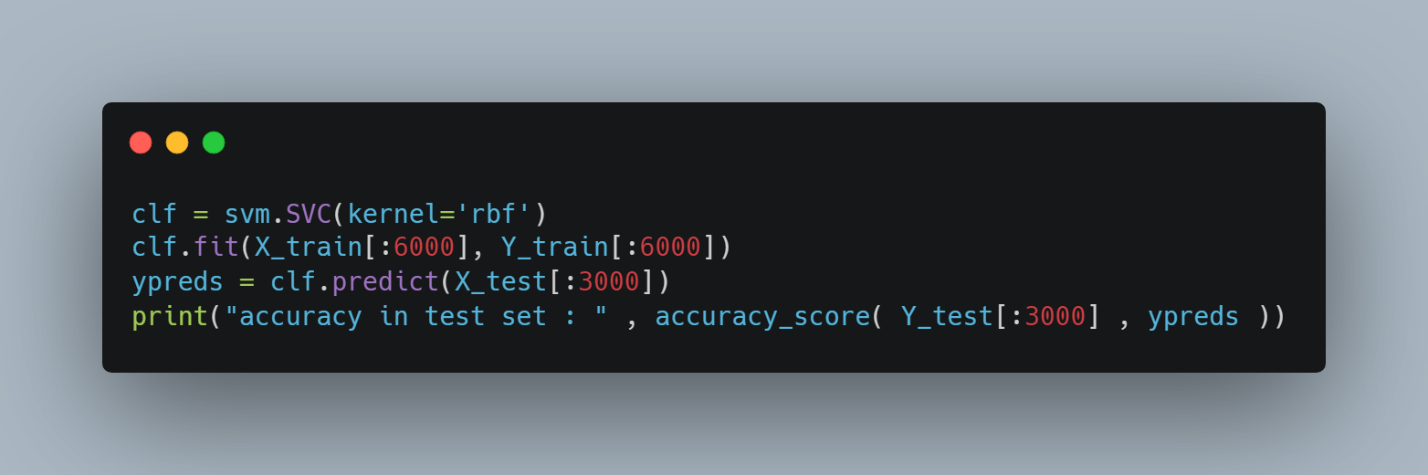
پس از import کردن کتابخانه های لازم، توابع مربوط به load کردن تصاویر را می نویسیم. پس از load کردن تصاویر با استفاده از load\_images آنها را به grayscale تبدیل می کنیم.



اندازه تصاویر را به 64 در 64 پیکسل تغییر می دهیم.



از تابع show\_image برای نمایش تصاویر استفاده می کنیم.



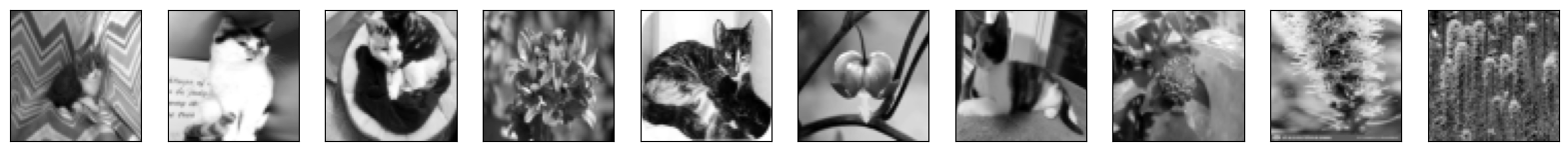
تصاویر را به صورت تصادفی shuffle کرده و به داده های train و test تقسیم می کنیم.

سپس با استفاده از rbf kernel در svm آنها را train می کنیم.

دقت:

accuracy in test set : 0.5

نتایج:



prediction of test set

cat cat flower flower flower cat cat