به نام خدا

نام و نام خانوادگی:

فرناز خوش دوست آزاد

شماره دانشجویی:

99521253

نام استاد:

دکتر عبدی

نام درس:

هوش مصنوعی و سیستم های خبره

نام پروژه:

پروژه support vector machine

بخش اول

در بخش اول ابتدا رندوم دیتا را با تنها یک feature تولید می‌کنیم که نمونه‌های آن در کد موجود می‌باشد و در مثال‌های جلوتر استفاده از یک تابع که تنها تعداد نمونه‌ها را به آن می‌دهیم داده‌ای رندوم تولید می‌کنیم و این کار را با استفاده از کتابخانه و متودهای از پیش تعریف شده می‌نویسیم و با استفاده از کتابخانه‌ی seaborn دو کلاس را در نمودار با دو رنگ متفاوت نمایش می‌دهیم. و در جلوتر از کتابخانه‌های آماده برای تولید شکل ماه و.. نیز استفاده کردیم و ضریب خطا و .. را نیز به آن داده‌ایم.

حال باید به سراغ پارامترهای دیگرsvm برویم و حالت‌های مختلف را مورد بررسی قرار دهیم.

الگوریتم SVC دارای دو پارامتر اصلی است:

Kernel

این پارامتر نوع هسته را برای مرز تصمیم مشخص می کند .هسته رایج ترین هسته، هسته تابع پایه شعاعی (RBF) است، اما هسته های دیگر مانند خطی، چند جمله ای و سیگموئید نیز می توانند استفاده شوند .انتخاب هسته به نوع داده ای که استفاده می شود و تعادل مورد نظر بین دقت طبقه بندی و بیش برازش بستگی دارد.

C

این پارامتر تعادل بین طبقه بندی نادرست نمونه‌های آموزشی و پیچیدگی مرز تصمیم را کنترل می‌کند . مقدار بالاتر C منجر به مرز تصمیم پیچیده‌تر می‌شود که بیشتر احتمال دارد نمونه‌های آموزشی را به درستی طبقه بندی کند، اما ممکن است بیشتر احتمال داشته باشد که داده‌ها را بیش از حد برازش دهد . مقدار پایین‌تر C منجر به مرز تصمیم ساده‌تری می‌شود که کمتر احتمال دارد داده‌ها را بیش از حد برازش دهد، اما ممکن است کمتر احتمال داشته باشد که نمونه‌های آموزشی را به درستی طبقه بندی کند.

علاوه بر این دو پارامتر اصلی، الگوریتم SVC دارای چندین پارامتر دیگر است که می توان برای بهبود عملکرد آن تنظیم کرد .این پارامترها عبارتند از:

- gamma: پارامتر gamma عرض هسته گاوسی استفاده شده در هسته RBF را کنترل می کند .مقدار بالاتر gamma منجر به هسته باریک‌تر و مرز تصمیم پیچیده‌تر می‌شود.
- Tol: پارامتر تحمل تعیین می کند که الگوریتم باید چقدر به همگرایی نزدیک شود قبل از اینکه آموزش را متوقف کند .مقدار پایین تر تحمل منجر به تکرارهای بیشتر و یک مدل دقیق تر می شود، اما ممکن است زمان بیشتری برای آموزش طول بکشد.
- Cache_size: پارامتر اندازه cache مشخص می کند که اندازه کش استفاده شده توسط الگوریتم برای ذخیره محاسبات میانی است .اندازه cache بزرگتر می‌تواند عملکرد را بهبود بخشد، اما همچنین استفاده از حافظه را افزایش می‌دهد.

در این بخش سه نوع داده مختلف از جمله دایره‌ای شکل، دو قسمت جدا و به صورت ماه با کمک از تابع های اماده کتابخانه Sklearn ساخته شده و با استفاده از کتابخانه plotly، و فیت کردن SVM بر روی داده ها، رسم شدند.

اگر پیچیدگی داده‌ها را بیشتر کنید، چه تاثیری در انتخاب هسته و چه تاثیری در پارامترهای آن هسته خواهد داشت؟ قدری تَفَلسُف کنید.

در صورتی که به جای دو cluster از کلاس‌های بیشتری استفاده کنیم، مدل ما نمی‌تواند از تنها یک خط استفاده کند و علاوه بر آن پیش‌بینی خوبی نخواهد داشت.

بخش دوم

همانطور که در کد ملاحظه می‌شود داده با ویژگی kernel = ‘rbf’ و c = 1 و cache_size = 900

بیشترین accuracy را دارد و kernel = ‘linear’ و c = 1 از دیگر نمونه‌های خوب ما می باشد.

بخش سوم

مراحلی که در روند پروژه انجام شده به شرح زیر می‌باشد:

1- پیدا کردن و لود کردن دادگان تصویری:
اولین قدم لود کردن دیتاست می‌باشد که صرفا با استفاده از کتابخانه os جهت دادن ادرس به تابع و همچنین استفاده از imread و as_gray جهت خوندن تصاویر به کار می‌رود و در نهایت تبدیل هر ماتریس به صورت مقادیر هر خانه پشت سر هم و flat کردن آن ها و در نهایت ذخیره کردن در دیتا فریم با استفاده از لیست از جمله کارهایی است که در این مرحله انجام می‌دهیم.

2- تقسیم دادگان به دو بخش x و y و تقسیم آن‌ها و کمی EDA:
با استفاده از تابع train_test_spilit، مقادیر بدست آمده را به چهار بخش برای train و test تقسیم می‌کنیم و سپس با استفاده از describe و info به بررسی داده‌ها پرداخته تا حس بهتری جهت مقدار دهی به پارامترها داشته‌باشیم و یا اگر نیاز به preproccessing بود، انجام بدهیم.

3- PCA :
با استفاده از PCA تعداد ویژگی ها را به 35 تا کاهش می‌دهیم با این کار از بیش برازش جلوگیری می‌کنیم و کمبود امکانات سخت افزاری جهت محاسبه و نبود زمان را پوشش می‌دهیم.

4- استفاده از gtrid search :
برای بررسی مقادیر مختلف هایپر پارامترهای SVM ، از جست و جوی شبکه ای استفاده میکنیم، به این صورت که اول مقادیر مختلفی را برای هریک از پارامتر های آن اختصاص داده و آن با جست و جو بر روی آن ها بهتر مدل را به ما میدهد همچنین با بهترین پارامتر ها.

5- Double grid search :
این ایده ابتکاری هست برای دقت بیشتر در نتیجه، برای اینکه بهترین مدل تخمین زده شود، برای بار دوم مدل با پارامتر های نهایی که در بخش قبل به دست آمدند در اطراف آن ها گرید سرچ شده و باز هم مقادیر دقیق تر و بهتری استفاده میشود.

6- Cross validation :
با استفاده از cross validation و درواقع بررسی مدل به دست آمده با اموزش در بخش های مختلف قسمت اموزشی و ازمایش آن، به میانگین دقت به دست آمده دست پیدا میکنیم و میتوانیم بعدا مقدار بیش برازش و یا دقت حدودی که در پیش بینی خواهد داشت را بررسی کنیم.

7- پیش‌بینی و بررسی بر روی داده‌های آزمایش:
با استفاده از توابع آماده، مدل را بر روی داده‌های آزمایش تست کرده و مقادیر

پیشبینی شده را بررسی می‌کنیم، با استفاده از classification مقادیر دقت و ... را بررسی می‌کنیم.

8- با استفاده از مقادیر به دست آمده در آخر و همچنین مقادیر cross validation ، متوجه می‌شویم که بیش برازش رخ نداده و داده ها به خوبی پیش بینی شده اند، گرچه دقت به دست آمده به نسبت بسیار کم می‌باشد.

چالش های در طول اجرا:

– تعداد ویژگی ها بسیار زیاد و مشکل نداشتن امکانات کافی برای محاسبه دادگان:
این مشکل هم با استفاده از pca واستفاده کردن از دادگانی دیگر حل شد.

– مشکل خواندن داده‌های تصویری :
توسط کتابخانه‌های توضیح داده شده این مشکل هم حل شد.

برای مشکل داده‌های حجم زیاد اول به فکر کم کردن با استفاده از حذف رندوم داده‌ها انجام شد، اما به دلیل تفاوت کردن توزیع ها مانده و اولیه، و همچنین نیازمند بودن به انجام خودکار آنها، این ایده به نتیجه نرسید.