تمرین سری دوم - مبانی یادگیری ماشین

محمد زیاری - ۹۷۲۲۲۰۴۷

بخش اول

در ابتدای کار قابل ذکر است که نسبت به تمرین قبلی ستون livingspacerange که حذف شده بود ، حذف نشد. به دلیل آنکه پس از اینکه دقت پایینی مشاهده کردم ، با بررسی فهمیدم این فیچر بیشترین کوورلیشن را داراست. پس تنها تغییر در داده ها در همین بخش بود.

برای این بخش نیاز است که برای حالت های مختلف ۵ فولد و ۱۰ فولد را پیاده سازی کنیم و معیار های MSE و ACCURACY را برای آنها بررسی کنیم.

ابتدا بدون استفاده از پکیجی رگرسیون را بدست میاوریم. نتایج که دقت و خطا است در کد مشخص شده است. در این روش ما هر سری یک بخش را تست میگیریم و بقیه را داده آموزشی و دقت و خطا را برای آن محاسبه میکنیم. من تنها ۵فولد را برای آن پیاده سازی کردم. فکر میکنم با کلاس بندی مناسبتر زدن ۱۰فولد هم سخت نبود اما با این روش تعداد خطوط کد بالا میرفت و به عنوان نمونه ۵فولد با استفاده از رگرسیون بدون پکیج وجود دارد.

برای سوال دوم و سوم هم طبق فیچرهایی که خواسته شده مدل رگرسیون را فیت میکنیم و نتایج مشخص است. برای مثال نتیجه سوال سوم بهتر از سوال دوم است زیرا از فیچرهای بیشتری در آن استفاده شده است.

در بخش های بعدی که خواسته شده به ازای فیچرهای دلخواه مدل فیت شود ، من از کل فیچرها استفاده کردم ابتدا رگرسیون خطی را بر روی تمامی فیچر ها اعمال کردم و خروجی هر fold را نشان دادم.

دو حالت آخر درمورد دو رگرسیون lasso , ridge است. هردو رگرسیون را توسط پکیج خود پیاده کردم و نتایج kfold نیز برای آن ها مشخص شده. همانطور که میدانید نتایج خروجی این رگرسیون ها به انتخاب درست ضریب آلفا نیز وابسته است.

در سوال دوم همین بخش گفته شده که خروجی ها را باهم مقایسه کنیم. همانطور که گفته شد خروجی هر fold و نیز میانگین میزان accuracy و mse برای هرکدام از رگرسیون ها مشخص شده است. خروجی ها تقریبا یکسان هستند و تفاوت جزئی در آنها دیده میشود اما خروجی رگرسیون خطی و رگرسیون lasso بسیار به هم نزدیک است.

بخش دوم

۱. این دو رگرسیون برای کم کردن خطای مدل استفاده میشوند و با اضافه کردن جمله های penalty به جمله خطا باعث کم شدن آن و کم شدن پیچیدگی مدل میشوند.

در رگریسون ridge جمله‌ای که برای regularization اضافه میشود همان توان دوم ضرایب ویژگی های ما (βi) است، که نرم دو نیز به آن گفته میشود. همچنین ضریب ترم λ (regularization) نیز باعث میشود تا مقدار این ضرایب کنترل شوند.اگر مقدار آن نزدیک به صفر باشد، مدل ما همان رگرسیون معمولی میشود، همچنین اگر مقدار آن خیلی بزرگ شود باعث میشود که مدل دچار مشکل شود و نتیجه خوبی ندهد. پس انتخاب کردن این ضریب بسیار مهم است. اگر مقدار ضرایب فیچر هابسیار بزرگ باشد، جمله خطا ما نیز بزرگ خواهد بود و میزان تاثیر آن کمتر میشود و باعث میشود تا این جمله به عنوان خطای مدل در نظر گرفته نشود. فیچر هایی که تاثیر کمتری دارند به سمت صفر میل میکنند و آن فیچرهایی که تاثیر بیشتری دارند، بزرگتر خواهند بود.

رگرسیون lasso نیز همانند ridge عمل میکند، با این تفاوت که جمله پنالتی که اضافه میشود جمع قدرمطلق ویژگی ها است، که نرم یک نیز به آن گفته میشود. تفاوت خیلی بزرگی که با رگرسیون ridge دارد آن است که میتواند برخی فیچرها را دقیقا صفر کند و این فیچرها کاملا در خروجی بی تاثیر میشوند. همین یک برتری نسبت به ridge محسوب میشود، زیرا بعد داده ها را نیز کم میکند و پیچیدگی مدل را خیلی کمتر خواهد کرد.

۲. پارامتر آلفا در این رو رگرسیون که به آن shrinkage parameter نیز گفته میشود بسیار مهم است زیرا در نتیجه نهایی و خروجی که این رگرسیون ها میدهند، تاثیر بسزایی دارد. به همین دلیل پیدا کردن بهترین آلفا خیلی مهم است. یکی از راه ها برای پیدا کردن آن، پیداره سازی مدل توسط چندین آلفا متفاوت به صورت cross validation است. به این صورت که توسط لیستی از آلفا های مختلف مدل را ارزیابی میکند و بهترین نتیجه را میدهد.

۳ . به نظر دلیل بیشتر کردن مقدار آن است که اگر تعداد داده ها بیشتر باشد مثلا تقسیم به ۴ قسمت یا فولد شاید خیلی نتیجه درستی به ما ندهد و ما باز هم به عنوان دیتا ست بزرگی به آن نگاه کنیم . البته از آن طرف بیشتر شدن بیش از حد تعداد فولد ها ممکن است در ابتدا بار محاسباتی بالایی داشته باشد و در وهله بعدی ممکن است داده های پرت را در یک فولد داشته باشیم و در نظر بگیریم که خیلی نمیتواند نتیجه ایده آلی داشته باشد. بنابراین طبق منابعی که مطالعه کردم اول از همه آنکه روی دیتاست های مختلف ۱۰ معمولا جواب خوبی میدهد و در آخر در جواب به سوال گفت که دلیل بیشتر کردن فولد همان تعداد instance هاست. هرچند باید بین هزینه محاسباتی و همچنین نتیجه درست معیار مشخصی قرار دهیم تا تعداد فولد مناسب را انتخاب کنیم

۴ . نمونه خاصی از cross-validation است که در آن تعداد instance ها با تعداد فولد ها برابر است. همانطور که در سوال قبلی هم اشاره شد بیشتر در مواقعی که دیتاست کوچکی داریم یا نتیجه حدس برایمان اهمیت بالایی دارد در حالی که بار محاسباتی

خیلی حایز اهمیت نیست از آن استفاده میکنیم. این متد برای همه instance ها به ترتیب یکی از آنها را train در نظر میگیرد و بقیه را تست در نظر میگیرد.

۵. Bootstrapping یک روش برای نمونه گیری از داده اصلی است که با جایگذاری انجام میشود یعنی هر داده ای که وارد نمونه گیری میشود دوباره پس از آنکه کار آن تمام شد به دیتاست برمیگردد و باز هم میتواند در نمونه گیری های بعدی شرکت داده شود. می توانیم این کار را بارها انجام دهیم و در نهایت برای ارزیابی مدل، تمامی نتایج این نمونه گیری ها را میانگین گرفته و به عنوان score اعلام کنیم. همین که در نهایت این ارزیابی ها با یکدیگر تجمیع می شوند، میتواند برای روش های ensemble نیز مفید باشد. در واقع از bootstrapping در روش های ensemble نیز استفاده میکنند.

تفاوت آن با cross validation آن است که در cross validation ما دیتا را به K بخش تقسیم میکنیم و با استفاده از k-1 بخش ترین و با یک بخش دیگر مدل را ارزیابی میکنیم. سپس در نهایت از تمام فولد ها میانگین میگیریم. درواقع cross validation بر کل داده احاطه دارد و از تمامی آن استفاده میکند و روش بدون جایگذاری است.

۶ . این روش به طور کلی آن است که ۵بار از ۲ فولد استفاده کنیم. در واقع داده ها با ۲ بخش تقسیم میشوند ، یکی تست و دیگری train و در هر مرتبه جایشان عوض میشود تا به جواب مناسبی برسیم

بخش سوم

در ابتدا داده ها را خواندیم. نیازی به تغییر داده ها نبود زیرا همه داده ها عددی بودند و داده نال نداشتیم. پس از آنکه تارگت را جدا کردیم داده ها را اسکیل کردیم تا به سراغ فیت کردن مدل برویم.

حالا در بخش اول مدلمان با استفاده از تمام فیچرها لاجستیک رگرسیون را پیاده سازی کردیم و نتایج خواسته شده نیز در کد قابل مشاهده است. قابل ذکر است هرکدام از موارد خواسته شده معین کننده چیزی هستند. مثلا recall تعداد درست هایی که درست حدس زده شده به همه درست ها میباشد و ... که هر کدام نسبت به هر لیبل درصدی دارند که نمایش داده شده است. نموداری در زیر آن رسم شده است که مربوط به سوال دوم است و نمایانگر آن است که داده ها به خوبی متوازن هستند.

در بخش بعدی کد داده ها طبق عمل خواسته شده به دو لیبل تبدیل شده اند. کلاس ۲ و۳ را توسط Dictionary به کلاس ۱ و کلاس ۰ را همانطور نگه داشتیم. سپس مدل برروی این دیتای جدید فیت شده و موارد خواسته شده گزارش شده اند.سپس خواسته شده تا داده ها بالانس شوند که برای این کار چند راه داریم. یکی upsampling که داده ها به اندازه بیشترین لیبل به صورت رندوم اضافه میشوند ، undersampling که به کوچکترین حد تعداد لیبل میرسند یا در واقع به اندازه کمترین لیبل حذف میشوند. میتوان از راه های دیگری هم استفاده کرد برای مثال از شبکه های مصنوعی یا حتی gan و نظیر آن که داده هایی مشابه با دیتاست بسازیم .
همچنین راهکار های دیگری نیز وجود دارد مانند آنکه از الگوریتم هایی استفاده کنیم که بر روی داده های نامتوازن به خوبی عمل کند، که ما در این سوال از undersampling استفاده کرده ایم و کلاسی که داده های زیادی داشت را به صورت رندم از ۱۵۰۰ تا به ۵۰۰ تا داده کاهش دادیم. سپس مدل را فیت کرده ایم. و نتایج را در کد مشاهده میکنیم.

برای بخش بعدی خواسته شده که از forward selection برای انتخاب ویژگی ها استفاده کنیم. در ابتدا برای ساده تر شدن کار تعداد ۴ لیبل را به ۲ لیبل کاهش دادیم و قیمت های پایین را LOW و قیمت های بالا را HIGH در نظر گرفتیم سپس به کمک پکیج feature_selection به پیاده سازی این روش می پردازیم. به انتخاب ۸ فیچر و با استفاده از ۵ فولد این عمل را انجام میدهد و در هر مرحله با استفاده از معیار auc و در جهت افزایش آن، فیچرهای مناسب را انتخاب میکند و به مجموعه اضافه میکند تا به ۸ تا برسد. برعکس برای backward selection انجام میدهیم یعنی از کل فیچرها شروع میکنیم و تا به ۸ فیچر برسیم، آن ویژگی هایی را که تاثیر منفی بر روی auc دارند حذف میکنیم و به این ترتیب بهترین فیچرها را انتخاب میکنیم و در هر حال مدل را با استفاده از آنها فیت میکنیم و نتایج را مشاهده میکنیم.

در ادامه کد خواسته شده برای pca با تعداد ۸ ویژگی زده شده، از آنجایی که pca به صورت دلخواه و بدون هیچ معیاری ویژگی ها از انتخاب میکند، به همین دلیل است که میزان دقت آن از دو حالت قبلی کمتر خواهد بود.

در بخش آخر هم ۵ فولد و ۱۰ فولد زده شده و نتایج خواسته شده گزارش شده است. روش دقیقا مطابق چیزی است که در بخش ۱ پیدا کردیم و مقدار accuracy نیز گزارش شده است.

بخش چهارم

۱ . همانطور که میدانیم مسایل لاجستیک رگرسیون برای دو کلاس تعریف شده است. اما برای آنکه بر روی چند کلاس آن را تعریف کنیم باید کلاس بندی مجدد انجام دهیم. کلاسبندی مجدد باید باینری یا در واقع به صورت صحیح و غلط جواب دهند و معیار ها هم بر این اساس تعریف میشوند که اگر کلاس را درست تشخیص بدهیم که tp یا tn اما اگر اشتباه تشخیص بدهیم fp یا fn است. در واقع تقسیم بندی بین چند کلاس و مقایسه نیز میان چند کلاس بررسی خواهد شد .

۲ . همانطور که میدانیم در حالتی که داده ها بالانس تقسیم نشده اند ، مدل به سمتی پیش میرود که بیشتر داده هایی که روی آن شک دارد را به طرف آنکه سهم بیشتری دارد حساب میکند. به نوعی به جای یادگیری در آن حفظ کردن صورت میگیرد. ما در اینجا برای حل این مشکل از روش undersampling استفاده کردیم. به طور کلی اتفاقی که باید شاهدش میبودیم آن بود که به جای آنکه recallپایینی در سمتی که داده کمتری وجود دارد داشته باشیم اما ... که البته شاید کمتر کردن دیتاها در واضح نبودن جواب تاثیر داشته و استفاده از upsampling شاید نتیجه بهتری می داشت. البته اتفاق مثمر ثمری که افتاد آن بود که به طور کلی f1score هر دو کلاس با هم برابر شد که نشان از آن است که مدل ما برخلاف سوال ۴ بیشتر به سمت یکی از داده ها کشیده نشده است

۳ . همانطور که در کد پیاده سازی کرده بودیم، مشاهده میشود که استفاده از forward selection تاثیر مثبتی بر روی نتایج به دست آمده داشته و دقت مدل کمی بیشتر از حالت قبل است. ما از feature selection به این خاطر استفاده میکنیم که تعدادی از آن فیچرهایی که داریم را انتخاب کنیم، طبیعتا وقتی با تعداد فیچرهای کمتری کار میکنیم پیچیدگی مدل ما کمتر خواهد بود و هرچه فیچرهایی که تاثیر مثبتی بر روی نتیجه نهایی دارند، تعداد بیشتری داشته باشند، نتیجه ما نیز بهتر خواهد بود.

۴. ۲ روش در پایین توضیح داده خواهند شد:

روش انتخاب ویژگی امتیاز خی ۲ :(Chi-squared) این روش انتخاب ویژگی از روش های فیلتر است. این روش اختلاف معنادار میان observed frequency و expected frequency دو ویژگی categorical را میسنجد. فرض صفر در این روش بیان میکند که هیچ گونه همبستگی میان این دو متغیر وجود ندارد. از این روش برای سنجش رابطه میان ویژگی های مختلف موجود در یک مجموعه داده و ویژگی هدف استفاده میشود و ویژگی هایم مناسب انتخاب میشوند.

$$X^2 = \frac{(Observed\ Frequency - Expected\ Frequency)^2}{Expected\ Frequency}$$

روش حذف بازگشتی ویژگی: این روش جز روش های انتخاب ویژگی wrapper است. روش recursive feature elimination(RFE) یک روش حریصانه برای انتخاب ویژگی است. در این روش ویژگی ها به طور بازگشتی و با در نظر گرفتن مجموعه های کوچک و کوچک تر از ویژگی ها انتخاب میشوند. در این روش ویژگی ها بر اساس مرتبه حذف شدن آن ها از فضای ویژگی رتبه بندی می شوند. ابتدا بر روی یک مجموعه اولیه از ویژگی ها آموزش داده میشود و سپس با استفاده از معیار های مختلف مانند ضریب و یا مهم بودن فیچر ها، فیچرهای خاصی انتخاب میشوند و تعداد خاصی از کل را انتخاب میکنیم.

۶. روش LDA مشابه روش PCA سعی دارد تا بعد داده ها را کم کند. اما به وسیله projection این کار را انجام میدهد. برای داده هایی که پراکندگی زیادی دارند و تفکیک پذیری آنها سخت است، LDA بسیار کارآمد است

LDA به ۲ مورد بسیار وابسته است. ۱. میانگین داده ها ۲. میزان پراکندگی آنها. در واقع تلاش دارد تا با تصویر کردن داده ها بر روی یک محور جدید، فاصله میان میانگین کلاس ها را زیاد کند و پراکندگی داده ها (scatter) را به حداقل برساند. درواقع هدف LDAآن است که نسبت این دو مورد را حداکثر کند و در نهایت پراکندگی داده ها را طوری ارائه دهد که تفکیک پذیری و کلاس بندی آنها به راحتی قابل انجام باشد.

۸ . این روش برای کلاس بندی ۲ تایی به کار میرود و یک عدد خروجی میدهد. ۱ نشانه موافقت بین حدس ما و مقدار واقعی ، ۱- نشانه مخالفت و ۰ نشان دهنده آن است که پیشبینی دقیقی نداشتیم. همانطور که گفته شد این روش برای کلاس های باینری و در واقع ۲ کلاس به کار میرود. این روش با استفاده از فرمولی محاسبه میشود که تشکیل شده از fn،fp،tn،tp ها است که همان جواب های درستی که درست تشخیص داده ایم ، غلطی که درست تشخیص داده ایم و ... است که گفتیم جواب به دست آمده از آن فرمول به چه منزله ای است و چه معنایی میدهد.