تمرین سری دوم – یادگیری ماشین

مهرانه مقتدایی فر - ۹۷۲۲۲۰۸۶

گزارش بخش ۱:

سوال ۱:

همانطور که در تمرین قبل داشتیم، دیتاست داده شده را پاکسازی کرده بودیم. با استفاده از همان داده های پردازش شده، سوالات این بخش را نیز پیاده سازی میکنیم. خواسته شده که k-fold cross validation را برروی رگرسیون های متفاوت پیاده سازی کنیم. برای اینکار در هر حالت ابتدا رگرسیون خواسته شده را پیاده کردیم (حالت ۱ مجزا نوشته شود)

برای حالت های دیگر با استفاده از پکیج model_selection و دستور cross_val_score دو معیار متفاوت را بررسی میکنیم. از آنجایی که 5-fold و 10-fold خواسته شده، هردو را درنظر گرفته و با دو معیار accuracy و MSE نتایج را بدست آورده‌ایم. به طور پیش فرض وقتی از cross_val_score استفاده میکنیم، چیزی که خروجی میدهد همان مقدار دقت مدل یا accuracy است. برای معیار mse نیز باید scoring = neg_mean_squared_error قرار دهیم، این مقدار منفی mse را به ما میدهد اما مقدار مطلق آن همان mse موردنظر است.

همچنین برای هر مدل، میزان دقت و mse بدون k-fold نیز محاسبه شده و برای مقایسه قرار داده شده است.

حالت ۲ رگرسیون برای یک فیچر با بیشتری کورولیشن که همان livingspacerange هست پیاده سازی شده و میزان mse و accuracy آن مشاده میشود.

حالت ۳ در مجموع ۴ فیچر همانطور که در سوال داده شده در نظر گرفتیم.

در حالت ۴، ۸ فیچر دلخواه در نظر گرفته شده. در حالت ۵ و ۶ نیز با استفاده از پکیج مورد نظر دو رگرسیون ridge و lasso را پیاده سازی کردیم.

در تمامی حالت ها خروجی های هر fold و در نهایت میانگین mse و accuracy ها بدست آورده شده.

سوال ۲:

رگرسیون خطی و رگرسیون ridge و lasso را برای حالت ۴ سوال قبل، یعنی با ۴ تا فیچر پیاده سازی کرده‌ایم.

همانطور که در خروجی کد برای هر رگرسیون میبینیم، رگرسیون خطی و رگرسیون lasso بسیار خروجی های نزدیک به هم دارن و از میزان accuracy آنها از رگرسیون ridge بیشتر است. تفاوت بسیار جزئی دیده میشود اما خروجی رگرسیون lasso بسیار مشابه به رگرسیون خطی ما است. البته خروجی این دو رگرسیون lasso و ridge بسیار به تعیین هایپرپارامتر آنها (alpha) بستگی دارد و با تغییر alpha به خصوص در رگرسیون lasso تغییر زیادی در خروجی مشاهده شد، تا با تغییر alpha به 0.001 خروجی آن مطلوب شد.

پاسخ سوالات بخش ۲:

۱.دو رگرسیون RIDGE و LASSO برای بهبود رگرسیون خطی ساخته شده‌اند. درواقع رگرسیون خطی خیلی از مواقع موجب overfittingمیشود و همچنین به ویژگی ها به یک وزن نگاه میکند، یعنی فیچرهایی که تاثیر بیشتری بر روی مدل دارند را از فیچرهایی با تاثیر کمتر، متمایز نمیبیند.

برای رفع این مشکل ها ، به بررسی دو رگرسیون RIDGE و LASSO میپردازیم.

رگرسیون RIDGE :

این رگرسیون با اضافه کردن یک جمله جدید به تابع هزینه، باعث میشود تا تاثیر برخی فیچرها، کمتر شود. این جمله همان جمع توان ۲ ضرایب فیچرهای ما است. به این جمله penalty گفته میشود. اضافه کردن این جمله درواقع آن ضرایبی را که بزرگ هستند، بزرگتر میکند و ضرایب کوچکتر را نیز کمتر میکند، همین باعث میشود تا تاثیر برخی فیچر ها کمتر شود. فیچرهایی با تاثیر بیشتر را بزرگتر میکند و فیچرهایی با تاثیر کمتر را کوچکتر. به همین علت باعث میشود تا تابع هزینه و میزان خطا کمتر شود و دقت مدل افزایش یابد.

همچنین یک ضریبی به اسم lambda یا alpha نیز در این جمله ضرب میشود، این ضریب یک هایپرپارامتر هست و اگر خیلی بزرگ باشد درواقع باعث میشود تا عملکرد مدل بدتر شود و به سمت underfitting برود و همچنین خیلی کوچک هم نباید باشد چون در آن صورت با رگرسیون خطی معمولی فرقی نخواهد داشت.

$$\widehat{\beta_R} = \sum (y_{true} - y_{pred})^2 + \lambda \cdot \sum_{i=1}^{n} \beta_i^2$$

رگرسیون:LASSO

این رگرسیون نیز مشابه RIDGE است اما با این تفاوت که جمله‌ای که به خطا اضافه میکند درواقع نرم ۱ ضرایب است. همین تفاوت باعث میشود تا فیچرهایی با تاثیر کمتر به نوعی ۰ شوند، اما در RIDGE این اتفاق نمی‌افتد و صرفا ضرایب ما به سمت صفر میل میکنند .LASSO را میتوان به نام feature selection نیز نامید، زیرا آن فیچرهایی که تاثیر زیادی دارند را نگه میدارد و بقیه فیچرهایی که تاثیر کمتر دارند را صفر میکند و کلا از بین میبرد.

$$\widehat{\beta_L} = \sum (y_{true} - y_{pred})^2 + \lambda \cdot \sum_{i=1}^{n} |\beta_i|$$

۲.برای پیدا کردن ضریب regularization یکی از راه کار ها استفاده از gridsearch است. بدین صورت که چندین alpha متفاوت را برای مدل مورد نظر در یک جا تکرار میکنیم و خود تابع آن alpha را که نتیجه بهتری دارد را به ما میدهد. به این صورت میتوانیم بهترین alpha را پیدا کنیم.

۳ .این تغییرات بستگی به تعداد متغیرهای ما دارد. به ازای هر میزان متغیری افزایش تعداد فولدها می تواند مثمر ثمر یا بیهوده باشد. از دید بیهوده بودن که واضح است مثلا اگر ۱۰ نمونه داریم دیگر نیازی به این که ۱۰ فولد بگیریم نداریم. اما از نگاه دیگر اگر تعداد بالا باشد آنگاه تعداد فولد بیشتر باعث تقسیم متوازن تر ، افزایش یادگیری و کاهش overfitting میشود که باید به میزان درستی از آن بهره بگیریم که با توجه به تعداد نمونه و همچنین خطا و آزمایش تعیین میگردد.

۴. این روش مانند روش k-fold است با آن تفاوت که تعداد فولدها برابر تعداد نمونه ها در نظر گرفته میشود و تنها یک بخش به عنوان بخش آموزشی در نظر گرفته میشود. موقعی از این روش استفاده میکنیم که دیتاست کوچکی داشته باشیم و یا هزینه محاسباتی به عنوان معیار برای ما مطرح نباشد.

۵ .روش bootstrapping درواقع یکی از روش های resampling داده است Bootstrap .انجام نمونه گیری با جایگذاری از یک نمونه اصلی به دفعات زیاد است. از یک نمونه ثابت با حجم محدود به دفعات زیاد نمونه گیری انجام میدهیم تا در نهایت بتوان با استفاده از نتایج کلیه‌ی دفعات نمونه گیری، در مجموع به یک توزیع نمونه‌ای درست یافت. به این دلیل جایگذاری انجام میدهیم که آن چیزهایی که نمونه گیری شده‌اند پس از انتخاب دوباره به مجموعه داده ها بازگردند و شانس انتخاب شدن در نمونه گیری های دیگر را نیز داشته باشند Cross validation .نیز مانند bootstrapping یک روش resamplingاست اما یکی از تفاوت های آن این است که cross validation بدون جایگذاری انجام میشود همچنین روش cross validation با تمامی دیتاست سر و کار دارد اما bootstrapping لزوما به همه ویژگی ها کار ندارد و تا هر زمان که بخواهیم میتوانیم با استفاده از آن resampling انجام دهیم. از bootstrapping در کار های مختلفی استفاده میکنیم، مثلا برای ارزیابی مدل یا روش های تجمعی (ensemble) یا حتی تخمین زدن بایاس و واریانس مدل استفاده میشود.

۶. 5x2 fold cross validation به این معناست که 2-fold را برای ۵ مرتبه تکرار کنیم. به این دلیل از 2fold استفاده میشود که مطمئن باشند هر مشاهده در مجموعه داده train یا test فقط برای یک بار آزمودن مدل استفاده میشوند.

گزارش بخش ۳:

در این بخش دیتاست جدید به ما داده شده، این دیتاست شامل داده‌ی nan نیست و همچنین داده های categorical نیز ندارد. داده های ما نیز تعداد کمی دارند، در کل ۲۰۰۰ تا است. رنج داده ها نیز مناسب است و در کل نیازی به تغییری در دیتاست نیست زیرا دیتای ما تمیز است و لازم نیست در دیتاست تغییر دهیم. اما در ابتدا برخی از فیچرها را visualize کرده‌ایم. همانطور که میبینید نمونه های هر کلاس در داده‌ی هدف ما نیز به تعداد ۵۰۰ تا است و کاملا متوازن است. نمودار های مختلف برای برخی ویژگی ها برای موبایل ها رسم شده و در بخش اول مشاهده میکنیم.

سپس ستون price_range را که همان target ما هست جدا کرده و سپس داده ها را scale میکنیم.

حال خواسته های مسئله و سوالات مختلف را پیاده سازی کرده‌ایم. این بخش بر روی پیاده سازی رگرسیون لاجستیک تمرکز دارد. ابتدا با استفاده از پکیج به پیاده سازی رگرسیون لاجستیک بر روی تمامی فیچرهایمان میپردازیم. در هر سوال نیز پس از پیاده سازی، میزان accuracy مدل و همچنین classification report که شامل معیار های خواسته شده است نشان داده شده.

در سوال اول دیتاست را تغییر نداده‌ایم و ۴ کلاس مختلف داریم. در نهایت با پیاده سازی رگرسیون لاجستیک برروی این داده ها، ماتریس confusion آن را نیز که یک ماتریس ۴*۴ هست، مشاهده میکنیم. همانطور که در ابتدا نیز گفته شد به علت آنکه داده های ما بسیار تمیز است، دقت مدل ما نیز بالا است.

در پاسخ به سوال ۲، در ابتدا در بخش visualization دیدیم که کلاس های ما کاملا متوازن هستند و از هرکدام ۵۰۰ تا داریم.

برای سوال ۳، یک دیکشنری تعریف کرده‌ایم که label های ۱ و ۲ و ۳ را به ۱ map میکند و کلاس ۰ را نیز همانطور باقی میگذارد. با اینکار به جای ۴تا کلاس، ۲ کلاس داریم. اتفاق که می‌افتد آن است که تعداد کلاس های ۱ بیشتر میشود و به ۱۵۰۰ تا میرد اما کلاس ۰ همان ۵۰۰ تا باقی میماند. حال دوباره مانند حالت قبل رگرسیون لاجستیک را پیاده سازی میکنیم اما این دفعه برای ۲ کلاس. همانطور که مشاهده میکنید دقت ما از مدل قبلی که شامل ۴ کلاس ببود، بیشتر شده است.

همانطور که گفته شد، با تبدیل کردن به ۲ کلاس، تعداد نمونه ها نامتوازن شده است. برای پاسخ به سوال ۵ برخی تکنیک هایی برای رفع این مشکل بیان میکنیم:

۱. Use Different Algorithms : میتوانیم از الگوریتم های مختلفی استفاده کنیم، تا جواب بهتری برروی داده های نامتوازن بگیریم. برخی الگوریتم ها خیلی بهتر برروی داده های ناموتازن عمل میکنند و فقط نباید از یک الگوریتم برا گونه های مختلف داده ها استفاده کرد. به عنوان مثال به کارگیری مدل های Decision Trees, Penalizing Models, Anomaly Detection نتایج خوبی بر روی داده های نامتوازن میدهند.

۲. Resample the Dataset : این روش به دو صورت میتواند انجام شود. حالت اول oversampling است. در این حالت به کلاسی که تعداد داده های کمتری دارد، به صورت رندم داده اضافه میکنیم تا تعداد داده های آن کلاس نیز برابر با کلاس دیگر شود. درواقع داده هایی که داریم را duplicate میکنیم تا به حدی که میخوایم برسیم. این روش ریسک overfitting را بالا میبرد.

۳. حالت دوم Undersampling است که برعکس حالت قبلی است. کاری که انجام میشود آن است که به صورت رندم داده های موجود در کلاس بزرگ تر را حذف میکنیم تا تعداد آن ها به تعداد داده های کلاس دیگر کاهش یابد. در این روش ممکن است اطلاعات زیادی را از دست دهیم و با ریسک underfitting مواجه هستیم.

در این بخش برای پیاده سازی ما از oversampling برای متوازن کردن دیتا ها استفاده کرده‌ایم و تعداد داده های کلاس ۰ را که ۵۰۰ تا بود که ۱۵۰۰ تا افزایش دادیم. و باز دوباره برروی این داده های جدید رگرسیون لاجستیک را پیاده سازی میکنیم و میبینید که میزان دقت آن کمی از حالت قبل که بدون متوازن کردن زده بودیم، کمتر شده است.

در سوال ۶ خواسته شده که روش forward selection را پیاده سازی کنیم و معیار مورد استفاده ما auc است. این روش یکی از روش های انتخاب ویژگی از نوع wrapper methods است. روش انتخاب مستقیم زیرمجموعه‌ای از فیچرها را در نظر میگیرد و در هر مرحله با توجه به معیار داده شده فیچر هایی را به این مجموعه اضافه میکند که باعث افزایش auc شود.

برای پیاده سازی این روش، چون میخواستیم از معیار auc استفاده کنیم، لازم بود تا ۴ کلاس را به ۲ کلاس تبدیل کنیم و اینگونه عمل کردیم که کلاس ۰ و ۱ را کلاس موبایل های ارزان قیمت (LOW PRICE L) در نظر گرفتیم و کلاس ۲ و ۳ را کلاس موبایل های گران قیمت (HIGH PRICE H) در نظر گرفتیم. سپس با استفاده از پکیج به انتخاب ویژگی ها به روش FORWARD عمل میکنیم. معیار مورد نظر نیز در scoring مشخص شده که همان معیار auc خواسته شده است.

تعداد فیچرهای انتخابی را ۱۱ تا قرار داده‌ایم، همانطور که میبینیم کد مورد نظر ۱۱ فیچر را توسط forward selection انتخاب کرده، در مرحله بعد با استفاده از این ۱۱ فیچر، رگرسیون لاجستیک را پیاده سازی میکنیم، همانطور که مشاهده میکنید، دقت مدل ما بیشتر از تمامی حالت های قبلی است زیرا فیچرهایی که تاثیر بیشتری داشتند انتخاب شده اند و دقت مدل به ۹۹/۵ رسیده.

در سوال بعدی گفته شده که PCA را با تعداد COMPONENT هایی مشابه حالت قبل (در اینجا ۱۱ تا) پیاده سازی کنیم. این کار را توسط پکیج و دستور pca انجام میدهیم و پس از آنکه x های مورد نظر را توسط pca پیدا کردیم، حال با استفاده از آنها مدل خود را fit میکنیم. نتایج نهایی همانطور که مشاهده میکنید، دقت پایین تری دارد زیرا pca به صورت رندم فیچرها را انتخاب میکند و معلوم نیست که کدام فیچرها باقی مانده، به همین علت دقت مدل پایین می آید.

در سوال ۱۰ هم در مورد backward selection سوال شده، که کد آن مشابه همین forward است با این تفاوت که بخش forward را FALSE قرار میدهیم. این روش از مجموعه کل فیچرها که ۲۰ تا است شروع میکند و برعکس فیچرهایی را که کمتر تاثیر دارند را حذف میکند تا به ۱۱ تا فیچر مورد نظر برسد. باز هم مدل لجستیک خود را بر روی این فیچر های بدست آمده پیاده میکنیم و میبینیم که دقت مدل ما از حالت forward selection نیز بیشتر شده و به ۹۹/۷۵ رسیده. بهترین نتیجه مربوط به این مدل است.

در پایان هم برای سوال ۱۱، خواسته شده که fold-k را پیاده سازی کنیم. مانند بخش ۱ عمل میکنیم و مدل لاجستیک خود را با 5-fold و 10-fold برروی تمامی فیچرها اعمال کرده‌ایم و accuracy را نیز مشاهده میکنیم.

پاسخ سوالات بخش ۴:

۱. مشابه این مثال در تمرین حل شد classification .در این حالت باید به صورت باینری تعریف شود . هر یک از داده ها به یک تارگت برخورد میکنند که میتواند درست یا غلط باشد که بحث باینری بودن در همینجاست. نکته ای که موجود است آن است که در حالت ۲ کلاسه ، کلاس به یک کلاس دیگر مقایسه میشد اما در این حالت اگر عضو کلاس مورد نظر نباشد میتواند عضو هرکدام از کلاس های دیگری باشد.

۲.همانطور که در کد پیاده سازی شده مشاهده میشود، داده ی ما پس از oversampling نتیجه بدتری را نسبت به قبل داشته است. یعنی در زمانی که دیتا های ما imbalanced بودند، میزان دقت مدل ۹۹ بوده است اما پس از آن که resamplingانجام داده ایم کمی کاهش یافته است. این به آن دلیل است که oversampling ممکن است که ریسک overfittingرا بالا ببرد و داده هایی را که به صورت رندم اضافه میکنیم ممکن است که تاثیر بدی بر روی مدل ما بگذارند و به همین دلیل دقت آن تا حدودی پایین می آید.

۳.در بخش ۳ پس از پیاده سازی میتوان مشاهده نمود که با انتخاب ۱۱ ویژگی از ۲۰ ویژگی که داشتیم و استفاده از forward selectionنتیجه نهایی بهتر شده و میزان دقت مدل از ۹۷/۵ به ۹۹/۵ رسیده است. این تغییر به وضوح قابل مشاهده است. درواقع feature selection به ما کمک میکند تا تعدادی از فیچر ها را انتخاب کنیم و با کل فیچرها کار نکنیم ، این روش سعی دارد تا با توجه به معیار انتخابی، آن ویژگی هایی را انتخاب کند که در جهت افزایش آن معیار کمک میکند و به این ترتیب ویژگی هایی که باعث پایین آوردن دقت مدل میشوند را به نوعی از بین میبرد و در برخی موارد به خصوص زمانی که تعداد فیچرهای ما زیاد است، خیلی خوب است که از این روش جهت کاهش فیچرهای خود استفاده کنیم و مدل بهتری داشته باشیم.

۴ .به طور کل روش های feature selection به ۴ دسته تقسیم میشوند :۱.Wrapper methods ۲. Filter methods ۳.Embedded methods ۴. Hybrid methods دو روش forward , backward در دسته روش های wrapperقرار میگیرند. دو روش دیگر را در اینجا توضیح میدهیم:

Filter methods:

در این روش به جای استفاده از cross validation از یک سری ازیابی های آماری برای پیدا کردن خصوصیات ذاتی فیچر ها استفاده میکنند و فیچرهایی را بر اساس آن معیار خاص انتخاب میکنند. این روش به مراتب محاسبات کمتری نسبت به روش wrapper دارد و سریعتر انجام میشود. یک روش آن استفاده از امتیاز فیشر است از این روش بیشتر برای کلاس بندی های باینری استفاده میشود و درواقع میزان اهمیت هر ویژگی با استفاده از نسبت فیشر پیدا میشود. نسبت فیشر اینگونه تعریف میشود: "فاصله میان میانگین نمونه کلاس ها به ازای یک ویژگی خاص تقسیم بر واریانس کلاس ها به ازای همان ویژگی"

روش دیگر روش انتخاب ویژگی relief است. در این روش، در هر مرحله و به طور تصادفی، یک نمونه از میان نمونه های موجود در مجموعه داده انتخاب میشود. سپس، میزان مرتبط بودن هر کدام از ویژگی ها بر اساس اختلاف میان نمونه

انتخاب شده و ددو نمونه همسایه نزدیک به روز رسانی میشود. اگریکی از ویژگی های نمونه انتخاب شده با ویژگی مشابه در نمونه همسایه از کلاس مشابه اختلاف داشته باشد، امتیاز این ویژگی کاهش می‌یابد. از سوی دیگر، اگر همان ویژگی در نمونه انتخاب شده با ویژگی مشابه در نمونه همسایه از کلاس مخالف اختلاف داشته باشد، این ویژگی افزایش می‌یابد.

۶ .روش LDA یک روش یادگیری نظارت شده است که برای تفکیک کلاس ها به کار میرود. درواقع هدف اصلی LDA آن است که میزان تفکیک پذیری میان کلاس ها را زیاد کند تا بتوان به راحتی آن ها را دسته بندی کرد LDA .سعی دارد تا با پیدا کردن یک خط جدید (محور) و تصویر کردن داده ها بر روی آن، این میزان تفکیک پذیری را بالا ببرد. معیار های اصلی برای اینکار، میانگین داده ها و همچنین پراکندگی(فراوانی) آن ها میباشد. این روش با کم کردن میزان پراکندگی داده ها و افزایش فاصله میانگین آنها، داده ها را بر روی آن محور مورد نظر، که توسط احتمالات بیزی پیدا میشود، تصویر میکند و پس از اینکار، داده ها به راحتی قابل تفکیک هستند زیرا کلاس های مختلف آنها کاملا جدا میشود و به راحتی میتوان آن ها را کلاس بندی کرد.

۸ .این معیار یکی از معیار های سنجش درستی مدل است، مانند بقیه معیار ها که قبلا استفاده میکردیم و درستی و دقت مدل را با آن بدست میاوردیم، مانند معیار .accuracy این معیار درواقع میزان درستی بین داده predict شده و میزان واقعی داده را به خوبی نشان میدهد و بیشتر برای کلاس بندی های باینری کاربرد دارد.این معیار توسط تست chi squared بدست می آید و اعدادی به عنوان خروجی میدهد. اگر این عدد ۱ باشد ه این معنی است که پیشبینی ما کاملا درست بوده، اگر -۱ باشد به این معنی است که پیشبینی ما کاملا غلط است(کلاس دیگر پیشبینی شده) و اگر ۰ باشد به این معنی است که یک پیشبینی رندم داشتیم و اصلا خوب نیست. این معیار توسط فرمول زیر بدست می آید:

$$|\mathrm{MCC}| = \sqrt{\frac{\chi^2}{n}}$$

همچنین با استفاده از مقادیر موجود در ماتریس confusion نیز میتوان این معیار را محاسبه کرد:

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$