گزارش تمرین دوم داده‌کاوی

علیرضا آزادبخت ۹۹۴۲۲۰۱۹

بخش ۱:

قسمت اولیه سوال و مراحل پیش پردازش داده و توضیحات مورد نیاز آن در گزارش تمرین قبل آورده شده و در این بخش از تمرین فعلی ما باید ۶ سری آزمایش مختلف را بر روی دیتای پیش پردازش شده قبلی انجام دهیم. اما با این تفاوت که هر آزمایش را یکبار برای 5-fold cv و یکبار برای 10-fold cv باید انجام دهیم.

سوال ۱:

- آزمایش یک: بررسی نتایج مدل رگرسیون خطی پیاده سازی شده از پایه را بر روی فیچر با بیشترین کرولیشن با فیچر هدف

  فیچر livingSpaceRange بیشترین کرولیشن با مقدار ۰.۹۵ را با فیچر هدف دارد.
- آزمایش دو: بررسی نتایج مدل رگرسیون خطی پکیج ها بر روی فیچر با بیشترین کرولیشن با فیچر هدف
- آزمایش سه: بررسی نتایج رگرسیون خطی پکیج ها بر روی ۲ فیچر با بیشترین کرولیشن و ۲ فیچر با کمترین کرولیشن با فیچر هدف

  فیچر های livingSpaceRange و noRooms بیشترین کرولیشن با مقادیر ۰.۹۵ و ۰.۶۶ و فیچر های خروجی از one-hot encoding با نام های Sachsen و normal کمترین کرولیشن با مقادیر -۰.۲۷ و – ۰.۱۰ را با فیچر هدف دارا میباشند.
- آزمایش چهار: بررسی نتایج رگرسیون خطی پکیج ها بر روی تمامی فیچر ها
- آزمایش پنجم: بررسی نتایج رگرسیون rigde پکیج ها بر روی تمامی فیچر ها
- آزمایش ششم: بررسی نتایج رگرسیون lasso پکیج ها بر روی تمامی فیچر ها

نتایج آزمایشات برای 5-fold به شکل زیر می‌باشد:

model	dataset	MSE_mean	MSE_variance	Accuracy_mean	Accuracy_variance
my linear regression	most correlation	31.47279	0.02465862	0.706759	1.48E-05
linear regression	most correlation	31.47279	0.02465862	0.706759	1.48E-05
linear regression	top 2 & tail 2 correlation	28.54525	0.014293933	0.76066	6.21E-06
linear regression	all features	4590.197	26680774.69	0.820647	2.05E-06
lasso	all features	28.82828	0.066131003	0.790563	4.84E-06
ridge	all features	24.21987	0.018064632	0.820879	2.48E-06

نتایج آزمایشات برای 10-fold به شکل زیر می‌باشد:

model	dataset	MSE_mean	MSE_variance	Accuracy_mean	Accuracy_variance
my linear regression	most correlation	31.47229	0.055784	0.706759	2.05E-05
linear regression	most correlation	31.47229	0.055784	0.706759	2.05E-05
linear regression	top 2 & tail 2 correlation	28.5444	0.047012	0.760401	1.46E-05
linear regression	all features	42969.22	7.65E+09	0.82078	9.98E-06
lasso	all features	28.82527	0.105322	0.790633	7.68E-06
ridge	all features	24.20772	0.047057	0.820809	9.79E-06

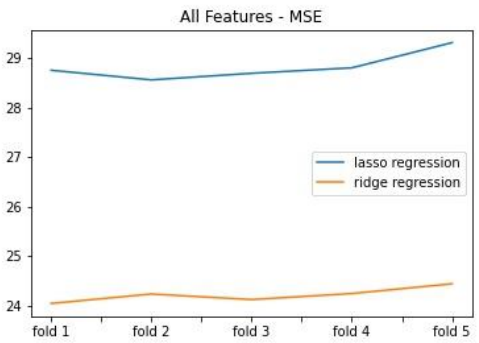
نتیجه گیری:

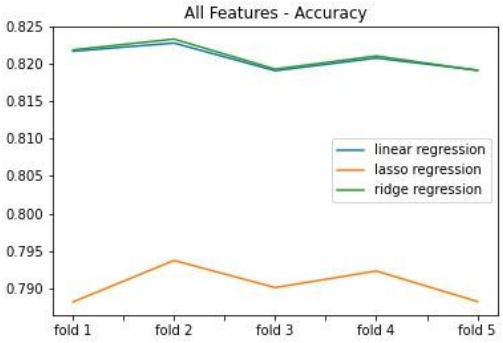
ملاک های بیشتری در نوت بوک محاسبه شده ولی برای گزارش به همین مقادیر اکتفا کردیم، تفاوت چندانی در نتایج کسب شده از دو روش ۵ تایی و ۱۰ تا دیده نمی‌شود و از نظر زمانی روش ۵ تایی خیلی سریع تر عمل کرد که بخاطر تعداد محاسبات کمتر اتفاقی طبیعی بود، نکته جالب قابل مشاهده که ارزش کل این تمرین به پیدا کردن چنین نتیجه ای است در سه سطر آخر نتایج است، در هر دو روش بعضی از fold های داده ها

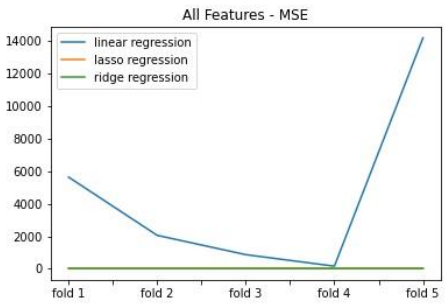
بخاطر ویژگی رندوم بودن آن ها داده هایی کنار هم قرار گرفته اند که بیشتر اگر مدلی بر روی آن ها آموزش داده شود انگار نسبت به داده های اصلی بر روی مجموعه داده ای شامل اوتلایر اموزش دیده است، به زبان دیگر پترنی که فولد های آموزش به خود میگیرند کاملا با پترن فولد تست متفاوت است و وقتی نتیجه خطای این فولد تست محاسبه می‌شود مقدار MSE خیلی بالایی خروجی می‌دهد و باعث ایجاد میانگین خطای خیلی بالا در MSE_mean است و از آنجایی که میانگین Accuracy همچنان قابل قبول است بدین معنی است که استدلالی که کردیم درست است و تنها تعداد کمی از داده های بد پخش شده اند و بر روی همین فولد ها دو رگرسیون ridgeو lasso به دلیل ماهیت ذاتی که دارند و فاکتور های خطایی که در فرمول آن ها است از این مشکل جلوگیری میکند و نتایج همچنان قابل قبول هستند.

سوال۲:

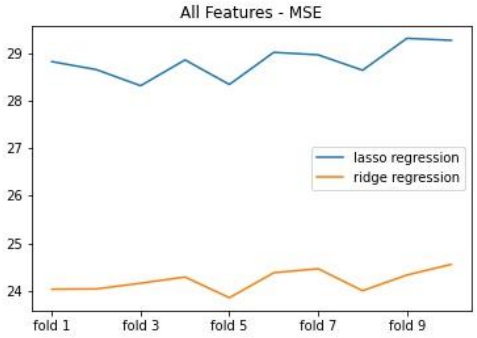
نتایج فولد ها در 5-fold cv:

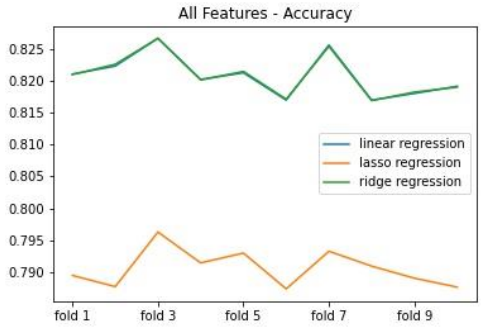


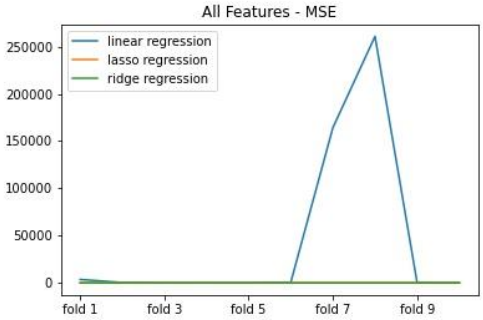




نتایج فولد ها در 10-fold cv:







توضیحاتی که در نتیجه‌گیری قسمت قبل گفتیم را در نمودار MSE فولد ها مشاهده می‌کنیم که در یکی از فولد ها الگوی فول های آموزش و فولد تست متفاوت است و رگرسیون خطی معمولی امکان حل چنین مسئله ای به درستی ندارد و باعث نتایج با خطای بیشتری می‌شود.

بخش ۲:

سوال ۱:

رگرسیون ridge مشابه رگرسیون خطی معمولی عمل میکند با این تفاوت که در فرمول خود علاوه بر مینیمایز کردن RSS بر روی داده ها سعی میکند ستی از پارامتری ها را پیدا کند که این کار را با مقادیر کوچک تر انجام دهد این کار کمک میکند که مدل در فرایند آموزش پارامتر های مربوط به ویژگی های کم اهمیت تر به صفر میل کنند، این کار به کمک اضافه کردن یک ترم جدید به فرمول کلی رگرسیون انجام می‌شود:

$$\lambda \sum_{j=1}^{p} \beta_j^2$$

رگرسیون lasso مشابه رگرسیون خطی معمولی عمل میکند با این تفاوت که در فرمول خود علاوه بر مینیمایز کردن RSS بر روی داده ها سعی میکند ستی از پارامتری ها را پیدا کند که این کار را با مقادیر کوچک تر انجام دهد این کار کمک میکند که مدل در فرایند آموزش پارامتر های مربوط به ویژگی های کم اهمیت تر به صفر میل کنند، این کار به کمک اضافه کردن یک ترم جدید به فرمول کلی رگرسیون انجام می‌شود:

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

هر دوی این مدل ها برای رگولاریزیشن استفاده میشوند و کمک می‌کنند که با مدل های پیچیده اورفیتینگ اتفاق نیافتد، در lasso مقادیر ممکن است صفر شوند و میتوان از آن برای فیچر سلکشن استفاده کرد، در ridge مقادیر به صفر میل می‌کنند و خیلی کوچک می‌شوند اما صفر نمی‌شوند، هر دو مدل هایپر پارامتری برای ست کردنت به اسم لاندا دارند که به کمک cross validation انتخاب می‌شود. در lasso فیچر هایی که کرولیشن دارند یکی از آن ها مقدار زیاد و باقی به صفر میل می‌کنند در ridge تقریبا این مقادیر یکسانند.
از مدل lasso می‌توان برای انتخاب ویژگی استفاده کرد، از این مدل میتوان در دیتا ست هایی که چند فیچر اصلی با تعداد کم و تاثیر گذار دارند استفاده کرد.
از مدل ridge می‌توان برای دیتا ست هایی که اکثر فیچر ها بر روی خروجی موثر هستند استفاده کرد.

منبع: اسلاید های Tibshirani و datacamp.com

سوال ۲:

می‌توان به کمک کراس ولیدیشن تمامی مقادیر ممکن برای پارامتر لاندا را به صورت گیرید سرچ بررسی کرد و مقداری که بهترین نتیجه را داشت انتخاب کنیم.

منبع: اسلاید های Tibshirani

سوال ۳:

رابطه رسمی برای انتخاب تعداد فولد ها وجود ندارد، هرچه تعداد فولد ها بیشتر می‌شود تعداد داده های درون فولد های اموزش به تعداد داده های دیتا ست اصلی نزدیک تر می‌شود و این امر باعث میشود که بایس کاهش پیدا کند، به طور کلی میتوان یک ترید اف بایس وارینس در انتخاب تعداد فولد ها مشاهده کرده به طوری که هر چه تعداد فولد ها افزایش یابد واریانس افزایش و بایس کاهش می‌یابد، و هرچه تعداد فولد ها کمتر باشد بایس افزایش و وارینس کاهش می‌یابد.
اما به طور کلی مقادیر ۵ و ۱۰ در بررسی ها نشان داده شده که به درستی مقدار خطای تست را میتواند مدل کند و از مقدار های زیاد بایاس و واریانس در امان است.

منبع: Applied Predictive Modeling و An Introduction to Statistical Learning

سوال ۴:

Loocv نوعی از k-fold cv است که در آن تعداد فولد ها با تعداد داده های برابر است، یعنی در هر مرتبه یک داده را به عنوان تست در نظر میگیریم و بر روی باقی دیتاست مدل را آموزش می‌دهیم، این روش واریانس زیادی دارد چون خروجی برابر به تعداد داده ها مدل مختلف است که همپوشانی زیادی دارند و خروجی ها وابسته به هم هستند.
اما این روش به دلایل گفته شده بایس خیلی کمی دارد.
هنگامی که دیتا ست کوچکی داریم استفاده از این روش مناسب تر است چون ممکن است داده ها به اندازه کافی زیاد نباشد که با فولد های کمتر دیتا ست آموزشی مناسبی را تشکیل دهند.

منبع: اسلاید Tabshirani و مقاله On estimation of characters obtained in statistical procedure of recognition

سوال ۵:

Bootstrappingیک متد نمونه گیری است که در آن امکان دارد از یک داده چند بار انتخاب شود، به زبان دیگر نمونه گیری با جایگذاری می‌باشد.
تفاوتی که با کراس ولیدیشن دارد این است که در کراس ولیدیشن ممکن نیست از داده ها تکراری نمونه گیری شود و در مجموعه داده های آموزش و تست استفاده شود.
در روش بوت استرپینگ دیتا ست خروجی معمولا اندازه دیتا ست اصلی است اما در کراس ولیدیشن اینگونه نیست.
استفاده اصلی از بوت استرپینگ در آمار و احتمال برای تولید ساب سمپلی با توزیع یکسان از دیتا ست اصلی استفاده می‌شود و در یادگیری ماشین بیشتر در مدل های انسمبل می‌باشد، مثلا برای ساختم مدل جنگ تصادفی درخت های تصمیم کوچکی بر روی مجموعه داده بوت استرپ شده اموزش داده می‌شود و مدل هایی با ایده و توانایی مختلفی ایجاد می‌شود و در نهایت خروجی مدل جنگل تصادفی حاصل رای گیری تک تک مدل ها می‌باشد، اما علاوه بر این ها میتوان از این روش برای محاسبه خطای آموزش و تست هم استفاده کرد به این صورت که مجموعه داده هایی که در دیتا ست بوت استرپ شده نیستند و انتخاب نشده اند را به عنوان مجموعه داده تست در نظر بگیریم و خطا تست را بر روی آن انجام شود.
بوت استرپینگ معمولا بایس بیشتر و واریانس کمتری نسبت به کراس ولیدیشن دارد.

منبع: مقاله A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection

سوال ۶:

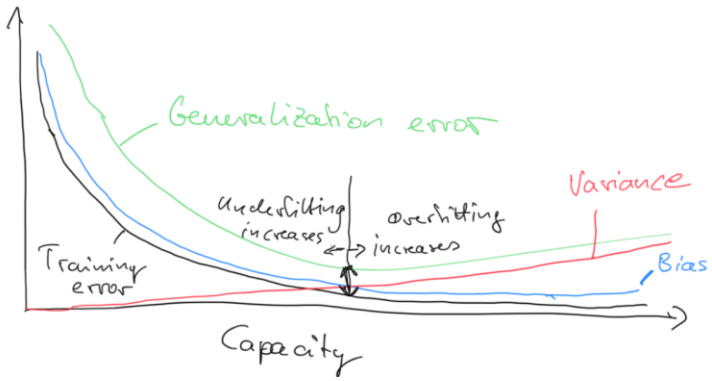
در این روش در دو لایه عمل میکنیم در لایه بیرونی یک 5-fold cv زده می‌شود که مطابق معمول برای بررسی دقت یک مدل استفاده می‌شود ولی در فرایند آموزش در لایه درونی یک 2-fold cv زده می‌شود که از آن برای تیون کردن پارامتر ها و انتخاب مدل استفاده می‌شود، لازم به ذکر است که دقت های لایه بیرونی با توجه به مدل برگزیده و یا پارامتر های تیون شده از لایه درونی محاسبه می‌شود.
این روش همزمان فرایند های انتخاب مدل و تیون کردن پارامتر ها در حین جلوگیری از اورفیت شدن انجام می‌دهد.

تیون کردن پارامتر های مدل در کراس ولیدیشن معمولی باعث ایجاد بایس در مدل می‌شود، پس برای مسائل و مدل هایی که در آن ها قصد تیون کردن پارامتر ها را داریم بهتر است از روش ۵ در ۲ استفاده کنیم.

منابع: مقاله On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation و مقاله Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

سوال ۷:

به طور کلی تا به حال در متد های مختلف از این روش استفاده کرده ایم مثلا در فرایند توقف زود هنگام در آموزش شبکه های عصبی هنگامی که خطا بر روی تست روبه افزایش می‌رود فرایند آموزش متوقف می‌شود که همان نمایان گر متد elbow که در سوال گفته است می‌باشد و به طور کلی میدانیم که هر تابع خطایی را میتوان به حاصل جمع دو خطای بایس و واریانس تبدیل کردو همواره خطای تست برابر جمع این دو خطا میباشد پس میتوان نقطه elbow را محلی که از آنجا واریانس روبه افزایش است در نظر گرفت و میتوان بایس و واریانس کمی داشت، پس میتوان همواره از این متد برای انتخاب مدل استفاده کرد.



منابع: A unified bias-variance decomposition

بخش ۳:

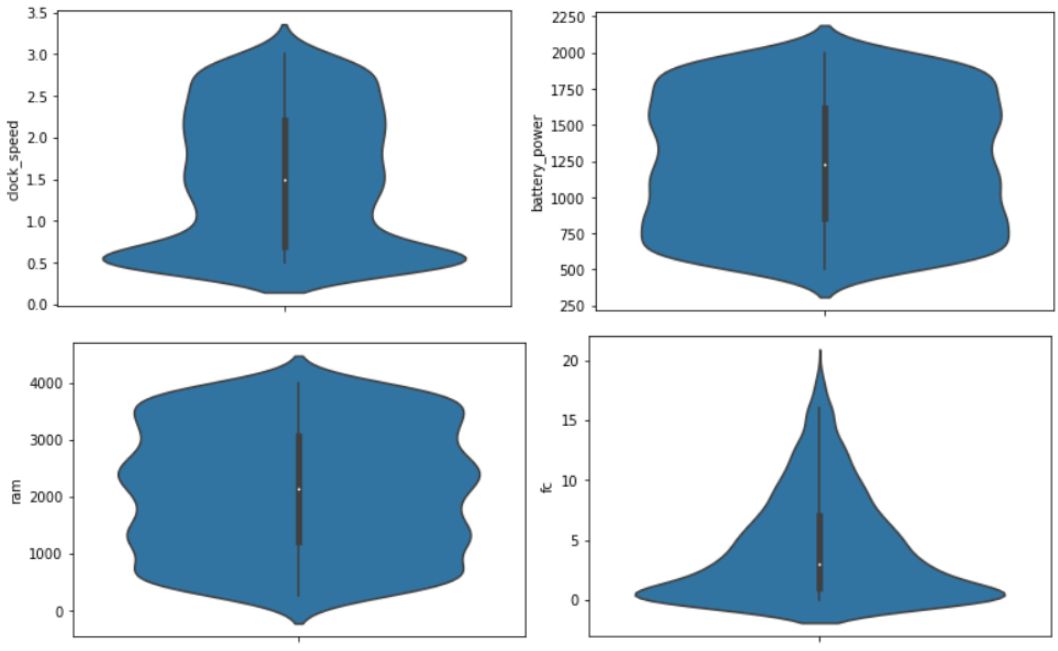
مقدمه:

در این تمرین دیتاستی شامل ۲۰۰۰ داده با بردار ویژگی ۲۱ از مشخصات تلفن های همراه و رنج قیمتی آن ها شامل ۴ دسته در اختیار ما قرار گرفته و در مورد این دیتاست باید به ۱۱ سوال مختلف پاسخ میدادیم، برای این کار ابتدا مراحل تمیز سازی داده ها را انجام دادیم و طی آن مقادیر null و داده های پرت رسیدگی کردیم و سپس مقداری EDA بر روی دیتاست انجام دادیم که تعدادی از سوالات را پاسخ دهیم، و سپس بر روی داده های آماده شده به سوالات پاسخ دادیم.

پیش پردازش داده ها:

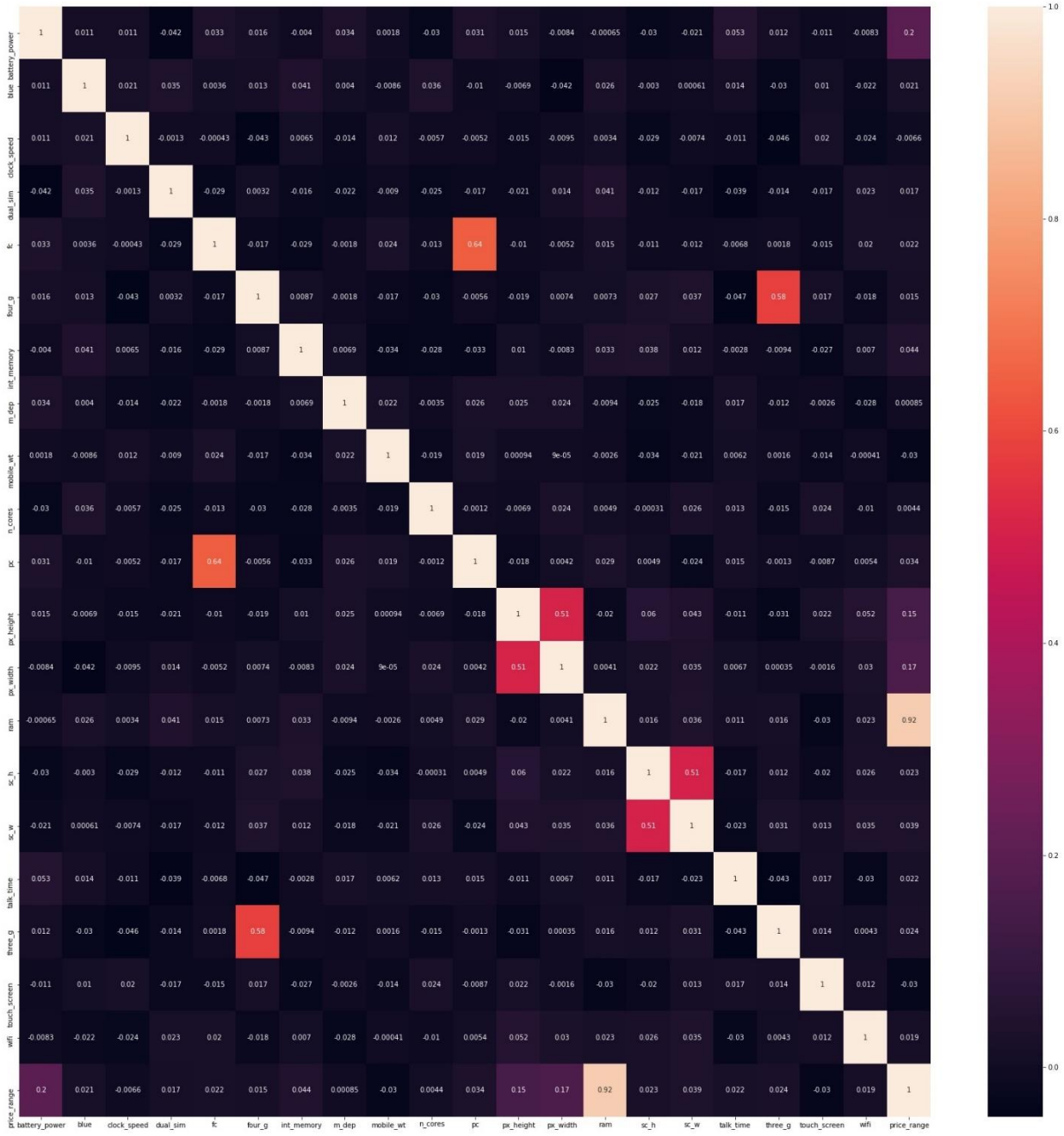
این مجموعه داده تماما شامل داده های عددی بوده و مراحل انکودینگ فیچر های کتگوریکال آن انجام شده و دیگر نیازی به این کار توسط ما نیست.

برای پیدا کردن داده های پرت و اوت‌لایر توزیع داده ها را بر روی تمامی فیچر های آن ها بررسی کردیم و هیچ یک شامل توزیع هایی با دم های طولانی نبودند پس داده ها شامل داده های پرت نیستند و نیازی به حذف بخشی از داده ها نیست، چند نمونه از این توزیع ها را مشاهده میکنیم:



EDA:

در این مرحله به بررسی ماتریس کرولیشن پرداختیم:



از آنجایی که فیچر هدف price_range است مشاهده می‌کنیم که کرولیشن زیادی بین آن و ram وجود دارد، و چند ارتباط کوچک هم بین متغییر های دیگر قایل مشاهده است مانند 3g و 4g .

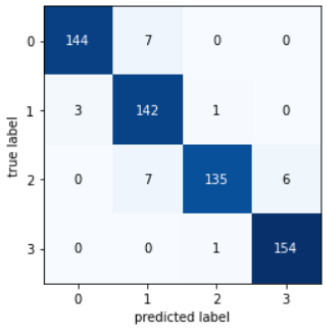
در تمامی مراحل زیر برای پاسخ به سوالات مجموعه داده ها را به دو مجموعه داده آموزش و تست با نسبت ۷۰ به ۳۰ تقسیم کردیم و نتایج گزارش شده حاصل دقت مدل ها بر روی مجموعه داده تست می‌باشد.

سوال ۱:

یک رگرسیون لاجستیک بر روی تمامی فیچر ها آموزش دادیم که چهار کلاس قیمت موبایل ها را دسته بندی کند:

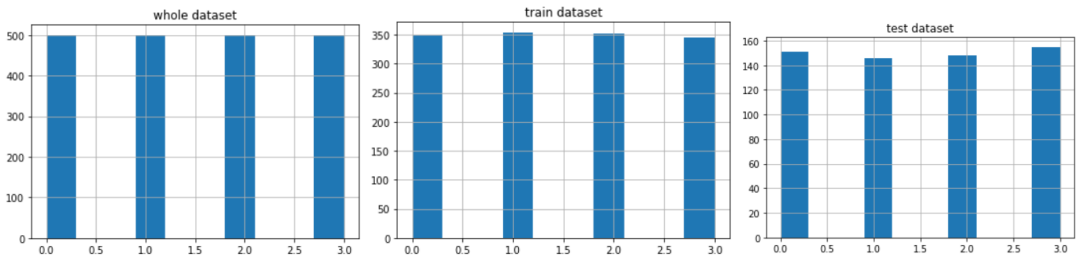
Classification Report:

	precision	recall	f1-score	support
0	0.98	0.95	0.97	151
1	0.91	0.97	0.94	146
2	0.99	0.91	0.95	148
3	0.96	0.99	0.98	155
accuracy			0.96	600
macro avg	0.96	0.96	0.96	600
weighted avg	0.96	0.96	0.96	600



سوال ۲:

برای بررسی اینکه آیا کلاس های از توزیع های یکسانی برخوردار هستند نمودار هیستوگرام متغیر هدف را بر روی کل مجموعه داده، مجموعه داده تست و مجموعه داده آموزش رسم کردیم:



و در تمامی حالات توزیع داده ها بین کلاس ها کاملا متوازن میباشد و خیلی نزدیک به هم هستند.

سوال ۳:

کلاس های ۱و ۲و۳ را با هم به کمک کد زیر به یک کلاس واحد تبدیل کردیم و کلاس ۰ را دست نزدیک حال کلاس ۰ شامل ۵۰۰ داده و کلاس ۱ شامل ۱۵۰۰ داده است.

```
new_y_train = (y_train>0).astype(int)
new_y_test = (y_test>0).astype(int)
```

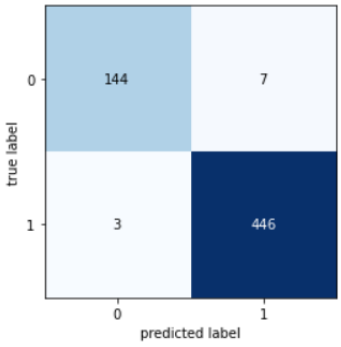
سوال ۴:

بر روی مجموعه داده های نا متوازن جدید یک رگرسیون لجستیک آموزش می‌دهیم و به نتایج زیر می‌رسیم:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.97       151
           1       0.98      0.99      0.99       449

    accuracy                           0.98       600
   macro avg       0.98      0.97      0.98       600
weighted avg       0.98      0.98      0.98       600
```



سوال ۵:

اگر مجموعه داده های ما شامل داده های نا متوازن باشند ممکن از مدل تواجه زیادی به کلاس غالب کند و از یاد گیری الگو های کلاس کوچک تر باز بماند و با اختصاص دادن تمام داده ها به کلاس غالب به نتایج زیاد و خوبی هم برسد، به طور کلی خیلی از مدل های روتین یادگیری ماشین با این نوع داده ها با مشکل روبرو میشوند و نیاز به بررسی خروجی مدل و انالیز آن برای پیدا کردن و رسیدگی به چنین مشکلاتی همواره احساس می‌شود. از معروف ترین راه های مقابله با این مشکل میتوان به موارد زیر اشاره کرد:

- روش اول oversampleing: در این روش از کلاس هایی که داده های کمتری دارند به صورت رندوم چند داده را انتخاب میکنند و دوباره وارد مجموعه داده های آن کلاس می‌کنند، به زبان دیگر به صورت

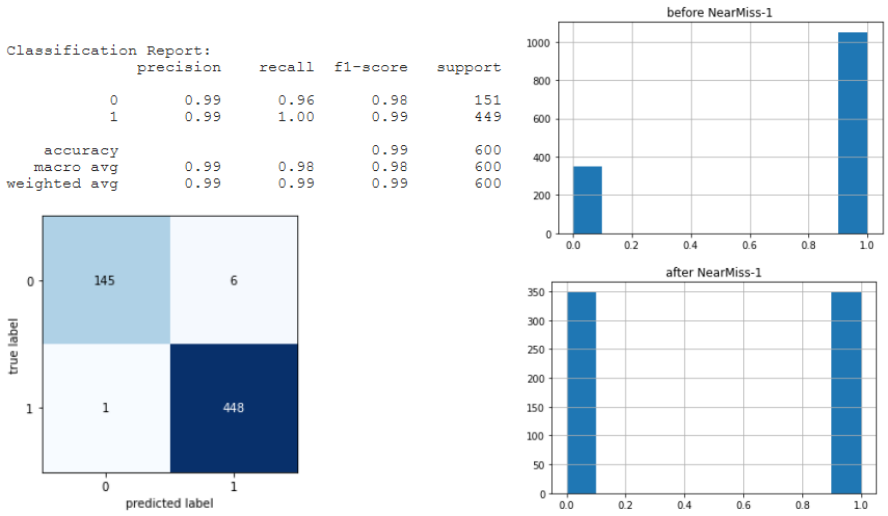
رندوم انقدر از داده های کلاس مغلوب تکرار می‌کنند که تعداد آن به تعداد کلاس غالب برسد و مجموعه داده بالانس شود.

- روش دوم undersampling: در این روش بر عکس روش قبل از کلاس غالب که تعداد بیشتری دارد با توجه به استراتژی های مختلفی که جلو تر مثال می‌زنیم به تعداد لازم نمونه گیری میکنیم و به جای کلاس مغلوب تنها نمونه گرفته شده را قرار می‌دهیم تا توزیع کلاس ها بالانس شود.
- روش سوم synthetic data: در این روش به کمک مدل های مولد سعی میکنیم باز داده های کلاس های کوچک دیتا تولید کنیم، می‌توان از مدل های gan برای تولید دیتا استفاده کرد یا میتوان به کمک data augmentation برای کلاس کوچک تر دیتا های جدید ساخت و یا می‌توان به کمک یک خط دیتا های کلاس کوچک را به هم وصل کرد و بر روی این خط متصل شده دیتا تولید کرد.

برای حل این مسئله در صورت سوال ما به کمک الگوریتم near-miss و سه نوع آن به حل این مسئله پرداختیم، در این الگوریتم در نوع یک: داده هایی از کلاس غالب انتخاب می‌شود که دارای کمترین میانگین فاصله از ۳ یا k (kیک هایپر پارامتر این روش است که معمولا ۳ قرار داده می‌شود) داده کلاس مغلوب هستند. در نوع دو: داده هایی از کلاس غالب انتخاب می‌شوند که دارای کمترین فاصله از ۳ یاk داده دور از کلاس مغلوب هستند. و در نوع سه: به تعداد مورد نیاز داده های کلاس مغلوب که نزدیک داده کلاس غالب هستند انتخاب می‌شود.

نتایج این سه الگوریتم به شکل زیر می‌باشد:

نوع یک:

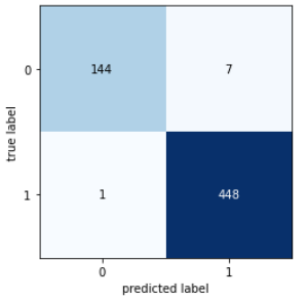


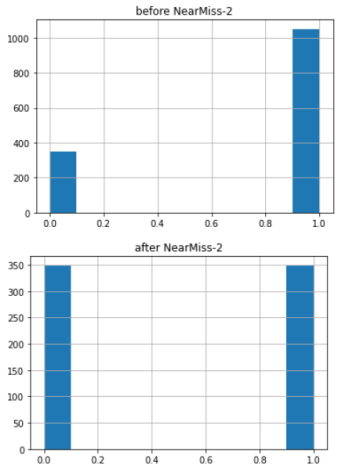
نوع دو:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       151
           1       0.98      1.00      0.99       449

    accuracy                           0.99       600
   macro avg       0.99      0.98      0.98       600
weighted avg       0.99      0.99      0.99       600
```



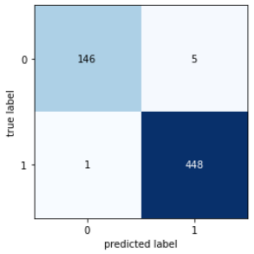


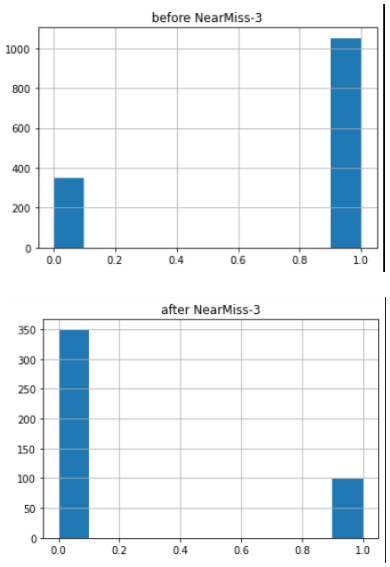
نوع سه:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       151
           1       0.99      1.00      0.99       449

    accuracy                           0.99       600
   macro avg       0.99      0.98      0.99       600
weighted avg       0.99      0.99      0.99       600
```





همان طور که مشاهده می‌شود و انتظار می‌رفت تنها روش یک و دو داده های کلاس ها بعد از این الگوریتم متوازن می‌شوند اما چون در روش سه تنها داده های مهم و مرز تصمیم انتخاب شده اند همچنان اطلاعات مورد نیاز مدل در دسترس هست و هر سه مدل به خوبی عملیات طبقه بندی را انجام داده اند. اما از نظر عددی بخواهیم بگوییم روش سوم دقت بیشتری کسب کرده است.

سوال ۶:

روش انتخاب ویژگی پیشرو را به صورت زیر پیاده سازی کردیم:

```
def forward_selection(X, y, model, score):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
    outer_max ={'adjusted_R2': -2, 'feature_set': []}
    fixed = []
    for j in range(len(X.columns)):
        inner_max = {'AUC': -2, 'feature': 0, 'adjusted_R2': -2}
        for i in X.columns:
            if i in fixed:
                continue
            current_features = fixed+[i]
            model.fit(X_train[current_features], y_train)
            pred = model.predict_proba(X_train[current_features])
            s = score(label_binarize(y_train, classes=[0, 1, 2, 3]), pred, multi_class="ovo", average="weighted")
            if s > inner_max['AUC']:
                inner_max['AUC'] = s
                inner_max['feature'] = i
                inner_max['adjusted_R2'] = 1 - (1-model.score(X_test[current_features], y_test))*(len(y_test)-1)/(len(y_test)-
X_test[current_features].shape[1]-1)
        if inner_max['feature'] != 0:
            fixed = fixed+[inner_max['feature']]
            if outer_max['adjusted_R2'] < inner_max['adjusted_R2']:
                outer_max['adjusted_R2'] = inner_max['adjusted_R2']
                outer_max['feature_set'] = fixed
        else:
            return outer_max['feature_set']
    return outer_max['feature_set']
```

در این روش ابتدا کل داده دریافتی را به با نسبت ۲۰ به ۸۰ تقسیم می‌کنیم با ۸۰ درصد داده ها فرایند انتخاب ویژگی را پیش میبریم و برای هر ست از پارامتر ها که بهترین ملاکauc را کسب کردند مقدار adjusted r2 که ملاکی است که مقدار خطا را با ترمی برای جریمه مدل های پیچیده محاسبه میکند بر روی ۲۰ درصد باقی مانده بدست می‌آوریم و در پایان الگوریتم ستی از فیچر ها که بیشترین ملاک adjusted r2 را دارند باز میگردانیم.

سوال ۷:

به کمک روش انتخاب ویژگی پیشرو که پیاده سازی کردیم ۷ فیچر برگزیده را پیدا کردیم که خروجی این الگوریتم بودند:

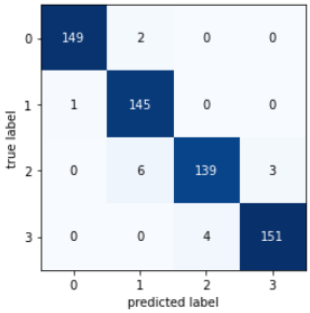
['ram', 'battery_power', 'px_height', 'px_width', 'mobile_wt', 'sc_h', 'pc']

سپس بر روی این ۷ فیچر مدل رگرسیون لاجستیک را اجرا کردیم و نتایج را بدست آوردیم:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       151
           1       0.95      0.99      0.97       146
           2       0.97      0.94      0.96       148
           3       0.98      0.97      0.98       155

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600
```



سوال ۸:

در روش قبلی متوجه شدیم که ۷ فیچر هستند که از اهمیت بیشتری بهره میبرند و میتوان به کمک آن ها عملیات طبقه بندی را به خوبی انجام داد در این سوال یک pca با ۷ کامپوننت اجرا میکنیم و فضای داده ها را به کمک این روش به فضای ۷ بعدی منتقل میکنیم و به دیتا ستی مشابه زیر میرسیم:

```
forward_selection_features len 7
```

	0	1	2	3	4	5	6
0	1533.788727	-147.867924	-334.273376	249.882553	18.898283	29.424814	1.179223
1	766.838150	8.467145	-65.138234	621.931840	9.926657	-11.737706	1.801129
2	790.969318	-323.190823	294.970226	325.231240	-61.372272	1.694497	-3.034388
3	-1553.815893	-276.087679	755.952930	-90.430166	-46.320396	10.056646	5.407686
4	-1371.177508	-559.457736	-317.777162	6.167178	-11.135402	-3.695607	7.385858

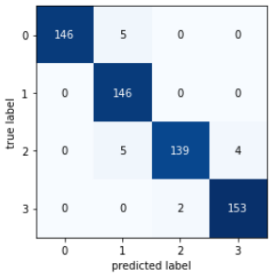
سوال ۹:

بر روی خروجی سوال قبل یک رگرسیون لاجستیک اجرا میکنیم و به نتایج زیر میرسیم:

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.97      0.98       151
           1       0.94      1.00      0.97       146
           2       0.99      0.94      0.96       148
           3       0.97      0.99      0.98       155

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600
```



نتایج حاصل از سوال ۷ و سوال ۹ کاملا مشابه هستند از نظر تعداد خطا ها در هر دو روش ۱۶ داده اشتباه پیشبینی شده اند که همه ان ها نزدیک قطر اصلی هستند و این نشانه نسبتا خوبیست چون فیچر هدف ما دارای خاصیت ترتیبی است پس وجود چنین نوع خطا هایی نشان میدهد که مدل الگو های درستی را یاد گرفته اما بعضی از داده ها اشتباه دسته بندی شده اند و یا این موبایل ها زیر قیمت ب فروش میرسند یا بیشت از اندازه و ارزش آن ها گران شده اند.

سوال ۱۰:

روش انتخاب ویژگی روبه عقب کاملا مشابه روش پیشرو پیاده سازی شده با این تفاوت که ابتدا لیست فیچر های مدل شامل همه فیچرهاست و یک به یک در هر مرحله یکی از آن ها حذف می‌شود:

```
def forward_selection(X, y, model, score):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
    outer_max ={'adjusted_R2': -2, 'feature_set': []}
    fixed = []
    for j in range(len(X.columns)):
        inner_max = {'AUC': -2, 'feature': 0, 'adjusted_R2': -2}
        for i in X.columns:
            if i in fixed:
                continue
            current_features = fixed+[i]
            model.fit(X_train[current_features], y_train)
            pred = model.predict_proba(X_train[current_features])
            s = score(label_binarize(y_train, classes=[0, 1, 2, 3]), pred, multi_class="ovo", average="weighted")
            if s > inner_max['AUC']:
                inner_max['AUC'] = s
                inner_max['feature'] = i
                inner_max['adjusted_R2'] = 1 - (1-model.score(X_test[current_features], y_test))*(len(y_test)-1)/(len(y_test)-
X_test[current_features].shape[1]-1)
        if inner_max['feature'] != 0:
            fixed = fixed+[inner_max['feature']]
            if outer_max['adjusted_R2'] < inner_max['adjusted_R2']:
                outer_max['adjusted_R2'] = inner_max['adjusted_R2']
                outer_max['feature_set'] = fixed
        else:
            return outer_max['feature_set']
    return outer_max['feature_set']
```

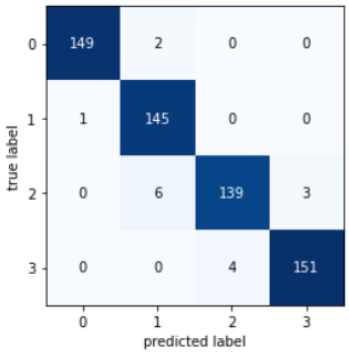
خروجی این روش هم دقیقا همان ۷ فیچر روش قبلی بود و نتایج رگرسیون خطی لاجستیک بر روی آن را میتوانیم در زیر مشاهده کنیم:

```
['battery_power', 'mobile_wt', 'pc', 'px_height', 'px_width', 'ram', 'sc_h']
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       151
           1       0.95      0.99      0.97       146
           2       0.97      0.94      0.96       148
           3       0.98      0.97      0.98       155

    accuracy                           0.97       600
   macro avg       0.97      0.97      0.97       600
weighted avg       0.97      0.97      0.97       600
```



سوال ۱۱:

به کمک 5-fold cv و 10-fold cv مدل لاجستیک رگرسیون را آموزش دادیم و به نتایج زیر رسیدیم:

cv	F1 weighted_ mean	F1 weighted_va riance	Accuracy_ mean	Accuracy_va riance	Precision weighted_ mean	Precision weighted_va riance	Recall weighted_ mean	Recall weighted_va riance
5	0.962488	2.30E-05	0.9625	2.25E-05	0.962674	2.38E-05	0.9625	2.25E-05
10	0.961939	9.59E-05	0.962	9.60E-05	0.962374	9.79E-05	0.962	9.60E-05

cv	F1 macro_mean	F1 macro_variance	Precision macro_mean	Precision macro_variance	Recall macro_mean	Recall macro_variance
5	0.962488	2.30E-05	0.962674	2.38E-05	0.9625	2.25E-05
10	0.961939	9.59E-05	0.962374	9.79E-05	0.962	9.60E-05

بخش ۴:

سوال ۱:

برای این کار از روش one-vs-rest استفاده می‌شود برای انجام لاجستیک رگرسیون برای چند دیتا ست های چند کلاسه به تعداد کلاس ها مدل رگرسیون خطی می‌سازیم هر یک از این مدل ها وظیفه تشخیص تنها یکی از کلاس ها از باقی کلاس ها را انجام می‌دهد و به صورت احتمالاتی احتمال تعلق داده ها به کلاس هدف خود را باز میگرداند، هنگامی که می‌خواهیم کلاس داده ای را تشخیص دهیم این داده را به تمامی رگرسیون های می‌دهیم و کلاسی که بیشترین احتمال تعلق را دارد به عنوان خروجی باز میگردانیم.

منبع: اسلاید های Andrew ng

سوال ۲:

تفاوت محسوسی دیده نمی‌شود در سوال ۴ تنها ۱۰ داده اشتباه طبقه بندی شده اند و در سوال ۵ در بهترین الگوریتم تنها ۶ داده اشتباه طبقه بندی شده اند و با اختلاف کمی نتایج سوال ۵ بهتر هستند.

سوال ۳:

تفاوت یک درصدی در دقت مدل بعد از انتخاب ویژگی مشاهده می‌شود به طور کلی در سوال ۱ تنها ۲۵ داده اشتباه طبقه بندی شده اند و دقت ۹۶ درصد بود و در سوال ۶ تنها ۱۶ داده اتشباه بودند و دقت ۹۷ درصد بود، به طور کلی ما به دنبال مدلی هستیم که پیچیدگی محاسباتی کمتری داشته باشد و در عین حال نتایج خوبی هم کسب کند به کمک انتخاب ویژگی میتوانیم از اورفیت شدن مدل ها جلوگیری کنیم و اثر نویز را کم کنیم و مدل های بهینه تری پیدا کنیم.

منبع: اسلاید tabshirani

سوال ۴:

Best subset selection: در این روش مشابه روش پیشرو عمل میکنیم اما در هر مرحله از حلقه داخلی بجای این که تنها یک فیچر را اضافه کنیم تمام جایگشت های مختلف قابل قبول از فیچر ها با تعداد آن چرخش از حلقه را بررسی میکنیم و باقی محاسبات و روش انتخاب کاملا مشابه روش های قبلی است.

Automatic Recommendation Method: این روش به عنوان ورودی دیتا ستی را دریافت می‌کند و از دیتا بیس درونی خود دیتاست های مشابه این دیتا ست را بر اساس ملاک فاصله پیدا میکند و چند دیتا ست مشابه را انتخاب می‌کند و بر اساس ملاک خطا و سرعت محاسبات و پیچیدگی محاسبات روش هایی که بر روی دیتا ست های کاندید خوب عمل کرده اند را انتخاب می‌کند ودقت مدل ها با ساب ست های کاندید را بر اساس ملاک ها خواسته شده انتخاب میکند و نهایتا بهترین ساب ست را بر میگرداند.

منابع: اسلاید های tabshirani و مقاله A Feature Subset Selection Algorithm Automatic Recommendation Method

سوال ۵:

هیچ یک از دو روش تضمین نمی‌کنند که بهترین ساب ست را پیدا کنند اما از نظر محاسباتی و پیچیدگی زمانی هر دو روش قابل دست یابی هستند.
روش پیشرو در دیتا ست هایی که تعداد داده ها از تعداد فیچر ها کمتر هستند میتواند استفاده شود اما روش پسرو حتما باید تعداد داده ها بیشتر از تعداد فیچر ها باشد.
برای رفع این مشکلات می‌توان این روش ها را با ترتیب های مختلف از بررسی فیچر ها در هر چرخش حلقه ها چند بار انجام داد تا شانس پیدا کردن بهترین ساب ست افزایش یابد، به زبان دیگر یک فاکتوری احتمالاتی به روش اضافه کنیم که هر بار اجرا مسیر متفاوتی را طی کند.
می‌توان در دیتا ست هایی که تعداد داده ها کمتر از فیچر ها است دیتا ست را تیکه تیکه کنیم نسبت به فیچر ها و روش پسرو را بر روی این تیکه های کوچک اجرا کنیم.

منبع: اسلاید های tabshirani

سوال ۶:

آنالیز افتراقی خطی Linear Discriminant Analysis

دراین روش به جای آنکه مستقیما $\Pr(Y = k|X = x)$ را مدل کنیم، ابتدا احتمال توزیع متغیر ورودی به شرط کلاس را به‌دست می آوریم و از روی آن و با استفاده از دانش پیشین، $\Pr(Y = k|X = x)$ را مدل می‌کنیم. چرا گاهی از این روش به جای لاجیستیک استفاده می کنیم:

- وقتی کلاس ها به خوبی از هم قابل تفکیک هستند، تخمین های روش لاجیستیک نااستوار است.
- وقتی تعداد داده ها کم است و توزیع متغیرها در هر کلاس تقریبا نرمال است، روش linear discriminant بهتر و استوارتر از روش لاجیستیک عمل می کند.
- روش linear discriminant برای حالت چندکلاسه معروف تر است.

استفاده از قضیه بیز برای کلاس بندی

فرض کنید که K تا کلاس داریم. فرض کنید π_k توزیع پیشین کلاس kام باشد. یعنی احتمال اینکه یک داده‌ی تصادفی عضو کلاس kام باشد. فرض کنید $f_k(x) = \Pr(X = x|Y = k)$ توزیع متغیر ورودی برای داده های کلاس kام باشد. آنگاه طبق قضیه بیز خواهیم داشت:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \qquad (3.1)$$

تخمین π_k از روی نمونه های آموزشی بسیار ساده است. کافی است محاسبه کنیم که چه کسری از داده ای آموزشی برچسب کلاس kام را دارند. اما تخمین $f_k(x)$ به این سادگی ها نیست؛ مگر اینکه فرض کنیم دارای توزیع ساده ای باشد. به $\Pr(Y = k|X = x)$ احتمال پسین می گوییم. به ازای هر داده ی جدید این احتمال پسین را برای هر کدام از K کلاس بدست می آوریم و نهایتا داده را به کلاسی نسبت می دهیم که احتمال پسین بیشتری داشته باشد. این قانون تصمیم گیری دارای کمترین نرخ خطا است.

حالت تک متغیره

ابتدا فرض کنیم فقط یک متغیر ورودی داریم و فرض کنیم $f_k(x)$ برای تمام کلاس ها دارای توزیع نرمال به فرم زیر است:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

اگر فرض کنیم که واریانس تمام کلاس ها با هم برابر است یعنی $\sigma_1 = \sigma_2 = \cdots = \sigma_k = \sigma$، با جایگذاری توزیع نرمال فوق در رابطه (3.1) خواهیم داشت:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}$$

حال با لگاریتم گرفتن و ساده کردن برخی جملات به فرمول زیر می رسیم که به آن discriminant score گویند:

$$\delta_k(x) = \frac{x\mu_k}{\sigma^2} - \frac{{\mu_k}^2}{2\sigma^2} + \log(\pi_k) \qquad (4.1)$$

هر داده X=xای را به کلاس kامی نسبت می دهیم که دارای مقدار $\delta_k(x)$ بزرگتری باشد.

همیشه مقادیر دقیق پارامترهای میانگین μ_k و واریانس σ را نداریم و باید این مقادیر را از روی نمونه های آموزشی به صورت زیر تخمین بزنیم:

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:\, y_i = k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:\, y_i = k} (x_i - \hat{\mu}_k)^2$$

که n تعداد کل نمونه های آموزشی و n_k تعداد نمونه های آموزشی با برچسب کلاس kام است. با جایگذاری این مقادیر تخمینی در فرمول (4.1) ، داده x را به کلاسی نسبت می دهیم که مقدار $\hat{\delta}_k(x)$ بزرگتری داشته باشد.

$$\hat{\delta}_k(x) = \frac{x\hat{\mu}_k}{\hat{\sigma}^2} - \frac{{\hat{\mu}_k}^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

اگرفرض کنیم K=2 و $\pi_1 = \pi_2$ باشد، داده را به کلاس 1 نسبت می دهیم اگر

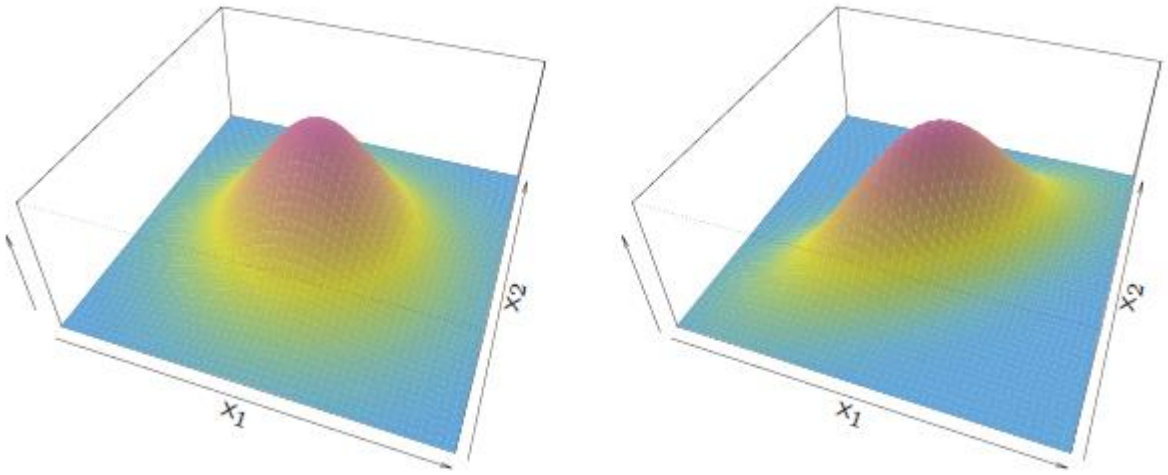
$$2x(\mu_1 - \mu_2) > {\mu_1}^2 - {\mu_2}^2$$

باشد. بنابراین مرز تصمیم گیری نقطه زیر است:

$$x = \frac{{\mu_1}^2 - {\mu_2}^2}{2x(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}$$

حالت چند متغیره

اگر بخواهیم کلاس‌بند را برای حالت چند متغیره تعمیم دهیم، باید از توزیع نرمال چند متغیره با میانگین وابسته به کلاس و ماتریس کوواریانس مشترک استفاده کنیم.



در شکل بالا دو توزیع نرمال دو متغیره را مشاهده می‌کنید که در شکل سمت چپ دو متغیر از هم مستقل و در شکل سمت راست دو متغیر همبستگی دارند. ارتفاع نمودار در هر نقطه نشان دهنده احتمال آن نقطه است.
توزیع نرمال چند متغیره به شکل زیر است:

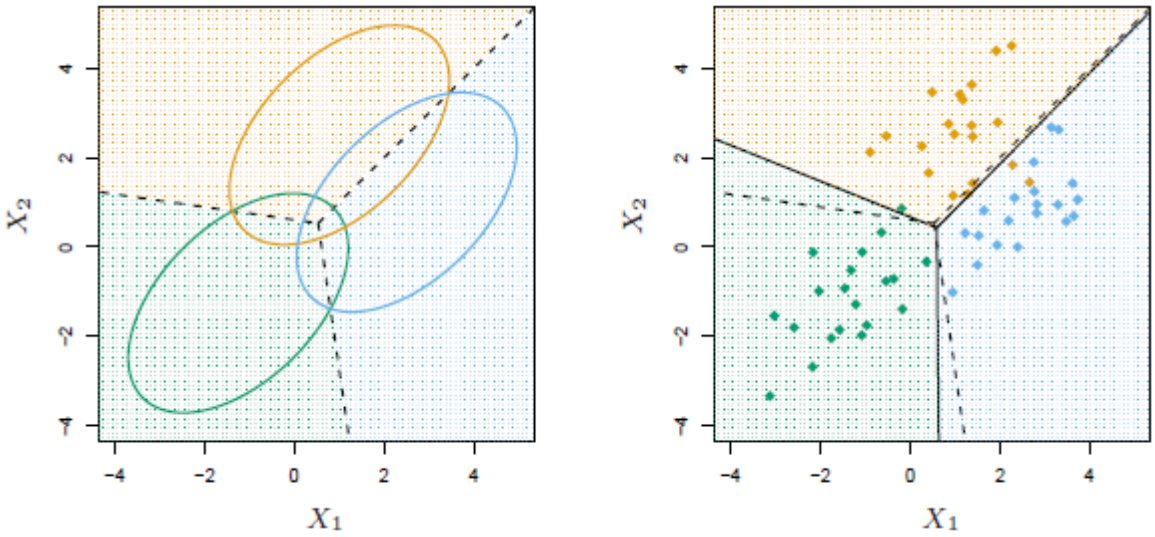
$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right) \qquad (5.1)$$

با جایگذاری این احتمال در فرمول (3.1) و لگرایتم از طرفین و حذف عبارات خواهیم داشت:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k{}^T\Sigma^{-1}\mu_k + \log(\pi_k)$$

مرز تصمیم گیری برای دو کلاس k و l جایی است که $\delta_k(x) = \delta_l(x)$. یعنی اگر هر دو کلاس دارای تعداد برابری عضو باشند، مرز تصمیم مجموعه همه نقاطی است که در رابطه زیر صدق می کنند:

$$x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k{}^T\Sigma^{-1}\mu_k = x^T\Sigma^{-1}\mu_l - \frac{1}{2}\mu_l{}^T\Sigma^{-1}\mu_l$$



در شکل سمت چپ سه توزیع نرمال و مرزهای تصمیم واقعی و در شکل سمت چپ نمونه هایی که از این توزیع ها تولید شده اند و خطوط پررنگی که نشان دهنده مرزهای تصمیم است که با پارامترهای تخمین زده شده به دست آمده اند. همان طور که مشاهده می کنید مرزهای تصمیمی که از پارامترهای تخمین زده شده با نمونه های آموزشی به دست آمده اند، اندکی از مرزهای تصمیم واقعی بیزین متفاوت هستند.

خروجی pca تضمین میکند که فیچر های جدید با هم کرولیشن ندارند و بعضی از مدل های یادگیری ماشین از این ویژگی استفاده می‌کنند و برای آن ها مفید است.
lda در جاهایی که با برچسب داده ها را داریم استفاده می‌شود و یک روش با ناظر است اما pca اینگونه نیست و میتواند بدون ناظر عمل کند. Lda بر روی داده های کم خوب عمل نمی‌کند.
از ldaمیتوان در طبقه بندی های چند کلاسه استفاده کرد و نتایج بهتری نسبت به لاجستیک رگرسیون گرفت.
استفاده از pcaبه عنوان طبقه بند ایده خیلی مناسبی نیست و بیشتر برای کاهش بعد میتوان از آن استفاده کرد.

منبع: اسلاید های tabshirani و جزوه درس

سوال ۷:

یکی از متد های معروف در این بخش 5x2 cv است(نه آن متدی که در سوالات قبل اشاره شد) در این روش برای دو مدلی که قصد مقایسه آن ها را داریم ۵ بار 2-fold cv را انجام می‌دهیم و نتایج آن ها نگه می‌داریم چون این نتایج به هم وابسته اند می‌توان بر روی آن ها یک t-test زد و اگر pvalue کمتر از ۰.۰۵ باید میگوییم که فرقی به دو مدل وجود ندارد در غیر این صورت یکی از مدل ها موفق تر عمل کرده است.

منبع: Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms

سوال ۸:

Matthews correlation coefficient یک ملاک برای پیدا کردن دقت مدل ها در تسک طبقه بندی دو کلاسه یا چند کلاسه می‌باشد و مانند f1-score میتواند در شرایطی که اندازه کلاس ها برابر نیستند نیز خوب عمل کند، در این ملاک مدل رندوم مقدار ۰ بهترین مدل مقدار ۱ و مدل کاملا برعکس مقادیر درست مقدار -۱ را می‌گیرد، و از طریق فرمول زیر محاسبه می‌شود:

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

F1 -score در قرمول خود به مقادیر tn توجه نمی‌کند اما این ملاک این کار را انجام می‌دهد، در جاهایی که نیازی به تفسیر ملاک نداریم و می‌خواهیم مدل بر روی هر دو کلاس یا همه کلاس ها خوب عمل کند و به طور کلی در دیتا ست های غیر بالانس ملاک خوبی برای استفاده است.

منبع: The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation