بسمه تعالی

دانشگاه شهید بهشتی
دانشکده علوم ریاضی
گروه علوم کامپیوتر

تمرین سری ۲ واحد درسی داده کاوی

سارا چرمچی
۴۰۰۴۲۲۰۶۶

استاد راهنما : جناب آقای دکتر فراهانی

دستیار آموزشی : آقای علی شریفی

اردیبهشت ۱۴۰۱

مقدمه

کلیه تحلیل های این گزارش با استفاده از زبان برنامه نویسی پایتون اجرا شده است و کدها به پیوست موجود است.

تمرین اول

دیتاست پیشرو شامل داده های فنی تلفن همراه با برچسب رده قیمتی است. داده ها شامل ۲۱ ستون متفاوت شامل داده های عددی و غیر عددی (باینری شده) و بالغ بر ۲۰۰۰ نمونه می باشد

ابتدا داده ها را load کرده و نگاهی کلی به صورت بندی دیتاست می کنیم شامل نوع متغیرها ، شکل دیتاست ، بررسی اطلاعات آماری داده های عددی و ...

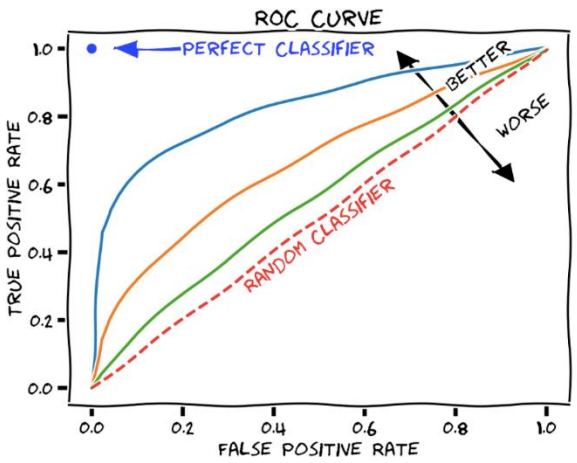
سپس در صورت نیاز داده ها را پاک سازی میکنیم شامل بررسی داده های null و تصمیم گیری در مورد آن ها و بررسی داده های تکراری و ...

داده های آموزش و آزمون را در هر مرحله جدا سازی کرده و روی داده های اموزش یک نرمال سازی ساده اجرا میکنیم.

سوال۱-روش پیشرو forward selection با متریکAUC

ابتدا داده های متغیر پاسخ را بصورت باینری در می اوریم و کلاس ارزان و گران قیمتی را بصورت ۰ و ۱ ایجاد می کنیم.

در این روش ابتدا با یک مدل خالی از متغیر شروع می‌کنیم و در هر مرحله یک متغیر به مدل اضافه می‌کنیم. متغیری برای اضافه شدن به مدل انتخاب می‌شود که اطلاعات اضافه بیشتری در اختیار مدل قرار دهد. این روند ادامه پیدا می‌کند تا همه متغیرها وارد مدل شوند.
اما معیار ما در انتخاب مدل بهتر مساحت زیر نمودار یا AUC است AUC (Area under Curve) -ROC(مشخصه عملیاتی گیرنده) معیار عملکردی است که بر اساس مقادیر آستانه متفاوت برای مشکلات طبقه بندی تنظیم شده است. همانطور که از نامش پیداست ، ROCیک منحنی احتمال است و AUC قابلیت تفکیک را اندازه گیری می کند. به عبارت ساده تر ، معیار AUC-ROC در مورد توانایی مدل در تشخیص کلاس ها به ما می گوید AUC .بالاتر ، مدل بهتر است. در شکل زیر نمایی از یک نمودار ROC نشان داده شده است:



مشخص است که roc_auc_score کمتر از ۰.۵ نشان دهنده کلاس بند رندوم است که در مدل موثر نخواهد بود.

در برنامه نویسی این روش ابتدا تابعی تعریف می کنیم شامل متغیرهای داده های اموزش x و داده های اموزش برچسب y و آستانه مورد نظر.

یک لیست اولیه از فیچرهای داده های آموزش می سازیم.، سپس با یک حلقه while فیچرهای باقی مانده را هر بار درون یک لیست میریزیم. با فیچرهای باقی مانده یک حلقه for می سازیم و هر بار یک فیچر را اضافه کرده به مدل و کارایی آن را با لیستی از متریک roc_auc سنجیم

با لیست به دست امده از حلقه for از متریک ها ، مقدار بیشنه را یافته و با مقدار آستانه می سنجسم و اگر با معیار همخوانی داشت به لیست roc_auc اضافه میکنیم. این حلقه تا تمام شده سنجش همه فیچرها ادامه می یابد و در نهایت لیستی از فیچرهای انتخابی را بر می گرداند.

به این ترتیب هر بار بهترین فیچر از نظر متریک roc انتخاب می شود و در انتخاب بعدی این فیچر ثابت است و فیچر بعدی اضافه می شود تا در نهایت یک لیست از بهترین فیچرها را بدست آوریم.

['ram', 'battery_power', 'px_height', 'px_width', 'n_cores', 'dual_sim']

سوال۲- مدل رگرسون لجستیک

در مقایسه با مدل خطی که روی کلاس بندی جواب نمی داد مدل رگسیون لاجستیک را داریم که از تابعی استفاده می کنیم که همواره خروجی بین ۰ و ۱ بدهد. یکی از تابع هایی که چنین ویژگی ای را دارد، تابع لاجیستیک است. اگرچه پیاده سازی مدل رگرسون لجستیک پیچیده است اما کتابخانه هایی برای پایتون وجود دارد که با یک خط کد قابل پیاده سازی است. در اینجا از کتابخانه و روش sklearn.linear_model.LogisticRegression استفاده شده است.

در تابع رگرسون لجستیک پس از فیت کردن مدل معیار شناخته‌شده‌ی ارزیابی نتایج داده کاوی مثل Precision - Recall - و F1-score را ارایه می کنیم.

معیار Recall : حداکثر مقدار این معیار یک ویا ۱۰۰ درصد و حداقل مقدار آن صفر است و هرچه مواردی که ما انتظار داشتیم پیش بینی شوند ولی برنامه پیش بینی نکرده‌است که به آن False Negative می‌گوییم نسبت به پیش بینی‌های درست یا True Positiveبیشتر باشد مقدار Recall کمتر خواهد شد.

معیار Precision : حداکثر مقدار این معیار یک ویا ۱۰۰ درصد و حداقل مقدار آن صفر است و هرچه مواردی که برنامه به غلط پیش بینی کرده است که به آن False Positive می‌گوییم نسبت به پیش بینی‌های درست یا True Positive بیشتر باشد مقدار Precision کمتر خواهد شد.

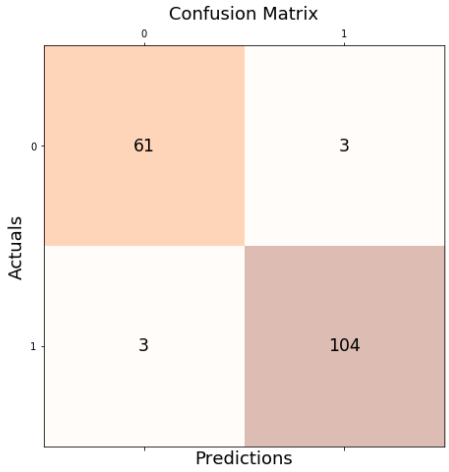
معیار f1-score : زمانی که می‌خواهید معیار ارزیابی شما میانگینی از دو مورد قبلی باشد یعنی همان Recall یا Precision می‌توانید از میانگین هارمونیک این دو معیار استفاده کنید که به آن معیار f1-score می‌گویند.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$



همه این معیارها با کتابخانه sklearn.metrics قابل پیاده سازی است.

در اینجا با استفاده از فیچرهای انتخاب شده در سوال-۱ که روش forward-selection پیشنهاد داده بود مدل رگرسیون لاجستیک را به همراه معیارهای ازریابی اجرا کرده و تنایج حاصل به صورت زیر است :

Precision: 0.968

Recall: 0.960

F1 Score: 0.964

سوال ۳ و ۴ - الگوریتم PCA و پیاده سازی رگرسیون لجستیک

تحلیل مولفه اساسی (PCA) همبستگی بین متغیرها را شناسایی میکند و از جمله روش‌های تبدیل خطی است. PCA در اصل راهی برای کاهش تعداد متغیرها و در عین حال حفظ اطلاعات مهم است. و تعدادی از متغیرها را که ممکن است همبستگی داشته باشند به تعداد کمتری از متغیرهای غیر همبسته تبدیل می کند که به عنوان اجزای اصلی شناخته می شوند. مولفه های اصلی ترکیب های خطی متغیرهای اصلی هستند که با واریانس ها (یا مقادیر ویژه) آنها در یک بعد متعامد خاص وزن شده اند. هدف اصلی آن سادگی در مصور سازی داده ها و همچنین افزایش سرعت مدل است. اما باید توجه داشته باشیم که PCA را فقط باید برای متغیرهای پیوسته اعمال کنیم، نه دسته بندی. اگرچه از نظر فنی می‌توان از PCA روی داده‌های کدگذاری شده یا باینری یک‌طرفه استفاده کرد، اما خیلی خوب کار نمی‌کند. این به این دلیل است که PCA برای به حداقل رساندن واریانس (انحرافات مربعی) طراحی شده است که وقتی روی متغیرهای باینری انجام می شود چندان معنی دار نیست. در صورتیکه داده ها مخلوط است بهتر است از سایر روش ها مانند Multiple Correspondence Analysis استفاده کرد.

در ادامه خواهیم دید که این روش روی کل داده های اموزشی ما با توجه به انکه شامل داده های باینری است خوب جواب نداده است

نسبت واریانس توضیح داده شده (pca.explained_variance_ratio_) به عنوان معیاری برای ارزیابی سودمندی اجزای و انتخاب تعداد مؤلفه های مورد استفاده در مدل استفاده می شود. نسبت واریانس توضیح داده شده درصدی از واریانس است که به هر یک از اجزای انتخاب شده نسبت داده می شود. در حالت ایده‌آل، با اضافه کردن نسبت واریانس توضیح‌داده‌شده هر مؤلفه، تعداد مؤلفه‌هایی را که در مدل گنجانده شود، انتخاب کرده تا به مجموع حدود ۰.۸ یا ۸۰ درصد برسد تا از برازش بیش از حد جلوگیری شود. این نسبت در اینجا 43.02 بدست آمده است که برای زیر ۶۰ درصد اصلا مناسب نیست.

مراحل اجرای روش pca در پایتون به شرح زیر است:

از پکیج scikit-learn استفاده می کنیم

PCA بر اساس مقیاس انجام می شود، بنابراین قبل از اعمال PCA باید ویژگی های موجود در داده های خود را مقیاس بندی کنید. از StandardScaler برای کمک به استانداردسازی ویژگی‌های مجموعه داده در مقیاس واحد (میانگین = ۰ و واریانس = ۱) استفاده کنید که لازمه عملکرد بهینه بسیاری از الگوریتم‌های یادگیری ماشین است.

داده های اصلی دارای 20 فیچر هستند ،در این بخش، کد داده های اصلی را که ۲۰ بعدی است به ۶ بعد کاهش می دهیم (انتخاب تعداد component ها براساس خروجی سوال قبل است)، باید توجه داشته باشم که پس از کاهش ابعاد، معمولاً معنای خاصی به هر جزء اصلی اختصاص داده نمی شود. اجزای جدید فقط ۶ بعد اصلی variation هستند. و در نهایت pca.explained_variance_ratio_ را بدست آورده برای هر component و مجموع و همگی را vizualize میکنیم.

با استفاده از تابع رگرسیون لجستیک پیشتر تعریف شده ، با فیچرهای حاصل از pca نیز مدل را اجرا کرده و معیارهای اصلی را بدست اوردیم که به وضوح کمتر از مدل پیشین است و یعنی کیفیت مدل افت کرده است :

Precision: 0.635

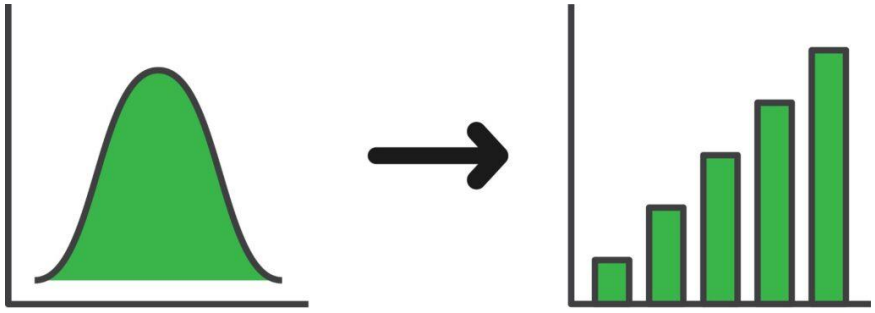
Recall: 0.546

F1 Score: 0.587

سوال۶- مهندسی ویژگی ها

الف. Binning

تکنیک binning جهت کاهش کاردینالیتی داده های پیوسته و گسسته به کار می رود. این روش با مرتبط کردن مقادیر در چند bin باعث کاهش تعداد مقادیر متمایز می شود. در نهایت این تکنیک منجر به افزایش دقت مدل خصوصا مدل های پیش بینی کننده می شود. و یک فیچر کتگوریکال از داده ها ایجاد می کند که غیر خطی بودن و نویز در دیتاست کاهش می دهد. استفاده از bin ها در اینجا اغلب به عنوان binning یا ایجاد k bin نامیده می شود، که در آن k به تعداد گروه هایی مربوط می شود که متغیر عددی از مجموعه داده به آنها نگاشت می شود .

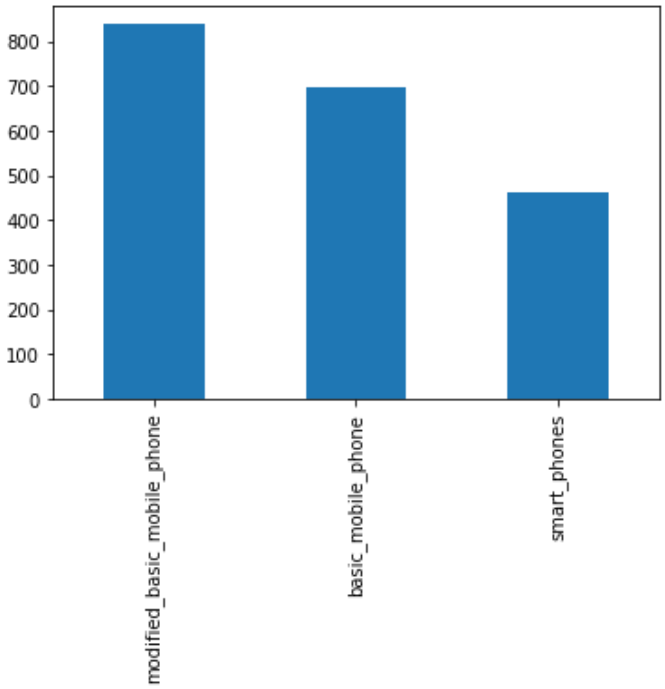


Discretization Process

روش های مختلفی برای binning وجود دارد اما در اینجا با توجه به برداشت ما از دیتاست اندازه bin ها را انتخاب می کنیم که لزوما برابر نیست. فیچر battery power می تواند نشان دهنده دسته بندی نسل های تلفن همراه باشد. با توجه به داده های موجود در مورد نسل باتری ها در تلفن همراه آن ها را به سه دسته تقسیم کرده و عدد مورد نظر را از منابع موجود بدست می اوریم:

bins = [150,1000,1650,2000]

labels = ['basic_mobile_phone','modified_basic_mobile_phone','smart_phones']



ب. one_hot_encoding

دیتاست ها گاهی شامل داده های متنی و کتگوریکال(منظور داده های غبرعددی) هستند. بسیاری از الگوریتم ها انتظار دارند مقادیر عددی باشند تا به نتایج پیشرفته دست یابند. بنابراین، چالش اصلی که یک تحلیلگر با آن مواجه است، تبدیل داده های متنی/ کتگوریکال به داده های عددی و ایجاد الگوریتم/مدلی برای معنا بخشیدن به آن است. کتابخانه SciKit-learn دارای تکنیک one_hot_encoding است که میتواند داده های غیرعددی را به داده های باینری تبدیل کند. و مزیتی که تسبت به label encoding دارد ان است که به صورت دتباله ای و ترتیب دار نیست که به داده ها اولویت خاصی بدهد.

در این استراتژی، هر مقدار دسته به یک ستون جدید تبدیل می شود و یک مقدار ۱ یا ۰ (نماد True/False) به ستون اختصاص می یابد. OneHotEncoder از کتابخانه SciKit فقط مقادیر عددی را می‌گیرد، بنابراین هر مقدار از نوع رشته باید قبل از یک one_hot_encoding برچسب‌گذاری یا label encoded شود.

_cores	...	talk_time	three_g	touch_screen	wifi	price_range	phone_types	0	1	2	3
2	...	19	0	0	1	1	basic_mobile_phone	0.0	1.0	0.0	0.0
3	...	7	1	1	0	2	modified_basic_mobile_phone	0.0	0.0	1.0	0.0
5	...	9	1	1	0	2	basic_mobile_phone	0.0	0.0	1.0	0.0
6	...	11	1	0	0	2	basic_mobile_phone	0.0	0.0	1.0	0.0
2	...	15	1	1	0	1	smart_phones	0.0	1.0	0.0	0.0

این کار با استفاده از پکیج sklearn.preprocessing و دادن ستون مورد نظر به دستور enc.fit_transform قابل اجراست.

ج. log_transform

Log Transformیکی از محبوب‌ترین تکنیک‌های تبدیل است. اساساً برای تبدیل توزیع اریب به توزیع معمولی/توزیع کم انحراف استفاده می شود. دلیل کارایی آن این است که تابع log برای مقابله با اعداد بزرگ خوب عمل می کند :

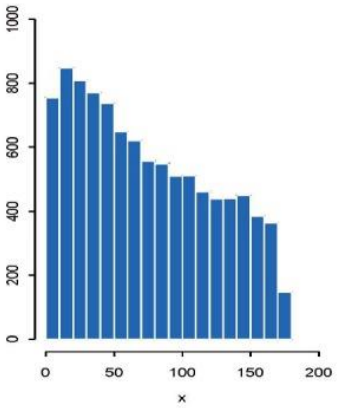
log(10) = 1

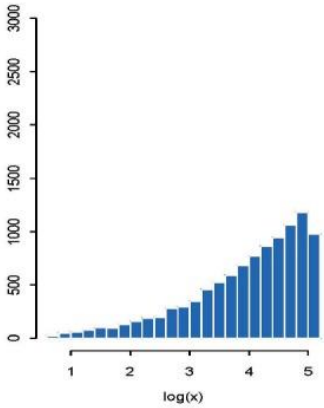
log(100) = 2, and

log(10000) = 4.

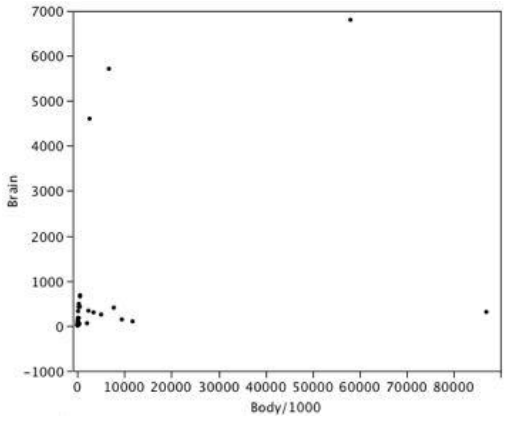
همانطور که میبینیم کاهش شدیدی در بازه داده ها ایجاد می کند پس به طور کلی :

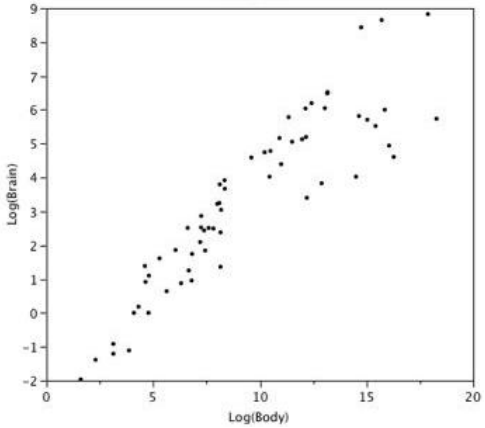
- کاهش تاثیر مقادیر خیلی کم
- کاهش تاثیر مقادیر خیلی زیاد





تبدیل log را می توان برای کم کردن توزیع های بسیار اریب (skewed)استفاده کرد. این می تواند هم برای تفسیرپذیرتر کردن الگوهای موجود در داده ها و هم برای کمک به برآورده کردن مفروضات آمار استنباطی ارزشمند باشد. شکل زیر مثالی دیگر نشان می دهد که چگونه یک تبدیل log می تواند الگوها را بیشتر نمایان کند :



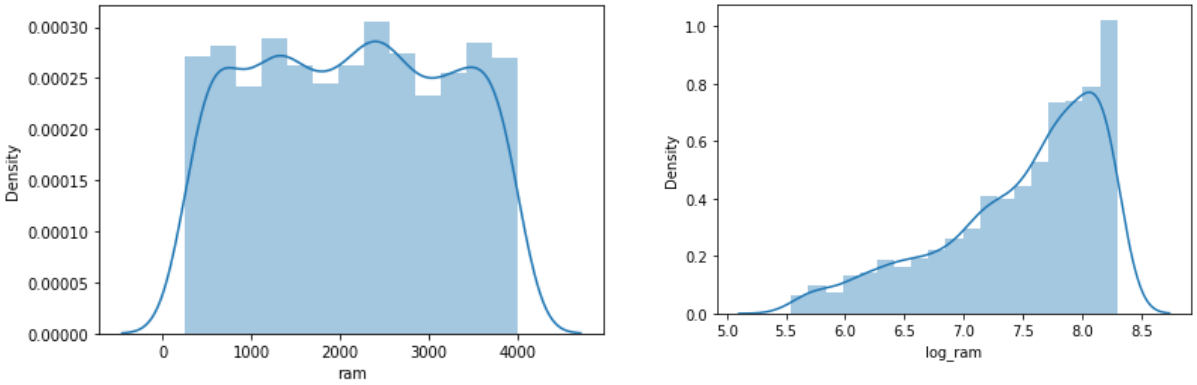


از طرفی متأسفانه، داده های حاصل از بسیاری از مطالعات، توزیع log نرمال را تقریب نمی زند، بنابراین اعمال این تبدیل، چولگی توزیع را کاهش نمی دهد. در واقع، در برخی موارد اعمال تبدیل می‌تواند توزیع را منحرف‌تر از داده‌های اصلی کند.

(https://dx.doi.org/10.3969%2Fj.issn.1002-0829.2014.02.009)

با کتابخانه numpy و دستور np.log می توان log transformation را روی هر ستون دلخواه اعمال کرد:

در زیر تاثیر log transformation را روی فیچر ram میبینیم:



به نظر می رسد normalization در اینجا transformation بهتری باشد.

د. ساخت فیچر جدید

با توجه داده ها می توان به سادگی فیچر های جدید مانند مساحت یا حجم یا باتری بر حسب زمان و ... را از روی فیچر های قبلی با عملیات ساده ریاضی بدست اورد :

$$px\_total=px\_height*px\_width$$

$$volume=sc\_w*sc\_h*m\_dep$$

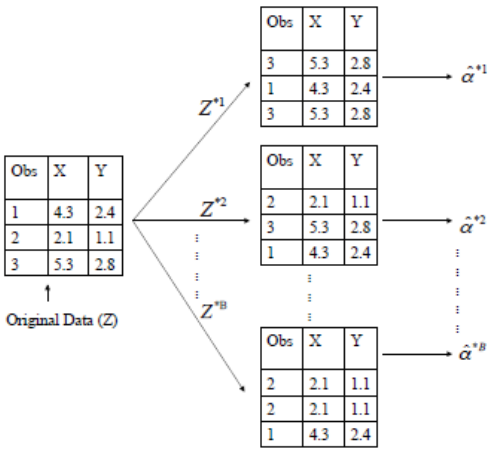
سوال۷- اعمال مدل رگرسون لجستیک روی سوال ۶

با تابع تعریف شده رگرسیون لجستیک در مراحل پیشین، تغییرات روی داده ها را در نظر گرفته و مدل را برای هر ۴ مورد مهندسی شده اجرا می کنیم.

سوال ۸ – bootstrap vs cross_validation

روش های bootstrapو cross_validation هر دو از روش های resampling داده هستند bootstrapیک ابزار آماری قابل قبول و قدرتمند است که می تواند برای تعیین کمیت عدم قطعیت مرتبط با یک estimator معین یا روش یادگیری آماری استفاده شود. می تواند standard error of a coeficient را تخمین بزند یا confidence interval ضرایب را تخمین بزند.

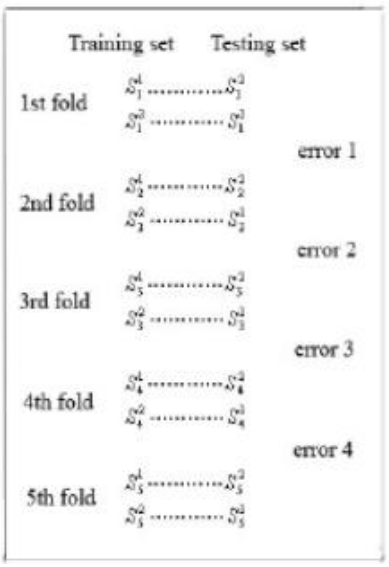
روش ان به این صورت است که از روی نسخه اورجینال دیتاست B تا bootstrap dataset می سازیم به طور تصادفی و برای این انتخاب هر بار یک نمونه از دیتاست رندوم برداشته و در دیتاست bootstrap کپی کرده اما از نسخه اورجینال حذف نمی کنیم. و دوباره این کار را تکرار می کنیم تا در نهایت B تا دیتاست هم اندازه با دیتاست اصلی اما به روش رندوم ساخته شود.



این روش به نوعی در کاهش واریانس خطا موثر است چون B تا تخمین زده و میانگین می گیرد و با میانگین گیری واریانس کاهش می یابد. با این کار مقادیر بدست امده به مقدار واقعی نزدیک تر است.

cross validation داده های اصلی را به دو مجموعه training و validation تقسیم می کنیم و در k-fold هم به همین صورت و فقط k-1 training و 1 به validation اختصاص می یابد. به طور کلی cross validation مجموعه داده موجود را برای ایجاد مجموعه داده های متعدد تقسیم می کند و روشbootstrap از مجموعه داده اصلی برای ایجاد مجموعه داده های متعدد پس از نمونه برداری مجدد با جایگزینی استفاده می کند.

سوال۹- 5*2 cross validation



در این روش ۵ همانندسازی از 2-fold cross validation اجرا می کنیم. در هر همانند سازی داده ها به دو مجموعه هم سایز s_1 , s_2 تقسیم می شوند. هر الگوریتم آموزش روی یک مجموعه train و مجموعه دیگر test می شود که منجر به ۴ تخمین خطا می شود (شکل روبرو) تعداد همانند سازی ها دلخواه نیست و بنا به نیاز test محاسبه شده است، به عبارتی مطالعات نشان داده است که در 2-fold cross validation تنها ۵ همانند سازی موثر است و هر عددی کمتر یا بیشتر از ۵ ریسک خطای نوع یک که باید کم باشد را افزایش می دهد. ایراد اصلی این روش برابر بودن داده های test train در هر fold است که باعث می شود هربار نیمی از داده ها از فرآیند اموزش حذف شوند. این روش نه تنها یک تخمین خوب از خطای کلی است بلکه تخمین خوب از واریانس خطا است.

چه جاهایی می تواند مفید باشد ؟ گاهی وقتی از k-fold cv paired t test استفاده می کنیم t statistic بدست امده بسیار بزرگ است. صورت t statistic اختلاف میانگین در اجرای دو الگوریتم را (در k-fold) تخمین می زند در حالیکه مخرج واریانس این اختلاف را اندازه می گیرد. با داده های مصنوعی ما kتا مجموعه اموزش غیرهمپوشان ایجاد می کنیم و میانگین و واریانس مجموهخ ها اموزش را محاسبه می کنیم. و میبینیم وقتی واریانس under estimate شده هنگامیکه مجموهخ های اموزش همپوشان باشند، میانگین ها گهگاه شدیدا بد تخمین زده می شوند و این علت مقادیر بزرگ t بود. این مشکل با correlation بین fold های مختلف قابل حل است. به عبارتی با جایگزیبنی صورت t statistic با اختلاف مشاهده شده حاصل از تک فولد از k fold cv ، اماره خوش رفتار می شود و این همان استفاده از 5*2 cross validation پیشتر توضیح داده شده است.

سوال۱۰- الگوریتم های ساخت درخت تصمیم

تصمیم در مورد تفکیک استراتژیک به شدت روی دقت درخت تصمیم اثر می گذارد. ملاک این تصمیم برای درخت های کلاسیندی و رگرسیون متفاوت است. درخت های تصمیم از الگوریتم های گوناگونی برای جداسازی گره ها به دو یا بیشتر وجود دارد. ساخت زیرگره ها منجر به افزایش همگنی حاصل زیر گره ها می شود. به عبارتی خلوص گره ها با توجه به متغیر هدف افزایش می یابد. انتخاب الگوریتم بر اساس نوع متغیر هدف است، اصلی ترین این الگوریتم ها عبارتند از :

- ID3 → (extension of D3)
- C4.5 → (successor of ID3)
- CART → (Classification And Regression Tree)
- CHAID → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)
- MARS → (multivariate adaptive regression splines)

الگوریتم ID3 به عنوان الگوریتم بیسیک در ساخت درخت محسوب می شود. این الگوریتم یک درخت multiway ایجاد می‌کند و برای هر گره (یعنی به شیوه‌ای حریصانه) ویژگی طبقه‌بندی را پیدا می‌کند که بیشترین information gain را برای اهداف طبقه‌بندی به همراه خواهد داشت. درختان تا حداکثر اندازه خود رشد می کنند و سپس یک مرحله هرس معمولاً برای بهبود توانایی درخت در تعمیم داده های unseen اعمال می شود.

الگوریتم C4.5 جانشین ID3 است و با تعریف پویای یک ویژگی گسسته (بر اساس متغیرهای عددی) که مقدار مشخصه پیوسته را به مجموعه‌ای از فواصل گسسته تقسیم می‌کند، محدودیت آنکه ویژگی‌ها باید categorical باشند حذف کرد. C4.5درختان آموزش دیده (یعنی خروجی الگوریتم ID3) به مجموعه ای از قوانین if-then تبدیل می کند. سپس این دقت(accuracy) هر قانون جهت تعیین ترتیبی که باید اعمال شوند، ارزیابی می‌شود. هرس با حذف پیش شرط یک قانون انجام می شود اگر دقت قانون بدون آن بهبود یابد. الگوریتم به شرح زیر است :

Algorithm 1.1 C4.5(D)

Input: an attribute-valued dataset D

```
Tree = {}
if D is "pure" OR other stopping criteria met then
    terminate
end if
for all attribute a ∈ D do
    Compute information-theoretic criteria if we split on a
end for
a_best = Best attribute according to above computed criteria
Tree = Create a decision node that tests a_best in the root
D_v = Induced sub-datasets from D based on a_best
for all D_v do
    Tree_v = C4.5(D_v)
    Attach Tree_v to the corresponding branch of Tree
end for
return Tree
```

الگوریتم CART (درخت طبقه بندی و رگرسیون) بسیار شبیه به C4.5 است، اما تفاوت آن در این است که از متغیرهای هدف عددی (رگرسیون) پشتیبانی می کند و مجموعه قوانین را محاسبه نمی کند. CART با استفاده از ویژگی و آستانه ای که بیشترین بهره اطلاعاتی(info gain) را در هر گره ایجاد می کند، درخت های دودویی را می سازد. الگوریتم CART از معیار Gini Index برای تقسیم یک گره به یک گره فرعی استفاده می کند. با مجموعه آموزشی به عنوان یک گره ریشه شروع می شود، پس از تقسیم موفقیت آمیز گره ریشه به دو بخش، زیر مجموعه ها را با استفاده از منطق مشابه تقسیم می کند و دوباره زیر مجموعه ها را تقسیم می کند، به صورت بازگشتی تا زمانی که split بیشتری پیدا کند، هیچ گره فرعی خالص یا حداکثر تعداد برگ در یک درخت در حال رشد ایجاد نمی کند یا آن را به عنوان هرس درخت می نامند.

در نگاهی کلی تفاوت الگوریتم های مشهور ساخت درخت تصمیم را در زیر میبینیم :

Features	*ID3*	*C4.5*	*CART*
Type of data	Categorical	Continuous and Categorical	continuous and nominal attributes data
Speed	Low	Faster than ID3	Average
Boosting	Not supported	Not supported	Supported
Pruning	No	Pre-pruning	Post pruning
Missing Values	Can't deal with	Can't deal with	Can deal with
Formula	Use information entropy and information Gain	Use split info and gain ratio	Use Gini diversity index

سوال۱۱-

ساخت درخت تصمیم از دیتاست با پکیج

کتابخانه scikit-learn دارای مدل درخت تصمیم برای مسایل کلاسبندی است : DecisionTreeClassifier

در نظر داریم که در کتابخانه scikit-learn مقادیر پیش‌فرض پارامترهای کنترل‌کننده اندازه درختان (به عنوان مثال max_depth، min_samples_leaf، و غیره) منجر به درختان کاملاً رشد کرده و هرس نشده می‌شوند که به طور بالقوه می‌توانند در برخی از مجموعه‌های داده بسیار بزرگ باشند. برای کاهش مصرف حافظه، پیچیدگی و اندازه درختان باید با تنظیم آن مقادیر پارامتر کنترل شود. در اینجا ابتدا یک درخت پیشفرض با داده های دیتاست اصلی می سازیم.

مراحل ساخت :

۱. مدل مورد نظر را import می کنیم
۲. داده های تست و اموزش را جداسازی کرده و مدل را initilize می کنیم
۳. مدل را به داده ها fit میکنیم
۴. پارامترهای مورد نیاز را ارزیابی می کنیم
۵. درخت را vizualize می کنیم

برای جلوگیری از overfit شدن مدل و کارایی بهتر می توان این مراحل را با اسکیل کردن(دستور StandardScaler()) داده ها تکرار کرد و نتایج را بررسی کرد همانطور که میبینیم اندکی روی داده های تست پاسخ بهتری دریافت کردیم

سوال۱۲- ارزیابی پارامترهای درخت تصمیم

پارامترهایی مثل accuracy,percision,recall,f1 score را روی درخت بررسی کردیم.

هنگامی که در مورد بهینه‌سازی عملکرد درخت تصمیم صحبت می‌شود، معمولاً به به دست آوردن بهترین معیار ممکن مدنظر نیست، بلکه حداقل کردن مشکل overfitting مورد بحث است. به طور کلی، اگر از مدل درخت تصمیم "out-of-the-box" استفاده کنیم، یک درخت بسیار پیچیده ایجاد می کند که هر نمونه گره برگ خود را دارد. مطمئناً چنین رویکردی به دستیابی به خطای صفر در مجموعه آموزشی کمک می کند. با این حال، مدل تعمیم ضعیفی خواهد داشت و عملکرد آن بر روی داده‌های دیده نشده بسیار مورد نظر باقی خواهد ماند.

در فایق آمدن بر این مشکل راهکارهایی وجود دارد:

۱. مهندسی ویژگی
۲. راهکارهای کاهش ابعاد :pca
۳. Hyperparameter Tuning
۴. راهکارهای مدیریت عدم تعادل در داده ها : Sampling techniques

در اینجا به Hyperparameter Tuning می پردازیم.

به طور کلی تنظیم Hyperparameter جستجوی فضای فراپارامتر برای مجموعه ای از مقادیر است که معماری مدل را بهینه می کند. در Decision tree classifier پارامتر criterion پارامتر اندازه گیری کیفیت یک split در درخت است. معیارهای پشتیبانی شده "gini" برای ناخالصی Gini و "آنتروپی" برای به دست آوردن اطلاعات هستند. اما بسیاری از محققان اشاره می کنند که در بیشتر موارد، انتخاب معیارهای تقسیم تفاوت چندانی در عملکرد درخت ایجاد نمی کند. تابع splitter استراتژی مورد استفاده برای انتخاب تقسیم در هر گره است. پارامتر max_depth حداکثر عمق درخت. اگر None باشد، گره‌ها تا زمانی که همه برگ‌ها خالص شوند یا تا زمانی که همه برگ‌ها کمتر از min_samples_split، نمونه داشته باشند، گسترش می‌یابند.

به طور کلی، هرچه عمیق‌تر اجازه دهید درختتان رشد کند، مدل شما پیچیده‌تر می‌شود، زیرا تقسیم‌های بیشتری خواهید داشت و اطلاعات بیشتری در مورد داده‌ها می‌گیرد و این یکی از دلایل ریشه‌ای بیش از حد برازش(overfitting) در درخت‌های تصمیم است، زیرا مدل شما برای داده های آموزشی کاملاً مناسب است و نمی تواند در مجموعه آزمایشی به خوبی تعمیم یابد. بنابراین، اگر مدل شما بیش از حد برازش دارد، کاهش عدد برای max_depth یکی از راه‌های مبارزه با overfitting است.

همچنین داشتن عمق بسیار کم بد است، بنابراین بهتر است که ابتدا به مدل اجازه داد حداکثر عمق را تعیین کند و سپس با مقایسه score آموزش و تست‌ها به دنبال overfitting یا underfitting بود و بسته به درجه، حداکثر عمق را کاهش یا افزایش داد.

پارامتر min_samples_split حداقل تعداد نمونه مورد نیاز برای تقسیم یک گره داخلی است. طبق یک مطالعه تجربی روی تنظیم فراپارامتر درخت‌های تصمیم ، مقادیر ایده‌آل min_samples_split بین ۱ تا ۴۰ برای الگوریتم CART که

الگوریتم پیاده‌سازی شده در scikit-learn است، می باشد. min_samples_split برای کنترل overfitting استفاده می شود. مقادیر بالاتر، یک مدل را از یادگیری روابطی که ممکن است برای نمونه خاص انتخاب شده برای یک درخت بسیار خاص باشد، باز می دارد. مقادیر بیش از حد بالا نیز می‌تواند منجر به underfitting شود، از این رو بسته به سطح underfitting یا overfitting ، می‌توانید مقادیر را برای min_samples_split تنظیم کنید.

روش Hyperparameter tunning برای کاهش overfitting مناسب است. هنگام تنظیم فراپارامترهای یک مدل، باید مراقب و دقیق بود، زیرا Hyperparameter ها می توانند عملکرد مدل را به شدت تحت تاثیر قرار دهند. تنظیم هایپرپارامتر با به حداکثر رساندن یا به حداقل رساندن متریک مشخص شده کار می کند. راه های بسیاری برای تنظیم هایپرپارامترها وجود دارند که در اینجا از روش RandomizedSeachCV استفاده شده است.

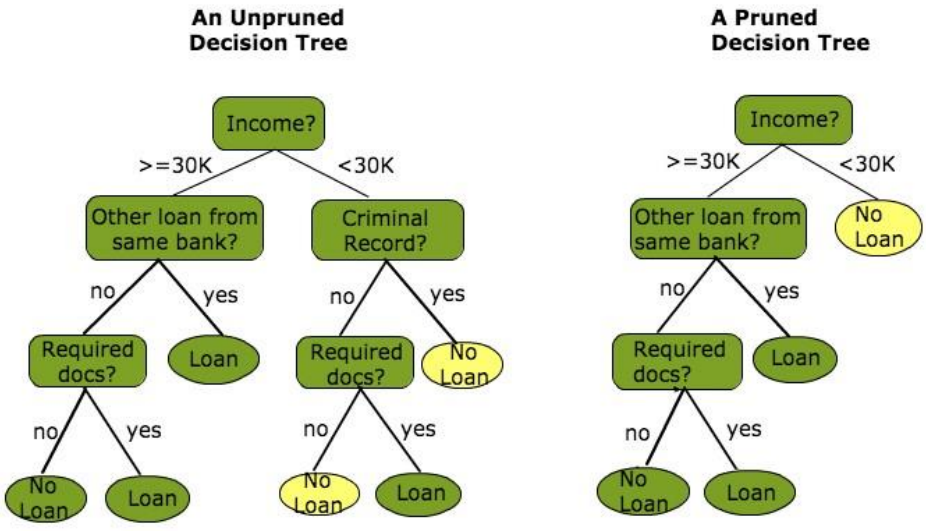
RandomizedSearchCVیک روش "fit" و "score" را پیاده سازی می کند. پارامترهای تخمینگر که برای اعمال این متد استفاده می شوند با cross validation بهینه می شوند cv در دستور منظور cross validation است. دقت مدل را پس از اعمال تنظیم هایپرپارامترها می ستنجیم و داریم :

Precision on train: 0.895

Precision on test: 0.815

سوال۱۳- هرس کردن درخت تصمیم، چگونه و چرا

هرس یک تکنیک فشرده‌سازی داده در الگوریتم‌های یادگیری ماشین و جستجو است که با حذف بخش‌هایی از درخت که برای طبقه‌بندی نمونه‌ها غیر بحرانی و زائد هستند، اندازه درخت‌های تصمیم را کاهش می‌دهد.



هرس درخت تصمیم را می توان به دو نوع تقسیم کرد: pre-prunnnig , post-prunning

پیش هرس (pre-prunnnig)، همچنین به عنوان قانون توقف زودهنگام شناخته می شود، روشی است که در آن ساخت زیردرخت در یک گره خاص پس از ارزیابی برخی اندازه گیری ها متوقف می شود. این اندازه گیری ها می تواند ناخالصی جینی (gini index) یا info gain باشد. در پیش هرس، شرایط هرس را بر اساس اقدامات فوق در هر گره ارزیابی می کنیم. شرایطی مانند :

informationGain(Attr)> minGain

treeDepth == MaxDepth

اگر شرط برآورده شد، درخت فرعی را هرس می کنیم. یعنی گره تصمیم را با یک گره برگ جایگزین می کنیم. در غیر این صورت، ساخت درخت را با استفاده از الگوریتم درخت تصمیم خود ادامه می دهیم. پیش هرس مزیت این را دارد که سریعتر و کارآمدتر باشد زیرا از ایجاد زیردرختهای بیش از حد پیچیده که بیش از حد بر داده های آموزشی منطبق هستند جلوگیری می کند. با این حال، در پیش هرس، رشد درخت پیش از موعد با شرایط توقف ما متوقف می شود.

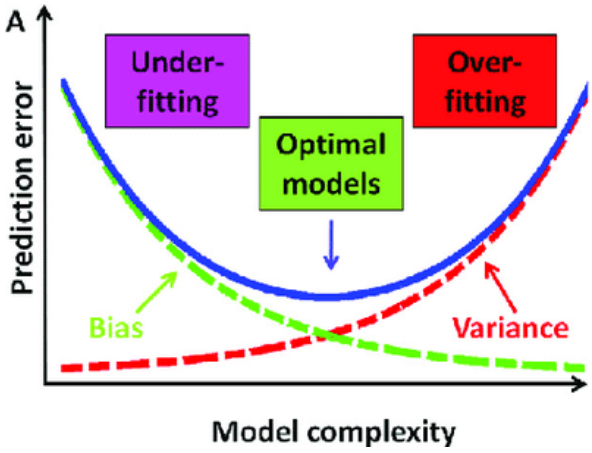
پس از هرس (post-prunning) همانطور که از نام آن پیداست، به معنای هرس کردن پس از ساخت درخت است. شما درخت را به طور کامل با استفاده از الگوریتم درخت تصمیم خود رشد می دهید و سپس درختان فرعی درخت را به صورت پایین به بالا هرس می کنید. شما از گره تصمیم پایین شروع می کنید و بر اساس معیارهایی مانند Gini Impurity یا Information Gain، تصمیم می گیرید که آیا این گره تصمیم را حفظ کنید یا آن را با یک گره برگ جایگزین کنید. برای مثال، فرض کنید می‌خواهیم درخت‌های فرعی را هرس کنیم که کمترین اطلاعات را به دست می‌آورند. هنگام تصمیم گیری برای گره برگ، می خواهیم بدانیم اگر الگوریتم درخت تصمیم ما این گره تصمیم را ایجاد نمی کرد، چه برگی ایجاد می کرد.

هرس درخت تصمیم به جلوگیری از برازش بیش از حد داده های آموزشی کمک می کند تا مدل ما به خوبی به داده های دیده نشده(unseen) تعمیم یابد. هرس درخت تصمیم به معنای حذف درخت فرعی است که زائد است و شکاف مفیدی ندارد و آن را با یک گره برگ جایگزین می کنیم. به طور کلی هرس پیچیدگی طبقه‌بندی‌کننده نهایی را کاهش می‌دهد و در نتیجه دقت پیش‌بینی را با کاهش بیش‌برازش بهبود می‌بخشد.

سوال۱۴- بهترین مرتبه مدل با استفاده از Elbow

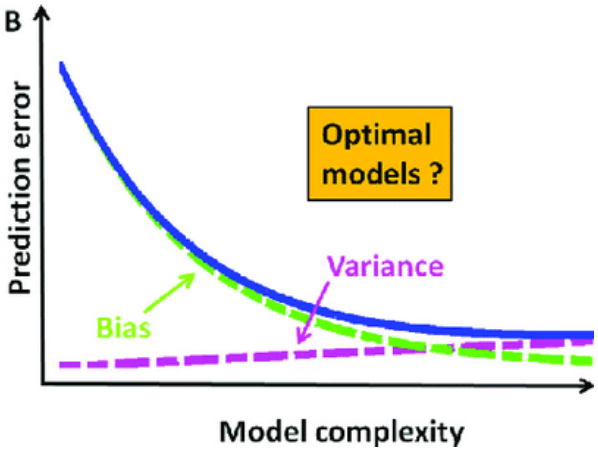
در خوشه‌بندی K-means، روش آرنج و تکنیک‌های تجزیه و تحلیل یا امتیاز برای یافتن تعداد خوشه‌ها در یک مجموعه داده استفاده می‌شود. روش elbow برای یافتن نقطه "ارنج" استفاده می شود، جایی که افزودن نمونه داده های اضافی عضویت خوشه

را تغییر چندانی نمی دهد. این زاویه در نمودار زمانی پیدا می شود که مجموعه داده پس از اعمال الگوریتم تحلیل خوشه ای، صاف یا خطی شود. نمودار elbow ، شکستگی را در نقطه ای نشان می دهد که تعداد خوشه ها شروع به افزایش می کند.



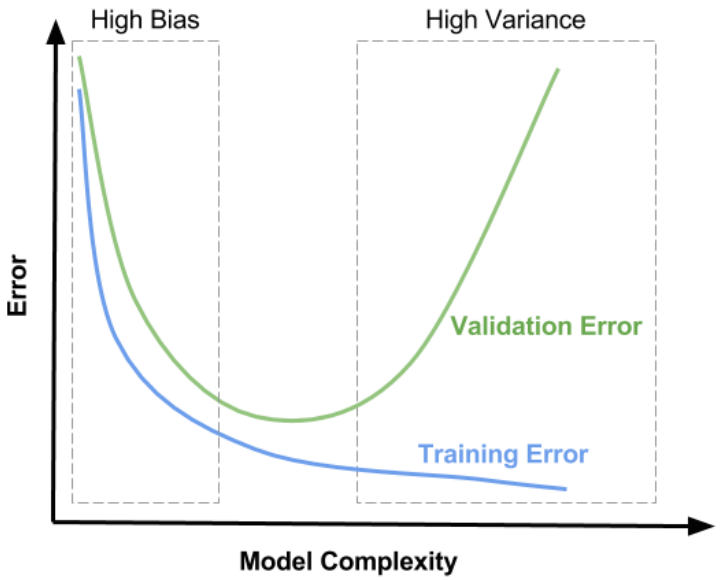
به طور شهودی، افزایش تعداد خوشه‌ها به طور طبیعی fit را بهبود می‌بخشد (تغییر را بیشتر توضیح میدهد)، زیرا پارامترهای بیشتری (خوشه‌های بیشتر) برای استفاده وجود دارد، اما در برخی مواقع باعث بیش برازش یا overfitting، و elbow این را منعکس می‌کند. در عمل ممکن است یک آرنج تیز(sharp elbow) وجود نداشته باشد، و به عنوان یک روش اکتشافی، چنین "آرنج" را نمی توان همیشه به طور واضح شناسایی کرد.

اما آیا همیشه چنین مدلی پاسخگو است ؟ اگر به نمودار زیر نگاه کنیم میبینیم افزایش واریانس در مقابل سرعت کاهش بایاس به کندی صورت می گیرد، در چنین موردی برآورد درجه بهینه مدل پیچیده می شود. به طوریکه global minimum به سادگی قابل تشخیص نیست و prediction error به مدل با درجه بالاتر انتخاب overfitting می شود. ممکن است نیاز به استفاده error باشیم.



نمی توان مدل بهینه را یافت و اگر عنوان معیار در نظر گرفته شود می شود که منجر به برای جلوگیری از چنین اتفاقی معیاری به غیر از prediction

با توجه به ان که overfitting منجر به عدم پاسخگویی مناسب مدل در برابر نمونه های تست می شود نمی توان چنین رویکردی را مورد استفاده قرار داد. نتیجه این انتخاب را در زیر می توان دید : اگر از روش elbow استفاده کرده بودیم مرتبه مدل به مراتب کوچکتر بوده است اما همانطور که میبینید این مرتبه در داده های تست به وضوح متفاوت است.


High Bias
High Variance
Error
Validation Error
Training Error
Model Complexity