تمرین ۲
فائزه علی‌نژاد
زهرا کشاورز رضائی
دانشگاه صنعتی امیرکبیر
دانشکده ریاضی و علوم کامپیوتر
مهر ۱۴۰۳

سوال ۱

Monte-Carlo Cross Validation: این رویکرد CV ابتدا یک کسری از مجموعه داده را به عنوان زیر مجموعه آزمایشی انتخاب می‌کند. در هر مرحله اعضای زیر مجموعه آزمایشی را به صورت تصادفی و با جایگذاری انتخاب می‌کند و باقی مجموعه داده را به عنوان داده‌ی آموزشی قرار می‌دهد. این فرایند را چندین بار انجام می‌دهد. برای ارزیابی عملکرد مدل در این اعتبارسنجی، میانگینی بر روی تمام تکرارهای این عملیات می‌گیرد. در این روش یک نمونه داده می‌تواند در بیش از یک تکرار جزو داده‌های آزمایشی قرار بگیرد. یکی از معایب این روش نسبت به k-fold CV این است که بعضی از نمونه داده‌ها ممکن است هرگز در دسته داده آموزشی قرار نگیرند. با توجه به اینکه ممکن است بعضی از نمونه داده‌ها به صورت تکراری انتخاب شوند و مجموعه آزمایشی‌های مختلف هم‌پوشانی داشته باشند، احتمال درز داده وجود دارد. از دیگر معایب این رویکرد هزینه محاسباتی بالای آن است. از مزایای این رویکرد می‌توان اشاره کرد که با توجه به آزادی عمل بیشتر در انتخاب مجموعه آزمایشی، این رویکرد برای محموعه داده‌های بزرگ بهتر عمل می‌کند.

بر روی محموعه داده‌های کوچک رویکرد k-fold عموما بهتر عمل می‌کند و از مزایای این رویکرد به شمار می‌رود. از دیگر مزایای رویکرد k-fold این است که این است هر داده دقیقا یکبار به عنوان داده آزمایشی قرار می‌گیرد. می‌توان گفت تعادلی میان واریانس و بایاس آن وجود دارد. از معایب آن هزینه محاسباتی بالای آن است. همچنین تعداد حالت‌هایی که در این رویکرد مجموعه داده می‌تواند تقسیم شود، محدود است.

بایاس در هر دو رویکرد k-fold CV و Monte-Carlo CV با توجه به اندازه مجموعه آموزشی، هرچقدر بزرگتر باشد مقدار بایاس کمتر است و هرچقدر مجموعه آزمایشی کوچک‌تر می‌شود، بایاس بیشتر می‌شود.

Monte- Carlo نسبت به k-fold واریانس بیشتری دارد، چرا که مجموعه آزمایشی در هر تکرار به صورت تصادفی انتخاب می‌شود. یک رابطه عکس بین واریانس و بایاس وجود دارد؛ هر چقدر که در جهت بهبود بایاس برویم و اندازه مجموعه آموزشی را کمتر کنیم، واریانس افزایش میابد.

Nested Cross Validation: این رویکرد عمل Cross Validation را در دو مرحله انجام می‌دهد. از CV برای تنظیم هایپرپارامترها و ارزیابی مدل استفاده می‌شود. اگر از دیتای یکسان برای آموزش و ارزیابی مدل استفاده کنیم، ارزیابی مدل خوشبینانه‌تر می‌شود. این مدل شامل دو لایه درونی و بیرونی است. در لایه بیرونی ابتدا داده‌ها را به چند بخش تقسیم می‌کنیم؛ سپس یکی از این بخش‌ها را به عنوان داده تست قرار می‌دهیم و باقی را داده آموزشی. حال این داده آموزشی وارد لایه داخلی می‌شود. در لایه داخلی با استفاده از CV هایپرپارامترها را تنظیم می‌کنیم. پس از تنظیم هایپرپارامترها به لایه بیرونی برمی‌گردیم و با توجه به مقادیر تنظیم شده ارزیابی مدل انجام می‌دهیم. این کار را برای تمامی ترکیب بخش‌ها تکرار می‌کنیم. برای ارزیابی نهایی عملکرد این اعتبارسنجی، بر روی تمامی مقادیر به دست آمده میانگین می‌گیریم.
با توجه به جداسازی داده آموزشی و آزمایشی، بایاس در اینجا کم است و واریانس بالاست.
از معایب این مدل می‌توان به هزینه محاسباتی بالا و پیچیدگی آن اشاره کرد. همچنین واریانس در اینجا بالاست. مزایا آن این است که از بیش برازش جلوگیری می‌کند. همچنین در این اعتبارسنجی، نتایج عادلانه‌تری به دست می‌آید.

سوال ۲

در کلاس‌هایی که داده کمتر است بایاس افزایش میابد چرا که داده در این کلاس‌ها کمتر است و یادگیری به خوبی انجام نمی‌شود و مدل به سمت کلاس‌هایی که داده بیشتری دارند، تمایل است. به دلیل مشابه واریانس مدل نیز افزایش میابد و چون مدل این کلاس‌ها را به خوبی یاد نگرفته، این کلاس‌ها پایدار نیستند. یکی از پرکاربردترین راهکارها برای حل این مشکل oversampling است؛ که در این حالت داده‌های کلاس‌هایی که کمتر شایع هستند را تکرار می‌کنیم تا داده جدید بسازیم. یا به صورت معادل می‌توان از راهکار Undersampling استفاده کرد و تعداد نمونه داده‌های کلاس شایع را کاهش داد. راهکار دیگری که در مواجه با چنین داده‌ای می‌توان به کار برد استفاده از مدل‌هایی‌ست که قابلیت این را دارند که به وزن حساس باشند، مانند درخت تصمیم که می‌توان به کلاس کمتر شایع وزن بیشتری اختصاص داد. همچنین می‌توان از معیار ارزیابی مناسب‌تری نسبت به Accuracy استفاده کرد؛ چرا که این معیار نسبت به کلاس‌های کمتر شایع نیز حساس است و عملا دقت ۹۹ درصد وقتی که داده بالانس نیست، معنایی ندارد. به عنوان مثال می‌توان از precision و recall استفاده کرد که ارزیابی بهتری می‌تواند ارائه دهد.

سوال ۳

در برداری سازی، عملیات‌هابه صورت موازی انجام می‌شوند در حالی که وقتی از حلقه استفاده می کنیم در هر مرحله فقط یک عملیات انجام می‌شود که این موجب افزایش سرعت محاسبات می‌شود. تعداد عملیات‌ها نسبت به روش‌های حلقه کاهش میابد و پیچیدگی کد کمتر می‌شود عموما وقتی از برداری‌سازی استفاده می‌کنیم کد کوتاه‌تر شده و ابن زمانی که بخواهیم کدها را بازبینی کنیم، بسیار کمک‌کننده خواهد بود. حافظه بهینه‌تر مدیریت می‌شود و سربار حافظه کاهش میابد.

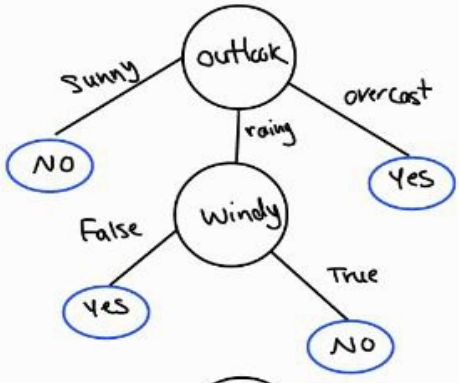
سوال ۴

درخت تصمیم ممکن است وابسته به شرایط می‌تواند ٪۱۰۰ بر روی داده آموزشی عمل نکند. به عنوان مثال فرض کنید بر روی عمق درخت تصمیم محدودیت داشته باشیم، در این حالت درخت تصمیم بر روی داده دقت ٪۱۰۰ نخواهد داشت. حالت دیگری وقتی‌ست که ویژگی پیوسته داشته باشیم، در این صورت باید داده را گسسته‌سازی کنیم و اگر در بین بازه‌هایی که داریم اشتراک و هم‌پوشانی داشته باشیم، مدل در تشخیص کلاس به مشکل برمی‌خورد و به درستی تشخیص نمی‌دهد.

سوال ۵

$R_\alpha(T) = R(T) + \alpha|T|$ $\quad R(T) = 2/14$

$\alpha = 0.1$



↳ To prune Humidity

$R_\alpha(H) = 2/14 + 0.1 \times 4 = 0.54$

$R_\alpha(T) = 0 + 0.1 \times 5 = 0.150$

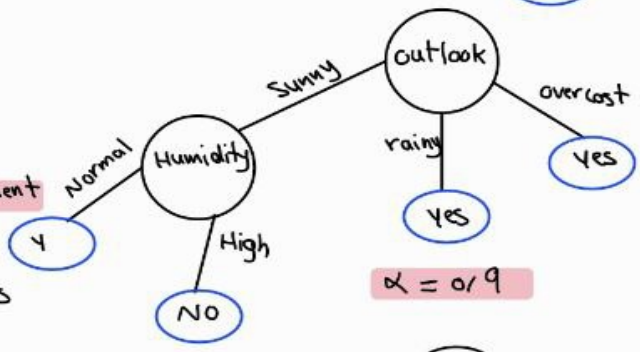
$R_\alpha(T) < R_\alpha(H) \rightarrow$ No improvement ×

↳ To prune Windy

$R_\alpha(W) = \frac{2}{14} + 0.1 \times 4 = 0.154$

$R_\alpha(T) = 0 + 0.1 \times 5 = 0.150$

$R_\alpha(T) < R_\alpha(W) \rightarrow$ No improvement ×



es

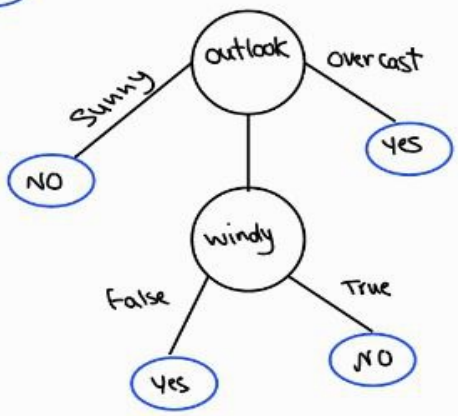
$\alpha = 0.9$

↳ To prune Humidity

$R_\alpha(H) = 2/14 + 0.9 \times 4 = 3.74$

$R_\alpha(T) = 0 + 9 \times 5 = 4.5$

$R_\alpha(T) > R_\alpha(H) \rightarrow$ improvement ✓

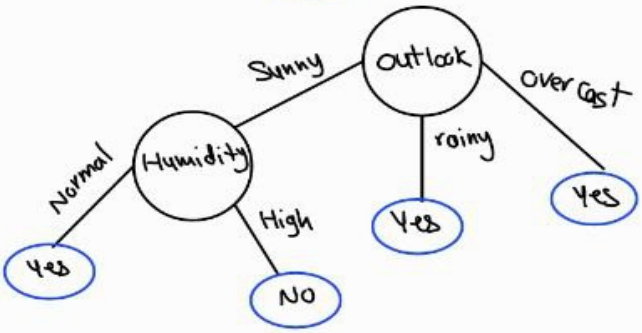


↳ To prune Windy

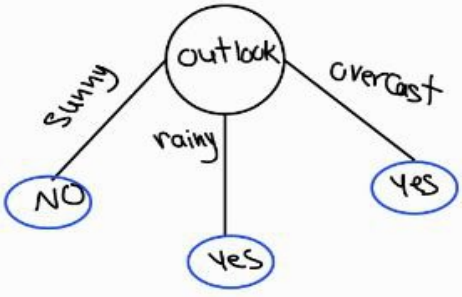
$R_\alpha(\text{windy}) = \frac{2}{14} + 0.9 \times 4 = 3.74$

$R_\alpha(T) = 0 + 0.9 \times 5 = 4.5$

$R_\alpha(T) > R_\alpha(W) \rightarrow$ improvement ✓



After pruning →



سوال ۶

ریشه این نام‌گذاری در اصطلاح "to pull oneself up by one's bootstraps" است که به دست‌یابی به موفقیت یا بهبود وضعیت بدون کمک خارجی و با تلاش خود اشاره دارد. در این روش به جای داده‌ی جدید مجموعه داده از داده‌های موجود ایجاد می کند.

مزایا:

- هیچ پیش‌فرضی برای توزیع داده‌ها وجود ندارد؛ که آن را گزینه مناسبی برای انواع مختلف مجموعه داده‌ها می کند.
- می‌تواند در جایگاه‌های مختلفی از مدل آماری و مجموعه داده‌ها، از جمله معیارهای پیچیده مانند ضرایب رگرسیون و سایر پارامترهای مدل استفاده کرد.
- از لحاظ مفهومی ساده و قابل پیاده‌سازی با ابزارهای محاسباتی مدرن است

معایب:

- به نمونه گیری‌های مجدد زیادی نیاز دارد که می تواند هزینه محاسباتی زیادی به خصوص برای مساله‌هایی با مجموعه داده بزرگ داشته باشد.
- اگر مجموعه داده‌ها بسیار کوچک باشد یا واریانس داده‌ها بالا باشد و داده‌ها پراکنده باشند، ممکن است خوب عمل نکند و دچار ناپایداری شود.
- نسبت به داده پرت حساس است. چرا که در این روش ما داده‌ها را با جایگذاری انتخاب می‌کنیم و یک داده پرت ممکن است چندین بار در نمونه ظاهر شود که بایاس را افزایش داده و باعث ضعف عملکرد می‌شود.

در سال ۱۹۸۳، بردلی افرون، این ارزیابی را مطرح کرد که برای کاهش خطای نمونه‌گیری به‌کار می‌رود. در این روش برای ارزیابی عملکرد مدل از ترکیبی از داده‌های آموزشی و آزمایشی استفاده می‌شود. عدد 0.632 از آنجا به دست می‌آید که به طور متوسط، تقریباً 63.2٪ از نمونه‌های اصلی در هر نمونه‌گیری Bootstrap قرار می‌گیرند و بقیه‌ی داده‌ها در نمونه‌گیری وارد نمی‌شوند.که به عنوان داده‌های آزمایش یا خارج از کیف (Out of Bag) می‌مانند. در هر تکرار نمونه‌گیری به روش Bootstrap داده‌های نمونه‌گیری‌شده برای آموزش مدل استفاده می‌شوند و داده‌هایی که در این نمونه نیستند به عنوان داده‌های آزمایش برای ارزیابی دقت مدل به کار می‌روند. در Bootstrap 0.632 دقت مدل با میانگین‌گیری از ارزیابی‌های انجام‌شده بر روی داده به شکل زیر محاسبه می‌شود:

$$ACC_{boot} = \frac{1}{b}\Sigma_{i=1}^{b}(0.632ACC_{OOB} + 0.368ACC_{train})$$

کد

۳.۱ درخت تصمیم

(پیاده‌سازی این بخش بر پایه داکیومنتیشن کتابخانه Scikit-learn انجام شده است.)

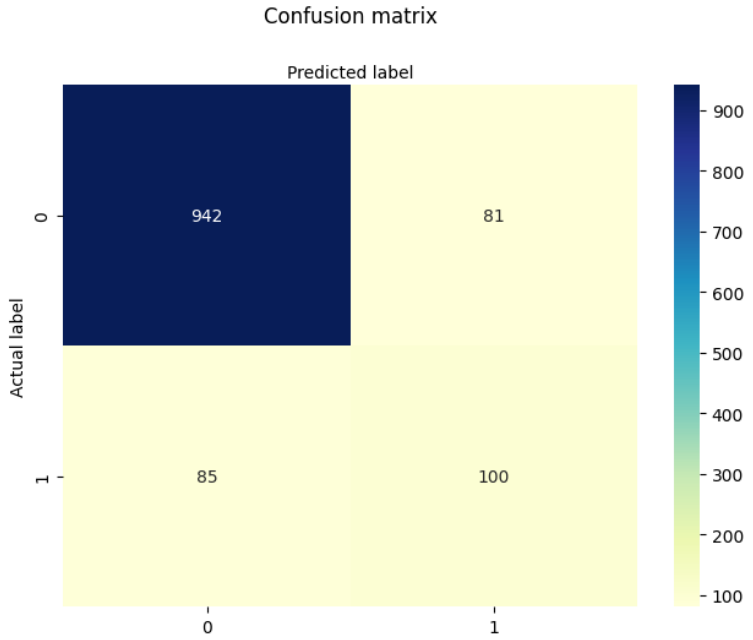
گام اول

در این گام به تحلیل کاوشگرانه بر روی داده‌ها می‌پردازیم.
با استفاده از کتابخانه Pandas مجموعه داده را می‌خوانیم. ابتدا با استفاده از دیتا فریم پانداز و تابع head() آن اطلاعات اولیه‌ای را با خواندن چند سطر ابتدایی در مورد داده کسب می‌کنیم.
با استفاده از info() در مورد نوع داده هر کدام از ویژگی‌ها اطلاعات کسب می‌کنیم.
در ادامه با استفاده از isnull().sum() در مورد ویژگی‌هایی که در یک نمونه مقدار ندارند اطلاعات کسب می‌کنیم. با توجه به خروجی این تابع یکی از ستون‌ها تمامی مقادیرش صفر است، پس این ستون را با استفاده از drop() حذف می‌کنیم. به جز این ستون همه ستون‌ها و دارای مقدار در تمامی سطرها هستند.
داده ما ممکن است شامل نمونه‌های تکراری باشد، با استفاده از duplicated() نمونه‌های تکراری را یافته و سپس حذف می‌کنیم.
در مجموعه داده‌ای که در اختیار داریم، بعضی از ویژگی‌ها به صورت توصیفی هستند و نه به صورت کمی. مدل فقط مقادیر کمی را متوجه می‌شود، برای همین به مقادیر توصیفی برچسب عددی می‌دهیم. این کار را با استفاده از preprocessing.LabelEncoder() انجام می‌دهیم. سپس با استفاده ازdescribe() توصیفی آماری از داده کسب می‌کنیم. با توجه به خروجی این تابع، نیاز به حذف داده پرت نداریم. در ادامه با استفاده از نمودار سعی می‌کنیم درک از داده را بالاتر ببریم. ماتریس کوواریانس داده ها را به صورت نموداری رسم می‌کنیم. در این پیاده سازی قصد نداریم که ابعاد را کاهش بدیم، اما اگر می‌خواستیم این کار را بکنیم، این نمودار بسیار کارآمد بود.
در نهایت به نرمال‌سازی این داده می‌پردازیم. با توجه به اینکه از تابع StandardScaler() برای نرمال‌سازی استفاده می‌کنیم، ابتدا مجموعه داده را به آموزشی و آزمایشی تقسیم می‌کنیم و سپس به سراغ نرمال سازی می‌رویم که از درز داده هنگام محاسبه میانگین جلوگیری کنیم

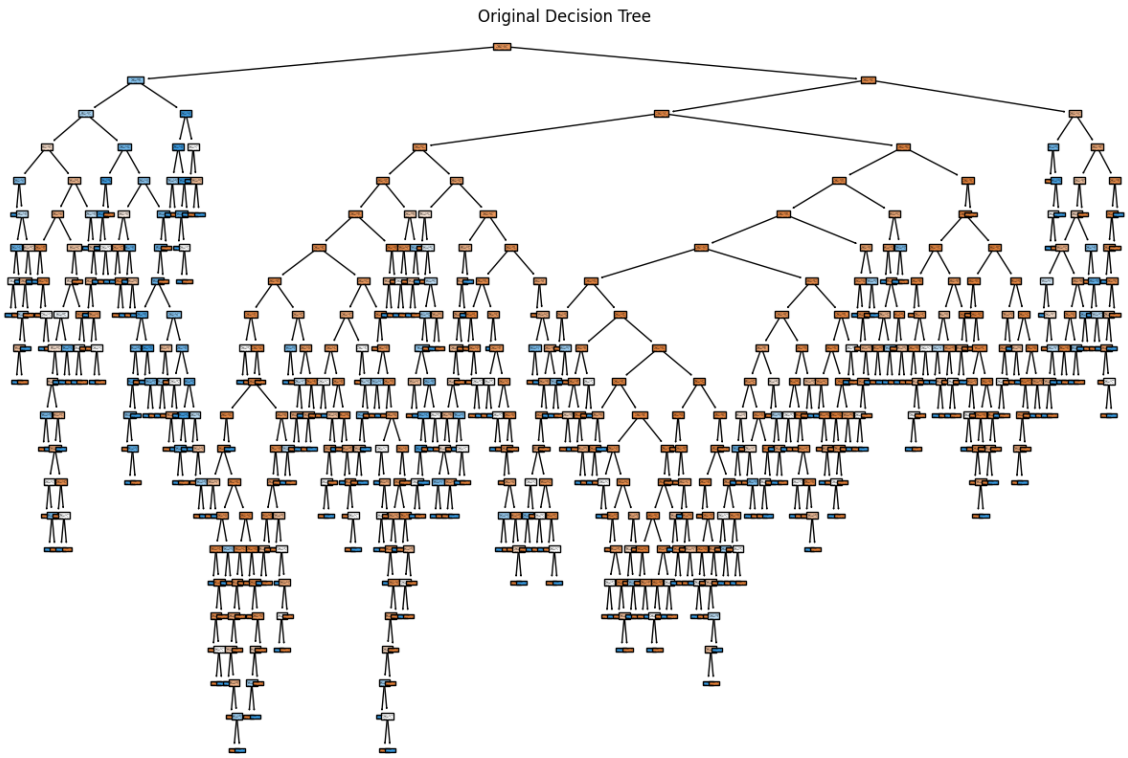
گام دوم

در این گام به پیاده‌سازی و تحلیل درخت تصمیم با استفاده از کتابخانه Scikit-Learn می‌پردازیم. از تابع DecisionTreeClassifier() در اینجا از تابع Gini استفاده می‌کنیم. هیچ محدودیتی بر روی تابع لحاظ نمی‌کنیم در مورد تعداد برگ‌ها یا ارتفاع درخت.

در ادامه با استفاده از معیارهای ارزیابی تعبیه شدن در Scikit-Learn مدل را ارزیابی می‌کنیم. عددی که معیار Accuracy به ما می‌دهد؛ مقدار ۸۶٪ است که نتیجه قابل قبولی به ما می‌دهد اما این معیار به تنهایی برای ارزیابی مدل کافی نیست، چرا که با توجه به نتایج مرحله قبل کلاس ۰ داده‌های بیشتری نسبت به کلاس ۱ ،تقریبا ۶ برابری، دارد و مدل را به سمت کلاس ۰ متمایل می‌کند. برای همین از معیارهای دیگر نیز استفاده می‌کنیم تا درک بهتری از وضعیت عملکرد مدل داشته باشیم. یکی از این معیارها Precision است که توجه به آن مدل برای تشخیص افرادی که فوت کرده‌اند به خوبی عمل نمی‌کند و نیمی از افرادی که فوت‌شده اعلام کرده در واقع زنده هستند. مقدار Precision، ۵۵٪ است. در ارزیابی با معیارRecall این را می‌بینیم که مدل به خوبی افرادی را که فوت کرده‌اند نسبت به تعداد کل افراد فوت شده به دست نیاورده. مقدار Recall برابر با ۵۴٪ است. در واقع موضع مدل به سمتی است که افراد را به اشتباه به گروه زنده‌ها نسبت می‌دهد این نزدیکی مقدارهای دو ارزیاب این را به ما می‌گوید که تعداد کمی از افراد زنده را به اشتباه فوت کرده نسبت داده‌اند. با توجه به ماتریس درهم ریختگی می‌توان دید که به داده‌های محدودی برچسب ۱ داده است. ماتریس درهم ریختگی آن برای داده آزمایشی به صورت زیر است:

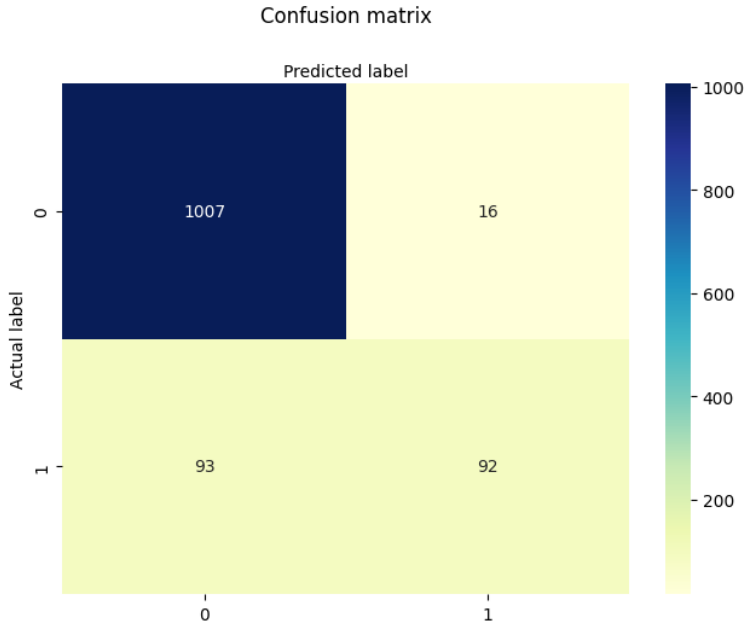


در ادامه درخت تصمیم را با استفاده از plot_tree رسم می‌کنیم.

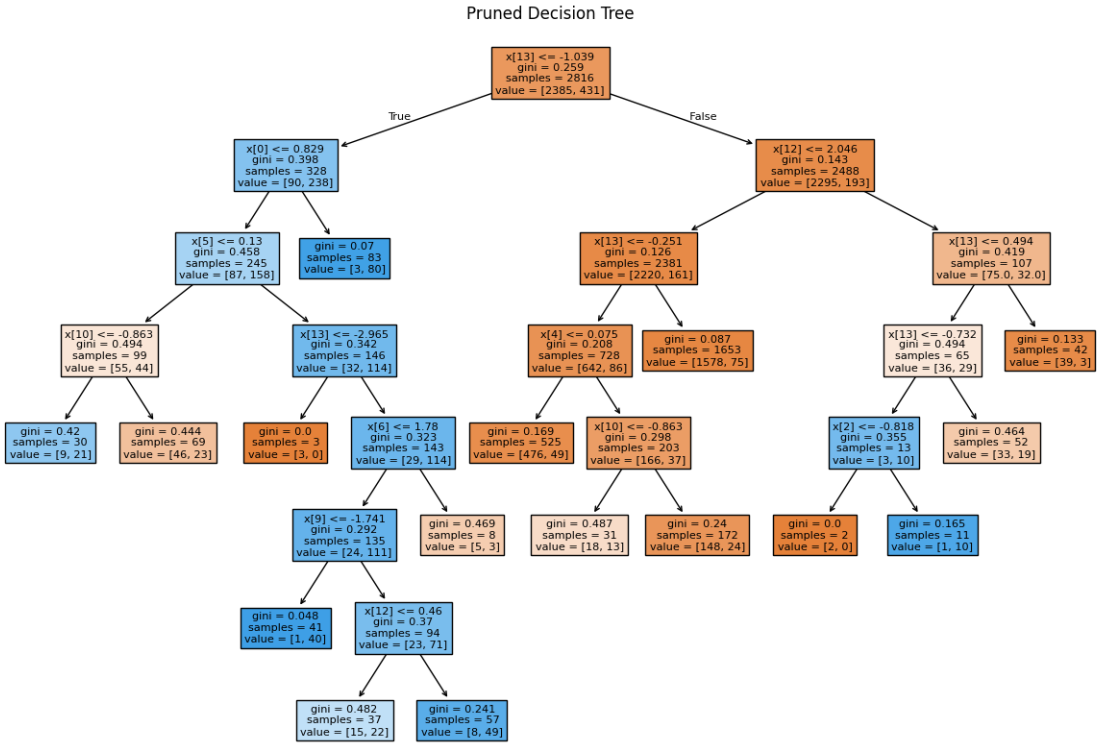


گام سوم

در این گام به پیاده‌سازی هرس پسین می‌پردازیم. در این روش یک زیر درخت را با یک برگ جایگزین می‌کنیم. از تابع cost_complexity_pruning_path استفاده می‌کنیم. با افزایش مقدارccp_alpha تعداد راس‌هایی که هرس می‌کنیم را افزایش می‌دهیم. سپس در میان تمام حالت‌های هرس، آن هرسی که بهترین دقت را به ما می‌دهد، را به عنوان هرس برمی‌گزینیم. دقت مدل بعد از هرس افزایش میابد. همچنین بقیه معیارها ارزیابی نیز بهبود میابند. این اتفاق به این دلیل است در درخت اصلی بیش برازش داریم؛ اما باید دقت کرد که هرچقدر تعداد رئوس هرس شده بیشتر باشد، مدل عملکرد بهتری نخواهد داشت و یک حالت بهینه‌ای وجود دارد که ما آن را انتخاب می‌کنیم. در اینجا مقدار Precision به شدت بهتر شده است و به عدد٪۸۵ رسیده است مه نشان می‌دهد مدل تعداد کمتری مقدار ۱ به نمونه‌ها نسبت داده و به صورت کلی با احتمال بهتری به نمونه‌ها برچسب ۱ داده و از حالت نسبتا رندوم بهتر شده است. اما این در مورد Recall صدق نمی‌کند. در اینجا هنوز با این مشکل رو به رو هستیم که مدل تمایل ندارد برچسب ۱ به نمونه ها بدهد. با نگاه کوتاهی به ماتریس درهم ریختگی می‌توان دید که مدل تمایل کمتری به ۱ دادن به نمونه دارد ولی وقتی اینکار را انجام می‌دهد، به خوبی نمونه را انتخاب می‌کند. ماتریس درهم ریختگی داده آموزشی بر روی مدل هرس شده به صورت زیر است:

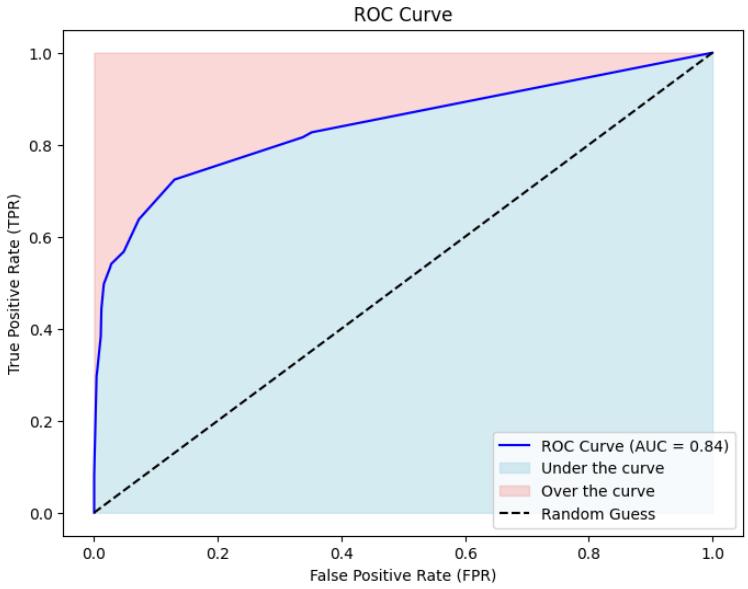


در ادامه درخت تصمیم را با استفاده از plot_tree رسم می‌کنیم که می‌توان دید عمق درخت به ۷ کاهش یافته که به شدت از پیچیدگی مدل می‌کاهد. این کاهش عمق درخت از آن‌جایی برای ما خوب است که مدل وابستگی کمتری به داده آموزشی دارد و در حالتی داده را از قبل ندیده بهتر عمل می‌کند. می‌توان دید که دقت مدل به ٪۹۱ بهبود یافته و اثرات بیش‌برازش را نداریم.



می‌دانیم هرچقدر مقدار TPR بیشتر و مقدار FPR کمتر باشد، به این معناست که مدل بهتر عمل می‌کند. مساحت زیر نمودار ROC نشان دهنده دقت آن است. این مقدار معادل با AUC است که هر چه

بیشتر باشد، عملکرد بهتری داریم. که با توجه به نمودار می‌توان دید مدل از یک انتخاب تصادفی بهتر عمل می‌کند و با توجه مقدار آن که تقریبا ۰.۸۴ است، و نزدیک به ۱ است می‌توان گفت که مدل به خوبی عمل می‌کند. .



گام چهارم

در اینجا بر روی داده رگرسیون لجستیک را با استفاده از تابع LogisticRegression() پیاده‌سازی می‌کنیم. این مدل به نسبت مدل هرس با معیار دقت بهتر شده است . در اینجا به رویه قبل به مقایسه دو معیار Precision و Recall می‌پردازیم. با توجه به جدولی که در ادامه آمده است می‌توان دید عملکرد بهتری نسبت به درخت هرس شده ارائه نمی‌دهد اما نسبت به درخت کامل توانسته آن‌هایی را که برچسب ۱ به آن‌ها داده به خوبی تشخیص دهد. اگر بخواهیم در مورد دقت مدل صحبت کنیم(Accuracy) می‌توان دید که مدل اکیدا از درخت کامل بهتر شده و مشابه درخت حرص شده عمل می‌کند.

LR	PDT	DT	
0.82	0.85	0.55	Precision
0.44	0.50	0.54	Recall

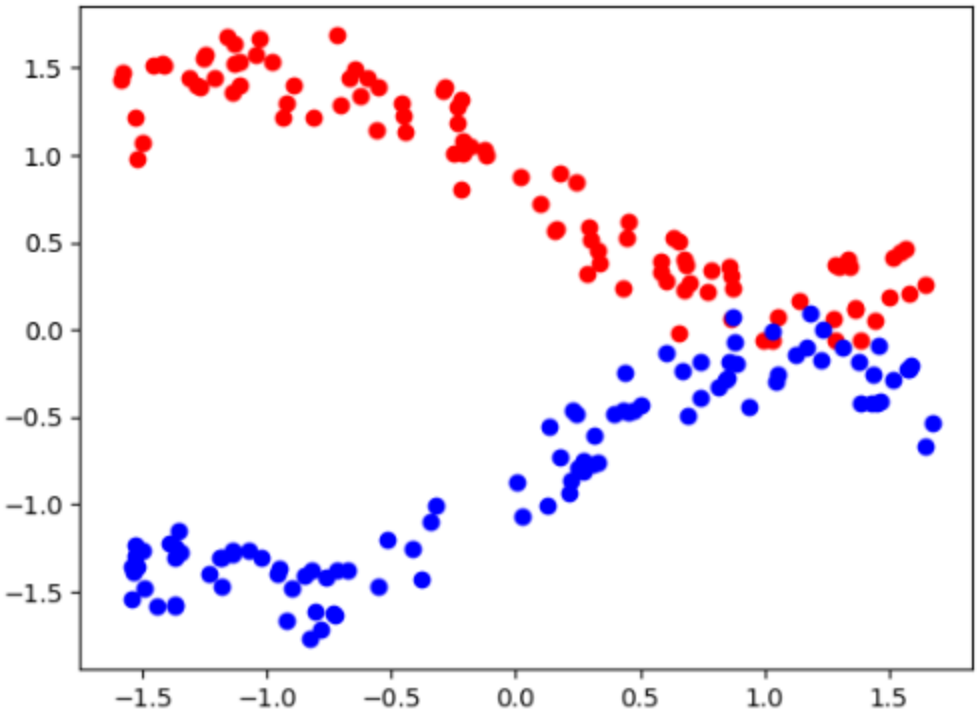
که بر اساس آن می‌توان گفت درخت تصمیم هرس شده بهتر از همه عمل می‌کند.

گام پنجم

در این گام از ما خواسته شده خروجی احتمالاتی هر دو مدل را به دست آورده و میانگین بگیریم. برای این‌کار از تابع predict_proba در هر دو مدل استفاده می‌کنیم. به عنوان خروجی به ما برای هر نمونه دو احتمال می‌دهد. این احتمالات را که نشانگر احتمال تعلق نمونه به هر کلاس است را میانگین می‌گیریم و مقدار بیشتر را به عنوان پیش‌بینی تحویل می‌دهیم. چیزی که از نتایج به نظر می‌رسد این روش از خروجی درخت تصمیم هرس شده بهتر نیست اما به سمت خروجی‌های آن تمرکز دارد نسبت به خروجی‌های رگرسیونی. به نظر می‌رسد با توجه به ناهماهنگی تعدد داده‌ها، میانگین‌گیری در جهت بهبود مدل عمل نکرده است و احتمالا باید برای کلاس‌ها وزن‌دهی کنیم تا عملکرد بهبود یابد.

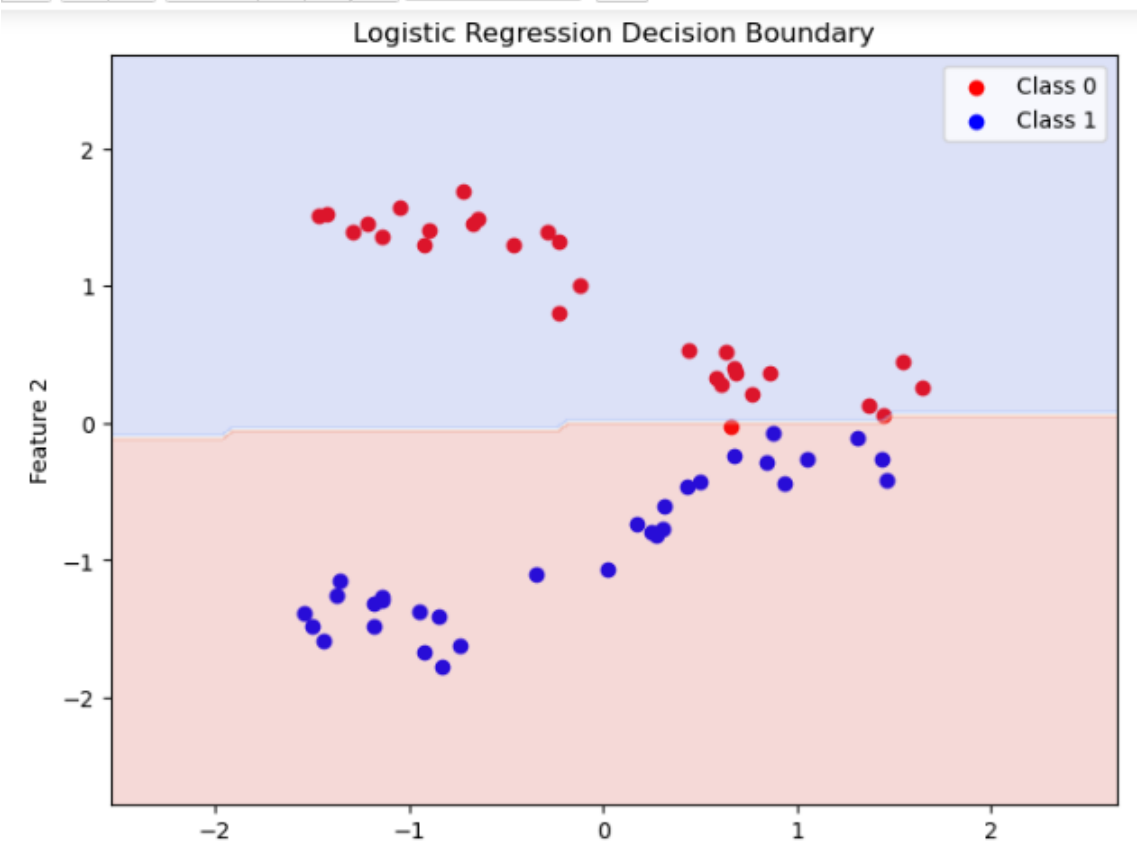
۲.۳ لجستیک رگرسیون

۰-۱ تحلیل مدل



نمودار شکل بالا نمایش دیتاست مدنظر با دیتای یونیفرم `uniform(low=-5,high=5,size=100)` برای مقادیر X و `uniform(low=-2,high=2,size=100)` برای مقادیر y هاست

وقتی مدل رگرسیون لجستیک رو با مقادیر دیفالت الفا ۰.۰۱ درجه ی ۱ ولامبدای ۰ و تعداد تکرار ۱۰۰۰ اجرا میکنیم به نمودار زیر میرسیم



که در این حالت مدل رگرسیون لجستیک به دقت ۹۸.۳۳ درصد روی داده های تست رسیده است . و به دقت ۹۵.۷۱ درصد روی داده های اموزشی رسیده است.

وقتی دقت تست و دقت آموزش نزدیک به هم باشند، به‌طور معمول نشان‌دهنده تعمیم خوب مدل است. یعنی مدل فقط الگوهای موجود در داده‌های آموزشی را یاد نگرفته بلکه می‌تواند به خوبی روی داده‌های جدید (تست) نیز پیش‌بینی‌های خوبی انجام بده. با اینحال ما دقت بالا داریم و دقت بالا در هر دو مجموعه به خودی خود نشان‌دهنده بیش‌برازش نیست. بیش‌برازش زمانی رخ می‌دهد که مدل روی داده‌های آموزشی دقت بالایی داشته باشد اما روی داده‌های تست عملکرد ضعیفی داشته باشد. در اینجا، تفاوت معناداری بین دقت تست و آموزش وجود ندارد، پس بیش‌برازش به احتمال زیاد رخ نداده است. در این حالت خاص، با توجه به نزدیک بودن دقت آموزش و تست و بالا بودن هر دو، نمی‌توان به سادگی نتیجه گرفت که بیش‌برازش رخ داده است. این وضعیت می‌تواند نشان‌دهنده عملکرد خوب مدل باشد، اما بررسی بیشتر با معیارهای دیگر و بررسی کلی داده‌ها و پیچیدگی مدل نیز می‌تواند به تحلیل دقیق‌تر کمک کند.

مرز تصمیم گیری ان به نظر مرز واضحی بین داد های دو کلاس ایجاد کرده است. این دقت بالا و مرز تصمیم گیری مناسب نشون میده این مدل برای این مجموعه دیتاست مناسب هست. همونطور که از رگرسیون لجستیک درجه ی یک انتظار میره مرز داده ها به شکل خطی هست. که به خوبی با داده ها همخونی داره و مرز مشخصی رو ایجاد کرده . دقت بالا روی داده هامون نشون میده که این مدل به خوبی روی داده های

تست عمل کرده و به خوبی تعمیم پذیر هست. همینطور دقت مدل برای داده های اموزشی نیز محاسبه شده و به مقدار ۹۸.۳۳ درصد با همان پارامترهای قبلی برای داده های تست شدیم که این اختلاف کم بین دقت مجموعه ازمایشی و تست نشان دهنده ی این هست که مدل دچار بیش برازش یا کم برازش نشده است.

در ادامه تنظیمات مدل رو عوض میکنیم تا ببینم به دقت های بهتری میرسیم یانه

همینطور توجه میکنید که از StandardScaler برای نرمال سازی داده استفاده شده است.

۲-۰ توضیح تابع features transform

ورودی‌ها: X داده‌های اصلی ویژگی‌ها و degree درجه چندجمله‌ای (پیش‌فرض ۲) هستند.

PolynomialFeatures بر اساس درجه مشخص‌شده با ترکیب ویژگی های موجود از طریق عملیات ضرب و جمع، ویژگی های جدیدی ایجاد می کند. این ویژگی ها به بهبود عملکرد مدل کمک میکنن خصوصا وقتی روابط غیرخطی بین ویژگی ها داشته باشیم. این می‌تواند هم اثرات اصلی (ویژگی‌های فردی ارتقا یافته به قدرت) و هم شرایط تعامل (محصولات جفت ویژگی) ایجاد کند.

only interaction: اگر True باشد، فقط عبارات تعامل را تولید می‌کند (پیش‌فرض=نادرست)

bias include: اگر True باشد، یک عبارت ثابت اضافه می‌کند (پیش‌فرض=True)

مزیتش اینه که ابعاد ویژگی هامون رو افزایش میده و روابط غیرخطی بین فیچر هارو هندل میکنه و باعث میشه مدل الگوهای پیچیده رو یاد بگیره

بدیشم اینه که اگه درجه ش بالا باشه میتونه باعث بیش برازش بشه همینطور هزینه ی محاسباتی و پیچیدگی رو افزایش میده همینطور ممکنه باعث اثر هم خطی چندگانه بشه.

راهکارش اینه از درجات پایین شروع بشه و به تدریج درجه رو افزایش بدیم و همزمان عملکرد مدل رو ببینیم تا از بیش برازش جلوگیری کنیم همینطور میتوان برای پیداکردن درجه ی بهینه از روش های cross validation استفاده کرد یا از تکنیک های منظم سازی مثل regression Ridge or Lasso استفاده کرد. پس در مجموع میشه گفت ابزار خوبی برای مهندسی فیچرهاست ولی باید محتاطانه ازش استفاده کرد.

۳-۰ three to degree change

وقتی درجه رو به ۳ افزایش میدیم با همون پارامترهای دیفالت کد قبلی که تو گام ۲ نمودار رو کشیدیم و دقت روبدست اوردیم حالا به دقت درصد ۱۰۰ داده های تست میرسیم این نشون دهنده ی اینه که مدل

بیش از حد تطبیق پیداکرده که یعنی بیش برازش رخ داده و مدل حتی نویزهارم به خوبی یاد گرفته برای حل این مشکل بهتره از درجه ی مناسبتری مثلا ۲ استفاده کرد که منجر به دقت هایی ارایه شده که بالاتر گفتیم که این اختلاف کم نشون دهنده ی این میتواند باشد که مدل تعمیم پذیر است و از احتمال بیش برازش جلوگیری کرده اما باید بیشتر بررسی کنیم

یا میتوان از روش های پیش پردازش داده مثل StandardScaler استفاده کرد که باعث نرمال سازی دیتا و بهبود نتایج میشوند استفاده کرد که استفاده کردیم همینطور میتوان برای کاهش دقت که میتواند نشان دهنده ی بیش برازش باشد از تکنیک های منظم سازی یا کراس ولیدیشن استفاده کرد.

به همین منظور مقدار لامبدا به اندازه ی کافی باید بزرگ باشه تا از بیش برازش جلوگیری کنه. از ۰ شروع میکنیم و به تدریج زیادش میکنیم تا ببینیم رفتار مدل رو .

همینطور مقدار پارامتر الفا یا همان لرنینگ ریت باید به اندازه ی کافی کوچیک باشه تا از بیش برازش جلوگیری بشه و چون درجه ی زیاد باعث بیش برازش میشه بنظر همون درجه ی ۲، ۳ مناسب هست.

در واقع با کاهش alpha می‌توانیم به مدل اجازه دهیم تا با سرعت کمتری یاد بگیرد و از نوسانات زیاد در تنظیم وزن‌ها جلوگیری کند. افزایش مقدار lambda کمک می‌کند که مدل به اندازه‌های وزنی بزرگ برای ویژگی‌های پیچیده (مانند ویژگی‌های چندجمله‌ای با درجه بالا) گرایش نداشته باشد و در نتیجه، از یادگیری بیش‌ازحد مدل جلوگیری شود. در انتها با تست الفا و بتا در بازه های مختلف و درجه ۲ به خروجی هایی با دقت های زیر روی مجموعه تست و اموزشی رسیدیم :

Accuracy Test	Accuracy Train	λ	α
۹۵.۰۰	۹۴.۲۹	۰.۰	۰.۰۱
۹۵.۰۰	۹۴.۲۹	۰.۱	۰.۰۱
۹۵.۰۰	۹۴.۲۹	۱.۰	۰.۰۱
۹۳.۳۳	۹۲.۸۶	۷.۰	۰.۰۱
۹۳.۳۳	۹۲.۸۶	۱۰.۰	۰.۰۱
۶۷.۹۱	۸۷.۸۶	۰.۰	۰.۰۰۱
۶۷.۹۱	۸۷.۸۶	۰.۱	۰.۰۰۱
۶۷.۹۱	۸۷.۸۶	۱.۰	۰.۰۰۱
۶۷.۹۱	۸۷.۸۶	۷.۰	۰.۰۰۱
۶۷.۹۱	۸۷.۸۶	۱۰.۰	۰.۰۰۱
۶۷.۹۱	۸۷.۸۶	۰.۰	۰.۰۰۰۱
۶۷.۹۱	۸۷.۸۶	۰.۱	۰.۰۰۰۱
۶۷.۹۱	۸۷.۸۶	۱.۰	۰.۰۰۰۱
۶۷.۹۱	۸۷.۸۶	۷.۰	۰.۰۰۰۱
۶۷.۹۱	۸۷.۸۶	۱۰.۰	۰.۰۰۰۱

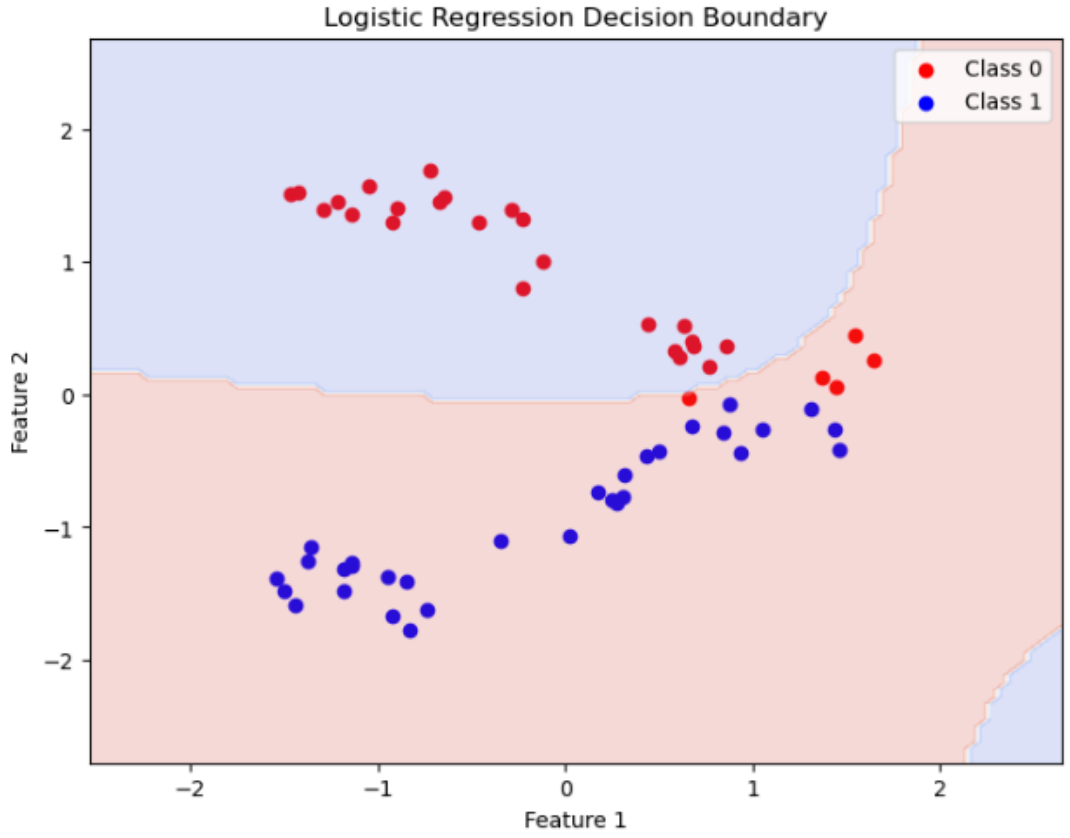
جدول ۱: Values lambda and alpha Different with Results Regression Logistic

به دنبال الفا و بتایی هستیم که دقت در داده های اموزشی و ازمایشی نزدیک به هم باشه مثلا۹۰ درصد

بدون اینکه اختلاف زیادی بین اونها وجودداشته باشه.

باتوجه به توضیحات بالا بنظر میرسد که بهترین تنظیمات برای جلوگیری از بیش‌برازش و تعمیم‌پذیری خوب مقادیر `alpha : 0.001` یا `alpha : 0.0001` باشد همینطور برای مقادیر لامبدا بنظر میرسد مقادیر `lambda : 1.0` یا `lambda : 7.0` مناسب هستند چون دقتی حدود ۹۵ درصد روی داده های ازمون و ۹۵ تا ۹۳ درصد روی داده های اموزشی میدهند که به احتمال زیاد از بیش برازش جلوگیری میکنند.

در ادامه تصویر نموداری که با مقادیر پارامتر بهینه ی لامبدا ۷ و الفا ۰.۰۰۱ و درجه ی ۲ رسم شده است اورده شده است. در تمامی مراحل تعداد تکرار همان مقدار دیفالت ۱۰۰۰ میباشد.



منحنی خط جداساز به خوبی نمایانگر تغییر درجه ای ست که داده ایم

۱ ۳.۳ مدل کالیبراسیون

توجه: نتایج این سوال و تمامی پروسه های ان وبا توجه به صورت سوال روی خروجی هایی که از درخت تصمیم تصمیم بدست امده(بدون هرس کردن) تمامی محاسبات انجام شده است.

تحلیل نتایج ارزیابی روی مجموعه ی ازمایشی: ما به این خروجی ها برای ارزیابی رسیدیم :

`precision: 0.80`

0.48 :recall

0.60 :f1score

0.90 :accuracy:

مقدار recall به این معناست که مدل توانسته ۴۸ درصد موارد مثبت واقعی را شناسایی کندمقدار بالایی نیست و نشون دهنده ی اینه که مدل تو شناسایی بعضی از موارد مثبت ضعف داردو احتمالا یه تعدادی از موارد مثبت رو به درستی تشخیص نداده است.

دقت مدل ۸۰ درصده به این معنی که که از بین مواردی که مدل به عنوان مثبت پیش‌بینی کرده، حدود ۸۰ درصد واقعاً مثبت بوده‌اند.

مقدار خوبی ست و نشون دهنده ی اینه که مدل تو پیش بینی های مثبت خود نسبتا مطمین عمل کرده.

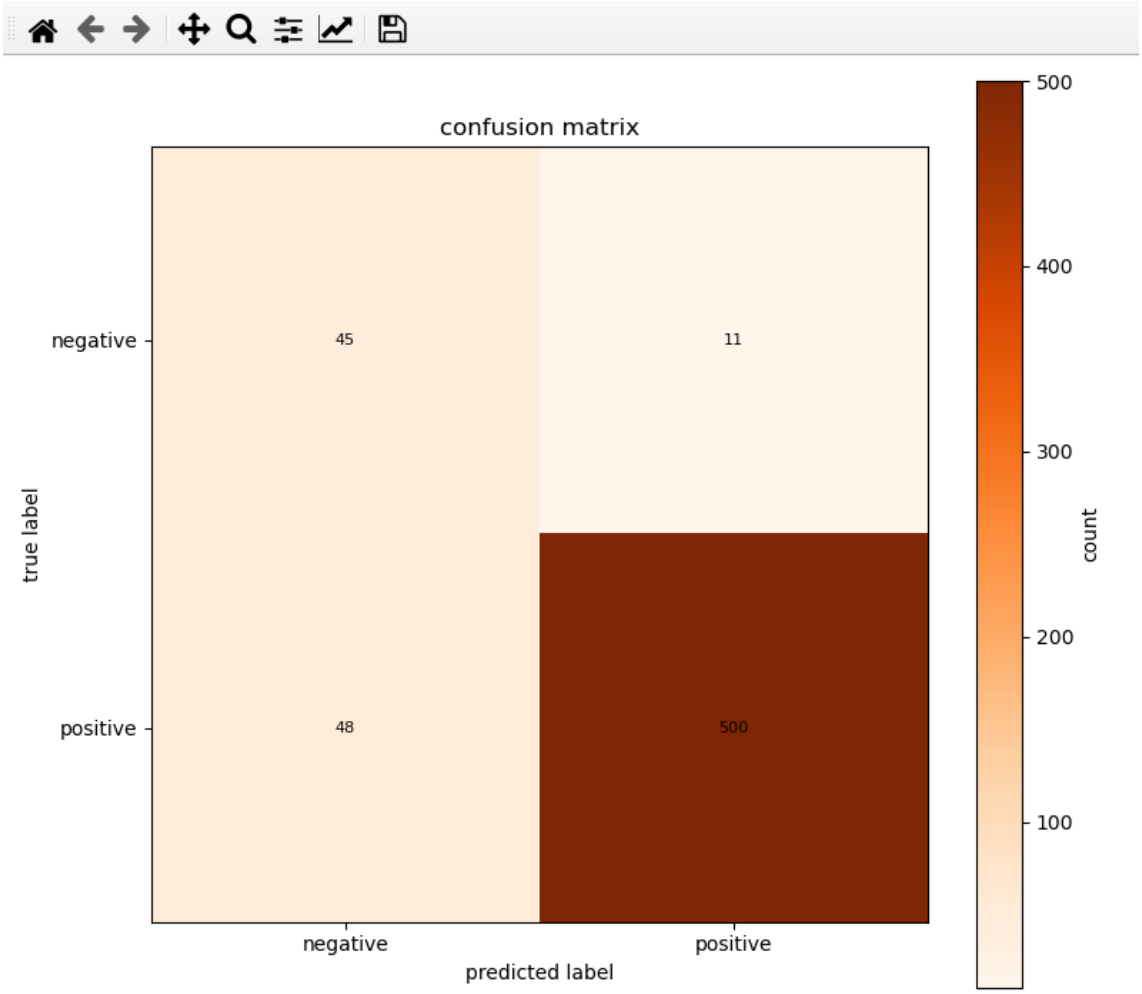
دقت مدل بالاست چون تعداد زیادی TP (500) و TN (45) داریم

امتیاز f1 score تقریبا ۶۰ درصده که به عنوان یک معیار متعادل بین دقت و فراخوانی عمل می‌کند. با توجه به این مقدار، مدل عملکرد متوسطی در تشخیص موارد مثبت دارد، چرا که به علت پایین بودن فراخوانی، F1 نیز کاهش یافته است

مقدار درستی ۹۰ درصد است که نشان می‌دهد مدل در بیش از ۹۰ درصد موارد پیش‌بینی‌ها را به درستی انجام داده است. این مقدار درستی بالا ممکن است به دلیل توانایی مدل در تشخیص صحیح موارد منفی باشد، اما باید توجه کرد که ممکن است هنوز برخی از موارد مثبت به درستی تشخیص داده نشده باشند.

مدل دقت بالایی دارد و اکثر موارد را به درستی پیش‌بینی کرده است، اما در شناسایی تمامی موارد مثبت ضعف دارد. در کاربردهایی که شناسایی دقیق موارد مثبت اهمیت بیشتری دارد، مانند تشخیص بیماری مثل این دیتاستمون، بهتر هست مدل بهبود یابد تا فراخوانی بیشتری داشته باشد

تحلیل ماتریس درهم ریختگی



Label True	Negative	Positive
Negative	45	11
Positive	48	500

- `Negative True` (TN): مقدار ۴۵ در خانه بالا سمت چپ، نشان‌دهنده تعداد نمونه‌های منفی است که به درستی پیش‌بینی شده‌اند.
- `Positive False` (FP): مقدار ۱۱ در خانه بالا سمت راست، نشان‌دهنده تعداد نمونه‌های منفی است که به اشتباه به عنوان مثبت پیش‌بینی شده‌اند.
- `Negative False` (FN): مقدار ۴۸ در خانه پایین سمت چپ، نشان‌دهنده تعداد نمونه‌های مثبتی است که به اشتباه به عنوان منفی پیش‌بینی شده‌اند.

- Positive True (TP): مقدار ۵۰۰ در خانه پایین سمت راست، نشان‌دهنده تعداد نمونه‌های مثبتی است که به درستی پیش‌بینی شده‌اند.

برای محاسبه ی ارزیابی یک کلاس نوشتیم که شامل هر کدوم از معیارهای خواسته شده صورت سوال نوشتیم که هرکدام مشخصا همونطور که از اسمشون مشخصه همون معیارها رو برمیگردونند.

کلاس ارزیابی ورودی‌های زیر را دریافت می‌کند:

prediction: برچسب‌های پیش‌بینی شده توسط مدل.

original: برچسب‌های واقعی.

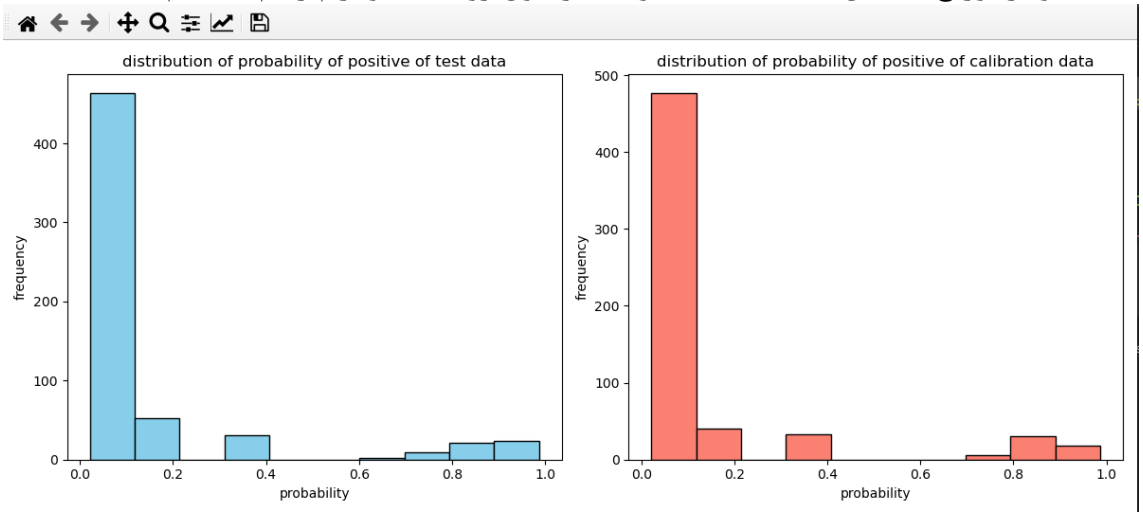
در متد confusion matrix اون ۴ مورد بالا که توضیح داده شد محاسبه میشه هر کدام از این موارد با بررسی مقدار صفر و یک بودن بررسی میشن و در نهایت با np.sum() مجموعشون محاسبه میشه.

متدهای این کلاس با در نظر گرفتن همون فرمول های سرکلاس وموجود در اسلایدهای کلاسی محاسبه شده اند. و بعدبا ساختن یک موجودیت از این کلاس هرکدام از متدهای مربوطه صدا زده میشوند و معیارها پرینت میشوند.

سپس در ادامه احتمالات پیش بینی در داده های تست با استفاده از متد اماده proba predict با استفاده از مدل DecisionTreeClassifier که قبلاً روی داده‌های آموزشی آموزش داده شده، احتمال تعلق نمونه‌های داده‌های تست test X را به هر کلاس (کلاس مثبت و منفی) پیش‌بینی می‌کنه

این احتمال‌ها به صورت یک آرایه دو بعدی برگردانده می‌شوند که در هر سطر احتمال‌های کلاس منفی و کلاس مثبت وجود دارد. سپس، با استفاده از data test of probas[:, 1] احتمال تعلق به کلاس مثبت جدا می‌شود و در متغیر data test of positive prob ذخیره می‌شود. درادامه مجددا همین کار را روی داده های کالیبراسیون انجام میدهیم.

در ادامه نمودار توزیع احتمال داده های تست و کالیبراسون رو باهیستوگرام رسم میکنیم.



تحلیل نمودار سمت چپ (داده‌های تست):

اکثر مقدارهای احتمال برای کلاس مثبت نزدیک به صفر هستند، که نشان می ده مدل برای بیشتر نمونه‌های داده‌ی تست، کلاس منفی را با احتمال بالا پیش‌بینی کرده و تعداد بسیار کمی از نمونه‌ها در داده‌های تست احتمال‌های بالاتری دارند (مثلاً در بازه‌های ۰.۶ تا ۱)، اما این نمونه‌ها بسیار محدودن. این نشان میده که مدل روی داده‌های تست عمدتا مطمئن هست که نمونه‌ها به کلاس منفی تعلق دارند. **نمودار سمت راست (داده‌های کالیبراسیون):**

مشابه نمودار تست، اکثر مقادیر احتمال برای کلاس مثبت نزدیک به صفر هستند. با این حال، توزیع احتمال‌ها در داده‌های کالیبراسیون کمی و بیشتر در حوالی مقادیر پایین‌تر از ۲.۰ است و تعداد کمی از نمونه‌ها در بازه‌های بالاتر قرار دارند. مدل برای داده‌های کالیبراسیون هم به طور عمده کلاس منفی را با اطمینان بالا پیش‌بینی کرده است.

گام سوم: این کد مرحله‌ی کالیبراسیون را با استفاده از رگرسیون لجستیک بر روی داده‌های احتمالاتی حاصل از مدل درخت تصمیم انجام میده چون مدل رگرسیون لجستیک با استفاده از تابع هزینه ی خود در پیش بینی های نادرست مدل باعث ایجاد جریمه میشود و این باعث میشه پیش بینی های مدل به واقعیت بیشتر نزدیک شه و این خصوصا توی موارد پزشکی مثل اینجا که پیش بینی های منفی اهمیت دارند بیشتر کاربرد داره.. این احتمال‌های کالیبره‌شده معمولاً به واقعیت نزدیک‌تر هستند

ابتدا یک مدل رگرسیون لجستیک رو به عنوان مدل کلیبراسیون تعریف میشه و سپس روی دیتای کالیبره شده ی مرحله ی قبل اموزش میبینه

متغیر `data calibration of positive prob` شامل احتمال تعلق به کلاس مثبت است که از مدل درخت تصمیم بر روی داده‌های کالیبراسیون استخراج شده است

متد `reshape` این احتمال‌ها را به فرمت مناسب برای ورودی‌های `fit` و `proba predict` تبدیل می‌کند

همونطور که گفتیم خروجی `proba predict` یک آرایه‌ی ۲ ستونی است که هر ستون شامل احتمال تعلق به یکی از کلاس‌هاست (ستون اول: احتمال کلاس منفی و ستون دوم: احتمال کلاس مثبت).

سپس در ادامه کلاس های مثبت ومنفی داده های تست داده های کالیبریت شده رو جدا میکنیم.

`calibration y`، هم که برچسب‌های واقعی داده‌های کالیبراسیون هست

گام چهارم و پنجم: این کد یه نمودار قابلیت اطمینان رسم میکنه که نشون بده که تا چه حد احتمال‌های پیش‌بینی‌شده مدل با احتمال‌های واقعی تطابق دارند. کد شامل تابع اصلی `reliability plot diagram` و یک تابع داخلی به نام `get reliability points` است که به طور دقیق‌تر نقاط مورد نیاز برای نمودار را محاسبه می‌کند. در ادامه بخش به بخش این کد توضیح داده شده

تابع اصلی `reliability plot diagram` به عنوان ورودی‌ها موارد زیر را دریافت می‌کند:

`true: y` لیستی از مقادیر واقعی (برچسب‌های داده‌های واقعی).

prob: y احتمال‌های پیش‌بینی‌شده توسط مدل بدون کالیبراسیون.

calibrated: prob y احتمال‌های پیش‌بینی‌شده توسط مدل بعد از کالیبراسیون.

bins: n تعداد دسته‌بندی‌ها (bins) برای تخمین نقاط روی نمودار

تابع داخلی points reliability get این تابع، نقاط مورد نیاز برای رسم نمودار قابلیت اطمینان را محاسبه می‌کند

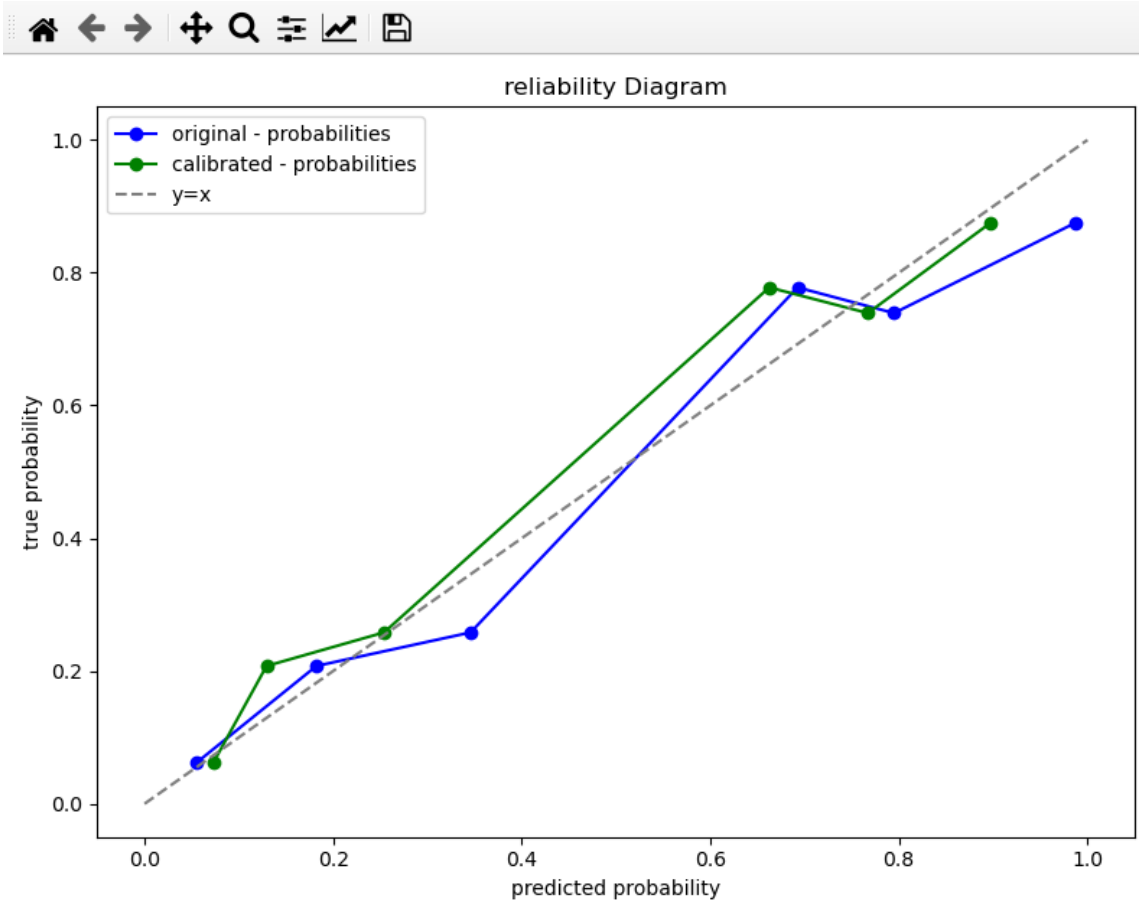
تو این قسمت اول اومدیم دسته های bins رو تعریف کردیم ینی دسته را بین مقادیر ۰ و ۱ به صورت خطی تقسیم کردیم و حدود دسته ها رو در اوردیم.

تو قسمت دوم اومدیم عرض دسته هارو بدست اوردیم این عرض رو برای محاسبه ی مرکز دسته نیاز داریم. بعدشم اومدیم مرکز دسته ها رو بدست اوردیم اینطوری که نصف عرض هر دسته به هر مرز پایینی میشه مرکز دسته مون بعدش اومدیم ایندکس هر مقدار تو دسته هارو بدست اوردیم تابع np.digitize مشخص می‌کند که هر مقدار در prob y تو کدام دسته قرار دارن. ۱- اضافه شده است تا ایندکس‌ها با صفر شروع شوند (بخاطر اینکه تو پایتون اندیس‌ها از ۰ شروع میشن).

بعدش اومدیم میانگین مقدار واقعی و احتمال پیش بینی شده ی هر دسته رو بدست اوردیم و اگه دسته خالی بود با مقدار خالی پرش کردیم . در ادامه اومدیم دسته های خالی رو حذف کردیم. و مقادیر ولید رو برگردوندیم. توضیح تابع diagram reliability plot در ابتدا اومدیم تابع reliability get points رو برای احتمال های اصلی و کالیبره شده صدا زدیم هر کدوم رو با یه رنگ متمایز صدا زدیم و بعد خط ایده ال y=x رو رسم کردیم و در انتها تنظیمات نهایی نموداراعم از لیبل زدن رو محورهاوعنوان و راهنمای نمودار رو صدا زدیم

گام ششم:

تفسیر نمودار



همونطور که مشخصه نمودار قابلیت اطمینان رو برای احتمالات اصلی و احتمالات کالیبره شده رسم کردیم

احتمالات اصلی با خط ابی و احتمالات کالیبره شده با خط سبز در شکل مشخص شده اند. محور افقی احتمالات پیش بینی شده و محور عمودی احتمالات واقعی هستند.

در این نمودار در نمودار قابلیت اطمینان، خط نقطه چین خاکستری رنگ y=x به عنوان خط ایده‌آل عمل می‌کند و نشان می‌دهد که مدل در حالتی کاملاً کالیبره‌شده و دقیق است. هر نقطه‌ای که روی این خط قرار گیرد، به این معنی است که احتمال پیش‌بینی‌شده مدل دقیقاً با احتمال واقعی وقوع آن رخداد مطابقت داره هر نقطه ای که روی این خط قرار بگیره قطه‌ای روی این خط قرار بگیرد، نشان‌دهنده این هست که مدل احتمال واقعی را به درستی تخمین زده.

احتمالات اصلی (آبی): نقاط آبی که به خط خاکستری y=x نزدیک‌تر هستند، نشان‌دهنده‌ی پیش‌بینی‌هایی هستند که مطابقت بیشتری با احتمال واقعی دارن. نقاطی که از خط فاصله دارند نسون میدن که احتمال پیش‌بینی شده با احتمال واقعی مغایرت داره.

احتمالات کالیبره‌شده (سبز): نقاط سبز نشان‌دهنده احتمالاتی هستند که پس از کالیبراسیون، به دست

آمده‌اند. در اینجا مشاهده می‌شود که این نقاط نزدیک‌تر به خط خاکستری y=x قرار دارند، که نشان‌دهنده بهبود دقت در تخمین احتمال‌ها پس از کالیبراسیون است. هر چی این نقاط به این خط خاکستری نزدیک تر باشن ینی به واقعیت نزدیک و قابل اعتماد ترن. حالا این فاصله ها دو جورن بعضیاشون زیر خط خاکستری قرار گرفتن و بعضیاشون بالاش و اینا دو معنی متفاوت دارن ینی اگر نقاط بالاتر از خط باشند، مدل تو حالتی خوش‌بینانه قرار داره و احتمال‌ها را بیشتر از واقعیت تخمین میزنه. اگر نقاط پایین‌تر از خط باشن، مدل بدبینانه است و احتمال‌ها را کمتر از واقعیت تخمین زده.

تو مدل اصلی بدون کالیبراسیون مدل تمایل داره تو بعضی قسمت ها احتمال رو کمتر از واقعیت و توبعضی بیشتر از واقعیت نشون بده ینی نقاط از خط ایده ال دور تر هستند.

تو مدل کالیبریت شده مدل بعد از کالیبراسیون تونسته پیش بینی هاشو به واقعیت نزدیکتر کنه و این نشون میده مدل تو کالیبراسیون تخمین دقیق تری تو احتمالات نسبت به وقتی که کالیبراسیون اعمال نکرده بودیم داره. خلاصه نتیجه اینکه کالیبراسیون مدل منجر به بهبود قابلیت اطمینان پیش‌بینی‌ها شده ینی مدل کالیبره‌شده به احتمال بیشتری احتمال‌های واقعی را به درستی تخمین می‌زند و می‌توان به خروجی آن اعتماد بیشتری داشت