



دانشگاه صنعتی اصفهان

دانشکده علوم ریاضی

پروژه کارشناسی آمار

مطالعه‌ای در مورد روشهای طبقه بندی داده ها

استاد راهنما : دکتر کلکین نما

پژوهشگر : یکتا شمس

۱۴۰۵

فهرست مطالب

فصل ۱: مقدمه.....	۴
• تفاوت رگرسیون و طبقه‌بندی.....	۶
• چرا نمی‌شود از رگرسیون خطی برای طبقه‌بندی استفاده کرد؟ (بخش ۴.۲ کتاب).....	۷
فصل ۲: رگرسیون لجستیک.....	۹
• مدل و فرمول ریاضی.....	۱۲
• تخمین ضرایب (Maximum Likelihood).....	۱۴
• تفسیر ضرایب و odds ratio.....	۱۶
• رگرسیون لجستیک چندکلاسه (Multinomial).....	۱۷
• پدیده درهم‌تنیدگی (Confounding).....	۱۸
• رگرسیون لجستیک برای بیش از دو کلاس پاسخ.....	۱۹
فصل ۳: نرخ خطا در طبقه بندی و طبقه بندی کننده بیز.....	۲۰
• طبقه کننده بیز.....	۲۰
• روش K نزدیک ترین همسایه.....	۲۲
• اثر انتخاب K روی عملکرد KNN.....	۲۳
فصل ۴: تحلیل تشخیصی خطی (LDA).....	۲۵
• استفاده از قضیه بیز برای طبقه‌بندی.....	۲۵
• تحلیل تشخیصی خطی برای حالت $p=1$	۲۷
• تحلیل تشخیصی خطی برای حالت $p>1$	۳۰
• مرزهای تصمیم‌گیری بیز.....	۳۱
• ماتریس درهم‌ریختگی (Confusion Matrix).....	۳۲
• آستانه تصمیم‌گیری و مبادله بین خطاها.....	۳۳
• منحنی ROC و سطح زیر منحنی (AUC).....	۳۵
• تحلیل تشخیصی درجه دو (QDA).....	۳۷
فصل ۵: مقایسه روش‌های طبقه‌بندی.....	۴۰
• ارتباط بین رگرسیون لجستیک و LDA.....	۴۰
• شش سناریوی مقایسه (خطی و غیرخطی).....	۴۲
• چه زمانی کدام روش بهتر عمل می‌کند؟.....	۴۴
• تعمیم مدل‌ها با استفاده از جمله‌های غیرخطی.....	۴۴

فصل ۶: آزمایشگاه رگرسیون لجستیک، LDA، QDA و KNN	۴۶
• داده‌های بازار سهام.....	۴۶
• رگرسیون لجستیک.....	۵۲
• تحلیل تشخیصی خطی.....	۵۸
• تحلیل تشخیصی درجه دو.....	۶۲
• K نزدیک‌ترین همسایه.....	۶۴
• کاربرد در داده‌های بیمه کاراوان.....	۶۶

مقدمه (Classification)

۱.۱ طبقه‌بندی داده‌ها چیست؟

در بسیاری از مسائل آماری و یادگیری ماشین، هدف پیش‌بینی یک متغیر پاسخ بر اساس یک یا چند متغیر توضیحی است. در حالتی که متغیر پاسخ کمی (Quantitative) باشد، از روش‌های رگرسیونی مانند رگرسیون خطی استفاده می‌شود. اما در بسیاری از موقعیت‌های واقعی، متغیر پاسخ کیفی (Qualitative) یا دسته‌ای (Categorical) است؛ یعنی مقادیر آن به‌جای اعداد پیوسته، برچسب‌ها یا دسته‌های مشخصی هستند. برای مثال:

- تشخیص اینکه یک ایمیل، اسپم است یا نه
 - پیش‌بینی اینکه یک بیمار به بیماری خاصی مبتلا است یا خیر
 - تشخیص نکول (Default) یا عدم نکول یک وام‌گیرنده
- فرآیند پیش‌بینی یک پاسخ کیفی را **طبقه‌بندی** یا **دسته‌بندی** (Classification) می‌نامند و روشی که برای این کار به‌کار می‌رود را **طبقه‌بندی‌کننده** (Classifier) می‌گویند.

۱.۲ چرا نمی‌توان از رگرسیون خطی برای طبقه‌بندی استفاده کرد؟

در نگاه اول ممکن است تصور شود که می‌توان دسته‌های یک متغیر کیفی را به اعداد تبدیل کرد (مثلاً ۰ و ۱) و از رگرسیون خطی معمولی استفاده کرد. اما این رویکرد با دو مشکل اساسی روبه‌رو است:

۱. وقتی متغیر پاسخ بیش از دو دسته دارد، کدگذاری اعداد به دسته‌ها یک فاصله و ترتیب کاذب بین دسته‌ها ایجاد می‌کند که در واقعیت وجود ندارد.

۲. حتی در حالت دو دسته‌ای (۰ و ۱)، مقادیر پیش‌بینی‌شده توسط رگرسیون خطی می‌توانند خارج از بازه [۰, ۱] باشند، در حالی که این مقادیر باید نشان‌دهنده احتمال باشند و نمی‌توانند منفی یا بزرگتر از یک شوند.

به همین دلیل، نیاز به روش‌های اختصاصی‌تری برای طبقه‌بندی داریم که این محدودیت‌ها را ندارند.

۱.۳ روش‌های مورد بررسی در این پروژه

در این پروژه، سه روش پرکاربرد طبقه‌بندی مورد بررسی و مقایسه قرار می‌گیرند:

- **رگرسیون لجستیک (Logistic Regression):** روشی که احتمال تعلق یک مشاهده به هر دسته را با استفاده از تابع لجستیک مدل می‌کند.
- **تحلیل تشخیصی خطی (Linear Discriminant Analysis - LDA):** روشی مبتنی بر قضیه بیز که با فرض توزیع نرمال برای متغیرهای توضیحی، مرز تصمیم‌گیری خطی بین دسته‌ها ایجاد می‌کند.
- **روش K نزدیکترین همسایه (K-Nearest Neighbors - KNN):** روشی ناپارامتری که بر پایه شباهت مشاهدات به یکدیگر عمل می‌کند.

۱.۴ ساختار پروژه

در فصل دوم، به بررسی دقیق مدل رگرسیون لجستیک پرداخته می‌شود. فصل سوم به تحلیل تشخیصی خطی و درجه دو اختصاص دارد. در فصل چهارم، این روش‌ها از نظر عملکرد با یکدیگر مقایسه می‌شوند و در فصل پنجم، پیاده‌سازی عملی این روش‌ها روی یک مجموعه داده واقعی ارائه می‌گردد.

تفاوت رگرسیون و طبقه‌بندی

مسائل طبقه‌بندی در دنیای واقعی به همان اندازه مسائل رگرسیون، بلکه شاید حتی بیشتر، رایج هستند. برای مثال:

- تشخیص اینکه بیمار مراجعه‌کننده به اورژانس با مجموعه‌ای از علائم مشخص، به کدامیک از چند بیماری احتمالی مبتلاست
 - تشخیص متقلبانه بودن یا نبودن یک تراکنش بانکی بر اساس آدرس IP و سابقه تراکنش‌های کاربر
 - تعیین اینکه کدام جهش‌های DNA بیماری‌زا هستند و کدامیک نیستند
- نکته مشترک در تمام این مسائل، همانند رگرسیون، وجود مجموعه‌ای از داده‌های آموزشی به صورت جفت $(x_1, y_1), \dots, (x_n, y_n)$ است که مدل بر اساس آن‌ها ساخته می‌شود. تفاوت اصلی رگرسیون و طبقه‌بندی در ماهیت متغیر پاسخ Y نهفته است: در رگرسیون، Y یک متغیر کمی (عددی و پیوسته) است، در حالی‌که در طبقه‌بندی، Y یک متغیر کیفی است که مقادیر آن به جای عدد، برچسب یا دسته هستند.

به عنوان مثال، در مجموعه داده Default (که در این کتاب و پروژه به عنوان مرجع استفاده می‌شود)، هدف پیش‌بینی این است که آیا فردی بر اساس درآمد سالانه (income) و مانده حساب ماهانه کارت اعتباری (balance) در پرداخت بدهی‌اش نکول می‌کند یا نه. در این مسئله، متغیر پاسخ default تنها دو مقدار ممکن (نکول یا عدم نکول) دارد و بنابراین کیفی است. به همین دلیل، مدل رگرسیون خطی ساده که برای متغیرهای پاسخ کمی طراحی شده، برای این نوع مسائل مناسب نیست و نیاز به روش‌های اختصاصی طبقه‌بندی مانند رگرسیون لجستیک، LDA و KNN داریم.

همچنین، درست مانند رگرسیون، هدف نهایی در طبقه‌بندی نیز صرفاً عملکرد خوب روی داده‌های آموزشی نیست، بلکه مدل باید بتواند روی مشاهدات آزمون (Test Observations)

که در فرآیند آموزش استفاده نشده‌اند نیز به‌درستی عمل کند؛ این ویژگی معیار اصلی ارزیابی کیفیت هر طبقه‌بندی‌کننده محسوب می‌شود.

چرا نمی‌شود از رگرسیون خطی برای طبقه‌بندی استفاده کرد؟

فرض کنید می‌خواهیم وضعیت پزشکی یک بیمار را بر اساس علائم در اورژانس پیش‌بینی کنیم. در این مثال ساده‌شده، سه تشخیص ممکن وجود دارد: سکته مغزی (stroke)، مصرف بیش‌ازحد دارو (drug overdose)، و تشنج صرعی (epileptic seizure). می‌توانیم این مقادیر را به‌صورت یک متغیر پاسخ کمی، Y ، کدگذاری کنیم:

if $Y = 1$: stroke

if $Y = 2$: drug overdose

if $Y = 3$: epileptic seizure

با استفاده از این کدگذاری، می‌توان از روش کمترین مربعات برای برازش یک مدل رگرسیون خطی جهت پیش‌بینی Y بر اساس مجموعه‌ای از متغیرهای پیش‌بین $X_1, \dots, X_p$ استفاده کرد. اما متأسفانه، این کدگذاری یک ترتیب خاص روی خروجی‌ها تحمیل می‌کند؛ به این ترتیب که drug overdose را بین stroke و epileptic seizure قرار می‌دهد و اصرار دارد که تفاوت بین stroke و drug overdose با تفاوت بین drug overdose و epileptic seizure برابر باشد. در عمل هیچ دلیل خاصی وجود ندارد که این‌طور باشد. برای مثال، می‌توان کدگذاری دیگری به همان اندازه معقول انتخاب کرد:

if $Y = 1$: epileptic seizure

if $Y = 2$:stroke

if $Y = 3$: drug overdose

که این کدگذاری، رابطه‌ای کاملاً متفاوت بین این سه بیماری را نتیجه می‌دهد. هر یک از این کدگذاری‌ها مدل‌های خطی اساساً متفاوتی تولید می‌کنند که در نهایت به مجموعه‌های متفاوتی از پیش‌بینی‌ها روی مشاهدات آزمون منجر می‌شوند.

اگر مقادیر متغیر پاسخ دارای یک ترتیب طبیعی بودند، مانند خفیف (mild)، متوسط (moderate) و شدید (severe)، و اگر احساس می‌کردیم که فاصله بین خفیف و متوسط مشابه فاصله بین متوسط و شدید است، آنگاه کدگذاری ۱، ۲، ۳ معقول می‌بود. اما متأسفانه، به‌طور کلی هیچ راه طبیعی‌ای برای تبدیل یک متغیر پاسخ کیفی با بیش از دو سطح به یک پاسخ کمی مناسب برای رگرسیون خطی وجود ندارد.

برای یک پاسخ کیفی دوتایی (binary، دو سطحی)، وضعیت بهتر است. برای مثال، فرض کنید تنها دو حالت ممکن برای وضعیت پزشکی بیمار وجود دارد: stroke و drug overdose. در این حالت می‌توانیم از رویکرد متغیر مجازی (dummy variable) استفاده کنیم و پاسخ را به این صورت کدگذاری کنیم:

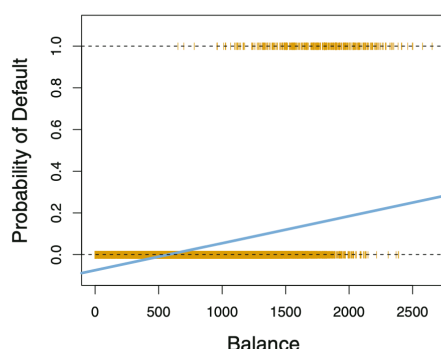
if $Y = 0$: stroke

if $Y = 1$: drug overdose

سپس می‌توانیم یک رگرسیون خطی روی این پاسخ دوتایی برازش دهیم و اگر $\hat{Y} > 0.5$ بود drug overdose و در غیر این صورت stroke را پیش‌بینی کنیم. در حالت دوتایی، به‌سادگی می‌توان نشان داد که حتی اگر کدگذاری بالا را برعکس کنیم، رگرسیون خطی همان پیش‌بینی‌های نهایی را تولید می‌کند.

برای یک پاسخ دوتایی با کدگذاری ۰ و ۱ مانند بالا، رگرسیون با روش کمترین مربعات منطقی است؛ می‌توان نشان داد که $\hat{XB}$ به‌دست‌آمده از رگرسیون خطی، در واقع برآوردی از

$\Pr(\text{drug overdose} | X)$ در این حالت خاص است. با این حال، اگر از رگرسیون خطی استفاده کنیم، ممکن است برخی از برآوردها خارج از بازه $[0, 1]$ باشند، که تفسیر آن‌ها را به عنوان احتمال دشوار می‌کند! با این حال، پیش‌بینی‌ها یک ترتیب ارائه می‌دهند و می‌توان آن‌ها را به عنوان برآوردهای خام احتمال تفسیر کرد. جالب اینکه، طبقه‌بندی‌هایی که از استفاده رگرسیون خطی برای پیش‌بینی یک پاسخ دوتایی به دست می‌آیند، با طبقه‌بندی‌های حاصل از روش تحلیل تشخیصی خطی (LDA) که در بخش ۴.۴ بحث می‌کنیم، یکسان خواهند بود.



این نمودار نشون می‌دهد که وقتی از رگرسیون خطی برای پیش‌بینی یک متغیر پاسخ دوتایی (که فقط دو مقدار ۰ و ۱ دارد) استفاده می‌کنیم، خط برازش شده می‌تواند در برخی نواحی مقداری خارج از بازه منطقی $[0, 1]$ تولید کند؛ یعنی احتمال‌های منفی یا بزرگتر از یک به دست آید که از نظر آماری قابل تفسیر نیستند.

با این حال، رویکرد متغیر مجازی را نمی‌توان به راحتی برای پاسخ‌های کیفی با بیش از دو سطح تعمیم داد. به همین دلایل، ترجیح داده می‌شود از یک روش طبقه‌بندی استفاده شود که واقعاً برای مقادیر پاسخ کیفی مناسب باشد، مانند روش‌هایی که در ادامه معرفی می‌شوند.

فصل ۲

رگرسیون لجستیک

استفاده از رگرسیون خطی برای پیش‌بینی یک پاسخ کیفی، مشکلاتی جدی به همراه دارد؛ ازجمله اینکه مقادیر پیش‌بینی‌شده می‌توانند خارج از بازه منطقی [۰, ۱] قرار بگیرند و به‌همین دلیل قابل تفسیر به‌عنوان احتمال نباشند. برای حل این مشکل، به‌جای مدل‌سازی مستقیم متغیر پاسخ Y ، رگرسیون لجستیک **احتمال** تعلق Y به یک دسته مشخص را مدل می‌کند.

برای مثال، فرض کنید می‌خواهیم بر اساس تعداد کلمات مشکوک موجود در یک ایمیل، احتمال اسپم‌بودن آن ایمیل را پیش‌بینی کنیم. در این حالت، به‌جای پیش‌بینی مستقیم اینکه ایمیل اسپم است یا نه، مدل رگرسیون لجستیک احتمال اسپم‌بودن را به شرط تعداد کلمات مشکوک برآورد می‌کند:

$$\Pr(\text{spam} = \text{Yes} \mid \text{suspicious_words})$$

این مقدار را می‌توان به‌اختصار $p(x)$ نامید که همواره بین ۰ و ۱ قرار دارد. سپس بر اساس این احتمال، می‌توان یک آستانه تعیین کرد؛ برای مثال، می‌توان $\text{spam} = \text{Yes}$ را برای هر ایمیلی که $p(x)$ بیشتر از ۰.۵ باشد پیش‌بینی کرد. اگر یک سرویس ایمیل بخواهد در تشخیص اسپم محافظه‌کارانه‌تر عمل کند (یعنی ایمیل‌های مهم را به‌اشتباه اسپم تشخیص ندهد)، می‌تواند آستانه بالاتری مانند ۰.۷ را انتخاب کند.

نکته مهم اینجا است که برخلاف رگرسیون خطی که مقدار پیش‌بینی‌شده می‌تواند منفی یا بزرگتر از یک باشد، تابعی که در رگرسیون لجستیک استفاده می‌شود (تابع لجستیک یا سیگموئید) به‌گونه‌ای طراحی شده که خروجی آن همواره در بازه [۰, ۱] باقی می‌ماند و بنابراین همیشه قابل تفسیر به‌عنوان یک احتمال معتبر است. این ویژگی، رگرسیون لجستیک

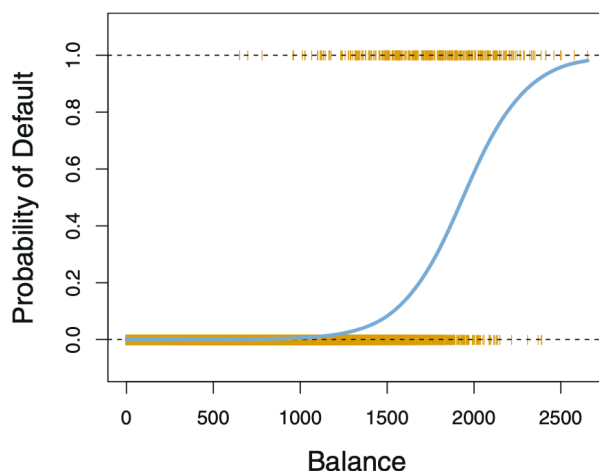
را به یکی از پرکاربردترین و مناسبترین روش‌ها برای مسائل طبقه‌بندی دوتایی (Binary Classification) تبدیل کرده است.

یکی از مزیت‌های اصلی رگرسیون لجستیک نسبت به روش‌های ساده‌تر طبقه‌بندی، این است که به جای اینکه صرفاً یک برچسب قطعی (مثلاً «اسپم» یا «غیراسپم») به هر مشاهده بدهد، یک عدد پیوسته بین ۰ و ۱ ارائه می‌دهد که میزان اطمینان مدل را نشان می‌دهد. این ویژگی چند مزیت مهم دارد:

نخست، امکان تنظیم آستانه تصمیم‌گیری را بر اساس نیاز کاربر فراهم می‌کند. برای مثال، در تشخیص اسپم، ممکن است ترجیح بدهیم آستانه را بالاتر از ۰.۵ تنظیم کنیم تا احتمال اینکه یک ایمیل مهم به اشتباه اسپم شناسایی شود کاهش یابد؛ حتی اگر این کار باعث شود برخی ایمیل‌های اسپم واقعی نیز از دست بروند.

دوم، خروجی احتمالی این امکان را می‌دهد که مشاهدات را بر اساس میزان اطمینان مدل رتبه‌بندی کنیم. برای مثال، ایمیلی با احتمال ۰.۹۸ برای اسپم‌بودن، قطعیت بسیار بیشتری نسبت به ایمیلی با احتمال ۰.۵۲ دارد، حتی اگر هر دو با آستانه ۰.۵ در یک دسته (اسپم) طبقه‌بندی شوند.

علاوه بر این، چون رابطه واقعی بین متغیر پیش‌بین و احتمال وقوع رویداد معمولاً به شکل غیرخطی و S مانند است، یک خط راست نمی‌تواند این رابطه رو به درستی مدل کند؛ در نواحی ابتدایی و انتهایی داده‌ها، خط خطی به طور قابل توجهی از رفتار واقعی داده‌ها فاصله می‌گیرد.



این نمودار نشون می‌دهد که وقتی از یک تابع لجستیک (به‌جای خط راست) برای مدل‌سازی احتمال یک پاسخ دوتایی استفاده می‌کنیم، منحنی حاصل همیشه بین ۰ و ۱ باقی می‌ماند و هیچ‌وقت از این بازه خارج نمی‌شود.

شکل این منحنی به‌صورت S یا سیگموئید است؛ یعنی در دو انتهای بازه (مقادیر خیلی کم یا خیلی زیاد متغیر پیش‌بین) تقریباً صاف و نزدیک به ۰ یا ۱ باقی می‌ماند، اما در ناحیه میانی، شیب تندی دارد و به‌سرعت از یک مقدار پایین به یک مقدار بالا تغییر می‌کند. این ناحیه میانی همون جایی است که مدل بیشترین حساسیت و عدم قطعیت رو در تصمیم‌گیری بین دو کلاس دارد.

این رفتار، دقیقاً همان چیزی است که از یک مدل احتمالاتی خوب انتظار می‌رود: در نواحی که داده‌ها به‌وضوح متعلق به یک دسته هستند، مدل با اطمینان بالا پیش‌بینی می‌کند، و فقط در ناحیه مرزی بین دو دسته، عدم قطعیت نشان می‌دهد. به همین دلیل، این نوع مدل‌سازی نسبت به رگرسیون خطی برای مسائل طبقه‌بندی مناسب‌تر و قابل اعتمادتر است.

این محدودیت‌ها دقیقاً همان دلیلی است که نشان می‌دهد چرا برای مسائل طبقه‌بندی، به‌جای رگرسیون خطی باید از روش‌هایی مثل رگرسیون لجستیک استفاده کرد که خروجی‌شان همیشه در بازه معتبر $[0, 1]$ باقی می‌ماند.

مدل و فرمول ریاضی

پرسش اصلی این است: چگونه می‌توان رابطه بین $p(X) = \Pr(Y=1|X)$ و متغیر پیش‌بین X را مدل کرد؟ همان‌طور که پیش‌تر اشاره شد، اگر از یک مدل خطی ساده مانند

$$p(X) = \beta_0 + \beta_1 X$$

استفاده کنیم، برای مقادیر کوچک X احتمال منفی و برای مقادیر بزرگ X احتمال بزرگتر از یک پیش‌بینی می‌شود؛ نتیجه‌ای که هیچ‌گاه از نظر آماری منطقی نیست، چون احتمال همواره باید بین ۰ و ۱ باقی بماند.

برای رفع این مشکل، باید $p(X)$ را با تابعی مدل کنیم که برای تمام مقادیر X ، خروجی‌ای بین ۰ و ۱ تولید کند. در رگرسیون لجستیک، از **تابع لجستیک** استفاده می‌شود.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

این تابع همواره یک منحنی S شکل تولید می‌کند؛ به این معنا که صرف‌نظر از مقدار X ، همیشه یک پیش‌بینی منطقی و در بازه $[0, 1]$ به دست می‌آید. برای مقادیر کوچک X ، احتمال به ۰ نزدیک می‌شود (اما هرگز منفی نمی‌شود) و برای مقادیر بزرگ X ، احتمال به ۱ نزدیک می‌شود (اما هرگز از ۱ بیشتر نمی‌شود).

مفهوم Odds :

با کمی جابه‌جایی در رابطه بالا، می‌توان نوشت:

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}$$

عبارت $p(X)/[1-p(X)]$ را **odds** یا **شانس** می‌نامند که می‌تواند هر مقداری بین ۰ و بی‌نهایت بگیرد. مقادیر نزدیک به ۰ نشان‌دهنده احتمال بسیار پایین و مقادیر نزدیک به بی‌نهایت نشان‌دهنده احتمال بسیار بالای وقوع رویداد هستند. برای مثال، اگر $p(X) = 0.2$ باشد، آنگاه odds برابر است با $\frac{0.2}{0.8} = \frac{1}{4}$ یعنی به‌طور میانگین از هر ۵ نفر، ۱ نفر دچار آن

رویداد می‌شود. به‌همین ترتیب، اگر $p(X) = 0.9$ باشد، odds برابر با 9 خواهد بود؛ یعنی از هر 10 نفر، به‌طور میانگین 9 نفر دچار آن رویداد می‌شوند.

تخمین ضرایب با روش Maximum Likelihood

ضرایب β_0 و β_1 در مدل لجستیک، مقادیر نامعلومی هستند که باید بر اساس داده‌های آموزشی موجود تخمین زده شوند. در رگرسیون خطی، از روش کمترین مربعات (Least Squares) برای تخمین ضرایب استفاده می‌شد. اگرچه می‌توان از کمترین مربعات غیرخطی برای برازش مدل لجستیک هم استفاده کرد، اما روش عمومی‌تر و ترجیح داده‌شده **درست‌نمایی بیشینه (Maximum Likelihood)** است، چون خواص آماری بهتری دارد.

ایده اصلی این روش به این صورت است: به‌دنبال مقادیری برای β_0 و β_1 می‌گردیم که احتمال پیش‌بینی‌شده $\hat{p}(x_i)$ برای هر مشاهده، تا حد امکان به وضعیت واقعی مشاهده‌شده آن نزدیک باشد. به عبارت دیگر، می‌خواهیم مقادیری برای $\hat{\beta}_0$ و $\hat{\beta}_1$ پیدا کنیم که وقتی در مدل جایگذاری شوند، برای مشاهداتی که واقعاً به کلاس مثبت تعلق دارند، عددی نزدیک به یک، و برای مشاهداتی که به کلاس منفی تعلق دارند، عددی نزدیک به صفر پیش‌بینی کنند.

این ایده را می‌توان با یک عبارت ریاضی به نام **تابع درست‌نمایی (Likelihood Function)** فرموله کرد:

$$\ell(\beta_0, \beta_1) = \prod_{i: y_i=1} p(x_i) \prod_{i': y_{i'}=0} (1 - p(x_{i'}))$$

مقادیر $\hat{\beta}_0$ و $\hat{\beta}_1$ به گونه ای انتخاب می شوند که این تابع درست نمایی را بیشینه کنند.

پس از برآزش مدل، معمولاً خروجی شامل جدولی از ضرایب تخمین زده شده، خطای استاندارد آن ها (Standard Error)، آماره (Z-statistic) و مقدار (P-value) است. بسیاری از این مفاهیم مشابه خروجی رگرسیون خطی هستند:

- **خطای استاندارد (SE):** میزان دقت تخمین هر ضریب را نشان می دهد.
 - **آماره Z:** نقشی مشابه آماره t در رگرسیون خطی دارد و از رابطه $\hat{\beta}_1 / SE(\hat{\beta}_1)$ به دست می آید. مقدار مطلق بزرگ این آماره، شواهدی بر رد فرضیه صفر $H_0: \beta_1 = 0$ است.
 - **فرضیه صفر ($H_0: \beta_1 = 0$):** این فرضیه به این معناست که احتمال وقوع رویداد، به متغیر پیش بین بستگی ندارد.
 - **مقدار P:** اگر این مقدار بسیار کوچک باشد (معمولاً کمتر از ۰.۰۵)، فرضیه صفر رد می شود و می توان نتیجه گرفت که بین متغیر پیش بین و احتمال وقوع رویداد، ارتباط معناداری وجود دارد.
- معمولاً مقدار تخمین زده شده برای عرض از مبدأ (Intercept) به خودی خود اهمیت تفسیری چندانی ندارد؛ نقش اصلی آن، تنظیم میانگین احتمال های پیش بینی شده مدل متناسب با نسبت واقعی وقوع رویداد در داده ها است.

تفسیر ضرایب و odds ratio

پیش بینی احتمال با استفاده از مدل برآزش شده

پس از تخمین ضرایب $\hat{\beta}_0$ و $\hat{\beta}_1$ ، محاسبه احتمال پیش‌بینی‌شده برای هر مقدار دلخواه از متغیر پیش‌بین کار ساده‌ای است؛ کافی است این مقادیر را در فرمول تابع لجستیک جایگذاری کنیم:

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}}$$

با جایگذاری هر مقدار مشخص از X ، می‌توان احتمال متناظر با آن را به‌دست آورد. برای مثال، اگر برای یک متغیر پیش‌بین خاص، مقدار کمی از آن احتمال بسیار پایینی (کمتر از ۱٪) پیش‌بینی کند، در حالی که مقدار بزرگتر همان متغیر احتمال بسیار بالاتری (مثلاً حدود ۵۸٪) نتیجه دهد، این نشان می‌دهد که رابطه بین متغیر پیش‌بین و احتمال وقوع رویداد، هرچند غیرخطی، اما قوی و معنادار است.

یکی از روش‌های رایج و کاربردی برای تفسیر نتایج رگرسیون لجستیک، محاسبه **نسبت شانس (Odds Ratio)** بین دو گروه است. برای این کار، می‌توان احتمال پیش‌بینی‌شده هر گروه را با جایگذاری مقدار متغیر مجازی (۰ یا ۱) در مدل محاسبه کرد:

$$\widehat{\Pr}(Y = 1 | X = 1) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 \times 1}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 \times 1}}$$

$$\widehat{\Pr}(Y = 1 | X = 0) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 \times 0}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 \times 0}}$$

با مقایسه این دو احتمال، می‌توان به‌طور مستقیم و ملموس نشان داد که تعلق به یک گروه خاص، چه میزان بر احتمال وقوع رویداد موردنظر تأثیر می‌گذارد. این رویکرد، به‌ویژه در گزارش نتایج به مخاطبانی که آشنایی کمتری با مفاهیم آماری دارند، بسیار کاربردی است؛ چراکه به‌جای بیان ضرایب خام مدل (که تفسیر مستقیم آن‌ها دشوار است)، تفاوت در احتمال یا نسبت شانس بین دو گروه را به‌صورت شفاف نشان می‌دهد.

رگرسیون لجستیک چندگانه (Multiple)

تا اینجا مدل رگرسیون لجستیک را تنها با یک متغیر پیش‌بین بررسی کردیم. اما در بسیاری از مسائل واقعی، پیش‌بینی یک پاسخ دوتایی نیازمند در نظر گرفتن همزمان چندین متغیر پیش‌بین است. درست همان‌طور که رگرسیون خطی ساده به رگرسیون خطی چندمتغیره تعمیم یافت، می‌توان مدل لجستیک را نیز برای حالتی که چند متغیر پیش‌بین $X_1, \dots, X_p$ داریم، به‌صورت زیر تعمیم داد:

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$$

که در آن $X = (X_1, \dots, X_p)$ نشان‌دهنده p متغیر پیش‌بین است. با کمی جابه‌جایی، این رابطه را می‌توان به شکل زیر بازنویسی کرد:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}$$

درست مانند حالت تک‌متغیره، ضرایب $\beta_0, \beta_1, \dots, \beta_p$ با استفاده از روش درست‌نمایی بیشینه (Maximum Likelihood) تخمین زده می‌شوند.

پدیده درهم‌تنیدگی (Confounding)

یکی از نکات مهم و ظریفی که در استفاده از رگرسیون با چند متغیر پیش‌بین باید به آن توجه کرد، پدیده‌ای به نام درهم‌تنیدگی (Confounding) است. این پدیده زمانی رخ می‌دهد که

نتیجه یک تحلیل با استفاده از یک متغیر پیش‌بین به‌تنهایی، با نتیجه‌ای که از تحلیل چندمتغیره (با در نظر گرفتن سایر متغیرها) به‌دست می‌آید، متفاوت یا حتی متناقض باشد.

برای مثال، ممکن است در یک مدل تک‌متغیره، ضریب مربوط به یک متغیر گروهی (مثلاً وضعیت عضویت در یک گروه خاص) مثبت باشد؛ یعنی نشان دهد افراد آن گروه احتمال بیشتری برای وقوع رویداد مورد نظر دارند. اما وقتی همان متغیر را در کنار سایر متغیرهای مرتبط (مثلاً یک متغیر کمی که با هر دو متغیر گروهی و پاسخ همبستگی دارد) وارد مدل کنیم، ممکن است علامت ضریب معکوس شود؛ یعنی نشان دهد که در شرایط یکسان از نظر آن متغیر کمی، افراد آن گروه در واقع احتمال کمتری برای وقوع رویداد دارند.

دلیل این تناقض ظاهری این است که متغیر گروهی و متغیر کمی با یکدیگر همبستگی دارند. اگر افراد یک گروه به‌طور متوسط مقدار بالاتری از آن متغیر کمی داشته باشند، و آن متغیر کمی خودش با افزایش احتمال وقوع رویداد مرتبط باشد، آنگاه در تحلیل تک‌متغیره (بدون کنترل آن متغیر کمی)، اثر گروه به‌اشتباه بزرگتر یا حتی معکوس به نظر می‌رسد. اما در تحلیل چندمتغیره، که اثر متغیر کمی به‌طور جداگانه کنترل می‌شود، اثر واقعی و مستقل متغیر گروهی آشکار می‌شود.

این مثال به‌خوبی نشان می‌دهد که تحلیل با یک متغیر پیش‌بین به‌تنهایی، زمانی که متغیرهای پیش‌بین دیگری نیز ممکن است مرتبط باشند، می‌تواند نتایج گمراه‌کننده‌ای به همراه داشته باشد. به‌طور کلی، وجود همبستگی بین متغیرهای پیش‌بین، می‌تواند باعث تفاوت قابل‌توجه بین نتایج مدل‌های تک‌متغیره و چندمتغیره شود؛ پدیده‌ای که در آمار به آن **درهم‌تنیدگی (Confounding)** گفته می‌شود و توجه به آن هنگام تفسیر نتایج مدل‌های آماری از اهمیت بالایی برخوردار است.

رگرسیون لجستیک برای بیش از دو کلاس پاسخ

گاهی نیاز داریم متغیر پاسخی را طبقه‌بندی کنیم که بیش از دو دسته دارد. برای مثال، فرض کنید سه تشخیص پزشکی احتمالی داریم: سکته مغزی (stroke)، مصرف بیش از حد دارو (drug overdose)، و تشنج صرعی (epileptic seizure). در چنین حالتی، می‌خواهیم هم $\Pr(Y = \text{stroke} | X)$ و هم $\Pr(Y = \text{drug overdose} | X)$ را مدل کنیم، و مقدار $\Pr(Y = \text{epileptic seizure} | X)$ نیز به‌طور خودکار برابر است با:

$1 - \Pr(Y = \text{stroke} | X) - \Pr(Y = \text{drug overdose} | X)$
با استفاده از یک کلاس پایه (baseline)، مثلاً «تشنج صرعی» (epileptic seizure)، می‌توان مدل رگرسیون لجستیک را برای K کلاس به‌صورت زیر تعمیم داد:

برای $k = 1, \dots, K-1$:

$$\Pr(Y = k | X = x) = \exp(\beta_{0k} + \beta_{1k}x_1 + \dots + \beta_{pk}x_p) / [1 + \sum \exp(\beta_{0l} + \beta_{1l}x_1 + \dots + \beta_{pl}x_p)]$$

و برای کلاس پایه (K):

$$\Pr(Y = K | X = x) = 1 / [1 + \sum \exp(\beta_{0l} + \beta_{1l}x_1 + \dots + \beta_{pl}x_p)]$$

که مجموع در هر دو رابطه روی 1 از 1 تا $K-1$ است.

با این کدگذاری، نسبت شانس لگاریتمی (log odds) بین هر کلاس k و کلاس پایه به‌صورت خطی نسبت به متغیرهای پیش‌بین است:

$$\log[\Pr(Y=k|X=x) / \Pr(Y=K|X=x)] = \beta_{0k} + \beta_{1k}x_1 + \dots + \beta_{pk}x_p$$

نکته‌ی مهم درباره‌ی تفسیر ضرایب:

انتخاب کلاس K به‌عنوان پایه کاملاً دلخواه است. تغییر کلاس پایه فقط مقدار ضرایب برآوردشده را تغییر می‌دهد، اما مقادیر برازش‌شده (fitted values)، احتمالات پیش‌بینی‌شده

فصل ۳ :

نرخ خطا در طبقه‌بندی و طبقه‌بندی‌کننده بیز

نرخ خطا در طبقه‌بندی

تاکنون بحث ما درباره دقت مدل، بر حالت رگرسیون متمرکز بوده است. اما بسیاری از مفاهیمی که با آن‌ها آشنا شدیم، مانند مصالحه بایاس-واریانس، با تغییراتی جزئی به حالت طبقه‌بندی نیز قابل‌تعمیم هستند؛ چراکه اکنون y_i دیگر عددی نیست، بلکه کیفی است.

فرض کنید می‌خواهیم f را بر اساس مشاهدات آموزشی $\{(x_n, y_n), \dots, (x_1, y_1)\}$ برآورد کنیم، که در آن $y_1, \dots, y_n$ کیفی هستند. رایج‌ترین رویکرد برای سنجش دقت برآورد $\hat{f}$ ، **نرخ خطای آموزشی (Training Error Rate)** است؛ یعنی نسبت اشتباهاتی که هنگام اعمال $\hat{f}$ روی مشاهدات آموزشی رخ می‌دهد:

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$$

در این رابطه، $\hat{y}_i$ برچسب کلاس پیش‌بینی‌شده برای مشاهده i ام با استفاده از $\hat{f}$ است، و $I(y_i \neq \hat{y}_i)$ یک **متغیر شاخص (Indicator Variable)** است که اگر $y_i \neq \hat{y}_i$ باشد برابر با ۱، و در غیر این صورت برابر با ۰ است. اگر $I(y_i \neq \hat{y}_i) = 0$ باشد، مشاهده i ام به‌درستی طبقه‌بندی شده؛ در غیر این صورت، به‌اشتباه طبقه‌بندی شده است.

این رابطه، **نرخ خطای آموزشی** نام دارد چون بر اساس داده‌هایی محاسبه می‌شود که برای آموزش طبقه‌بندی‌کننده استفاده شده‌اند. اما مانند حالت رگرسیون، آنچه بیشتر اهمیت

دارد، نرخ خطایی است که از اعمال طبقه‌بندی‌کننده روی مشاهدات آزمون (که در آموزش استفاده نشده اند) به دست می‌آید. نرخ خطای آزمون مرتبط با مجموعه‌ای از مشاهدات آزمون به شکل (x_0, y_0) به صورت زیر تعریف می‌شود :

$$\text{Avg}(I(y_0 \neq \hat{y}_0))$$

یک طبقه‌بندی‌کننده **خوب**، طبقه‌بندی‌کننده‌ای است که این نرخ خطای آزمون را تا حد امکان کوچک کند.

طبقه‌بندی‌کننده بیز (Bayes Classifier)

می‌توان نشان داد (اگرچه اثبات آن خارج از حوصله این پروژه است) که نرخ خطای آزمون، به‌طور میانگین، توسط یک طبقه‌بندی‌کننده بسیار ساده که هر مشاهده را به محتمل‌ترین کلاس (با توجه به مقادیر متغیرهای پیش‌بین آن) نسبت می‌دهد، کمینه می‌شود. به عبارت دیگر، باید یک مشاهده آزمون با بردار پیش‌بین x_0 را به کلاس j نسبت دهیم که در آن مقدار زیر بیشینه باشد:

$$\Pr(Y = j \mid X = x_0)$$

توجه داشته باشید که این یک **احتمال شرطی** است؛ یعنی احتمال اینکه $Y=j$ باشد، به شرط اینکه بردار پیش‌بین مشاهده‌شده x_0 باشد. این طبقه‌بندی‌کننده بسیار ساده، **طبقه‌بندی‌کننده بیز** نام دارد. در یک مسئله دوکلاسه (با دو مقدار ممکن پاسخ، مثلاً کلاس ۱ و کلاس ۲)، طبقه‌بندی‌کننده بیز معادل است با پیش‌بینی کلاس یک، اگر $\Pr(Y=1|X=x_0) > 0.5$ باشد، و در غیر این صورت پیش‌بینی کلاس دو.

مرزی که در آن این احتمال دقیقاً برابر با ۵۰٪ است، **مرز تصمیم‌گیری بیز (Bayes Decision Boundary)** نامیده می‌شود؛ پیش‌بینی طبقه‌بندی‌کننده بیز توسط این مرز تعیین می‌شود.

طبقه‌بندی‌کننده بیز کمترین نرخ خطای آزمون ممکن، که **نرخ خطای بیز (Bayes Error Rate)** نامیده می‌شود، را تولید می‌کند. از آنجا که طبقه‌بندی‌کننده بیز همواره کلاسی را انتخاب می‌کند که احتمال بالاتری دارد، نرخ خطا در نقطه $X=x_0$ برابر است با ۱ منهای بیشینه این احتمال. به‌طور کلی، نرخ خطای بیز کلی به‌صورت زیر است:

$$1 - E \left(\max_j \Pr(Y = j | X) \right)$$

که در آن امید ریاضی، میانگین این احتمال روی تمام مقادیر ممکن X را محاسبه می‌کند. نرخ خطای بیز همیشه بزرگتر از صفر است، چون در جمعیت واقعی، کلاس‌ها معمولاً با یکدیگر همپوشانی دارند. این نرخ خطا، مشابه همان مفهوم خطای غیرقابل‌کاهش (Irreducible Error) است که پیش‌تر در حالت رگرسیون بررسی شد.

روش K نزدیکترین همسایه (K-Nearest Neighbors)

در حالت ایده‌آل، همیشه دوست داریم برای پیش‌بینی پاسخ‌های کیفی از طبقه‌بندی‌کننده بیز استفاده کنیم. اما برای داده‌های واقعی، توزیع شرطی Y به شرط X را نمی‌دانیم، و بنابراین محاسبه طبقه‌بندی‌کننده بیز غیرممکن است. به همین دلیل، طبقه‌بندی‌کننده بیز به‌عنوان یک استاندارد طلایی دست‌نیافتنی عمل می‌کند که سایر روش‌ها با آن مقایسه می‌شوند.

بسیاری از رویکردها تلاش می‌کنند توزیع شرطی Y به شرط X را برآورد کنند، و سپس مشاهده را به کلاسی با بالاترین احتمال **برآورد شده** نسبت دهند. یکی از این روش‌ها، طبقه‌بندی‌کننده **K نزدیکترین همسایه (KNN)** است. با در نظر گرفتن یک عدد صحیح

مثبت K و یک مشاهده آزمون x_0 ، طبقه‌بندی‌کننده KNN ابتدا K نقطه‌ای از داده‌های آموزشی را که نزدیکترین به x_0 هستند شناسایی می‌کند (این مجموعه را با N_0 نشان می‌دهیم). سپس، احتمال شرطی کلاس j را به صورت نسبتی از نقاط N_0 که مقدار پاسخشان برابر با j است، برآورد می‌کند:

$$\Pr(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

در نهایت، KNN قاعده بیز را اعمال کرده و مشاهده آزمون x_0 را به کلاسی با بالاترین احتمال برآوردشده نسبت می‌دهد.

با وجود اینکه KNN رویکردی بسیار ساده است، می‌تواند طبقه‌بندی‌کننده‌هایی تولید کند که به طرز شگفت‌انگیزی به طبقه‌بندی‌کننده بهینه بیز نزدیک هستند.

اثر انتخاب K روی عملکرد KNN

انتخاب مقدار K تأثیر بسیار زیادی روی طبقه‌بندی‌کننده KNN حاصل دارد. وقتی $K=1$ باشد، مرز تصمیم‌گیری بیش از حد منعطف است و الگوهایی در داده پیدا می‌کند که متناظر با مرز تصمیم‌گیری بیز نیستند؛ این حالت متناظر است با یک طبقه‌بندی‌کننده با اریب پایین اما واریانس بسیار بالا. با افزایش K ، روش کمتر منعطف می‌شود و مرز تصمیم‌گیری‌ای تولید می‌کند که به یک خط نزدیکتر است؛ این حالت متناظر است با یک طبقه‌بندی‌کننده با واریانس پایین اما اریب بالا.

درست مانند حالت رگرسیون، بین نرخ خطای آموزشی و نرخ خطای آزمون رابطه قوی‌ای وجود ندارد. وقتی $K=1$ باشد، نرخ خطای آموزشی KNN برابر با صفر است، اما نرخ خطای آزمون ممکن است بسیار بالا باشد. به طور کلی، هرچه از روش‌های طبقه‌بندی

منعطفتری استفاده کنیم، نرخ خطای آموزشی کاهش می‌یابد، اما نرخ خطای آزمون لزوماً کاهش نمی‌یابد.

با رسم نرخ خطای آموزشی و آزمون KNN به عنوان تابعی از $K/1$ (که میزان انعطاف‌پذیری روش را نشان می‌دهد)، مشاهده می‌شود که نرخ خطای آموزشی همواره با افزایش انعطاف‌پذیری کاهش می‌یابد. اما نرخ خطای آزمون یک شکل مشخص U به خود می‌گیرد؛ ابتدا کاهش می‌یابد (تا نقطه بهینه‌ای از K)، و سپس دوباره افزایش می‌یابد، زمانی که روش بیش‌ازحد منعطف شده و دچار بیش‌برازش (Overfitting) می‌شود.

در هر دو حالت رگرسیون و طبقه‌بندی، انتخاب سطح مناسب انعطاف‌پذیری، عاملی حیاتی برای موفقیت هر روش یادگیری آماری است. مصالحه اریبی-واریانس، و شکل U حاصل در نرخ خطای آزمون، می‌تواند این کار را دشوار سازد. در فصل بعدی این پروژه (فصل مربوط به آزمایش عملی)، روش‌های مختلفی برای برآورد نرخ خطای آزمون و در نتیجه انتخاب بهترین سطح انعطاف‌پذیری برای یک روش یادگیری آماری مشخص بررسی می‌شوند.

فصل ۴

تحلیل تشخیصی خطی (LDA)

رگرسیون لجستیک، احتمال $\Pr(Y=k|X=x)$ را مستقیماً با استفاده از تابع لجستیک مدل می‌کند. به عبارت آماری‌تر، این روش توزیع شرطی پاسخ Y را به شرط متغیرهای پیش‌بین X مدل می‌کند.

در این فصل، رویکرد جایگزین و کمی غیرمستقیم‌تری برای برآورد این احتمال‌ها بررسی می‌شود. در این رویکرد، به جای مدل‌سازی مستقیم Y به شرط X ، توزیع متغیرهای پیش‌بین X را به‌طور جداگانه در هر یک از کلاس‌های پاسخ (یعنی به شرط Y) مدل می‌کنیم، و سپس با استفاده از قضیه بیز (Bayes' Theorem)، این توزیع‌ها را معکوس کرده و به برآوردی برای $\Pr(Y=k|X=x)$ تبدیل می‌کنیم. وقتی این توزیع‌ها نرمال فرض شوند، مدل حاصل از نظر شکل ریاضی بسیار شبیه به رگرسیون لجستیک خواهد بود.

استفاده از قضیه بیز برای طبقه‌بندی :

فرض کنید می‌خواهیم یک مشاهده را در یکی از K کلاس ممکن طبقه‌بندی کنیم، که $K \geq 2$ است؛ به عبارت دیگر، متغیر پاسخ کیفی Y می‌تواند K مقدار متمایز و بدون ترتیب داشته باشد.

فرض کنید π_k نشان‌دهنده احتمال پیشین (Prior Probability) کلی باشد؛ یعنی احتمال اینکه یک مشاهده به‌طور تصادفی انتخاب‌شده، متعلق به کلاس k ام باشد. همچنین فرض کنید $f_k(X) \equiv \Pr(X=x|Y=k)$ نشان‌دهنده تابع چگالی (Density Function) متغیر X برای مشاهداتی باشد که از کلاس k ام می‌آیند. به‌بیان دیگر، $f_k(x)$ زمانی مقدار نسبتاً بزرگی دارد که احتمال بالایی وجود داشته باشد که یک مشاهده از کلاس k ام، مقدار $X \approx x$ داشته باشد؛ و برعکس، اگر بعید باشد که مشاهداتی از کلاس k ام مقدار $X \approx x$ داشته باشد، $f_k(x)$ کوچک خواهد بود.

با این تعاریف، قضیه بیز بیان می‌کند که:

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

مطابق با نمادگذاری قبلی، این مقدار را به‌اختصار $p_k(X) = \Pr(Y=k|X)$ می‌نامیم. این رابطه نشان می‌دهد که به‌جای محاسبه مستقیم $p_k(X)$ ، می‌توان به‌سادگی برآوردهایی از π_k و $f_k(X)$ را در رابطه بالا جایگذاری کرد. برآورد π_k معمولاً کار ساده‌ای است؛ کافی است نسبت مشاهدات آموزشی متعلق به کلاس k ام را از کل نمونه محاسبه کنیم. اما برآورد $f_k(X)$ معمولاً چالش‌برانگیزتر است، مگر اینکه فرض‌های ساده‌ای درباره شکل این توزیع‌ها در نظر بگیریم.

مقدار $p_k(x)$ را **احتمال پسین (Posterior Probability)** می‌نامند؛ یعنی احتمال اینکه یک مشاهده با مقدار پیش‌بین $X=x$ ، متعلق به کلاس k ام باشد، به شرط دانستن مقدار متغیر پیش‌بین آن مشاهده.

همان‌طور که پیش‌تر دیدیم، **طبقه‌بندی‌کننده بیز (Bayes Classifier)** که هر مشاهده را به کلاسی نسبت می‌دهد که در آن $p_k(X)$ بیشینه است، کمترین نرخ خطای ممکن را در بین

تمام طبقه‌بندی‌کننده‌ها دارد (البته به شرطی که تمام عبارات رابطه بالا به‌درستی مشخص شده باشند). بنابراین، اگر بتوانیم راهی برای برآورد $f_k(X)$ پیدا کنیم، می‌توانیم طبقه‌بندی‌کننده‌ای بسازیم که به طبقه‌بندی‌کننده بیز نزدیک باشد. این رویکرد، موضوع اصلی بخش‌های بعدی این فصل است.

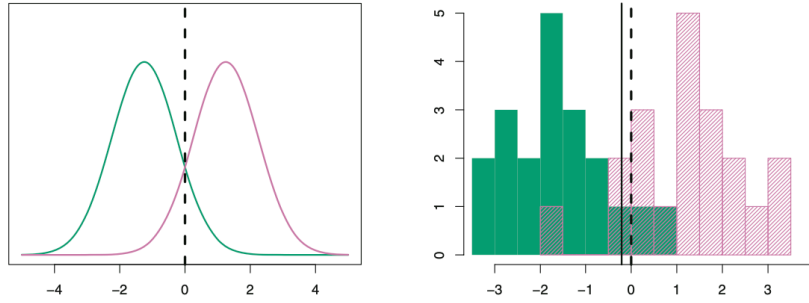
تحلیل تشخیصی خطی برای حالت $p=1$:

در ابتدا فرض می‌کنیم $p=1$ است؛ یعنی تنها یک متغیر پیش‌بین داریم. هدف این است که برآوردی برای $f_k(x)$ به‌دست آوریم تا بتوانیم آن را در رابطه قضیه بیز جایگذاری کرده و $p_k(x)$ را برآورد کنیم. سپس هر مشاهده را به کلاسی نسبت می‌دهیم که در آن $p_k(x)$ بیشینه باشد. برای این کار، ابتدا باید فرض‌هایی درباره شکل $f_k(x)$ در نظر بگیریم.

فرض می‌کنیم $f_k(x)$ از توزیع نرمال (Gaussian) پیروی می‌کند. در حالت تک‌بعدی، تابع چگالی نرمال به شکل زیر است:

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

که در آن μ_k و σ_k^2 به‌ترتیب میانگین و واریانس مربوط به کلاس k ام هستند. در این مرحله، فرض ساده‌کننده‌ای در نظر می‌گیریم: واریانس تمام K کلاس با یکدیگر برابر است ($\sigma_1^2 = \dots = \sigma_K^2 = \sigma^2$)، و این مقدار مشترک را با σ^2 نشان می‌دهیم. با جایگذاری این تابع چگالی در رابطه قضیه بیز، خواهیم داشت:



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

طبقه‌بندی‌کننده بیز، هر مشاهده $X=x$ را به کلاسی نسبت می‌دهد که در آن رابطه بالا بیشینه باشد. با گرفتن لگاریتم از این رابطه و مرتب‌سازی جملات، می‌توان نشان داد که این کار معادل است با نسبت‌دادن مشاهده به کلاسی که در آن مقدار تابع زیر بیشینه است:

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

برای مثال، اگر $K=2$ و $\pi_1=\pi_2$ باشد، طبقه‌بندی‌کننده بیز یک مشاهده را به کلاس ۱ نسبت می‌دهد اگر $2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$ باشد، و در غیر این صورت به کلاس ۲. در این حالت، مرز تصمیم‌گیری بیز متناظر با نقطه‌ای است که:

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

در عمل، حتی اگر کاملاً مطمئن باشیم که X در هر کلاس از یک توزیع نرمال پیروی می‌کند، همچنان باید پارامترهای $\mu_1, \dots, \mu_K, \pi_1, \dots, \pi_K$ و σ^2 را برآورد کنیم. روش **تحلیل تشخیصی خطی (LDA)** دقیقاً همین کار را انجام می‌دهد؛ یعنی طبقه‌بندی‌کننده بیز را با جایگذاری برآوردهایی برای این پارامترها تقریب می‌زند. برآوردهای مورد استفاده به شکل زیر هستند:

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

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

که در آن n تعداد کل مشاهدات آموزشی و n_k تعداد مشاهدات آموزشی متعلق به کلاس k است. برآورد $\hat{\mu}_k$ ، میانگین ساده تمام مشاهدات آموزشی کلاس k است، در حالی که $\hat{\sigma}^2$ را می‌توان به عنوان میانگین وزنی واریانس‌های نمونه‌ای هر یک از K کلاس در نظر گرفت. در صورتی که احتمال‌های پیشین $\pi_1, \dots, \pi_K$ از قبل مشخص نباشند، LDA این مقادیر را با استفاده از نسبت مشاهدات آموزشی متعلق به هر کلاس برآورد می‌کند:

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

طبقه‌بندی‌کننده LDA این برآوردها را در رابطه $\delta_k(x)$ جایگذاری کرده و هر مشاهده را به کلاسی نسبت می‌دهد که در آن مقدار زیر بیشینه باشد:

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

واژه **خطی (Linear)** در نام این طبقه‌بندی‌کننده، به این دلیل است که توابع تشخیص $\delta_k(x)$ توابعی خطی نسبت به x هستند (برخلاف توابع پیچیده‌تر ممکن دیگر).

تحلیل تشخیصی خطی برای حالت $p > 1$:

اکنون طبقه‌بندی‌کننده LDA را از حالت تک‌متغیره به حالتی که چند متغیر پیش‌بین داریم تعمیم می‌دهیم. فرض می‌کنیم $X = (X_1, X_2, \dots, X_p)$ از یک توزیع نرمال چندمتغیره (Multivariate Gaussian) با بردار میانگین اختصاصی برای هر کلاس و یک ماتریس کوواریانس مشترک بین تمام کلاس‌ها پیروی می‌کند. پیش از ادامه، مروری کوتاه بر این توزیع می‌کنیم.

برای نشان دادن اینکه یک متغیر تصادفی p -بعدی X دارای توزیع نرمال چندمتغیره است، می‌نویسیم $X \sim N(\mu, \Sigma)$. در این نماد، $E(X) = \mu$ میانگین X است (برداری با p مؤلفه) و $\text{Cov}(X) = \Sigma$ ماتریس کوواریانس $p \times p$ متغیر X است. به‌طور رسمی، تابع چگالی نرمال چندمتغیره به‌صورت زیر تعریف می‌شود:

$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)$$

فرض LDA در حالت چندبعدی: در حالت $p > 1$ ، طبقه‌بندی‌کننده LDA فرض می‌کند مشاهدات کلاس k ام از یک توزیع نرمال چندمتغیره $N(\mu_k, \Sigma)$ پیروی می‌کنند؛ که در آن μ_k بردار میانگین اختصاصی کلاس k ام است، و Σ ماتریس کوواریانس است که بین تمام K کلاس مشترک است (یعنی هیچ‌کدام از کلاس‌ها واریانس یا همبستگی اختصاصی خودشان را ندارند). با جایگذاری تابع چگالی کلاس k ام، یعنی $f_k(X=x)$ ، در رابطه قضیه بیز و انجام کمی محاسبات جبری، می‌توان نشان داد که طبقه‌بندی‌کننده بیز، مشاهده $X=x$ را به کلاسی نسبت می‌دهد که در آن مقدار زیر بیشینه باشد:

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

این رابطه، نسخه برداری/ماتریسی همان تابع تشخیص $\delta_k(x)$ است که پیشتر برای حالت تکبعدی معرفی کردیم.

مرزهای تصمیم‌گیری بیز:

این مرزها، مجموعه نقاطی x هستند که در آن‌ها $\delta_k(x) = \delta_l(x)$ برای هر دو کلاس متفاوت $k \neq l$ برقرار باشد؛ یعنی:

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

اگر K کلاس داشته باشیم، تعداد مرزهای تصمیم بیز برابر با تعداد جفت‌های ممکن بین این K کلاس است؛ برای مثال، با سه کلاس، سه مرز تصمیم‌گیری وجود خواهد داشت که هر یک مربوط به یکی از سه جفت ممکن کلاس‌هاست. این مرزها فضای متغیرهای پیش‌بین را به چند ناحیه مجزا تقسیم می‌کنند، و طبقه‌بندی‌کننده بیز هر مشاهده جدید را بر اساس ناحیه‌ای که در آن قرار می‌گیرد، طبقه‌بندی می‌کند.

برآورد پارامترها و طبقه‌بندی نهایی: درست مانند حالت تکبعدی، در عمل باید پارامترهای نامعلوم $\mu_1, \dots, \mu_K, \pi_1, \dots, \pi_K$ و Σ را برآورد کنیم؛ فرمول‌های برآورد این پارامترها مشابه فرمول‌هایی هستند که پیشتر برای حالت یکبعدی معرفی شدند. برای طبقه‌بندی یک مشاهده جدید LDA، $X=x$ این برآوردها را در رابطه $\delta_k(x)$ جایگذاری کرده و مشاهده را به کلاسی نسبت می‌دهد که در آن $\hat{\delta}_k(x)$ بیشینه باشد.

نکته مهم این است که $\delta_k(x)$ یک تابع خطی نسبت به x است؛ به عبارت دیگر، قاعده تصمیم‌گیری LDA تنها از طریق یک ترکیب خطی از مؤلفه‌های x به آن وابسته است، نه یک

تابع پیچیده‌تر. این ویژگی دقیقاً همان دلیلی است که واژه **خطی** در نام این روش به کار رفته است.

ماتریس درهم‌ریختگی

یکی از نکات مهمی که باید در ارزیابی یک طبقه‌بندی‌کننده دوتایی مدنظر قرار داد، این است که تنها گزارش کردن نرخ خطای کلی (Overall Error Rate)، تصویر کاملی از عملکرد مدل ارائه نمی‌دهد. در عمل، یک طبقه‌بندی‌کننده دوتایی می‌تواند دو نوع خطای متفاوت مرتکب شود: نسبت‌دادن اشتباه یک مشاهده متعلق به کلاس مثبت به کلاس منفی، یا برعکس، نسبت‌دادن اشتباه یک مشاهده متعلق به کلاس منفی به کلاس مثبت. تعیین اینکه کدامیک از این دو نوع خطا رخ می‌دهد، اغلب اهمیت بسیار زیادی دارد.

ابزار مناسب برای نمایش این اطلاعات، **ماتریس درهم‌ریختگی (Confusion Matrix)** است. عناصر روی قطر این ماتریس، مشاهداتی را نشان می‌دهند که وضعیت واقعی‌شان به‌درستی پیش‌بینی شده است، در حالی که عناصر خارج از قطر، مشاهداتی را نشان می‌دهند که به اشتباه طبقه‌بندی شده‌اند. جدول کلی این ماتریس به صورت زیر است:

جمع	پیش‌بینی: مثبت (Non-null)	پیش‌بینی: منفی (Null)	
N	False Positive (FP)	True Negative (TN)	واقعی: منفی (Null)
P	True Positive (TP)	False Negative (FN)	واقعی: مثبت (Non-null)
	*P	*N	جمع

در این جدول، «+» معمولاً نشان‌دهنده رویداد یا وضعیت مورد بررسی (مشابه «بیماری» در ادبیات اپیدمیولوژی، یا فرضیه خلاف در ادبیات آزمون فرض آماری) است، و «-» نشان‌دهنده وضعیت مقابل (مشابه «عدم بیماری»، یا فرضیه صفر) است.

نام معیار	تعریف	نام‌های مترادف
نرخ مثبت کاذب	FP/N	خطای نوع اول، ۱ منهای ویژگی (Specificity = TN/N)
نرخ مثبت واقعی	TP/P	۱ منهای خطای نوع دوم، توان آزمون، حساسیت، فراخوانی (Recall)
ارزش اخباری مثبت	TP/P*	دقت (Precision)، ۱ منهای نسبت کشف کاذب
ارزش اخباری منفی	TN/N*	—

بر اساس این چهار مقدار پایه (TN، FP، FN، TP)، معیارهای مهمی برای ارزیابی عملکرد کلاس‌محور یک طبقه‌بندی‌کننده تعریف می‌شوند که در حوزه‌های پزشکی، زیست‌شناسی و آمار کاربرد گسترده‌ای دارند:

نکته مهم درباره مخرج این کسرها: مخرج نرخ مثبت کاذب و نرخ مثبت واقعی، تعداد واقعی مشاهدات هر کلاس در جامعه است (یعنی N و P)، در حالی که مخرج ارزش اخباری مثبت و منفی، تعداد کل مشاهداتی است که مدل برای هر کلاس پیش‌بینی کرده است (یعنی *N و *P).

آستانه تصمیم‌گیری و مبادله بین خطاها :

طبقه‌بندی‌کننده بیز، یک مشاهده را به کلاس مثبت نسبت می‌دهد اگر احتمال پسین آن، بیش از ۰.۵ باشد:

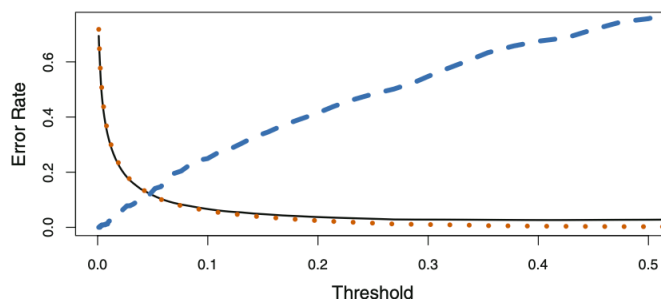
$$\Pr(Y = 1 \mid X = x) > 0.5$$

بنابراین، طبقه‌بندی‌کننده بیز (و به تبع آن، LDA) به‌طور پیش‌فرض از آستانه ۰.۵٪ برای احتمال پسین استفاده می‌کند تا یک مشاهده را به کلاس مثبت نسبت دهد. استفاده از این آستانه، کمترین نرخ خطای کلی ممکن را نتیجه می‌دهد (به شرط صحیح‌بودن فرض توزیع نرمال).

اما اگر نگران خطای خاصی باشیم (برای مثال، از دست دادن موارد واقعاً مثبت اهمیت بیشتری نسبت به طبقه‌بندی نادرست موارد منفی داشته باشد)، می‌توانیم این آستانه را کاهش دهیم. برای مثال، به جای استفاده از آستانه ۰.۵، می‌توانیم مشاهده را به کلاس مثبت نسبت دهیم اگر:

$$\Pr(Y = 1 \mid X = x) > 0.2$$

کاهش آستانه معمولاً باعث افزایش حساسیت (شناسایی بهتر و بیشتر موارد واقعاً مثبت) می‌شود، اما در عوض تعداد مثبت‌های کاذب و در نتیجه نرخ خطای کلی افزایش می‌یابد؛ یعنی یک نوع مبادله (Trade-off) بین دو نوع خطا برقرار است. انتخاب آستانه مناسب معمولاً یک تصمیم صرفاً آماری نیست، بلکه باید بر اساس دانش حوزه (Domain Knowledge) و هزینه نسبی هر یک از دو نوع خطا در کاربرد مورد نظر اتخاذ شود.



- **خط توپر مشکی: نرخ خطای کلی (Overall Error Rate)**
 - **خط چین آبی: نرخ خطا در بین مشاهدات واقعاً مثبت (یعنی نسبت مواردی که واقعاً مثبت بوده‌اند اما به اشتباه منفی طبقه‌بندی شده‌اند)**
 - **خط نقطه‌چین نارنجی: نرخ خطا در بین مشاهدات واقعاً منفی (یعنی نسبت مواردی که واقعاً منفی بوده‌اند اما به اشتباه مثبت طبقه‌بندی شده‌اند)**
- همان‌طور که در نمودار مشخص است، استفاده از آستانه ۰.۵ (که همان آستانه پیش‌فرض طبقه‌بندی‌کننده بیز است)، نرخ خطای کلی را به کمترین مقدار ممکن می‌رساند (خط مشکی

در این نقطه به کمینه خود نزدیک است). این نتیجه قابل انتظار است، چون طبقه‌بندی‌کننده بیز دقیقاً با همین آستانه، کمترین نرخ خطای کلی را تضمین می‌کند.

اما نکته مهم این است که وقتی آستانه ۰.۵ استفاده می‌شود، نرخ خطا در بین مشاهدات واقعاً مثبت (خط چین‌دار آبی) نسبتاً بالاست. با کاهش تدریجی آستانه، این نرخ خطا به‌طور پیوسته کاهش می‌یابد؛ اما در عوض، نرخ خطا در بین مشاهدات واقعاً منفی (خط نقطه‌چین نارنجی) افزایش می‌یابد.

این نمودار به‌وضوح نشان می‌دهد که بین این دو نوع خطا یک **مبادله (Trade-off)** وجود دارد: کاهش یکی از آن‌ها، به‌طور طبیعی باعث افزایش دیگری می‌شود. بنابراین، انتخاب بهترین مقدار آستانه، تصمیمی صرفاً آماری نیست و باید بر اساس دانش حوزه کاربردی و هزینه نسبی هر یک از دو نوع خطا در آن کاربرد خاص اتخاذ شود.

منحنی ROC و سطح زیر منحنی (AUC) :

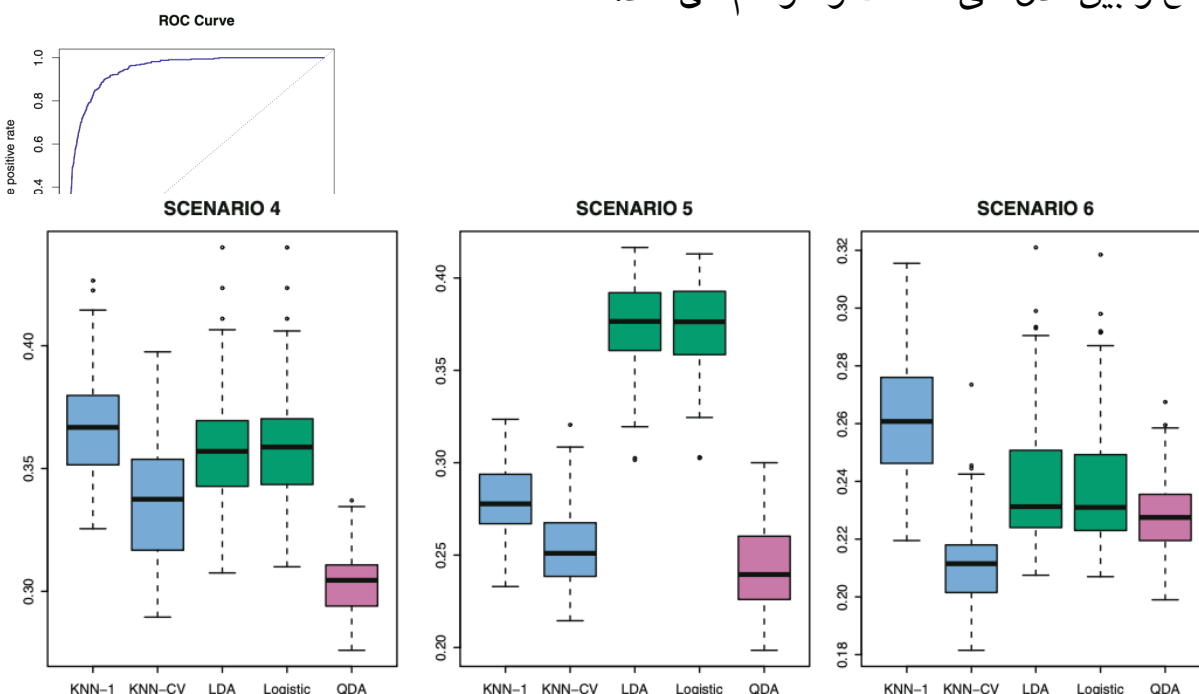
منحنی ROC (Receiver Operating Characteristic) ابزاری بسیار محبوب برای نمایش همزمان هر دو نوع خطا در تمام مقادیر ممکن آستانه تصمیم‌گیری است. نام «ROC» ریشه تاریخی دارد و از نظریه مخابرات گرفته شده؛ این نام مخفف عبارت «مشخصه عملکرد گیرنده» است.

این منحنی، نرخ مثبت واقعی (که همان حساسیت است) را در محور عمودی، و نرخ مثبت کاذب (که برابر با ۱ منهای ویژگی است) را در محور افقی نمایش می‌دهد. با تغییر مقدار آستانه از ۰ تا ۱، مسیری روی این نمودار ترسیم می‌شود که رفتار کلی طبقه‌بندی‌کننده را در تمام آستانه‌های ممکن نشان می‌دهد.

عملکرد کلی یک طبقه‌بندی‌کننده، با در نظر گرفتن تمام آستانه‌های ممکن، توسط **سطح زیر منحنی ROC (Area Under the Curve - AUC)** خلاصه می‌شود. یک منحنی ROC ایده‌آل به گوشه بالا-چپ نمودار نزدیک می‌شود، که نشان‌دهنده نرخ مثبت واقعی بالا و نرخ مثبت کاذب پایین به‌طور همزمان است؛ بنابراین هرچه AUC به عدد ۱ نزدیکتر باشد، طبقه‌بندی‌کننده بهتر عمل کرده است.

یک طبقه‌بندی‌کننده که هیچ اطلاعات مفیدی نداشته باشد و بهتر از حالت کاملاً تصادفی عمل نکند (یعنی طبقه‌بندی‌کننده «بدون اطلاعات»)، مقدار AUC برابر با ۰.۵ خواهد داشت؛ این حالت با یک خط مورب (قطری) روی نمودار ROC نمایش داده می‌شود.

منحنی‌های ROC به‌ویژه برای مقایسه عملکرد چند طبقه‌بندی‌کننده مختلف بسیار مفید هستند، زیرا برخلاف نرخ خطای کلی که تنها به یک آستانه خاص وابسته است، این منحنی‌ها تمام آستانه‌های ممکن را به‌طور همزمان در نظر می‌گیرند و امکان مقایسه‌ای جامع‌تر بین مدل‌های مختلف را فراهم می‌کنند.



در مقابل، منحنی آبی‌رنگ که در این نمودار دیده می‌شود، به‌وضوح به گوشه بالا-چپ نمودار متمایل است؛ یعنی در بسیاری از آستانه‌ها، نرخ مثبت واقعی بالایی دارد در حالی که نرخ مثبت کاذب پایین باقی می‌ماند. هرچه یک منحنی ROC بیشتر به این گوشه نزدیک شود، سطح زیر منحنی (AUC) آن به عدد ۱ نزدیکتر می‌شود و طبقه‌بندی‌کننده عملکرد بهتری خواهد داشت.

این نوع نمودار، ابزاری بصری و قدرتمند برای ارزیابی کیفیت کلی یک طبقه‌بندی‌کننده فراهم می‌کند، بدون آنکه نیاز باشد یک آستانه خاص را از پیش انتخاب کنیم.

تحلیل تشخیصی درجه دو (QDA)

همان‌طور که پیش‌تر بحث شد، LDA فرض می‌کند که مشاهدات هر کلاس از یک توزیع نرمال چندمتغیره با بردار میانگین اختصاصی و یک ماتریس کوواریانس مشترک بین تمام K کلاس پیروی می‌کنند.

تحلیل تشخیصی درجه دو (Quadratic Discriminant Analysis - QDA) رویکردی جایگزین ارائه می‌دهد. مانند LDA، طبقه‌بندی‌کننده QDA نیز از فرض توزیع نرمال برای مشاهدات هر کلاس استفاده می‌کند و برآوردهای پارامترها را در قضیه بیز جایگذاری می‌کند تا پیش‌بینی انجام دهد. اما برخلاف LDA، QDA فرض می‌کند که هر کلاس ماتریس کوواریانس اختصاصی خودش را دارد. به عبارت دیگر، فرض می‌شود مشاهدات کلاس k ام به صورت $X \sim N(\mu_k, \Sigma_k)$ هستند، که در آن Σ_k یک ماتریس کوواریانس اختصاصی برای کلاس k ام است.

با این فرض، طبقه‌بندی‌کننده بیز، مشاهده $X=x$ را به کلاسی نسبت می‌دهد که در آن مقدار زیر بیشینه باشد:

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

که با باز کردن این عبارت می‌توان آن را به صورت زیر نیز نوشت:

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

بنابراین، طبقه‌بندی‌کننده QDA شامل جایگذاری برآوردهایی برای μ_k ، Σ_k و π_k در رابطه بالا و سپس نسبت دادن مشاهده $X=x$ به کلاسی است که در آن این مقدار بیشینه باشد. تفاوت کلیدی با رابطه $\delta_k(x)$ در LDA این است که در اینجا، متغیر x به صورت **درجه دو** (Quadratic) ظاهر می‌شود؛ به همین دلیل این روش QDA نامگذاری شده است.

چرا انتخاب بین LDA و QDA اهمیت دارد؟

این سؤال مطرح می‌شود که چرا اهمیت دارد که K کلاس، ماتریس کوواریانس مشترک داشته باشند یا نه؛ به عبارت دیگر، چرا کسی LDA را به QDA ترجیح می‌دهد یا برعکس؟ پاسخ این سؤال در **مصالحه اریبی-واریانس (Bias-Variance Trade-off)** نهفته است.

وقتی p متغیر پیش‌بین داریم، برآورد یک ماتریس کوواریانس نیازمند برآورد $p(p+1)/2$ پارامتر است. QDA برای هر کلاس یک ماتریس کوواریانس جداگانه برآورد می‌کند؛ یعنی در مجموع $Kp(p+1)/2$ پارامتر. برای مثال، با ۵۰ متغیر پیش‌بین، این عدد می‌تواند مضربی از ۱۲۲۵ باشد که تعداد پارامتر بسیار زیادی است.

در مقابل، با فرض اینکه K کلاس یک ماتریس کوواریانس مشترک دارند، مدل LDA نسبت به x خطی می‌شود؛ به این معنا که تنها Kp ضریب خطی باید برآورد شود. در نتیجه، LDA طبقه‌بندی‌کننده‌ای با انعطاف‌پذیری بسیار کمتر نسبت به QDA است و در نتیجه واریانس بسیار پایین‌تری دارد؛ این ویژگی می‌تواند به بهبود عملکرد پیش‌بینی منجر شود. اما یک

مصالحه در کار است: اگر فرض LDA مبنی بر مشترکبودن ماتریس کوواریانس بین K کلاس به شدت نادرست باشد، LDA می تواند دچار اریبی بالا شود.

به طور کلی می توان گفت: LDA زمانی گزینه بهتری است که تعداد مشاهدات آموزشی نسبتاً کم باشد و کاهش واریانس اهمیت حیاتی داشته باشد. در مقابل، QDA زمانی توصیه می شود که مجموعه داده آموزشی بسیار بزرگ باشد (به طوری که واریانس طبقه بندی کننده نگرانی چندانی ایجاد نکند)، یا وقتی فرض مشترکبودن ماتریس کوواریانس بین K کلاس آشکارا غیرقابل قبول باشد.

فصل ۵

مقایسه روش‌های طبقه‌بندی

ارتباط بین رگرسیون لجستیک و LDA

در این پروژه، سه روش طبقه‌بندی متفاوت را بررسی کردیم: رگرسیون لجستیک، LDA و QDA. پیش از این، روش K نزدیکترین همسایه (KNN) را نیز معرفی کرده بودیم. اکنون به بررسی این می‌پردازیم که در چه شرایطی هر یک از این روش‌ها ممکن است بر دیگری برتری داشته باشد.

اگرچه انگیزه‌های رگرسیون لجستیک و LDA متفاوت است، این دو روش ارتباط نزدیکی با یکدیگر دارند. حالت دوکلاسه با یک متغیر پیش‌بین ($p=1$) را در نظر بگیرید، و فرض کنید $p_1(x)$ و $p_2(x)=1-p_1(x)$ به ترتیب احتمال تعلق مشاهده $X=x$ به کلاس ۱ و کلاس ۲ باشند. در چارچوب LDA، می‌توان نشان داد که log-odds به صورت زیر است:

$$\log \left(\frac{p_1(x)}{1 - p_1(x)} \right) = \log \left(\frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x$$

که در آن c_0 و c_1 توابعی از μ_1 ، μ_2 و σ^2 هستند. از طرف دیگر، می‌دانیم که در رگرسیون لجستیک:

$$\log \left(\frac{p_1}{1 - p_1} \right) = \beta_0 + \beta_1 x$$

هر دو رابطه بالا توابعی خطی نسبت به x هستند. بنابراین، هر دو روش رگرسیون لجستیک و LDA مرزهای تصمیم‌گیری خطی تولید می‌کنند. تنها تفاوت بین این دو رویکرد در این است که β_0 و β_1 با روش درست‌نمایی بیشینه (Maximum Likelihood) تخمین زده می‌شوند، در حالی که c_0 و c_1 با استفاده از میانگین و واریانس برآورد شده از یک توزیع نرمال محاسبه می‌شوند. این ارتباط مشابه، برای داده‌های چندبعدی ($p > 1$) نیز برقرار است.

از آنجا که رگرسیون لجستیک و LDA تنها در روش برازش خود متفاوت هستند، انتظار می‌رود که این دو رویکرد نتایج مشابهی ارائه دهند. این موضوع اغلب صادق است، اما همیشه چنین نیست. LDA فرض می‌کند مشاهدات از یک توزیع نرمال با ماتریس کوواریانس مشترک در هر کلاس پیروی می‌کنند؛ بنابراین وقتی این فرض تقریباً برقرار باشد، LDA می‌تواند نسبت به رگرسیون لجستیک بهبود ارائه دهد. برعکس، اگر فرض‌های نرمال بودن برقرار نباشند، رگرسیون لجستیک می‌تواند از LDA بهتر عمل کند.

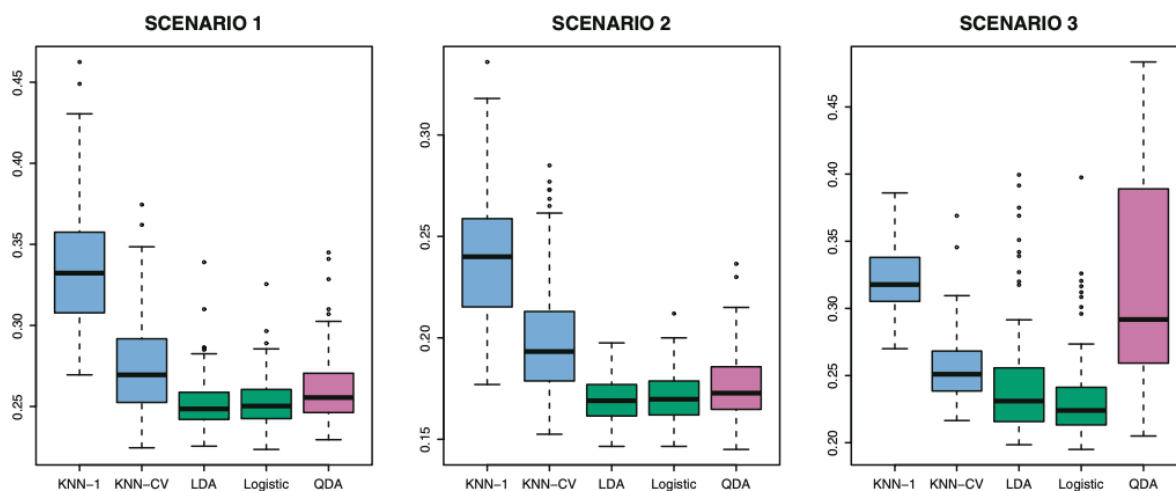
روش KNN رویکردی کاملاً متفاوت نسبت به روش‌های این فصل دارد. برای پیش‌بینی یک مشاهده $X=x$ ، ابتدا K مشاهده آموزشی نزدیکترین به x شناسایی می‌شوند و سپس X به کلاسی که اکثریت این مشاهدات به آن تعلق دارند، نسبت داده می‌شود. بنابراین، KNN یک رویکرد کاملاً ناپارامتری است: هیچ فرضی درباره شکل مرز تصمیم‌گیری در نظر گرفته نمی‌شود. به همین دلیل، انتظار می‌رود این روش زمانی که مرز تصمیم‌گیری بسیار غیرخطی باشد، بر LDA و رگرسیون لجستیک برتری داشته باشد. از سوی دیگر، KNN اطلاعاتی درباره اهمیت هر یک از متغیرهای پیش‌بین ارائه نمی‌دهد؛ یعنی برخلاف رگرسیون لجستیک، جدولی از ضرایب در اختیار نداریم.

در نهایت، QDA به‌عنوان یک راحل میانه بین روش ناپارامتری KNN و روش‌های خطی LDA و رگرسیون لجستیک عمل می‌کند. از آنجا که QDA مرز تصمیم‌گیری درجه دوم را فرض می‌کند، می‌تواند طیف وسیع‌تری از مسائل را نسبت به روش‌های خطی به‌درستی مدل کند. اگرچه QDA به‌اندازه KNN منعطف نیست، اما در شرایطی که تعداد مشاهدات

آموزشی محدود باشد، می‌تواند عملکرد بهتری داشته باشد؛ چون برخلاف KNN، QDA فرض‌هایی درباره شکل مرز تصمیم‌گیری در نظر می‌گیرد.

شش سناریوی مقایسه‌ای :

برای نشان‌دادن عملکرد این چهار روش طبقه‌بندی، شش سناریوی متفاوت مورد بررسی قرار گرفت. در سه سناریو، مرز تصمیم‌گیری بیز خطی است، و در سه سناریوی دیگر، این مرز غیرخطی است. در هر سناریو، ۱۰۰ مجموعه داده آموزشی تصادفی تولید و برای هر یک، هر روش برازش شده و نرخ خطای آزمون آن روی یک مجموعه داده آزمون بزرگ محاسبه شد.



سناریوی ۱: در هر یک از دو کلاس، ۲۰ مشاهده آموزشی وجود داشت. مشاهدات هر کلاس، متغیرهای تصادفی نرمال بدون همبستگی با میانگین متفاوت در هر کلاس بودند. در این حالت، LDA عملکرد خوبی داشت، که قابل انتظار است چون این دقیقاً همان مدلی است که LDA فرض می‌کند. KNN عملکرد ضعیفی داشت، چون واریانس اضافی آن با کاهش متناظری در اریب جبران نشد. QDA نیز عملکرد ضعیفتری نسبت به LDA داشت، چون مدلی منعطفتر از حد نیاز برآزش کرد. رگرسیون لجستیک، چون مرز تصمیم‌گیری خطی فرض می‌کند، نتایجی تنها کمی ضعیفتر از LDA داشت.

سناریوی ۲: جزئیات مشابه سناریوی ۱ بود، با این تفاوت که در هر کلاس، دو متغیر پیش‌بین همبستگی ۰.۵- داشتند. نتایج نشان داد تغییر چندانی در عملکرد نسبی روش‌ها نسبت به سناریوی قبل ایجاد نشد.

سناریوی ۳: متغیرهای X_1 و X_2 از توزیع t با ۵۰ مشاهده در هر کلاس تولید شدند. توزیع t شکلی مشابه توزیع نرمال دارد، اما تمایل بیشتری به تولید نقاط دورافتاده (نقاطی که فاصله زیادی از میانگین دارند) دارد. در این سناریو، مرز تصمیم‌گیری همچنان خطی بود، بنابراین با چارچوب رگرسیون لجستیک سازگار بود. اما این سناریو فرض‌های LDA را نقض می‌کرد، چون مشاهدات از توزیع نرمال گرفته نشده بودند. نتایج نشان داد رگرسیون لجستیک بهتر از LDA عمل کرد، هرچند هر دو روش نسبت به رویکردهای دیگر برتری داشتند. به‌طور خاص، عملکرد QDA به‌طور قابل‌توجهی به‌دلیل عدم نرمال‌بودن داده‌ها بدتر شد.

سناریوی ۴: داده‌ها از یک توزیع نرمال تولید شدند، با همبستگی ۰.۵ بین متغیرهای پیش‌بین در کلاس اول و همبستگی ۰.۵- در کلاس دوم. این ساختار دقیقاً منطبق با فرض QDA بود و منجر به مرزهای تصمیم‌گیری درجه‌دوم شد. نتایج نشان داد QDA در این سناریو، از تمام روش‌های دیگر بهتر عمل کرد.

سناریوی ۵: در هر کلاس، مشاهدات از یک توزیع نرمال با متغیرهای پیش‌بین بدون همبستگی تولید شدند. اما پاسخ‌ها با استفاده از تابع لجستیک و با در نظر گرفتن X_1^2 ، X_2^2 و $X_1 \times X_2$ به‌عنوان متغیرهای پیش‌بین نمونه‌گیری شدند. در نتیجه، مرز تصمیم‌گیری درجه‌دوم بود. نتایج نشان داد QDA بار دیگر بهترین عملکرد را داشت، و پس از آن KNN با انتخاب خودکار (KNN-CV) K قرار گرفت. روش‌های خطی عملکرد ضعیفی داشتند.

سناریوی ۶: جزئیات مشابه سناریوی قبل بود، اما پاسخ‌ها از یک تابع غیرخطی پیچیده‌تر نمونه‌گیری شدند. در نتیجه، حتی مرزهای تصمیم‌گیری درجه‌دوم QDA نیز نتوانستند داده‌ها را به‌درستی مدل کنند. نتایج نشان داد QDA عملکردی کمی بهتر از روش‌های خطی داشت، در حالی که روش بسیار منعطف‌تر KNN-CV بهترین نتایج را ارائه داد. اما KNN با $K=1$ بدترین نتایج را در بین تمام روش‌ها داشت؛ این موضوع نشان می‌دهد که حتی وقتی داده‌ها رابطه غیرخطی پیچیده‌ای دارند، یک روش ناپارامتری مانند KNN، اگر سطح هموارسازی به‌درستی انتخاب نشود، همچنان می‌تواند نتایج ضعیفی ارائه دهد.

چه زمانی کدام روش بهتر عمل می‌کند ؟

این شش مثال نشان می‌دهند که هیچ روشی در تمام موقعیت‌ها بر دیگر روش‌ها برتری ندارد. زمانی که مرزهای تصمیم‌گیری واقعی خطی باشند، رویکردهای LDA و رگرسیون لجستیک عملکرد خوبی خواهند داشت. زمانی که مرزها به‌طور متوسط غیرخطی باشند، QDA می‌تواند نتایج بهتری ارائه دهد. و در نهایت، برای مرزهای تصمیم‌گیری بسیار پیچیده‌تر، یک رویکرد ناپارامتری مانند KNN می‌تواند برتر باشد. اما سطح هموارسازی برای یک روش ناپارامتری باید با دقت انتخاب شود.

تعمیم مدل‌ها با استفاده از جمله‌های غیرخطی

همان‌طور که در فصل رگرسیون دیدیم، می‌توان با استفاده از تبدیل‌های غیرخطی متغیرهای پیش‌بین (مانند X^2 ، X^3 و حتی X^4)، رابطه غیرخطی بین متغیرهای پیش‌بین و پاسخ را مدل

کرد. رویکرد مشابهی را می‌توان در چارچوب طبقه‌بندی نیز به‌کار برد. برای مثال، می‌توان با افزودن جمله‌های X^2 ، X^3 و X^4 به‌عنوان متغیرهای پیش‌بین، نسخه‌ای منعطف‌تر از رگرسیون لجستیک ساخت. البته این کار ممکن است عملکرد رگرسیون لجستیک را بهبود دهد یا نه؛ این موضوع بستگی دارد به اینکه آیا افزایش واریانس ناشی از افزایش انعطاف‌پذیری، با کاهش کافی در اریب جبران می‌شود یا خیر.

همین رویکرد را می‌توان برای LDA نیز به‌کار برد. اگر تمام جمله‌های درجه‌دوم و حاصل‌ضرب‌های متقابل ممکن را به LDA اضافه کنیم، شکل مدل حاصل مشابه مدل QDA خواهد بود، هرچند برآورد پارامترها متفاوت خواهد بود. این رویکرد به ما امکان می‌دهد در جایی بین مدل LDA و مدل QDA حرکت کنیم و انعطاف‌پذیری مدل را به‌صورت تدریجی تنظیم کنیم.

فصل ۶

آزمایشگاه رگرسیون لجستیک، QDA، LDA و KNN

داده‌های بازار سهام

بررسی ساختار داده Smarket

پیش از ساخت مدل‌های طبقه‌بندی، ابتدا باید با ساختار داده Smarket آشنا شویم. در این مرحله نام متغیرها، ابعاد داده و خلاصه‌ی آماری هر یک از متغیرها بررسی می‌شود.

```
> library(ISLR)
> names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"      "Volume"
[8] "Today"     "Direction"
> dim(Smarket)
[1] 1250  9
> summary(Smarket)
```

Year	Lag1	Lag2	Lag3	Lag4
Min. :2001	Min. :-4.922000	Min. :-4.922000	Min. :-4.922000	Min. :-4.922000
1st Qu.:2002	1st Qu.: -0.639500	1st Qu.: -0.639500	1st Qu.: -0.640000	1st Qu.: -0.640000
Median :2003	Median : 0.039000	Median : 0.039000	Median : 0.038500	Median : 0.038500
Mean :2003	Mean : 0.003834	Mean : 0.003919	Mean : 0.001716	Mean : 0.001636
3rd Qu.:2004	3rd Qu.: 0.596750	3rd Qu.: 0.596750	3rd Qu.: 0.596750	3rd Qu.: 0.596750
Max. :2005	Max. : 5.733000	Max. : 5.733000	Max. : 5.733000	Max. : 5.733000

Lag5	Volume	Today	Direction
Min. :-4.92200	Min. :0.3561	Min. :-4.922000	Down:602
1st Qu.: -0.64000	1st Qu.:1.2574	1st Qu.: -0.639500	Up :648
Median : 0.03850	Median :1.4229	Median : 0.038500	
Mean : 0.00561	Mean :1.4783	Mean : 0.003138	
3rd Qu.: 0.59700	3rd Qu.:1.6417	3rd Qu.: 0.596750	
Max. : 5.73300	Max. :3.1525	Max. : 5.733000	

```
> |
```

با اجرای (Smarket) names مشخص می‌شود که مجموعه داده Smarket شامل ۹ متغیر است:

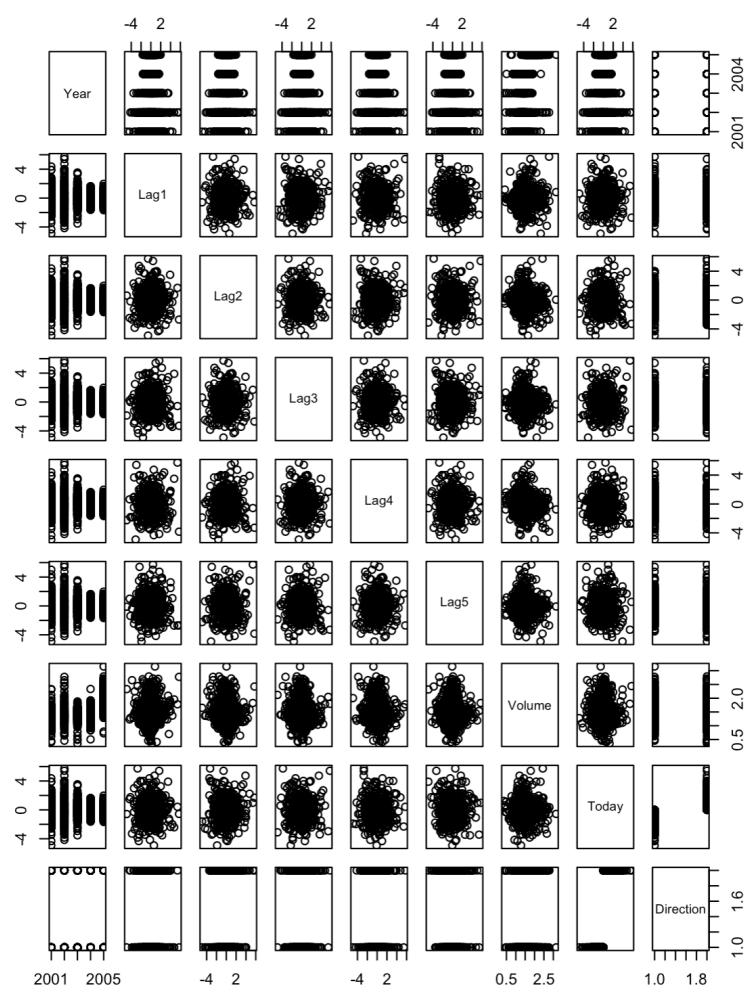
- **Year**: سال مربوط به هر مشاهده
 - **Lag1 تا Lag5**: بازده درصدی هر یک از پنج روز معاملاتی قبل
 - **Volume**: حجم معاملات روز قبل (بر حسب میلیارد سهم)
 - **Today**: بازده درصدی همان روز
 - **Direction**: جهت حرکت بازار (Up یا Down)
- خروجی (dim(Smarket) نشان می‌دهد داده شامل ۱۲۵۰ مشاهده (روز معاملاتی) و ۹ متغیر است.

خروجی (summary(Smarket) خلاصه‌ای آماری هر متغیر را نشان می‌دهد:

- متغیر **Year** از ۲۰۰۱ تا ۲۰۰۵ را پوشش می‌دهد و میانگین و میانه هر دو ۲۰۰۳ هستند، یعنی داده‌ها به‌طور یکنواخت در این بازه‌ی زمانی پخش شده‌اند.
- متغیرهای **Lag1 تا Lag5** دامنه‌ی تقریباً یکسانی دارند (بین ۴.۹۲- تا ۵.۷۳ درصد)، که منطقی است چون همگی بازده روزانه‌ی همان شاخص با تأخیرهای مختلف هستند. میانگین همه‌ی آن‌ها نزدیک به صفر است، یعنی به‌طور میانگین بازار نه سود قابل‌توجهی داشته و نه زیان قابل‌توجهی.
- متغیر **Volume** از ۰.۳۵۶ تا ۳.۱۵۲ میلیارد سهم متغیر است با میانگین ۱.۴۷۸ میلیارد، که نشان‌دهنده‌ی نوسان و رشد حجم معاملات در طول این پنج سال است.
- متغیر **Today** مشابه متغیرهای Lag رفتار می‌کند، چون خودش هم بازده درصدی روزانه است، با میانگین ۰.۰۰۳۱۳۸ درصد.
- متغیر **Direction** کیفی است و دو مقدار دارد: ۶۰۲ روز Down (نزولی) و ۶۴۸ روز Up (صعودی). این توزیع تقریباً متوازن (حدود ۴۸٪ به ۵۲٪) نشان می‌دهد

داده دچار عدم توازن شدید کلاسی نیست، که برای مدل‌سازی طبقه‌بندی در مراحل بعد (Logistic Regression، LDA، QDA و KNN) نکته‌ی مهمی است.

`pairs(Smarket)`



: pairs(Smarket)

دستور `pairs()` ماتریسی از نمودارهای پراکندگی (scatter plot) بین تمام جفت‌متغیرهای مجموعه‌داده Smarket رسم می‌کند. هر خانه از این ماتریس، رابطه‌ی بین دو متغیر مربوط به سطر و ستون خودش را نشان می‌دهد.

```

Error in cor(Smarket[, 1:8]) : must be numeric
> cor(Smarket[, -9])

```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume
Year	1.00000000	0.029699649	0.030596422	0.033194581	0.035688718	0.029787995	0.53900647
Lag1	0.02969965	1.000000000	-0.026294328	-0.010803402	-0.002985911	-0.005674606	0.04090991
Lag2	0.03059642	-0.026294328	1.000000000	-0.025896670	-0.010853533	-0.003557949	-0.04338321
Lag3	0.03319458	-0.010803402	-0.025896670	1.000000000	-0.024051036	-0.018808338	-0.04182369
Lag4	0.03568872	-0.002985911	-0.010853533	-0.024051036	1.000000000	-0.027083641	-0.04841425
Lag5	0.02978799	-0.005674606	-0.003557949	-0.018808338	-0.027083641	1.000000000	-0.02200231
Volume	0.53900647	0.040909908	-0.043383215	-0.041823686	-0.048414246	-0.022002315	1.000000000
Today	0.03009523	-0.026155045	-0.010250033	-0.002447647	-0.006899527	-0.034860083	0.01459182
Today							
Year	0.030095229						
Lag1	-0.026155045						
Lag2	-0.010250033						
Lag3	-0.002447647						
Lag4	-0.006899527						
Lag5	-0.034860083						
Volume	0.014591823						
Today	1.000000000						

با بررسی این نمودار چند نکته مشخص می‌شود:

۱. رابطه‌ی بین متغیرهای Lag1 تا Lag5:

نمودارهای پراکندگی بین این متغیرها به شکل ابری بدون الگوی مشخص (تصادفی) هستند، که نشان می‌دهد بین بازده روزهای مختلف رابطه‌ی خطی یا الگوی قابل‌توجهی وجود ندارد.

۲. رابطه‌ی بین Year و Volume:

در سطر و ستون مربوط به این دو متغیر، یک روند صعودی نسبتاً مشخص دیده می‌شود؛ یعنی با گذشت سال‌ها (از ۲۰۰۱ به ۲۰۰۵)، مقدار Volume افزایش پیدا کرده است. این با همبستگی عددی بالا (۰.۵۳۹) که قبلاً محاسبه شد، همخوانی دارد.

۳. متغیر Direction:

چون این متغیر کیفی (دسته‌ای) است، نمودارهای مربوط به آن به‌صورت دو خط افقی/عمودی (برای دو مقدار Up و Down) نمایش داده می‌شوند، نه به شکل ابر پراکنده. با نگاه به این ردیف و ستون، تفاوت محسوسی بین مقادیر Lag برای روزهای Up و Down دیده نمی‌شود؛ یعنی به‌صورت بصری هم مشخص است که پیش‌بینی جهت بازار تنها از روی این متغیرها کار ساده‌ای نیست.

نتیجه‌گیری کلی:

این نمودار به‌صورت بصری همان نتیجه‌ای را تأیید می‌کند که از ماتریس همبستگی به‌دست

آمد: بین بازده روزهای گذشته و بازده امروز رابطه‌ی خطی قوی وجود ندارد، و تنها رابطه‌ی قابل‌توجه بین Year و Volume است.

تفسیر ماتریس همبستگی

دستور $\text{cor}(\text{Smarket}[, -9], \dots)$ همبستگی دوبه‌دوی بین تمام متغیرهای عددی مجموعه داده Smarket را محاسبه می‌کند (متغیر Direction چون کیفی است، با $[, -9]$ از محاسبه حذف شده است).

مقدار همبستگی بین دو متغیر عددی بین -۱ تا +۱ است:

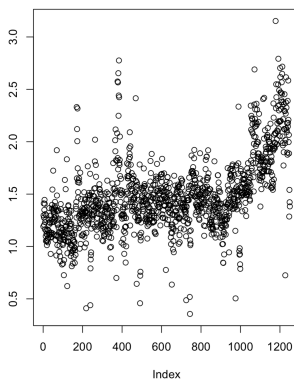
- نزدیک به +۱ یعنی رابطه‌ی مستقیم قوی
 - نزدیک به -۱ یعنی رابطه‌ی معکوس قوی
 - نزدیک به صفر یعنی عملاً رابطه‌ی خطی وجود ندارد
- با بررسی جدول، چند نکته‌ی کلیدی دیده می‌شود:

۱. همبستگی بین متغیرهای Lag با یکدیگر و با Today:

تمام این مقادیر بسیار کوچک هستند (بین -۰.۰۴۸ تا +۰.۰۴۱)، یعنی عملاً نزدیک به صفر. این نشان می‌دهد بازده هیچ‌کدام از روزهای گذشته (Lag1 تا Lag5) رابطه‌ی خطی معناداری با بازده امروز (Today) یا با یکدیگر ندارند.

۲. تنها همبستگی قابل‌توجه: Year و Volume

مقدار همبستگی بین Year و Volume برابر با ۰.۵۳۹ است که در مقایسه با بقیه‌ی مقادیر جدول (که همه نزدیک صفرند) بسیار بزرگتر است. این یعنی با گذشت سال‌ها، حجم معاملات (Volume) به‌طور محسوسی افزایش پیدا کرده است.



۳. قطر اصلی ماتریس

مقادیر روی قطر اصلی جدول همیشه برابر ۱ هستند، چون همبستگی هر متغیر با خودش کامل (۱.۰۰) است.

نتیجه‌گیری:

این جدول به صورت عددی همان چیزی را تأیید می‌کند که در نمودار pairs () به صورت بصری دیده شد: نبود رابطه‌ی خطی معنادار بین بازده گذشته و بازده امروز، و وجود یک رابطه‌ی صعودی واضح بین زمان (Year) و حجم معاملات (Volume).

```
attach(Smarket)
plot(Volume)
```

با بررسی این نمودار چند نکته مشخص می‌شود:

۱. روند کلی صعودی:

هرچند داده‌ها در هر بازه نوسان زیادی دارند، اما با حرکت از ابتدای محور (Index نزدیک صفر، مربوط به سال ۲۰۰۱) به سمت انتها (Index نزدیک ۱۲۵۰، مربوط به سال ۲۰۰۵)، مقدار Volume به طور محسوس افزایش پیدا کرده است.

۲. پراکندگی داده‌ها:

در بخش ابتدایی نمودار (سال‌های اولیه)، اغلب مقادیر Volume بین ۱.۰ تا ۱.۵ میلیارد سهم متمرکز هستند. اما در بخش انتهایی نمودار (سال‌های پایانی)، مقادیر به طور کلی بالاتر رفته و بین ۱.۵ تا ۳.۰ میلیارد سهم پراکنده شده‌اند.

۳. نوسانات موضعی:

با وجود روند کلی صعودی، نوسانات کوتاه مدت زیادی هم دیده می‌شود (مثلاً افزایش ناگهانی

حوالی Index=400 یا Index=1150)، که نشان می‌دهد حجم معاملات علاوه بر روند بلندمدت، تحت تأثیر رویدادهای کوتاهمدت بازار هم قرار می‌گیرد.

نتیجه‌گیری:

این نمودار به‌صورت بصری همان چیزی را نشان می‌دهد که از همبستگی عددی بین Year و Volume (برابر با ۰.۵۳۹) به‌دست آمد: میانگین حجم سهام معامله‌شده روزانه از سال ۲۰۰۱ تا ۲۰۰۵ روند رشد داشته است.

رگرسیون لجستیک

ساخت مدل :

```
> glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
+ data=Smarket, family=binomial)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000    0.240736  -0.523   0.601
Lag1         -0.073074    0.050167  -1.457   0.145
Lag2         -0.042301    0.050086  -0.845   0.398
Lag3          0.011085    0.049939   0.222   0.824
Lag4          0.009359    0.049974   0.187   0.851
Lag5          0.010313    0.049511   0.208   0.835
Volume        0.135441    0.158360   0.855   0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

در این مرحله یک مدل رگرسیون لجستیک برازش داده می‌شود تا متغیر Direction با استفاده از Lag1 تا Lag5 و Volume پیش‌بینی شود. تابع glm () مدل‌های خطی تعمیم‌یافته (generalized linear models) را برازش می‌کند که رگرسیون لجستیک یکی از

انواع آن است. نحوه‌ی استفاده از glm () شبیه به lm () است، با این تفاوت که باید آرگومان family=binomial را وارد کنیم تا به R بگوییم رگرسیون لجستیک اجرا کند.

خروجی نشان می‌دهد کوچکترین p-value مربوط به Lag1 است. ضریب منفی این متغیر نشان می‌دهد اگر بازار دیروز بازده مثبت داشته باشد، احتمال کمتری دارد که امروز صعودی باشد؛ اما چون p-value برابر ۰.۱۵ نسبتاً بزرگ است، شواهد قطعی برای وجود رابطه‌ی واقعی بین Lag1 و Direction وجود ندارد.

۲. استخراج ضرایب و پیش‌بینی احتمالات

```
> coef(glm.fit)
(Intercept)      Lag1      Lag2      Lag3      Lag4      Lag5
-0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
Volume
0.135440659
> summary(glm.fit)$coef
              Estimate Std. Error   z value Pr(>|z|)
(Intercept) -0.126000257  0.24073574 -0.5233966  0.6006983
Lag1         -0.073073746  0.05016739 -1.4565986  0.1452272
Lag2         -0.042301344  0.05008605 -0.8445733  0.3983491
Lag3          0.011085108  0.04993854  0.2219750  0.8243333
Lag4          0.009358938  0.04997413  0.1872757  0.8514445
Lag5          0.010313068  0.04951146  0.2082966  0.8349974
Volume        0.135440659  0.15835970  0.8552723  0.3924004
> summary(glm.fit)$coef[,4]
(Intercept)      Lag1      Lag2      Lag3      Lag4      Lag5      Volume
0.6006983      0.1452272  0.3983491  0.8243333  0.8514445  0.8349974  0.3924004
> glm.probs=predict(glm.fit, type="response")
> glm.probs[1:10]
      1      2      3      4      5      6      7      8
0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
      9     10
0.5176135 0.4888378
```

تابع `coef` () فقط ضرایب مدل برازش‌شده را نمایش می‌دهد. تابع `predict` () احتمال صعودی‌شدن بازار را پیش‌بینی می‌کند. گزینه‌ی `type="response"` به `R` می‌گوید خروجی را به‌صورت احتمال $P(Y=1|X)$ بدهد (نه لاجیت). چون داده‌ای به `predict` () داده نشده، احتمالات برای همان داده‌ی آموزشی محاسبه شده‌اند.

با استفاده از تابع `contrasts` () مشخص می‌شود که `R` به مقدار `Up` عدد ۱ اختصاص داده است، پس این احتمالات مربوط به صعودی‌شدن بازارند نه نزولی‌شدن.

```
> contrasts(Direction)
      Up
Down  0
Up    1
```

۳. تبدیل احتمالات به برچسب و ساخت جدول درهم‌ریختگی (Confusion Matrix)

```
> glm.pred=rep("Down", 1250)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred, Direction)
      Direction
glm.pred Down  Up
      Down  145 141
      Up    457 507
> (507+145)/1250
[1] 0.5216
> mean(glm.pred==Direction)
[1] 0.5216
```

ابتدا برداری از ۱۲۵۰ مقدار "Down" ساخته می‌شود. سپس هر عنصری که احتمال پیش‌بینی شده‌اش بیش از ۰.۵ باشد، به "Up" تغییر می‌کند. تابع table () جدول درهم‌ریختگی می‌سازد تا مشخص شود چند مشاهده درست یا غلط طبقه‌بندی شده‌اند.

عناصر روی قطر اصلی جدول، پیش‌بینی‌های درست را نشان می‌دهند. مدل در مجموع $652 = 145 + 507$ پیش‌بینی درست داشته، که برابر با دقت ۵۲.۲٪ است.

نکته‌ی مهم: این عدد گمراه‌کننده است، چون مدل روی همان ۱۲۵۰ مشاهده هم آموزش دیده و هم آزمایش شده. یعنی ۴۷.۸٪ خطا در واقع خطای آموزشی (training error) است، که معمولاً خوش‌بینانه‌تر از خطای واقعی مدل روی داده‌ی جدید است.

۴. تفکیک داده به آموزش (Train) و آزمون (Test)

```
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252    9
> Direction.2005=Direction[!train]
```

برای ارزیابی واقعی‌تر، داده به دو بخش تقسیم می‌شود: مشاهدات قبل از ۲۰۰۵ برای آموزش، و مشاهدات سال ۲۰۰۵ به‌عنوان داده‌ی نگه‌داشته‌شده (held-out) برای آزمون.

شیء `train` یک بردار بولی (Boolean) با ۱۲۵۰ عنصر است: مقادیر مربوط به قبل از ۲۰۰۵ برابر `TRUE` و مقادیر مربوط به ۲۰۰۵ برابر `FALSE` هستند. علامت ! مقادیر بولی را معکوس می‌کند، پس `Smarket[!train,]` فقط داده‌های سال ۲۰۰۵ (۲۵۲ مشاهده) را برمی‌گرداند.

```
> glm.pred=rep("Down", 252)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred, Direction.2005)
      Direction.2005
glm.pred Down Up
Down      77 97
Up        34 44
> mean(glm.pred==Direction.2005)
[1] 0.4801587
> mean(glm.pred!=Direction.2005)
[1] 0.5198413
```

نتیجه: نرخ خطای آزمون این بار ۵۲٪ است — یعنی حتی بدتر از حدس تصادفی! این نشان می‌دهد که به‌طور کلی نمی‌توان از بازده روزهای قبل، عملکرد آینده‌ی بازار را پیش‌بینی کرد.

۵. بازسازی مدل با متغیرهای منتخب

از آنجا که Lag1 و Lag2 بیشترین قدرت پیش‌بینی را در مدل اولیه داشتند، مدل با حذف بقیه‌ی متغیرها دوباره ساخته می‌شود:

```
> glm.fit=glm(Direction~Lag1+Lag2, data=Smarket, family=binomial,
+             subset=train)
> glm.probs=predict(glm.fit, Smarket.2005, type="response")
> glm.pred=rep("Down", 252)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred, Direction.2005)
      Direction.2005
glm.pred Down  Up
      Down   35  35
      Up    76 106
> mean(glm.pred==Direction.2005)
[1] 0.5595238
> 106/(106+76)
[1] 0.5824176
```

این بار نتیجه کمی بهتر است: ۵۶٪ از حرکات روزانه به‌درستی پیش‌بینی شده‌اند. اما نکته‌ی مهم این‌جاست: یک استراتژی ساده‌تر که همیشه پیش‌بینی کند بازار صعودی می‌شود، هم ۵۶٪ درست از آب درمی‌آید! پس از نظر نرخ خطای کلی، رگرسیون لجستیک بهتر از حدس ساده نیست.

با این حال، جدول درهم‌ریختگی نشان می‌دهد در روزهایی که مدل پیش‌بینی صعودی می‌کند، دقت آن ۵۸٪ است — که می‌تواند مبنای یک استراتژی معاملاتی باشد (خرید در روزهایی که مدل صعود پیش‌بینی می‌کند و اجتناب از معامله در روزهای نزولی). البته باید بررسی دقیق‌تری شود که آیا این بهبود واقعی است یا صرفاً شانس.

۶. پیش‌بینی برای مقادیر خاص جدید

```
> predict(glm.fit, newdata=data.frame(Lag1=c(1.2,1.5),
+                                     Lag2=c(1.1,-0.8)), type="response")
      1      2
0.4791462 0.4960939
```

این دستور احتمال صعودی شدن بازار را برای دو روز فرضی جدید پیش‌بینی می‌کند: یکی با $Lag1=1.2$ و $Lag2=1.1$ ، و دیگری با $Lag1=1.5$ و $Lag2=-0.8$. نتیجه احتمال‌های ۰.۴۷۹۱ و ۰.۴۹۶۱ است — یعنی هر دو زیر ۰.۵، پس مدل برای هر دو روز پیش‌بینی نزولی می‌کند.

تحلیل تشخیصی خطی (LDA)

اکنون مدل LDA را روی داده‌های Smarket پیاده‌سازی می‌کنیم. در نرم‌افزار R، برای برازش مدل LDA از تابع `lda()` که بخشی از کتابخانه MASS است استفاده می‌شود. نحوه فراخوانی این تابع دقیقاً مشابه توابع `lm()` و `glm()` است، با این تفاوت که آرگومان `family` در آن وجود ندارد. مدل را تنها با استفاده از مشاهدات قبل از سال ۲۰۰۵ (مجموعه آموزشی) برازش می‌دهیم:

```

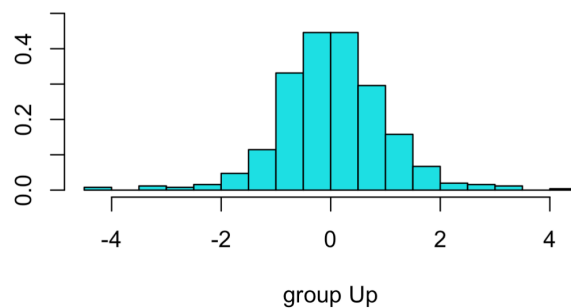
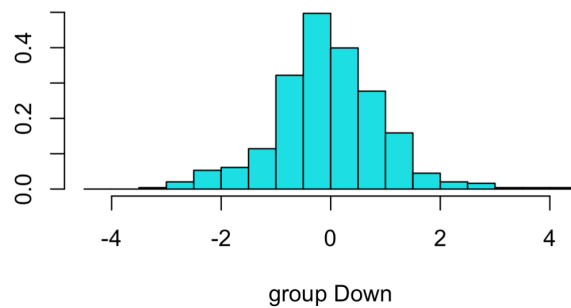
> library(ISLR)
> library(MASS)
> lda.fit = lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
> lda.fit
Call:
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
      Down      Up 
0.491984 0.508016 

Group means:
      Lag1      Lag2 
Down 0.04279022 0.03389409 
Up   -0.03954635 -0.03132544 

Coefficients of linear discriminants:
      LD1 
Lag1 -0.6420190 
Lag2 -0.5135293 
> plot(lda.fit)

```



خروجی مدل LDA نشان می‌دهد که $\hat{\pi}_1 = 0.492$ و $\hat{\pi}_2 = 0.508$ ؛ به عبارت دیگر، ۴۹.۲٪ از مشاهدات آموزشی مربوط به روزهایی است که بازار روند نزولی داشته است. همچنین، مدل میانگین‌های گروهی را نیز ارائه می‌دهد؛ این مقادیر، میانگین هر متغیر پیش‌بین در هر کلاس هستند و به عنوان برآورد μ_k توسط LDA استفاده می‌شوند. این میانگین‌ها نشان می‌دهند که

بازده دو روز قبل، تمایل به منفی بودن دارد در روزهایی که بازار صعودی است، و تمایل به مثبت بودن دارد در روزهایی که بازار نزولی است.

بخش «ضرایب تشخیص خطی» (Coefficients of Linear Discriminants) در خروجی، ترکیب خطی متغیرهای Lag1 و Lag2 را نشان می‌دهد که برای تشکیل قاعده تصمیم‌گیری LDA استفاده می‌شود؛ به عبارت دیگر، این‌ها ضرایب مربوط به عناصر $X=x$ در تابع تشخیص $\delta_k(x)$ هستند. اگر مقدار زیر بزرگ باشد، طبقه‌بندی‌کننده LDA روند صعودی بازار را پیش‌بینی می‌کند، و اگر کوچک باشد، روند نزولی را پیش‌بینی می‌کند:

$$-0.642 \times \text{Lag1} - 0.514 \times \text{Lag2}$$

تابع `plot()` نمودارهایی از این «تشخیص‌های خطی» را رسم می‌کند؛ این مقادیر با محاسبه همین عبارت خطی برای هر یک از مشاهدات آموزشی به دست می‌آیند.

پیش‌بینی با مدل LDA

تابع `predict()` در این حالت، یک لیست با سه عنصر باز می‌گرداند: عنصر اول، `class`، حاوی پیش‌بینی‌های LDA درباره جهت حرکت بازار است. عنصر دوم، `posterior`، یک ماتریس است که ستون k ام آن، احتمال پسین تعلق مشاهده متناظر به کلاس k ام را (بر اساس رابطه قضیه بیز) نشان می‌دهد. در نهایت، عنصر x حاوی همان مقادیر تشخیص خطی است که پیش‌تر توضیح داده شد.

```
> lda.pred = predict(lda.fit, Smarket.2005)
> names(lda.pred)
[1] "class"      "posterior" "x"
~
```

همان‌طور که پیش‌تر مشاهده شد، پیش‌بینی‌های LDA و رگرسیون لجستیک تقریباً یکسان هستند:

```
> lda.class = lda.pred$class
> table(lda.class, Direction.2005)
      Direction.2005
lda.class Down  Up
      Down   35  35
      Up    76 106
> mean(lda.class == Direction.2005)
[1] 0.5595238
```

نتیجه نشان می‌دهد که دقت پیش‌بینی مدل روی مجموعه آزمون برابر با ۵۶٪ است.

با اعمال آستانه ۵۰٪ روی احتمال‌های پسین، می‌توان همان پیش‌بینی‌های موجود در lda.pred\$class را بازتولید کرد:

```
> sum(lda.pred$posterior[,1] >= .5)
[1] 70
> sum(lda.pred$posterior[,1] < .5)
[1] 182
```

نکته مهم این است که احتمال پسین خروجی‌شده توسط مدل، متناظر با احتمال کاهش بازار است:

```
> lda.pred$posterior[1:20, 1]
      999      1000      1001      1002      1003      1004      1005      1006
0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016 0.4872861
      1007      1008      1009      1010      1011      1012      1013      1014
0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761 0.4744593 0.4799583
      1015      1016      1017      1018
0.4935775 0.5030894 0.4978806 0.4886331
> lda.class[1:20]
[1] Up Up Up Up Up Up Up Up Up Up Down Up Up Up Up
[17] Up Down Up Up
Levels: Down Up
```

استفاده از آستانه دلخواه برای تصمیم‌گیری

اگر بخواهیم به جای آستانه پیش‌فرض ۵۰٪، از آستانه دیگری برای تصمیم‌گیری استفاده کنیم، این کار به راحتی امکان‌پذیر است. برای مثال، فرض کنید فقط زمانی می‌خواهیم پیش‌بینی کاهش بازار انجام دهیم که اطمینان بسیار بالایی (مثلاً بیش از ۹۰٪) از این کاهش داشته باشیم:

```
> sum(lda.pred$posterior[,1] > .9)
[1] 0
```

نتیجه این دستور برابر با صفر است؛ یعنی هیچ‌کدام از روزهای سال ۲۰۰۵ به این آستانه نمی‌رسند. در واقع، بیشترین احتمال پسین کاهش بازار در کل سال ۲۰۰۵ تنها برابر با ۵۲.۰۲٪ بوده است.

تحلیل تشخیصی درجه دو (QDA)

اکنون مدل QDA را روی داده‌های Smarket برآزش می‌دهیم. در R، پیاده‌سازی QDA با استفاده از تابع `qda()` انجام می‌شود که این تابع نیز بخشی از کتابخانه MASS است. نحوه فراخوانی این تابع دقیقاً مشابه `lda()` است:

```
> qda.fit = qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
> qda.fit
Call:
qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
      Down      Up 
0.491984 0.508016 

Group means:
      Lag1      Lag2 
Down 0.04279022 0.03389409 
Up   -0.03954635 -0.03132544
```

خروجی این مدل شامل میانگین‌های گروهی است، اما برخلاف LDA، شامل ضرایب تشخیص خطی نیست؛ چون طبقه‌بندی‌کننده QDA بر پایه یک تابع درجه دوم (نه خطی) از متغیرهای پیش‌بین عمل می‌کند و بنابراین چنین ضرایبی برای آن تعریف نمی‌شود.

تابع predict () دقیقاً به همان شکلی که برای LDA استفاده شد، برای QDA نیز به‌کار می‌رود:

```
> qda.class = predict(qda.fit, Smarket.2005)$class
> table(qda.class, Direction.2005)
      Direction.2005
qda.class Down  Up
      Down    30  20
      Up     81 121
> mean(qda.class == Direction.2005)
[1] 0.5992063
```

نتیجه جالب‌توجه این است که پیش‌بینی‌های QDA در حدود ۶۰٪ از موارد صحیح هستند، در حالی که داده‌های سال ۲۰۰۵ اصلاً در برآزش مدل استفاده نشده‌اند. این سطح از دقت برای داده‌های بازار سهام، که به‌طور معمول مدل‌سازی آن‌ها بسیار دشوار است، بسیار قابل‌توجه است. این نتیجه نشان می‌دهد که ممکن است شکل درجه دوم فرض‌شده توسط QDA، رابطه واقعی بین متغیرها را دقیق‌تر از شکل خطی فرض‌شده در LDA و رگرسیون لجستیک نمایش دهد. با این حال، توصیه می‌شود پیش از هرگونه نتیجه‌گیری قطعی، عملکرد این روش روی یک مجموعه داده آزمون بزرگتر نیز ارزیابی شود؛ چراکه نمی‌توان تنها بر اساس یک نتیجه محدود، به این جمع‌بندی رسید که این رویکرد همواره عملکرد بهتری نسبت به سایر روش‌ها خواهد داشت.

روش K نزدیکترین همسایه (KNN)

اکنون KNN را با استفاده از تابع knn () که بخشی از کتابخانه class است، پیاده‌سازی می‌کنیم. این تابع نسبت به سایر توابع برازش مدل که تاکنون با آن‌ها کار کرده‌ایم، رفتار کاملاً متفاوتی دارد. به جای رویکرد دمرحله‌ای که ابتدا مدل را برازش می‌دهیم و سپس با استفاده از آن پیش‌بینی انجام می‌دهیم، تابع knn () پیش‌بینی‌ها را با یک دستور واحد انجام می‌دهد. این تابع به چهار ورودی نیاز دارد:

۱. ماتریسی شامل متغیرهای پیش‌بین مربوط به داده‌های آموزشی (با نام train.X)
۲. ماتریسی شامل متغیرهای پیش‌بین مربوط به داده‌هایی که می‌خواهیم برایشان پیش‌بینیو معیارهای ارزیابی مدل ثابت می‌مانند. به همین دلیل، تفسیر ضرایب در رگرسیون لجستیک چندکلاسه باید با احتیاط و همیشه با توجه به کلاس پایه‌ی انتخاب‌شده انجام شود.

انجام دهیم (با نام test.X)

۳. برداری شامل برچسب‌های کلاس مربوط به مشاهدات آموزشی (با نام train.Direction)

۴. مقداری برای K، یعنی تعداد نزدیکترین همسایه‌هایی که طبقه‌بندی‌کننده استفاده می‌کند

از تابع cbind () (مخفف column bind) برای ترکیب متغیرهای Lag1 و Lag2 در قالب دو ماتریس، یکی برای مجموعه آموزشی و دیگری برای مجموعه آزمون، استفاده می‌کنیم:

```
> library(class)
> train.X = cbind(Lag1, Lag2)[train,]
> test.X = cbind(Lag1, Lag2)[!train,]
> train.Direction = Direction[train]
```

اکنون می‌توان از تابع knn () برای پیش‌بینی حرکت بازار در تاریخ‌های سال ۲۰۰۵ استفاده کرد. پیش از فراخوانی knn ()، یک بذر تصادفی (Random Seed) تنظیم می‌کنیم؛ چون اگر چند مشاهده به‌طور مساوی به‌عنوان نزدیکترین همسایه در نظر گرفته

شوند، R به‌طور تصادفی بین آن‌ها یکی را انتخاب می‌کند. بنابراین، برای تضمین تکرارپذیری نتایج، باید یک بذر تصادفی تنظیم شود:

```
> set.seed(1)
> knn.pred = knn(train.X, test.X, train.Direction, k = 1)
> table(knn.pred, Direction.2005)
      Direction.2005
knn.pred Down Up
      Down   43 58
      Up    68 83
> (83+43) / 252
[1] 0.5
```

نتایج با $K=1$ چندان مطلوب نیستند، چون تنها ۵۰٪ از مشاهدات به‌درستی پیش‌بینی شده‌اند. البته ممکن است این نتیجه ضعیف به این دلیل باشد که $K=1$ منجر به یک مدل بیش‌ازحد منعطف می‌شود. در ادامه، همین تحلیل را با $K=3$ تکرار می‌کنیم:

```
> knn.pred = knn(train.X, test.X, train.Direction, k = 3)
> table(knn.pred, Direction.2005)
      Direction.2005
knn.pred Down Up
      Down   48 54
      Up    63 87
> mean(knn.pred == Direction.2005)
[1] 0.5357143
```

نتایج اندکی بهبود یافته‌اند. اما افزایش بیشتر K هیچ بهبود دیگری ایجاد نمی‌کند. به نظر می‌رسد که برای این مجموعه داده، QDA بهترین نتایج را در میان تمام روش‌هایی که تاکنون بررسی کرده‌ایم ارائه می‌دهد.

کاربرد روی داده‌های بیمه کاروان (Caravan Insurance Data)

اکنون روش KNN را روی مجموعه داده Caravan که بخشی از کتابخانه ISLR است، اعمال می‌کنیم. این داده شامل ۸۵ متغیر پیش‌بین است که ویژگی‌های جمعیت‌شناختی ۵۸۲۲ فرد را نشان می‌دهند. متغیر پاسخ، Purchase، نشان می‌دهد که آیا هر فرد بیمه قایق کاروان (Caravan) خریداری کرده است یا نه. در این مجموعه داده، تنها ۶٪ از افراد بیمه را خریداری کرده‌اند.

اهمیت استانداردسازی متغیرها: از آنجا که KNN بر پایه فاصله بین مشاهدات عمل می‌کند، مقیاس متغیرهای پیش‌بین اهمیت زیادی دارد. هر متغیری که مقیاس بزرگتری داشته باشد (برای مثال، یک متغیر با واحد سالانه در مقابل ماهانه)، تأثیر بسیار بیشتری روی فاصله بین مشاهدات، و در نتیجه روی طبقه‌بندی‌کننده KNN، خواهد گذاشت؛ حتی اگر از نظر واقعی اهمیت کمتری داشته باشد. برای رفع این مشکل، معمولاً داده‌ها را استاندارد می‌کنیم؛ یعنی هر متغیر را طوری تبدیل می‌کنیم که میانگین صفر و انحراف معیار یک داشته باشد. این کار با تابع scale () انجام می‌شود:

```
> standardized.X = scale(Caravan[, -86])
```

پس از این تبدیل، هر ستون از standardized.X دارای انحراف معیار برابر با یک و میانگین برابر با صفر خواهد بود.

تقسیم داده‌ها به مجموعه آموزشی و آزمون: اکنون مشاهدات را به دو مجموعه تقسیم می‌کنیم؛ مجموعه آزمون شامل ۱۰۰۰ مشاهده اول، و مجموعه آموزشی شامل باقی مشاهدات است. سپس مدل KNN را با $K=1$ روی داده‌های آموزشی برازش داده و عملکرد آن را روی داده‌های آزمون ارزیابی می‌کنیم:

```
> library(ISLR)
> attach(Caravan)
> test = 1:1000
> train.X = standardized.X[-test,]
> test.X = standardized.X[test,]
> train.Y = Purchase[-test]
> test.Y = Purchase[test]
> set.seed(1)
> knn.pred = knn(train.X, test.X, train.Y, k = 1)
> mean(test.Y != knn.pred)
[1] 0.118
> mean(test.Y != "No")
[1] 0.059
```

نرخ خطای KNN روی ۱۰۰۰ مشاهده آزمون، کمی کمتر از ۱۲٪ است. در نگاه اول، این عدد نسبتاً خوب به نظر می‌رسد. اما از آنجا که تنها ۶٪ از مشتریان بیمه را خریداری کرده‌اند، می‌توانستیم با پیش‌بینی همیشگی «عدم خرید» (بدون توجه به مقادیر متغیرهای پیش‌بین) نیز به نرخ خطای ۶٪ برسیم!

اهمیت انتخاب معیار ارزیابی مناسب: فرض کنید فروش بیمه به هر فرد هزینه غیرقابل چشم‌پوشی دارد (مثلاً یک فروشنده باید شخصاً به دیدار هر مشتری بالقوه برود). اگر شرکت بخواهد بیمه را به مجموعه‌ای تصادفی از مشتریان بفروشد، نرخ موفقیت تنها ۶٪ خواهد بود که با توجه به هزینه‌های درگیر، احتمالاً بسیار پایین است. به همین دلیل، شرکت ترجیح می‌دهد بیمه را فقط به مشتریانی بفروشد که احتمال خرید بالایی دارند. بنابراین، نرخ خطای کلی دیگر معیار موردنظر نیست؛ بلکه نسبت افرادی که به‌درستی پیش‌بینی شده‌اند بیمه را خواهند خرید، اهمیت دارد.

نتایج نشان می‌دهد که KNN با $K=1$ عملکردی به مراتب بهتر از حدس تصادفی، در میان مشتریانی که پیش‌بینی شده بیمه خواهند خرید، دارد. از میان ۷۷ مشتری‌ای که پیش‌بینی شده بود بیمه بخرند، ۹ نفر (۱۱.۷٪) واقعاً خرید کردند؛ این نرخ دو برابر نرخ است که از حدس تصادفی به‌دست می‌آید:

```
> table(knn.pred, test.Y)
      test.Y
knn.pred  No  Yes
      No  873  50
      Yes   68   9
> 9/(68+9)
[1] 0.1168831
```

اثر تغییر K: با افزایش K به ۳، نرخ موفقیت به ۱۹٪ افزایش می‌یابد، و با $K=5$ این نرخ به ۲۶.۷٪ می‌رسد؛ یعنی بیش از چهار برابر نرخ حدس تصادفی. این نشان می‌دهد که KNN موفق به یافتن الگوهای واقعی در این مجموعه داده دشوار شده است:

```
> knn.pred = knn(train.X, test.X, train.Y, k = 3)
> table(knn.pred, test.Y)
      test.Y
knn.pred  No  Yes
      No  920  54
      Yes   21   5
> 5/26
[1] 0.1923077
> knn.pred = knn(train.X, test.X, train.Y, k = 5)
> table(knn.pred, test.Y)
      test.Y
knn.pred  No  Yes
      No  930  55
      Yes   11   4
> 4/15
[1] 0.2666667
```

مقایسه با رگرسیون لجستیک: برای مقایسه، مدل رگرسیون لجستیک را نیز روی همین داده‌ها برازش می‌دهیم. اگر از آستانه پیش‌فرض ۰.۵ برای پیش‌بینی استفاده کنیم، با مشکلی مواجه می‌شویم: تنها ۷ مشاهده از داده‌های آزمون پیش‌بینی می‌شوند که بیمه بخرند، و حتی بدتر از آن، پیش‌بینی همه این ۷ مورد اشتباه است! با این حال، لازم نیست از آستانه ۰.۵ استفاده کنیم. اگر به جای آن، هر زمان که احتمال پیش‌بینی شده خرید بیشتر از ۰.۲۵ باشد، خرید را پیش‌بینی کنیم، نتایج به مراتب بهتر می‌شود: ۳۳ نفر پیش‌بینی می‌شوند که بیمه بخرند، و پیش‌بینی برای حدود ۳۳٪ از آن‌ها صحیح است؛ این نرخ بیش از پنج برابر بهتر از حدس تصادفی است:

```

> glm.fit = glm(Purchase ~ ., data = Caravan, family = binomial, subset = -test)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> glm.probs = predict(glm.fit, Caravan[test,], type = "response")
> glm.pred = rep("No", 1000)
> glm.pred[glm.probs > .5] = "Yes"
> table(glm.pred, test.Y)
      test.Y
glm.pred  No  Yes
      No  934  59
      Yes   7   0
> glm.pred = rep("No", 1000)
> glm.pred[glm.probs > .25] = "Yes"
> table(glm.pred, test.Y)
      test.Y
glm.pred  No  Yes
      No  919  48
      Yes   22  11
> 11/(22+11)
[1] 0.3333333

```

این مثال به خوبی نشان می‌دهد که در مسائل واقعی با کلاس‌های نامتوازن (مانند این داده که تنها ۶٪ از مشاهدات به کلاس مثبت تعلق دارند)، انتخاب آستانه مناسب و معیار ارزیابی درست، اهمیت بسیار بیشتری نسبت به صرفاً نگاه کردن به نرخ خطای کلی مدل دارد.