امیرهشام - ۹۸۳۱۱۳۳

۱-

هنگامی که از یک مجموعه داده جداگانه برای هرس درخت استفاده می‌کنیم، خطر وارد کردن سوگیری (Bias) در فرآیند هرس را داریم. این به این دلیل است که درخت هرس شده بر اساس مجموعه‌ای متفاوت از داده‌هایی که برای ساختن درخت اصلی استفاده شده است، در نتیجه، درخت هرس شده ممکن است به خوبی به داده‌های جدید تعمیم ندهد.

دلیل دوم آن است که اگر یک مجموعه داده جداگانه برای هرس استفاده کنیم، ارزیابی دقیق عملکرد درخت هرس شده دشوار می‌شود. این به این دلیل است که درخت بر روی مجموعه‌ای متفاوت از داده‌ها نسبت به داده‌هایی که در نهایت به آن اهمیت می‌دهیم بهینه شده است. اگر بخواهیم عملکرد درخت هرس شده را روی داده‌های جدید آزمایش کنیم، باید از مجموعه‌های داده‌های جداگانه دیگری استفاده کنیم که می‌تواند فرآیند ارزیابی را پیچیده‌تر و مستعد خطا کند.

امیر بهنام - ۹۸۳۱۱۳۳

۲-

ابتدا مقدار Entropy را حساب می‌کنیم:

$$Entropy(D) = -\frac{۲}{۵}\log\frac{۲}{۵} - \frac{۳}{۵}\log\frac{۳}{۵} = ۰.۵۲۸ + ۰.۴۴۲ = ۰.۹۷$$

سپس Entropy یعنی شوزها را محاسبه می‌کنیم و در ادامه Gain هر یک را بدست می‌آوریم:

$$Entropy(\text{رنگ لباس}) = \frac{۱}{۵}\underbrace{\left(-\frac{۱}{۱}\log\frac{۱}{۱} - ۰\right)}_{۰} + \frac{۲}{۵}\underbrace{\left(-\frac{۲}{۲}\log\frac{۲}{۲} - ۰\right)}_{۰} + \frac{۲}{۵}\underbrace{\left(-\frac{۱}{۲}\log\frac{۱}{۲} - \frac{۱}{۲}\log\frac{۱}{۲}\right)}_{۱}$$

$$\Rightarrow Entropy(\text{رنگ لباس}) = ۰.۴$$

$$Entropy(\text{قد}) = \frac{۲}{۵}\left(-\frac{۱}{۲}\log\frac{۱}{۲} - \frac{۱}{۲}\log\frac{۱}{۲}\right) + \frac{۳}{۵}\left(-\frac{۱}{۳}\log\frac{۱}{۳} - \frac{۲}{۳}\log\frac{۲}{۳}\right) = ۰.۹۵۹$$

$$Entropy(\text{وزن}) = \frac{۳}{۵}\underbrace{\left(-\frac{۲}{۳}\log\frac{۲}{۳} - \frac{۱}{۳}\log\frac{۱}{۳}\right)}_{۰.۵۵۹} + \frac{۲}{۵}\underbrace{\left(-\frac{۱}{۲}\log\frac{۱}{۲} - \frac{۱}{۲}\log\frac{۱}{۲}\right)}_{۰.۴} = ۰.۹۵۹$$

برای قد اعداد را به دسته تقسیم کردیم: ۱۶۰، ۱۶۲، ۱۶۵ ← دسته ۱

۱۷۰، ۱۷۲ ← دسته ۲

$$Gain(\text{رنگ لباس}) = Entropy(D) - Entropy(\text{رنگ لباس}) = ۰.۹۷ - ۰.۴ = ۰.۵۴ \rightarrow \text{Root}$$

$$Gain(\text{قد}) = Entropy(D) - Entropy(\text{قد}) = ۰.۹۷ - ۰.۹۵۹ = ۰.۰۱۱$$

$$Gain(\text{وزن}) = Entropy(D) - Entropy(\text{وزن}) = ۰.۹۷ - ۰.۹۵۹ = ۰.۰۱۱$$

از آنجایی که Gain(رنگ لباس) از بقیه بیشتر بود آن را به عنوان Root انتخاب می‌کنیم.

با توجه به جدول داده شده، می‌توان برای رنگ‌های قرمز و آبی مشخص کرد که آیا دعوت می‌شوند یا خیر. برای آبی هر ۲ بله است پس دعوت می‌شوند و برای قرمز فقط یک گزینه داریم که خیر است پس دعوت نمی‌شوند.

امیربهام - ۹۸۳۱۱۳۳

۲- ادامه)

برای رنگ سبز Entropy آن را محاسبه می‌کنیم:

Entropy (رنگ لباس ، سبز) = $-\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} = 1$

و در ادامه ستون‌های قد و وزن مربوط به آن را محاسبه می‌کنیم:

Entropy (وزن ، ۵) = $\frac{1}{2}$ ($\underbrace{\text{بله ۱ ، نه ۰}}_{0}$) + $\frac{1}{2}$ ($\underbrace{\text{بله ۱ ، نه ۰}}_{0}$) = ۰

Entropy (قد ، ۵) = $\frac{1}{2}$ (بله ۱ ، نه ۰) + $\frac{1}{2}$ (بله ۱ ، نه ۰) = ۰

از اینجا به بعد Entropy برای تک تک متغیرها در نتیجه Gain آنها نیز برابر می‌شوند. پس برای "سبز" می‌توانیم هر کدام را انتخاب کنیم و فرق نمی‌کند.

امیرحسام - ۹۸۳۱۳۳

۳-

داده‌ای باید به کدام class نسبت داده شود

$x = ($ معدل = عالی , مطالعه = بله , حضور = خیر $)$

class:

c_1: pass = 'yes'

c_2: pass = 'no'

$P(\text{pass} = \text{'yes'}) = \frac{4}{6} = \frac{2}{3}$

$P(\text{pass} = \text{'no'}) = \frac{2}{6} = \frac{1}{3}$

$P(\text{معدل} = \text{عالی} \mid \text{pass} = \text{'yes'}) = \frac{2}{4} = \frac{1}{2}$

$P(\text{مطالعه} = \text{بله} \mid \text{pass} = \text{'yes'}) = \frac{3}{4}$

$P(\text{حضور} = \text{خیر} \mid \text{pass} = \text{'yes'}) = \frac{1}{4}$

$P(\text{معدل} = \text{عالی} \mid \text{pass} = \text{'No'}) = \frac{0}{2}$

$P(\text{مطالعه} = \text{بله} \mid \text{pass} = \text{'No'}) = \frac{0}{2}$

$P(\text{حضور} = \text{خیر} \mid \text{pass} = \text{'No'}) = \frac{2}{2} = 1$

$\xrightarrow{\text{نرمالایز}}$ $\frac{1}{3}$, $\frac{1}{3}$, 1

$P(\text{معدل} = \text{عالی} , \text{مطالعه} = \text{بله} , \text{حضور} = \text{خیر} \mid \text{pass} = \text{'yes'}) = \frac{1}{2} \times \frac{3}{4} \times \frac{1}{4} = \frac{3}{32}$

$P(\text{معدل} = \text{عالی} , \text{مطالعه} = \text{بله} , \text{حضور} = \text{خیر} \mid \text{pass} = \text{'No'}) = \frac{1}{3} \times \frac{1}{3} \times 1 = \frac{1}{9}$

$P(\text{معدل} = \text{عالی} , \text{مطالعه} = \text{بله} , \text{حضور} = \text{خیر} \mid \text{pass} = \text{'yes'}) = \frac{3}{32} \times \frac{2}{3} = \frac{1}{16}$

$P(\text{معدل} = \text{عالی} , \text{مطالعه} = \text{بله} , \text{حضور} = \text{خیر} \mid \text{pass} = \text{'No'}) = \frac{1}{9} \times \frac{1}{3} = \frac{1}{27}$

پس با توجه به اعداد به دست آمده می‌توان گفت x مطلق به کلاس (pass = 'yes') است.

امیر بهنام _ ۹۸۳۱۱۴۳

_۴

۱: Imblanced data sets ← در مواردی که توزیع کلاس ها در مجموعه داده ها به شدت نامتعادل است، دقت می‌تواند گمراه‌کننده باشد. به عنوان مثال، اگر مجموعه داده‌ای با ۹۵٪ نمونه متعلق به کلاس A و تنها ۵٪ متعلق به کلاس B داشته باشیم، مدلی که همیشه A را پیش‌بینی می‌کند به دقت ۹۵٪ دست می‌یابد. با این حال، چنین مدلی در سناریوهای دنیای واقعی مفید نخواهد بود.

۲: cost sensitive classification ← در برخی از برنامه‌ها طبقه‌بندی اشتباه کلاس‌های خاص می‌تواند پرهزینه‌تر از سایرین باشد. در چنین مواردی، دقت هزینه طبقه‌بندی اشتباه را در نظر نمی‌گیرد و بنابراین معیار مناسبی نباشد. به عنوان مثال، در یک سناریوی تشخیص پزشکی، false negative (یعنی عدم شناسایی یک بیماری) می‌تواند پرهزینه‌تر از false positive باشد (یعنی شناسایی بیماری در صورت عدم وجود آن).

۳: Multi class classification ← در مواردی که بیش از ۲ کلاس برای پیش‌بینی وجود دارد، ممکن است دقت تصویر کاملی از عملکرد مدل ارائه نکند. این بدین دلیل است که دقت یک عدد واحد است که نشان می‌دهد درصد موارد طبقه‌بندی صحیح است. با این حال، نشان نمی‌دهد که کدام کلاس‌ها به عنوان کلاس‌های دیگر به اشتباه طبقه‌بندی شده‌اند. بنابراین، معیارهای اضافی مانند دقت، یادآوری و امتیاز F1 ممکن است برای ارزیابی عملکرد یک مدل طبقه‌بندی چندطبقه مورد نیاز باشد.

۴: overfitting ← اگر یک مدل به داده‌های آموزشی بیش از حد برازش شود، ممکن است در مجموعه آموزشی به دقت بالایی دست یابد اما در داده‌های دیده نشده عملکرد ضعیفی داشته باشد. بنابراین، دقت ممکن است معیار قابل اعتمادی برای ارزیابی چنین مدل‌هایی نباشد.

۵: Noise ← برخی از مجموعه‌های داده ممکن است دارای ویژگی‌های پرسروصدا یا نامربوط باشند که به کار پیش‌بینی کمک نمی‌کنند. در چنین مواردی، دقت ممکن است مستقیم از عملکرد مدل ایجاد نکند، زیرا مدل ممکن است به جای یادگیری الگوهای اساسی در داده‌ها، به سادگی یادبگیرد که این ویژگی‌های نویزدار را نادیده بگیرد.

امیربهنام _ ۹۸۳۱۱۳۳

۵-

پس از استفاده از روش 10 fold cross validation برای انتخاب پارامتر بهتر و تیمرد برای انتخاب مدل نمایه و تخمین خطا استفاده از کل مجموعه داده امین نه... فقط قسمت آموزش) برای آموزش تیمدل با مقدار به انتخابی است.

سپس می‌توان از این مدل برای پیش‌بینی مجموعه‌ای از داده‌ها استفاده کرد که holdset مورد استفاده قرار نگرفته. طول فرآیند cross validation اعمال می‌شود.

مجموعه‌ی holdout باید نمونه‌ای صرف از داده‌هایی باشد که مدل نمایه روی آن هنگامی که مدل نمایه آموزش داده شد می‌توان عملکرد آن را در مجموعه نگهدارنده با استفاده از معیارهای مناسب مانند accuracy ، precision ، recall یا F1 score ارزیابی کرد.

نرخ خطای تخمین را می‌توان بر اساس معیارهای عملکرد به دست آمده در مجموعه نگهدارنده محاسبه کرد.

یک روش دیگر می‌تواند استفاده از sampling bootstrap برای تخمین پذیری عملکرد مدل است. در این رویکرد، چندین نمونه bootstrap از مجموعه داده اصلی ترسیم می‌شود و هر نمونه برای برازش مدل با مقدار به انتخاب شده استفاده می‌شود. تخمین‌های خطای به دست آمده از مدل‌های چندگانه را می‌توان میانگین کرد تا تخمین پایدارتری از عملکرد مدل ارائه شود.

۱) به طور کلی انتخاب روش برای انتخاب مدل نمایه و تخمین خطا به شکل قابل میزان داده‌های موجود و عوامل دیگری مانند منابع محاسباتی بستگی دارد.

امیرپنام - ۹۸۳۰۱۱۳۳

۶-

مورد ۱: لازم نیست نرخ خطای طبقه‌بندی‌کننده ترکیبی در داده‌های آموزشی اصلی قبل از توقف فرآیند یادگیری Boosting باشد.

این به این دلیل است که هدف به حداقل رساندن میزان خطای کل مجموعه داده‌ها از جمله داده‌های آموزشی و آزمایشی است. نرخ خطای طبقه‌بندی‌کننده ترکیبی در داده‌های آموزشی ممکن است با اضافه شدن طبقه‌بندی‌کننده‌های ضعیف به کاهش ادامه دهد، اما ممکن است نرخ خطا در داده‌های آزمایشی به دلیل برازش بیش از حد شروع به افزایش کند. بنابراین، میزان خطا در داده‌های تست، شاخص بهتری برای توقف یادگیری Boosting است. پس یادگیری را متوقف نمی‌کنیم.

مورد ۲: اگر میزان خطای طبقه‌بندی‌کننده ضعیف روی داده‌های تمرین وزن‌دار باشد، می‌توانیم آموزش را متوقف کنیم زیرا به طبقه‌بندی کامل روی داده‌های آموزشی دست یافته‌ایم. این به این دلیل است که وزن‌های اختصاص داده شده به نمونه‌های طبقه‌بندی اشتباه در هر تکرار افزایش می‌یابد و وقتی میزان خطا صفر می‌شود به این معنی است که همه نمونه‌ها به درستی و کاملاً طبقه‌بندی شده‌اند و تقویت بیشتر عملکرد مجموعه تمرین را بهبود نمی‌بخشد.

امیرپناه - ۹۸۳۱۱۳۳

۷-

الف)

هرچه C کوچکتر باشد margin بیشتر است و هر چه C بزرگتر باشد margin نیز کمتر خواهد بود و در اینجا چون C را بسیار بزرگ در نظر گرفته ایم، پس margin کم خواهیم داشت.

ب) نقطه (۲,۲) - اگر این نقطه را حذف کنیم فاصله از x=1 تا x=4 در نظر باعث افزایش فاصله می‌شود و خطی که به دست می‌آید با حالت الف در تناقض است. باید بگوییم که در حالت این نقاط (۴,۱) و (۴,۳) - با حذف این نقاط فاصله را از x=2 تا x=5 باید در نظر گرفت که در این حالت نیز با افزایش فاصله مواجه می‌شویم و در نتیجه خطی که در حالت الف به دست آمده با این حالت در تناقض خواهد بود.

نقاط (۱,۴)، (۴,۳) و (۲,۲) - با این حالت فاصله از x=1 تا x=5 می‌شود و با افزایش فاصله مواجه می‌شویم و در نتیجه خطی که رسم می‌شود با حالت الف در تناقض خواهد بود.

به طور کلی عبارت هایی که در شکل با دایره مشخص شده اند.

امید بهنام - ۹۸۳۰۱۳۳

۸-

الف) درست - الگوریتم بایز ساده نمی تواند وابستگی بین متغیرها را محاسبه کند زیرا تمامی متغیرها مستقل از یکدیگر هستند و می توان گفت که متغیرهای ورودی هیچ گونه وابستگی به همدیگر ندارند.

ب) درست - هنگامی که یک درخت تصمیم به سمت یک درخت کامل حرکت می کند، تمایل دارد که داده ها را بیش از حد برازش کند و به این معنی که بیش از حد پیچیده می شود و نویز یا تغییرات تصادفی در داده ها را به جای روابط و الگوهای زیربنایی ثبت می کند. بنابراین با پیچیده شدن درخت، احتمال پوشاندن نویز افزایش می یابد.

ج) غلط - در روش k نزدیکترین همسایه، اگر k=1 باشد، الگوریتم در مقابل داده های نویز مقاومت کمتری نسبت به k=5 دارد. وقتی k=1، الگوریتم به سادگی نزدیکترین نقطه را به عنوان پیش بینی انتخاب می کند که می تواند تحت تاثیر نقاط نویز نزدیک به نقطه پرس و جو قرار گیرد. با این حال، زمانی که k=5، الگوریتم 5 نزدیکترین نقطه را در نظر می گیرد و رای میانگین یا اکثریت آنها را می گیرد که تاثیر نقاط پرسر و صدا را کاهش می دهد. بنابراین، افزایش k به طور کلی استحکام الگوریتم را در برابر نویز را بهبود می بخشد.

امیرپنام - ۹۸۳۰۱۳۳

۹- برای سیستمی که برای تشخیص راننده در خودرو طراحی شده است، بهترین معیار ارزیابی Recall است.

Recall معیاری است که نشان می‌دهد مدل چقدر می‌تواند موارد مثبت را شناسایی کند. به عبارت دیگر نسبت موارد مثبت واقعی را که به درستی توسط مدل مثبت شناسایی شده‌اند، اندازه‌گیری می‌کند. برای این سناریو تشخیص تمام موارد خستگی بسیار مهم است و تشخیص ندادن یک مورد از خستگی موجب عواقب جبران ناپذیر می‌شود. بنابراین داشتن ارزش Recall تضمین می‌کند که سیستم قابل به تشخیص تمام موارد خستگی و به حداقل رساندن آن است.

Accuracy، پیش بینی های صحیح کلی مدل را اندازه‌گیری می‌کند اما بین موارد مثبت واقعی و منفی واقعی تمایز قائل نمی‌شود.

precision نسبت مثبت های واقعی را در بین تمام مثبت های پیش بینی شده اندازه‌گیری می‌کند اما منفی های کاذب را در نظر نمی‌گیرد.

Loss value میزان خطای مدل را اندازه‌گیری می‌کند اما بینشی در مورد توانایی سیستم برای تشخیص تمام موارد خستگی راننده ارائه نمی‌دهد.

با توجه به دلایل فوق Recall بهترین و مناسب‌ترین معیار برای ارزیابی است.

امیرپنام - ۹۸۳۱۱۳۳

۱۵-

شاخص دیگری وجود دارد که به جای آنتروپی می‌توان از آن برای ساخت درخت تصمیم استفاده کرد که این شاخص Gini impurity index نام دارد.

Gini impurity index، احتمال طبقه‌بندی نادرست عنصری که به طور تصادفی انتخاب شده است را می‌سنجد. اگر قرار باشد به طور تصادفی بر اساس توزیع برچسب‌ها در نود برچسب‌گذاری شود. محدوده آن بین ۰ تا ۱ است، جایی که ۰ نشان‌دهنده طبقه‌بندی کامل (همه عناصر متعلق به یک کلاس هستند) و 1 نشان‌دهنده حداکثر ناخالصی (عناصر به طور مساوی در همه کلاس‌ها توزیع شده‌اند).

perfect classification

maximum impurity

فرمول آن نیز به صورت زیر است:

$$\text{Gini index} = 1 - \sum (p_i)^2$$

که در آن p_i نشان‌دهنده احتمال یک عنصر متعلق به کلاس i ام است.

تفاوت اصلی بین Gini impurity index و Entropy در این است که شاخص جینی به دلیل تمایل به حداقل رساندن طبقه‌بندی اشتباه تمایل به ایجاد پارتیشن‌های بزرگتر از آنتروپی دارد. به عبارت دیگر، شاخص جینی گره‌های تقسیم‌کننده را ترجیح می‌دهد که اکثریت یک کلاس را دارند. از سوی دیگر، آنتروپی قطعیت داده‌ها را اندازه‌گیری می‌کند و تقسیم‌های متعادل‌تر را ترجیح می‌دهد.

اگر شاخص جینی بالا باشد، نشان می‌دهد که یک نود ناخالصی بالایی دارد، به این معنی که اختلاط قابل توجهی از کلاس‌های مختلف در آن نود وجود دارد. اگر شاخص جینی پایین باشد، نشان می‌دهد که یک نود ناخالصی پایین دارد و اکثر نمونه‌ها متعلق به یک کلاس واحد هستند.

یک درخت تصمیم با استفاده از شاخص جینی ساخته شده است با هدف به حداقل رساندن شاخص ناخالصی جینی در تمام شاخه‌های درخت است.

امیرنام - ۹۸۳۰۱۱۳۳

-۱۱

الف) simple Linear regression و multiple Linear Regression هر ۲ تکنیک هایی هستند که در مدل سازی آماری برای تجزیه و تحلیل رابطه بین یک متغیر وابسته و یک یا چند متغیر مستقل استفاده می شوند.

تفاوت اصلی بین simple Linear regression و multiple Linear Regression، تعداد متغیرهای مستقل استفاده شده در مدل است. در simple Linear regression، تنها یک متغیر مستقل وجود دارد در حالی که در multiple Linear Regression، ۲ یا چند متغیر مستقل وجود دارد.

تفاوت مهم دیگر تفسیر ضرایب در هر مدل است. در simple Linear regression، ضرایب نشان دهنده تغییر متغیر وابسته برای افزایش یک واحد در متغیر مستقل است. در multiple Linear Regression، ضرایب نشان دهنده تغییر متغیر وابسته زمانی هستند که همه متغیرهای مستقل دیگر ثابت نگه داشته شوند.

علاوه بر این، در multiple Linear Regression امکان بررسی اثرات منحصر به فرد هر متغیر مستقل بر متغیر وابسته بیشتر کنترل اثرات سایر متغیرهای مستقل را فراهم می کند. این در simple Regression امکان پذیر نیست زیرا تنها یک متغیر مستقل وجود دارد.

همچنین از نظر complexity، multiple Linear Regression پیچیده تر از simple Linear Regression است زیرا شامل تجزیه و تحلیل اثر متغیرهای مستقل چندگانه بر روی یک متغیر وابسته است و از نظر overfitting نیز multiple Linear Regression مستعد بیش برازش است به خصوص زمانی که متغیرهای مستقل زیادی وجود دارد. این ممکن است در داده های آموزشی عملکرد خوبی داشته باشد اما داده های جدید و دیده نشده ضعیف عمل کند. نسبت به simple Linear Regression بدان معناست که مدل علیرغم تفاوت های آنها، هر ۲ شباهت هایی نیز دارند. هر ۲ مدل از مفروضات اساسی یکسان از جمله خطی بودن، استقلال، نرمال بودن و همسانی استفاده می کنند. علاوه بر این هر ۲ مدل می توانند پیش بینی نتیجه بر اساس مقادیر متغیرهای مستقل استفاده کنند. همچنین از هر ۲ روش می توان برای شناسایی outlier و مشاهدات تاثیرگذار که ممکن است به کیفیت برازش تاثیر بگذارد استفاده کرد و بر این مورد می توان نیز اشاره کرد که اگر همبستگی بالایی بین متغیرهای مستقل وجود داشته باشد، هر ۲ روش می توانند از چند خطی بودن رنج ببرند.

امیرهام - ۹۸۳۱۱۳۳

۱۱-

ب)

Lasso Regression <- این روش از جریمه L1 (Penalty L1) در اندازه ضرایب استفاده می‌کند. این بدان معنی است که یک عبارت جریمه برابر با مقدار مطلق ضرایب ضرب در یک ثابت به تابع هزینه در اقل مربعات معمولی (OLS) اضافه می‌کند. جریمه L1 ضرایب ویژگی‌های کمتر مهم را به صفر کاهش می‌دهد و به طور موثر انتخاب ویژگی را انجام می‌دهد. بنابراین، Lasso Regression زمانی می‌تواند مفید باشد که تعداد زیادی ویژگی داشته باشیم و گمان می‌کنیم که تعداد کمی از آنها برای پیش‌بینی متغیر هدف ضروری هستند.

Ridge Regression <- این روش از جریمه L2 (Penalty L2) در اندازه ضرایب استفاده می‌کند. این بدان معنی است که یک عبارت جریمه برابر با مجذور ضرایب در یک ثابت به تابع هزینه OLS اضافه می‌کند. جریمه L2 تمام مقادیر ضرایب را به سمت صفر کوچک می‌کند اما برخلاف Lasso Regression، ضرایب دقیق صفر را به همراه ندارد. در عوض، مقادیر ضرایب کوچکتر تاثیر کمتری بر متغیر هدف ایشان می‌گذارند و مقادیر بزرگتر نشان‌دهنده تاثیر بیشتر است. Ridge Regression زمانی می‌تواند مفید باشد که ویژگی‌های همبسته زیادی داشته باشیم، زیرا می‌تواند تاثیر هم‌خطی چندگانه را بر عملکرد مدل کاهش دهد.

به طور خلاصه، Lasso Regression از منظم‌سازی L1 برای انجام انتخاب ویژگی استفاده می‌کند در حالی که Ridge Regression از منظم‌سازی L2 برای کوچک کردن مقادیر ضرایب به سمت صفر (اما نه حذف آنها) استفاده می‌کند.

امیر بهام - ۹۸۳۱۱۳۳

ال
ج)

$$X = \begin{bmatrix} 1 & 42 \\ 1 & 74 \\ 1 & 48 \\ 1 & 35 \\ 1 & 56 \\ 1 & 26 \\ 1 & 60 \end{bmatrix} \qquad Y = \begin{bmatrix} 98 \\ 130 \\ 120 \\ 88 \\ 182 \\ 80 \\ 135 \end{bmatrix}$$

$$\Rightarrow X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 42 & 74 & 48 & 35 & 56 & 26 & 60 \end{bmatrix} \begin{bmatrix} 1 & 42 \\ 1 & 74 \\ 1 & 48 \\ 1 & 35 \\ 1 & 56 \\ 1 & 26 \\ 1 & 60 \end{bmatrix}$$

$$\Rightarrow X^T X = \begin{bmatrix} 7 & 341 \\ 341 & 18181 \end{bmatrix}$$

$$\Rightarrow X^T Y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 42 & 74 & 48 & 35 & 56 & 26 & 60 \end{bmatrix} \begin{bmatrix} 98 \\ 130 \\ 120 \\ 88 \\ 182 \\ 80 \\ 135 \end{bmatrix}$$

$$\Rightarrow X^T Y = \begin{bmatrix} 833 \\ 49348 \end{bmatrix}$$

$$\begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \begin{bmatrix} 7 & 341 \\ 341 & 18181 \end{bmatrix}^{-1} \begin{bmatrix} 833 \\ 49348 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \frac{1}{10986} \begin{bmatrix} 18181 & -341 \\ -341 & 7 \end{bmatrix} \begin{bmatrix} 833 \\ 49348 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} = \frac{1}{10986} \begin{bmatrix} 449705 \\ 16583 \end{bmatrix}$$

$$\Rightarrow Y = \frac{449705}{10986} + \frac{449705}{10986} X \Rightarrow Y = 40.469 + 1.5059 X$$

$$X = 40 \Rightarrow Y = 105.82 \rightarrow$$ فشارخون

امیر بهنام-9831133- تمرین دوم داده کاوی -قسمت پیاده سازی

آماده سازی داده ها:

در ابتدا کتاب خانه های مورد نیاز را import می کنیم و سپس فایل Csv را به صورت زیر بار گذاری می کنیم:

```
df = pd.read_csv(r'C:\Users\dr. Behnam\Videos\HousePrice\housePrice.csv' , names=['Area', 'Room', 'Parking', 'Warehouse', 'Elevator' , 'Address'
```

در ادامه با استفاده از isna().sum() تعداد ردیف های که دارای NaN هستند را برای هر ستون پیدا می کنیم:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split



df = pd.read_csv(r'C:\Users\dr. Behnam\Videos\HousePrice\housePrice.csv' , names=['Area', 'Room', 'Parking', 'Warehous

print(df.isna().sum())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python Debug Console

```
PS C:\Users\dr. Behnam\Pictures>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\dr. Behnam\.vscode\extensions
s-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '50688' '--' 'c:\Users\dr. Behnam\Pictures\HousePrices.py'
Area          0
Room          0
Parking       0
Warehouse     0
Elevator      0
Address      23
Price         0
dtype: int64
```

سپس با استفاده از دستور df.shape[0] تعداد سطر ها را قبل از حذف مشخص می کنیم. سپس با استفاده از دستور df.dropna(how ='any’) سطر های که حاوی مقدار NaN هستند را حذف می کنیم ودر نهایت با استفاده از دستور df.shape[0] تعداد سطر ها را بعد از حذف را مشخص می کنیم:

```python
number_of_rows_before_deletion = df.shape[0]
df = df.dropna(how ='any')
number_of_rows_after_deletion = df.shape[0]
print(df.isna().sum())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python Debug Console

```
:\Users\dr. Behnam\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '50714
r. Behnam\Pictures\HousePrices.py'
Area         0
Room         0
Parking      0
Warehouse    0
Elevator     0
Address      0
Price        0
dtype: int64
PS C:\Users\dr. Behnam\Pictures>
```

Quartile های pricelevel:

```python
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
quartiles = df['Price'].quantile([0, 0.25, 0.5, 0.75, 1])
labels = ['cheap', 'underMean', 'upperMean', 'expensive']
df['priceLevel'] = pd.qcut(df['Price'], q=4, labels=labels)

print("Quartiles of Price column:\n", df['Price'].describe())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Quartiles of Price column:
 count    3.450000e+03
mean     5.375563e+09
std      8.125918e+09
min      3.600000e+06
25%      1.419250e+09
50%      2.900000e+09
75%      6.000000e+09
max      9.240000e+10
Name: Price, dtype: float64
```

در کد بالا , ستون Price را به چهار quartile تقسیم کردیم و در ادامه به هر کدام label مورد نظر را می هیم و یک ستون به نام pricelevel می سازیم و در نهایت با استفاده از تابع describe نتایج مورد نظر را چاپ می کنیم.

و در ادامه داده ها به صورت زیر می باشند:

```python
print("\nUpdated dataframe:\n", df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Updated dataframe:
       Area  Room  Parking  Warehouse  Elevator             Address       Price priceLevel
0        63     1     True       True      True             Shahran  1850000000  underMean
1        60     1     True       True      True             Shahran  1850000000  underMean
2        79     2     True       True      True              Pardis   550000000      cheap
3        95     2     True       True      True       Shahrake Qods   902500000      cheap
4       123     2     True       True      True      Shahrake Gharb  7000000000  expensive
...     ...   ...      ...        ...       ...                 ...         ...        ...
3468     86     2     True       True      True  Southern Janatabad  3500000000  upperMean
3469     83     2     True       True      True            Niavaran  6800000000  expensive
3470     75     2    False      False     False              Parand   365000000      cheap
3471    105     2     True       True      True              Dorous  5600000000  upperMean
3472     82     2    False       True      True              Parand   360000000      cheap

[3450 rows x 8 columns]
PS C:\Users\dr. Behnam>
```

با استفاده از Label Encoding نتایج به صویت زیر می باشد:

```python
selected_str_cols = ['Parking', 'Warehouse', 'Elevator' , 'Address']

le_Parking = LabelEncoder()
le_Warehouse = LabelEncoder()
le_Elevator = LabelEncoder()
le_Address = LabelEncoder()

df['Parking'] = le_Parking.fit_transform(df['Parking'])
df['Warehouse'] = le_Warehouse.fit_transform(df['Warehouse'])
df['Elevator'] = le_Elevator.fit_transform(df['Elevator'])
df['Address'] = le_Address.fit_transform(df['Address'])

print(df[selected_str_cols])
print("\nUpdated dataframe For Label Encoding:\n", df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
[3450 rows x 8 columns]
PS C:\Users\dr. Behnam>  c:; cd 'c:\Users\dr. Behnam'; & 'C:\Users\dr. Behnam\AppData\Local\Pro
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\deb
.py'
      Parking  Warehouse  Elevator  Address
0           1          1         1      156
1           1          1         1      156
2           1          1         1      117
3           1          1         1      152
4           1          1         1      150
...       ...        ...       ...      ...
3468        1          1         1      163
3469        1          1         1      105
3470        0          0         0      115
3471        1          1         1       39
3472        0          1         1      115
```

و همچنین داده ها به صورت کلی به شکل زیر است:

```python
selected_str_cols = ['Parking', 'Warehouse', 'Elevator' , 'Address']

le_Parking = LabelEncoder()
le_Warehouse = LabelEncoder()
le_Elevator = LabelEncoder()
le_Address = LabelEncoder()

df['Parking'] = le_Parking.fit_transform(df['Parking'])
df['Warehouse'] = le_Warehouse.fit_transform(df['Warehouse'])
df['Elevator'] = le_Elevator.fit_transform(df['Elevator'])
df['Address'] = le_Address.fit_transform(df['Address'])

print(df[selected_str_cols])
print("\nUpdated dataframe For Label Encoding:\n", df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
      Area  Room  Parking  Warehouse  Elevator  Address       Price priceLevel
0       63     1        1          1         1      156  1850000000  underMean
1       60     1        1          1         1      156  1850000000  underMean
2       79     2        1          1         1      117   550000000      cheap
3       95     2        1          1         1      152   902500000      cheap
4      123     2        1          1         1      150  7000000000  expensive
...    ...   ...      ...        ...       ...      ...         ...        ...
3468    86     2        1          1         1      163  3500000000  upperMean
3469    83     2        1          1         1      105  6800000000  expensive
3470    75     2        0          0         0      115   365000000      cheap
3471   105     2        1          1         1       39  5600000000  upperMean
3472    82     2        0          1         1      115   360000000      cheap

[3450 rows x 8 columns]
```

نرمال سازی داده ها به صورت زیر است:

```python
scaler = StandardScaler()
scaler.fit(df.get(['Area', 'Room', 'Parking', 'Warehouse', 'Elevator' , 'Address' , 'Price']))
scaled_data = scaler.transform(df.get(['Area', 'Room', 'Parking', 'Warehouse', 'Elevator' , 'Address' , 'Price']))
scaler.fit(scaled_data)
df['Area'] = scaled_data[:, 0]
df['Room'] = scaled_data[:, 1]
df['Parking'] = scaled_data[:, 2]
df['Warehouse'] = scaled_data[:, 3]
df['Elevator'] = scaled_data[:, 4]
df['Address'] = scaled_data[:, 5]
df['Price'] = scaled_data[:, 6]
print(df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python Debug Consol

```
          Area      Room   Parking  Warehouse  Elevator   Address     Price priceLevel
0    -0.631533 -1.422380  0.424135   0.304647  0.519406  1.013652 -0.433929  underMean
1    -0.674673 -1.422380  0.424135   0.304647  0.519406  1.013652 -0.433929  underMean
2    -0.401453 -0.106774  0.424135   0.304647  0.519406  0.238524 -0.593934      cheap
3    -0.171372 -0.106774  0.424135   0.304647  0.519406  0.934151 -0.550548      cheap
4     0.231268 -0.106774  0.424135   0.304647  0.519406  0.894401  0.199937  expensive
...        ...       ...       ...        ...       ...       ...       ...        ...
3468 -0.300793 -0.106774  0.424135   0.304647  0.519406  1.152777 -0.230846  upperMean
3469 -0.343933 -0.106774  0.424135   0.304647  0.519406  0.000023  0.175321  expensive
3470 -0.458973 -0.106774 -2.357739  -3.282491 -1.925275  0.198774 -0.616704      cheap
3471 -0.027572 -0.106774  0.424135   0.304647  0.519406 -1.311732  0.027624  upperMean
3472 -0.358313 -0.106774 -2.357739   0.304647  0.519406  0.198774 -0.617320      cheap

[3450 rows x 8 columns]
PS C:\Users\dr. Behnam>
```

در ادامه داده ها را به 80 و 20 تقیسم می کنیم:

```python

X = df.drop(['priceLevel'], axis=1)
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS C:\Users\dr. Behnam>  c:; cd 'c:\Users\dr. Behnam'; & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\P
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher'
.py'
PS C:\Users\dr. Behnam>  c:; cd 'c:\Users\dr. Behnam'; & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\P
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher'
.py'
Training set shape: (2760, 7)
Testing set shape: (690, 7)
PS C:\Users\dr. Behnam>
```

در اینجا ستون PriceLevel را حذف کردیم و ستون Price را به عنوان برچسپ درنظر گرفتیم(می توانستیم بر عکس نیز در نظر بگیریم و اینجا برای نمونه به این شکل انجام شده) و در ادامه 80٪ را برای آموزش و 20٪ را برای تست انتخاب کردیم و سپس آن را چاپ کردیم.

رگرسیون:

رگرسیون خطی به صورت زیر است:

```python
#Linear Regression

model = LinearRegression()
model.fit(X_train, y_train)
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
train_r2, test_r2 = r2_score(y_train, y_train_pred), r2_score(y_test, y_test_pred)
train_mse, test_mse = mean_squared_error(y_train, y_train_pred), mean_squared_error(y_test, y_test_pred)

print(f'Training set:\nR-squared: {train_r2:.3f}\nMSE: {train_mse:.3f}')
print(f'Testing set:\nR-squared: {test_r2:.3f}\nMSE: {test_mse:.3f}')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python Debug

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\dr. Behnam\.vscc
python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '55936' '--' 'c:\Users\dr. Behnam\Official.py'
Training set:
R-squared: 0.540
MSE: 0.443
Testing set:
R-squared: 0.532
MSE: 0.536
PS C:\Users\dr. Behnam>
```

این قطعه کد تحلیل رگرسیون خطی را روی یک مجموعه داده ما با متغیرهای ورودی X_train و متغیر خروجی y_train انجام می دهد. تابع LinearRegression() نمونه ای از مدل رگرسیون خطی را ایجاد می کند. متد fit() . با استفاده از X_train و y_train به عنوان آرگومان برای آموزش مدل فراخوانی می شود. سپس از مدل آموزش‌دیده برای پیش‌بینی پاسخ‌های مجموعه آموزشی و آزمایشی با استفاده از روش predict() استفاده می کنیم که مقادیر پیش‌بینی‌شده y_train و y_test را برمی‌گرداند.

سپس ضریب تعیین (R-squared) و میانگین مربعات خطا (MSE) برای مجموعه های آموزشی و آزمایشی به ترتیب با استفاده از توابع r2_score() و mean_squared_error() محاسبه می شود. R-squared میزان تناسب خط رگرسیون با داده ها را اندازه گیری می کند و MSE میانگین مجذور اختلاف بین مقادیر پیش بینی شده و واقعی را اندازه گیری می کند.در نهایت، مقادیر R-squared و MSE را برای مجموعه های آموزشی و آزمایشی چاپ می کنیم.

رگرسیون چند جمله ای 2 درجه:

```
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model = LinearRegression()
model.fit(X_train_poly, y_train)
y_train_pred = model.predict(X_train_poly)
y_test_pred = model.predict(X_test_poly)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)

print(f'Training set R-squared: {train_r2:.3f}, MSE: {train_mse:.3f}\n')
print(f'Testing set R-squared: {test_r2:.3f}, MSE: {test_mse:.3f}\n')
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c
python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '55991' '--' 'c:\Users\d
Training set R-squared: 0.674, MSE: 0.314

Testing set R-squared: 0.721, MSE: 0.320
```

در گام اول یک شی PolynomialFeatures با درجه 2 ایجاد می کنیم که با ایجاد ویژگی های چند جمله ای تا درجه 2، ماتریس ویژگی اصلی X_train را به یک ماتریس جدید X_train_poly تبدیل می کند. همان تبدیل با استفاده از روش transform() روی داده های تست اعمال می شود. از شیء PolynomialFeatures.

سپس، یک مدل رگرسیون خطی ایجاد می شود و بر روی داده‌های آموزشی تبدیل شده (X_train_poly) و متغیر هدف مربوطه (y_train) آموزش داده می‌شود. سپس این مدل برای پیش‌بینی متغیر هدف برای داده‌های آموزشی (y_train_pred) و داده‌های آزمون (y_test_pred) استفاده می کنیم.

در نهایت، عملکرد مدل با استفاده از دو معیار - R-squared (r2_score) و میانگین مربعات خطا (mean_squared_error) ارزیابی می شوند. این معیارها برای هر دو مجموعه آموزشی و آزمایشی محاسبه شده و به ترتیب در متغیرهای train_r2، test_r2، train_mse و test_mse ذخیره می‌شوند و در ادامه عبارت های مورد نیاز را چاپ می کنیم.

رگلیسیون چند جمله ای 3 درجه:

```python
#Polynomial Regression With 3 Degrees
poly = PolynomialFeatures(degree=3)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model = LinearRegression()
model.fit(X_train_poly, y_train)
y_train_pred = model.predict(X_train_poly)
y_test_pred = model.predict(X_test_poly)
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)

print(f'Training set R-squared: {train_r2:.3f}, MSE: {train_mse:.3f}\n')
print(f'Testing set R-squared: {test_r2:.3f}, MSE: {test_mse:.3f}\n')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
PS C:\Users\dr. Behnam>  c:; cd 'c:\Users\dr. Behnam'; & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\pyt
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56010' '--'
.py'
Training set R-squared: 0.754, MSE: 0.237

Testing set R-squared: 0.726, MSE: 0.314
```

توضیحات این قسمت همانند بخش قبل است با این تفاوت که از Degree =3 به جای 2 استفاده می کنیم.

دسته بندی:

درخت تصمیم:

```python
#Decision Tree
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
training_accuracy = accuracy_score(y_train, clf.predict(X_train))
testing_accuracy = accuracy_score(y_test, y_pred)
print("Training Accuracy:", training_accuracy)
print("Testing Accuracy:", testing_accuracy)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\dr. Beh
python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56186' '--' 'c:\Users\dr. Behnam\Offic
Training Accuracy: 0.9746376811594203
Testing Accuracy: 0.7695652173913043
PS C:\Users\dr. Behnam>
```

در اینجا یک طبقه‌بندی درخت تصمیم را بر روی برخی از داده‌های ورودی با استفاده از معیار آنتروپی برای اندازه‌گیری کیفیت هر تقسیم آموزش می‌دهیم. داده های ورودی به مجموعه های آموزشی و آزمایشی تقسیم می شوند، جایی که X_train و y_train به ترتیب ویژگی ها و برچسب های مجموعه آموزشی را نشان می دهند، در حالی که X_test نشان دهنده ویژگی های مجموعه آزمایشی است. هنگامی که طبقه‌بندی‌کننده با استفاده از clf.fit (X_train، y_train) آموزش داده شد، برای پیش‌بینی برچسب‌های مجموعه آزمایشی با استفاده از y_pred = clf.predict (X_test) استفاده می‌شود. دو خط کد بعدی دقت طبقه‌بندی کننده را در مجموعه‌های آموزشی و آزمایشی با استفاده از تابع accuracy_score از sklearn.metrics محاسبه می‌کنند و این مقادیر را چاپ می‌کنند. خروجی چاپ شده به ترتیب دقت طبقه بندی کننده را در مجموعه های آموزشی و آزمایشی نمایش می دهد.

جنگل تصادفی:

ابتدا یک شی طبقه‌بندی‌کننده clf را به‌عنوان RandomForestClassifier با معیار پارامتر روی «Entropy» مقداردهی اولیه می‌کنیم. این پارامتر کیفیت تقسیم در الگوریتم درخت تصمیم مورد استفاده توسط جنگل تصادفی

را مشخص می کند. مرحله بعدی برازش طبقه‌بندی کننده بر روی داده‌های آموزشی، X_train و y_train با استفاده از روش ()fit است. این مدل را بر روی داده های داده شده آموزش می دهد. سپس خروجی y_pred را روی داده های تست X_test با استفاده از روش ()predict پیش بینی می کند.

در ادامه با استفاده از تابع ()accuracy_score از scikit-learn، دقت مدل را روی داده‌های آموزش و آزمایش محاسبه می‌کنیم..

در نهایت، دقت آموزش و دقت تست چاپ می شود. این مقادیر به ترتیب نشان دهنده میزان عملکرد خوب مدل بر روی داده های آموزش و آزمایش هستند.

```
#Random Forest
clf = RandomForestClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
training_accuracy = accuracy_score(y_train, clf.predict(X_train))
testing_accuracy = accuracy_score(y_test, y_pred)
print("Training Accuracy:", training_accuracy)
print("Testing Accuracy:", testing_accuracy)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\
python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56186' '--' 'c:\Users\dr. Behna
Training Accuracy: 0.9746376811594203
Testing Accuracy: 0.7695652173913043
PS C:\Users\dr. Behnam>  c:; cd 'c:\Users\dr. Behnam'; & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Pyth
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '5
2.py'
Training Accuracy: 0.9746376811594203
Testing Accuracy: 0.7478260869565218
```

: KNN

```python
#KNN algorithm for k=3, 5, and 7
for k in [3, 5, 7]:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Evaluate the accuracy of the model on both training and testing sets
    train_acc = knn.score(X_train, y_train)
    test_acc = knn.score(X_test, y_test)
    print(f"k={k}, Training Accuracy: {train_acc:.3f}, Testing Accuracy: {test_acc:.3f}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\c
python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56237' '--' 'c:\Users\dr. Behnar
k=3, Training Accuracy: 0.840, Testing Accuracy: 0.699
k=5, Training Accuracy: 0.792, Testing Accuracy: 0.722
k=7, Training Accuracy: 0.771, Testing Accuracy: 0.720
```

پیاده سازی آن به این صورت است که ابتدا فهرستی [3، 5، 7] حاوی سه مقدار ممکن برای تعداد همسایه‌هایی که در الگوریتم KNN در نظر گرفته می‌شود، مقداردهی اولیه می‌کند. برای هر مقدار در این لیست، یک شی KNeighborsClassifier جدید با تعداد همسایه های مشخص شده ایجاد می شود و آن را با داده های آموزشی X_train و y_train متناسب می کند.

پس از برازش مدل، با استفاده از روش () score شی KNeighborsClassifier دقت آن را در مجموعه های آموزشی و آزمایشی ارزیابی می کند. دقت آموزش معیاری است از میزان تناسب مدل با داده‌های آموزشی، در حالی که دقت تست تخمینی از تعمیم مدل به داده‌های جدید و نادیده می‌دهد.

در نهایت، کد نتایج را خروجی می‌دهیم، مقدار k و همچنین دقت آموزش و آزمایش مربوطه را برای هر k چاپ می‌کند.

: SVM

```python
#SVM algorithm in linear mode
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)
train_acc_linear = svm_linear.score(X_train, y_train)
test_acc_linear = svm_linear.score(X_test, y_test)
print(f"Linear SVM: Training Accuracy: {train_acc_linear:.3f}, Testing Accuracy: {test_acc_linear:.3f}")

#SVM algorithm in non-linear mode
svm_nonlinear = SVC(kernel='rbf')
svm_nonlinear.fit(X_train, y_train)
train_acc_nonlinear = svm_nonlinear.score(X_train, y_train)
test_acc_nonlinear = svm_nonlinear.score(X_test, y_test)
print(f"Non-Linear SVM: Training Accuracy: {train_acc_nonlinear:.3f}, Testing Accuracy: {test_acc_nonlinear:.3f}")
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER                    Python Debug Console

Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dr. Behnam>  & 'C:\Users\dr. Behnam\AppData\Local\Programs\Python\Python38-32\python.exe' 'c:\Users\dr. Behnam\.vscode\ext
python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56253' '--' 'c:\Users\dr. Behnam\Official2.py'
Linear SVM: Training Accuracy: 0.594, Testing Accuracy: 0.596
Non-Linear SVM: Training Accuracy: 0.630, Testing Accuracy: 0.642
PS C:\Users\dr. Behnam>
```

این کد دو طبقه‌بندی‌کننده ماشین بردار پشتیبان (SVM) را بر روی یک مجموعه داده را آموزش و ارزیابی می‌کند. اولین مدل SVM با استفاده از هسته خطی آموزش داده می شود و دقت آموزش و تست آن محاسبه و چاپ می شود. سپس مدل دوم SVM با استفاده از هسته غیرخطی (تابع پایه شعاعی) آموزش داده می شود و دقت آموزش و تست آن نیز محاسبه و چاپ می شود.

چند خط اول کد نمونه ای از کلاس SVC برای SVM با هسته خطی ایجاد می کند و آن را با داده های آموزشی متناسب می کند. سپس دقت آموزش و تست این مدل با استفاده از روش score() محاسبه شده و به ترتیب در متغیرهای train_acc_linear و test_acc_linear ذخیره می شود. سپس این دقت ها با را چاپ می کنیم. خطوط بعدی نمونه دیگری از کلاس SVC را برای SVM با هسته تابع پایه شعاعی (RBF) ایجاد کرده و آن را به داده های آموزشی برازش می دهند. دقت آموزش و تست این مدل به طور مشابه محاسبه شده ودر نهایت چاپ می شود.

به طور کلی، این کد راهی برای مقایسه عملکرد مدل‌های SVM با هسته‌های مختلف در یک مجموعه داده ارائه می‌کند.

دسته بندی داده ها با استفاده از یادگیری عمیق:

در ابتدا ستون pricelevel را به صورت زیر one hot encode می کنیم:

```
# Encode the labels as integers using LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# Perform one-hot encoding on the integer-encoded labels
y = to_categorical(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
#print("Training set shape:", X_train.shape)
#print("Testing set shape:", X_test.shape)


```

در ادامه مدل و شبکه ی عصبی را می سازیم و آن را آموزش و در نهایت مورد ارزیابی قرار می دهیم:

```
# Build the model
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.2))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(4, activation='softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
opt = Adam(learning_rate=0.001)
history=model.fit(X_train, y_train, epochs=12000, batch_size=256, validation_data=(X_test, y_test))

# Evaluate the model
train_acc=model.evaluate(X_train, y_train)
print('Training accuracy:', train_acc[1])
test_acc=model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc[1])

```

قسمت اول کد یک مدل ترتیبی ایجاد می کند. این بدان معنی است که لایه ها به صورت متوالی روی هم قرار می گیرند. در این مورد، سه لایه متراکم با عملکرد فعال سازی relu، و دو لایه حذفی وجود دارد که به جلوگیری از برازش بیش از حد کمک می کند. شکل ورودی به صورت (X_train.shape[1]) تعریف می‌شود، به این معنی که مدل انتظار ورودی‌هایی با ویژگی‌های X_train.shape[1] را دارد.

مرحله بعدی کامپایل مدل با مشخص کردن تابع ضرر، بهینه ساز و متریک ارزیابی است. در این مورد، تابع تلفات متقابل آنتروپی طبقه‌ای استفاده می‌شود، زیرا این یک مشکل طبقه‌بندی چند کلاسه است، از بهینه‌ساز Adam استفاده می‌شود و متریک دقت در طول آموزش ردیابی می‌شود.

پس از کامپایل مدل، با استفاده از روش fit() آموزش داده می شود. داده های آموزشی (X_train، y_train) به همراه برخی از فراپارامترها مانند تعداد دوره ها، اندازه دسته و داده های اعتبارسنجی به مدل منتقل می شود. بهینه ساز را نیز می توان در این مرحله مشخص کرد. در این حالت از بهینه ساز Adam با نرخ یادگیری 0.001 استفاده می شود.

در نهایت، مدل آموزش‌دیده شده بر روی داده‌های آموزشی و آزمون با استفاده از روش ارزیابی () ارزیابی می‌شود. دقت آموزش و دقت تست بر روی کنسول چاپ می شود.

هچنین کد مربوط به ماتریس درهم ریختگی نیز به صورت زیر می باشد:

```
# Predict the test data and print the confusion matrix
y_pred = model.predict(X_test)
cm = confusion_matrix(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1))
print('Confusion matrix:\n', cm)
```

پس از تست کردن انواع هایپرپارامتر ها بهترین نتایج و همچنین ماتریس درهم ریختگی به صورت زیر می باشد:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

11/11 [==============================] - 0s 5ms/step - loss: 0.6525 - accuracy: 0.7370 - val_loss: 0.8422 - val_accuracy: 0.7058
87/87 [==============================] - 0s 1ms/step - loss: 0.5528 - accuracy: 0.7844
Training accuracy: 0.7844203114509583
22/22 [==============================] - 0s 1ms/step - loss: 0.8422 - accuracy: 0.7058
Test accuracy: 0.7057970762252808
Confusion matrix:
 [[108   6  22  28]
 [  2 149   3  31]
 [ 15  10 110  35]
 [ 12  24  15 120]]
```

از learning rate با مقدار بسیار کمی استفاده کردیم زیرا نرخ یادگیری کمتر منجر به همگرایی کندتر می شود، اما به طور بالقوه می تواند منجر به عملکرد بهتر شود زیرا به روز رسانی ها دقیق تر هستند و مقدار آن را ۰.۰۰۱ که کمترین مقدار است قرار دادیم.

از softmax به جای sigmoid استفاده شده است زیرا برای طبقه بندی چند کلاسه بسیار مناسب تر و کاربردی تر است.

از batch size و epochs زیادی استفاده شده است به طوری هر چه مقدار آنها بیشتر شد دقت بیشتری نیز بدست آمد. با ۱۲۰۰۰ epochs و ۲۵۶ batch size بهترین نتیجه حاصل شد و با مقادیر کمتری نیز تست شدند اما دقت کمتر از بهترین عملکرد بود و حتی زمانی که تعداد epochs را ۲۰ و تعداد batch size را ۳۲ قرار داده ام. دقت در بازه ۶۰ قرار گرفت که خیلی از بهترین عملکرد فاصله دارد . در نتیجه باید نرخ یادگیری را کم یا حتی کمتیرن مقدار در نظر بگیریم تا دقت و آموزش بالاتر رود و تعداد batch size و epochs را نیز باید مقادیر بالا و زیادی در نظر بگیریم تا عملکرد بسیار خوبی حاصل شود.