به نام خدا

پاسخ به سوالات ورکشاپ درخت تصمیم

ایلیا حبیبی

402481207

تمرین 1: دانلود یک دیتاست از سایت kaggle و کلین کردن آن.

به عنوان بخشی از پروژه انجام شده است.

تمرین 2: الگوریتم درخت تصمیم جز یادگیری نظارت‌شده است یا نظارت‌نشده؟

الگوریتم درخت تصمیم، یک رویکرد یادگیری نظارت‌شده (Supervised Learning) است که به طور گسترده در حوزه‌های آمار، داده‌کاوی و یادگیری ماشین به کار می‌رود. اساس این الگوریتم بر یادگیری یک تابع از ویژگی‌های ورودی (Input Features) به یک متغیر هدف خروجی (Output Target Variable) استوار است. به عبارت دیگر، الگوریتم بر روی یک مجموعه داده آموزش می‌بیند که در آن "پاسخ‌های صحیح" یا برچسب‌ها از قبل مشخص شده‌اند. این ویژگی، اساس تعریف یادگیری نظارت‌شده است.

تمرین 3: بیشتر درباره underfitting و overfitting تحقیق کنید.

مدل نویز را یاد گرفته است. یعنی چه؟

درباره هایپرپارامترهای درخت تصمیم تحقیق کنید و توضیح دهید چگونه هرکدام باعث overfit و underfit می‌شوند.

کم‌برازش (Underfitting) و **بیش‌برازش (Overfitting)**.

- **کم‌برازش** زمانی رخ می‌دهد که مدل بیش از حد ساده است و نمی‌تواند الگوهای پیچیده و اساسی موجود در داده‌ها را بیاموزد. چنین مدلی دارای بایاس (Bias) بالا بوده و هم بر روی داده‌های آموزشی و هم بر روی داده‌های آزمون عملکرد ضعیفی از خود نشان می‌دهد.
- **بیش‌برازش** سناریوی مقابل است؛ در این حالت، مدل بیش از حد پیچیده شده و داده‌های

آموزشی را با جزئیات افراطی، شامل نویزها و داده‌های پرت، "حفظ" می‌کند. این مدل دارای واریانس (Variance) بالا است و با اینکه بر روی داده‌های آموزشی عملکردی عالی دارد، در مواجهه با داده‌های جدید و دیده‌نشده، به دلیل عدم توانایی در تعمیم، عملکرد ضعیفی خواهد داشت.

"مدل نویز را یاد گرفته است"

عبارت "یادگیری نویز" به وضعیتی اشاره دارد که در آن یک مدل پیچیده، به جای یادگیری رابطه واقعی و تعمیم‌پذیر بین ویژگی‌ها و متغیر هدف (که به آن "سیگنال" می‌گویند)، نوسانات تصادفی، خطاها و ویژگی‌های بی‌اهمیت موجود در داده‌های آموزشی (که "نویز" نامیده می‌شوند) را نیز به عنوان الگوهای معنادار شناسایی می‌کند.

در یک درخت تصمیم، این پدیده زمانی رخ می‌دهد که درخت بیش از حد عمیق و شاخه‌شاخه می‌شود. چنین درختی مرزهای تصمیم‌گیری بسیار پیچیده و نامنظمی ایجاد می‌کند تا هر نمونه آموزشی، حتی نمونه‌های نویزی یا پرت، را به درستی طبقه‌بندی کند.[8] برای مثال، ممکن است یک شاخه و برگ مجزا تنها برای طبقه‌بندی یک نمونه داده غیرعادی ایجاد شود. این قانون خاص، نمونه‌ای از یادگیری نویز است و قابلیت تعمیم به داده‌های جدید را نخواهد داشت.

هایپرپارامترها

هایپرپارامترها (Hyperparameters) تنظیمات خارجی مدل هستند که پیش از شروع فرآیند آموزش مشخص می‌شوند و به عنوان ابزار اصلی برای کنترل پیچیدگی درخت تصمیم عمل می‌کنند. سه هایپرپارامتر کلیدی در این زمینه عبارتند از:

max_depth (حداکثر عمق)

- **تعریف:** این هایپرپارامتر حداکثر تعداد سطوح یا لایه‌هایی که یک درخت می‌تواند رشد کند را محدود می‌سازد.
- **تأثیر:** مقدار بسیار کم برای max_depth، تعداد تقسیم‌ها را محدود کرده و مدلی ساده ایجاد می‌کند که ممکن است دچار **کم‌برازش** شود. در مقابل، مقدار بسیار زیاد یا نامحدود به درخت اجازه می‌دهد تا جایی رشد کند که تمام برگ‌ها خالص شوند. این امر منجر به ایجاد مدلی بسیار پیچیده و مستعد **بیش‌برازش** از طریق یادگیری نویز می‌شود.

min_samples_split (حداقل نمونه برای تقسیم)

- **تعریف:** این پارامتر حداقل تعداد نمونه‌هایی که یک گره باید داشته باشد تا برای تقسیم شدن در نظر گرفته شود را تعیین می‌کند.
- **تأثیر:** مقدار کم (مانند مقدار پیش‌فرض ۲) به الگوریتم اجازه می‌دهد تا گره‌هایی با تعداد نمونه‌های بسیار کم را نیز تقسیم کند. این کار پیچیدگی را افزایش داده و ریسک **بیش‌برازش** را به همراه دارد، زیرا مدل ممکن است از الگوهای کوچک و نویزی یاد بگیرد. مقدار بالا، مدل را محافظه‌کارتر کرده و از ایجاد تقسیم‌های بسیار جزئی جلوگیری می‌کند که این امر منجر به درختی

ساده‌تر و احتمالا **کم‌برازش** می‌شود.

min_samples_leaf (حداقل نمونه در برگ)

- **تعریف:** این هایپرپارامتر حداقل تعداد نمونه‌هایی که باید در یک گره پایانی (برگ) وجود داشته باشد را مشخص می‌کند.
- **تأثیر:** این پارامتر یکی از مؤثرترین ابزارها برای مقابله با بیش‌برازش است. مقدار کم (مانند ۱) به درخت اجازه می‌دهد برگ‌هایی ایجاد کند که تنها یک نمونه داده را شامل می‌شوند، که این تعریف دقیق **بیش‌برازش** است. مقدار بالاتر تضمین می‌کند که هر تصمیم نهایی (هر برگ) توسط گروه قابل‌توجهی از نمونه‌ها پشتیبانی می‌شود که این امر به تعمیم‌پذیری بهتر کمک کرده و از تصمیم‌گیری بر اساس داده‌های پرت جلوگیری می‌کند. مقدار بسیار بالا نیز می‌تواند منجر به **کم‌برازش** شود.
- **تأثیر هایپرپارامترها بر پیچیدگی و تعمیم‌پذیری مدل**

هایپرپارامتر	تأثیر مقدار کم	تأثیر مقدار زیاد	گرایش
max_depth	مدل ساده، عدم توانایی در یادگیری الگوهای پیچیده	مدل بسیار پیچیده، یادگیری جزئیات و نویز داده‌ها	کم: کم‌برازش (بایاس بالا) / زیاد: بیش‌برازش (واریانس بالا)
min_samples_split	اجازه تقسیم بر روی گروه‌های کوچک و احتمالاً نویزی	نیاز به شواهد قابل توجه برای ایجاد یک قانون تصمیم جدید	کم: بیش‌برازش (واریانس بالا) / زیاد: کم‌برازش (بایاس بالا)
min_samples_leaf	ایجاد برگ‌ها برای نمونه‌های خاص یا پرت	تضمین اینکه هر تصمیم توسط گروهی از نمونه‌ها پشتیبانی می‌شود	کم: بیش‌برازش (واریانس بالا) / زیاد: کم‌برازش (بایاس بالا)

تمرین 4: درخت را برای دیتاست زیر رسم کنید، محاسبات ریاضی کامل انجام شوند

یک بار براساس information gain و یک بار براساس gini index محاسبات انجام شود.

تمرین

ID	Age	Has_job	Own_house	Credit_rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

مجموعه داده و هدف

در این بخش، درخت تصمیم برای مجموعه داده ارائه‌شده ساخته می‌شود. هدف، ساخت یک مدل طبقه‌بندی برای پیش‌بینی متغیر Class (با مقادیر 'Yes' یا 'No') بر اساس چهار ویژگی Age، Has_job، Own_house و Credit_rating است.

شمارش اولیه داده‌ها به شرح زیر است:

- تعداد کل نمونه‌ها: ۱۵
- توزیع کلاس: ۹ نمونه 'Yes' و ۶ نمونه 'No'

محاسبات گام به گام برای گره ریشه

محاسبه آنتروپی اولیه سیستم

مجموعه داده اولیه شامل ۹ نمونه 'Yes' و ۶ نمونه 'No' است.

$p(Yes)=9/15=0.6$
$p(No)=6/15=0.4$

Entropy(Sroot)=-(0.6×log2(0.6))-(0.4×log2(0.4))
Entropy(Sroot)=-(0.6×-0.737)-(0.4×-1.322)=0.4422+0.5288=0.971

محاسبه بهره اطلاعاتی برای هر ویژگی

محاسبات بهره اطلاعاتی برای ویژگی‌ها

ویژگی	مقدار	تعداد کل	تعداد Yes	تعداد No	آنتروپی زیرمجموعه	بهره اطلاعاتی (Gain)
Age	young	5	2	3	E(2,3)=0.971	**0.082**
	middle	5	4	1	E(4,1)=0.722	
	old	5	3	2	E(3,2)=0.971	
Has_job	true	5	5	0	E(5,0)=0	**0.324**
	false	10	4	6	E(4,6)=0.971	
Own_house	true	6	6	0	E(6,0)=0	**0.420**
	false	9	3	6	E(3,6)=0.918	
Credit_rating	fair	5	1	4	E(1,4)=0.722	**0.254**
	good	6	5	1	E(5,1)=0.650	
	excellent	4	3	1	E(3,1)=0.811	

- محاسبه برای :Age
Eweighted(Age)=155(0.971)+155(0.722)+155(0.971)=0.324+0.241+0.324=0.889

Gain(S,Age)=0.971-0.889=0.082

- محاسبه برای :Has_job
Eweighted(Has_job)=155(0)+1510(0.971)=0.647
Gain(S,Has_job)=0.971-0.647=0.324
- محاسبه برای :Own_house
Eweighted(Own_house)=156(0)+159(0.918)=0.551
Gain(S,Own_house)=0.971-0.551=0.420
- محاسبه برای :Credit_rating
Eweighted(Credit_rating)=155(0.722)+156(0.650)+154(0.811)=0.241+0.260+0.216=0.717
Gain(S,Credit_rating)=0.971-0.717=0.254

انتخاب گره ریشه و تقسیم بازگشتی

بر اساس محاسبات، ویژگی Own_house با بهره اطلاعاتی 0.420 بیشترین مقدار را دارد و به عنوان گره ریشه انتخاب می‌شود.

- **شاخه Own_house = true:** این شاخه شامل ۶ نمونه است که همگی 'Yes' هستند. این گره خالص است و به یک برگ با برچسب **'Yes'** تبدیل می‌شود.
- **شاخه Own_house = false:** این شاخه شامل ۹ نمونه است (۳ 'Yes' و ۶ 'No'). این گره ناخالص است و باید دوباره تقسیم شود.

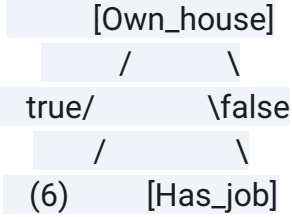
محاسبات برای گره Own_house = false (زیرمجموعه 'S'):
S'=9 نمونه (۳ 'Yes', ۶ 'No'). Entropy(S')=E(3,6)=0.918.
بهره اطلاعاتی برای ویژگی‌های باقی‌مانده (Age, Has_job, Credit_rating) بر روی این زیرمجموعه محاسبه می‌شود.

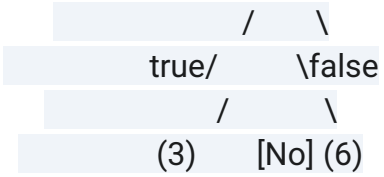
- Gain(S',Has_job)=0.918-[93E(3,0)+96E(0,6)]=0.918-0=0.918

ویژگی Has_job بیشترین بهره اطلاعاتی را در این زیرمجموعه دارد و برای تقسیم بعدی انتخاب می‌شود.

- **شاخه Has_job = true:** شامل ۳ نمونه است که همگی 'Yes' هستند. این گره خالص است و به برگ **'Yes'** تبدیل می‌شود.
- **شاخه Has_job = false:** شامل ۶ نمونه است که همگی 'No' هستند. این گره خالص است و به برگ **'No'** تبدیل می‌شود.

درخت نهایی (بر اساس بهره اطلاعاتی)





ساخت بر اساس شاخص جینی

محاسبات گام به گام برای گره ریشه

محاسبه شاخص جینی اولیه سیستم

p(Yes)=9/15, p(No)=6/15
Gini(Sroot)=1-[(159)2+(156)2]=1-[0.36+0.16]=0.48

محاسبه بهره جینی برای هر ویژگی

محاسبات بهره جینی برای ویژگی‌ها

ویژگی	مقدار	تعداد کل	تعداد Yes	تعداد No	شاخص جینی زیرمجموعه	بهره جینی (Gain)
Age	young	5	2	3	G(2,3)=0.48	**0.037**
	middle	5	4	1	G(4,1)=0.32	
	old	5	3	2	G(3,2)=0.48	
Has_job	true	5	5	0	G(5,0)=0	**0.16**
	false	10	4	6	G(4,6)=0.48	
Own_house	true	6	6	0	G(6,0)=0	**0.2136**
	false	9	3	6	G(3,6)=0.4	

	44					
Credit_rating	fair	5	1	4	G(1,4)=0.32	**0.166**
	good	6	5	1	G(5,1)=0.278	
	excellent	4	3	1	G(3,1)=0.375	

- محاسبه برای :Own_house
Gweighted(Own_house)=156(0)+159(0.444)=0.2664
GiniGain(S,Own_house)=0.48-0.2664=0.2136

با انجام محاسبات دقیق برای همه ویژگی‌ها، Own_house همچنان بالاترین بهره جینی را خواهد داشت. فرآیند تقسیم بازگشتی مشابه روش بهره اطلاعاتی است و به همان درخت نهایی منجر می‌شود. در این مجموعه داده خاص، هر دو معیار به یک درخت یکسان رسیدند، اگرچه این امر همیشه صادق نیست.

تمرین 5: فهمیدیم که قرار نیست همیشه درختی که از روش حریصانه به دست آوردیم، بهینه هم باشه. الان باید تحقیق کنید که چطور درخت بهینه رو به دست بیاریم.

الگوریتم‌های استاندارد ساخت درخت تصمیم مانند ID3، C4.5 و CART از یک استراتژی **حریصانه (Greedy)، بالا به پایین و بازگشتی** استفاده می‌کنند. در هر گره، این الگوریتم‌ها تقسیمی را انتخاب می‌کنند که به صورت محلی بهینه باشد، یعنی بیشترین بهره اطلاعاتی یا بهره جینی را در همان لحظه فراهم کند.

مشکل اصلی این رویکرد، "کوته‌بینی" آن است. یک تقسیم که در یک گره به صورت محلی بهینه به نظر می‌رسد، لزوماً به یک درخت بهینه منجر نمی‌شود. ممکن است یک تقسیم اولیه که در ظاهر ضعیف‌تر است، در مراحل بعدی امکان تقسیم‌های بسیار بهتری را فراهم آورد و در نهایت به درختی کوچک‌تر و دقیق‌تر منجر شود. یافتن درخت تصمیم واقعاً بهینه یک مسئله NP-Hard است، به این معنی که هزینه محاسباتی آن با افزایش ابعاد داده به صورت نمایی رشد می‌کند. الگوریتم‌های حریصانه یک **رهیافت (Heuristic)** یا میان‌بر محاسباتی هستند که بدون جستجو در کل فضای عظیم درخت‌های ممکن، یک درخت "به اندازه کافی خوب" می‌سازند. بهای این میان‌بر، عدم تضمین بهینگی است؛ الگوریتم در هر مرحله به یک تقسیم متعهد می‌شود و هرگز برای بازنگری آن باز نمی‌گردد.

هرس کردن: یک رویکرد عملی برای بهینه‌سازی

هرس کردن (Pruning) نه تنها یک تکنیک برای مقابله با بیش‌برازش، بلکه یک روش عملی برای بهبود درخت غیربهینه‌ای است که توسط الگوریتم حریصانه تولید شده است.

پیش‌هرس (Pre-Pruning)

این رویکرد که به آن توقف زودهنگام (Early Stopping) نیز گفته می‌شود، با اعمال محدودیت‌های هایپرپارامتری (مانند max_depth و min_samples_leaf)، رشد درخت را پیش از آنکه بیش از حد پیچیده شود متوقف می‌کند. این کار از ابتدا مانع از ایجاد یک درخت بیش‌برازش‌شده توسط الگوریتم حریصانه می‌شود.

پس‌هرس (Post-Pruning)

این یک رویکرد اصلاحی است. ابتدا یک درخت تا عمق کامل خود (و احتمالاً تا حد بیش‌برازش) رشد می‌کند. سپس، الگوریتم به سمت عقب حرکت کرده و شاخه‌ها و گره‌ها را ارزیابی می‌کند. شاخه‌هایی که قدرت پیش‌بینی کمی در یک مجموعه داده اعتبارسنجی (Validation Set) دارند، حذف (هرس) می‌شوند و درخت ساده‌تر می‌گردد. یکی از تکنیک‌های کلیدی در این زمینه **هرس پیچیدگی-هزینه (Cost-Complexity Pruning - CCP)** است که به طور سیستماتیک زیردرخت‌ها را ارزیابی کرده و "ضعیف‌ترین حلقه" (زیردرختی که کمترین بهبود را به ازای هر برگ ارائه می‌دهد) را هرس می‌کند.

بهینه‌سازی جهانی: جستجو برای بهینگی واقعی

فراتر از هرس کردن، روش‌های پیشرفته و غیرحریصانه‌ای وجود دارند که با فرموله‌بندی ساخت درخت به عنوان یک مسئله بهینه‌سازی رسمی، به دنبال یافتن درخت بهینه جهانی هستند.

برنامه‌ریزی ریاضی (بهینه‌سازی عدد صحیح مختلط - MIO)

این رویکرد کل فرآیند ساخت درخت را به عنوان یک مسئله بهینه‌سازی واحد با یک تابع هدف (مانند حداکثرسازی دقت) و مجموعه‌ای از شروط تعریف می‌کند. از آنجایی که متغیرهای تصمیم می‌توانند هم پیوسته و هم گسسته (صحیح) باشند، این یک مسئله MIO است. مزیت این روش آن است که به صورت تئوری می‌تواند درخت بهینه قابل اثبات را پیدا کند، اما عیب بزرگ آن هزینه محاسباتی بسیار بالا و عدم مقیاس‌پذیری برای مجموعه داده‌های بزرگ است.

این حوزه تحقیقاتی که به آن درختان تصمیم بهینه (Optimal Decision Trees - ODTs) گفته می‌شود، یک مرز نوین در یادگیری ماشین است. برای سال‌ها، عدم بهینگی درختان تصمیم منفرد پذیرفته شده بود و تمرکز به سمت مدل‌های گروهی (Ensemble) مانند رندوم فارست و گرادیان بوستینگ معطوف شد که با ترکیب تعداد زیادی یادگیرنده ضعیف، به عملکرد بالا به قیمت از دست دادن تفسیرپذیری دست می‌یابند. پیشرفت‌های اخیر در حل‌کننده‌های بهینه‌سازی، علاقه به ساخت یک درخت منفرد، بسیار دقیق و در عین حال قابل تفسیر را احیا کرده است. هدف، دستیابی به عملکردی قابل رقابت با مدل‌های گروهی ضمن حفظ ماهیت "وایت باکس" یک درخت منفرد است؛ امری که به ویژه در حوزه‌های پرمخاطره مانند پزشکی و مالی که در آن‌ها تفسیرپذیری یک الزام است، ارزش فوق‌العاده‌ای دارد.