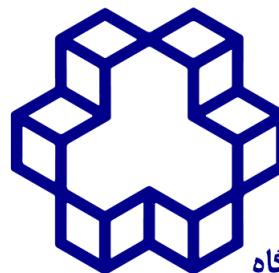


بسم الله الرحمن الرحيم



دانشگاه
خواجہ نصیرالدین طوسی
K. N. Toosi University
of Technology

پاسخ مینی پروژه دوم یادگیری ماشین

Google Colab
GitHub

نگارش: علی شعبانپور مقدم - هدیه شوشیان

شماره دانشجویی: ۴۰۲۰۷۳۰۴-۴۰۳۰۸۰۵۴

استاد درس: دکتر مهدی علیاری شوره دلی

فهرست مطالب

| | |
|----|---|
| ۳ | ۱ پرسش اول |
| ۳ | ۱.۱ الف |
| ۳ | ۲.۱ ب |
| ۳ | ۳.۱ ج |
| ۴ | ۴.۱ د |
| ۵ | ۵.۱ ه |
| ۶ | ۶.۱ و |
| ۸ | ۲ پرسش دوم |
| ۸ | ۱.۲ الف و ب |
| ۹ | ۲.۲ ج و د |
| ۱۰ | ۳.۲ ه |
| ۱۱ | ۳ پرسش سوم |
| ۱۱ | ۱.۳ |
| ۱۲ | ۲.۳ پیش پردازش داده ها |
| ۱۵ | ۳.۳ محاسبه آنتروپی داده ها |
| ۱۶ | ۴.۳ محاسبه بهره اطلاعات یا سود اطلاعاتی |
| ۱۷ | ۵.۳ درخت تصمیم |



۱ پرسش اول

۱.۱ الف

طبقه بندی بیز با استفاده از قضیه بیز و فرض بر استقلال بین متغیرها و ویژگیها برای دسته بندی برمنای احتمال کاربرد دارد. طبقه بندی بیز بهینه فرض می کند استقلال بین ویژگیها وجود ندارد و الگوریتم سعی می کند از توزیع کامل احتمال شرطی بین ویژگیها استفاده کند اما به دلیل اینکه محاسبات آن دشوار است کمتر استفاده می شود. پس در بیز ساده فرض می شود ویژگیها استقلال دارند، محاسبات پیچیده هستند، نیاز به داده های زیاد لازم نیست، و دقت آن کم است این در حالی است که بیز بهینه کاملا عکس می باشد.

۲.۱ ب

انواع الگوریتم طبقه بند بیز ساده بصورت زیر است: *GaussianNaive* که برای داده های پیوسته مناسب است و ویژگیها در هر کلاس از توزیع نرمال پیروی می کند.

برای داده هایی که ویژگی های دو حالتی یا باينری *multinomialnavie* برای داده هایی که ویژگی های دو حالتی یا باينری *BernoulliNaive* در این دادگان دو ستون داریم که یکی عبارات پیامکی دیگری ستون برچسب های آن است که *ham* و *spam* هستند. از آنجایی که کامپیوتر متوجه پیام نمی شود پس با پیدا کردن کلمات اسپم موجود در پیام تشخیص می دهد که یک پیام اسپم است یا خیر. پس چون به فرم باينری یا صفر و یک می باشد از الگوریتم برنولی بهره می بریم.

۳.۱ ج

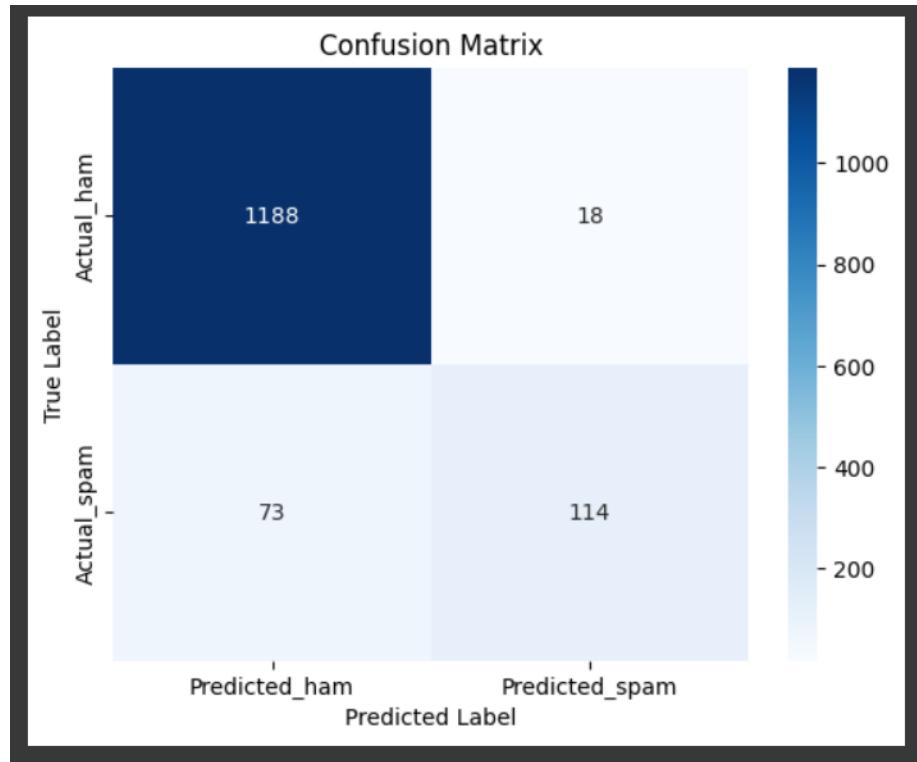
دادگان را بارگذاری می کنیم کتابخانه ها را فراخوانی می کنیم دیتا فریم را به شکل معین ذخیره می کنیم. سپس تمام داده ها را بر اساس برچسب های آنها بصورت زیر جدا می کنیم و تغییراتی روی متغیرهای اعمال می کنیم از جمله: حذف علائم نگارشی با کتابخانه استرینگ، کوچک کردن حرف متن، حذف اعداد موجود با کتابخانه `re`، حذف کلماتی که اطلاعی از اسپ بودن یا نبودن نمی دهند با کتابخانه `nltk` که نیاز به دانلود شدن دارد، استفاده از کتابخانه کلاکشن برای بدست اوردن کلمات با تعداد تکرار آنها، تعدادی از کلمات پرتکرار را جدا کرده و در یک لیست ذخیره می کنیم.
ده کلمه پرتکرا در دو کلاس را با دستورات مربوطه نمایش می دهیم. بررسی میکنیم ایا این کلمات پرتکرار در متنی وجود دارند؟ اگر وجود دارند عدد یک و در غیر اینصورت عدد صفر برای آن درایه خاص لحظه می گردد. برای انجام این کار بصورت زیر عمل میکنیم:
ابتدا پس از اضافه کردن کتابخانه ها یک لیست خالی تعریف می کنیم. حلقه `for` ای تعریف میکنیم که روی سطها پیش پردازش ها را انجام داده و بررسی میکنیم که ایا کلمات موجود در متن جزو کلمات پرتکرار بوده است؟ نتیجه را بصورت یک دیتا فریم ذخیره می کنیم.
سپس داده ها را به دسته آزمون و آموزش تقسیم می کنیم و الگوریتم زیر را روی آن بکار می بریم:

$$P_1 = \frac{P(\text{word1} = (0,1), \text{word2} = (0,1), \dots, \text{word20}(0,1) | \text{class1}) \times P(\text{class1})}{P(\text{word1} = (0,1), \text{word2} = (0,1), \dots, \text{word20}(0,1))}$$
$$P_2 = \frac{P(\text{word1} = (0,1), \text{word2} = (0,1), \dots, \text{word20}(0,1) | \text{class2}) \times P(\text{class2})}{P(\text{word1} = (0,1), \text{word2} = (0,1), \dots, \text{word20}(0,1))}$$

دنبال آن هستیم تشخیص دهیم که با داده با احتمال بالاتر متعلق به کدام کلاس است.
با دستور یونیک انواع لیبل ها استخراج می گردد و سپس احتمال انها را محاسبه می کنیم. برای جلوگیری از صفر شدن کل رابطه عدد یک را با صورت مسر ها جمع میکنیم. با استفاده از لگاریتم ضرب را به جمع تبدیل میکنیم و سپس تغلق نمونه را مورد بررسی قرار می دهیم. برای ذخیره کلاس های با احتمال بالاتر

برای هر نمونه یک لیست خالی درنظر می‌گیریم. از یک دیکشنری برای ذخیره کلاس و احتمال هر کلاس برای هر نمونه بهره می‌بریم. بالاترین مقدار لگاریتم همان کلاس نهایی است.

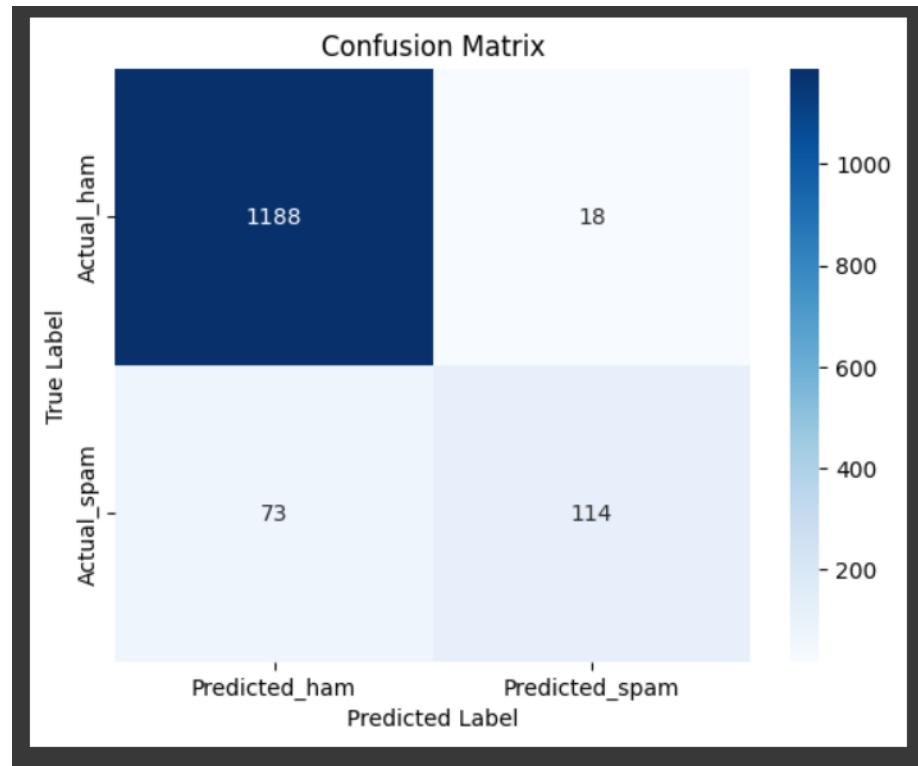
مقدار دقیق برابر ۰.۹۳۴۶ بودست امد. و دقت بازیابی و مشتبه برای هر دو کلاس برابر شد با ۰.۶۰۹، ۰.۸۶۳، ۰.۹۸۵، ۰.۹۴۲ و ماتریس درهم ریختگی به شکل زیر بدست می‌اید:



شکل ۱: ماتریس درهم ریختگی

۴.۱

حال با استفاده از *sklearn* مدل را بصورت آماده تبدیل میکنیم و خروجی‌ها به شکل زیر خواهد بود: ۰.۸۰۲، ۰.۶۶۰، ۰.۹۳۶، ۰.۹۶۸ و ماتریس درهم ریختگی بصورت زیر خواهد بود



شکل ۲: ماتریس درهم ریختگی

نتایج در این حالت بهم نزدیک هستند

۵.۱

فرض می‌کنیم مجموعه کلاس‌ها به صورت زیر تعریف شده باشد:

$$C = \{C_1, C_2, \dots, C_k\}$$

و x یک نمونه مشاهده شده باشد. احتمال پسین برای کلاس C_k به صورت زیر تعریف می‌شود:

$$P(C_k | x)$$

اگر تصمیم بگیریم که x به کلاس $\hat{C}(x)$ تعلق دارد، آنگاه احتمال خطای تصمیم به صورت زیر است:

$$P(\text{خطا} | x) = 1 - P(\hat{C}(x) | x)$$

برای کمینه کردن خطای تصمیم، باید کلاسی را انتخاب کنیم که بیشترین احتمال پسین را داشته باشد:



$$\hat{C}(x) = \arg \max_k P(C_k | x)$$

بنابراین، خطای کلی برابر خواهد بود با:

$$P(\text{خطا}) = \int P(\text{خطا} | x) \cdot p(x) dx = \int [1 - \max_k P(C_k | x)] \cdot p(x) dx$$

۶.۱ و

تابع ریسک برای انتخاب تصمیم α_i در شرایط مشاهده‌ی x به شکل زیر تعریف می‌شود:

$$R(\alpha_i | x) = \sum_{j=1}^k \lambda(\alpha_i | y_j) \cdot P(y_j | x)$$

که در آن: $\lambda(\alpha_i | y_j)$ هزینه‌ی انتخاب تصمیم α_i وقتی کلاس واقعی y_j باشد. $P(y_j | x)$ احتمال وقوع کلاس y_j با توجه به x است.
در این سؤال، ما با دو کلاس spam و ham سر و کار داریم. ماتریس هزینه‌ها به شکل زیر است:

وقتی پیش‌بینی spam است:

| هزینه | کلاس واقعی |
|-------|------------|
| . | spam |
| ۱ | ham |

| هزینه | کلاس واقعی |
|-------|------------|
| ۵ | spam |
| . | ham |

با توجه به این جدول‌ها، ریسک پیش‌بینی‌ها به صورت زیر خواهد بود:

$$R(\text{اسپم} | x) = 0 \cdot P(\text{اسپم} | x) + 1 \cdot P(\text{هم} | x)$$

$$R(\text{هم} | x) = 5 \cdot P(\text{اسپم} | x) + 0 \cdot P(\text{هم} | x)$$

ما تصمیم می‌گیریم که x متعلق به کلاس spam باشد اگر:

$$R(\text{اسپم} | x) < R(\text{هم} | x) \Rightarrow P(\text{هم} | x) < 5 \cdot P(\text{اسپم} | x)$$



از آنجایی که:

$$P(\text{هم} | x) = 1 - P(\text{اسپم} | x) \Rightarrow P(\text{اسپم} | x) > \frac{1}{6} \approx 0.167$$

بنابراین، مرز تصمیم از ۵.۰ به حدود ۱۶۷.۰ تغییر می‌کند.

در این تمرین، اهمیتتابع ریسک در تصمیم‌گیری آماری و نقش هزینه‌های متفاوت در خطای نوع اول و دوم را بررسی کردیم. دیدیم که حتی با یک تغییر ساده در هزینه‌ی اشتیاه، مرز تصمیم باین‌ری بیز از مقدار کلاسیک خود (۵.۰) به مقدار بسیار متفاوتی (۱۶۷.۰) جابجا می‌شود. این نتیجه‌گیری نشان‌دهنده‌ی اهمیت تنظیم مدل‌ها متناسب با شرایط واقعی مسئله و نه فقط داده‌های آماری خام است.



۲ پرسش دوم

۱.۲ الف و ب

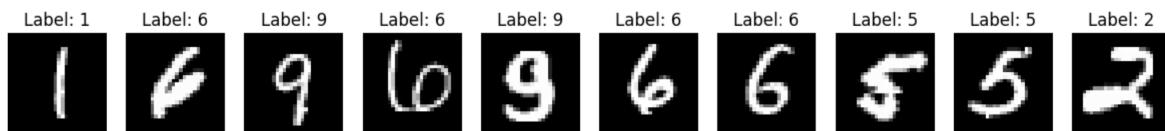
داده‌های مربوط به تصاویر ارقام دستنویس در دیتاست MNIST را در نظر بگیرید. این دیتاست شامل تصاویر خاکستری با اندازه 28×28 پیکسل می‌باشد که هر تصویر به صورت یک بردار با ۷۸۴ ویژگی نمایش داده شده است. مراحل زیر را انجام دهید:

(آ) از دیتاست MNIST، تعداد 10000 نمونه انتخاب کرده و داده‌ها را به کمک یکی از روش‌های StandardScaler یا MinMaxScaler نرمال‌سازی کنید. سپس دلیل انتخاب روش نرمال‌سازی را با توجه به ماهیت داده‌ها بیان نمایید.

(ب) داده‌های نرمال‌سازی شده را به دو بخش آموزش و تست با نسبت 70% به 30% تقسیم کنید.

ما داده‌های مجموعه MNIST را با استفاده از تابع `fetch_openml` از کتابخانه `sklearn.datasets` بارگذاری کردیم و ۱۰۰۰۰ نمونه اول را انتخاب نمودیم. هر تصویر در این مجموعه، یک تصویر ۲۸ در ۲۸ پیکسلی سیاه‌وسفید است که به صورت یک بردار ۷۸۴ بعدی مسطح شده ذخیره شده است. ابتدا تعدادی نمونه از داده را مشاهده می‌کنیم در زیر:

10 Sample Images from Normalized Training Set



برای نرمال‌سازی این داده‌ها پیش از اعمال الگوریتم KNN، از MinMaxScaler استفاده کردیم.

چرا MinMaxScaler؟

یکی از دلایل اصلی استفاده از MinMaxScaler این است که داده‌های تصویر دارای محدوده مشخصی بین ۰ تا ۲۵۵ هستند. این مقیاس‌ساز داده‌ها را به بازه‌ی ثابت $[0, 1]$ نگاشت می‌کند، در حالی که فاصله‌ی نسبی میان شدت پیکسل‌ها را حفظ می‌کند.

الگوریتم KNN به دلیل استفاده از فاصله‌ی اقلیدسی نسبت به بزرگی مقادیر ویژگی‌ها حساس است. بنابراین اگر داده‌ها نرمال‌سازی نشده باشند، ویژگی‌هایی با مقیاس بزرگ‌تر می‌توانند تأثیر ناعادلانه‌ای بر نتایج داشته باشند.

از طرف دیگر، چون داده‌های تصویری به صورت طبیعی غیر منفی و در بازه‌ای مشخص هستند، استفاده از StandardScaler به جای MinMaxScaler که داده‌ها را حول میانگین صفر می‌برد و فرض می‌کند داده‌ها توزیع نرمال دارند، منطقی‌تر و مؤثرتر است.

در نتیجه انتخاب ما برای نرمال‌سازی بر اساس ماهیت داده‌ها و نیازهای الگوریتم KNN انجام شد.

داده‌های نرمال‌سازی شده به دو بخش تقسیم شده‌اند: ۷۰٪ برای آموزش و ۳۰٪ برای ارزیابی.

برای انجام این تقسیم‌بندی از تابع `train_test_split` در کتابخانه `sklearn.model_selection` استفاده کردیم و یک مقدار تصادفی ثابت برای `random_state` تعیین کردیم تا نتایج قابلیت تکرار داشته باشند.

۲.۰ ج و د

- ج) با استفاده از الگوریتم K-Nearest Neighbors (KNN)، مدل را برای حداقل سه مقدار مختلف k (برای مثال $k = 3, 5, 9$) آموزش دهید. سپس با تغییر مقدار k در بازه‌ای مانند ۱ تا ۲۵ با گام ۲، عملکرد مدل را ارزیابی کرده و بهترین مقدار k را تعیین کنید.
- د) دقت مدل را برای مقادیر مختلف k در یک نمودار رسم کنید تا تاثیر پارامتر k بر عملکرد مدل به صورت بصری مشخص شود.

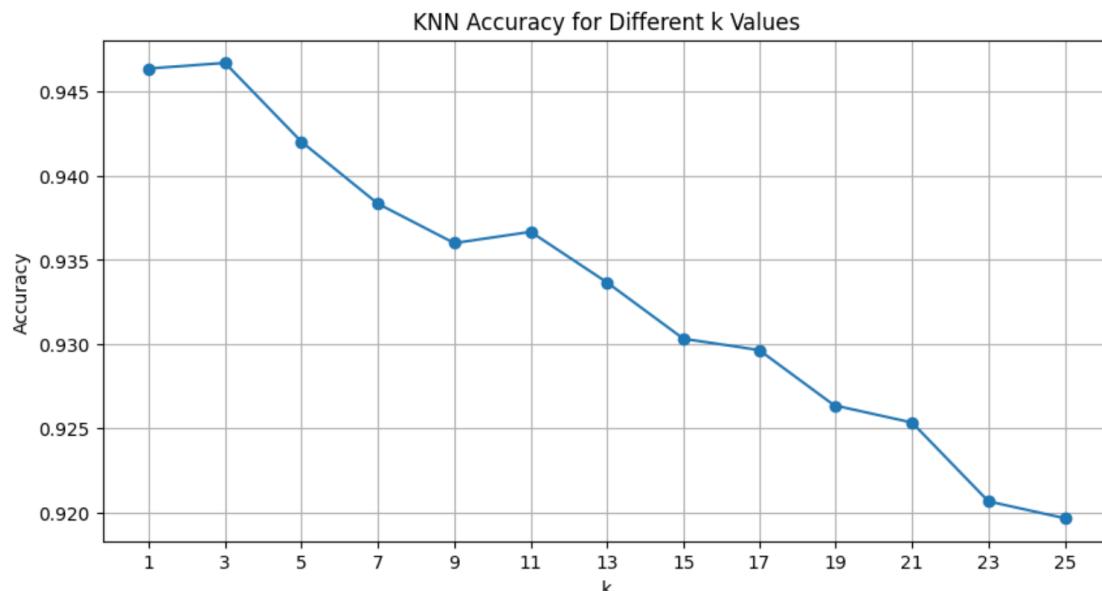
ج - الگوریتم (KNN)

ما مدل‌های KNN را با مقادیر $k = 3, 5, 9$ آموزش دادیم و دقت آن‌ها را روی مجموعه‌ی تست اندازه‌گیری کردیم. سپس یک جستجوی سراسری روی مقادیر فرد k از ۱ تا ۲۵ انجام دادیم تا مقدار بهینه را پیدا کنیم. برای ارزیابی دقت مدل از تابع `accuracy_score` در کتابخانه‌ی `sklearn.metrics` استفاده کردیم و مقدار k با بهترین عملکرد گزارش شد.

بهترین مقدار k : ۳ با دقت ۹۴.۶۷٪

د - نمودار دقت بر حسب k

نموداری از دقت مدل نسبت به مقدار k ترسیم شده تا روند تغییرات عملکرد مدل با افزایش تعداد همسایه‌ها قابل مشاهده باشد.



۳.۲ ۵

(۴) یک روش بهبود برای افزایش دقت الگوریتم KNN پیشنهاد دهید. برای مثال می‌توانید از کاهش بعد با روش PCA استفاده کنید. توضیح دهید که چرا این روش می‌تواند موثر باشد و سپس عملکرد مدل را برای مقادیر مختلف مؤلفه‌های اصلی (مثلاً ۲۰، ۴۰، ۶۰، ...، ۱۰۰) ارزیابی کنید. نتیجه را در قالب نمودار Accuracy vs. Number of PCA Components نمایش دهید.

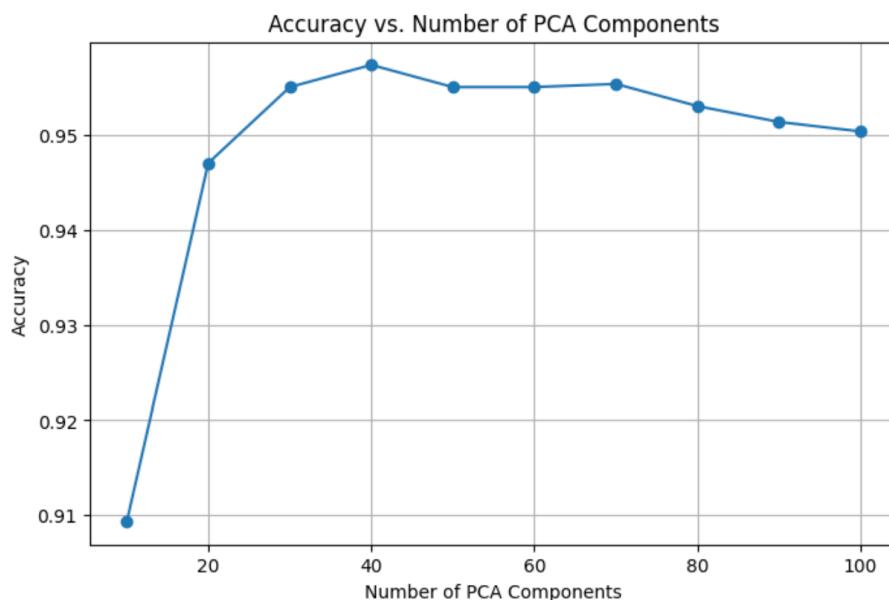
۵- بهبود عملکرد KNN با استفاده از PCA

ما از PCA برای کاهش ابعاد دیتاست MNIST پیش از آموزش مدل KNN استفاده کردیم. **PCA چه کمکی می‌کند؟**

داده‌های اولیه‌ی MNIST شامل ۷۸۴ ویژگی هستند که بسیاری از آن‌ها با یکدیگر همبستگی زیادی دارند یا حاوی اطلاعات مفیدی نیستند. PCA با یافتن جهاتی (مولفه‌هایی) که بیشترین واریانس داده را توضیح می‌دهند، داده‌ها را فشرده‌سازی می‌کند. این کار باعث کاهش نویز، کاهش هزینه‌ی محاسباتی، و در بعضی موارد بهبود عملکرد مدل می‌شود. ما مدل KNN را با بهترین مقدار $k=3$ روی داده‌های کاهش‌بعدیافته آموزش دادیم. برای بررسی اثر کاهش ابعاد، از تعداد مؤلفه‌های مختلف، ۲۰، ۴۰، ۶۰، ۸۰، ۱۰۰ استفاده کردیم.

دقت حاصل از مدل‌ها رسم شده تأثیر استفاده از PCA به صورت تصویری قابل مشاهده باشد.
مقایسه قبل و بعد:

بهترین دقت مدل KNN بدون استفاده از PCA برابر ۹۴.۶۷% با $k=3$ بود. استفاده از PCA نه تنها باعث بهبود کارایی و جلوگیری از بیش‌برازش می‌شود، بلکه در این آزمایش دقت ۹۵.۶۳% را با $k=3$ به دست آوردیم که نشان‌دهنده‌ی عملکرد بهتر مدل و تأثیر مثبت PCA است.





۳ پرسش سوم

یک شرکت تولید پوشак به دنبال این است که بفهمد کدام ویژگی‌ها یا بخش‌ها باعث افزایش فروش می‌شوند. برای این منظور، از الگوریتم‌های مختلفی برای تحلیل داده‌ها استفاده می‌شود. داده‌هایی که به شما داده شده شامل ۱۰ ویژگی و ۴۰۰ رکورد است. هدف این است که از درخت تصمیم و الگوریتم‌های دیگر برای شناسایی ویژگی‌هایی استفاده کنید که بیشترین تاثیر را بر افزایش فروش دارند.

۱.۳

مجموعه section دادگان^۱ دیتاستی که در این پروژه استفاده می‌شود در قالب Comma Separated Values (CSV) یا به اختصار pandas.read_csv() می‌باشد و بایستی ابتدا آنرا به شکل DataFrame بخوانید. (راهنمایی: با استفاده ازتابع() می‌توانید این کار را انجام دهید). برای دریافت دیتاست می‌توانید از این پیوند استفاده کنید.

- با استفاده ازتابع().head() سطر اول دیتاست را بخوانید و نمایش دهید.

۱.۴

بارگذاری و پیش‌نمایش داده‌های شرکت

ما فایل Company_Data.csv را از مخزن GitHub و با استفاده از لینک خام آن بارگذاری کردیم. این دیتاست شامل 240 رکورد و 10 ویژگی مختلف است که عواملی مؤثر بر فروش را نمایش می‌دهند. با استفاده از دستور head().. ردیف اول داده را نمایش دادیم تا ساختار دیتاست را پیش از انجام تحلیل بررسی کنیم.

| | Sales | CompPrice | Income | Advertising | Population | Price | ShelveLoc | Age | Education | Urban | US |
|---|-------|-----------|--------|-------------|------------|-------|-----------|-----|-----------|-------|-----|
| 0 | 9.50 | 138 | 73 | 11 | 276 | 120 | Bad | 42 | 17 | Yes | Yes |
| 1 | 11.22 | 111 | 48 | 16 | 260 | 83 | Good | 65 | 10 | Yes | Yes |
| 2 | 10.06 | 113 | 35 | 10 | 269 | 80 | Medium | 59 | 12 | Yes | Yes |
| 3 | 7.40 | 117 | 100 | 4 | 466 | 97 | Medium | 55 | 14 | Yes | Yes |
| 4 | 4.15 | 141 | 64 | 3 | 340 | 128 | Bad | 38 | 13 | Yes | No |
| 5 | 10.81 | 124 | 113 | 13 | 501 | 72 | Bad | 78 | 16 | No | Yes |
| 6 | 6.63 | 115 | 105 | 0 | 45 | 108 | Medium | 71 | 15 | Yes | No |
| 7 | 11.85 | 136 | 81 | 15 | 425 | 120 | Good | 67 | 10 | Yes | Yes |
| 8 | 6.54 | 132 | 110 | 0 | 108 | 124 | Medium | 76 | 10 | No | No |
| 9 | 4.69 | 132 | 113 | 0 | 131 | 124 | Medium | 76 | 17 | No | Yes |



- ابتدا مشخص نمایید که در درون دیتاست داده‌های ناقص یا گمشده^۴ وجود دارد یا خیر؟ در صورت وجود داده‌های ناقص یا گمشده، آنها را از دیتاست حذف نمایید.
- یکی از مشکلات رایج در درون دیتاست‌ها وجود داده‌های تکراری^۵ می‌باشد. ابتدا توضیح دهید که داده‌های تکراری چگونه می‌توانند برای مدل آموزش دیده و تحلیل داده‌ها مشکل ایجاد نمایند. سپس مشخص نمایید که چه تعداد داده تکراری در درون دیتاست است و در صورت وجود آنها را حذف نمایید.
- ویژگی‌هایی مانند محل فروختن محصول (شهر یا غیرشهر)، در آمریکا یا خارج از آمریکا فروخته شدن محصول و محل قفسه‌های فروش محصول جزو ویژگی‌های دسته‌ای می‌باشند. برای اینکه از چنین ویژگی‌هایی بتوانیم در آموزش مدل استفاده نماییم، می‌بایست آنها را به ویژگی‌های عددی^۶ تبدیل نماییم بنابراین این ویژگی‌ها را Encode نمایید.

۲.۳ پیش‌پردازش داده‌ها

بررسی داده‌های ناقص یا گمشده

در ابتدا بررسی شد که آیا داده‌های ناقص یا گمشده در دیتاست وجود دارد یا خیر. نتیجه بررسی با استفاده از دستور `df.isnull().sum()` نشان داد که هیچ داده‌ی گمشده‌ای در هیچ‌یک از ستون‌های دیتاست وجود ندارد. در صورت وجود چنین داده‌هایی، باید آن ردیف‌ها را با استفاده از `dropna()` حذف کردیم؛ زیرا وجود داده‌های ناقص می‌تواند مدل را دچار خطا و عدم دقت کند.

بررسی داده‌های تکراری

با استفاده از دستور `df.duplicated().sum()` بررسی شد که آیا داده‌های تکراری در دیتاست وجود دارد. در این دیتاست هیچ داده‌ی تکراری مشاهده نشد. اما در صورت وجود، این داده‌ها باید با دستور `drop_duplicates()` حذف می‌شدند، چرا که داده‌های تکراری ممکن است باعث ایجاد data leakage و `بایاس` در مدل شوند و دقت پیش‌بینی را کاهش دهند.

بررسی ویژگی‌های دسته‌ای مانند محل فروش محصول و قفسه

ویژگی‌هایی مانند Urban (فروش در شهر یا خارج از شهر)، US (فروش در آمریکا یا خارج از آمریکا)، و ShelveLoc (موقعیت قفسه در فروشگاه) از نوع داده‌های دسته‌ای هستند. برای استفاده از این ویژگی‌ها در مدل‌های یادگیری ماشین، باید آن‌ها را به فرمت عددی تبدیل کرد. برای Urban و US از کدگذاری دودویی استفاده شد (بله: ۱، خیر: ۰). برای ShelveLoc که سه دسته Bad, Medium, Good دارد، از کدگذاری عدد صحیح استفاده شد: `Good → 2`, `Medium → 1`, `Bad → 0`.



- همانطور که بیان شد، هدف دیتاست این است که ببیند این ویژگی‌ها چطور می‌توانند باعث افزایش فروش شوند، بنابراین متغیر هدف در چنین دیتاستی میزان فروش محصول (sales) می‌باشد، اما همانطور که مشاهده می‌شود جنس خروجی‌ها از جنس متغیرهای عددی می‌باشند و هدف ما نیز آموزش مدل با درخت تصمیم است. برای این منظور نیاز است تا متغیرهای عددی را به متغیرهای دسته‌ای تبدیل نماییم. این کار را برای متغیر هدف پیاده‌سازی نمایید. (تعداد کلاس‌ها را با توجه به برآورد خودتان از متغیر هدف تنظیم نمایید.)

بدین ترتیب مرحله پیش‌پردازش بر روی دیتاست به پایان می‌رسد.

- ماتریس همبستگی^۱ بین ویژگی‌های دیتاست و متغیر هدف را ترسیم نمایید.
- پس از ترسیم ماتریس تحلیل نمایید که کدام ویژگی‌ها همبستگی بیشتری با یکدیگر و متغیر هدف دارند.

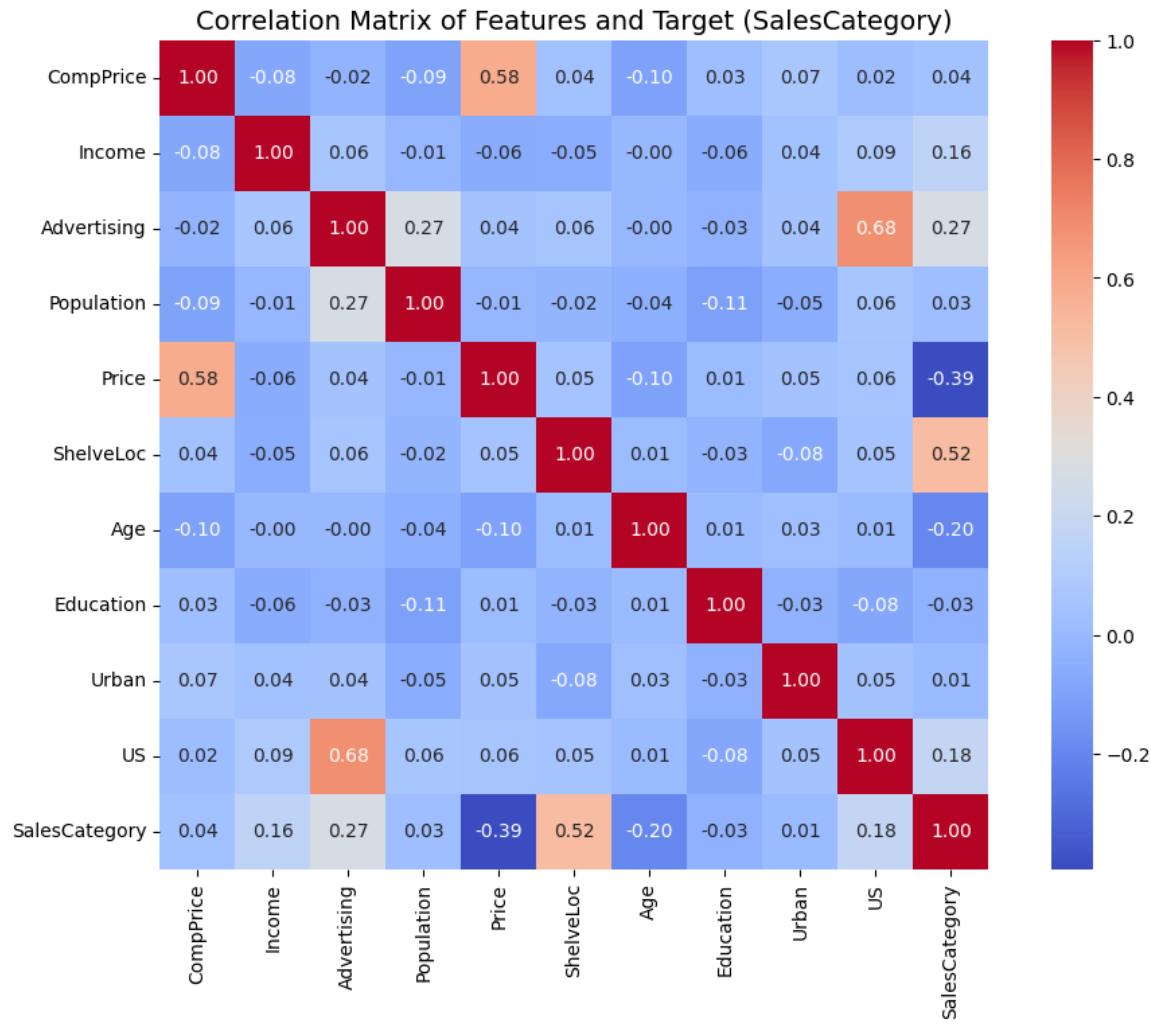
هدف مدل‌سازی و نحوه‌ی تعریف متغیر هدف

همان‌طور که مشخص است، متغیر Sales نشان‌دهنده‌ی میزان فروش محصول است و از جنس عددی پیوسته می‌باشد. اما برای آموزش مدل‌هایی مانند درخت تصمیم که بر اساس دسته‌بندی عمل می‌کنند، نیاز است این متغیر به صورت طبقه‌بندی‌شده تعریف شود. برای این کار، با استفاده از آستانه‌های مشخص شده مبتنی بر واقعیت تجاری، مقادیر Sales را به سه دسته تقسیم کردیم: low برای مقادیر کمتر یا مساوی 6 - medium برای مقادیر بیشتر از 6 و کمتر یا مساوی 10 - high برای مقادیر بیشتر از 10 - این دسته‌بندی باعث شد که فقط درصد کمی از محصولات در دسته‌ی high قرار بگیرند، که هم‌راستا با رفتار واقعی بازار است (یعنی محصولاتی که فروش خلی بالایی دارند، نادرتر هستند).

متغیر هدف جدید تحت عنوان SalesCategory تعریف شد و ستون عددی اصلی Sales حذف گردید تا از آن برای مدل‌سازی استفاده نشود.

تحلیل همبستگی بین ویژگی‌ها و متغیر هدف

برای درک این که چه ویژگی‌هایی بیشترین اثر را بر فروش دارند، ضریب همبستگی بین ویژگی‌ها و متغیر دسته‌ای SalesCategory را بررسی کردیم. نتایج زیر حاصل شد:



ویژگی‌هایی با بیشترین همبستگی مثبت:

- ShelveLoc با ضریب 0.52+ محل بهتر محصول در فروشگاه تأثیر قوی مثبتی بر فروش دارد.

- Advertising با ضریب 0.27+ تبلیغات بیشتر با فروش بهتر در ارتباط است.

- US با ضریب 0.18+ فروش در آمریکا اندکی با فروش بالاتر مرتبط است.

ویژگی‌هایی با بیشترین همبستگی منفی:

- Price با ضریب -0.39- قیمت بالاتر تأثیر منفی قوی بر فروش دارد.

- Age با ضریب -0.20- مشتریان مسن‌تر تمایل کمتری به خرید محصولات پرفروش دارند.

این نتایج می‌توانند برای تحلیل اهمیت ویژگی‌ها و بهینه‌سازی عملکرد مدل مورد استفاده قرار بگیرند.



۳.۳ محاسبه آنتروپی داده ها

همانطور که بیان شد، هدف دیتاست این است که بیند این ویژگی‌ها چطور می‌توانند باعث افزایش فروش شوند، بنابراین متغیر هدف در چنین دیتاستی میزان فروش محصول (sales) می‌باشد، اما همانطور که مشاهده می‌شود جنس خروجی‌ها از جنس متغیرهای عددی می‌باشند و هدف ما نیز آموزش مدل با درخت تصمیم است. برای این منظور نیاز است تا متغیرهای عددی را به متغیرهای دسته‌ای تبدیل نماییم. این کار را برای متغیر هدف پیاده‌سازی نمایید. (تعداد کلاس‌ها را با توجه به برآورد خودتان از متغیر هدف تنظیم نمایید.)

بدین ترتیب مرحله پیش‌پردازش بر روی دیتاست به پایان می‌رسد.

- ماتریس همبستگی^۱ بین ویژگی‌های دیتاست و متغیر هدف را ترسیم نمایید.
- پس از ترسیم ماتریس تحلیل نمایید که کدام ویژگی‌ها همبستگی بیشتری با یکدیگر و متغیر هدف دارند.

3.3 Entropy Calculation

Entropy is a metric that quantifies the uncertainty or impurity in a set of class labels.

We implemented a function `calculate_entropy(y)` that:

1. Counts the frequency of each class label in `y`
2. Converts those counts into probabilities
3. Applies the entropy formula: [$Entropy(S) = -\sum p_i \log_2(p_i)$]
 - Entropy is **0** when all samples belong to the same class.
 - Entropy is **maximum** when the classes are equally distributed.

This function is useful for evaluating data splits in decision tree learning.

```
[ ] import numpy as np

def calculate_entropy(y):

    # Convert to NumPy array if not already
    y = np.array(y)

    # Count the frequency of each class label
    _, counts = np.unique(y, return_counts=True)

    # Convert counts to probabilities
    probabilities = counts / counts.sum()

    # Compute entropy using the formula
    entropy = -np.sum(probabilities * np.log2(probabilities))

    return entropy
```

```
[ ] # Example: a balanced sample

y_example = [1, 2 ,3,1, 3, 2,4,4]
entropy_value = calculate_entropy(y_example)
print(f"Entropy: {entropy_value:.4f}")
```

Entropy: 2.0000



۴.۳ محاسبه بهره اطلاعات یا سود اطلاعاتی

- تابع `info_gain(parent, children)` را به صورت صحیح پیاده سازی کنید.(راهنمایی: `parent` لیستی از برچسب‌های مجموعه اصلی، `children` لیستی از زیرمجموعه‌های حاصل از تقسیم است که هرکدام شامل برچسب‌های کلاس خود هستند).

ما تابعی با عنوان `info_gain(parent, children)` پیاده‌سازی کردیم که بر پایه‌ی فرمول انتروپی تعریف می‌شود:

$$IG(S, A) = Entropy(S) - \sum weight_v \cdot Entropy(S_v)$$

پارامترها:

`parent`: برچسب‌های کلاسی کل مجموعه‌ی داده `children`: لیستی از زیرمجموعه‌های ایجاد شده از تقسیم مجموعه داده بر اساس یک ویژگی

نحوه عملکرد تابع:

۱. انتروپی کل مجموعه داده (والد) را محاسبه می‌کند.
۲. انتروپی هر زیرمجموعه (فرزنده) را با وزن نسبت اندازه‌اش محاسبه می‌کند.
۳. مجموع انتروپی‌های وزنی فرزندان را از انتروپی والد کم می‌کند تا سود اطلاعاتی به دست آید.

از این معیار در درخت تصمیم برای انتخاب ویژگی‌ای که بیشترین اطلاعات را برای تقسیم داده فراهم می‌کند استفاده می‌شود.

```
[ ] def info_gain(parent, children):  
  
    total_len = len(parent)  
    parent_entropy = calculate_entropy(parent)  
  
    weighted_entropy = 0  
    for child in children:  
        weight = len(child) / total_len  
        entropy_child = calculate_entropy(child)  
        weighted_entropy += weight * entropy_child  
  
    return parent_entropy - weighted_entropy  
  
[ ] # Example labels (parent dataset)  
parent_labels = ['low', 'medium', 'low', 'high', 'medium', 'high', 'low']  
  
# Split by a feature (example children groups)  
children_labels = [  
    ['low', 'low', 'low'],           # group A  
    ['medium', 'medium'],          # group B  
    ['high', 'high']              # group C  
]  
  
# Compute information gain  
ig = info_gain(parent_labels, children_labels)  
print(f"Information Gain: {ig:.4f}")  
  
⇒ Information Gain: 1.5567
```



۵.۳ درخت تصميم

همانطور که قبلا توضیح داده شد، درخت تصمیم یک مدل یادگیری نظارتی است که ساختاری مشابه یک درخت دارد و برای پیش‌بینی مقدار یک متغیر هدف استفاده می‌شود. این ساختار شامل گره‌ها شاخه‌هایی است که در هر گره، داده‌ها بر اساس یک ویژگی و مقدار مشخص تقسیم می‌شوند. این تقسیم‌ها به گونه‌ای انجام می‌شوند که داده‌ها به سمت گره‌های برگ با خلوص بیشتر هدایت شوند. فرآیند رشد درخت تا زمانی ادامه دارد که معیار توقف برآورده شود (مانند رسیدن به عمق مشخص یا تعداد نمونه کم در هر گره)

- مفهوم prune کردن در درخت‌های تصمیم گیری چیست؟ مزایا و معایب استفاده از این روش را ذکر کنید.
- به کمک تابع GridSearchCV یک مدل درخت تصمیم را آموزش داده و مقادیر بهینه برای پارامترها را بدست آورید. نحوه عملکرد این تابع را نیز به طور خلاصه توضیح دهید.
- درخت تصمیم نهایی خود را رسم کنید. برای این کار میتوانید از plot_tree استفاده کنید.
- آیا در مدل شما overfitting یا underfitting رخ داده است؟ به طور کلی چه زمانی این پدیده رخ می‌دهد؟ هر یک را توضیح دهید. همچنین چه راهکارهایی برای کنترل آن برای مدل‌های درخت تصمیم وجود دارد؟
- برای مدل، خروجی معیارهای ارزیابی را با استفاده از داده‌های تست گزارش کنید و تحلیل کنید.
- ماتریس درهم‌ریختگی را برای مدل پیاده‌سازی شده برای داده‌های تست^۱ رسم نموده و تحلیل کنید.

تحلیل درخت تصمیم و ارزیابی مدل

مفهوم prune در درخت‌های تصمیم چیست؟ مزایا و معایب آن را توضیح دهید.

Pruning یا هرس کردن در درخت تصمیم فرآیندی است برای حذف شاخه‌هایی که تأثیر زیادی بر دقت مدل ندارند. این کار باعث جلوگیری از overfitting می‌شود.

مزایا: - افزایش توان تعمیم مدل برای داده‌های جدید - کاهش پیچیدگی مدل

معایب: - در صورت هرس بیش از حد، خطر underfitting وجود دارد - نیاز به تنظیم و اعتبارسنجی دقیق دارد

استفاده از GridSearchCV برای آموزش مدل و بهینه‌سازی پارامترها

ما از تابع GridSearchCV برای جستجوی مقادیر بهینه پارامترهای درخت تصمیم استفاده کردیم. این تابع ترکیب‌های مختلفی از پارامترهای زیر را بررسی کرد:

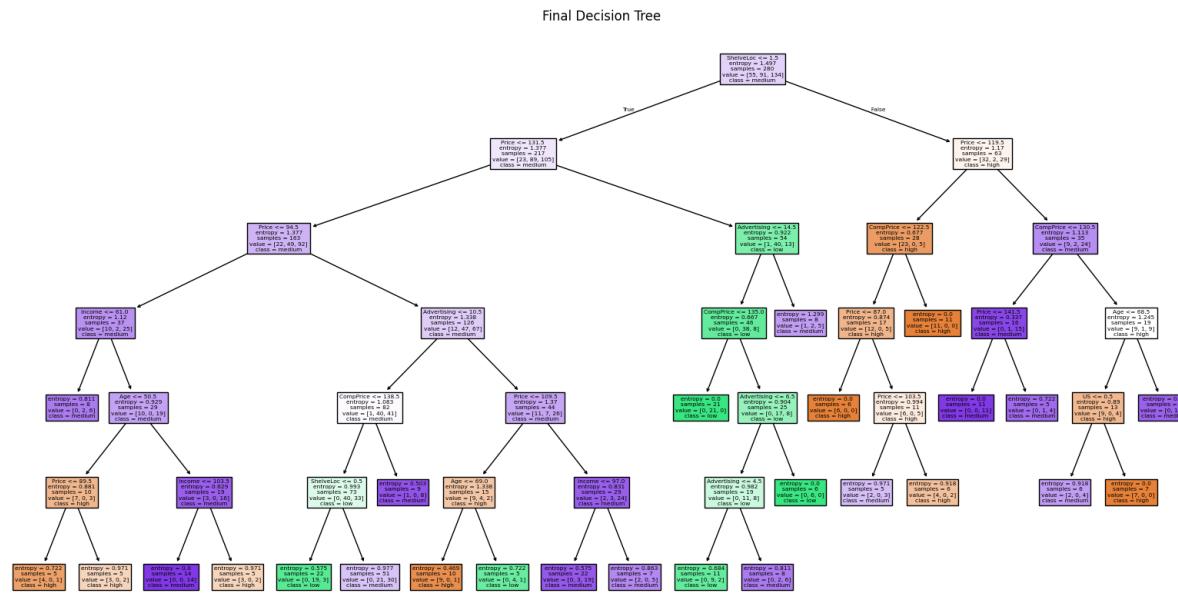
(entropy و gini و criterion - min_samples_leaf - min_samples_split - max_depth -

مدلی که بهترین دقت میانگین در cross-validation را داشت، انتخاب شد. بهترین پارامترها به شرح زیر بودند:

min_samples_split=2 ,min_samples_leaf=5 ,max_depth=6 ,criterion='entropy' و دقت میانگین برابر ۶۵% بود.

رسم درخت تصمیم نهایی

برای رسم درخت نهایی از تابع plot_tree در کتابخانه sklearn.tree استفاده کردیم. این تصویر ساختار درخت را نشان می‌دهد و کمک می‌کند فهم بهتری از نحوه تصمیم‌گیری مدل داشته باشیم.



بررسی وضعیت underfitting یا overfitting

مدل نهایی روی داده‌های تست دقت ۵۷.۵% را نشان داد. این مقدار نسبت به دقت cross-validation (که ۶۵% بود) کمتر است، اما اختلاف شدید نیست. این نشان می‌دهد مدل اندکی دچار underfitting است، یعنی نتوانسته پیچیدگی کامل داده‌ها را یاد بگیرد.

برای کنترل این وضعیت، از پارامترهایی مانند min_samples_leaf و max_depth pruning استفاده کردیم و فرآیند pruning نیز به طور ضمی در مدل اعمال شد.



معیار های ارزیابی و ماتریس درهم ریختگی در زیر قابل مشاهده هستند:

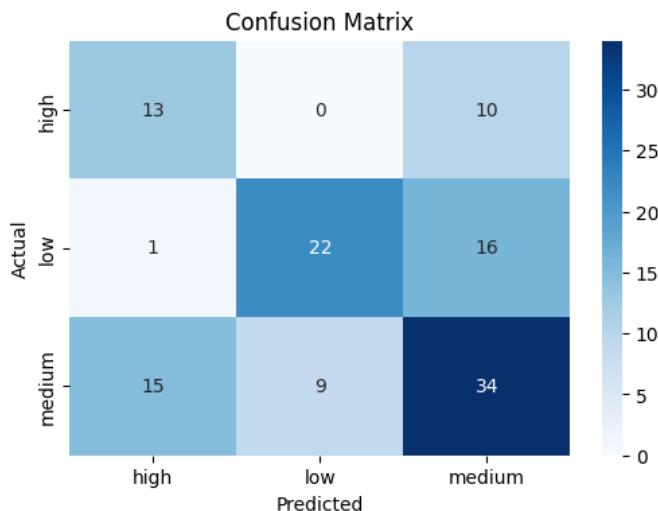
| classification_report | | | | |
|-----------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| high | 0.45 | 0.57 | 0.50 | 23 |
| low | 0.71 | 0.56 | 0.63 | 39 |
| medium | 0.57 | 0.59 | 0.58 | 58 |
| accuracy | | | 0.57 | 120 |
| macro avg | 0.57 | 0.57 | 0.57 | 120 |
| weighted avg | 0.59 | 0.57 | 0.58 | 120 |

ارزیابی عملکرد مدل بر اساس داده های تست

ماتریس درهم ریختگی نشان می دهد که مدل در تشخیص کلاس high عملکرد ضعیف تری نسبت به بقیه دارد. مقدار precision برای high برابر 0.45 و recall آن برابر 0.57 است.

به طور کلی:

- دقیق کلی مدل: 57% - F1-score → 0.50 .high → 0.63 .low → 0.58 .medium → 0.57 برای کلاس ها:



ماتریس درهم ریختگی مدل در تست: مدل بیشتر نمونه های medium را درست پیش بینی کرده، اما در تشخیص بین high و medium اشتباه هایی رخ داده که احتمالاً به دلیل هم پوشانی ویژگی هاست.

جمع بندی:

درخت تصمیم نهایی با استفاده از تنظیم بهینه پارامترها آموزش داده شد، با وجود اینکه مدل عملکرد متوسطی داشت، اما فرآیند ارزیابی و کنترل پیچیدگی مدل به خوبی پیاده سازی شده و نتایج قابل قبول هستند.