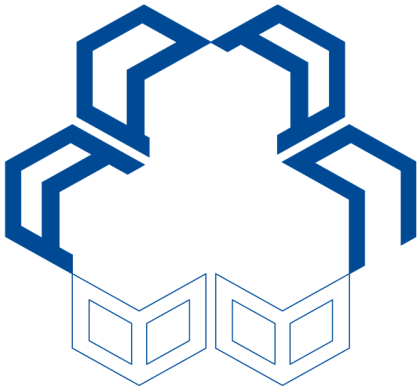


یادگیری ماشین

مینی پروژه دوم

نام و نام خانوادگی	امیر جهانگرد | علیرضا امیری
شمارهٔ دانشجویی	۴۰۳۱۵۳۸۴ | ۴۰۲۰۲۴۱۴
تاریخ	اردیبهشت ماه ۱۴۰۴

فهرست مطالب

۱ پرسش ۱

لینک درایو گوگل کولب حاوی کدهای این تمرین
لینک Github

۱.۱ الف

طبقه بندی بر اساس مدل بیز، یک روش مبتنی بر داده های آماری برای تفکیک کلاس های داده ها با در اختیار داشتن توزیع داده های هر کلاس می باشد. در تئوری، در صورت در اختیار داشتن توزیع دقیقی برای داده ها مطابق با معادلات حاکم بر بیز، می توانیم با بالاترین دقت دسته بندی را انجام دهیم. پیش از ادامه، معادله ی بیز را در رابطه ی زیر مشاهده می کنیم.

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)\,P(C_k)}{P(\mathbf{x})} \tag{۱}$$

که در آن،

- C_k دسته ی k،
- $\mathbf{x}$ بردار ویژگی های رویت شده ی یک نمونه،
- $P(C_k|\mathbf{x})$ خواسته ی مسئله که همان احتمال تعلق به دسته ی K پس از مشاهده ی ویژگی های x ،
- $P(\mathbf{x}|C_k)$ توزیع احتمال ویژگی ها در دسته ی k،
- $P(C_k)$ توزیع داده های دسته ی k،
- و $P(\mathbf{x})$ توزیع کلی تمام داده ها است.

بر اساس این رابطه، در صورت در اختیار داشتن تعدا بی نهایت داده، می توانیم احتمال تعلق داده های جدید به هر کلاس را به صورت قطعی محاسبه کرده و کلاسی که بیشترین احتمال را دارد به عنوان پاسخ قطعی انتخاب کنیم. بر اساس دانش آماری موجود در این روش، می توان دریافت که همواره در صورت برقرار بودن مفروضات این رابطه، پاسخ به دست آمده از آن بهینه خواهد بود. بر این مبنا، روش طبقه بندی بیز بهینه به صورت زیر مطرح می شود:

پیدا کردن کلاس C^* به گونه ای که $C^* = \arg\max_{C_k} P(C_k|\mathbf{x})$

با این حال، محاسبه ی مقدار فوق مستلزم دانستن $P(\mathbf{x}|C_k)$ و $P(C_k)$ می باشد که در واقعیت امکان پذیر نیست.

برای فائق آمدن بر این مشکل و ساده سازی معادلات، در روش طبقه بندی بر اساس بیز ساده فرض می شود که ویژگی ها همه مستقل از یکدیگر هستند. بنابراین خواهیم داشت:

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{n} P(x_i|C_k) \tag{۲}$$

آنگاه رابطه ی بیز به صورت زیر تقریب زده می شود:

$$P(C_k|\mathbf{x}) \propto P(C_k) \prod_{i=1}^{n} P(x_i|C_k) \tag{۳}$$

و در روش طبقه بندی مطابق با هدف زیر تلاش می کنیم:

$$C^* = \arg\max_{C_k} \left(P(C_k) \prod_{i=1}^{n} P(x_i|C_k) \right) \quad (۴)$$

بدین ترتیب، علی رغم فرض اشتباهی که در این رابطه در نظر گرفته شده است، می توانیم با در اختیار داشتن داده های محدود، با دقت مناسبی طبقه بندی را انجام دهیم. علاوه بر این، به دلیل ساده سازی های مفروض در این روش، generalization بیشتری نیز به دست خواهیم آورد.

۲.۱ ب

از میان انواع مختلف پیاده سازی های روش طبقه بندی بیز، در اینجا به بررسی سه روش ،Bernoulli Gaussian و multinomial می پردازیم.
به طور مختصر، روابط ریاضی هر یک به شرح زیر است:

- multinomial

$$P(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_{i=1}^{n} p_{ki}^{x_i}$$

که در آن p_{ki} توزیع احتمال کلمه ی امi در کلاس k است

- Bernoulli

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{1-x_i}$$

که در آن p_{ki} توزیع احتمال وجود داشتن کلمه ی امi در کلاس k است

- Gaussian

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{ki}^2}} \exp\left(-\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2} \right)$$

که در آن μ_{ki} میانگین ویژگی i در کلاس k و σ_{ki}^2 واریانس ویژگی i در کلاس k است.

بر این اساس، همان طور که در پروژه های انجام شده بر روی این دیتاست می توان مشاهده کرد، غالبا از روش multinomial استفاده شده است، از آنجا که در این روش، نه تنها حضور داشتن یا نداشتن، بلکه تعداد حضور کلمات در متن پیامک لحاظ شده است که در مقایسه با روش برنولی، مناسب تر به نظر می رسد. علاوه بر این، استفاده از روشی گوسی در مواقعی مناسب است که داده ها دارای مقادیر پیوسته باشند که در مورد این دیتاست قابل پیاده سازی نیست.

۳.۱ ج. پیاده سازی مدل طبقه بندی به صورت دستی

چنان که در بخش قبل نتیجه گیری شد، پیاده سازی مدل multinomial به عنوان مدل بهینه ی طبقه بندی برای دیتاست Spam که دارای محتوای حروف می باشد برای پیاده سازی انتخاب شده است. در این بخش، به پیاده سازی این مدل بدون استفاده از پکیج های موجود خواهیم پرداخت.

۱.۳.۱ بارگذاری دیتاست

در بخش اول، دیتاست را با استفاده از کتابخانه ی pandas داخل محیط پایتون بارگذاری کرده و داده های ابتدایی آن را مشاهده می کنیم. (شکل ۱)

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

شکل ۱: ساختار دیتاست خام

با بررسی این ساختار، متوجه می شویم که برچسب های دیتاست در ستون اول و محتوای پیام ها در ستون دوم قرار گرفته است. علاوه بر این، تعداد ۵۰ نمونه نیز در ستون ستوم دارای محتوای متنی هستند که از آنها صرف نظر کرده و حذف شان می کنیم. در پایان، پس از نام گذاری ستون ها و حذف مقادیر نامعتبر ،NaN دیتافریم به صورت نمایش داده شده در شکل شکل ۲ به دست می آید.

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

شکل ۲: دیتافریم اصلاح شده

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
✓ 0.0s
((4457, 8265), (1115, 8265), (4457,), (1115,))
```

شکل ۳: داده های تست و آموزش پس از پیش پردازش

۲.۳.۱ پیش پردازش

تا به اینجا، دیتاست شامل داده های پیام ها و برچسب ها مشخص شده اند. با این حال، برای استفاده در مدل های یادگیری ماشین، لازم است داده ها از حالت حروف خارج شده و شامل محتوای عددی باشند.

برای این منظور، در طی چند مرحله متن ها را به محتوای عددی تبدیل می کنیم.

۱. فیلتر کردن متن ها

در ابتدا، برای خالص سازی متن های موجود، نیاز داریم تا کلمات موجود در آنها را از فیلتر هایی بگذرانیم. برای این منظور با استفاده از پکیج NLTK به انتخاب و حذف این عوامل می پردازیم.

- حذف کاراکتر های خاص مانند @!
- کوچک کردن تمامی حروف
- حذف عبارات پرکاربرد مانند ،The ،is and

۲. تغییر برچسب های متنی به عددی

در اینجا، برچسب ها را به صورت زیر تغییر می دهیم و نگاشت می کنیم.

$$\text{label} = \begin{cases} 0 & \text{iflabel} = \text{"ham"} \\ 1 & \text{iflabel} = \text{"spam"} \end{cases}$$

۳. توکن بندی

در این بخش که گامی مهم در پیش پردازش دادگان و استخراج ویژگی است، کلمات مورد استفاده در متن پیام ها به صورت عددی مطابق آنچه که پس از این توضیح داده می شود تبدیل می شوند. برای این منظور، از روش CountVectorizer در پکیج Sklearn استفاده می شود. این روش، به هر کلمه که به عنوان ورودی داده می شود، یک توکن ثبت کرده و در نهایت لغت نامه ای شامل تمام کلماتی که با آن آموزش دیده است ایجاد می کند. زین پس، با دریافت هر جمله یا متن، با شمردن تعداد حضور هر کلمه در متن، عددی را برای توکن متناظر قرار می دهد که ما می توانیم از این توکن ها به عنوان ویژگی ها برای آموزش مدل استفاده کنیم.

پیش از اجرای این مرحله، لازم به ذکر است که برای جلوگیری از نشت اطلاعات از داده های Train به داده های ،Test ما تنها می توانیم فرایند tokenization را بر روی داده های Train انجام دهیم و سپس به داده های Test منتقل کنیم. بنابراین، ابتدا داده ها را به دو دسته ی آموزش و تست تقسیم کرده و سپس vectorizer را بر روی آن آموزش می دهیم.

در پایان این مرحله، داده های تست و آموزش به صورت نمایش داده شده در شکل ۳ به دست می آیند.

۳.۳.۱ آموزش مدل MultiNB به صورت دستی

به منظور پیاده سازی دستی این کد، کلاسی به عنوان MultiNB تعریف می شود که در ادامه به توضیح روش های آن خواهیم پرداخت:

- prior

در این کلاس، مطابق رابطه ی زیر، مقدار احتمال پیشین هر دسته به صورت زیر محاسبه می شود.

$$P(y=c) = \frac{\text{count}(y=c)}{n_{\text{samples}}} \tag{۵}$$

- fit

در این روش، به ازای هر کلاس مجموع تعداد هر ویژگی و مجموع تمام ویژگی ها در هر کلاس به صورت زیر به دست می آید.

$$N_{c,i} = \sum_{x \in D_c} x_i \qquad \text{and} \qquad N_c = \sum_i N_{c,i} \tag{۶}$$

- theta

در این بخش، تابع مشابهت likelihood داده های هر کلاس مطابق رابطه ی زیر محاسبه می شود.

$$\theta_{c,i} = P(x_i \mid y=c) = \frac{N_{c,i} + \alpha}{N_c + \alpha \cdot d} \tag{۷}$$

آنگاه با فرض استقلال ویژگی ها از یکدیگر، توزیع احتمال هر کلاس به صورت زیر محاسبه می شود.

$$P(\mathbf{x} \mid y=c) = \prod_{i=1}^{d} \theta_{c,i}^{x_i} \tag{۸}$$

لازم به ذکر است که در این روابط، از مقدار α برای جلوگیری از بروز خطا در شرایطی که داده های موجود در دیتاهای تستی در توکن های آموزش دیده وجود نداشته باشند قرار داده شده است.

- predict

در نهایت در این روش، احتمال پسین، یعنی احتمال تعلق داشتن به هر دسته، با فرض در اختیار داشتن ماتریس ویزگی ها را محاسبه کرده و در نهایت، کلاسی با بیشترین احتمال را به عنوان خروجی اعلام می کنیم. رابطه ی احتمال در این بخش از کد به صورت زیر محاسبه می شود.

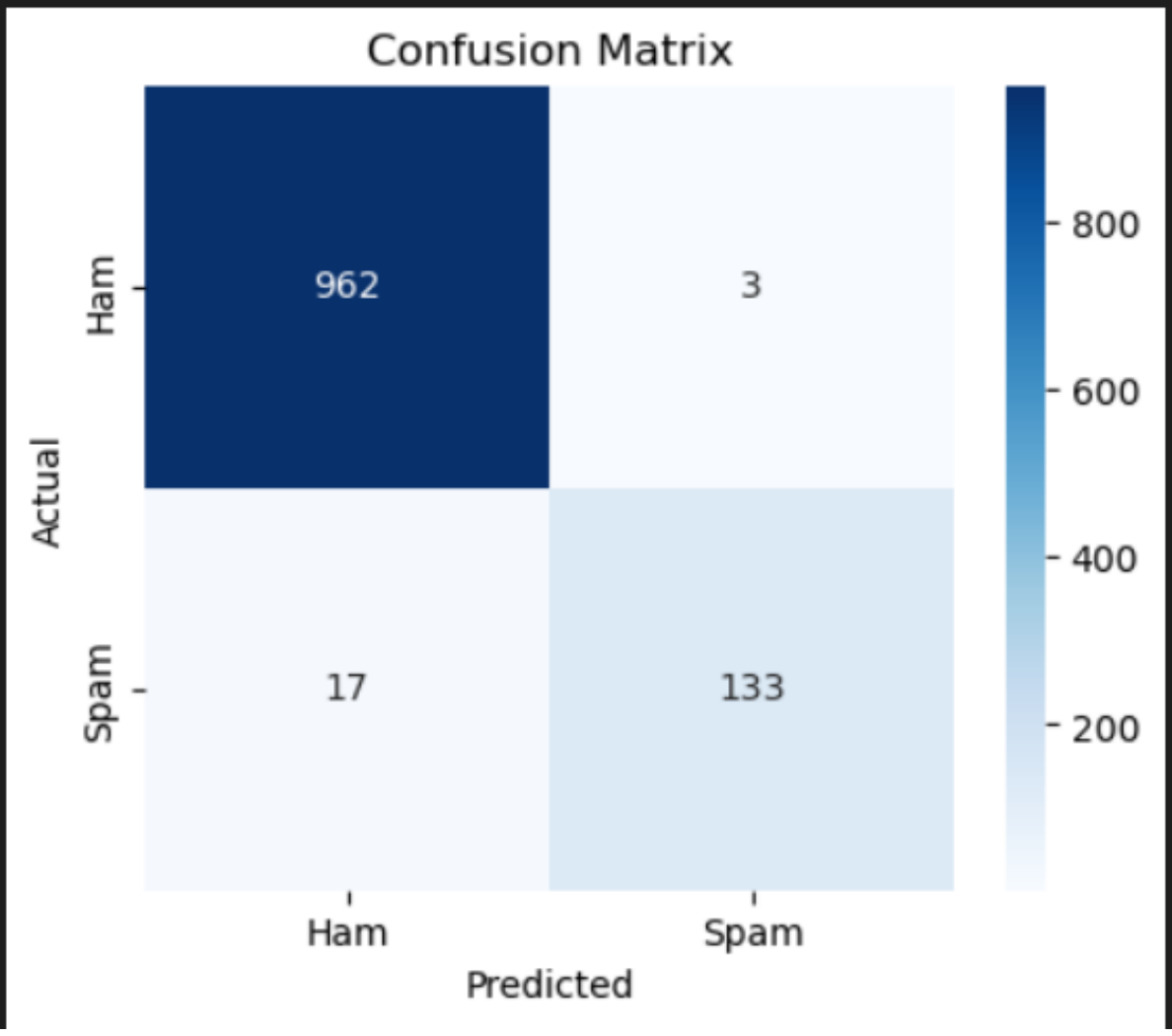
$$P(y=c \mid \mathbf{x}) = \frac{P(y=c) \cdot P(\mathbf{x} \mid y=c)}{\sum_{c'} P(y=c') \cdot P(\mathbf{x} \mid y=c')} \tag{۹}$$

$$\hat{y} = \arg\max_c \; P(y=c \mid \mathbf{x}) \tag{۱۰}$$

۴.۳.۱ آموزش مدل

با تعریف مدل در بخش قبل، می توانیم به پیاده سازی آن بر روی دیتاست بپردازیم. پس از آموزش مدل با دستور fit و آزمودن مدل بر داده های تستی خواهیم داشت:شکل ۴

```
Accuracy: 0.9821

Classification Report:
              precision    recall  f1-score   support

         Ham       0.98      1.00      0.99       965
        Spam       0.98      0.89      0.93       150

    accuracy                           0.98      1115
   macro avg       0.98      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

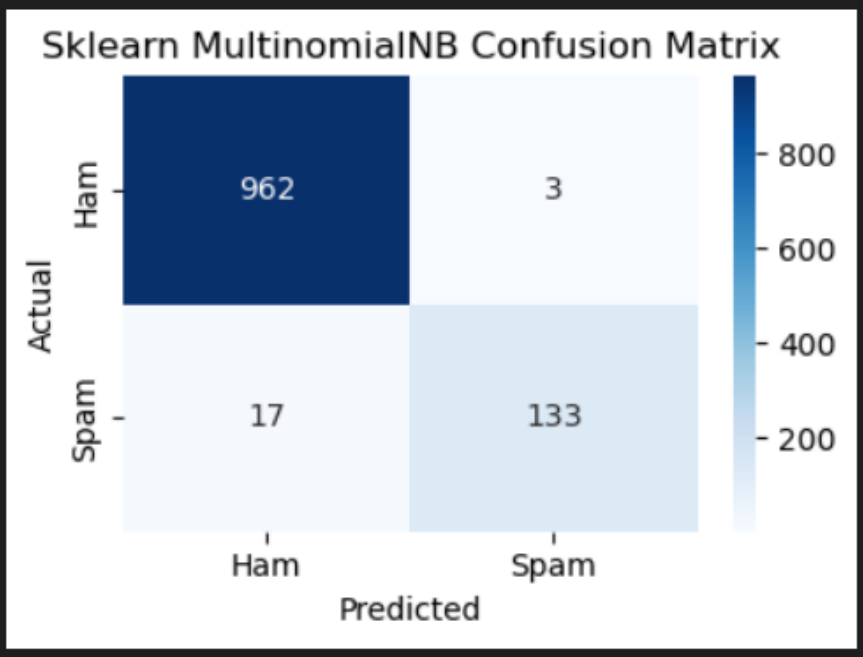


شکل ۴: گزارش طبقه بندی برای مدل MultiNB دستی

۴.۱ د. پیاده سازی دو مدل طبقه بندی با استفاده از sklearn

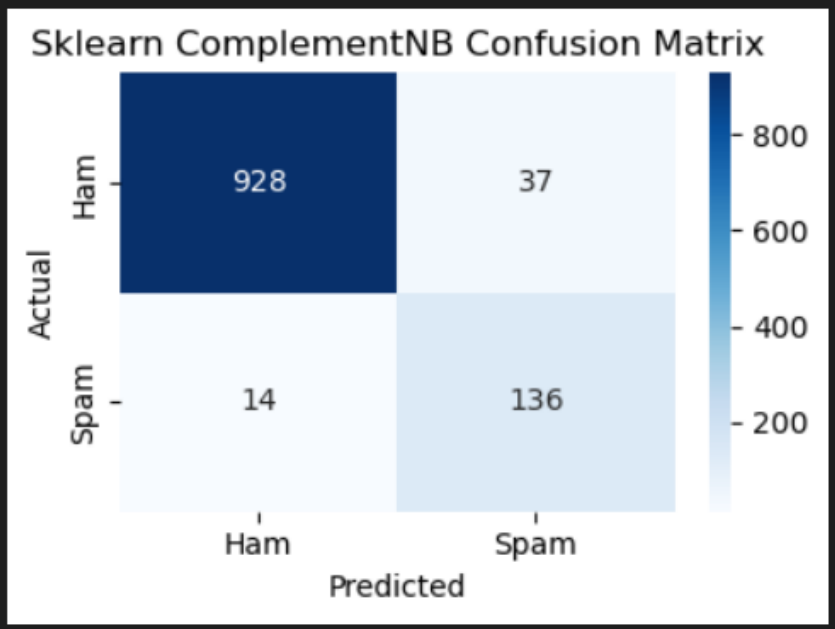
با در اختیار داشتن دیتاست چنان که در بخش پیشین توضیح داده شد، می توان مدل های دیگری از کتابخانه ی sklearn را بر روی داده ها آموزش داد. در اینجا، دو مدل MultinomialNB و ComplementNB انتخاب شده و داده ها بر اساس آن آموزش داده می شوند. گزارش طبقه بندی این مدل ها به شرح زیر خواهد بود. شکل ۵ ، شکل ۶

```
--- Sklearn MultinomialNB ---
Accuracy: 0.9820627802690582
              precision    recall  f1-score   support

         Ham       0.98      1.00      0.99       965
        Spam       0.98      0.89      0.93       150

    accuracy                           0.98      1115
   macro avg       0.98      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

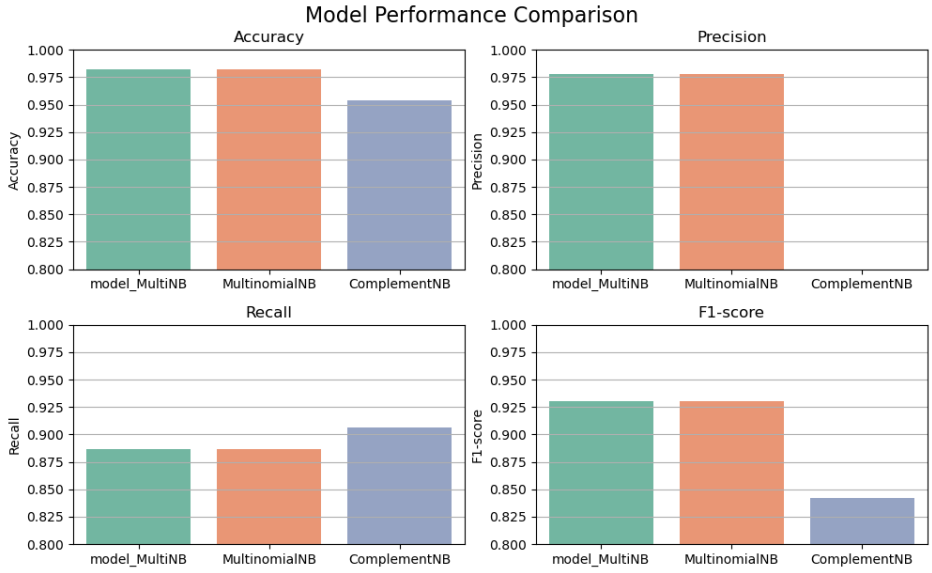


شکل ۵: گزارش طبقه بندی مدل MultiNB با استفاده از sklearn

```
--- Sklearn ComplementNB ---
Accuracy: 0.9542600896860987
              precision    recall  f1-score   support

         Ham       0.99      0.96      0.97       965
        Spam       0.79      0.91      0.84       150

    accuracy                           0.95      1115
   macro avg       0.89      0.93      0.91      1115
weighted avg       0.96      0.95      0.96      1115
```



شکل ۶: گزارش طبقه بندی مدل ComplementNB با استفاده از sklearn



شکل ۷: مقایسه عملکرد مدل ها

پس از این، برای مقایسه ی عملکرد مئدل ها با یکدیگر خواهیم داشت. شکل ۷

از مقایسه این داده ها و بررسی پارامتر های گزارش شده، در می یابیم که اگرچه مدل ComplementNB دارای امتیاز recall بیشتری است، اما دقت آن در تشحیص پیام های spam تنها ۷۰ درصد است که نشان می دهد این مدل توانسته spam های بیشتری را تشخیص دهد، اما با قیمت اینکه تعداد بسیار بیشتری از تایپ های ham را نیز به عنوان spam تشخیص داده باشد. به غیر از این، دو مدل دیگر که مربوط به multiNB هستند عملکرد یکسانی دارند و این نشان می دهد که پیاده سازی دستی و با استفاده از پکیج با یکدیگر مطابقت دارند.

```
Custom MultiNB       | Total Risk: 88 | Average Risk: 0.0789
sklearn MultiNB      | Total Risk: 88 | Average Risk: 0.0789
ComplementNB         | Total Risk: 107 | Average Risk: 0.0960
```

شکل ۸: نتایج تابع ریسک

۵.۱ ه. اثبات ریاضی بهینه بودن بیز

فرض کنید مجموعه ویژگی‌ها $\mathcal{X}$ و مجموعه برچسب‌ها $\mathcal{Y} = \{1, 2, \ldots, K\}$ باشد.
برای هر $x \in \mathcal{X}$، هدف انتخاب برچسب $\hat{y}(x)$ به گونه‌ای است که احتمال خطا کمینه شود:

$$P(\hat{y}(x) \neq y)$$

با داشتن توزیع پسین $P(y \mid x)$، اگر برچسب a را به x نسبت دهیم، آنگاه خطای مورد انتظار:

$$E[\text{error} \mid x, a] = \sum_{y \in \mathcal{Y}, y \neq a} P(y \mid x) = 1 - P(a \mid x)$$

است. بنابراین برای کمینه کردن خطا باید $P(a \mid x)$ را بیشینه کنیم. در نتیجه قانون تصمیم‌گیری بهینه:

$$\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} P(y \mid x)$$

است که قانون بیز نام دارد.
در نهایت، قانون تصمیم‌گیری بیز با تابع زیان $1 - 0$، احتمال خطا را به صورت زیر کمینه می‌کند:

$$\min_{\hat{y}(x)} P(\hat{y}(x) \neq y)$$

۱.۵.۱ و. محاسبه ی تابع ریسک و تعیین بهترین مدل

در این بخش، معیار دیگری برای ارزیابی عملکرد مدل ها معرفی شده است. بر اساس توضیحات داده شده، علاوه بر معیار های دقت و صحت، معیار هزینه نیز مطرح است. در این حالت، به ازای هر گزارش FP، ۵ برابر هزینه ی بیشتری پرداخت میشود در مقایسه با حالت TN. بنابراین، با محاسبه ی تعداد و مقدار حالت های خطا که برای سیستم ریسک ایجاد می کنند و در نهایت محاسبه ی هزینه ی آن، می توانیم بهترین مدل را به دست بیاوریم. بنابراین، ریسک ها به صورت زیر محاسبه می شوند.

$$\text{Risk Average} = \frac{5 \cdot FN + 1 \cdot FP}{N} \tag{۱۱}$$

در نتیجه، با محاسبه ی این مقدار بر مدل های آموزش داده شده خواهیم داشت. شکل ۸

بر اساس این داده ها، مشاهده می کنیم که اگرچه مدل complement تعداد بیشتری از پیامک های spam را تشخیص داده است، با این حال هزینه ای که بابت خطاهای دیگر این مدل پرداخت شده است بیشتر بوده و مدل را در معرض ریسک بیشتری قرار می دهد. بنابراین، همچنان در این مسئله مدل Multi پیشنهاد می شود.

۲ پرسش ۲

لینک کولب حاوی کدهای این پرسش

۱.۲ آ

توزیع داده‌ها:

تصاویر در دیتاست MNIST معمولاً مقادیر پیکسل‌هایی از ۰ تا ۲۵۵ دارند که به صورت تقریباً نرمال توزیع می‌شوند.

با استفاده از StandardScaler، داده‌ها به گونه‌ای تغییر می‌کنند که میانگین هر ویژگی (هر پیکسل) برابر صفر و انحراف معیار آن برابر یک خواهد شد. این عمل برای مدل‌هایی که به توزیع داده‌ها و نرمال بودن آن‌ها حساس هستند، مانند SVM یا خطی رگرسیون، مناسب است.

الگوریتم‌های استفاده شده:

اگر قصد استفاده از الگوریتم‌هایی مثل SVM، KNN، خطی رگرسیون یا عصبی شبکه‌های را داشته باشیم، StandardScaler به دلیل نرمال کردن داده‌ها با توجه به میانگین و انحراف معیار، عملکرد بهتری خواهد داشت.

به‌ویژه برای مدل‌هایی که برای محاسبه فاصله (مثل KNN) یا تصمیم‌گیری بر اساس توزیع داده‌ها (مثل SVM) طراحی شده‌اند، این نرمال‌سازی بسیار مفید است.

چرا MinMaxScaler مناسب نیست؟

تصاویر MNIST معمولاً مقادیر گسترده‌ای از پیکسل‌ها دارند (از ۰ تا ۲۵۵) و نرمال‌سازی آن‌ها به بازه [0, 1] با MinMaxScaler ممکن است باعث از دست رفتن اطلاعات مربوط به توزیع واقعی داده‌ها شود. به عبارت دیگر، اگر داده‌ها به شدت نامتوازن یا با ویژگی‌هایی با مقادیر بسیار متفاوت باشند، MinMaxScaler ممکن است باعث فشرده شدن داده‌ها به بازه کوچکی شود و اطلاعات ارزشمند از بین برود.

استفاده در شبکه‌های عصبی: در صورتی که مدل شما از نوع عصبی شبکه باشد، MinMaxScaler ممکن است مناسب باشد، ولی در اکثر مواقع با توجه به توزیع نرمال داده‌های MNIST، StandardScaler بهتر عمل می‌کند.

۲.۲ ب

پس از اعمال کد پایتون نتیجه و خروجی ما مطابق شکل ۹ شد.

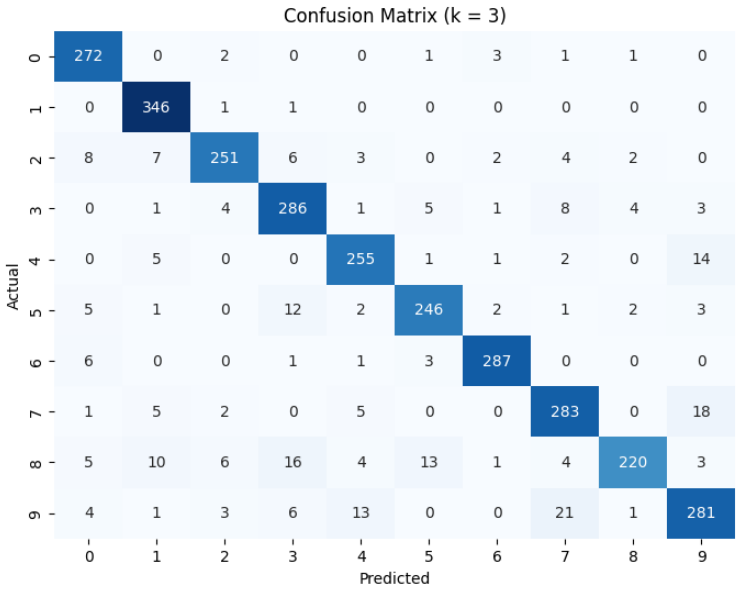
```
Training data shape: (7000, 784)
Testing data shape: (3000, 784)
```

شکل ۹: تقسیم داده ها به آموزش و آزمون

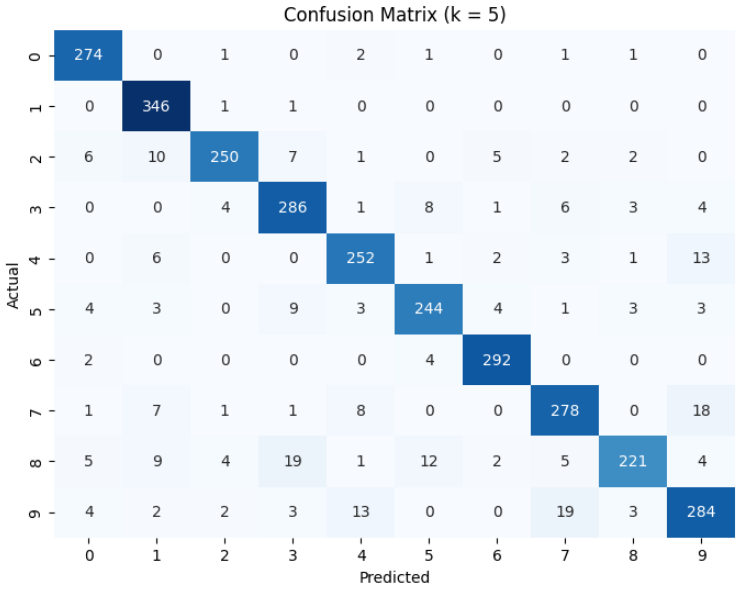
۳.۲ ج

نتایج ما برای این بخش مطابق زیر است:

حال باید بررسی کنیم که مدل ما برای کدام مقدار k از ۱ تا ۲۵ مناسب است. نتایج ما در شکل ۱۳ قابل مشاهده است. طبق این شکل بهترین k برای ما مقدار ۳ و ۵ می باشد.



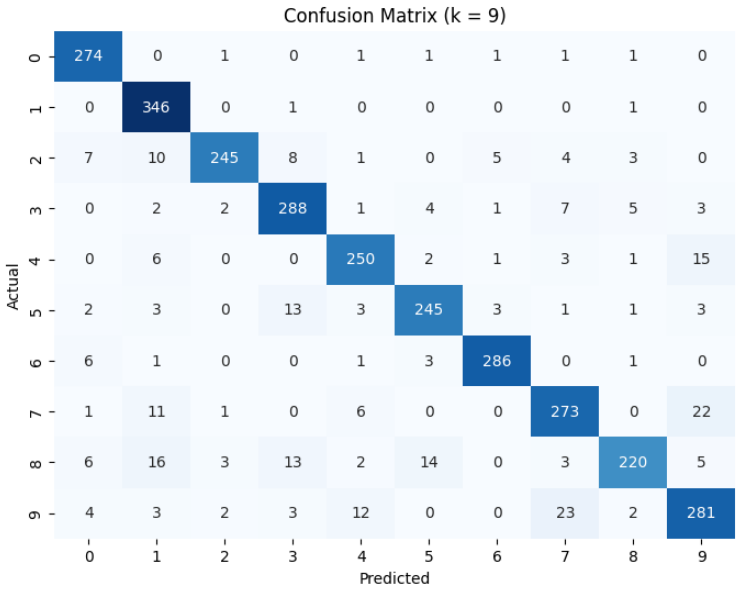
شکل ۱۰: $k = ۳$ ، Accuracy: ۹۰.۹۰%



شکل ۱۱: $k = ۵$ ، Accuracy: ۹۰.۹۰%

۴.۲ د

نمودار خواسته شده از ما برای این بخش در شکل ۱۴ مشاهده می شود. همانطور که دیده می شود بهترین مقدار k برای ما ۳ و ۵ است.



شکل ۱۲: $k = ۹$ ، Accuracy: ۹۰.۲۷%

```
Accuracy for k=1: 90.70%
Accuracy for k=3: 90.90%
Accuracy for k=5: 90.90%
Accuracy for k=7: 90.53%
Accuracy for k=9: 90.27%
Accuracy for k=11: 89.93%
Accuracy for k=13: 89.37%
Accuracy for k=15: 89.37%
Accuracy for k=17: 89.23%
Accuracy for k=19: 88.87%
Accuracy for k=21: 88.90%
Accuracy for k=23: 88.70%
Accuracy for k=25: 88.57%
```

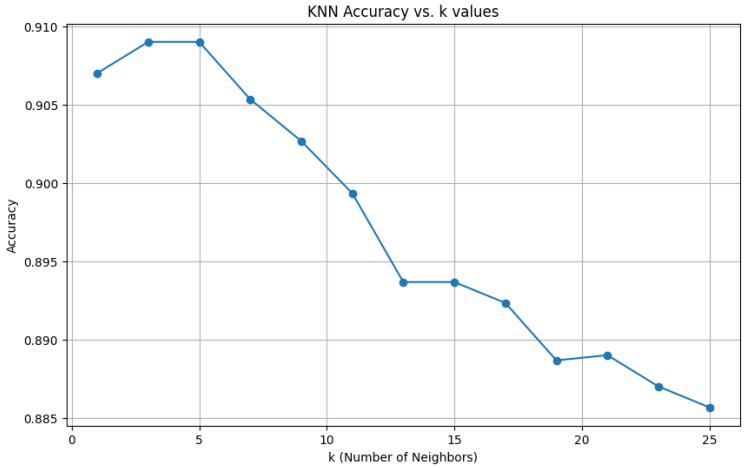
شکل ۱۳: ارزیابی مدل برای k های مختلف

۵.۲ ه

چرا PCA می‌تواند موثر باشد؟

الگوریتم K-Nearest Neighbors (KNN) به شدت به ابعاد بالا حساس است که این پدیده به نفرین ابعاد بالا یا Curse of Dimensionality معروف است. در فضای با ابعاد زیاد، فاصله بین داده‌ها کمتر معنی‌دار می‌شود و کارایی مدل‌های یادگیری کاهش می‌یابد.

روش PCA با کاهش تعداد ابعاد ویژگی‌ها می‌تواند این مشکل را تا حد زیادی حل کند. مزایای استفاده از PCA عبارتند از:



شکل ۱۴: نمودار k های مختلف برای ارزیابی

- فاصله‌ها معنی‌دارتر می‌شوند: با کاهش ابعاد، داده‌ها در فضایی فشرده‌تر و معنادارتر قرار می‌گیرند، و الگوریتم KNN بهتر می‌تواند همسایگان نزدیک را تشخیص دهد.
- نویز حذف می‌شود: مؤلفه‌های کم‌اهمیت که معمولاً حامل نویز هستند، حذف می‌شوند و فقط مؤلفه‌های مهم باقی می‌مانند.
- محاسبات سریع‌تر می‌شوند: کاهش ابعاد منجر به کاهش بار محاسباتی می‌شود که در حجم بالای داده‌ها بسیار مؤثر است.

در نتیجه، با اعمال PCA ممکن است مدل KNN دقت بالاتری کسب کند و زمان اجرای کمتری نیز داشته باشد.
حال طبق این توضیحات به بررسی سوال می پردازیم. ابتدا مدل را برای مقادیر مختلف مولفه های اصلی بررسی می کنیم. ارزیابی ما در شکل ۱۵ قابل مشاهده است.

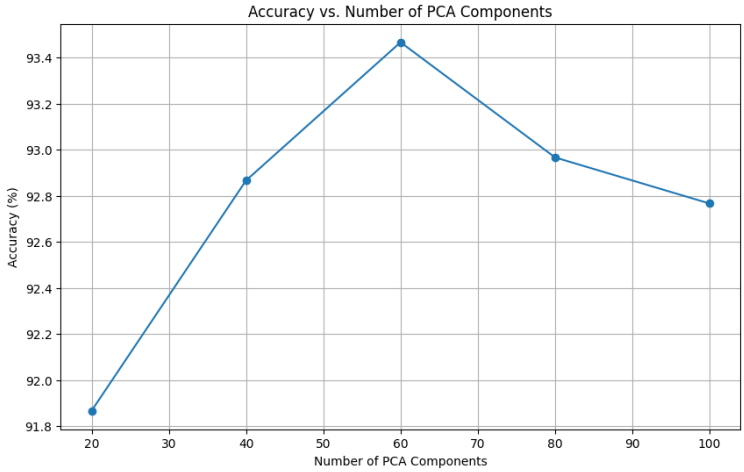
```
PCA Components: 20 -> Accuracy: 91.87%
PCA Components: 40 -> Accuracy: 92.87%
PCA Components: 60 -> Accuracy: 93.47%
PCA Components: 80 -> Accuracy: 92.97%
PCA Components: 100 -> Accuracy: 92.77%
```

شکل ۱۵: عملکرد مدل برای مولفه های مختلف

سپس در شکل ۱۶ نتیجه نمودار را مشاهده می کنیم. مشاهده می شود که بهترین معیار مولفه اصلی ۶۰ می باشد.

۳ پرسش ۳

لینک کولب حاوی کدهای این پرسش



شکل ۱۶: نمودار عملکرد مدل برای مولفه های مختلف

۱.۳

طبق خواسته سوال، ۱۰ سطر اول دیتاست مطابق شکل ۱۷ است.

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	\
0	9.50	138	73	11	276	120	Bad	42	
1	11.22	111	48	16	260	83	Good	65	
2	10.06	113	35	10	269	80	Medium	59	
3	7.40	117	100	4	466	97	Medium	55	
4	4.15	141	64	3	340	128	Bad	38	
5	10.81	124	113	13	501	72	Bad	78	
6	6.63	115	105	0	45	108	Medium	71	
7	11.85	136	81	15	425	120	Good	67	
8	6.54	132	110	0	108	124	Medium	76	
9	4.69	132	113	0	131	124	Medium	76	

	Education	Urban	US
0	17	Yes	Yes
1	10	Yes	Yes
2	12	Yes	Yes
3	14	Yes	Yes
4	13	Yes	No
5	16	No	Yes
6	15	Yes	No
7	10	Yes	Yes
8	10	No	No
9	17	No	Yes

شکل ۱۷: ۱۰ سطر اول دیتاست

۲.۳ پیش پردازش دیتاست

۱.۲.۳ بررسی داده های ناقص یا گمشده

در شکل ۱۸ نتیجه کد بررسی داده های ناقص را مشاهده می کنیم. طبق این نتیجه همانطور که میبینیم دیتاست ما هیچ داده ناقصی ندارد.

```
Missing data in each column:
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64

No missing data found.
```

شکل ۱۸: بررسی داده های ناقص در دیتاست

۲.۲.۳ بررسی داده های تکراری

توضیحات درباره داده‌های تکراری:

داده‌های تکراری در دیتاست‌ها می‌توانند مشکلات مختلفی برای مدل‌های یادگیری ماشین و تحلیل داده‌ها ایجاد کنند. برخی از این مشکلات عبارتند از:

- اثرگذاری بیش از حد بر مدل: داده‌های تکراری باعث می‌شوند که برخی نمونه‌ها وزن زیادی در مدل پیدا کنند، که می‌تواند باعث شود مدل بر اساس این نمونه‌های تکراری بیش از حد تطبیق پیدا کند .(Overfitting)
- کاهش دقت مدل: داده‌های تکراری می‌توانند دقت مدل را کاهش دهند، زیرا مدل ممکن است تمرکز زیادی بر روی داده‌های تکراری بگذارد و از ویژگی‌های دیگر دیتاست به درستی استفاده نکند.

- منحرف کردن تحلیل‌ها: در تحلیل داده‌ها، داده‌های تکراری می‌توانند نتایج نادرستی ایجاد کنند، به طوری که آنالیزها نمی‌توانند به درستی نتایج مطلوب را ارائه دهند.
- کاهش کارایی: در مدل‌هایی که به زمان محاسباتی حساس هستند، داده‌های تکراری می‌توانند زمان آموزش و پیش‌بینی را افزایش دهند و در نتیجه کارایی مدل را کاهش دهند.

در شکل ۱۸ نتیجه کد بررسی داده های تکراری را مشاهده می کنیم. طبق این نتیجه همانطور که میبینیم دیتاست ما هیچ داده تکراری ندارد.

```
Number of duplicate rows: 0

No duplicate data found.
```

شکل ۱۹: بررسی داده های تکراری در دیتاست

تبدیل ویژگی‌های دسته‌ای به ویژگی‌های عددی:

برای تبدیل ویژگی‌های دسته‌ای (categorical features) به ویژگی‌های عددی (numerical features) در مدل‌های یادگیری ماشین، می‌توان از چند روش مختلف استفاده کرد. متداول‌ترین روش‌ها برای انجام این تبدیل عبارتند از:

- Encoding One-Hot: برای ویژگی‌هایی که مقادیر دسته‌ای محدود دارند، استفاده از Encoding One-Hot بسیار رایج است. این روش برای هر مقدار از ویژگی یک ستون جدید ایجاد می‌کند که در آن مقدار "۱" برای آن مقدار خاص و "۰" برای بقیه مقادیر قرار می‌گیرد.
- Encoding Label: اگر ویژگی‌ها تنها شامل چند مقدار مختلف باشند (مثلاً "شهر" و "غیرشهر" برای ویژگی محل فروش)، می‌توان از Encoding Label استفاده کرد. این روش هر مقدار را به یک عدد منحصر به فرد تبدیل می‌کند.

راه‌حل پیشنهادی:

برای ویژگی‌های دسته‌ای مانند محل فروش محصول (شهر یا غیرشهر) و فروخته شدن در آمریکا یا خارج از آمریکا، می‌توان از Encoding Label یا Encoding One-Hot استفاده کرد. در صورتی که تعداد مقادیر دسته‌ای محدود باشد (برای مثال فقط دو دسته "شهر" و "غیرشهر" وجود داشته باشد)، استفاده از Encoding Label می‌تواند مناسب باشد. اما برای ویژگی‌هایی که چندین دسته دارند، بهتر است از Encoding One-Hot استفاده کنیم.

در شکل ۲۰ نتیجه کد پایتون برای این بخش را مشاهده می کنیم. طبق توضیحات، داده های بخش US که شامل Yes و No بود به ترتیب به ۱ و ۰ تبدیل شدند. همچنین داده های ستون Urban نیز که Yes و No بودند به ترتیب به ۱ و ۰ تبدیل شدند. در نهایت داده های ستون ShelveLoc که شامل Good و Bad و Medium بودند به ترتیب به ۱ و ۰ و ۲ تبدیل شدند. شکل ۲۰ خروجی اصلی و نهایی جدول ما می باشد.

۳.۲.۳ تبدیل متغیر هدف به متغیر دسته ای

برای تبدیل متغیر هدف (که در اینجا sales است) به متغیر دسته‌ای (categorical variable)، ابتدا باید میزان فروش (متغیر عددی) را به دسته‌های مختلف تقسیم کنیم. این کار می‌تواند با استفاده از تکنیک‌هایی مانند تقسیم بازه‌ای (binning) انجام شود.

پیشنهاد بازه‌ها برای دسته‌بندی:

با توجه به میانگین ۷.۴۹، می‌توانیم `sales` را به دسته‌های زیر تقسیم کنیم:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	\
0	9.50	138	73	11	276	120	0	42	
1	11.22	111	48	16	260	83	1	65	
2	10.06	113	35	10	269	80	2	59	
3	7.40	117	100	4	466	97	2	55	
4	4.15	141	64	3	340	128	0	38	
5	10.81	124	113	13	501	72	0	78	
6	6.63	115	105	0	45	108	2	71	
7	11.85	136	81	15	425	120	1	67	
8	6.54	132	110	0	108	124	2	76	
9	4.69	132	113	0	131	124	2	76	

	Education	Urban	US
0	17	1	1
1	10	1	1
2	12	1	1
3	14	1	1
4	13	1	0
5	16	0	1
6	15	1	0
7	10	1	1
8	10	0	0
9	17	0	1

شکل ۲۰: مشاهده ۱۰ سطر اول دیتاست که همه داده های آن عددی هستند.

- کم فروش Low) (Sales: مقادیری که کمتر از میانگین هستند.
- متوسط فروش Medium) (Sales: مقادیری که در نزدیکی میانگین قرار دارند.
- زیاد فروش High) (Sales: مقادیری که بیشتر از میانگین هستند.

برای انجام این کار، پیشنهاد می‌شود که محدوده‌ها را بر اساس درصدی از میانگین و دامنه داده‌ها تنظیم کنیم.
پیشنهاد بازه‌ها:

- کم فروش (Low): `sales` کمتر از ۴.۵
- متوسط فروش (Medium): `sales` بین ۴.۵ و ۱۰
- زیاد فروش (High): `sales` بیشتر از ۱۰

سپس در شکل ۲۱ ۱۰ سطر اول داده های فروش و داده های دسته بندی شده را مشاهده می کنیم.
در نهایت این داده های متنی را به داده های عددی تبدیل می کنیم. بدین صورت که Low به ۰، Medium به ۱ و High به ۲ تبدیل شدند. ۱۰ سطر اول ما در شکل ۲۲ قابل مشاهده است.

۴.۲.۳ ماتریس همبستگی

ماتریس همبستگی ما مطابق شکل ۲۳ می باشد.

	Sales	Sales_Category
0	9.50	Medium
1	11.22	High
2	10.06	High
3	7.40	Medium
4	4.15	Low
5	10.81	High
6	6.63	Medium
7	11.85	High
8	6.54	Medium
9	4.69	Medium

شکل ۲۱: مشاهده ۱۰ سطر اول دیتاست داده های فروش و داده های دسته شده

۵.۲.۳ بررسی همبستگی ها

در شکل ۲۴ ما ترتیب میان بیشترین همبستگی ما دسته بندی Sales را داریم. طبق این شکل بیشترین میزان همبستگی این دسته با Sales ، Adver-tising ، ShelveLoc ، US و Income می باشد. همچنین همبستگی میان CompPrice با Price و US با Advertising نیز زیاد و قابل توجه است.

۳.۳ محاسبه آنتروپی داده ها

آنتروپی (Entropy) یک معیار برای اندازه‌گیری عدم‌قطعیت یا بی‌نظمی در داده‌ها است. در مدل‌های یادگیری ماشین، محاسبه آنتروپی می‌تواند به ما کمک کند تا میزان اطلاعات یا عدم‌قطعیت موجود در یک مجموعه داده را ارزیابی کنیم.

برای محاسبه آنتروپی یک مجموعه از داده‌ها، از فرمول زیر استفاده می‌کنیم:

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

که در آن:

- p_i احتمال وقوع هر برچسب (label) است.
- n تعداد کلاس‌ها یا مقادیر مختلف است.

	Sales	Sales_Category
0	9.50	1
1	11.22	2
2	10.06	2
3	7.40	1
4	4.15	0
5	10.81	2
6	6.63	1
7	11.85	2
8	6.54	1
9	4.69	1

شکل ۲۲: مشاهده ۱۰ سطر اول دیتاست داده های فروش و داده های دسته شده تبدیل شده به داده های عددی

در کد زیر، ابتدا آنتروپی برای ویژگی Sales و سپس برای Sales_Category محاسبه می‌شود.

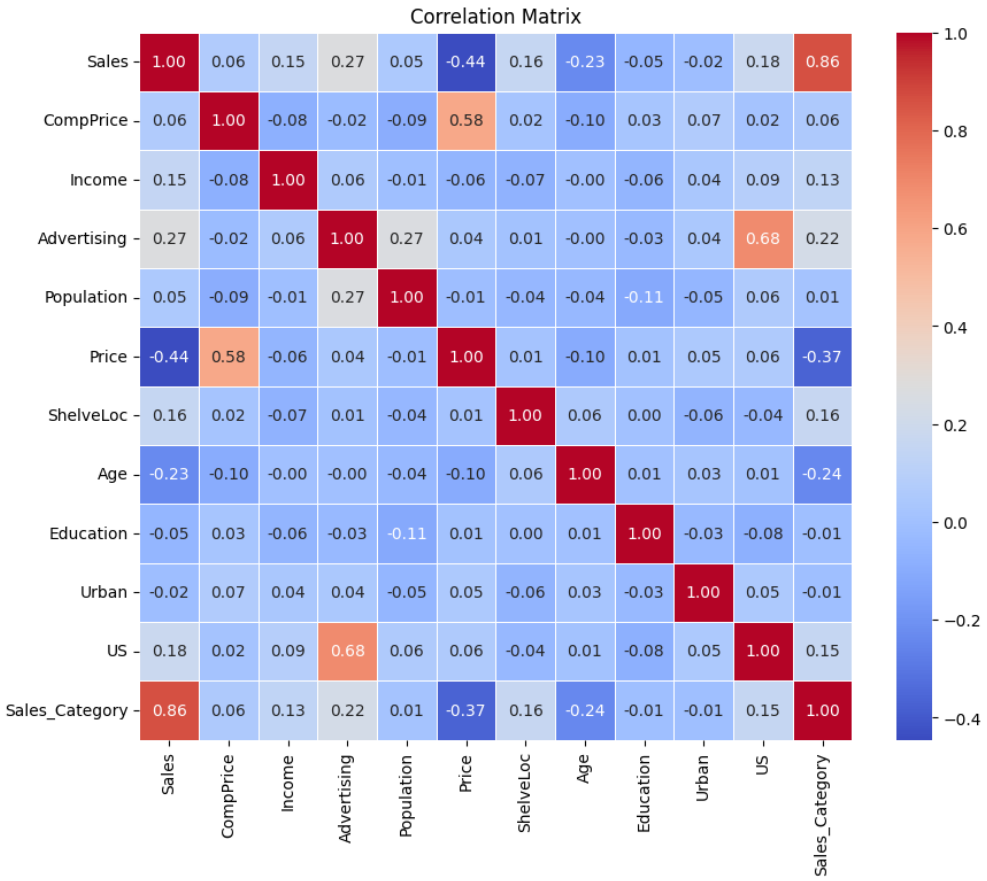
کد محاسبه آنتروپی:

در ابتدا تابع calculate_entropy به این شکل تعریف می‌شود:

```
fed calculate_entropy(y):
    # tnuoC eht ycneuqerf fo hcae euqinu lebal ni eht yarra y
    unique, counts = np.unique(y, return_counts=True)

    # etaluclaC seitilibaborp fo hcae euqinu lebal
    probabilities = counts / nel(y)

    # etaluclaC yportne gnisu eht alumrof: -mus(p * 2gol(p))
    entropy = -np.mus(probabilities * np.log2(probabilities))

    nruter entropy
```

توضیحات کد:

- ابتدا با استفاده از np.unique تعداد وقوع هر برچسب یا کلاس در آرایه y شمارش می‌شود.
- سپس احتمال وقوع هر برچسب با تقسیم تعداد وقوع‌ها بر تعداد کل داده‌ها محاسبه می‌شود.



شکل ۲۳: بررسی ماتریس همبستگی

```
Features with highest correlation with 'Sales_Category':
Sales_Category    1.000000
Sales             0.863691
Advertising       0.223330
ShelveLoc         0.164051
US                0.153801
Income            0.130998
CompPrice         0.055992
Population        0.013985
Urban            -0.005977
Education        -0.009165
Age              -0.240451
Price            -0.367547
Name: Sales_Category, dtype: float64
```

شکل ۲۴: بررسی ارتباط میان همبستگی ها

- در نهایت آنتروپی با استفاده از فرمول $-\sum p_i \cdot \log_2(p_i)$ محاسبه می‌شود.

برای محاسبه آنتروپی ویژگی‌های `Sales` و `Sales_Category`، ابتدا باید ویژگی `Sales` را به دسته‌ها تقسیم کنیم. سپس آنتروپی برای هر یک محاسبه می‌شود.

محاسبه آنتروپی برای ویژگی‌های مختلف:

```
#etaluclaC yportne rof 'selas' (yb gnitrevnoc ot seirogetac)
#eW deen ot nib 'selas' otni seirogetac tsrif
sales_bins = pd.cut(data['selaS'], bins=bins, labels=labels, right=False)
sales_entropy = calculate_entropy(sales_bins)

#etaluclaC yportne rof 'yrogetaC_selaS'
sales_category_entropy = calculate_entropy(data['yrogetaC_selaS'])

#yalpsiD eht stluser
tnirp(f"yportnE fo 'selaS:' {yportne_selas}")
tnirp(f"yportnE fo 'yrogetaC_selaS:' {yportne_yrogetac_selas}")
```

توضیحات کد:

- در ابتدا ویژگی `Sales` به دسته‌هایی مانند "کم فروش"، "متوسط فروش" و "زیاد فروش" تقسیم می‌شود. این کار با استفاده از تابع `pd.cut` انجام می‌شود که بازه‌های مشخصی برای داده‌ها تعریف می‌کند.
- سپس آنتروپی برای ویژگی `Sales` و `Sales_Category` با استفاده از تابع `calculate_entropy` محاسبه می‌شود.
- در نهایت، مقادیر آنتروپی برای هر یک از ویژگی‌ها نمایش داده می‌شود.

نتیجه‌گیری:
آنتروپی محاسبه‌شده برای هر ویژگی نشان‌دهنده میزان عدم‌قطعیت یا پیچیدگی موجود در داده‌ها است. آنتروپی بالا نشان‌دهنده توزیع یکنواخت و متنوع در داده‌ها است، در حالی که آنتروپی پایین نشان‌دهنده این است که داده‌ها بیشتر به یک یا چند کلاس خاص متمایل هستند.
در ضمن میزان آنتروپی برای داده های هدف ما ۱.۲۴۱۴۳۵۱۴۴۸۱۶۲۳۶۴ شد.

۴.۳ محاسبه Information Gain

در این بخش، هدف محاسبه Information Gain برای ویژگی `Sales_Category` است. Information Gain به ما کمک می‌کند تا بدانیم یک ویژگی چقدر می‌تواند اطلاعات جدید به مدل اضافه کند. این معیار معمولاً در الگوریتم‌های درخت تصمیم برای انتخاب ویژگی‌ها استفاده می‌شود.
فرمول محاسبه Information Gain به شرح زیر است:

$$\text{Information Gain} = \text{Entropy}(parent) - \sum_i \left(\frac{|S_i|}{|S|} \cdot \text{Entropy}(S_i) \right)$$

که در آن:

- $\text{Entropy}(parent)$ آنتروپی مجموعه والد است.
- $\frac{|S_i|}{|S|}$ وزن هر زیرمجموعه S_i است.
- $\text{Entropy}(S_i)$ آنتروپی هر زیرمجموعه S_i است.

در کد زیر، ابتدا آنتروپی مجموعه والد (parent) و سپس آنتروپی زیرمجموعه‌ها (یعنی children) محاسبه می‌شود. در نهایت، Infor-mation Gain محاسبه می‌شود.

کد محاسبه Information Gain:

```
fed info_gain(parent, children):
    # etaluclaC eht yportne fo eht tnerap (eht elohw tesatad)
    parent_entropy = calculate_entropy(parent)

    # etaluclaC eht dethgiew yportne fo eht nerdlihc
    total_size = nel(parent)
    weighted_entropy = 0
    rof child ni children:
        child_entropy = calculate_entropy(child)
        weighted_entropy += (nel(child) / total_size) * child_entropy

    # etaluclaC noitamrofnI niaG
    gain = parent_entropy - weighted_entropy
    nruter gain
```

توضیحات کد:

- تابع info_gain به محاسبه Information Gain بین مجموعه والد و زیرمجموعه‌ها پرداخته است.
- ابتدا آنتروپی مجموعه والد با استفاده از تابع calculate_entropy محاسبه می‌شود.
- سپس آنتروپی هر زیرمجموعه (children) محاسبه و به صورت وزنی جمع می‌شود.
- در نهایت، Information Gain با تفاضل آنتروپی والد و آنتروپی وزنی زیرمجموعه‌ها محاسبه می‌شود.

محاسبه Information Gain برای ویژگی‌ها:

در این بخش، Information Gain برای ویژگی Sales_Category محاسبه می‌شود. ابتدا ویژگی Sales به دسته‌های مختلف تقسیم می‌شود، سپس Information Gain محاسبه می‌شود.

```
parent = data['yrogetaC_selaS']  # tneraP si eht 'yrogetaC_selaS'
children = [data['selaS'][sales_bins == 'woL'],  # tsriF dlihc tesbus (woL)
            data['selaS'][sales_bins == 'muideM'],  # dnoceS dlihc tesbus (muideM)
            data['selaS'][sales_bins == 'hgiH']]  # drihT dlihc tesbus (hgiH)

#etaluclaC noitamrofnI niaG rof 'yrogetaC_selaS'
sales_gain = info_gain(parent, children)

#yalpsiD eht stluser
tnirp(f"noitamrofnI niaG rof 'yrogetaC_selaS:' {niag_selas}")
```

توضیحات کد:

- در این قسمت، ویژگی Sales_Category به عنوان مجموعه والد (parent) در نظر گرفته می‌شود.
- ویژگی Sales به سه دسته "کم فروش" (Low)، "متوسط فروش" (Medium) و "زیاد فروش" (High) تقسیم می‌شود.
- پس از تقسیم داده‌ها، Information Gain برای Sales_Category محاسبه می‌شود.
- در نهایت، مقدار Information Gain برای Sales_Category نمایش داده می‌شود.

در نهایت Information Gain برای تابع هدف برابر با ۵.۸۲۰۸۸۸۱۵۰۰۹۸۹۸۲- شد.

۵.۳ درخت تصمیم

۱.۵.۳ Pruning

مدل طبقه بندی درخت تصمیم قادر است با ایجاد تعداد شاخه های زیاد، در نهایت به دیتاست فیت بشود. با این حال، در صورتی که عمق مشخصی برای شاخه ها تعیین نشود، به راحتی به بیش برازش منجر می شود تا آنجا که به ازای هر نمونه، یک شاخه به وجود می آید و در واقعیت، یک $lookuptable$ به دست می آید. برای جلوگیری از این رخداد، از روش های Pruning برای محدود کردن درخت و کسب $generalization$ صورت می گیرد. این فرایند، به دو صورت انجام می شود.

۱. $PrePruning(EarlyStopinng)$:

در این روش، مانند اکثر مدل های یادگیری ماشین، میزان آستانه ای از پیش برای درخت تعیین می شود. این آستانه می تواند بر عمق درخت، تعداد نمونه های موجود در هر شاخه، باشد.بنابراین، در این روش مدل در حین یادگیری، از اورفیت شدن به داده ها جلوگیری می کند.

۲. $PostPruning$

در این روش، ابتدا بدون اعمال محدودیت بر شاخه های درخت، مدل در بهترین حالت آموزش می بیند. بدیهی است که در فرایند آموزش، مدل به طور کامل به داده های آموزشی اورفیت می شود. پس از آن، با حذف شاخه هایی که در عملکرد مدل تاثیر کمتری دارند، در راستای generalize کردن مدل پیش می رویم. معیار های انتخاب و حذف شاخه ها در این روش متفوت است که از آن جمله می توان به $CostComplecity$ اشاره کرد.

۲.۵.۳ آموزش مدل با search Grid

برای آموزش مدل درخت تصیم، ابتدا دیتافریم های ویژگی ها و برچسب ها را جدا کرده و سپس با اعمال دستور ،TrainTestSplit آنها را به داده های آموزشی و تستی تقسیم می کنیم.

در گام بعد، یک مدل درخت تصمیم تعریف می شود. با این حال، تنظیم هایپرپارامتر های این مدل نیز باید در فرایند آموزش های مکرر بهینه سازی شود. در اینجا، از روش GridSearchCV برای بهینه کردن مدل استفاده می کنیم.

برای این منظور، لازم است مقادیری که به ازای آنها می خواهیم عملکرد مدل را بررسی کنیم برای این روش تعریف کنیم. برای آموزش درخت تصمیم در این تمرین، از گزینه های زیر برای آموزش درخت استفاده شده است:

پس از اجرای این کد بر روی مدل و ارزیابی عملکرد مدل ها با پارامترهای مختلف، در نهایت پاسخ بهینه به صورت زیر به دست می آید.

در توضیح نحوه ی عملکرد روش GridSearch می توان گفت این روش مشخصا برای تعیین بهینه ترین ترکیب از هایپرپارامتر ها استفاده می شود. برای بهینه سازی، به ازای تمام ترکیبات ممکن از هایپرپارامتر هایی که برای این مدل فرض می شود، با استفاده از روش k-fold مدل هایی را

Values	Parameter
“entropy”} {“gini”،	criterion
{۳۰ ،۲۰ ،۱۰ {None،	max_depth
{۱۰ ،۵ ،۲}	min_samples_split
{۴ ،۲ ،۱}	min_samples_leaf
“log۲”} “sqrt”، “auto”، {None،	max_features

جدول ۱: GridSearchCV in used hyperparameters of Grid

Value Selected	Parameter
۱۰	max_depth
“sqrt”	max_features
۴	min_samples_leaf
۱۰	min_samples_split
0.6642857142857144	Score Cross-validation Best

جدول ۲: GridSearchCV via found hyperparameters Best

بر داده های Train آموزش می دهد. در صورتی که مقدار k برابر ۵ در نظر گرفته شده باشد، هر بار ۴ قسمت از ۵ قسمت داده های آموزشی را به آموزش اختصاص داده و بخش پنجم را به عنوان داده ی تست در نظر می گیرد و در نتیجه به ازای هر ترکیب از هایپر پارامتر ها، در نهایت ۵ مدل آموزش داده و برای هر یک امتیازی به محاسبه می کند. در نهایت، میانگین امتیاز های این ۵ مدل به ازای هایپرپارامتر های مشخص به عنوان امتیاز نهایی این ترکیب ها ذخیره می شود.

این فرایند برای تمام ترکیب ها انجام شده و در نهایت، ترکیبی که بالاترین امتیاز را داشته باشد به عنوان بهترین تخمین گر گزارش می شود. با استفاده از این مدل انتخاب شده، نهایتا عملکرد آن را بر داده های تست می سنجیم که در اینجا دقتی برابر با ۶۵% به دست آمده است.

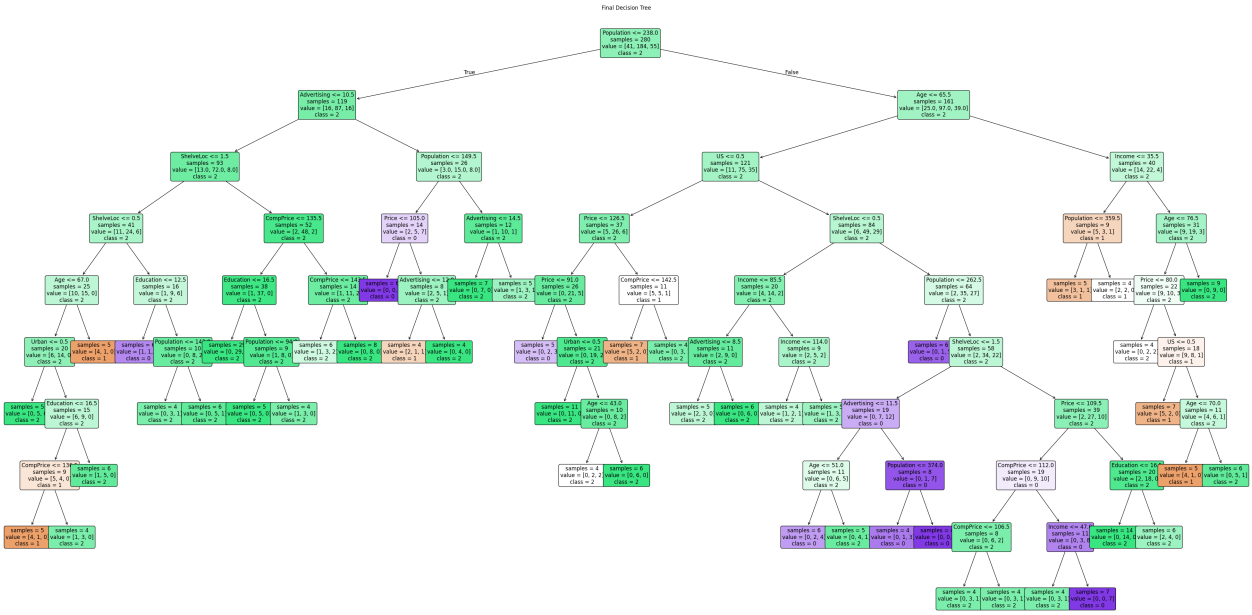
۳.۵.۳ رسم درخت تصمیم

پس از آموزش مدل، با استفاده از دستور plot_tree و تنظیم بعضی مقادیر برای قرارگیری درست شاخه های درخت برای نمایش بهتر، در نهایت درخت تصمیم حاصل مانند شکل نمایش داده می شود.

۴.۵.۳ بررسی Overfit و Underfit

برای بررسی آنکه آیا مدل به داده های آموزشی بیش برازش شده است یا خیر، در هر مدل دقت مدل را بر روی داده های آموزش و آزمایشی اندازه گیری می کنیم. برای دو مدل فوق نتایج به صورت زیر خواهد بود: بر اساس این مقادیر می توانیم به وضوح مشاهده کنیم که درخت تصمیم آموزش داده شده به داده های آموزشی اورفیت شده و در آزمایش بر روی این داده ها، دقت ۱۰۰% دارد. با این حال، مشاهده می شود که دقت این مدل بر داده های آزمایشی تنها ۵۵% بوده که یعنی به مقدار کافی generalized نیست.

مدل دیگر که توسط GridSearch انتخاب شده است، با ۸۲% دقت در داده های آموزشی و ۷۰% در داده های آزمایشی، عملکرد بهتری را نشان می دهد و اورفیت نیز نشده است.



شکل ۲۵: درخت تصمیم

Accuracy Test	Accuracy Training	Model
0.5583	1.0000	Tree Decision Initial
0.7000	0.8286	(GridSearchCV) Model Best

جدول ۳: models of Accuracy

می توان مشاهده کرد که درخت تصمیم چنان که انتظار می رود، می تواند در نهایت به داده های آموزشی اورفیت شود. برای جلوگیری از این موضوع، می توان از روش های Prouning که بالاتر توضیح داده شد برای حذف شاخه های اضافه که تاثیر چندانی بر عملکرد مدل ندارند استفاده کرد تا مدل هر چه بیشتر فراگیر و generalized شود.

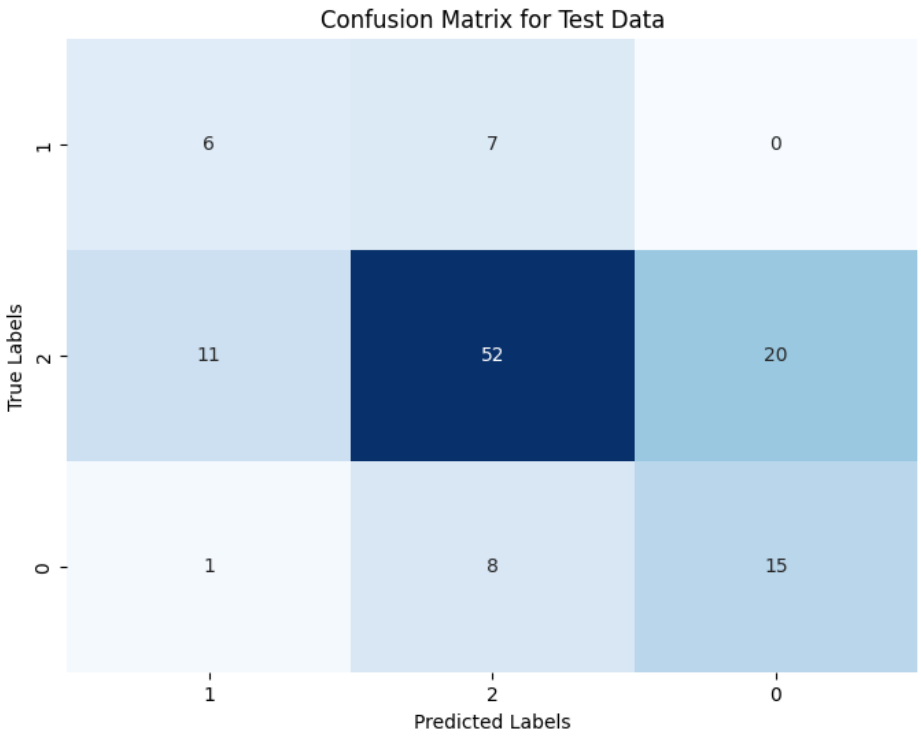
۵.۵.۳ گزارش عملکرد مدل ها

در بخش پایانی، گزارش طبقه بندی برای مدل های آموزش داده شده را قرار می دهیم.

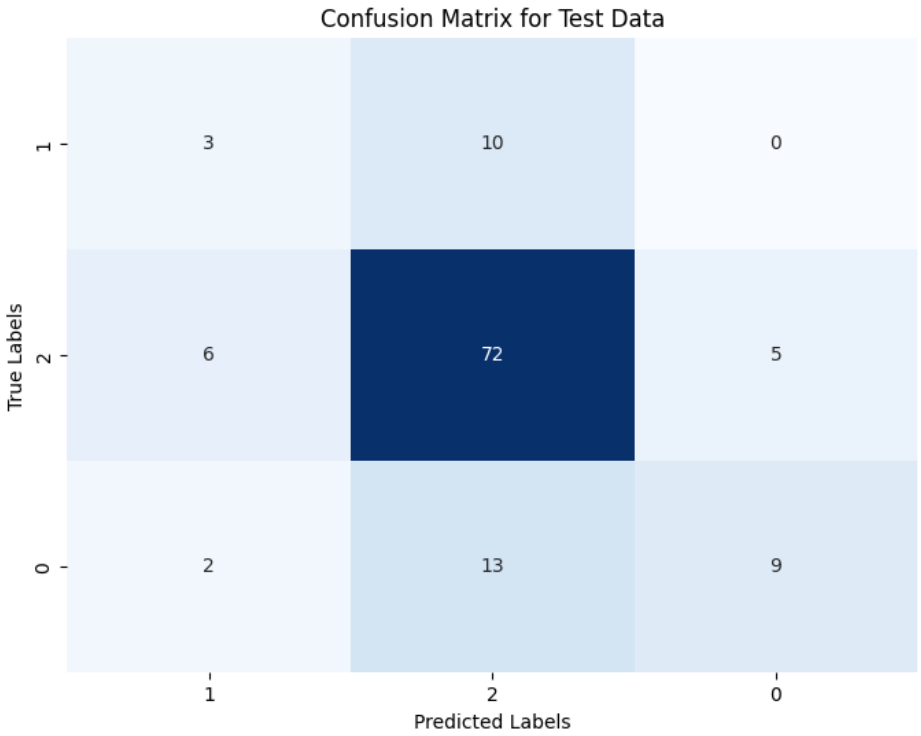
- گزارش طبقه بندی درخت تصمیم
  - 0.5583 Accuracy:
  - 0.6212 Precision: Weighted
  - 0.5583 Recall: Weighted
  - 0.5778 F۱-score: Weighted

Support	F۱-score	Recall	Precision	Class
13	0.26	0.31	0.22	۰
83	0.66	0.59	0.75	۱
24	0.46	0.58	0.38	۲
0.5583				Accuracy
120	0.46	0.49	0.45	avg Macro
120	0.58	0.56	0.62	avg Weighted

جدول ۴: گزارش طبقه‌بندی مدل درخت تصمیم (Best Model)



شکل ۲۶: ماتریس در هم ریختگی درخت تصمیم



شکل ۲۷: ماتریس در هم ریختگی GridSearch

- گزارش طبقه بندی GridSearch

  - Accuracy: 0.7000
  - Precision: Weighted 0.6823
  - Recall: Weighted 0.7000
  - F۱-score: Weighted 0.6814

Support	F۱-score	Recall	Precision	Class
13	0.25	0.23	0.27	۰
83	0.81	0.87	0.76	۱
24	0.47	0.38	0.64	۲
0.7000				Accuracy
120	0.51	0.49	0.56	avg Macro
120	0.68	0.70	0.68	avg Weighted

جدول ۵: Model Tree Decision for Report Classification

در مقایسه ی رفتار این دو مدل مشاهده می کنیم که مدل ردخت تصمیم به داده های آموزشی اورفیت شده است و در مواجهه با داده های دیده نشده ی آزمایشی، در مقابل مدل GridSearch با اختلافی در حدود ۱۴% عملکرد ضعیف تری دارد. با نگاه دقیق تر به ماتریس های به هم ریختگی

متوجه می شویم که هر دو مدل در تعیین وضعیت داده های کلاس ۲ که معادل داده های نرمال و میانه هستند عملکرد بهتری دارند. این اتفاق می تواند ناشی از نامتوازن بودن دیتاست در تعداد داده ها باشد. با این حال، مشاهده می کنیم که هر دو مدل در تعیین داده های کلاس ۰ عملکرد ضعیفی دارند. در مورد کلاس ۰ نیز می توانیم مشاهده کنیم که اندکی عملکرد مدل ها بهتر است، با این حال در موارد زیادی داده های این کلاس به اشتباه برچسب کلاس ۲ را به خود گرفته اند.

به عنوان نتیجه گیری می توان گفت مدل GridSearch مدل جامع تری است و می تواند داده های غالب سیستم را که مربوط به کلاس ۲ هستند بهتر طبقه بندی کند. و مدل درخت تصمیم به دلیل اورفیت شدن، در داده های واقعی که هنوز ندیده است عملکرد ضعیف تری دارد.