دانشکده مهندسی برق

دوره کارشناسی ارشد مهندسی برق-کنترل

مینی‌پروژه دوم درس یادگیری ماشین

توسط:

محمدرضا امانی – احمدرضا طاهری

استاد راهنما:

دکتر مهدی علیاری شوره‌دلی

بهار ۱۴۰۴

Question 1 & 2 code

Question 3 code


بسم الله الرحمن الرحیم

فهرست مطالب

عنوان **صفحه**

فهرست جدول‌ها ..... ج

فهرست شکل‌ها ..... د

-فصل 1 پرسش یک ۱

الف) ۱

ب) ۱

ج) ۲

د) ۳

ه) ۴

و) ۵

فصل ۲- پرسش دو ۶

آ - ب) ۶

ج) ۷

د) ۷

د – PCA) ۹

فصل ۳- پرسش سوم ۱۱

بخش ۱.۳ ..... ۱۱

بخش ۲.۳ ..... ۱۱

ماتریس همبستگی : ۱۵

بخش ۳.۳ ..... ۱۶

بخش ۴.۳ ..... ۱۶

بخش ۵.۳ ..... ۱۷

مفهوم prune کردن ۱۷

تابع GridSearchCv ۱۸

آموزش مدل درخت تصمیم ۱۸

وقوع overfit ۲۰

گزارش طبقه بندی و ماتریس درهم ریختگی برای دادگان Test ۲۰

فهرست جدول‌ها

عنوان **صفحه**

جدول ۱-۱: جدول تابع هزینه ریسک ۵

فهرست شکل‌ها

عنوان **صفحه**

شکل ۱-۱: دادگان اسپم پیامک ۱
شکل ۲-۱: تبدیل دادگان متن به ویژگی‌های عددی ۲
شکل ۳-۱: تابع Multinomial scratch ۲
شکل ۴-۱: نتایج شبیه‌سازی Multinomial Scratch ۳
شکل ۵-۱: نتایج شبیه‌سازی Multinomial Sklearn ۳
شکل ۶-۱: نتایج شبیه‌سازی تابع Gaussian ۴
شکل ۱-۲: جداسازی دادگان برای آموزش و آزمایش و نرمالسازی آن‌ها ۶
شکل ۲-۲: آموزش مدل KNN با مقادیر مختلف K ۷
شکل ۳-۲: ترسیم مقادیر صحت بر حسب مقدار K در الگوریتم KNN ۷
شکل ۴-۲: ترسیم confusion matrix برای بهترین K در الگوریتم KNN ساده ۸
شکل ۵-۲: پیاده‌سازی PCA ۹
شکل ۶-۲: نتایج شبیه‌سازی PCA-KNN ۹
شکل ۷-۲: ترسیم confusion matrix برای بهترین ترکیب در PCA-KNN ۱۰
شکل ۱-۳: دیتافریم دادگان خام ۱۱
شکل ۲-۳: دیتافریم دادگان پیش‌پردازش شده ۱۲
شکل ۳-۳: دیتافریم دادگان با متغیر هدف کلاسبندی شده ۱۳
شکل ۴-۳: مقدار همبستگی ویژگی ها با متغیر هدف sales به ترتیب نزولی ۱۴
شکل ۵-۳: ماتریس همبستگی ویژگی ها و متغیر هدف ۱۵
شکل ۶-۳: کد تابع محاسبه انتروپی ۱۶
شکل ۷-۳: کد تابع محاسبه Information gain ۱۶
شکل ۸-۳: بهترین هایپرپارامتر های محاسبه شده توسط GridsearchCV ۱۹
شکل ۹-۳: درخت تصمیم آموزش داده شده ۱۹
شکل ۱۰-۳: گزارش طبقه بندی دادگان تست ۲۱
شکل ۱۱-۳: ماتریس درهم ریختگی دادگان تست ۲۱

فصل ۱- پرسش یک

الف)

بیز ساده

طبقه‌بند بیز ساده از قضیه بیز استفاده می‌کند اما فرض می‌کند ویژگی‌ها نسبت به یکدیگر مستقل‌اند، مشروط بر کلاس (که به آن فرض "استقلال شرطی" گفته می‌شود). این فرض در دنیای واقعی اغلب برقرار نیست، اما باعث ساده‌تر شدن محاسبات و اجرای سریع می‌شود.

$$P(C_k \mid x_1, x_2, \dots, x_n) \propto P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

بیز بهینه

طبقه‌بند بیز بهینه (که به آن Bayes classifier نیز گفته می‌شود) بهترین عملکرد ممکن را در تئوری دارد و هیچ فرضی درباره‌ی استقلال ویژگی‌ها نمی‌زند. این طبقه‌بند با داشتن توزیع‌های احتمال واقعی برای داده‌ها و کلاس‌ها، کلاس با بیشترین احتمال پسین واقعی را انتخاب می‌کند.

$$P(C_k \mid x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

این طبقه‌بند بهترین تصمیم‌گیر ممکن است اگر توزیع احتمال $P(x|C_k)$ به طور کامل در اختیار باشد. اما به دلیل در دسترس نبودن این توزیع احتمال، پیاده‌سازی این روش محدودیت‌هایی دارد.

ب)

از میان سه مدل Multinomial، Bernoulli و Gaussian، مناسب‌ترین مدل برای کار با داده‌های متن Sms Spam، مدل Multinomial است. دادگان به شرح زیر هستند:

	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
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

شکل ۱-۱: دادگان اسپم پیامک

در تحلیل این نوع از داده، ما به دنبال تعداد تکرار کلمات در جملات مختلف هستیم. این ویژگی‌ها (تعداد تکرار هر کلمه) با استفاده از روش‌هایی مانند Bag of words یا TF-IDF استخراج می‌شوند. حال مناسب‌ترین روش، روش Multinomial است، زیرا شمارش کلمات را به عنوان ویژگی در نظر می‌گیرد. به طور مثال بیان می‌کند که احتمال دیدن تعدادی از هر کلمه در یک کلاس خاص چقدر است.

```
vectorizer = CountVectorizer()
#vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['text'].values).toarray()
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=rs)
```

شکل ۲-۱: تبدیل دادگان متن به ویژگی‌های عددی

این در حالی است که مدل Bernoulli، فرض می‌کند هر ویژگی مقدار باینری ۰ یا ۱ دارد. مدل Bernoulli برای حالتی مناسب است که به دنبال وجود یا عدم وجود کلمه‌ای خاص در دادگان هستیم، نه تعداد یا وزن ویژگی‌ها. همچنین مدل Guassian، فرض می‌کند که ویژگی‌ها از توزیع نرمامل پیروی می‌کنند. این مدل برای داده‌های پیوسته واقعی مانند دما مناسب است.

بنابراین بهترین روش Multinomial است.

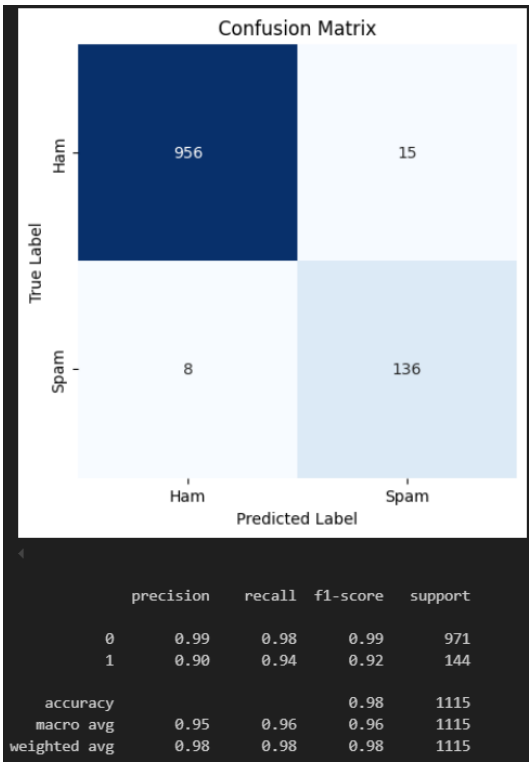
ج)

مدل Mutinomial را با استفاده از کد زیر پیاده‌سازی می‌کنیم:

```
class MultinomialNB_Scratch:
    def fit(self, X, y, alpha=1.0):
        self.alpha = alpha
        self.classes = np.unique(y)
        self.class_count = np.bincount(y)
        self.class_prior = self.class_count / len(y)  # P(class)

        self.feature_count = np.zeros((len(self.classes), X.shape[1]))
        for idx, c in enumerate(self.classes):
            self.feature_count[idx, :] = X[y == c].sum(axis=0)


        self.feature_log_prob = np.log((self.feature_count + alpha) /
                                       (self.feature_count.sum(axis=1)[:, np.newaxis] + alpha * X.shape[1]))

    def predict(self, X):
        log_probs = np.log(self.class_prior) + X @ self.feature_log_prob.T
        return np.argmax(log_probs, axis=1)
```

شکل ۳-۱: تابع Multinomial scratch

ارزیابی انجام شده بر حسب confusion matrix و معیارهای مختلف سنجش دقت بر حسب زیر است:

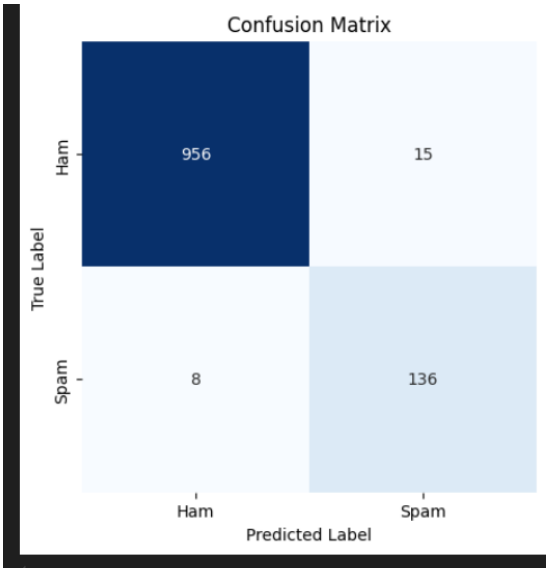


	precision	recall	f1-score	support
0	0.99	0.98	0.99	971
1	0.90	0.94	0.92	144
accuracy			0.98	1115
macro avg	0.95	0.96	0.96	1115
weighted avg	0.98	0.98	0.98	1115

شکل ۴-۱: نتایج شبیه‌سازی Multinomial Scratch

د)

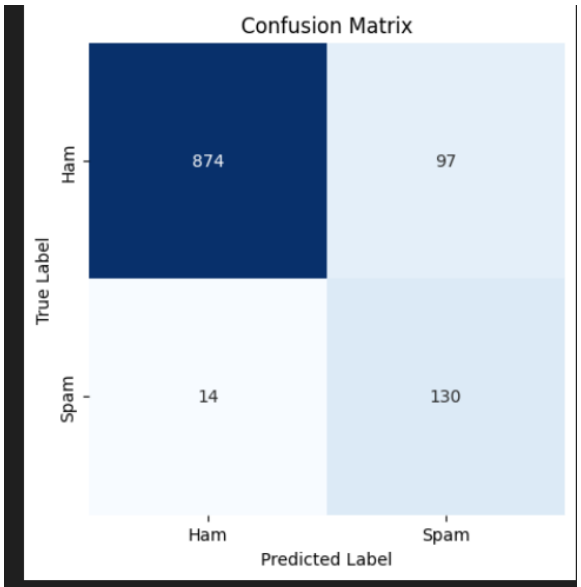
نتایج شبیه‌سازی تابع آماده Multinomial از کتابخانه sklearn به شرح زیر است:



	precision	recall	f1-score	support
0	0.99	0.98	0.99	971
1	0.90	0.94	0.92	144
accuracy			0.98	1115
macro avg	0.95	0.96	0.96	1115
weighted avg	0.98	0.98	0.98	1115

شکل ۵-۱: نتایج شبیه‌سازی Multinomial Sklearn

مشاهده می‌شود که نتایج شبیه‌سازی با استفاده از تابع scratch و با استفاده از تابع sklearn کاملا مشابه یکدیگر شده‌اند. اکنون به شبیه‌سازی تابع آماده Gaussian می‌پردازیم:



	precision	recall	f1-score	support
0	0.98	0.90	0.94	971
1	0.57	0.90	0.70	144
accuracy			0.90	1115
macro avg	0.78	0.90	0.82	1115
weighted avg	0.93	0.90	0.91	1115

شکل ۶-۱: نتایج شبیه‌سازی تابع Gaussian

به طور کلی هر دو مدل، صحت بالایی داشته و تعداد miss classification نسبت به False classification به طور چشم‌گیری کمتر است. این ویژگی بدان معناست که هر دو مدل توانایی بالایی در تشخیص spam بودن پیامک‌ها دارند. در نهایت همانگونه که انتظار می‌رفت، مدل Multinomial عملکرد بسیار مناسب‌تری، به دلیل ویژگی‌های بیان شده برا این مدل در بخش ب، از خود نشان داده است. مدل Gaussian دقت پایینی برای تشخیص spam دارد و تعداد زیادی از پیامک‌های معتبر را اسپم تشخیص می‌دهد.

۵)

فرض کنید بردار ویژگی یا وقایع را x در نظر بگیریم. مجموعه‌ای از کلاس‌های مورد نظر شامل $C = \{c_1, c_2, \dots, c_n\}$. می‌خواهیم کلاس $\hat{c}(x)$ را پیش‌بینی کنیم. هدف کمینه کردن احتمال خطا است. احتمال خطا را به صورت زیر تعریف می‌کنیم:

$$P(error|x) = 1 - P(\hat{c}(x)|x)$$

که در آن $P(\hat{c}(x)|x)$، احتمال پسین نام دارد.

$$P(correct|x) = P(c_i|x)$$

برای بیشینه کردن ترم بالا (کمینه کردن خطا)، باید کلاسی انتخاب شود که بالاترین $P(c_i|x)$ را داشته باشد:

$$P(error|x) = 1 - \max_{c_i} P(c_i|x)$$

حال برای محاسبه خطای کلی داریم:

$$Overall\ error = \int \left(1 - \max_{c_i} P(c_i|x)\right) p(x)dx$$

عبارت بالا در حالتی کمینه می‌شود، که همیشه کلاسی انتخاب شود، که دارای بیشینه احتمال پسین یا $P(c_i|x)$ باشد.

و)

تابع ریسک به صورت زیر تعریف می‌شود:

$$R(\hat{c} \mid x) = \sum_{c \in C} L(\hat{c}, c).P(c|x)$$

فرض می‌کنیم که هزینه اشتباه در تشخیص پیام معتبر به جای اسپم (در اصل اسپم بوده است) ۵ برابر اسپم تشخیص دادن پیام معتبر (در اصل معتبر بوده) باشد.

ماتریس هزینه به صورت زیر تعریف می‌شود:

جدول ۱-۱: جدول تابع هزینه ریسک

	True Spam	True Ham
Predict Spam	0	1
Predict Ham	5	0

$$R(spam|x) = 0 \times P(spam|x) + 1 \times P(ham|x) = P(ham|x)$$
$$R(ham|x) = 5 \times P(spam|x) + 0 \times P(ham|x) = 5P(spam|x)$$

در نهایت تابع تصمیم به صورت زیر خواهد بود:

$$R(spam|x) \overset{?}{\Leftrightarrow} R(ham|x)$$
$$P(ham|x) \overset{?}{\Leftrightarrow} 5P(spam|x)$$

فصل ۲- پرسش دو

آ - ب)

دقت داشته باشید که داده‌ها ابتدا باید به دو بخش train و test تقسیم شوند و سپس فرآیند نرمال‌سازی بر روی دادگان train، fit شده، و در نهایت تنها بر دادگان test، transform شود. نرمالایز کردن کل داده به صورت یکجا اشتباه است. بنابراین پرسش‌های آ و ب در کنار هم در این بخش پاسخ داده می‌شوند.

از میان دو روش MinMaxScaler و StandardScaler، روش MinMax برای این دادگان مناسب‌تر است زیرا در دادگان تصویر mnist، هر ستون یا ویژگی بیانگر یک پیسکل در تصویر است. مقدار این ویژگی برابر یکی از اعداد ۰ تا ۲۵۵ است که بیانگر ۲۵۶ طیف رنگی برای ساخت پیسکل مورد نظر است. بنابراین تمامی مقادیر درون این مجموعه دیتا، دارای مقدار مثبت بوده و در یک بازه به خصوصی قرار دارند. روش MinMax، تنها دادگان را در بازه بین ۰ تا ۱ قرار می‌دهد و همچنین علامت آن‌ها را تغییر نداده و منفی نمی‌کند. این در حالی است که در روش StandardScaler، دادگان به صورت تابع نرمال با میانگین صفر و واریانس ۱ در نظر گرفته می‌شوند. این روش می‌تواند باعث شود که در برخی از دادگان، تغییر علامت به وجود بیاید.

```
mnist_train, mnist_test = train_test_split(mnist, train_size=0.7, test_size=0.3, random_state=rs)
 0.0s


mnist_train_label = mnist_train['label'].values
mnist_test_label = mnist_test['label'].values

mnist_train = mnist_train.drop('label', axis=1)
mnist_test = mnist_test.drop('label', axis=1)
 0.0s


#scaler = StandardScaler()
scaler = MinMaxScaler()


scaler.fit(mnist_train)

mnist_train_scaled=scaler.transform(mnist_train)
mnist_train_scaled = pd.DataFrame(data=mnist_train_scaled, columns=mnist_train.columns)

mnist_test_scaled=scaler.transform(mnist_test)
mnist_test_scaled = pd.DataFrame(data=mnist_test_scaled, columns=mnist_test.columns)

mnist_test_scaled
```

شکل ۱-۲: جداسازی دادگان برای آموزش و آزمایش و نرمالسازی آن‌ها

ج)

با استفاده از مدل KNN از کتابخانه sklearn، دادگان را برای K در بازه ۱ تا ۲۵ آموزش می‌دهیم و بهترین K را در ارزیابی مدل بر روی دادگان آزمایش، استخراج می‌کنیم. بهترین مدل برای K=5 با صحت ۹۴ درصدی:

```
score_list=[]
best_score=0
best_k=0


for k in range(1,26):

    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(mnist_train_scaled,mnist_train_label)
    predict=knn.predict(mnist_test_scaled)
    acc = accuracy_score(mnist_test_label,predict)
    score_list.append(acc)
    if acc>best_score:
        best_score=acc
        best_k = k

print(f'Accuracy of KNN, best:{best_score} at K: {best_k}')
```

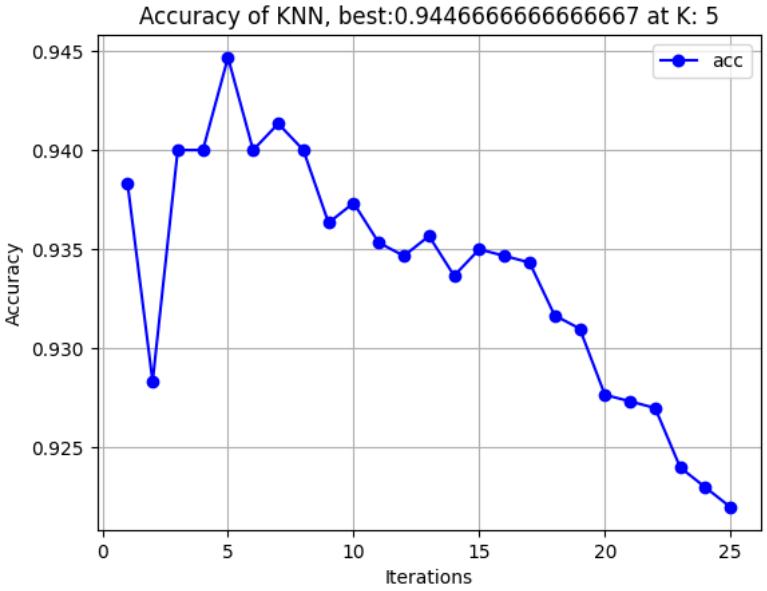
✓ 6.9s

```
Accuracy of KNN, best:0.9446666666666667 at K: 5
```

شکل ۲-۲: آموزش مدل KNN با مقادیر مختلف K

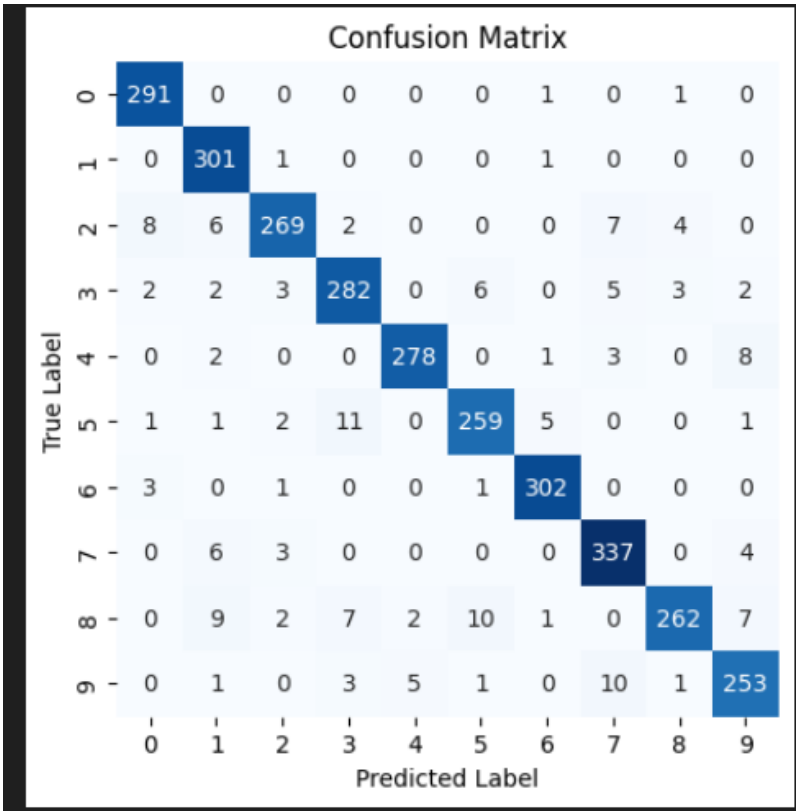
د)

اکنون با ذخیره تمامی مقادیر accuracy برای تمامی Kها، نمودار زیر را رسم می‌کنیم:



شکل ۳-۲: ترسیم مقادیر صحت بر حسب مقدار K در الگوریتم KNN

در نهایت confusion matrix را برای بهترین K در این مدل که برابر ۵ است، رسم می‌کنیم:



	precision	recall	f1-score	support
0	0.95	0.99	0.97	293
1	0.92	0.99	0.95	303
2	0.96	0.91	0.93	296
3	0.92	0.92	0.92	305
4	0.98	0.95	0.96	292
5	0.94	0.93	0.93	280
6	0.97	0.98	0.98	307
7	0.93	0.96	0.95	350
8	0.97	0.87	0.92	300
9	0.92	0.92	0.92	274
accuracy			0.94	3000
macro avg	0.95	0.94	0.94	3000
weighted avg	0.95	0.94	0.94	3000

شکل ۴-۲: ترسیم confusion matrix برای بهترین K در الگوریتم KNN ساده

د – PCA)

الگوریتم PCA، راستاهایی (مولفه) را بررسی می‌کند که در آن‌ها بیشترین واریانس وجود داشته باشد و همچنین این مولفه‌ها بر یکدیگر عمود باشند. بنابراین با استفاده از این الگوریتم می‌توان ابعاد داده را کاهش داد و تا حد زیادی اطلاعات مربوط به دادگان را حفظ کرد.

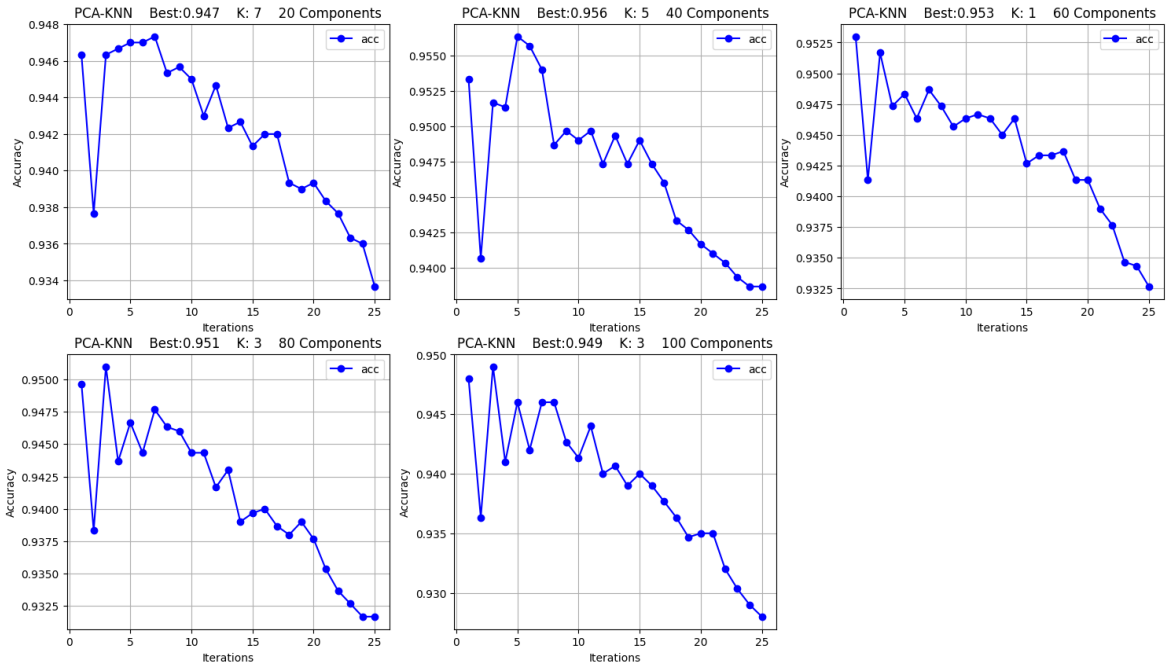
با استفاده از کد زیر، الگوریتم PCA را پیاده‌سازی می‌کنیم:

```
pca = PCA(n_components=j,random_state=rs)
pca.fit(mnist_train_scaled)

pca_train = pca.transform(mnist_train_scaled)
pca_test = pca.transform(mnist_test_scaled)
```

شکل ۵-۲: پیاده‌سازی PCA

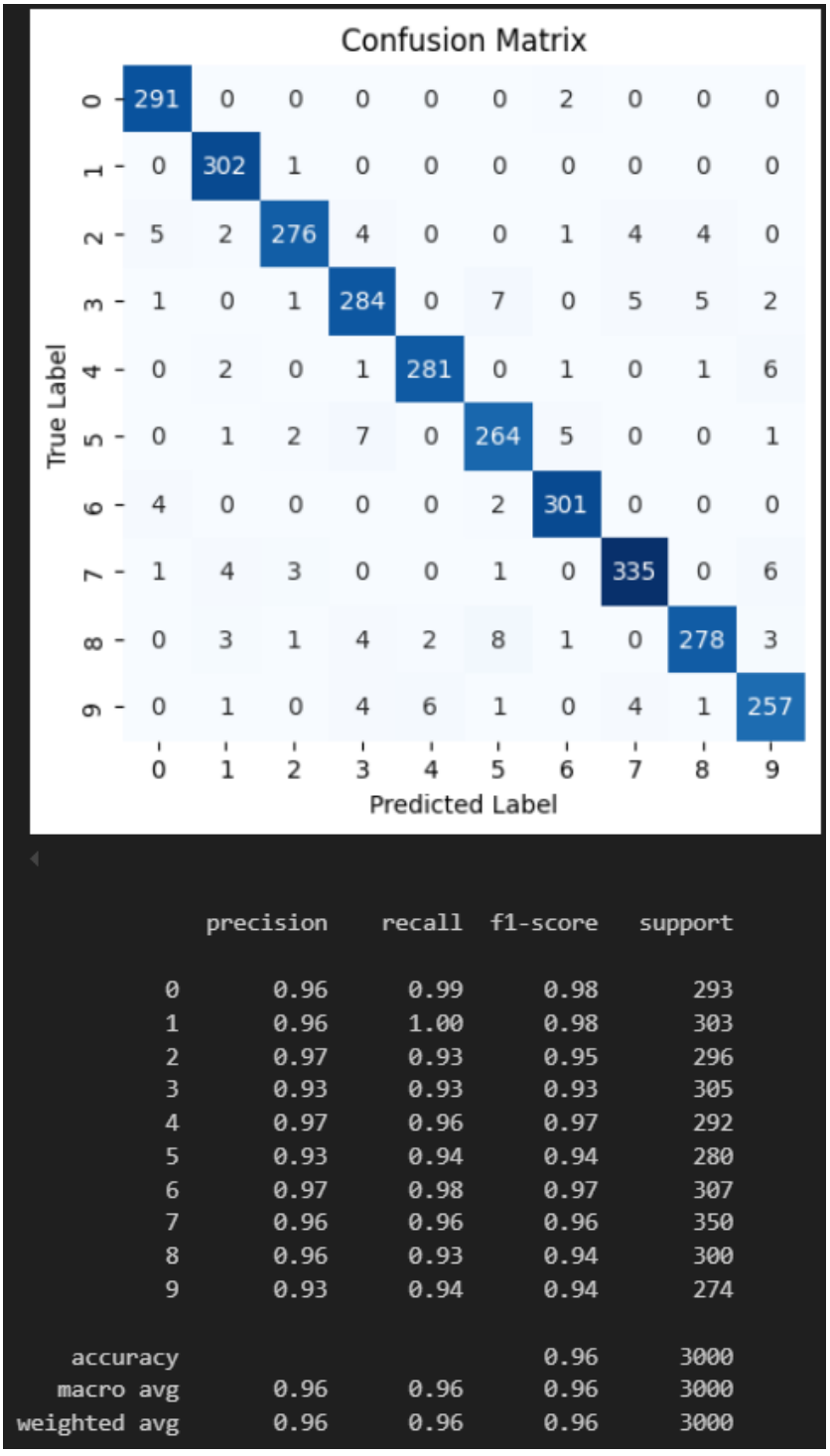
در نهایت، با ترکیب تعداد مولفه‌های PCA و مقدار پارامتر K در الگوریتم KNN، سعی می‌کنیم بهترین مدل را استخراج کنیم. نتایج شبیه‌سازی به شرح زیر خواهند بود:

```
The best combination is: PCA Components:40 with K:5 ---------> Accuracy:0.9563333333333334
```



شکل ۶-۲: نتایج شبیه‌سازی PCA-KNN

در نهایت confusion matrix بهترین ترکیب را رسم می‌کنیم:



	precision	recall	f1-score	support
0	0.96	0.99	0.98	293
1	0.96	1.00	0.98	303
2	0.97	0.93	0.95	296
3	0.93	0.93	0.93	305
4	0.97	0.96	0.97	292
5	0.93	0.94	0.94	280
6	0.97	0.98	0.97	307
7	0.96	0.96	0.96	350
8	0.96	0.93	0.94	300
9	0.93	0.94	0.94	274
accuracy			0.96	3000
macro avg	0.96	0.96	0.96	3000
weighted avg	0.96	0.96	0.96	3000

شکل ۷-۲: ترسیم confusion matrix برای بهترین ترکیب در PCA-KNN

در نهایت با استفاده از کاهش ابعاد PCA، هم الگوریتم سریع‌تر شده و هم صحت الگوریتم از ۹۴ درصد به ۹۶ درصد رسید.

فصل ۳- پرسش سوم

بخش ۱.۳

پس از تبدیل فایل دادگان دیتاست از فرمت csv. به pandas_dataframe ده سطر اول دیتاست به صورت زیر میباشد.

	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

شکل ۱-۳: دیتافریم دادگان خام

بخش ۲.۳

در اولین مرحله از پیش پردازش دیتاست یک کد مینویسیم که اگر سطری دارای missing value به ازای هر کدام از ستون ها یا همان ویژگی ها بود آن را مشخص کند و سپس آن سطر را از دیتاست حذف کند که نتیجه کد به صورت زیر می باشد و نشان می دهد که هیچ سطر ناقص یا missing value در دیتاست یافت نشده است.

```
No missing values found.
```

در مرحله بعدی از پیش پردازش به دنبال حذف سطر های تکراری میگردیم زیرا مشکلاتی از قبیل موارد زیر در آموزش مدل و تحلیل نتایج ایجاد خواهند کرد :

۱-ایجاد سوگیری در یادگیری مدل(Bias) :

وقتی یک رکورد (سطر) چندین بار تکرار شده باشد، مدل به اشتباه تصور می‌کند که این داده اهمیت بیشتری دارد. این باعث می‌شود وزن بیشتری به آن داده نسبت داده شود و روند یادگیری منحرف شود.

۲- افزایش احتمال بیش‌برازش(Overfitting) :

داده‌های تکراری باعث می‌شوند مدل الگوهای خاصی را بیش از حد حفظ کند و نتواند به‌خوبی به داده‌های جدید تعمیم پیدا کند. یعنی مدل بیش از حد به داده‌های آموزش وابسته می‌شود.

۳- خراب شدن آمارهای تحلیلی :

تکرار داده‌ها می‌تواند مقادیری مانند میانگین، فراوانی، یا ضریب همبستگی را به‌صورت نادرست نشان دهد. این تحلیل‌ها به جای اینکه تصویری واقعی از داده‌ها بدهند، گمراه‌کننده می‌شوند.

۴- افزایش مصرف منابع محاسباتی:

داشتن سطرهای اضافی باعث می‌شود زمان آموزش مدل بیشتر شود، حافظه بیشتری اشغال شود و عملکرد کلی سیستم پایین بیاید، در حالی‌که آن داده‌ها هیچ ارزش افزوده‌ای ندارند.

پس از چک کردن این مورد نیز نتیجه به صورت زیر نشان میدهد که هیچ سطر تکراری در دیتاست یافت نشد.

```
No duplicate rows found.
```

در آخرین مرحله پیش پردازش که تبدیل ویژگی های Categorical به Numerical می باشند به این صورت عمل میکنیم که با توجه به اینکه ستون های US , Urban دارای مقادیر Yes / No هستند آنها را با یک mapping ساده به 1 , 0 تعبیر میکنیم و برای ستون Shelveloc از تکنیک one-hot استفاده میکنیم و دیتافریم اصلی را به گونه ای تغییر میدهیم که به جای ستون Shelveloc سه ستون Shelveloc_bad , Shelveloc_medium , Shelveloc_good با مقادیر باینری 1 , 0 به صورت زیر خواهیم داشت.

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education	Urban	US	ShelveLoc_Bad	ShelveLoc_Good	ShelveLoc_Medium
0	9.50	138	73	11	276	120	42	17	1	1	1	0	0
1	11.22	111	48	16	260	83	65	10	1	1	0	1	0
2	10.06	113	35	10	269	80	59	12	1	1	0	0	1
3	7.40	117	100	4	466	97	55	14	1	1	0	0	1
4	4.15	141	64	3	340	128	38	13	1	0	1	0	0
5	10.81	124	113	13	501	72	78	16	0	1	1	0	0
6	6.63	115	105	0	45	108	71	15	1	0	0	0	1
7	11.85	136	81	15	425	120	67	10	1	1	0	1	0
8	6.54	132	110	0	108	124	76	10	0	0	0	0	1
9	4.69	132	113	0	131	124	76	17	0	1	0	0	1

شکل ۲-۳: دیتافریم دادگان پیش‌پردازش شده

حال به منظور کلاس بندی متغیر هدف دیتافریم جدید را به گونه ای تعریف می کنیم که یک ستون جدید با نام Sales_Class داشته باشد و اگر مقدار متغیر هدف (Sales) کوچکتر یا مساوی میانگین بود آن را Low_sales و اگر بالاتر از میانگین بود آن را High_sales کلاسبندی کند.

تصویر دیتافریم جدید به همراه اطلاعاتی در مورد sales صورت زیر خواهد بود :

```
Mean Sales value: 7.50

Sales class distribution:
Sales_Class
Low_sales     201
High_sales    199
Name: count, dtype: int64

First 10 rows of the classified DataFrame:
   Sales  CompPrice  Income  Advertising  Population  Price  Age  Education
0   9.50        138      73           11         276    120   42         17
1  11.22        111      48           16         260     83   65         10
2  10.06        113      35           10         269     80   59         12
3   7.40        117     100            4         466     97   55         14
4   4.15        141      64            3         340    128   38         13
5  10.81        124     113           13         501     72   78         16
6   6.63        115     105            0          45    108   71         15
7  11.85        136      81           15         425    120   67         10
8   6.54        132     110            0         108    124   76         10
9   4.69        132     113            0         131    124   76         17

   Urban  US  ShelveLoc_Bad  ShelveLoc_Good  ShelveLoc_Medium Sales_Class
0      1   1              1               0                 0  High_sales
1      1   1              0               1                 0  High_sales
2      1   1              0               0                 1  High_sales
3      1   1              0               0                 1   Low_sales
4      1   0              1               0                 0   Low_sales
5      0   1              1               0                 0  High_sales
6      1   0              0               0                 1   Low_sales
7      1   1              0               1                 0  High_sales
8      0   0              0               0                 1   Low_sales
9      0   1              0               0                 1   Low_sales
```

شکل ۳-۳: دیتافریم دادگان با متغیر هدف کلاسبندی شده

همانطور که مشاهده می شود میانگین ستون sales برابر ۷.۵ می باشد و پراکندگی هم کاملا یکسان بوده و مقدار آنتروپی دیتاست تقریبا ۱ می باشد (با توجه به اینکه شرط کوچکتر مساوی برای low_sales قرار داده شده است یک داده توزیع به صورت ۲۰۱ به ۱۹۹ می باشد.)

روابط هم بستگی :

با توجه به فرمول همبستگی بین دو متغیر داریم :

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \cdot \sigma_Y}$$

که کوواریانس بین آنها و همچنین انحراف معیار هر یک از متغیر ها از روابط زیر محاسبه می شود :

$$\mathrm{Cov}(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

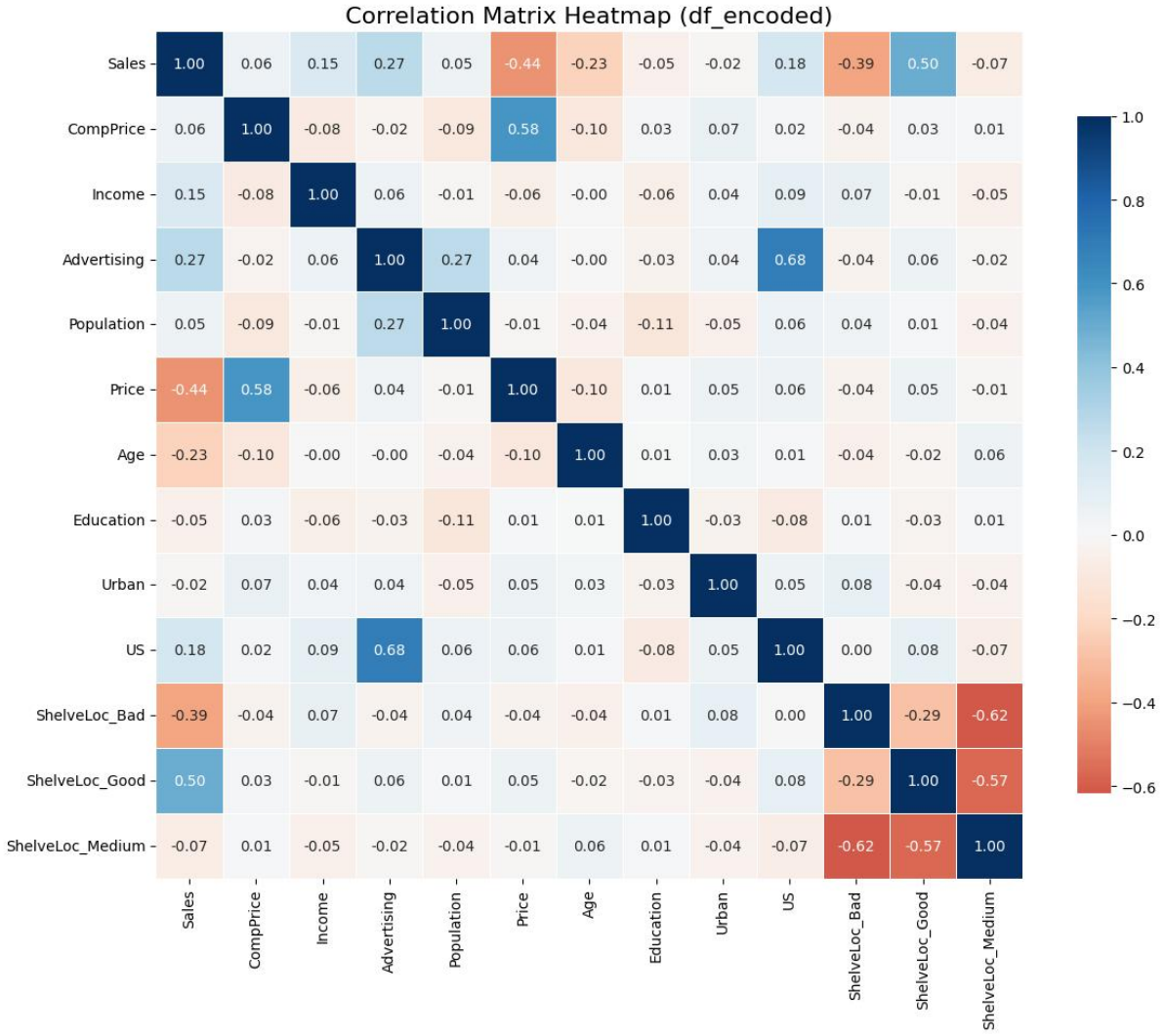
با توجه به فرمول همبستگی، قابل استنتاج است که Corr(X,Y) بدون واحد و نرمالایز می باشد که نشان دهنده ارتباط بین دو متغیر می باشد به نحوی که اگر تغییرات دو متغیر هم جهت باشد مقدار آن بین ۰ و ۱ (مثبت) و اگر در خلاف جهت هم باشد مقدار آن بین ۰ و ۱- (منفی) می باشد.

مقدار همبستگی متغیر هدف (sales) و ویژگی ها به ترتیب نزولی به صورت زیر می باشد :

```
Correlation of numeric features with 'Sales':
ShelveLoc_Good      0.500510
Advertising         0.269507
US                  0.177093
Income              0.151951
CompPrice           0.064079
Population          0.050471
Urban              -0.015419
Education          -0.051955
ShelveLoc_Medium   -0.073999
Age                -0.231815
ShelveLoc_Bad      -0.393167
Price              -0.444951
```

شکل ۴-۳: مقدار همبستگی ویژگی ها با متغیر هدف sales به ترتیب نزولی

ماتریس همبستگی :



شکل ۵-۳: ماتریس همبستگی ویژگی ها و متغیر هدف

همانطور که مشاهده میشود ماتریس همبستگی متقارن بوده و اگر سطر اول این ماتریس را بررسی کنیم فی الواقع همبستگی متغیر هدف (sales) با سایر ویژگی های encoded را میتوانیم مشاهده کنیم که همانطور که مشخص است بیشترین میزان همبستگی با فیچر shelveloc_good با مقدار ۰.۵ می باشد بدین معنی که قرار گرفتن کالا در موقعیت خوبی از قفسه بیشترین تاثیر را در افزایش میانگین فروش خواهد داشت همچنین همانطور که قابل انتظار هم بود بدترین رابطه در مورد قیمت و میانگین فروش با مقدار ۰.۴۴- می باشد، بی تاثیر ترین ویژگی هم مربوط به اینکه آیا آن کالا درون منطقه شهری (urban) فروش میرود یا خیر می باشد.

سایر ارتباطات قوی نیز مربوط به ارتباط price و compPrice و همچنین US و advertising می باشد.

بخش ۳.۳

تابع نوشته شده برای محاسبه انتروپی یک بردار ndarray به صورت زیر می باشد.

```python
def calculate_entropy(y):
    # Step 1: Count occurrences of each class in y
    class_counts = np.bincount(y)

    # Step 2: Remove zero counts to avoid division/log issues
    class_probs = class_counts[class_counts > 0] / len(y)

    # Step 3: Compute entropy using the formula: -sum(p * log2(p))
    entropy = -np.sum(class_probs * np.log2(class_probs))

    return entropy
```

شکل ۶-۳: کد تابع محاسبه انتروپی

توضیحات :

در ابتدا با استفاده از دستور bincount تعداد تکرار هر کلاس در یک دیتاست را به دست می آوریم (که اولین مقدار آن هم برابر تعداد تکرار اولین کلاس در دیتاست می باشد)

سپس آن را بر طول بردار (تعداد کل نمونه ها) تقسیم میکنیم تا احتمال وقوع هر کلاس را به دست آوریم

سپس مطابق فرمول آنتروپی خروجی تابع را مشخص می کنیم.

بخش ۴.۳

تابع نوشته شده برای محاسبه Information gain یک فیچر با دریافت دیتاست اصلی (والد) در آرگومان اول و دیتاست های تفکیک شده زیر شاخه ها (فرزندان) به صورت زیر می باشد.

```python
def info_gain(parent, children):
    # Step 1: Compute entropy of the parent set
    entropy_parent = calculate_entropy(parent)

    # Step 2: Compute weighted average entropy of children
    total_count = len(parent)
    weighted_entropy = 0

    for child in children:
        weight = len(child) / total_count
        entropy_child = calculate_entropy(child)
        weighted_entropy += weight * entropy_child

    # Step 3: Information gain is the reduction in entropy
    return entropy_parent - weighted_entropy
```

شکل ۷-۳: کد تابع محاسبه Information gain

توضیحات :

در یک حلقه ابتدا وزن زیر شاخه که برابر احتمال آن زیرشاخه یا همان unique value در آن فیچر است محاسبه میشود و در آنتروپی دیتاستی که آن زیرشاخه تفکیک کرده است ضرب می شود و پس از اجرای این روند به ازای تمامی زیرشاخه ها، حاصل جمع از انتروپی دیتاست والد کاسته خواهد شد.

بخش ۵.۳

مفهوم prune کردن

درخت تصمیم، اگر بدون کنترل رشد کند، ممکن است به شدت بزرگ و پیچیده شود، به‌طوری‌که به تمام جزئیات موجود در داده‌های آموزش واکنش نشان دهد. این موضوع باعث می‌شود مدل بیش‌برازش (Overfitting)کند؛ یعنی روی داده‌های آموزش عملکرد خوبی داشته باشد، اما نتواند به‌درستی روی داده‌های جدید تعمیم یابد.

برای مقابله با این مشکل، از تکنیکی به نام هرس کردن (Pruning) استفاده می‌شود. در این تکنیک، برخی گره‌ها یا زیرشاخه‌های درخت که اطلاعات مفیدی برای تصمیم‌گیری ندارند، پس از ساخت درخت حذف یا اصلاح می‌شوند.

انواع هرس کردن:

۱- پیش‌هرس(Pre-Pruning)

قبل از اینکه درخت کاملاً رشد کند، با استفاده از شروطی مثل موارد زیر سعی میکنیم فرآیند رشد درخت را متوقف کنیم.

- عمق بیشینه درخت(max_depth)
- حداقل تعداد نمونه برای تقسیم(min_samples_split)
- حداقل کاهش خطا یا آنتروپی

۲- پس‌هرس(Post-Pruning)

ابتدا اجازه می‌دهیم درخت کامل ساخته شود، سپس با بررسی عملکرد مدل (مثلاً روی داده‌ی اعتبارسنجی)، گره‌هایی که به عملکرد کمک نمی‌کنند را حذف می‌کنیم.

مزایای هرس کردن:

- کاهش بیش‌برازش :(Overfitting) مدل تعمیم‌پذیری بهتری پیدا می‌کند.
- ساده‌تر شدن مدل :درخت کوتاه‌تر و خواناتر می‌شود.
- افزایش سرعت پیش‌بینی :چون درخت کوچک‌تر است، زمان پیش‌بینی کمتر می‌شود.

معایب احتمالی:

- اگر به‌درستی انجام نشود، ممکن است منجر به کم‌برازش (Underfitting) شود.
- نیاز به تنظیم پارامترهایی مثل min_samples_split, alpha یا پارامتر های اعتبارسنجی دارد.
- پس‌هرس ممکن است به داده‌های اعتبارسنجی اضافی نیاز داشته باشد.

تابع GridSearchCv

برای بهینه‌سازی عملکرد مدل درخت تصمیم(Decision Tree) ، لازم است که مقادیر مناسبی برای ابرپارامترها (Hyperparameters) انتخاب کنیم. این پارامترها شامل مواردی مثل:

- max_depth (عمق بیشینه درخت)
- min_samples_split (حداقل تعداد نمونه برای تقسیم یک گره)
- criterion معیار تقسیم مانند gini یا entropy

کتابخانه sklearn ابزاری به نام GridSearchCV ارائه می‌دهد که به ما کمک می‌کند تا با استفاده از جستجوی شبکه‌ای و اعتبارسنجی متقابل(Cross Validation) ، بهترین ترکیب پارامترها را برای مدل پیدا کنیم.

آموزش مدل درخت تصمیم

برای آموزش مدل درخت تصمیم ابتدا ستون sales را از دیتافریم حذف میکنیم و ستون sales_classified را به عنوان خروجی در نظر میگیریم سپس تقسیم بندی داده ها به دو بخش آموزش و تست را انجام می دهیم.

سپس با تعیین هایپرپارامتر های :

criterion به ۲ صورت ناخالصی جینی و انتروپی

max_depth به ۴ حالت [۳, ۵, ۱۰, None]

min_samples_split به ۳ حالت [۲, ۵, ۱۰]

مجموعا ۲۴ درخت تصمیم تشکیل میدهیم که با کمک تابع GridSearchCV و تکنیک Cross Validation

که در اینجا ۵ قرار گرفته است (5 folds) بهترین تنظیمات برای hyperparameter ها را به دست می آوریم.

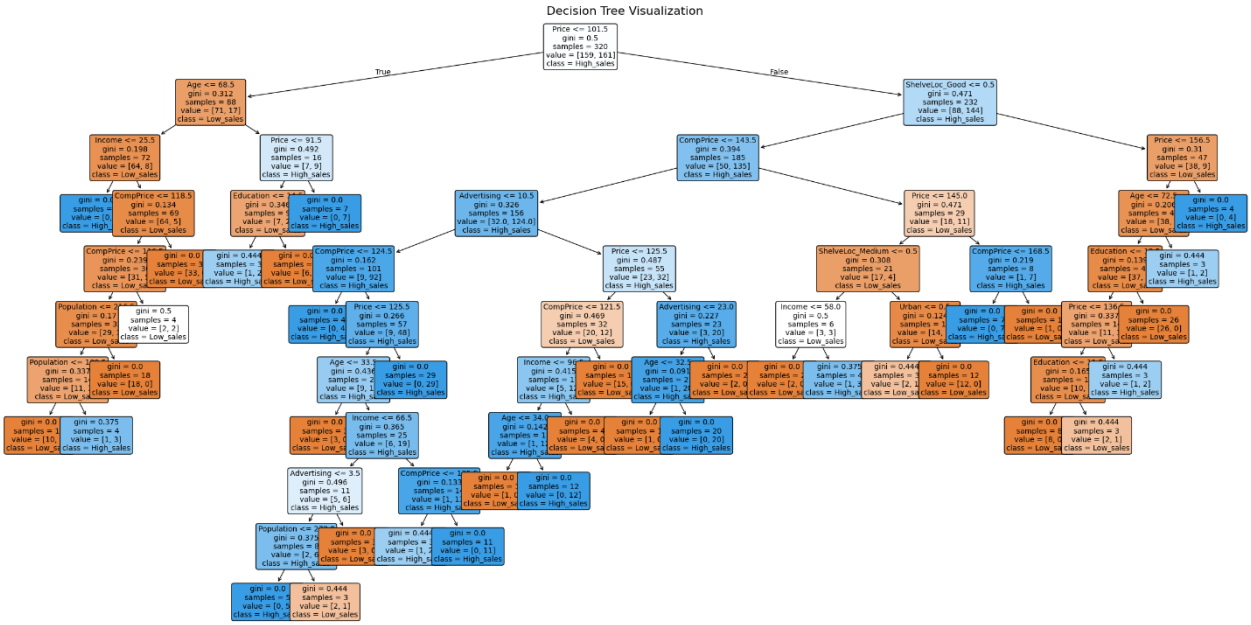
که خروجی به صورت زیر می باشد :

```
Best Parameters: {'criterion': 'gini', 'max_depth': 10, 'min_samples_split': 5}
Best Cross-Validation Score: 0.784375
```

شکل ۸-۳: بهترین هایپرپارامتر های محاسبه شده توسط GridsearchCV

امتیاز به دست آمده نشان دهنده میانگین دقت حاصل از cross-validation روی داده‌های آموزش می باشد.

درخت تصمیم به دست آمده به صورت زیر می باشد که در صورت زوم کردن کیفیت آن حفظ خواهد شد.



شکل ۹-۳: درخت تصمیم آموزش داده شده

تحلیل نوشته های داخل هر گره تصمیم :

برای مثال اگر root node را بخواهیم تحلیل کنیم بدین صورت می باشد که با توجه به criterion که در به دست آوردن بهترین هایپرپارمتر ها gini قرار گرفت در درخت تصمیم ما فیچرprice بیشترین کاهش gini را از دیتاست والد به دیتاست های فرزند ایجاد کرد و باعث شد در root node قرار گیرد حال مربعی که در root node به تصویر کشیده شده است بیانگر یکی از زیر شاخه های price می باشد که برحسب best splitting value =101.5 این فیچر با مقادیر عددی را به ۲ دسته کلی بزرگتر یا کوچکتر مساوی این آستانه تقسیم کرده است همچنین در این مربع بیان شده است که gini دیتاست والد برابر ۰.۵ می باشد و تعداد کل نمونه های دیتاست آموزش برابر ۳۲۰ بوده که با توجه به ماکزیمم gini به دست آمده همانطور که در سطر بعد نشان داده شده است ۱۵۹ داده متعلق به کلاس Low_sales و

۱۶۱ داده متعلق به کلاس high_sales می باشد که با توجه به بیشتر بودن کلاس high_sales این کلاس به عنوان پیشبینی این گره انتخاب شده است.
در مرحله بعدی همانطور که مشخص شده است اگر price≥101.5 یا به عبارت دیگه price≤101.5 غلط باشد سراغ فیچر shelveloc_good میرویم و اگر price≤101.5 سراغ فیچر Age می رویم و به همین ترتیب درخت تصمیم را بر اساس هایپر پارامتر های به دست آمده ایجاد میکنیم.
لازم به ذکر است که درخت تصمیمی که کتابخانه scikit-learn رسم می کند در هر گره، فارغ از نوع گره (تصمیم یا برگ) پیشبینی را در سطر آخر اعلام میکند اما گره های برگ آن گره هایی هستند که بعد از آنها هیچگونه split اتفاق نیفتد.

وقوع overfit

به طور کلی بیش برازش حالتی است که مدل آن‌قدر به جزئیات داده‌های آموزش حساس می‌شود که الگوهای تصادفی و نویز را هم یاد می‌گیرد در نتیجه مدل روی داده آموزش دقت بالایی دارد، اما روی داده تست یا داده‌های جدید عملکرد ضعیفی دارد.
در درخت تصمیمی که رسم شده است با توجه به عمق درخت که توسط بهترین هایپر پارامتر ۱۰ انتخاب شده است در شاهخه های پایین تعداد نمونه های کم با gini خیلی کوک و در بسیاری از موارد gini=0 مشاهده می شود همین ۳ مورد (عمق درخت زیاد، تعداد نمونه کم در زیرشاخه های پایین و ناخالصی جینی صفر) درخت تصمیم را مستعد بروز overfit خواهند کرد.
در مقابل، underfit در حالتی که مدل بسیار ساده است و حتی روی داده‌های آموزش هم نمی‌تواند خوب یاد بگیرد مثلاً درختی که فقط ۱ یا ۲ تقسیم داشته باشد و دقت پایینی روی داده آموزش داشته باشد، اتفاق می افتد.
موثرترین راهکار برخورد با این پدیده در درخت تصمیم تکنیک هرس کردن یا pruning می باشد که در قسمت قبل انواع آن شرح داده شد همچنین یکی دیگر از راهکار های جلوگیری از این موضوع استفاده از تکنیک‌هایensemble learning مانند Random Forest یا Gradient Boosting می باشد که با ترکیب چند درخت، دقت را بالا می‌برند ولی خطر overfitting را کاهش می‌دهند.

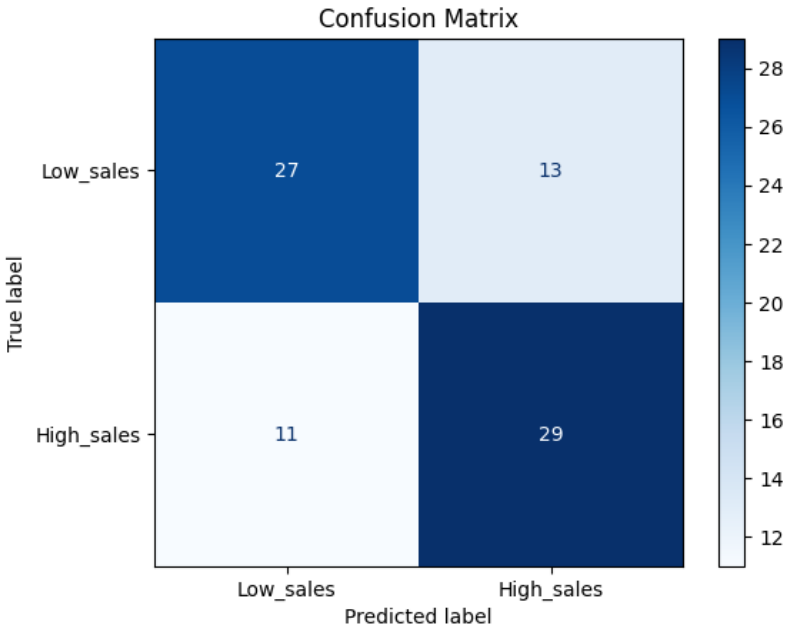
گزارش طبقه بندی و ماتریس درهم ریختگی برای دادگان Test

در صورت استفاده از درخت تصمیم آموزش داده شده برای کلاسبندی دادگان تست نتایج طبقه بندی به صورت زیر می باشد.

```
Test Accuracy: 0.7

Classification Report:
              precision    recall  f1-score   support

  High_sales       0.69      0.72      0.71        40
   Low_sales       0.71      0.68      0.69        40

    accuracy                           0.70        80
   macro avg       0.70      0.70      0.70        80
weighted avg       0.70      0.70      0.70        80
```

شکل ۱۰-۳: گزارش طبقه بندی دادگان تست

همچنین ماتریس درهم ریختگی نیز به صورت زیر می باشد.



شکل ۱۱-۳: ماتریس درهم ریختگی دادگان تست

تحلیل نتایج طبقه بندی و ماتریس درهم ریختگی:

با توجه به accuracy که دقت کلی کلاسبندی را بیان میکند ۷۰ درصد از نمونه هایی که پیشبینی شده اند درست پیشبینی شده اند و برابر trace (جمع عناصر قطری) ماتریس درهم ریختگی تقسیم بر کل داده های تست که ۸۰ نمونه می باشد را بیان میکند.

در مورد precision که دقت جزئی برای هر کلاس را به این نحو محاسبه میکند که عنصر قطری را بر مجموع درایه های هر ستون تقسیم خواهد کرد برای کلاس High_ class اندکی کمتر از Low_class می باشد.

از ماتریس درهم ریختگی نیز مشاهده می شود که ۱۳ نمونه باوجود اینکه واقعا از کلاس Low بوده اند اما به اشتباه کلاس High کلاس بندی شده اند و ۱۱ نمونه باوجود اینکه واقعا از کلاس High بوده اند اما به اشتباه کلاس Low کلاس بندی شده اند.