

دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق

درس مبانی سیستم‌های هوشمند

استاد: دکتر مهدی علیاری

مینی پروژه اول

نام و نام خانوادگی	محمدامین محمدیون شبستری
شماره دانشجویی	۴۰۱۲۲۵۰۳
نام و نام خانوادگی	محمد سبحان سخایی
شماره دانشجویی	۴۰۱۱۹۲۵۳
نام و نام خانوادگی	میانا یوسفی مقدم
شماره دانشجویی	۴۰۱۲۴۰۹۳
تاریخ	۱۴۰۴ آبان
لینک‌های کولب و گیت‌هاب:	لینک کولب سؤال ۴ لینک کولب سؤال ۵ لینک گیت‌هاب



فهرست مطالب

۵	۱	پرسش‌های تحلیلی
۵	۱.۱	پاسخ پرسش یک
۶	۱.۱.۱	قسمت الف
۶	۲.۱.۱	قسمت ب
۹	۲.۱	پاسخ پرسش دو
۹	۱.۲.۱	الف: حل با روش پرسپترون (Perceptron)
۱۲	۲.۲.۱	ب: حل با روش حداقل مربعات (Least Square)
۱۵	۳.۲.۱	پ: حل با روش فیشر (LDA)
۱۹	۴.۲.۱	مقایسه هر سه روش و بررسی نتایج
۲۱	۳.۱	پاسخ پرسش سه
۲۴	۲	سوالات پیاده‌سازی
۲۴	۱.۲	پرسش چهارم
۲۴	۱.۱.۲	بخش اول: تحلیل و کاوش داده‌ها (EDA)
۳۱	۲.۱.۲	بخش دوم: پیش‌پردازش داده‌ها
۳۱	۳.۱.۲	بخش سوم: انتخاب ویژگی و مدل‌سازی کلاسیک
۳۷	۴.۱.۲	بخش چهارم: نمایش ویژگی‌ها با استفاده از کاهش ابعاد
۴۱	۲.۲	پاسخ پرسش پنجم
۴۱	۱.۲.۲	بخش اول: مطالعه مقاله
۴۳	۲.۲.۲	بخش دوم: دادگان
۴۵	۳.۲.۲	بخش سوم: تحلیل اکتشافی داده‌ها (EDA)
۴۶	۴.۲.۲	بخش چهارم: پیاده‌سازی نوت‌بوک (q5.ipynb)



فهرست تصاویر

۹	داده‌های ارائه شده در صورت سوال.	۱
۱۲	داده‌های جدا شده توسط خط پیدا شده توسط الگوریتم پرسپترون (Perceptron).	۲
۱۵	داده‌های جدا شده توسط خط پیدا شده توسط الگوریتم کمترین مربعات (Least Square).	۳
۱۹	داده‌های جدا شده توسط خط پیدا شده توسط الگوریتم فیشر (LDA).	۴
۲۵	df.describe()	۷
۲۵	df.info()	۶
۲۵	df.head()	۵
۲۶	نقشه حرارتی همبستگی ویژگی‌ها (seaborn)	۸
۲۷	نقشه حرارتی همبستگی ویژگی‌ها (plotly)	۹
۲۸	نمودار Pairplot	۱۰
۲۹	Hexbinplot with Numerical features	۱۱
۲۹	Hexbinplot with Categorical features	۱۲
۳۰	نمودار دایره‌ای	۱۴
۳۰	نمودار میله‌ای	۱۳
۳۲	ویژگی‌های مهم استخراج شده از Regression Lasso	۱۵
۳۲	ویژگی‌های مهم استخراج شده از RFE	۱۶
۳۴	AUC-ROC (RFE)	۱۸
۳۴	Confusion Matrix (RFE)	۱۷
۳۴	ضرایب مدل رگرسیون لجستیک برای ویژگی‌های استخراج شده از RFE	۱۹
۳۶	AUC-ROC (LassoRegression)	۲۱
۳۶	Confusion Matrix (LassoRegression)	۲۰
۳۷	ضرایب مدل رگرسیون لجستیک برای ویژگی‌های استخراج شده از LassoRegression	۲۲
۳۸	PCA Projection to 2 Components	۲۳
۳۹	LDA Projection to 2 Components	۲۴
۴۱	MLP Projection to 2 Components	۲۵



فهرست جداول

۸	خلاصه‌ی نتایج ارزیابی مدل برای کلاس‌های مختلف	۱
۴۴	ویژگی‌های مجموعه‌داده، نوع داده و شرح مفهومی هر ستون	۲
۴۵	تعداد مقادیر یکتاوی هر ویژگی در مجموعه‌داده	۳
۴۷	نمونه‌ای از ۵ سطر اول داده‌ها	۴



فهرست برنامه‌ها

۲۴	بارگذاری داده‌ها و نمایش اطلاعات اولیه	۱
۲۵	کد مربوط به ترسیم نمودار همبستگی و بیزگی‌ها	۲
۲۷	کد مربوط به ترسیم نمودار pairplot	۳
۲۸	کد مربوط به ترسیم نمودار Hexbin	۴
۲۹	کد مربوط به ترسیم توزیع کلاس‌ها	۵
۳۱	کد مربوط به نرمالسازی	۶
۳۱	Extract Important Features	۷
۳۲	Train model and evaluation (RFE)	۸
۳۳	Evaluation (RFE)	۹
۳۵	Train model and evaluation (LassoRegression)	۱۰
۳۵	Evaluation (LassoRegression)	۱۱
۳۷	PCA Projection to 2 Components	۱۲
۳۸	LDA Projection to 2 Components	۱۳
۳۹	Training MLP	۱۴
۴۳	بررسی ابعاد داده	۱۵
۴۳	نمایش نوع داده‌ی هر ستون	۱۶
۴۴	محاسبه تعداد مقادیر یکتا برای هر ستون	۱۷



۱ پرسش‌های تحلیلی

۱.۱ پاسخ پرسش یک

در این قسمت، محاسبه‌ی معیارهای حساسیت (Sensitivity) و ویژگی (Specificity) مورد نیاز است. برای انجام این محاسبات، ابتدا باید با مفاهیم پایه‌ای مرتبط آشنا شویم.

برای درک بهتر، از یک مثال استفاده می‌کنیم؛ فرض کنید در حال طراحی یک سیستم تشخیص سرطان هستیم:

- کلاس مثبت (Positive): کلاسی است که در آن، مراجعه‌کننده بیمار و مبتلا به سرطان است.

- کلاس منفی (Negative): کلاسی است که در آن، مراجعه‌کننده سالم بوده و هیچ‌گونه بیماری ندارد.

چهار حالت مختلف برای پیش‌بینی مدل در مورد هر بیمار وجود دارد:

• مثبت حقیقی (True Positive - TP)

- مقدار واقعی: بیمار مبتلا به سرطان است.

- مقدار پیش‌بینی شده توسط مدل: بیمار مبتلا به سرطان است.

- نتیجه: تشخیص مدل در مورد بیمار بودن فرد، درست بوده است.

• منفی حقیقی (True Negative - TN)

- مقدار واقعی: بیمار سالم است.

- مقدار خروجی مدل: بیمار سالم است.

- نتیجه: مدل به درستی فرد سالم را (که بیماری ندارد) تشخیص داده است.

• منفی کاذب (False Negative - FN)

- مقدار واقعی: بیمار مبتلا به سرطان است.

- مقدار پیش‌بینی شده: بیمار سالم است.

- نتیجه: این خطرناک‌ترین حالت است؛ اشتباهی که می‌تواند منجر به عواقب جبران‌ناپذیر (مانند مرگ بیمار) شود. در این مثال، FN‌ها اهمیت حیاتی دارند.

• مثبت کاذب (False Positive - FP)

- مقدار واقعی: بیمار سالم است.

- مقدار پیش‌بینی شده: بیمار مبتلا به سرطان است.

- نتیجه: تشخیص اشتباه بوده و منجر به یک هشدار کاذب (False Alarm) شده است. گرچه این حالت نیز خطاست، اما در این سناریوی خاص، خطر آن کمتر از FN است و نهایتاً با یک تست مجدد، اشتباه مشخص می‌شود.



(Sensitivity)

این متریک، که یکی از متریک‌های ارزیابی است، با نام‌های نرخ مثبت حقیقی (True Positive Rate) یا (Recall) نیز شناخته می‌شود. هدف این متریک آن است که مشخص کند از میان تمامی موارد مثبت واقعی (در مثال ما، تمام افراد مبتلا به سرطان)، مدل چه درصدی را به درستی شناسایی کرده است؟ در واقعیت، مانمی خواهیم حتی یک بیمار را از دست بدھیم (به اشتباہ Miss کنیم) و به او بگوییم سالم است. هزینه‌ی FN بسیار زیاد است.

$$\text{(Recall) Sensitivity} = \frac{TP}{TP + FN} \quad (1)$$

(Specificity)

Specificity، یا نرخ منفی حقیقی (True Negative Rate)، مشخص می‌کند که از میان تمامی موارد منفی واقعی (افرادی که بیمار نیستند)، مدل چه درصدی را به درستی تشخیص داده است؟ این معیار می‌سنجد که مدل چقدر در شناسایی افراد سالم موفق عمل می‌کند.

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (2)$$

با دانستن این مفاهیم، به حل سوال می‌پردازیم.

۱.۱.۱ قسمت الف

فرمول‌های محاسبه‌ی حساسیت و ویژگی به شرح زیر است:

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (3)$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (4)$$

۲.۱.۱ قسمت ب

در این سوال، یک ماتریس درهم‌زننگی (Confusion Matrix) چهار کلاسه به ما داده شده است. کلاس‌های واقعی c1، c2، c3 و c4 هستند و کلاس‌های پیش‌بینی شده c'1، c'2، c'3 و c'4 می‌باشند. در ابتدا، باید مقادیر TN، TP، FN و FP را برای هر کلاس به دست آوریم.

کلاس c1:

\bullet $TP = 45$: مقدار واقعی، c1 بوده و مقدار پیش‌بینی شده توسط مدل نیز c'1 است.

\bullet $FN = 6$: مقدار واقعی، c1 بوده، اما مقدار پیش‌بینی شده توسط مدل، c'1 نبوده است.

\bullet $FP = 5$: مقدار پیش‌بینی شده توسط مدل، c'1 بوده، در حالی که مقدار واقعی c1 نبوده است.



\bullet $TN = 90$: مقدار واقعی، $c1$ نبوده و مقدار پیش‌بینی شده توسط مدل نیز $c1$ نبوده است.

$$(محاسبه: 146 - (45 + 6 + 5) = 90)$$

محاسبات:

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{45}{45 + 6} = \frac{45}{51} \approx 0.8824 \quad (88.24\%) \quad (5)$$

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{90}{90 + 5} = \frac{90}{95} \approx 0.9474 \quad (94.74\%) \quad (6)$$

کلاس $c2$:

\bullet $TP = 32$: مقدار واقعی، $c2$ بوده و مقدار پیش‌بینی شده توسط مدل نیز $c2$ است.

\bullet $FN = 11$: مقدار واقعی، $c2$ بوده، اما مقدار پیش‌بینی شده توسط مدل، $c2$ نبوده است.

\bullet $FP = 7$: مقدار پیش‌بینی شده توسط مدل، $c2$ بوده، در حالی که مقدار واقعی $c2$ نبوده است.

\bullet $TN = 96$: مقدار واقعی، $c2$ نبوده و مقدار پیش‌بینی شده توسط مدل نیز $c2$ نبوده است.

محاسبات:

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{32}{32 + 11} = \frac{32}{43} \approx 0.7442 \quad (74.42\%) \quad (7)$$

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{96}{96 + 7} = \frac{96}{103} \approx 0.9320 \quad (93.20\%) \quad (8)$$

کلاس $c3$:

\bullet $TP = 16$: مقدار واقعی، $c3$ بوده و مقدار پیش‌بینی شده توسط مدل نیز $c3$ است.

\bullet $FN = 14$: مقدار واقعی، $c3$ بوده، اما مقدار پیش‌بینی شده توسط مدل، $c3$ نبوده است.

\bullet $FP = 4$: مقدار پیش‌بینی شده توسط مدل، $c3$ بوده، در حالی که مقدار واقعی $c3$ نبوده است.

\bullet $TN = 112$: مقدار واقعی، $c3$ نبوده و مقدار پیش‌بینی شده توسط مدل نیز $c3$ نبوده است.

محاسبات:

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{16}{16 + 14} = \frac{16}{30} \approx 0.5333 \quad (53.33\%) \quad (9)$$

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{112}{112 + 4} = \frac{112}{116} \approx 0.9655 \quad (96.55\%) \quad (10)$$



کلاس c4

\bullet $TP = 20$: مقدار واقعی، C4 بوده و مقدار پیش‌بینی شده توسط مدل نیز C4 است.

\bullet $FN = 2$: مقدار واقعی، C4 بوده، اما مقدار پیش‌بینی شده توسط مدل، C4 نبوده است.

\bullet $FP = 17$: مقدار پیش‌بینی شده توسط مدل، C4 بوده، در حالی که مقدار واقعی C4 نبوده است.

\bullet $TN = 107$: مقدار واقعی، C4 نبوده و مقدار پیش‌بینی شده توسط مدل نیز C4 نبوده است.

محاسبات:

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{20}{20 + 2} = \frac{20}{22} \approx 0.9091 \quad (90.91\%) \quad (11)$$

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{107}{107 + 17} = \frac{107}{124} \approx 0.8629 \quad (86.29\%) \quad (12)$$

نتیجه‌گیری

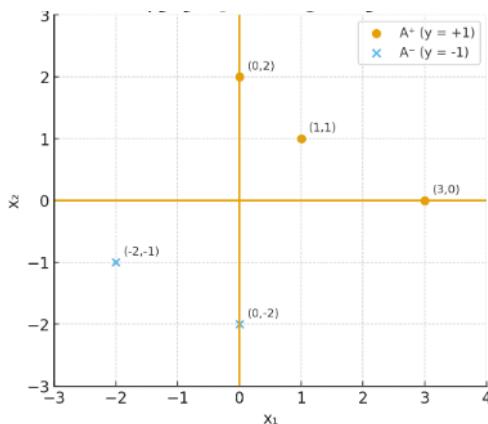
جدول زیر خلاصه‌ی مقادیر و نتایج محاسبات را برای هر چهار کلاس نشان می‌دهد:

جدول ۱: خلاصه‌ی نتایج ارزیابی مدل برای کلاس‌های مختلف

Specificity	Sensitivity (Recall)	TN	FP	FN	TP	کلاس
94.74%	88.24%	۹۰	۵	۶	۴۵	C1
93.20%	74.42%	۹۶	۷	۱۱	۳۲	C2
96.55%	53.33%	۱۱۲	۴	۱۴	۱۶	C3
86.29%	90.91%	۱۰۷	۱۷	۲	۲۰	C4



۲.۱ پاسخ پرسش دو



شکل ۱: داده‌های ارائه شده در صورت سوال.

برای حل این سوال، ابتدا از مفهوم «بردار افزوده» (Augmented Vector) استفاده می‌کنیم. مطابق معادله‌ی خط، داریم:

$$y = wx + b \quad (13)$$

این معادله شامل یک بایاس (bias)، یعنی b ، است. برای ساده‌سازی محاسبات، تمام بردارهای داده x را که متشکل از x_1 و x_2 هستند، به x' تبدیل می‌کنیم:

$$x' = (x_1, x_2, 1) \quad (14)$$

متناسب با این موضوع، بردار وزن مانیز تغییر کرده و خواهیم داشت:

$$w' = (w_1, w_2, b) \quad (15)$$

با این کار، معادله‌ی خط ما به صورت $y = w' \cdot x' + b$ ساده می‌شود.

ماتریس X' ، که شامل تمام بردارهای افزوده‌ی داده‌ها است، به شکل زیر خواهد بود (بافرض اینکه کلاس‌های داده‌ها $\{+1, -1\}$ باشند تا با نتایج سوال هماهنگ شود):

$$X' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 3 & 0 & 1 \\ -2 & -1 & 1 \\ 0 & -2 & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} +1 \\ +1 \\ +1 \\ -1 \\ -1 \end{bmatrix} \quad (16)$$

۱.۲.۱ الف: حل با روش پرسپترون (Perceptron)

در این الگوریتم، ما به یک وزن اولیه (Initial Weight) برای شروع نیاز داریم که مطابق گفته‌ی سوال، $w' = (0, 0, 0)$ را در نظر می‌گیریم.



نحوه تصمیم‌گیری پرسپترون: اگر $w' \cdot x' > 0$ باشد، نقطه به کلاس مثبت تعلق دارد و اگر $w' \cdot x' < 0$ باشد، نقطه به کلاس منفی تعلق دارد.

در الگوریتم پرسپترون، وزن برای هر بردار داده بررسی شده و اگر طبقه‌بندی به درستی انجام نشده باشد، وزن‌ها به روزرسانی می‌شوند:

$$w(t+1) = w(t) + \eta \cdot y \cdot x' \quad (17)$$

که η همان نرخ یادگیری (Learning Rate) است (در این سوال $\eta = 1$ فرض می‌شود).

: ۱ Epoch بررسی

- نقطه ($y_1 = +1$) کلاس $x'_1 = (1, 1, 1)$
وزن فعلی: $w'(0) = (0, 0, 0)$

$$w'(0) \cdot x'_1 = (0, 0, 0) \cdot (1, 1, 1)^T = 0 \quad (18)$$

بررسی شرط:

$$y_1 \cdot (w'(0) \cdot x'_1) = (+1) \cdot (0) = 0 \quad (19)$$

مطابق الگوریتم پرسپترون، اگر $y \cdot w \leq 0$ باشد، نقطه اشتباه طبقه‌بندی شده است. در نتیجه وزن باید آپدیت شود:

$$w'(1) = w'(0) + \eta \cdot y_1 \cdot x'_1 = (0, 0, 0) + 1 \cdot (+1) \cdot (1, 1, 1) = (1, 1, 1) \quad (20)$$

- نقطه ($y_2 = +1$) کلاس $x'_2 = (0, 2, 1)$
وزن فعلی: $w'(1) = (1, 1, 1)$

$$w'(1) \cdot x'_2 = (1, 1, 1) \cdot (0, 2, 1)^T = 0 + 2 + 1 = 3 \quad (21)$$

بررسی شرط:

$$y_2 \cdot (w'(1) \cdot x'_2) = (+1) \cdot (3) = 3 > 0 \quad (22)$$

طبقه‌بندی درست است. وزن آپدیت نمی‌شود.

- نقطه ($y_3 = +1$) کلاس $x'_3 = (3, 0, 1)$
وزن فعلی: $w'(* - +96) = (1, 1, 1)$

$$w'(2) \cdot x'_3 = (1, 1, 1) \cdot (3, 0, 1)^T = 3 + 0 + 1 = 4 \quad (23)$$

بررسی شرط:

$$y_3 \cdot (w'(2) \cdot x'_3) = (+1) \cdot (4) = 4 > 0 \quad (24)$$

طبقه‌بندی درست است. وزن آپدیت نمی‌شود.



- نقطه $(y_4 = -1)$ کلاس $x'_4 = (-2, -1, 1)$
وزن فعلی: $w'(3) = (1, 1, 1)$

$$w'(3) \cdot x'_4 = (1, 1, 1) \cdot (-2, -1, 1)^T = -2 - 1 + 1 = -2 \quad (25)$$

بررسی شرط:

$$y_4 \cdot (w'(3) \cdot x'_4) = (-1) \cdot (-2) = 2 > 0 \quad (26)$$

طبقه‌بندی درست است. وزن آپدیت نمی‌شود.

- نقطه $(y_5 = -1)$ کلاس $x'_5 = (0, -2, 1)$
وزن فعلی: $w'(4) = (1, 1, 1)$

$$w'(4) \cdot x'_5 = (1, 1, 1) \cdot (0, -2, 1)^T = 0 - 2 + 1 = -1 \quad (27)$$

بررسی شرط:

$$y_5 \cdot (w'(4) \cdot x'_5) = (-1) \cdot (-1) = 1 > 0 \quad (28)$$

طبقه‌بندی درست است. وزن آپدیت نمی‌شود.

پس از پایان Epoch 1، مشاهده می‌شود که وزن $w' = (1, 1, 1)$ همه‌ی داده‌ها را به درستی طبقه‌بندی کرده است. نکته‌ای که وجود دارد، این است که این الگوریتم تا جایی تکرار می‌شود که یک دور کامل روی تمامی داده‌ها چرخیده باشد و دیگر وزن w' آپدیت نشود. در این حالت (چون پس از آپدیت اول، هیچ آپدیت دیگری در این Epoch رخ نداد)، الگوریتم پایان می‌یابد. در غیر این صورت، برای Epoch 2, 3, ... ادامه می‌یافتد.

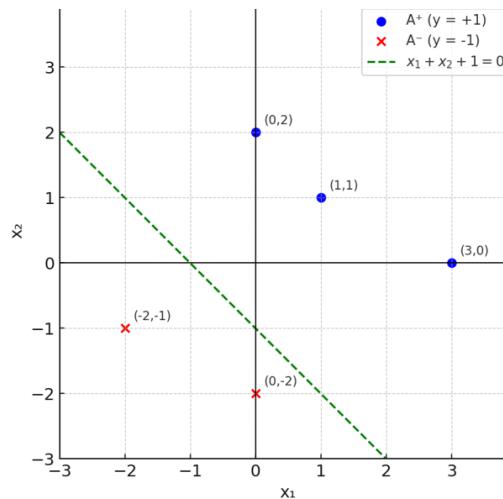
نتیجه قسمت الف: بردار وزن نهایی ما برابر شد با:

$$w' = (w_1, w_2, b) = (1, 1, 1) \quad (29)$$

به سبب همین موضوع، خط جداساز ما برابر است با:

$$1 \cdot x_1 + 1 \cdot x_2 + 1 = 0 \quad (30)$$

که می‌تواند به بهترین شکل، داده‌های ارائه شده را جدا کند.



شکل ۲: داده‌های جدا شده توسط خط پیدا شده توسط الگوریتم پرسپترون (Perceptron).

۲.۲.۱ ب: حل با روش حداقل مربعات (Least Square)

در الگوریتم حداقل مربعات (Least Square)، ما به مسئله به شیوه‌ای مشابه رگرسیون نگاه می‌کنیم. ما در صدد یافتن تابعی به شکل f هستیم که به بهترین شکل کلاس‌های ما از هم جدا کند. این امر با کمینه کردن خطای مجموع مربعات رخ می‌دهد:

$$\min \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (31)$$

ماتریس داده‌های افزوده X' (با ابعاد 3×5) و بردار برچسب‌ها y (با ابعاد 1×5) به صورت زیر تعریف می‌شوند:

$$X' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 3 & 0 & 1 \\ -2 & -1 & 1 \\ 0 & -2 & 1 \end{bmatrix} \quad (32)$$

$$y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \quad (33)$$

قرار است که خطای زیر را کمینه کنیم:

$$J(w') = \sum_{i=1}^5 (y_i - \hat{y}_i)^2 \quad (34)$$



که در آن $\hat{y} = X'w'$ است. اگر این معادله به فرم ماتریسی نوشته شود، تابع هزینه $J(w')$ برابر است با:

$$J(w') = (y - X'w')^T(y - X'w') \quad (35)$$

برای یافتن بردار w' (که ضرایب معادله خط ما است) و کمینه کردن $J(w')$ ، باید از این تابع نسبت به w' مشتق بگیریم و آن را برابر با صفر قرار دهیم.

محاسبات مشتق و ساده‌سازی:

۱. ابتدا تابع هزینه را بسط می‌دهیم:

$$J(w') = (y^T - (X'w')^T)(y - X'w') \quad (36)$$

$$J(w') = (y^T - w'^T X'^T)(y - X'w') \quad (37)$$

$$J(w') = y^T y - y^T X'w' - w'^T X'^T y + w'^T X'^T X'w' \quad (38)$$

۲. از آنجایی که $y^T X'w'$ یک اسکالر است، ترانهادهی آن با خودش برابر است: $(y^T X'w')^T = w'^T X'^T y$. بنابراین:

$$J(w') = y^T y - 2w'^T X'^T y + w'^T X'^T X'w' \quad (39)$$

۳. اکنون نسبت به w' مشتق می‌گیریم (با استفاده از قواعد مشتق ماتریسی) می‌رسیم:

$$\nabla_{w'} J(w') = \frac{\partial}{\partial w'}(y^T y) - \frac{\partial}{\partial w'}(2w'^T X'^T y) + \frac{\partial}{\partial w'}(w'^T X'^T X'w') \quad (40)$$

$$\nabla_{w'} J(w') = 0 - 2X'^T y + 2(X'^T X')w' \quad (41)$$

۴. با قرار دادن مشتق برابر با صفر، به «معادله نرمال» (Normal Equation) می‌رسیم:

$$-2X'^T y + 2(X'^T X')w' = 0 \quad (42)$$

$$(X'^T X')w' = X'^T y \quad (43)$$

در نهایت، بردار وزن‌های ما با حل معادله بالا به صورت زیر خواهد بود (که به آن «شبه‌وارون» یا Pseudo-inverse نیز گفته می‌شود):

$$w' = (X'^T X')^{-1} X'^T y \quad (44)$$



حل معادله و محاسبه وزن‌ها:

۱. محاسبه X'^T :

$$X'^T = \begin{bmatrix} 1 & 0 & 3 & -2 & 0 \\ 1 & 2 & 0 & -1 & -2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (45)$$

۲. محاسبه $X'^T X'$:

$$X'^T X' = \begin{bmatrix} 1 & 0 & 3 & -2 & 0 \\ 1 & 2 & 0 & -1 & -2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 3 & 0 & 1 \\ -2 & -1 & 1 \\ 0 & -2 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 3 & 2 \\ 3 & 10 & 0 \\ 2 & 0 & 5 \end{bmatrix} \quad (46)$$

۳. محاسبه $X'^T y$:

$$X'^T y = \begin{bmatrix} 1 & 0 & 3 & -2 & 0 \\ 1 & 2 & 0 & -1 & -2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix} \quad (47)$$

۴. محاسبه $(X'^T X')^{-1}$ (وارون ماتریس $(X'^T X')$):

$$(X'^T X')^{-1} = \begin{bmatrix} 14 & 3 & 2 \\ 3 & 10 & 0 \\ 2 & 0 & 5 \end{bmatrix}^{-1} = \frac{1}{615} \begin{bmatrix} 50 & -15 & -20 \\ -15 & 66 & 6 \\ -20 & 6 & 131 \end{bmatrix} \quad (48)$$

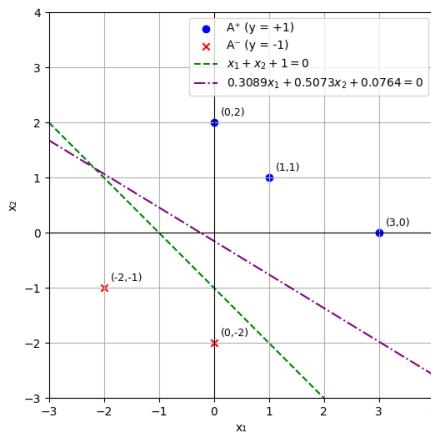
۵. محاسبه نهایی w' :

$$w' = (X'^T X')^{-1} (X'^T y) = \frac{1}{615} \begin{bmatrix} 50 & -15 & -20 \\ -15 & 66 & 6 \\ -20 & 6 & 131 \end{bmatrix} \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix} \quad (49)$$

$$w' = \frac{1}{615} \begin{bmatrix} (300 - 90 - 20) \\ (-90 + 396 + 6) \\ (-120 + 36 + 131) \end{bmatrix} = \frac{1}{615} \begin{bmatrix} 190 \\ 312 \\ 47 \end{bmatrix} \approx \begin{bmatrix} 0.3089 \\ 0.5073 \\ 0.0764 \end{bmatrix} \quad (50)$$

معادلهٔ خط نهایی: خطی که به دست آمده، مطابق $w_1x_1 + w_2x_2 + b = 0$ یا $w' \cdot x' = 0$ است. با جایگذاری وزن‌های محاسبه شده:

$$0.3089x_1 + 0.5073x_2 + 0.0764 = 0 \quad (51)$$



شکل ۳: داده‌های جدا شده توسط خط پیدا شده توسط الگوریتم کمترین مربعات (Least Square).

۳.۲.۱ ب: حل با روش فیشر (LDA)

در این قسمت، مسئله با استفاده از روش تحلیل ممیزی خطی فیشر، که به الگوریتم LDA (Linear Discriminant Analysis) معروف است، حل می‌شود. اساس این الگوریتم با دو روش قابلی (که مبتنی بر بردار افزوده بودند) متفاوت است. در LDA، ما به دنبال یافتن یک جهت (بردار w) برای تصویرسازی (Projection) داده‌ها هستیم؛ به گونه‌ای که داده‌ها پس از تصویر شدن بر روی این جهت، بهترین تفکیک‌پذیری را داشته باشند.

از دیدگاه فیشر، بهترین تفکیک‌پذیری به معنای بیشینه کردن فاصله بین میانگین کلاس‌ها (پراکندگی بین کلاسی) و کمینه کردن پراکندگی (واریانس) درون هر کلاس (پراکندگی درون کلاسی) است.

۱. محاسبه میانگین کلاس‌ها (μ_i): ابتدا داده‌های هر کلاس را (بدون افزودن بایاس) مشخص می‌کنیم:

$$\bullet \text{ کلاس } C_1: N_1 = 3, x_1 = (0, 2)^T, x_2 = (1, 1)^T : (y = +1)$$

$$\bullet \text{ کلاس } C_2: N_2 = 2, x_3 = (3, 0)^T, x_4 = (-2, -1)^T : (y = -1)$$

میانگین کلاس C_1 :

$$\mu_1 = \frac{1}{N_1} \sum_{x \in C_1} x = \frac{1}{3} \left(\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \right) = \frac{1}{3} \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 4/3 \\ 1 \\ 1 \end{bmatrix} \quad (52)$$

میانگین کلاس C_2 :

$$\mu_2 = \frac{1}{N_2} \sum_{x \in C_2} x = \frac{1}{2} \left(\begin{bmatrix} -2 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ -2 \end{bmatrix} \right) = \frac{1}{2} \begin{bmatrix} -2 \\ -1 \\ -3 \end{bmatrix} = \begin{bmatrix} -1 \\ -3/2 \end{bmatrix} \quad (53)$$



۲. محاسبه‌ی ماتریس پراکنده‌ی بین کلاسی (S_B): این ماتریس، پراکنده‌ی بین میانگین‌های کلاس‌ها را اندازه‌گیری می‌کند. برای دو کلاس، این ماتریس متناسب است با ضرب خارجی تفاضل میانگین‌ها:

$$S_B \propto (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \quad (54)$$

ابتدا بردار تفاضل میانگین‌ها را محاسبه می‌کنیم:

$$\mu_1 - \mu_2 = \begin{bmatrix} 4/3 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ -3/2 \end{bmatrix} = \begin{bmatrix} 4/3 + 1 \\ 1 + 3/2 \end{bmatrix} = \begin{bmatrix} 7/3 \\ 5/2 \end{bmatrix} \quad (55)$$

سپس S_B را محاسبه می‌کنیم (برای سادگی در محاسبات بعدی، از همین فرم استفاده می‌کنیم):

$$S_B = \begin{bmatrix} 7/3 \\ 5/2 \end{bmatrix} \begin{bmatrix} 7/3 & 5/2 \end{bmatrix} = \begin{bmatrix} 49/9 & 35/6 \\ 35/6 & 25/4 \end{bmatrix} \quad (56)$$

(توجه: تعریف کلی‌تر $S_B = \sum N_i(\mu_i - \mu)(\mu_i - \mu)^T$ نیز به ماتریسی متناسب با ماتریس بالا منجر می‌شود و جهت نهایی w را تغییر نمی‌دهد).

۳. محاسبه‌ی ماتریس پراکنده‌ی درون کلاسی (S_W): این ماتریس، مجموع پراکنده‌ی های داخلی هر کلاس را نشان می‌دهد:

$$S_W = S_1 + S_2 \quad \text{که} \quad S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \quad (57)$$

ابتدا S_1 (برای کلاس C_1) را محاسبه می‌کنیم:

$$x_1 - \mu_1 = (1, 1)^T - (4/3, 1)^T = (-1/3, 0)^T \bullet$$

$$x_2 - \mu_1 = (0, 2)^T - (4/3, 1)^T = (-4/3, 1)^T \bullet$$

$$x_3 - \mu_1 = (3, 0)^T - (4/3, 1)^T = (5/3, -1)^T \bullet$$

$$S_1 = \begin{bmatrix} -1/3 \\ 0 \end{bmatrix} \begin{bmatrix} -1/3 & 0 \end{bmatrix} + \begin{bmatrix} -4/3 \\ 1 \end{bmatrix} \begin{bmatrix} -4/3 & 1 \end{bmatrix} + \begin{bmatrix} 5/3 \\ -1 \end{bmatrix} \begin{bmatrix} 5/3 & -1 \end{bmatrix} \quad (58)$$

$$S_1 = \begin{bmatrix} 1/9 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 16/9 & -4/3 \\ -4/3 & 1 \end{bmatrix} + \begin{bmatrix} 25/9 & -5/3 \\ -5/3 & 1 \end{bmatrix} = \begin{bmatrix} 42/9 & -9/3 \\ -9/3 & 2 \end{bmatrix} = \begin{bmatrix} 14/3 & -3 \\ -3 & 2 \end{bmatrix} \quad (59)$$

سپس S_2 (برای کلاس C_2) را محاسبه می‌کنیم:

$$x_4 - \mu_2 = (-2, -1)^T - (-1, -3/2)^T = (-1, 1/2)^T \bullet$$

$$x_5 - \mu_2 = (0, -2)^T - (-1, -3/2)^T = (1, -1/2)^T \bullet$$



$$S_2 = \begin{bmatrix} -1 \\ 1/2 \end{bmatrix} \begin{bmatrix} -1 & 1/2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1/2 \end{bmatrix} \begin{bmatrix} 1 & -1/2 \end{bmatrix} \quad (60)$$

$$S_2 = \begin{bmatrix} 1 & -1/2 \\ -1/2 & 1/4 \end{bmatrix} + \begin{bmatrix} 1 & -1/2 \\ -1/2 & 1/4 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ -1 & 1/2 \end{bmatrix} \quad (61)$$

در نهایت، S_W را جمع می‌زنیم:

$$S_W = S_1 + S_2 = \begin{bmatrix} 14/3 & -3 \\ -3 & 2 \end{bmatrix} + \begin{bmatrix} 2 & -1 \\ -1 & 1/2 \end{bmatrix} = \begin{bmatrix} 20/3 & -4 \\ -4 & 5/2 \end{bmatrix} \quad (62)$$

۴. بهینه‌سازی و یافتن w : در نهایت، ما به دنبال بردار w هستیم که معیار فیشر $J(w)$ (نسبت پراکندگی بین کلاسی به پراکندگی درون کلاسی در جهت w) را بیشینه کند:

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \quad (63)$$

برای یافتن w که $J(w)$ را بیشینه می‌کند، از $J(w)$ نسبت به w مشتق گرفته و آن را برابر صفر قرار می‌دهیم (با استفاده از قاعده مشتق خارج قسمت):

$$\frac{\partial J}{\partial w} = \frac{(2S_B w)(w^T S_W w) - (w^T S_B w)(2S_W w)}{(w^T S_W w)^2} = 0 \quad (64)$$

که منجر به تساوی زیر می‌شود:

$$(S_B w)(w^T S_W w) = (S_W w)(w^T S_B w) \quad (65)$$

با تقسیم طرفین بر اسکالر $(w^T S_W w)$ ، داریم:

$$S_B w = \left(\frac{w^T S_B w}{w^T S_W w} \right) S_W w \quad (66)$$

عبارت داخل پرانتز همان $J(w)$ است که یک اسکالر (مقدار ویژه، λ) می‌باشد:

$$S_B w = \lambda S_W w \implies S_W^{-1} S_B w = \lambda w \quad (67)$$

این یک مسئله‌ی مقدار ویژه تعمیم‌یافته است. حال، تعریف S_B (برای دو کلاس) را جایگذاری می‌کنیم:

$$S_W^{-1}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = \lambda w \quad (68)$$

از آنجایی که λ یک اسکالر است (آن را v می‌نامیم)، سمت چپ معادله، برداری در جهت $S_W^{-1}(\mu_1 - \mu_2)^T w$ خواهد بود. بنابراین:

$$w \propto S_W^{-1}(\mu_1 - \mu_2) \quad (69)$$

این همان نتیجه‌ی مورد نظر است.



۵. محاسبه‌ی بردار w : مطابق نتیجه‌ی به دست آمده، w را محاسبه می‌کنیم. ابتدا وارون S_W مورد نیاز است:

$$\det(S_W) = (20/3)(5/2) - (-4)(-4) = 100/6 - 16 = 50/3 - 48/3 = 2/3 \quad (70)$$

$$S_W^{-1} = \frac{1}{2/3} \begin{bmatrix} 5/2 & 4 \\ 4 & 20/3 \end{bmatrix} = \frac{3}{2} \begin{bmatrix} 5/2 & 4 \\ 4 & 20/3 \end{bmatrix} = \begin{bmatrix} 15/4 & 6 \\ 6 & 10 \end{bmatrix} \quad (71)$$

اکنون w را (تا یک ضریب ثابت) محاسبه می‌کنیم:

$$w = S_W^{-1}(\mu_1 - \mu_2) = \begin{bmatrix} 15/4 & 6 \\ 6 & 10 \end{bmatrix} \begin{bmatrix} 7/3 \\ 5/2 \end{bmatrix} \quad (72)$$

$$w = \begin{bmatrix} (15/4)(7/3) + (6)(5/2) \\ (6)(7/3) + (10)(5/2) \end{bmatrix} = \begin{bmatrix} 35/4 + 15 \\ 14 + 25 \end{bmatrix} = \begin{bmatrix} 95/4 \\ 39 \end{bmatrix} \quad (73)$$

از آنجایی که w یک جهت است، می‌توانیم آن را در ۴ ضرب کنیم تا کسر از بین برود:

$$w \propto \begin{bmatrix} 95 \\ 156 \end{bmatrix} \quad (74)$$

۶. محاسبه‌ی بایاس (b): خط تصمیم به صورت $w^T x + b = 0$ خواهد بود. آستانه‌ی تصمیم ($b = w_0 = -b$) معمولاً در میانگین میانگین‌های تصویرشده‌ی دو کلاس قرار داده می‌شود:

$$w_0 = \frac{1}{2}(w^T \mu_1 + w^T \mu_2) = \frac{1}{2}w^T(\mu_1 + \mu_2) \quad (75)$$

ابتدا $\mu_1 + \mu_2$ را محاسبه می‌کنیم:

$$\mu_1 + \mu_2 = \begin{bmatrix} 4/3 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -3/2 \end{bmatrix} = \begin{bmatrix} 1/3 \\ -1/2 \end{bmatrix} \quad (76)$$

با استفاده از $w = (95, 156)^T$

$$w_0 = \frac{1}{2} \begin{bmatrix} 95 & 156 \end{bmatrix} \begin{bmatrix} 1/3 \\ -1/2 \end{bmatrix} = \frac{1}{2} \left(\frac{95}{3} - \frac{156}{2} \right) = \frac{1}{2} \left(\frac{95}{3} - 78 \right) \quad (77)$$

$$w_0 = \frac{1}{2} \left(\frac{95 - 234}{3} \right) = \frac{1}{2} \left(\frac{-139}{3} \right) = -\frac{139}{6} \quad (78)$$

بایاس b برابر با $-w_0$ است:

$$b = \frac{139}{6} \approx 23.167 \quad (79)$$

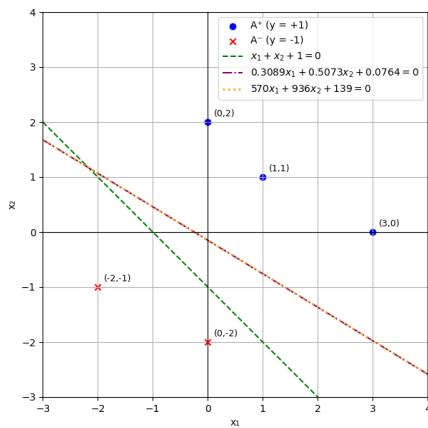
۷. معادله‌ی خط نهایی: در نهایت، معادله‌ی خط جداساز به دست آمده در این قسمت به شکل $w^T x + b = 0$ است. با جایگذاری مقادیر w و b :

$$\begin{bmatrix} 95 & 156 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{139}{6} = 0 \quad (80)$$

$$95x_1 + 156x_2 + \frac{139}{6} = 0 \quad (81)$$

برای حذف مخرج کسر، می‌توان کل معادله را در ۶ ضرب کرد:

$$570x_1 + 936x_2 + 139 = 0 \quad (82)$$



شکل ۴: داده‌های جدا شده توسط خط پیدا شده توسط الگوریتم فیشر (LDA).

۴.۲.۱ مقایسه هر سه روش و بررسی نتایج

برای مقایسه‌ی بصری، معادلات خطی به دست آمده از هر سه روش را بر حسب x_2 بازنویسی می‌کنیم:

• الف (پرسپترون):

$$x_1 + x_2 + 1 = 0 \implies x_2 = -x_1 - 1 \quad (83)$$

• ب (کمترین مربعات):

$$0.3089x_1 + 0.5073x_2 + 0.0764 = 0 \implies x_2 = -\frac{0.3089}{0.5073}x_1 - \frac{0.0764}{0.5073} \quad (84)$$

$$x_2 \approx -0.6089x_1 - 0.1506 \quad (85)$$



• پ (فیشر - LDA)

$$95x_1 + 156x_2 + \frac{139}{6} = 0 \implies x_2 = -\frac{95}{156}x_1 - \frac{139}{6 \times 156} \quad (86)$$

$$x_2 \approx -0.6090x_1 - 0.1485 \quad (87)$$

همان‌طور که مشاهده می‌شود، خطوط به دست آمده از روش فیشر (LDA) و روش کمترین مربعات (Least Square) تقریباً یکسان هستند و تفاوت بسیار جزئی دارند؛ اما خط به دست آمده از روش پرسپترون به طور قابل توجهی متفاوت است.

کدام روش پیشنهاد می‌شود و بهتر است؟

الف: پرسپترون (Perceptron)

- نقطه ضعف: راه حل پرسپترون منحصر به فرد نیست. این راه حل به نقطه شروع $w^{(0)}$ و ترتیب ارائه‌ی داده‌ها بستگی دارد.
- «قضیه‌ی همگرایی پرسپترون» (Perceptron Convergence Theorem) فقط تضمین می‌کند که اگر داده‌ها خطی جداپذیر باشند، الگوریتم یک راه حل پیدا خواهد کرد، اما تضمین نمی‌کند که این راه حل «خوب» باشد (مثلاً با حاشیه‌ی اطمینان بالا).
- در مثال ما، خط $0 = x_1 + x_2 + 1$ بسیار به نقاط کلاس منفی (مانند $(-1, -2)$ و $(0, -2)$) نزدیک است و حاشیه‌ی امن (Margin) خوبی ندارد.

ب: کمترین مربعات (Least Square)

- نقطه ضعف: این روش ذاتاً یک روش رگرسیون (Regression) است و سعی دارد $x' \cdot w'$ را به مقادیر هدف $(+1 \text{ و } -1)$ نزدیک کند.
- این روش به نقاط پرت (Outliers) بسیار حساس است. اگر یک نقطه از کلاس مثبت، بسیار دور از بقیه نقاط آن کلاس بود (مثلاً $(10, 10)$)، این نقطه به تنهایی خط جداکننده را به سمت خود «می‌کشید» تا خطای رگرسیون را کم کند، حتی اگر این کار به قیمت بدتر شدن مرز طبقه‌بندی تمام شود.

پ: فیشر (LDA)

- نقطه قوت: این روش به طور خاص برای جداسازی کلاس‌ها طراحی شده است. LDA به دنبال جهتی است که نسبت پراکنده‌ی بین-کلاسی به پراکنده‌ی درون-کلاسی را بیشینه کند.
- این یک معیار آماری قوی است و معمولاً (به ویژه در صورت نرمال بودن توزیع داده‌های هر کلاس) یک مرز جداکننده‌ی بسیار خوب و مستحکم (Robust) ایجاد می‌کند.



۳.۱ پاسخ پرسش سه

اگر تحلیل مؤلفه‌های اصلی (PCA) را مستقیماً (بدون هیچ پیش‌پردازشی) روی این داده‌ها اعمال کنیم، ویژگی x_1 به طور کامل بر مؤلفه‌های اصلی غالب خواهد شد.

هدف PCA پیدا کردن یک سری محورهای مختصات جدید (به نام مؤلفه‌های اصلی یا Principal Components) است به طوری که واریانس داده‌ها در امتداد این محورها به ترتیب بیشینه شود. PCA این کار را با تجزیه‌ی مقادیر ویژه (Eigenvalue Decomposition) داده‌ها انجام می‌دهد. روی ماتریس کوواریانس (Covariance Matrix) داده‌ها انجام می‌دهد. برای داده‌های دو بعدی ما، ماتریس کوواریانس (Σ) به این شکل است:

$$\Sigma = \begin{pmatrix} \text{Var}(x_1) & \text{Cov}(x_1, x_2) \\ \text{Cov}(x_1, x_2) & \text{Var}(x_2) \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix} \quad (۸۸)$$

از آنجایی که x_1 در بازه‌ی ۰ تا ۱۰۰۰ تغییر می‌کند، واریانس آن ($\sigma_1^2 = \text{Var}(x_1)$) بسیار بزرگ خواهد بود. از طرف دیگر، x_2 در بازه‌ی ۰ تا ۱ تغییر می‌کند. واریانس آن ($\sigma_2^2 = \text{Var}(x_2)$) بسیار کوچک خواهد بود. ($\sigma_1^2 \gg \sigma_2^2$).

بررسی کوواریانس: حتی اگر x_1 و x_2 همبستگی کامل داشته باشند، کوواریانس آن‌ها (σ_{12}) توسط واریانس آنها محدود می‌شود و در مقایسه با σ_1^2 ناچیز خواهد بود. کوواریانس توسط واریانس‌ها محدود می‌شود. ما می‌دانیم که ضریب همبستگی (ρ) همیشه بین -۱ و ۱ است:

$$|\rho| = \left| \frac{\sigma_{12}}{\sqrt{\sigma_1^2 \sigma_2^2}} \right| = \left| \frac{\sigma_{12}}{\sigma_1 \sigma_2} \right| \leq 1 \quad (۸۹)$$

بنابراین:

$$|\sigma_{12}| \leq \sigma_1 \sigma_2 \quad (۹۰)$$

حالا باید σ_1^2 را با $|\sigma_{12}| / \sigma_1$ مقایسه کنیم: چون $\sigma_2 \gg \sigma_1$ (و هر دو مثبت هستند)، می‌توانیم نتیجه بگیریم که $\sigma_1 \sigma_2 \gg \sigma_1^2$. و چون $|\sigma_{12}| \leq \sigma_1 \sigma_2$ پس قطعاً:

$$\sigma_1^2 \gg |\sigma_{12}| \quad (۹۱)$$

تحلیل مقادیر ویژه (Eigenvalues): مقادیر ویژه (λ)، واریانس‌های توضیح داده شده توسط هر مؤلفه اصلی هستند. آن‌ها از طریق «معادله مشخصه» (Characteristic Equation) به دست می‌آیند:

$$\det(\Sigma - \lambda I) = 0 \quad (۹۲)$$

$$\det \begin{pmatrix} \sigma_1^2 - \lambda & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 - \lambda \end{pmatrix} = 0 \quad (۹۳)$$

$$(\sigma_1^2 - \lambda)(\sigma_2^2 - \lambda) - \sigma_{12}^2 = 0 \quad (۹۴)$$

$$\lambda^2 - (\sigma_1^2 + \sigma_2^2)\lambda + (\sigma_1^2 \sigma_2^2 - \sigma_{12}^2) = 0 \quad (۹۵)$$

حالا از فرض $\sigma_2^2 \gg \sigma_1^2$ و $|\sigma_{12}| \ll \sigma_1^2$ استفاده می‌کنیم:



تحلیل λ_1 (بزرگترین مقدار ویژه): از معادله Trace (اثر ماتریس):

$$\lambda_1 + \lambda_2 = \text{tr}(\Sigma) = \sigma_1^2 + \sigma_2^2 \quad (96)$$

از آنجایی که $\sigma_2^2 \gg \sigma_1^2$, مجموع تقریباً برابر σ_1^2 است:

$$\lambda_1 + \lambda_2 \approx \sigma_1^2 \quad (97)$$

از آنجایی که λ_1 بزرگترین مقدار ویژه است ($\lambda_1 \geq \lambda_2 \geq 0$), بخش عمده‌ی این مجموع باید متعلق به λ_1 باشد. پس می‌توانیم با تقریب خوبی بگوییم:

$$\lambda_1 \approx \sigma_1^2 \quad (98)$$

تحلیل λ_2 (کوچکترین مقدار ویژه): از معادله دترمینان استفاده می‌کنیم: $\lambda_1 \lambda_2 = \det(\Sigma) = \sigma_1^2 \sigma_2^2 - \sigma_{12}^2$. حالا که $\lambda_1 \approx \sigma_1^2$. را داریم، آن را جایگزین می‌کنیم:

$$(\approx \sigma_1^2) \lambda_2 \approx \sigma_1^2 \sigma_2^2 - \sigma_{12}^2 \quad (99)$$

با تقسیم طرفین بر σ_1^2 :

$$\lambda_2 \approx \sigma_2^2 - \frac{\sigma_{12}^2}{\sigma_1^2} \quad (100)$$

از آنجایی که σ_2^2 خود عددی بسیار کوچک بود و $\frac{\sigma_{12}^2}{\sigma_1^2} \gg \sigma_1^2$ (به دلیل $|\sigma_{12}| \ll \sigma_1^2$) عددی بسیار کوچک است، λ_2 یک عدد بسیار کوچک خواهد بود.

این اثبات می‌کند که اولین مؤلفه اصلی (PCA) تقریباً تمام واریانس کل سیستم را به تنها یک توضیح خواهد داد.

تحلیل بردارهای ویژه (Eigenvectors): بردارهای ویژه، جهت‌های مؤلفه‌های اصلی را نشان می‌دهند. بردار ویژه v مربوط به λ از این معادله به دست می‌آید:

$$(\Sigma - \lambda I)v = 0 \quad (101)$$

$$\begin{pmatrix} \sigma_1^2 - \lambda & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 - \lambda \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (102)$$

این معادل دو معادله است. ما فقط به یکی از آنها نیاز داریم (چون به هم وابسته‌اند). از معادله دوم استفاده می‌کنیم:

$$\sigma_{12}v_1 + (\sigma_2^2 - \lambda)v_2 = 0 \quad (103)$$



محاسبه $v^{(1)}$ (بردار ویژه PC1، مربوط به $\lambda_1 \approx \sigma_1^2$): در معادلهٔ بالا $\lambda = \lambda_1 \approx \sigma_1^2$ را جایگزین می‌کنیم:

$$\sigma_{12}v_1 + (\sigma_2^2 - \sigma_1^2)v_2 \approx 0 \quad (104)$$

می‌خواهیم نسبت $\frac{v_2}{v_1}$ را پیدا کنیم:

$$(\sigma_2^2 - \sigma_1^2)v_2 \approx -\sigma_{12}v_1 \quad (105)$$

$$\frac{v_2}{v_1} \approx \frac{-\sigma_{12}}{\sigma_2^2 - \sigma_1^2} = \frac{\sigma_{12}}{\sigma_1^2 - \sigma_2^2} \quad (106)$$

با استفاده از فرض $\sigma_2^2 \gg \sigma_1^2$ ، مخرج کسر تقریباً برابر σ_1^2 است:

$$\frac{v_2}{v_1} \approx \frac{\sigma_{12}}{\sigma_1^2} \quad (107)$$

با توجه به اینکه $|\sigma_{12}| \gg \sigma_1^2$ ، این نسبت (آن را ϵ می‌نامیم) بسیار کوچک و نزدیک به صفر است ($\epsilon \approx 0$). پس $\epsilon v_1 \approx v_2$. بردار ویژه $v^{(1)}$ متناسب است با $\begin{pmatrix} 1 \\ \epsilon \end{pmatrix}$ یا ساده‌تر $\begin{pmatrix} v_1 \\ \epsilon v_1 \end{pmatrix}$.

$$v^{(1)} = \frac{1}{\sqrt{1^2 + \epsilon^2}} \begin{pmatrix} 1 \\ \epsilon \end{pmatrix} \quad (108)$$

چون $0 \approx \epsilon$ ، مخرج تقریباً 1 است.

$$v^{(1)} \approx \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad (109)$$

نتیجه (PC1): اولین مؤلفه اصلی (PC1) بردار $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ است که دقیقاً همان محور x_1 است.

محاسبه $v^{(2)}$ (بردار ویژه PC2، مربوط به λ_2): در یک ماتریس متقارن (مانند ماتریس کوواریانس)، بردارهای ویژه بر هم عمود هستند. بردار عمود بر $v^{(1)}$ باید متناسب با $\begin{pmatrix} -\epsilon \\ 1 \end{pmatrix}$ باشد. (زیرا $v^{(1)} \cdot v^{(2)} \approx (1)(-\epsilon) + (\epsilon)(1) = 0$). با نرمال‌سازی و با توجه به اینکه $0 \approx \epsilon$:

$$v^{(2)} \approx \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (110)$$

نتیجه (PC2): دومین مؤلفه اصلی (PC2) بردار $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ است که دقیقاً همان محور x_2 است.



نتیجه‌گیری نهایی

اثبات ریاضی نشان داد که:

۱. مقدار ویژه اول (λ_1) تقریباً برابر با کل واریانس x_1 (σ_1^2) است و مقدار ویژه دوم (λ_2) بسیار ناچیز است.
۲. بردار ویژه اول (PC1) تقریباً بر محور x_1 منطبق است.
۳. بردار ویژه دوم (PC2) تقریباً بر محور x_2 منطبق است.

این به وضوح یک نتیجه‌گیری اشتباه است که صرفاً به دلیل تفاوت واحدها یا مقیاس‌های را خ داده است، نه به دلیل اهمیت واقعی ویژگی‌ها. PCA فریب خورده و فکر می‌کند ویژگی x_1 تنها ویژگی مهم است، صرفاً به این دلیل که واریانس عددی آن بزرگتر است.

۲ سوالات پیاده‌سازی

۱.۲ پرسش چهارم

۱.۱.۲ بخش اول: تحلیل و کاوش داده‌ها (EDA)

در این بخش ابتدا از ما خواسته شده است تا فایل داده teleCust1000t.csv را با استفاده از کتابخانه pandas بخوانیم و از توابع df.info() و df.describe() استفاده کنیم.

```
 1 import pandas as pd
 2 file_path = '/content/teleCust1000t.csv'
 3 df = pd.read_csv("/content/teleCust1000t.csv", sep=",")
 4 df.head()
 5 df.info()
 6 df.head()
```

۱: بارگذاری داده‌ها و نمایش اطلاعات اولیه Code

خروجی‌های مربوط به این بخش از کد به صورت زیر خواهند بود:



<class 'pandas.core.frame.DataFrame'>															
RangeIndex: 1000 entries, 0 to 999															
Data columns (total 12 columns):															
#	Column	Non-Null Count	Dtype												
0	region	1000	non-null	int64											
1	tenure	1000	non-null	int64											
2	age	1000	non-null	int64											
3	marital	1000	non-null	int64											
4	address	1000	non-null	int64											
5	income	1000	non-null	float64											
6	ed	1000	non-null	int64											
7	employ	1000	non-null	int64											
8	retire	1000	non-null	float64											
9	gender	1000	non-null	int64											
10	reside	1000	non-null	int64											
11	custcat	1000	non-null	int64											
dtypes: float64(2), int64(10)															
memory usage: 93.9 KB															

region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
0	2	3	13	44	1	9	64.0	4	5	0.0	1
1	3	11	33	1	7	136.0	5	5	0.0	0	6
2	3	68	52	1	24	116.0	1	29	0.0	1	2
3	2	33	33	0	12	33.0	2	0	0.0	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4

df.describe() :

df.info()

df.head()

همانطور که مشاهده می‌شود هیچ داده NaN‌ای وجود ندارد بنابراین نیازی به استفاده از این روش‌های مطرح شده برای جایگزین کردن داده‌های گمشده نیست.

در قسمت بعد از ما سوال شده است که نوع ویژگی‌ها (عددی یا طبقه‌ای) را مشخص نماییم و تفاوت آنها را توضیح دهیم که جواب آن بدین صورت است که: ویژگی‌های عددی، ویژگی‌هایی هستند که با یک عدد بیان می‌شوند که می‌توانند از نوع پیوسته یا گسسته باشند. مثلاً ویژگی‌هایی مثل سن، درآمد، مقدار سال سکونت مشتری در آدرس فعلی، مدت اشتغال در شغل فعلی، مدت همکاری مشتری با شرکت و تعداد افراد ساکن در خانه از نوع ویژگی‌های عددی می‌باشند. ویژگی‌هایی که از نوع اسم، از نوع ترتیب و یا از نوع درست با غلط می‌باشند از جنس ویژگی‌های طبقه‌ای هستند. این نوع ویژگی‌ها عبارتند از: منطقه جغرافیایی مشتری، وضعیت تأهل مشتری، سطح تحصیلات، بازنیسته بودن یا نبودن، جنسیت مشتری.

باید توجه داشت که در این دیتابست نوع داده مربوط به ویژگی‌های طبقه‌ای به عدد تبدیل شده است و لازم نیست که ما آنها را به عدد تبدیل نماییم.

در سوال بعدی از ما خواسته شده است تا با استفاده از کتابخانه‌های seaborn و plotly Heatmap

همبستگی ویژگی‌ها را رسم و ویژگی‌هایی که بیشترین همبستگی را با متغیر هدف دارند مشخص نماییم. بدین منظور کد زیر زده شده است:

```

1 import pandas as pd
2 import seaborn as sns
3 import plotly.express as px
4 import matplotlib.pyplot as plt
5 numeric_df = df.select_dtypes(include=['int64', 'float64'])
6 corr_matrix = numeric_df.corr()
7 plt.figure(figsize=(10, 8))
8 sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
9 plt.title('Heatmap of Feature Correlations With Seaborn Library', fontsize=14)
10 plt.tight_layout()
11 plt.savefig('/content/HeatmapofFeatureCorrelationsWithSeabornLibrary.png', dpi=300,
bbox_inches='tight')
```



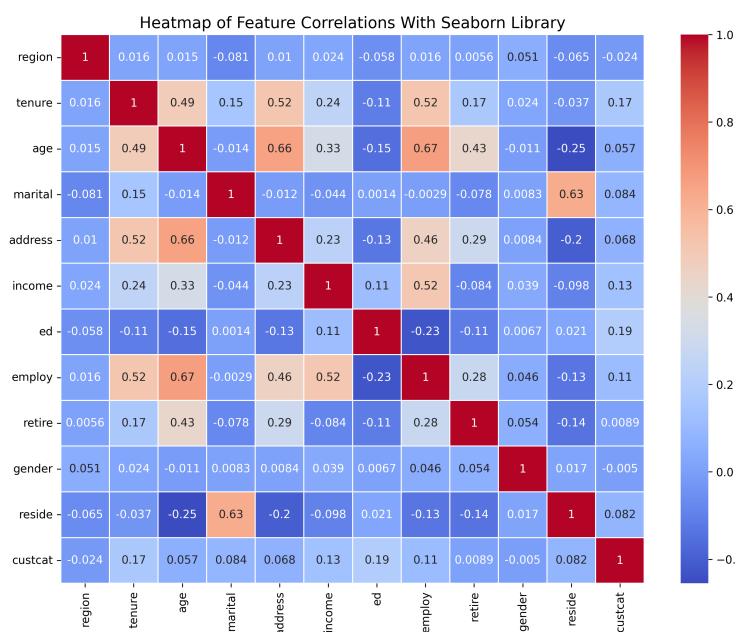
```

12 plt.show()
13 numeric_df = df.select_dtypes(include=['int64', 'float64'])
14 corr_matrix = numeric_df.corr()
15 plt.figure(figsize=(10, 8))
16 sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
17 plt.title('Heatmap of Feature Correlations With Seaborn Library', fontsize=14)
18 plt.tight_layout() #
19 plt.savefig('/content/HeatmapofFeatureCorrelationsWithSeabornLibrary.png', dpi=300,
20 bbox_inches='tight')
21 plt.show()

```

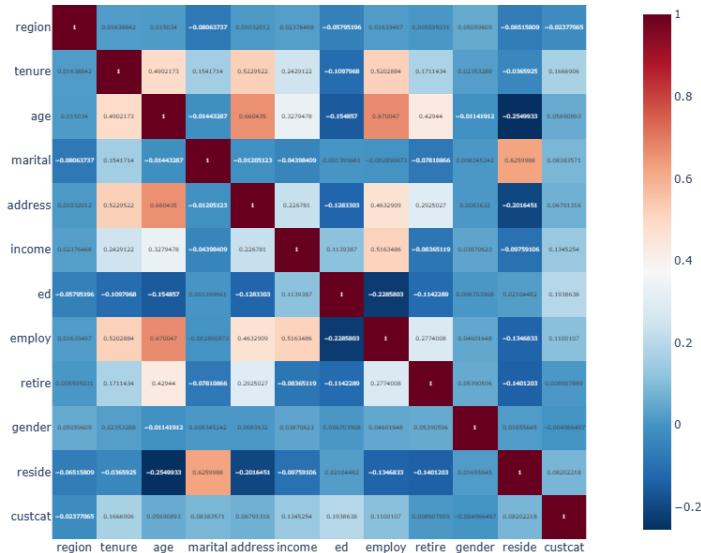
۲: کد مربوط به ترسیم نمودار همبستگی ویژگی‌ها

خروجی‌های این کد به صورت شکل زیر می‌باشد:



شکل ۸: نقشه حرارتی همبستگی ویژگی‌ها (seaborn)

Interactive Correlation Heatmap With Plotly Library



شکل ۹: نقشه حرارتی همبستگی ویژگی‌ها(pltly)

همانطور که مشاهده می‌شود با توجه به نمودار همبستگی گرفته شده، ویژگی‌هایی مثل tenure و income به ترتیب بیشترین همبستگی را با متغیر هدف (custcat) دارند.

برای بخش بعدی نمودارهای pairplot برای ویژگی‌هایی ترسیم شده است که بیشترین تفکیک را بر روی خروجی ایجاد می‌کنند. هنگامیکه برای ویژگی‌هایی به غیر از ویژگی‌های انتخاب شده این نمودارها را ترسیم می‌نماییم، مشاهده می‌نماییم که خیلی کلاس‌ها به داخل یکدیگر فشرده شده‌اند. برای این بخش نیز کد زیر زده شده است:

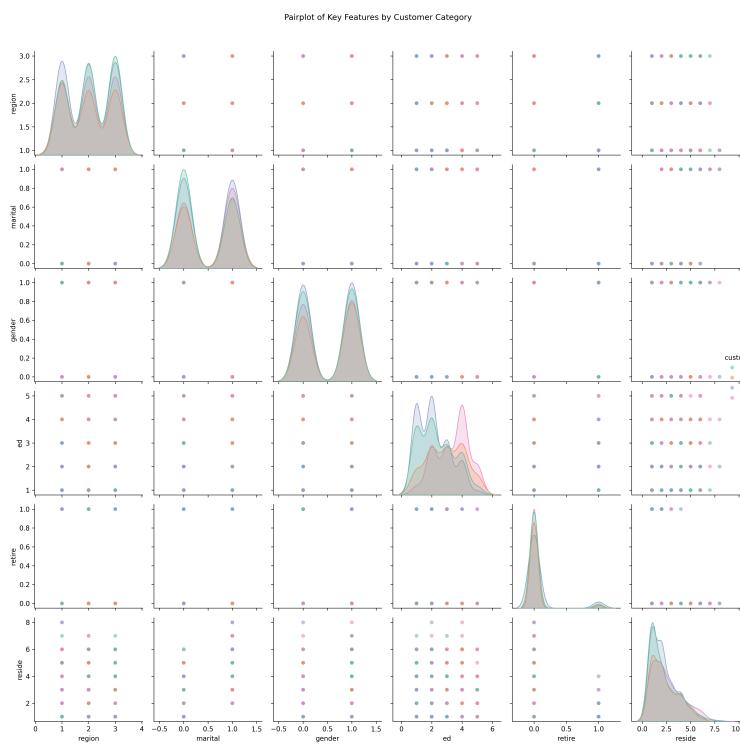
```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 features = ['region', 'marital', 'gender', 'ed', 'retire', 'reside', 'custcat']
4 g = sns.pairplot(df[features], hue='custcat', palette='Set2', diag_kind='kde', plot_kws={'alpha':0.6})
5 g.fig.suptitle('Pairplot of Key Features by Customer Category', y=1.02)
6 plt.tight_layout()
7 g.savefig('/content/pairplot_custcat.png', dpi=300, bbox_inches='tight')
8 plt.show()

```

pairplot کد مربوط به ترسیم نمودار

خروجی این کد شکل زیر می‌باشد:



شکل ۱۰: نمودار Pairplot

در سوال بعدی از ما خواسته شده است تا برای دو ویژگی عددی مهم نمودار Hexbin ترسیم نماییم و ارتباطش را با خروجی توضیح دهیم. در واقع ویژگی‌هایی که بیشترین تفکیک را روی کلاس‌های ما ایجاد می‌کنند بیشتر ویژگی‌های طبقه‌ای هستند ولی از آن جهت که در این سوال از ما خواسته شده تا دو ویژگی عددی مهم را با این نمودار بررسی نماییم سراغ دو ویژگی درآمد و مقدار ساعت اشتغال رفته‌ی چون مقدار همبستگی بیشتری با متغیر هدف خروجی داشتند. کد زیر بدین منظور زده شده است:

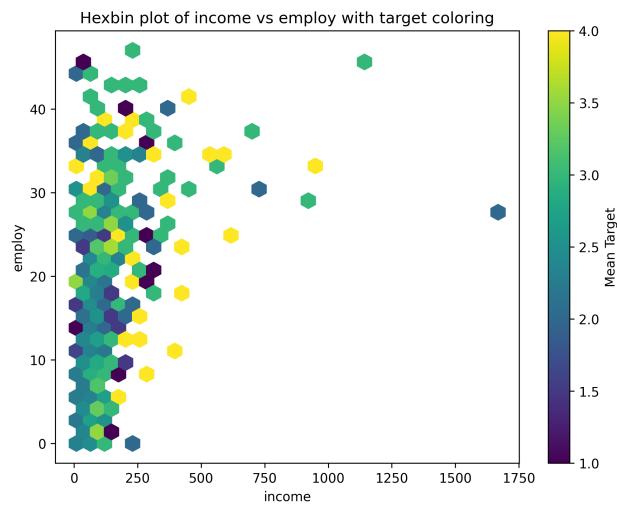
```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 x = df['income']
4 y = df['employ']
5 c = df['custcat']
6 plt.figure(figsize=(8,6))
7 hb = plt.hexbin(x, y, gridsize=30, cmap='viridis', C=c, reduce_C_function=np.mean)
8 plt.colorbar(hb, label='Mean Target')
9 plt.xlabel('income')
10 plt.ylabel('employ')
11 plt.title('Hexbin plot of income vs employ with target coloring')
12 plt.savefig('/content/Hexbinplot.png', dpi=300, bbox_inches='tight')
13 plt.show()

```

کد مربوط به ترسیم نمودار Hexbin

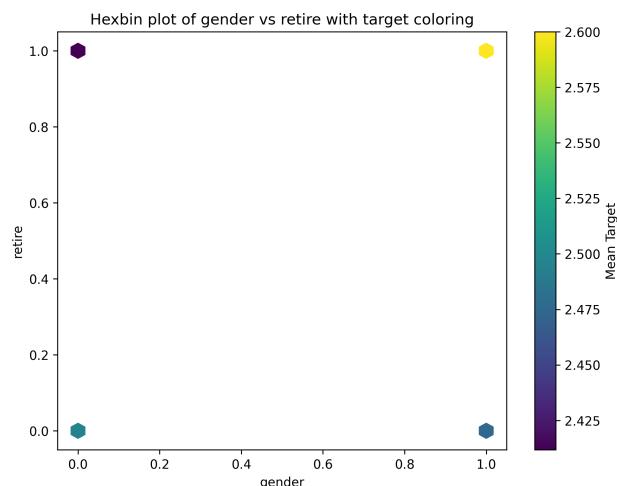
خروجی این کد به صورت زیر است:



شکل :۱۱ Hexbinplot with Numerical features

همانطور که مشاهده می‌شود با اینکه این دو ویژگی عددی همبستگی بیشتری با متغیر هدف دارند، این همبستگی باعث نشده است تا بتوانیم کلاس‌ها را به خوبی از یکدیگر تفکیک کنیم و عملاً مشاهده می‌شود که با افزایش مقدار هر کدام از ویژگی‌ها کلاس‌ها تقریباً در یکجا متراکم شده‌اند.

خروجی مربوط به استفاده از دو ویژگی طبقه‌ای که مقداری تفکیک‌پذیری بهتری دارد به صورت زیر است:



شکل :۱۲ Hexbinplot with Categorical features

در بخش بعدی با استفاده از دوتابع countplot و plotpie توزیع کلاس‌ها را نمایش می‌دهیم. کد این بخش به صورت زیر می‌باشد:

```

1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4 target_col = 'custcat'
5 plt.figure(figsize=(6,4))

```



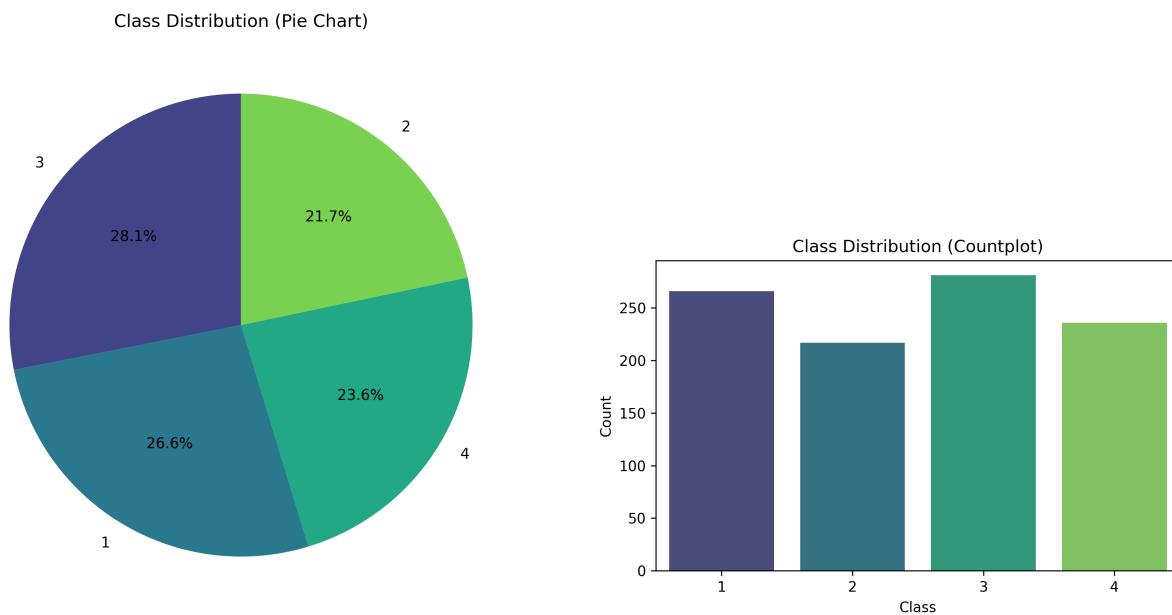
```

 9  sns.countplot(x=target_col, data=df, palette='viridis')
10 plt.title('Class Distribution (Countplot)')
11 plt.xlabel('Class')
12 plt.ylabel('Count')
13 plt.tight_layout()
14 plt.savefig('/content/countplot.png', dpi=300, bbox_inches='tight')
15 plt.show()
16 class_counts = df[target_col].value_counts()
17 plt.figure(figsize=(6,6))
18 plt.pie(
19     class_counts,
20     labels=class_counts.index,
21     autopct='%.1f%%',
22     startangle=90,
23     colors=sns.color_palette('viridis', len(class_counts))
24 )
25 plt.title('Class Distribution (Pie Chart)')
26 plt.tight_layout()
27 plt.savefig('/content/piechart.png', dpi=300, bbox_inches='tight')
28 plt.show()

```

Code ۵: کد مربوط به ترسیم توزیع کلاس‌ها

خروجی‌های مربوط به کد بالا به صورت زیر می‌باشد:



شکل ۱۴: نمودار دایره‌ای

شکل ۱۳: نمودار میله‌ای



همانطور که مشاهده می‌شود تقریباً تعادل بین کلاس‌ها برقرار است و تعداد داده‌های هر کلاس تقریباً با کلاس دیگر برابر است. این تعادل از آن جهت مهم است که مدلمان در هنگام آموزش بر روی کلاسی که داده بیشتری از آن موجود است، بایاس نشود.

۲.۱.۲ بخش دوم: پیش‌پردازش داده‌ها

در اولین سوال از این بخش از ما خواسته شده است تا ویژگی‌های طبقه‌ای را به داده عددی تبدیل نماییم. همانطور که قبلاً نیز اشاره شده است در این دیتاست ویژگی‌های طبقه‌ای به داده عددی تبدیل شده‌اند و نیازی نیست که با این کار آنها را به داده عددی تبدیل نماییم. در بخش بعد می‌بایست یک نرمالسازی بر روی ویژگی‌ها انجام دهیم. هدف از نرمالسازی این است که تمامی مقادیر مربوط به هر ویژگی در یک بازه قرار بگیرند. قرار گرفتن در یک مقیاس مشابه باعث می‌شود تا مدل یادگیری ماشین پردازش درستی را بر روی داده‌ها انجام دهد. در بسیاری از الگوریتم‌های یادگیری ماشین (مثل شبکه‌های عصبی، رگرسیون خطی و لجستیک) مقدار عددی ویژگی‌ها روی عملکرد مدل تأثیر مستقیم دارد. اگر مثلاً یکی از ویژگی‌ها بین $0 \text{ تا } 1$ تغییر کند ولی دیگری بین $0 \text{ تا } 10,000$ باشد، مدل به ویژگی بزرگ‌تر وزن بیشتری می‌دهد، حتی اگر از نظر معنا اهمیتش کمتر باشد. کد زیر در این بخش زده شده است:

```

1 import pandas as pd
2 from sklearn.preprocessing import MinMaxScaler
3 target_col = 'custcat'
4 numeric_features = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
5 numeric_features = [col for col in numeric_features if col != target_col]
6 df_normalized = df.copy()
7 scaler = MinMaxScaler()
8 df_normalized[numeric_features] = scaler.fit_transform(df[numeric_features])
9 df_normalized.head()

```

۶: کد مربوط به نرمالسازی Code

با توجه به نمودارهایی که در بخش‌های قبل ترسیم کردیم متوجه شدیم که یکسری از ویژگی‌ها بهتر می‌توانند فضای کلاس‌بندی ما را از یکدیگر تفکیک کنند و یکسری از ویژگی‌ها تاثیر چندانی در تفکیک کلاس‌ها از یکدیگر ندارند. بنابراین با توجه به نمودارهای گرفته شده ۶ ویژگی را که بیشتر از همه باعث تفکیک پذیری کلاس‌ها از یکدیگر شده‌اند انتخاب می‌نماییم و بقیه را حذف می‌نماییم. مثلاً در انتخاب ویژگی ed می‌توان گفت که همبستگی بیشتری با متغیر هدف ما داشته است. انتخاب دیگر ویژگی‌ها نیز براساس نمودارهای pairplot و Hexbin انجام شده است.

۳.۱.۲ بخش سوم: انتخاب ویژگی و مدل‌سازی کلاسیک

در این بخش از بین ۶ ویژگی که در بخش قبل به عنوان ویژگی‌های مهم‌تر انتخاب کردیم استفاده می‌نماییم. همانطور که سوال از ما خواسته است با استفاده از LassoRegression و RFE ویژگی‌هایی که اهمیت بیشتری در پیش‌بینی کلاس مورد دارند را به دست می‌آوریم. در زیر کد مربوط به هر بخش زده شده است:

```

1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.linear_model import Lasso, LinearRegression
4 from sklearn.feature_selection import RFE
5 target_col = 'custcat'

```



```

9 selected_features = ['region', 'marital', 'gender', 'ed', 'retire', 'reside']
10 X = df_normalized[selected_features]
11 y = df_normalized[target_col]
12 lasso = Lasso(alpha=0.01)
13 lasso.fit(X, y)
14 lasso_coef = pd.Series(lasso.coef_, index=X.columns)
15 important_features_lasso = lasso_coef[lasso_coef != 0].sort_values(ascending=False)
16 print("Important Features With Lasso Regression:")
17 print(important_features_lasso)
18 estimator = LinearRegression()
19 selector = RFE(estimator, n_features_to_select=3, step=1)
20 selector = selector.fit(X, y)
21 rfe_features = X.columns[selector.support_]
22 print("Important Features With RFE:")
23 print(rfe_features)

```

Extract Important Features :啖 Code

خروجی‌های این کد به صورت زیر می‌باشند:

```

Important Features With Lasso Regression:
ed          0.603171
marital    0.147239
dtype: float64

```

شکل ۱۵: ویژگی‌های مهم استخراج شده از Regression Lasso

```

Important Features With RFE:
Index(['ed', 'retire', 'reside'], dtype='object')

```

شکل ۱۶: ویژگی‌های مهم استخراج شده از RFE

حال از ویژگی‌های به دست آمده استفاده می‌نماییم و یک مدل LogisticRegression با توزیع داده ۹۰ درصد ترین و ۱۰ درصد تست آموزش می‌دهیم. کد زده شده برای آموزش و به دست آوردن دقیق مدل از ویژگی‌های به دست آمده از RFE به صورت زیر است:

```

1 import pandas as pd
2 from sklearn.preprocessing import MinMaxScaler
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
6 X_rfe = X[rfe_features]
7 y = df_normalized['custcat']
8 X_train, X_test, y_train, y_test = train_test_split(
9     X_rfe, y, test_size=0.1, random_state=42, stratify=y
10 )

```



```
11 print(f"Train: {X_train.shape[0]}, Test: {X_test.shape[0]}")  
12 logreg = LogisticRegression(max_iter=1000)  
13 logreg.fit(X_train, y_train)  
14 y_test_pred = logreg.predict(X_test)  
15 y_train_pred = logreg.predict(X_train)  
16 print("\nTest Accuracy:", accuracy_score(y_test, y_test_pred))  
17 print("\nTrain Accuracy:", accuracy_score(y_train, y_train_pred))  
18 print("Test Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))  
19 print("\nClassification Report (Test):\n", classification_report(y_test, y_test_pred))
```

Train model and evaluation (RFE) :Λ Code

مقدار دقیق دست آمده در این قسمت برای داده‌های آموزش برابر با ۳۶ درصد و برای داده‌های آزمون برابر با ۴۵ درصد می‌باشد. این نشان دهنده این است که مدل به خوبی بر روی داده‌های آموزش یاد نگرفته است. البته اینطور انتظار داریم که با یک خط توان این ۴ کلاس را از هم تفکیک کرد چرا که در نموذارهای گرفته شده در اول این سوال کاملاً مشخص بود که داده‌ها به صورت خطی از هم تفکیک‌پذیر نیستند. کد زده شده برای به دست آوردن ماتریس درهم‌ریختگی، نمودار ROC و مقدار AUC به صورت زیر است:

```
1 import matplotlib.pyplot as plt  
2 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc,  
3     roc_auc_score  
4 cm = confusion_matrix(y_test, y_test_pred)  
5 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=logreg.classes_)  
6 disp.plot(cmap=plt.cm.Blues)  
7 plt.title("Confusion Matrix on Test Set")  
8 plt.savefig('/content/confusionRfe.png', dpi=300, bbox_inches='tight')  
9 plt.show()  
10 import pandas as pd  
11 import numpy as np  
12 import matplotlib.pyplot as plt  
13 from sklearn.preprocessing import MinMaxScaler, label_binarize  
14 from sklearn.linear_model import LogisticRegression  
15 from sklearn.model_selection import train_test_split  
16 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,  
17     roc_curve, auc, roc_auc_score  
18 y_test_bin = label_binarize(y_test, classes=logreg.classes_)  
19 n_classes = y_test_bin.shape[1]  
20 y_score = logreg.predict_proba(X_test)  
21 plt.figure(figsize=(8,6))  
22 for i in range(n_classes):  
23     fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_score[:, i])  
24     roc_auc = auc(fpr, tpr)  
25     plt.plot(fpr, tpr, lw=2, label=f'Class {logreg.classes_[i]} (AUC = {roc_auc:.2f})')  
26 plt.plot([0,1], [0,1], color='navy', lw=1, linestyle='--')  
27 plt.xlim([0.0, 1.0])  
28 plt.ylim([0.0, 1.05])  
29 plt.xlabel('False Positive Rate')
```



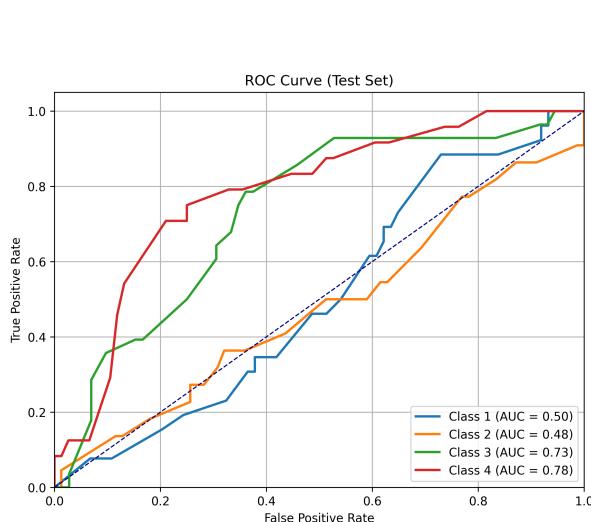
```

۲۸ plt.ylabel('True Positive Rate')
۲۹ plt.title('ROC Curve (Test Set)')
۳۰ plt.legend(loc="lower right")
۳۱ plt.grid("True")
۳۲ plt.savefig('/content/AUCROC.RFE.png', dpi=300, bbox_inches='tight')
۳۳ plt.show()
۳۴ auc_score = roc_auc_score(y_test_bin, y_score, average='macro')
۳۵ print(f"\nMacro-average AUC on Test Set: {auc_score:.3f}")

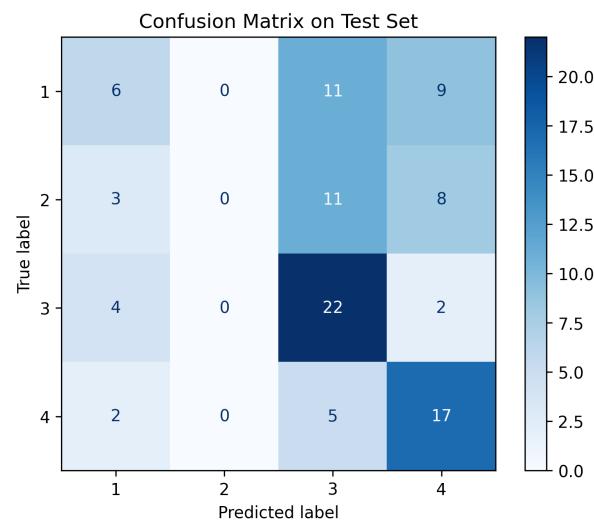
```

Evaluation (RFE) :۹ Code

خروجی‌های مربوط به این کد در ادامه ارسال می‌گردد:



شکل ۱۸: AUC-ROC (RFE)



شکل ۱۷: Confusion Matrix (RFE)

ضرایب مدل به صورت زیر به دست می‌آیند:

◆ Logistic Regression Coefficients by Class:					
	class	ed	retire	reside	bias
0	1	-1.034495	-0.143629	-0.535953	0.649686
1	2	0.741347	-0.355602	0.087600	-0.399004
2	3	-1.483760	0.429354	-0.307495	0.772063
3	4	1.776908	0.069877	0.755847	-1.022745

شکل ۱۹: ضرایب مدل رگرسیون لجستیک برای ویژگی‌های استخراج شده از RFE

همانطور که مشاهده می‌شود ویژگی ed دارای ضرایب بزرگتری برای پیش‌بینی هر کلاس می‌باشد و نشان می‌دهد که تاثیر بیشتری را برای پیش‌بینی مدل ایفا می‌کند. هرچند که اینطور هم انتظار می‌رفت چرا که هم ماتریس همبستگی بین ویژگی‌ها و هم ترتیب اهمیت ویژگی‌های



استخراج شده به وسیله RFE این موضوع را تایید می‌کند. حال تمامی بخش‌های قبل را برای LassoRegression تکرار می‌نماییم. کد زده شده برای آموزش مدل و ارزیابی بر روی آن به صورت زیر است:

```

1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LogisticRegression
3 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
4 X_rfe = df_normalized[['ed', 'marital']]
5 y = df_normalized['custcat']
6 X_train, X_test, y_train, y_test = train_test_split(
7     X_rfe, y, test_size=0.1, random_state=42, stratify=y
8 )
9 print(f"Train: {X_train.shape[0]}, Test: {X_test.shape[0]}")
10 logreg = LogisticRegression(max_iter=1000)
11 logreg.fit(X_train, y_train)
12 y_test_pred = logreg.predict(X_test)
13 y_train_pred = logreg.predict(X_train)
14 print("\nTest Accuracy:", accuracy_score(y_test, y_test_pred))
15 print("\nTrain Accuracy:", accuracy_score(y_train, y_train_pred))
16 print("Test Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
17 print("\nClassification Report (Test):\n", classification_report(y_test, y_test_pred))

```

Train model and evaluation (LassoRegression) :۱۰ Code

مقدار دقت به دست آمده در این قسمت برای داده‌های آموزش برابر با ۴۱ درصد می‌باشد. این نشان دهنده این است که مدل به خوبی بر روی داده‌های آموزش یادگرفته است. البته اینطور انتظار داریم که با یک خط نتوان این ۴ کلاس را از هم تفکیک کرد چرا که در نمودارهای گرفته شده در اول این سوال کاملاً مشخص بود که داده‌ها به صورت خطی از هم تفکیک‌پذیر نیستند. مقدار دقت مدل بر روی داده‌های آزمون در این روش ۴ درصد نسبت به روش قبل افت داشته است. اما برخلاف این در مقادیر AUC و نمودار ROC مقداری بهبود مشاهده می‌شود. کد زده شده برای به دست آوردن ماتریس درهم‌ریختگی، نمودار ROC و مقدار AUC به صورت زیر است:

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from sklearn.preprocessing import label_binarize
4 from sklearn.metrics import (
5     ConfusionMatrixDisplay,
6     roc_curve,
7     auc,
8     roc_auc_score
9 )
10 cm = confusion_matrix(y_test, y_test_pred)
11 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=logreg.classes_)
12 disp.plot(cmap=plt.cm.Blues)
13 plt.title("Confusion Matrix on Test Set")
14 plt.savefig('/content/confusionLasoo.png', dpi=300, bbox_inches='tight')
15 plt.show()
16 y_test_bin = label_binarize(y_test, classes=logreg.classes_)

```



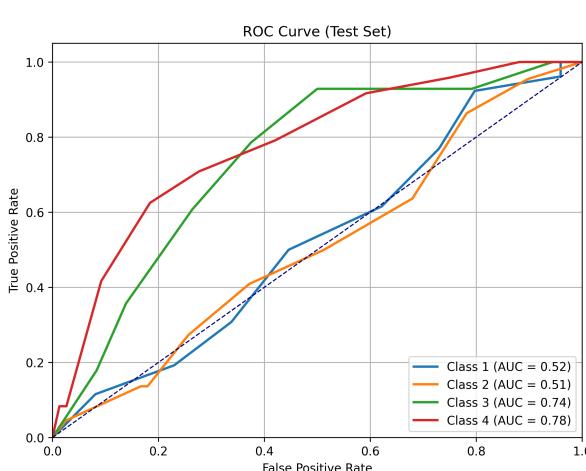
```

 ۱۷ n_classes = y_test_bin.shape[1]
 ۱۸ y_score = logreg.predict_proba(X_test)
 ۱۹ plt.figure(figsize=(8,6))
 ۲۰ for i in range(n_classes):
 ۲۱     fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_score[:, i])
 ۲۲     roc_auc = auc(fpr, tpr)
 ۲۳     plt.plot(fpr, tpr, lw=2, label=f'Class {logreg.classes_[i]} (AUC = {roc_auc:.2f})')
 ۲۴ plt.plot([0,1], [0,1], color='navy', lw=1, linestyle='--')
 ۲۵ plt.xlim([0.0, 1.0])
 ۲۶ plt.ylim([0.0, 1.05])
 ۲۷ plt.xlabel('False Positive Rate')
 ۲۸ plt.ylabel('True Positive Rate')
 ۲۹ plt.title('ROC Curve (Test Set)')
 ۳۰ plt.legend(loc="lower right")
 ۳۱ plt.grid("True")
 ۳۲ plt.savefig('/content/AUCROCLasoo.png', dpi=300, bbox_inches='tight')
 ۳۳ plt.show()
 ۳۴ auc_score = roc_auc_score(y_test_bin, y_score, average='macro')
 ۳۵ print(f"\nMacro-average AUC on Test Set: {auc_score:.3f}")

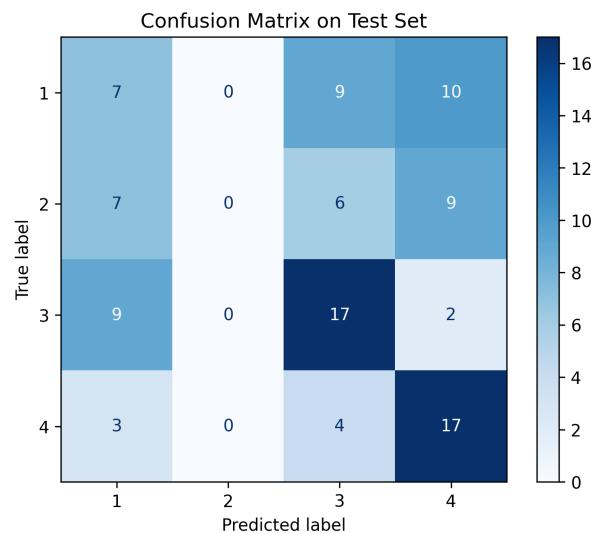
```

Evaluation (LassoRegression) :۱۱ Code

خروجی‌های مربوط به این کد در ادامه ارسال می‌گردد:



شکل ۲۱: AUC-ROC (LassoRegression)



شکل ۲۰: Confusion Matrix (LassoRegression)

ضرایب مدل به صورت زیر به دست می‌آیند:



◆ Logistic Regression Coefficients by Class:				
	class	ed	marital	bias
0	1	-1.041500	-0.325390	0.697060
1	2	0.774520	0.165056	-0.495850
2	3	-1.516567	-0.027211	0.765593
3	4	1.783547	0.187544	-0.966803

شکل ۲۲: ضرایب مدل رگرسیون لجستیک برای ویژگی‌های استخراج شده از LassoRegression

همانطور که مشاهده می‌شود ویژگی ed دارای ضرایب بزرگتری برای پیش‌بینی هرکلاس می‌باشد و نشان می‌دهد که تاثیر بیشتری را برای پیش‌بینی مدل ایفا می‌کند. هرچند که اینطور هم انتظار می‌رفت چرا که هم ماتریس همبستگی بین ویژگی‌ها و هم عدد اهمیت ویژگی‌های استخراج شده به وسیله LassoRegression این موضوع را تایید می‌کند.

۴.۱.۲ بخش چهارم: نمایش ویژگی‌ها با استفاده از کاهش ابعاد

همانطور که سوال از ما خواسته با استفاده از PCA یک کاهش ابعاد بر روی دیتاپی که در اختیار داریم، می‌زنیم. کد آن به صورت زیر است:

```

1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.decomposition import PCA
4 import matplotlib.pyplot as plt
5 features = ['region', 'tenure', 'age', 'marital', 'address',
6             'income', 'ed', 'employ', 'retire', 'gender', 'reside']
7 X = df[features]
8 y = df['custcat']
9 scaler = StandardScaler()
10 X_scaled = scaler.fit_transform(X)
11 pca = PCA(n_components=2)
12 X_pca = pca.fit_transform(X_scaled)
13 pca_df = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
14 pca_df['custcat'] = y.values
15 print("Explained Variance Ratio:", pca.explained_variance_ratio_)
16 print("Total Variance Explained:", sum(pca.explained_variance_ratio_))
17 plt.figure(figsize=(8,6))
18 for label in pca_df['custcat'].unique():
19     subset = pca_df[pca_df['custcat'] == label]
20     plt.scatter(subset['PC1'], subset['PC2'], label=f'Class {label}', alpha=0.6)
21 plt.title('PCA Projection to 2 Components')
22 plt.xlabel('Principal Component 1')
23 plt.ylabel('Principal Component 2')
24 plt.legend()
25 plt.grid(True)

```

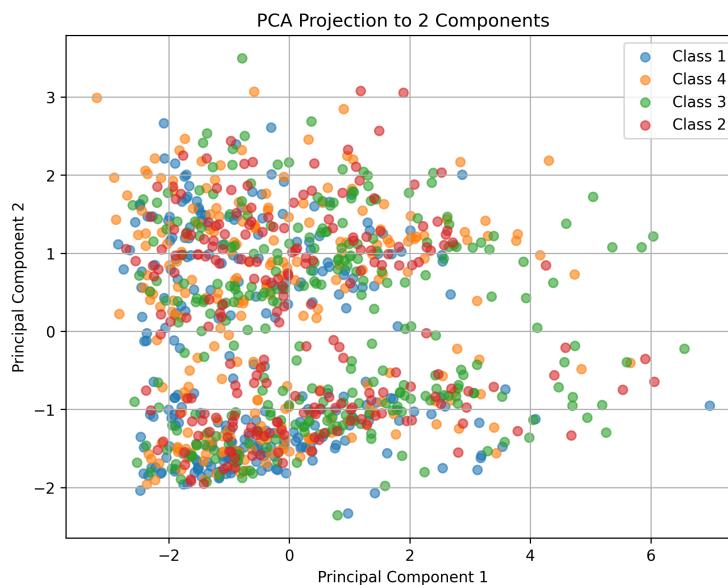
```

۲۶ plt.savefig('/content/PCA.png', dpi=300, bbox_inches='tight')
۲۷ plt.show()

```

PCA Projection to 2 Components :۱۲ Code

توجه شود که در اینجا از استانداردسازی standardscaler استفاده شده است. خروجی مربوط به این کد به صورت زیر است:



شکل ۲۳: PCA Projection to 2 Components

حال در بخش بعد از یک کاهش ابعاد LDA استفاده می‌نماییم. کد زیر بدین منظور زده شده است:

```

۱ import pandas as pd
۲ from sklearn.preprocessing import StandardScaler
۳ from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
۴ import matplotlib.pyplot as plt
۵ features = ['region', 'tenure', 'age', 'marital', 'address',
۶             'income', 'ed', 'employ', 'retire', 'gender', 'reside']
۷ X = df[features]
۸ y = df['custcat']
۹ scaler = StandardScaler()
۱۰ X_scaled = scaler.fit_transform(X)
۱۱ lda = LDA(n_components=2)
۱۲ X_lda = lda.fit_transform(X_scaled, y)
۱۳ lda_df = pd.DataFrame(data=X_lda, columns=['LD1', 'LD2'])
۱۴ lda_df['custcat'] = y.values
۱۵ explained_ratio = lda.explained_variance_ratio_
۱۶ print("Explained Variance Ratio:", explained_ratio)
۱۷ print("Total Variance Explained:", sum(explained_ratio))
۱۸ plt.figure(figsize=(8,6))
۱۹ for label in lda_df['custcat'].unique():

```



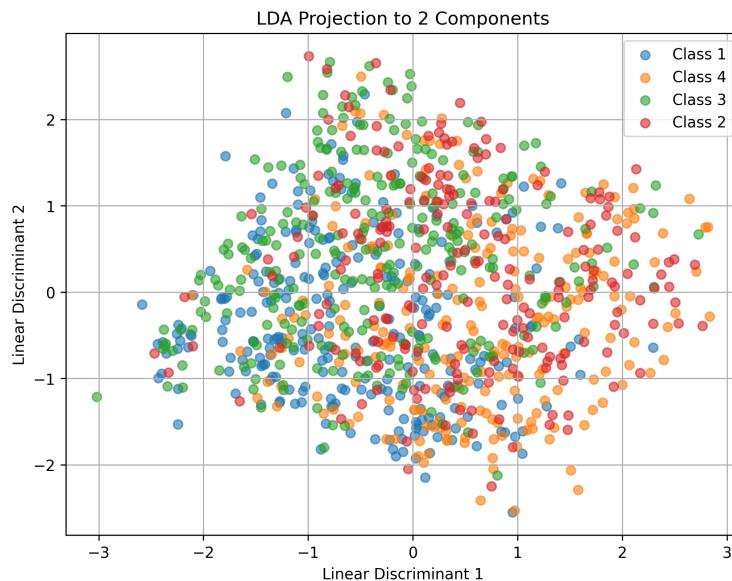
```

20     subset = lda_df[lda_df['custcat'] == label]
21     plt.scatter(subset['LD1'], subset['LD2'], label=f'Class {label}', alpha=0.6)
22     plt.title('LDA Projection to 2 Components')
23     plt.xlabel('Linear Discriminant 1')
24     plt.ylabel('Linear Discriminant 2')
25     plt.legend()
26     plt.grid(True)
27     plt.savefig('/content/LDA.png', dpi=300, bbox_inches='tight')
28     plt.show()

```

LDA Projection to 2 Components :۱۳ Code

در کاهش ابعاد LDA حتما باید توجه داشته باشیم که حداقل مقدار کاهش ابعادی که می‌توانیم داشته باشیم برابر با مینیمم تعداد کلاس‌ها منهای یک و ابعاد ویژگی‌هایمان است. در اینجا مقدار ۲ برای کاهش بعد عدد منطقی است. خروجی این بخش از کد به صورت زیر است:



LDA Projection to 2 Components :۲۴ شکل

حال از MLP به عنوان کاهش بعد استفاده می‌نماییم. کد زیر را برای آموزش مدل استفاده کردیم. ۱۰ درصد دیتارو برای تست و درصد آن را برای آموزش مدل استفاده نمودیم. فرآیند آموزش آنقدر خوب نیست و هرچه قدر هم که مدل را تغییر می‌دهیم و نورون‌ها را کم یا زیاد می‌نماییم و یا لایه‌ها را اضافه یا کم می‌نماییم نمی‌توانیم به دقت بالاتر از ۴۰٪ خودرهای درصد دست پیدا نماییم. در اینجا در آخرین اپک توانتیم به ۴۱ درصد اکیورسی دست پیدا نماییم که خیلی خوب نیست. کد زیر برای این آموزش زده شده است:

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5 from tensorflow.keras.models import Sequential
6 from tensorflow.keras.layers import Dense, Input

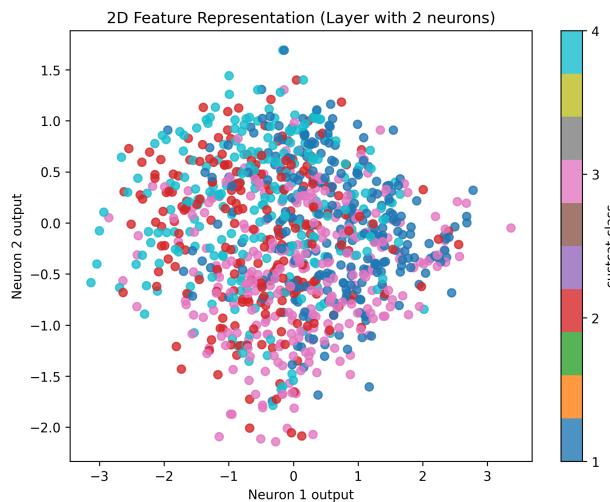
```



```
v  from tensorflow.keras.utils import to_categorical
^  X = df[features]
^  y = df['custcat']
10  scaler = StandardScaler()
11  X_scaled = scaler.fit_transform(X)
12  y_cat = to_categorical(y - 1)
13  X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_cat,
14                                              test_size=0.1, random_state=42)
15  model = Sequential([
16      Input(shape=(X_scaled.shape[1],)),
17      Dense(5),
18      Dense(4),
19      Dense(3),
20      Dense(2),
21      Dense(4, activation='softmax')
22  ])
23  model.compile(optimizer='adam',
24                  loss='categorical_crossentropy',
25                  metrics=['accuracy'])
26  history = model.fit(X_train, y_train,
27                      epochs=100,
28                      batch_size=32,
29                      validation_data=(X_test, y_test),
30                      verbose=1)
31  loss, acc = model.evaluate(X_test, y_test, verbose=0)
32  print(f"Test Accuracy: {acc:.4f}")
```

Training MLP : ۱۴ Code

از استانداردسازی standardscaler استفاده شده است. مدل دارای ۵ نورون در ورودی بعد از آن ۴ نورون، بعد ۳ نورون، بعد ۲ نورون و درنهایت دارای ۴ نورون به همراه یک تابع فعالساز softmax در خروجی است. از تابع بهینه‌ساز Adam استفاده شده است و تابع هزینه نیز categoricalcrossentropy است. مقدار سایز هر batch برابر با ۳۲ است. خروجی به دست آمده از نگاشت دو نورون قبل از لایه آخر که به منظور کاهش بعد استفاده شده است به صورت زیر است:



شکل ۲۵: MLP Projection to 2 Components

همانطور که مشاهده می‌شود هیچ‌کدام از سه روش مورد استفاده شده باعث نشده که کلاس‌هایمان دارای تفکیک‌پذیری بیشتری شوند اما اگر بخوایم مقداری مقایسه نماییم؛ MLP از LDA و PCA بـهتر عمل کرده است. اگر مقداری دقیق شود، یک مقدار بهتر توانسته‌اند دیتای هرکلاس را در یک مکان نزدیک به هم جمع آوری کنند اما باز هم هیچ‌کدام باعث نشده‌اند که دیتای هرکلاس از یکدیگر تفکیک گردد.

۲.۲ پاسخ پرسش پنجم

۱.۲.۲ بخش اول: مطالعه مقاله

صورت سوال:

- مقاله مدل خود را برای چه شهری و چه دادگانی (شامل چه ویژگی‌هایی) انجام داده است؟
- مقاله چه پیش‌پردازش‌هایی روی دادگان خود انجام داده است؟
- مقاله از چه مدل‌هایی استفاده کرده است؟ (صرفاً نام ببرید)

پاسخ:

“Housing Price Prediction via Improved Machine Learning” با عنوان Truong et al. (2020) مطالعه مورد نظر توسط Truong et al. (2020) بر روی داده‌های قیمت مسکن شهر پکن (Beijing) انجام شده است [Truong ۲۰۲۰]. این مجموعه داده با نام “Housing Techniques” در دسترس است و اطلاعات معاملات ملکی از سال‌های ۲۰۰۹ تا ۲۰۱۸ را شامل می‌شود.

۱. شهر و دادگان مورد استفاده

- نسخه خام داده‌ها شامل بیش از ۳۰۰،۰۰۰ رکورد و ۲۶ ویژگی مختلف است که جزئیاتی از معاملات ملک مانند زیربنا، سن‌بنا، موقعیت جغرافیایی، تعداد اتاق‌ها، نوع طبقه و سایر مشخصات ساخت را دربر دارد.



- پس از انجام فرایند پاکسازی و مهندسی ویژگی، داده نهایی دارای ۹۶۲ نمونه و ۱۹ ویژگی شد.
- در ادامه با اجرای استانداردسازی بر روی مقادیر عددی و کدگذاری وان‌هات (One-Hot Encoding) برای مقادیر دسته‌ای، مجموعه داده نهایی ۵۸ ویژگی داشت.

۲. پیش‌پردازش‌های انجام‌شده فرایند آماده‌سازی داده در این مقاله شامل چند مرحله کلیدی است:

۱. حذف داده‌های گمشده: ویژگی‌هایی که بیش از ۵٪ مقادیرشان خالی بود (مانند Day on market) حذف شدند. همچنین هر سطر دارای مقدار گمشده از داده حذف گردید.

۲. مهندسی ویژگی:

- ویژگی constructionTime با age جایگزین شد (سن بنا بر حسب سال = ۲۰۱۹ منهای سال ساخت).
- ویژگی جدیدی به نام distance برای نمایش فاصله ملک از مرکز پکن افزوده شد.
- ویژگی floor به دو زیرویژگی floorType و floorHeight تقسیک گردید.
- تعداد اتاق‌های نشیمن، حمام و آشپزخانه به دلیل ابهام داده حذف شدند و ویژگی livingRoom (که در واقع بیانگر تعداد اتاق‌خواب است) در بازه ۱ تا ۴ محدود شد.

۳. حذف داده‌های پرت: مقادیر پرت با استفاده از قاعده IQR (فاصله بین چارک‌ها) شناسایی و حذف شدند:

$$x < Q_1 - 1.5 \times IQR \quad \text{یا} \quad x > Q_3 + 1.5 \times IQR$$

۴. آماده‌سازی نهایی: داده‌های عددی نرمال‌سازی شدند، داده‌های دسته‌ای وان‌هات گردیدند و داده‌ها با نسبت ۸۰/۲۰ بین مجموعه‌های آموزش و آزمون تقسیم شدند. معیار ارزیابی نیز RMSLE (خطای میانگین مربعات لگاریتمی ریشه‌دار) انتخاب شد.

۳. مدل‌های مورد استفاده نویسنده‌گان مقاله برای پیش‌بینی قیمت مسکن، از ترکیبی از مدل‌های سنتی و پیشرفته استفاده کرده‌اند که عبارت‌اند از:

Random Forest •

XGBoost •

LightGBM •

(LightGBM و XGBoost و RF، Hybrid Regression •

(XGBoost و LightGBM، RF و Stacked Generalization •

در مجموع، این مقاله نشان داده است که ترکیب مدل‌های یادگیری ماشین و اعمال پیش‌پردازش دقیق بر داده‌ها می‌تواند دقیق‌تر باشد. پیش‌بینی قیمت مسکن را به صورت چشم‌گیری افزایش دهد.



۲.۰.۲ بخش دوم: دادگان

صورت سؤال:

- این دادگان چند نمونه و چند ویژگی دارد؟
- نوع داده‌ی هر ویژگی چیست؟
- هر ویژگی چند مقدار منحصر به فرد دارد؟

پاسخ:

در این بخش از مجموعه داده‌ی housing.csv استفاده شده است. هدف از این بخش، شناسایی ساختار کلی داده، نوع متغیرها و تنوع مقادیر هر ستون پیش از انجام مراحل تحلیل و پیش‌پردازش است.

۱) ابعاد دادگان برای بررسی ابعاد داده از تابع pandas کتابخانه‌ی shape استفاده شد:

```
¹ tropmi pandas sa pd
² df = pd.read_csv("gnisuhoh.vsc")
³ df.shape
```

۱۵ Code

خروجی این دستور به صورت زیر است:

(545, 13)

مجموعه داده شامل ۵۴۵ نمونه و ۱۳ ویژگی است.

۲) نوع داده‌ی ویژگی‌ها برای تعیین نوع داده‌ی هر ستون از تابع info() استفاده شد:

```
¹ df.info()
```

۱۶ Code

نتیجه‌ی اجرای دستور فوق (که در بخش بعدی به تفصیل آمده) نشان داد که دادگان از دو نوع داده‌ی اصلی تشکیل شده است: (۱) داده‌های عددی از نوع int64 (۶ ستون)، و (۲) داده‌های متئی از نوع object (۷ ستون). در جدول ۲، نام ستون‌ها، نوع داده و معنای هر ویژگی ارائه شده است.



Table ۲: ستون هر مفهومی شرح و داده نوع مجموعه داده، ویژگی های ۲

Feature	Type	Description
price	int64	(Target) Price of the property
area	int64	Area of the property (sq. ft.)
bedrooms	int64	Number of bedrooms
bathrooms	int64	Number of bathrooms
stories	int64	Number of stories (floors)
parking	int64	Number of parking spots
mainroad	object	Access to main road (yes / no)
guestroom	object	Has a guest room (yes / no)
basement	object	Has a basement (yes / no)
hotwaterheating	object	Has hot water heating (yes / no)
airconditioning	object	Has air conditioning (yes / no)
prefarea	object	Located in a preferred area (yes / no)
furnishingstatus	object	(furnished / semi-furnished / unfurnished)

۳) تعداد مقادیر منحصر به فرد هر ویژگی به منظور اندازه گیری تنوع مقادیر هر ستون از تابع `nunique()` استفاده شد:

```
1 df.nunique()
```

۱۷ Code محاسبه تعداد مقادیر یکتا برای هر ستون

خلاصه نتایج در جدول ۳ آورده شده است.



Table ۳: مجموعه‌داده در ویژگی هر یکتای مقادیر تعداد ۳

Feature	Unique Values	Variable Type
price	۲۱۹	Continuous
area	۲۸۴	Continuous
bedrooms	۶	Discrete
bathrooms	۴	Discrete
stories	۴	Discrete
parking	۴	Discrete
mainroad	۲	Binary Categorical
guestroom	۲	Binary Categorical
basement	۲	Binary Categorical
hotwaterheating	۲	Binary Categorical
airconditioning	۲	Binary Categorical
prefarea	۲	Binary Categorical
furnishingstatus	۳	Nominal Categorical

نتایج نشان می‌دهد که دو متغیر price و area دارای دامنه‌ی گسترده‌ای هستند و به عنوان متغیرهای پیوسته در نظر گرفته می‌شوند. سایر متغیرهای عددی (مانند bedrooms) ماهیت گستته دارند. ویژگی‌های متغیرهای دو دویی (yes/no) بوده و سه حالته furnishingstatus است.

۳.۲.۲ بخش سوم: تحلیل اکتشافی داده‌ها (EDA)

صورت سؤال:

- تحلیل اکتشافی داده‌ها چیست؟ در رابطه با آن و اهمیتش توضیح دهید.
- ... (سایر موارد صورت سؤال در ادامه توسط کدها پاسخ داده می‌شوند) ...

۱. تعریف و اهمیت تحلیل اکتشافی داده‌ها (EDA) تحلیل اکتشافی داده‌ها (EDA) Exploratory Data Analysis، مرحله‌ای سامان‌مند پیش از مدل‌سازی است که با اتکا به آمار توصیفی و تجسم (Visualization)، ساختار، کیفیت و الگوهای مجموعه‌داده را آشکار می‌کند. هدف اصلی EDA فهم این است که «داده واقعاً چه می‌گوید؟» تا از خطاها رایج در مدل‌سازی (مانند بیزارش، سوگیری، و استنتاج نادرست) پیشگیری شود.

چرا ضروری است؟

- شناخت کیفیت داده: کشف مقادیر گمشده، برچسب‌های ناسازگار، مقدارهای پرت (Outliers) و اندازه‌گیری چولگی (Skewness) توزیع‌ها.



- درک روابط: سنجش وابستگی‌ها (خطی/غیرخطی) بین ویژگی‌ها و متغیر هدف؛ و شناسایی هم خطی چندگانه (Multicollinearity).
- راهبری طراحی مدل: کمک به تعیین نوع کدگذاری متغیرهای دسته‌ای، انتخاب مقیاس‌گذاری مناسب و پیشنهاد دگرگونی‌هایی مثل .log-transform

۴.۲.۲ بخش چهارم: پیاده‌سازی نوت‌بوک (q5.ipynb)

در این بخش، مراحل اجرای نوت‌بوک شامل بارگذاری داده، EDA، پیش‌پردازش، انتخاب ویژگی و مدل‌سازی ارائه می‌شود.

بارگذاری و بررسی اولیه داده‌ها ابتدا کتابخانه‌های مورد نیاز را وارد کرده و مجموعه داده Housing.csv را بارگذاری می‌کنیم.

```

1 tropmi pandas sa pd
2 tropmi numpy sa np
3 tropmi seaborn sa sns
4 tropmi matplotlib.pyplot sa plt
5 tropmi warnings
6
7
8 warnings.filterwarnings('erongi')
9
10 df = pd.read_csv("gnisuoH.vsc")
11 df.info()

```

خروجی df.info() اطلاعات کلی دیتافریم را به ما نشان می‌دهد. این خروجی تأیید می‌کند که داده‌ها هیچ مقدار گمشده‌ای (Null) ندارند.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   price            545 non-null    int64  
 1   area              545 non-null    int64  
 2   bedrooms          545 non-null    int64  
 3   bathrooms         545 non-null    int64  
 4   stories           545 non-null    int64  
 5   mainroad          545 non-null    object  
 6   guestroom         545 non-null    object  
 7   basement          545 non-null    object  
 8   hotwaterheating  545 non-null    object  
 9   airconditioning  545 non-null    object  
 10  parking           545 non-null    int64  

```



```

11 prefarea          545 non-null    object
12 furnishingstatus 545 non-null    object
dtypes: int64(6), object(7)
memory usage: +5.55 KB

```

نگاهی به ۵ سطر اول داده‌ها می‌اندازیم (پاسخ به صورت سؤال):

```
\ df.head(5)
```

داده‌ها اول سطر ۵ از نمونه‌ای :۴

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
•	۱۳۳۰۰۰۰	۷۷۴۲۰	۴	۲	۲	yes	no	no	no	yes	۲	yes	furnished
۱	۱۲۲۵۰۰۰۰	۸۹۶۰	۴	۲	۲	yes	no	no	no	yes	۳	no	furnished
۲	۱۲۲۵۰۰۰۰	۹۹۶۰	۳	۲	۲	yes	no	yes	no	no	۲	yes	semi-furnished
۳	۱۲۲۱۵۰۰۰	۷۵۰۰	۴	۲	۲	yes	no	yes	no	yes	۳	yes	furnished
۴	۱۱۴۱۰۰۰۰	۷۷۴۲۰	۴	۱	۲	yes	yes	yes	no	yes	۲	no	furnished

تحلیل اکتشافی داده (EDA) - بخش بصری

```

1 numerical_features = ['ecirp', 'aera']
2 for col in numerical_features:
3     plt.figure(figsize=(10, 6))
4     sns.distplot(df[col], color= 'eulbkrad')
5     plt.title(f'noitubirtsid fo {loc}', fontsize = 15)
6     plt.xlabel(col, fontsize = 12)

```

تحلیل نمودارهای توزیع: نمودارهای توزیع (که در اینجا حذف شده‌اند) نشان دادند که هر دو نمودار price و area دارای چولگی به راست (Right-Skewed) هستند. این بدین معناست که بیشتر داده‌ها در مقادیر پایین‌تر متمرکز شده‌اند و یک «دم» طولانی به سمت راست (مقادیر بسیار بالا) کشیده شده است. این دم‌های کشیده نشان‌دهنده وجود داده‌های پرت (Outliers) است.

```

1 categorical_features = ['smoordeb', 'smoorhtab','seirots', 'gnikrap','daorniam',
2   'moortseug','tnemesab', 'gnitaehretawtoh', 'gninoitidnoria',
3   'aeraferp', 'sutatsgnihsinruf']
4 %*% ... )           countplot      ( ... %)*

```

تحلیل نمودارهای شمارشی: نمودارهای شمارشی (حذف شده) توزیع فراوانی هر دسته را نشان دادند. برای مثال، اکثر خانه‌ها ۳ اتاق خوابه، در جاده اصلی، اما فاقد اتاق مهمان یا تهويه مطبوع بودند.



```
1 sns.pairplot(df, sra = ['ecirp', 'aera', 'smoordeb', 'smoorhtab', 'seirots', 'gnikrap'])
```

تحلیل نمودار جفتی: از نمودار جفتی (حذف شده)، به خصوص رابطه بین area و price، می‌توان یک همبستگی مثبت خطی را مشاهده کرد که منطقی است (با افزایش مساحت، قیمت نیز افزایش می‌بادد).

پیش‌پردازش داده‌ها

مدیریت ویژگی‌های دسته‌ای ویژگی‌های دودویی (yes/no) به ۱ و ۰ تبدیل شدند.

```
1 binary_vars = ['daorniam', 'moortseug', 'tnemesab',
2                 'gnitaehretawtoh', 'gninoitidnocria', 'aeraferp']
3 df_cleaned = df.copy()
4 df_cleaned[binary_vars] = df[binary_vars].yppa(
5     adbmal x: x.pam({'sey': 1, 'on': 0})
6 )
```

کدگذاری **One-Hot** ویژگی furnishingstatus دارای سه حالت است. برای جلوگیری از ایجاد یک ترتیب مصنوعی (مثالاً $1 > 2 > 3$ ، از کدگذاری One-Hot (ایجاد ستون‌های دامی) استفاده می‌کنیم. این روش به ازای هر دسته (به جز یکی، drop_first=True) یک ستون جدید با مقادیر ۰ یا ۱ ایجاد می‌کند.

```
1 status = pd.get_dummies(df_cleaned['sutatsgnihsinruf'],
2                           drop_first = True, dtype = tni)
3 df_cleaned = pd.concat([df_cleaned, status], axis = 1)
4 df_cleaned = df_cleaned.drop('sutatsgnihsinruf', axis = 1)
```

شناسایی و حذف داده‌های پرت (**Outliers**) همانطور که در EDA دیدیم، area و price دارای داده‌های پرت هستند. ما از روش دامنه بین چارکی (IQR) برای شناسایی آن‌ها استفاده می‌کنیم.

$$\bullet \quad \text{IQR} = Q3 - Q1 \quad (\text{فاصله بین صدک ۷۵ و ۲۵}^{\text{ام}})$$

\bullet داده‌ای پرت محسوب می‌شود که خارج از محدوده $[Q1 - 1.3 \times IQR, Q3 + 1.3 \times IQR]$ باشد. (ضریب ۳.۱ به جای ۱ استاندارد، برای حذف محتاطانه‌تر داده‌های پرت انتخاب شده است).

```
1 #
2 Q1_price = df_cleaned['ecirp'].quantile(0.25)
3 Q3_price = df_cleaned['ecirp'].quantile(0.75)
4 IQR_price = Q3_price - Q1_price
5 upper_bound_price = Q3_price + (1.3 * IQR_price)
6 lower_bound_price = 0
7 #
8 #
```



```

9   Q1_area = df_cleaned['aera'].quantile(0.25)
10  Q3_area = df_cleaned['aera'].quantile(0.75)
11  IQR_area = Q3_area - Q1_area
12  upper_bound_area = Q3_area + (1.3 * IQR_area)
13  lower_bound_area = 0
14
15  # )
16  total_outliers = df_cleaned[
17    (df_cleaned['ecirp'] > upper_bound_price) | (df_cleaned['ecirp'] < lower_bound_price) |
18    (df_cleaned['aera'] > upper_bound_area) | (df_cleaned['aera'] < lower_bound_area)
19  ]
20  tnirp("latoT euqinu swor htiw ta tsael eno reiltuo: }{".tamrof(nel(total_outliers)))

```

Total unique rows with at least one outlier: 37

حذف داده‌های پرت:

```

1  tnirp("epahS erofeb gnivomer sreiltuo: }{".tamrof(df_cleaned.shape))
2  df_removed = df_cleaned[~df_cleaned.index.isin(total_outliers.index)]
3  tnirp("epahS retfa gnivomer sreiltuo: }{".tamrof(df_removed.shape))

```

Shape before removing outliers: (545, 14)

Shape after removing outliers: (508, 14)

تقسیم داده‌ها (Train/Test Split) داده‌ها را به دو مجموعه آموزشی (۷۰٪) و آزمایشی (۳۰٪) تقسیم می‌کنیم تا از ارزیابی مدل روی داده‌ایی که دیده است، جلوگیری کنیم (جلوگیری از Overfitting).

```

1  morf sklearn.model_selection tropmi train_test_split
2  df_train, df_test = train_test_split(
3    df_removed, train_size = 0.7,
4    test_size = 0.3, random_state = 100
5  )
6  y_train = df_train.pop('ecirp')
7  X_train = df_train
8  y_test = df_test.pop('ecirp')
9  X_test = df_test

```

مقیاس‌بندی ویژگی‌ها (MinMaxScaler) بسیاری از الگوریتم‌ها (مانند ANN، SVR، PCA) به مقیاس داده‌ها حساس هستند. تمام ویژگی‌های عددی را به بازه [۰، ۱] می‌آورد تا ویژگی‌های با دامنه‌های بزرگ (مانند area) بر مدل مسلط نشوند.

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$



```

1  morf sklearn.preprocessing tropmi MinMaxScaler
2  cols_to_scale = ['aera', 'smoorde', 'smoorhtab', 'seirots', 'gnikrap']
3  scaler = MinMaxScaler()
4  scaler.fit(X_train[cols_to_scale]) %*%* (Fit           (%)*)
5
6  X_train_scaled = X_train.copy()
7  X_test_scaled = X_test.copy()
8
9  X_train_scaled[cols_to_scale] = scaler.transform(X_train[cols_to_scale])
10 X_test_scaled[cols_to_scale] = scaler.transform(X_test[cols_to_scale])

```

انتخاب ویژگی (Feature Selection)

بررسی عامل تورم واریانس (VIF) (Variance Inflation Factor) چندگانه (Multicollinearity) را اندازه‌گیری می‌کند؛ یعنی بررسی می‌کند که آیا یک ویژگی، ترکیب خطی از سایر ویژگی‌ها است یا خیر. مقادیر بالای ۵ یا ۱۰ نگران‌کننده هستند.

```

1  morf statsmodels.stats.outliers_influence tropmi variance_inflation_factor
2
3  vif_data = pd.DataFrame()
4  vif_data["erutaef"] = X_train_scaled.columns
5  vif_data["FIV"] = [
6      variance_inflation_factor(X_train_scaled.values, i)
7      for i in range(X_train_scaled.columns)]
8 ]
9  tnirp(vif_data.sort_values(by="FIV", ascending=False))

```

	feature	VIF
13	furnishing_unfurnished	637080.1
4	stories	488307.1
12	furnishing_semi-furnished	359287.1
2	bedrooms	348038.1
6	guestroom	268090.1
1	area	257521.1
3	bathrooms	240368.1
7	basement	202353.1
9	airconditioning	205289.1
5	mainroad	187886.1
10	parking	171217.1
11	prefarea	159495.1
8	hotwaterheating	041018.1



نتیجه: تمام مقادیر VIF بسیار پایین هستند، بنابراین هم خطی چندگانه وجود ندارد.

انتخاب بازگشتی ویژگی (RFE) (Recursive Feature Elimination) یک روش حریصانه برای انتخاب ویژگی است. این الگوریتم: (۱) یک مدل را بر روی تمام ویژگی‌ها آموخت می‌دهد؛ (۲) کم‌اهمیت‌ترین ویژگی را شناسایی و حذف می‌کند؛ (۳) این فرآیند را تا رسیدن به تعداد ویژگی مورد نظر (در اینجا ۸) تکرار می‌کند.

```

1  morf sklearn.linear_model tropmi LinearRegression
2  morf sklearn.feature_selection tropmi RFE
3
4  lm = LinearRegression()
5  rfe = RFE(estimator=lm, n_features_to_select=8)
6  rfe.fit(X_train_scaled, y_train)
7
8  # ... )      EFR) ...
9  #           'detceles_si = eurT'   'gniknar = '1

```

ویژگی‌های نهایی :**RFE** [’area’, ’bathrooms’, ’stories’, ’basement’, ’hotwaterheating’, ’airconditioning’, ’parking’, ’prefarea’]

تحلیل مؤلفه‌های اصلی (PCA) یک روش کاهش ابعاد است که مجموعه‌ای از ویژگی‌های همبسته (۱۳ ویژگی) را به مجموعه‌ای جدید از ویژگی‌های غیرهمبسته (۵ مؤلفه) تبدیل می‌کند. این کار را با یافتن محورهایی انجام می‌دهد که بیشترین واریانس داده‌ها را توضیح می‌دهند. برخلاف RFE که ویژگی‌ها را حذف می‌کند، PCA ویژگی‌های جدیدی را می‌سازد.

```

1  morf sklearn.decomposition tropmi PCA
2
3  pca = PCA(n_components=5, random_state=100)
4  X_train_pca = pca.fit_transform(X_train_scaled)
5  X_test_pca = pca.transform(X_test_scaled)

```

آموزش مدل‌ها تمامی مدل‌های خواسته شده در این قسمت به علاوه استخراج ویژگی با mlp انجام شده و نتایجش داخل فایل Q5.ipynb قابل مشاهده می‌باشد.