سوال1

ابتدا کتابخانه‌های لازم بارگذاری شده و داده‌ها از فایل CSV خوانده می‌شوند. ماتریس همبستگی با data.corr() محاسبه و با استفاده از seaborn به صورت نمودار حرارتی (heatmap) رسم می‌شود.در بخش بعد، ویژگی‌های عددی به جز ستون 'country' انتخاب و با استفاده از StandardScaler نرمال‌سازی می‌شوند. داده‌های نرمال شده در دیتافریم جدید ذخیره و نمایش داده می‌شوند.سپس روش Elbow برای تعیین تعداد بهینه خوشه‌ها استفاده می‌شود. با تغییر تعداد خوشه‌ها از 1 تا 10، مقدار

SSE (Sum of Squared Errors) محاسبه و رسم می‌شود. در حلقه kmeansرا از k برابر 1تا 10 اعمال کرده و در sseمیریزیم و سپس نمودار را رسم میکنیم.

روش امتیاز Silhouette برای تعیین تعداد بهینه خوشه‌ها استفاده می‌شود. امتیاز Silhouette برای تعداد خوشه‌های 2 تا 10 با کد silhouette\_score(data\_scaled[features], labels)محاسبه و رسم میکنیم.

حال تعداد بهینه خوشه‌ها براساس روش‌های Elbow و Silhouette تعیین می‌شود.

optimal\_clusters\_silhouette = sil\_scores.index(max(sil\_scores)) + 2

optimal\_clusters = optimal\_clusters\_elbow if optimal\_clusters\_elbow == optimal\_clusters\_silhouette else optimal\_clusters\_silhouette

سپس، الگوریتم k-means با تعداد بهینه خوشه‌ها اجرا شده و برچسب خوشه‌ها به داده‌ها اضافه می‌شود.

در مرحله بعد سه ویژگی child\_mort، income و gdpp برای بصری‌سازی انتخاب شده و نمودارهای پراکندگی این ویژگی‌ها با رنگ‌بندی براساس خوشه‌ها رسم می‌شود.

sns.pairplot(data\_scaled, vars=selected\_features, hue='Cluster', palette='Set1', diag\_kind='kde')

plt.suptitle('Clusters Visualization with Selected Features', y=1.02)

plt.show()

برای کاهش ابعاد ،الگوریتم PCA با کتابخانه PCA()بر روی داده‌های نرمال شده اعمال و واریانس توضیح داده شده توسط مؤلفه‌های اصلی محاسبه می‌شود. سپس تعداد مؤلفه‌های اصلی که حداقل ۹۰٪ واریانس را توضیح می‌دهند تعیین می‌شود.

n\_components = next(i for i, total\_variance in enumerate(explained\_variance.cumsum()) if total\_variance >= 0.9) + 1

در اخر داده‌ها با استفاده از تعداد بهینه مؤلفه‌های اصلی کاهش ابعاد داده شده و الگوریتم k-means روی داده‌های کاهش یافته اجرا می‌شود. نتایج خوشه‌بندی اصلی و مبتنی بر PCA با هم مقایسه می‌شوند.

comparison = data\_scaled[['country', 'Cluster', 'Cluster\_PCA']]

print(comparison.head())

خوشه‌ها در ابعاد کاهش یافته بصری‌سازی شده و داده‌های کاهش یافته در یک فایل CSV ذخیره می‌شوند.

سوال2

برای این سوال دو راه حل نوشته شده است اولی بر اساس دیتاست ساخته شده در سوال اول و دومی بر اساس دیتاست رندوم و ساختگی:

پاسخ اول:

ابتدا تمام مراحل سوال اول را با تغییراتی در جزییات دوباره انجام میدهیم به عنوان مثال از روش MinMaxScaler برای پیش پردازش استفاده میکنیم.در روشelbowمقدار WCSS (مجموع مربعات درون خوشه‌ها) محاسبه میشود و برای pca حداقل ۹۵٪ واریانس را در نظر میگیریم.

حال به حل سوال 2 میپردازیم: داده‌های نهایی بارگذاری شده و ویژگی‌ها و هدف انتخاب می‌شوند. به منظور ساده‌سازی، هدف به دو کلاس باینری تبدیل شده و داده‌ها به مجموعه‌های آموزشی و اعتبارسنجی تقسیم می‌شوند.

یک تابع برای آموزش مدل‌ها و ثبت خطاهای آموزشی و اعتبارسنجی تعریف میکنیم و سپس مدل‌های پرسپترون و آدالاین با استفاده از داده‌های آموزشی و در تعداد ایپاک 100 آموزش داده می‌شوند:

for epoch in range(epochs):

        model.partial\_fit(X\_train, y\_train, classes=np.unique(y\_train))

        train\_errors.append(1 - model.score(X\_train, y\_train))

        val\_errors.append(1 - model.score(X\_val, y\_val))

و در نهایت نمودار خطاهای آموزشی و اعتبارسنجی برای مدل‌های پرسپترون و آدالاین رسم می‌شود. همچنین دقت نهایی هر مدل محاسبه و نمایش داده می‌شود.

پاسخ دوم:

ابتدا داده‌های مصنوعی را با استفاده از تابع make\_classification از کتابخانه sklearn تولید می‌کنیم. یک مجموعه داده مصنوعی دو کلاسه با 10000 نمونه و دو ویژگی تولید می‌شود که کلاس‌ها به خوبی از هم جدا هستند.

X, y = make\_classification(n\_samples=10000, n\_features=2, n\_informative=2, n\_redundant=0,

                           n\_clusters\_per\_class=1, class\_sep=2.0, random\_state=42)

سپس داده‌ها را ترسیم می‌کنیم.

# Plot the distribution of the data

plt.figure(figsize=(10, 6))

plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='red', label='Class 0')

plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='blue', label='Class 1')

plt.title('Linearly Separable Data')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

داده‌های دو کلاسه را بر روی نمودار پراکندگی ترسیم می‌کنیم. نقاط قرمز مربوط به کلاس 0 و نقاط آبی مربوط به کلاس 1 هستند.

مدل پرسپترون و مدل ادالاین را با (SGDClassifier و loss برابر با 'squared\_error' )مقدار دهی ،داده ها را اموزش و ارزیابی میکنیم:

perceptron = Perceptron(max\_iter=1000, random\_state=42)

adaline = SGDClassifier(loss='squared\_error', max\_iter=1000, random\_state=42)

داده‌ها را به دو مجموعه آموزشی و اعتبارسنجی تقسیم می‌کنیم، به دو بخش ۸۰ درصد آموزشی و ۲۰ درصد اعتبارسنجی تقسیم می‌شوند.

تابعی برای آموزش مدل‌ها و ثبت خطاهای آموزشی و اعتبارسنجی تعریف می‌کنیم. مدل به صورت تدریجی (partial\_fit) آموزش داده می‌شود و خطاهای آموزشی و اعتبارسنجی در هر دوره ثبت می‌شوند.

سپس مدل‌ها با استفاده از تابع train\_and\_plot\_errors آموزش داده می‌شوند و خطاهای آموزشی و اعتبارسنجی ثبت می‌شوند. نمودار خطاهای آموزشی و اعتبارسنجی برای مدل‌های پرسپترون و آدالاین رسم می‌شوند ودقت نهایی مدل‌ها براساس خطای اعتبارسنجی در آخرین دوره آموزشی محاسبه و نمایش داده می‌شود.

سوال 3

حل مسئله "Frozen Lake" با استفاده از روش تکرار ارزش (Value Iteration) : در این مسئله، هدف رسیدن به حالت هدف (G) از حالت شروع (S) بدون افتادن در حالت‌های حفره (H) است.

تکرار ارزش: در این قسمت، مسئله با استفاده از روش تکرار ارزش حل می‌شود. هدف، پیدا کردن سیاست بهینه است که به عامل کمک کند تا به حالت هدف (G) برسد بدون اینکه وارد حالت‌های حفره (H) شود.

این کد شامل دو تابع اصلی است:

1. value\_iteration:
   * این تابع برای محاسبه جدول ارزش بهینه استفاده می‌شود.
   * پارامترها: env (محیط)، gamma (ضریب تخفیف).
   * ابتدا یک جدول ارزش صفر ایجاد می‌شود.
   * در هر تکرار، جدول ارزش بروز شده محاسبه می‌شود.
   * برای هر حالت، مقدار Q برای هر عمل محاسبه می‌شود و بیشترین مقدار Q به عنوان ارزش جدید آن حالت انتخاب می‌شود.
   * تکرارها تا زمانی که تفاوت جدول ارزش بروز شده و قدیمی کمتر از یک آستانه مشخص باشد ادامه می‌یابد.
2. extract\_policy:
   * این تابع برای استخراج سیاست بهینه از جدول ارزش بهینه استفاده می‌شود.
   * پارامتر: value\_table (جدول ارزش)، env (محیط)،gamma (ضریب تخفیف).
   * برای هر حالت، مقدار Q برای هر عمل محاسبه می‌شود.
   * عملی که بیشترین مقدار Q را داشته باشد به عنوان عمل بهینه برای آن حالت انتخاب می‌شود.

در نهایت، این توابع برای محیط FrozenLake-v1 اجرا می‌شوند و سیاست بهینه و تابع ارزش بهینه محاسبه و نمایش داده می‌شود.

خروجی نیز بهترین اقذام را در هر مرحله نمایش میدهد.

سوال4

ابتدا دیتاست اعداد دست‌نویس digits از sklearn بارگیری و نمونه‌ای از تصاویر هر عدد را نمایش می‌دهیم:

digits = load\_digits()

داده‌های تصویری را با استفاده از StandardScaler استانداردسازی می‌کنیم تا مقادیر آن‌ها به صورت یکنواخت در بیایند. مدل خوشه‌بندی KMeans را مقداردهی اولیه و آموزش می‌دهیم و برچسب‌های خوشه‌ها را بدست می‌آوریم

kmeans = KMeans(n\_clusters=10, random\_state=42)

سپس یک تصویر از هر خوشه را نمایش می‌دهیم:

fig, axes = plt.subplots(2, 5, figsize=(10, 4))

for i, ax in enumerate(axes.flat):

    # Find an image with the current KMeans label i

    idx = np.where(kmeans\_labels == i)[0][0]

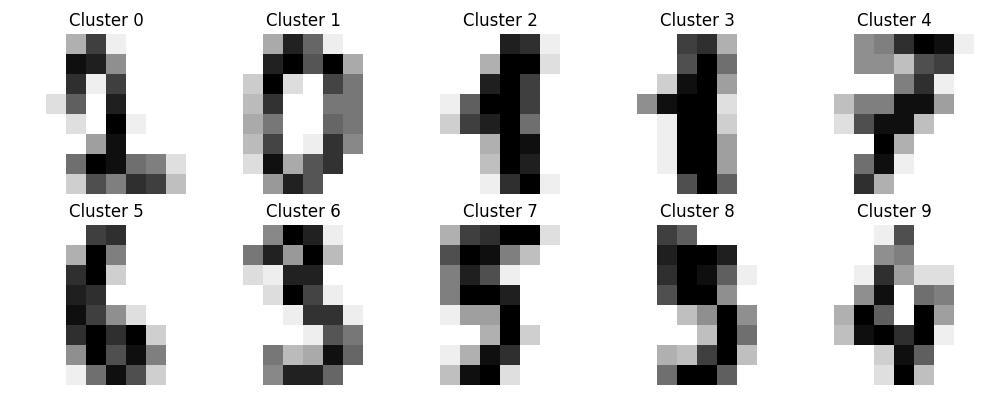
    ax.imshow(digits.images[idx], cmap='binary')

    ax.set\_title(f"Cluster {i}")

    ax.axis('off')

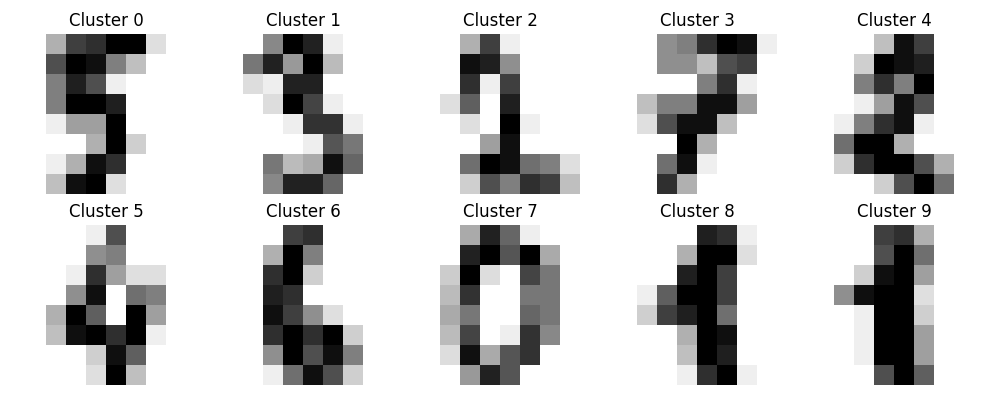
plt.tight\_layout()

plt.show()



در مرحله بعد،مدل خوشه‌بندی سلسله‌مراتبی را مقداردهی اولیه و آموزش می‌دهیم و برچسب‌های خوشه‌ها را بدست می‌آوریم و یک خوشه از هرکدام را نمایش میدهیم.

hierarchical = AgglomerativeClustering(n\_clusters=10)



سپس با استفاده از شاخص سیلوئت (Silhouette Score)، مدل‌های KMeans و Hierarchical Clustering را ارزیابی می‌کنیم

kmeans\_silhouette = silhouette\_score(digits.data, kmeans\_labels)

print(f"Silhouette Score for KMeans: {kmeans\_silhouette}")

hierarchical\_silhouette = silhouette\_score(digits.data, hierarchical\_labels)

print(f"Silhouette Score for Hierarchical Clustering: {hierarchical\_silhouette}")

برای هر خوشه، برچسب مربوط به بیشترین تعداد داده‌های صحیح (واقعی) را اختصاص می‌دهیم تا برچسب‌های پیش‌بینی شده برای هر خوشه را بدست آوریم.

سوال5

ابتدا دو دیتاست را از مسیر مشخص شده بارگذاری می‌کنیم و نمودار پراکندگی هر کدام را رسم می‌کنیم تا توزیع داده‌ها را مشاهده کنیم.سپس مدل KMeans را با تعداد خوشه‌های مشخص شده برای هر دیتاست آموزش می‌دهیم و نتایج را ترسیم می‌کنیم.بر اساس نمودار مقدار k را برای دیتاست اول3 و برای دیتاست دوم 2 در نظر میگیریم.

برای ارزیابی کیفیت خوشه‌بندی از نمره سیلوئت استفاده می‌کنیم. همچنین، از روش Elbow برای تعیین تعداد بهینه خوشه‌ها بهره می‌بریم.

silhouette\_score1 = silhouette\_score(dataset1, labels1)

silhouette\_score2 = silhouette\_score(dataset2, labels2)

elbow:

sse1 = []

for k in range(1, 10):

    kmeans = KMeans(n\_clusters=k, random\_state=42)

    kmeans.fit(dataset1)

    sse1.append(kmeans.inertia\_)

در بخش بعد مدل DBSCAN را برای هر دو دیتاست آموزش می‌دهیم و نتایج را ترسیم می‌کنیم ونمره سیلوئت را برای خوشه‌بندی DBSCAN محاسبه می‌کنیم.

# آموزش DBSCAN بر روی Dataset1

dbscan1 = DBSCAN(eps=0.5, min\_samples=5)

dbscan1.fit(dataset1)

labels\_dbscan1 = dbscan1.labels\_

# آموزش DBSCAN بر روی Dataset2

dbscan2 = DBSCAN(eps=0.3, min\_samples=5)

dbscan2.fit(dataset2)

labels\_dbscan2 = dbscan2.labels\_

در بخش بعدی مدل KMeans را بر روی دیتاست سوم آموزش می‌دهیم و نقاط دورافتاده را شناسایی و نمایش می‌دهیم.

# شناسایی نقاط دورافتاده با استفاده از KMeans

kmeans3 = KMeans(n\_clusters=3, random\_state=42)

kmeans3.fit(dataset3)

labels3 = kmeans3.labels\_

( مراحل در کد توضیح داده شده اند.)

سوال 6

این سوال مراحل مختلفی برای کاهش تعداد رنگ‌ها در یک تصویر با استفاده از الگوریتم K-means Clustering است.

ابتدا کتابخانه‌های مورد نیاز مانند numpy برای کار با آرایه‌ها، matplotlib برای ترسیم نمودارها، KMeans از sklearn.cluster برای انجام خوشه‌بندی و PIL برای کار با تصاویر بارگیری می‌شوند.

تابعی به نام quantize\_image تعریف می‌شود که یک تصویر و تعداد رنگ‌های مورد نظر را به عنوان ورودی می‌پذیرد و تصویر کوانتایز شده را برمی‌گرداند. برای جاگذاری هر تصویر در رنگ بندی جدید در تابع به روش kmeansنیاز داریم .

سپس تصویر با استفاده از PIL بارگذاری شده و به یک آرایه‌ی numpy تبدیل می‌شود و یک لیست از تعداد خوشه‌های مورد نظر برای کوانتایز کردن تصویر تعریف می‌شود

تصویر اصلی و تصاویر کوانتایز شده با استفاده از تعداد خوشه‌های مختلف ترسیم می‌شوند.

for i, n\_colors in enumerate(n\_colors\_list, 2):

    quantized\_image = quantize\_image(image, n\_colors)

    plt.subplot(2, 4, i)

    plt.imshow(quantized\_image)

    plt.title(f'{n\_colors} colors')

    plt.axis('off')

تصاویر در notebook jupyter اورده شده است.