دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق

درس مبانی سیستم‌های هوشمند

استاد: دکتر مهدی علیاری

پایانترم

نام و نام خانوادگی	محمدامین محمدیون شبستری
شمارهٔ دانشجویی	۴۰۱۲۲۵۰۳
نام و نام خانوادگی	محمدسبحان سخایی
شمارهٔ دانشجویی	۴۰۱۱۹۲۵۳
تاریخ	بهمن ۱۴۰۴
لینک گیت‌هاب	

فهرست مطالب

۱ پرسش اول ۶
۱.۱ معرفی دیتاست (Dataset Overview) ۶
۲.۱ پاک‌سازی داده‌ها (Data Cleaning) ۶
۳.۱ تحلیل توزیع داده‌ها (Distribution Analysis) ۷
۴.۱ کدگذاری ویژگی‌ها (Feature Encoding) ۸
۵.۱ تحلیل همبستگی (Correlation Analysis) ۸
۶.۱ نتیجه‌گیری اولیه از بررسیه اولیه داده ۱۰
۷.۱ مبانی نظری و ریاضیاتی الگوریتم PSO ۱۰
۱.۷.۱ فرمول‌بندی ریاضی حرکت ذرات ۱۰
۲.۷.۱ ریاضیات Binary PSO برای انتخاب ویژگی ۱۱
۸.۱ پیاده‌سازی با pyswarms و تحلیل تابع هدف ۱۱
۹.۱ تحلیل نتایج بهینه‌سازی و همگرایی ۱۲
۱۰.۱ ارزیابی نهایی با مدل Random Forest ۱۲
۱۱.۱ انتخاب ویژگی با الگوریتم‌های تکاملی – الگوریتم ژنتیک ۱۳
۱.۱۱.۱ ۱. نمایش کروموزوم (Representation) ۱۳
۲.۱۱.۱ ۲. تابع برازندگی (Fitness Function) ۱۴
۳.۱۱.۱ ۳. عملگر انتخاب (Selection Operator) ۱۴
۴.۱۱.۱ ۴. عملگر تقاطع (Crossover Operator) ۱۴
۵.۱۱.۱ ۵. عملگر جهش (Mutation Operator) ۱۵
۱۲.۱ تحلیل پیاده‌سازی با کتابخانه DEAP ۱۵
۱۳.۱ تحلیل نتایج و نمودارهای همگرایی ۱۵
۱۴.۱ ارزیابی مدل Random Forest با ویژگی منتخب ۱۷
۱۵.۱ تحلیل تطبیقی و مقایسه عملکرد PSO و GA ۱۸
۱.۱۵.۱ تفسیر نمودار مقایسه‌ای (Comparative Convergence Analysis) ۱۸
۲.۱۵.۱ نتیجه‌گیری نهایی: کدام الگوریتم برنده است؟ ۱۹

۲ پرسش دوم ۱۹
۱.۲ توضیحات داده ۱۹
۱.۱.۲ تحلیل توزیع جنسیت ۲۰
۲.۱.۲ تحلیل توزیع متغیرهای عددی ۲۱
۳.۱.۲ تحلیل رابطه درآمد و امتیاز خرید ۲۲
۲.۲ ویژگی‌های عددی و استانداردسازی ۲۳
۳.۲ استفاده از PCA برای نمایش دوبعدی ۲۳
۴.۲ الگوریتم K-Means ۲۴

۱.۴.۲ توضیح الگوریتم ۲۴
۲.۴.۲ استفاده از K-Means ۲۵
۳.۴.۲ بررسی نتایج ۲۶
۵.۲ خوشه‌بندی سلسله‌مراتبی (Agglomerative Clustering) ۲۷
۱.۵.۲ توضیح الگوریتم ۲۷
۲.۵.۲ دسته بندی با Agglomerative Clustering ۲۸
۳.۵.۲ نتایج ۲۸
۶.۲ الگوریتم DBSCAN ۳۰
۱.۶.۲ توضیح الگوریتم ۳۰
۲.۶.۲ دسته بندی با DBSCAN ۳۱
۳.۶.۲ نتایج ۳۱
۷.۲ مقایسه سه روش دسته‌بندی ۳۳
۱.۷.۲ تحلیل بصری و ساختاری خوشه‌ها ۳۳

۳ پرسش سه ۳۴
۱.۳ مبانی نظری و مفاهیم پایه ۳۴
۱.۱.۳ الگوریتم Q-Learning و مفهوم تابع ارزش ۳۵
۲.۳ قسمت الف: کالبدشکافی معادله به‌روزرسانی ۳۵
۱.۲.۳ تحلیل دقیق پارامترها ۳۵
۲.۲.۳ چرا Q-Learning یک روش Off-Policy است؟ ۳۶
۳.۳ قسمت ب: حل تشریحی و قدم‌به‌قدم مثال عددی ۳۶
۴.۳ بخش ج: عوامل ناپایداری و راهکارهای عملی دقیق ۳۷
۱.۴.۳ ۱. مشکل مقیاس نامناسب پاداش‌ها (Improper Reward Scaling) ۳۷
۲.۴.۳ ۲. مشکل نرخ اکتشاف ثابت (Fixed Exploration Rate) ۳۸

فهرست تصاویر

۱ توزیع ویژگی‌های دموگرافیک و مالی متقاضیان ۷
۲ ماتریس همبستگی پیرسون بین ویژگی‌ها ۹
۳ منحنی همگرایی (Convergence Curve) الگوریتم PSO ۱۲
۴ گزارش طبقه‌بندی (Classification Report) مدل نهایی ۱۳
۵ آمار توصیفی نسل‌ها (میانگین و مینیمم هزینه) ۱۶
۶ منحنی همگرایی (Convergence Curve) الگوریتم ژنتیک ۱۷
۷ مقایسه منحنی همگرایی: PSO (آبی) در برابر GA (قرمز) ۱۸
۸ توزیع فراوانی جنسیت مشتریان ۲۰
۹ توزیع آماری سن، درآمد سالانه و امتیاز خرید ۲۱
۱۰ رابطه بین درآمد سالانه و امتیاز خرید با تفکیک جنسیت ۲۲
۱۱ نمایش فضای ویژگی‌ها پس از کاهش ابعاد با PCA ۲۴
۱۲ نمودار آرنج (Inertia) و امتیاز سیلوئت برای مقادیر مختلف k ۲۷
۱۳ نمایش خوشه‌بندی Agglomerative با پیوندهای مختلف در فضای PCA ۲۹
۱۴ دندروگرام (Dendrogram) برای پیوندهای مختلف ۳۰
۱۵ مقایسه بصری خوشه‌بندی‌ها در فضای دوبعدی PCA ۳۳

فهرست جداول

۱ جدول تبدیل متغیرهای کیفی به کمی ۸
۲ مقایسه آماری بهترین نتایج سه الگوریتم ۳۳

فهرست برنامه‌ها

۱ پرسش اول

۱.۱ معرفی دیتاست (Dataset Overview)

دیتاست مورد بررسی با نام loan_train.csv شامل اطلاعات مالی و دموگرافیک متقاضیان وام مسکن است. هدف نهایی در این پروژه، استفاده از این ویژگی‌ها برای پیش‌بینی وضعیت تایید وام (Loan Status) است. این دیتاست شامل ۶۱۴ نمونه (Rows) و ۱۲ ویژگی (Columns) می‌باشد. ساختار داده‌ها ترکیبی از متغیرهای عددی و طبقه‌بندی شده (Categorical) است.

جزئیات ویژگی‌های موجود به شرح زیر است:

- **Gender**: جنسیت متقاضی (شامل مقادیر Male/Female). حدود ۱۳ داده گم‌شده دارد.
- **Married**: وضعیت تأهل. افراد متأهل معمولاً ثبات مالی بیشتری دارند.
- **Dependents**: تعداد افراد تحت تکفل. تعداد فرزندان می‌تواند بر توانایی بازپرداخت تأثیر بگذارد.
- **Education**: وضعیت تحصیلات (Graduate/Not Graduate). انتظار می‌رود افراد تحصیل‌کرده درآمد بالاتری داشته باشند.
- **Self_Employed**: وضعیت خوداشتغالی. شغل‌های آزاد ممکن است ریسک بازپرداخت متفاوتی داشته باشند.
- **Applicant_Income**: درآمد متقاضی (عدد صحیح). توزیع این داده‌ها معمولاً چولگی دارد.
- **Coapplicant_Income**: درآمد ضامن یا همسر.
- **Loan_Amount**: مبلغ وام درخواستی (بر حسب هزار دلار).
- **Term**: مدت زمان بازپرداخت وام به ماه (مثلاً ۳۶۰ ماه معادل ۳۰ سال).
- **Credit_History**: سابقه اعتباری (۱ برای خوش‌حساب، ۰ برای بدحساب). این ویژگی معمولاً مهم‌ترین فاکتور بانکی است.
- **Area**: منطقه مسکونی (Urban, Semiurban, Rural).
- **Status**: متغیر هدف (Target Variable) که وضعیت نهایی وام (Y/N) را نشان می‌دهد.

بررسی اولیه با دستور ()info نشان می‌دهد که ستون‌های Credit_History (۵۰ مورد)، Self_Employed (۳۲ مورد) و Loan_Amount (۲۲ مورد) دارای بیشترین مقادیر گم‌شده (Null Values) هستند که باید در مرحله پیش‌پردازش مدیریت شوند.

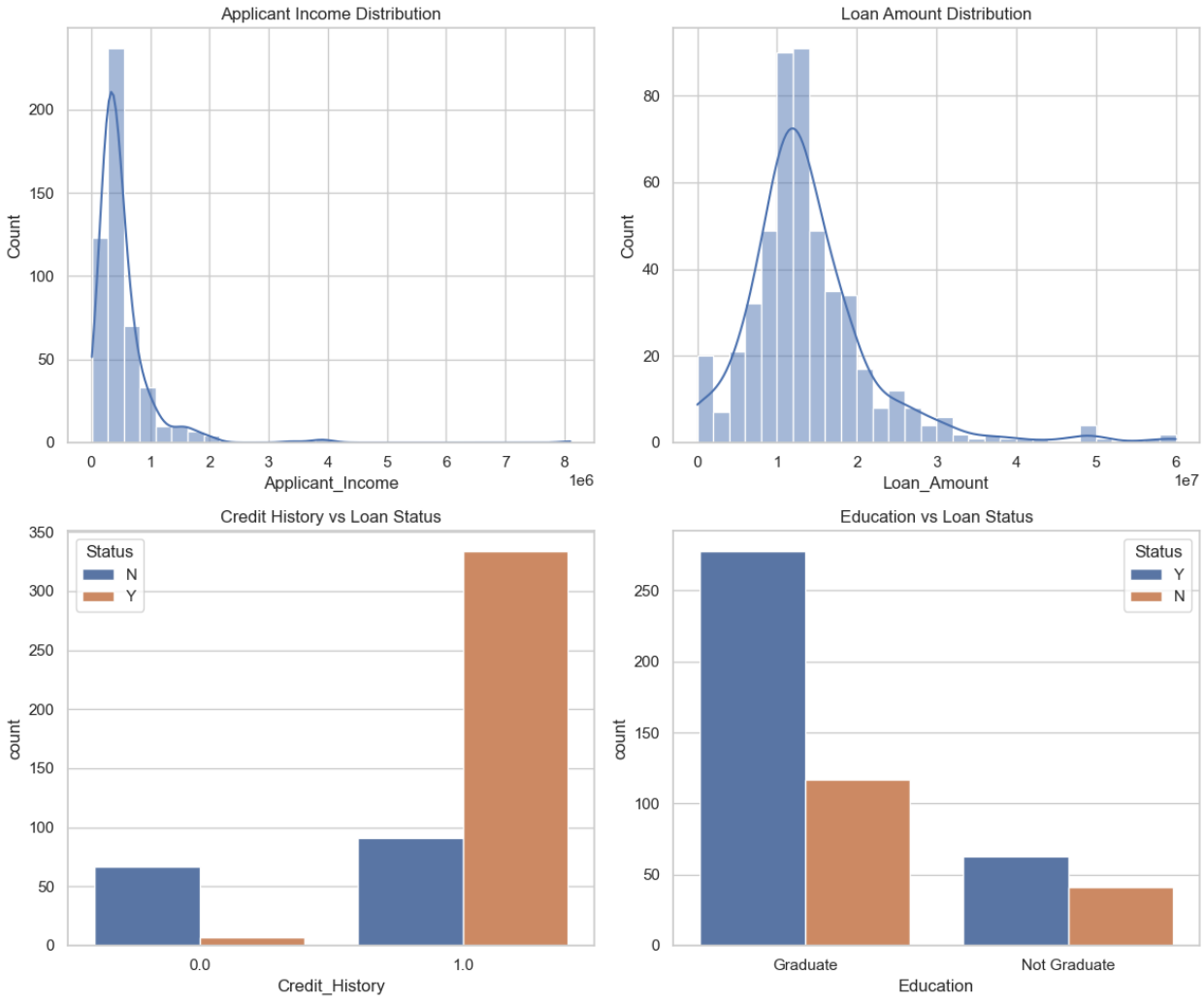
۲.۱ پاک‌سازی داده‌ها (Data Cleaning)

وجود مقادیر گم‌شده (NaN) می‌تواند باعث خطای الگوریتم‌های یادگیری ماشین شود. در این پروژه، استراتژی «حذف سطرها» (Drop Rows) انتخاب شده است. اگرچه روش‌های دیگری مانند جایگزینی با میانگین (Imputation) وجود دارد، اما برای حفظ اصالت داده‌های آموزشی، سطرهای ناقص حذف شدند.

با اجرای دستور ()dropna، تعداد نمونه‌ها از ۶۱۴ به ۴۹۹ کاهش یافت (حدود ۱۹٪ از داده‌ها حذف شد). این کاهش حجم داده اگرچه قابل توجه است، اما تضمین می‌کند که مدل بر روی داده‌های کاملاً واقعی و تمیز آموزش می‌بیند. خروجی نهایی دستور ()isnull().sum نشان می‌دهد که تمامی ستون‌ها اکنون دارای ۰ مقدار گم‌شده هستند.

۳.۱ تحلیل توزیع داده‌ها (Distribution Analysis)

در این بخش به تحلیل نمودارهای موجود در تصویر plot1.png می‌پردازیم. این نمودارها توزیع تک‌متغیره ویژگی‌ها را نشان می‌دهند.



شکل ۱: توزیع ویژگی‌های دموگرافیک و مالی متقاضیان

تحلیل دقیق آماری نمودارهای شکل ۱:

۱. عدم تعادل جنسیتی (**Gender Imbalance**): حدود ٪۸۱ از متقاضیان مرد هستند. این موضوع ممکن است باعث شود مدل در پیش‌بینی برای زنان دچار سوگیری (Bias) شود.

۲. وضعیت تأهل: نزدیک به ٪۶۵ متقاضیان متأهل هستند که نشان‌دهنده تمایل بیشتر خانواده‌ها به دریافت وام مسکن است.

۳. تحصیلات: حدود ٪۷۸ متقاضیان دارای تحصیلات دانشگاهی (Graduate) هستند.

۴. سابقه اعتباری (**Credit History**): بیش از ٪۸۴ افراد دارای سابقه اعتباری مثبت (1.0) هستند. این عدم تعادل شدید می‌تواند پیش‌بینی موارد بدحساب را دشوار کند.

۵. توزیع درآمد (**Applicant Income**): هیستوگرام درآمد نشان‌دهنده «چولگی شدید به راست» (Right Skewness) است. میانگین درآمد (۵۴۰۳) از میانه (۳۸۱۲) بیشتر است که نشان‌دهنده وجود متقاضیان با درآمد بسیار بالا (Outliers) در دیتاست است.

۶. متغیر هدف (**Status**): حدود ٪۶۹ وام‌ها تایید شده (Y) و ٪۳۱ رد شده‌اند. این یعنی با یک مسئله نامتوازن روبرو هستیم و دقت مدل باید با معیارهایی مثل F1-Score سنجیده شود، نه فقط Accuracy.

۴.۱ کدگذاری ویژگی‌ها (**Feature Encoding**)

از آنجا که مدل‌های ریاضی قادر به درک متن نیستند، باید متغیرهای کیفی (Categorical) به کمی (Numerical) تبدیل شوند. برای این کار از روش Label Encoding استفاده شده است.

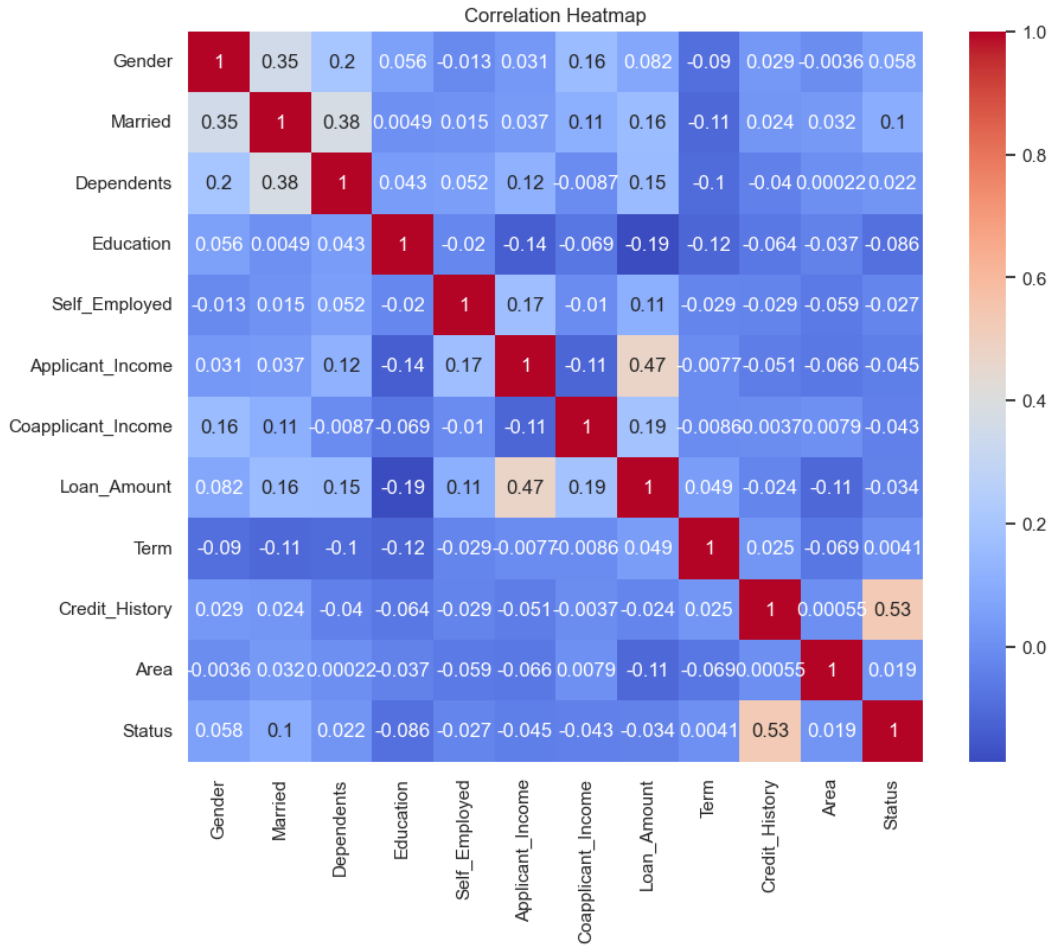
تحلیل انتخاب روش: استفاده از Label Encoder برای متغیرهای دو دویی (مثل جنسیت و تأهل) کاملاً مناسب است. اما برای متغیرهایی مثل Area (که سه حالت دارد)، این روش یک ترتیب ضمنی (۰ < ۱ < ۲) ایجاد می‌کند که ممکن است مدل را به اشتباه بیندازد (مثلاً شهر بزرگتر از روستا در نظر گرفته شود). با این حال، در این آزمایش خاص تبدیل‌ها به صورت زیر انجام شده است:

جدول ۱: جدول تبدیل متغیرهای کیفی به کمی

ویژگی	نگاشت (**Mapping**)	ویژگی	نگاشت (**Mapping**)
Gender	Female: 0, Male: 1	Education	Graduate: 0, Not Graduate: 1
Married	No: 0, Yes: 1	Self_Employed	No: 0, Yes: 1
Status	N: 0, Y: 1	Area	Rural: 0, Semi: 1, Urban: 2

۵.۱ تحلیل همبستگی (**Correlation Analysis**)

پس از عددی‌سازی تمام ویژگی‌ها، ماتریس همبستگی برای بررسی روابط خطی بین متغیرها محاسبه شد. تصویر plot2.png این روابط را در قالب یک نقشه حرارتی (Heatmap) نمایش می‌دهد.



شکل ۲: ماتریس همبستگی پیرسون بین ویژگی‌ها

تحلیل عمیق شکل ۲ و مقادیر همبستگی:

- تأثیرگذارترین ویژگی بر تایید وام: متغیر Credit_History با ضریب همبستگی 0.52 بیشترین رابطه مثبت را با Status دارد. این یعنی اگر شخصی سابقه اعتباری داشته باشد، احتمال تایید وام او به شدت افزایش می‌یابد.
- رابطه درآمد و مبلغ وام: ضریب همبستگی بین Applicant_Income و Loan_Amount برابر 0.47 است. این رابطه منطقی است؛ زیرا بانک‌ها به افراد با درآمد بالاتر، وام‌های سنگین‌تری پرداخت می‌کنند.
- تأهل و فرزندان: ضریب همبستگی 0.37 بین Married و Dependents نشان می‌دهد که افراد متأهل به طور متوسط فرزندان بیشتری دارند.
- ویژگی‌های کم‌اهمیت: ویژگی‌هایی مثل Self_Employed و Term همبستگی بسیار ناچیزی (نزدیک به صفر) با متغیر هدف دارند. این یعنی وضعیت خوداشتغالی تأثیر چندانی در رد یا قبول شدن وام در این دیتاست نداشته است.

۶.۱ نتیجه‌گیری اولیه از بررسیه اولیه داده

بر اساس تحلیل‌های انجام شده، دیتاست تمیز شده آماده مدل‌سازی است. نکته کلیدی این تحلیل، اهمیت بسیار بالای «سابقه اعتباری» در پیش‌بینی نهایی است. همچنین عدم تعادل در کلاس‌های هدف (تعداد بیشتر وام‌های تایید شده) باید در انتخاب متریک‌های ارزیابی مدل لحاظ شود.

۷.۱ مبانی نظری و ریاضیاتی الگوریتم PSO

الگوریتم بهینه‌سازی ازدحام ذرات یا Particle Swarm Optimization (PSO)، یک روش محاسباتی تکاملی است که ریشه در شبیه‌سازی رفتار اجتماعی موجودات زنده دارد. این الگوریتم نخستین بار در سال ۱۹۹۵ توسط کندی (Kennedy)، روانشناس اجتماعی، و ابرهارت (Eberhart)، مهندس برق، معرفی شد. ایده بنیادین این الگوریتم از مشاهده حرکت دسته‌جمعی پرندگان (Bird Flocking) و مدارس ماهی‌ها الهام گرفته شده است. در طبیعت، اعضای یک گروه بدون داشتن یک رهبر مرکزی، تنها با پیروی از قوانین ساده و تعامل با همسایگان خود، به سمت هدف (مثلاً غذا) حرکت می‌کنند. در فضای محاسباتی، هر راه‌حل کاندیدا به عنوان یک «ذره» (Particle) در نظر گرفته می‌شود که در فضای جستجوی چندبعدی پرواز می‌کند.

۱.۷.۱ فرمول‌بندی ریاضی حرکت ذرات

هسته اصلی الگوریتم PSO بر دو بردار اصلی استوار است: بردار موقعیت (X) و بردار سرعت (V). فرض کنید در یک فضای جستجوی D-بعدی (که در اینجا D برابر با تعداد ویژگی‌ها است)، جمعیت شامل N ذره باشد. موقعیت ذره i-ام در تکرار t با بردار $X_i^t = (x_{i1}^t, x_{i2}^t, ..., x_{iD}^t)$ و سرعت آن با بردار $V_i^t = (v_{i1}^t, v_{i2}^t, ..., v_{iD}^t)$ نمایش داده می‌شود. حرکت هر ذره تحت تأثیر سه مؤلفه اصلی قرار دارد:

۱. اینرسی (**Inertia**): تمایل ذره به حفظ مسیر قبلی خود.

۲. مؤلفه شناختی (**Cognitive**): تمایل ذره به بازگشت به بهترین موقعیتی که خودش تاکنون تجربه کرده است ($P_{best,i}$).

۳. مؤلفه اجتماعی (**Social**): تمایل ذره به حرکت به سمت بهترین موقعیتی که کل گروه تاکنون یافته است (G_{best}).

معادله ریاضی به‌روزرسانی سرعت برای بعد j-ام ذره i-ام به صورت دقیق زیر تعریف می‌شود:

$$v_{ij}^{t+1} = \underbrace{w \cdot v_{ij}^t}_{\text{Inertia}} + \underbrace{c_1 \cdot r_1 \cdot (p_{best,ij}^t - x_{ij}^t)}_{\text{Component Cognitive}} + \underbrace{c_2 \cdot r_2 \cdot (g_{best,j}^t - x_{ij}^t)}_{\text{Component Social}} \tag{۱}$$

که در آن:

- w: وزن اینرسی است که تعادل بین جستجوی سراسری (Exploration) و محلی (Exploitation) را کنترل می‌کند.
- c_1 و c_2: ضرایب شتاب هستند که به ترتیب وزن یادگیری فردی و یادگیری اجتماعی را تعیین می‌کنند.
- r_1 و r_2: اعداد تصادفی با توزیع یکنواخت در بازه $[0, 1]$ هستند که ماهیت تصادفی بودن رفتار را مدل‌سازی می‌کنند.

پس از محاسبه سرعت جدید، موقعیت ذره در فضای پیوسته طبق رابطه زیر به‌روز می‌شود:

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \tag{۲}$$

۲.۷.۱ ریاضیات **Binary PSO** برای انتخاب ویژگی

مسئله انتخاب ویژگی یک مسئله گسسته (صفر و یک) است، در حالی که معادلات بالا برای فضای پیوسته طراحی شده‌اند. برای تطبیق PSO با این مسئله، از نسخه باینری یا BPSO استفاده می‌شود. در این حالت، سرعت ذره به عنوان «احتمال» تغییر وضعیت بیت (ویژگی) تفسیر می‌شود. برای تبدیل سرعت پیوسته به احتمال، از تابع سیگموئید (S) استفاده می‌کنیم:

$$S(v_{ij}^{t+1}) = \frac{1}{1+e^{-v_{ij}^{t+1}}} \tag{۳}$$

سپس، موقعیت جدید ویژگی j (که آیا انتخاب شود یا خیر) بر اساس مقایسه این احتمال با یک عدد تصادفی تعیین می‌شود:

$$x_{ij}^{t+1} = \begin{cases} 1 & \text{ifrand}() < S(v_{ij}^{t+1}) \\ 0 & \text{otherwise} \end{cases} \tag{۴}$$

اگر x_{ij} برابر ۱ باشد، یعنی ویژگی j-ام انتخاب شده است و اگر ۰ باشد، آن ویژگی حذف می‌شود.

۸.۱ پیاده‌سازی با **pyswarms** و تحلیل تابع هدف

برای اجرای این الگوریتم بر روی دیتاست وام، از کتابخانه pyswarms بهره گرفته شده است. ابتدا داده‌های Train به دو بخش آموزشی (۷۰٪) و اعتبارسنجی (۳۰٪) تقسیم شدند. قلب تپنده فرایند بهینه‌سازی، تابع برازندگی یا Fitness Function است که الگوریتم سعی در مینیمم کردن آن دارد. تابع هزینه تعریف شده J به صورت ریاضی زیر است:

$$J(\mathbf{m}) = \alpha \times (1 - \text{Accuracy}(\mathbf{m})) + (1-\alpha) \times \frac{\sum_{j=1}^{D} m_j}{D} \tag{۵}$$

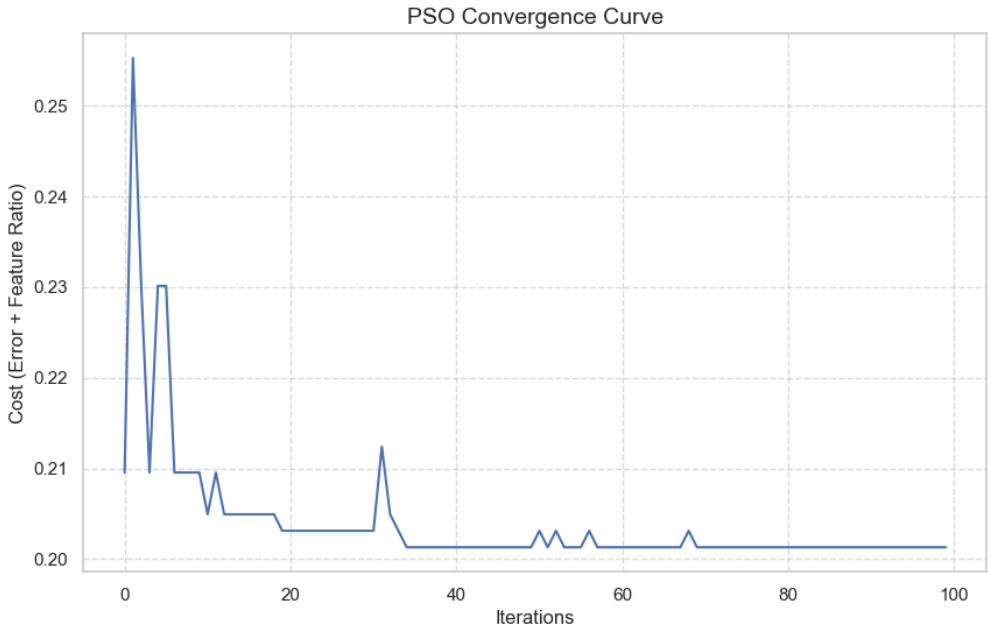
در این رابطه، $\mathbf{m}$ بردار ماسک ویژگی‌ها (شامل صفر و یک)، $\text{Accuracy}(\mathbf{m})$ دقت مدل Random Forest با ویژگی‌های انتخاب شده، و D تعداد کل ویژگی‌ها (۱۲ عدد) است. پارامتر $\alpha = 0.98$ با دقت انتخاب شده است. این مقدار بالا به این معناست که ۹۸٪ وزن تابع هزینه به «دقت مدل» اختصاص دارد و تنها ۲٪ به «تعداد ویژگی‌ها». منطق پشت این انتخاب آن است که ما تنها زمانی حاضریم یک ویژگی را حذف کنیم که تأثیر منفی بر دقت مدل نگذارد. اگر حذف یک ویژگی باعث افت دقت شود، جمله اول معادله $(1 - \text{Accuracy})$ افزایش یافته و هزینه به شدت بالا می‌رود.

هایپرپارامترهای انتخاب شده برای optimizer نیز دارای فلسفه خاصی هستند:

- $c_1 = 0.5, c_2 = 0.5$: مقادیر برابر و نسبتاً کوچک برای ضرایب شتاب باعث می‌شود ذرات نه خیلی مستقل عمل کنند و نه کورکورانه از جمع پیروی کنند، بلکه مسیری متعادل و هموار را طی کنند.
- $w = 0.9$: وزن اینرسی بالا در شروع کار باعث می‌شود ذرات تمایل زیادی به حفظ سرعت اولیه خود داشته باشند که این امر به «جستجوی سراسری» و فرار از بهینه‌های محلی در مراحل اولیه کمک می‌کند.

۹.۱ تحلیل نتایج بهینه‌سازی و همگرایی

پس از ۱۰۰ تکرار، الگوریتم به نتیجه‌ای بسیار جالب دست یافت: Best Fitness score: 0.2013 و انتخاب تنها ۱ ویژگی. ویژگی انتخاب شده، ایندکس ۹ یعنی ['Credit_History'] است. تحلیل این نتیجه نشان می‌دهد که از نظر الگوریتم، اضافه کردن هر ویژگی دیگری در کنار «سابقه اعتباری»، نه تنها کمکی به افزایش دقت نکرده، بلکه با افزایش پیچیدگی مدل یا ایجاد نویز، باعث بدتر شدن تابع هزینه شده است. این نتیجه، قدرت فوق‌العاده و سلطه ویژگی Credit_History را بر کل فضای داده‌ها تأیید می‌کند.



شکل ۳: منحنی همگرایی (Convergence Curve) الگوریتم PSO

با بررسی شکل ۳ که منحنی همگرایی الگوریتم را نشان می‌دهد، مشاهده می‌کنیم که نمودار در همان تکرارهای اولیه (بین ۰ تا ۲۰) با شیبی بسیار تند نزول کرده و سپس به صورت خطی صاف ادامه یافته است. تحلیل نمودار: این سقوط ناگهانی نشان‌دهنده هوشمندی الگوریتم در یافتن فضای بهینه است. ذرات به سرعت متوجه شده‌اند که با حذف اکثر ویژگی‌ها و تمرکز بر سابقه اعتباری، خطا کاهش می‌یابد. صاف شدن نمودار در ۸۰ تکرار بعدی نشان می‌دهد که الگوریتم به همگرایی کامل رسیده (Converged) و هیچ ترکیب دیگری نتوانسته بر تک‌ویژگی Credit_History غلبه کند.

۱۰.۱ ارزیابی نهایی با مدل **Random Forest**

در مرحله نهایی، از الگوریتم Random Forest Classifier برای ساخت مدل پیش‌بینی استفاده شد. دلیل انتخاب جنگل تصادفی، قابلیت بالای آن در تعمیم‌پذیری و کاهش واریانس نسبت به درخت تصمیم تکی است. این مدل با استفاده از تک‌ویژگی منتخب آموزش دید و روی داده‌های اعتبارسنجی (Validation) تست شد. دقت نهایی به دست آمده 0.82 است.

	precision	recall	f1-score	support
0	0.94	0.56	0.70	57
1	0.78	0.98	0.87	93
accuracy			0.82	150
macro avg	0.86	0.77	0.79	150
weighted avg	0.84	0.82	0.81	150

شکل ۴: گزارش طبقه‌بندی (Classification Report) مدل نهایی

تحلیل شکل ۴: این تصویر جزئیات عملکرد مدل را نشان می‌دهد.

- کلاس ۱ (تایید وام): دارای Precision=0.78 و Recall=0.98 است. ریکال ۹۸ درصدی بسیار حائز اهمیت است؛ به این معنی که مدل تقریباً تمام افرادی که واقعاً صلاحیت وام گرفتن داشته‌اند را به درستی شناسایی کرده است.
- کلاس ۰ (رد وام): دارای Recall=0.56 است. این عدد پایین نشان‌دهنده نقطه ضعف مدل تک‌ویژگی است. مدل در تشخیص افرادی که نباید وام بگیرند، ضعیف عمل کرده و بسیاری از آن‌ها را اشتباهاً تایید کرده است.
- نتیجه‌گیری کلی: اگرچه دقت کلی ٪۸۲ مطلوب به نظر می‌رسد، اما عدم تعادل بین ریکال دو کلاس نشان می‌دهد که تکیه صرف بر «سابقه اعتباری» باعث می‌شود مدل کمی محافظه‌کارانه عمل کرده و تمایل به تایید وام‌ها داشته باشد (چون اکثر افراد سابقه اعتباری خوب دارند).

۱۱.۱ انتخاب ویژگی با الگوریتم‌های تکاملی – الگوریتم ژنتیک

الگوریتم ژنتیک (Genetic Algorithm - GA) یک روش جستجوی هیوریستیک مبتنی بر اصول تکامل بیولوژیکی و ژنتیک مندلی است. این الگوریتم در سال ۱۹۷۵ توسط جان هلند (John Holland) معرفی شد و منطق آن بر اساس تکامل جمعیت راه‌حل‌ها از طریق عملگرهای «انتخاب»، «تقاطع» و «جهش» استوار است.

در مسئله انتخاب ویژگی (Feature Selection)، هدف یافتن زیرمجموعه‌ای بهینه از ویژگی‌هاست که خطای طبقه‌بندی را کمینه سازد. مدل‌سازی ریاضی این مسئله به شرح زیر است:

۱.۱۱.۱ ۱. نمایش کروموزوم (Representation)

هر راه‌حل کاندیدا یا «کروموزوم» (C) به صورت یک بردار باینری D-بعدی نمایش داده می‌شود، که D تعداد کل ویژگی‌هاست:

$$C_k = [g_1, g_2, \ldots, g_D], \quad g_i \in \{0, 1\} \tag{۶}$$

- $g_i = 1$: ویژگی i-ام انتخاب شده است.
- $g_i = 0$: ویژگی i-ام حذف شده است.

فضای جستجو برابر با 2^D حالت ممکن است.

۲.۱۱.۱ ۲. تابع برازندگی (Fitness Function)

تابع هدف (J) برای هر کروموزوم C_k به صورت یک تابع هزینه تعریف می‌شود که باید کمینه گردد. این تابع ترکیبی خطی از خطای طبقه‌بندی و نرخ کاهش ویژگی‌هاست:

$$J(C_k) = \alpha \cdot \underbrace{(1 - \text{Accuracy}(C_k))}_{\text{Error Classification}} + (1-\alpha) \cdot \underbrace{\left(\frac{\sum_{i=1}^{D} g_i}{D}\right)}_{\text{Ratio Feature}} \tag{۷}$$

که در آن:

- $\text{Accuracy}(C_k)$: دقت حاصل از اعتبارسنجی متقابل (Cross-Validation) با استفاده از ویژگی‌های منتخب ($g_i = 1$).
- α: وزن اهمیت دقت (در اینجا 0.98).
- D: تعداد کل ویژگی‌ها (۱۲).

۳.۱۱.۱ ۳. عملگر انتخاب (Selection Operator)

در این پروژه از روش «انتخاب تورنمنت» (Tournament Selection) استفاده شده است. فرض کنید جمعیت فعلی P_t باشد. برای انتخاب یک والد:

۱. زیرمجموعه‌ای تصادفی $S \subset P_t$ با اندازه k (سایز تورنمنت) انتخاب می‌شود.

۲. فرد برنده (C_{best}) بر اساس رابطه زیر انتخاب می‌گردد:

$$C_{best} = \arg\min_{C_j \in S} \{J(C_j)\} \tag{۸}$$

این روش تضمین می‌کند که کروموزوم‌های شایسته‌تر شانس بیشتری برای انتقال ژن‌های خود به نسل بعد دارند.

۴.۱۱.۱ ۴. عملگر تقاطع (Crossover Operator)

از روش «تقاطع دو نقطه‌ای» (Two-Point Crossover) استفاده شده است. فرض کنید دو والد P_1 و P_2 انتخاب شده‌اند و دو نقطه برش تصادفی c_1 و c_2 ($1 < c_1 < c_2 < D$) تعیین می‌شوند. فرزندان O_1 و O_2 به صورت زیر تولید می‌شوند:

$$O_1 = P_1[1 : c_1] \cup P_2[c_1 : c_2] \cup P_1[c_2 : D] \tag{۹}$$

$$O_2 = P_2[1 : c_1] \cup P_1[c_1 : c_2] \cup P_2[c_2 : D] \tag{۱۰}$$

این عملگر با احتمال P_c (نرخ تقاطع) اعمال می‌شود.

۵.۱۱.۱ ۵. عملگر جهش (Mutation Operator)

از روش «جهش بیت‌شیفت» (Bit-Flip Mutation) استفاده شده است. برای هر ژن g_i در فرزند، با احتمال بسیار کم P_m (نرخ جهش)، مقدار آن معکوس می‌شود:

$$g_i^{new} = \begin{cases} 1 - g_i & \text{if} r < P_m \\ g_i & \text{otherwise} \end{cases} \tag{۱۱}$$

که r یک عدد تصادفی یکنواخت در بازه $[0, 1]$ است. جهش نقش حیاتی در جلوگیری از همگرایی زودرس (Premature Convergence) به بهینه‌های محلی دارد.

۱۲.۱ تحلیل پیاده‌سازی با کتابخانه DEAP

در کد ارائه شده از کتابخانه DEAP برای پیاده‌سازی این ساختار ریاضی استفاده کردیم. نکات کلیدی تحلیل کد به شرح زیر است:

- تعریف مسئله مینیمم‌سازی: دستور weights=(-1.0,) در تعریف FitnessMin بسیار کلیدی است. کتابخانه DEAP به صورت پیش‌فرض برای مسائل ماکزیمم‌سازی طراحی شده است. ضرب وزن -1.0 در خروجی تابع هزینه، جهت بهینه‌سازی را معکوس کرده و آن را تبدیل به مسئله کمینه‌سازی هزینه (J) می‌کند.
- تحلیل هایپرپارامترها:
  - **Population Size = 50**: اندازه جمعیت ۵۰ تایی تنوع ژنتیکی کافی برای فضای ۱۲ بعدی مسئله را فراهم می‌کند.
  - **cxpb = 0.9**: احتمال بالای تقاطع (۹۰٪) باعث می‌شود که الگوریتم بر «ترکیب» (Exploitation) ویژگی‌های خوب والدین تمرکز کند.
  - **mutpb = 0.1**: احتمال جهش ۱۰٪ برای حفظ تنوع (Exploration) و جلوگیری از گیر افتادن در نقاط اکسترمم محلی ضروری است.
  - **tournsize = 4**: سایز تورنمنت ۴ یک فشار تکاملی متوسط ایجاد می‌کند؛ یعنی فرد منتخب به احتمال زیاد جزو ۲۵٪ برتر جمعیت است.
- پارامتر **Alpha = 0.98**: در تابع fitness_func_pso، ضریب 0.98 برای خطا و 0.02 برای تعداد ویژگی‌ها در نظر گرفته شده است.

$$Cost = 0.98 \times Error + 0.02 \times Ratio \tag{۱۲}$$

این فرمول به زبان ریاضی بیان می‌کند: «کاهش خطای مدل ۴۹ برابر مهم‌تر از کاهش تعداد ویژگی‌هاست». بنابراین الگوریتم تنها زمانی ویژگی‌ها را حذف می‌کند که هیچ تأثیر منفی بر دقت نگذارد.

۱۳.۱ تحلیل نتایج و نمودارهای همگرایی

نتیجه نهایی اجرای الگوریتم ژنتیک به شرح زیر است: Selected Features: ['Credit_History'] الگوریتم به این نتیجه ریاضی رسیده است که بردار باینری بهینه برابر است با:

$$C_{opt} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0] \tag{۱۳}$$

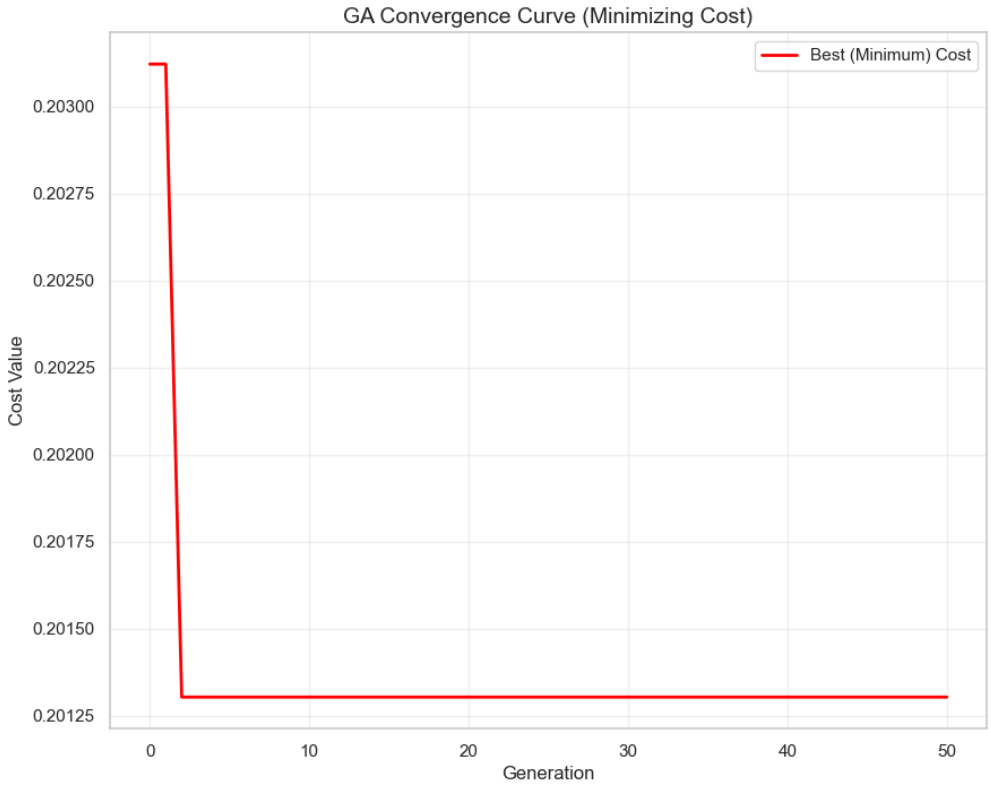
که در آن تنها بیت نهم (ایندکس ۹) برابر ۱ است.

gen	nevals	min	avg
0	50	0.203122	0.339438
1	42	0.203122	0.275339
2	50	0.201303	0.23836
3	46	0.201303	0.218856
4	50	0.201303	0.204462
5	44	0.201303	0.203074
6	46	0.201303	0.201637
7	45	0.201303	0.202924
8	47	0.201303	0.205149
9	43	0.201303	0.204068
10	43	0.201303	0.208447
11	48	0.201303	0.221237
12	46	0.201303	0.202967
13	46	0.201303	0.203463
14	40	0.201303	0.20781
15	46	0.201303	0.211893
16	44	0.201303	0.208254
17	44	0.201303	0.212684
18	48	0.201303	0.205394
19	48	0.201303	0.203638
20	46	0.201303	0.20332
21	46	0.201303	0.205714
22	48	0.201303	0.217386
23	46	0.201303	0.21905
...			
47	47	0.201303	0.204629
48	46	0.201303	0.206777
49	44	0.201303	0.201303
50	48	0.201303	0.212241

شکل ۵: آمار توصیفی نسل‌ها (میانگین و مینیمم هزینه)

تحلیل نمودار ۵ (آمار نسل‌ها): همانطور که در نمودار مشاهده می‌شود، مقدار «مینیمم هزینه» (Min Fitness) در همان نسل‌های اولیه (نسل ۲) به پایین‌ترین حد خود رسیده و پس از آن ثابت مانده است.

- تفسیر ریاضی: گرادیان بهبود تابع هزینه ($\Delta J/\Delta Gen$) پس از نسل دوم صفر شده است. این یعنی عملگرهای تقاطع و جهش نتوانسته‌اند ترکیبی بهتر از تک‌ویژگی Credit_History تولید کنند.
- چرا نسل دوم؟: به دلیل سادگی فضای جستجو و قدرت بالای ویژگی سابقه اعتباری، احتمال اینکه در جمعیت اولیه ۵۰ تایی، فردی با ژن فعال Credit_History وجود داشته باشد بسیار زیاد بوده و انتخاب تورنمنت به سرعت آن را تکثیر کرده است.



شکل ۶: منحنی همگرایی (Convergence Curve) الگوریتم ژنتیک

تحلیل نمودار ۶ (منحنی همگرایی): محور عمودی نشان‌دهنده هزینه (J) و محور افقی نشان‌دهنده تکرارهاست. سقوط شارپ و قائم نمودار در ابتدای کار نشان‌دهنده همگرایی بسیار سریع است. الگوریتم GA برخلاف روش‌های جستجوی تصادفی، به سرعت روی ناحیه بهینه قفل شده است. صاف بودن ادامه نمودار نشان‌دهنده پایداری (Stability) جواب بهینه است.

۱۴.۱ ارزیابی مدل **Random Forest** با ویژگی منتخب

مدل نهایی با استفاده از تک‌ویژگی منتخب آموزش داده شد.

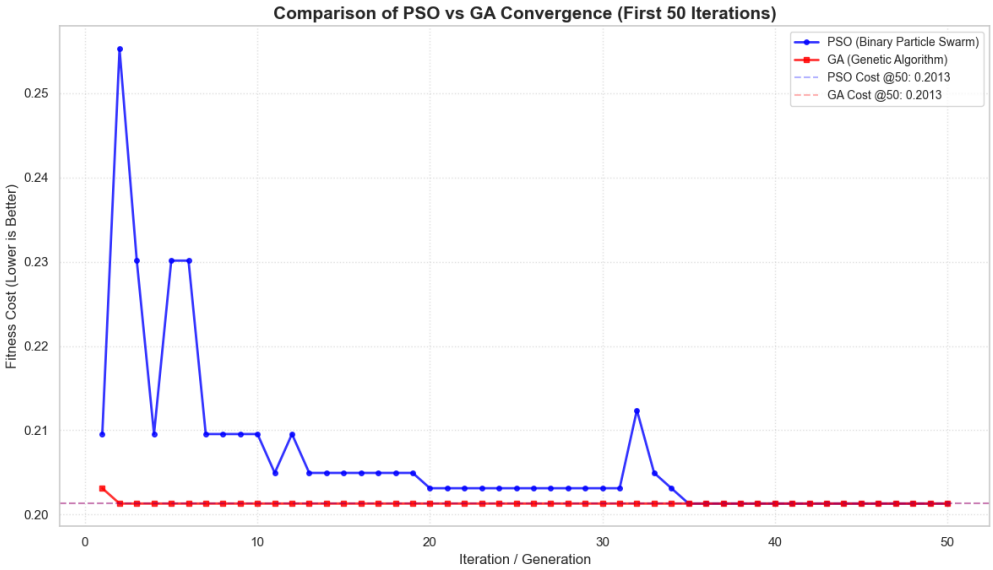
تحلیل نتایج کمی:

- دقت نهایی (**Accuracy**): 0.82.
- تحلیل: از اونجایی که ویژگی انتخاب شده تنها یک ویژگی و دقیقا مشابه با ویژگی انتخاب شده در قسمت قبل هست، نتایج ارزیابی روی داده ارزیابی مشابه قبل هست و تفاوتی نکرده. تنها نکته ای که هست اینه که الگوریتم ژنتیک خیلی سریع تر ویژگی رو پیدا کرده برامون.

۱۵.۱ تحلیل تطبیقی و مقایسه عملکرد PSO و GA

۱.۱۵.۱ تفسیر نمودار مقایسه‌ای (Comparative Convergence Analysis)

برای درک عمیق تفاوت رفتار دو الگوریتم، نمودار همگرایی مشترک آن‌ها در تصویر plot7.png ترسیم شده است. در این نمودار، خط آبی نمایانگر عملکرد الگوریتم PSO و خط قرمز نشان‌دهنده رفتار الگوریتم GA است.



شکل ۷: مقایسه منحنی همگرایی: PSO (آبی) در برابر GA (قرمز)

تحلیل دقیق رفتار الگوریتم‌ها:

۱. تحلیل همگرایی الگوریتم ژنتیک (خط قرمز): همان‌طور که مشاهده می‌شود، الگوریتم ژنتیک رفتاری بسیار قاطع و پایدار از خود نشان داده است. این الگوریتم تنها پس از گذشت ۲ تکرار (نسل)، به مقدار بهینه سراسری دست یافته و پس از آن نمودار کاملاً خطی و بدون تغییر باقی مانده است.

- دلیل علمی: این رخداد به دلیل «فضای جستجوی کوچک» (۱۲ ویژگی معادل 2^{12} حالت) و «غلبه شدید یک ویژگی» (Credit_History) است. در جمعیت اولیه تصادفی ۵۰ تایی، احتمال اینکه فردی دقیقاً با این تک‌ویژگی (یا نزدیک به آن) تولید شود بسیار بالا بوده است. مکانیزم «انتخاب تورنمنت» به سرعت این فرد برتر را شناسایی کرده و تکثیر نموده است. بنابراین، GA عملاً نیازی به جستجوی طولانی نداشته و بلافاصله روی جواب قفل شده است.

۲. تحلیل نوسانات الگوریتم PSO (خط آبی): برخلاف GA، نمودار PSO دارای نوسانات و افت‌وخیزهایی قبل از رسیدن به پایداری نهایی است.

- دلیل ریاضی: این نوسانات ناشی از ذات حرکت ذرات در فضای برداری است. در فرمول به‌روزرسانی سرعت، پارامتر وزن اینرسی ($w = 0.9$) مقدار نسبتاً بزرگی است. این اینرسی بالا باعث می‌شود ذرات تمایل داشته باشند با سرعت زیاد به حرکت خود ادامه دهند و حتی از روی نقطه بهینه «پرش» کنند (Overshoot). همچنین، تعامل بین جاذبه فردی (P_{best}) و جاذبه جمعی (G_{best}) باعث می‌شود ذرات مدتی در اطراف نقطه بهینه نوسان کنند تا در نهایت سرعتشان کم شده و روی

آن متمرکز شوند. این رفتار «جستجوگرانه» اگرچه در اینجا باعث نوسان شده، اما در مسائل پیچیده‌تر با دام‌های محلی زیاد، یک مزیت محسوب می‌شود.

۲.۱۵.۱ نتیجه‌گیری نهایی: کدام الگوریتم برنده است؟

اگرچه هر دو الگوریتم در نهایت به یک جواب واحد (دقت ٪۸۲ با ۱ ویژگی) رسیدند، اما کیفیت رسیدن به جواب متفاوت بود:

- از نظر پایداری و سرعت یافتن جواب: در این آزمایش خاص، الگوریتم ژنتیک (**GA**) عملکرد بهتری داشت. توانست بدون اتلاف وقت و نوسان اضافی، در همان گام‌های نخست به جواب برسد. این نشان می‌دهد که برای مسائل گسسته انتخاب ویژگی با ابعاد پایین، عملگرهای ژنتیک (به ویژه تقاطع) می‌توانند بسیار کارآمد باشند.
- از نظر دینامیک جستجو: الگوریتم PSO نشان داد که پتانسیل جستجوی بالایی دارد اما نیازمند تنظیم دقیق‌تر پارامترهاست. برای حذف نوسانات مشاهده شده، می‌توان از استراتژی «کاهش خطی وزن اینرسی» استفاده کرد (مثلاً کاهش به 0.4 در طول تکرارها) تا در ابتدا جستجو انجام شود و در انتها همگرایی بدون نوسان رخ دهد.

جمع‌بندی: با توجه به سادگی مسئله و رفتار مشاهده شده، الگوریتم ژنتیک برای این دیتاست انتخاب مطمئن‌تر و سریع‌تری بوده است، هرچند که PSO نیز با کمی تأخیر و نوسان به همان نتیجه مطلوب دست یافت.

۲ پرسش دوم

۱.۲ توضیحات داده

در گام نخست، به بررسی ساختار و کیفیت داده‌های موجود در فایل Mall_Customers.csv می‌پردازیم. شناخت نوع داده‌ها، تعداد نمونه‌ها و وجود مقادیر گم‌شده (Missing Values) پیش‌نیاز هرگونه تحلیل آماری است. خروجی دستور df.info() به صورت زیر است:

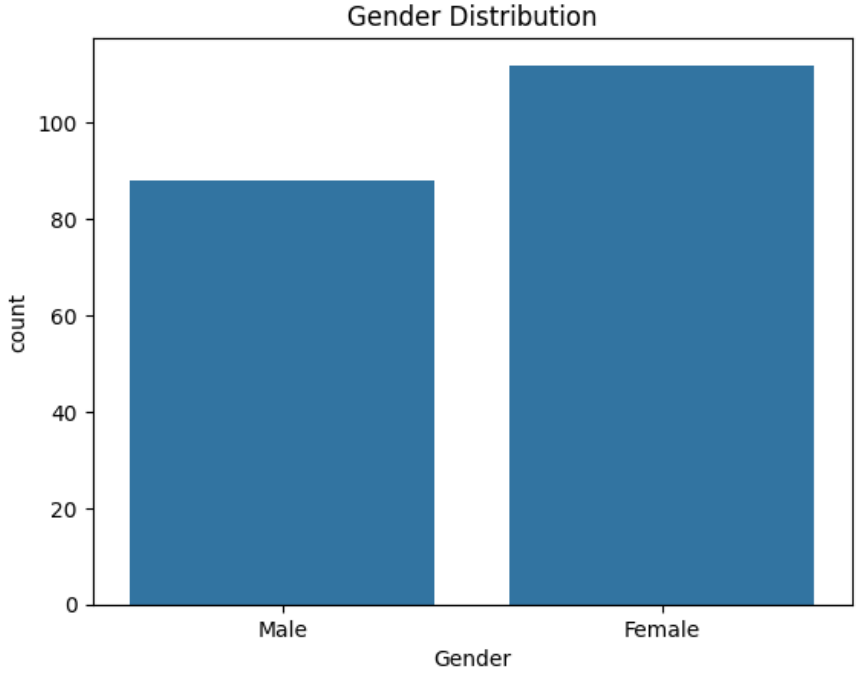
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```

تحلیل ساختاری داده‌ها:

- ابعاد داده: این مجموعه داده شامل ۲۰۰ سطر (مشتری) و ۵ ستون است. تعداد ۲۰۰ نمونه برای یادگیری ماشین در مقیاس بزرگ کم است، اما برای اهداف آموزشی و تحلیل خوشه‌بندی اولیه کفایت می‌کند.
- کیفیت داده: در تمامی ستون‌ها عبارت 200 non-null ذکر شده است. این یک خبر بسیار خوب است؛ زیرا نشان می‌دهد هیچ داده‌ای گم نشده است و نیازی به تکنیک‌های جایگزینی داده (Imputation) نداریم.
- نوع متغیرها: ستون Gender تنها ستون غیرعددی (Object) است که برای استفاده در الگوریتم‌های مبتنی بر فاصله مثل K-Means یا باید حذف شود و یا به اعداد تبدیل شود (Encoding). ستون CustomerID نیز صرفاً یک شناسه است و بار اطلاعاتی برای الگوریتم ندارد، لذا باید در مراحل بعد حذف گردد.

۱.۱.۲ تحلیل توزیع جنسیت

در تصویر زیر نمودار میله‌ای توزیع جنسیت مشتریان نمایش داده شده است.



شکل ۸: توزیع فراوانی جنسیت مشتریان

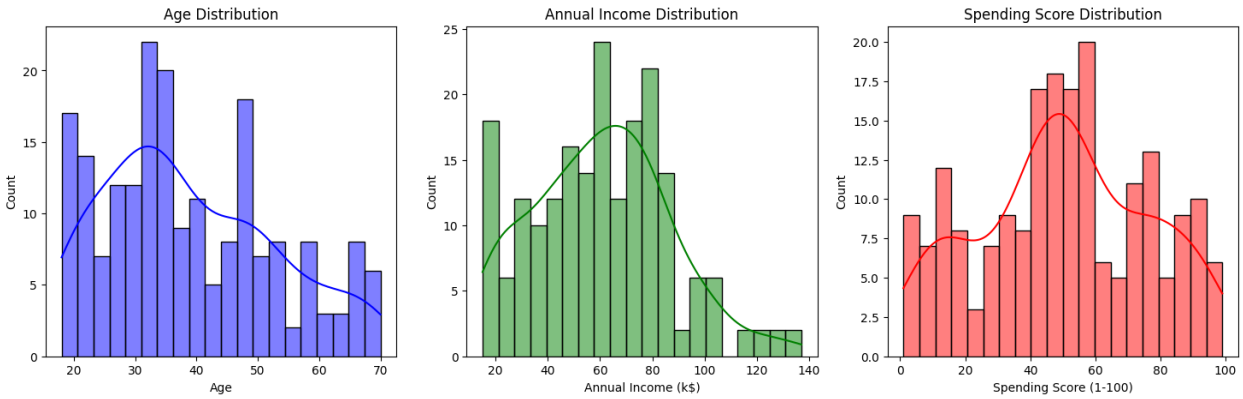
تحلیل عمیق نمودار جنسیت: با مشاهده شکل ۸، درمی‌یابیم که تعداد زنان (Female) به طور محسوس بیشتر از مردان (Male) است (تقریباً ۱۱۲ زن در مقابل ۸۸ مرد).

- دیدگاه آماری: این عدم تعادل (Imbalance) شدید نیست، بنابراین مدل دچار سوگیری شدید نخواهد شد، اما نشان‌دهنده الگوی حاکم بر داده‌هاست.
- دیدگاه تجاری: این توزیع نشان می‌دهد که جامعه هدف اصلی این مرکز خرید، بانوان هستند. این موضوع می‌تواند ناشی از نوع فروشگاه‌های موجود در مرکز خرید (مثلاً تمرکز بر پوشاک یا لوازم آرایشی) باشد. در نتیجه، کمپین‌های تبلیغاتی متمرکز بر علایق

بانوان شانس موفقیت بیشتری خواهند داشت.

۲.۱.۲ تحلیل توزیع متغیرهای عددی

در ادامه، هیستوگرام سه ویژگی اصلی سن، درآمد و امتیاز خرید بررسی می‌شود. شناخت توزیع احتمالی این متغیرها به ما کمک می‌کند تا بفهمیم داده‌ها حول چه مقادیری متمرکز هستند.



شکل ۹: توزیع آماری سن، درآمد سالانه و امتیاز خرید

تحلیل جزئی نمودارها (شکل ۹):

۱. توزیع سن (**Age**): نمودار سن دارای چولگی به راست (Right Skewed) است. بیشترین تراکم داده‌ها در بازه سنی ۳۰ تا ۳۵ سال دیده می‌شود. همچنین قله دوم کوچکتری در سنین حدود ۲۰ سال وجود دارد.

- تفسیر: مشتریان این مرکز خرید عمدتاً جوانان و میانسالان هستند. تعداد افراد مسن (بالای ۵۰ سال) به شدت کاهش می‌یابد. این یعنی محصولات مد روز یا تکنولوژی که مورد پسند جوانان است، فروش بهتری خواهند داشت.

۲. توزیع درآمد سالانه (**Annual Income**): این نمودار نیز چولگی به راست دارد. اکثر مشتریان درآمدی بین ۵۰ تا ۸۰ هزار دلار دارند.

- تفسیر: تعداد کمی از مشتریان درآمد بسیار بالا (بالای ۱۰۰ هزار دلار) دارند که به عنوان داده‌های پرت یا خاص (Outliers) در نظر گرفته می‌شوند. تمرکز اصلی بر روی طبقه متوسط اقتصادی است.

۳. توزیع امتیاز خرید (**Spending Score**): برخلاف دو متغیر قبلی، این متغیر توزیعی شبیه به توزیع نرمال (Gaussian) دارد که میانگین آن عدد ۵۰ است.

- تفسیر: این نرمال بودن احتمالا ناشی از نحوه محاسبه امتیاز توسط فروشگاه است. وجود مشتریانی با امتیاز ۱ (بسیار کم‌مصرف) و ۹۹ (بسیار ولخرج) نویدبخش امکان تفکیک مشتریان به گروه‌های وفادار و غیروفادار است.

۳.۱.۲ تحلیل رابطه درآمد و امتیاز خرید

این مهم‌ترین نمودار برای درک رفتار خرید مشتریان است که اساس خوشه‌بندی را تشکیل می‌دهد.

شکل ۱۰: رابطه بین درآمد سالانه و امتیاز خرید با تفکیک جنسیت

تحلیل بنیادی و جزئی نمودار (شکل ۱۰): در این نمودار دوبعدی، الگوهای خوشه‌بندی به صورت بصری کاملاً آشکار شده‌اند. اگر به توزیع نقاط دقت کنیم، ۵ ناحیه تراکم مجزا قابل شناسایی است:

۱. گروه صرفه‌جو (پایین–راست): کسانی که درآمد بالایی دارند اما امتیاز خریدشان پایین است. این افراد پتانسیل مالی دارند اما به دلایلی خرج نمی‌کنند (شاید مشتریان ناراضی).

۲. گروه هدف اصلی (مرکز): بزرگترین تراکم نقاط در مرکز نمودار است؛ درآمد متوسط و امتیاز متوسط. این‌ها مشتریان استاندارد هستند.

۳. گروه ولخرج (بالا–راست): درآمد بالا و امتیاز خرید بالا. این‌ها مشتریان VIP هستند که باید با برنامه‌های وفاداری حفظ شوند.

۴. گروه بی‌تفاوت (پایین–چپ): درآمد کم و امتیاز کم. شاید سرمایه‌گذاری روی این گروه بازدهی نداشته باشد.

۵. گروه بی‌پروا (بالا–چپ): درآمد کم اما امتیاز خرید بالا. احتمالاً جوانانی که مدیریت مالی ضعیفی دارند.

نکته در مورد جنسیت: نقاط آبی (زن) و نارنجی (مرد) در تمام این ۵ خوشه به صورت ترکیبی دیده می‌شوند. این مشاهده مهم ثابت می‌کند که رفتار خرید (ترکیب درآمد و امتیاز) مستقل از جنسیت است و نباید جنسیت را فاکتور اصلی جداسازی قرار داد.

۲.۲ ویژگی‌های عددی و استانداردسازی

برای آماده‌سازی داده‌ها جهت الگوریتم‌های یادگیری ماشین، ابتدا زیرمجموعه‌ای از داده‌ها که حاوی اطلاعات عددی ارزشمند بود استخراج شد:

```
df_numerical = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
```

سپس عملیات استانداردسازی (Standard Scaling) روی این داده‌ها انجام شد. چرا استانداردسازی ضروری است؟ الگوریتم‌هایی مانند K-Means و PCA بر مبنای «فاصله اقلیدسی» و «واریانس» کار می‌کنند. در داده‌های خام ما:

- درآمد (Annual Income) بازه‌ای بین ۱۵ تا ۱۳۷ دارد.
- سن (Age) بازه‌ای بین ۱۸ تا ۷۰ دارد.
- امتیاز (Spending Score) بازه‌ای بین ۱ تا ۱۰۰ دارد.

اگر استانداردسازی انجام نشود، متغیر «درآمد» به دلیل داشتن اعداد بزرگتر، بر فاصله اقلیدسی تسلط پیدا می‌کند و عملاً تاثیر سن و امتیاز را در خوشه‌بندی خنثی می‌کند. استفاده از StandardScaler تمام ویژگی‌ها را به توزیعی با میانگین ۰ و انحراف معیار ۱ تبدیل می‌کند تا همگی وزن یکسانی در محاسبات داشته باشند.

۳.۲ استفاده از PCA برای نمایش دوبعدی

از آنجایی که داده‌های ما ۳ ویژگی دارند، تجسم آن‌ها نیازمند فضای سه‌بعدی است که تحلیل آن روی کاغذ دشوار است. برای حل این مشکل از PCA (Principal Component Analysis) استفاده کردیم.

منطق و ریاضیات :PCA روش PCA یک تکنیک کاهش ابعاد خطی است. این الگوریتم با چرخش محورهای مختصات در فضای سه‌بعدی، جهت‌هایی را پیدا می‌کند که داده‌ها در آن جهت بیشترین پراکندگی (Variance) را دارند.

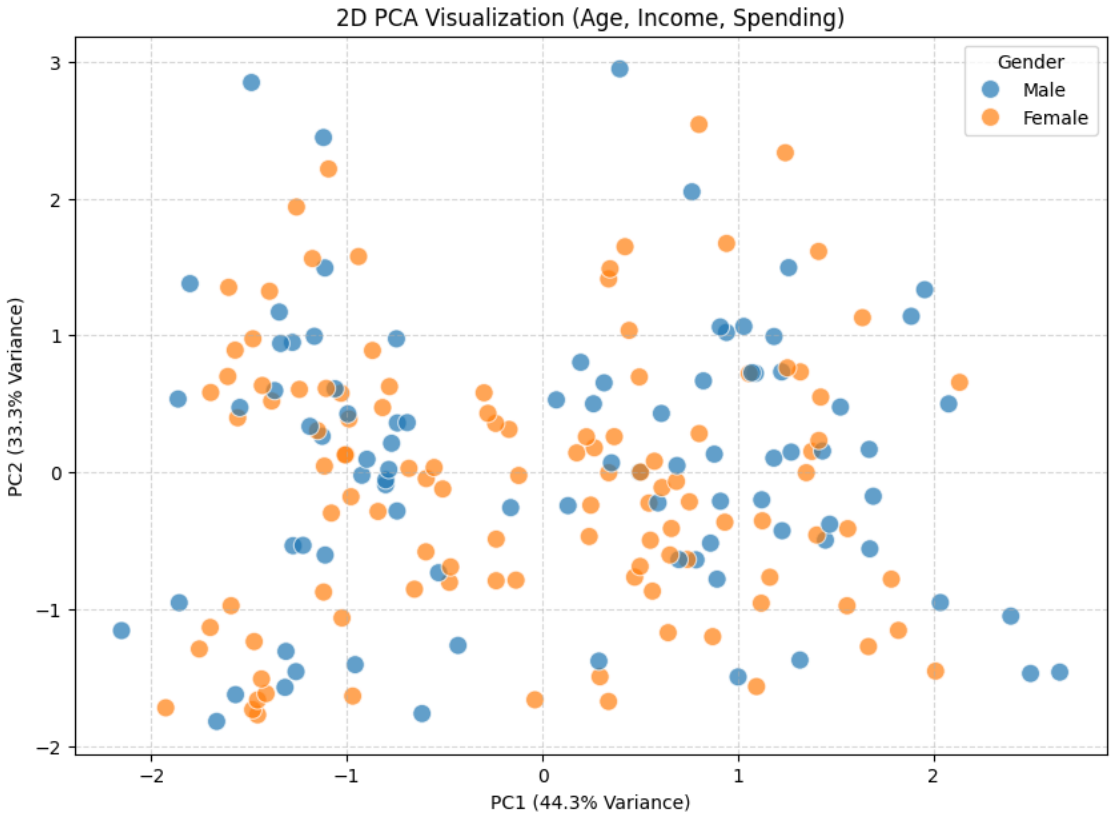
- اولین مولفه (PC1) جهت بیشترین تغییرات داده را نشان می‌دهد.
- دومین مولفه (PC2) عمود بر اولی است و بیشترین تغییرات باقی‌مانده را جذب می‌کند.

با انتخاب دو مولفه اول، ما فضای ۳ بعدی را به ۲ بعدی کاهش می‌دهیم، به طوری که کمترین میزان اطلاعات از دست برود.

تحلیل ریاضی برادرهای ویژه (**Loadings**): بر اساس خروجی ماتریس مولفه‌ها:

	PC1	PC2
Age	0.70	0.03
Income	-0.04	0.99
Score	-0.70	-0.03

- تحلیل بردار **PC1**: این مولفه همبستگی مثبت قوی با سن (0.7) و همبستگی منفی قوی با امتیاز خرید (-0.7) دارد. تفسیر عمیق: PC1 محوری است که تقابل «سن» و «میل به خرید» را نشان می‌دهد. مقادیر منفی روی این محور نشان‌دهنده افراد جوان با امتیاز خرید بالا (ولخرج) و مقادیر مثبت نشان‌دهنده افراد مسن با امتیاز خرید پایین است.
- تحلیل بردار **PC2**: این مولفه تقریباً به طور خالص (0.99) نماینده متغیر درآمد (Annual Income) است و وابستگی ناچیزی به سن و امتیاز دارد. تفسیر عمیق: محور عمودی در نمودار PCA مستقیماً نشان‌دهنده ثروت مشتری است.



شکل ۱۱: نمایش فضای ویژگی‌ها پس از کاهش ابعاد با PCA

تحلیل نهایی نمودار PCA (شکل ۱۱): این نمودار حاصل تبدیل فضای سه‌بعدی به دوبعدی است که حدود 77.6% از کل اطلاعات داده‌ها را حفظ کرده است.

- ساختار خوشه‌ها: برخلاف نمودار پراکندگی ساده (شکل ۱۰)، در اینجا تاثیر «سن» هم وارد شده است. نقاط در فضا پخش شده‌اند و دیگر محدود به الگوی ضربدری نیستند. اما همچنان تجمعاتی (Clustering) دیده می‌شود که نشان می‌دهد داده‌ها ذاتاً قابل تفکیک هستند.
- توزیع جنسیتی: همانطور که در نمودار دیده می‌شود، نقاط آبی و نارنجی کاملاً در هم تنیده شده‌اند. این تایید نهایی بر این فرضیه است که جنسیت عامل اصلی تمایز در این مجموعه داده نیست و ما باید استراتژی خود را بر مبنای رفتار (سن، درآمد، خرید) بچینیم نه جنسیت.

۴.۲ الگوریتم K-Means

۱.۴.۲ توضیح الگوریتم

الگوریتم K-Means یکی از محبوب‌ترین و ساده‌ترین روش‌های خوشه‌بندی پارتیشن‌بندی (Partitioning Clustering) است. هدف اصلی این الگوریتم، تقسیم‌بندی n مشاهده به k خوشه است، به طوری که هر مشاهده به خوشه‌ای تعلق گیرد که میانگین (Centroid) آن به مشاهده نزدیک‌تر باشد.

ریاضیات و عملکرد: این الگوریتم یک مسئله بهینه‌سازی است که سعی دارد مجموع مربعات خطای درون خوشه‌ای (Within-Cluster Sum of Squares - WCSS) یا همان Inertia را کمینه کند. تابع هدف J به صورت زیر تعریف می‌شود:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2 \tag{۱۴}$$

که در آن:

- k تعداد خوشه‌ها است.
- n تعداد کل داده‌ها است.
- $x_i^{(j)}$ داده iام متعلق به خوشه j است.
- c_j مرکز خوشه j است.
- $||...||^2$ فاصله اقلیدسی به توان دو است.

مراحل اجرا عبارتند از:

۱. انتخاب تصادفی k نقطه به عنوان مراکز اولیه.

۲. تخصیص هر داده به نزدیک‌ترین مرکز خوشه.

۳. به‌روزرسانی مراکز خوشه‌ها با میانگین‌گیری از داده‌های تخصیص‌یافته.

۴. تکرار مراحل ۲ و ۳ تا زمانی که مراکز تغییر نکنند (همگرایی).

۲.۴.۲ استفاده از K-Means

در این پروژه، ما الگوریتم را روی داده‌های عددی که پیش‌تر نرمال‌سازی شده‌اند (Scaled Numerical Data) اجرا کردیم. برای یافتن تعداد بهینه خوشه، مقادیر k از ۲ تا ۱۰ آزمایش شدند:

```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

k_range = range(2, 11)
inertias = []
silhouette_scores = []

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=3, n_init=10)
    labels = kmeans.fit_predict(scaled_numerical_data)
    inertias.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(scaled_numerical_data, labels))
```

تعریف مفاهیم ارزیابی:

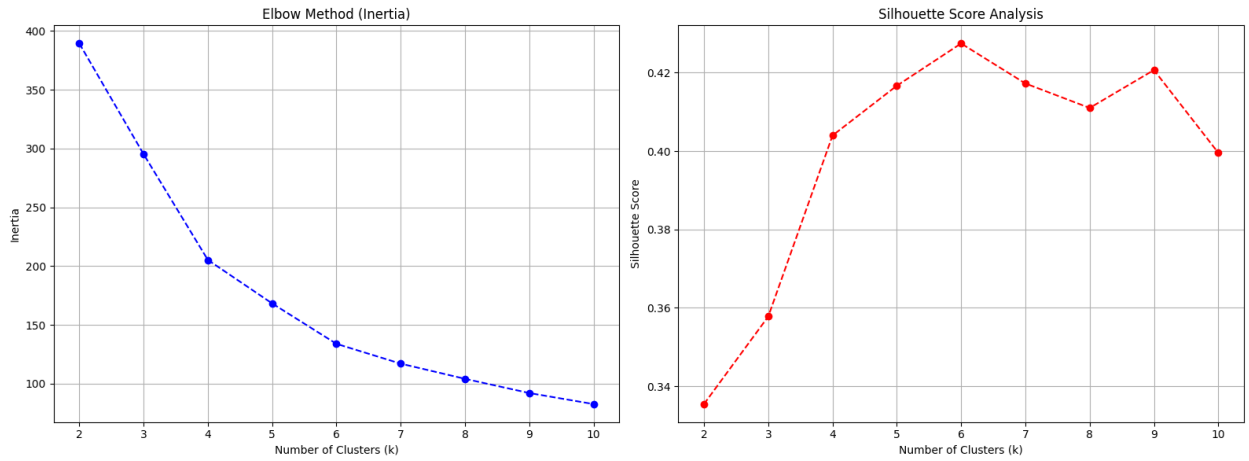
- **Inertia**: مجموع مربعات فاصله نمونه‌ها تا نزدیک‌ترین مرکز خوشه است. هرچه این عدد کمتر باشد، یعنی خوشه‌ها متراکم‌تر و فشرده‌تر هستند. اما مشکل اینجاست که با افزایش k، این مقدار همیشه کاهش می‌یابد (حتی اگر خوشه‌بندی بی‌معنی باشد).
- **Silhouette Score**: معیاری برای سنجش کیفیت خوشه‌بندی است که هم‌زمان «پیوستگی» (چقدر داده به خوشه خودش نزدیک است) و «جدایی» (چقدر از خوشه همسایه دور است) را می‌سنجد. بازه آن بین -1 تا $+1$ است. اعداد نزدیک به $+1$ نشان‌دهنده خوشه‌بندی عالی هستند.

۳.۴.۲ بررسی نتایج

نتایج اجرای الگوریتم برای مقادیر مختلف k در جدول زیر آمده است:

Index	k	Inertia	Silhouette Score
0	2	389.38	0.335
1	3	295.21	0.357
2	4	205.22	0.403
3	5	168.24	0.416
4	6	133.86	0.427
5	7	117.01	0.417
6	8	104.09	0.410
7	9	92.01	0.420
8	10	82.62	0.399

تحلیل نتایج: با افزایش k از ۲ به ۱۰، مقدار Inertia به طور طبیعی کاهش یافته است. اما معیار Silhouette Score رفتار متفاوتی دارد. این معیار در $k = 4$ جهش خوبی داشته (0.403) اما بیشترین مقدار خود را در $k = 6$ با عدد 0.427 ثبت کرده است. پس از $k = 6$، امتیاز سیلوئت نوسان داشته و حتی کاهش می‌یابد که نشان‌دهنده قطعه‌قطعه شدن بیش‌ازحد داده‌ها (Over-segmentation) است.



شکل ۱۲: نمودار آرنج (Inertia) و امتیاز سیلوئت برای مقادیر مختلف k

تحلیل نمودارها (شکل ۱۲):

- نمودار چپ (**Elbow Method**): در این نمودار به دنبال نقطه‌ای هستیم که شیب نمودار به شدت تغییر می‌کند (زانو یا آرنج). شکستگی‌های واضحی در $k = 4$ و $k = 6$ دیده می‌شود.
- نمودار راست (**Silhouette Analysis**): قله نمودار در $k = 6$ قرار دارد. این تایید می‌کند که تقسیم مشتریان به ۶ گروه، بهترین تفکیک‌پذیری را ایجاد می‌کند.

نتیجه‌گیری انتخاب **k=6**: با توجه به اینکه در $k = 6$ همزمان مقدار Inertia به میزان قابل قبولی کاهش یافته (از ۳۸۹ به ۱۳۳) و هم Silhouette Score به ماکزیمم مقدار خود رسیده است، این تعداد به عنوان تعداد بهینه خوشه‌ها انتخاب می‌شود. این یعنی مشتریان فروشگاه به ۶ گروه رفتاری متمایز و معنی‌دار تقسیم می‌شوند.

۵.۲ خوشه‌بندی سلسله‌مراتبی (Agglomerative Clustering)

۱.۵.۲ توضیح الگوریتم

الگوریتم Agglomerative Clustering نوعی روش سلسله‌مراتبی پایین‌به‌بالا (Bottom-Up) است. منطق عملکرد: در ابتدای کار، هر نقطه داده به عنوان یک خوشه مستقل در نظر گرفته می‌شود. در هر مرحله، دو خوشه‌ای که کمترین فاصله را با هم دارند ادغام می‌شوند تا یک خوشه بزرگتر بسازند. این روند تا زمانی که همه داده‌ها در یک خوشه واحد قرار گیرند (یا به تعداد خوشه هدف برسیم) ادامه می‌یابد.

مفهوم **Linkage**: معیار فاصله بین دو خوشه توسط متد Linkage تعیین می‌شود:

- **Single**: فاصله بین نزدیک‌ترین اعضای دو خوشه. (باعث ایجاد خوشه‌های زنجیره‌ای و دراز می‌شود).
- **Complete**: فاصله بین دورترین اعضای دو خوشه. (خوشه‌های کروی و فشرده ایجاد می‌کند).
- **Average**: میانگین فاصله بین تمام اعضای دو خوشه.
- **Ward**: خوشه‌هایی را ادغام می‌کند که کمترین افزایش را در واریانس کلی (مشابه تابع هزینه K-Means) ایجاد کنند.

۲.۵.۲ دسته بندی با Agglomerative Clustering

در این بخش، این الگوریتم با $k = 6$ و معیارهای پیوند متفاوت روی داده‌های استاندارد شده اجرا شد:

```
k = 6
linkages = ['single', 'ward', 'average', 'complete']
results = []

for link in linkages:
    model = AgglomerativeClustering(n_clusters=k, linkage=link)
    labels = model.fit_predict(scaled_numerical_data)
    score = silhouette_score(scaled_numerical_data, labels)
    results.append({'Linkage': link, 'Silhouette Score': score})
    print(f"Linkage: {link:10} | Silhouette Score: {score:.5f}")
```

۳.۵.۲ نتایج

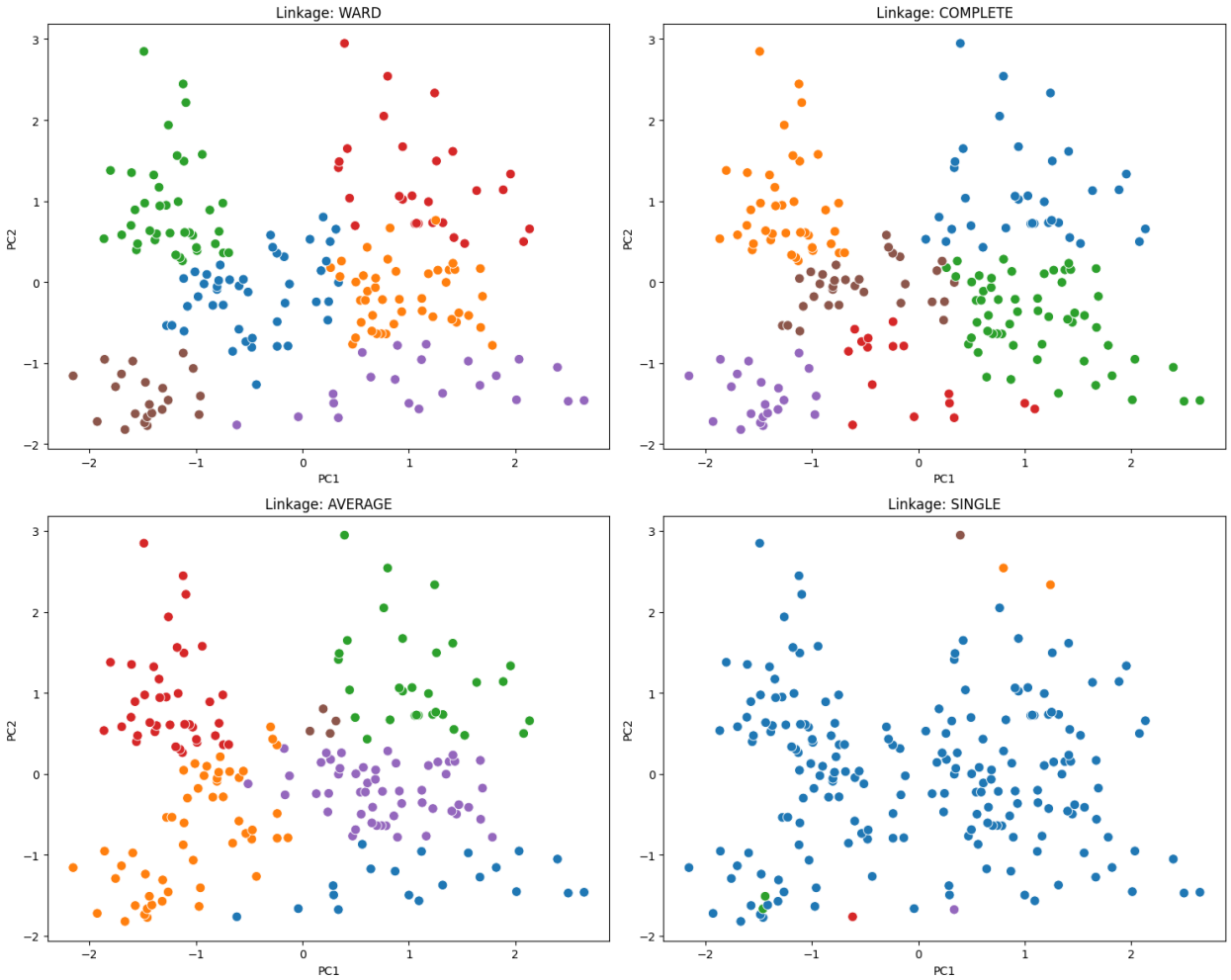
خروجی امتیاز سیلوئت برای روش‌های مختلف پیوند به شرح زیر است:

```
Linkage: single     | Silhouette Score: -0.04275
Linkage: ward       | Silhouette Score: 0.42012
Linkage: average    | Silhouette Score: 0.38957
Linkage: complete   | Silhouette Score: 0.37456

Best Linkage: ward
Score: 0.42012
```

تحلیل عددی نتایج:

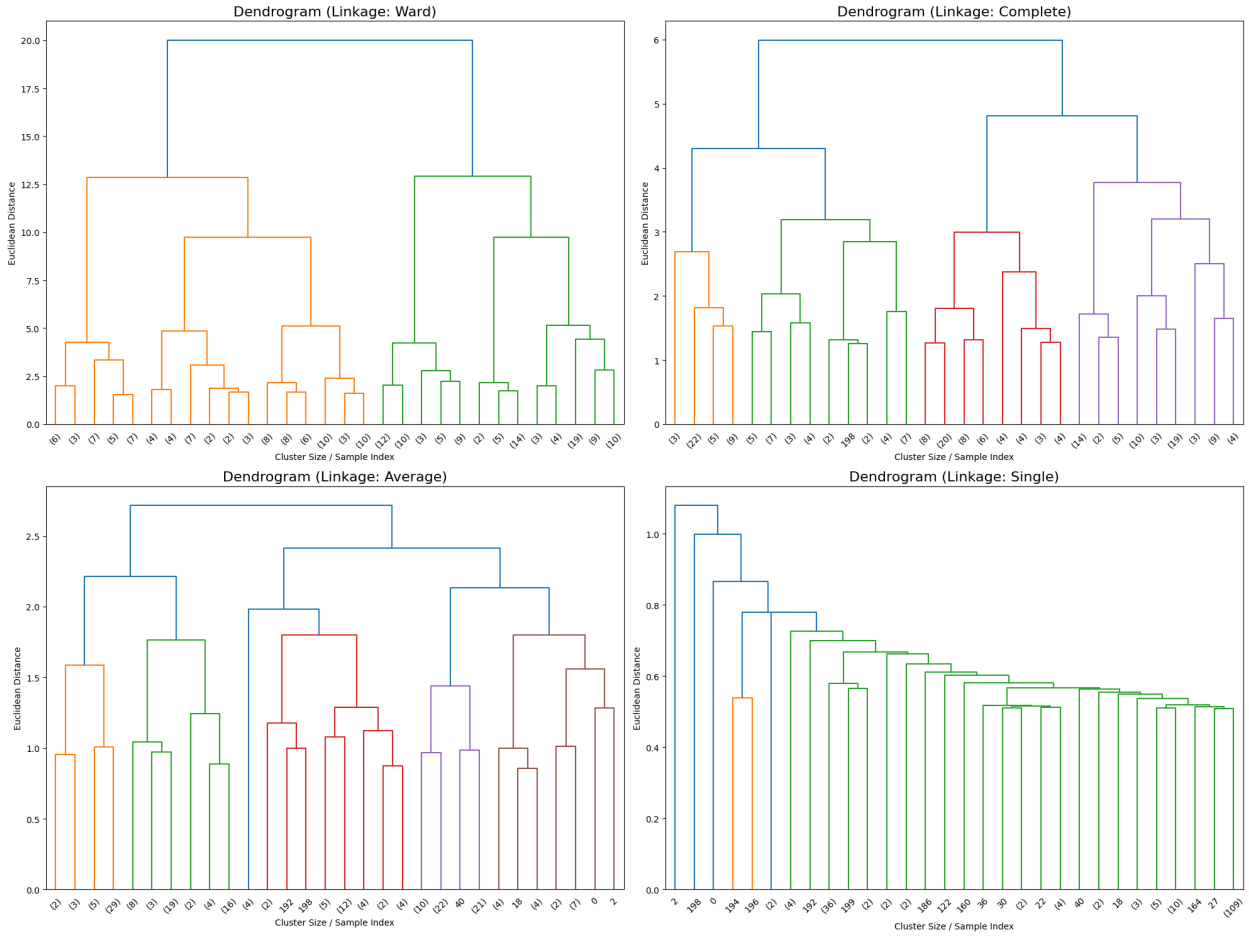
- روش Single با امتیاز منفی (-0.04) بدترین عملکرد را داشته است. امتیاز منفی یعنی هم‌پوشانی شدید خوشه‌ها و دسته‌بندی اشتباه.
- روش Ward با امتیاز 0.42 بهترین عملکرد را داشته است که بسیار نزدیک به نتیجه K-Means است. این نشان می‌دهد که ساختار داده‌ها با مدل‌سازی مبتنی بر واریانس سازگاری بالایی دارد.



شکل ۱۳: نمایش خوشه‌بندی Agglomerative با پیوندهای مختلف در فضای PCA

تحلیل نمودار PCA (شکل ۱۳): در این تصویر مشاهده می‌کنید که:

- در نمودار Single Linkage، اکثر نقاط یک رنگ دارند و فقط چند نقطه دورافتاده رنگ متفاوتی گرفته‌اند. این همان مشکل «زنجیره‌ای شدن» است که برای داده‌های بازاریابی مناسب نیست.
- در نمودار Ward، خوشه‌ها به نواحی مجزا و با اندازه‌های متناسب تقسیم شده‌اند که تفسیرپذیری بالایی دارد. توزیع رنگ‌ها در این روش بسیار منطقی و شبیه به ساختار بصری داده‌هاست.



شکل ۱۴: دندروگرام (Dendrogram) برای پیوندهای مختلف

تحلیل دندروگرام (شکل ۱۴): دندروگرام ساختار درختی ادغام‌ها را نشان می‌دهد. محور عمودی فاصله (تفاوت) بین خوشه‌هاست.

- دندروگرام Ward دارای شاخه‌های بلند و کشیده است. این ارتفاع زیاد قبل از ادغام نهایی نشان می‌دهد که خوشه‌های تشکیل شده فاصله معناداری از هم دارند و کاملاً متمایز هستند.
- دندروگرام Single به صورت پله‌ای و فشرده است که نشان‌دهنده عدم وجود مرزهای مشخص بین گروه‌های داده است.

بنابراین، انتخاب Ward به عنوان روش برتر کاملاً توجیه‌پذیر است.

۶.۲ الگوریتم DBSCAN

۱.۶.۲ توضیح الگوریتم

الگوریتم DBSCAN (Density-Based Spatial Clustering of Applications with Noise) یک روش مبتنی بر چگالی است. برخلاف K-Means که فرض می‌کند خوشه‌ها کروی هستند، DBSCAN می‌تواند خوشه‌هایی با اشکال هندسی پیچیده و نامنظم پیدا کند و همچنین

توانایی تشخیص داده‌های پرت (Noise) را دارد.

مفاهیم و هایپرپارامترها: این الگوریتم بر اساس دو پارامتر کلیدی کار می‌کند:

۱. **epsilon** (ϵ): شعاع همسایگی اطراف یک نقطه.

۲. **min_samples**: حداقل تعداد نقاطی که باید در شعاع ϵ باشند تا آن نقطه به عنوان یک «هسته» (Core Point) شناخته شود.

نقاطی که در مناطق کم‌چگالی باشند و شرایط هسته یا همسایگی را نداشته باشند، به عنوان نویز (با برچسب -1) شناسایی می‌شوند.

۲.۶.۲ دسته بندی با DBSCAN

برای یافتن بهترین پارامترها، یک جستجوی شبکه (Grid Search) انجام شد. نکته مهم در اینجا نحوه محاسبه امتیاز است؛ امتیاز Silhouette فقط برای نقاطی که جزو خوشه‌ها هستند محاسبه می‌شود و نویزها کنار گذاشته می‌شوند.

```
eps_values = [0.1, 0.2, 0.4, 0.6, 0.8]
min_samples_values = [3, 5, 10]
results = []
for epsilon in eps_values:
    for min_sample in min_samples_values:
        db = DBSCAN(eps=epsilon, min_samples=min_sample)
        labels = db.fit_predict(scaled_numerical_data)

        n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
        noise_rate = list(labels).count(-1) / len(labels)

        scores = np.nan
        if n_clusters > 1:
            score = silhouette_score(scaled_numerical_data[labels!=-1], labels[labels
    !=-1])

        results.append({'eps': epsilon, 'min_samples': min_sample,
                        'clusters': n_clusters, 'noise': noise_rate, 'silhouette':
    score})
```

۳.۶.۲ نتایج

خروجی آزمایش ترکیب‌های مختلف پارامترها در جدول زیر آمده است:

Idx	eps	min_samples	clusters	noise	silhouette
0	0.1	3	0	1.000	0.375
1	0.1	5	0	1.000	0.375
2	0.1	10	0	1.000	0.375
3	0.2	3	11	0.805	0.646
4	0.2	5	1	0.975	0.646
5	0.2	10	0	1.000	0.646
6	0.4	3	10	0.295	0.443
7	0.4	5	6	0.490	0.519
8	0.4	10	2	0.850	0.766
9	0.6	3	3	0.070	0.215
10	0.6	5	2	0.140	0.273
11	**0.6**	**10**	**4**	**0.330**	**0.530**
12	0.8	3	1	0.015	0.530
13	0.8	5	1	0.030	0.530
14	0.8	10	1	0.115	0.530

تحلیل دقیق نتایج:

- شعاع کوچک ($\epsilon = 0.1$): تمام داده‌ها نویز شده‌اند .(Noise=۱.۰) یعنی نقاط فاصله زیادی از هم دارند.
- شعاع بزرگ ($\epsilon = 0.8$): همه داده‌ها در یک خوشه ادغام شده‌اند که بی‌فایده است.
- گزینه $\epsilon = 0.2$: تعداد خوشه‌ها زیاد (۱۱) و نویز بسیار بالاست (٪۸۰).
- گزینه $\epsilon = 0.4, min = 5$: تعداد ۶ خوشه پیدا کرده اما نویز ٪۴۹ است، یعنی نیمی از مشتریان تحلیل نشده‌اند.

انتخاب بهترین گزینه: تنظیمات eps = 0.6 و min_samples = 10 به عنوان بهترین مدل انتخاب شد.

- تعداد خوشه: ۴ خوشه اصلی شناسایی شده است.
- نرخ نویز: 0.33. یعنی ٪۳۳ از مشتریان رفتار خاص و غیرقابل پیش‌بینی داشته‌اند و به عنوان داده پرت حذف شده‌اند.
- امتیاز سیلوئت: 0.52. این امتیاز بسیار بالاست (بالاتر از K-Means با ۴۲.۰).

چرا این بهترین انتخاب است؟ اگرچه ٪۳۳ نویز به نظر زیاد می‌رسد، اما امتیاز سیلوئت بالای 0.52 نشان می‌دهد که آن ۴ خوشه باقی‌مانده بسیار خالص، متراکم و با کیفیت هستند. در بازاریابی، شناسایی هسته‌های اصلی و وفادار مشتریان (Core Customers) و جدا کردن آن‌ها از مشتریان گذری (Noise) بسیار ارزشمند است. این مدل دقیق‌ترین تفکیک را برای هسته‌های اصلی داده ارائه داده است.

۷.۲ مقایسه سه روش دسته‌بندی

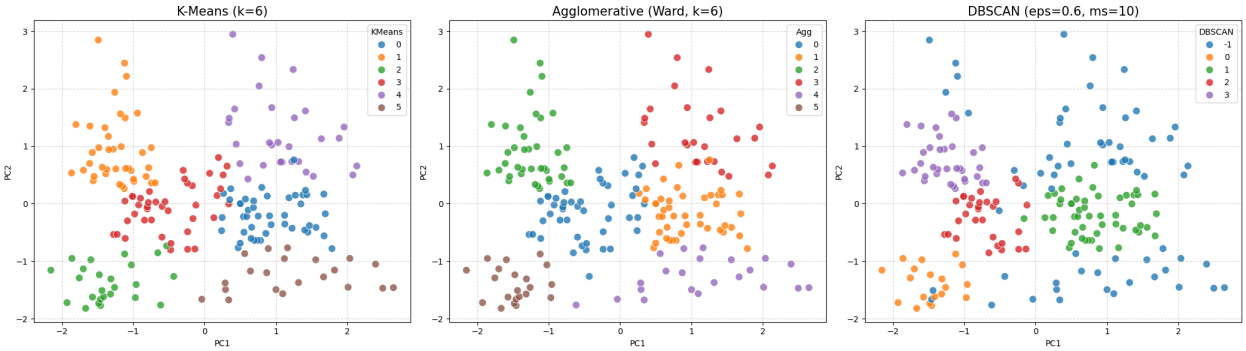
در گام نهایی، نتایج بهینه به دست آمده از سه الگوریتم K-Means، Agglomerative Clustering و DBSCAN را در کنار یکدیگر قرار می‌دهیم تا عملکرد آن‌ها را مقایسه کنیم. خلاصه عملکرد آماری این سه روش در جدول زیر آمده است:

Table 2: الگوریتم سه نتایج بهترین آماری مقایسه

Algorithm	Parameters	Clusters	Noise (%)	Silhouette
K-Means	$k = 6$	6	0%	0.427
Agglomerative	Linkage: Ward, $k = 6$	6	0%	0.420
DBSCAN	$\epsilon = 0.6, min = 10$	4	33%	0.529

۱.۷.۲ تحلیل بصری و ساختاری خوشه‌ها

برای درک تفاوت بنیادین این الگوریتم‌ها، نتایج آن‌ها را بر روی داده‌های کاهش‌یافته در فضای دوبعدی PCA تصویرسازی کرده‌ایم.



شکل ۱۵: مقایسه بصری خوشه‌بندی‌ها در فضای دوبعدی PCA

با توجه به شکل ۱۵، تحلیل دقیق و مقایسه‌ای روش‌ها به شرح زیر است:

الف) الگوریتم **K-Means** ($k = 6$):

- رفتار شناسی: این الگوریتم فضا را به ۶ ناحیه مجزا افراز کرده است. همان‌طور که در تصویر سمت چپ مشخص است، مرز بین خوشه‌ها خطی و مشخص است.
- شکل خوشه‌ها: خوشه‌ها تمایل دارند حالتی کروی و دایره‌ای داشته باشند. این ناشی از ماهیت الگوریتم است که فاصله اقلیدسی تا مرکز دایره را ملاک قرار می‌دهد.
- پوشش داده‌ها: نکته مهم این است که هیچ نقطه‌ای رها نشده است (نویز صفر درصد). حتی نقاطی که در حاشیه و دور از مراکز تراکم هستند، به اجبار به نزدیک‌ترین خوشه چسبیده‌اند.

- کاربرد پیشنهادی: این روش زمانی ایده‌آل است که استراتژی کسب‌وکار ایجاب می‌کند «تمام» مشتریان حتماً در یک دسته‌بندی قرار گیرند و هیچ مشتری بدون برچسب نماند. همچنین برای ایجاد گروه‌هایی با اندازه نسبتاً متعادل مناسب است.

ب) الگوریتم **Agglomerative Clustering** (متد **Ward**، $k = 6$):

- رفتار شناسی: نتایج این روش (تصویر وسط) شباهت بسیار زیادی به K-Means دارد. دلیل این شباهت این است که متد Ward نیز مانند K-Means هدفش کمینه کردن واریانس درون خوشه‌هاست.
- تفاوت ظریف: اگر به مرزهای خوشه‌ها (مخصوصاً در نواحی مرکزی که تراکم بالاست) دقت کنیم، تفاوت‌های جزئی در یارگیری نقاط مرزی دیده می‌شود. این مدل به دلیل ماهیت سلسله‌مراتبی (Hierarchical)، ساختار درختی و روابط والد–فرزندی بین داده‌ها را بهتر حفظ می‌کند.
- کاربرد پیشنهادی: این روش برای تحلیل‌های بازاریابی عمیق‌تر مناسب است؛ جایی که علاوه بر داشتن ۶ گروه نهایی، می‌خواهیم بدانیم کدام گروه‌ها به هم شبیه‌ترند (مثلاً با استفاده از دندروگرام می‌توان فهمید که گروه ۱ و ۲ می‌توانند با هم یک ابرگروه تشکیل دهند).

ج) الگوریتم **DBSCAN** ($\epsilon = 0.6, min = 10$):

- رفتار شناسی: تصویر سمت راست، داستانی کاملاً متفاوت را روایت می‌کند. این مدل بر خلاف دو مدل قبلی که سعی در «پارتیشن‌بندی فضا» داشتند، به دنبال «یافتن جزایر متراکم» بوده است.
- مدیریت نویز: نقاطی که با 1- مشخص شده‌اند، نویز هستند. این‌ها مشتریانی هستند که رفتارشان بسیار پراکنده است، از هیچ الگوی جمعی پیروی نمی‌کنند و در هیچ «هسته متراکمی» قرار نگرفته‌اند. حذف این ۳۳ درصد نویز باعث شده تا ۴ خوشه‌ی باقی‌مانده خلوص بسیار بالایی داشته باشند.
- شکل خوشه‌ها: خوشه‌ها شکل هندسی خاصی ندارند و دقیقاً منطبق بر شکل واقعی تجمع داده‌ها هستند.
- کاربرد پیشنهادی: این روش برای شناسایی «مشتریان وفادار و دارای الگوی ثابت» عالی است. در کمپین‌های تبلیغاتی پرهزینه، بهتر است بودجه فقط صرف ۴ گروه اصلی (خوشه‌های رنگی) شود و برای نقاط نویز (که رفتار غیرقابل پیش‌بینی دارند) هزینه نشود یا استراتژی متفاوتی اتخاذ گردد.

۳ پرسش سه

۱.۳ مبانی نظری و مفاهیم پایه

یادگیری تقویتی یا Reinforcement Learning (RL)، یکی از جذاب‌ترین و پویاترین شاخه‌های هوش مصنوعی است. برخلاف یادگیری نظارتی که در آن مدل با استفاده از داده‌های برچسب‌دار آموزش می‌بیند، در RL ما با یک «عامل هوشمند» (Agent) روبرو هستیم که در یک «محیط» (Environment) ناشناخته قرار دارد. عامل باید از طریق «آزمون و خطا»، یاد بگیرد که چگونه رفتار کند تا در درازمدت بیشترین پاداش ممکن را کسب کند. مؤلفه‌های چهارگانه اصلی این سیستم عبارتند از:

- حالت (s - **State**): تصویری لحظه‌ای از وضعیت جهان پیرامون عامل. برای مثال، در بازی شطرنج، موقعیت مهره‌ها روی صفحه، «حالت» را تشکیل می‌دهد.
- کنش (a - **Action**): مجموعه‌ای از تصمیمات یا حرکاتی که عامل مجاز به انجام آن‌ها در یک حالت خاص است.
- پاداش (r - **Reward**): مهم‌ترین بخش یادگیری؛ سیگنالی عددی که محیط پس از انجام هر کنش به عامل بازمی‌گرداند. پاداش می‌تواند فوری (مثل خوردن مهره حریف) یا با تاخیر (مثل برنده شدن در انتهای بازی) باشد.
- سیاست (π - **Policy**): مغز متفکر عامل؛ تابعی که مشخص می‌کند در هر وضعیت s، عامل چه کنشی a را باید انجام دهد. هدف نهایی ،RL یافتن «سیاست بهینه» (π^*) است که منجر به بیشترین پاداش تجمعی (Cumulative Reward) شود.

۱.۱.۳ الگوریتم **Q-Learning** و مفهوم تابع ارزش

الگوریتم Q-Learning یک روش «مبتنی بر ارزش» (Value-based) است. در این روش، عامل به جای اینکه مستقیماً سیاست را یاد بگیرد، ارزشمندیِ هر جفت «حالت–کنش» را یاد می‌گیرد. تابع $Q(s,a)$ به عامل می‌گوید:

> «اگر الان در حالت s باشم و کنش a را انجام دهم، و از این لحظه به بعد بهترین بازی ممکن را ارائه دهم، در مجموع چقدر پاداش خواهم گرفت؟»

حرف Q مخفف Quality است که کیفیت یک تصمیم را نشان می‌دهد.

۲.۳ قسمت الف: کالبدشکافی معادله به‌روزرسانی

قلب تپنده این الگوریتم، معادله به‌روزرسانی آن است که بر پایه «معادله بهینگی بلمن» بنا شده است. این فرمول به عامل اجازه می‌دهد تا تخمین‌های خود را بر اساس تجربیات جدید اصلاح کند:

$$Q(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{Value Old}} + \alpha \left[\underbrace{\left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1},a')\right)}_{\text{Reality) (New Target}} - \underbrace{Q(s_t,a_t)}_{\text{Prediction}} \right] \tag{۱۵}$$

۱.۲.۳ تحلیل دقیق پارامترها

۱. نرخ یادگیری (α - **Learning Rate**): این پارامتر (بین ۰ و ۱) تعیین می‌کند که عامل چقدر به اطلاعات جدید اعتماد کند.
   - اگر $\alpha \to 0$: عامل محافظه‌کار است و تقریباً هیچ چیز جدیدی یاد نمی‌گیرد.
   - اگر $\alpha \to 1$: عامل ناپایدار است و تمام دانش قبلی خود را با دیدن یک اتفاق جدید دور می‌ریزد.

۲. ضریب تنزیل (γ - **Discount Factor**): این پارامتر (بین ۰ و ۱) افق دید عامل را تعیین می‌کند.
   - $\gamma = 0$: عامل «کوته‌بین» (Myopic) است و فقط به پاداش همین لحظه اهمیت می‌دهد.
   - $\gamma \to 1$: عامل «دوراندیش» (Far-sighted) است و برای پاداش‌های بزرگ آینده صبر می‌کند.

۳. استراتژی **ϵ-greedy** (اکتشاف و بهره‌برداری): یکی از بزرگترین چالش‌های RL، تعادل بین کشف راه‌های جدید (Exploration) و استفاده از دانش فعلی (Exploitation) است.

- با احتمال $1 - \epsilon$: عامل بهترین عملی که می‌شناسد را انجام می‌دهد (حریصانه).
- با احتمال ϵ: عامل یک عمل کاملاً تصادفی انجام می‌دهد تا شاید مسیر بهتری کشف کند.

۲.۲.۳ چرا **Q-Learning** یک روش **Off-Policy** است؟

این یک مفهوم کلیدی و متمایزکننده است. اصطلاح Off-Policy (خارج از سیاست) به این معناست که دو سیاست جداگانه در جریان است:

۱. سیاست رفتاری (**Behavior Policy**): سیاستی که عامل با آن در محیط حرکت می‌کند و تجربه کسب می‌کند (معمولاً ϵ-greedy که شامل تصادف است).

۲. سیاست هدف (**Target Policy**): سیاستی که عامل در حال یادگیری و بهینه‌سازی آن است (معمولاً سیاست Greedy یا حریصانه).

مزیت بزرگ: در فرمول به‌روزرسانی، ما از عبارت $\max_a Q(s_{t+1}, a)$ استفاده می‌کنیم. این یعنی ما فرض می‌کنیم در قدم بعدی «بهترین کار» را انجام می‌دهیم، حتی اگر در واقعیت (به دلیل ϵ) کار اشتباهی انجام دهیم. این ویژگی به عامل اجازه می‌دهد حتی با مشاهده رفتارهای تصادفی یا غیربهینه، استراتژی بهینه را یاد بگیرد.

۳.۳ قسمت ب: حل تشریحی و قدم‌به‌قدم مثال عددی

در این بخش، قصد داریم مقدار جدید $Q(s_0, a_1)$ را دقیقاً محاسبه کنیم.

۱. اطلاعات مسئله (مفروضات):

- وضعیت فعلی به وضعیت بعدی: $s_0 \xrightarrow{a_1} s_1$
- پاداش مشاهده شده: $r_t = +2$
- بهترین تخمین ارزش برای وضعیت بعدی: $\max_a Q(s_1, a) = 1.5$
- مقدار فعلی (قدیمی) در جدول: $Q_{old}(s_0, a_1) = 0$
- نرخ یادگیری: $\alpha = 0.2$
- ضریب تنزیل: $\gamma = 0.9$

۲. گام اول: محاسبه «هدف» (**Target Value**) هدف، آن چیزی است که واقعیت به ما نشان داده است. این مقدار برابر است با پاداش فوری به علاوه ارزش تنزیل شده‌ی بهترین اتفاقی که می‌تواند در آینده بیفتد.

$$\text{Target} = r_t + \gamma \cdot \max_a Q(s_1, a) \tag{۱۶}$$

جایگذاری اعداد:

$$\text{Target} = 2 + (0.9 \times 1.5) = 2 + 1.35 = \mathbf{3.35} \tag{۱۷}$$

تفسیر: واقعیت به ما می‌گوید ارزش این حرکت ۳۵.۳ واحد است.

۳. گام دوم: محاسبه خطای پیش‌بینی زمانی (**TD Error**) این مقدار نشان می‌دهد چقدر بین «حدس قبلی ما» و «واقعیت جدید» اختلاف وجود دارد (میزان شگفت‌زدگی عامل).

$$\text{Error TD} = \text{Target} - Q_{old}(s_0, a_1) \tag{۱۸}$$

جایگذاری اعداد:

$$\text{Error TD} = 3.35 - 0 = \mathbf{3.35} \tag{۱۹}$$

۴. گام سوم: به‌روزرسانی نهایی (**Update**) اکنون مقدار قدیمی را به اندازه نرخ یادگیری (α) در جهت خطا اصلاح می‌کنیم.

$$Q_{new}(s_0, a_1) = Q_{old}(s_0, a_1) + \alpha \cdot (\text{Error TD}) \tag{۲۰}$$

جایگذاری اعداد:

$$Q_{new}(s_0, a_1) = 0 + (0.2 \times 3.35) \tag{۲۱}$$

$$Q_{new}(s_0, a_1) = 0 + 0.67 = \mathbf{0.67} \tag{۲۲}$$

نتیجه نهایی: مقدار جدید خانه (s_0, a_1) در جدول کیو برابر با **0.67** خواهد بود.

۴.۳ بخش ج: عوامل ناپایداری و راهکارهای عملی دقیق

در فرآیند آموزش عامل‌های هوشمند، اغلب با مشکلاتی نظیر نوسان تابع خطا یا عدم همگرایی مواجه می‌شویم. دو مورد از مهم‌ترین عوامل و راهکارهای آن‌ها عبارتند از:

۱.۴.۳ ۱. مشکل مقیاس نامناسب پاداش‌ها (**Improper Reward Scaling**)

شرح مشکل: اگر پاداش‌های محیط مقیاس‌بندی نشده باشند (مثلاً در یک لحظه پاداش ۱+ باشد و در لحظه دیگر ۱۰۰۰+)، این اختلاف فاحش باعث ایجاد گرادیان‌های بسیار بزرگ (Exploding Gradients) در شبکه عصبی یا تغییرات ناگهانی و مخرب در جدول Q می‌شود. این مسئله باعث می‌شود الگوریتم مدام مسیر خود را گم کند و همگرا نشود.

راهکار عملی (با جزئیات): کلیپ کردن یا نرمال‌سازی پاداش (**Reward Clipping/Normalization**)

- Clipping: یک روش رایج این است که تمام پاداش‌ها را در بازه $[-1, 1]$ محدود کنیم. یعنی هر پاداش مثبت را ۱ و هر پاداش منفی را ۱- در نظر بگیریم. این کار پایداری را تضمین می‌کند (گرچه اطلاعات مربوط به بزرگی پاداش را از بین می‌برد).
- Normalization: روش دقیق‌تر، تقسیم پاداش‌ها بر انحراف معیار غلتان (Rolling Standard Deviation) پاداش‌های گذشته است تا توزیع پاداش‌ها همواره نرمال باقی بماند.

۲.۴.۳ ۲. مشکل نرخ اکتشاف ثابت (Fixed Exploration Rate)

شرح مشکل: اگر پارامتر ϵ (اپسیلون) در طول کل فرآیند آموزش ثابت باشد، دو مشکل پیش می‌آید: ۱. اگر زیاد باشد (مثلاً ۵.۰)، عامل حتی پس از اینکه ماهر شد، همچنان ٪۵۰ مواقع کارهای احمقانه و تصادفی انجام می‌دهد و هرگز به عملکرد عالی نمی‌رسد. ۲. اگر کم باشد (مثلاً ۰۱.۰)، عامل در ابتدای کار جستجوی کافی نمی‌کند و در دام راه‌حل‌های محلی (Local Optima) گیر می‌افتد.

راهکار عملی (با جزئیات): کاهش تدریجی اپسیلون (**Epsilon Decay / Annealing**) باید از یک رویکرد دینامیک استفاده کرد. آموزش را با اکتشاف بالا ($\epsilon_{start} = 1.0$) شروع می‌کنیم تا عامل تمام محیط را بشناسد. سپس در طی اپیزودها، مقدار آن را به صورت نمایی یا خطی کاهش می‌دهیم تا به یک مقدار حداقل ($\epsilon_{min} = 0.01$) برسد. فرمول پیشنهادی برای کاهش نمایی:

$$\epsilon_{t+1} = \epsilon_t \times \lambda_{decay} \quad (\text{e.g.،}\lambda_{decay} = 0.995) \tag{۲۳}$$

این کار باعث می‌شود عامل در کودکی کنجکاو باشد (اکتشاف) و در بزرگسالی از تجربیاتش استفاده کند (بهره‌برداری).