دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

هوش محاسباتی

گزارش کتبی تمرین اول

اعضای گروه:

رضا چراخ

فرناز موحدی

اسفند ۱۴۰۳

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فصل اول
مبانی و مفاهیم الگوریتم ژنتیک (GA) و ویژگی‌های آن

۱-۱- مقدمه

فصل اول گزارش، به پاسخ به سوالات عنوان شده در مستند تمرین اول می‌پردازد و نیازمند داشتن دانش عمیقی از مفاهیم مطرح شده در کلاس و همچنین دیگر مفاهیم مرتبط است.

۱-۲- الگوریتم‌های تکاملی را توضیح دهید. دلایل اصلی برتری الگوریتم‌های تکاملی نسبت به الگوریتم‌های یادگیری تقویتی در برخی مسائل چیست؟

۱-۲-۱- الگوریتم‌های تکاملی

الگوریتم‌های تکاملی (Evolutionary Algorithms) زیرمجموعه‌ای از روش‌های بهینه‌سازی مبتنی بر هوش مصنوعی هستند که از فرآیندهای طبیعی تکامل زیستی مانند انتخاب طبیعی، جهش و ترکیب الهام گرفته‌اند. این الگوریتم‌ها با ایجاد و تکامل جمعیتی از راه‌حل‌های ممکن، به‌صورت تکراری به سمت یافتن راه‌حل‌های بهینه حرکت می‌کنند. هر عضو از جمعیت نشان‌دهنده یک راه‌حل ممکن برای مسئله است و با گذشت نسل‌ها، اعضای بهتر انتخاب شده و ترکیب می‌شوند تا بهبود یابند.

۱-۲-۲- دلایل اصلی برتری الگوریتم‌های تکاملی نسبت به الگوریتم‌های یادگیری تقویتی

در سال‌های اخیر، استراتژی‌های تکاملی به‌عنوان جایگزینی مقیاس‌پذیر برای یادگیری تقویتی مطرح شده‌اند. دلایل اصلی برتری الگوریتم‌های تکاملی نسبت به الگوریتم‌های یادگیری تقویتی در برخی مسائل عبارت‌اند از:

- **موازی‌سازی آسان‌تر:** الگوریتم‌های تکاملی به‌راحتی قابل موازی‌سازی هستند، زیرا ارزیابی هر عضو جمعیت مستقل از دیگران است. این ویژگی امکان استفاده مؤثرتر از منابع محاسباتی را فراهم می‌کند. در واقع در یادگیری تقویتی، روش‌های رایج مانند Q-learning و Policy Gradients نیاز

به ارتباط مداوم بین پردازنده‌ها دارند. اما الگوریتم‌های تکاملی فقط یک مقدار اسکالر (امتیاز کل) را بین پردازنده‌ها رد و بدل می‌کنند که باعث کاهش حجم ارتباطات و موازی‌سازی بهتر روی هزاران هسته پردازشی می‌شود.

- **عدم وابستگی به گرادیان و مشتق پذیری:** یادگیری تقویتی (RL) معمولاً نیاز به محاسبه گرادیان تابع ارزش و استفاده از روش‌های پس‌انتشار خطا (Backpropagation) دارد که در برخی موارد دشوار یا حتی غیرممکن است. برخلاف بسیاری از روش‌های یادگیری تقویتی که به محاسبه گرادیان نیاز دارند، الگوریتم‌های تکاملی بدون نیاز به اطلاعات گرادیان عمل می‌کنند. این امر آن‌ها را برای مسائلی که محاسبه گرادیان دشوار یا غیرممکن است و همچنین مسائل با محیط‌های ناپیوسته، مناسب می‌سازد.
- **مقاومت در برابر تأخیر پاداش و افق‌های زمانی بلند:** یکی از مشکلات اصلی یادگیری تقویتی (RL)در مسائل با افق زمانی بلند (Long-Horizon task) این است که اثر پاداش‌ها به مرور کاهش می‌یابد. این مشکل به دلیل تخفیف زمانی پاداش (Temporal Discounting) رخ می‌دهد که در روش‌هایی مانند Q-learning و Policy Gradient ضروری است اما باعث کاهش کارایی یادگیری در برخی مسائل می‌شود. در الگوریتم‌های تکاملی (Evolutionary Strategies - ES) به مجموع کل پاداش در یک اپیزود توجه می‌شود، نه به اینکه این پاداش چه زمانی اتفاق افتاده است. ESنیازی به تخفیف پاداش ندارد، زیرا فقط نتیجه‌ی نهایی کل اپیزود را در نظر می‌گیرد. درواقع تصمیمات اولیه و نهایی به یک اندازه در یادگیری نقش دارند. به این ترتیب مسائل بلندمدت و پاداش‌های دیرهنگام را بهتر مدیریت می‌کند، چرا که هیچ اطلاعاتی در طول زمان از بین نمی‌رود.
- **بهبود رفتارهای کاوش و پایداری در برابر بهینه‌های محلی:** الگوریتم‌های تکاملی به دلیل جستجوی سراسری در فضای راه‌حل‌ها، کمتر در بهینه‌های محلی گرفتار می‌شوند و احتمال یافتن بهینه سراسری را افزایش می‌دهند.
- **عدم نیاز به تقریب تابع ارزش و شبکه‌های پیچیده:** روش‌های یادگیری تقویتی معمولاً نیاز به مدل‌سازی تابع ارزش (Value Function Approximation) دارند که ممکن است منجر به بایاس و خطا در یادگیری شود. اما الگوریتم‌های تکاملی مستقیماً پارامترهای سیاست را بهینه می‌کنند و نیازی به تخمین تابع ارزش ندارند.

۱-۳- الگوریتم ژنتیک چیست و چه تفاوتی با سایر الگوریتم‌های تکاملی مانند برنامه‌نویسی تکاملی (EP) یا استراتژی‌های تکامل (ES) دارد؟

الگوریتم ژنتیک (Genetic Algorithm - GA) یکی از الگوریتم‌های تکاملی است که از اصول انتخاب طبیعی و ژنتیک الهام گرفته شده و برای حل مسائل بهینه‌سازی و جستجو به کار می‌رود. این الگوریتم با ایجاد یک جمعیت اولیه از راه‌حل‌های ممکن آغاز می‌شود و سپس طی چندین نسل با استفاده از عملگرهای ژنتیکی مانند ترکیب (Crossover) و جهش (Mutation) به تدریج راه‌حل‌های بهتری تولید می‌کند. کروموزوم‌های هر نسل بر اساس یک تابع هدف ارزیابی شده و بهترین‌ها برای تولید نسل بعدی انتخاب می‌شوند. این فرایند تا رسیدن به یک راه‌حل بهینه یا برآورده شدن یک معیار توقف ادامه پیدا می‌کند.

در مقایسه با سایر الگوریتم‌های تکاملی، برنامه‌نویسی تکاملی (Evolutionary Programming - EP) بیشتر بر جهش تأکید دارد و کمتر از ترکیب استفاده می‌کند. همچنین، نمایش راه‌حل‌ها در EP معمولاً عددی است و بیشتر برای یادگیری ماشین و بهینه‌سازی سیستم‌های دینامیکی به کار می‌رود. استراتژی‌های تکامل (Evolution Strategies - ES) نیز مشابه EP، عمدتاً از جهش و انتخاب بهره می‌برد و معمولاً در مسائل بهینه‌سازی عددی و پیوسته استفاده می‌شود. یکی از ویژگی‌های کلیدی ES استفاده از روش‌های خاص انتخاب مانند ($\mu + \lambda$-Selection) است که تفاوت‌هایی در نحوه انتخاب نسل بعدی نسبت به GA ایجاد می‌کند.

به طور کلی، الگوریتم ژنتیک انعطاف‌پذیرتر بوده و در مسائل متنوعی از بهینه‌سازی گسسته تا پیوسته به کار می‌رود، در حالی که EP و ES بیشتر در بهینه‌سازی سیستم‌های عددی و پارامتری مورد استفاده قرار می‌گیرند. تفاوت کلیدی دیگر این است که GA به ترکیب توجه بیشتری دارد، در حالی که EP و ES بیشتر به جهش متکی هستند.

۱-۴- عملیات جهش (Mutation) و ترکیب (Crossover) معمولاً چگونه روی رشته‌های بیتی (Bitstring) در یک الگوریتم ژنتیک اعمال می‌شوند؟ این عملگرها هنگام استفاده برای جایگشت‌ها یا سایر نمایش‌های غیر باینری چگونه باید تغییر یابند؟

۱-۴-۱- عملیات جهش (Mutation)

در الگوریتم‌های ژنتیک، جهش فرآیندی است که با ایجاد تغییرات تصادفی در کروموزوم‌ها، تنوع ژنتیکی را حفظ می‌کند و از همگرایی زودهنگام جلوگیری می‌کند. در نمایش رشته‌های بیتی، روش‌های مختلفی برای جهش وجود دارد:

- جهش تک نقطه‌ای ((Bit Flip Mutation: در این روش، یک یا چند بیت به‌طور تصادفی انتخاب

شده و مقدار آن‌ها تغییر می‌کند.

- جهش چند نقطه‌ای ((Multi-bit Mutation: چندین بیت به‌طور تصادفی انتخاب شده و مقدارشان تغییر می‌کند.
- جهش یکنواخت ((Uniform Mutation: هر بیت با احتمال مشخصی (مثلاً ۵٪) تغییر می‌کند.
- جهش معکوس‌سازی ((Inversion Mutation: یک بازه تصادفی از بیت‌ها انتخاب شده و مقادیر آن معکوس می‌شوند.

۱-۴-۲- عملیات ترکیب (Crossover)

ترکیب (Crossover) فرآیندی است که دو کروموزوم (والدین) را ترکیب کرده و یک یا چند فرزند جدید تولید می‌کند. روش‌های رایج برای رشته‌های بیتی عبارتند از:

- ترکیب تک‌نقطه‌ای(Single-Point Crossover) : یک نقطه تصادفی در کروموزوم انتخاب شده و بیت‌های دو والد بعد از آن نقطه با یکدیگر جابجا می‌شوند.
- ترکیب دو نقطه‌ای(Two-Point Crossover) : دو نقطه تصادفی انتخاب شده و بخش بین این دو نقطه بین والدین جابجا می‌شود.
- ترکیب یکنواخت(Uniform Crossover) : هر بیت فرزند با احتمال ۵۰٪ از یکی از والدین انتخاب می‌شود.

۱-۴-۳- عملیات جهش و ترکیب برای جایگشت‌ها و نمایش‌های غیر باینری

هنگام استفاده از این دو عملیات در جایگشت‌ها یا نمایش های دیگری که ترتیب یا نحوه قرار گیری ژن‌ها مهم و دارای محدودیت است، می‌توانیم برای عملیات crossover از عملگر cut-and-crossfill و همچنین برای جهش نیز از swap استفاده کنیم. در ادامه این موارد شرح داده می‌شوند:

- روش ترکیب cut-and-crossfill: این روش دارای دو بخش است:

برش (Cut):یک نقطه تصادفی در کروموزوم‌های والد انتخاب می‌شود. بخش قبل از این نقطه از والد اول در فرزند اول و از والد دوم در فرزند دوم کپی می‌شود.

پر کردن (Crossfill):باقی‌مانده ژن‌ها از والد دیگر گرفته می‌شود، اما به ترتیب ظاهر شدن در والد دوم و بدون تکرار ژن‌ها.

- جهش جابجایی Swap Mutation):) دو موقعیت تصادفی در جایگشت انتخاب شده و مقدارشان جابجا می‌شود.

۱-۵- چه خواص پایداری یا خصوصیات تغییرناپذیری (Invariance Properties) باید هنگام اجرای یک الگوریتم ژنتیک روی رشته‌های بیتی حفظ شوند؟ نمونه‌هایی ارائه دهید که نشان دهند این ویژگی‌ها چگونه بر کارایی و رفتار الگوریتم تأثیر می‌گذارند.

در الگوریتم‌های ژنتیکی که روی رشته‌های بیتی اجرا می‌شوند، برخی خصوصیات تغییرناپذیر باید حفظ شوند تا عملکرد و کارایی الگوریتم دچار انحراف نشود. این خصوصیات عبارت‌اند از:

- **حفظ اندازه جمعیت(Population Size Invariance)**

در هر نسل، اندازه جمعیت باید ثابت بماند مگر اینکه تغییر اندازه جمعیت به‌عنوان بخشی از استراتژی تکاملی طراحی شده باشد.

مثال تأثیر:

اگر در یک الگوریتم ژنتیک اندازه جمعیت تغییر کند (مثلاً برخی افراد حذف شوند یا تعداد بیش از حدی اضافه شوند)، توازن بین تنوع ژنتیکی و فشار انتخابی به‌هم می‌ریزد. اگر اندازه جمعیت کاهش یابد، ممکن است تنوع ژنتیکی به سرعت از بین برود، که منجر به همگرایی زودرس (Premature Convergence) شود. از طرفی در برخی موارد خاص، افزایش اندازه جمعیت می‌تواند از گیر افتادن در بهینه‌های محلی جلوگیری کند.

- **حفظ طول کروموزوم(Chromosome Length Invariance)**

هر کروموزوم (رشته بیتی) باید دارای طول ثابت باشد، مگر اینکه یک روش خاص مانند تکامل ساختاری استفاده شود که اجازه تغییر طول کروموزوم را بدهد.

مثال تأثیر:

اگر عملگرهای جهش و ترکیب باعث تغییر تصادفی طول رشته بیتی شوند، دیگر کروموزوم‌ها با هم سازگار نخواهند بود و مقایسه و ترکیب آن‌ها دچار مشکل می‌شود. البته در برخی کاربردهای خاص (مثلاً تکامل شبکه‌های عصبی)، تغییر طول کروموزوم می‌تواند مفید باشد.

- **حفظ احتمال جهش و ترکیب (Mutation & Crossover Probability Invariance)**

احتمالاتی که برای جهش (Mutation) و ترکیب (Crossover) تعیین می‌شوند باید در طول فرایند اجرا ثابت بمانند (مگر در روش‌هایی که احتمال را به‌صورت تطبیقی تغییر می‌دهند).

مثال تأثیر:

اگر احتمال جهش بیش از حد بالا رود، الگوریتم به‌جای جستجوی هدایت‌شده، به جستجوی تصادفی تبدیل می‌شود و کارایی کاهش می‌یابد. همچنین اگر احتمال جهش بسیار پایین باشد، تنوع ژنتیکی کاهش می‌یابد و الگوریتم ممکن است در بهینه‌های محلی گیر کند.

استراتژی تطبیقی: در برخی الگوریتم‌ها، احتمال جهش و ترکیب در طول زمان تغییر می‌کنند تا هم تنوع اولیه حفظ شود و هم همگرایی در مراحل نهایی بهتر انجام شود.

۱-۶- اگر یک الگوریتم ژنتیک برای رشته‌های بیتی با طول n برای یافتن جواب بهینه، زمان مورد انتظار $O(n3)$ را صرف کند، این مقدار چگونه با تعداد مراحل مورد انتظار برای جستجوی تصادفی با (توزیع یکنواخت) مقایسه می‌شود؟ علاوه بر این، چه عواملی می‌توانند بر عملکرد الگوریتم در عمل تاثیر بگذارند؟

"تعداد مراحل مورد انتظار" به میانگین تعداد ارزیابی‌هایی که انتظار داریم قبل از یافتن جواب بهینه انجام شود اشاره دارد. این مقدار معمولاً از طریق تحلیل احتمالاتی محاسبه شده و بیانگر تعداد دفعاتی است که الگوریتم باید اجرا شود تا به نتیجه مطلوب برسد.

۱-۶-۱- مقایسه دو روش: الگوریتم ژنتیک و جستجوی تصادفی

- الگوریتم ژنتیک با زمان مورد انتظار $O(n^3)$: یعنی به‌طور میانگین، الگوریتم ژنتیک در حداکثر حدود $O(n^3)$ مرحله، جواب بهینه را پیدا می‌کند. می‌دانیم هر مرحله شامل ارزیابی جمعیت، انتخاب، ترکیب (Crossover) و جهش (Mutation) است.
- این مقدار چندجمله‌ای (Polynomial) بوده که در مقایسه با جستجوی تصادفی، بسیار بهینه‌تر و سریع‌تر است.
- جستجوی تصادفی یکنواخت با زمان مورد انتظار $O(2^n)$: در این روش، هر رشته بیتی با احتمال برابر انتخاب می‌شود. چون فضای جستجو دارای 2^n ترکیب ممکن است، به‌طور میانگین حدود $O(2^n)$ تلاش لازم است تا به جواب بهینه برسیم.
- این مقدار نمایی (Exponential) است، یعنی با افزایش مقدار n، تعداد مراحل به‌شدت افزایش می‌یابد و این روش برای مسائل بزرگ غیرعملی و ناکارآمد است.

۱-۶-۲- عوامل موثر بر عملکرد الگوریتم ژنتیک در عمل

چندین عامل می‌توانند بر عملکرد واقعی الگوریتم ژنتیک تأثیر بگذارند:

- طراحی تابع برازش (Fitness Function Design)
  - تابع برازش باید به گونه‌ای تعریف شود که افراد باکیفیت‌تر را به‌درستی امتیازدهی کند. در غیر این صورت، ممکن است الگوریتم در بهینه‌های محلی گرفتار شود یا نرخ همگرایی کاهش یابد.
- نرخ جهش و ترکیب (Mutation & Crossover Rates)

- مقدار بهینه نرخ جهش (Mutation Rate) معمولاً کم ولی غیر صفر است تا از کاهش تنوع ژنتیکی جلوگیری کند. نرخ ترکیب (Crossover) باید متناسب با ساختار مسئله تنظیم شود، زیرا مقدار نامناسب می‌تواند تکامل را مختل کند.

- اندازه جمعیت (Population Size)
  - اگر اندازه جمعیت خیلی کوچک باشد، ممکن است تنوع ژنتیکی کم شده و الگوریتم در بهینه‌های محلی گیر کند. از طرف دیگر اگر اندازه جمعیت خیلی بزرگ باشد، ممکن است هزینه محاسباتی بالا رود و سرعت همگرایی کاهش یابد.
- نحوه انتخاب والدین (Selection Strategy)
  - روش‌های مختلف مانند انتخاب چرخ رولت(Roulette Wheel Selection) ، انتخاب تورنمنت (Tournament Selection) و انتخاب رتبه‌ای (Rank Selection) می‌توانند تأثیر زیادی بر سرعت همگرایی و پیشگیری از افتادن در بهینه‌های محلی داشته باشند.
- ساختار مسئله و نحوه نمایش داده‌ها (Problem Structure & Representation)
  - برخی مسائل ساختار جستجوی ساده‌تری دارند و الگوریتم ژنتیک به‌راحتی می‌تواند در آن‌ها جواب بهینه را پیدا کند. برخی مسائل دارای چشم‌اندازهای پیچیده‌ای در فضای جستجو هستند که باعث کاهش کارایی الگوریتم می‌شود.

فصل دوم
درک و حل مسائل با الگوریتم ژنتیک

۲-۱- مقدمه

فصل دوم گزارش، به حل تمرین‌های عنوان شده در مستند تمرین اول و پاسخ به سوالات این بخش‌ها می‌پردازد و نیازمند داشتن دانش عمیقی از مفاهیم مطرح شده در کلاس و همچنین دیگر مفاهیم مرتبط است.

۲-۲- مسئله فروشنده دوره گرد

مسئله فروشنده دوره گرد (TSP) را در نظر بگیرید، که هدف آن تعیین کوتاه ترین مسیر ممکن است که از هر مجموعه شهر داده شده دقیقاً یک بار بازدید کرده و به شهر مبدأ بازگردد . برای حل این مسئله با بکارگیری یک الگوریتم ژنتیک (GA) ، فرض کنید که هر ژن در یک کروموزوم نشان دهنده یک یال بدون جهت بین دو شهر است. به عنوان مثال، ژن 'TI' نشاندهنده یک اتصال مستقیم بین تهران و اصفهان است، و با توجه به فرض بدون جهت بودن، 'TI'معاد ل 'IT' در نظر گرفته می‌شود.

۲-۲-۱- اگر تعداد کل شهرها ۱۰ باشد، هر یک کروموزوم به چند ژن نیاز دارد؟

یک مسیر کامل شامل ۱۰ یال (اتصال بین دو شهر) خواهد بود، زیرا اگر ۱۰ شهر داشته باشیم، ۱۰ حرکت لازم است تا به نقطه‌ی شروع بازگردیم. در نتیجه هر کروموزوم به ۱۰ ژن نیاز دارد.

۲-۲-۲- الفبای الگوریتم (مجموعه ژن‌های منحصر به فرد) شامل چند ژن یکتاست؟

از آنجا که هر ژن نشان‌دهنده‌ی یک یال بدون جهت است، تعداد یال‌های ممکن بین ۱۰ شهر را محاسبه

می‌کنیم:

$$C(n,2) = \frac{(10 * 9)}{2} = 45$$

پس تعداد ژن‌های یکتا (مجموعه ژن‌های منحصر به فرد) برابر ۴۵ است.

۲-۳- الگوریتم ژنتیک با طول ثابت هشت ژن

فرض کنید یک الگوریتم ژنتیک از کروموزوم‌هایی به شکل $x=abcdefghx$ با طول ثابت هشت ژن استفاده می‌کند. هر ژن می‌تواند هر عددی بین 0 تا 9 باشد. مقدار برازش (**Fitness**) یک فرد/کروموزم x به صورت زیر محاسبه می‌شود:

$f(x) = (a+b) - (c+d) + (e+f) - (g+h)$

و فرض کنید که جمعیت اولیه شامل چهار فرد با کروموزوم‌های زیر باشد:

$x1$=65413532
$x2$=87126601
$x3$=23921285
$x4$=41852094

۲-۳-۱- برازندگی (Fitness) هر فرد/کروموزم را با نشان دادن تمام مراحل محاسبه کنید، و آ نها را به ترتیب از بیشترین مقدار برازش تا کمترین مرتب کنید.

$$f(x_2) = 15 - 3 + 12 - 1 = 23$$
$$f(x_1) = 11 - 5 + 8 - 5 = 9$$
$$f(x_3) = 5 - 11 + 3 - 13 = -16$$
$$f(x_4) = 5 - 13 + 2 - 13 = -21$$

۲-۳-۲- عملیات ترکیب (Crossover) زیر را انجام دهید:

۲-۳-۲-۱- دو فرد با بالاترین مقدار براز ش را با استفاده از ترکیب تک نقطه‌ای در نقطه میانی

ترکیب کنید.

$x_1 = 6541|3532$ ---------- $x_5 = 6541|6601$

$x_2 = 8712|6601$ ---------- $x_6 = 8712|3532$

۲-۳-۲-۲- دومین و سومین فرد برتر را با استفاده از ترکیب دو نقطه‌ای در نقاط bc و fg ترکیب کنید.

$x_1 = 65|4135|32$ ---------- $x_7 = 65|9212|32$

$x_3 = 23|9212|85$ ---------- $x_8 = 23|4135|85$

۲-۳-۲-۳- فرد اول و سوم برتر را با استفاده از ترکیب یکنواخت (Uniform Crossover) ترکیب کنید.

$x_2 = 8|7|1|2|6|6|0|1$ ---------- $x_9 = 8|3|1|2|6|2|5|0$

$x_3 = 2|3|9|2|1|2|8|5$ ---------- $x_{10} = 2|7|9|2|1|6|8|1$

در روش ترکیب یکنواخت از احتمال 0.5 برای انتخاب هر کدام از ژن‌ها از والد خود استفاده شده و فرض شده که به صورت یکی در میان هر ژن از یک والد انتخاب شده.

۲-۳-۳- فرض کنید جمعیت جدید شامل شش فرد حاصل از عملیات ترکیب در سوال قبل باشد. مقدار برازش این جمعیت جدید را محاسبه کنید و تمامی مراحل محاسبات را نشان دهید. آیا مقدار برازش کلی بهبود یافته است؟

$$f(x_5) = 11 - 5 + 12 - 1 = 17$$
$$f(x_6) = 15 - 3 + 8 - 5 = 15$$
$$f(x_7) = 11 - 11 + 3 - 5 = -2$$
$$f(x_8) = 5 - 5 + 8 - 13 = -5$$
$$f(x_9) = 11 - 3 + 8 - 5 = 11$$

$$f(x_{10}) = 9 - 11 + 7 - 9 = -4$$

میانگین برازندگی جمعیت اول $= \frac{(9+23-16-12)}{4} = -1.25$

میانگین برازندگی جمعیت دوم $= \frac{(17+15+11-2-5-4)}{6} = 5.33$

به صورت میانگین برازش جمعیت بهبود یافته اما بالاترین برازندگی یکی راه حل در جمعیت جدید از بالاترین برازندگی یک راه حل در جمعیت اولیه پایین تر است.

۲-۳-۴- با بررسی تابع برازش و در نظر گرفتن این که ژن‌ها فقط می‌توانند اعداد 0 تا 9 باشند، کروموزومی را بیابید که بیشترین مقدار برازش ممکن را داشته باشد (جواب بهینه). همچنین مقدار بیشینه‌ی برازش را محاسبه کنید.

بهترین کروموزوم = 99009900

$$f(x) = (9 + 9) - (0 + 0) + (9 + 9) - (0 + 0) = 36$$

۲-۳-۵- با بررسی جمعیت اولیه‌ی الگوریتم، آیا می‌توان گفت که این الگوریتم بدون استفاده از عملگر جهش (Mutation) می‌تواند به بهترین راه‌حل ممکن دست یابد؟

خیر. زیرا تمامی مقادیر ژن‌های مورد نظر برای رسیدن به راه حل بهینه، در ژن‌های کروموزوم‌های جمعیت اولیه وجود ندارد. برای مثال ژن a هیچکدام از راه‌حل‌ها برابر 9 نیست.

۲-۴- الگوریتم ژنتیک با جمعیت ده نفری

یک الگوریتم ژنتیکی را در نظر بگیرید که روی یک جمعیت ۱۰ نفری با ترکیب زیر اعمال شده است.

x=1دو نمونه

x=2سه نمونه

x=3سه نمونه

x=4دو نمونه

تابع برازندگی به صورت زیر تعریف شده است:

$$f(x) = x^3 - 4x^2 + 7$$

۲-۴-۱- مقادیر خام برازندگی را برای هر مقدار متمایز از x محاسبه کنید.

$$x = 1 \rightarrow f(1) = 1^3 - 4(1)^2 + 7 = 4$$
$$x = 2 \rightarrow f(2) = 2^3 - 4(2)^2 + 7 = -1$$
$$x = 3 \rightarrow f(3) = 3^3 - 4(3)^2 + 7 = -2$$
$$x = 4 \rightarrow f(4) = 4^3 - 4(4)^2 + 7 = 7$$

۲-۴-۲- بررسی کنید که آیا مقدار برازندگی خام برای هر x منفی است یا نه. در صورت وجود مقادیر منفی، یک مقدار ثابت را به تمام مقادیر برازندگی اضافه کنید تا احتمال‌های انتخاب غیرمنفی شوند.

همه مقادیر را با عدد 3 جمع می‌کنیم.

$$x = 1 \rightarrow f(1) = 4 + 3 = 7$$
$$x = 2 \rightarrow f(2) = -1 + 3 = 2$$
$$x = 3 \rightarrow f(3) = -2 + 3 = 1$$
$$x = 4 \rightarrow f(4) = 7 + 3 = 10$$

۲-۴-۳- مجموع کل برازندگی جمعیت (پس از اعمال هرگونه تغییر لازم) را محاسبه کنید.

$$2(7) + 3(2) + 3(1) + 2(10) = 43$$

۲-۴-۴- با استفاده از روش انتخاب چرخ رولت، احتمال انتخاب یک فرد با x=1 ، x=2 ، x=3 و x=4 را براساس مقادیر برازندگی (اصلاح شده) تعیین کنید.

$$P_{FPS}(x_i) = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$

$$x = 1 \rightarrow P(x_1) = \frac{7 + 7}{43} = \frac{14}{43}$$

$$x = 2 \rightarrow P(x_2) = \frac{2 + 2 + 2}{43} = \frac{6}{43}$$

$$x = 3 \rightarrow P(x_3) = \frac{1 + 1 + 1}{43} = \frac{3}{43}$$

$$x = 4 \rightarrow P(x_4) = \frac{10 + 10}{43} = \frac{20}{43}$$

۲-۴-۵- فرض کنید که اکنون احتمال‌های انتخاب با استفاده از تابع برازندگی تغییریافته زیر محاسبه می‌شوند:

$$g(x) = [f(x)]^2$$

مزیت تابع برازندگی جدید چیست؟ احتمال انتخاب هر فرد را با استفاده از $g(x)$ مجدداً محاسبه کنید.

- **تقویت تفکیک افراد با عملکرد بهتر:** با استفاده از تابع برازش g(x)، مقادیر بزرگ‌تر تابع برازش f(x)با افزایش نمایی تقویت می‌شوند. این باعث می‌شود که تفاوت‌ها میان افراد با عملکرد خوب و بد بیشتر مشخص شود به این معنا که اگر f(x)یک مقدار مثبت بزرگ تولید کند، $[f(x)]^2$ این مقدار را بیشتر بزرگ می‌کند و در نتیجه این فرد در فرآیند انتخاب اولویت بیشتری خواهد داشت. این موضوع می‌تواند به تسریع در همگرایی الگوریتم و تمرکز سریع‌تر بر روی بهینه‌سازی نهایی کمک کند.
- **تشویق به جستجوی دقیق‌تر:** تابع برازش g(x)در شرایطی که مقدار تابع برازش f(x)بزرگ است، باعث می‌شود که افرادی که نتایج بهتری دارند با سرعت بیشتری در فرآیند انتخاب قرار بگیرند. در نتیجه، احتمال اینکه این افراد در فرآیند ترکیب و جهش نقش بیشتری ایفا کنند، افزایش می‌یابد.

۲-۴-۶- توضیح دهید که استفاده از مقادیر برازندگی به توان دو، یعنی $g(x)$ به جای $f(x)$ ، چگونه فشار انتخاب (Selection Pressure) را تحت تاثیر قرار می‌دهد. این موضوع چه تاثیری بر همگرایی و تنوع جمعیت در الگوریتم ژنتیکی خواهد داشت؟

فشار انتخاب را بالاتر می‌برد و باعث می‌شود کروموزوم‌ها با برازندگی کمتر با احتمال کمتری انتخاب شوند. بالا رفتن فشار انتخاب به معنای همگرایی سریع تر و پایین تر آمدن تنوع در جمعیت می‌شود.

فصل سوم
پیاده‌سازی، ارزیابی و تجزیه تحلیل الگوریتم ژنتیک جهت انتخاب بهترین ویژگی‌ها برای بهبود مدل‌های طبقه‌بندی مشتریان فروشگاه

۳-۱- مقدمه

در فصل دوم گزارش، الگوریتم ژنتیک (GA) از ابتدا پیاده‌سازی شده است (با انتخاب پارامترهای کلیدی مانند اندازه‌ی جمعیت، تابع برازش، نرخ جهش و غیره) تا مهم‌ترین ویژگی‌ها برای مسئله‌ی طبقه‌بندی مشتریان یک فروشگاه انتخاب شوند. بهترین ۳ ، ۵ و ۷ ویژگی از میان ویژگی‌های مجموعه داده انتخاب شده و عملکرد یک مدل طبقه‌بندی را با استفاده از آنها باهم مقایسه خواهیم کرد. نتایجی که در ادامه شرح داده می‌شوند بر اساس دقت مدل در validation هستند.

۳-۲- آشنایی با مجموعه داده

پس از خواندن داده ستون‌های نامرتبط را حذف می‌کنیم.

```
# Load datasets
train_df = pd.read_csv("Train.csv")
test_df = pd.read_csv("Test.csv")

# Remove unnecessary columns
train_df.drop(columns=["ID", "Var_1"], inplace=True)
test_df.drop(columns=["ID", "Var_1"], inplace=True)
```

۳-۳- پیش‌پردازش داده‌ها

۳-۳-۱- مدیریت داده‌های از دست رفته

درصد مقادیر گمشده برای هر ستون محاسبه می‌شود. اگر مقدار گمشده در یک ستون بیشتر از ۵٪ باشد، مقدار میانه (median) برای جایگزینی مقادیر عددی و مقدار مد (mode) یا بیشترین تکرار برای جایگزینی مقادیر categorical استفاده می‌شود.

اگر مقادیر گمشده‌ی یک ستون کمتر از ۵٪ باشد، آن سطرها حذف می‌شوند.

```
print("Missing values percentage:\n", missing_percent)

# Define a threshold (e.g., 5%) for deciding whether to drop or impute
threshold = 5

# Separate numerical and categorical columns
numerical_cols = train_df.select_dtypes(include=[np.number]).columns
categorical_cols = train_df.select_dtypes(include=[object]).columns

# Handle missing values
for col in train_df.columns:
    if train_df[col].isnull().sum() > 0:  # If there are missing values
        if missing_percent[col] > threshold:
            if col in numerical_cols:  # Numerical features
                train_df[col].fillna(train_df[col].median(), inplace=True)
            elif col in categorical_cols:  # Categorical features
                train_df[col].fillna(train_df[col].mode()[0], inplace=True)
        else:
            train_df.dropna(subset=[col], inplace=True)  # Drop entire row if missing < threshold

# Apply the same strategy to test data but without dropping rows
for col in test_df.columns:
    if test_df[col].isnull().sum() > 0:
        if col in numerical_cols:
            median_value = train_df[col].median()  # Use median from training data
            test_df[col].fillna(median_value, inplace=True)
        elif col in categorical_cols:
            mode_value = train_df[col].mode()[0]  # Use mode from training data
            test_df[col].fillna(mode_value, inplace=True)

# Display missing values after handling
print("Missing values after handling:\n", train_df.isnull().sum())
```

۳-۳-۲- حذف داده‌های پرت

در این قسمت داده‌های پرت با روش IQR (Interquartile Range) شناسایی و حذف می‌شوند. ابتدا چارک اول (Q1) و چارک سوم (Q3) **محاسبه شده و فاصله بین چارک‌ها** (IQR) به دست می‌آید. محدوده‌ی مجاز داده‌ها بین (Q1 - 1.5×IQR) و (Q3 + 1.5×IQR) تعریف می‌شود و داده‌هایی که خارج از این محدوده باشند حذف می‌شوند.

```
### 2) Removing Outliers (Using IQR Method) ###
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Apply outlier removal to numerical columns
numeric_cols = train_df.select_dtypes(include=[np.number]).columns
for col in numeric_cols:
    train_df = remove_outliers(train_df, col)
```

۳-۳-۳- رمزگذاری ویژگی‌های دسته‌ای

برخی از ستون‌ها دارای داده‌های دسته‌ای (Categorical) هستند که مدل نمی‌تواند مستقیماً از آن‌ها استفاده کند و این ستون‌ها باید به مقدار عددی تبدیل شوند. این کار از دو روش انجام می‌شود:

- Label Encoding (برای ویژگی‌های دو مقداری و یا آن‌هایی که دارای ترتیب ذاتی هستند.)
- One-Hot Encoding (برای ویژگی‌هایی که بیش از دو مقدار دارند و از ترتیب خاصی نیز پیروی نمی‌کنند.

در این داده‌ها ستون‌های Gender، Ever_Married و Graduated دارای دو مقدار ممکن هستند (مثلاً "Male/Female" یا "Yes/No"). بنابراین این ستون‌ها با Label Encoding به عدد ۰ و ۱ تبدیل می‌شوند. همچنین مقادیر ستون Spending_Score به دلیل اینکه دارای ترتیب ذاتی هستند، (low/medium/high) می‌توانند با label encoding به عدد تبدیل شوند. اما این کار به صورت دستی انجام شده است زیرا در label encoding نسبت دادن اعداد به کلمات بر اساس حروف الفبا انجام می‌شود؛ درصورتی که نیاز داریم مقادیر فاصله‌های درست و منطقی از یکدیگر داشته باشند. برای مثال مقدار low به medium نزدیک تر از high

باشد.

```
### 3) Encoding Categorical Features ###
# List of binary categorical columns for Label Encoding
binary_cols = ["Gender", "Ever_Married", "Graduated"]

# List of multi-category columns for One-Hot Encoding
one_hot_cols = ["Profession"]

# Apply Label Encoding
label_encoders = {}
for col in binary_cols:
    le = LabelEncoder()
    train_df[col] = le.fit_transform(train_df[col])
    test_df[col] = le.transform(test_df[col])  # Ensure consistency
    label_encoders[col] = le

# Manually encode Spending_Score
spending_mapping = {"Low": 0, "Average": 1, "High": 2}
train_df["Spending_Score"] = train_df["Spending_Score"].map(spending_mapping)
test_df["Spending_Score"] = test_df["Spending_Score"].map(spending_mapping)

# Apply One-Hot Encoding for remaining categorical columns
train_df = pd.get_dummies(train_df, columns=one_hot_cols, drop_first=True)
test_df = pd.get_dummies(test_df, columns=one_hot_cols, drop_first=True)
```

۳-۴- پیاده‌سازی الگوریتم ژنتیک (GA) برای انتخاب ویژگی‌ها

در این بخش الگوریتم ژنتیک بدون استفاده از کتابخانه‌های آماده از ابتدا کدنویسی شده است.

۳-۴-۱- تعریف اجزای الگوریتم

نمایش کروموزوم : هر کروموزوم به اندازه تعداد ویژگی‌هایی که میخواهیم ژن دارد(۳،۵ یا ۷). که مقدار هر ژن نام مربوط به آن ویژگی است.(نیازی به تغییر آن به داده‌های عددی نبود)

جمعیت اولیه : ابتدا اندازه جمعیت ۱۰ مقدار دهی شد ولی پس از تست چند مقدار متفاوت به عدد ۲۵ رسیدیم. همچنین جمعیت اولیه به صورت رندوم مقدار دهی می‌شود.

```
# Generate Initial Population
def generate_population(features, population_size, max_features):
    return [random.sample(features, k=max_features) for _ in range(population_size)]
```

۳-۴-۲- تعریف تابع برازش

تابع برازش میزان عملکرد یک مجموعه‌ی ویژگی را در طبقه‌بندی Decision Tree Classifier ارزیابی

می‌کند و درواقع همان دقت مدل DT است.

۳-۴-۳- استراتژی انتخاب

استراتژی‌های انتخابی شامل موارد زیر هستند:

انتخاب تورنمنتی(Tournament Selection)

```python
def tournament_selection(population, scores, k=3):
    selected = random.sample(list(zip(population, scores)), k)
    return max(selected, key=lambda x: x[1])[0]
```

انتخاب چرخ رولت(Roulette Wheel Selection)

```python
def roulette_wheel_selection(population, scores):
    total_fitness = sum(scores)
    if total_fitness == 0:
        return random.choice(population)  # Avoid division by zero
    probabilities = [f / total_fitness for f in scores]
    return random.choices(population, weights=probabilities, k=1)[0]
```

۳-۴-۴- عملگر ترکیب

عملگرهای پیاده‌سازی شده شامل موارد زیر هستند:

ترکیب چندنقطه‌ای(Multi-point Crossover) : پیاده‌سازی شده به صورت دو نقطه‌ای

```python
def multi_point_crossover(parent1, parent2, max_features, num_points=2):
    child1, child2 = parent1.copy(), parent2.copy()

    if len(parent1) < num_points or len(parent2) < num_points:
        return parent1, parent2


    crossover_points = sorted(random.sample(range(len(parent1)), num_points))
    start, end = crossover_points


    child1[start:end], child2[start:end] = child2[start:end], child1[start:end]


    child1 = fix_duplicates(child1, parent1, max_features)
    child2 = fix_duplicates(child2, parent2, max_features)

    return child1, child2
```

ترکیب یکنواخت(Uniform Crossover)

```python
def uniform_crossover(parent1, parent2, max_features):
    child1, child2 = [], []

    for g1, g2 in zip(parent1, parent2):
        if random.random() > 0.5:
            child1.append(g1)
            child2.append(g2)
        else:
            child1.append(g2)
            child2.append(g1)

    # Fix duplicates and ensure max_features limit
    child1 = fix_duplicates(child1, parent1, max_features)
    child2 = fix_duplicates(child2, parent2, max_features)

    return child1, child2
```

تابع کمکی fix_duplicates

```python
def fix_duplicates(child, parent, max_features):

    unique_features = set()
    new_child = []

    for feature in child:
        if feature not in unique_features:
            unique_features.add(feature)
            new_child.append(feature)

    missing_features = [f for f in parent if f not in new_child]
    new_child.extend(missing_features[:max_features - len(new_child)])

    return new_child[:max_features]
```

۳-۴-۵- عملگر جهش

در این بخش تغییرات تصادفی کوچک در فرزندان ایجاد می‌کنیم تا تنوع حفظ شود .نرخ جهش ۶.۲۵ درصد تعیین شده است. زیرا تعداد ویژگی‌ها پس از اعمال پیش پردازش ۱۶ تا است و ۱/۱۶ برابر با این مقدار خواهد بود.

```python
# Mutation Operator
def mutation(individual, features, max_features, mutation_rate=0.05):
    if random.random() < mutation_rate:
        idx = random.randint(0, len(individual) - 1)
        possible_new_features = list(set(features) - set(individual))
        if possible_new_features:
            new_feature = random.choice(possible_new_features)
            individual[idx] = new_feature

    return individual[:max_features]
```

۳-۴-۶- معیار توقف

الگوریتم در شرایط زیر متوقف می‌شود: رسیدن به تعداد نسل‌های ثابت، عدم همگرایی جمعیت در چندین تکرار

```python
# Check for convergence (if the best accuracy does not improve significantly)
if current_best_score - best_score < improvement_threshold:
    no_improvement_count += 1
else:
    no_improvement_count = 0  # Reset if improvement occurs
    best_score = current_best_score  # Update best score

if no_improvement_count >= patience:
    print(f"Stopping early due to convergence (No improvement in {patience} generations).")
    break
```

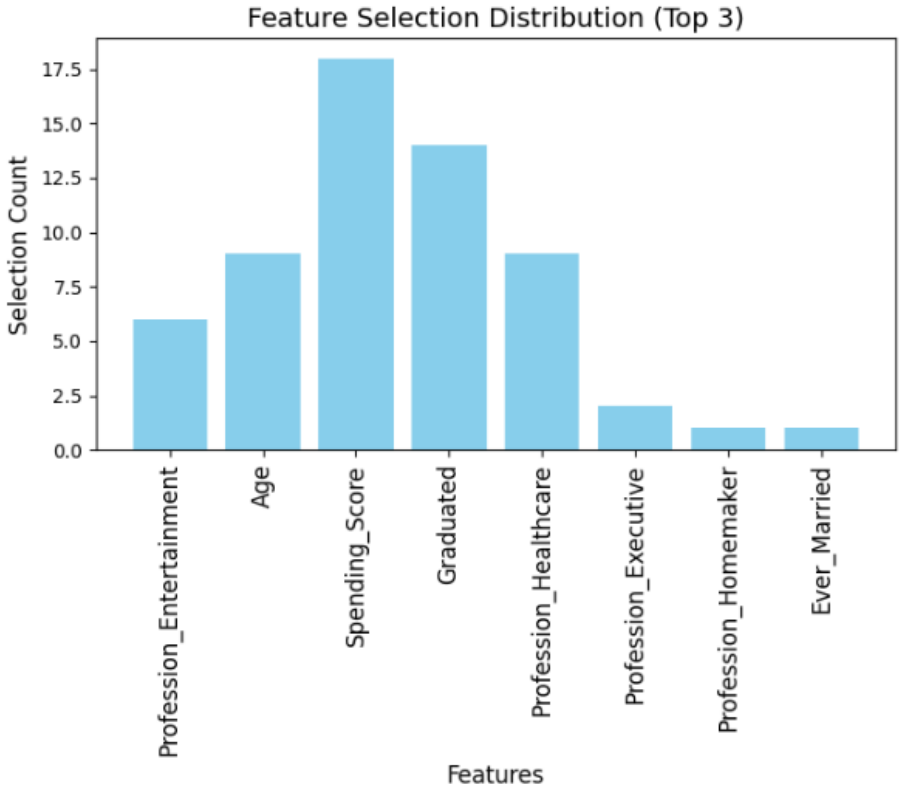
۳-۵- انتخاب بهترین ویژگی‌ها

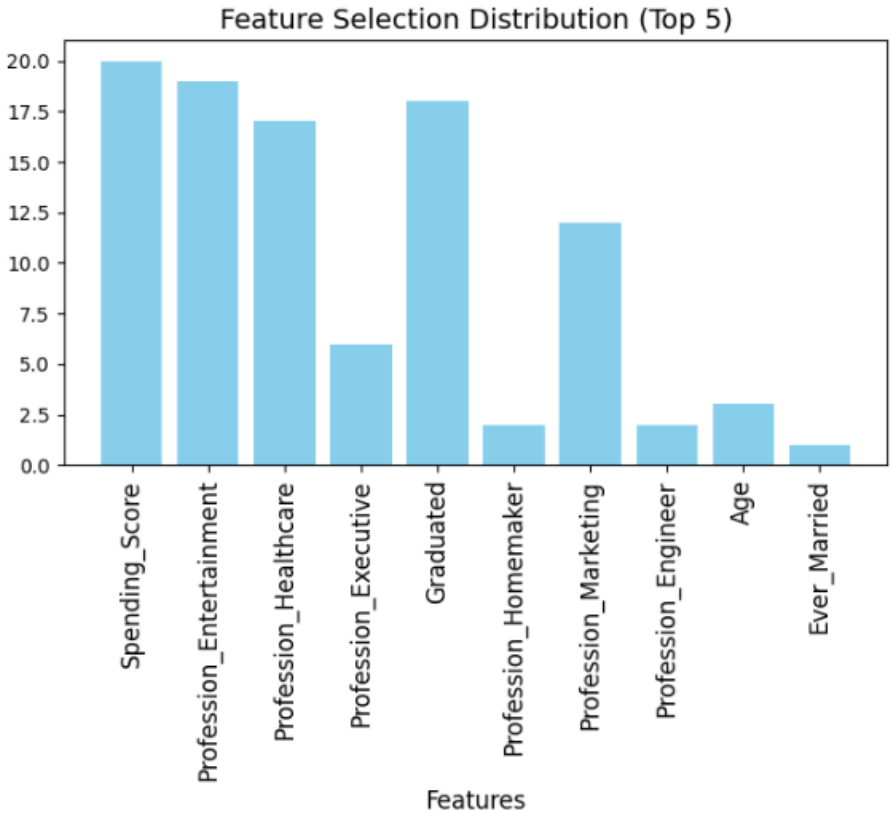
الگوریتم را ۲۰ بار برای هر مورد زیر اجرا کرده‌ایم:

- برای انتخاب ۳ ویژگی برتر.
- برای انتخاب ۵ ویژگی برتر
- برای انتخاب ۷ ویژگی برتر

```python
import numpy as np
import random
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


def run_experiment(X, y, features, num_features, num_runs=20):
    feature_counts = Counter()

    for _ in range(num_runs):
        best_features = genetic_algorithm(X, y, features, max_features=num_features)
        feature_counts.update(best_features)

    return feature_counts


def run_decision_tree(X, y, selected_features):
    X_subset = X[selected_features]
    X_train, X_test, y_train, y_test = train_test_split(X_subset, y, test_size=0.2, random_state=42)

    clf = DecisionTreeClassifier(random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    return accuracy_score(y_test, y_pred)


feature_sizes = [3, 5, 7]
feature_counts_dict = {size: run_experiment(train_df[features], train_df["Segmentation"], features, num_features=size) for size in feature_sizes}


selected_features = {size: [f for f, _ in feature_counts_dict[size].most_common(size)] for size in feature_sizes}


results = {}
results["All Features"] = run_decision_tree(train_df[features], train_df["Segmentation"], features)
results["Top 3 Features"] = run_decision_tree(train_df[features], train_df["Segmentation"], selected_features[3])
results["Top 5 Features"] = run_decision_tree(train_df[features], train_df["Segmentation"], selected_features[5])
results["Top 7 Features"] = run_decision_tree(train_df[features], train_df["Segmentation"], selected_features[7])
```

زیرمجموعه‌های ویژگی‌هایی را که بالاترین امتیاز برازش را دارند، استخراج و با رسم نمودار توزیع هر ویژگی تحلیل کرده‌ایم. همانطور که در شکل‌های زیر نیز مشخص است، بهترین ویژگی‌های هر بخش انتخاب شده است. این ویژگی‌ها در بخش پاسخ سوال‌های نهایی (۳-۷-۱) به طور دقیق عنوان شده‌اند.




Feature Selection Distribution (Top 5)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Spending_Score
Profession_Entertainment
Profession_Healthcare
Profession_Executive
Graduated
Profession_Homemaker
Profession_Marketing
Profession_Engineer
Age
Ever_Married
Features


Feature Selection Distribution (Top 7)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Graduated
Profession_Healthcare
Profession_Executive
Profession_Marketing
Ever_Married
Profession_Entertainment
Spending_Score
Profession_Lawyer
Age
Profession_Engineer
Gender
Profession_Homemaker
Profession_Doctor
Family_Size
Work_Experience
Features

۳-۶- آموزش و ارزیابی مدل طبقه‌بندی

پس از انتخاب ۳ ، ۵ و ۷ ویژگی برتر توسط GA ، موارد زیر را انجام داده و بررسی می‌کنیم.

۳-۶-۱- آموزش مدل DT

در این بخش مدل Decision Tree Classifier برای طبقه‌بندی با استفاده از هر یک از ۳ ویژگی منتخب، ۵ ویژگی منتخب، ۷ ویژگی منتخب و همه‌ی ویژگی‌ها آموزش داده شده است.

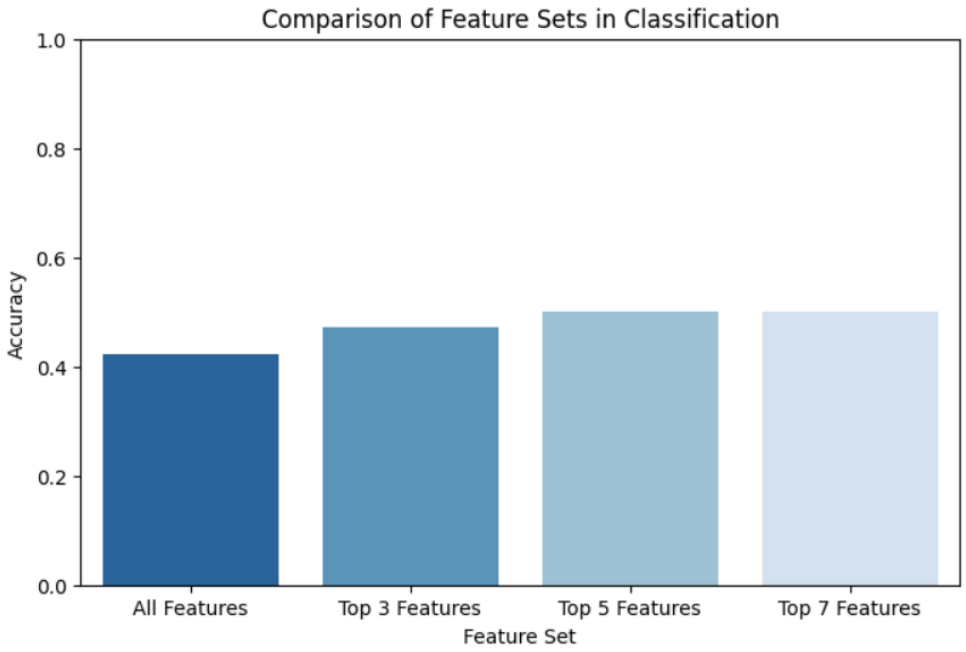
۳-۶-۲- ارزیابی نحوه عملکرد مدل‌های DT با استفاده از معیارهای دقت

پس از آموزش مدل‌ها، عملکرد آنها را با استفاده از معیارهای دقت ارزیابی کنید.

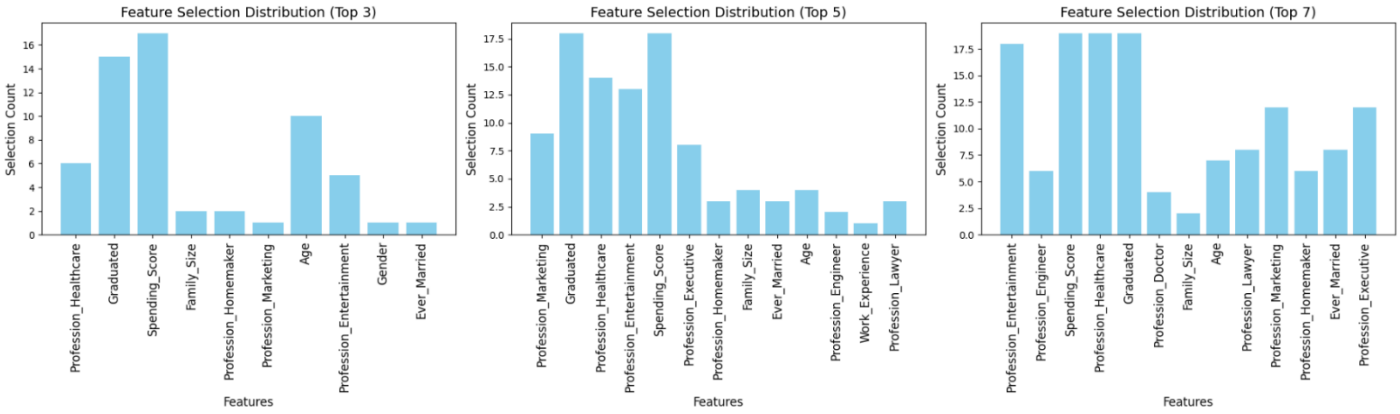
۳-۶-۳- مقایسه نتایج بدست آمده با استفاده از نمودار

نتایج بدست آمده از دقت مدل در حالتی که استراتژی انتخاب تورنمنت و عملگر ترکیب به صورت یکنواخت پیاده سازی شده باشد (بهترین حالت):

	Feature Set	Accuracy
0	All Features	0.421695
1	Top 3 Features	0.471864
2	Top 5 Features	0.501017
3	Top 7 Features	0.501695



رسم دلیل انتخاب ویژگی‌های برتر در هر سه حالت:

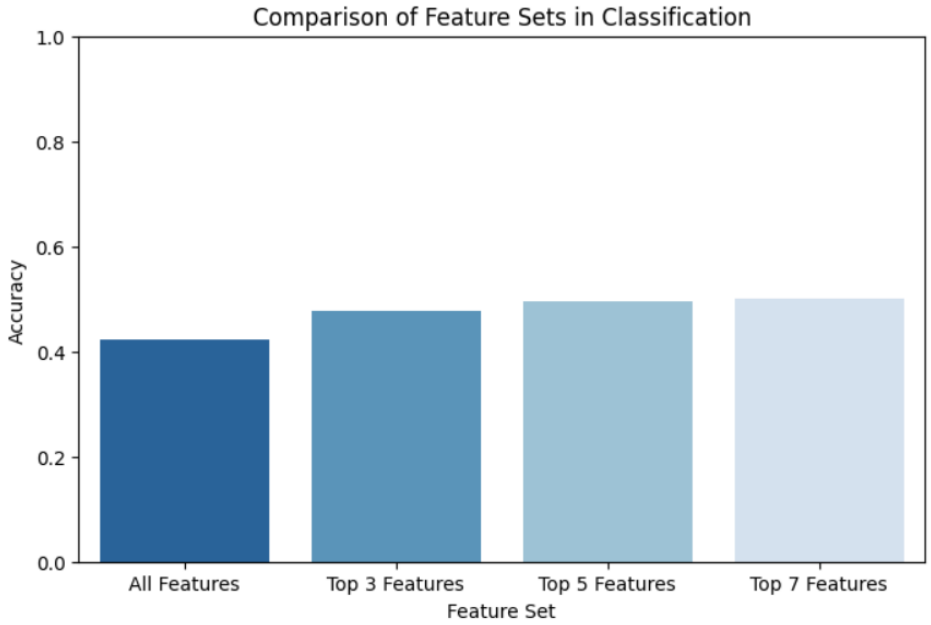


نتایج بدست آمده از این مدل روی داده‌های تست(test_df) :

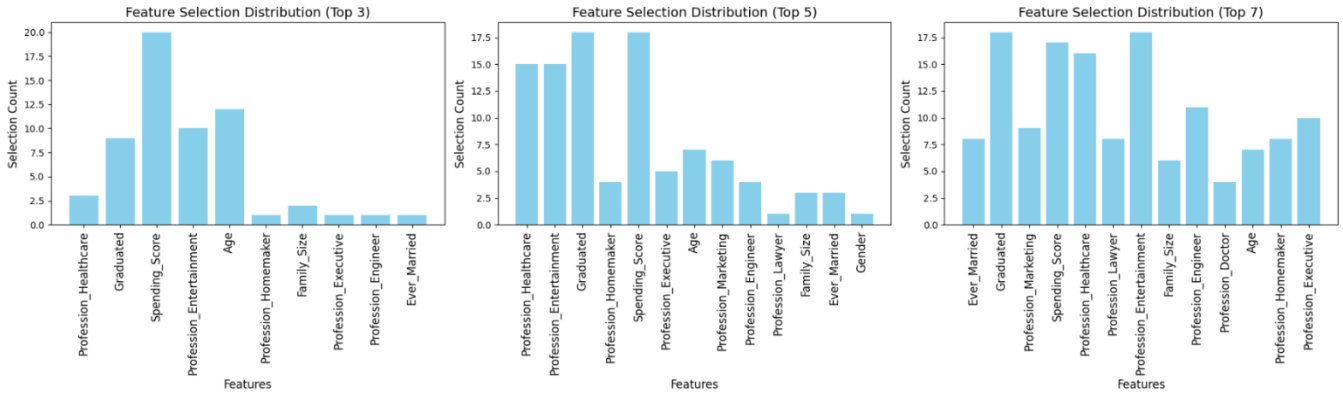
```
'All Features': 0.30376855728968405, 'Top 3 Features': 0.326227636086791, 'Top 5 Features': 0.3269889607917777, 'Top 7 Features': 0.323943661971831}
```

نتایج بدست آمده از دقت مدل در حالتی که استراتژی انتخاب تورنمنت و عملگر ترکیب به صورت دو نقطه‌ای پیاده سازی شده باشد:

	Feature Set	Accuracy
0	All Features	0.421695
1	Top 3 Features	0.477288
2	Top 5 Features	0.494237
3	Top 7 Features	0.501017

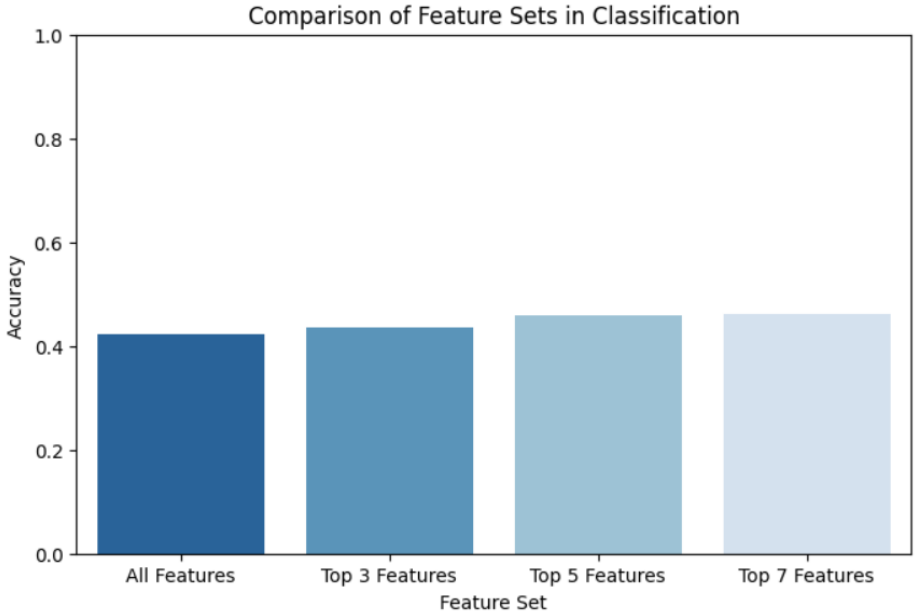


رسم دلیل انتخاب ویژگی‌های برتر در هر سه حالت:

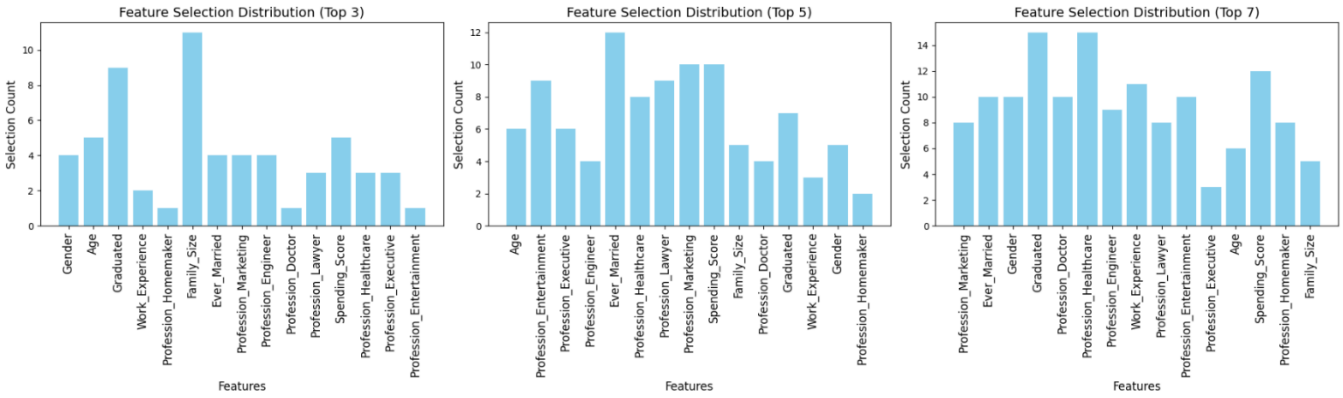


نتایج بدست آمده از دقت مدل در حالتی که استراتژی انتخاب چرخ رولت و عملگر ترکیب به صورت دو نقطه‌ای پیاده سازی شده باشد:

	Feature Set	Accuracy
0	All Features	0.421695
1	Top 3 Features	0.435254
2	Top 5 Features	0.458305
3	Top 7 Features	0.460339

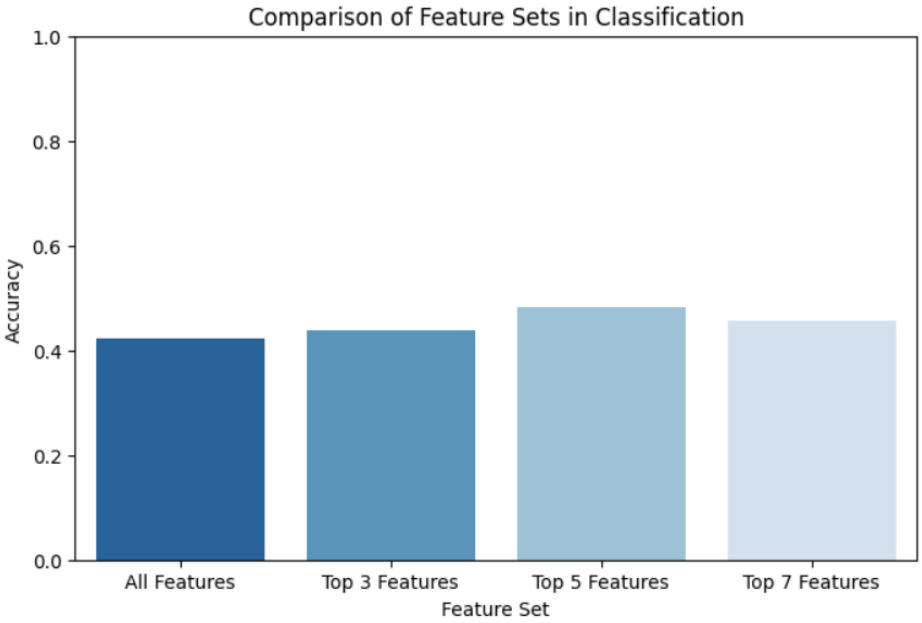


رسم دلیل انتخاب ویژگی‌های برتر در هر سه حالت:

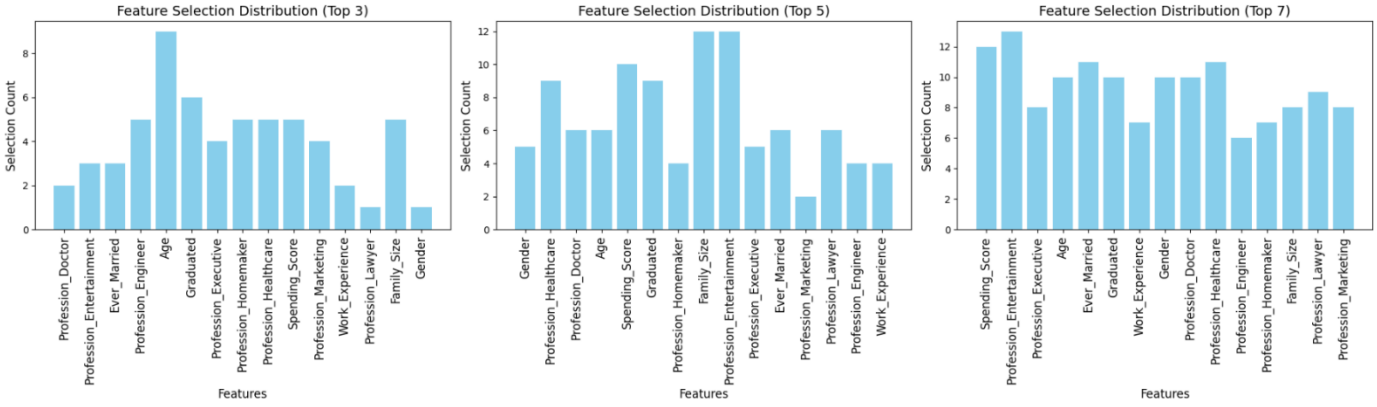


نتایج بدست آمده از دقت مدل در حالتی که استراتژی انتخاب چرخ رولت و عملگر ترکیب به صورت یکنواخت پیاده سازی شده باشد:

	Feature Set	Accuracy
0	All Features	0.421695
1	Top 3 Features	0.439322
2	Top 5 Features	0.482712
3	Top 7 Features	0.456949



رسم دلیل انتخاب ویژگی‌های برتر در هر سه حالت:



۳-۷- پاسخ به سوالات انتهایی

۳-۷-۱- کدام مجموعه‌ی ویژگی بهترین عملکرد طبقه‌بندی را داشت؟

مجموعه ویژگی هفت تایی graduated, spending_score, profession_healthcare, profession_entertainment, profession_engineer, profession_executive, profession_marketing بهترین عملکرد را داشته است.

۳-۷-۲- آیا استفاده از GA برای انتخاب ویژگی باعث بهبود مدل شد، یا عملکرد مشابهی

با همه‌ی ویژگی‌ها داشت؟

بله تا ۳ درصد باعث بهبود دقت مدل نسبت به حالتی که همه ویژگی‌ها را در نظر بگیریم، شده است.

۳-۷-۳- مزایا و معایب استفاده از تعداد ویژگی‌های کمتر در مقایسه با همه‌ی ویژگی‌ها چیست؟

استفاده از تعداد ویژگی‌های کمتر در مقایسه با تمامی ویژگی‌های موجود، مزایا و معایب متعددی دارد که باید هنگام طراحی مدل به آن‌ها توجه شود.

مزایا: یکی از مزایای اصلی کاهش تعداد ویژگی‌ها، کاهش پیچیدگی مدل است. با کاهش تعداد ویژگی‌ها، تعداد پارامترهای مدل نیز کاهش می‌یابد و این امر منجر به یادگیری سریع‌تر و کاهش هزینه‌های محاسباتی خواهد شد. درنتیجه حذف ویژگی‌های غیرضروری می‌تواند باعث افزایش کارایی پردازش و کاهش زمان آموزش مدل شود. همچنین، استفاده از ویژگی‌های کمتر می‌تواند به کاهش احتمال بیش‌برازش (Overfitting) کمک کند، زیرا ویژگی‌های غیرضروری و نویزی ممکن است باعث شوند مدل بیش از حد به داده‌های آموزشی وابسته شده و در تعمیم‌پذیری به داده‌های جدید دچار مشکل شود. در واقع، حذف این ویژگی‌ها می‌تواند عملکرد مدل را در مواجهه با داده‌های ندیده بهبود بخشد.

علاوه بر این، کاهش تعداد ویژگی‌ها باعث افزایش تفسیرپذیری مدل می‌شود. زمانی که تعداد ویژگی‌ها زیاد باشد، درک نحوه تأثیر هر ویژگی بر خروجی مدل دشوار می‌شود. در مقابل، داشتن تعداد محدودی از ویژگی‌های مهم، باعث می‌شود که مدل ساده‌تر شده و بتوان نتایج آن را به‌صورت واضح‌تر تحلیل کرد. از دیگر مزایای انتخاب ویژگی، کاهش تأثیر ویژگی‌های نامرتبط یا همبسته است. برخی از ویژگی‌ها ممکن است اطلاعات تکراری داشته باشند یا هیچ تأثیر خاصی بر روی خروجی مدل نداشته باشند. حذف این ویژگی‌ها می‌تواند منجر به بهبود دقت مدل و جلوگیری از ورود اطلاعات زائد به فرآیند یادگیری شود.

معایب: با وجود تمامی این مزایا، حذف برخی ویژگی‌ها می‌تواند باعث از دست دادن اطلاعات مهم شود. در صورتی که ویژگی‌های کلیدی و تأثیرگذار در فرآیند انتخاب ویژگی نادیده گرفته شوند، مدل ممکن است عملکرد ضعیف‌تری داشته باشد و دقت آن کاهش یابد. علاوه بر این، انتخاب ویژگی‌های مناسب نیازمند استفاده از روش‌های دقیق و مناسب برای انتخاب ویژگی‌ها است. برخی از این روش‌ها مانند تحلیل مؤلفه‌های اصلی (PCA)، ممکن است به تنظیمات دقیق نیاز داشته باشند و استفاده نادرست از آن‌ها می‌تواند موجب حذف ویژگی‌های مهم شود.

در برخی موارد، کاهش تعداد ویژگی‌ها ممکن است عملکرد مدل‌های پیچیده را کاهش دهد. مدل‌هایی مانند شبکه‌های عصبی عمیق توانایی یادگیری روابط پیچیده بین ویژگی‌ها را دارند و حذف برخی ویژگی‌ها می‌تواند اطلاعاتی را از بین ببرد که در لایه‌های عمیق‌تر شبکه می‌توانند پردازش و استخراج شوند. بنابراین، در

مدل‌های پیچیده، استفاده از تمامی ویژگی‌ها می‌تواند مفید باشد.

۳-۷-۴- کدام پارامترهای GA (اندازه‌ی جمعیت، روش انتخاب، نرخ جهش) بیشترین تأثیر را بر عملکرد داشتند؟

روش انتخاب و نرخ جهش بیشترین تاثیر را داشتند. با بالا بردن نرخ جهش تاثیر مثبت در دقت مدل مشاهده می‌شود. همچنین تفاوت نسبتا محسوسی (نسبت به بقیه پارامترها) میان استراتژی چرخ رولت و تورنمنت وجود دارد. تورنمنت مقدار کمی بهتر عمل کرده است.

منابع

[1] J. H. X. C. S. S. I. S. Tim Salimans, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning".

[2] "chatgpt," [Online]. Available: https://chatgpt.com/.

[3] "datacamp," [Online]. Available: https://www.datacamp.com/tutorial/genetic-algorithm-python.

[4] "medium," [Online]. Available:

https://medium.com/@Data_Aficionado_1083/genetic-algorithms-optimizing-success-through-evolutionary-computing-f4e7d452084f.

[5] "youtube," [Online]. Available: https://www.youtube.com/watch?v=nhT56blfRpE.