دانشکده مهندسی کامپیوتر

خوشه‌بندی رکوردهای فراخوانی سیستم در نرم‌افزارها

پروژه کارشناسی مهندسی کامپیوتر

شایان موسوی‌نیا

استاد راهنما

دکتر وحید ازهری

اسفند ۱۴۰۱


بسم الله الرحمن الرحيم

تأییدیهٔ هیئت داوران جلسه دفاع از پروژه پایانی

نام دانشکده: دانشکده مهندسی کامپیوتر

نام دانشجو: شایان موسوی‌نیا

عنوان پروژه: خوشه‌بندی رکوردهای فراخوانی سیستم در نرم‌افزارها

تاریخ دفاع: اسفند ۱۴۰۱

گرایش:

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امضا
۱	استاد راهنما	دکتر وحید ازهری	استادیار	دانشگاه علم و صنعت	
۲	استاد داور داخلی	دکتر ابوالفضل دیانت	استادیار	دانشگاه علم و صنعت	

تأییدیهٔ صحت و اصالت نتایج

بسمه‌تعالی

اینجانب شایان موسوی‌نیا به شماره دانشجویی ۹۷۵۲۲۲۳۸ دانشجوی رشته مهندسی کامپیوتر مقطع تحصیل کارشناس مهندس کامپیوتر تأیید می‌نمایم که کلیهٔ نتایج این پروژه پایان حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. در صورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوت، ضوابط و مقررات آموزش، پژوهش و انضباط ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض در خصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسئولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهدهٔ اینجانب خواهد بود و دانشگاه هیچ‌گونه مسئولیت دراین‌خصوص نخواهد داشت.

نام و نام خانوادگی: شایان موسوی‌نیا

تاریخ و امضا:

مجوز بهره‌بری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و باتوجه‌به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

□ بهره‌برداری از این پایان‌نامه برای همگان بلامانع است.

□ بهره‌برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

□ بهره‌برداری از این پایان‌نامه تا تاریخ ..... ممنوع است.

استاد راهنما: دکتر وحید ازهری

تاریخ و امضا:

قدردانی

سپاس خداوندگار حکیم را که با لطف بی‌کران خود، آدمی را زیور عقل آراست.

در آغاز وظیفه خودم می‌دانم از زحمات بی‌دریغ اســـتاد راهنمای خود، جناب آقای دکتر وحید ازهری صمیمانه تشکر و قدردان می‌کنم که قطعاً بدون راهنمایی‌های ارزنده ایشان، این مجموعه به انجام نمی‌رسید.

در پایان، بوســـه می‌زنم بر دســـتان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، وجود مقدســـشـــان را ســـتایش می‌کنم و تشـــکر می‌کنم از خانواده عزیزم به‌پاس عاطفه ســـرشـــار و گرمای امیدبخش وجودشان که بهترین پشتیبان من بودند.

شایان موسوی‌نیا

اسفند ۱۴۰۱

چکیده

یکی از کمک‌های قابل توجه این رویکرد توانایی آن در شناسایی خوشه ای است که فرآیند کاربر به آن تعلق دارد، حتی بدون هیچ گونه اطلاعات دیگری به غیر از تماس‌های سیستمی که در طول فرآیند انجام می‌شود. برچسب خوشه ای به دست آمده می‌تواند بینشی در مورد ماهیت عملکرد انجام شده توسط کاربر ارائه دهد. به عنوان مثال، برچسب خوشه ممکن است نشان دهد که عمل مربوط به کار بوده است.

این مطالعه نشان می‌دهد که برای اجرای الگوریتم خوشه‌بندی، ابعاد ورودی‌ها باید یکسان باشند تا بردارهای یکسان آماده شوند. پس از خوشه‌بندی داده‌ها، فرایند خوشه‌بندی مورد ارزیابی قرار گرفت و تعداد خوشه‌های در نظر گرفته شده مناسب تشخیص داده شد. رویکرد پیشنهادی در این مطالعه، خوشه‌بندی کنش‌های یک سیستم را امکان‌پذیر می‌سازد، و این امکان را فراهم می‌آورد که مشخص شود یک عمل معین در کدام خوشه تعلق دارد.

فرایند جمع‌آوری داده‌ها یکی از مهم‌ترین مراحل این مطالعه بود، زیرا هیچ نمونه داده‌ای موجود در اینترنت وجود نداشت. استفاده از نخ‌ها در فرایند جمع‌آوری داده‌ها برای اطمینان از عملکرد صحیح بسیار مهم بود. علاوه بر این، ذخیره رکوردها در فریم‌های مناسب با استفاده از کتابخانه پانداها مدیریت بهتر داده‌ها را تسهیل می‌کند.

رویکرد پیشنهادی پیامدهایی برای زمینه‌های مختلف از جمله امنیت سایبری و بهینه‌سازی عملکرد سیستم دارد. با خوشه‌بندی اقدامات سیستم، شناسایی فعالیت‌های مخرب در یک سیستم، شناسایی گلوگاه‌های عملکرد و بهینه‌سازی عملکرد سیستم با نظارت بر تماس‌های سیستم امکان‌پذیر است.

در نتیجه، این مطالعه با تجزیه‌وتحلیل سوابق فراخوانی سیستم، رویکرد جدیدی را برای خوشه‌بندی اقدامات سیستم ارائه کرد. این رویکرد پیامدهایی برای زمینه‌های مختلف از جمله امنیت سایبری و بهینه‌سازی عملکرد سیستم دارد. توانایی این رویکرد برای شناسایی خوشه‌ای که فرایند کاربر به آن تعلق دارد، حتی بدون هیچ‌گونه اطلاعات دیگری به‌غیراز تماس‌های سیستمی که در طول فرایند انجام می‌شود، آن را به ابزاری ارزشمند برای درک رفتار سیستم تبدیل می‌کند. کار آینده می‌تواند کاربرد این رویکرد را برای برنامه‌ها و سیستم‌های دیگر بررسی کند و عملکرد آن را در مقایسه با سایر الگوریتم‌های خوشه‌بندی ارزیابی کند.

فهرست مطالب

مقدمه

تجزیه‌وتحلیل خوشه‌ای تکنیکی است که درداده کاوی و یادگیری ماشین استفاده می‌شود که شامل گروه‌بندی مجموعه‌ای از نقاط داده یا اشیا به زیرمجموعه‌ها یا "خوشه‌ها" بر اساس شباهت یا نزدیکی آنها به یکدیگر است. در زمینه فراخوانی سیستم، خوشه‌بندی می‌تواند برای گروه‌بندی توالی‌های مشابهی از فراخوان‌های سیستمی که احتمالاً با اعمال یا رفتارهای خاص مطابقت دارند، استفاده شود.

چندین دلیل وجود دارد که چرا فراخوانی‌های سیستم خوشه‌بندی می‌تواند برای درک اقداماتی که آنها مطابقت دارند مفید باشد:

- درک بهبودیافته: خوشه‌بندی به ارائه یک نمای سازمان‌یافته‌تر و ساختارمند از توالی فراخوانی سیستم کمک می‌کند و شناسایی الگوها و درک اقداماتی را که آنها نشان می‌دهند آسان‌تر می‌کند.
- کاهش پیچیدگی: با گروه‌بندی توالی‌های فراخوانی سیستم مشابه، خوشه‌بندی می‌تواند به کاهش پیچیدگی کلی داده‌ها و قابل‌مدیریت‌تر کردن آن‌ها کمک کند.
- تشخیص ناهنجاری‌ها: خوشه‌بندی می‌تواند برای شناسایی توالی‌های فراخوانی غیرعادی یا غیرعادی که ممکن است نشان‌دهنده فعالیت مخرب یا سایر انواع سوءاستفاده از سیستم باشد، استفاده شود.
- بهبود عملکرد: با شناسایی الگوهای رایج در دنباله‌های فراخوانی سیستم، خوشه‌بندی همچنین می‌تواند برای بهینه‌سازی عملکرد سیستم با پیشنهاد راه‌هایی برای ساده‌سازی و ساده‌سازی اعمال یا رفتارهای خاص مورداستفاده قرار گیرد.

به‌طورکلی، خوشه‌بندی فراخوانی‌های سیستم یک تکنیک مفید برای درک اعمال و رفتارهای مرتبط با توالی فراخوانی سیستم است و می‌تواند به بهبود عملکرد سیستم، شناسایی ناهنجاری‌ها و ارائه بینش‌هایی در مورد رفتار کلی یک سیستم کمک کند.

توانایی تجزیه‌وتحلیل تماس‌های سیستمی برای به‌دست‌آوردن بینش در مورد رفتار سیستم به یک حوزه تحقیقاتی روبه‌افزایش تبدیل شده است. در این پایان‌نامه، ما یک رویکرد جدید برای خوشه‌بندی اقدامات سیستم بر اساس سوابق فراخوانی‌های سیستم برای یک عملیات خاص در یک برنامه، مانند `Discord`، پیشنهاد می‌کنیم. این رویکرد می‌تواند برای جداسازی اقدامات مختلف که می‌تواند در یک برنامه انجام شود، استفاده می‌شود و درک دقیق‌تری از رفتار سیستم ارائه می‌دهد.
تهیه یک مجموعه‌داده مناسب و بدون خطا برای موفقیت الگوریتم خوشه‌بندی بسیار مهم است. بااین‌حال، ازآنجایی‌که هیچ نمونه داده‌ای موجود در اینترنت وجود نداشت، نویسندگان مجبور بودند مجموعه‌داده را از ابتدا جمع‌آوری کنند. برای جمع‌آوری داده‌ها در پلتفرم `Discord`، نویسندگان از دستور `perf` استفاده کردند و استفاده از نخ‌ها را برای عملکرد مناسب توصیه کردند. سپس داده‌ها در چارچوب‌های مناسب با استفاده از کتابخانه پانداها برای تسهیل مدیریت بهتر داده‌ها ذخیره شدند.

برای اجرای الگوریتم خوشه بندی، ابعاد ورودی‌ها باید یکسان باشد، بنابراین نویسندگان اطمینان حاصل کردند که بردارهای یکسانی برای الگوریتم آماده شده است. پس از خوشه‌بندی داده‌ها، نویسندگان فرآیند خوشه‌بندی را ارزیابی کردند و تشخیص دادند که تعداد خوشه‌های در نظر گرفته شده برای هدف مناسب است.

رویکرد پیشنهادی پیامدهای قابل‌توجهی برای زمینه‌های مختلف از جمله امنیت سایبری و بهینه‌سازی عملکرد سیستم دارد. با خوشه‌بندی اقدامات سیستم، شناسایی فعالیت‌های مخرب در یک سیستم، شناسایی گلوگاه‌های عملکرد و بهینه‌سازی عملکرد سیستم با نظارت بر تماس‌های سیستم امکان‌پذیر است.

یکی از مهم‌ترین کمک‌های این رویکرد، توانایی آن در شناسایی خوشه‌ای است که فرایند کاربر به آن تعلق دارد، حتی بدون هیچ اطلاعات دیگری به‌غیراز تماس‌های سیستمی که در طول فرایند انجام می‌شود. برچسب خوشه‌ای به‌دست‌آمده می‌تواند بینشی در مورد ماهیت عملکرد انجام شده توسط کاربر ارائه دهد، مانند اینکه آیا این کار به کار مربوط است یا خیر.

در نتیجه، در این مقاله یک رویکرد جدید برای خوشه‌بندی اقدامات سیستم بر اساس سوابق فراخوانی‌های سیستم برای یک عملیات خاص در یک برنامه پیشنهاد می‌کنند. این رویکرد پیامدهای قابل‌توجهی برای زمینه‌های مختلف دارد و درک دقیق‌تری از رفتار سیستم ارائه می‌دهد. رویکرد نویسندگان به جمع‌آوری، آماده‌سازی و خوشه‌بندی داده‌ها ابزار ارزشمندی برای درک رفتار سیستم و شناسایی فعالیت‌های مخرب فراهم می‌کند. کار آینده می‌تواند کاربرد این رویکرد را برای برنامه‌ها و سیستم‌های دیگر بررسی کند و عملکرد آن را در مقایسه با سایر الگوریتم‌های خوشه‌بندی ارزیابی کند.

ادبیات موضوع:

فراخوانی سیستم (system call)

فراخوانی سیستم در یک سیستم‌عامل (OS) مکانیزمی است که توسط برنامه‌ها برای درخواست خدمات از هسته، هسته سیستم‌عامل استفاده می‌شود. این خدمات می‌تواند شامل دسترسی به دستگاه‌های سخت‌افزاری، سیستم‌های فایل، منابع شبکه و سایر منابع سیستم باشد. فراخوانی سیستم پلی بین فضای کاربر و فضای هسته در یک سیستم‌عامل ایجاد می‌کند و برنامه‌ها را قادر می‌سازد با سیستم‌عامل تعامل داشته باشند و به ویژگی‌های آن دسترسی داشته باشند.

هنگامی که یک برنامه نیاز به دسترسی به یک منبع سیستمی دارد که توسط سیستم‌عامل کنترل می‌شود، یک تماس سیستمی برقرار می‌کند. برنامه مجموعه‌ای از پارامترها را به سیستم‌عامل ارسال می‌کند که نشان می‌دهد به چه سرویسی نیاز دارد و سیستم‌عامل عملیات درخواستی را از طرف برنامه انجام می‌دهد. پس از اتمام عملیات، سیستم‌عامل کنترل را به برنامه باز می‌گرداند و برنامه می‌تواند به اجرا ادامه دهد.

تماس‌های سیستمی را می‌توان بر اساس انواع خدماتی که ارائه می‌کنند به چند دسته طبقه‌بندی کرد. برخی از دسته‌های رایج تماس‌های سیستمی عبارت‌اند از:

- مدیریت فرایند: این فراخوانی‌های سیستمی به برنامه‌های کاربردی اجازه می‌دهند تا فرایندها، رشته‌ها و سایر موجودیت‌های اجرایی در سیستم را ایجاد، خاتمه و کنترل کنند.
- مدیریت فایل: این فراخوانی‌های سیستمی به برنامه‌ها امکان خواندن، نوشتن و دست‌کاری فایل‌ها و دایرکتوری‌ها را می‌دهد.
- مدیریت حافظه: این فراخوان‌های سیستمی خدماتی را برای تخصیص و تخصیص حافظه و همچنین مدیریت حافظه مجازی ارائه می‌دهند.
- مدیریت دستگاه: این فراخوانی‌های سیستمی، برنامه‌ها را قادر می‌سازد تا با دستگاه‌های سخت‌افزاری مانند درایوهای دیسک، چاپگرها و رابط‌های شبکه تعامل داشته باشند.
- مدیریت شبکه: این فراخوان‌های سیستمی خدماتی را برای مدیریت اتصالات شبکه، سوکت‌ها و سایر منابع مرتبط با شبکه ارائه می‌دهند.

اجرای تماس‌های سیستمی بسته به سیستم‌عامل می‌تواند متفاوت باشد. معمولاً فراخوانی‌های سیستمی به‌عنوان دستورالعمل‌های خاص یا وقفه‌های نرم‌افزاری پیاده‌سازی می‌شوند که هنگام اجرا در هسته به دام می‌افتند. هنگامی که یک فراخوانی سیستمی اجرا می‌شود، هسته زمینه فعلی برنامه را ذخیره می‌کند، به حالت هسته تغییر می‌کند و عملیات درخواستی را انجام می‌دهد. پس از اتمام عملیات، هسته زمینه ذخیره شده را بازیابی می‌کند و کنترل را به برنامه باز می‌گرداند.

فراخوانی‌های سیستمی جزء حیاتی هر سیستم‌عامل هستند، زیرا برنامه‌ها را قادر می‌سازند تا با سیستم اصلی تعامل داشته باشند و از منابع آن استفاده کنند؛ بنابراین، کارایی و قابلیت اطمینان تماس‌های سیستمی برای عملکرد کلی و پایداری سیستم‌عامل بسیار مهم است. علاوه بر این، طراحی و اجرای فراخوان‌های سیستمی می‌تواند بر امنیت و جداسازی سیستم‌عامل و همچنین توانایی برنامه‌ها برای اشتراک‌گذاری و همکاری در منابع سیستم تأثیر بگذارد. در نتیجه، فراخوانی سیستم یک حوزه مهم مطالعه و تحقیق در سیستم‌عامل‌ها و علوم کامپیوتر به طور گسترده‌تر است.

کتابخانه perf:

کتابخلنه perf مجموعه‌ای قدرتمند از ابزارها را برای پروفایل و تجزیه‌وتحلیل عملکرد یک سیستم لینوکس فراهم می‌کند. یکی از ویژگی‌های کلیدی perf امکان ثبت یک‌سری فرایندها و تجزیه‌وتحلیل دقیق آنهاست.

برای شروع ضبط، می‌توانید از دستور "perf record" که بخشی از مجموعه ابزار perf است استفاده کنید. این دستور به شما امکان می‌دهد رویدادهایی را که می‌خواهید ضبط کنید، مانند چرخه‌های CPU یا ازدست‌رفتن حافظه پنهان، و همچنین فرایند یا فرایندهای هدف را مشخص کنید.

پس از اتمام ضبط، می‌توانید نتایج را با استفاده از دستور "perf report" تجزیه‌وتحلیل کنید. این دستور یک نمای کلی از رویدادهای ضبط شده به همراه آمار و تجسم‌های مختلف ارائه می‌دهد تا به شما در شناسایی تنگناهای عملکرد و سایر مسائل کمک کند.

علاوه بر ضبط و تجزیه‌وتحلیل رویدادها، کتابخلنه perf همچنین طیف وسیعی از ابزارهای دیگر را برای پروفایل و بهینه‌سازی سیستم‌های لینوکس فراهم می‌کند. این ابزارها عبارت‌اند از "perf stat" برای جمع‌آوری آمار در سراسر سیستم، "perf top" برای نظارت بر عملکرد در زمان واقعی، و "perf annotate" برای تجزیه‌وتحلیل دقیق کد منبع.

با استفاده از کتابخانه perf برای ثبت و تجزیه‌وتحلیل فرایندهای خود، می‌توانید بینش ارزشمندی در مورد عملکرد سیستم خود به دست آورید و زمینه‌های بهینه‌سازی و بهبود را شناسایی کنید.

کتابخانه panda

کتابخانه Pandas یک ابزار محبوب و قدرتمند دست‌کاری داده‌ها برای زبان برنامه‌نویسی پایتون است. این بر روی بسته NumPy ساخته شده است و ساختارهای داده سریع، منعطف و گویا را ارائه می‌دهد که کار با داده‌های "رابطه‌ای" یا "برچسب" را آسان و شهودی می‌کند.

این کتابخانه توسط Wes McKinney در سال 2008 در زمانی که او در مدیریت سرمایه AQR کار می‌کرد، توسعه یافت و از آن زمان به یکی از پرکاربردترین کتابخانه‌های پایتون برای تجزیه‌وتحلیل و دستیاری داده‌ها تبدیل شده است.

Pandas دو کلاس اصلی برای دستکاری داده‌ها ارائه می‌دهد: Series و DataFrame. سری یک آرایه برچسب‌دار تک‌بعدی است که می‌تواند هر نوع داده‌ای از جمله اعداد صحیح، شناورها، رشته‌ها و اشیاء پایتون را در خود جای دهد. DataFrame یک ساختار داده با برچسب دو بعدی با ستون‌هایی از انواع مختلف است. این شبیه به یک صفحه گسترده یا یک جدول SQL است و می‌تواند به عنوان مجموعه ای از اشیاء سری در نظر گرفته شود.

یکی از قدرتمندترین ویژگی‌های پانداها توانایی آن در مدیریت داده‌های از دست رفته است. این کتابخانه چندین روش برای شناسایی، حذف و پر کردن مقادیر از دست رفته ارائه می‌کند، که اغلب هنگام کار با داده‌های دنیای واقعی می‌تواند چالش‌برانگیز باشد.

پانداها همچنین شامل طیف گسترده ای از روش‌ها برای دستکاری و تبدیل داده‌ها هستند. به عنوان مثال، کتابخانه روش‌هایی را برای فیلتر کردن و انتخاب داده‌ها، محاسبه آمار خلاصه، اعمال توابع برای گروه‌های داده و ادغام، پیوستن و تغییر شکل مجموعه‌های داده ارائه می‌کند.

علاوه بر این قابلیت‌های دستکاری داده‌های اولیه، پانداها بسیاری از ویژگی‌های پیشرفته را برای موارد استفاده تخصصی‌تر ارائه می‌دهد. به عنوان مثال، کتابخانه شامل روش‌هایی برای کار با داده‌های سری زمانی است که می‌تواند به ویژه برای کاربردهای مالی یا علمی مفید باشد. Pandas همچنین از مدیریت داده‌ها در فرمت‌های مختلف، از جمله CSV، Excel، پایگاه‌های داده SQL و JSON پشتیبانی می‌کند.

مزیت دیگر پانداها توانایی آن در ادغام با سایر کتابخانه‌های محبوب پایتون، مانند Matplotlib برای تجسم داده‌ها و Scikit-learn برای یادگیری ماشین است.

به طور خلاصه، کتابخانه Pandas یک مجموعه ابزار قدرتمند و بصری برای کار با داده‌های برچسب گذاری شده در پایتون ارائه می‌دهد. مجموعه ای غنی از روش‌ها و ویژگی‌های آن، آن را به ابزاری ضروری برای تحلیلگران داده، دانشمندان و توسعه دهندگانی که با مجموعه داده‌های دنیای واقعی کار می‌کنند، تبدیل می‌کند. فرقی نمی‌کند با داده‌های مالی، داده‌های علمی یا هر نوع دیگر از داده‌های ساختاریافته کار می‌کنید، پانداها ابزاری ضروری است که می‌تواند به شما کمک کند داده‌هایتان را به راحتی تمیز، تغییر داده و تجزیه و تحلیل کنید.

خوشه بندی

خوشه‌بندی یک تکنیک یادگیری بدون نظارت محبوب در یادگیری ماشینی و تجزیه و تحلیل داده است که شامل گروه‌بندی اشیاء مشابه با هم در خوشه‌ها بر اساس معیارهای شباهت یا متریک فاصله است. هدف از خوشه بندی یافتن ساختار معنادار در داده‌های بدون برچسب و شناسایی الگوها و روابط بین اشیاء است.

در اصل، خوشه‌بندی راهی برای تقسیم‌بندی داده‌ها به گروه‌ها یا خوشه‌ها است، به‌گونه‌ای که شباهت در خوشه‌ها را به حداکثر می‌رساند و شباهت بین خوشه‌ها را به حداقل می‌رساند. نتیجه خوشه‌بندی مجموعه‌ای از خوشه‌ها است که هر کدام شامل گروهی از اشیاء مشابه یکدیگر و بی‌شبهه با اشیاء در خوشه‌های دیگر است.

الگوریتم‌های خوشه‌بندی با تخصیص نقاط داده به خوشه‌ها بر اساس شباهت یا اندازه‌گیری فاصله کار می‌کنند. انتخاب معیار تشابه به ماهیت داده‌ها و مشکل خاص در دست بستگی دارد. به عنوان مثال، اگر داده‌ها متشکل از متغیرهای عددی باشند، یک اندازه گیری فاصله رایج، فاصله اقلیدسی است. اگر داده‌ها متشکل از متغیرهای طبقه بندی شده باشند، اندازه گیری فاصله مانند فاصله جاکارد ممکن است مناسب‌تر باشد.

یکی دیگر از الگوریتم‌های خوشه بندی محبوب، خوشه بندی سلسله مراتبی است. خوشه‌بندی سلسله مراتبی شامل ایجاد سلسله مراتبی از خوشه‌ها با ادغام یا تقسیم متوالی خوشه‌ها بر اساس معیار شباهت است. دو نوع اصلی خوشه بندی سلسله مراتبی وجود دارد: تجمعی و تقسیمی. خوشه‌بندی انباشته‌ای با هر نقطه داده به‌عنوان خوشه خاص خود شروع می‌شود و به‌طور متوالی خوشه‌ها را تا زمانی که همه نقاط داده متعلق به یک خوشه واحد باشند، ادغام می‌کند. خوشه‌بندی تقسیم‌بندی با تمام نقاط داده در یک خوشه شروع می‌شود و متوالی خوشه‌ها را تا زمانی که هر نقطه داده متعلق به خوشه خودش باشد، تقسیم می‌کند.

خوشه بندی در طیف گسترده ای از برنامه‌ها، از جمله تقسیم بندی مشتری، تشخیص ناهنجاری، تقسیم بندی تصویر، خوشه بندی اسناد و موارد دیگر استفاده می‌شود. این ابزار مفیدی برای کاوش داده، تشخیص الگو و داده کاوی است و می‌تواند به کشف ساختارهای پنهان در مجموعه داده‌های بزرگ و پیچیده کمک کند.

با این حال، خوشه بندی همیشه کار ساده ای نیست و می‌تواند تحت تأثیر انتخاب الگوریتم، تعداد خوشه‌ها و معیار تشابه مورد استفاده قرار گیرد. مهم است که این عوامل را هنگام اعمال خوشه بندی برای مسائل دنیای واقعی به دقت در نظر بگیرید و نتایج خوشه بندی را با استفاده از معیارهای مناسب مانند ضریب شبح یا شاخص دیویس-بولدین ارزیابی کنید.

به طور خلاصه، خوشه‌بندی یک تکنیک قدرتمند و پرکاربرد برای یادگیری بدون نظارت و تجزیه و تحلیل داده‌ها است که شامل گروه‌بندی اشیاء مشابه با هم به خوشه‌ها بر اساس معیارهای شباهت یا متریک فاصله است. این ابزار مفیدی برای کشف الگوها و روابط در مجموعه داده‌های بزرگ و پیچیده است و می‌تواند برای طیف گسترده ای از مشکلات در حوزه‌های مختلف اعمال شود.

الگوریتم k-means

الگوریتم k-means یک الگوریتم خوشه‌بندی محبوب است که برای تقسیم یک مجموعه داده به k خوشه مجزا بر اساس شباهت نقاط داده استفاده می‌شود. الگوریتم k-means یک روش ساده، سریع و کارآمد برای خوشه بندی داده‌ها است و به طور گسترده در علم داده، یادگیری ماشین و برنامه‌های کاربردی تشخیص الگو استفاده می‌شود.

الگوریتم k-means با تخصیص مکرر هر نقطه داده به خوشه‌ای که مرکز آن به آن نزدیک‌تر است، کار می‌کند و سپس مرکز هر خوشه را بر اساس میانگین نقاط داده‌ای که به آن اختصاص داده شده است، دوباره محاسبه می‌کند. این روند تا زمانی ادامه می‌یابد که مراکز خوشه‌ها دیگر تغییر نکنند یا معیارهای همگرایی دیگر برآورده شود.

الگوریتم k-means را می‌توان در مراحل زیر خلاصه کرد:

- مقداردهی اولیه: به طور تصادفی k مرکز خوشه اولیه را از نقاط داده انتخاب کنید.
- تخصیص: هر نقطه داده را بر اساس برخی متریک فاصله، معمولاً فاصله اقلیدسی، به نزدیک‌ترین مرکز خوشه اختصاص دهید.
- محاسبه مجدد: مرکز هر خوشه را بر اساس میانگین نقاط داده اختصاص داده شده به آن دوباره محاسبه کنید.
- مراحل 2 و 3 را تا زمان همگرایی تکرار کنید: مراحل 2 و 3 را تا زمانی تکرار کنید که تخصیص خوشه‌ها و مراکز خوشه دیگر تغییر نکنند یا معیارهای همگرایی دیگر برآورده شود.

الگوریتم k-means را می‌توان برای طیف گسترده ای از مسائل، از جمله تقسیم بندی تصویر، تقسیم بندی مشتری، تشخیص ناهنجاری و بسیاری دیگر اعمال کرد. این الگوریتم ساده، کارآمد است و می‌تواند مجموعه داده‌های بزرگ را با ویژگی‌های زیادی مدیریت کند.

با این حال، الگوریتم k-means دارای محدودیت‌ها و اشکالاتی است. یکی از محدودیت‌های اصلی این است که کاربر باید تعداد خوشه‌ها، k را از قبل مشخص کند. این می‌تواند در عمل یک کار دشوار باشد، به خصوص زمانی که داده‌ها ابعاد بالایی دارند یا خوشه‌ها به خوبی از هم جدا نشده اند. علاوه بر این، الگوریتم k-means به مراکز خوشه اولیه حساس است و می‌تواند در حداقل‌های محلی گیر کند و در نتیجه خوشه بندی بهینه نیست.

چندین تکنیک وجود دارد که می‌توان برای رفع این محدودیت‌ها و بهبود عملکرد الگوریتم k-means استفاده کرد. به عنوان مثال، یک رویکرد استفاده از گونه‌ای از k-means به نام الگوریتم k-means++ است که مراکز خوشه اولیه را به روشی هوشمندتر انتخاب می‌کند تا از گیر کردن در حداقل‌های محلی جلوگیری شود. روش دیگر استفاده از یک الگوریتم خوشه بندی متفاوت، مانند خوشه بندی سلسله مراتبی یا DBSCAN است که نیازی به تعیین تعداد خوشه‌ها از قبل توسط کاربر ندارد.

به طور خلاصه، الگوریتم k-means یک الگوریتم خوشه بندی پرکاربرد و کارآمد است که می‌تواند برای طیف گسترده ای از مسائل اعمال شود. با این حال، محدودیت‌ها و اشکالاتی دارد، از جمله نیاز به تعیین تعداد خوشه‌ها از قبل و حساسیت به مراکز اولیه خوشه. هنگام استفاده از الگوریتم k-means در عمل، بررسی و ارزیابی دقیق نتایج خوشه بندی ضروری است.

روش‌شناسی

دستور perf :

برای اینکه بتوانیم system call های موردنظر را جمع‌آوری کنیم، نیاز است از کتابخانه perf استفاده کنیم که به‌صورت اجمال، با اجرا کردند خط زیر در command line می‌توان شروع کرد:

```
perf
```

در اکثر مواقع، ما نیاز داریم با کمک این کتابخانه، داده‌هایی را ثبت یا ضبط کنیم که متعلق به برنامه‌هایی است که در حالت عادی در دسترس ما نیستند. بدین منظور ما نیاز است دستور بالا را همراه با عبارت sudo استفاده کنیم که با کمک آن، سطح دسترسی ما در سیستم‌عامل، به‌عنوان administrator لحاظ می‌شود و قابلیت انجام تمامی کارهای موردنیاز را داریم:

```
sudo perf
```

هدف ما ثبت یک‌سری فرایند است که با کمک دستور record که کتابخانه perf موجود است، این فرایند را ادامه می‌دهیم:

```
sudo perf record
```

به‌وسیله این دستور، ما با کمک لایبرری perf شروع به ثبت این یک‌سری وقایع می‌کنیم.

حال نیاز داریم این eventها را انحصاراً بر روی system call یک پراس قرار دهیم. با کمک پرچم e می‌توان به کتابخانه perf نشان داد که ما به دنبال ثبت eventهایی با محوریت system call هستیم. روند این کار با فرمان زیر صورت می‌گیرد:

```
sudo perf record -e 'syscalls:sys_*'
```

در مرحله بعدی، باید مشخص کنیم این ضبط‌کردن eventهایی که مدنظر ما هست، با چه فرکانسی انجام شود. منطقاً با افزایش این عامل، نمونه‌هایی بیشتری در ثانیه می‌توان ضبط کرد. در اینجا ما مقدار ۹۹ را برای این عبارت در نظر گرفتیم که آن را با پرچم f به کتابخانه perf معرفی می‌کنیم و به فرمت زیر می‌توان از آن بهره برد:

```
&& sudo perf record -e 'syscalls:sys_*' -F 99
```

قدم بعدی این است که مشخص کنیم کدام برنامه را می‌خواهیم به‌وسیله این کتابخانه مورد بررسی قرار دهیم. با کمک پرچم process identifier ، p را که می‌خواهیم آن را مورد بررسی قرار دهیم به این کتابخانه انتقال می‌دهیم، ولی نکته مهم این است که قبل از آن نیاز به پیداکردن این process identifier ها داریم و در کنار آن باید بدانیم که هر برنامه ممکن است تعداد متعددی process identifier داشته باشد. به کمک دستور pidof، در command line می‌توانیم تمامی process identifier های یک برنامه را دریافت کنیم. در بررسی که ما انجام

دادیم، نرم‌افزار discord را به‌عنوان ملاک اصـــلی برای بررســـی قرار دادیم که این برنامـه بـه طور معمول دارای ۴ process identifier بود :

حال که ما pidهای متناظر را برای برنامه خود (discord) یافتیم، باید این pidها را با اسـتفاده از پرچم p که در بالاتر توضیح دادم مورداستفاده قرار بدهیم:

```
sudo perf record -e 'syscalls:sys_*' -F 99 -p "pid"
```

در مرحله آخر نیز تنها باید مشـخص کنیم که این ضـبط eventها چه مدت صـورت گیرد و این مورد را با پرچم sleep به این کتابخانه معرفی می‌کنیم و نوع استفاده از به شکل زیر است:

```
sudo perf record -e 'syscalls:sys_*' -F 99 -p "pid" sleep "seconds"
```

به طور مثال با فرض برابر بودن بود process identifier با 1000 و زمان ضـبط داده‌ها با ۱۰ ثانیه، شـکل نهایی آن به این صورت خواهد بود:

```
sudo perf record -e 'syscalls:sys_*' -F 99 -p 1000 sleep 10
```

با هر بار اجرا برنامه بالا فایلی به اسم perf.data دریافت خواهیم کرد که شامل تمامی اطلاعاتی است که توسط دستور بالا دریافت شده است.

توسط دستور زیر می‌توان مشاهده کرد که خروجی تهیه شده توسط دستور بالا به چه شکلی خواهد بود.

در ادامه برای راحتی کار، ما این فایل را به فرمت text تبدیل می‌کنیم و تا بررسی مواردی که می‌خواهیم در نظر داشته باشیم دیگر نیازمند استفاده از کتابخانه perf نباشد. این کار به‌سادگی توسط دستور زیر انجام می‌شود:

```
sudo perf script > perf.data.txt -f
```

حال، فایل تکسـتی در اختیار داریم که در هر خط آن به ترتیب اسم پراسـسی که یک system call را صدا زده اسـت، process identifier این پراسس، time step که این پراسـس در آن این system call را فراخوانده اسـت و در نهایت اسـم system call مربوطه را داریم که در قسـمت بعدی، باید بتوانیم با اسـتفاده از این داده‌هایی که داریم، یک ساختار مناسب برای گروه‌بندی این موارد بکنیم که خوشه‌بندی را بر روی آن بتوان اعمال کرد.

قبل از اجرا این فرایند باید به این نکته توجه داشـت که برنامه اصـلی discord شـامل چندین پراسـس متفاوت اسـت که هرکدام از آنها در کنار دیگری در حال فراخوانی system call ها هســـتند و به‌صـــورت یکپارچه عمل می‌کنند؛ بنا بر همین موضوع، ما در هر لحظه که قصد داریم یک ضبطی برای system call ها را اعمال کنیم باید به‌صورت هم زمان، برای تمامی پراسـس‌هایی که برنامه discord دارد، این کار را اعمال کنیم. در غیر این صـورت اگر به‌صـورت سـری، برای هرکدام از پراســـس‌ها این عملیات را انجام دهیم، به‌ازای هر پراســـس، یک‌ســـری داده داریم که هر گروه از آنها، به گروه بعدی ربطی ندارد و این موضوع باعث می‌شود که خوشه‌بندی ما نتواند به‌درستی کار کند.

برای اجرایی کردند این فرایند، با threadها کمک می‌گیریم. به این صورت عمل می‌کنیم که به‌ازای هر پراسس، یک thread درست می‌کنیم و به‌عنوان آرگومان ورودی آن، شماره پراسسی را که می‌خواهیم آن thread شروع به ضبط کردند با آن بکند را به آن می‌دهیم. به این صورت، ما می‌توانیم به طور هم‌زمان، system call های تمامی پراسس‌های discord را ضبط بکنیم.

تجزیه‌کننده (parser):

حال بعد از ساخت فایل‌های تکست برای هر ضبطی که انجام دادیم، آنها را به‌صورت یک دیکشنری در میاوریم. این کار را باید با پارس (parse) کردن فایل تکست خود انجام دهیم. برای درک بهتر این موضوع بهتر است در ابتدا فرمت یک فایل تکستی را که تولید کردیم را ببینیم:

```
Discord  3186 [003]   495.056872:                    syscalls:sys_exit_poll: 0x1
```

هر خط از قایل تکست را می‌توان به با کد زیر، استخراج کرد:

```
file = pidPath+"/"+pid+"/perf.data.txt"
f = open(file, "r")
lines = f.readlines()
```

ابتدا مسیری که فایل تکست در آن داده شده است را به تابع ()open می‌دهیم و سپس با فقط دسترسی read (خواندن) آن را باز می‌کنیم. در انتها با کمک تابع readlines تمامی خطوط آن را در یک تغییر ذخیره می‌کنیم.

از فایل تکست بالا ما تنها دو عبارت را نیاز است که داشته باشیم:

- اسم system call در این خط که در اینجا sys_exit_poll است.
- اسم پراسسی که این system call را صدا زده است که در اینجا Discord است.

در ابتدا، راحت‌ترین کار، استخراج اسم این system call است. اگر توجه کنیم می‌بینیم که اسم آن دقیقاً بعد از عبارت syscalls آمده است. برای همین، در ابتدا هر خط را با ":" به قسمت‌های کوچک‌تری تقسیم می‌کنیم و آن‌ها را درون یک آرایه ذخیره می‌کنیم. این کار به‌صورت زیر انجام می‌شود:

```
split = line.split(":")
```

برای مثال بالا، حاصل به‌صورت زیر است:

- Discord 3186 [003]   495.056872
- Syscalls
- sys_exit_poll
- 0x1

با توجه با خروجی بالا، برای برداشـــتند اســـم این system call ،هم می‌توانیم خانه دوم آرایه‌ای که در بالاتر تعریف کردیم را به‌عنوان اسم ذخیره کنیم و هم اینکه ببینیم کدام یک از خانه‌های آرایه، عبارت Syscalls در آن آمده است. با پیداکردن شماره خانه آن، خانه بعدی را به‌عنوان اسم این system call در نظر می‌گیریم.

در اینجا برای اینکه عملکرد فرایندی که درست کردیم، بهتر باشد، از روش دوم که در بالا گفته شد استفاده کردیم.

```
if("syscalls" in split[counter_]):
      sysName = split[counter_+1]
```

برای بهبود عملکرد و سرعت ضبط فرایندها، فقط system call هایی را ضبط می‌کنیم که به‌صورت enter باشند. حال برای برداشت اسم پراسس مربوط باید از یک نوع parser جدید استفاده کنیم. برای این کار، هر خط را بر اساس کاراکتر "]" جداسازی می‌کنیم:

```
if "enter" in sysName:
   key = split[0].replace(" ","").split("[")[0]
```

خروجی این parser برای متن نمونه بالا، به‌صورت زیر خواهد بود:

- Discord 3186
- 003] 495.056872: syscalls:sys_exit_poll: 0x1

در اینجا ما تنها نیاز به عبارت اولی هســتیم که در بالانشــان دادیم؛ ولی همچنان یک بخش عددی در آن وجود دارد که موردنیاز ما نیســت. در اینجا باتوجه‌به خط زیر، می‌توانیم این بخش عددی را حذف کنیم و فقط عبارت Discord را نگه داریم:

```
key_ = ''.join(i for i in key if not i.isdigit())
```

با انجام خط بالا، تنها عبارتی که باقی می‌ماند که ما آن را به‌صــورت key نگه می‌داریم، در مثال بالا، Discord اســت. حال باید این system call ها را به دیکشنری که برای این اسامی درست کردیم اضافه می‌کنیم.

```
if key_ in sysCall:
  sysCall[key_].append(sysName)
else:
  sysCall[key_] = [sysName]
```

تنها نکته این است این دیکشنری که درست کردیم، دارای کلیدهایی است که اسم پراسس‌های ضبط شده است و مقدارهای هر کلید، system call هایی است که توسط آن پراسس مربوطه صدا زده شده است. چیزی که ما نیاز داریم، تعداد هرکدام از این system call ها است که در این دیکشنری‌ها ذخیره شده است. برای این کار از دستور زیر استفاده می‌کنیم:

```
unique, counts = np.unique(np.array(sysCall[key]), return_counts=True)
hold = dict(zip(unique, counts))
```

با انجام این کار، ما یک دیکشـنری داریم که کلید آن، اسـم system call های ما درون آن اسـت و مقدار آن، برابر با تعداد تکرار هرکدام از آن system call ها است. حال ما باید یک ستون دیگر به این داده‌ها اضافه کنیم که اسم همان پراسسی است که این system call ها را فراخوانده است. این روند را توسط خطوط زیر عملیاتی می‌کنیم:

```
d = hold.copy()
d.update({"name":key})
df_hold =pd.DataFrame.from_dict([d])
df_hold =df_hold[ ['name'] + [ col for col in df_hold.columns if col != 'name' ] ]
df.append(df_hold)
```

مدیریت کردند فریم‌ها با کتابخانه panda

حال برای راحتی کار، لیسـت بالا را که مجموعه از یک‌سـری کلید اسـت را درون یک فریم از کتابخانه پاندا می‌کنیم. تنها نکته این است که ما به‌ازای هر نام پراسسی که داریم، یک فهرست درست کرده بودیم که شرایط بالا را داشت، حال باید وقتی که این لیسـت‌ها را به فریم تبدیل می‌کنیم، همه آن‌ها را نیز باید با همدیگر ادغام کنیم که به‌صـورت زیر این کار انجام می‌شود:

```
output = pd.DataFrame()
for hold in df:
    output = pd.concat([output, hold], ignore_index=True)
```

رکوردهایی که در قسمت‌های قبلی، ضبط کردیم، به ۴ بخش تقسیم می‌کنیم:

1. idle-homepage: در زمان ضبط‌کردن برای این گروه، در صفحه شخصی کاربرد و به‌صورت idle بوده است.
2. typing: در حال تایپ‌کردن در چت باکس، در چت شخصی با یک اکانت دیگر.
3. idle-inChannel: در حال دیدن چت‌های کاربران موجود در یک چنل.
4. voice-chat: در حال تماس صوتی با یک اکانت دیگر.

تمامی گروه‌های بالا دارای ۱۰ رکورد هستند که با بَسامد ۹۹ هرتز و به مدت ۵ ثانیه، رکورد شده است.

به از انجام تمامی موارد بالا، ما یک‌سری پیرابند برای هرکدام از رکوردها داریم که هر سطر از آن، نشان‌دهنده تعداد تکرار یک system call به‌خصوص توسط یک پراسس منحصربه‌فرد است.

به طور مثال، نمونه زیر، یکی از نتایج رکوردها است که بعضی از system call ها و اسم پراسس‌های مربوط به آن، نمایش‌داده‌شده است.

name	sys_enter_clock_gettime	sys_enter_close	sys_enter_futex	sys_enter_getpid	sys_enter_getrandom
AudioEncoder	0.0	0.0	0.0	0.0	0.0
AudioOutputDevi	0.0	0.0	0.0	0.0	0.0
CacheThread_Blo	0.0	0.0	0.0	0.0	0.0
Chrome_ChildIOT	0.0	0.0	44.0	0.0	0.0
Chrome_IOThread	0.0	0.0	0.0	0.0	0.0
Compositor	40.0	0.0	40.0	0.0	0.0
CompositorTileW	0.0	0.0	160.0	0.0	0.0
DedicatedWorker	0.0	0.0	0.0	0.0	0.0
Discord	8.0	20.0	89.0	60.0	19.0

شکل 1.نمونه ای از ریکرد های ضبط شده

تنظیم بُعد داده‌ها:

نکته مهمی که به آن باید توجه داشت، این است که بُعد (dimension) هر رکوردی که صورت فریم داریم، یکسان باشد. این موضوع برای اینکه بتوانیم خوشه‌بندی مناسبی داشته باشیم، الزامی است. دلیل این موضوع هم این است که در الگوریتم k-means، ما نیاز است داده‌هایی که برای آموزش الگوریتم به آن می‌دهیم دارای بعدها یکسان باشد و مشخصات هر سطر، مانند سطر متناظر در رکوردهای دیگر باشد. به طور مثال در همه رکوردها، خانه (1,1) مربوط به تعداد تکرارهای system call با اسم sys_enter_clock_gettime که توسط پراسس AudioEncoder صدا زده شده است.

برای تنظیم‌کردن این موضوع، ابتدا تمامی اسامی system call هایی که در رکوردهای ما آماده است را در یک فهرست به اسم columns_name ذخیره می‌کنیم.

برای این موضوع، تمامی ۴۰ رکوردی که داریم را باید بررسی کنیم و تمامی system call ها موجود در آنها را در فهرستی ذخیره کنیم. نکته مهم این است که هر گروه از رکوردها را در task_out ها ذخیره کرده بودیم که هرکدام از این گروه‌ها، دارای ۱۰ رکورد است که با استفاده از دستور columns. می‌تواند لیست columns های موجود در آن فریم را دریافت کنیم. بررسی می‌کنیم که اگر موردی در این columnsها بود که در لیست موجود نبود، آن را به لیست اضافه می‌کنیم و در غیر این صورت و در حالتی که در لیست بود، آن را به لیست اضافه نمی‌کنیم. نحوه انجام این فرایند، در کد زیر قابل‌بررسی است.

```
columns_name = []
for task in task_out:
  for rec in task:
    for item in rec.columns:
      if item not in columns_name:
        columns_name.append(item)
```

در مرحله بعدی، فرایندی مانند بالا را باید برای پیدا کردند تمامی اسامی ممکن برای پراسس‌های موجود انجام داد. در اینجا نیز باید تمامی ۴۰ رکوردی که داریم را باید بررسی کنیم و تمامی پراسس‌های موجود در آنها را در لیستی ذخیره کنیم. با استفاده از ["name"] می‌تواند لیست پراسس‌های موجود در آن فریم را دریافت کنیم. بررسی می‌کنیم که اگر موردی در این اسامی بود که در لیست موجود نبود، آن را به لیست اضافه می‌کنیم و در غیر این صورت و در حالتی که در لیست بود، آن را به لیست اضافه نمی‌کنیم. نحوه انجام این فرایند، در کد زیر قابل‌بررسی است.

```
proc_name = []
for task_counter in range(len(task_out)):
  for rec_counter in range(len(task_out[task_counter])):
    for proc in task_out[task_counter][rec_counter]["name"]:
      if proc not in proc_name:
        proc_name.append(proc)
```

حال ما تمامی اسامی system call ها و پراسس‌هایی که در رکوردهایمان داشتیم را در لیست‌هایی که بالاتر تعریف کردیم، ذخیره کردیم. حال می‌دانیم که فریم‌های ما تمامی اسامی system call ها و پراسس‌های موجود را دارا نیستند. پس در مرحله بعدی، باید این موارد را در فریم‌هایی که دارای این موارد نیستند، اضافه کنیم که در آخر، شکل و بُعد همه فریم‌ها عین هم شود.

برای بررسی این مورد برای پراسس‌ها، همانند بالا، باید تمامی ۴۰ رکورد را بررسی کرد در هرکدام چک کرد که آیا پراسسی وجود دارد که در لیست اصلی پراسس‌ها نباشد. در صورت بودن همچنین حالتی، این پراسس به فریم آن رکورد اضافه می‌شود. این کار به این صورت انجام می‌شود که هر پراسسی که موجود نباشد به یک لیستی اضافه می‌شود و در اهر هر for، این لیست به فریم آن رکورد اضافه می‌شود. بعد از به‌روزرسانی تمامی حالات، به فریم اصلی رکوردها، این فریم اضافه می‌شود.

```
for task_counter in range(len(task_out)):
  for rec_counter in range(len(task_out[task_counter])):
    holdProc = []
    for proc in proc_name:
      if proc not in task_out[task_counter][rec_counter]["name"].tolist():
        holdProc.append(proc)
    df = pd.DataFrame(holdProc,columns =['name'])
    task_out[task_counter][rec_counter] =
task_out[task_counter][rec_counter].append(df, ignore_index = True)
```

مرحله بعدی، تنظیم‌کردن اسامی system call ها در هر پیرابند هستند. مانند بالا عمل می‌کنیم و تمامی ۴۰ پیرابندی که داریم را باید بررسی کنیم که کدام system call ها در آن نیامده است و در صورت موجود نبودن system call ای در آن، باید این system call را به این اضافه کنیم. به این صورت عمل می‌کنیم که یک فریم با اسامی system call ها تعریف می‌کنیم. در هر مرحله گردش در for، این فریم را، با ۴۰ فریم اصلی که داریم concate می‌کنیم. این دستور به این صورت عمل می‌کند که در خروجی آن، تمامی system call ها در فریم نهایی باشند. نکته مهم این است که برای این که ترتیب system call ها در همه فریم‌ها یکسان باشد، بعد از concate کردن فریم‌ها، آنها را بر اساس فیلد "name" ای که در فریم‌ها موجود است، sort می‌کنیم. فریم نهایی ما در این ۴۰ رکورد به‌اندازه‌ای برابر با (31 rows × 87 columns) خواهد بود.

```
df_ = pd.DataFrame(columns=columns_name)

for task_counter in range(len(task_out)):
  for rec_counter in range(len(task_out[task_counter])):
    task_out[task_counter][rec_counter] =
pd.concat([df_,task_out[task_counter][rec_counter]],
ignore_index=True).sort_values(by=['name']).reset_index(drop=True).fillna(0)
```

اجرای الگوریتم خوشه‌بندی k-means:

در این مرحله، بعد از مرتب‌سازی اطلاعات و رکوردها، از کتابخانه sklearn برای الگوریتم KMeans استفاده کردیم. این تابع در ورودی خود یک numpy array دریافت می‌کند که حتماً بُعد دوم آن باید ۱ باشد. برای همین موضوع باید فریم‌هایی که در مراحل بالا آماده کردیم را باید reshape بکنیم تا بُعد دوم آن، برابر با ۱ باشد. برای این کار از تابع به فرم reshape(-1). استفاده می‌کنیم که این کار را برای ما انجام می‌دهد. بعد از تغییر شکل هر فریم، آن را درون لیست‌هایی قرار می‌دهیم که بتوانیم آن‌ها را به تابع KMeans بدهیم.

```
task_out_list = []
main = []
for task_counter in range(len(task_out)):
  rec_out_list=[]
  for rec_counter in range(len(task_out[task_counter])):
    hold = task_out[task_counter][rec_counter].drop('name',
axis=1).to_numpy().reshape(-1)
    rec_out_list.append(hold)
    main.append(hold)
  task_out_list.append(rec_out_list)
```

در مرحله آخر، ما فهرســـت‌هایی که فراهم کردیم را به numpy array به کمک تابع ()array تبدیل می‌کنیم. تابع KMeans علاوه بر داده‌هایی که در ورودی دریافت می‌کند، باید تعداد خوشه‌هایی که می‌خواهیم این الگوریتم به تعداد

```
X = np.array(main)
kmeans = KMeans(n_clusters=4).fit(X)
kmeans.labels_
```

آن‌ها خوشه‌بندی کند را به‌عنوان ورودی نیز به آنها بدهیم. در آخرسر با دستور _lalels. می‌توانیم نتایج خوشه‌بندی را بررسی کنیم.

خروجی دستور _labels. به‌صورت زیر است :

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3,3, 3, 3,
3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

ملاحظه شود که خوشه‌بندی به طور کاملاً مناسب انجام شده است. ۴۰ رکوردی را که متعلق به ۴ گروه اصلی بودند را، ما به ترتیب به این الگوریتم خوشـــه‌بندی دادیم و در خروجی این الگوریتم می‌توان دید که این ۴ گروه به‌درســـتی از هم تفکیک شده‌اند.

بررسی عملکرد خوشه‌بندی:

روش‌های مختلفی برای اندازه گیری عملکرد خوشـــه بندی بر روی داده‌های قطار وجود دارد. در اینجا چند روش رایج وجود دارد:

امتیاز silhouette: امتیاز silhouette کیفیت خوشـه بندی را با محاســبه فاصـله بین نقاط داده در یک خوشـه و فاصله بین نقاط داده در خوشه‌های مختلف اندازه گیری می‌کند. امتیاز سیلوئت بالاتر نشان دهنده خوشه‌های با تعریف بهتر اســت. برای محاســبه امتیاز silhouette در داده‌های قطار، می‌توانید از تابع silhouette_score از کتابخانه scikit-learn استفاده کنید.

شاخص Calinski-Harabasz: شاخص Calinski-Harabasz معیاری از کیفیت خوشه ای است که واریانس بین خوشه ای را با واریانس درون خوشه ای مقایسه می‌کند. شاخص Calinski-Harabasz بالاتر نشان دهنده خوشه‌های با تعریف بهتر اســـت. برای محاســـبه شـــاخص Calinski-Harabasz بر روی داده‌های قطار، می‌توانید از تابع calinski_harabasz_score از کتابخانه scikit-learn استفاده کنید.

شاخص Davies-Bouldin: شاخص Davies-Bouldin یکی دیگر از معیارهای کیفیت خوشه است که فاصله بین مرکزهای خوشه را با فاصله بین نقاط داده در یک خوشه مقایسه می‌کند. شاخص Davies-Bouldin پایین‌تر نشان

دهنده خوشه‌های با تعریف بهتر است. برای محاسبه شاخص Davies-Bouldin بر روی داده‌های قطار، می‌توانید از تابع davies_bouldin_score از کتابخانه scikit-learn استفاده کنید.

بازرسی بصری: همچنین می‌توانید به صورت بصری خوشه‌ها را برای ارزیابی عملکرد خوشه بندی بررسی کنید. می‌توانید از ابزارهایی مانند نمودارهای پراکنده، هیستوگرام، یا نقشه‌های حرارتی برای تجسم خوشه‌ها استفاده کنید و بررسی کنید که آیا آن‌ها منطقی هستند یا اینکه نقاط پرت یا اشتباه طبقه‌بندی شده وجود دارد.

توجه به این نکته مهم است که ارزیابی عملکرد خوشه‌بندی روی داده‌های قطار همیشه قابل اعتماد نیست، زیرا ممکن است مدل بیش از حد با داده‌ها مطابقت داشته باشد و در داده‌های جدید و دیده نشده ضعیف عمل کند. بنابراین، ارزیابی عملکرد خوشه‌بندی در یک مجموعه آزمایشی جداگانه یا با استفاده از تکنیک‌های اعتبارسنجی متقابل توصیه می‌شود.

یکی از مهم‌ترین سؤالاتی که می‌توان در اینجا بررسی کرد این است که آیا واقعاً این ۴۰ رکورد ما، در ۴ گروه باید خوشه‌بندی شوند؟ منطقاً به دلیل اینکه در ابتدا ما ۴ گروه داده مختلف درست کردیم، انتظار این را داریم که این موضوع را ببینیم؛ ولی آیا در عمل هم به همین‌گونه صورت‌گرفته است؟

برای بررسی این موضوع از تابعی به اسم inertia استفاده می‌کنیم. این تابع اندازه‌گیری می‌کند که چگونه یک مجموعه‌داده توسط K-Means خوشه‌بندی‌شده است. این کار به این روش انجام می‌شود که با اندازه‌گیری فاصله بین هر نقطه داده و مرکز آن، مجذور کردن این فاصله و جمع‌کردن این مربع‌ها در یک خوشه محاسبه می‌شود که خروجی این تابع چه مقدار است. یک مدل خوب مدلی با اینرسی کم و تعداد خوشه کم (K) است.

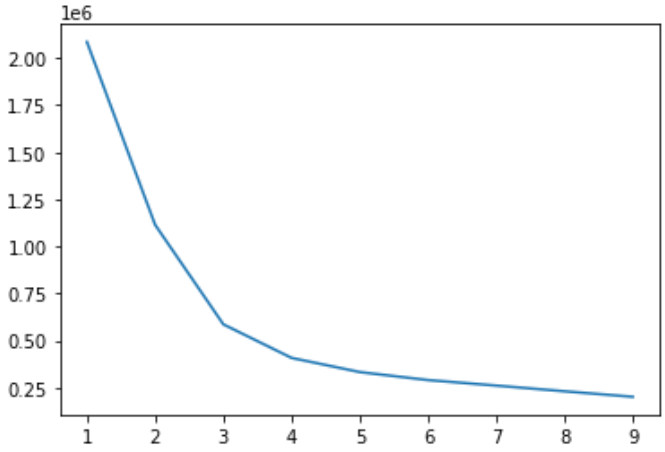
برای اجرایی کردند این موضوع، داده‌هایی که داریم را در بازه ۱ تا ۱۰ خوشه‌بندی می‌کنیم که این مقدارها گفته شده، تعداد خوشه‌های هدف هر خوشه‌بندی هستند. در هر بار خوشه‌بندی، خروجی الگوریتم k-means را بعد از اجرای تابع در یک آرایه ذخیره می‌کنیم که در انتها نمودار مربوطه را بکشیم.

```
sse = []
for i in range(1,10):
    kmeans = KMeans(n_clusters=i , max_iter=300)
    kmeans.fit(X)  # <- fit here.....
    sse.append(kmeans.inertia_)
```

حال ما یک آرایه از خروجی‌های این تابع برای تعداد خوشه‌های بین ۱ تا ۱۰ هستیم و تنها کافی است نمودار مربوط به این ۱۰ حالت را رسم کنیم. با تحلیل روی آن بررسی کنیم مقدار خوشه مناسب برای این مسئله چه مقداری است.

```
plt.plot(range(1,10),sse)
plt.show()
```

خروجی عبارات بالا، نمودار زیر خواهد بود:



شکل2 : نمودار inertia

برای بررسی اینکه چه مقدار برای در نظر گرفتند مقدار خوشه مناسب، باید انتخاب شود باید به شیب‌خط آبی‌رنگ در نمودار بالا نگاه کرد. هر جا که از آن جا به بعد، شیب‌خط تفاوتی آن‌چنانی با مقادیر بعدی خود ندارد، می‌توان این موضوع را حساب کرد. همان‌طور که ملاحظه می‌توان کرد (شکل2)، بعد از افزایش تعداد خوشه‌ها از ۴ به بعد، شیب‌خط به حدود خوبی، ثابت بوده است و به همین دلیل تعداد خوشه مناسب را برابر با ۴ در نظر می‌گیریم.

تعداد گروه‌های رکوردهای ما ۴ بوده است و همان‌طور که در بالانشان دادیم، تعداد خوشه‌های مناسب برای این رکوردها نیز ۴ بوده است. این موضوع نشان‌دهنده عملکرد خوب الگوریتم خوشه‌بندی را نشان می‌دهد.

برای بررسی عملکرد خوشه‌بندی‌شده، می‌توان از ۲ تابع زیر برای محاسبه امتیاز آن استفاده کرد:

- `transform` : این تابع ،فاصله هرکدام از نمونه‌ها را با مرکز خوشه‌هایی که داریم (در اینجا ۴ مورد است) را حساب می‌کند. در یک خوشه‌بندی خوب، انتظار داریم که فاصله یک نمونه که در یک خوشه مشخصی قرار دارد، با باقی خوشه‌ها به‌اندازه معقولی دور باشد.

جدول زیر، ۴ نمونه مختلف را نشان می‌دهد که هر ردیف، خروجی تابع `transform` را برای آن نمونه نشان می‌دهد. به طور مثال، برای ردیف اول با استفاده از این تابع و ورودی دادن نمونه اول به آن، ما ۴ مقدار خروجی دریافت کردیم. هرکدام از این ۴ مقدار، فاصله این نمونه با مرکز ۴ خوشه دیگر است. کمترین مقدار آن (که با رنگ خاکستری مشخص شده است) به‌عنوان خوشه مربوط به آن نمونه تلقی خواهد شد.

```
kmeans.transform([X[0],X[10],X[20],X[30]])
```

۴۱۶/۸۵۱۲۹۲۴۳	۴۳/۶۳۱۷۵۴۴۹	۳۶۲/۰۵۲۶۲۰۴۸	۱۹۶/۶۰۲۸۷۳۸۳
۷۴/۷۸۷۶۹۹۵۲	۴۴۹/۸۹۷۲۴۳۸۲	۳۵۰/۱۰۰۹۸۵۴۳	۳۴۷/۰۳۶۷۲۷۱۶
۳۲۲/۳۳۷۴۰۰۸۷	۲۱۲/۳۳۷۷۷۳۳۷	۳۴۸/۰۹۰۰۷۴۵۵	۶۶/۱۷۳۱۸۱۸۸
۳۳۷/۹۰۴۴۲۴۳۶	۳۷۸/۵۰۸۲۹۵۸۱	۷۷/۷۸۶۲۴۵۵۷	۳۴۸/۵۸۶۱۲۹۹۶

جدول 1. فاصله هر نمونه با خوشه های دیگر

می‌توان دید که فاصله هر نمونه با خوش‌های دیگر، به مقدار قابل‌قبولی متفاوت است که این موضوع نشان‌دهنده عملکرد خوبی خوشه‌بندی‌شده است.

- Score: هدف در K-means کاهش مجموع مربعات فواصل نقاط از مرکز خوشه مربوطه آنهاست. با استفاده از این تابع ما یک مقدار منفی را دریافت می‌کنیم که این مقدار نشان می‌دهد که خوشه‌ها چقدر از نظر داخلی منسجم هستند (هرچه کمتر بهتر است).
  با اجرا دستور زیر به نتیجه "-408404.80000000005" می‌رسیم.

```
kmeans.score(X)
```

با داشتن ۴۰ نمونه متفاوت، می‌توان گفت به طور میانگین، مقدار مربعات فواصل نقاط از مرکز خوشه آنها، برابر با ۱۰۲۱۰ است. این موضوع نیز نشان‌دهنده عملکرد خوب این الگوریتم خوشه‌بندی است.

نتیجه‌گیری:

در این پایان‌نامه مانشان دادیم که با داشتند رکوردهایی از system call ها برای یک عملیات خاص در یک برنامه (مانند discord) می‌توانیم یک خوشه‌بندی مناسب را ارائه دهیم که با آن، بتوانیم actionهای مختلفی را که در یک برنامه قابل‌انجام است را از همدیگر تفکیک کنم.

یکی از مهم‌ترین عملیات جایی که در اینجا انجام شده است، تهیه یک مجموعه‌داده مناسب و بدون خطا برای انجام الگوریتم خوشه‌بندی بر روی آن است. جمع‌آوری این مجموعه‌داده یکی از بزرگ‌ترین مراحل این پایان‌نامه بود که به دلیل موجود نبود نمونه داده در بستر اینترنت، باید به‌صورت کامل از اول جمع‌آوری می‌شد.

ما در ابتدا نحوه جمع‌آوری داده‌های را در بستر discord با کمک دستور perf توضیح دادیم و نشان دادیم برای استفاده و کارکرد مناسب باید از threadها برای این موضوع استفاده کرد. در ادامه نیاز بود که این رکورد در یک‌سری فریم مناسب برای مدیریت بهتر آنها ذخیره شوند که با به کمک کتابخانه panda این کار را انجام دادیم. نشان دادیم

برای اجرا الگوریتم خوشه‌بندی باید بعدهای ورودی‌های ما یکسان باشند تا بتوان بردارهای یکسانی را برای این الگوریتم آماده کرد. در انتها بعد از خوشه‌بندی کردن داده، نشان دادیم که خوشه‌بندی انجام شده و همین‌طور تعداد خوشه‌هایی که برای این منظور، مدنظر قرار دادیم، مناسب بوده‌اند.

با انجام فرایندهای بالا، می‌توان دید که با این عملکرد، ما می‌توانیم actionهای یک سیستم را خوشه‌بندی کنیم. بدین‌وسیله با داشتند که مجموعه‌داده مناسب از همه actionهای یک سیستم می‌توان با داشتند یک action که به ما داده شده است، تشخیص دهیم در کدام‌یکی از خوشه‌هایی که داریم قرار می‌گیرد.

امکانی که این روش به ما می‌دهد این است که با داشتند system call هایی که یک کاربر در طی یک فرایند انجام داده است، بدون داشتند هیچ اطلاع دیگری می‌توانیم تشخیص دهیم که فرایندی که کاربرانی‌ام داده است زیرمجموعه کدام یک از خوشه‌ها قرار می‌گرید و با داشتند برچسب آن خوشه، می‌فهمیم که آن action مربوط کاری بوده است.

کارهای آینده:

مهم‌ترین دلیل که از خوشه‌بندی داده‌ها در بررسی یک سیستم می‌توان به دستاورد، تمامی حالت‌هایی است که یک سیستم ممکن است در خود ایجاد کند. حال با داشتند که مجموعه‌داده کامل از تمامی actionهای موجود در یک برنامه، ما می‌توانیم یک action که به‌صورت نامعلوم به ما داده شده است را بررسی کنیم و تشخیص دهیم در کدام شاخه قرار دارد. به کمک این ایده ما می‌توانیم داده‌ها و actionهای غیرنرمال را در یک سیستم پیدا کنیم.

برای این کار بدین منظور عمل می‌کنیم که اگر یک action یک کاربر انجام داد و این عملیات در خوشه مناسبی قرار نمی‌گرفت و یا فاصله زیادی از همه آنها داشت، می‌توان نتیجه گرفت این عملیات یک عملیات غیرنرمال از طرف کاربر بوده است و به همین دلیل، این دیدگاه می‌تواند برای شناسایی الگوهای رفتاری برنامه/سیستم مختلف و همچنین رفتار غیرعادی یا بدخیم مفید باشد.

انجام خوشه بندی بر روی انواع مختلف برنامه‌ها می‌تواند بینش و اطلاعات ارزشمندی را ارائه دهد که می‌تواند برای بهینه سازی فرآیندهای توسعه نرم افزار مورد استفاده قرار گیرد. با گسترش داده‌هایی که ضبط می‌کنیم، مانند ثبت زمان اجرای هر تابع یا میزان استفاده از حافظه هر متغیر، می‌توانیم دید جامع‌تری از نحوه رفتار برنامه به دست آوریم. سپس می‌توان از این داده‌های اضافی برای شناسایی رکورد متعلق به کدام خوشه و حتی کدام عملیات در برنامه به هر خوشه استفاده کرد.

هنگامی که خوشه‌های مختلف را شناسایی کردیم، می‌توانیم ویژگی‌های هر خوشه را تجزیه و تحلیل کنیم تا درک عمیق‌تری از رفتار برنامه به دست آوریم. برای مثال، ممکن است متوجه شویم که خوشه‌های خاصی با انواع خاصی از عملیات مرتبط هستند یا خوشه‌های خاص بیشتر مستعد خطا یا مشکلات عملکرد هستند.

با درک رفتار برنامه در سـطح دانه ای، می‌توانیم برای بهینه سـازی کد و بهبود عملکرد آن گام برداریم. این ممکن اسـت شامل بازسازی کد، بهینه سازی الگوریتم‌ها یا ایجاد تغییرات دیگر برای بهبود کارایی و اثربخشی کلی برنامه باشد.

به طور خلاصه، انجام خوشه‌بندی بر روی انواع مختلف برنامه‌ها و گسترش داده‌هایی که ثبت می‌کنیم، می‌تواند بینش‌های ارزشـمندی در مورد رفتار برنامه ارائه دهد و ما را قادر می‌سـازد تا فرآیند توسـعه نرم‌افزار را بهینه کنیم و کیفیت کلی کد را بهبود بخشیم.

مراجع

Perf:

- Linux perf Examples (brendangregg.com)
- BPF Performance Tools (Book) (brendangregg.com)
- Systems Performance 2nd Edition Book (brendangregg.com)
- Tutorial - Perf Wiki (kernel.org)

Panda library:

- pandas - Python Data Analysis Library (pydata.org)

K-means:

- Understanding K-means Clustering in Machine Learning | by Education Ecosystem (LEDU) | Towards Data Science
- K-Means Clustering Explained - neptune.ai
- sklearn.cluster.KMeans — scikit-learn 1.2.1 documentation
- K means clustering python

Clustring:

- The 5 Clustering Algorithms Data Scientists Need to Know | by George Seif | Towards Data Science
- 10 Clustering Algorithms With Python - MachineLearningMastery.com
- How to Evaluate Clustering Models
- data clustering
- What is Cluster Analysis?