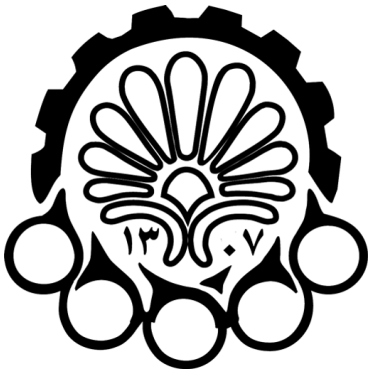


گزارش پروژه نهایی

تحلیل کلان‌داده

عارف عزیزیان

۲۲ تیر ۱۴۰۳

فهرست مطالب

مقدمه

هدف این پروژه توسعه یک سیستم توصیه‌گر مبتنی بر Collaborative Filtering است که از Locality-Sensitive Hashing (LSH) برای ارائه پیشنهادهای کارآمد استفاده می‌کند و در عین حال حریم خصوصی کاربران را حفظ می‌نماید. در دنیای امروز که اطلاعات به سرعت در حال گسترش است، سیستم‌های توصیه‌گر نقش مهمی در کمک به کاربران برای پیدا کردن محتوای مرتبط و مورد علاقه‌شان ایفا می‌کنند.

Collaborative Filtering یکی از روش‌های پرکاربرد در سیستم‌های توصیه‌گر است که بر اساس تجزیه و تحلیل الگوهای رفتاری کاربران، پیشنهادهایی را به آن‌ها ارائه می‌دهد. با این حال، یکی از چالش‌های اصلی در این روش، حجم زیاد داده‌ها و حفظ حریم خصوصی کاربران است. استفاده از LSH به ما امکان می‌دهد تا با کاهش پیچیدگی محاسباتی، همسایگان نزدیک کاربران را به صورت کارآمد پیدا کنیم و در عین حال اطلاعات حساس کاربران را محافظت کنیم.

علاوه بر این، در این پروژه، داده‌های اینستاگرام را به دقت جمع‌آوری کردیم تا سیستم توصیه‌گر را بر روی این داده‌ها نیز آزمایش کنیم. این داده‌ها شامل تعاملات کاربران با صفحات مختلف اینستاگرام می‌شود که می‌تواند اطلاعات ارزشمندی برای تحلیل رفتار کاربران و ارائه پیشنهادهای شخصی‌سازی شده فراهم کند.

به طور کلی، این پروژه به دو بخش اصلی تقسیم می‌شود:

۱. **پیاده‌سازی سیستم توصیه‌گر مبتنی بر LSH**: در این بخش، ابتدا الگوریتم LSH را پیاده‌سازی کرده و سپس از آن برای ایجاد سیستم توصیه‌گر استفاده می‌کنیم.

۲. **استفاده از سیستم توصیه‌گر بر روی داده‌های اینستاگرام**: در این بخش، داده‌های جمع‌آوری شده از اینستاگرام را تحلیل کرده و سیستم توصیه‌گر را بر روی این داده‌ها آزمایش می‌کنیم تا کارایی و دقت آن را ارزیابی کنیم.

۱ بررسی و ساختار داده‌ها

۱-۱ توضیحات مجموعه داده MovieLens 20M

مجموعه داده MovieLens 20M شامل ۲۰ میلیون رتبه‌بندی و یک میلیون نقد از ۱۳۸،۰۰۰ کاربر در مورد ۲۷،۰۰۰ فیلم است. این مجموعه داده توسط GroupLens Research در دانشگاه مینه‌سوتا ایجاد شده است و یکی از بزرگترین و پرکاربردترین مجموعه داده‌ها در حوزه توصیه‌گرها می‌باشد. داده‌های MovieLens برای ارزیابی سیستم‌های توصیه‌گر استفاده می‌شوند و به دلیل حجم بزرگ و تنوع فیلم‌ها و کاربران، چالشی جدی برای مدل‌های توصیه‌گر محسوب می‌شوند.

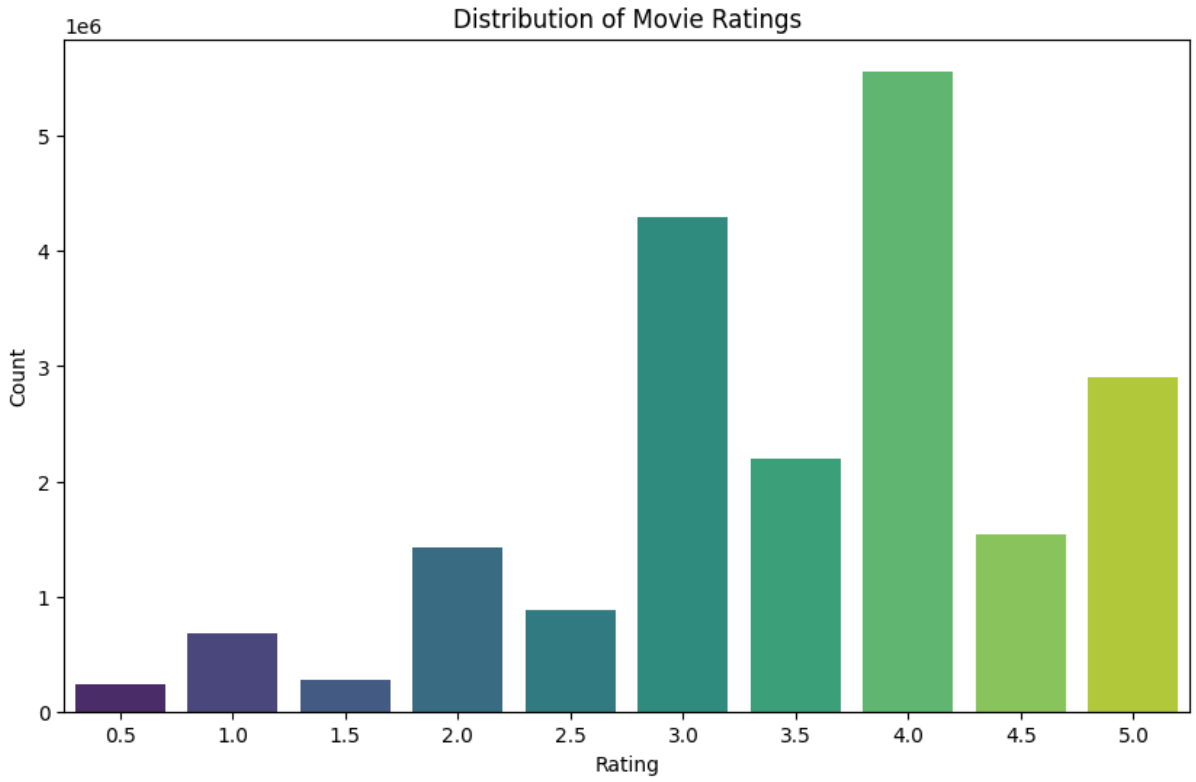
۱-۲ بارگذاری و بررسی اولیه داده‌ها

ابتدا، داده‌ها از فایل‌های CSV مربوطه بارگذاری شدند. این مجموعه داده شامل دو فایل اصلی ratings.csv و movies.csv است که اطلاعات مربوط به رتبه‌بندی‌ها و فیلم‌ها را به ترتیب در خود دارند. فایل ratings.csv شامل ستون‌هایی برای شناسه کاربر، شناسه فیلم، امتیاز و تاریخ ثبت امتیاز می‌باشد. فایل movies.csv شامل ستون‌هایی برای شناسه فیلم، عنوان فیلم و ژانرهای مربوط به فیلم است. برای مراحل visualization این دو مجموعه داده را باهم join می‌کنیم تا بتوانیم تحلیل‌های بصری و آماری انجام دهیم.

۱-۳ Visualization

بصری‌سازی داده‌ها به ما کمک می‌کند تا الگوها و روندهای موجود در داده‌ها را بهتر درک کنیم. در این بخش، چندین نمودار برای نمایش توزیع و ویژگی‌های مختلف مجموعه داده MovieLens 20M تهیه شده است.

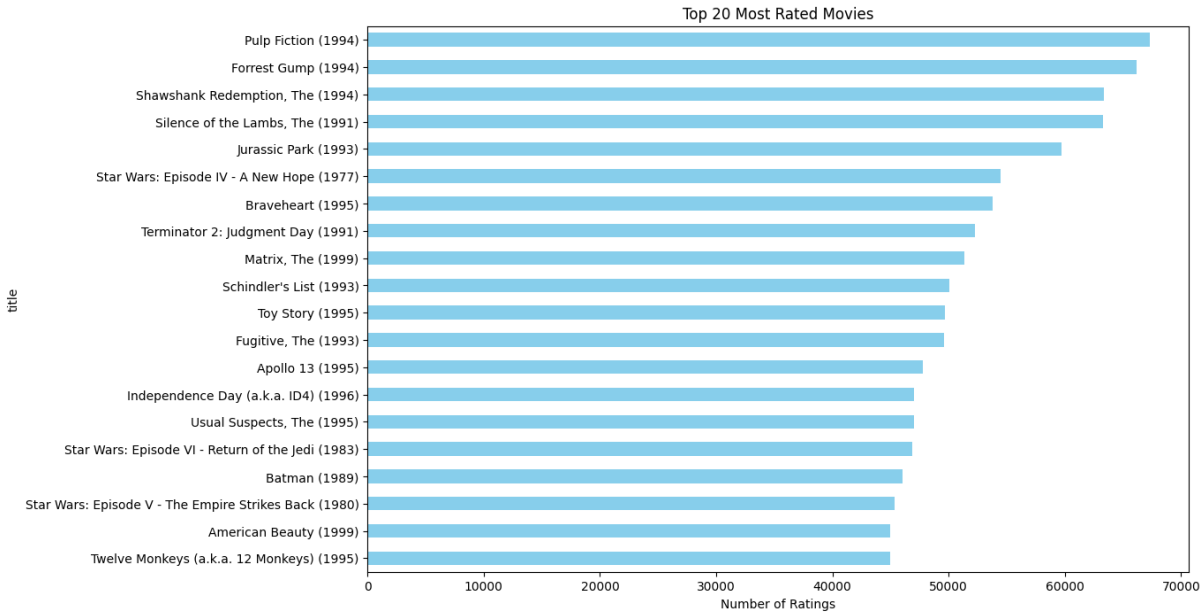
توزیع امتیازها



شکل ۱: توزیع رتبه‌بندی‌های داده شده به فیلم‌ها

این نمودار نشان می‌دهد که توزیع رتبه‌بندی‌های داده شده به فیلم‌ها چگونه است. همانطور که از نمودار مشخص است، اکثر رتبه‌بندی‌ها در مقیاس ۳ تا ۴ قرار دارند. این اطلاعات می‌تواند به شناسایی الگوهای رتبه‌بندی کاربران کمک کند و در بهبود مدل توصیه‌گر مؤثر باشد. رتبه‌بندی‌های بالا (۴ و ۵) نشان‌دهنده رضایت بالای کاربران از فیلم‌ها می‌باشد و رتبه‌بندی‌های پایین (۱ و ۲) نشان‌دهنده نارضایتی کاربران است.

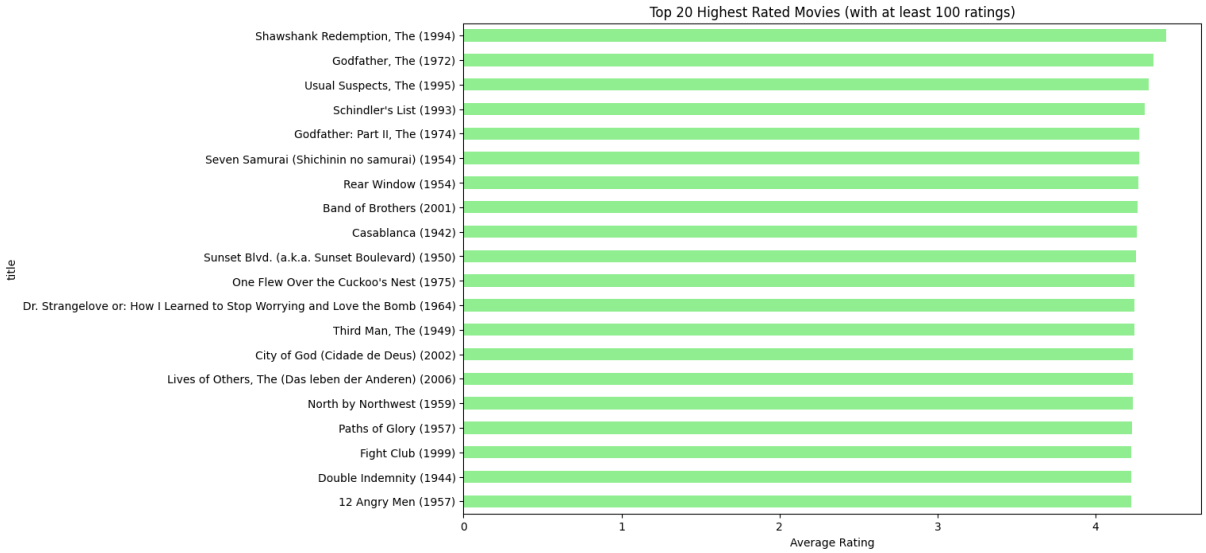
فیلم‌ها با بیشترین تعداد رای



شکل ۲: فیلم‌هایی با بیشترین تعداد رای

این نمودار ۲۰ فیلم را به ما نشان می‌دهد که بیشترین تعداد رای‌ها در بین تمام فیلم‌ها را دارند. نکته جالب حضور سه فیلم اول از سال ۱۹۹۴ است که نشان‌دهنده محبوبیت ماندگار این فیلم‌ها می‌باشد. فیلم‌هایی با تعداد رای بالا معمولاً فیلم‌هایی هستند که توجه بسیاری از کاربران را به خود جلب کرده‌اند و به همین دلیل می‌توانند برای تحلیل‌های بیشتر و بهبود مدل‌های توصیه‌گر مفید باشند.

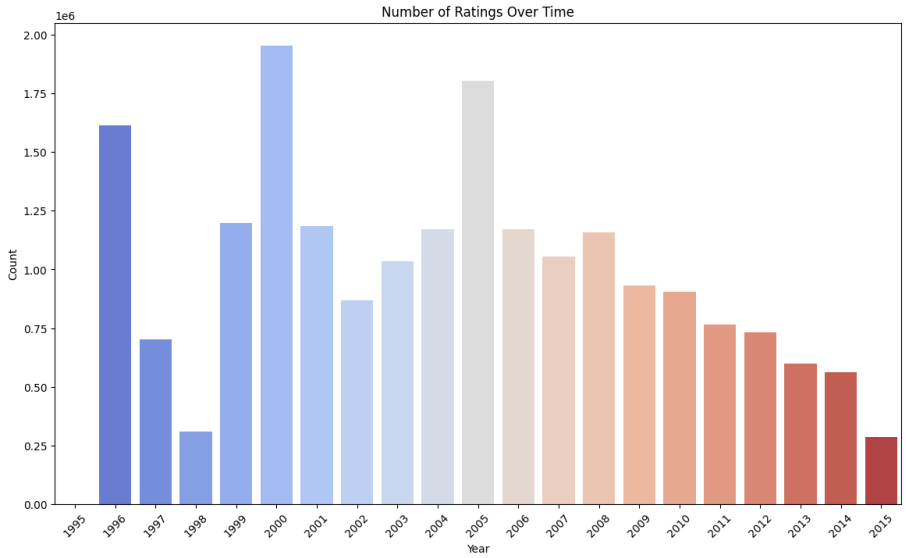
فیلم‌ها با بیشترین میانگین امتیاز



شکل ۳: فیلم‌هایی با بیشترین میانگین امتیاز

نمودار بالا از بین فیلم‌هایی که حداقل ۱۰۰ رای توسط کاربران به آن‌ها داده شده، ۲۰ فیلمی که بیشترین میانگین امتیاز را دارا هستند را نمایش می‌دهد. نکته جالب این نمودار حضور فیلم The Shawshank Redemption از نمودار قبلی به عنوان رتبه اول نمودار فعلی می‌باشد که نشان می‌دهد این فیلم نه تنها پربیننده بوده بلکه فیلم خوب و پرطرفداری نیز هست. فیلم‌هایی با میانگین امتیاز بالا معمولاً فیلم‌هایی هستند که کیفیت بالایی دارند و توانسته‌اند رضایت تعداد زیادی از کاربران را جلب کنند.

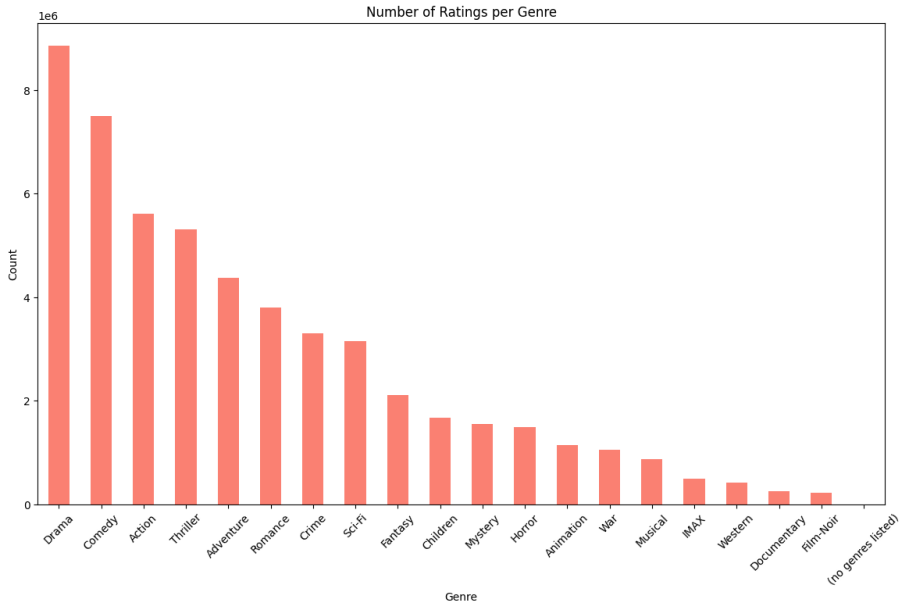
توزیع تعداد رای‌ها بر اساس زمان



شکل ۴: توزیع تعداد رای‌های کاربران بر حسب زمان

این نمودار با نمایش توزیع تعداد رای‌های کاربران برحسب زمان اخذ این آرا به ما نشان می‌دهد که در سال‌های ۲۰۰۰ و ۲۰۰۵ قبل‌تر در سال ۱۹۹۶ کاربران علاقه بیشتری به امتیاز دادن به این فیلم‌ها داشتند و از سال ۲۰۰۵ با افت شدید و نزولی شدن تعداد آرا همراه است. این اطلاعات می‌تواند نشان‌دهنده تغییرات در تمایلات کاربران و تاثیر تغییرات فناوری و رسانه‌ها بر رفتار کاربران باشد.

توزیع فیلم‌ها بر اساس ژانر



شکل ۵: توزیع فیلم‌ها بر اساس ژانر

این نمودار توزیع فیلم‌ها را براساس ژانر آن‌ها نمایش می‌دهد. ممکن است یک فیلم که در دو ژانر قرار می‌گیرد یک بار در مثلاً ژانر ترسناک و یک بار در ژانر کمدی شماره بشود. در کل ژانر درام با اختلاف پرطرفدارترین ژانر و پس از آن ژانر کمدی می‌باشد و ژانرهای بعدی خیلی اختلاف فاحشی ندارند. این اطلاعات می‌تواند به مدل توصیه‌گر کمک کند تا با در نظر گرفتن ژانرهای محبوب، پیشنهادات بهتری به کاربران ارائه دهد.

۴-۱ نتیجه‌گیری

با بررسی و تحلیل مجموعه داده MovieLens 20M، به بینش‌های مهمی در مورد الگوهای رفتاری کاربران و ویژگی‌های فیلم‌ها دست یافتیم. این اطلاعات می‌تواند به بهبود مدل‌های توصیه‌گر کمک کرده و تجربه کاربری بهتری را برای کاربران فراهم کند. در بخش‌های بعدی، به پیاده‌سازی و ارزیابی سیستم توصیه‌گر مبتنی بر LSH می‌پردازیم.

۲ پیش‌پردازش و نمایش sparse

۲-۱ تشکیل ماتریس اسپارس

یکی از مهم‌ترین مراحل در پیاده‌سازی سیستم‌های توصیه‌گر، پیش‌پردازش داده‌ها و تبدیل آن‌ها به فرمتی است که بتوان به راحتی با آن‌ها کار کرد. برای این منظور، ابتدا تابعی با نام $create\_sparse\_matrix$ نوشته شد که با استفاده از تابع $pivot$ کتابخانه pandas، داده را به یک ماتریس dense تبدیل کرده و سپس با استفاده از کتابخانه scipy این ماتریس را به یک ماتریس اسپارس تبدیل می‌کرد.

این روش به دلیل عدم تبدیل مستقیم داده به فرمت اسپارس، عملکرد بسیار کندی داشت و حافظه بسیار زیادی را در مرحله اولیه اشغال می‌کرد که حتی منجر به kernel crash نیز می‌شد. برای بهبود این مسئله، تابعی به نام df_to_sparse نوشته شد که به طور مستقیم یک ماتریس اسپارس با داده‌های داده شده می‌سازد. این تابع با استفاده از شناسه‌های کاربران و آیتم‌ها، ماتریس اسپارس را تشکیل می‌دهد که ابعاد آن مساوی با ابعاد کلی داده بوده و با توجه به split دیتا، خانه‌ها را پر می‌کند.

تشکیل ماتریس اسپارس به ما امکان می‌دهد تا با داده‌های حجیم به صورت کارآمدتر کار کنیم و از منابع سیستم بهینه‌تر استفاده کنیم. این مرحله، پایه‌ای برای مراحل بعدی پردازش و تحلیل داده‌ها است.

۲-۲ نرمال‌سازی داده

نرمال‌سازی داده‌ها مرحله‌ای مهم در پیش‌پردازش داده‌ها است که به بهبود دقت مدل‌های توصیه‌گر کمک می‌کند. در این بخش، برای هر کاربر میانگین امتیازاتش محاسبه می‌شود و این میانگین از همه امتیازات کاربر کسر می‌شود. به این ترتیب، داده‌ها در بازه -۴ تا ۴ قرار می‌گیرند. این نرمال‌سازی باعث می‌شود تا تأثیر اختلافات بین کاربران کاهش یابد و مدل بتواند الگوهای کلی را بهتر شناسایی کند.

در روش Collaborative Filtering، یکی از مراحل اصلی محاسبه پیش‌بینی‌ها، کسر میانگین امتیازات هر کاربر از امتیازات وی است. با نرمال‌سازی داده‌ها در این مرحله، این عملیات از همان ابتدا برای داده‌های آموزشی انجام می‌شود که به بهبود دقت و سرعت محاسبات کمک می‌کند.

۲-۳ مزایای استفاده از ماتریس اسپارس

استفاده از ماتریس اسپارس مزایای زیادی دارد که از جمله آن‌ها می‌توان به موارد زیر اشاره کرد:

- **صرفه‌جویی در حافظه**: ماتریس‌های اسپارس تنها عناصر غیرصفر را ذخیره می‌کنند که باعث کاهش چشمگیر استفاده از حافظه می‌شود.

- **افزایش سرعت محاسبات**: عملیات ریاضی بر روی ماتریس‌های اسپارس به دلیل کمبود عناصر غیرصفر سریع‌تر انجام می‌شود.
- **بهبود کارایی**: در کاربردهای بزرگ مقیاس مانند سیستم‌های توصیه‌گر، استفاده از ماتریس اسپارس کارایی کلی سیستم را بهبود می‌بخشد.

۴-۲ نتیجه‌گیری

پیش‌پردازش داده‌ها و تبدیل آن‌ها به فرمتی قابل استفاده برای مدل‌های توصیه‌گر، مرحله‌ای حیاتی در فرایند توسعه سیستم‌های توصیه‌گر است. با استفاده از روش‌های مناسب برای تشکیل ماتریس اسپارس و نرمال‌سازی داده‌ها، می‌توان به بهبود دقت و کارایی این مدل‌ها کمک کرد. در بخش‌های بعدی، به پیاده‌سازی و ارزیابی سیستم توصیه‌گر مبتنی بر LSH می‌پردازیم.

۳ تقسیم آموزش و تست

برای تقسیم مجموعه داده به آموزش و آزمایش راه‌های متفاوتی وجود دارد که ما دو روش از بین این روش‌ها را بررسی کردیم:

- **انتخاب تصادفی از بین رای‌ها**: در این روش، خود فایل rating.csv به دو قسمت آزمایش و آموزش با نسبت ۱ به ۹ تقسیم می‌شود. این بدان معناست که ۱۰ درصد از داده‌ها به عنوان مجموعه آزمایش و ۹۰ درصد به عنوان مجموعه آموزش در نظر گرفته می‌شود. این روش ممکن است باعث شود خانه‌های تصادفی در ماتریس آموزش صفر شود و به ماتریس آزمایش انتقال پیدا کند. مزیت این روش، سادگی و اجرای سریع آن است؛ اما ممکن است نتایج نهایی تحت تأثیر عدم یکنواختی در توزیع داده‌ها قرار گیرد.
- **انتخاب تصادفی از بین کاربران**: در این روش، پس از ساخت ماتریس که هر سطر آن یک کاربر و هر ستون آن یک فیلم است، از بین سطرها یا کاربران به طور تصادفی ۱۰ درصد آن‌ها انتخاب شده و تشکیل ماتریس آزمایش را می‌دهد. به این صورت که ۱۰ درصد از کاربران به عنوان داده آزمایش و ۹۰ درصد به عنوان داده آموزش در نظر گرفته می‌شوند. این روش به حفظ یکنواختی در توزیع داده‌ها کمک می‌کند و احتمالاً نتایج دقیق‌تری را به همراه خواهد داشت.

از بین روش‌های گفته شده، هر دو آزمایش شدند و روش دوم منجر به نتایج بهتری شد که این روش انتخاب شد. انتخاب تصادفی از بین کاربران باعث می‌شود تا تعادل بهتری در تقسیم داده‌ها ایجاد شود و نتایج بهتری حاصل شود.

۳-۱ Cross-Validation

برای ارزیابی بهتر و دقیق‌تر مدل، مطابق با کاری که در مقاله انجام شده است، از روش Cross-Validation استفاده شد. در این روش، هربار ۱۰ درصد از داده به عنوان داده آزمایش در نظر گرفته می‌شود و ۸۰ درصد به عنوان داده آموزش، سپس این کار را ۱۰ بار انجام می‌دهیم تا تمام داده‌ها یک بار در جایگاه آزمایش و یک بار در جایگاه آموزش قرار گیرند. این فرآیند به نام 10-fold Cross-Validation شناخته می‌شود.

مزیت استفاده از Cross-Validation این است که به ما اجازه می‌دهد تا مدل را بر روی بخش‌های مختلف داده‌ها آزمایش کنیم و نتایج به دست آمده را با هم ترکیب کنیم تا به یک نتیجه جامع و دقیق برسیم. این کار به کاهش واریانس و جلوگیری از بیش‌برازش (overfitting) کمک می‌کند.

روش ارزیابی

در هر بار اجرای Cross-Validation، مراحل زیر انجام می‌شود:

۱. **تقسیم داده‌ها**: ۱۰ درصد از داده‌ها به عنوان داده آزمایش و ۹۰ درصد به عنوان داده آموزش انتخاب می‌شود.

۲. **آموزش مدل**: مدل توصیه‌گر بر روی داده‌های آموزش آموزش داده می‌شود.

۳. **ارزیابی مدل**: مدل آموزش‌دیده بر روی داده‌های آزمایش ارزیابی می‌شود و معیارهای عملکرد مانند دقت (precision) و یادآوری (recall) محاسبه می‌شوند.

۴. **ذخیره نتایج**: نتایج هر بار اجرای Cross-Validation ذخیره و در نهایت میانگین گرفته می‌شود.

با میانگین گرفتن از نتایج به دست آمده از تمام مراحل Cross-Validation، به نتیجه جامع و دقیقی در مورد عملکرد مدل دست می‌یابیم. این روش ارزیابی به ما اطمینان می‌دهد که مدل توصیه‌گر به خوبی تعمیم‌پذیر است و می‌تواند بر روی داده‌های جدید نیز عملکرد مناسبی داشته باشد.

نتایج ارزیابی و دقت مدل در بخش‌های بعدی به تفصیل مورد بحث قرار خواهند گرفت.

۴ درک حفظ حریم خصوصی

حفظ حریم خصوصی در سیستم‌های توصیه‌گر از اهمیت ویژه‌ای برخوردار است. با گسترش استفاده از سیستم‌های توصیه‌گر در برنامه‌های مختلف، نگرانی‌های مرتبط با حفظ حریم خصوصی کاربران نیز افزایش یافته است. در این پروژه، تکنیک‌های حفظ حریم خصوصی در قالب استفاده از Locality-Sensitive Hash-ing (LSH) برای پیشنهاد دادن مورد بررسی قرار گرفته‌اند. این روش‌ها به ما امکان می‌دهند تا بدون نیاز به دسترسی مستقیم به داده‌های حساس کاربران، توصیه‌های مؤثری ارائه دهیم و امنیت و حریم خصوصی آن‌ها را حفظ کنیم.

۴-۱ تئوری پشت حفظ حریم خصوصی

تکنیک LSH از خواصی برخوردار است که می‌تواند ضمن حفظ تشابه‌های ساختاری یا به اصطلاح دقیق‌تر با حفظ فاصله‌های قبلی با یک معیار فاصله خاص و مشخص، امکان پنهان‌سازی داده‌های واقعی کاربران را فراهم آورد. این ویژگی‌ها از طریق تبدیل داده‌ها به بردارهای هش که تنها براساس تشابه و نه بر اساس مقادیر دقیق داده‌ها تولید می‌شوند، به دست می‌آیند. به این ترتیب، حتی در صورت دسترسی غیرمجاز به این نمایه‌ها، اطلاعات شخصی کاربران محفوظ می‌ماند.

LSH به این صورت عمل می‌کند که داده‌های ورودی را به چندین فضای هش با استفاده از توابع هش مختلف نگاشت می‌کند. این توابع هش طوری طراحی شده‌اند که اشیای مشابه با احتمال بیشتری در یک باکت هش قرار می‌گیرند. با استفاده از LSH، می‌توان به سرعت همسایگان نزدیک یک داده را پیدا کرد، بدون اینکه نیاز باشد به داده‌های اصلی دسترسی داشت. این روش به حفظ حریم خصوصی کاربران کمک می‌کند و در عین حال کارایی سیستم توصیه‌گر را افزایش می‌دهد.

۴-۲ کاربرد در پروژه

در پیاده‌سازی ما و طبق مقاله، LSH برای ایجاد یک ماتریس همسایگی از کاربران استفاده شده است. این ماتریس بر اساس تشابه میان رتبه‌بندی‌های کاربران بدون نیاز به افشای اطلاعات شخصی تک تک کاربران ساخته شده است. علاوه بر این، در تمام مراحل تجزیه و تحلیل داده‌ها، اطلاعات کاربران به گونه‌ای پردازش شده است که امکان بازیابی اطلاعات شخصی وجود ندارد.

LSH به ما این امکان را می‌دهد که بدون دسترسی مستقیم به داده‌های خام و حساس کاربران، بتوانیم الگوهای تشابه میان کاربران را شناسایی کرده و پیشنهادهای شخصی‌سازی شده ارائه دهیم. این امر به ویژه در محیط‌هایی که حفظ حریم خصوصی کاربران از اهمیت بالایی برخوردار است، بسیار حائز اهمیت است.

۳-۴ چالش‌ها و راهکارها

یکی از چالش‌های عمده در پیاده‌سازی تکنیک‌های حفظ حریم خصوصی، تعادل بین دقت توصیه‌ها و سطح حریم خصوصی است. حفظ حریم خصوصی ممکن است به کاهش دقت مدل‌های توصیه‌گر منجر شود، زیرا اطلاعات کمتری برای آموزش مدل در دسترس خواهد بود. در این پروژه، با استفاده از تنظیمات پارامتریک مناسب و ارزیابی دقیق، تلاش شده است تا این تعادل به نحو احسن انجام شود.

- **تنظیمات پارامتریک**: پارامترهای مختلف LSH مانند تعداد توابع هش و تعداد باکت‌ها به دقت تنظیم شده‌اند تا به تعادل مطلوب بین دقت و حفظ حریم خصوصی برسیم.
- **ارزیابی دقیق**: ارزیابی‌های متعدد با استفاده از Cross-Validation انجام شده است تا از دقت و کارایی مدل اطمینان حاصل شود. این ارزیابی‌ها شامل بررسی دقت توصیه‌ها و میزان حفظ حریم خصوصی کاربران است.
- **تجزیه و تحلیل نتایج**: نتایج به دقت مورد بررسی قرار گرفته‌اند تا اطمینان حاصل شود که هم حریم خصوصی حفظ شده و هم کیفیت توصیه‌ها در سطح قابل قبولی است. این تجزیه و تحلیل شامل بررسی مواردی مانند precision، recall و MSE است.

نتایج نشان داد که استفاده از LSH می‌تواند به طور موثری به حفظ حریم خصوصی کاربران کمک کند، بدون اینکه به طور قابل توجهی از دقت و کارایی سیستم توصیه‌گر کاسته شود. این روش می‌تواند به عنوان یک راهکار موثر برای سیستم‌های توصیه‌گری که نیاز به حفظ حریم خصوصی کاربران دارند، مورد استفاده قرار گیرد.

۵ تحلیل پیچیدگی زمانی

در این بخش به بررسی و تحلیل پیچیدگی زمانی الگوریتم توصیه‌گر مبتنی بر LSH می‌پردازیم. این تحلیل شامل مقایسه الگوریتم استفاده شده در نوت‌بوک با الگوریتم ارائه شده در مقاله است.

۵-۱ الگوریتم در مقاله

مقاله یک الگوریتم توصیه‌گر مبتنی بر LSH را معرفی می‌کند که شامل مراحل زیر است:

- **پیش‌پردازش داده‌ها**: تبدیل داده‌های خام تعامل کاربر-آیتم به فرمت مناسب.
- **ساخت جداول LSH**: هش کردن نقاط داده به داخل باکت‌ها با استفاده از LSH.
- **تولید کاندیداها**: یافتن همسایگان کاندیدا از باکت‌های LSH.
- **محاسبه تشابه**: محاسبه تشابه بین آیتم‌ها یا کاربران.
- **تولید توصیه‌ها**: تولید توصیه‌ها بر اساس امتیازات تشابه.

۵-۲ پیاده‌سازی در نوت‌بوک

نوت‌بوک شامل مراحل زیر است:

- **بارگذاری و پیش‌پردازش داده‌ها**: بارگذاری مجموعه داده MovieLens و پیش‌پردازش آن.
- **ساخت جداول LSH**: پیاده‌سازی LSH برای هش کردن تعاملات کاربر-آیتم.
- **تولید همسایگان کاندیدا**: استفاده از LSH برای تولید همسایگان کاندیدا.
- **محاسبه تشابه**: محاسبه امتیازات تشابه بین کاربران.
- **تولید توصیه‌ها**: ایجاد توصیه‌ها بر اساس امتیازات تشابه محاسبه شده.

۵-۳ تحلیل پیچیدگی زمانی

در اینجا به تحلیل پیچیدگی زمانی هر مرحله از الگوریتم می‌پردازیم:

- **پیش‌پردازش داده‌ها**:
  - **مقاله**: پیش‌پردازش شامل نرمال‌سازی و تبدیل به فرمت اسپارس.

- **نوت‌بوک**: بارگذاری داده‌ها و انجام پیش‌پردازش مشابه.
- **پیچیدگی زمانی**: $O(n \log n)$ که n تعداد تعاملات است.

- **ساخت جداول LSH**:
  - **مقاله**: هش کردن هر نقطه داده به چندین جدول LSH.
  - **نوت‌بوک**: ایجاد توابع هش و هش کردن زوج‌های کاربر-آیتم.
  - **پیچیدگی زمانی**: برای هر نقطه داده که به k جدول هش می‌شود: $O(k \cdot n)$.

- **تولید کاندیداها**:
  - **مقاله**: پرس‌وجو از جداول LSH برای یافتن همسایگان کاندیدا.
  - **نوت‌بوک**: بازیابی همسایگان کاندیدا از باکت‌های LSH.
  - **پیچیدگی زمانی**: برای هر پرس‌وجو با m جدول هش و L کاندیدا در هر جدول: $O(m \cdot L)$.

- **محاسبه تشابه**:
  - **مقاله**: محاسبه امتیازات تشابه بین آیتم‌ها/کاربران.
  - **نوت‌بوک**: محاسبه تشابه با استفاده از معیار انتخاب شده.
  - **پیچیدگی زمانی**: فرض بر اینکه d ابعاد باشد: $O(d \cdot n^2)$.

- **تولید توصیه‌ها**:
  - **مقاله**: تولید توصیه‌های برتر- N بر اساس امتیازات تشابه.
  - **نوت‌بوک**: مرتب‌سازی و انتخاب آیتم‌ها/کاربران برتر- N.
  - **پیچیدگی زمانی**: $O(n \log n)$ برای مرتب‌سازی.

۴-۵ پیچیدگی فرآیند دسته‌بندی و ساخت ماتریس همسایگی

فرآیند دسته‌بندی و ساخت ماتریس همسایگی شامل مراحل زیر است:

- **دسته‌بندی**: هر کاربر به چندین باکت LSH هش می‌شود تا همسایگان بالقوه پیدا شوند.
- **ساخت ماتریس همسایگی**: بر اساس همسایگان بالقوه، ماتریس همسایگی ساخته می‌شود که تشابه میان کاربران را نشان می‌دهد.
- **پیچیدگی زمانی**: $O(k \cdot n)$ برای دسته‌بندی و $O(m \cdot L)$ برای ساخت ماتریس همسایگی.

۵-۵ تجزیه ماتریس

در پروژه، ابتدا تجزیه ماتریس قابل یادگیری SVD مطابق مقاله پیاده‌سازی شد. اما برای تسریع فرآیند Cross-Validation، از Randomized SVD استفاده شد.

- **تجزیه ماتریس قابل یادگیری (Learnable SVD)**: این روش دقیق است ولی زمان زیادی برای آموزش نیاز دارد.
- **تجزیه ماتریس تصادفی (Randomized SVD)**: این روش سریع‌تر است و برای Cross-Validation استفاده شد.
- **پیچیدگی زمانی**:
  - Learnable SVD: پیچیدگی زمانی $O(k \cdot n^3)$ که k تعداد دفعات اجرای الگوریتم گرادیان کاهشی است.
  - Randomized SVD: پیچیدگی زمانی $O(n^2 \log n)$

۵-۶ مقایسه

ساختار کلی و پیاده‌سازی در هر دو مقاله و نوت‌بوک مشابه است، با تفاوت‌های اصلی در جزئیات پیاده‌سازی و بهینه‌سازی. پیچیدگی‌های زمانی هم‌راستا هستند، به طوری که مقاله مبنای نظری را فراهم می‌کند و نوت‌بوک کاربرد عملی را ارائه می‌دهد.

در مجموع، تجزیه و تحلیل پیچیدگی زمانی به ما این امکان را می‌دهد که کارایی الگوریتم را بهتر درک کنیم و بهینه‌سازی‌های لازم را برای بهبود عملکرد سیستم توصیه‌گر اعمال کنیم. این تحلیل‌ها نشان می‌دهند که استفاده از LSH می‌تواند به طور مؤثری زمان محاسبات را کاهش داده و کارایی سیستم را افزایش دهد.

۶ پیاده‌سازی در معماری تابعی

در این بخش به پیاده‌سازی روش پیشنهادی در معماری تابعی پرداخته شده است. هدف از این معماری، اطمینان از قابلیت استفاده مجدد و نگهداشت آسان کد است.

۶-۱ پیاده‌سازی تابعی در نوت‌بوک

در نوت‌بوک، پیاده‌سازی به گونه‌ای انجام شده که از اصول معماری تابعی بهره‌مند باشد و هر تکه کدی که به صورت پرتکرار در پروژه نیاز بوده تبدیل به تابع شده تا کدهای تکراری نوشته نشوند. تمامی توابع به صورت جداگانه تعریف شده‌اند تا سازماندهی و خوانایی کد بهبود یابد. این جداسازی به افزایش قابلیت استفاده مجدد و نگهداشت کد کمک می‌کند.

۶-۲ توضیحات توابع استفاده شده

- **تابع normalize_data**:
  - این تابع داده‌های ورودی را نرمال‌سازی می‌کند.
  - ابتدا میانگین امتیازات هر کاربر را محاسبه کرده و یک جدول جدید با نام mean_rating ایجاد می‌کند.
  - سپس داده‌ها را با استفاده از mean_rating ادغام می‌کند.
  - ستون normalized_rating را با کسر mean_rating از rating ایجاد می‌کند.
  - ستون‌های rating و mean_rating را حذف می‌کند تا داده‌های نهایی فقط شامل normalized_rating باشند.
  - در نهایت، داده‌های نرمال‌شده و mean_ratings را برمی‌گرداند.
- **تابع create_sparse_matrix**:
  - این تابع داده‌های ورودی را به یک ماتریس اسپارس تبدیل می‌کند.
  - ابتدا یک جدول محوری (pivot table) از داده‌ها با شاخص userId و ستون movieId ایجاد می‌کند که مقادیر آن normalized_rating هستند.
  - مقادیر خالی (NaN) را با صفر جایگزین می‌کند.
  - سپس، ماتریس حاصل را به یک ماتریس اسپارس (csr_matrix) تبدیل می‌کند و برمی‌گرداند.

- این متد به علت کندی با متد دیگری عوض شد.

- **تابع create_hash_vectors:**
  - این تابع بردارهای هش را ایجاد می‌کند.
  - ورودی‌ها شامل طول بردار (length) و تعداد بردارهای هش (count) می‌باشند.
  - برای هر بردار هش، یک بردار تصادفی با طول مشخص از توزیع یکنواخت در بازه 1- تا 1 ایجاد می‌کند.
  - در نهایت، لیستی از بردارهای هش ایجاد شده را برمی‌گرداند.

- **تابع create_hash:**
  - این تابع یک بردار را هش می‌کند.
  - ورودی شامل یک بردار و لیستی از بردارهای مین‌هش (min_hash) می‌باشد.
  - برای هر بردار مین‌هش، حاصل ضرب نقطه‌ای (dot product) بردار ورودی و بردار مین‌هش را محاسبه می‌کند.
  - اگر حاصل بیشتر از صفر باشد، مقدار 1 و در غیر این صورت مقدار 0 به لیست نتیجه افزوده می‌شود که دقیقا مشابه مقاله است.
  - در نهایت، لیست مقادیر هش شده را برمی‌گرداند.

- **تابع calculate_bucket_number:**
  - این تابع شماره باکت را برای یک بردار هش محاسبه می‌کند.
  - ورودی شامل یک بردار هش (hash_vector) می‌باشد.
  - بردار هش را به یک رشته باینری تبدیل می‌کند.
  - رشته باینری را به یک عدد صحیح در مبنای ۲ تبدیل کرده و آن را به عنوان شماره باکت برمی‌گرداند.

- **تابع create_neighborhood_matrix:**
  - این تابع ماتریس همسایگی را برای یک بردار کاربر جدید ایجاد می‌کند.
  - ورودی‌ها شامل بردار کاربر جدید (new_user_vector)، باکت‌ها (buckets)، بردارهای مین‌هش (min_hash)، و ماتریس اسپارس (sparse_matrix) می‌باشند.

- ابتدا بردار هش برای بردار کاربر جدید ایجاد می‌شود.
- شماره باکت بر اساس بردار هش محاسبه می‌شود.
- اگر شماره باکت در بین باکت‌ها موجود باشد، کاربران همسایه از آن باکت استخراج می‌شوند؛ در غیر این صورت، یک ماتریس اسپارس خالی برگردانده می‌شود.
- سپس، ردیف‌های ماتریس اسپارس مرتبط با کاربران همسایه به صورت یک ماتریس جدید ایجاد می‌شوند.
- در نهایت، ماتریس اسپارس همسایگی برگردانده می‌شود.

- **تابع matrix_factorization**:

- این تابع تجزیه ماتریس را برای ماتریس اسپارس انجام می‌دهد.
- ورودی شامل ماتریس اسپارس (sparse_ratings) و تعداد مؤلفه‌ها (k) می‌باشد.
- ابتدا تجزیه مقدارهای منفرد تصادفی (randomized SVD) بر روی ماتریس اسپارس انجام می‌شود.
- ماتریس‌های U، S و V از تجزیه مقدارهای منفرد به دست می‌آیند.
- عوامل آیتم (item_factors) با ضرب ماتریس U در قطر ماتریس S ایجاد می‌شوند.
- عوامل کاربر (user_factors) همان ماتریس V هستند.
- تجزیه ماتریس نهایی با ضرب عوامل آیتم در عوامل کاربر به دست می‌آید.
- در نهایت، ماتریس تجزیه شده برگردانده می‌شود.

- **تابع pearson_correlation_coefficient**:

- این تابع ضریب همبستگی پیرسون را بین دو آرایه محاسبه می‌کند.
- ورودی‌ها شامل دو آرایه (array1 و array2) می‌باشند.
- ابتدا بررسی می‌شود که طول هر دو آرایه یکسان باشد؛ در غیر این صورت، یک خطا ایجاد می‌شود.
- میانگین هر دو آرایه محاسبه می‌شود.
- آرایه‌های ورودی حول میانگین خود مرکزدهی می‌شوند.
- کوواریانس بین آرایه‌های مرکزدهی شده محاسبه می‌شود.

- انحراف معیار هر دو آرایه محاسبه می‌شود.
- اگر حاصل ضرب انحراف معیارها برابر با صفر باشد، ضریب همبستگی پیرسون برابر با صفر بازگردانده می‌شود.
- در غیر این صورت، ضریب همبستگی پیرسون با تقسیم کوواریانس بر حاصل ضرب انحراف معیارها محاسبه می‌شود.

- **تابع df_to_sparse:**
  - این تابع یک دیتافریم را به ماتریس اسپارس تبدیل می‌کند.
  - ورودی‌ها شامل دیتافریم رتبه‌بندی‌ها (rating_df)، دیتافریم (df) و کلید (key) برای مقادیر می‌باشند که پیش‌فرض آن normalized_rating است.
  - شناسه‌های منحصر به فرد کاربران از دیتافریم رتبه‌بندی‌ها استخراج شده و به ترتیب مرتب می‌شوند.
  - یک نگاشت (mapping) از شناسه‌های کاربران به شاخص‌های منحصر به فرد ایجاد می‌شود.
  - شناسه‌های منحصر به فرد فیلم‌ها از دیتافریم رتبه‌بندی‌ها استخراج شده و به ترتیب مرتب می‌شوند.
  - یک نگاشت (mapping) از شناسه‌های فیلم‌ها به شاخص‌های منحصر به فرد ایجاد می‌شود.
  - شاخص‌های کاربران و فیلم‌ها در دیتافریم df با استفاده از نگاشت‌های ایجاد شده، استخراج می‌شوند.
  - مقادیر مربوط به کلید (key) از دیتافریم df استخراج می‌شوند.
  - ماتریس coo_matrix با استفاده از شاخص‌های کاربران، شاخص‌های فیلم‌ها و مقادیر استخراج شده، ایجاد می‌شود.
  - ابعاد و تعداد عناصر غیرصفر ماتریس ایجاد شده چاپ می‌شوند.
  - در نهایت، ماتریس coo_matrix به ماتریس csr_matrix تبدیل شده و برگردانده می‌شود.

- **تابع diversify_recommendations_for_user:**
  - این تابع توصیه‌های متنوعی برای یک کاربر ایجاد می‌کند.
  - ورودی‌ها شامل بردار کاربر (user_vector)، نگاشت فیلم‌ها (movie_mapping) و دیکشنری ژانرها (genres_dict) می‌باشد.

- لیست diversified_recs برای ذخیره‌سازی توصیه‌های متنوع و مجموعه seen_genres برای ذخیره‌سازی ژانرهای مشاهده شده تعریف می‌شود.
- شاخص‌های مرتب‌سازی شده بر اساس امتیازات توصیه‌ها به صورت نزولی در sorted_rec_indices ذخیره می‌شوند.
- برای هر شاخص فیلم در sorted_rec_indices:
  - اگر تعداد توصیه‌های متنوع برابر یا بیشتر از تعداد ژانرها باشد، حلقه متوقف می‌شود.
  - اگر شاخص فیلم بزرگتر از طول بردار کاربر باشد، ادامه داده می‌شود.
  - شناسه فیلم از نگاشت فیلم‌ها استخراج می‌شود.
  - اگر شناسه فیلم در دیکشنری ژانرها موجود باشد، ژانرهای مرتبط با فیلم استخراج می‌شود.
  - برای هر ژانر در ژانرهای فیلم:
    - اگر ژانر قبلاً مشاهده نشده باشد، فیلم و امتیاز آن به لیست توصیه‌های متنوع افزوده می‌شود و ژانر به مجموعه ژانرهای مشاهده شده افزوده می‌شود.
  - در نهایت، لیست توصیه‌های متنوع بازگردانده می‌شود.

- **تابع sort_recommendation**:
  - این تابع توصیه‌های یک کاربر را بر اساس امتیازها مرتب می‌کند.
  - ورودی‌ها شامل بردار کاربر (user_vector) و نگاشت فیلم‌ها (movie_mapping) می‌باشند.
  - شاخص‌های توصیه‌ها بر اساس امتیازات به صورت نزولی مرتب می‌شوند.
  - برای هر شاخص فیلم در sorted_rec_indices، شناسه فیلم و امتیاز مربوطه به لیست توصیه‌ها افزوده می‌شود.
  - در نهایت، لیست توصیه‌های مرتب شده بازگردانده می‌شود.

- **تابع get_top_n_recommendations**:
  - این تابع برترین n توصیه‌ها را برای هر کاربر برمی‌گرداند.
  - ورودی‌ها شامل دیکشنری توصیه‌ها (recommendations) و تعداد توصیه‌ها (n) می‌باشند.
  - شناسه‌های منحصر به فرد فیلم‌ها از rating_df استخراج شده و به ترتیب مرتب می‌شوند.
  - برای هر کاربر در دیکشنری توصیه‌ها:

- توصیه‌ها بر اساس امتیازات مرتب می‌شوند.
- برترین n توصیه برای کاربر ذخیره می‌شود.

- در نهایت، دیکشنری شامل برترین توصیه‌ها برای هر کاربر بازگردانده می‌شود.

- **تابع precision_at_k**:

- این تابع دقت (precision) را در k توصیه برتر محاسبه می‌کند.
- ورودی‌ها شامل آیتم‌های توصیه شده (recommended_items)، آیتم‌های واقعی (actual_items) و مقدار k می‌باشند.
- مجموعه‌ای از آیتم‌های توصیه شده در k توصیه برتر و آیتم‌های واقعی ایجاد می‌شود.
- آیتم‌های مرتبط به عنوان اشتراک این دو مجموعه محاسبه می‌شوند.
- دقت به صورت نسبت آیتم‌های مرتبط به k محاسبه می‌شود.
- در نهایت، مقدار دقت بازگردانده می‌شود.

- **تابع recall_at_k**:

- این تابع بازخوانی (recall) را در k توصیه برتر محاسبه می‌کند.
- ورودی‌ها شامل آیتم‌های توصیه شده (recommended_items)، آیتم‌های واقعی (actual_items) و مقدار k می‌باشند.
- مجموعه‌ای از آیتم‌های توصیه شده در k توصیه برتر و آیتم‌های واقعی ایجاد می‌شود.
- آیتم‌های مرتبط به عنوان اشتراک این دو مجموعه محاسبه می‌شوند.
- بازخوانی به صورت نسبت آیتم‌های مرتبط به تعداد آیتم‌های واقعی محاسبه می‌شود.
- در نهایت، مقدار بازخوانی بازگردانده می‌شود.

۳-۶ پیاده‌سازی کراس‌ولیدیشن

در این بخش، پیاده‌سازی فرآیند کراس‌ولیدیشن برای ارزیابی مدل توصیه‌گر توضیح داده شده است. مراحل این فرآیند به صورت جداگانه و دقیق تشریح شده‌اند:

- **اجرای حلقه کراس‌ولیدیشن**:
  - متغیر fold برای شمارش تعداد حلقه‌ها و total_loss برای جمع‌آوری خطاها تنظیم می‌شود.
  - برای هر حلقه از KFold:
    - توقف زمانی ۱۰ ثانیه برای جلوگیری از استفاده بیش از حد منابع.
    - تقسیم کاربران به داده‌های آموزش (train_users) و اعتبارسنجی (val_users) بر اساس شاخص‌های تولید شده.
    - استخراج داده‌های مربوط به کاربران آموزش و اعتبارسنجی از rating_df.
    - نرمال‌سازی داده‌های آموزش با استفاده از تابع normalize_data.
    - تبدیل داده‌های آموزش به ماتریس اسپارس با استفاده از تابع df_to_sparse.
    - تبدیل داده‌های اعتبارسنجی به ماتریس اسپارس با استفاده از تابع df_to_sparse با کلید rating.
- **ایجاد باکت‌ها**:
  - ایجاد بردارهای مین‌هش با استفاده از تابع create_hash_vectors.
  - ایجاد باکت‌ها و افزودن کاربران به باکت‌های مناسب با استفاده از توابع create_hash و calculate_bucket_number.
  - پردازش ماتریس اسپارس اعتبارسنجی برای استخراج شاخص‌های کاربران مورد نظر.
- **محاسبه خطا**:
  - توقف زمانی ۱۰ ثانیه برای جلوگیری از استفاده بیش از حد منابع.
  - متغیرهای loss و processed_count برای محاسبه و شمارش خطاها تنظیم می‌شوند.
  - برای هر کاربر در داده‌های اعتبارسنجی:
    - استخراج بردار کاربر از ماتریس اسپارس.
    - پیش‌بینی رتبه‌بندی‌ها با استفاده از تابع predict_ratings.
    - محاسبه خطای میانگین مربعات بین رتبه‌بندی‌های واقعی و پیش‌بینی شده.

- افزودن خطا به loss و افزایش شمارش processed_count.
- چاپ خطای حلقه و افزودن آن به total_loss.
- محاسبه و چاپ خطای اعتبارسنجی برای هر حلقه.

- **محاسبه خطای میانگین**:
  - محاسبه و چاپ خطای میانگین کراس‌ولیدیشن.

۷ بهینه‌سازی هایپرپارامترها و تنظیمات متداول LSH

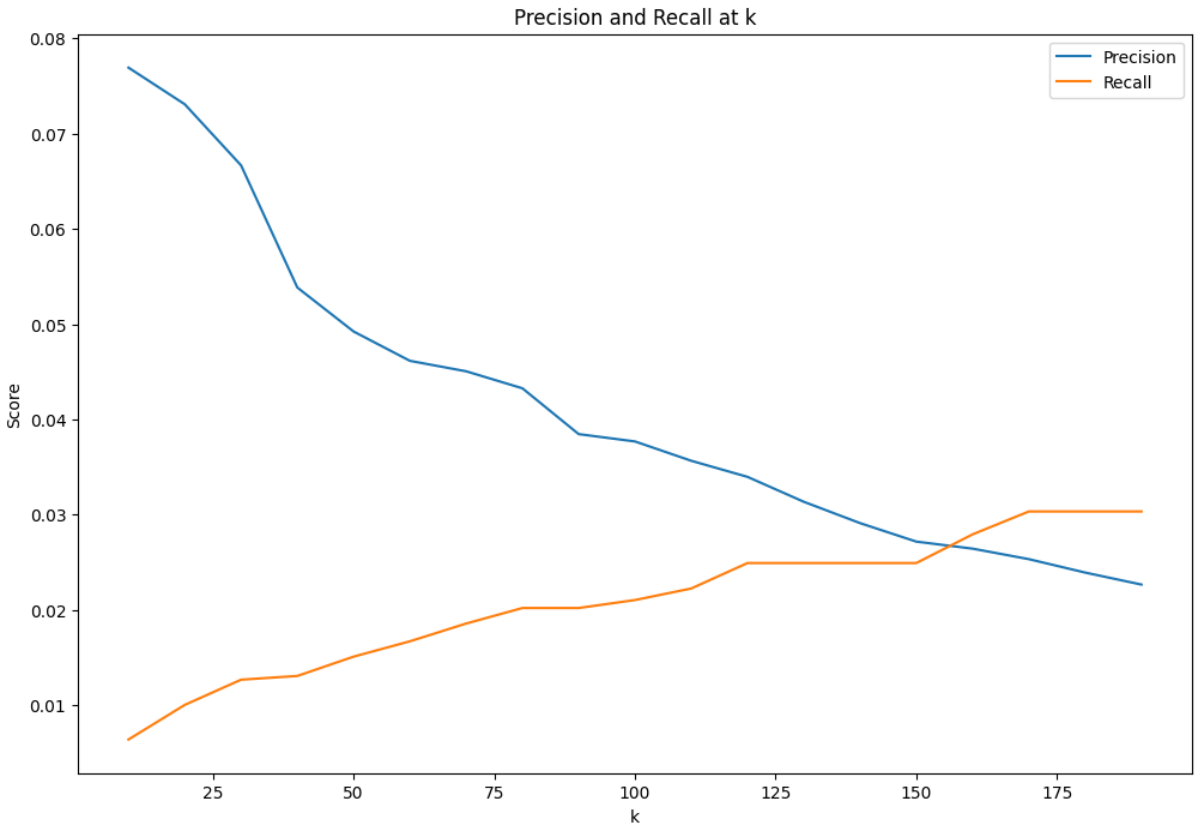
هایپرپارامترها نقش مهمی در بهینه‌سازی عملکرد الگوریتم‌های توصیه‌گر دارند. در این بخش، به چالش‌های تنظیم هایپرپارامترها در طول پیاده‌سازی روش پرداخته می‌شود. همچنین، تنظیمات متداول LSH و تأثیر آنها بر عملکرد توصیه‌گر مورد بررسی قرار می‌گیرد. با انجام آزمایشات، تولید نمودارها و مقایسه نتایج با آنچه در مقاله ارائه شده است، درک جامعی از کارایی روش و پتانسیل بهبود آن به دست خواهد آمد.

۷-۱ آزمایشات و نتایج

برای ارزیابی تأثیر هایپرپارامترها و تنظیمات LSH، آزمایشات مختلفی انجام شد:

- **تأثیر تعداد بردارهای هش بر عملکرد LSH**:
  - آزمایش با تعداد مختلف بردارهای هش (۲۰۰، ۱۰۰، ۲۰، ۱۶، ۱۴، ۱۲) انجام شد.
  - نتایج نشان داد که با افزایش تعداد بردارهای هش، ممکن است تعداد اجزای هر باکت آنقدر کم شود که برای بردار کاربرهای جدید همسایه‌ای پیدا نشود. از سوی دیگر، کاهش تعداد بردارهای هش می‌تواند منجر به افزایش تعداد همسایگان کاندیدا و در نتیجه کاهش دقت توصیه‌ها شود.
- **تأثیر تعداد مؤلفه‌های تجزیه ماتریس**:
  - نتایج نشان داد که با افزایش تعداد مؤلفه‌ها، دقت بهبود می‌یابد اما ممکن است بیش‌برازش رخ دهد. همچنین، خانه‌های صفر خیلی تغییری نکند و با داشتن عدد ثابت مانند ۲ یا ۳ هر ماتریسی به صورت دلخواه انجام نشود. بنابراین، نصف مینیمم ابعاد ماتریس به عنوان تعداد فاکتورها در نظر گرفته شد تا تعادلی بین دقت و کارایی حاصل شود.

۲-۷ صحت و بازیابی (Precision and Recall)



نمودار بالا نشان می‌دهد که با افزایش مقدار k، صحت کاهش می‌یابد اما بازخوانی افزایش می‌یابد. این نتیجه مطابق با یافته‌های مقاله است که بیان می‌کند با افزایش تعداد توصیه‌ها، احتمال یافتن آیتم‌های مرتبط بیشتر می‌شود اما صحت کلی کاهش می‌یابد زیرا تعداد آیتم‌های غیرمرتبط نیز افزایش می‌یابد.

۳-۷ مقایسه با نتایج مقاله

نتایج به دست آمده با نتایج ارائه شده در مقاله مقایسه شدند:

- میزان خطای میانگین مربعات ما حدوداً ۰/۸۸ شد که مشابه نتایج مقاله بود.
- زمان اجرای الگوریتم با تنظیمات بهینه LSH بهبود یافت و نتایج به دست آمده نشان‌دهنده عملکرد بهتر بود.
- از نظر صحت و بازیابی، نتایج پیاده‌سازی شده با مقاله بسیار تفاوت داشت و کمتر بود که شاید به دلیل استفاده نکردن از فاکتوریزیشن با یادگیری باشد. این امر نشان می‌دهد که استفاده از روش‌های پیشرفته‌تر تجزیه ماتریس می‌تواند بهبودهای قابل توجهی در عملکرد مدل ایجاد کند.

۴-۷ تنوع‌بخشی به توصیه‌ها

این روش با توجه به ژانرهای مختلف فیلم‌ها، تنوع بیشتری به توصیه‌ها می‌بخشد. با این کار، کاربران تجربه‌ای متنوع‌تر و جذاب‌تر از سیستم توصیه‌گر خواهند داشت. نتایج نشان می‌دهد که استفاده از ژانرها برای تنوع‌بخشی به توصیه‌ها می‌تواند به بهبود تجربه کاربری کمک کند. به عنوان مثال، توصیه فیلم‌ها بر اساس ژانر می‌تواند کاربران را به کشف محتوای جدید و متفاوت ترغیب کند.

۵-۷ توصیه‌های نهایی پس از تنوع‌بخشی

در اینجا یک مثال را می‌بینیم که برای یک کاربر از هر ژانر یک فیلم توصیه شده:

- **King Kong (1933)** - Predicted Rating: 4.42
- **Steel Magnolias (1989)** - Predicted Rating: 4.34
- **Phantasm (1979)** - Predicted Rating: 4.18
- **Forces of Nature (1999)** - Predicted Rating: 4.15
- **Deep Impact (1998)** - Predicted Rating: 4.10
- **Naked Gun: From the Files of Police Squad!, The (1988)** - Predicted Rating: 4.07
- **Love Bug, The (1969)** - Predicted Rating: 4.06
- **Tears of the Sun (2003)** - Predicted Rating: 4.04
- **Friends & Lovers (1999)** - Predicted Rating: 4.01
- **Stomp the Yard (2007)** - Predicted Rating: 4.01
- **Wyatt Earp (1994)** - Predicted Rating: 3.99
- **The Odyssey (1997)** - Predicted Rating: 3.95
- **Hollywoodland (2006)** - Predicted Rating: 3.95
- **Young Frankenstein (1974)** - Predicted Rating: 3.93
- **4 Little Girls (1997)** - Predicted Rating: 3.87

- **South Pacific (1958)** - Predicted Rating: 3.86
- **Tangled (2010)** - Predicted Rating: 3.83
- **Postman Always Rings Twice, The (1946)** - Predicted Rating: 3.72
- **Megamind (2010)** - Predicted Rating: 3.67
- **Sputnik (2013)** - Predicted Rating: 0.0

این مثال نشان می‌دهد که چگونه سیستم توصیه‌گر می‌تواند تنوعی از ژانرهای مختلف را برای کاربران فراهم کند و تجربه کاربری را بهبود بخشد. با استفاده از این رویکرد، می‌توانیم توصیه‌های متنوع‌تری ارائه دهیم که به کاربران کمک می‌کند تا محتوای جدید و جذاب را کشف کنند و تجربه‌ای متنوع و فراگیر از سیستم توصیه‌گر داشته باشند.

مقدمه بخش اینستاگرام

در این بخش، ما به جمع‌آوری داده‌های مرتبط با کاربران و صفحات اینستاگرام پرداختیم تا بتوانیم سیستم توصیه‌گر خود را بر روی این داده‌ها اعمال کنیم. فرآیند جمع‌آوری داده‌ها به صورت جداگانه در یک نوتبوک با نام insta_crawling.ipynb انجام شد.

هدف از این بخش این است که با استفاده از داده‌های واقعی اینستاگرام، عملکرد سیستم توصیه‌گر مبتنی بر LSH را ارزیابی کنیم و نتایج را تحلیل نماییم. در این فرآیند، داده‌های جمع‌آوری شده شامل اطلاعات پروفایل‌های اینستاگرام، تعداد دنبال‌کنندگان، تعداد دنبال‌شوندگان، تعداد پست‌ها، میانگین لایک‌ها و نظرات در هر پست، و سایر ویژگی‌های مرتبط می‌باشد.

داده‌های جمع‌آوری شده به فرمت مناسبی تغییر نام داده شدند تا با ساختار داده‌های MovieLens سازگار باشند. سپس این داده‌ها به مجموعه‌های آموزش و اعتبارسنجی تقسیم شده و مورد نرمال‌سازی قرار گرفتند. پس از آن، مدل توصیه‌گر با استفاده از تکنیک LSH پیاده‌سازی شد و رتبه‌بندی‌ها برای مجموعه اعتبارسنجی پیش‌بینی شدند.

نهایتا، توصیه‌ها بر اساس تعاملات کاربران با صفحات مختلف اینستاگرام تولید شده و نتایج به صورت گرافیکی نمایش داده شدند. در بخش نهایی فایل اصلی main.ipynb، تحلیل‌های انجام شده و نتایج به دست آمده به تفصیل بررسی شدند تا میزان دقت و کارایی سیستم توصیه‌گر مشخص شود.

در ادامه به شرح جزئیات مراحل مختلف این فرآیند و نتایج به دست آمده خواهیم پرداخت.

۸ انتخاب صفحات اینستاگرام

موضوع انتخابی ما در این بخش، تکنولوژی و اخبار فناوری است که ۳۰ صفحه معروف تا نسبتا معروف را برای این بخش انتخاب کردیم که به شرح زیر است:

No.	Username	Full Name	Follower Count
1	techcrunch	TechCrunch	1566321
2	wired	WIRED	1831334
3	mashable	Mashable	993525
4	thenextweb	TNW	110557
5	gizmodo	GIZMODO	125442
6	engadget	Engadget	377847
7	digitaltrends	Digital Trends	1293207
8	cnet	CNET	1171076
9	theapplehub	Apple Hub	1001445
10	androidauthority	Android Authority	740808
11	gadgetflow	Tech, Gadgets and Crowdfunding	243517
12	unboxtherapy	Lewis Hilsenteger	2881415
13	linustech	Linus Tech Tips	1612756
14	androidcentral	Android Central	282266
15	appleinsider	AppleInsider	98236
16	thegadgetshow	The Gadget Show	16223

No.	Username	Full Name	Follower Count
17	techradar	techradar	145070
18	mkbhd	Marques Brownlee	4905227
19	alcontech	Technology/gadgets	59221
20	verge	The Verge	1577526
21	csharp_dotnet	C#.NET Programming (C# | .NET)	12212
22	madewithcode	Made with Code	75115
23	gitHub	GitHub	482312
24	insidertech	Insider Tech	1806842
25	microsoft	Microsoft	4401197
26	openai	OpenAI	1688992
27	python.hub	Python helper	1651203
28	linux.teach	Linux Helper	170689
29	javascript.js	JavaScript	750543
30	mobiscrub	Mobiscrub	402676

۹ جمع‌آوری داده و استخراج ویژگی‌ها

۹-۱ مقدمه

در این بخش، فرآیند جمع‌آوری داده‌های اینستاگرام و استخراج ویژگی‌های مختلف از صفحات انتخابی توضیح داده می‌شود. داده‌های جمع‌آوری شده شامل تعداد دنبال‌کننده‌ها، تعداد دنبال‌شوندگان، تعداد پست‌ها، میانگین لایک‌ها و نظرات در هر پست، میانگین تعداد اسلایدها در هر پست، میانگین طول کپشن‌ها و برچسب‌های متداول می‌باشد. این داده‌ها در قالب فایل JSON ذخیره شده‌اند. این فرآیند به منظور تحلیل دقیق تعاملات کاربران با محتوای صفحات مختلف اینستاگرام و ایجاد توصیه‌های شخصی‌سازی شده به کار گرفته شده است.

۹-۲ کتابخانه‌های مورد استفاده

برای جمع‌آوری داده‌ها از کتابخانه Instaloader استفاده شده است. این کتابخانه یک ابزار قدرتمند و متن‌باز برای استخراج داده‌های اینستاگرام است که به ما امکان می‌دهد به سادگی به پروفایل‌ها و پست‌ها دسترسی پیدا کنیم و اطلاعات لازم را استخراج کنیم.

۹-۳ فرآیند جمع‌آوری داده‌ها

برای جمع‌آوری داده‌های اینستاگرام، از کتابخانه‌های متن‌باز استفاده شده است. این فرآیند شامل مراحل زیر می‌باشد:

- **استخراج پروفایل‌ها**:
  - ابتدا پروفایل‌های اینستاگرام صفحات انتخابی بارگذاری شده و اطلاعات عمومی آن‌ها از جمله نام کامل، تعداد دنبال‌کننده‌ها، تعداد دنبال‌شوندگان و تعداد پست‌ها استخراج می‌شود.
- **استخراج پست‌ها**:
  - سپس پست‌های منتشر شده توسط هر پروفایل جمع‌آوری می‌شوند. برای هر پست، اطلاعاتی نظیر تعداد لایک‌ها، تعداد نظرات، تعداد اسلایدها و طول کپشن استخراج می‌شود.
- **استخراج هشتگ‌ها**:
  - از کپشن هر پست، هشتگ‌های استفاده شده استخراج و شمارش می‌شوند تا پرکاربردترین هشتگ‌ها شناسایی شوند.

- **محاسبات آماری**:

■ میانگین تعداد لایک‌ها، نظرات، اسلایدها و طول کپشن برای هر پروفایل محاسبه می‌شود. همچنین، ده هشتگ برتر و پرکاربرد برای هر پروفایل تعیین می‌گردد.

۹-۴ ذخیره‌سازی داده‌ها

داده‌های جمع‌آوری شده در قالب یک فایل CSV و یک فایل JSON مطابق با خواسته پروژه ذخیره می‌شوند. این فایل شامل ویژگی‌های زیر برای هر پروفایل است:

- **Username**: نام کاربری صفحه اینستاگرام.
- **Full Name**: نام کامل کاربر یا صفحه.
- **Follower Count**: تعداد دنبال‌کنندگان.
- **Following Count**: تعداد دنبال‌شوندگان.
- **Number of Posts**: تعداد پست‌ها.
- **Average Likes**: میانگین لایک‌ها در هر پست.
- **Average Comments**: میانگین نظرات در هر پست.
- **Average Slides**: میانگین تعداد اسلایدها در هر پست.
- **Average Caption Length**: میانگین طول کپشن‌ها.
- **Top 10 Tags**: ده هشتگ برتر و پرکاربرد.

۹-۵ نمونه خروجی JSON

در ادامه یک نمونه از خروجی داده‌های جمع‌آوری شده در قالب JSON ارائه شده است:


```
{
    "username": "techcrunch",
    "full_name": "TechCrunch",
    "follower_count": 1566321,
```

```
    "following_count": 103,
    "num_posts": 5246,
    "avg_likes": 1226.05,
    "avg_comments": 28.55,
    "avg_slides": 1.45,
    "avg_caption_length": 778.3,
    "top_10_tags": [
        "#TechCrunch",
        "#technews",
        "#artificialintelligence",
        "#GenAI",
        "#Meta",
        "#ElonMusk",
        "#TimCook",
        "#Apple",
        "#socialmedia",
        "#Zuck"
    ]
}
```


۹-۶ توضیحات اضافی

برای جلوگیری از استفاده بیش از حد منابع و مدیریت نرخ درخواست‌ها به سرور اینستاگرام، یک توقف زمانی کوتاه بین هر درخواست قرار داده شده است. این کار به منظور جلوگیری از بلوکه شدن توسط سرور و حفظ بهره‌وری انجام شده است. در صورتی که پروفایلی وجود نداشته باشد یا مشکلی در اتصال به سرور رخ دهد، این موارد ثبت و مدیریت می‌شوند.

داده‌های نهایی به صورت دوره‌ای ذخیره و به‌روز می‌شوند تا از دست رفتن داده‌ها جلوگیری شود و در نهایت فایل CSV شامل تمام اطلاعات مورد نیاز برای استفاده در سیستم توصیه‌گر می‌باشد.

۱۰ جمع‌آوری داده‌های کاربران و گردآوری

۱۰-۱ مقدمه

در این بخش، نحوه جمع‌آوری داده‌های کاربران اینستاگرام که با پست‌های صفحات انتخابی تعامل داشته‌اند (دنبال کردن یا کامنت) توضیح داده شده است. هدف از این کار، جمع‌آوری داده‌هایی نظیر تعداد دنبال‌کنندگان، تعداد دنبال‌شوندگان، تعداد پست‌ها و نظرات به ازای هر صفحه و معیارهای تعامل کلی کاربران است. این داده‌ها در قالب فایل JSON سازماندهی و ذخیره شده‌اند. این اطلاعات به ما امکان می‌دهد تا رفتارهای تعاملی کاربران را تحلیل کرده و توصیه‌های دقیق‌تری ارائه دهیم.

۱۰-۲ فرآیند جمع‌آوری داده‌ها

برای جمع‌آوری داده‌ها از کاربران، از مراحل زیر استفاده شد:

- **استخراج کاربران فعال**:
  - کاربران فعال که با پست‌های صفحات انتخابی تعامل داشته‌اند شناسایی شدند.
  - برای هر پست، لیست کاربران کامنت‌گذار استخراج شد.
- **جمع‌آوری اطلاعات پروفایل کاربران**:
  - برای هر کاربر شناسایی شده، اطلاعات عمومی پروفایل شامل تعداد دنبال‌کنندگان، تعداد دنبال‌شوندگان و تعداد پست‌ها استخراج شد.
- **محاسبات آماری**:
  - معیارهای تعامل کلی برای هر کاربر شامل میانگین تعداد اسلایدها در هر پست و میانگین طول کپشن‌ها محاسبه شدند. این اطلاعات به ما کمک می‌کند تا میزان تعامل و فعالیت کاربران را به‌طور دقیق‌تری بسنجیم و بتوانیم الگوهای رفتاری آن‌ها را شناسایی کنیم.

۱۰-۳ ساختار داده‌های جمع‌آوری شده

داده‌های جمع‌آوری شده در قالب یک فایل JSON ذخیره شده‌اند. ساختار داده‌ها به شرح زیر است:

- **user_id**: نام کاربری
- **is_private**: آیا پیج پرایوت است

- **followers**: تعداد دنبال‌کنندگان
- **following**: تعداد دنبال‌شوندگان
- **posts**: تعداد پست‌ها اگر پیج پرایوت نیست
- **scores_per_page**: امتیاز تعاملی کاربر با صفحات مختلف

۱۰-۴ نمونه خروجی JSON

در ادامه یک نمونه از خروجی داده‌های جمع‌آوری شده در قالب JSON ارائه شده است:


```
{
    "user_id":"@0_monika_yadav1",
    "is_private":false,
    "followers":6,
    "following":197,
    "posts":1,
    "scores_per_page":{
        "@wired":0.3,
        "@digitaltrends":0.3,
        "@gadgetflow":0.3
    }
}
```


۱۰-۵ توضیحات اضافی

در فرآیند جمع‌آوری داده‌ها، توجه ویژه‌ای به جلوگیری از استفاده بیش از حد منابع و مدیریت نرخ درخواست‌ها به سرور اینستاگرام شده است. به منظور جلوگیری از بلوکه شدن توسط سرور و حفظ بهره‌وری، یک توقف زمانی کوتاه بین هر درخواست قرار داده شده است. این اقدامات کمک می‌کند تا داده‌ها به صورت پایدار و بدون وقفه جمع‌آوری شوند.

در صورتی که پروفایلی وجود نداشته باشد یا مشکلی در اتصال به سرور رخ دهد، این موارد ثبت و مدیریت می‌شوند. این اطلاعات به ما کمک می‌کند تا داده‌های دقیق و کاملی را برای تحلیل‌های بعدی فراهم کنیم.

داده‌های نهایی به صورت دوره‌ای ذخیره و به‌روز می‌شوند تا از دست رفتن داده‌ها جلوگیری شود و در نهایت فایل CSV شامل تمام اطلاعات مورد نیاز برای استفاده در سیستم توصیه‌گر می‌باشد. این اطلاعات به ما امکان می‌دهد تا به طور مؤثرتری رفتارهای کاربران را تحلیل کرده و توصیه‌های دقیق‌تری ارائه دهیم.

۱۱ تولید رتبه‌بندی عددی

در این بخش، قوانین و فرآیندهای مربوط به تخصیص رتبه‌بندی عددی به کاربران بر اساس تعاملات آن‌ها با صفحات اینستاگرام توضیح داده شده است. این تعاملات شامل دنبال کردن یک صفحه، کامنت‌گذاری بر روی پست‌ها و دیگر معیارهای تعامل می‌باشد. به دلیل محدودیت‌های اینستاگرام در دسترسی به داده‌های مربوط به لایک‌ها، این معیار در محاسبات در نظر گرفته نشده است.

۱۱-۱ قوانین رتبه‌بندی

برای تخصیص رتبه‌بندی عددی به کاربران، از قوانین زیر استفاده شده است:

- **دنبال کردن یک صفحه**: ۰/۳ امتیاز
- **کامنت‌گذاری بر روی پست‌ها**: ۰/۲ امتیاز

۱۱-۲ فرآیند جمع‌آوری داده‌ها

برای جمع‌آوری داده‌های تعاملات کاربران، از روش‌های زیر استفاده شد:

- **جمع‌آوری اطلاعات دنبال‌کنندگان**:
  - برای هر صفحه انتخابی، لیست دنبال‌کنندگان استخراج شد.
- **جمع‌آوری اطلاعات کامنت‌گذاران**:
  - برای هر پست، لیست کاربران کامنت‌گذار استخراج شد.
  - برای هر پست، تعداد محدودی کامنت بررسی شد تا از مصرف بیش از حد منابع جلوگیری شود.
- **ذخیره‌سازی داده‌ها**:
  - داده‌های جمع‌آوری شده در قالب فایل‌های CSV و JSON ذخیره شدند تا برای مراحل بعدی تحلیل و ارزیابی آماده باشند.

۱۱-۳ محاسبه امتیاز کاربران

امتیاز هر کاربر بر اساس تعاملات او با صفحات مختلف به شرح زیر محاسبه شده است:

- **استخراج تعاملات**:

  ■ برای هر کاربر، لیست صفحاتی که با آن‌ها تعامل داشته، استخراج شد.

- **محاسبه امتیاز**:

  ■ برای هر صفحه‌ای که کاربر با آن تعامل داشته، امتیاز مربوط به هر نوع تعامل (دنبال کردن، کامنت) محاسبه شد.

  ■ مجموع امتیازات تعاملات مختلف برای هر صفحه جمع‌آوری و به عنوان امتیاز نهایی کاربر ثبت شد.

۴-۱۱ چالش‌ها و راهکارها

در فرآیند تخصیص رتبه‌بندی عددی به کاربران، چالش‌های مختلفی وجود دارد:

- **رفتارهای تعاملی جانب‌دارانه**:

  ■ برخی کاربران ممکن است بیشتر به کامنت‌گذاری تمایل داشته باشند و برخی دیگر به دنبال کردن صفحات، که ممکن است باعث توزیع نامتناسب امتیازات شود. برای مقابله با این چالش، وزن‌دهی دقیق به هر نوع تعامل براساس میزان اهمیت آن انجام شد.

- **محدودیت‌های اینستاگرام**:

  ■ به دلیل محدودیت‌های اینستاگرام در دسترسی به داده‌های مربوط به لایک‌ها، این معیار در محاسبات در نظر گرفته نشده است. به منظور جبران این محدودیت، تمرکز بیشتری بر روی داده‌های کامنت‌ها و دنبال کردن‌ها قرار گرفت.

- **تنوع تعاملات کاربران**:

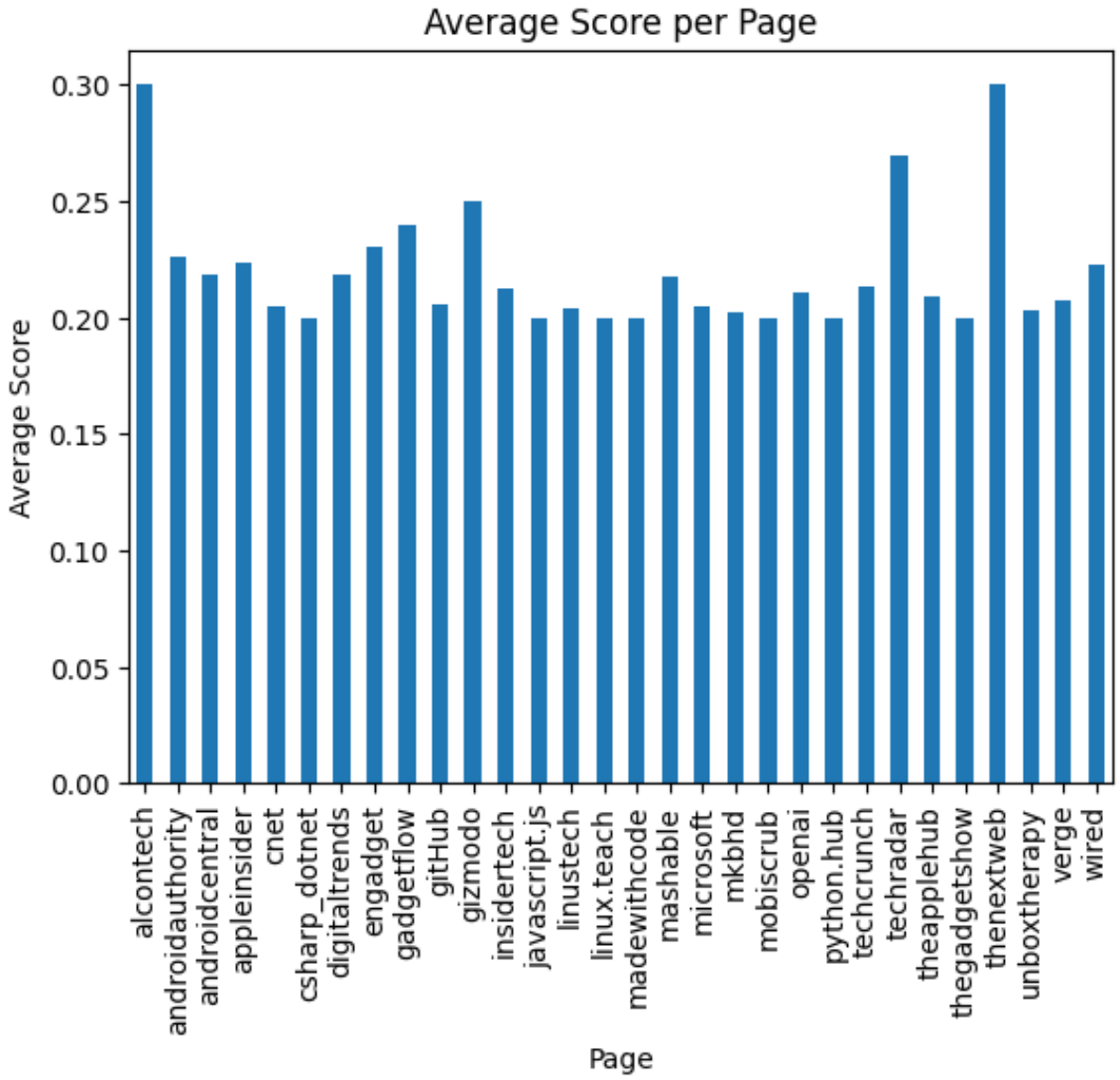
  ■ کاربران ممکن است تعاملات متفاوتی با صفحات مختلف داشته باشند. برای مدیریت این تنوع، الگوریتمی برای تحلیل و دسته‌بندی تعاملات کاربران توسعه داده شد.

- **زمان‌بندی جمع‌آوری داده‌ها**:

  ■ برای جلوگیری از مشکلات مرتبط با زمان‌بندی و حجم بالای درخواست‌ها، یک توقف زمانی بین هر درخواست قرار داده شد. این توقف‌ها به طور تصادفی تنظیم شدند تا از بلوکه شدن توسط سرور اینستاگرام جلوگیری شود.

۵-۱۱ نمایش امتیازات صفحات

در این بخش، میانگین امتیازات هر صفحه با توجه به تعاملات کاربران با آن‌ها نمایش داده شده است. نمودار زیر میانگین امتیازات هر صفحه را نشان می‌دهد که بر اساس تعاملات کاربران با این صفحات محاسبه شده است.



شکل ۶: میانگین امتیازات هر صفحه بر اساس تعاملات کاربران

این نمودار نشان می‌دهد که کاربران چگونه با صفحات مختلف تعامل داشته‌اند و کدام صفحات بیشتر مورد توجه و تعامل قرار گرفته‌اند. این اطلاعات به ما کمک می‌کند تا صفحات محبوب‌تر و با تعامل بیشتر را شناسایی کرده و توصیه‌های بهتری ارائه دهیم.

۱۲ پیاده‌سازی سیستم توصیه‌گر با داده‌های اینستاگرام

۱۲-۱ مقدمه

در این بخش، سیستم توصیه‌گر مبتنی بر LSH که پیش‌تر پیاده‌سازی شده است، بر روی داده‌های جمع‌آوری شده از اینستاگرام اعمال می‌شود. هدف این است که بر اساس تعاملات کاربران با صفحات مختلف اینستاگرام، توصیه‌های شخصی‌سازی شده‌ای برای کاربران تولید شده و عملکرد سیستم توصیه‌گر ارزیابی شود.

۱۲-۲ روش اجرا

۱. **آماده‌سازی داده‌ها**:

- داده‌های اینستاگرام بارگذاری شده و ستون‌ها برای مطابقت با فرمت داده‌های MovieLens تغییر نام داده شدند.
- داده‌ها به مجموعه‌های آموزش و اعتبارسنجی تقسیم شدند.
- داده‌های آموزش نرمال‌سازی شده و به فرمت ماتریس اسپارس تبدیل شدند.

۲. **فیلترینگ مشارکتی مبتنی بر LSH**:

- بردارهای هش ایجاد شده و کاربران بر اساس این هش‌ها به باکت‌ها تقسیم‌بندی شدند.
- رتبه‌بندی‌ها برای مجموعه اعتبارسنجی پیش‌بینی شده و خطای اعتبارسنجی محاسبه شد.

۳. **تولید توصیه‌ها**:

- از ماتریس پیش‌بینی شده برای تولید سه‌تایی‌های کاربر-صفحه-امتیاز استفاده شد.
- امتیازات به اقداماتی مانند 'کامنت/لایک'، 'فالو' و 'فالو/لایک/کامنت' تبدیل شدند.

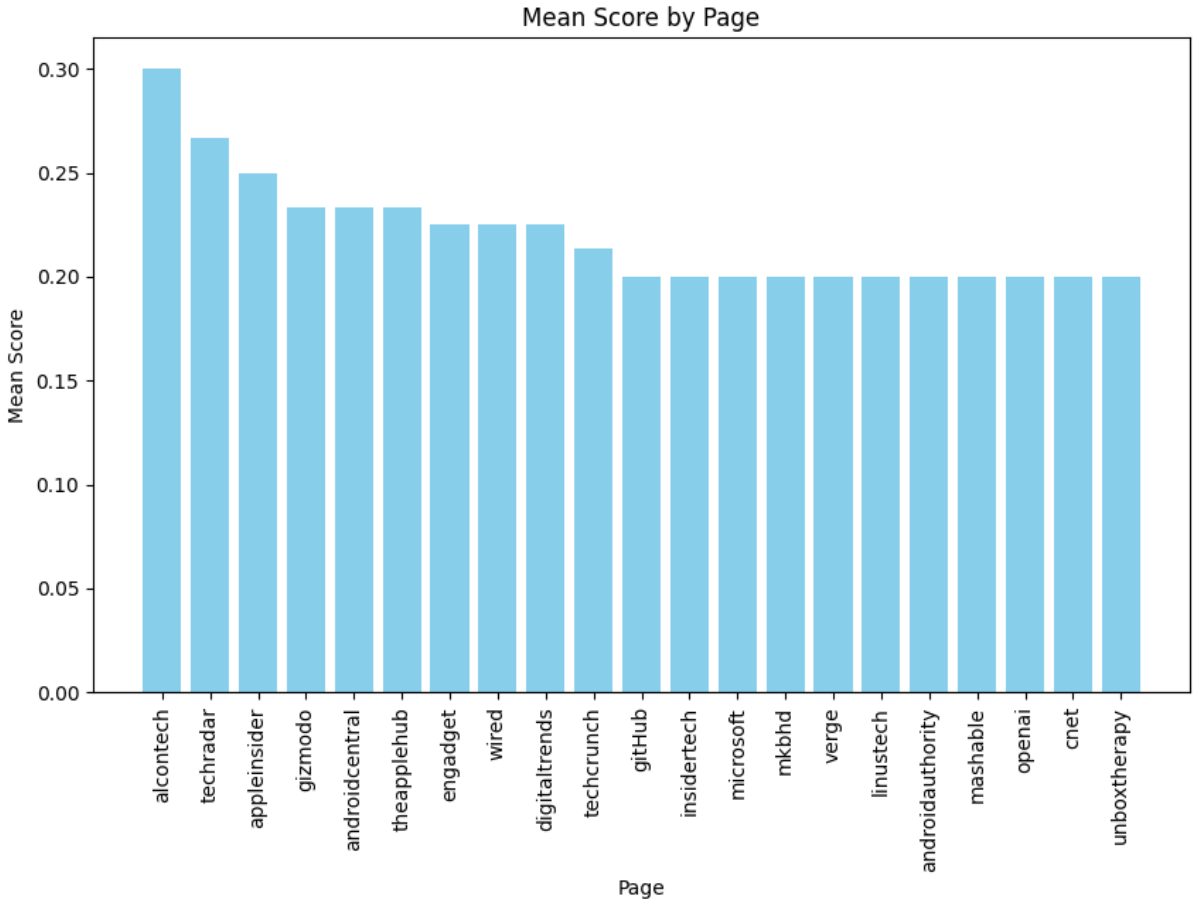
۴. **ارزیابی**:

- میانگین امتیازات بر اساس صفحه ترسیم شد.
- توزیع اقدامات توصیه شده تحلیل شد.

۱۲-۳ ارزیابی نتایج

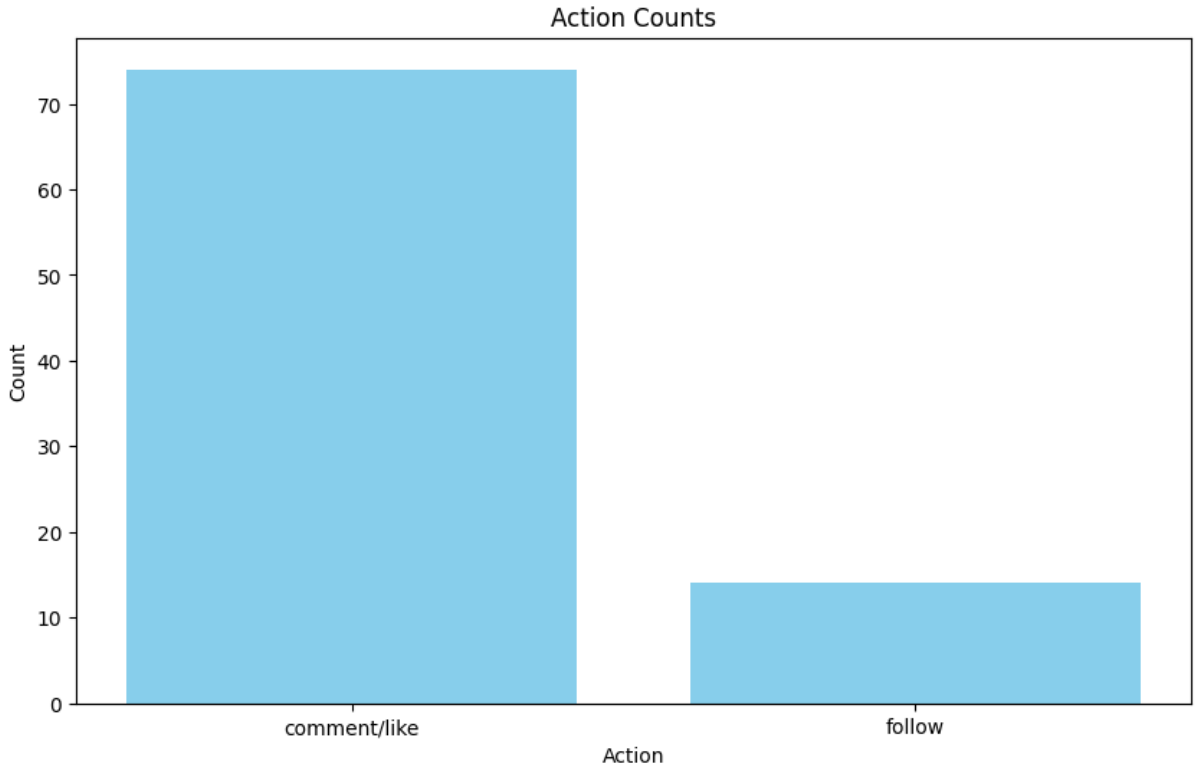
نتایج نشان داد که سیستم توصیه‌گر مبتنی بر LSH می‌تواند توصیه‌های معتبری برای کاربران اینستاگرام ارائه دهد. میانگین مربعات خطا (MSE) برای این داده‌ها تقریبا برابر با 1.5×10^{-34} محاسبه شد که

نشان‌دهنده دقت بالای مدل است. با بررسی میانگین امتیازات به ازای هر صفحه و تحلیل توزیع اقدامات توصیه شده، می‌توان به دقت و کارایی سیستم پی برد. نمودارهای زیر میانگین امتیازات بر اساس صفحه و توزیع اقدامات توصیه شده را نشان می‌دهند.



شکل ۷: میانگین امتیازات هر صفحه اینستاگرام

نمودار بالا نشان می‌دهد که کدام صفحات اینستاگرام بالاترین میانگین امتیاز را دارند و می‌توان گفت که اکثر کاربران صفحه alcontech را دوست خواهند داشت و احتمالا دنبال خواهند کرد.



شکل ۸: توزیع اقدامات توصیه شده برای کاربران

نمودار بالا توزیع اقدامات توصیه شده برای کاربران را نشان می‌دهد که احتمالا در مواجه شدن با یک پست از صفحات بالا چه واکنشی خواهند داشت. این تحلیل‌ها کمک می‌کنند تا بتوانیم رفتارهای احتمالی کاربران را پیش‌بینی کرده و توصیه‌های دقیق‌تری ارائه دهیم.

۴-۱۲ نمونه‌گیری از نتایج ارزیابی و تحلیل آن‌ها

در این بخش، ۳۰ نمونه از نتایج ارزیابی سیستم توصیه‌گر بر روی داده‌های اینستاگرام انتخاب شده و رتبه‌بندی‌های آن‌ها به تفصیل مورد بحث قرار می‌گیرد. هدف از این تحلیل، ارائه بینش‌هایی در مورد اثربخشی سیستم توصیه‌گر بر روی داده‌های سفارشی شما می‌باشد.

۱. **کاربر: ‘0btuseprocessor’**

- صفحه: cnet
- امتیاز: ۲.۰
- صفحه: digitaltrends

- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات cnet و digitaltrends علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲. **کاربر: 'adamsarwar'**

- صفحه: techcrunch
- امتیاز: ۲.۰
- صفحه: wired
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات techcrunch و wired علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۳. **کاربر: 'add12'**

- صفحه: insidertech
- امتیاز: ۲.۰
- صفحه: mashable
- امتیاز: ۲.۰
- صفحه: techcrunch
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات insidertech، mashable و techcrunch علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۴. **کاربر: 'afoteyannum'**

- صفحه: androidauthority
- امتیاز: ۲.۰
- صفحه: mkbhd
- امتیاز: ۲.۰

- صفحه: techcrunch
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات androidauthority، mkbhd و techcrunch علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۵. **کاربر: 'agentwalker'**

- صفحه: mashable
- امتیاز: ۲.۰
- صفحه: verge
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات mashable و verge علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۶. **کاربر: 'alberto.dsc'**

- صفحه: gitHub
- امتیاز: ۲.۰
- صفحه: techcrunch
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات gitHub و techcrunch علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۷. **کاربر: 'allmightsgone'**

- صفحه: microsoft
- امتیاز: ۲.۰
- صفحه: techradar
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات microsoft و techradar علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۸. **کاربر: 'astroeconomist'**

- صفحه: techcrunch
- امتیاز: ۲.۰
- صفحه: wired
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات techcrunch و wired علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۹. **کاربر: 'barefooteddragonkick'**

- صفحه: digitaltrends
- امتیاز: ۲.۰
- صفحه: engadget
- امتیاز: ۲.۰
- صفحه: mashable
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات digitaltrends، engadget و mashable علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۰. **کاربر: 'caulmseh'**

- صفحه: androidcentral
- امتیاز: ۲.۰
- صفحه: mashable
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات androidcentral و mashable علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۱. **کاربر: 'deborahn.1974'**

- صفحه: cnet
- امتیاز: ۲.۰
- صفحه: wired
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات cnet و wired علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۲. **کاربر: 'electrumojo'**

- صفحه: gizmodo
- امتیاز: ۲.۰
- صفحه: mashable
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات gizmodo و mashable علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۳. **کاربر: 'evandarkdj'**

- صفحه: digitaltrends
- امتیاز: ۲.۰
- صفحه: techcrunch
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات digitaltrends و techcrunch علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۴. **کاربر: 'exception_null'**

- صفحه: digitaltrends
- امتیاز: ۲.۰
- صفحه: mkbhd

- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات digitaltrends و mkbhd علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۵. **کاربر: 'frankthe1tank'**

- صفحه: androidauthority
- امتیاز: ۲.۰
- صفحه: verge
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات androidauthority و verge علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۶. **کاربر: 'gaurav_daryanani'**

- صفحه: microsoft
- امتیاز: ۲.۰
- صفحه: openai
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات microsoft و openai علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۱۷. **کاربر: 'hashim.alsughayer'**

- صفحه: appleinsider
- امتیاز: ۳.۰
- صفحه: theapplehub
- امتیاز: ۳.۰
- تحلیل: این کاربر به صفحات appleinsider و theapplehub علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند فالو، لایک یا کامنت واکنش نشان خواهد داد.

۱۸. **کاربر: 'jonty.pants'**

- صفحه: digitaltrends
- امتیاز: ۳.۰
- صفحه: engadget
- امتیاز: ۳.۰
- تحلیل: این کاربر به صفحات digitaltrends و engadget علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند فالو، لایک یا کامنت واکنش نشان خواهد داد.

۱۹. **کاربر: 'jordan_andrew22'**

- صفحه: digitaltrends
- امتیاز: ۳.۰
- صفحه: techcrunch
- امتیاز: ۳.۰
- صفحه: wired
- امتیاز: ۳.۰
- تحلیل: این کاربر به صفحات digitaltrends، techcrunch و wired علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند فالو، لایک یا کامنت واکنش نشان خواهد داد.

۲۰. **کاربر: 'kindofanmol'**

- صفحه: androidauthority
- امتیاز: ۲.۰
- صفحه: mkbhd
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات androidauthority و mkbhd علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۱. **کاربر: 'luxehouseplants'**

- صفحه: androidcentral
- امتیاز: ۳.۰
- صفحه: gizmodo
- امتیاز: ۳.۰
- صفحه: techradar
- امتیاز: ۳.۰
- تحلیل: این کاربر به صفحات androidcentral، gizmodo و techradar علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند فالو، لایک یا کامنت واکنش نشان خواهد داد.

۲۲. **کاربر: 'manu_pasta'**

- صفحه: techcrunch
- امتیاز: ۲.۰
- صفحه: verge
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات techcrunch و verge علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۳. **کاربر: 'mr.lost.music'**

- صفحه: gitHub
- امتیاز: ۲.۰
- صفحه: linustech
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات gitHub و linustech علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۴. **کاربر: 'nuno_hipolito'**

- صفحه: gizmodo

- امتیاز: ۲.۰
- صفحه: mashable
- امتیاز: ۲.۰
- صفحه: techcrunch
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات gizmodo، mashable و techcrunch علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۵. **کاربر: 'oniwura'**

- صفحه: mkbhd
- امتیاز: ۲.۰
- صفحه: unboxtherapy
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات mkbhd و unboxtherapy علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۶. **کاربر: 'ovi_wan_kenobi'**

- صفحه: androidcentral
- امتیاز: ۲.۰
- صفحه: engadget
- امتیاز: ۲.۰
- صفحه: linustech
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات androidcentral، engadget و linustech علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۷. **کاربر: 'r_sharma5'**

- صفحه: appleinsider

- امتیاز: ۲.۰
- صفحه: theapplehub
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات appleinsider و theapplehub علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۸. **کاربر: 'rob.gpt'**

- صفحه: gitHub
- امتیاز: ۲.۰
- صفحه: linustech
- امتیاز: ۲.۰
- صفحه: mkbhd
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات gitHub، linustech و mkbhd علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۲۹. **کاربر: 'seizethade'**

- صفحه: microsoft
- امتیاز: ۲.۰
- صفحه: techcrunch
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات microsoft و techcrunch علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.

۳۰. **کاربر: 'socialmediaisfake21'**

- صفحه: mashable
- امتیاز: ۲.۰

- صفحه: wired
- امتیاز: ۲.۰
- تحلیل: این کاربر به صفحات mashable و wired علاقه‌مند است و احتمالاً با تعاملات بیشتری مانند لایک یا کامنت واکنش نشان خواهد داد.