RECOMMENDATION SYSTEM

گزارش پروژه درس یادگیری ماشین

استاد پروژه

دکتر جواد سلیمی سرتختی

FEBRUARY 25, 2024

پویا طوقیان
Kashanu.ac.ir


بسم الله الرحمن الرحيم

۱- بخش اول

۱-۱ مقدمه :

مقاله ای که در ابتدا برای ارائه اتخاب شده بود ، پژوهشی بود که بر روی استفاده از یک الگوریتم مناسب برای پیشنهاد دادن فیلم بر اساس فیلترینگ بر محتوا انجام شده بود .

نام مقاله :

"Movie Popularity and Target Audience Prediction Using the Content-Based Recommender System"

این مقاله با عنوان فوق در مجله ی IEEE Access چاپ شده است ، همچنین این مقاله در تاریخ ۱۸ آوریل ۲۰۲۲ میلادی منتشر شده است . نویسندگان این مقاله عبارتند از :

- Sandipan Sahu
- Raghvendra Kumar
- Mohd Shafi Pathan
- Jana Shafi
- Yogesh Kumar
- Muhammad Fazal Ijaz

این مقاله به پیش‌بینی محبوبیت یک فیلم و گروه هدف آن با استفاده از یک سیستم توصیه‌گر مبتنی بر محتوا می‌پردازد. سیستم توصیه‌گر مبتنی بر محتوا بر اساس ویژگی‌های محتوای فیلم مثل ژانر، بازیگران، کارگردان و غیره، فیلم‌های مشابه را پیشنهاد می‌دهد. با استفاده از اطلاعات فیلم‌های مشابه، می‌توان محبوبیت فیلم جدید و گروه هدف آن را پیش‌بینی کرد .

۱-۲ خلاصه ایی از بخش مقدمه و چکیده این مقاله :

- این مقاله در مورد پیش بینی محبوبیت فیلم و مخاطب هدف آن با استفاده از سیستم توصیه گر مبتنی بر محتوا است .
- مقدمه بیان می کند که صنعت فیلم سازی در سراسر جهان یکی از صنایع پررونق و مهم است اما تنها تعداد معدودی از فیلم ها موفق می شوند .
- پژوهشگران اغلب احساس می کنند که نیاز به سیستم های خبره ای دارند که بتواند احتمال موفقیت یک فیلم را پیش از تولید آن با دقت قابل قبولی پیش بینی کند .
- اکثر مطالعات گذشته بر پیش بینی محبوبیت فیلم در مراحل پس از تولید متمرکز بوده اند که برای سرمایه گذاران مفید نیست .

- این مطالعه یک سیستم توصیه گر مبتنی بر محتوا با استفاده از ویژگی های اولیه فیلم مانند ژانر، بازیگران، کارگردان و غیره پیشنهاد می کند .
- سپس با استفاده از خروجی سیستم توصیه گر و اطلاعات رتبه بندی و رأی فیلم های مشابه، یک مجموعه ویژگی جدید ایجاد می شود .
- در نهایت یک مدل یادگیری عمیق **CNN** برای طبقه بندی چندکلاسه محبوبیت فیلم پیشنهاد شده است .

۱-۳ داده های که این مقاله برای پیش بینی از آنها استفاده کرده است :

- برای سیستم توصیه‌گر مبتنی بر محتوا از دو دیتاست tmdb_5000_movies و tmdb_5000_credits استفاده شده است که داده‌های عمومی هستند.
- برای پیش‌بینی موفقیت فیلم و گروه هدف از دیتاست امتیازدهی IMDb استفاده شده است.
- به دلیل استفاده از دو دیتاست مجزا، مشکل همگام‌سازی بین آن‌ها وجود داشته که با استفاده از دیتاست لینک کوچک برطرف شده است.

۱-۴ الگوریتم پیشنهادی در مقاله :

الگوریتم این مقاله به پیشنهاد دادن فیلم **با استفاده از KNN چندبعدی** می پردازد .

این الگوریتم برای توصیه فیلم‌های مشابه با استفاده از روش K نزدیک‌ترین همسایه چند بعدی استفاده شده است.

- ویژگی‌های فیلم شامل توضیحات، کلمات کلیدی، ژانر، کارگردان و بازیگران استفاده شده‌اند .
- برای محاسبه شباهت بین توضیحات فیلم از TF-IDF استفاده شده است.
- سایر ویژگی‌ها به داده‌های دودویی تبدیل شده‌اند.
- فاصله بین هر دو فیلم با استفاده از محاسبه فاصله ویژگی‌ها و cosine similarity محاسبه می‌شود.
- در نهایت N فیلم مشابه با کمترین فاصله به عنوان خروجی الگوریتم برگردانده می‌شود

۲- بخش دوم

۲-۱ پیاده سازی الگوریتم :

برای پیاده سازی این کار انجام شده (پیشنهاد دهنده ی فیلم) با استفاده از یک الگوریتم متفاوت از آنچه در مقاله ذکر شده است ، لازم است یک سری توضیحات در مورد بحث recommendation system مخصوصا بخش فیلترینگ مبتنی بر محتوا (Content-Based Filtering) بدهم :

در کل فیلترینگ مبتنی بر محتوا (Content-Based Filtering) یکی از روش‌های مؤثر در سیستم‌های پیشنهادگر است که بر اساس خصوصیات و ویژگی‌های آیتم‌ها (مانند فیلم‌ها) و ترجیحات کاربران کار می‌کند . برای رسیدن به یک نتیجه ی بهینه و قابل قبول میتوان از استراتژی و تکنیک هایی استفاده کرد :

۲-۱-۱ استخراج ویژگی‌ها (Feature Extraction) :

- **تجزیه و تحلیل متن** :برای فیلم‌ها، توضیحات، داستان‌ها، برچسب‌ها(tags) ، و بررسی‌ها می‌توانند منابع غنی از اطلاعات باشند. استفاده از روش‌های پردازش زبان طبیعی (NLP) مانند TF-IDF (Term Frequency-Inverse Document Frequency) یا Word2Vec برای تبدیل متن به وکتورهای عددی که می‌توانند به عنوان ویژگی‌هایی برای تجزیه و تحلیل مورد استفاده قرار گیرند.
- **تحلیل تصویر** :در مواردی که پوسترهای فیلم یا تصاویر مرتبط در دسترس باشند، می‌توان از تکنیک‌های بینایی ماشین برای استخراج ویژگی‌های بصری استفاده کرد.

۲-۱-۲ مدل‌سازی پروفایل کاربر (User Profile Modeling) :

- **ترکیب ویژگی‌ها** :پس از استخراج ویژگی‌ها، ساخت یک پروفایل کاربری که ترجیحات او را نشان می‌دهد با استفاده از این ویژگی‌ها ممکن است. این کار می‌تواند شامل میانگین‌گیری وزن‌دار ویژگی‌های فیلم‌هایی باشد که کاربر قبلاً از آن‌ها لذت برده است.
- **به‌روزرسانی دینامیک** :پروفایل‌های کاربری باید به طور پیوسته به‌روزرسانی شوند تا تغییرات در سلیقه‌ها و ترجیحات کاربران را منعکس کنند.

۲-۱-۳ الگوریتم‌های یادگیری ماشین:

- **دسته‌بندی** : استفاده از الگوریتم‌های دسته‌بندی مانند درخت تصمیم، رگرسیون لجستیک، یا ماشین‌های بردار پشتیبان (SVM) برای پیش‌بینی اینکه آیا یک فیلم خاص به یک کاربر پیشنهاد داده شود یا نه، بر اساس ویژگی‌های فیلم و پروفایل کاربر.

- **فیلتر کردن مبتنی بر محتوا با استفاده از شبکه‌های عصبی** :استفاده از شبکه‌های عصبی پیچشی (CNN) برای تحلیل تصویر و شبکه‌های عصبی بازگشتی (RNN) یا مدل‌های توجهی برای تجزیه و تحلیل متن می‌تواند در درک بهتر محتوا و تطابق آن با سلیقه‌ی کاربران بسیار مؤثر باشد.

۲-۱-۴ ارزیابی و بهینه‌سازی:

- **ارزیابی دقت** : استفاده از معیارهایی مانند دقت، فراخوانی ، و امتیاز F1 (f1_score) برای ارزیابی عملکرد مدل.
- **آزمایش A/B**[1] : اجرای آزمایش‌های A/B برای مقایسه عملکرد روش‌های مختلف و بهینه‌سازی تجربه‌ی کاربری بر اساس بازخورد واقعی.

استفاده از ترکیبی از این تکنیک‌ها و به کارگیری دقیق استراتژی‌های یادگیری ماشین می‌تواند به ساخت سیستم پیشنهادگر فیلم مبتنی بر محتوا کمک کند که توانایی ارائه پیشنهادهای معنادار و دقیق بر اساس سلیقه‌های فردی کاربران را دارد .

موارد بالا بحث های کلی از فیلترینگ مبتنی بر محتوا بود ، آنچه که میتواند برای بازنویسی و پیاده سازی الگوریتم پیشنهادی مقاله مدنظر به ما کمک کند بخش ۳ دسته بندی فوق است .

در این بخش انواع الگوریتم های یادگیری ماشین مورد استفاده را برای دسته بندی داده ها و پیش بینی کردن یک یا چند فیلم خاص به یک کار بر را نام برده است . **الگوریتم‌های دسته‌بندی مانند درخت تصمیم، رگرسیون لجستیک، یا ماشین‌های بردار پشتیبان (SVM) .**

انتخاب ما برای ادامه ی کار پیاده سازی الگوریتم پیشنهادی در مقاله ، استفاده از ماشین های بردار پشتیبان SVM است . برای استفاده از SVM در فیلترینگ مبتنی بر محتوا، می‌توانیم یک سناریو ساده را در نظر بگیریم که در آن قصد داریم بر اساس ویژگی‌های فیلم‌ها (مثل ژانر، مدت زمان، و امتیاز و .....) به کاربران فیلم‌هایی را پیشنهاد دهیم که ممکن است مورد علاقه‌شان باشد. در این مثال، فرض می‌کنیم که ویژگی‌های هر فیلم به صورت عددی ارائه شده‌اند و ما از SVM برای دسته‌بندی فیلم‌ها استفاده می‌کنیم تا تعیین کنیم که آیا یک فیلم خاص برای کاربر پیشنهاد شود یا نه.

۲-۲ توضیح روش کار

ابتدا در مورد روش کار پایه ای در فیلترینگ مبتنی بر محتوا صحبت میکنیم و در ادامه الگوریتمم را به سمت SVM میبریم .

برای توضیح فرآیندی که در فیلترینگ مبتنی بر محتوا برای تبدیل داده‌ها به فرمت قابل استفاده توسط الگوریتم‌ها انجام می‌شود، ابتدا باید یک سری مراحل پیش‌پردازش داده‌ها را در نظر گرفت. این فرآیند شامل خواندن داده‌ها، پاک‌سازی و انتخاب ویژگی‌ها می‌باشد . در اینجا به توضیح مفصل هر بخش و نحوه پیاده‌سازی آن می‌پردازیم:

[1]- آزمون آ/ب به انگلیسی (A/B testing) : عنوان فرایندی است که برای تشخیص اینکه از میان دو ویژگی «آ» و «ب» کدام‌یک مناسب‌تر است به کار می‌رود. در آزمون‌های آ/ب، دو پیاده‌سازی متفاوت به صورت آزمایشی به دو گروه از کاربران ارائه می‌شود. مقایسهٔ نتایج به‌دست‌آمده از گروه‌ها می‌تواند به انتخاب پیاده‌سازی مناسب‌تر کمک کند

۲-۲-۱ خواندن داده‌ها

در ابتدا، داده‌های مورد نیاز از منابع مختلفی مانند پایگاه‌داده‌ها، فایل‌های متنی یا API ها خوانده می‌شوند. این داده‌ها ممکن است شامل اطلاعات کاربران، آیتم‌ها و تعاملات بین آن‌ها باشد.

۲-۲-۲ پیش‌پردازش داده‌ها

پیش‌پردازش شامل چندین مرحله است:

- **تمیزکاری داده‌ها** :

تمیزکاری داده‌ها فرایندی است که در آن داده‌های نامناسب، ناقص، تکراری یا نادرست از مجموعه داده حذف یا اصلاح می‌شوند تا کیفیت داده‌ها برای تحلیل‌های بعدی بهبود یابد. این فرایند شامل بررسی دقیق داده‌ها و اعمال اقداماتی مانند پر کردن مقادیر گمشده، حذف رکوردهای تکراری، و اصلاح اشتباهات تایپی یا فرمت‌بندی است. تمیز کردن داده‌ها به حفظ دقت و قابلیت اعتماد تحلیل‌ها کمک می‌کند

- **نرمال‌سازی داده‌ها** :

نرمال‌سازی داده‌ها به فرایند تبدیل ویژگی‌ها به یک مقیاس مشترک برای جلوگیری از بی‌تاثیری ویژگی‌های با مقیاس بزرگ‌تر اشاره دارد. این کار مهم است زیرا الگوریتم‌های یادگیری ماشین ممکن است به طور ناعادلانه توسط ویژگی‌هایی با مقیاس‌های بزرگ‌تر تاثیر پذیرند. نرمال‌سازی به اطمینان از اینکه هر ویژگی به طور عادلانه در تحلیل دخیل باشد کمک می‌کند .

- **تبدیل داده‌ها** :

تبدیل داده‌ها شامل اعمال تغییرات یا تبدیلات روی داده‌ها برای ایجاد ویژگی‌های جدید یا تغییر ویژگی‌های موجود است. این تبدیلات ممکن است شامل ایجاد ویژگی‌های مشتق شده، تبدیل لگاریتمی، برداری‌سازی متن، و دیگر انواع پیش‌پردازش برای بهبود کیفیت مدل‌سازی یادگیری ماشین باشد

۲-۲-۳ انتخاب ویژگی‌ها

انتخاب ویژگی‌های مهم و کاربردی از میان تمام ویژگی‌های موجود، بر اساس این که کدام یک از آن‌ها کمترین خطا و اشکال را در داده‌ها ایجاد می‌کنند و بیشترین اطلاعات مرتبط با تعاملات کاربران و آیتم‌ها را فراهم می‌آورند. این فرآیند می‌تواند شامل روش‌های آماری، مدل‌های مبتنی بر داده و تکنیک‌های یادگیری ماشین برای ارزیابی اهمیت ویژگی‌ها باشد

۲-۲-۴ وزن‌دهی ماتریس ویژگی‌های آیتم

پس از انتخاب ویژگی‌های اساسی، برای هر ویژگی در ماتریس آیتم، وزنی بر اساس اهمیت آن ویژگی در تعیین ترجیحات کاربر اختصاص می‌دهیم. این کار می‌تواند با استفاده از تکنیک‌هایی مانند تحلیل عاملی یا روش‌های انتخاب ویژگی مبتنی بر داده‌ها انجام شود.

۲-۲-۵ ایجاد ماتریس پروفایل کاربر

ماتریس پروفایل کاربر با استفاده از امتیازات (ratings) داده‌شده توسط کاربران به آیتم‌ها و وزن ویژگی‌های مرتبط با آن آیتم‌ها ساخته می‌شود. این کار از طریق ضرب ماتریسی امتیازات کاربران در وزن‌های ویژگی‌های آیتم‌ها انجام می‌شود تا نشان‌دهنده میزان ترجیح کاربر به هر ویژگی باشد.

۲-۲-۶ وزن‌دهی بر اساس امتیازات کاربر

برای اعمال امتیازات کاربر به کل ویژگی‌ها، هر ورودی در ماتریس پروفایل کاربر باید با توجه به امتیازات کلی که کاربر به آیتم‌ها داده است، تنظیم شود. این امر می‌تواند با محاسبه میانگین وزن‌دار امتیازات بر اساس اهمیت هر آیتم یا ویژگی انجام شود.

پس از ایجاد ماتریس پروفایل کاربر و وزن‌دهی به ویژگی‌های آیتم‌ها، مرحله نهایی فرآیند فیلترینگ مبتنی بر محتوا، پیش‌بینی و ارائه پیشنهادات به کاربران است. این مرحله شامل مقایسه پروفایل کاربر با ویژگی‌های آیتم‌هایی است که هنوز توسط کاربر مشاهده یا ارزیابی نشده‌اند. در ادامه به نحوه انجام این فرآیند پرداخته می‌شود.

۲-۲-۷ محاسبه امتیاز پیش‌بینی شده

برای هر آیتم جدید، یک امتیاز پیش‌بینی شده بر اساس میزان تطابق ویژگی‌های آیتم با پروفایل کاربر محاسبه می‌شود. این کار معمولاً از طریق ضرب داخلی بین بردار ویژگی‌های آیتم و بردار پروفایل کاربر انجام می‌شود. امتیاز پیش‌بینی شده نشان‌دهنده میزان علاقه‌مندی پیش‌بینی شده کاربر به آیتم است.

۲-۲-۸ رتبه‌بندی و ارائه پیشنهادات

پس از محاسبه امتیازات پیش‌بینی شده برای تمام آیتم‌های جدید، آیتم‌ها بر اساس امتیازات پیش‌بینی شده به ترتیب نزولی رتبه‌بندی می‌شوند. آیتم‌هایی با بالاترین امتیاز پیش‌بینی شده به عنوان پیشنهادات به کاربر ارائه می‌شوند. این رویکرد اطمینان حاصل می‌کند که پیشنهادات ارائه شده بیشترین تطابق را با علایق و ترجیحات فردی کاربر دارند.

۲-۲-۹ بهینه‌سازی پیشنهادها

- **تنوع** :برای جلوگیری از ارائه توصیه‌های تکراری و افزایش تنوع در پیشنهادها، می‌توان از الگوریتم‌های تنوع‌بخش استفاده کرد.
- **جدیدترین‌ها** :ترجیح دادن آیتم‌های جدیدتر برای حفظ ارتباط پیشنهادها با زمان حال.
- **فیلتر کردن پس از پیش‌بینی** :اعمال محدودیت‌ها و فیلترها برای حذف آیتم‌های نامناسب یا غیرقابل دسترس.

۲-۲-۱۰ محاسبه پیش‌بینی‌ها

برای هر کاربر و آیتم، یک پیش‌بینی امتیاز بر اساس میزان تطابق بین پروفایل کاربر و ویژگی‌های آیتم تولید می‌شود. این تطابق می‌تواند از طریق محاسبه محصول نقطه‌ای بین وکتور ویژگی‌های کاربر در ماتریس پروفایل کاربر و وکتور ویژگی‌های آیتم انجام شود. هرچه این مقدار بزرگ‌تر باشد، نشان‌دهنده تطابق بالاتر و احتمال بیشتر برای پسندیدن آیتم توسط کاربر است.

۲-۳ چرا SVM:

سیستم‌های توصیه‌گر مبتنی بر محتوا (Content-Based Recommendation Systems) به این شکل کار می‌کنند که ویژگی‌های مربوط به آیتم‌ها (مانند فیلم‌ها) و ترجیحات کاربر را تحلیل می‌کنند تا توصیه‌هایی را ارائه دهند که به طور خاص برای هر کاربر مناسب باشد. ماشین بردار پشتیبان (Support Vector Machine) یا (SVM) یک ابزار قدرتمند برای دسته‌بندی است که می‌تواند در سیستم‌های توصیه‌گر مبتنی بر محتوا برای پیش‌بینی اینکه آیا یک کاربر خاص یک فیلم را دوست خواهد داشت یا نه، مورد استفاده قرار گیرد. در ادامه به توضیح دلایل استفاده از SVM در این زمینه می‌پردازیم:

۲-۳-۱ کارایی در دسته‌بندی دوگانه

SVM به طور خاص برای مسائل دسته‌بندی دوگانه طراحی شده است. در مورد سیستم‌های توصیه‌گر مبتنی بر محتوا، مسئله می‌تواند به صورت پیش‌بینی اینکه آیا یک کاربر یک فیلم را می‌پسندد (مثبت) یا نمی‌پسندد (منفی) تبدیل شود SVM . با یافتن یک حاشیه (margin) حداکثر بین دو کلاس می‌تواند به خوبی این دسته‌بندی را انجام دهد.

۲-۳-۲ قابلیت تعمیم بالا

SVM با استفاده از تکنیک‌هایی مانند کرنل تریک (Kernel Trick) قادر است در فضاهای ویژگی با ابعاد بالا عمل کند، بدون اینکه دچار مشکل بیش‌برازش (Overfitting) شود. این ویژگی به SVM امکان می‌دهد تا الگوهای پیچیده‌ای را که ممکن است بین ویژگی‌های فیلم‌ها و ترجیحات کاربر وجود داشته باشد، کشف کند.

۲-۳-۳ انعطاف‌پذیری در انتخاب کرنل

SVM اجازه می‌دهد تا از کرنل‌های مختلفی استفاده شود، مانند خطی، چندجمله‌ای، رادیال بیس (RBF) و سیگموئید. این انعطاف‌پذیری به توسعه‌دهندگان اجازه می‌دهد تا کرنل مناسب را برای ماهیت داده‌ها و مسئله مورد نظر انتخاب کنند. به عنوان مثال، اگر روابط غیرخطی قوی بین ویژگی‌های فیلم‌ها و امتیازات کاربران وجود داشته باشد، استفاده از کرنل RBF می‌تواند مفید باشد.

۲-۳-۴ مقاومت در برابر نویز

SVM توانایی مقاومت در برابر نویز داده‌ها و نمونه‌های پرت (Outliers) را دارد. این ویژگی به ویژه در موقعیت‌هایی که داده‌ها شامل خطاهای اندازه‌گیری یا امتیازدهی‌های نادرست هستند، اهمیت پیدا می‌کند.

۲-۳-۵ کاربرد در محتوای متنوع

SVM نه تنها می‌تواند برای داده‌های عددی بلکه برای داده‌های متنی نیز مورد استفاده قرار گیرد، که این امر آن را برای سیستم‌های توصیه‌گر که ممکن است شامل توصیفات متنی فیلم‌ها باشند، ایده‌آل می‌سازد. به کمک تکنیک‌هایی مانند TF-IDF یا Word2Vec ، می‌توان ویژگی‌های متنی را به فضاهای عددی تبدیل کرد که SVM قادر به کار با آن‌ها است.

بنابراین ، با توجه به قابلیت‌های فوق، SVM یک ابزار قدرتمند و انعطاف‌پذیر برای سیستم‌های توصیه‌گر مبتنی بر محتوا است که می‌تواند به طور مؤثری در پیش‌بینی پسند یا نپسندیدن فیلم‌ها توسط کاربران به کار رود.

۳- بخش سوم

۳-۱ توضیح خط به خط کد پیاده سازی :

۳-۱-۱ وارد کردن کتابخانه های مورد نیاز :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
import seaborn as sns
import datetime
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
```

۳-۱-۲ خواندن دیتاست های مورد استفاده

```
movies = pd.read_csv("tmdb_5000_movies.csv")
credits = pd.read_csv("tmdb_5000_credits.csv")
```

۳-۱-۳ استخراج سال انتشار فیلم :

```
movies.release_date = pd.to_datetime(movies["release_date"])
movies['year'] = movies.release_date.dt.year
```

۳-۱-۴ تبدیل داده از فرمت json به string :

```
from pandas.io.json import json_normalize
import json
def json_decode(data , key) :
    """
description :this function can be helpful to perform decodeing in python of json string .
Arguments :
    data : the data that we want decoding
    key : the key that we want ti return its value

Returns : list of values
"""
    result = []
    data = json.loads(data)
    for ithem in data :
        result.append(ithem[key])

    return result
```

```
columns = ["genres" , "keywords" , "production_companies"  ,
           "production_countries", "cast" , "crew" , "spoken_languages"]

for i in columns :
   movies[i] = movies[i].apply( json_decode , key = 'name')
```

کد ارائه شده شامل یک تابع به نام json_decode است که برای استخراج و بازگرداندن مقادیر خاص از یک رشته JSON در پایتون استفاده می‌شود. ابتدا، دو کتابخانه json و json_normalize از ماژول pandas.io.json وارد می‌شوند .

این تابع دو پارامتر دریافت می‌کند:

- **data** :داده‌ای که می‌خواهیم از آن رمزگشایی انجام دهیم. انتظار می‌رود که این داده یک رشته JSON باشد.
- **key** :کلیدی که می‌خواهیم مقادیر مرتبط با آن را بازیابی کنیم

همچنین در بخش دوم این بخش دیتافریم movies با فرمت جدید داده ها آپدیت میشود .

۳-۱-۴ تمیز کردن داده ها از کلمات نامفهوم (Nan)

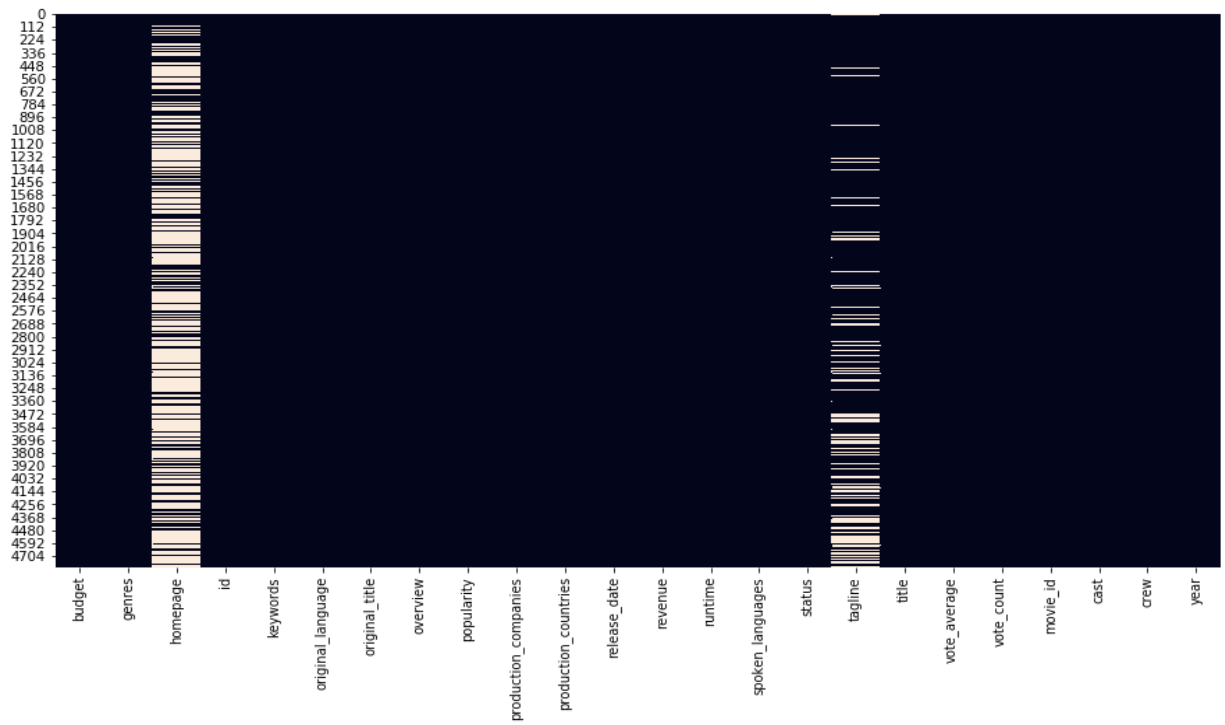
```
#clean data
missing = movies.isnull().sum()
print("the sum of error data : " , missing.sum())

null_columns = [feature for feature in movies.columns if movies[feature].isnull().sum() > 0]
for col in null_columns:
    print(col + "------" + str(movies[col].isnull().sum()))
```

۳-۱-۴-۱ بخش کد ارائه شده برای پیدا کردن و چاپ کردن تعداد داده‌های گمشده (مقادیر null) در یک DataFrame به نام movies طراحی شده است .

- پیدا کردن داده‌های گمشده : از تابع isnull() روی DataFrame movies استفاده می‌شود تا یک DataFrame جدیدی تولید شود که هر سلول آن نشان‌دهنده وضعیت گمشده (با مقدار True) یا غیرگمشده با مقدار (False) مقادیر اصلی است.
- جمع‌بندی داده‌های گمشده : سپس، با استفاده از تابع sum() روی نتیجه حاصل از isnull() تعداد داده‌های گمشده در هر ستون از DataFrame movies محاسبه می‌شود. تابع sum() اولین بار برای جمع‌آوری تعداد True ها (که نشان‌دهنده مقادیر گمشده هستند) در هر ستون اجرا می‌شود.

این روش یک بخش مهم از پیش‌پردازش داده‌ها در تجزیه و تحلیل داده و یادگیری ماشین است، که به شناسایی و رسیدگی به مشکلات کیفیت داده کمک می‌کند.



the picture of errors (Nan) data in database : شکل ۱

```
high_missing = missing[missing > 500].index
movies.drop(high_missing , axis = 1 , inplace = True)
print(high_missing)
print(movies.shape)
```

۳-۱-۴-۲ در این بخش کد، مراحل زیر برای حذف ستون‌هایی با تعداد زیاد مقادیر گمشده (بیش از ۵۰۰ مورد) از DataFrame movies انجام می‌شود:

- **تعیین ستون‌های با تعداد زیاد مقادیر گمشده :**ابتدا، ستون‌هایی که تعداد مقادیر گمشده آن‌ها بیشتر از ۵۰۰ است، با استفاده از missing[missing > 500] شناسایی می‌شوند. index. این دستور، ایندکس (نام) این ستون‌ها را برمی‌گرداند و در متغیر high_missing ذخیره می‌کند.
- حذف ستون‌های با تعداد زیاد مقادیر گمشده : سپس، با استفاده از تابع drop() روی DataFrame movies, ستون‌های شناسایی شده در high_missing حذف می‌شوند. پارامتر axis=1 نشان می‌دهد که حذف بر اساس ستون‌ها انجام می‌گیرد. Inplace = True به معنای این است که تغییرات مستقیماً روی DataFrame اصلی اعمال می‌شوند، بدون نیاز به اختصاص دادن نتیجه به یک متغیر جدید .

۳-۱-۵ پیش پردازش ویژگی های مد نظر :

این بخش کد دو تابع را برای رمزگذاری یک‌باره (one-hot encoding) برای ویژگی‌های چندبرچسب (multilabel) در یک DataFrame pandas ارائه می‌دهد. این روش به خصوص در پیش‌پردازش داده‌ها برای مدل‌سازی یادگیری ماشین استفاده می‌شود.

```
# functions

def one_hot_encode_multilabel (df, column_name):

    mlb_series = df[column_name].apply(lambda x : ' , '.join(x))


    mlb_df = mlb_series.str.get_dummies(sep=',')

    return mlb_df

def one_hot_encode_multilabel_original_language (df, column_name):

    mlb_series = df[column_name].apply(lambda x : ' '.join(x))


    mlb_df = mlb_series.str.get_dummies(sep = ' , ' )

    return  mlb_df


genres = one_hot_encode_multilabel(movies , "genres")
countries = one_hot_encode_multilabel(movies , "production_countries")
language = one_hot_encode_multilabel_original_language(movies , "original_language")

encoded_features = pd.concat([genres , language ,  countries], axis = 1)

encoded_features
```

- ابتدا، با استفاده از متد , apply(lambda x : ' , '.join(x)) هر لیست از برچسب‌ها در سلول‌های ستون مورد نظر به یک رشته تبدیل می‌شود که برچسب‌ها با کاما (,) از یکدیگر جدا شده‌اند.
- سپس، با استفاده از متد , str.get_dummies(sep=',') رمزگذاری یک‌باره بر اساس این رشته‌های تولید شده انجام می‌شود، که در نتیجه‌ی آن ، برای هر برچسب منحصر به فرد یک ستون جدید ایجاد می‌شود.

در نهایت، pd.concat برای ترکیب ویژگی های genres, language و countries در یک DataFrame واحد استفاده می‌شود که هر سه به صورت رمزگذاری یک‌باره پردازش شده‌اند. این ترکیب به شما امکان می‌دهد تا ویژگی‌هایی که برای مدل‌های یادگیری ماشین آماده شده‌اند را در یک مکان جمع‌آوری کنید .

در آخر، دیتا فریم encoded features ماتریسی از ویژگی های مورد استفاده میشود که خروجی آن در زیر آمده :

	Action	Adventure	Animation	Comedy	Crime	Documentary	Drama	Family	Fantasy	Foreign	...	Sweden	Switzerland
0	1	1	0	0	0	0	0	0	1	0	...	0	0
1	1	1	0	0	0	0	0	0	1	0	...	0	0
2	1	1	0	0	1	0	0	0	0	0	...	0	0
3	1	0	0	0	1	0	1	0	0	0	...	0	0
4	1	1	0	0	0	0	0	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4798	1	0	0	0	1	0	0	0	0	0	...	0	0
4799	0	0	0	1	0	0	0	0	0	0	...	0	0
4800	0	0	0	1	0	0	1	0	0	0	...	0	0
4801	0	0	0	0	0	0	0	0	0	0	...	0	0
4802	0	0	0	0	0	1	0	0	0	0	...	0	0

4799 rows × 145 columns

شکل ۲ : features matrix

۳-۱-۶ وارد کردن دادهای user به همراه امتیاز هایی که او به فیلم های مختلف داده است .

در واقع بر اساس تعیین نظرات این کاربر امتیاز ها ثبت شده اند و تاثیر این امتیاز ها در تمام داده اعمال میشود و بر اساس نظر این کاربر در آخر ۱۰ فیلم که بیشترین امتیاز را نسبت به تمام فیلم ها بدست آورده پیشنهاد میشود .

```
title = [ 'Harry Potter and the Prisoner of Azkaban', 'Half Baked',
          'The Chronicles of Riddick', 'Love Jones', 'My Name Is Khan',
          'The Bad Lieutenant: Port of Call - New Orleans', 'Chill Factor',
          'Aliens', 'Little Miss Sunshine', 'The Butterfly Effect' ]

Movie_id  = [673 , 9490 , 2789 ,27322 , 26022 , 11699 , 2162 , 679 , 773 , 1954 ]
Ratings = [8 , 6.5 , 6 ,7 , 8 , 6.1 , 5 , 7.8 , 7.6 , 6.9 ]
```

۳-۱-۷ تطابق داده های ورودی با دیتا فریم :

در این مرحله داده های ورودی کاربر در بین کل دیتاست تطبیق میشود و سایر ویژگی ها به آن اضافه میشود .

```
input_id = movies[movies["original_title"].isin(input_user_data[ "title" ].tolist() )]
input_user_data = pd.merge(input_id , input_user_data )
input_user_data
```

ما در این کد در حال تلاش برای پیدا کردن فیلم‌هایی در DataFrame movies هستیم که عنوان‌های اصلی آن‌ها با عناوین موجود در لیست داده های کاربر مطابقت داشته باشند . سپس، با استفاده از تابع ما pd.merge, اطلاعاتی را که بر اساس این عنوان‌ها پیدا کرده‌ایم با داده‌های کاربر (input_user_data) ادغام می‌کنیم . این ادغام به ما امکان می‌دهد تا اطلاعات مرتبط با فیلم‌های مورد علاقه کاربر را بر اساس DataFrame movies بازیابی و ترکیب کنیم.

۳-۱-۸ تطابق داده های ورودی با ماتریس تشکیل شده ویژگی ها :

به علاوه بر این که داده های ورودی را در دیتا فریم اصلی پیدا میکنیم ، باید این داده ها در دیتا فریم encoded features نیز پیدا کنیم و تطبیق دهیم تا بتوانیم ماتریس ویژگی های داده ی ورودی را هم بدست آوریم .

```
user_encoded_features = encoded_features[encoded_features["movie_id"].isin(input_user_data["movie_id"].tolist())]
```

کدی که در بالا مشاهده میکنید ، به منظور انتخاب ویژگی‌های رمزگذاری شده مربوط به فیلم‌های خاصی از DataFrame encoded_features است که شناسه‌های فیلم (movie_id) آن‌ها در لیست شناسه‌های موجود در input_user_data ["movie_id"] قرار دارد . این کار به ما این امکان را می‌دهد که فقط داده‌های مربوط به فیلم‌های مورد علاقه یا انتخاب شده توسط کاربر را فیلتر کنیم .

۳-۱-۹ تشکیل پروفیل کاربر برای تعین وزن هر یک از ویژگی ها :

در اینجا با توجه به ماتریس ویژگی های بدست آمده و امتیاز های کابر، یک لیست به نام پروفایل کابر میسازیم که تعیین کننده میزان اهمیت این کاربر به هریک از ویژگی هایی مد نظر ما بوده است .

```
user_profile = user_encoded_features.transpose().dot(input_user_data[" rating "])
user_profile.head (20)
```

در نتیجه، این کد به ما امکان می‌دهد تا بفهمیم کدام ویژگی‌های فیلم‌ها (مانند ژانرها، کشورهای تولید، زبان‌ها) برای کاربر مهم‌تر هستند ، بر اساس امتیازاتی که به فیلم‌های مختلف داده‌اند. این اطلاعات می‌توانند در سیستم‌های پیشنهادی برای ارائه پیشنهادهای دقیق‌تر به کاربران استفاده شوند.

۳-۱-۱۰ محاسبه امتیازات پیشنهادی بر اساس پروفایل کاربر :

```
recommendation_table = (( encoded_features * user_profile ).sum(axis = 1)) / (user_profile.sum())
recommendation_table
```

کد فوق یک جدول پیشنهادی را برای ما محاسبه می‌کند که در آن امتیازات پیشنهادی برای هر فیلم بر اساس پروفایل کاربر تعیین می‌شود. این فرآیند از دو مرحله اصلی تشکیل شده است:

- **ضرب ویژگی‌های رمزگذاری شده در پروفایل کاربر** : ابتدا، ویژگی‌های رمزگذاری شده هر فیلم (مثلاً ژانرها، کشورهای تولید کننده، زبان‌ها) در وزن‌ها یا اهمیت‌های مربوط به هر ویژگی که در پروفایل کاربر محاسبه شده، ضرب می‌شوند. این عمل باعث می‌شود که هر فیلم بر اساس علاقه‌مندی‌های شناخته شده کاربر امتیازدهی شود.

- **محاسبه میانگین وزن‌دار** : سپس، مجموع امتیازات حاصل از مرحله قبل برای هر فیلم، تقسیم بر مجموع کل وزن‌های پروفایل کاربر می‌شود. این کار نتایج را نرمالیزه می‌کند و امتیاز نهایی پیشنهادی برای هر فیلم را بر اساس علاقه‌مندی‌های کلی کاربر تعیین می‌کند.

نتیجه نهایی، یک جدول امتیازات پیشنهادی است که فیلم‌ها را بر اساس میزان مطابقت آن‌ها با پروفایل کاربر رتبه‌بندی می‌کند. این امتیازات می‌توانند برای ما تعیین کند که کدام یک از فیلم‌ها به احتمال زیاد بیشترین علاقه را از طرف کاربر خاص ما جلب می‌کنند.

۳-۱-۱۱ تعیین فیلم‌های پیشنهادی بر اساس آستانه امتیاز :

این بخش از کد به منظور دسته‌بندی فیلم‌ها به دو دسته "پیشنهادی" و "غیر پیشنهادی" بر اساس آستانه امتیازی مشخص شده است.

- **تعیین آستانه** :ابتدا آستانه‌ای برای امتیازدهی به فیلم‌ها تعیین می‌شود. این آستانه برابر است با بیشترین امتیاز دریافتی منهای ۰.۲. این روش امکان مقایسه فیلم‌ها را فراهم می‌آورد و فیلم‌هایی که امتیازشان نزدیک به بالاترین امتیاز است را به عنوان پیشنهادهای ممکن شناسایی می‌کند.

- **دسته‌بندی فیلم‌ها** : سپس، هر فیلم بر اساس امتیازش نسبت به آستانه تعیین شده، دسته‌بندی می‌شود. فیلم‌هایی که امتیازشان برابر یا بالاتر از آستانه است به عنوان "پیشنهادی" طبقه‌بندی شده و در لیست پیشنهادات قرار می‌گیرند. در مقابل، فیلم‌هایی که امتیازشان پایین‌تر از آستانه است به عنوان "غیر پیشنهادی" شناخته شده و ممکن است برای کاربر جذابیت کمتری داشته باشند.

```
scores = recommendation_table.values
threshold = max(scores) - 0.2
Final = []
for i in range(len(scores)) :
    if scores[i] >= threshold :
        title = "recommendation"
        Final.append(title)

    if scores[i] <= threshold :
        title = "not_recommendation"
        Final.append(title)
```

این رویکرد امکان می‌دهد تا کاربران فیلم‌هایی را که بیشترین تطابق را با علایق شخصی‌شان دارند به راحتی شناسایی کنند و در عین حال، از پیشنهادهای کمتر مرتبط دوری کنند .

۳-۱-۱۲ مدل SVM :

```
X = encoded_features.drop(["final_rating" , "Final opinion"] , axis = 1)
y = encoded_features["Final opinion"]
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.3, random_state = 42)
model = SVC(kernel = 'linear' )
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
report = classification_report(y_test, predictions)
print(accuracy, report)
```

این بخش از کد مراحل آماده‌سازی داده‌ها، آموزش یک مدل دسته‌بندی با استفاده از ماشین بردار پشتیبان (SVM) با هسته خطی، و سپس ارزیابی عملکرد این مدل را توصیف می‌کند. برای این منظور، از کتابخانه scikit-learn استفاده شده است.

- **آموزش مدل** :یک مدل دسته‌بندی با استفاده از ماشین بردار پشتیبان (SVM) با هسته خطی (kernel='linear') آموزش داده می‌شود. این مدل با استفاده از داده‌های آموزشی آموزش می‌بیند.
- **پیش‌بینی و ارزیابی مدل** :پس از آموزش مدل، برای ارزیابی عملکرد آن، پیش‌بینی‌هایی بر روی داده‌های آزمایشی انجام می‌شود. سپس، دقت کلی مدل با استفاده از accuracy_score ارزیابی می‌شود، و گزارش دسته‌بندی شامل معیارهای دیگری مانند دقت، بازخوانی، و امتیاز F1 (f1-score) برای هر کلاس به دست می‌آید.

این فرآیند به درک بهتری از توانایی مدل در پیش‌بینی نظرات کاربران بر اساس ویژگی‌های فیلم‌ها کمک می‌کند و می‌تواند برای بهبود توصیه‌های فیلم به کاربران استفاده شود .

۳-۲ نتایج بدست آمده

جدول ۱ : مشخصات عملکرد مدل

Metric	Precision	Recall	F1-Score	Support
not_recommendation	1.00	0.99	1.00	1018
recommendation	0.98	1.00	0.99	422
accuracy	0.94	-	-	1440
macro avg	0.99	1.00	0.99	1440
weighted avg	0.99	0.99	0.99	1440

نتایج ارائه شده از یک گزارش طبقه‌بندی هستند که عملکرد یک مدل طبقه‌بندی در تشخیص دو کلاس **not_recommendation** و **recommendation** را نشان می‌دهد. در اینجا، هر معیار دقت، بازیابی، f1-score و پشتیبانی برای هر کلاس به همراه میانگین‌های کلی گزارش شده‌اند. اجازه دهید هر کدام از این معیارها و نتایج کلی را توضیح دهیم:

۳-۲-۱ معیارها

- دقت : (Precision) نسبت تعداد نمونه‌های درست مثبت به کل نمونه‌هایی که به عنوان مثبت پیش‌بینی شده‌اند (درست مثبت + غلط مثبت). دقت برای **not_recommendation 1.00** و برای **recommendation 0.98** است، که نشان‌دهنده دقت بسیار بالای مدل در تشخیص هر دو کلاس است.
- بازیابی : (Recall) نسبت تعداد نمونه‌های درست مثبت به کل نمونه‌های واقعاً مثبت (درست مثبت + غلط منفی). بازیابی برای **not_recommendation 0.99** و برای **recommendation 1.00** است، نشان دهنده توانایی بالای مدل در بازیابی اکثر نمونه‌های واقعی هر دو کلاس است.
- (f1-Score) : میانگین هارمونیک دقت و بازیابی، که تعادل بین دقت و بازیابی را فراهم می‌کند. f1-scre برای **not_recommendation 1.00**و برای **recommendation 0.99** است، که نشان دهنده عملکرد فوق‌العاده مدل در هر دو جنبه است.
- پشتیبانی : (Support) تعداد نمونه‌های واقعی در هر کلاس، که برای **not_recommendation** 1018 و برای recommendation 422 است. این مقدار نشان دهنده توزیع نمونه‌ها در داده‌های آزمون است.

البته این اعداد نسبت به کل داده های test است .

۳-۲-۲ نتایج کلی

- دقت کلی(Accuracy) : نسبت تعداد تمام پیش‌بینی‌های صحیح به کل نمونه‌ها، که ۰.۹۹ (یا ۹۹.۴۴٪) است. این مقدار بیانگر عملکرد کلی بسیار بالای مدل در تمامی نمونه‌ها است.
- میانگین کلی (Macro Avg) و(Weighted Avg) :
- **Macro Avg** میانگین ساده‌ی معیارهای دقت، بازیابی و f1-score برای هر دو کلاس، که تأثیر مساوی برای هر کلاس دارد، صرف‌نظر از تعداد نمونه‌های آن‌ها. میانگین ماکرو برای دقت، بازیابی و f1-score حدود ۰.۹۹ است.
- **Weighted Avg** میانگین وزن‌دار معیارها بر اساس تعداد نمونه‌ها در هر کلاس، که نشان دهنده عملکرد مدل با توجه به توزیع نمونه‌های واقعی است. میانگین وزنی برای دقت، بازیابی f1-score نیز حدود ۰.۹۹ است.

۳-۳ نتیجه‌گیری

این نتایج نشان دهنده عملکرد بالای مدل در تشخیص دقیق دو کلاس not_recommendation و recommendation است، با دقت بسیار بالا، بازیابی عالی و f1-score برجسته ب. توزیع نمونه‌ها و عملکرد مدل در هر دو کلاس به خوبی مدیریت شده، که منجر به نتایج کلی بسیار مطلوب شده است.

۴ - بخش چهارم :

۴-۱ نتیجه گیری کلی :

این پروژه به بررسی و ارائه یک سیستم توصیه‌گر مبتنی بر محتوا می‌پردازد که با استفاده از الگوریتم‌های پیشرفته یادگیری ماشین و پردازش زبان طبیعی، قادر به پیش‌بینی محبوبیت فیلم‌ها و شناسایی دقیق گروه‌هدف آنها است. این سیستم با تحلیل ویژگی‌های فیلم‌ها نظیر ژانر، بازیگران، کارگردان، و دیگر عناصر محتوایی، پیشنهادات شخصی‌سازی‌شده‌ای را به کاربران ارائه می‌دهد که بر اساس علایق فردی آنها شکل گرفته است. استفاده از داده‌های امتیازدهی و رفتاری کاربران به عنوان بخشی از فرایند یادگیری ماشین، این سیستم را قادر می‌سازد تا با دقت بالایی نسبت به پیش‌بینی تمایلات کاربران و ارائه پیشنهادات مرتبط عمل کند. این تحقیق نشان می‌دهد که چگونگی کاربرد فناوری‌های نوین در بهبود تجربه کاربری و کمک به تولیدکنندگان فیلم برای دستیابی به موفقیت اقتصادی بیشتر از طریق درک بهتر و جذب مؤثرتر مخاطبان هدف.

تشکر از توجه شما .

۴۰۲۲۱۵۵۰۰۰۴