به نام خدا

هوش مصنوعی **پروژه ششم**

علی دارابی – شماره دانشجویی : ۸۱۰۱۰۰۲۶۴

عنوان پروژه : Clustering

آشنایی با مجموعه داده

۱ – در صورتی که داده ها نامتوازن بودند، چه مشکلاتی در فرآیند خوشه بندی پیش می آمد؟ چه راهکاری را برای برطرف کردن این مشکل ارائه می دهید؟ توضیح دهید.

اگر داده ها نامتوازن بودند، یعنی تعداد سمپل ها در هر کلاس به طور قابل توجهی متفاوت بود، چندین مشکل ایجاد میشد. برای مثال : مدل به کلاس های خاصی گرایش پیدا می‌کرد و کلاس‌هایی که داده بیشتری دارند را بهتر یاد میگرفت. درنتیجه دقت پیشبینی ها برای کلاس هایی که سمپل های کمتری داشتند، پایین تر میشد. در واقع مدل احتمالا روی داده های train کلاس هایی با سمپل بیشتر overfit میشد، و توانایی تعمیم پذیری آن کاهش پیدا می‌کرد.

راهکار برای حل این مشکل : resampling داده ها به نحوی که تعداد نمونه در هر کلاس برابر شود. البته باید مراقب باشیم که اطلاعات مفید از دست نرود. استفاده از معیارهای ارزیابی مثل F1-score به جای دقت کلی تا کلاس های کم نمونه تر هم در نظر گرفته شوند. استفاده از الگوریتم‌هایی مثل oversampling یا undersampling برای متوازن سازی داده ها. اعمال وزن های متفاوت برای نمونه های متفاوت در تابع loss مدل (وزن بیشتر به نمونه های کلاس با نمونه کمتر).

تاثیر انواع دیگر پیش پردازش ها :

میتوانیم در لیست stop words علامت ها (مانند نقطه ، ویرگول) و اعداد فارسی اضافه کنیم تا آنها را حذف کنیم که اینکار را در کد انجام دادیم و نتیجه بهتری گرفتیم. جایگزینی اعداد و مقادیر عددی با یک توکن مشخص که تاثیر خوبی نداشت.

۲ – در گزارش کار خود، جایگزین کردن کلمات با روش stemming یا lemmatization را توضیح دهید.

Stemming یک روش ساده است که با اعمال قوانینی، پسوند کلمات را حذف می‌کند تا به ریشه آنها برسد. البته ممکن است گاهی نتیجه صحیح نباشد.

Lemmatization پیچیده تر است و با تجزیه ساختاری کلمه، lemma یا ریشه دقیق آن را به دست می‌آورد. این روش نیاز به دانش زبان شناسی دارد تا بتواند ریشه صحیح را تشخیص دهد.

برای مثال فرض میکنیم کلمات "walking", "walked", "walks" را داریم:

Stemming: با حذف پسوندهای "ing", "ed", "s" این کلمات را به "walk" تبدیل می‌کند.

Lemmatization: این کلمات را به "walk" تبدیل می‌کند، چون "walk" ریشه یا lemma این کلمات است.

به طور دقیق تر : مثلا در مورد کلمه "walking" ، lemmatizer می‌فهمد که این فرم حال جاری از فعل "walk" است و بنابراین آن را به ریشه اصلیش که "walk" است تبدیل می‌کند. اما stemmer صرفا پسوند "ing" را حذف می‌کند و به "walk" می‌رسد. درنتیجه lemmatizer با درک بهتری از ساختار کلمات، ریشه دقیق تری را برمی‌گرداند.

مزیت lemmatization این است که شکل استاندارد کلمات را به دست می‌دهد اما stemming سریع تر است و نیاز به منابع زبان شناسی کمتری دارد. در نتیجه وقتی دقت بیشتری نیاز است lemmatization ترجیح داده می‌شود. اما stemming می‌تواند در مواردی که سرعت مهم تر است مفید باشد.

---

۳ – دلیل استفاده از بردار ویژگی و ویژگی های آن را در گزارش توضیح دهید.

بردارهای ویژگی به ما اجازه نمایش عددی نمونه های داده ای که شامل ویژگی‌ها هستند را می‌دهند. بسیاری از الگوریتم های یادگیری ماشین نیاز به داده‌های عددی دارند. بردارهای ویژگی داده هایی مانند داده های متنی و تصویری و ... را به اعداد تبدیل میکنند. آنها به ما امکان استاندارد کردن انواع مختلف داده ها و ویژگی ها تحت یک نمایش برداری مشترک را می‌دهند. این امکان مقایسه بین نمونه‌های داده را فراهم می‌کند. هنگامی که داده‌ها به صورت بردارهای ویژگی نمایش داده می‌شوند، فاصله بین آنها به راحتی با معیارهای فاصله‌ای مانند فاصله اقلیدسی قابل محاسبه است. این فواصل برای مقایسه تشابه ها به کار می‌روند. مدل‌های یادگیری ماشین می‌توانند الگوهای مهمی را در فضای برداری ویژگی‌ها تشخیص دهند که برای طبقه‌بندی داده‌ها، پیش‌بینی نتایج و خوشه بندی داده‌ها به کار می‌رود. نزدیکی مکانی در این فضا نشان دهنده تشابه معنایی است. داده‌های پرابعاد به شکل برداری فشرده با طول ثابت درمی‌آیند. این باعث بهینگی حافظه و فضا می‌شود.

---

۴ – در مورد نحوه استفاده از word2vec و doc2vec و تبدیل متن به بردار ویژگی توضیح دهید.

Word2vec روشی برای تبدیل کردن کلمات به بردارهای اعدادی است. این بردارها به گونه ای ساخته می‌شوند که کلمات مشابه از نظر معنایی، بردارهای مشابهی داشته باشند. مثلاً بردار برای کلمه "گربه" به بردار کلمه "سگ" شباهت زیادی دارد. زیرا هر دو حیوان هستند. اما بردار کلمه "گربه" با بردار کلمه "میز" تفاوت زیادی دارد. با استفاده از این بردارهای کلمه، می‌توان به الگوریتم‌های یادگیری ماشین کمک کرد تا معانی و روابط بین کلمات را درک کنند. Doc2Vec هم همین کار را برای جملات و پاراگراف‌ها انجام می‌دهد. برداری برای کل متن ایجاد می‌کند تا بتوان متون را بر اساس محتوایشان مقایسه کرد. در مجموع، این دو تکنیک باعث می‌شوند مدل ما بتواند معنا و محتوای متون را تا حدی درک کند.

---

۵ – در مورد روش های K-means و DBSCAN و مزایا و معایب این روش ها نسبت به هم توضیح دهید.

K-Means: داده‌ها را با اختصاص هر نقطه داده به یکی از k خوشه تعریف شده توسط مرکز خوشه ها خوشه‌بندی میکند. مزایا: ساده و سریع است. برای خوشه‌های کروی شکل عملکرد خوبی دارد. همچنین می‌تواند برای داده‌های بسیار بزرگ هم مثل داده هایی با میلیون ها نقطه، به خوبی خوشه بندی انجام دهد. یعنی با افزایش حجم داده‌ها، سرعت کم نمیشود.

معایب: نیاز به مشخص کردن تعداد خوشه ها k از ابتدا دارد. برای خوشه‌های غیرکروی مناسب نیست. به نقاط پرت و دور افتاده حساس است. یعنی اگر ویژگی های مختلف دامنه های متفاوتی داشته باشند، باید قبل از اعمالK-Mean ، آن ها را استاندارد کنیم تا مقیاس شان تقریبا یکسان شود.

DBSCAN: داده‌ها را بر اساس تراکم با برچسب زنی نواحی پرتراکم به عنوان خوشه و نواحی کم تراکم به عنوان نویز خوشه بندی می‌کند.

مزایا: نیاز به مشخص کردن تعداد خوشه ها ندارد. نقاط نویز و پرت را شناسایی می‌کند. با خوشه‌های شکل دلخواه به خوبی کار می‌کند. در برابر نقاط پرت مقاوم است.

معایب: در خوشه‌های با تراکم متفاوت مشکل دارد. DBSCAN وقتی حجم داده ها خیلی زیاد میشود، از نظر محاسباتی سنگین و کند می‌شود به دلیل اینکه برای هر نقطه باید فاصله آن را با تمام نقاط دیگر محاسبه کنیم تا ببینیم آیا در یک خوشه قرار می‌گیرد یا خیر. به پارامترهای ورودی حساس است.

به طور خلاصه میتوان گفت که K-Means ساده‌تر و سریع‌تر است اما DBSCAN بهتر با شکل‌های نامنظم خوشه کنار می‌آید. KMeans برای داده‌های بزرگ مقیاس‌پذیری بهتری نسبت به DBSCAN دارد. DBSCAN در برابر داده های پرت مقاوم تر است.

۶ – خروجی حاصل از دو نوع خوشه بندی را با هم مقایسه کنید.

با توجه به نتایج بدست آمده در کد، الگوریتم Kmeans ، Homogeneity Score بالاتری نسبت به DBSCAN دارد یعنی خوشه بندی آن بهتر با برچسب های واقعی داده ها مطابقت دارد. همچنین الگوریتم Kmeans ، Silhouette Score بالاتری دارد، بنابراین خوشه‌بندی آن از لحاظ تفکیک‌پذیری درون و بین خوشه ای بهتر است. درنتیجه به نظر میرسد الگوریتم Kmeans برای این مجموعه داده خاص، خوشه بندی بهتری از لحاظ همگنی با برچسب های واقعی انجام داده است و همچنین خوشه های فشرده تر و مجزا تری تولید کرده است.

---

۷ – درباره PCA تحقیق کنید و نحوه عملکرد آن را به اختصار توضیح دهید.

PCA یک روش برای کاهش ابعاد داده‌ها است. فرض کنید داده های ما شامل ۲۰ مشخصه یا ستون است. ما می‌خواهیم این ۲۰ مشخصه را به مثلا ۴ مشخصه کلیدی کاهش دهیم. PCA این کار را با این مراحل انجام می‌دهد:

۱ – داده‌ها را استانداردسازی می‌کند. (میانگین صفر و انحراف معیار یک)

۲ – رابطه بین مشخصه ها را بررسی می‌کند و مشخصه‌هایی که با هم همبستگی بالایی دارند را پیدا میکند.

۳ – مشخصه هایی که با هم همبستگی بالایی دارند را در یک مولفه اصلی جدید خلاصه میکند.

۴ – ۴ مؤلفه اصلی را که بیشترین اطلاعات داده ها را نگه میدارند، انتخاب میکند.

۵ – داده‌ها را بر اساس این ۴ مولفه اصلی جدید نمایش می‌دهد.

بنابراین PCA با خلاصه کردن اطلاعات در مؤلفه‌های اصلی، ابعاد داده را کاهش می‌دهد در حالی که حداکثر اطلاعات مفید را نگه میدارد.

---

۸ – در مورد نحوه محاسبه معیار silhouette و homogeneity توضیح دهید.

Silhouette Score: میزان شباهت هر نقطه داده به خوشه خودش را با خوشه های دیگر مقایسه می‌کند. مقادیر آن بین منفی ۱ تا ۱ هستند و مقدار بالاتر بهتر است. نحوه محاسبه آن :

$$Silhoutte = \frac{b - a}{\max(a, b)}$$

a: میانگین فاصله نقطه تا تمام نقاط دیگر در خوشه خودش است.

b: کمترین فاصله میانگین نقطه تا نقاط خوشه دیگر است.

در نهایت میانگین s برای تمام نقاط به دست می‌آید.

Homogeneity Score: درصد نمونه هایی که درست به برچسب خوشه واقعی‌شان اختصاص یافته اند را می سنجد، مقادیر آن بین ۰ تا ۱ است. مقدار بالاتر بهتر است. نحوه محاسبه آن :

اگر بخواهیم به عدم قطعیت در برچسب های واقعی کلاس های داده ها توجه کنیم، فرمول بدین شکل است :

$$Homogeneity = 1 - \frac{H(Y_{true}|Y_{pred})}{H(Y_{true})}$$

در این فرمول سهم هر داده را بر اساس احتمال تعلق آن به کلاس صحیح وزن دهی میکند. این بدان معناست که داده هایی که احتمال بیشتری دارد درست طبقه بندی شده باشند، تاثیر بیشتری برنتیجه خواهند داشت تا داده هایی که احتمال کمتری دارد درست طبقه بندی شده باشند.

ولی باتوجه به اینکه در این مسئله ما راجب به برچسب های واقعی کلاس های داده ابهامی نداریم، فرض میکنیم داده های ما N تا نمونه دارد و K تا خوشه داریم. برچسب های واقعی را با C = {c1, c2, ..., cn} نشان میدهیم و برچسب های پیش‌بینی شده توسط مدل را با K = {k1, k2, ..., kn} نشان میدهیم. حال یک متغیر شمارنده تعریف می‌کنیم (correct) سپس برای هر نمونه i از ۱ تا N اگر ci == ki بود، به correct یکی اضافه میکنیم. در نهایت بدین شکل خروجی را محاسبه میکنیم :

$$Homogeneity = \frac{correct}{N}$$

۹ – نتایج حاصل از معیار های ذکر شده را برای هر یک روش ها گزارش کنید.

در Notebook انجام شده است.

---

۱۰ – راهکار هایی پیشنهاد کنید که بتوان عملکرد مدل ها بهبود داد.

۱ – امتحان روش های مختلف برداری کردن: به جای Bag-of-Words ساده، از بردارهای TF-IDF یا Word2Vec استفاده کنیم. با این روش ها ممکن است معنای متن را بهتر بگیریم.

۲ – بهینه سازی پارامترهای Doc2Vec: اندازه بردار، پنجره کلمات، حداقل تعداد کلمات و غیره را تنظیم کنیم. برای داده های بزرگتر بردارها و پنجره های بزرگتر نیاز است.

۳ – کاهش ابعاد قبل از خوشه بندی: از PCA یا UMAP برای کاهش ابعاد بردارها به ۲ یا ۳ بعد استفاده کنیم. ابعاد بالا الگوریتم‌های خوشه‌بندی را تحت تأثیر قرار می‌دهد.

۴ – ترکیب مدل‌های خوشه‌بندی: چند الگوریتم را اجرا کنیم و خوشه های هر متن را مقایسه کنیم. برچسب نهایی می‌تواند بیشترین رای باشد.

۵ – بهینه‌سازی کامل پارامترهای تراکم برای DBSCAN: این الگوریتم بسیار حساس است اما اشکال دلخواه را خوب پیدا می‌کند.
