دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

درس تحلیل کلان داده ها

استاد درس جناب آقای دکتر چهرقانی

(گزارش پروژه پایانی)

محسن عبادپور | ۴۰۰۱۳۱۰۸۰ | m.ebadpour@aut.ac.ir

نیمسال دوم سال تحصیلی ۱۴۰۲-۱۴۰۱

فهرست مطالب

بخش اول: مقدمه و معرفی

طبق خواسته پروژه، در این بخش به صورت خلاصه موضوع مورد بحث در مقاله[1] و راهکار ارائه شده بررسی شده و سپس مزایا و معایب احتمالی آن مورد اشاره قرار میگیرد.

الف) موضوع مورد بحث

در بحث یادگیری ماشین و خصوصا یادگیری ماشین بدون نظارت(بدون برچسب) یکی از task های مهم و چالشی خوشه‌بندی داده ها می باشد که هدف آن قرار دادن نمونه‌های مشابه در یک خوشه می باشد بطوری که اعضای یک خوشه با اعضای خوشه‌ی دیگر مشابه نباشد. هدف این مقاله نیز حل مسئله خوشه‌بندی در شرایط چالشی تر می باشد که در بیان می شود.

همانطور که در کلاس و محتوای درس بحث شده است، خوشه‌بندی دارای چالش های متعددی می باشد مانند در نظر گرفتن/نگرفتن شکل یا توزیع برای خوشه‌ها، تعیین مراکز خوشه و مقداردهی اولیه‌ی آنها، شناسایی داده های پرت، بروزرسانی مراکز خوشه‌ها و... . یکی از چالش هایی که در خوشه‌بندی میتواند وجود داشته باشد و مسئله را سخت کند نامتعادل (imbalance) بودن تعداد اعضای خوشه های مفروض باشد چرا که باعث می شود الگوریتم های خوشه بند در معرض شناسایی آن خوشه‌ها به عنوان داده های پرت یا به عنوان یک خوشه دیگر قرار گرفته و عملکرد آن با خطا مواجه شود. دومین چالشی که مسئله خوشه بندی را سخت‌تر می کند، نامتعادل (imbalance) بودن توزیع و چگالی اعضای خوشه های متفاوت می باشد مثلا اعضای یک خوشه خیلی بهم نزدیک باشند و خوشه dense باشد اما اعضای خوشه

[1] An Adaptive Clustering Algorithm Based on Local-Density Peaks for Imbalanced Data Without Parameters.

دیگری از هم فاصله داشته و خوشه sparse باشد؛ در این حالت چالشی که وجود دارد تعیین مرز خوشه‌ها و توزیع آنها و همچنین شناسایی هر دو نوع خوشه بدرستی باشد. چالش دیگری که در الگوریتم های خوشه بندی وجود دارد، نیازمندی به تعیین برخی هایپر پارامترها نظیر تعداد خوشه از قبل می باشد که بر کیفیت خوشه بندی تاثیر مستقیم دارد.

مقاله‌ی مورد اشاره، خوشه بندی را هدف خود قرار داده به طوری که بتواند به هر سه مورد مذکور پاسخ دهد که مشتمل بر (۱) حل چالش نامتعادل بودن تعداد نمونه‌ها، (۲) حل چالش نامتعادل بودن توزیع و پراکندگی و (۳) عدم نیاز به هایپرپارامتر می باشد.

ب) راهکار ارائه شده و روند حل

روش و راه حل کلی مورد بحث در مقاله مبتنی بر استفاده از قله‌های محلی پراکندگی (Local-Density Peaks) می‌باشد که در آن خوشه بندی بر اساس نقاط و نمونه هایی که به صورت محلی از چگالی بالایی برخوردار بوده و به تعداد نمونه‌های بیشتری از یک حد آستانه(dc) نزدیک است انجام می پذیرد. در ادامه بصورت خلاصه گام‌ها و روند حل مسئله که در مقاله ارائه شده است، بیان می‌شود.

در شروع الگوریتم بایستی یکسری محاسبات به ازای هر نمونه انجام شود که شامل محاسبه چگالی (طبق فرمول شماره ۵ مقاله) و فاصله‌ی رو به بالا (upward distance: طبق فرمول شماره ۲ مقاله) می باشد که اولی معیاری است که نشان میدهد حول هر نمونه با حد آستانه dc نمونه های دیگر چگونه قرار گرفته اند و دومی نیز معیاری است که نشان میدهد هر نمونه در چه حداقل فاصله‌ای از نقاط چگال قرار گرفته است. لازم به ذکر است که حد آستانه‌ی dc توسط خود نمونه ها تعیین شده و طبق فرمول شماره ۳ مقاله مقدار میگیرد که برابر است با حداکثر فاصله ای نمونه‌ها با نزدیک ترین همسایه خود فاصله دارند. در کار های قبلی، با استفاده از این دو معیار یک گراف تصمیم دو بعدی رسم می شود و سپس بر اساس آن با در نظر گرفتن یک حد مناسب مراکز خوشه ها مشخص می شود.

مشکلی که این رویکرد دارد این است که در مقابله با خوشه هایی که sparse هستند با شکست مواجه شده و آنها را به عنوان نویز شناسایی می شود. برای حل این مشکل یک معیار(بعد) دیگر به گراف تصمیم با عنوان RNN اضافه می‌شود که نشان میدهد هر نمونه در چه تعداد از همسایه‌های نزدیک (nearest neighbor) سایر این نمونه ها واقع شده است.

در این صورت طبق تعریف شماره ۴ مقاله نقاطی که چگالی و RNN پایین و upward distance بالا داشته باشد میتواند نویز در نظر گرفته شود و همچنین نقاطی که upward distance و RNN بالا یا upward distance و چگالی بالایی داشته باشد مستعد مرکز بودن خوشه‌های احتمالی قرار می گیرد.(Determining the Noise)

حال که نویز ها شناسایی شده است و از محاسبات کنار گذاشته شده است، بایستی خوشه‌بندی آغاز شود. برای اجتناب از تعیین دستی تعداد خوشه‌ها، هر خوشه تعدادی زیر خوشه در نظر گرفته و زیر خوشه ها تولید می شود؛ نقاطی که upward distance آنها از مجموع انحراف معیار و میانگین upward distance بیشتر باشد به عنوان مراکز زیر خوشه‌ها در نظر گرفته می شود که میتواند هر عددی باشد. حال بایستی نقاط باقی مانده به زیر خوشه ها تخصیص یابد. اختصاص نقاط باقیمانده به این صورت است که نقاط باقیمانده بر اساس چگالی شان و نزولی مرتب می شود؛ اولین نقطه از لیست به نزدیکترین مرکز زیر خوشه‌ای که چگالی آن از خودش بزرگتر است اختصاص پیدا میکند و بطور مشابه دومین نقطه نیز به نزدیکترین نقطه‌ای که چگالی آن بیشتر از خودش است مرتبط شده و به خوشه‌اش تعلق پیدا می‌کند و این روند برای همه‌ی نقاط به ترتیب مذکور اجرا می شود. شبه کد توضیحات ارائه شده تا این قسمت در الگوریتم ۱ مقاله آورده شده است که پیاده سازی انجام شده نیز بر آن اساس بوده است.(Initial Sub-Clusters)

حال که زیر خوشه‌ها و اعضایشان محاسبه شده است، محتمل است که تعدادی زیر خوشه اشتباها شناسایی شده باشد و اعضای آن در اصل متعلق به سایر زیر خوشه ها باشد؛ لذا بایستی زیرخوشه ها را آپدیت نموده و در خصوص آنها تصمیم گیری کرد که آیا همچنان زیرخوشه بمانند یا زیر خوشه حذف شده و اعضایش همانند روش قبل به سایر زیر خوشه ها اختصاص یابد. (Sub-Cluster Updating) برای این منظور و برای هر زیر خوشه یک شرط بررسی می شود که آیا تعداد اعضای زیر خوشه از نصف تعداد همسایه های مرکز زیر خوشه با شعاع dc کمتر است یا خیر. اگر کمتر باشد یعنی قسمت عمده‌ی اعضای این زیر خوشه به زیر خوشه های دیگر جذب شده‌اند و در نتیجه این زیر خوشه میتواند حذف شده و اعضایش به سایر خوشه ها انتقال یابد. شبه کد توضیحات ارائه شده تا این قسمت در الگوریتم ۲ مقاله آورده شده است که پیاده سازی انجام شده نیز بر آن اساس بوده است.

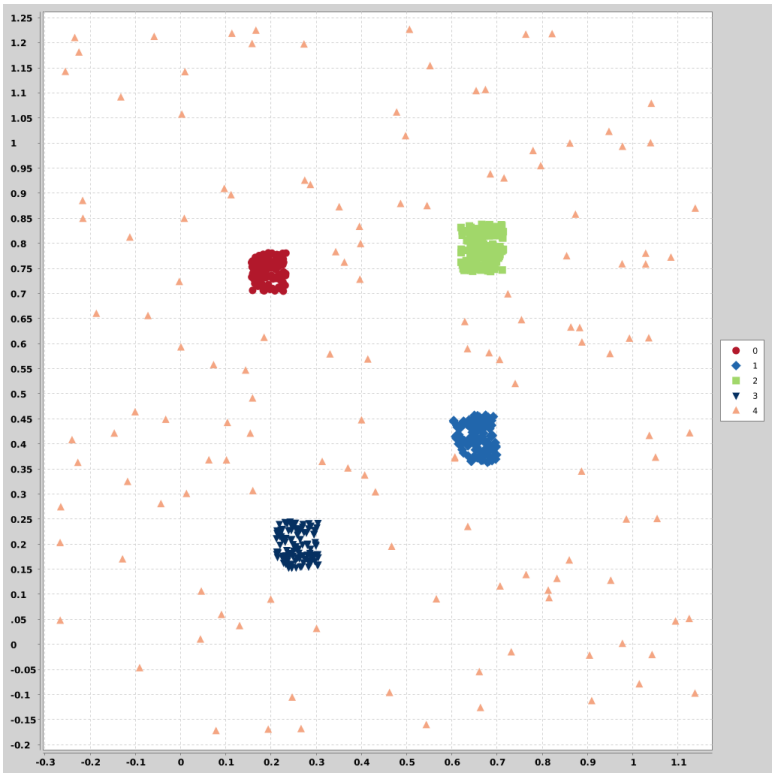
حال بایستی زیر خوشه‌ها را باهم ادغام کرده و خوشه‌های مورد نظر را نهایی را بدست آورد. برای این منظور نقاط مرزی در نظر گرفته می‌شوند به طوری که چگالی آنها از میانگین چگالی در خوشه‌ی متناظر کمتر است؛ همچنین زیر مجموعه ای از اعضای یک زیر خوشه که شامل همه‌ی نمونه ها جز نمونه های مرزی است را In آن زیر خوشه در نظر میگیریم.

همچنین بایستی یک شعاع ادغام(r) در نظر گرفته شود که براساس آن بتوان تصمیم گرفت که آیا ادغام صورت گیرد یا خیر. در صورتی که در مرحله ابتدایی، نمونه های نویزی شناسایی نشده باشد، شعاع ادغام برابر با dc تعیین می شود و در غیر این صورت همانند رابطه‌ی محاسبه‌ی dc اقدام می‌کند با این تفاوت که نقاط نویز را در محاسبه نظر نمی‌گیرد. حال فاصله‌ی دو خوشه‌ی n و m بصورت d(m,n) تعریف می شود که برابر است با کمینه فاصله ای که یک عضو مانند x از m با یک عضو مانند y از n میتواند داشته باشد. بر اساس تعریف و مفروضات فوق، اگر فاصله d(m,n) از r بزرگتر باشد، یعنی دو زیر خوشه باهم فاصله قابل توجهی داشته و نباید باهم ادغام کرد. در غیر این صورت که d(m,n) کوچکتر– مساوی r باشد در دو حالت ادغام را انجام میدهیم و در صورت نقص هر دو ادغام صورت نمیگیرد: (۱) اگر x و y جزو In خوشه هایشان باشند یعنی نقاط میانی خوشه‌ها به مقدار قابل توجهی بهم نزدیک اند پس میتوانیم آنها را ادغام کنیم. (۲) اگر مجموع چگالی نقاط x و y از میانگین چگالی نقاط مراکز زیرخوشه های n و m بیشتر باشد یعنی در مرز دو خوشه چگالی بالاست پس میتوانیم آنها را ادغام کنیم. فرآیند بررسی مذکور را به ازای تمامی جفت خوشه‌ها انجام و در خصوص ادغام یا عدم ادغام آنها تصمیم میگیریم و در انتها عملیات ادغام به خوشه های واحد را برای هر جفت ادغام شده‌ی ممکن انجام میدهیم. در این وضعیت فرآیند خوشه‌بندی به اتمام می‌رسد و تنها کافی است نقاط نویز را نیز خوشه بندی کنیم که هر یک را به نزدیک ترین خوشه اختصاص می دهیم.

ج) مزایا و معایب احتمالی مقاله

از وجه های مختلفی میتوان مقاله را بررسی و برای آن نقاط مثبت و منفی بررسی کرد. به عنوان مزیت اساسی میتوان مدیریت و خوشه‌بندی صحیح داده های نامتعادل(چه از جهت تعداد و چه از جهت چگالی و تراکم) بیان کرد. مزیت بعدی را میتوان عدم نیاز به تنظیم هایپرپارامتر نام برد. نقطه مثبت دیگر این است که برای خوشه‌ها و اعضایشان شکل(محدب/مقعر) یا توزیع خاص(نرمال/...) و ثابت پیشفرض در نظر نمیگیرد فارغ از اینکه بتواند درست خوشه بندی کند یا خیر.

اگر مسئله‌ی خوشه‌بندی مورد نظر ما در مقیاس Big Data و Data Stream تعریف شود الگوریتم معرفی شده در مقاله از کارآیی مناسب برخوردار نبوده و با ضعف مواجه می شود چرا که مرتبه زمانی آن $O(n^2)$ می باشد که خارج از منابع و هدف تعیین شده می باشد. نقطه ضعف بعدی این است که نقاط نویزی را در انتها به یک خوشه تخصیص می دهد که محتمل است اشتباه باشد چرا که ممکن است نویز ها همگی در فضای ویژگی یکنواخت پخش شده باشد و نمونه‌های نویز حاصل از یک خوشه نبوده و به هیچ خوشه ای تعلق نداشته باشد؛ به عبارتی دیگر خوشه‌ی نویزی نمیتواند ایجاد و تعیین کند(همانند شکل زیر که نخواهد توانست به درستی خوشه بندی کند). مورد بعدی که میتواند چالشی باشد اعمال الگوریتم در فضاهای غیر اقلیدسی و برداری می باشد که غیرساختاری می باشد. (مجموعه داده zelnik4)



بخش دوم: پیاده‌سازی

در این بخش به صورت خلاصه روند توابع پیاده‌سازی انجام شده طبق توضیحات بخش قبل آورده شده است.

الف) بارگذاری مجموعه داده

بارگذاری مجموعه داده و انجام پیش پردازش‌ها و همچنین تبدیل آن به فرمت X و Y در تابع پیاده سازی شده‌ی load_dataset انجام می پذیرد که نام مجموعه داده را دریافت و آن را از پوشه‌ی دیتاست میخواند. کل مجموعه داده‌های مورد استفاده در این پیاده سازی توسط این تابع لود و مورد استفاده قرار گرفته است. تنها پیش پردازش اعمالی نرمال سازی می باشد تا حساسیت نسبت به رنج فاصله ها از بین رود چرا که ما با فاصله سنجی و محاسبه distance مواجه هستم.

ب) پیاده سازی الگوریتم

برای پیاده سازی الگوریم مورد اشاره در مقاله توابع متعددی بر اساس منطق الگوریتم و گام های آن پیاده سازی شده است در ادامه بطور خلاصه مورد اشاره قرار میگیرد.

تابع distance_measure برای سنجش فاصله بین دو نمونه پیاده سازی شده است که دو نمونه را ورودی و فاصله‌ی آن را بر اساس اقلیدسی محاسبه و بر میگرداند. هدف از پیاده سازی این تابع آن بوده است که در صورت استفاده از معیار فاصله‌ای جز اقلیدسی در زمانی دیگر بتوان آن را اینجا پیاده سازی نمود و در کل الگوریتم آن را اعمال کرد.

تابع calculate_dc برای محاسبه‌ی حد آستانه‌ی dc مورد استفاده در الگوریتم پیاده سازی شده است که مجموعه را دریافت و معیار dc را به همراه ماتریس فاصله بر میگرداند. در ادامه و برای ساده سازی حجم پردازش‌ها و به جای محاسبه فاصله‌ی نمونه ها این ماتریس ذخیره و نگهداری می شود.

تابع calculate_density برای محاسبه‌ی چگالی هر یک از نمونه‌ها پیاده سازی شده است که ورودی آن dc و ماتریس فاصله‌ها بوده و خروجی آن یک برداری به اندازه مجموعه داده است که هر عنصر چگالی متناظر با آن نمونه را نشان می‌دهد.

تابع calculate_upward_distance تابعی است که معیار upward distance را به ازای نمونه ها محاسبه و برمیگرداند. ورودی تابع بردار چگالی نمونه ها و ماتریس فاصله بوده و خروجی آن یک برداری به اندازه مجموعه داده است که هر عنصر upward distnace متناظر با آن نمونه را نشان می‌دهد.

تابع calculate_neighbors تابعی است که نزدیکترین همسایگی نمونه ها را محاسبه و نتیجه را برمیگرداند. ماتریس فاصله به همراه تعداد همسایه‌ی مورد نظر ورودی تابع بوده و خروجی نیز دو لیست به اندازه مجموعه داده است. لیست اول همسایه‌های هر نمونه‌ی متناظر را نشان میدهد و لیست دوم نیز همسایه‌های معکوس هر نمونه‌ی متناظر را نشان می دهد به این معنا که عنصر i در نزدیکترین همسایگی های چه نمونه هایی ظاهر شده است.

تابع determine_noise_and_init تابعی است که نقاط نویزی مشخص و سپس زیر خوشه‌های اولیه را ایجاد و نمونه‌های باقی مانده را به آنها تخصیص می دهد. ورودی تابع مقادیر upward distance، مقادیر چگالی، لیست همسایگی‌های معکوس و ماتریس فاصله است و خروجی نیز بردار برچسب خوشه‌ها برای مجموعه داده، اندیس مراکز زیرخوشه‌ها و تعداد زیر خوشه‌ها می باشد(تعداد نقاط نویزی را نیز بخاطر گزارش گیری بر میگرداند). در پیاده سازی این تابع، ابتدا میانگین و انحراف معیار چگالی و upward distance محاسبه می شود و سپس تعداد همسایگی معکوس برای هر نمونه حساب شده و میانگین و انحراف معیارش نیز بدست می آید. سپس بر اساس الگوریتم بیان شده در بخش قبل، نقاط نویزی تعیین شده و سپس مراکز زیر خوشه‌ها مشخص می شود و در ادامه سایر نقاط به خوشه ها اختصاص پیدا می‌میکند.

تابع update_clusters تابعی است که زیر خوشه‌ها را بروز کرده و آنهایی که به اشتباه به عنوان یک زیر خوشه تعیین شده اند را حذف و اعضایش را به سایر خوشه ها تخصیص می دهد(false positive sub-cluster). ورودی این تابع مقادیر upward distance، مقادیر چگالی، ماتریس فاصله، اندیس مراکز زیر خوشه‌ها، حد آستانه dc و تعداد خوشه‌ها

می باشد. خروجی آن نیز بردار بروز شده‌ی برچسب خوشه بندی برای مجموعه داده، تعداد خوشه‌ها کنونی و اندیس مراکز زیر خوشه های حذف می باشد. روند بروزرسانی زیرخوشه‌ها و تصمیم گیری بر اساس شرط بیان شده در بخش توضیحات می باشد که بایستی تعداد اعضای زیر خوشه از نصف تعداد همسایه های مرکز زیر خوشه با شعاع dc بزرگتر باشد.

تابع clusters_distance تابع کمکی است که در بخش ادغام زیرخوشه ها مورد استفاده قرار میگیرد و فاصله دو خوشه را نشان میدهد$d(m,n)$. ورودی تابع ماتریس فاصله به همراه اعضای دو خوشه بوده و خروجی نیز فاصله محاسبه شده به همراه اندیس عناصر از دو خوشه می باشد.(به عبارتی دیگر argmin)

تابع get_In_set تابعی است که اعضای In یک خوشه را محاسبه میکند و ورودی آن مقادیر چگالی ها و اعضای خوشه است و خروجی آن نیز مجموعه اعضای خوشه است که مرزی نبوده و درونی می باشد. این تابع کمکی است که در بخش ادغام زیرخوشه ها مورد استفاده قرار میگیرد.

تابع update_merge_labels دیکشنری خوشه هایی که بایستی باهم ادغام شوند را بروز می کند. دیکشنری مورد نظر بدین صورت است که نشان میدهد هر خوشه بایستی با چه خوشه های دیگری بصورت مستقیم ادغام شوند.(به عبارتی دیکشنری است که کلید ها شماره خوشه و مقادیر لیست های ادغامی با آن خوشه می باشد). ورودی این تابع دیکشنری ادغام به همراه شماره دو خوشه ای است بایستی ادغام شوند و خروجی نیز دیکشنری بروز شده می باشد. این تابع کمکی است که در بخش ادغام زیرخوشه ها مورد استفاده قرار میگیرد.

تابع merging_clusters تابعی است که ادغام خوشه ها را بر اساس الگوریتم بیان شده انجام می دهد. ورودی تابع حد آستانه dc، مقادیر چگالی، ماتریس فاصله، بردار برچسب خوشه‌ها برای مجموعه داده و مراکز زیرخوشه ها اولیه می باشد. در ابتدای این تابع ابتدا شعاع ادغام (r) محاسبه می شود و سپس به ازای هر دو نمونه‌ی ممکن طبق الگوریتم اعضای In محاسبه و فاصله‌ی بین آن دو خوشه سنجیده میشود و شرط های الگوریتم بررسی و در صورت صحیح بودن، ادغام دو خوشه توسط تابع update_merge_labels ذخیره می شود. در نهایت توسط دیکشنری ادغام، ادغام خوشه‌ها انجام و خوشه‌های نهایی حاصل می شود و نقاط نویزی نیز توسط در این زمان در خوشه های نزدیک خود قرار میگیرند. خروجی این تابع بردار برچسب خوشه‌ها برای مجموعه داده و لیست ادغام شده‌ی زیر خوشه‌ها می باشد.

تابع prediction_report معیار های ارزیابی مورد نیاز نظیر محاسبه و نتیجه را بر میگرداند. ممکن است شماره خوشه‌ی بدست آمده با شماره برچسب ابتدایی تفاوت داشته باشد، لذا در این تابع بررسی می شود که مد هر خوشه برچسب متناظر را دریافت نماید.

ج) معیارهای ارزیابی

معیار های مورد استفاده در این مقاله مجموعا پنج مورد می باشد. اولین معیار accuracy می باشد که نشان میدهد خوشه بندی با چه دقتی انجام شده است و چه نسبتی از مجموعه داده به درستی باهم خوشه بندی شده اند. این معیار میتواند بین ۰ الی ۱ تغییر پیدا کند چرا که نسبت خوشه بندی صحیح به مجموع کل نمونه ها را نشان میدهد و همچنین هر چقدر بیشتر باشد یعنی تعداد نمونه های بیشتری درست خوشه بندی شده اند و این بهتر است. برای محاسبه‌ی این معیار از تابع آماده‌ی accuracy_score از کتابخانه‌ی sklearn استفاده شده است.

دومین معیار recall است که نشان میدهد چه نسبتی از نمونه‌هایی که مثبت خوشه بندی شده اند واقعا مثبت بوده اند. این معیار برای خوشه بندی باینری و دو خوشه ای معنادارتر است و برای چند خوشه ای بصورت macro مثبت/منفی در نظر گرفته و محاسبه می‌شود. این معیار میتواند بین ۰ الی ۱ تغییر پیدا کند چرا که نسبت خوشه بندی های واقعا مثبت را به مجموع کل خوشه بند های مثبت نشان میدهد و همچنین هر چقدر بیشتر باشد خوشه بند در مواجه با نمونه های مثبت مناسب و دقیق عمل کرده و این بهتر است. برای محاسبه‌ی این معیار از تابع آماده‌ی accuracy_score از کتابخانه‌ی sklearn استفاده شده است.

سومین معیار ارزیابی مورد اشاره در مقاله معیار اطلاعات متقابل نرمال(Normalized Mutual Information) شده می باشد که مبتنی بر تئوری اطلاعات بوده و نشان دهنده‌ی این است که دو متغیر تصادفی چه میزان از یکدیگر اطلاعات دارند. اینکه چه میزان از هم اطلاعات دارند را میتوان در قالب احتمالات وقوع عادی و توام مقادیر در نظر گرفت و آن را بر اساس بی نظمی (آنتروپی) نرمال کرد. ما اگر پیش‌بینی‌ها را یک متغیر تصادفی و برچسب‌ها را یک متغیر تصادفی در نظر بگیریم آنگاه میتوانیم این معیار را برای آن حساب کنیم. این معیار بین ۰ الی ۱ تغییرات دارد و هر چه بیشتر باشد بهتر است چرا که نشان میدهد متغیر تصادفی پیش‌بینی ها از متغیر تصادفی برچسب های حقیقی مطلع تر بوده و همانند

آن عمل می کند و این بهتر است. معادلات ریاضی و محاسباتی آن خارج از هدف توضیح معیار است لذا آورده نشده است.

چهارمین معیار ارزیابی تعداد خوشه های استخراجی است که بر حسب هر نوع دیتاست تعیین می شود که چند باشد و بایستی خروجی با تعداد خوشه های حقیقی یکی باشد.(در برخی موارد ۲ و در برخی موارد ۳ و....)

پنجمین معیار ارزیابی زمان اجرایی بر اساس ثانیه است که نشان میدهد چند ثانیه طول کشیده است الگوریتم اجرا و نتیجه خوشه بندی را خروجی دهد و هر چه کم باشد بهتر است. در مقاله اشاره نشده است که این زمان آیا مشتمل بر لود مجموعه داده بود است یا نه، یا مشتمل بر محاسبه ارزیابی بوده است یا نه و ... . بنده در پیاده سازی زمان کل را برای یک مجموعه داده گزارش کرده ام که بیشتر از زمان گزارشی توسط مقاله خواهد بود. نتایج اجرا ها در بخش بعدی (بخش سوم) آورده شده است.

د) چالش‌های پیاده سازی و راهکار های ابتکاری

بنده در پیاده سازی های انجام شده با چندین چالش مختلف مواجه بودم که دلیل آن عدم اشاره مقاله به آن جزییات بوده است که در ادامه به برخی از آنها اشاره و روند پیش گرفته شده را بیان می کنم که بترتیب پیشروی مقاله می باشد.

اولین چالشی که مواجه بودم، بحث پیش پردازش ها روی مجموعه داده و تاثیر آن بر عملکرد مدل مدل بود؛ از جایی که بحث سنجش فاصله مطرح بوده و رنج آن در ویژگی های مختلف میتوانست تاثیر سو داشته باشد، بنده بردارهای ویژگی را نرمال کرده ام؛ وقتی نرمال سازی انجام نمیشد نتایج نهایی در همه‌ی مجموعه داده ها حد ۱.۲٪ اختلاف داشت.

دومین چالشی که مواجه بودم تعیین تعداد و چگونگی همسایه‌ها در زمان تعیین زیرخوشه های اولیه بود. مقاله اشاره کرده بود که میتوان از Natural Neighbors استفاده کرد که خود این یک الگوریتم و مقاله دیگر بوده و مستلزم پیاده سازی کلا دیگری می بود. در جزییات پیاده سازی اشاره نکرده بود که برای مجموعه داده های مختلف چگونه رفتار کرده است. بنده برای این چالش از nearest neighbors استفاده کرده و برای از بین بردن تاثیر پارامتر K به ازای K های مختلف فرآیند خوشه بندی را انجام و نمودار های آن را رسم کرده ام و بر اساس آن ماکسیمم نتیجه را گزارش کرده ام. بازه‌ی

K های تست شده ۹۰ درصد حول جذر تعداد نمونه ها بوده است. (تعداد جذر نمونه ها یک مقدار توصیه شده برای K در کاربرد می باشد)

سومین چالشی که داشتم نحوه محاسبه‌ی فاصله‌ی دو خوشه و بدست آوردن argmin آن بصورت بهینه طبق فرمول مقاله بوده است. برای این منظور از خاصیت symmetric بودن ماتریس فاصله استفاده کردم که intersection سطر های متناظر با اعضای خوشه‌ی اول را با ستون های متناظر با اعضای خوشه‌ی دوم محاسبه کرده و فاصله های دو به دوی اعضا بصورت مستقیم حاصل می شود. حال با محاسبه min و argmin در ماتریس جدید و نگاشت اندیس های آن به ماتریس اولیه توانستم پاسخ را بدست آورم.

چهارمین چالشی که داشتم در زمان ادغام زیرخوشه‌ها با یکدیگر بود. چالش اینجا بود که چگونه میتوان تشخیص داد در نهایت کدام زیرخوشه‌ها همگی باهم یک خوشه را تشکیل میدهد و یک برچسب به آنها اختصاص می یابد. مثلا زیرخوشه ۱ ممکن است با زیر خوشه ۲ ادغام شود و زیر خوشه ۲ هم با زیر خوشه‌ی ۳ ادغام شود؛ در این حال چگونه می‌توان تشخیص داد که هر سه‌ی این یک خوشه را تشکیل میدهد؟ بنده برای حل این مشکل از گراف و همبندی گراف استفاده کردم. هر زیر خوشه را یک گره در نظر گرفته و در صورتی که هر دو زیر خوشه بایکدیگر ادغام شود بین آنها یال ایجاد می شود. در نهایت Connected Components های گراف و اعضای آن نشان میدهد که کدام زیر خوشه ها توام تشکیل یک خوشه نهایی را می دهد. برای این مورد در قسمت رو به رو از دیتاست Lithuanian شکل آورده شده است. مثلا زیر خوشه های ۸، ۱۰، ۵ و... باهم یک خوشه می شوند و ۱۱، ۱۲ و... باهم یک خوشه می شوند.

پنجمین چالشی که مواجه بودم تطبیق شماره خوشه‌ها با شماره برچسب‌ها بود. برای حل این مورد بنده از یک mapping اعضای خوشه با محاسبه mode در هر خوشه استفاده کرده و فضای شماره خوشه ها را به فضای شماره برچسب ها انتقال دادم. در مقاله این مورد اشاره نشده بود که چگونه انجام میپذیرد و بنده بر حسب یادگیری ماشین آن را انجام داده‌ام.

همچنین در خصوص چگونگی سنجش زمان اجرایی، معیار سنجش فاصله موردی اشاره نشده بود که بنده فرض های معمول را اتخاد کردم که عبارت بود از محاسبه از ابتدای اولین پردازش تا پایان محاسبه‌ی نتایج برای هر مجموعه داده و استفاده از معیار فاصله اقلیدسی.

بخش سوم: نتایج و بهبود

در این بخش به نتایج و گزارش های حاصل از پیاده سازی طبق توضیحات بخش قبل آورده شده است.

الف) نتایج پیاده سازی و تحلیل

برای ارائه نتایج و گزارش های حاصل از پیاده سازی طبق مجموعه داده های مقاله، ۵ مورد را انتخاب و آزمایش‌های مورد نیاز را روی آن انجام میدهیم. نتایج تجمیع شده بصورت زیر قابل مشاهده است:

Dataset	Accuracy	Recall	NMI	# Noises	Run-Time sec.	# Clusters	# Labels
Thyroid	0.73023	0.41111	0.14035	2	0.38799	2	3
Guassian	0.98950	0.99257	0.93707	54	33.21498	4	4
Ids2	0.99594	0.99830	0.97783	43	74.87539	5	5
Banana	1.00000	1.00000	1.00000	16	39.93293	2	2
Lithuanian	1.00000	1.00000	1.00000	2	42.26875	2	2

بنده از بین مجموعه داده‌های { Ecoli, Thyroid, Robot navigation} همه‌شان، بین مجموعه داده های {Lithuanian, Banana, } مجموعه داده‌ی Thyroid و از بین سایر مجموعه داده‌ها هم Ids2 را انتخاب و گزارش کرده ام. همانطور که در بخش چالش ها مطرح شد، بنده برای محاسبه زمان اجرایی از اولین لحظه محاسبات تا لحظه‌ی ثبت نتایج استفاده کرده ام لذا نتایج حاصل در این ستون کلا متفاوت است.

طبق جدول فوق قابل مشاهده است که نتایج حاصل از خوشه‌بندی و پیاده‌سازی انجام شده دقیقا منطبق و برابر با گزارش مقاله می باشد؛ این مطابقت عینی برای مجموعه داده های بنچمارک آزمایشگاهی(سنتز) نشان از پیاده سازی دقیق الگوریتم مقاله می باشد که حتی تا رقم اعشار نیز با آن‌ها یکی است. البته این مورد برای مجموعه داده Thyroid که Real-Life dataset می باشد صادق نبوده و صرفا اندکی متفاوت است؛ برای معیار accuracy و recall نتایجی که بنده طبق پیاده سازی بدست آورده ام بترتیب ۱.۳۸٪ و ۳.۳۳٪ بیشتر از نتایج مقاله بوده و برای معیار NMI نیز ۱۳.۹۶٪

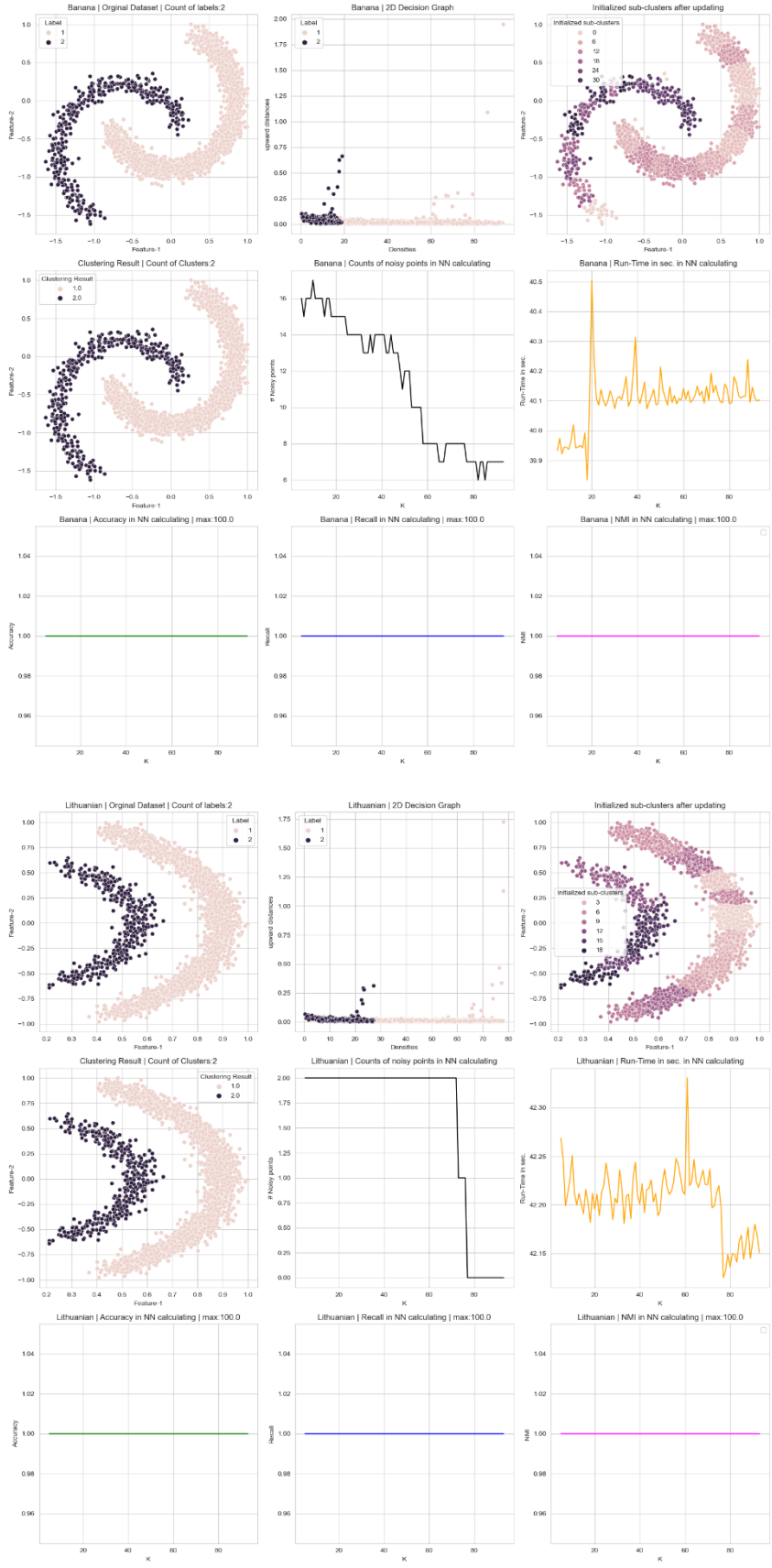
کمتر می باشد. این عدم مطابقت میتواند از دلایل متعددی نشات گرفته باشد؛ مثلا اعمال پیش پردازش‌های ابتدایی روی مجموعه داده های Real-Life (که در مقاله در خصوص آن صحبتی نشده است)، حذف داده های پرت، استفاده از معیار سنجش فاصله متفاوت(اقلیدسی یا کسینوسی یا ...)، انتخاب تعداد همسایه‌های متفاوت (بحث natural neighbor یا تعداد ثابت) و... .

علاوه بر توضیحات و تحلیل‌های فوق که مشتمل بر مقایسه‌ی نتایج نیز بود، بنده برای مجموعه داده‌ها یک گزارش آماده کرده ام که در زیر دو مورد از آنها قابل مشاهده است. همانطور که قبلا اشاره شد یکی از مواردی که در پیاده سازی وجود داشت، چگونگی انتخاب همسایه ها بود که مقاله آن را به یک مقاله‌ی دیگر ارجاع(natural neighbors) داده بود و طبق گفت و گوی گروه تلگرامی قرار شده بود که ما خود برای اینگونه موارد تصمیم گیری کنیم.

بنده از K-NN استفاده کرده و برای K های مختلف آزمایش را تکرار و بهترین نتیجه‌ی خوشه بندی را گزارش کرده ام. بازه‌ی آزمایش K ۹۰ درصد حول جذر تعداد نمونه ها بوده است. (تعداد جذر نمونه ها یک مقدار توصیه شده برای K است.) بر اساس این K ها میتوان تغییرات همه‌ی معیار های ارزیابی را رسم نموده و مقایسه انجام داد. در نمودار های مربوط به معیار های ارزیابی زیر محور افقی نشان دهنده‌ی این تغییرات می باشد.

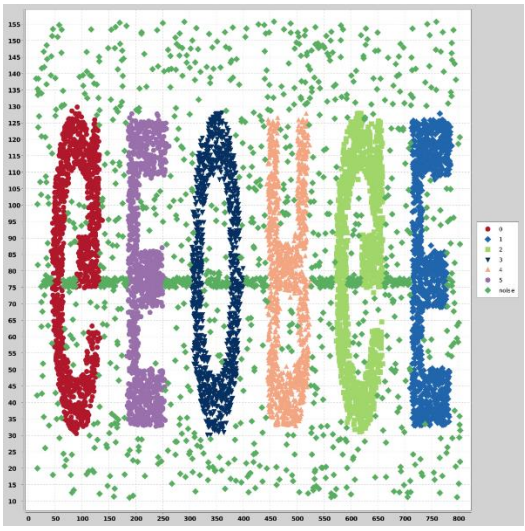
بترتیب از چپ به راست و از بالا به پایین نمودار های آورده شده مربوط به نمایش مجموعه داده به شکل اورجینال، گرافت تصمیم دو بعدی مورد اشاره در مقاله، زیرخوشه های ابتدایی، نتیجه‌ی خوشه بندی، تغییرات تعداد نویز شناسایی شده، تغییرات زمان اجرایی، تغییرات accuracy، تغییرات Recall و تغییرات NMI می‌باشد.

قابل مشاهده است که با افزایش K تعداد نویز های شناسایی شده کاهش پیدا میکند.


Banana | Orginal Dataset | Count of labels:2
Banana | 2D Decision Graph
Initialized sub-clusters after updating
Clustering Result | Count of Clusters:2
Banana | Counts of noisy points in NN calculating
Banana | Run-Time in sec. in NN calculating
Banana | Accuracy in NN calculating | max:100.0
Banana | Recall in NN calculating | max:100.0
Banana | NMI in NN calculating | max:100.0
Lithuanian | Orginal Dataset | Count of labels:2
Lithuanian | 2D Decision Graph
Initialized sub-clusters after updating
Clustering Result | Count of Clusters:2
Lithuanian | Counts of noisy points in NN calculating
Lithuanian | Run-Time in sec. in NN calculating
Lithuanian | Accuracy in NN calculating | max:100.0
Lithuanian | Recall in NN calculating | max:100.0
Lithuanian | NMI in NN calculating | max:100.0

ب) بخش امتیازی – بهبود عملکرد مدل

همانطور که در بخش معایب گفته شد، یک ضعف مقاله این است که نقاط نویزی را در انتها به یک خوشه‌ی آماده تخصیص می‌دهد که محتمل است اشتباه باشد چرا که ممکن است نویزها همگی در فضای ویژگی یکنواخت پخش شده باشد و نمونه‌های نویز حاصل از یک خوشه نبوده و به هیچ خوشه ای تعلق نداشته باشد؛ به عبارتی دیگر در مقاله خوشه‌ی نویزی نمیتواند ایجاد و تعیین کند؛ مثال این مورد در مجموعه داده‌ی cluto-t5-8k وجود دارد که در شکل زیر قابل نمایش است:



اگر مجموعه داده‌ی فوق به الگوریتم مقاله ورودی داده شود نخواهد توانست به دقت خوبی در خو شه بندی دست یابد؛ در پیاده سازی انجام شده معیار accuracy حاصل ۳۹٪ بوده است. برای حل این مشکل و ایجاد بهبود در الگوریتم مقاله، راه حلی ارائه و پیاده سازی شده است که در ادامه مورد مطالعه قرار میگیرد.

اولین تغییر ارائه شده در مرحله بروزرسانی زیرخوشه‌ها(update sub-clusters) بوده است؛ در این مرحله وقتی زیر خوشه‌ای در شرط صدق و بایستی از بین رفته و اعضای آن به سایر زیرخوشه‌ها تخصیص یابد، اعضای زیرخوشه حذف شده را به زیرخوشه دیگر اختصاص نمیدهیم بلکه اعضای آن را به عنوان نویز و برچسب صفر در نظر میگیریم. دلیل و استدلال این تغییر این است که ممکن است نویز های حاشیه‌ای و مرزی بین دو خوشه وجود داشته باشد و لزوما به هیچ زیر خوشه‌ای تعلق نداشته باشد و لذا تخصیص آن به زیر خوشه‌ی دیگر محتمل است اشتباه باشد.

دومین تغییر ارائه شده است در مرحله‌ی اختصاص داده‌های نویزی به خوشه‌ها پس از ادغام زیرخوشه ها می باشد؛ در این مرحله مکانیزمی بایستی در نظر گرفته شود که بین داده های نویزی که به هیچ خوشه تعلق نداشته و داده هایی که به یک خوشه تعلق داشته ولی از آن فاصله دارد تمایز ایجاد شود. برای ایجاد این تمایز، بهبود و راهکار ارائه شده مبتنی بر خوشه بندی مبتنی بر چگالی DBScan است. بین داده های نویزی الگوریتم مذکور اعمال می شود؛ پارامتر مورد استفاده برای حدآستانه فاصله eps برای خوشه بندی برابر با dc تعیین و در نظر گرفته ام. دلیل و استدلال این امر این است که مطمئن باشیم برای هر نقطه نویزی حداقل یک همسایه تعلق می گیرد. برای پارامتر حداقل تعداد نمونه min_samples برای یک خوشه در الگوریتم DBScan نیز از نصف میانگین ۷۵٪ خوشه‌های استخراجی استفاده می‌شود. دلیل این تصمیم این است که اگر تعداد اعضای خوشه ها نامتعادل بوده باشد، برای تصمیم گیری در خصوص تعداد نویز های بدست آمده و خوشه بندی آنها تاثیر سو نداشته باشد؛ به عبارتی حد بالای اعضای خوشه‌ها را در نظر گرفته نمی شود.

پس از خوشه بندی نویزها با DBScan، اگر نویز ها همچنان نویز شناسایی بشوند یعنی بقدر کافی از هم نیز دور هستند لذا میتوانند یک خوشه‌ی نویز بوجود آورند که در مجموعه داده و در فضای ویژگی حضور داشته اند(برچسب منفی یک). در صورتی که نویزهایی باهم خوشه شوند یعنی فاصله‌ی آنها مناسب و معنی دار بوده و میتوانند نسبت بهم رابطه نیز داشته باشند و این رابطه نشان میدهد که بطور تصادفی نویز شناسایی نشده و بلکه متعلق به یکی از خوشه‌مان قرار دارند و احتمالا در حاشیه‌ی خوشه قرار گرفته اند؛ برای این نقاط ایده مقاله انجام شده و به نزدیکترین خوشه تخصیص پیدا می کنند.

این راه حل ارائه شده بروی مجموعه داده‌های معرفی شده مقاله آزمایش شده و با خود مقاله نیز در جدول زیر مقایسه و آورده شده است. همچنین تعدادی مجموعه داده جدید(zelnik2, zelnik4, cluto-t5-8k) که نویزی هستند در دو حالت پایه و بهبود یافته آزمایش شده اند که نتایج در جدول زیر نیز آورده شده است. تصاویر این مجموعه داده نیز در ادامه آورده شده است.

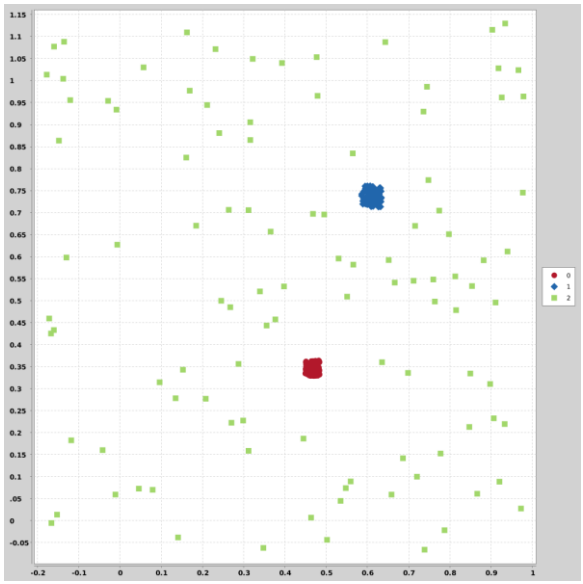
Dataset	Accuracy	Recall	NMI	# Noises	# Clusters	# Labels	Improved Version
zelnik2	0.41584	0.40881	0.13053	12	2	3	False
zelnik2	0.62046	0.60377	0.56968	12	2	3	True
zelnik4	0.60289	0.60000	0.62642	36	3	5	False
zelnik4	0.74920	0.73913	0.79819	36	4	5	True
Guassian	0.98950	0.99257	0.93707	53	4	4	False
Guassian	0.97350	0.94958	0.86872	53	4	4	True
cluto-t5-8k	0.39350	0.38381	0.39979	242	4	7	False
cluto-t5-8k	0.79450	0.78420	0.82836	242	6	7	True
Lithuanian	1.00000	1.00000	1.00000	2	2	2	False
Lithuanian	1.00000	1.00000	1.00000	2	2	2	True
Ids2	0.99594	0.99830	0.97783	72	5	5	False
Ids2	0.93469	0.79440	0.88233	72	4	5	True
Thyroid	0.73023	0.41111	0.14035	10	2	3	False
Thyroid	0.79070	0.55556	0.37867	10	2	3	True
Banana	1.00000	1.00000	1.00000	12	2	2	False
Banana	0.99833	0.99900	0.97635	12	2	2	True

ستون improved version نشان میدهد نسخه بهبوده یافته‌ی ارائه شده مورد آزمایش قرار گرفته است یا نسخه‌ی پایه از خود مقاله.

قابل مشاهده است که بهبود ارائه شده باعث افزایش دقت چشمگیر در مجموعه داده های نویزی و حتی غیر نویزی (Thyroid) شده است. در مجموعه داده های zelnik دقت بین ۱۵ الی ۲۰ درصد افزایش یافته است. در مجموعه داده های خود مقاله نیز دقت صرفا اندکی کاهش(بین ۱ الی ۶ درصد) وجود داشته است.

کاهش ۱–۶ درصد در مقابل افزایش ۱۵ الی ۲۰ درصد نسبت کمی تلقی شده و میتوان جمع بندی نمود که بهبود ارائه شده دارای توجیه استفاده می باشد. بنظر بنده مقاله بیش از حد مجموعه داده را عاری از نویز عادی دانسته و مجموعه داده هایی را سعی کرده انتخاب کند که نویز نداشته باشد و روی آن مانور دهد.

تصویر مجموعه داده zelnik2:



تصویر مجموعه داده zelnik4: