مقاله پژوهشی

راهکاری مبتنی بر ساخت درخت دودویی تقریبی برای سرعت‌بخشیدن به جستجوی نزدیک‌ترین همسایگی در داده‌های حجیم

حسین کلاته و نگین دانشپور

چکیده: با توجه به سرعت روزافزون تولید اطلاعات و نیاز تبدیل اطلاعات به دانش، روش‌های یادگیری ماشین قدیمی دیگر پاسخگو نیستند. هنگام استفاده از طبقه‌بندی‌ها با روش‌های یادگیری ماشین قدیمی، به ویژه استفاده از طبقه‌بندی‌های ذاتاً تنبل مانند روش k- نزدیک‌ترین همسایگی (KNN)، عملیات طبقه‌بندی داده‌های حجیم بسیار کند است.

نزدیک‌ترین همسایگی به دلیل سادگی و دقت عملی که ارائه می‌دهد یک روش محبوب در زمینه طبقه‌بندی داده‌ها می‌باشد. روش پیشنهادی مبتنی بر مرتب‌سازی بردارهای ویژگی داده‌های آموزشی در یک درخت جستجوی دودویی است تا طبقه‌بندی داده‌های بزرگ را با استفاده از روش نزدیک‌ترین همسایگی تسریع بخشد. این کار با استفاده از یافتن تقریبی دو دورترین داده محلی در هر گره درخت انجام می‌شود. این دو داده به عنوان معیار برای تقسیم داده‌های موجود در گره فعلی بین دو گروه، مورد استفاده قرار می‌گیرند. مجموعه داده‌های موجود در هر گره بر اساس شباهت آنها به این دو داده، به فرزند چپ یا راست گره فعلی تخصیص داده می‌شوند. نتایج آزمایش‌های متعدد انجام‌شده بر روی مجموعه داده‌های مختلف از مخزن UCI، میزان دقت خوب با توجه به زمان اجرای کم روش پیشنهادی را نشان می‌دهد.

کلیدواژه: **بافر همپوشانی، داده‌های حجیم، درخت تصمیم دودویی، طبقه‌بندی نزدیک‌ترین همسایگی.**

۱- مقدمه

در طی دهه گذشته، تقاضاهای روزافزون جهانی برای انواع ارتباطات و اطلاعات، منجر به پیشرفت‌های چشم‌گیری در زمینه فناوری اطلاعات شده است که اشتراک داده‌ها و ارتباطات جهان را تسهیل می‌کند [۱]. این تعاملات تکنولوژیکی جهانی در بین انسان‌ها، حجم زیادی از داده‌ها را ایجاد می‌کند. مدیریت این داده‌ها مزایای رقابتی برای شرکت‌ها دارد و یا به صورت اکتشافات مهم در زمینه‌های مختلف علمی منعکس می‌شود [۲]. شرکت‌ها و محققان برای ذخیره، تجزیه و تحلیل این مقدار زیاد از داده‌ها با مشکلات اساسی روبه‌رو هستند. این مقدار از داده‌ها تحت عنوان داده‌های حجیم[۱] معرفی می‌شوند [۳].

معمولاً داده‌های حجیم به ۲ دسته داده‌های جریانی[۲] و داده‌های ایستا[۳] یا دسته‌ای[۴] تقسیم می‌شوند. در مورد داده‌های جریانی، داده‌ها به طور مداوم در حال تولیدشدن هستند، در حالی که در مورد داده‌های دسته‌ای تمام داده‌های مورد استفاده از قبل موجود هستند؛ مثل تشخیص تخصیص وام به مشتری جدید که با توجه به مجموعه داده‌ای که از قبل جمع‌آوری شده است، این تصمیم‌گیری انجام می‌شود. کار ارائه‌شده در این مقاله بر روی داده‌های دسته‌ای تمرکز دارد.

KNN (نزدیک‌ترین همسایگی) [۴] به صورت گسترده در مسائل طبقه‌بندی[۵] استفاده می‌شود و از معروف‌ترین تکنیک‌های داده‌کاوی است [۵]. این روش بر اساس مفهوم شباهت کار می‌کند و اساس و ایده آن به این صورت است که نمونه‌هایی که مشابه هستند باید به کلاس یکسانی مربوط باشند. شباهت بر اساس معیار فاصله سنجیده می‌شود.

روش نزدیک‌ترین همسایگی، یک روش بر اساس یادگیری تنبل[۶] است [۶]. در این نوع یادگیری، مجموعه آموزش ذخیره شده و فرایند یادگیری تا زمان ورود نمونه آزمایش جدید به تأخیر می‌افتد. با توجه به حجم زیاد داده در مجموعه آموزش، این کار برای داده‌های حجیم مناسب نیست. دقیقاً بر خلاف این روش، نوع دیگری از یادگیری، یادگیری مشتاق[۷] است که به محض ورود نمونه آزمایش شروع به ساخت مدل می‌کند.

در مورد داده‌های حجیم، یافتن نزدیک‌ترین همسایگی به مقدار زیادی هزینه محاسباتی نیاز دارد. برخی از محققان از چارچوب‌های[۸] توزیع‌شده مانند آپاچی هدوپ[۹] برای نمونه‌های آموزش استفاده می‌کنند [۷]. این رویکردها معمولاً به یافتن دقیق‌ترین همسایگی نزدیک می‌شوند، اما برای این کار نیاز به استفاده از یک سیستم توزیع‌شده بزرگ است [۸].

بیشتر روش‌های شناخته‌شده یادگیری ماشین برای مرحله آموزش پرونده[۱۰]های حجیم به زمان زیادی نیاز دارند. برای مثال، طبقه‌بندی K نزدیک‌ترین همسایگی (KNN) بیش از ۶ هفته طول کشید تا مجموعه داده HIGGS را با استفاده از ماشین‌های محاسباتی قدرتمند طبقه‌بندی کند [۹]. بنابراین در مورد پرونده‌های داده‌های حجیم برای تأمین وسیله‌ای برای غلبه بر مشکل اندازه داده‌ها، تسهیل روند تجزیه و تحلیل و به دست آوردن اطلاعات حیاتی که برای برنامه‌های جدید مهم است، نیاز به تفکری مبتکرانه می‌باشد [۱۰].

این مقاله در تاریخ ۲۳ آبان ماه ۱۴۰۰ دریافت و در تاریخ ۱۰ اسفند ماه ۱۴۰۰ بازنگری شد.

حسین کلاته، دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجائی، تهران، ایران، (email: h.kalateh@sru.ac.ir).

نگین دانشپور (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجائی، تهران، ایران، (email: ndaneshpour@sru.ac.ir).

1. Big Data
2. Stream Data
3. Static Data
4. Batch Data
5. Classification
6. Lazy Learning
7. Eager Learning
8. Framework
9. Apache Hadoop
10. File

روش نزدیک‌ترین همسایگی، یک روش مبتنی بر نمونه می‌باشد که مشکل اساسی آن ناکارآمدی ذاتی و عدم مقیاس‌پذیری آن است، زیرا همه نمونه‌های مجموعه آموزشی باید هنگام ورود داده جدید مورد بررسی قرار گیرند [۱۱]. در هنگام مواجه‌شدن با داده‌های حجیم، پیداکردن نزدیک‌ترین همسایگی از میان داده‌های موجود بسیار سخت می‌شود. با توجه به توضیحات و به دلیل تولید اطلاعات زیاد در جامعه مدرن امروز، استفاده از چنین الگوریتم‌هایی در برنامه‌های جدید دارای محدودیت است [۱۲]. مشکلات موجود در این زمینه به صورت گسترده مورد مطالعه قرار گرفته‌اند که منجر به ارائه طیف گسترده‌ای از پیشنهادها شده که هدف این پیشنهادها کاهش تعداد کل جستجوها و در نتیجه تسریع روند طبقه‌بندی است. به صورت کلی در زمینه یافتن نزدیک‌ترین همسایگی در داده‌های حجیم، رویکردهای ارائه‌شده به گروه‌های زیر تقسیم می‌شوند: گروه اول حذف نمونه‌های نادرست و بدون تأثیر ایده‌آل بر عملکرد [۱۳] که باعث می‌شود رکوردهای کمتری مورد جستجو قرار گیرند. گروه دوم بهره‌گیری از ویژگی‌های فاصله برای اجتناب از محاسبات غیر ضروری [۱۴] که با کم‌شدن تعداد ویژگی‌های مورد جستجو باعث تسریع می‌شوند. گروه سوم ایجاد ساختارهای جستجو برای یافتن سریع نزدیک‌ترین همسایگی [۱۵] که هدف آنها تسریع فرایند در کنار حفظ دقت رقابتی با روش نزدیک‌ترین همسایگی می‌باشد.

در راستای طبقه‌بندی داده‌های حجیم، کار ارائه‌شده توسط حسنات [۱۶] که جزو گروه سوم طبقه‌بندی بالا می‌باشد، از انتخاب داده تصادفی و روش تپه‌نوردی[1] برای یافتن دو دورترین نقطه و ساخت درخت تصمیم استفاده می‌کند و هدف آن سرعت‌بخشیدن به فرایند یافتن نزدیک‌ترین همسایه برای پیداکردن طبقه داده آزمایش می‌باشد. این کار در بخش بعد به طور کامل شرح داده شده است.

این مقاله با ارائه روشی برای انتخاب نقطه اول به صورت هوشمندانه و همچنین استفاده از بافر همپوشانی، از منظر دقت باعث بهبود نتایج کار مذکور گردیده است. روش ارائه‌شده با روش‌های معروف موجود، بر روی مجموعه داده‌های UCI مقایسه می‌شود. ارزیابی دقیق نتایج بر روی چند مجموعه داده طبیعی نشان می‌دهد که این الگوریتم، با توجه به زمان کم اجرا و با دقت بالا می‌تواند بر روی مجموعه داده‌های بزرگ مورد استفاده قرار گیرد.

روش ارائه‌شده در این مقاله به طور خلاصه دارای مزایای زیر است:

۱) انتخاب نقطه اول به صورت هوشمندانه که باعث می‌شود با تعداد پیمایش کمتر، دو دورترین نقطه را یافت که نتیجه کار باعث افزایش سرعت مرحله آموزش می‌شود.

۲) با انتخاب نقطه اول به صورت هوشمندانه، همچنین می‌توان موازنه‌بودن فاصله دو دورترین نقطه را سنجید و از انتخاب مقادیر پرت[2] به عنوان دورترین نقاط جلوگیری کرد. این کار باعث می‌شود درخت بهتر شکل بگیرد و از تشکیل درختان مورب یا با عمق زیاد جلوگیری شود.

۳) عدم پیمایش درخت تا زمانی که در هر برگ درخت فقط یک داده موجود باشد، به دلیل استفاده کمتر از ساخت خوشه تقریبی باعث دقت بالاتر کار می‌گردد. همچنین افزودن بافر همپوشانی باعث می‌شود که یک حاشیه امن برای داده‌هایی که ممکن است نزدیک‌ترین همسایگی آنها در شاخه دیگر درخت باشد ایجاد گردد.

۴) در اجراهای متفاوت بر روی مجموعه داده یکسان، دقت روش تغییر نکرده و این باعث می‌شود که کار قابل اتکا بوده و بتوان آن را قضاوت کرد.

در ادامه و در بخش دوم، مطالعات و روش‌های پیشین مرور گردیده و در بخش سوم به بیان روش ارائه‌شده پرداخته شده است. در بخش چهارم، نتایج آزمایش‌های انجام‌گرفته ارزیابی گردیده‌اند. بخش پنجم نیز یک نتیجه‌گیری کلی از کار ارائه‌شده را بیان می‌کند.

۲- کارهای پیشین

در این بخش، کارهای پیشین انجام‌گردیده برای سرعت‌بخشیدن و افزایش دقت روش k- نزدیک‌ترین همسایگی در محیط داده‌های حجیم بررسی می‌شوند.

استفاده از طبقه‌بندی KNN روی داده‌های حجیم به قدرت محاسباتی بالایی نیاز دارد. در این روش طبقه‌بندی، برچسب کلاس یک نمونه آزمایش بر اساس k تا از نزدیک‌ترین نمونه‌های موجود در مجموعه آموزش تعیین می‌شود [۱۷].

به طور کلی کارهای انجام‌شده در این رابطه به ۲ دسته تقسیم می‌شود: کارهایی که از چارچوب موازی و توزیع‌شده مانند آپاچی هدوپ و آپاچی اسپارک[3] برای سرعت‌بخشیدن به یافتن نزدیک‌ترین همسایگی استفاده می‌کنند [۱۸] تا [۲۴] و دسته دیگر که از روش‌های تقسیم و غلبه و خوشه‌بندی برای کاهش حجم نمونه‌های آموزش استفاده می‌کنند [۲۵] و [۲۶]. کار این مقاله به دسته دوم مرتبط است.

به عنوان نمونه‌ای از گروه اول، بن هَیم و اِلَد تامتو [۲۷] یک روش سریع ساخت درخت تصمیم پیشنهاد می‌دهند و از روش ارباب- برده[4] استفاده می‌کنند. ابتدا داده‌های ورودی به سیستم‌های برده وارد شده و این سیستم‌های برده از این داده‌ها هیستوگرام می‌سازند و یک خلاصه آماری از داده‌ها را در اختیار پردازنده مرکزی قرار می‌دهند. این روش، داده‌ها را در مقدار معینی از حافظه فشرده می‌کند. پردازنده اصلی از این نقاط برای پیداکردن نقاط بهینه تقریبی برای شکستن گره‌های نهایی درخت تصمیم و به روز نگه داشتن درخت استفاده می‌کند. در این گروه کار، ژیا و همکاران [۲۸] برای تعیین ترافیک شهری در ساعات مختلف روشی ارائه کردند که بر اساس آن برای تعیین ترافیک در هر لحظه، علاوه بر استفاده از نزدیک‌ترین همسایگان زمانی برای حدس ترافیک از نزدیک‌ترین همسایگان مکانی نیز استفاده کردند که موجب بهبود دقت کار آنها شده بود و برای پیاده‌سازی کارشان نیز از روش نگاشت- کاهش[5] بهره بردند. در کار دیگری، گالگو و بنیتز [۲۲] روشی ارائه کردند که بر اساس آن ابتدا از میان داده‌های آموزشی، تعدادی داده با استفاده از نمونه‌برداری هوشمندانه انتخاب می‌شود و از این داده‌ها برای ساخت درخت در رایانه مرکزی استفاده می‌گردد. باقی داده‌ها بر اساس این درخت به برگ‌ها منتسب می‌شوند. به ازای هر برگ، یک رایانه قرار دارد که در آن داده‌های موجود با استفاده از الگوریتم RNGE یک گراف را تشکیل می‌دهند. در مرحله به روز رسانی و طبقه‌بندی داده‌های جدید در ابتدا با پیمایش درخت، داده جدید به یک رایانه می‌رسد. در اطراف داده جدید که برای طبقه‌بندی وارد می‌شود، گراف RNGE به روز رسانی می‌گردد. بر اساس همسایه‌های داده موجود که با یال به آن متصل هستند، تصمیم گرفته

1. Hill Climbing
2. Outlier
3. Apache Spark
4. Master-Slave
5. Map-Reduce

می‌شود که داده واردشده مورد قبول است و یا این که به عنوان داده پرت با آن برخورد شده و از مجموعه داده‌ها حذف گردد. همچنین به ازای تمام داده‌های موجود که با یال به داده جدید متصل هستند، بررسی می‌شود که آیا این داده‌ها به داده قدیمی تبدیل شده‌اند؟ در صورت مثبت‌بودن پاسخ، داده قدیمی حذف شده و گراف به روز رسانی می‌شود. معایب این کار عبارتند از پیچیدگی زمانی بالای الگوریتم RNGE و همچنین وابستگی سرعت انجام الگوریتم به تعداد مراحل نگاشت که در نظر گرفته شده است. از مزایای این کار هم می‌توان به پیداکردن نزدیک‌ترین همسایگان به صورت موازی اشاره کرد.

به عنوان نمونه‌ای از گروه دوم، آنجیولی و همکاران [۲۹] یک روش افزایشی جدید برای محاسبه یک زیرمجموعه سازگار از داده‌های آموزشی ارائه کردند. رویکرد پیشنهادی ایشان، نادیده‌گرفتن داده‌هایی بود که به داده‌های موجود شباهت داشتند و اطلاعات جدیدی اضافه نمی‌کردند. راه حل ارائه‌شده در این مقاله در حالی که هنوز دقت قابل قبولی داشت، سرعت یادگیری بهتر و پیچیدگی فضایی پایین‌تری را نسبت به روش نزدیک‌ترین همسایگی نشان می‌داد. در نتیجه برای زمانی که مسئله کمبود حافظه مطرح است، این روش مناسب می‌باشد. از سوی دیگر این روش پیچیدگی محاسباتی بیشتری نسبت به روش نزدیک‌ترین همسایگی دارد. در تحقیق انجام‌شده توسط دنگ و همکاران [۳۰] از روش خوشه‌بندی k-means برای تقسیم فضای نمونه به k تا خوشه استفاده شده است. پس از ساخت خوشه‌ها با توجه به مشخص‌بودن مراکز خوشه‌ها با ورود داده آزمایش، فاصله آن تا مراکز خوشه‌ها سنجیده شده و خوشه‌ای که مرکز آن کمترین فاصله را تا داده آزمایش دارد انتخاب می‌گردد. سپس از داده‌های آن به عنوان مجموعه داده آموزشی استفاده شده و با استفاده از روش نزدیک‌ترین همسایگی، برچسب طبقه داده آزمایش را حدس می‌زند. این روش به دو پارامتر برای تعیین تعداد خوشه‌ها و همچنین برای مشخص‌کردن تعداد همسایه‌ها برای روش نزدیک‌ترین همسایگی نیاز دارد که از مشکلات آن هستند. همچنین روش بیان‌شده به دلیل کم‌کردن فضای جستجو، از روش نزدیک‌ترین همسایگی سریع‌تر است اما نسبت به آن روش دقت کمتری دارد. سیدل و کریگل [۳۱] روش جدید چندمرحله‌ای ارائه دادند که از مکانیزم فیلتر برای کاهش تعداد نامزدها و در نتیجه کمترشدن اندازه‌گیری فاصله، در زمان ورود نمونه آزمایش جدید استفاده می‌کند. در نتیجه روش ارائه‌شده در این مقاله سرعت بهتری نسبت به روش نزدیک‌ترین همسایگی دارد. لی و همکاران [۳۲] بر اساس یک رویکرد مبتنی بر تراکم، مقدار داده‌های آموزشی را کاهش دادند و سپس از مجموعه آموزش جدید برای طبقه‌بندی استفاده کردند. در نتیجه سرعت یافتن طبقه داده آزمایش جدید افزایش یافت. در کار انجام‌شده توسط سعادت‌فر و همکاران [۸] برای بهبود کار تحقیقاتی دنگ، عنوان شده که معیار فاصله تا مرکز خوشه به تنهایی کفایت نمی‌کند. بنابراین پس از خوشه‌بندی مجموعه داده به وسیله روش k-means، از معیارهای چگالی و همچنین نحوه گسترش داده‌ها در خوشه‌ها استفاده کردند که باعث افزایش دقت کار شده است. همچنین در کار ارائه‌شده توسط گالگو و همکاران [۳۳] از دو روش خوشه‌بندی برای سرعت‌بخشیدن به روش نزدیک‌ترین همسایگی با نام‌های cKNN و cKNN+ استفاده شده است. روش دوم از رویکردی فازی[1] برای تقویت فرایند خوشه‌بندی استفاده می‌کند. دقت کار آنها به تعداد خوشه‌هایی که انتخاب می‌کنند، بستگی دارد اما با این حال زمانی که از روش شبکه عصبی عمیق[2] برای یادگیری و حدس بهتر تعداد خوشه‌ها استفاده می‌کنند، کارشان دقت خوبی را ارائه می‌دهد. همچنین در کار دیگری که توسط ایشان منتشر شد [۳۴]، در بهبود کار قبلی خود روشی به نام caKD+ ارائه کردند که در آن ابتدا با استفاده از شبکه عصبی عمیق سعی نمودند که ویژگی‌های نامرتبط مجموعه داده را نادیده بگیرند و در نتیجه، سرعت یافتن نزدیک‌ترین همسایگی را افزایش دهند. سپس با استفاده از خوشه‌بندی، فضای جستجو را با فرض این که تمام نزدیک‌ترین همسایگان در همان خوشه هستند، محدود کردند. برای این منظور در هر خوشه ابتدا تمام نزدیک‌ترین همسایگان هر عضو خوشه نیز به آن خوشه اضافه شدند. علاوه بر این به جای استفاده از یک مقدار k ثابت برای تمام مجموعه داده، با استفاده از یک فرایند پیش‌پردازش، مقدار k مناسب برای هر خوشه مشخص می‌شود. همچنین از یک ساختار داده درخت KD[3] برای تسریع در یافتن خوشه مناسب و همچنین جستجو در هر خوشه استفاده کردند. این کار باعث شد که دقت کار آنها بسیار بالا اما سرعت کارشان پایین باشد. در کار دیگری که توسط اوگیاروگلو، استفانو، اونگلیدیس و جرجیوس [۳۵] ارائه گردیده است، دو روش برای طبقه‌بندی داده‌های حجیم مطرح شده است. روش اول برای داده‌های ایستا مناسب می‌باشد و ابتدا مراکز هر طبقه را پیدا می‌کند و از این مراکز به عنوان نقطه شروع برای الگوریتم k-means استفاده می‌نماید. پس از پایان خوشه‌بندی، اگر در خوشه‌ای داده ناهمگن وجود داشته باشد این مراحل را تکرار می‌کند. پس از پایان مرحله آموزش با ورود داده آزمایش، طبقه نزدیک‌ترین خوشه به عنوان طبقه داده آزمایش انتخاب می‌شود. با این کار سرعت پیداکردن طبقه نسبت به روش نزدیک‌ترین همسایگی افزایش اما دقت کاهش پیدا می‌کند. در روش دوم پیشنهادی در مقاله ایشان از وزن برای داده‌ها و همچنین از بافر استفاده می‌کنند که برای داده‌های جریانی مناسب است.

در کار دیگری، ونگ و همکاران [۳۶] روشی مبتنی بر ساخت درخت دودویی متوازن برای طبقه‌بند چندمتغیره ارائه کردند که دقت بالایی دارد. در این مقاله از دو روش برای محاسبه وزن برای تشکیل درخت استفاده شده است: یکی به صورت تصادفی که با MDT RND و دیگری با استفاده از روش PCA که با MDT PCA نمایش داده شده است. از این درخت‌ها برای سرعت‌بخشیدن به مرحله آزمایش استفاده می‌شود و همچنین در هر برگ این درخت‌ها تنها یک داده وجود دارد که در نتیجه برای پیداکردن همسایگی نیاز به پیچیدگی بالا ندارد. در این مقاله در بخش آزمایش‌ها از این دو روش برای مقایسه کار استفاده شده است.

همچنین حسنات [۱۶] روشی را برای ساخت درخت دودویی ارائه کرده که مبتنی بر پیداکردن دو دورترین نقطه با استفاده از روش تپه‌نوردی محلی است. در این کار در هر لحظه فقط دو نقطه اصلی داریم. ابتدا یک نقطه به صورت تصادفی انتخاب شده و بعد دورترین نقطه از این داده را پیدا کرده و به عنوان نقطه اصلی در نظر می‌گیرد. در جستجوی بعدی به دنبال یافتن دورترین نقطه از نقطه اول می‌گردد و با یافتن این نقطه، آن را نقطه اصلی دوم قرار می‌دهد. در این مرحله از روش بازگشتی استفاده کرده و دومین نقطه اصلی را معیار قرار داده و به دنبال دورترین نقطه از این نقطه می‌گردد. اگر نقطه‌ای پیدا کند که فاصله آن تا نقطه اصلی دوم از دورترین نقاط یافته‌شده فعلی بیشتر باشد، نقاط یافته‌شده فعلی را به عنوان دورترین نقاط موجود، جایگزین نقاط قبلی می‌کند و این فرایند را

1. Fuzzy

2. Deep Neural Network

3. KD Tree

تکرار می‌کند تا دورترین نقاط را پیدا کند. حال این دو نقطه را فرزندان گره فعلی در نظر می‌گیرد. سپس داده‌های موجود در گره فعلی را با توجه به شباهتشان به این دو داده به دو گروه تقسیم می‌کند. برای هر کدام از فرزندان همین کار را تکرار می‌کند تا درخت تصمیمی ایجاد شود که در هر گره آن یک داده قرار داشته باشد. در این کار چون نقطه اول به صورت هدفمند انتخاب نمی‌شود و به صورت تصادفی این کار انجام می‌گردد، در هر اجرا ممکن است در دام قله محلی گرفتار شده و جواب متفاوتی ارائه دهد که این برای کار طبقه‌بندی مناسب نیست. همچنین برای پیداکردن دو دورترین نقطه نیاز است به تعداد دفعات زیاد داده‌های موجود در گره فعلی پیمایش گردند. از ایده این کار برای روش ارائه‌شده در این مقاله استفاده گردیده است. این کار نیز در بخش آزمایش‌ها برای مقایسه با روش ارائه‌شده با نام FPBST آمده است.

در دیگر کار ارائه‌شده توسط حسنات [۹]، ابتدا نرم دوم تمام داده‌های موجود در گره محاسبه می‌شود. در مرحله بعد با استفاده از دو روش ذکرشده در مقاله، کمترین و بیشترین نرم داده‌های موجود محاسبه شده و از آنها برای تقسیم فضای داده به دو زیردرخت استفاده می‌کند. این گام‌ها را برای هر دو زیردرخت تا رسیدن به گره برگ نیز تکرار می‌کند. این روش سرعت بیشتر اما در عین حال دقت به مراتب کمتری نسبت به روش FPBST دارد. در دیگر کار ارائه‌شده توسط حسنات [۳۷]، روش‌های مختلف که توسط ایشان ارائه شده است مورد آزمایش قرار گرفته‌اند. این روش‌ها عبارت هستند از Tabu، Beam و EPBST که همگی دارای پیچیدگی زمانی بیشتر و نیز دقت کمتری نسبت به FPBST هستند. همچنین در کار دیگری که توسط ایشان منتشر شد [۳۸]، برای ساخت درخت از دو روش RPBST و EPBST استفاده می‌کند. در روش RPBST از روش تصادفی برای انتخاب دو نقطه برای ساخت درخت استفاده می‌کند و داده‌ها را بر اساس شباهتشان به این دو داده به دو گروه تقسیم می‌نماید و این کار را ادامه می‌دهد. در روش EPBST ابتدا بیشینه و کمینه نرم داده‌ها را در گره ریشه پیدا می‌کند و به این داده‌ها بیشینه و کمینه سراسری می‌گوید، سپس داده‌ها را بر اساس شباهتشان به این دو داده به دو گروه تقسیم می‌کند. در هر کدام از فرزندان نیز دو داده که بیشترین شباهت را به کمینه و بیشینه سراسری دارند، انتخاب می‌کند و تقسیم‌بندی داده‌ها را بر اساس شباهتشان به این دو نقطه به دو گروه تقسیم می‌کند. این کار به صورت بازگشتی تا رسیدن به گره برگ انجام می‌شود. این روش‌ها از روش FPBST که توسط آقای حسنات ارائه شد سرعت بیشتر و دقت کمتری دارند.

در روش ارائه‌شده در این مقاله از بافر همپوشانی برای افزایش دقت استفاده گردیده که در واقع شکل ساده و تغییریافته از spill-tree که در کار [۳۹] ارائه شده است، می‌باشد. در بخش بعد به طور مفصل در رابطه با روش ارائه‌گردیده صحبت می‌شود.

۳- رویکرد پیشنهادی

در این بخش، مراحل مختلف رویکرد پیشنهادی برای طبقه‌بندی داده‌های ورودی شرح داده می‌شود. همان طور که در بخش گذشته نیز ذکر شد، پیداکردن نزدیک‌ترین همسایگی در محیط داده‌های حجیم از نظر زمان مصرفی کار بسیار پرهزینه‌ای است. برای مقابله با این مشکل دو راه حل وجود دارد: راه اول استفاده از چارچوب توزیع‌شده است و راه دوم استفاده از تکنیک‌های خوشه‌بندی یا روش‌های تقسیم و غلبه می‌باشد که در این مقاله از راه دوم استفاده شده است.

معمولاً در یادگیری ماشین[1] مجموعه‌ای تحت عنوان مجموعه آموزش شامل تعدادی نمونه[2] که هر نمونه با برچسب طبقه‌اش مشخص شده است، در دسترس می‌باشد. یک نمونه داده با بردار ویژگی نمایش داده می‌شود. پیچیدگی زمانی برای پیداکردن نزدیک‌ترین همسایگی یک نمونه ورودی از مرتبه $O(n.d)$ است که n تعداد رکوردهای موجود در مجموعه داده و d نشان‌دهنده ابعاد آن می‌باشد و به این دلیل که در محیط داده‌های حجیم n مقدار بزرگی دارد، روش پیداکردن نزدیک‌ترین همسایگی غیر بهینه است.

در تکنیک پیشنهادی برای کوچک‌کردن فضای جستجو و در نتیجه افزایش سرعت، از درخت تصمیم دودویی استفاده شده است. هدف از ساخت درخت، تسهیل در یافتن برچسب طبقه داده آزمایش است که با استفاده از مرتب‌سازی داده‌های آموزشی صورت می‌گیرد. برای این کار ابتدا از مجموعه داده‌های آموزشی، دو دورترین داده (یا به اصطلاح قطر مجموعه) در فضای ویژگی اقلیدسی انتخاب می‌شوند و از این داده‌ها برای تقسیم‌بندی سایر داده‌های موجود بر اساس میزان شباهتشان به این نقاط استفاده می‌گردد. برای درک منطق استفاده‌شده در کار، انتخاب دو داده دور به این دلیل است که این داده‌ها کمترین شباهت را به یکدیگر دارند که از این جهت به احتمال زیاد متعلق به دو طبقه مختلف می‌باشند. بنابراین از این داده‌ها برای طبقه‌بندی سایر داده‌ها استفاده می‌شود. البته این منطق ممکن است لزوماً همیشه جواب درست ندهد. به طور مثال زمانی که داده‌های مربوط به یک طبقه توسط داده‌های مربوط به طبقه دیگر محاط شده باشند، با این کار لزوماً داده‌هایی که شباهت بیشتری به هم دارند بدون توجه به برچسب طبقه در یک گروه قرار می‌گیرند و داده‌های دورتر و کم‌شباهت‌تر در گروهی جداگانه قرار می‌گیرند. این فرایند دقیقاً همان کاری است که روش نزدیک‌ترین همسایگی به دنبال آن است.

در فاز آموزش، درخت تصمیمی ایجاد می‌شود و برای جستجو در مرحله آزمایش مورد استفاده قرار می‌گیرد. این درخت باعث سرعت‌بخشیدن به تعیین نوع طبقه داده در مقایسه با زمان مورد نیاز در روش نزدیک‌ترین همسایگی می‌شود که در مورد داده‌های حجیم، زمانی غیر قابل قبول دارد. برای ساخت درخت، نیاز به معیاری برای تصمیم‌گیری در هر مرحله داریم. کار اصلی این مقاله، یافتن رویکرد بهینه برای پیداکردن دو دورترین داده است که به عنوان معیار در ساخت درخت تصمیم استفاده می‌شوند. اگر فرض کنیم که در این مرحله از ساخت درخت، دو دورترین داده با P۱ و P۲ مشخص شوند، داده‌هایی که به P۱ شبیه‌تر هستند در زیردرخت P۱ و داده‌هایی که به P۲ شبیه‌تر هستند در زیردرخت P۲ گره موجود در درخت تصمیم طبقه‌بندی می‌شوند. با فرض توزیع یکسان داده‌ها در مجموعه داده، با این کار تعداد داده‌های مورد جستجو در هر مرحله نصف می‌شود. با تکرار مداوم یافتن دو دورترین نقطه در هر مرحله، بر اساس داده‌های محلی موجود در آن گره، درخت مورد نظر تشکیل می‌شود. خلاصه کارهای پیشین که در این بخش توضیح داده شد، به همراه معایب و مزایای هر روش به صورت خلاصه در جدول ۱ ارائه شده است.

از فاصله اقلیدسی که یکی از شناخته‌شده‌ترین معیارهای اندازه‌گیری موجود می‌باشد، برای اندازه‌گیری میزان شباهت استفاده شده است. فاصله بین دو نقطه $d(x,y)$ با استفاده از (۱) تعریف می‌شود

1. Machine Learning
2. Instance

جدول ۱: خلاصه روش‌های ارائه‌شده در کارهای پیشین.

	نام مقاله	سال ارائه	توضیح مختصر	مزایا	معایب
گروه اول (کارهای ارائه‌شده برای داده‌های جریانی و با استفاده از سیستم‌های توزیع‌شده)	[۲۷]	۲۰۱۰	کار این الگوریتم، ساخت سریع درخت تصمیم از هیستوگرام‌های ساخته‌شده در پردازنده‌های slave است که داده‌ها را به مقدار معینی فشرده می‌کند و یک دید آماری از داده‌ها ارائه می‌دهد. یک پردازنده اصلی نیز از این نقاط برای پیداکردن نقاط بهینه تقریبی برای شکستن گره‌های نهایی درخت استفاده می‌کند.	پیداکردن نقاط split به صورت موازی نیاز به یک بار رصد داده‌ها دارد. قوانین قابل فهم برای انسان (ویژگی DT)	افت دقت (کم) نرخ خطا (کم)
	[۲۸]	۲۰۱۶	این روش برای پیش‌بینی ترافیک در یک محل خاص از دو تابع ODT و OPP استفاده می‌کند و برخلاف روش‌های گذشته که برای پیش‌بینی فقط جاده مد نظر را در زمان‌های مختلف در نظر می‌گرفتند، وابستگی مکانی (یعنی جاده‌های مجاور) را نیز برای پیش‌بینی در نظر می‌گیرد و به همسایگی‌های نزدیک‌تر وزن بیشتر می‌دهد تا اثر انتخاب k کمرنگ‌تر شود.	در نظر گرفتن وابستگی مکانی علاوه بر وابستگی زمانی با استفاده از ویژگی وزن‌دهی به رأی‌ها بر اساس نزدیکی به محل مورد نظر، اثر k را کم‌رنگ می‌کند.	داده تست را به همه گره‌ها می‌فرستد و گامی برای پیش‌بینی گره مناسب ندارد.
	[۲۲]	۲۰۱۷	الگوریتم ارائه‌شده از درخت m-tree استفاده می‌کند و kNN را که روشی بر اساس یادگیری تنبل است، برای جریان داده‌ها مناسب می‌کند و به صورت موازی از آن استفاده می‌نماید.	انجام k نزدیک‌ترین همسایه که روشی بر پایه یادگیری تنبل است به صورت موازی برای داده‌های جریانی.	پیچیدگی زمانی بالا سرعت وابسته به تعداد مراحل نگاشت
	[۲۹]	۲۰۰۵	با نادیده‌گرفتن داده‌هایی که اطلاعات جدید به سیستم اضافه نمی‌کنند، باعث کم‌شدن حجم داده می‌شود.	دقت قابل قبول	پیچیدگی زیاد سرعت کم
	[۳۰]	۲۰۱۶	ابتدا با استفاده از k-means خوشه می‌سازد و سپس در قسمت آزمایش، ابتدا نزدیک‌ترین خوشه را برای داده ورودی پیدا می‌کند و سپس درون نزدیک‌ترین خوشه، روش kNN را برای پیداکردن برچسب داده‌های ورودی جدید اعمال می‌نماید.	فضای آموزش را کاهش می‌دهد، در نتیجه زمان کمتری نسبت به kNN دارد.	به انتخاب k و m (تعداد خوشه‌ها) حساس است. در مقایسه با kNN کمی افت دقت دارد.
	[۹]	۲۰۱۸	در این کار ابتدا نرم دوم تمام داده‌ها محاسبه شده و سپس با توجه به دو راه حل برای کمینه و بیشینه نرم، شروع به ساخت درخت دودویی می‌کند.	سرعت بیشتر نسبت به FPBST	دقت کمتر نسبت به FPBST
	[۳۷]	۲۰۱۸	روش‌هایی از قبیل تپه‌نوردی، BEAM و TABU را ارائه می‌کند که همگی بر اساس ساخت درخت دودویی هستند.	مناسب برای داده‌های بزرگ	پیچیدگی زمانی زیاد دقت کمتر نسبت به FPBST
گروه دوم (کارهای ارائه‌شده برای داده‌های ایستا و با استفاده از یک رایانه)	[۳۸]	۲۰۱۸	ابتدا کوچک‌ترین و بزرگ‌ترین نرم داده‌ها را پیدا می‌کند و از این داده‌ها برای تقسیم‌کردن داده‌ها در تمام مراحل استفاده می‌نماید.	سرعت بهتر نسبت به FPBST	دقت پایین‌تر نسبت به FPBST
	[۳۵]	۲۰۱۴	ابتدا مرکز هر کلاس را برای نقطه شروع k-means انتخاب می‌نماید. پس از اتمام خوشه‌بندی، این کار را برای خوشه‌های ناهمگن تکرار می‌کند. در روش بعدی هم برای داده‌های جریانی مناسب بوده و از وزن و بافر استفاده می‌کند.	افزایش سرعت نسبت به kNN دقت قابل قبول	کاهش دقت به خاطر محاسبه نزدیک‌ترین همسایگی به صورت تقریبی
	[۳۳]	۲۰۱۸	از دو روش خوشه‌بندی برای سرعت‌بخشیدن به روش نزدیک‌ترین همسایگی با نام‌های cKNN و cKNN+ استفاده کرده‌اند. روش دوم از خوشه‌هایی که هم‌پوشانی دارند، برای بهبود فرایند خوشه‌بندی استفاده می‌کند. دقت کار آنها به تعداد خوشه‌هایی که انتخاب می‌کنند، بستگی دارد اما با این حال زمانی که از روش شبکه عصبی عمیق برای یادگیری و حدس بهتر تعداد خوشه‌ها استفاده می‌کنند، کارشان دقت خوبی را ارائه می‌دهد.	دقت خوب کار	پیچیدگی بالا زمان زیاد اجرا
	[۳۴]	۲۰۲۲	روش caKD+ را ارائه کردند که در آن، ابتدا با استفاده از شبکه عصبی عمیق ویژگی‌های مناسب را انتخاب نمودند. سپس با استفاده از روش خوشه‌بندی، فضای جستجو برای نزدیک‌ترین همسایگی را محدود کردند. برای جلوگیری از خوشه‌های همپوشان در هر خوشه نزدیک‌ترین همسایگان اعضای خوشه نیز به خوشه اضافه شدند. علاوه بر این به جای استفاده از مقدار ثابت k برای هر خوشه، از k مناسب استفاده شد. همچنین از روش KD-Tree برای انتخاب خوشه مناسب و یافتن نزدیک‌ترین همسایگان داخل خوشه استفاده شد.	دقت بالاتر نسبت به روش‌های cKNN و cKNN+	سرعت پایین
	[۳۶]	۲۰۱۸	ابتدا داده‌ها را بر روی بردار تصادفی و PCA نگاشت و سپس آنها را نصف می‌کند. اگر خلوص هر گروه کافی بود که ادامه نمی‌دهد و در غیر این صورت دو گره جدید ایجاد می‌کند.	دقت بالا و قابل قبول	پیچیدگی زیاد و سرعت کم
	[۸]	۲۰۲۰	در بهبود کار آقای دنگ ذکر کردند که فاصله به تنهایی کفایت نمی‌کند و پس از خوشه‌بندی با k-means از معیارهای چگالی و نحوه گسترش داده‌ها نیز استفاده کردند.	بهبود دقت کار آقای دنگ	احتیاج به زمان بیشتری دارد. وابستگی به معیارهای α و β

```
1.  Input: Numerical training data set,
        a list of DATA with size n and d features,
        MN.
2.  Output: A Root pointer to the NFPBST.

3.  RootN = Create a Node
4.  RootN.Examples ← FVs //all indexes of FVs from the training set

5.  Method Furthest(FVs)
6.      sum =  0
7.      count = 0
8.      for each point Pi in FVs:
9.          count = count + 1
10.         sum = sum + Pi
11.     if count ≤ MN : return
12.     P0 =  sum/count       // Calculate the average and assign to P0
13.     max =  0              // for calculate maximum distance
14.     for each point Pi in FVs:
15.         D ← ED (P0, Pi)
16.         if D > max :
17.             max = D
18.             P1 =  Pi
19.
20.     max = 0
21.     for each point Pi in FVs:
22.         D ← ED (P1, Pi)
23.         if D > max :
24.             max = D
25.             P2 = Pi
26.     return (P0, P1, P2, max)

27. Method swap(RootN,P1,P2)
28.     if euclidianNorm(P1) > euclidianNorm(P2): Xchange(P1, P2)
29.         RootN.P1 ← P1
30.         RootN.P2 ← P2
31.     return (P1,P2)

32. Method BuildBST(RootN)
33.     Points p ← Furthest(RootN.examples)
34.     If p==null : return
35.     If (isUnbalance(P0, P1, P2)) : Check the next point
36.     t = p.max * τ
37.     for each Pi in Node :
38.         if(ED(Pi, p.P1) ≤ ED(Pi, p.P2))+t
39.             Add index of FVi to Node.Left.Examples
40.         else
41.         if(ED(Pi, p.P2) ≤ ED(Pi, p.P1))+t
42.             Add index of FVi to Node.Right.Examples
43.     BuildBST(Node.Left)
44.     BuildBST(Node.Right)
```

شکل ۱: مرحله آموزش.

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(a_i(x)-a_i(y))^2} \qquad (1)$$

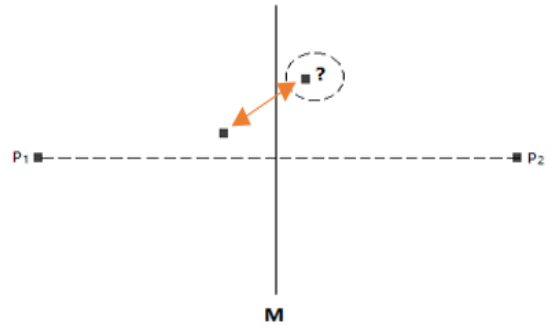
در (۱)، x و y دو نمونه داده می‌باشند که با استفاده از بردار ویژگی $\langle a_1(x), a_2(x), \ldots, a_d(x)\rangle$ توصیف می‌گردند که $a_i(x)$ نشان‌دهنده iامین ویژگی نمونه x است.

روش توضیح داده شده برای مرحله آموزش، در شکل ۱ نشان داده شده است. برای یافتن دو دورترین نقطه، ابتدا میانگین داده‌های موجود در گره محاسبه و P۰ نامیده می‌شود. با انتخاب هدفمند نقطه اول، کار یافتن دو دورترین نقطه تسهیل می‌گردد. به این دلیل که وقتی از دو دورترین نقطه سخن گفته می‌شود، یکی از این نقاط به احتمال زیاد بیشترین فاصله را نسبت به میانگین داده‌ها دارد. دورترین داده نسبت به میانگین جستجو شده و P۱ نامگذاری می‌شود. حال کافی است دورترین داده از داده P۱ را پیدا کرده و با P۲ مشخص کنیم که این دو نقطه به احتمال زیاد دو دورترین نقطه موجود در مجموعه داده‌های محلی هستند. این داده‌ها دو دورترین نقطه اصلی محلی نام‌گذاری می‌شوند. حال ممکن است که در مجموعه داده، توزیع یکسانی بر روی داده‌ها نباشد یا در مجموعه داده، مقادیر پرت وجود داشته باشد که در بدترین حالت در هر مرحله، مقادیر پرت به عنوان دورترین نقطه اصلی انتخاب شوند و پیچیدگی مسئله را به پیچیدگی n^2 ببرند. برای مقابله با این مسئله می‌توان توازن دو نقطه یافته‌شده به عنوان دورترین نقاط را نسبت به میانگین داده‌ها مقایسه کرد و در صورت عدم توازن به سراغ دورترین نقطه بعدی نسبت به میانگین به عنوان نقطه اصلی رفت.

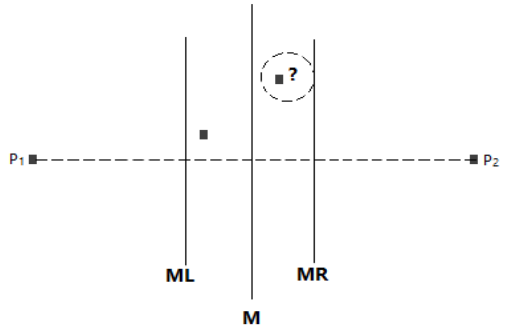
همچنین برای یکپارچگی در ساخت درخت، در هر مرحله داده‌ای که به مرکز مختصات نزدیک‌تر و در نتیجه قدرمطلق اندازه آن کوچک‌تر است، به عنوان فرزند چپ و متعاقباً داده دیگر به عنوان فرزند راست گره موجود انتخاب می‌شود.

با این کار دو دورترین نقطه مورد نیاز برای ساخت درخت پیدا می‌شوند. به این خاطر که در هر مرحله، فضای جستجو نصف می‌گردد و نصف داده‌ها به گره سمت چپ و نصف دیگر به گره سمت راست می‌روند، عمق درخت حداکثر $\log n$ می‌شود. برای ساخت درخت می‌توان این مراحل را تکرار کرد. با ورود هر داده برای آزمایش، درخت پیمایش شده و طبقه داده مورد نظر پیدا می‌شود. این کار باعث بالارفتن سرعت جستجو و روش پیشنهادی می‌گردد. چون ساخت درخت با یافتن نقاط تقریبی شکل می‌گیرد، این کار ممکن است باعث کاهش دقت شود. برای مقابله با این مشکل می‌توان دو کار انجام داد: به عنوان کار اول می‌توان ساخت درخت را تا سطح خاصی از ایجاد گره‌های جدید درخت ادامه داد و به عنوان کار دوم می‌توان ساخت درخت را تا مرحله‌ای ادامه داد که داده‌های موجود در هر برگ از یک تعداد مشخصی کمتر شود. در این مقاله از روش دوم استفاده شده است. متغیری به نام MN وجود دارد که نشان‌دهنده این تعداد است. در هر مرحله اگر تعداد داده‌ها بیشتر از مقدار موجود در متغیر MN باشد، ساخت درخت ادامه پیدا می‌کند. بدیهی است که هرچه عدد MN بزرگ‌تر باشد، مرحله آموزش سریع‌تر به اتمام می‌رسد. همچنین با انتخاب عدد بزرگ برای MN با توجه به این که پیداکردن طبقه داده مورد آزمایش در هر شاخه درخت به روش نزدیک‌ترین همسایگی سوق پیدا می‌کند، دقت افزایش می‌یابد. با این کار مرحله آزمایش طولانی‌تر می‌شود چون پس از رسیدن داده آزمایش به گره برگ مورد نظر باید نزدیک‌ترین همسایه از بین حداکثر MN داده دیگر پیدا شود. در نتیجه سرعت اجرا کندتر و به زمان اجرای نزدیک‌ترین همسایگی نزدیک‌تر می‌شود. با مقداردهی MN به اندازه کل داده‌های آموزش، دقیقاً روش نزدیک‌ترین همسایگی که دقت بالاتر و سرعت پایین‌تری دارد، اجرا می‌شود. این مقدار می‌تواند با توجه به نیاز و شرایط تغییر کند اما برای کارهای انجام‌شده در این مقاله، این مقدار با توجه به آزمایش‌های صورت‌گرفته، عدد ۱۲۰ قرار داده شده است.

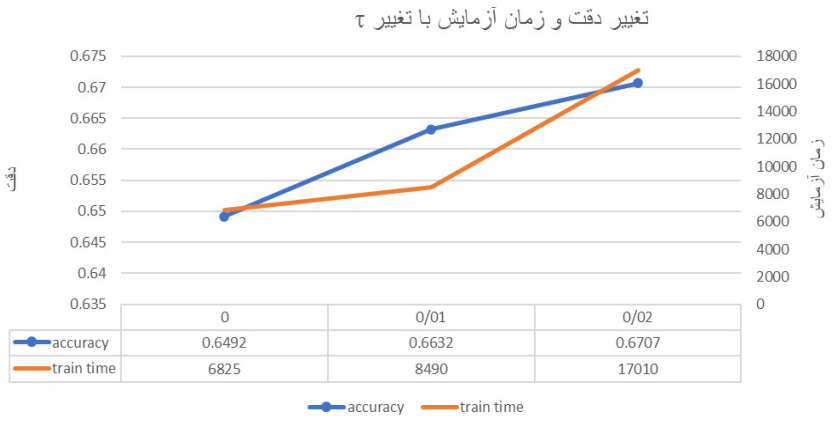
حال با فرض این که دو دورترین نقطه یافته‌شده به ترتیب نقاط P۱ و P۲ باشند، با توجه به شکل ۲ برای پیداکردن برچسب طبقه داده‌های موجود در گره از وسط فاصله دو دورترین نقطه یعنی خط M داده‌ها به دو گروه تقسیم می‌شوند. با استفاده از روش مذکور، نقطه‌ای که با علامت سؤال مشخص شده است، به زیردرخت راست و نقطه‌ای که از لحاظ فاصله با نقطه مورد نظر کمترین فاصله را دارد به زیردرخت سمت چپ تعلق می‌گیرد. این برخلاف خواسته ما که استفاده از روش نزدیک‌ترین همسایگی می‌باشد است.



شکل ۲: مشکل ساخت درخت تقریبی.



شکل ۳: استفاده از بافر همپوشانی برای مقابله با مشکل ساخت درخت تقریبی.



شکل ۴: نحوه تغییرات دقت و زمان آموزش با تغییر مقدار τ.

جدول ۲: مقادیر مختلف τ به ازای مجموعه داده‌های مختلف.

نام مجموعه داده	τ	نام مجموعه داده	τ
covtype	٪۱۲	Normalized Covtype	٪۲
HIGGS	٪۱	Normalized HIGGS	٪۱
letter	٪۱۱	Normalized letter	٪۱۱
SUSY	٪۴	Normalized SUSY	٪۴
waveform	٪۱۰	Normalized waveform	٪۷
satimage	٪۱۶	Normalized satimage	٪۱۷
Mnist	٪۴	Physical Unclonable Functions	٪۶

برای مقابله با این مشکل طبق شکل ۳ به جای این که از خط M داده‌ها به دو قسمت تقسیم شوند از بافر همپوشانی استفاده می‌شود. بافر همپوشانی یک حاشیه امن است که در اطراف خط M ایجاد می‌گردد و داده‌های درون آن در هر ۲ فرزند گره موجود نگهداری می‌شود. به عبارت دیگر، داده‌های موجود در سمت چپ MR در زیردرخت چپ گره فعلی و داده‌های موجود در سمت راست ML در شاخه راست گره فعلی ذخیره می‌شوند. این کار باعث ایجاد داده‌های تکراری و اضافی در درخت می‌شود و در سرعت اجرای روش تأثیر می‌گذارد، اما باعث افزایش دقت و از بین رفتن مشکل عنوان‌شده می‌شود. برای تعیین مقدار بافر همپوشانی دو راه وجود دارد: راهکار اول آن است که این مقدار را یک مقدار ثابت در نظر گرفت. برای مثال τ را ۰/۲ در نظر گرفت و این کار را تا زمانی که مقدار τ بزرگ‌تر از نصف فاصله دو دورترین نقطه شود در هر مرحله تکرار کرد، چون در غیر این صورت درخت به سمت بی‌نهایت کشیده می‌شود. راهکار دوم این است که مقدار τ در هر مرحله برابر درصدی از فاصله دو دورترین نقطه باشد که با این کار می‌توان تا آخرین مرحله ساخت درخت از بافر همپوشانی استفاده کرد. در این مقاله از روش دوم استفاده شده و مقدار τ در هر مجموعه داده با توجه به آزمایش‌های مختلف بر اساس موازنه بین سرعت و دقت تعیین شده است. زیرا با افزایش τ، دقت کار افزایش اما سرعت کاهش می‌یابد. برای تعیین حد بالای τ به این نحو عمل شده که در هر مرحله، مقدار τ در الگوریتم به مقدار ۰/۰۱ افزایش داده می‌شود و زمان آن به نسبت اجرای قبل محاسبه می‌گردد. اگر زمان انجام مرحله آموزش نسبت به مرحله قبل دو برابر یا بیشتر شده باشد، اجرا به پایان می‌رسد. در این میان با توجه به داده‌های test و train موجود، مقدار τ برای الگوریتم به گونه‌ای انتخاب می‌شود که مقدار دقت را بیشینه کند. همچنین مقدار τ انتخابی باید کمتر از ۰/۵ باشد، در غیر این صورت درخت به سمت بی‌نهایت میل خواهد کرد. مثلاً در شکل ۴ برای مجموعه داده Normalized covtype مشاهده می‌شود که با تغییر مقدار τ از مقدار ۰/۰۱ به مقدار ۰/۰۲، زمان اجرا از ۸۴۹۰ میلی‌ثانیه به ۱۷۰۱۰ میلی‌ثانیه افزایش می‌یابد. در نتیجه اجرا پایان یافته و مقدار مناسب τ برای این کار با توجه به دقت به دست آمده برابر ۰/۰۲ خواهد بود. به همین طریق برای مجموعه داده‌های استفاده‌شده، مقدار τ به دست آمده که در جدول ۲ قابل مشاهده است.

هزینه پیچیدگی زمانی برای ساخت درخت $O(5n.\log n)$ است که $3n$ برای یافتن ۲ دورترین نقطه صرف می‌شود و $\log n$ هزینه مصرفی برای پیمایش عمق درخت می‌باشد. $2n$ نیز در هر مرحله هزینه مقایسه داده‌ها با ۲ دورترین داده موجود در گره و تقسیم داده‌ها به ۲ طبقه است. از آنجا که عدد ثابت در پیچیدگی زمانی لحاظ نمی‌شود، هزینه پیچیدگی زمانی نهایی برای ساخت مرحله آموزش برابر $O(n.\log n)$ است.

در مرحله آزمایش، داده ورودی از گره اصلی درخت شروع به بررسی می‌کند و با توجه به شباهت به داده‌های انتخاب‌شده به عنوان فرزندان گره فعلی، به یکی از دو شاخه درخت می‌رود. در هر گره با تکرار این مقایسه‌ها به یکی از برگ‌های درخت می‌رسد و در آن گره برگ که تعداد داده‌ها کم است از روش نزدیک‌ترین همسایگی برای تعیین برچسب استفاده می‌شود.

```
1.  Input: test data set TESTDT with size n and d features.
2.  Output: Testing Accuracy Acc.
3.
4.  Method test:
5.  |  Acc = 0
6.  |  Node = Root
7.  |  for each data in TESTDT:
8.  |  |  Node ← GetTreeNode(Node, data)
9.  |  |  class ← KNN(data, Node)
10. |  |  if class==Class(data):
11. |  |  |  Acc = Acc +1
12. |  Acc ←Acc/n
13. |  return Acc
14.
15. Method GetTreeNode(Node, data):
16. |  D1 ←ED(data, Node.P1)
17. |  D2 ←ED(data, Node.P2)
18. |  if (D1 < D2 and Node.Left is exist):
19. |  |  return GetTreeNode(Node.Left, data)
20. |  else if (D2 ≥ D1 and Node.Right is exist):
21. |  |  return GetTreeNode(Node.Right, FVi)
22. |  else:
23. |  |  return Node
24.
25. Method KNN(FVi, Node):
26. |  Array Distance;
27. |  for each data in Node :
28. |  |  Distance[j] ←ED[data, FVi]
29. |  index ←argmin(Distance[j])
30. |  class ←Class(Node.data[index])
31. |  return class
32.
```

شکل ۵: مرحله آزمایش.

اگر در گره برگ فقط یک داده وجود داشته باشد، نیاز به استفاده از روش نزدیک‌ترین همسایگی نیست چون نزدیک‌ترین همسایه مشخص است. اما در کار ارائه‌شده در هر صورت از روش نزدیک‌ترین همسایگی برای یافتن برچسب مناسب استفاده شده است. روش توضیح داده‌شده برای مرحله آزمایش، در شکل ۵ نشان داده شده است.

هزینه پیچیدگی زمانی برای مرحله آزمایش برابر با $O(2d.\log n)$ است. $2d$ هزینه مقایسه داده آزمایش با فرزند چپ و فرزند راست گره جاری است و $\log n$ هزینه پیمایش درخت به ازای هر داده می‌باشد که با توجه به مجموعه داده‌های انتخابی و با در نظر گرفتن $n \gg d$، هزینه مرحله آزمایش برابر با $O(\log n)$ است که بسیار سریع می‌باشد.

٤- آزمایش‌ها

برای انجام آزمایش‌ها و ارزیابی کار ارائه‌شده از سیستمی با پردازنده AMD Ryzen ۵ ۳۴۰۰G به همراه ۱۶ گیگابایت حافظه استفاده گردیده و برنامه شرح داده شده به زبان جاوا نوشته شده است. البته به این دلیل که باقی روش‌ها هم برای مجموعه داده HIGGS با این مقدار حافظه قابل پیاده‌سازی نبودند، برای این مجموعه داده خاص، برای تمام روش‌های مقایسه‌شده از ۳۲ گیگابایت حافظه استفاده گردیده است. همچنین برای اجرا از حالت تک‌نخی استفاده شده که راهکار ارائه‌شده هم در زمان محاسبه میانگین و به دست آوردن دورترین نقاط و تخصیص داده‌های محلی به فرزندان و هم در زمان ورود و آزمایش داده جدید، قابلیت اجرا به صورت چندنخی را دارد که باعث بهبود سرعت می‌شود.

در این مقاله از مجموعه داده‌های مختلف استفاده شده که اندازه آنها از ۵۰۰۰ تا ۱۱ میلیون رکورد و ابعاد آنها از ۱۸ تا ۷۸۴ ویژگی[1] متغیر بوده که مشخصات آنها در جدول ۳ آمده است. همچنین در آزمایش‌ها به جز مجموعه داده‌های MNIST و Physical Unclonable Functions که به صورت مجموعه آموزش و آزمایش از پیش مشخص شده در مخزن UCI قرار داده شده‌اند، برای باقی مجموعه داده‌ها از اعتبارسنجی متقابل ۵تایی[2] برای آزمایش روش‌ها استفاده شده و از نتایج، میانگین گرفته شده است. همچنین به دلیل این که این دو مجموعه داده به صورت نرمال‌شده در دسترس هستند، تنها یک بار مورد آزمایش قرار گرفته‌اند. نیز در روش MDT PCA و MDT RND طبق پیشنهاد نویسندگان مقاله ارائه‌شده برای هر قسمت از $\delta = ۱,۰/۹,۰/۸,۰/۷,۰/۶$ استفاده گردیده و میانگین گرفته شده است. برای مجموعه داده‌های بزرگ‌تر مانند HIGGS از مقدار $\delta = ۰/۶$ و SUSY و Physical Unclonable Functions از میانگین نتایج حاصل از $\delta = ۰/۷,۰/۶$ استفاده شده است.

لازم به ذکر است که برای رعایت عدالت و مقایسه منصفانه کارها، کد مقالات مربوط از نویسندگان مقالات دریافت شده و در شرایط یکسان و روی مجموعه داده‌های یکسان اجرا گردیده و نتایج به دست آمده گزارش شده است. برای نمایش و بیان این امر که کار مربوط هم روی مجموعه داده نرمال‌شده و هم روی مجموعه داده غیر نرمال بهتر نتیجه می‌دهد، نتایج روی هر دو حالت مجموعه داده‌ها آزمایش گردیده است.

عملکرد روش پیشنهادی با جدیدترین روش‌های ارائه‌شده برای طبقه‌بندی داده‌های بزرگ که شامل MDT RND، MDT PCA و FPBST می‌باشند، مقایسه شده است. از مزایای کار ارائه‌شده نسبت به FPBST می‌توان به نکاتی که در ادامه بیان می‌شود توجه کرد. در روش ارائه‌شده در مقاله FPBST از روش تپه‌نوردی استفاده کرده و به صورت حریصانه به دنبال دو دورترین نقطه در مجموعه داده می‌گردد. این کار با توجه به حریصانه‌بودن ممکن است در اجراهای متفاوت در دام قله محلی گرفتار شده و دو دورترین نقطه در اجراهای متفاوت، مقادیر متغیری بگیرند. به همین جهت در اجراهای متفاوت، کار ارائه‌شده دقت‌های متفاوتی دارد. از آنجا که این مقادیر متغیر هستند پس نمی‌توان به عنوان یک روش قابل اتکا در امر طبقه‌بندی از آن استفاده کرد. اما برخلاف آن، در روش ارائه‌شده در این مقاله دو دورترین داده در اجراهای متفاوت مقادیر ثابتی دارند و به ازای اجراهای متفاوت، دقت تغییر نمی‌کند که این یک جنبه مثبت کار ارائه‌شده است. نکته دیگری که در مورد روش ارائه‌شده در مقاله FPBST وجود دارد، این است که با انتخاب داده‌های پرت به عنوان دو دورترین نقطه، باعث زیادشدن عمق درخت می‌شود و در نتیجه زمان اجرا افزایش می‌یابد به طوری که اگر در هر مرحله، این انتخاب غلط صورت گیرد، درخت به شکل مورب رشد کرده و دچار پیچیدگی نمایی می‌گردد. اما در روش ارائه‌شده در این مقاله با داشتن مقدار میانگین و محاسبه فاصله دو نقطه یافته‌شده به عنوان دورترین نقاط از نقطه میانگین، می‌توان با بررسی عدم توازن فاصله‌ها این داده‌های پرت را تشخیص داده و به دنبال دو دورترین نقطه بعدی رفت که این یک جنبه مثبت دیگر کار ارائه‌شده به نسبت روش FPBST است. همچنین برتری دیگر روش ارائه‌شده این است که در هر گره داخلی درخت با سه بار پیمایش داده‌های محلی دو دورترین نقطه را می‌یابد، در صورتی که روش FPBST طبق آنچه در مقاله مربوط ذکر شده است، به صورت

1. Attribute
2. 5-Fold-Cross-Validation

جدول ۳: مشخصات مجموعه داده‌های استفاده‌شده در آزمایش‌ها.

نام مجموعه داده	تعداد رکورد	نوع داده	اندازه مجموعه تست	اندازه مجموعه آموزش	تعداد ویژگی	تعداد کلاس
HIGGS	۱۱۰۰۰۰۰۰	حقیقی	۲۲۰۰۰۰۰	۸۸۰۰۰۰۰	۲۸	۲
Physical Unclonable Functions	۶۰۰۰۰۰۰	صحیح	۱۰۰۰۰۰۰	۵۰۰۰۰۰۰	۱۲۸	۲
SUSY	۵۰۰۰۰۰۰	حقیقی	۱۰۰۰۰۰۰	۴۰۰۰۰۰۰	۱۸	۲
covtype	۵۸۱۰۱۲	صحیح	۱۱۶۲۰۳	۴۶۴۸۰۹	۵۴	۷
mnist	۷۰۰۰۰	صحیح	۱۰۰۰۰	۶۰۰۰۰	۷۸۴	۱۰
letter	۲۰۰۰۰	حقیقی	۴۰۰۰	۱۶۰۰۰	۱۶	۲۶
satimage	۶۴۳۰	صحیح	۱۲۸۶	۵۱۴۴	۸۶	۶
waveform	۵۰۰۰	حقیقی	۱۰۰۰	۴۰۰۰	۲۱	۳

جدول ۴: متوسط زمان اجرا برای مرحله آموزش بر حسب میلی‌ثانیه (مقادیر توپر نشان‌دهنده بهترین نتایج به ازای هر مجموعه داده است).

نام مجموعه داده	MDT RND [۳۶]	MDT PCA [۳۶]	FPBST [۱۶]	روش ارائه‌شده همراه با بافر همپوشانی	روش ارائه‌شده بدون بافر همپوشانی
covtype	۲۸۲۵۳	۲۳۷۳۵	۶۳۰۸	۱۵۰۹۹	۵۶۴۸
Normalized Covtype	۶۷۳۶۵۵	۴۴۵۶۱۸	۸۹۷۵	۱۷۰۱۰	۸۰۵۹
HIGGS	۹۸۲۲۸۴۴	۴۳۱۹۷۱۱	۳۹۸۰۵۴	۴۳۴۸۶۸	۳۷۳۷۴۶
Normalized HIGGS	۱۳۴۸۹۰۰۷	۷۳۶۵۱۵۱	۳۸۶۴۲۱	۴۰۰۴۸۶	۳۵۸۰۴۱
letter	۳۸۳۶	۳۶۱۶	۳۴۴	۷۰۶	۲۹۴
Normalized letter	۳۹۱۶	۳۷۰۲	۳۳۱	۶۱۲	۳۲۵
SUSY	۳۸۴۷۶۴	۲۴۹۳۵۸	۷۴۴۹۹	۱۲۰۱۲۱	۶۸۵۵۵
Normalized SUSY	۲۶۵۶۲۰۰	۲۴۲۹۰۴۰	۸۰۴۵۸	۱۳۳۳۷۴	۷۷۷۶۵
waveform	۳۲۲	۴۲۹	۲۰۸	۴۶	۱۷۵
Normalized waveform	۳۵۸	۵۱۰	۲۹۱	۶۲	۲۵۶
Mnist	۵۵۶۲۵۱	۱۱۹۵۶۸	۸۷۵۲	۲۱۶۵۷	۸۵۸۰
satimage	۳۹۴	۵۴۷	۲۶۹	۶۶۵	۲۶۶
Normalized satimage	۳۹۷	۵۶۴	۲۷۶	۵۴۰	۲۴۴
Physical Unclonable Functions	۳۱۱۶۲۵۲۷	۳۴۵۵۹۲۸۸	۱۲۴۰۸۵	۱۵۶۷۸۳	۱۲۳۶۶۲

میانگین به ۴ تا ۷ بار پیمایش نیاز دارد و این مقدار حتی می‌تواند بیشتر هم باشد. بنابراین در روش ارائه‌شده، اغلب اوقات هم زمان مرحله آموزش و هم تعداد سطوح درخت کاهش می‌یابد که نتیجه آن، اجرای سریع‌تر و دقیق‌تر مرحله آزمایش است. همچنین از آنجا که مؤلفان در [۱۶] روش FPBST را بیان کرده‌اند و برای مقایسه کارشان کارهای MDT RND و MDT PCA را انتخاب کرده‌اند و دلایل برتری کارشان نسبت به این کارها توضیح داده شده است، در این کار نیز برای مقایسه، این کارها انتخاب شده‌اند.

در ادامه و در جدول ۴، متوسط زمان اجرای مرحله آموزش بر حسب میلی‌ثانیه برای روش‌های مورد مقایسه نمایش داده شده است. نتایج نشان می‌دهند که روش ارائه‌شده در این مقاله که در ستون‌های ۵ و ۶ جدول مذکور قرار دارد به نسبت روش‌های MDT PCA و MDT RND سرعت خیلی بیشتری دارد. همچنین بر اساس توضیحات داده‌شده در همین بخش، روش ارائه‌شده نسبت به روش FPBST نیز سریع‌تر است. در این جدول به عنوان مثال در مورد مجموعه داده HIGGS که با ۱۱ میلیون رکورد بیشترین داده را دارا می‌باشد، همان طور که مشخص است روش ارائه‌شده بدون بافر همپوشانی بهترین نتیجه را دارد. همچنین در مورد مجموعه داده Mnist که بیشترین تعداد ویژگی را دارا است، طبق نتایج حاصل‌شده، روش ارائه‌گردیده بدون بافر همپوشانی به دلایل توضیح داده شده بهترین زمان را ثبت کرده است. در جدول ۵ نیز متوسط زمان اجرای مرحله آزمایش بر حسب میلی‌ثانیه نمایش داده شده است. همان طور که در جدول مشخص است در بیشتر آزمایش‌ها با مجموعه داده‌های مختلف، روش ارائه‌شده بدون بافر همپوشانی بهترین زمان را ثبت کرده است. در بعضی از مجموعه داده‌ها روش ارائه‌شده با بافر همپوشانی بهترین زمان را داشته و این زمان خوب با وجود دقت بسیار بالای این روش می‌باشد. به عنوان نمونه در حالت نرمال، کار ارائه‌شده بدون بافر همپوشانی هم در مورد مجموعه داده HIGGS و Mnist که به ترتیب دارای بیشترین نمونه و بیشترین ویژگی هستند، بهترین زمان را ثبت کرده است. در جدول ۶ متوسط دقت به ازای هر مجموعه داده نمایش داده شده است. همان طور که در جدول آمده است، در همه مجموعه داده‌ها به جز SUSY و Physical Unclonable Functions روش ارائه‌شده همراه با بافر همپوشانی، بهترین دقت را دارد. در این دو مجموعه داده نیز زمان بسیار کمتر بوده و البته میزان دقت، بسیار نزدیک به بهترین دقت به دست آمده می‌باشد. این دقت‌ها برای هر مجموعه داده از میانگین ۵ اجرای انجام‌شده محاسبه گردیده است. برای آن که این کارها به صورت کلی قابل قیاس باشند، در ردیف آخر جدول ۶ میانگین دقت به ازای هر روش محاسبه شده است.

۵- نتیجه‌گیری

روش نزدیک‌ترین همسایگی به دلیل قابل درک بودن و سادگی مفهوم و همچنین دقت بالا و محدوده خطای پایین، به صورت گسترده‌ای در کارهای طبقه‌بندی با ناظر مورد استفاده قرار می‌گیرد. از سوی دیگر به

جدول ۵: متوسط زمان اجرا برای مرحله آزمایش بر حسب میلی‌ثانیه (مقادیر توپر نشان‌دهنده بهترین نتایج به ازای هر مجموعه داده است).

نام مجموعه داده	[۳۶] MDT RND	[۳۶] MDT PCA	[۱۶] FPBST	روش ارائه‌شده همراه با بافر همپوشانی	روش ارائه‌شده بدون بافر همپوشانی
Covtype	۲۵۴۵۶	۲۴۷۸۸	۱۳۶۰	۱۷۰۰	۱۱۸۲
Normalized Covtype	۷۳۸۵	۵۴۷۲	۱۶۹۹	۲۰۰۰	۱۷۱۵
HIGGS	۱۷۵۰۰۸	۱۵۸۰۲۹	۳۴۲۷۱۴	۱۳۰۰۶۳	۳۵۵۶۳۷
Normalized HIGGS	۱۹۵۵۹۳	۱۶۱۲۲۹	۱۹۱۶۳۰	۱۳۱۱۹۳	۱۱۸۵۱۱
Letter	۱۸۸	۱۱۲	۳۱	۴۵	۳۱
Normalized letter	۱۹۱	۱۰۹	۳۱	۴۰	۳۱
SUSY	۱۵۸۹۷۲	۱۵۰۵۸۰	۱۸۵۱۰	۳۳۷۶۶	۱۸۰۲۱
Normalized SUSY	۲۰۴۱۱۹	۱۷۲۷۴۷	۲۲۱۱۶	۳۷۷۲۳	۲۱۰۲۴
Waveform	۷۹	۸۵	۱۶	۵	۱۵
Normalized waveform	۱۱۲	۷۴	۲۸	۱۵	۱۵
Mnist	۹۰۱۵	۷۰۸۳	۹۸۵	۱۷۱۸	۹۸۵
Satimage	۱۳۹	۱۳۶	۲۵	۱۸	۲۰
Normalized satimage	۱۴۲	۱۳۱	۲۷	۱۲	۱۵
Physical Unclonable Functions	۱۵۷۴۷۳	۱۵۵۷۱۲	۶۲۰۴۷	۷۹۹۷۷	۷۳۷۹۴

جدول ۶: متوسط دقت آزمایش‌های انجام‌شده به صورت درصد (مقادیر توپر نشان‌دهنده بهترین نتایج به ازای هر مجموعه داده است).

نام مجموعه داده	[۳۶] MDT RND	[۳۶] MDT PCA	[۱۶] FPBST	روش ارائه‌شده همراه با بافر همپوشانی	روش ارائه‌شده بدون بافر همپوشانی
Covtype	٪۴۵٫۷۰۲	٪۴۴٫۲۶۴	٪۵۷٫۸۳۰۱	٪۶۴٫۳۴	٪۵۸٫۵۴۰۳
Normalized Covtype	٪۶۲٫۵۶	٪۵۸٫۵۹۸	٪۶۱٫۸۵۵	٪۶۷٫۰۷	٪۶۲٫۳۶۸
HIGGS	٪۵۸٫۲۶۲	٪۵۹٫۷۰۴	٪۵۸٫۱۸	٪۶۰٫۸۹۲۴۲	٪۵۸٫۲۰۷۸
Normalized HIGGS	٪۵۷٫۳۹۸	٪۵۸٫۸۳۲	٪۵۷٫۲۳۳۲	٪۵۸٫۸۹۸۸	٪۵۸٫۶۱۸۳
Letter	٪۵۹٫۹۵	٪۷۵٫۴۱	٪۷۷٫۳۴	٪۹۵٫۲۷	٪۷۸٫۵۲
Normalized letter	٪۵۹٫۹۶۸	٪۷۵٫۴۰۶	٪۷۷٫۳۹	٪۹۵٫۲۷	٪۷۸٫۵۲
SUSY	٪۶۸٫۷۳۸	٪۷۲٫۴۷۶	٪۷۰٫۹۲۹۴۶	٪۷۱٫۷۸۶۵	٪۷۰٫۹۶۸۸
Normalized SUSY	٪۶۶٫۳۳	٪۶۷٫۲۰۸	٪۶۶٫۷۵۹۱	٪۶۸٫۸۷۷۲	٪۶۶٫۸۸۱۲
Waveform	٪۶۶٫۳۰۸	٪۷۵٫۷۶	٪۷۶٫۲۴	٪۷۸٫۹۸	٪۷۷٫۳۶
Normalized waveform	٪۶۵٫۶۶	٪۷۵٫۷۴	٪۷۵٫۰۶	٪۷۸٫۵۴	٪۷۵٫۴۲
Mnist	٪۵۲٫۳۴	٪۸۳٫۳	٪۸۴٫۸۶	٪۹۵٫۵۳	٪۸۴٫۹۲
Satimage	٪۷۶٫۳۰۶	٪۸۱٫۶۹	٪۸۵٫۳۱۶	٪۹۰٫۸۶	٪۸۵٫۵۵۴
Normalized satimage	٪۷۵٫۸۱۶	٪۸۰٫۴۵۴	٪۸۵٫۶	٪۹۰٫۴۱	٪۸۵٫۸۸
Physical Unclonable Functions	٪۴۹٫۹۳	٪۵۰٫۰	٪۴۹٫۹۳۴۱	٪۴۹٫۹۹۷۳	٪۴۹٫۹۶۹۷
میانگین	٪۶۱٫۳۸۰۶	٪۶۸٫۴۸۸۷	٪۷۰٫۳۲۳۴	٪۷۶٫۱۹۴۵	٪۷۰٫۸۳۷۸

این دلیل که روش نزدیک‌ترین همسایگی برای طبقه‌بندی از داده‌ها، مدل ایجاد نمی‌کند، در مورد داده‌های حجیم با ورود هر داده آزمایش باید فاصله داده با تمام مجموعه داده محاسبه شود، در نتیجه این روش سرعت بسیار پایینی دارد. در کار ارائه‌شده در این مقاله دو روش جدید توضیح داده شد که یکی از کارها برای سرعت‌بخشیدن به طبقه‌بندی داده‌ها همراه با دقت عالی مناسب است. این کار با استفاده از ساخت درخت دودویی موجب سرعت‌بخشیدن به جستجوی نزدیک‌ترین همسایگی می‌شود. همچنین این روش با تشخیص فاصله نامتوازن فرزندان یافته‌شده گره موجود از میانگین داده‌ها در هر مرحله، باعث تشخیص داده‌های پرت می‌شود که باعث عدم گسترش عمق درخت گردیده و مانع از مورب‌شدن درخت تشکیل‌شده خواهد شد. تشخیص داده‌های پرت باعث افزایش سرعت مرحله آزمایش می‌شود. در کار دیگر از بافر همپوشانی استفاده شده که باعث دقت بسیار بالای کار ارائه‌شده در کنار زمان پیش‌بینی مطلوب می‌شود. بافر همپوشانی باعث شد که دقت کار برای نقاط نزدیک به خط تقسیم داده‌ها در هر گره بالا رود. برخلاف روش FPBST که در اجراهای متفاوت، دقت‌های متفاوت ارائه می‌کند، این روش به ازای اجراهای متفاوت، دقت یکسانی را ارائه می‌دهد. این روش با توجه به مقایسات بالا از روش MDT RND و MDT PCA سرعت خیلی بهتر و نزدیک به سرعت FPBST دارد. به همین ترتیب مقایسات انجام‌شده نشان می‌دهند که کار ارائه‌شده، دقت بالاتری نسبت به روش‌های جدید موجود دارد، در نتیجه این کار برای زمانی که دقت بالا مورد نیاز است مناسب می‌باشد.

مراجع

[1] X. Lv, "The big data impact and application study on the like ecosystem construction of open internet of things," *Cluster Computing*, vol. 22, no. 2, pp. 3563-3572, Mar. 2019.

[2] V. Marx, *Biology: the Big Challenges of Big Data*, Ed: Nature Publishing Group, 2013.

using k-nearest neighbor joins," *J. of Big Data*, vol. 5, no. 1, pp. 1-27, Feb. 2018.
[24] S. Bagui, A. K. Mondal, and S. Bagui, "Improving the performance of kNN in the mapreduce framework using locality sensitive hashing," *International J. of Distributed Systems and Technologies*, vol. 10, no. 4, pp. 1-16, Oct. 2019.
[25] Z. Yong, L. Youwen, and X. Shixiong, "An improved KNN text classification algorithm based on clustering," *J. of Computers*, vol. 4, no. 3, pp. 230-237, Mar. 2009.
[26] H. Neeb and C. Kurrus, *Distributed k-Nearest Neighbors*, Ed: Stanford University Publishing: Stanford, CA, USA, 2016.
[27] Y. Ben-Haim and E. Tom-Tov, "A streaming parallel decision tree algorithm," *J. of Machine Learning Research*, vol. 11, no. 2, pp. 849-872, Feb. 2010.
[28] D. Xia, H. Li, B. Wang, Y. Li, and Z. Zhang, "A map reduce-based nearest neighbor approach for big-data-driven traffic flow prediction," *IEEE Access*, vol. 4, pp. 2920-2934, Jun. 2016.
[29] F. Angiulli, "Fast condensed nearest neighbor rule," in *Proc. of the 22nd Int. Conf. on Machine learning*, pp. 25-32, New York, NY, USA, 7-11Aug. 2005.
[30] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," *Neurocomputing*, vol. 195, no. 100, pp. 143-148, Jun. 2016.
[31] T. Seidl and H. P. Kriegel, "Optimal multi-step k-nearest neighbor search," *SIGMOD Rec.*, vol. 27, no. 2, pp. 154-165, Jun. 1998.
[32] L. R. Lu and H. Y. Fa, "A density-based method for reducing the amount of training data in kNN text classification," *J. of Computer Research and Development*, vol. 45, no. 4, pp. 539–544, Aug. 2004.
[33] A. J. Gallego, J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan, "Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation," *Pattern Recognition*, vol. 74, no. 1, pp. 531-543, Feb. 2018.
[34] A. J. Gallego, J. R. Rico-Juan, and J. J. Valero-Mas, "Efficient k-nearest neighbor search based on clustering and adaptive k values," *Pattern Recognition*, vol. 122, Article ID: 108356, Feb. 2022.
[35] S. Ougiaroglou and G. Evangelidis, "RHC: a non-parametric cluster-based data reduction for efficient k kNN classification," *Pattern Analysis and Applications*, vol. 19, no. 1, pp. 93-109, Feb. 2016.
[36] F. Wang, Q. Wang, F. Nie, W. Yu, and R. Wang, "Efficient tree classifiers for large scale datasets," *Neurocomputing*, vol. 284, no. 2, pp. 70-79, Apr. 2018.
[37] A. Hassanat, "Greedy algorithms for approximating the diameter of machine learning datasets in multidimensional euclidean space: experimental results," 2018.
[38] A. B. Hassanat, "Two-point-based binary search trees for accelerating big data classification using KNN," *PloS One*, vol. 13, no. 11, Article ID: e0207772, Nov. 2018.
[39] T. Liu, A. W. Moore, K. Yang, and A. G. Gray, "An investigation of practical approximate nearest neighbor algorithms," in *Proc. 17th Int. Conf. on Information Processing Systems*, pp. 825-832, Vancouver, Canada,13-18 Dec. 2004.

حسین کلاته مدرک کارشناسی خود را در سال ۱۳۹۵ از دانشگاه سمنان اخذ نموده است. نام‌برده همچنین مدرک کارشناسی ارشد خود را در سال ۱۳۹۹ از دانشگاه تربیت دبیر شهید رجایی تهران دریافت نموده است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: داده کاوی و طبقه‌بندی داده‌های حجیم.

نگین دانشپور مدرک کارشناسی خود را در سال ۱۳۷۷ از دانشگاه شهید بهشتی اخذ نموده است. همچنین نام‌برده درجهٔ کارشناسی ارشد و دکتری خود را از دانشگاه صنعتی امیرکبیر در سال‌های ۱۳۸۰ و ۱۳۸۹ اخذ نموده است. در حال حاضر، ایشان عضو هیأت علمی و دانشیار دانشکده مهندسی کامپیوتر دانشگاه تربیت دبیر شهید رجایی است. زمینه‌های پژوهشی مورد علاقه ایشان عبارتند از: تحلیل و مدیریت داده، پایگاه داده تحلیلی ، داده کاوی و پیش‌پردازش داده.

[3] I. Triguero, J. Maillo, J. Luengo, S. García, and F. Herrera, "From big data to smart data with the k-nearest neighbours algorithm," in *Proc. IEEE Int. Conf. on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 859-864, Chengdu, China, 15- 18 Dec. 2016.
[4] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37-66, Jan. 1991.
[5] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. on Information Theory*, vol. 13, no. 1, pp. 21-27, Jan. 1967.
[6] D. W. Aha, *Lazy Learning*, in Lazy learning: Kluwer Academic Publishers, pp. 7-10, 1997.
[7] P. P. Anchalia and K. Roy, "The k-nearest neighbor algorithm using mapreduce paradigm," in *Proc. 5th IEEE Int. Conf. on Intelligent Systems, Modelling and Simulation,* pp. 513-518, Langkawi, Malaysia, 27- 9 Jan. 2014.
[8] H. Saadatfar, S. Khosravi, J. H. Joloudari, A. Mosavi, and S. Shamshirband, "A new k-nearest neighbors classifier for big data based on efficient data pruning," *Mathematics*, vol. 8, no. 2, p. 286, Feb. 2020.
[9] A. Hassanat, "Norm-based binary search trees for speeding up knn big data classification," *Computers*, vol. 7, no. 4, pp. 5, Oct. 2018.
[10] D. Zhu, "Humor robot and humor generation method based on big data search through IOT," *Cluster Computing*, vol. 22, no. 4, pp. 9169-9175, Jul. 2019.
[11] J. Calvo-Zaragoza, J. J. Valero-Mas, and J. R. Rico-Juan, "Improving kNN multi-label classification in prototype selection scenarios using class proposals," *Pattern Recognition*, vol. 48, no. 5, pp. 1608-1622, May 2015.
[12] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97-107, Jun. 2013.
[13] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, Springer, 2015.
[14] L. Micó and J. Oncina, "A constant average time algorithm to allow insertions in the LAESA fast nearest neighbour search index," in *Proc. 20th IEEE Int. Conf. on Pattern Recognition,* pp. 3911-3914, Istanbul, Turkey, 23- 26 Aug. 2010.
[15] A. Bifet, *et al.*, "Extremely fast decision tree mining for evolving data streams," in *Proc. of the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 1733-1742, Halifax, Canada, 13-17 Aug. 2017.
[16] A. B. Hassanat, "Furthest-pair-based binary search tree for speeding big data classification using k-nearest neighbors," *Big Data*, vol. 6, no. 3, pp. 225-235, Sept 2018.
[17] X. Wu, *et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1-37, Jan. 2008.
[18] E. Plaku and L. E. Kavraki, "Distributed computation of the kNN graph for large high-dimensional point sets," *J. of Parallel and Distributed Computing*, vol. 67, no. 3, pp. 346-359, Mar. 2007.
[19] C. Zhang, F. Li, and J. Jestes, "Efficient parallel kNN joins for large data in mapreduce," in *Proc. of the 15th Int. Conf. on Extending Database Technology*, pp. 38-49, Berlin, Germany, 27-30 Mar. 2012.
[20] K. Sun, H. Kang, and H. H. Park, "Tagging and classifying facial images in cloud environments based on KNN using mapreduce," *Optik*, vol. 126, no. 21, pp. 3227-3233, Nov. 2015.
[21] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera, "kNN-IS: an iterative spark-based design of the k-nearest neighbors classifier for big data," *Knowledge-Based Systems*, vol. 117, no. 1, pp. 3-15, Feb. 2017.
[22] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, J. M. Benítez, and F. Herrera, "Nearest neighbor classification for high-speed big data streams using spark," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2727-2739, Jul. 2017.
[23] G. Chatzigeorgakidis, S. Karagiorgou, S. Athanasiou, and S. Skiadopoulos, "FML-kNN: scalable machine learning on big data