مقاله پژوهشی

ساخت درخت تصمیم مقیاس‌پذیر مبتنی بر تقسیم سریع داده‌ها و پیش‌هرس

سمیه لطفی، محمد قاسم‌زاده، مهران محسن‌زاده و میترا میرزارضایی

چکیده: دسته‌بندی، یکی از وظایف مهم داده‌کاوی و یادگیری ماشین است و درخت تصمیم به عنوان یکی از الگوریتم‌های پرکاربرد دسته‌بندی، دارای سادگی و قابلیت تفسیر نتایج است. اما در مواجهه با داده‌های حجیم، درخت تصمیم بسیار پیچیده خواهد شد و با محدودیت‌های حافظه و زمان اجرا مواجه است. الگوریتم‌های ساخت درخت باید همه مجموعه داده آموزش و یا بخش زیادی از آن را درون حافظه نگه دارند. الگوریتم‌هایی که به علت انتخاب زیرمجموعه‌ای از داده با محدودیت حافظه مواجه نیستند، زمان اضافی جهت انتخاب داده صرف می‌کنند. جهت انتخاب بهترین ویژگی برای ایجاد انشعاب در درخت هم باید محاسبات زیادی بر روی این مجموعه داده انجام شود. در این مقاله، یک رویکرد مقیاس‌پذیر افزایشی بر مبنای تقسیم سریع و هرس، جهت ساخت درخت تصمیم بر روی مجموعه داده‌های حجیم ارائه شده است. الگوریتم ارائه‌شده درخت تصمیم را با استفاده از کل مجموعه داده آموزش اما بدون نیاز به ذخیره‌سازی داده در حافظه اصلی می‌سازد. همچنین جهت کاهش پیچیدگی درخت از روش پیش‌هرس استفاده شده است. نتایج حاصل از اجرای الگوریتم بر روی مجموعه داده‌های UCI نشان می‌دهد الگوریتم ارائه‌شده با وجود دقت و زمان ساخت قابل رقابت با سایر الگوریتم‌ها، بر مشکلات حاصل از پیچیدگی درخت غلبه کرده است.

***کلیدواژه:* پیش‌هرس، داده‌کاوی، درخت تصمیم، مقیاس‌پذیر.**

۱- مقدمه

داده‌کاوی، وظیفه اکتشاف، تجزیه و تحلیل حجم زیادی از داده‌ها را به منظور استخراج قوانین و الگوها به عهده دارد [۱]. دسته‌بندی به عنوان یکی از مهم‌ترین وظایف داده‌کاوی شامل یافتن یک تابع یا مدل بوده که برچسب کلاس نمونه‌های مختلف را بر حسب ویژگی‌های آنها مشخص می‌کند. تا کنون تکنیک‌های مختلفی مانند درخت تصمیم، شبکه‌های عصبی و دسته‌بند بیز برای دسته‌بندی ارائه شده است. درخت تصمیم به علت قابلیت تفسیر و نمایش قوانین در قالب سلسله‌مراتبی پرکاربردترین الگوریتم دسته‌بندی است. همچنین این الگوریتم به دلیل سادگی و قابلیت توصیف، هزینه و زمان ساخت مناسب، توانایی کار با داده‌های پیوسته و گسسته، عدم نیاز به دانش قبلی و ارائه دقت مناسب، یکی از الگوریتم‌های پرکاربرد در حوزه تشخیص الگو و یادگیری ماشین است. هر درخت از مؤلفه‌های ریشه، گره‌های میانی، گره‌های برگ و یال‌ها تشکیل شده است. گره‌های میانی دارای پارامترهایی جهت تصمیم‌گیری و انشعاب هستند و هر گره یک یا چند فرزند دارد. برای هر یک از فرزندان پارامتر تصمیم‌گیری دارای مقداری است که مسیر طی‌شده در پیمایش درخت را نشان می‌دهد. به هر گره برگ یک برچسب انتساب داده می‌شود که بیان‌کننده کلاس است. بعد از ساخت درخت تصمیم، جهت دسته‌بندی نمونه جدید، پیمایش درخت از ریشه شروع شده و تا رسیدن به یک برگ ادامه می‌یابد. در نهایت برچسب آن برگ به عنوان کلاس نمونه مورد نظر برگردانده می‌شود [۲].

عمومی‌ترین الگوریتم‌های ساخت درخت تصمیم C۴.۵، ID۳ و CART است اما با توجه به این که داده‌های موجود در پایگاه داده‌ها اغلب شامل تعداد زیادی ویژگی و رکورد هستند، این روش‌ها در مواجهه با این داده‌ها قابل استفاده نیستند [۳]. دلیل ناکارآمدی درخت تصمیم در مواجهه با مجموعه داده‌های حجیم، معیار انتخاب بهترین ویژگی جهت انشعاب گره‌ها است. معیارهای سنتی جهت تعیین بهترین ویژگی نیازمند انجام محاسبات بر روی کل داده‌ها هستند. برخی از روش‌های موجود جهت انتخاب ویژگی‌های عددی از روش‌های گسسته‌سازی و برخی دیگر از ارزیابی‌های پرهزینه استفاده نموده‌اند که هیچ یک از این روش‌ها برای مواجهه با مجموعه داده‌های حجیم مناسب نیستند.

الگوریتم‌های مختلفی جهت غلبه بر این چالش توسعه یافته‌اند که هر یک از آنها نیز نیازمند حافظه و یا زمان زیادی هستند. از ساده‌ترین روش‌های مقابله با داده‌های حجیم می‌توان نمونه‌برداری، روش‌های کاهش ابعاد، تجمیع، پردازش موازی و روش‌های افزایشی را نام برد. برخی از این الگوریتم‌ها، تمام مجموعه داده را به حافظه می‌آورند و برخی دیگر بخشی از داده‌های آموزش را به داخل حافظه آورده و بر مشکل محدودیت حافظه غلبه می‌کنند. اما برای انتخاب زیرمجموعه‌ای از داده‌ها مجبور به تحمل هزینه زمانی و محاسباتی هستند. در برخی دیگر از روش‌ها، ساخت درخت تصمیم به صورت افزایشی و با استفاده از تمام داده‌های آموزش انجام می‌شود. در این روش‌ها داده‌ها به ترتیب در ساخت درخت مشارکت می‌کنند و معمولاً برای داده‌هایی از نوع جریان کاربرد دارند. این دسته از روش‌ها برای مواجهه با داده‌های زیاد، مناسب نیستند [۴]. اخیراً الگوریتمی جهت ساخت درخت تصمیم با تمرکز بر روی زمان ساخت ارائه شده به نحوی که حداکثر دقت را داشته باشد. نقطه قوت این الگوریتم ساخت درخت در مدت زمان محدود است. ایده اصلی کار این است که وقتی زمان کافی برای ساخت درخت وجود دارد، ویژگی‌ای را

این مقاله در تاریخ ۲ دی ماه ۱۳۹۹ دریافت و در تاریخ ۱ فروردین ماه ۱۴۰۰ بازنگری شد.

سمیه لطفی، دانشکده مهندسی کامپیوتر، واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران، (email: lsomayeh@gmail.com).

محمد قاسم‌زاده (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران، (email: m. ghasemzadeh@yazd.ac.ir).

مهران محسن‌زاده، دانشکده مهندسی کامپیوتر، واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران، (email: mohsenzadeh@srbiau.ac.ir).

میترا میرزارضایی، دانشکده مهندسی کامپیوتر، واحد علوم و تحقیقات، دانشگاه آزاد اسلامی، تهران، ایران، (email: mirzarezaee@srbiau.ac.ir).

برای انشعاب انتخاب کند که بیشترین سود را داشته باشد و وقتی زمان محدود است کاراترین ویژگی را از نظر زمان انتخاب کند. اما این روش بر روی داده حجیم بررسی نشده و همچنین به پیچیدگی درخت حاصل توجهی نشده است [۵].

جهت غلبه بر مشکلات ذکرشده، در این مقاله الگوریتم مقیاس‌پذیر افزایشی بر مبنای تقسیم سریع و هرس درخت، ارائه شده است. با ساخت درخت تصمیم از کل داده‌های آموزش به نحوی که جهت ساخت درخت نیازی به ذخیره تمام داده‌ها در حافظه نباشد و همچنین حذف رکوردهای استفاده‌شده بعد از توسعه یک گره، از اتلاف زمان و حافظه جلوگیری خواهد شد. با استفاده از پیش‌هرس $J_{\max}$ [۶] از افزایش پیچیدگی درخت جلوگیری شده و با توجه به این که از تمام داده‌های آموزش جهت ساخت درخت استفاده شده است، قابلیت اعتماد مدل افزایش می‌یابد.

در ادامه ساختار مقاله به این شرح است: در قسمت مروری بر سابقه تحقیق، روش‌های ارائه‌شده برای مواجهه با مجموعه داده‌های حجیم از دیدگاه‌های مختلف بررسی گردیده است. در بخش ۳ به شرح عملکرد الگوریتم پیشنهادی پرداخته شده و در نهایت، تحلیل و نتیجه‌گیری در بخش‌های ۴ و ۵ آمده است.

۲- مروری بر سوابق تحقیق

مقیاس‌پذیری یکی از چالش‌های مهم هر الگوریتمی است یعنی الگوریتم بتواند بر مشکل حجم زیاد داده‌ها غلبه نماید. روش‌های زیادی تا کنون برای حل این مشکل توسعه یافته‌اند که برخی از این روش‌ها با محدودیت دو عامل زمان و حافظه مواجه بوده‌اند. با توجه به مطالعات انجام‌شده در داده‌های حجیم، در یک دسته‌بندی کلی می‌توان روش‌های ارائه‌شده در این حوزه را در سه رویکرد: ۱) نمونه‌برداری، ۲) لیست‌های داده‌ای و ۳) روش‌های افزایشی، دسته‌بندی نمود. این دسته‌بندی حاصل تحقیقات انجام‌شده در این مطالعه بوده و بدین معنا نیست که روش دیگری به جز رویکردهای بیان‌شده وجود ندارد. در این بخش مروری بر سوابق مربوط به برخی از روش‌های پیشین ارائه‌شده مبتنی بر هر یک از رویکردهای بیان‌شده فوق در حوزه مجموعه داده‌های حجیم بررسی می‌گردد.

۲-۱ رویکرد نمونه‌برداری

در این رویکرد ابتدا نمونه‌هایی از مجموعه داده اصلی، انتخاب و سپس الگوریتم ساخت درخت بر روی نمونه انتخاب‌شده اعمال می‌شود. از جمله در الگوریتم‌های ICE و BOAT که برای جلوگیری از ذخیره کل داده درون حافظه اصلی، داده‌های آموزش به بخش‌هایی تقسیم می‌شوند. برای هر بخش با استفاده از الگوریتم‌های سنتی (مانند ID۳، C۴.۵ و CART) یک درخت تصمیم ساخته می‌شود. هر یک از درخت‌های تصمیم هر بخش به صورت مجزا مورد پردازش قرار می‌گیرد و یا با هم ترکیب می‌شوند. بدین ترتیب دیگر نیازی به قرارگرفتن تمام داده‌ها در حافظه نیست. انتخاب این نمونه‌ها یا به صورتی تصادفی صورت می‌گیرد یا از الگوریتم‌های خاصی برای انتخاب داده بهره می‌گیرند. از مزایای این روش انعطاف‌پذیری بالا در برخورد با افزایش یا کاهش داده‌ها در پارتیشن‌ها است، اما به دلیل عدم استفاده از کل داده‌ها در این روش، قابلیت اعتماد مدل پایین است. عمدتاً زمان الگوریتم نمونه‌برداری، وابستگی نتایج به تکنیک نمونه‌برداری و نیاز به زمان زیاد جهت ساخت درخت برای مجموعه داده‌های مختلف از چالش‌های این روش به حساب می‌آید [۷] و [۸].

۲-۲ رویکرد لیست‌های داده‌ای

در این رویکرد جهت جلوگیری از نگهداری تمام داده‌ها در حافظه اصلی، ساختارهای لیست ایجاد و عمدتاً در حافظه دیسک نگهداری می‌شوند. برای تقسیم و توسعه به جای استفاده از رکوردها از این لیست استفاده می‌شود. روش SLIQ[1] از جمله این روش‌هاست که در آن به ازای هر ویژگی یک ساختار لیست نگهداری می‌شود [۹]. در هر لیست، کلاس رکوردها و تعداد گره‌های مرتبط با هر رکورد نگهداری می‌شوند. در مرحله تقسیم و توسعه به جای استفاده از رکوردها از این لیست استفاده می‌شود به نحوی که بزرگی این لیست وابسته به تعداد رکوردها است. اساس این روش C۴.۵ است. همچنین در این الگوریتم داده‌های پیوسته به صورت صعودی و داده‌های گسسته بر اساس شناسه مرتب می‌شوند. به طور کلی هدف از ارائه این رویکرد، جلوگیری از افزونگی بالای مرتب‌سازی و کاهش هزینه‌ها در فاز ارزیابی انشعابات بدون از دست دادن دقت است. از مزایای این روش افزایش سرعت ساخت درخت با حفظ دقت آن بوده و از معایب آن اتلاف حافظه است.

روش SPRINT[2] از دیگر الگوریتم‌های مبتنی بر لیست است که در نحوه نگهداری لیست با روش SLIQ تفاوت دارد. این روش نیازی به ذخیره در حافظه اصلی ندارد ولی برای هر بار توسعه باید تمام لیست از حافظه ثانویه خوانده شود. پردازش در لیست ویژگی‌ها به صورت چندپردازنده‌ای و موازی انجام می‌شود و در هر دو روش بالا، نیازمند فضایی به اندازه دو برابر داده‌های آموزشی هستیم [۱۰].

روش Rainforest نیز برای نمایش ویژگی‌ها از لیست استفاده می‌کند ولی تنها مقادیر متفاوت ویژگی‌ها را نگهداری می‌نماید. از این طریق تعداد رکوردها کاهش پیدا کرده اما لیست باید در حافظه نگهداری شود. در صورت متنوع‌بودن مقادیر یک متغیر، اندازه لیست بالا می‌رود. در هر بار ساخت لیست باید دو بار تمام داده‌ها خوانده شده و یک بار هم نوشته شوند که این مورد برای داده‌هایی با حجم بالا مناسب نیست [۱۱].

۲-۳ رویکرد افزایشی

در روش‌های افزایشی، داده‌ها به ترتیب در ساخت درخت مشارکت کرده و درخت تصمیم از تمام داده‌ها ساخته می‌شود [۱۲]. در الگوریتم VFDT[3]، درخت تصمیم در زمان ثابت و مستقل از تعداد نمونه‌ها ساخته می‌شود. این الگوریتم جهت ساخت درخت اولیه، نیازمند رکوردهایی است که به صورت تصادفی انتخاب شده باشند. سپس تمام رکوردها به صورت افزایشی به درخت وارد شده و در آن پیمایش می‌شوند. هنگامی که تعداد رکوردهای موجود در یک برگ به تعداد مشخصی رسید، معیار Information Gain برای آن محاسبه می‌شود. این الگوریتم باید برای تمام ویژگی‌های عددی، تمام حالت‌های مختلف تقسیم را محاسبه نماید که این مسئله در حالتی که داده‌ها تنوع زیادی داشته باشند، بسیار زمان‌بر است [۱۳].

روش DTFS[4] در جهت عدم قرارگیری داده‌های آموزشی در حافظه اصلی و همچنین حل مشکل سربار محاسباتی، داده‌های آموزش را به صورت افزایشی به درخت تزریق می‌کند. در این روش هر رکورد در درخت پیمایش می‌شود و در برگ‌ها ذخیره می‌گردد. سپس در صورتی که

1. Supervised Learning in Quest
2. Scalable Parallelizable Induction of Decision Trees
3. Very Fast Decision Trees
4. Decision Tree Using Fast Splitting

جدول ۱: مقایسه رویکردهای ساخت درخت تصمیم از مجموعه داده‌های حجیم.

رویکرد	مزایا	معایب
نمونه‌برداری	ساخت درخت با دو بار اسکن داده‌ها عدم نگهداری کل داده‌ها درون حافظه	استفاده از زیرمجموعه کوچکی از داده‌ها اتلاف زمان به دلیل انتخاب داده‌ها و انتخاب تکنیک مناسب پایین‌بودن دقت مدل ساخته‌شده
لیست داده‌ای	ارائه روشی در ذخیره‌سازی غلبه بر کاهش سربار حافظه کاهش زمان انتخاب	نیاز به فضای اضافی جهت نگهداری لیست‌های تعریفی هزینه زمانی ساخت لیست
افزایشی	سرعت بالا عدم نگهداری کل داده در حافظه و سربار فضای حافظه	نیاز به پیش‌پردازش پیچیدگی درخت کاهش سادگی و قابلیت تفسیر درخت نیاز به تنظیم پارامترها

تعداد رکوردهای موجود در یک برگ از تعداد مشخص‌شده توسط کاربر بیشتر شود، الگوریتم در مورد توسعه و یا به روز رسانی برگ تصمیم می‌گیرد. در این الگوریتم همانند روش C۴.۵ از معیار Gain Ratio برای انتخاب بهترین معیار انشعاب استفاده شده است با این تفاوت که تنها s رکورد ذخیره‌شده در گره‌ای که باید توسعه یابد، مورد بررسی قرار می‌گیرند. انتخاب ویژگی‌های انشعاب از میان مجموعه‌ای با s رکورد موجب تسریع در روند اجرای الگوریتم می‌شود [۱۴].

با توجه به مطالعات انجام‌شده، ویژگی‌های هر دسته از الگوریتم‌های فوق را می‌توان در جدول ۱ مشاهده نمود.

۳- روش پیشنهادی

همان طور که بیان شد، در این مقاله الگوریتم جدیدی برای ساخت درخت تصمیم مبتنی بر روش افزایشی و الگوریتم DTFS ارائه شده که علاوه بر غلبه بر چالش داده‌های زیاد، توانسته است به توازنی بین دقت و پیچیدگی دست یابد. در این الگوریتم سعی شده وابستگی بین دقت و پیچیدگی درخت تصمیم را با پارامترهای الگوریتم به حداقل برسانیم. در درخت تصمیم، در مرحله تعیین بهترین ویژگی جهت توسعه، باید هم‌زمان تمام داده‌های در حافظه موجود باشند. با افزایش تعداد رکوردها، ممکن است محدودیت حافظه مانع از محاسبه بهترین ویژگی شود. همچنین در داده‌های زیاد و ساخت درخت بزرگ، چالش از دست رفتن قابلیت تفسیر و افزایش خطا وجود دارد. در روش پیشنهادی روی هر دو چالش مطرح‌شده، تمرکز گردیده است.

اصول و خطوط روش پیشنهادی را می‌توان به صورت زیر بیان نمود:

۳-۱ تعیین اولویت ورود رکوردها به درخت

در مرحله اول، جهت تعیین اولویت ورودی رکوردها به درخت از معیارهای شباهت (با توجه به نوع داده‌های ورودی) استفاده شده است. به کمک این معیار، میزان شباهت هر رکورد با سایر رکوردها محاسبه شده و یک عدد بیانگر شباهت با کل رکوردها به دست می‌آید. سپس رکوردها بر اساس میزان شباهت به صورت نزولی مرتب می‌شوند. پس در نتیجه، داده‌ای در ابتدا قرار می‌گیرد که دارای بیشترین شباهت با سایرین بوده و در انتها نیز رکوردی قرار می‌گیرد که کمترین شباهت را با بقیه دارد. با این رویکرد، در مراحل ابتدایی ساخت درخت، از رکوردهایی استفاده خواهد شد که دارای کیفیت بهتری هستند.

۳-۲ انتخاب بهترین ویژگی جهت انشعاب، تصمیم‌گیری در مورد وقوع و تعیین شرایط توسعه

یکی از مسایل اصلی در ساخت درخت تصمیم، انتخاب ویژگی جهت انشعاب است. انشعاب در هر گره بسته به نوع داده‌ای مقادیر، می‌تواند به صورت دودویی یا چندگانه باشد. اگر مقادیر ویژگی‌ها به صورت پیوسته باشد، نوع انشعاب در بیشتر مواقع دودویی بوده و اگر مقادیر ویژگی گسسته باشد، انشعاب ممکن است به صورت دودویی یا چندگانه باشد. جهت انتخاب بهترین ویژگی در تولید هر گره، باید به میزان خلوص و توزیع همگن کلاس در هر دسته توجه نمود. جهت تشخیص درجه خلوص و انتخاب ویژگی مناسب برای گسترش گره، در الگوریتم پیشنهادی از (۱) تا (۴) استفاده شده است.

آنتروپی، میزان خلوص مجموعه‌ای از داده‌ها را مشخص می‌کند. این ویژگی آماری نشان‌دهنده کیفیت تقسیم‌کنندگی مثال‌های آموزشی توسط یک ویژگی است. اگر ویژگی هدف c مقادیر مختلفی داشته باشد که احتمال پیشامد هر یک به صورت P_i باشد، آنتروپی I نسبت به این دسته‌بندی به صورت زیر تعریف می‌شود

$$Entropy(I) = \sum_{i=1}^{c} -p_i \log_{2} p_i \qquad (۱)$$

با داشتن بی‌نظمی به عنوان ناخالصی در یک مجموعه مثال آموزشی، می‌توان معیار مؤثربودن یک صفت در دسته‌بندی داده‌های آموزشی را تعریف نمود. این معیار، کاهش مورد انتظار در بی‌نظمی است که با تفکیک‌کردن مثال‌ها بر پایه این صفت حاصل می‌شود.

بهره اطلاعات یک ویژگی عبارت است از مقدار کاهش آنتروپی که به واسطه جداسازی مثال‌ها از طریق این ویژگی حاصل می‌شود. بهره اطلاعاتی یک ویژگی A مربوط به مجموعه مثال‌های I در (۲) نشان داده شده است. جمله اول مقدار آنتروپی داده‌ها و عبارت دوم مقدار آنتروپی مورد انتظار بعد از جداسازی داده‌هاست. $values(A)$ مجموعه تمام مقادیر ممکن برای صفت A است. در (۲)، I_v زیرمجموعه‌ای از I است که در آن صفت A مقدار v را دارد

$$Gain(I, A) = Entropy(I) - \sum_{v \in values(A)} \frac{|I_v|}{|I|} Entropy(I_V) \qquad (۲)$$

عدم قطعیت نمونه‌ها نسبت به ویژگی A از (۳) به دست می‌آید

$$SplitInfo(A) = -\sum_{j=1}^{c} \frac{\#classj}{I} \log_{2} \frac{\#classj}{I} \qquad (۳)$$

رابطه نرمال‌شده (۴) در محاسبه نسبت بهره اطلاعاتی استفاده می‌شود. خاصیت Gain Ratio نشان‌دهنده این است که یک ویژگی با چه گستردگی و یکنواختی داده‌ها را جدا می‌سازد. مخرج کسر باعث می‌شود تا ویژگی‌هایی که مقادیر زیادی با توزیع یکنواخت دارند حذف شوند [۱۵] و [۱۶]

$$GainRatio(A) = \frac{Gain(I, A)}{SplitInfo(A)} \quad (۴)$$

به علت این که برای محاسبه Gain Ratio در هر انشعاب باید برای تمام رکوردها، حالت‌های موجود محاسبه و از بین آنها ویژگی‌ای با بیشترین مقدار نتیجه انتخاب شود، محاسبه این معیار از لحاظ زمانی بسیار پرهزینه است.

با توجه به زمان‌بربودن محاسبه این معیار، در الگوریتم پیشنهادی تنها از رکوردهای ذخیره‌شده در همان گره‌ای که تصمیم به انشعابش گرفته‌ایم در محاسبه و انتخاب بهترین ویژگی استفاده می‌کنیم. برای متغیرهای دسته‌ای به ازای هر یک از مقادیر و برای متغیرهای پیوسته، میانگین مقادیر ظاهرشده برای تمام رکوردهای در راستای یک کلاس، به عنوان معیار انشعاب محاسبه می‌شود. در نتیجه با توجه به اعداد به دست آمده از محاسبه Gain Ratio بهترین ویژگی را انتخاب می‌کنیم. انتخاب ویژگی‌های انشعاب از میان رکوردهای درون یک گره موجب تسریع در روند انشعاب می‌شود.

۳-۳ کاهش پیچیدگی در برگ‌های مستعد توسعه توسط معیار $J_{\max}$

یکی از روش‌های پیش‌هرس استفاده از معیار j است. معیار j به عنوان ابزار تئوری اطلاعات برای اندازه‌گیری محتوای قوانین استخراج‌شده از درخت، استفاده می‌شود. با فرض این که فرم قوانین استخراج‌شده از درخت به صورت if $Y = y$ then $X = x$ باشد، ارزش اطلاعات محتوای قوانین با استفاده از (۵) محاسبه می‌گردد [۱۷]

$$J(X;Y=y) = p(x)j(X;Y=y) \quad (۵)$$

که $p(y)$ احتمال این که مقدم قانون اتفاق بیفتد و $j(X;Y=y)$، معیار j ای است که باید توسط (۶) محاسبه شود

$$j(X;Y=y) = p(x|y)\log_2(\frac{p(x|y)}{p(x)}) + (1-p(x|y))\log_2(\frac{1-p(x|y)}{1-p(x)}) \quad (۶)$$

روش هرس زودهنگام هرس j بر اساس معیار اندازه‌گیری j با هدف کاهش رخداد پدیده جادهی بیش از اندازه ارائه شده است. مزیت این روش هرس کاهش تعداد قوانین یا گره‌های درخت با حفظ دقت قابل قبول می‌باشد. مهم‌ترین عیب روش هرس j این است که این روش ممکن است دچار بهینگی محلی گردد. دلیل این امر آن است که ارزش گره‌ای که به دلیل کمتربودن ارزش j آن از پدرش هرس شود، ممکن است در انشعاب‌های بعدی افزایش یابد. جهت غلبه بر این مشکل مذکور رویکردی به نام $J_{\max}$ ارائه شده است. در این روش برای هر گره علاوه بر J_{value} با استفاده از (۷) مقدار $J_{\max}$ آن گره نیز محاسبه می‌گردد. اگر ارزش j یک گره کمتر از ارزش j پدرش بود به ارزش $J_{\max}$ آن نگاه می‌شود. اگر ارزش $J_{\max}$ آن گره بیشتر از J_{value}های قبلی آن باشد، رشد درخت ادامه می‌یابد زیرا مقدار J_{value} ممکن است دوباره افزایش یابد. این رشد تا زمانی که ارزش J_{value} و $J_{\max}$ برابر شود، ادامه می‌یابد. فرمول محاسبه ارزش $J_{\max}$ در (۷) مشاهده می‌شود [۶]

$$J_{\max} = p(y)\max\{p(x|y)\log_2(\frac{1}{p(x)}), (1-p(x|y))\log_2(\frac{1}{1-p(x)})\} \quad (۷)$$

۴-۳ الگوریتم ساخت درخت

ساخت درخت با یک گره خالی به نام ریشه آغاز می‌شود. در ابتدا ریشه درخت یک برگ هم هست. رکوردهای آموزشی به ترتیب به ریشه وارد می‌شوند و درخت را پیمایش کرده تا به یک برگ ختم شوند و در برگ ذخیره می‌گردند. بعد از این که تعداد رکوردهای ذخیره‌شده در یک برگ به حداکثر تعداد مجاز خود رسید، آن گاه یکی از شرایط زیر برقرار می‌شود:

۱) تمام رکوردهای ذخیره‌شده در راستای یک کلاس باشند.

۲) رکوردهای موجود در این برگ از کلاس‌های مختلفی باشند.

در حالت اول برگ توسعه نمی‌یابد و فقط لبه ورودی به آن به روز شده و رکوردهای ذخیره‌شده در این برگ، حذف می‌شوند.

در حالت دوم بعد از انتخاب ویژگی مناسب، به ازای هر یک از مقادیر آن صفت باید لبه‌ای تشکیل شود. به ازای هر یک از لبه‌های احتمالی در برگ مستعد توسعه، مقادیر J و $J_{\max}$ با استفاده از (۶) و (۷) محاسبه و شرایط ایجاد گره جدید به صورت زیر بررسی می‌شود:

۱) اگر مقدار j یک گره بیشتر از مقدار j پدرش باشد، توسعه گره انجام می‌شود.

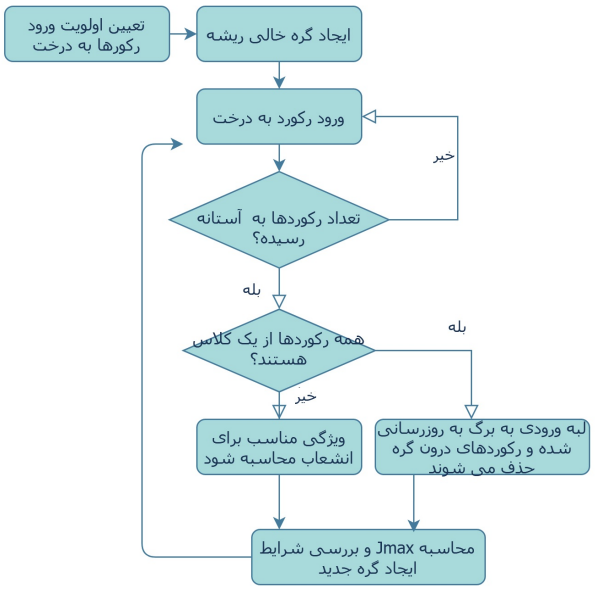
۲) اگر ارزش j گره کمتر از ارزش j پدرش باشد، مقدار $J_{\max}$ آن گره بررسی می‌شود. اگر مقدار $J_{\max}$ آن گره بیشتر از J_{value}های قبلی آن باشد، توسعه گره درخت انجام می‌شود.

۳) اگر ارزش j گره با ارزش $J_{\max}$ گره برابر باشد، توسعه گره متوقف می‌شود.

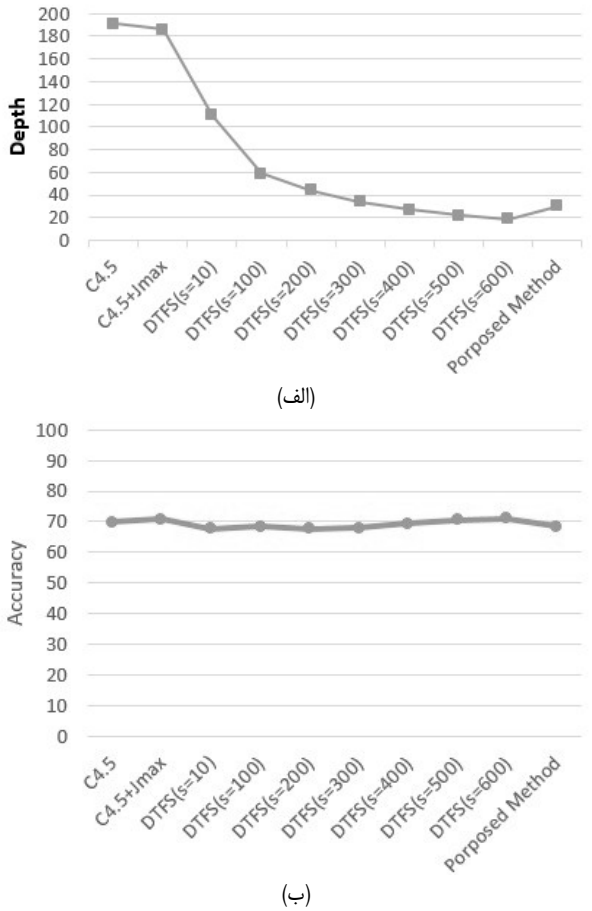
رکوردهای مربوط به هر لبه درون برگ ایجادشده برای آن لبه ذخیره می‌شوند و میانگین مقادیر در راستای آن کلاس به عنوان برچسب لبه انتساب داده می‌شود. گره برگ سطح قبلی به یک گره میانی تبدیل می‌شود و رکوردهای ذخیره‌شده در این گره، حذف خواهند شد. فاز استنتاج بعد از آن که تمام رکوردها پیمایش و پردازش شدند به اتمام می‌رسد. سپس به تمامی برگ‌ها کلاس اکثریت را به عنوان برچسب می‌زنیم. در صورتی که گرهی تهی باشد اکثریت کلاس پدر داده می‌شود. روندنمای روش پیشنهادی در شکل ۱ نشان داده شده است. بعد از تعیین اولویت ورود رکوردها و ایجاد گره ریشه، بقیه مراحل به ازای همه رکوردهای آموزشی انجام می‌شود.

۴- نتایج آزمایش

در این بخش، کارایی روش پیشنهادی بررسی شده و با الگوریتم‌های C۴.۵ و DTFS از نظر دقت، زمان اجرای الگوریتم و پیچیدگی درخت تصمیم مورد ارزیابی و مقایسه قرار می‌گیرد. جهت اجرای روش پیشنهادی و سایر الگوریتم‌های درخت تصمیم از سیستمی با مشخصات RAM ۶ گیگابایت، پردازنده ۲٫۱۰ GHz و سیستم عامل Win۱۰ ۶۴ بیتی استفاده شده است. در حین مقایسه الگوریتم‌ها سعی بر آن بود تا تمام شرایط اجرا یکسان باشد. مجموعه داده‌های استفاده‌شده جهت آزمایش در جدول ۲ نشان داده شده‌اند که از UCI گرفته شده و هیچ یک

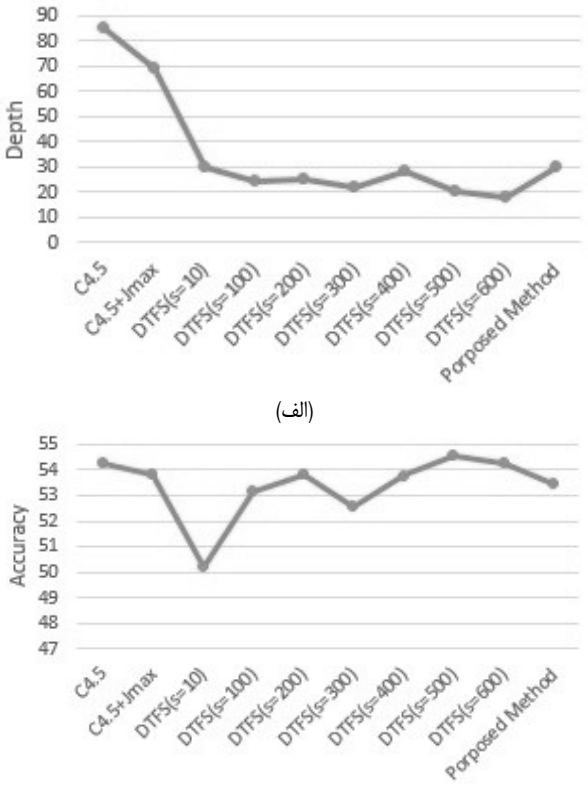


شکل ۱: روندنمای الگوریتم پیشنهادی.



شکل ۲: (الف) عمق درخت تصمیم و (ب) دقت در درخت‌های تصمیم ساخته‌شده برای مجموعه داده MagicGamma.

Missing value ندارد. جهت تقسیم داده‌ها از روش Hold out استفاده شده است به نحوی که در هر بار اجرا ۷۰ درصد داده‌ها برای آموزش و ۳۰ درصد مجموعه داده برای ارزیابی در نظر گرفته شده‌اند. سپس برای تأیید صحت نتایج روش Hold out را ۱۰ بار تکرار کرده و میانگین نتایج گزارش شده است. در همه آزمایش‌ها الگوریتم‌های C۴.۵، C۴.۵ با اعمال روش هرس J_{max}، DTFS با مقادیر مختلف پارامتر s و در نهایت روش پیشنهادی با یکدیگر مقایسه شده‌اند.



شکل ۳: (الف) عمق درخت تصمیم و (ب) دقت در درخت‌های تصمیم ساخته‌شده برای مجموعه داده Poker.

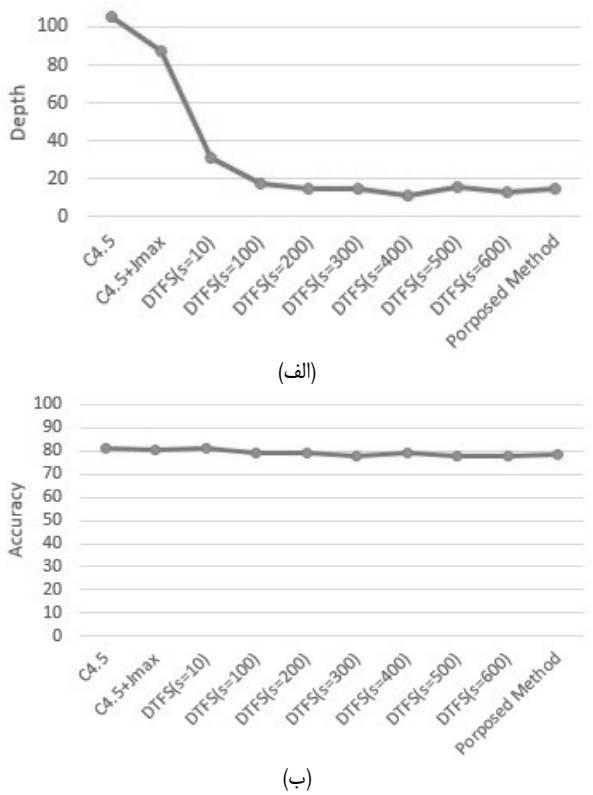
جدول ۲: مجموعه داده‌های استفاده‌شده در آزمایش‌ها.

مجموعه داده	تعداد ویژگی	تعداد رکورد
Magic Gamma	۱۱	۱۹۰۲۰
Statlog (Shuttle)	۹	۵۸۰۰۰
Poker	۱۱	۱۰۲۵۰۱۰

در شکل ۲ عمق درخت و دقت الگوریتم‌های ساخت درخت بر روی مجموعه داده Magic Gamma نشان داده شده است. عمق درخت یکی از معیارهای نشان‌دهنده میزان پیچیدگی درخت است. هر چه عمق درخت بیشتر باشد به همان میزان پیچیدگی درخت زیاد است. همان گونه که نتایج اجرا در شکل ۲ نشان می‌دهد، در الگوریتم DTFS به ازای مقادیر مختلف s عمق درخت تغییر کرده است. در الگوریتم DTFS اگر مقدار s را پایین در نظر بگیریم دقت کاهش می‌یابد. همچنین در الگوریتم C۴.۵ پیچیدگی درخت بالاست و پیش‌هرس هم تأثیر چندانی در کاهش پیچیدگی درخت نداشته است. الگوریتم پیشنهادی توانسته است با حفظ دقت به مقدار میانگینی در پیچیدگی دست یابد.

همان گونه که در شکل ۳ مشخص است در مجموعه داده Poker به ازای $s=۱۰$ دقت الگوریتم ۵۰ درصد است که با افزایش مقدار s دقت افزایش یافته اما این روال همیشه صادق نیست، مثلاً در $s=۳۰۰,۶۰۰$ مجدداً دقت کاهش یافته است. هرچند الگوریتم C۴.۵ بیشترین دقت خروجی را داشته است اما درخت تصمیم حاصل از آن بسیار پیچیده است. نتایج نشان می‌دهد در الگوریتم پیشنهادی به موازنه‌ای بین دقت و پیچیدگی دست یافته‌ایم.

در شکل ۴ نتایج حاصل از اجرای الگوریتم‌ها بر روی مجموعه داده Shuttle نشان داده شده است. در همه الگوریتم‌ها دقت درخت تصمیم ایجادشده حدود ۸۰ درصد است اما عمق درخت حاصل که نشان‌دهنده



شکل ۴: (الف) عمق درخت تصمیم و (ب) دقت در درخت‌های تصمیم ساخته‌شده برای مجموعه داده Shuttle.

پیچیدگی است، در الگوریتم پایه C۴.۵ بالا است که با استفاده از روش پیش‌هرس این پیچیدگی تا حدی کم شده است. استفاده از رویکرد افزایشی برای ساخت درخت در این مجموعه تا حد زیادی باعث کاهش پیچدگی درخت و نگه‌داشتن دقت در حد قابل قبول شده است.

۵- نتیجه‌گیری

در این مقاله روشی جهت ساخت درخت تصمیم بر روی مجموعه داده‌هایی با تعداد رکورد زیاد ارائه شد. در رویکرد پیشنهادی ابتدا اولویت ورود رکوردها به درخت تعیین شده و سپس مجموعه داده آموزش بر اساس اولویت‌های مشخص‌شده رکورد به رکورد به درخت تصمیم وارد شده و در برگ مناسب قرار می‌گیرند. برای هر گره در زمان مقتضی پیچیدگی آن محاسبه شده و در مورد انشعاب یا عدم انشعاب آن تصمیم‌گیری می‌شود. جهت تعیین بهترین ویژگی برای انشعاب از معیار Gain Ratio استفاده شده است. در نهایت جهت جلوگیری از پیچیدگی درخت و ایجاد توازن بین دقت و پیچیدگی، برگ‌های مستعد توسعه با استفاده از معیار $J_{\max}$ مشخص می‌شوند. در مجموع از مهم‌ترین مزایای این روش می‌توان به موارد زیر اشاره نمود: استفاده از تمام داده‌های آموزش در ساخت درخت، عدم سربار حافظه و زمان به دلیل استفاده‌نشدن از ساختمان داده‌های ویژه، هزینه زمانی مناسب و کاهش پیچیدگی درخت با استفاده از روش پیش‌هرس.

نتایج آزمایش‌ها نشان داده که در الگوریتم پیشنهادی حداکثر تعداد مجاز رکورد ذخیره‌شده در یک گره تأثیری بر روی دقت، زمان اجرا و پیچیدگی درخت حاصل ندارد. همچنین استفاده از روش پیش‌هرس باعث جلوگیری از پیچیدگی درخت شده است. در ادامه کار می‌توان معیارهای دیگری جهت تصمیم‌گیری در مورد انشعاب گره را بررسی و استفاده نمود. همچنین با افزودن محاسبات موازی می‌توان سرعت اجرای الگوریتم را بهبود بخشید.

مراجع

[1] S. Agarwal, "Data mining: data mining concepts and techniques," in *Proc. IEEE Int. Conf. on Machine Intelligence and Research Advancement,* . pp. 203-207, Katra, India, 21-23 Dec. 2013.

[2] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261-283, 2013.

[3] Z. Ruiz, J. Salvador, and J. Garcia-Rodriguez, "A survey of machine learning methods for big data," in *International Work-Conf. on the Interplay Between Natural and Artificial Computation*, pp. 259-267, Springer, Cham, Jun. 2017.

[4] S. Lomax and S. Vadera, "A survey of cost-sensitive decision tree induction algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, pp. 1-35, Feb. 2013.

[5] Y. L. Chen, C. C. Wu, and K. Tang, "Time-constrained cost-sensitive decision tree induction," *Information Sciences*, vol. 354, pp. 140-152, Mar. 2016.

[6] F. Stahl and M. Bramer, "Jmax-pruning: a facility for the information theoretic pruning of modular classification rules," *Knowledge-Based Systems*, vol. 29, pp. 12-19, 2012.

[7] S. Hwang, H. G. Yeo, and J. S. Hong, "A new splitting criterion for better interpretable trees," *IEEE Access*, vol. 8, pp. 62762-62774, 2020.

[8] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Y. Loh, "BOAT-optimistic decision tree construction," *ACM SIGMOD Record*, vol. 28, no. 2, pp. 169-180, Jun. 1999.

[9] M. Mehta, R. Agrawal, and J. Rissanen, "SLIQ: a fast scalable classifier for data mining," *Advances in Database Technology-EDBT'96*, pp. 18-32, 1996.

[10] J. Shafer, R. Agrawal, and M. Mehta, "SPRINT: a scalable parallel classifier for data mining," in *Proc. Int. Conf. Very Large Data Bases*, vol. 96, pp. 544-555, 3-6 Sept. 1996.

[11] J. Gehrke, R. Ramakrishnan, and V. Ganti, "Rainforest-a framework for fast decision tree construction of large datasets," *Data Mining and Knowledge Discovery*, vol. 4, pp. 127-162, 2000.

[12] G. Hulten and P. Domingos, "Mining decision trees from streams," in Garofalakis M., Gehrke J., Rastogi R. (eds.) *Data Stream Management. Data-Centric Systems and Applications.* pp. 189-208, Springer Berlin Heidelberg, 2016.

[13] C. C. Wu, Y. L. Chen, Y. H. Liu, and X. Y. Yang, "Decision tree induction with a constrained number of leaf nodes," *Applied Intelligence*, vol. 45, no. 3, pp. 673-685, Oct. 2016.

[14] A. Franco-Arcega, J. A. Carrasco-Ochoa, G. Sanchez-Diaz, and J. F. Martinez-Trinidad, "Decision tree induction using a fast splitting attribute selection for large datasets," *Expert Systems with Applications*, vol. 38, no. 11, pp. 14290-14300, Oct. 2011.

[15] B. Chandra, R. Kothari, and P. Paul, "A new node splitting measure for decision tree construction," *Pattern Recognition*, vol. 43, no. 8, pp. 2725-2731, Aug. 2010.

[16] P. S. Akash, M. E. Kadir, A. A. Ali, and M. Shoyaib, "Inter-node hellinger distance based decision tree," in *Proc. 28th Int. Joint Conf. on Artificial Intelligence, IJCAI'19*, pp. 1967-1973, Aug. 2019.

[17] M. Bramer, "Using J-pruning to reduce overfitting in classification trees," *Knowledge-Based Systems*, vol. 15, no. 5, pp. 301-308, Jul. 2002.

سمیه لطفی در سال ۱۳۸۲ مدرک کارشناسی مهندسی کامپیوتر (نرم‌افزار) خود را از دانشگاه آزاد اسلامی واحد نجف آباد و در سال ۱۳۸۸ مدرک کارشناسی ارشد مهندسی نرم افزار خود را از دانشگاه شیراز دریافت نمود. در حال حاضر در دوره دکتری مهندسی کامپیوتر گرایش سیستم های نرم افزاری در دانشگاه آزاد اسلامی واحد علوم و تحقیقات مشغول به تحصیل و پژوهش هستند. ایشان همچنین از سال ۱۳۹۰ تا کنون به عنوان عضو هیئت علمی دانشگاه آزاد اسلامی واحد بندرعباس مشغول به تدریس و تحقیق می باشند. زمینه‌های علمی مورد علاقه نام‌برده عبارتند از: داده‌کاوی ، یادگیری ماشین ، تحلیل و طراحی الگوریتم‌ها.

محمد قاسم‌زاده در سال ۱۳۶۸ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه شیراز و در سال ۱۳۷۴ مدرک کارشناسی ارشد مهندسی کامپیوتر (هوش ماشین و رباتیک) خود را از دانشگاه صنعتی امیرکبیر دریافت نمود. از بهمن ماه ۱۳۸۰ الی بهمن ماه ۱۳۸۴ نام‌برده جهت انجام دوره دکتری (علوم کامپیوتر) در دانشگاه تربیر و دانشگاه پتسدام هر دو در کشور آلمان مشغول به تحصیل و پژوهش بودند. ایشان در سال ۱۳۸۴ موفق به اخذ درجه دکتری علوم کامپیوتر گردید. محمد قاسم‌زاده از سال ۱۳۷۵ تا کنون به عنوان عضو هیأت علمی در دانشگاه یزد مشغول به تدریس و تحقیق می‌باشند.

زمینه‌های علمی مورد علاقه نام‌برده عبارتند از: طراحی و تحلیل الگوریتم‌ها، پردازش زبان طبیعی، سیستم‌های هوشمند و محاسبات نرم.

مهران محسن‌زاده در سال ۱۳۷۶ مدرک کارشناسی مهندسی کامپیوتر (نرم افزار) خود را از دانشگاه شهید بهشتی، در سال ۱۳۷۸ کارشناسی ارشد و در سال ۱۳۸۳ دکتری مهندسی کامپیوتر خود را از دانشگاه آزاد اسلامی واحد علوم و تحقیقات دکتری مهندسی کامپیوتر خود را دریافت نمود. ایشان هم اکنون استادیار گروه مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد علوم و تحقیقات می‌باشند. زمینه های علمی مورد علاقه نام‌برده: رایانش ابری ، مهندسی نرم افزار و داده های بزرگ است و تا کنون بیش از ۸۰ مقاله در کنفرانس ها و مجلات بین المللی منتشرکرده است.

میترا میرزارضایی استادیار مهندسی کامپیوتر دانشگاه آزاد اسلامی واحد علوم و تحقیقات ایران می‌باشند. زمینه‌های علمی مورد علاقه ایشان شامل سیستم‌های هوشمند، بیوانفورماتیک ، شناسایی الگو و یادگیری ماشین است.