ارائهٔ روشی جدید مبتنی‌بر برنامه‌نویسی ژنتیک برای وزن‌دهی قواعد فازی در طبقه‌بندی نامتوازن

محبوبه مهدی‌زاده و مهدی‌افتخاری
بخش مهندسی کامپیوتر، دانشکدهٔ فنی و مهندسی، دانشگاه شهید باهنر کرمان، کرمان، ایران

چکیده

در زمینهٔ مسائل طبقه‌بندی، اغلب با طبقه‌هایی مواجه می‌شویم که تعداد نمونه‌های متفاوتی دارند؛ یعنی کلاس‌هایی با تعداد نمونهٔ زیاد و کلاس‌هایی با تعداد نمونهٔ کم؛ این مسائل «مسائل طبقه‌بندی با مجموعه‌داده‌های نامتوازن» نامیده می‌شوند. سامانه‌های طبقه‌بندی مبتنی‌بر قوانین فازی(FRBCSs) یکی از رایج‌ترین سامانه‌های مدل‌سازی فازیِ استفاده‌شده، برای حل مسائل طبقه‌بندی است. وزن‌دهیِ قوانین اغلب برای بهبود دقت طبقه‌بندی استفاده می‌شود و به‌طور معمول نسخه‌های فازیِ *confidence* و *support* برای تولید وزن قوانین فازی بکار می‌روند. در این مقاله، یک روش تکاملی بر مبنای برنامه‌نویسی ژنتیک برای تولید عبارات مربوط به وزن ارائه می‌شود. برای تولید عبارات از چهار معیار *confidence*، *support*، *lift* و *recall* به‌عنوان پایانه‌های برنامه‌نویسی ژنتیک استفاده می‌کنیم. آزمایش را بر روی بیست مجموعه‌داده از مجموعه‌داده‌های keel اجرا و سپس نتایج به‌دست آمده را با استفاده از آزمون‌های آماری تحلیل می‌کنیم. نتایج حاصل، نشان می‌دهد که کارایی FRBCS با استفاده از روش پیشنهادی بهبود می‌یابد.

واژگان کلیدی: مسائل با مجموعه‌داده‌های نامتوازن، سامانه‌های طبقه‌بندی مبتنی بر قوانین فازی، وزن‌دهی قوانین، برنامه‌نویسی ژنتیک.

۱- مقدّمه

تکنیک‌های یادگیری ماشین در بسیاری از حوزه‌های جهان واقعی مانند اینترنت، مطالعات تجاری و علمی، صنعت و غیره کاربرد دارند. یکی از مهم‌ترین مسائل در زمینه داده‌کاوی و یادگیری ماشین، مسأله کلاس نامتوازن[1] است (Lopez et al., 2013). مسألهٔ دادهٔ نامتوازن زمانی رخ می‌دهد که نمونه‌های یک یا چند کلاس ذاتاً نادرند و یا به‌سختی جمع‌آوری می‌شوند؛ بنابراین مسألهٔ کلاس نامتوازن بسیار حائز اهمیت است زیرا به‌طور ضمنی در اکثر کاربردهای واقعی مانند تشخیص کلاهبرداری، مدیریت ریسک، تحقیقات پزشکی، تشخیص تولد زودرس و غیره مشاهده می‌شود (Williams et al., 2009; Sun et al., 2009).

مسألهٔ طبقه‌بندی نامتوازنِ باینری بدین صورت تعریف می‌شود: یک مسألهٔ طبقه‌بندی است که در آن تفاوت قابل توجهی میان میزان نمونه‌های دو طبقه وجود دارد. طبقهٔ مثبت یا اقلیت به طبقه‌ای گفته می‌شود که به‌طور معمول از دیدگاه یادگیری بیش‌ترین علاقه و توجه را دارد و درصورتی‌که نادرست طبقه‌بندی شود، منجر به هزینهٔ بیش‌تری می‌شود؛ درحالی‌که طبقه با تعداد نمونه‌های بیش‌تر طبقهٔ اکثریت یا طبقهٔ منفی نامیده می‌شود.

اغلب الگوریتم‌های یادگیری ماشین فرض می‌کنند تعداد نمونه‌های آموزشی در طبقه‌های متفاوت برابرند و براین اساس، یک طبقه‌بند را یاد می‌گیرند. بنابراین زمانی‌که این الگوریتم‌ها را به داده‌های نامتوازن اعمال می‌کنیم، طبقه‌بند یادگرفته‌شده بیش‌تر از طبقهٔ اکثریت منتج می‌شود که این موضوع به پیش‌بینی بسیار ضعیف از طبقهٔ اقلیت

1- Class imbalanced

منجر می‌شود؛ زیرا آموزش طبقهٔ اقلیت به‌درستی انجام نشده است. به‌دلیل عدم کارایی الگوریتم‌های یادگیری استاندارد در مواجهه با مسائل نامتوازن، روش‌های متفاوتی برای حل این نوع مسائل ارائه شده است که به دو دستهٔ اصلی تقسیم می‌شوند: نمونه‌گیری از داده‌ها[1] و اصلاح الگوریتم‌های موجود.

هدف روش‌های نمونه‌گیری، توازن در توزیع داده‌ها با متناسب کردن نمونه‌های موجود در طبقه‌هاست (Yu et al., 2013; Yen and Lee, 2009) ؛ در این روش‌ها پس از متوازن کردن مجموعه‌داده‌های آموزشی، می‌توان طبقه‌بندهای[2] استاندارد را برای انجام طبقه‌بندی به‌کار برد. در دستهٔ دوم، روش‌های سطح الگوریتم سعی دارند با استفاده از بهبود الگوریتم‌های موجود و یا با ارائهٔ روش‌های جدید، مسائل طبقه‌بندی نامتوازن را حل کرده و سبب می‌شوند الگوریتم‌های یادگیری با موضوع عدم‌توازن طبقه‌ها هماهنگی بیش‌تری پیدا کنند (Galar et al., 2012).

در این مقاله بر روی دستهٔ دوم تمرکز می‌کنیم و الگوریتمی ارائه می‌دهیم که با مسائل نامتوازن سازگار باشد. برای دستیابی به این هدف از سامانه‌های طبقه‌بندی مبتنی بر قوانین فازی[3] استفاده می‌کنیم. سامانه‌های طبقه‌بندی مبتنی بر قوانین فازی یکی از رایج‌ترین سامانه‌های مدل‌سازی فازیِ استفاده شده، برای حل مسائل طبقه‌بندی هستند و وزن‌دهی به قوانین اغلب برای بهبود دقت طبقه‌بندی استفاده می‌شود. در روش پیشنهادی برای تولید وزن قوانین از الگوریتم تکاملی برنامه‌نویسی ژنتیک استفاده می‌کنیم و معادلات تولیدشده با این روش را به‌عنوان فرمولی برای محاسبه وزن قوانین به‌کار می‌بریم.

در روش پیشنهادی، بر روی مسائل طبقه‌بندی باینری تمرکز کرده و از مجموعه دادهای keel[4] برای انجام آزمایش‌ها استفاده می‌کنیم. برای بررسی نتایج نیز معیار *AUC* را به‌کار می‌بریم که کارایی الگوریتم پیشنهادی را اندازه‌گیری می‌کند (Tahir et al., 2012).

۲- مجموعه داده‌های نامتوازن در مسائل طبقه‌بندی

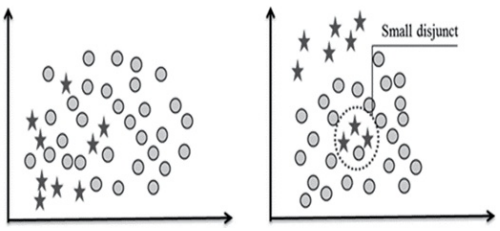
در این بخش، ابتدا مسألهٔ مجموعه‌داده‌های نامتوازن را معرفی و سپس معیارهای ارزیابی برای این نوع از مسائلِ طبقه‌بندی را ارائه می‌کنیم.

۲-۱- مسألهٔ مجموعه‌داده‌های نامتوازن

در مسائل طبقه‌بندی، تعداد نمونه‌های هر کلاس می‌تواند متفاوت باشد. در مسائل طبقه‌بندی باینری، مسألهٔ عدم توازن هنگامی رخ می‌دهد که یک طبقه، تعداد نمونه‌های بسیاری دارد؛ درحالی‌که طبقهٔ دیگر توسط تعداد نمونه‌های کمی نمایش داده می‌شود. این مسأله، بسیار حائز اهمیت است؛ زیرا به‌طور ضمنی در اکثر کاربردهای واقعی (He & Garcia, 2009) ازجمله مخابرات، تجارت، پزشکی و غیره وجود دارد. در این مسائل، نرخ عدم توازن[5] متفاوت است. نرخ عدم توازن یا *IR* برابر با نسبت تعداد نمونه‌های طبقهٔ اکثریت به تعداد نمونه‌های طبقهٔ اقلیت است.

الگوریتم‌های یادگیری ماشین با این فرضیه که «عدم توازن میان طبقه‌ها مانع اصلی در داشتن طبقه‌بندهای مناسب است» موافق می‌باشند. با این‌حال در بعضی از حوزه‌ها با وجود نامتوازن بودن طبقه‌ها، الگوریتم‌های یادگیری ماشین قادرند طبقه‌بندهای خوبی را حتی با افزایش عدم توازن ایجاد کنند. این موضوع نشان می‌دهد که عدم توازن به‌تنهایی دلیل کاهش کارایی الگوریتم‌های یادگیری نیست.

نتایج آزمایش‌ها نشان می‌دهد که مشکل کاهش کارایی به میزان هم‌پوشانی[6] داده‌های طبقه‌ها نیز بستگی دارد (Lopez et al., 2012). موضوع مهم دیگر در این نوع مسائل، وجود شکاف‌های کوچک[7] در داده‌هاست که اغلب الگوریتم‌های یادگیری در تشخیص این نواحی دچار مشکل می‌شوند. در شکل (۱-۱) این موارد نشان داده شده است (Lopez et al., 2012; Fern´andez et al., 2009).



(شکل ۱-۱): مثالی از عدم توازن میان طبقه‌ها. شکاف کوچک (شکل سمت راست) و هم‌پوشانی میان طبقه‌ها (شکل سمت چپ)

1- Data sampling
2- Classifiers
3 - Fuzzy Rule Based Classification Systems(FRBCS)
4- Keel dataset repository
5- Imbalanced Ratio (*IR*)
6- Overlapping
7- Small disjunct

۲-۲- معیارهای ارزیابی در مسائل نامتوازن

معیارهای ارزیابی نقشی اساسی در مسائل طبقه‌بندی ایفا می‌کنند. برای ارزیابی کارایی طبقه‌بندها، ساده‌ترین روش تحلیل براساس ماتریس گیجی[1] است؛ این ماتریس نمونه‌هایی را که برای هر طبقه به‌درستی و به‌اشتباه تشخیص داده شده‌اند حفظ می‌کند. رایج‌ترین معیار استفاده‌شده، دقت است:

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \qquad (۱)$$

در معادلهٔ بالا، TP و TN به‌ترتیب تعداد نمونه‌های طبقهٔ مثبت و منفی را که به‌درستی طبقه‌بندی شده‌اند نشان می‌دهند. FN تعداد نمونه‌های طبقهٔ مثبت است که به‌اشتباه منفی پیش‌بینی شده‌اند؛ درحالی‌که FP تعداد نمونه‌های منفی است که به‌اشتباه مثبت معرفی شده‌اند.

معیار دقت نمی‌تواند تعداد برچسب‌های[2] درست تشخیص داده شده با طبقه‌های متفاوت را مشخص کند و این موضوع ممکن است در چارچوب مسائل نامتوازن منجر به نتیجه‌گیری‌های اشتباه شود. برای مثال، فرض کنید بانکی قصد ایجاد یک طبقه‌بند را دارد که بتواند در آینده پیش‌بینی کند به کدام‌یک از مشتریان وام دهد. تعداد مشتریانی که شرایط وام گرفتن را ندارند تنها ۲٪ از کل مشتریان را تشکیل می‌دهند. حال اگر طبقه‌بند پیش‌بینی کند که تمام مشتریان شرایط وام گرفتن را دارند، دقت بسیار بالایی خواهد داشت؛ یعنی ۹۸٪؛ اما طبقه‌بند نمی‌تواند مشتریان بدحساب را در میان مشتریان دیگر مشخص کند. بنابراین اگر طبقه‌بندی بتواند سبب پیش‌بینی درست در طبقهٔ اقلیت شود، بسیار مناسب خواهد بود (Tong et al., 2011). به‌همین دلیل، در مسائل نامتوازن از معیارهای دیگری استفاده می‌شود. در این مقاله معیار AUC را برای ارزیابی کارایی الگوریتم‌های بیان‌شده به‌کار می‌بریم .

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \qquad (۲)$$

در معادلهٔ بالا TP_{rate} میزان نمونه‌های مثبتی را که به‌درستی مثبت پیش‌بینی شده‌اند، نسبت به کل نمونه‌های مثبت نشان می‌دهد و FP_{rate} میزان نمونه‌های منفی که به‌اشتباه مثبت پیش‌بینی شده‌اند، نسبت به کل نمونه‌های منفی را مشخص می‌کند.

1- Confusion Matrix
2- Label

۳- مفاهیم پایه

در این بخش، مفاهیمی که برای درک بهتر روش پیشنهادی نیاز می‌باشد به‌اختصار شرح داده شده است. درابتدا سامانه‌های طبقه‌بندی مبتنی بر قوانین فازی(FRBCS) را توضیح می‌دهیم؛ سپس دو روش تولید قوانین فازی را که اغلب برای ساخت پایگاه قوانین استفاده می‌شود، بیان می‌کنیم؛ و پس از آن، روش‌های تخصیص وزنی که به‌طور معمول در FRBCSها به‌کار می‌روند، بررسی می‌کنیم؛ و در انتها برنامه‌نویسی ژنتیک را شرح می‌دهیم.

۳-۱- سامانه‌های طبقه‌بندی مبتنی بر قوانین فازی

سامانه‌های طبقه‌بندی مبتنی بر قوانین فازی از تعدادی قوانین اگر- آن‌گاه تشکیل شده است که این قوانین برای مدل‌کردن رفتار ورودی-خروجی سامانه و طبقه‌بندی الگوها به‌کار می‌روند. هر مسألهٔ طبقه‌بندی شامل m الگوی آموزشی $x_p = (x_{p1},...,x_{pn}), p = 1,2,...,m$ از Mطبقه است که x_{pi}، مقدار iاُمین ویژگی ($i = 1,2,...,n$) از p اُمین الگوی آموزشی است.

$$\begin{aligned} Rule\ \ R_j &: If\ x_1\ is\ A_{j1}\ and\ ...\ a\ nd\ x_n\ is\ A_{jn}\ then \\ & Cla\ ss\ =\ C_j\ \ with\ \ R\ W_j \end{aligned} \qquad (۳)$$

که R_j برچسب jاُمین قانون و $x = (x_1,...,x_n)$ یک بردار الگوی n بعدی است، A_{ji} یک مجموعهٔ فازی مقدم[3]، C_j برچسب طبقه و RW_j وزن قانون است. ما از توابع عضویت مثلثی به‌عنوان مجموعه‌های فازی مقدم استفاده می‌کنیم (Alcala-Fdez et al., 2011).

۳-۲- مدل‌های یادگیری قوانین فازی

تولید پایگاه قوانین برای FRBCSها با روش‌های متفاوتی انجام می‌گیرد؛ اما از میان آنها دو روش بیش از سایر روش‌ها استفاده می‌شود. روش اول که توسط چای[4] پیشنهاد شده است (Chi et al., 1996)، روش ونگ-مندل[5] (Wang & Mendel, 1992) را برای مسائل طبقه‌بندی توسعه می‌دهد. دومین روش، توسط ایشیبوچی[6] استفاده شده است (Ishibuchi & Yamamoto, 2004) که در آن تمامی قوانینِ ممکن در فضای جستجوی مسأله ایجاد می‌شود.

3- Antecedent
4- Chi
5- Wang – Mendel
6- Ishibuchi

در روش اوّل (معروف به روش تولید قانون چای) ارتباط بین متغیرهای مسأله تشخیص داده می‌شود و یک وابستگی میان فضای ویژگی‌ها و فضای طبقه توسط گام‌های زیر ایجاد می‌شود:

۱. **ساخت نواحی زبانی**[1] : دامنهٔ تغییر هر ویژگی مشخص می‌شود و نواحی فازی محاسبه می‌شوند.

۲. **تولید قوانین فازی برای هر نمونه**: برای انجام این کار لازم است:

۱.۲. درجهٔ تطبیق نمونه با نواحیِ فازی متفاوت به کمک یک نرم مثلثی[2] محاسبه شود.

۲.۲. نمونهٔ مورد نظر به ناحیهٔ فازی با بیش‌ترین درجهٔ تطبیق[3] اختصاص یابد.

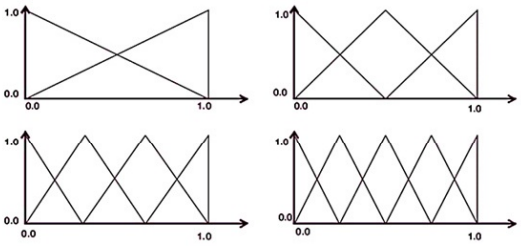
۳.۲. یک قانون برای نمونهٔ مورد نظر تولید کنیم که مقدم[4] آن توسط ناحیهٔ فازیِ انتخاب‌شده مشخص شود و تالی[5] آن، برچسب طبقه نمونه باشد.

۴.۲. وزن قانون محاسبه شود.

در روش دوم یعنی روش ایشیبوچی، به‌طور هم‌زمان از چهار ناحیهٔ فازی برای هر ویژگی استفاده می‌شود (شکل (۲-۳)). الگوریتم درابتدا تمامی ترکیب‌های ممکن از مقدم‌های مجموعه‌های فازی را می‌سازد و سپس با یک تالی ترکیب می‌کند و یک قانون را تشکیل می‌دهد. برای کاهش محاسبات از قوانین به طول کم‌تر از سه استفاده می‌شود.

تالی‌های اختصاص‌یافته به قانون، براساس بیش‌ترین مقدار اطمینان[6] به‌دست می‌آیند. قوانین با توجه به برچسب طبقه، به M گروه تقسیم می‌شوند و هر گروه براساس معیار انتخاب قوانین مرتب می‌شود. درنهایت FRBCS با انتخاب Q قانون فازی از هر گروه ساخته می‌شود. به‌طور معمول Q مقداری کوچک انتخاب می‌شود تا پیچیدگی سامانه کاهش یابد و در اغلب کاربردها مقداری بین ۱۵ تا ۴۰ دارد.

در این مقاله از روش دوم برای تولید قوانین فازی استفاده می‌کنیم.



(شکل ۲-۳): نواحی فازی تعریف‌شده توسط ایشیبوچی

۳-۳- روش‌های تخصیص وزن قوانین

برای وزن‌دهی به قوانین، روش‌های متفاوتی ارائه شده است. از میان تمامی معیارهای ارائه شده، چهار معیار بیش از سایر معیارها کاربرد دارند. این چهار فرمول وزن‌دهی که کاربرد گسترده‌تری دارند (با درنظر گرفتن قوانینی به فرم معادله (۳))، عبارتند از (Ishibuchi & Yamamoto, 2005):

- **معیار *Confidence***: بعد از تولید مجموعه قوانین، از این معیار می‌توان برای انتخاب بهترین قانون استفاده کرد. این معیار، میزان درجهٔ تطبیق نمونه‌هایی که برچسب طبقهٔ خروجی آنها با تالی قانون یکسان می‌باشد؛ نسبت به میزان درجهٔ تطبیق تمامی نمونه‌ها با قانون مورد نظر را، محاسبه می‌کند و به‌صورت زیر تعریف می‌شود:

$$conf = \sum_{X_p \in Class\ C_j} \mu_j(X_p) \Big/ \sum_{p=1}^{m} \mu_j(X_p) \qquad (۴)$$

زمانی‌که چندین قانون با *confidence* یکسان وجود دارد، می‌توان از معیار *support* استفاده کرد؛ اما به‌صورت منطقی همواره تقریب *confidence* از معیار *support* بهتر است.

- **معیار *support***: از معایب این معیار امکان حذف قوانینی است که تعداد اندکی از نمونه‌های موجود در مجموعه‌داده را پوشش می‌دهند؛ اما قوانین پراهمیتی هستند و به‌صورت زیر تعریف می‌شود:

$$supp = 1/m. \sum_{X_p \in Class\ C_j} \mu_j(X_p) \qquad (۵)$$

- **معیار *lift***: معیار *confidence* به‌تنهایی برای توصیف درجهٔ کیفیت یک قانون کافی نیست. قوانینی که دارای *confidence* بالایی هستند، ممکن است به‌صورت تصادفی یا شانسی باشند؛ چنین قوانینی زمانی به‌وجود می‌آیند که مشخص شود مقدم و تالی نسبت به یکدیگر مستقل

1- Linguistic Partitions
2- T-Norm
3- Compatibility Degree
4- Antecedent
5- Consequent
6- Confidence

هستند. یکی از معیارهایی که می‌تواند درجهٔ استقلال و یا وابستگی را مشخص کند معیار *lift* است که به‌صورت زیر تعریف می‌شود:

$$lift = \frac{\sum_{X_p \in Class\,C_j} \mu_j(X_p) \Big/ \sum_{p=1}^{m} \mu_j(X_p)}{r/m} \qquad (۶)$$

•**معیار *recall*:** این معیار، میزان درجهٔ تطبیق نمونه‌هایی که برچسب طبقهٔ خروجی آنها با تالی قانون یکسان است را نسبت به تعداد کل نمونه‌ها نشان می‌دهد و به‌صورت زیر تعریف می‌شود:

$$recall = \sum_{X_p \in Class\,C_j} \mu_j(X_p) \Big/ m \qquad (۷)$$

در معادلات (۴) تا (۶) X_p نمونهٔ آموزشی، $ClassC_j$ کلاس قانون jام، $\mu_j(x_p)$ درجهٔ تطبیق نمونهٔ x_p با قانون j اُم و m تعداد نمونه‌ها را نشان می‌دهد و r بیان‌گر تعداد نمونه‌هایی است که طبقهٔ آنها با طبقهٔ قانون j اُم یکسان است. هر یک از چهار معیار فوق، با توجه به نوع کاربرد، می‌توانند برای وزن‌دهی قوانین مورد استفاده قرار گیرند.

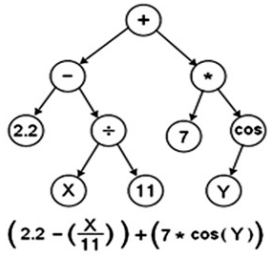
۳-۴- برنامه‌نویسی ژنتیک

یکی از الگوریتم‌های پردازش تکاملی، برنامه‌نویسی ژنتیک است که در اوائل دههٔ نود میلادی توسط جان‌کوزا مطرح و به‌تدریج توسعه داده شد.

تفاوت عمدهٔ آن با الگوریتم ژنتیک (GA) در نحوهٔ نمایش جواب مسأله است. الگوریتم ژنتیک از یک رشته از اعداد برای نمایش جواب استفاده می‌کند؛ درحالی‌که برنامه‌نویسی ژنتیک برای رگرسیون خطی به‌کار می‌رود و در نسخهٔ اصلی آن، هر کروموزوم یک عبارت یا معادله است که به‌صورت درخت نمایش داده می‌شود. در برنامه‌نویسی ژنتیک، مجموعه‌ای از توابع و پایانه‌ها[1] را در اختیار داریم که درخت‌های مزبور با استفاده از این توابع و پایانه‌ها ایجاد می‌شوند (شکل (۳-۳)). برنامه‌نویسی ژنتیک برای حل مسائل، شامل چهار مرحله است:

۱. ایجاد جمعیتی اولیه از ترکیب تصادفی توابع و پایانه‌ها برای مسأله؛

۲. اجرای هر برنامه در جمعیت و محاسبهٔ مقدار شایستگی آن؛



(شکل ۳-۳): نمونه‌ای از مدل تک‌ژنی

۳. تولید جمعیت جدیدی از برنامه‌های رایانه‌ای براساس:
- کپی بهترین برنامه‌های موجود؛
- ایجاد برنامه‌های جدید با عمل جهش؛
- تولید برنامه‌های جدید توسط عمل همبری.

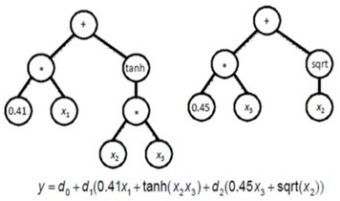
۴. انتخاب بهترین برنامهٔ رایانه‌ای در هر نسل (بهترین جواب تا این مرحله) به‌عنوان نتیجهٔ برنامه‌نویسی ژنتیک.

برخلاف برنامه‌نویسی ژنتیک سنتی، در برنامه‌نویسی ژنتیک چندژنی[2] هر کروموزوم از تعدادی ژن تشکیل شده است که هرکدام از ژن‌ها یک نمایش درختی دارند (Morrison et al., 2010).

یک کروموزوم چندژنی شامل یک یا تعداد بیش‌تری ژن است که هرکدام یک درخت هستند. وزن‌های بهینه برای ژن‌ها با استفاده از روش کم‌ترین مربعات خطا به‌دست می‌آید. خروجیِ یک رگرسیونِ چندژنی از مجموع وزن‌دار ژن‌ها به‌دست می‌آید و می‌تواند به‌صورت معادلهٔ زیر باشد:

$$y = d_0 + d_1 \times tree_1 + \ldots + d_M \times tree_M \qquad (۸)$$

d_0 ترم بایاس، $d_1 \ldots d_M$ وزن ژن‌ها و M تعداد ژن‌ها (درخت‌ها) و $tree_1 \ldots tree_M$ ژن‌های تشکیل‌دهندهٔ کروموزوم فعلی است. شکل (۴-۳) یک مدل چندژنی شبه‌خطی با دو ژن و خروجی y با ورودی‌های x_3, x_2, x_1 را نشان می‌دهد (Searson.D et al., 2010).



(شکل ۴-۳): مثالی از مدل چندژنی

1- Terminal

2- Multi-Gene Genetic Programming

تمامی عملگرهای استفاده‌شده در الگوریتم ژنتیک در اینجا نیز قابل استفاده است؛ با این تفاوت که باید بر روی درخت‌ها اعمال شود (Searson.D, 2009).

۴- روش پیشنهادی

سامانه‌های طبقه‌بندی مبتنی‌بر قوانین فازی یکی از رایج‌ترین سامانه‌های مدل‌سازی فازیِ استفاده شده، برای حل مسائل‌طبقه‌بندی هستند و وزن‌دهی به قوانین اغلب برای بهبود دقت طبقه‌بندی استفاده می‌شود. در روش پیشنهادی برای تولید وزن قوانین از روشی تکاملی استفاده می‌کنیم و معادلات تولیدشده با این روش را به‌عنوان فرمولی برای محاسبهٔ وزن قوانین به‌کار می‌بریم.

به‌منظور پیاده‌سازی روش پیشنهادی، ابتدا قوانین را براساس روش ایشیبوچی تولید می‌کنیم که نحوهٔ تولید این قوانین در بخش ۳-۲ شرح داده شده است. پس از تولید قوانین، برای دستیابی به کارایی بهتر از روش وزن‌دهی به قوانین استفاده می‌کنیم. همان‌طور که در بخش ۳-۳ توضیح دادیم، معیارهای متفاوتی برای وزن‌دهی وجود دارد که در اغلب روش‌های ارائه شده از یک معیار (که ویژگیِ خاصی را درنظر می‌گیرد) برای وزن‌دهی استفاده می‌شود؛ درحالی‌که ممکن است استفاده از عبارتی خطی یا غیرخطی براساس ترکیب چند معیار، وزن مناسب‌تر و درنتیجه قوانین بهتری را ایجاد کند. به همین‌منظور و برای استفادهٔ همزمان از چند معیار در وزن‌دهی قوانین، برنامه‌نویسی ژنتیک را به‌کار می‌بریم.

در الگوریتم برنامه‌نویسی ژنتیک، هر کروموزوم نشان‌دهندهٔ یک یا چند درخت (یا عبارت) است که براساس مجموعهٔ توابع و پایانه‌ها به‌دست آمده‌اند. بنابراین تعریف توابع و پایانه‌ها، دو گام مهم در برنامه‌نویسی ژنتیک است. از آن‌جایی که هدف از اجرای برنامه‌نویسی ژنتیک، دستیابی به عبارتی مناسب برای وزن قوانین است (که این عبارت ترکیبی از چند معیار وزن‌دهی است)، مجموعهٔ پایانه‌ها را به‌صورت زیر تعریف می‌کنیم:

$$Terminal = \{\ confidence, suppor,t\ lift, recall\ \}$$

این مجموعه از چهار معیار وزن‌دهی که در بخش ۳-۳ توضیح دادیم، تشکیل شده است. مجموعه توابعی که در این الگوریتم استفاده می‌شود، متناسب با مسألهٔ مورد نظر است؛ توابع به‌کار گرفته‌شده در این مقاله عبارتند از:

$$\{+,-,\times,\div,\hat{},\sin,\cos,\exp\}$$

برای اجرای الگوریتم برنامه‌نویسی ژنتیک، در ابتدا یک جمعیت تصادفی اولیه از N کروموزوم (با توجه به مجموعهٔ توابع و پایانه‌ها) تولید می‌کنیم و در هر کروموزوم، وزن تمامی ژن‌ها را به‌دست می‌آوریم.

به‌طورکلی در یک مدل چندژنی از برنامه‌نویسی ژنتیک، وزن هر ژن (d_i در معادلهٔ (۸)) به‌وسیلهٔ روش کم‌ترین مربعات خطا به‌دست می‌آید. برای دستیابی به وزنی دقیق‌تر در روش پیشنهادی، برای محاسبهٔ وزن هر ژن یک سامانهٔ طبقه‌بندی مبتنی‌بر قانون ایجاد می‌کنیم و معادلهٔ حاصل از ژن مورد بررسی را به‌عنوان وزن قوانین این سامانه درنظر می‌گیریم؛ سپس داده‌های آموزشی را طبقه‌بندی کرده و در انتها مقدار حاصل از معادلهٔ (۹) را به‌عنوان وزن آن ژن در نظر می‌گیریم؛ همچنین، مقدار ترم بایاس (d_0) را عددی تصادفی در بازه [۰,۱] قرار می‌دهیم.

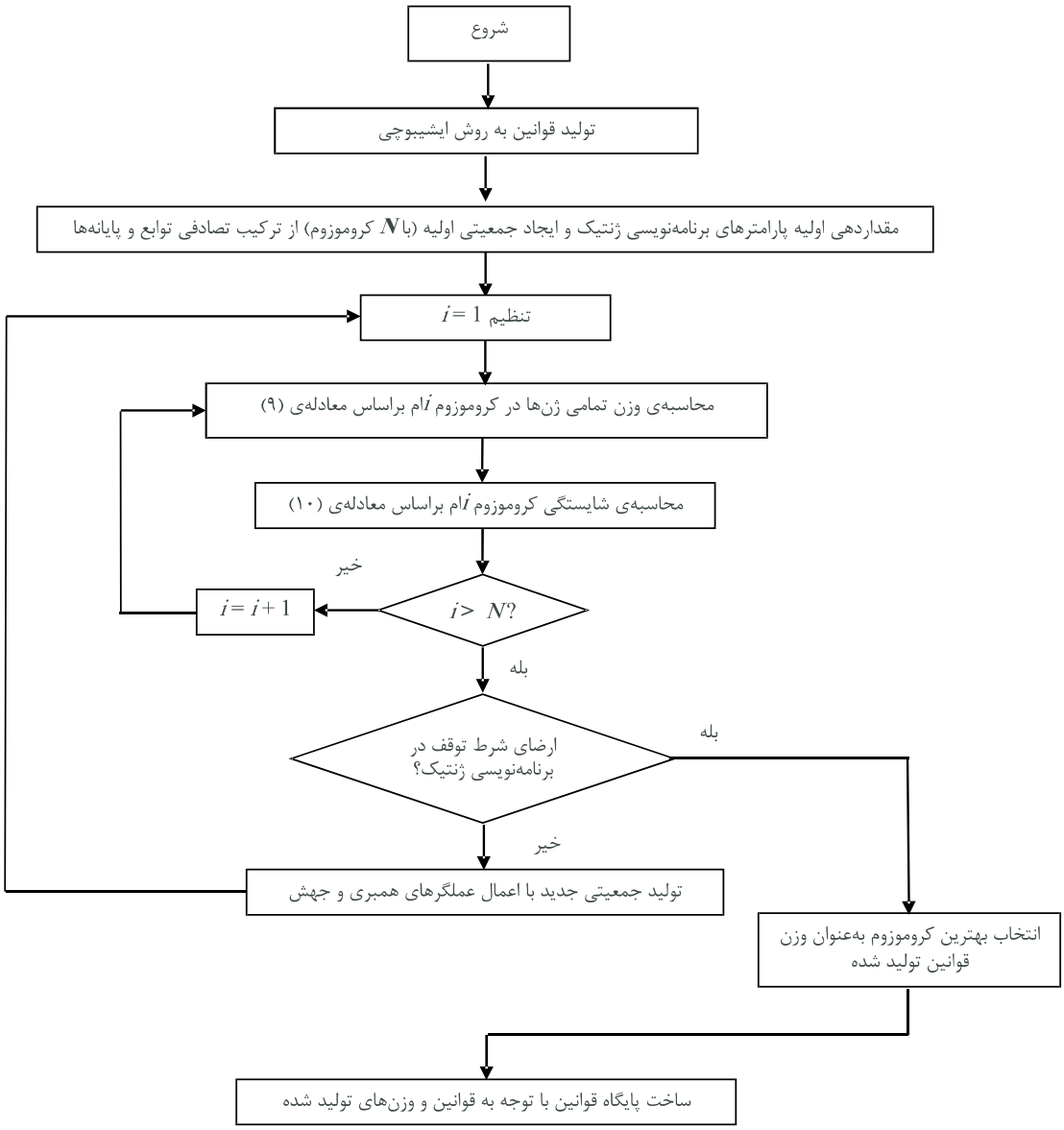
$$d_i = TP/(TP+FN) \qquad (۹)$$

پس از محاسبهٔ وزن تمامی ژن‌ها، کروموزوم از ترکیب این ژن‌ها و ترم بایاس تشکیل می‌شود. مابقی کروموزوم‌ها نیز بدین‌شکل تولید می‌شوند و فضای جستجوی برنامه‌نویسی ژنتیک را تشکیل می‌دهند. بنابراین به‌طور غیرمستقیم فضای جستجوی برنامه‌نویسی‌ژنتیک را معین کردیم؛ با این وجود، هنوز نمی‌دانیم که چه عناصر یا قسمت‌هایی از این فضای جستجو مناسب می‌باشند. تابع‌شایستگی می‌تواند سودمندی هر نمونهٔ حل (حاصل از این فضای جستجو) را اندازه‌گیری کند. تابع شایستگی مورد استفاده برای ارزیابی کروموزوم در برنامه‌نویسی ژنتیک، میانگین هندسی نرخ‌های درست (یا همان GM) می‌باشد:

$$GM = \sqrt{\frac{TP}{TP+FN}\cdot\frac{TN}{FP+TN}} \qquad (۱۰)$$

درنهایت کروموزومی که بیش‌ترین شایستگی را در کل‌تکرارها به‌دست‌آورد به‌عنوان خروجی این مرحله درنظر گرفته می‌شود و درحقیقت این کروموزوم، بهترین عبارت به‌دست آمده برای وزن قوانین است.

قوانین ساخته‌شده در مرحلهٔ اول از الگوریتم پیشنهادی، به‌همراه وزن‌های تولیدشدهٔ بالا، یک طبقه‌بند را تشکیل می‌دهند که از آن برای پیش‌بینی طبقهٔ نمونه‌های آزمون استفاده می‌شود. شکل (۵-۴) فلوچارت روش پیشنهادی را نشان می‌دهد.



(شکل ۵-۴): فلوچارت الگوریتم پیشنهادی

۵- تحلیل رفتار روش پیشنهادی

در این روش برای ساخت مدل از توابع عضویت مثلثی تعریف‌شده توسط ایشیبوچی استفاده می‌کنیم و مقدار Q را برابر بیست در نظر می‌گیریم؛ بنابراین تعداد کل قوانین تولیدشده برابر چهل است. هم‌چنین به‌منظور دستیابی به وزن مناسب برای این قوانین، از برنامه‌نویسی ژنتیک استفاده می‌شود که در جدول (۱-۵) مقادیر نسبت داده شده به عمل‌گرهای الگوریتم برنامه‌نویسی ژنتیک نشان داده شده است. مقادیر سایر عملگرهای این الگوریتم که در جدول (۱-۵) ذکر نشده‌اند، برابر با مقادیر پیش‌فرض برنامه‌نویسی ژنتیک در نظر گرفته می‌شود.

در این مطالعه، از بیست مجموعه‌داده از KEEL با نرخ عدم توازن متفاوت استفاده می‌کنیم. جدول (۲-۵) داده‌های استفاده شده را نمایش می‌دهد و برای هر مجموعه داده، تعداد نمونه‌ها، تعداد ویژگی‌ها، نام طبقه‌ها (اقلیت و اکثریت)، توزیع طبقه و نسبت تعداد نمونه‌های طبقهٔ اکثریت به تعداد نمونه‌های طبقهٔ اقلیت (نرخ عدم توازن) را نشان می‌دهیم.

(جدول۱-۵): مقادیر پارامترهای کنترلی در برنامه‌نویسی ژنتیک به‌منظور تنظیم وزن قوانین

پارامترهای کنترلی	مقدار
اندازه جمعیت	۱۰۰
نرخ همبری	۰/۸۵
نرخ جهش	۰/۱
ماکزیمم تعداد ژن (طول عبارات ایجاد شده)	۴
تعداد تکرار	۱۰۰

(جدول ۲-۵): مجموعه داده‌های استفاده‌شده در مقاله

مجموعه‌داده‌ها	نمونه‌ها	ویژگی‌ها	کلاس	توزیع کلاس	عدم‌توازن
Ecoli0vs1	220	7	(im, cp)	(35.00, 65.00)	1.86
Wisconsin	683	9	(malignant, benign)	(35.00, 65.00)	1.86
Pima	768	8	(tested-positive, tested-negative)	(34.84,66.16)	1.90
Iris0	150	4	(Iris-Setosa, reminder)	(33.33,66.67)	2.00
Vehicle2	846	18	(bus, reminder)	(28.37, 71.63)	2.52
Glass0123vs456	214	9	(non-window glass, reminder)	(23.83, 76.17)	3.19
Vehicle0	846	18	(van, reminder)	(23.64, 76.17)	3.23
Ecoli1	336	7	(im, reminder)	(22.92, 77.08)	3.36
New-thyroid2	215	5	(hypo, reminder)	(16.89, 83.11)	4.92
New-thyroid1	215	5	(hyper, reminder)	(16.28, 83.72)	5.14
Ecoli2	336	7	(pp, reminder)	(15.48, 84.52)	5.46
Segment0	2308	19	(brickface, reminder)	(14.26, 85.74)	6.01
Glass6	214	9	(headlamps, reminder)	(13.55, 86.45)	6.38
Yeast3	1484	8	(ME3, reminder)	(10.98, 89.02)	8.11
Ecoli3	336	7	(iMU, reminder)	(10.88,89.12)	8.19
Page-blocks0	5472	10	(reminder, text)	(10.23, 89.77)	8.77
Vowel0	988	13	(hid, reminder)	(9.01, 90.99)	10.10
Glass4	214	9	(containers, reminder)	(6.07, 93.94)	15.47
Yeast1458vs7	482	8	(POX, CYT)	(4.15, 95.85)	23.10
Yeast5	1484	8	(ME1, reminder)	(2.96, 97.04)	32.78

برای دستیابی به نتایج دقیق‌تر، روش صحت متقاطع پنج‌تایی بر روی مجموعه داده‌ها اعمال می‌شود. در این روش، مجموعه داده به پنج قسمت با تعداد نمونهٔ برابر افراز می‌شود و در فرآیند ساخت مدل، چهار قسمت به‌عنوان مجموعه‌دادهٔ آموزشی و یک قسمت به‌عنوان مجموعه‌دادهٔ آزمون در نظر گرفته می‌شود. به عبارت دیگر ۸۰٪ از داده‌ها برای آموزش و ۲۰٪ به‌منظور آزمون به‌کار می‌رود. این عمل را پنج مرتبه تکرار می‌کنیم، به‌طوری که تمامی داده‌ها یک‌بار به‌عنوان دادهٔ آزمون مورد استفاده قرار گیرند. بنابراین تعداد مدل‌های ساخته‌شده برابر با تعداد افرازها خواهد بود.

جدول (۳-۵)، میانگین *AUC* به‌دست آمده از اعمال روش پیشنهادی بر روی مجموعه‌داده‌های آزمون را نشان می‌دهد. ستون آخر این جدول، عبارات حاصل از اعمال برنامه‌نویسی ژنتیک چندژنی بر روی هر مجموعه‌داده را بیان می‌کند.

(جدول ۳-۵): بهترین عبارت به‌دست آمده با استفاده از روش پیشنهادی

مجموعه داده‌ها	نتایج روش پیشنهادی	بهترین عبارت به‌دست آمده در روش پیشنهادی
Ecoli0vs1	0.9893	0.974 * *lift* + 1.082
Wisconsin	0.9814	0.9707 * cos(*conf*) - 0.0159
Pima	0.7528	0.8022 * (*lift* ^4) + 0.6306 * *lift* + 0.4692
Iris0	1.0000	*conf* + 0.1715
Vehicle2	0.7625	0.7752 * *lift* + 0.7752 * (*lif* ^2) + 0.8605
Glass0123vs456	0.9174	0.9804 * *lift* - 0.9804 * *conf* + 0.9804 * *lift* ^2 + 0.5077
Vehicle0	0.8981	7.785 * *lift* + 0.9541
Ecoli1	0.9264	0.974 * *conf* * *lift* + 0.2343
New-thyroid2	0.9839	*conf* - 2 * *sup* + 0.1864
New-thyroid1	0.9834	0.4571 * cos(*recall*) + 0.9714 * *conf* * *lift* + 0.1034
Ecoli2	0.8766	0.9615 * *lift* + 0.9502
Segment0	0.9841	*lift* + 0.9422 * *lift* * (*conf* ^2) + 0.8038
Glass6	0.9401	0.8966 * *conf* - 0.8966 * *recall* + 0.931 * cos(*lift*) + 0.4918
Yeast3	0.9137	*lift* - 0.271
Ecoli3	0.8882	0.9714 * *lift* + 0.508
Page-blocks0	0.7358	0.542 * exp(*lift*) + 0.5302
Vowel0	0.9222	0.8556 * *lift* + 0.3535
Glass4	0.6987	abs(*conf* - *lift*) + 0.1818
Yeast1458vs7	0.7841	0.55 * *lift* + 0.8762
Yeast5	0.8966	0.8864 * (*lift* ^2) + 0.3001

همان‌طور که در این جدول مشاهده می‌کنید در اکثر مجموعه‌داده‌ها، معیار *lift* در ساخت عبارت مربوط به وزن قوانین استفاده شده است و با افزایش نرخ عدم توازن این معیار از عناصر اصلی تشکیل‌دهندهٔ وزن قوانین است. یکی از دلایل این امر تمایل نسبی این معیار به نمونه‌های مثبت است؛ زیرا وجود *r* (در مخرج کسر) در معیار *lift* سبب می‌شود قوانین با خروجی اکثریت وزن کم‌تری نسبت به قوانین با خروجی اقلیت داشته باشند (*r* بیان‌گر تعداد نمونه‌هایی است که خروجی آن‌ها با خروجی قانون مورد نظر یکسان است؛ بنابراین در قوانین با خروجی کلاس منفی، *r* مقدار بزرگ‌تری دارد؛ درنتیجه مقدار *lift* کاهش می‌یابد). این موضوع نشان‌دهندهٔ توجه طبقه‌بند به نمونه‌های مثبت و درنتیجه افزایش احتمال پیش‌بینی درست این نمونه‌هاست.

تحلیل روش پیشنهادی از وظایف اصلی در یک تحقیق محسوب می‌شود. به همین منظور، برای بررسی نتایج و مقایسهٔ الگوریتم پیشنهادی از سه الگوریتم شناخته‌شدهٔ طبقه‌بندی استفاده می‌کنیم که عبارتند از:

- **C4.5** (Lopez et al., 2012): C4.5 یک الگوریتم تولیدکنندهٔ درخت تصمیم است. درخت تصمیم به‌صورت بالا-پایین با استفاده از information gain (تفاوت در بی‌نظمی)- ناشی از انتخاب یک ویژگی برای شکاف داده- ساخته می‌شود. ویژگی با بالاترین normalized information gain برای تصمیم‌گیری انتخاب می‌شود. سپس الگوریتم در زیرفهرست‌های کوچک‌تر تکرار می‌شود.
- **SVM** (Lopez et al., 2012): یکی از روش‌هایی که در حال حاضر به‌صورت گسترده‌ای برای مسألهٔ دسته‌بندی مورد استفاده قرار می‌گیرد، روش ماشین بردار پشتیبان

یا SVM است. دراصل SVMها برای طبقه‌بندی دو طبقهٔ خطی با استفاده از margin طراحی شده‌اند، که margin به معنی کمینهٔ فاصله از ابرصفحهٔ جداکننده تا نزدیک‌ترین نقاط داده است.

- **3-NN** (Lopez et al., 2012): 3-NN گروهی از سه نمونه در مجموعهٔ داده‌ها که به الگوی آزمون نزدیک‌ترند را پیدا می‌کند. یک نمونهٔ آزمون به الگوریتم داده می‌شود؛ الگوریتم فاصلهٔ (یا شباهت) بین نمونهٔ آزمون و تمامی نمونه‌های آموزشی را محاسبه می‌کند تا سه نزدیک‌ترین همسایه‌اش را پیدا کند. طبقهٔ نمونهٔ آزمون با بیش‌ترین کلاس در میان سه نزدیک‌ترین نمونه به‌دست می‌آید.

همچنین روش پیشنهادی را با چهار سامانهٔ طبقه‌بندی مبتنی‌بر قانون مقایسه می‌کنیم که عبارتند از:

- **FH-GBML** (Lopez et al., 2010): این سامانهٔ طبقه‌بندی با استفاده از روش Pittsburgh ساخته می‌شود که در آن هر کروموزوم نشان‌دهندهٔ یک پایگاه قوانین است. همچنین روش GCCL را برای اصلاح قوانین به‌کار می‌برد که باعث ایجاد قوانینی کاراتر می‌شود.
- **SGERD** (Mansoori et al., 2008): این تکنیک، با به‌کارگیری الگوریتم ژنتیک یک سامانهٔ طبقه‌بندی مبتنی بر قانون تکاملی را ایجاد و درحقیقت یک مجموعهٔ فشرده از قوانین مناسب را استخراج می‌کند. تعداد تکرار در الگوریتم ژنتیک محدود است و به ابعاد مسأله بستگی دارد. انتخاب کروموزوم در این روش غیرتصادفی است و تنها بهترین کروموزوم می‌تواند به حیات خود ادامه دهد.
- **Chi-FRBCS** (Fernandez et al., 2008): در این روش تعداد برچسب‌های فازی، روش‌های وزن‌دهی و استفاده از نرم‌های مثلثی مختلف مورد بررسی قرار گرفت و درنهایت یک سامانهٔ طبقه‌بندی مبتنی بر قوانین فازی با کارایی بالاتر به عنوان طبقه‌بند ایجاد شد.
- **GFS-Max LogitBoost** (Sánchez & Otero, 2007): در این روش، یک مدل طبقه‌بندی با استفاده از طبقه‌بندهای logistic regression boosting ایجاد می‌شود.

در جدول (۴-۵)، نتایج حاصل از اعمال این الگوریتم‌ها بر روی مجموعه‌داده‌های مورد بررسی نشان داده شده است. این نتایج، میانگینِ مربوط به پنج‌بار اجرای داده‌هایِ آزمون است. . در این جدول، روشی که از دیگر روش‌ها عملکرد بهتری دارد به‌صورت **پررنگ** مشخص شده است. با مشاهدهٔ جدول درمی‌یابیم که روش پیشنهادی عملکرد بهتری نسبت به سایر روش‌ها دارد. نتایج آزمایش‌ها در جدول (۴-۵) نشان می‌دهد که روش پیشنهادی میانگین بالاتری در اکثر مجموعه‌داده‌ها به‌دست آورده است. با این‌حال، ارزیابی تنها براساس مشاهده مشخص نمی‌کند که آیا میان روش‌های مختلف، تفاوت قابل ملاحظه‌ای وجود دارد یا خیر.

در اینجا از آزمون‌های آماری برای اطمینان از وجود تفاوت معنی‌دار میان روش‌ها استفاده می‌کنیم. منظور از تفاوت معنی‌دار این است که اختلاف میان روش‌ها به حدّ کافی بزرگ باشد تا بتوان اطمینان حاصل کرد که نتایج به‌طور تصادفی یا بر اثر شانس، بهتر نشده‌اند. بنابراین برای تحلیل دقت‌های گزارش‌شده، از روش‌های ارائه‌شده در نرم‌افزار KEEL (Alcalá-Fdez et al., 2011) استفاده می‌کنیم.

برای ارزیابی کارایی روش پیشنهادی آزمون *فریدمن* را به‌کار می‌بریم که یک آزمون آماری، مبتنی بر مقایسهٔ چندین روش است. این آزمون ، هر الگوریتم را به‌طور جداگانه برای هر مجموعه‌داده رتبه‌بندی می‌کند و به الگوریتمی که بهترین نتیجه را به‌دست آورد رتبهٔ یک و به بدترین الگوریتم، رتبه M (در صورتی که M الگوریتم را با هم مقایسه کنیم) می‌دهد. در مواردی که مقادیر و درنتیجه رتبه‌های یکسان وجود دارد، میانگین رتبه‌ها محاسبه می‌شود. فرضیهٔ صفر، بیان می‌کند که تمامی الگوریتم‌ها یکسان هستند؛ درحالی‌که رد این فرضیه، وجود تفاوت میان الگوریتم‌های مورد بررسی را نشان می‌دهد. روش کار آزمون فریدمن به‌صورت زیر می‌باشد:

فرض می‌کنیم که r_i^j رتبهٔ j اُمین از میان k الگوریتم در i اُمین مجموعه داده (از میان N مجموعه داده) است. آزمون فریدمن میانگین رتبه‌ها را با هم مقایسه می‌کند که با استفاده از معادلهٔ (۱۱) محاسبه می‌شود:

$$R_j = \frac{1}{N} \cdot \sum_i r_i^j \qquad (۱۱)$$

(جدول۴-۵): مقایسهٔ میانگین هندسی روش پیشنهادی با هفت روش طبقه‌بندی دیگر

Dataset	C4.5	SVM	3-NN	FH-GBML	SGERD	Chi-FRBCS	GFS-MLB	Current method
Ecoli0vs1	0.9832	0.9671	0.5000	0.9762	0.9754	0.9651	0.9600	**0.9893**
Wisconsin	0.9454	0.9666	0.9658	0.9620	0.9245	0.4080	0.9564	**0.9814**
Pima	0.7012	0.7194	0.6703	0.6980	0.6438	0.7041	0.7048	**0.7528**
Iris0	0.9900	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9900	**1.0000**
Vehicle2	0.9561	0.9527	**0.9736**	0.8204	0.6244	0.9636	0.8612	0.7625
Glass0123vs456	0.9155	0.9043	0.8888	0.8888	0.8401	0.8625	0.8962	**0.9174**
Vehicle0	0.9296	**0.9490**	0.9379	0.8348	0.5930	0.8591	0.8669	0.8981
Ecoli1	0.8586	0.8192	0.7636	0.8480	0.4945	0.8600	0.8511	**0.9264**
New-thyroid2	0.9373	0.9829	0.9373	0.9546	0.7257	0.8409	0.8488	**0.9839**
New-thyroid1	0.9143	0.9829	0.9659	**0.9931**	0.7428	0.8413	0.9571	0.9834
Ecoli2	0.8641	0.7351	0.8302	0.8550	0.4963	0.9184	0.8813	**0.8766**
Segment0	0.9826	**0.9927**	0.5000	0.9709	0.4997	0.9931	0.8029	0.9841
Glass6	0.8132	0.9198	0.9140	0.9032	0.8885	0.8372	0.8864	**0.9401**
Yeast3	0.8597	0.6299	0.8171	0.8321	0.5523	0.5007	0.8278	**0.9137**
Ecoli3	0.7280	0.5000	0.6598	0.7674	0.4665	0.7933	0.6976	**0.8882**
Page-blocks0	**0.9221**	0.8218	0.9075	0.8116	0.5986	0.7209	0.7011	0.7358
Vowel0	0.9706	0.8950	0.9939	0.8256	0.7710	**0.9976**	0.9133	0.9222
Glass4	**0.9997**	0.5592	0.8425	0.6479	0.5666	0.8210	0.7849	0.6987
Yeast1458vs7	0.5000	0.5000	0.5144	0.4985	0.5000	0.6574	0.5000	**0.7841**
Yeast5	0.6135	0.5000	0.8128	0.6783	0.5000	0.4399	0.5236	**0.8966**

براساس فرضیهٔ صفر (که بیان می‌کند تمامی الگوریتم‌ها برابرند و درنتیجه رتبهٔ آنها (R_j) باید یکسان باشد) احتمال فریدمن براساس χ^2 با $k-1$ درجهٔ آزادی به‌دست می‌آید و به‌صورت زیر محاسبه می‌شود (Demsar, 2006):

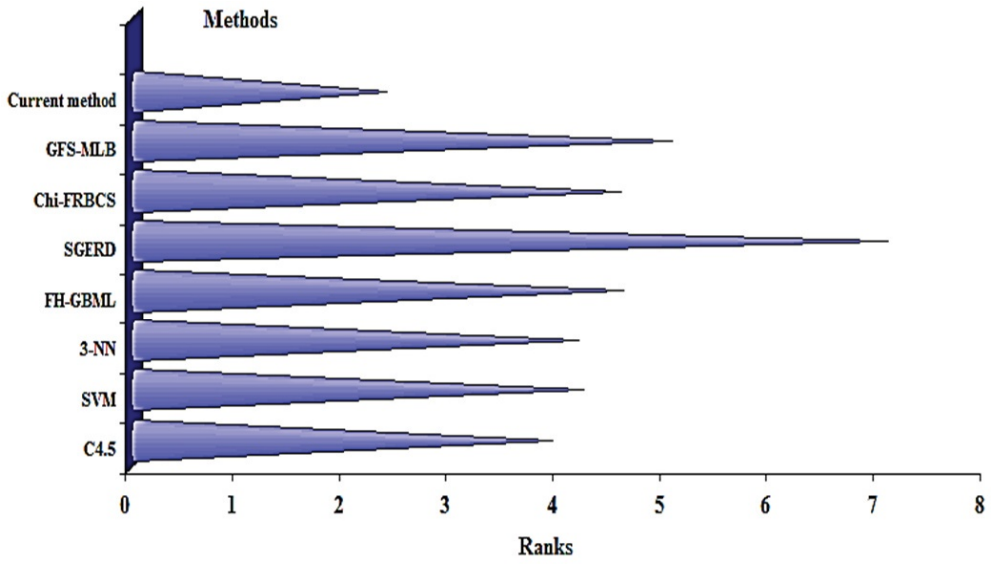
$$\chi_F^2 = \frac{12N}{k(k+1)} \cdot [\sum_j R_j^2 - \frac{k(k+1)^2}{4}]. \qquad (۱۲)$$

میانگین رتبه‌های به‌دست‌آمده برای هر روش با استفاده از آزمون فریدمن در شکل (۶-۵) نشان داده شده است. مقدار احتمال فریدمن، ۳۹.۹۲۰۸۳۳ با درجهٔ آزادی هفت است و *p-value* محاسبه شده با این آزمون ۰/۰۰۰۰۰۱ است.

به‌دلیل این‌که مقدار احتمال فریدمن از *p-value* محاسبه‌شده بزرگ‌تر است؛ فرضیهٔ صفر رد می‌شود. با توجه به این‌که روش با بهترین میانگین هندسی، کم‌ترین رتبه را دارد؛ بنابراین روشی که کم‌ترین میانگین رتبه را دارد، بهترین عملکرد را خواهد داشت. آشکار است که، میانگین رتبهٔ نسبت داده شده به روش پیشنهادی کم‌تر از رتبهٔ روش‌های دیگر است. بنابراین روش پیشنهادی، بهترین روش اجرایی است.

بعد از اطمینان از وجود تفاوت با استفاده از آزمون فریدمن، از آزمون post hoc استفاده می‌کنیم تا تشخیص دهیم که آیا روش کنترلی (که بهترین روش در آزمون فریدمن است) تفاوت معناداری با روش‌های دیگر شرکت‌کننده در مقایسه دارد یا خیر (Garcia et al., 2010). Post hoc استفاده‌شده در این مقاله، آزمون Finner است.

رویهٔ Finner مقدار α را در یک حالت گام پایین تنظیم می‌کند. این روش، فرضیهٔ H_1 تا H_{i-1} را رد می‌کند؛ اگر i کوچک‌ترین عدد صحیح باشد که $p_i > 1-(1-\alpha)^{i/(k-1)}$. *p-value*های به‌دست آمده با اعمال روش‌های Finner (به‌عنوان روش Post hoc انتخاب‌شده در این مقاله) بر روی نتایج آزمون فریدمن، در جدول (۵-۵) نشان داده شده است.



(شکل ۶-۵) : میانگین رتبهٔ الگوریتم‌ها با استفاده از آزمون فریدمن

(جدول۵-۵): جدول مقایسهٔ post hoc برای روش پیشنهادی $\alpha = 0.05$ (فریدمن)

i	Algorithms	z	P	Finner
7	SGERD	6.066674	0	0.007301
6	GFS-MLB	6.43341	0.000553	0.014548
5	FH-GBML	2.862463	0.003972	0.021743
4	Chi-FRBCS	2.830048	0.004312	0.028885
3	SVM	2.26732	0.015852	0.035975
2	3-NN	2.20167	0.020114	0.043013
1	C4.5	2.000014	0.043578	0.05

رویهٔ Finner تمام فرضیه‌هایی را که $p - value \leq 0.05$ دارند، رد می‌کند. همان‌طور که جدول (۵-۵) نشان می‌دهد، رویهٔ Finner ثابت می‌کند که روش پیشنهادی بهتر از هفت روش دیگر انجام می‌شود؛ زیرا تمامی روش‌ها $P\_Finner \leq 0.05$ دارند و فرضیهٔ صفر که نشان‌دهندهٔ عملکرد یکسان روش‌هاست؛ رد می‌شود.

بعد از انجام post hoc، *p-value*های تنظیم‌نشده برای محاسبهٔ p-valueهای دقیق استفاده می‌شود.

*p-value*های تنظیم‌شدهٔ به‌دست‌آمده، در جدول (۶-۵) نشان داده شده است. در این جدول نیز مقدار *p-value* تمامی روش‌ها کم‌تر از ۰/۰۵ است که نشان می‌دهد روش پیشنهادی عملکرد بهتری نسبت به سایر روش‌های مورد مقایسه دارد.

(جدول۶-۵) : p-valueهای تنظیم‌شده (فریدمن)

i	Algorithms	Unadjusted p	p_{Finner}
7	SGERD	0	0
6	GFS-MLB	0.000554	0.001939
5	FH-GBML	0.004073	0.009478
4	Chi-FRBCS	0.004509	0.009478
3	SVM	0.016925	0.023614
2	3-NN	0.020137	0.023614
1	C4.5	0.045388	0.045388

۶- نتیجه‌گیری

در میان سامانه‌های طبقه‌بندی، کارایی سامانه‌های فازی سبب شده است که به‌طور گسترده مورد استفاده قرار گیرند. از طرفی دیگر، استفاده از الگوریتم‌های تکاملی در مواردی که نیاز به پیداکردن جوابی بهینه (نه لزوماً بهترین جواب) داریم باعث سرعت‌بخشی در عملکرد طبقه‌بند می‌شود. هدف اصلی این تحقیق ایجاد بهبود در الگوریتم‌های طبقه‌بندی نامتوازن با استفاده از تئوری فازی و الگوریتم‌های تکاملی است.

از جمله موفق‌ترین و پرکاربردترین سامانه‌های طبقه‌بند فازی، سامانه‌های طبقه‌بندی مبتنی‌بر قوانین فازی است که در حوزهٔ مسائل نامتوازن نیز با اعمال تغییرات و اصلاحات بر روی روش‌های موجود، نتایج قابل قبولی کسب کرده‌اند. در این مقاله، مسألهٔ طبقه‌بندی مجموعه‌داده‌های نامتوازن را با استفاده از سامانه‌های طبقه‌بندی مبتنی‌بر قانون مورد بررسی قرار دادیم و برای این منظور، یک روش جدید تکاملی، برای تخصیص وزن در سامانه‌های طبقه‌بندی مبتنی‌بر قوانین فازی ارائه دادیم. این تکنیک براساس برنامه‌نویسی ژنتیک چندژنی است که با استفاده از gptips پیاده‌سازی شده است. بهترین عبارت به‌دست‌آمده با استفاده از این الگوریتم برای قوانین در هر مجموعهٔ داده، معین شد؛ که این عبارت، معیارِ ترکیبیِ جدیدی از *confidence*، *support*، *lift* و *recall* برای وزن‌دهی قوانین است.

همچنین کارایی قوانین فازی استخراج‌شده از داده‌های عددی را (برای مسائل طبقه‌بندی که از روش وزن‌دهی پیشنهادی استفاده می‌کنند)، بررسی کردیم. نتایج شبیه‌سازی بر روی بیست مجموعه‌دادهٔ شناخته‌شدهٔ keel نشان می‌دهد که در اکثر مجموعه‌داده‌های مورد بررسی، روش پیشنهادی عملکرد بهتری نسبت به سایر روش‌های مورد مقایسه دارد و بنابراین می‌توان نتیجه گرفت عبارات جدیدی که به‌عنوان وزن قوانین درنظر گرفته شده‌اند، منجر به بهبود کارایی طبقه‌بندی می‌شوند. درنهایت برای اثبات نتایج آزمایش، از آزمون‌های آماری استفاده کردیم.

۷- منابع

Alcala-Fdez.J, Alcala.R and Herrera.F, "A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning", IEEE Trans. On Fuzzy Systems, 2011, Vol. 19, 857-872.

Alcalá-Fdez.J , Fernández.A, Luengo.J, Derrac.J, García.S, Sánchez.L, Herrera.F, "KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework", Journal of Multi-Valued Logic and Soft Computing, 2011, vol. 17, 255–287.

Chi.Z, Yan.H and Pham.T, "Fuzzy alghorithms with applications to image processing and pattern recognition", World Scientific, 1996, Vol. 10, 139-145.

Demsar.J, "Statistical comparisons of classifiers over multiple data sets", Journal of Machine Learning Reasearch, 2006, Vol. 7, 1-30.

Fern´andez.A, Berlanga.F, Jesus.M and Herrera.F, "Genetic Cooperative-Competitive Fuzzy Rule Based Learning Method using Genetic Programming for Highly Imbalanced Data-Sets",IFSA-EUSFLAT onference,2009.

Fernandez.a, Garcia.S, Jesus.M.J and Herrera.F, "A study of the behavior of linguistic fuzzy rule based classification system in framework of imbalanced data-sets", Fuzzy Sets and Systems, 2008, Vol. 159,

Oh. S, Lee.M.S and Zhang.B, "Ensemble Learning active example selection for imbalanced biomedical data classification", IEEE/ACM Trans. On computational biology and bioinformatics, 2011, Vol. 8, No. 2, 316-324.

Sánchez.L, Otero.J, "Boosting fuzzy rules in classification problems under single-winner inference", International Journal of Intelligent Systems, 2007, vol. 22, 1021-1034.

Searson,D. "GPTIPS: Genetic Programming & Symbolic Reression for MATLAB", http://gptips.sourceforge.net,2009.

Searson.D, Leahy.D and Willis.M, "GPTIPS:An Open Source Genetic Programming Toolbox For Multigene Symbolic Regression", 2011, Proceeding of International MultiConference of Engineers and Computer Scientist, 2010.

Sun.Y, Wong.A.K.C and Kamel.M.C, "Classification of imbalanced data: a review", International Journal of Pattern Recognition and Artificial Intelligence, 2009, vol. 23, 687–719.

Tahir.M, Kittler.J and Yan.F, "Inverse random under sampling for class imbalance problem and its application to multi-label classification", Pattern Recognition, 2012, Vol. 45, 3738-3750.

Tong.L.I, Chang.Y.Cand Lin.C.S, "Determining the optimal re-sampling strategy for classification model with imbalanced data using design of experiments and response surface methodologies" Expert Systems With Applications, 2011, Vol. 38, 4222-4227.

Wang.L and Mendel.J, "Generating fuzzy rules by learning from examples" IEEE Trans. Systems Man Cybernet, 1992, Vol. 25, 353-361.

Williams.D.P, Myers.V, and Silvious.M.S, "Mine classification with imbalanced data", IEEE Transactions on Geoscience and Remote Sensing Letters, 2009, vol. 6, 528-532.

Yen.S and Lee.Y.S, "Cluster-based under-sampling approaches for imbalanced data distributions", Expert Systems with Applications, 2009, vol. 36, 5718-5727.

Yu.H, Ni.J, and Zhao.J, "ACOSampling: An ant colony optimization-based undersampling method for classifying imbalanced DNA microarray data", Neurocomputing, 2013, vol. 101, 309-318.

2378-2398.

Galar.M, Fernandez.A, Barrenechea.E and Bustince.H, "A review on ensembles for the class imbalance problem: Bagging-,Boosting-, and Hybrid-based approaches", IEEE Trans. On systems, man, and cybernetics-Part C: Applications and Review, 2012, Vol. 42, 463-484.

Garcia.S, Derrac.J, Triguero.I, Carmona.C, Herrera.F, "Evolutionary-based selection of generalized instances for imbalanced classification", Knowledge-Based Systems, 2012, Vol. 25, 3-12.

Garcia.S, Fernandeaz.A, Luengo.J, Herrera.F, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power", Information Sciences, 2010, Vol. 180, 2044-2064.

He.H and Garcia.E, "Learning from imbalanced data", IEEE Transactions on Knowledge and Data Engineering, 2009, vol. 21, 1263-1284.

Ishibuchi.H and Yamamoto.T, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining", Fuzzy Set and Systems, 2004, Vol. 141, 59-88.

Ishibuchi.H, Yamamoto.T, "Rule weight specification in fuzzy rule-based classification systems", IEEE Trans. On fuzzy systems, 2005, Vol. 13, No. 4, 428-435.

Lopez.V, Fernandez.A and G.Moreno-Torres.J, Herrera.F, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics", Expert Systems With Application, 2012 ,Vol.39, 6585-6608.

Lopez.V, Fernandez.A and Herrera.F, "A first approach for cost-sensitive classification with linguistic genetic fuzzy systems in imbalanced data-sets", IN 10th International conference on intelligent systems design and applications, ISDA, 2010, 676-681.

Lopez.v, Fernandez.A, Jesus.M, Herrera.F, "A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline datasets", Knowledge-Based Systems, 2013, Vol. 38, 85-104.

Mansoori.E, Zolghadri.M, Katebi.S, "SGERD: A Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules From Data", IEEE Transactions on Fuzzy Systems, 2008, vol. 16, 1061-1071.

Morrison.G, Searson.D, Willis.M, "Using genetic programming to evolve a team of data classification", World Academy of Science, Engineering and Technology, 2010, Vol. 72, pp. 261-264.

محبوبه مهدی‌زاده مدرک کارشناسی خود را در رشتهٔ مهندسی کامپیوتر گرایش نرم‌افزار در سال ۱۳۸۹ از دانشگاه آزاد اسلامی واحد مشهد اخذ کرد. وی تحصیلات خود را در مقطع کارشناسی ارشد کامپیوتر گرایش هوش مصنوعی در دانشگاه شهید باهنر کرمان در سال ۱۳۹۲ به پایان رسانده است. زمینه‌های پژوهشی مورد علاقهٔ ایشان پردازش تصویر، یادگیری ماشین، داده‌کاوی و الگوریتم‌های تکاملی است.
نشانی رایانامهٔ ایشان عبارتست از:
mh.mahdizadeh@gmail.com

مهدی افتخاری مدرک کارشناسی خود را در سال ۱۳۷۹در رشتهٔ مهندسی کامپیوتر گرایش سخت‌افزار از دانشگاه شیراز، و مدارک کارشناسی‌ارشد و دکترای خود را نیز به‌ترتیب در سال‌های ۱۳۸۲ و ۱۳۸۶ در رشتهٔ مهندسی کامپیوتر گرایش هوش مصنوعی از همان دانشگاه اخذ کرد. وی از سال ۱۳۸۶ تاکنون عضو هیأت علمی دانشکدهٔ مهندسی کامپیوتر دانشگاه شهید باهنرکرمان است. حوزه‌های تخصصی ایشان شامل مدل‌سازی و سامانه‌های فازی، الگوریتم‌های تکاملی، داده‌کاوی، یادگیری ماشین و کاربرد روش‌های هوشمند در بیوانفورماتیک است. وی تاکنون بیش از شصت مقاله در نشریات معتبر علمی و کنفرانس‌ها منتشر کرده است.
نشانی رایانامهٔ ایشان عبارتست از:
m.eftekhari@mail.uk.ac.ir