The 9th International Conference on Web Research
ICWR2023
May 3th-4th, 2023, Tehran, Iran

Online Learning in Dynamic Feature Space

[1] Reza Sajedi, [2] Mohammadreza Razzazi

[1] MSc Student of Computer Engineering, Amirkabir University of Technology, Tehran, Iran
r.sajedi@aut.ac.ir

[2] Professor of Computer Engineering, Amirkabir University of Technology, Tehran, Iran
razzazi@aut.ac.ir

Abstract

Nowadays, most of our daily activities are carried out on the web. The high speed and volume of data production on the web have made the use of online machine learning algorithms in processing and analyzing data streams very efficient. Many of these algorithms have been developed assuming a fixed feature space; however, in real-world problems, this assumption may not hold and each instance of a data stream may have different features. In this study, this new problem that has recently attracted a lot of attention is investigated. Also, a novel general algorithm for data stream classification is proposed, which exploits the relationships between features and estimates the values of unavailable features to achieve the maximum potential classifier. Finally, through empirical experiments and comparison with two recent algorithms, it is shown that the proposed algorithm has higher accuracy.

Keywords: Online Machine Learning, Data Stream Classification, Feature Evolution, Algorithm

The 9th International Conference on Web Research
ICWR2023
May 3th-4th, 2023, Tehran, Iran

یادگیری برخط در فضای ویژگی پویا

رضا ساجدی[1]، محمدرضا رزازی[2]

[1] دانشجوی کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران
r.sajedi@aut.ac.ir

[2] استاد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران
razzazi@aut.ac.ir

چکیده

امروزه غالب فعالیت‌های روزمره انسان بر بستر وب انجام می‌شود. سرعت و حجم بالای تولید داده در وب، استفاده از الگوریتم‌های برخط یادگیری ماشین در پردازش و تحلیل جویبارداده را بسیار کارآمد جلوه داده است. بسیاری از این الگوریتم‌ها با فرض ثابت بودن فضای ویژگی ارائه شده‌اند؛ اما در مسائل دنیای واقعی ممکن است چنین فرضی رعایت نشود و هر نمونه از جویبارداده ویژگی‌های متفاوتی داشته باشد. در این پژوهش، این مسئله جدید که اخیراً توجه زیادی را به خود جلب کرده است بررسی می‌شود. همچنین یک الگوریتم عمومی نوین برای طبقه‌بندی جویبارداده ارائه می‌شود که با کشف روابط بین ویژگی‌ها و تخمین مقادیر مربوط به ویژگی‌های ناموجود، از حداکثر پتانسیل طبقه‌بند در پیش‌بینی بهره‌برداری می‌کند. درنهایت با انجام آزمایش‌های تجربی و مقایسه نتایج آن با دو مورد از الگوریتم‌های اخیر، نشان داده می‌شود که الگوریتم ارائه شده دقت بالاتری دارد.

کلمات کلیدی

یادگیری برخط، طبقه‌بندی جویبارداده، تحول ویژگی، الگوریتم

۱- مقدمه

گسترش روزافزون وب باعث شده است تا غالب نیازهای روزمره انسان در این محیط پاسخ داده شود. با فعالیت افراد در وب، در هر لحظه حجم انبوهی از داده تولید می‌شود. برای تحلیل و کشف دانش از این داده‌ها، دیگر الگوریتم‌های برون‌خط[1] یادگیری ماشین که با رویکرد پردازش دسته‌ای[2] اجرا می‌شوند، کارایی چندانی ندارند. در عوض استفاده از الگوریتم‌های یادگیری برخط برای پردازش این جویبارداده‌ها[3] می‌تواند کارآمد باشد.

اندازه نامتناهی، رانش مفهوم[4]، تحول مفهوم[5]، رانش ویژگی[6] و تحول ویژگی[7] از جمله چالش‌های یادگیری برخط هستند. پدیده رانش مفهوم با تغییر تدریجی رابطه بین داده ورودی و متغیر هدف (تغییر توزیع داده‌ها) اتفاق می‌افتد [1]. تحول مفهوم به‌معنای ظهور طبقه‌های جدید در جویبارداده است [2]. در تحول ویژگی، هر نمونه از جویبارداده ممکن است متغیرهای مستقل متفاوتی داشته باشد [3]. این مورد با رانش ویژگی تفاوت دارد. در رانش ویژگی فرض می‌شود که فضای ویژگی، ثابت و متناهی است و اهمیت ویژگی‌های انتخابی به‌مرور زمان کاهش می‌یابد [4].

در این پژوهش، تمرکز اصلی بر روی چالش تحول ویژگی است. این چالش با نام‌های فضای ویژگی پویا و فضای ویژگی متغیر[8] نیز شناخته شده است. بسیاری از کابردهای دنیای واقعی با این چالش روبه‌رو هستند. مثلاً در طبقه‌بندی اسناد، هر سند ممکن است شامل کلمه‌های متفاوتی باشد. در سامانه‌های نظارت بر محیط یا سامانه‌های سلامت که مبتنی‌بر اینترنت اشیا هستند، ممکن است برخی از حسگرهای قدیمی با حسگرهای جدید جایگزین شوند و فضای ویژگی نمونه‌ها با فضای ویژگی طبقه‌بند متفاوت باشد. در سامانه‌های توصیه‌گر دائماً اقلام جدیدی اضافه می‌شود. در مجموع هر مسئله‌ای که با جویبارداده سروکار دارد، ممکن است با این چالش مواجه باشد.

در این پژوهش، یک الگوریتم عمومی جدید برای طبقه‌بندی جویبارداده در فضای ویژگی پویا با استفاده از نگاشت ویژگی‌ها (DCDF2M)[9] ارائه می‌شود. در این پژوهش فرض می‌شود که ویژگی‌ها بتوانند با هر ترتیب دلخواهی تغییر کنند؛ یعنی ویژگی‌های موجود در هر نمونه از جویبارداده ممکن است با نمونه قبل یا بعد از آن متفاوت باشد. هر نمونه ممکن است شامل ویژگی‌های تازه‌ای باشد که طبقه‌بند[10] تاکنون آنها را مشاهده نکرده است. همچنین ممکن است برخی از ویژگی‌هایی که طبقه‌بند قبلاً آنها را مشاهده کرده است در آن نمونه، ناموجود باشد. الگوریتم پیشنهادی با کشف روابط بین ویژگی‌ها و تخمین ویژگی‌های ناموجود، از حداکثر پتانسیل طبقه‌بند برای

پیش‌بینی طبقه مناسب بهره‌مند می‌شود. این الگوریتم، عمومی است؛ یعنی مبتنی‌بر طبقه‌بند خاصی نیست و می‌تواند در هر کابرد از طبقه‌بندی که بهترین عملکرد را دارد استفاده کند.

درنهایت با انجام آزمایش‌های تجربی بر روی ۱۰ مجموعه‌داده و مقایسه نتایج آن با دو الگوریتم GLSC [5] و OLVF [6] که فرض‌های یکسانی در تعریف مسئله داشته‌اند، نشان داده می‌شود که الگوریتم پیشنهادی دقت بالاتری دارد.

سهم علمی این پژوهش عبارت است از:

– ارائه یک الگوریتم عمومی جدید برای طبقه‌بندی جویبارداده در فضای ویژگی پویا با استفاده از نگاشت ویژگی‌ها

– استفاده از یک مدل پایه در تخمین ویژگی‌های ناموجود و اثبات یک حد بالا برای خطای تخمین

– ارزیابی الگوریتم با انجام آزمایش‌های تجربی، مقایسه عملکرد آن با دو مورد از الگوریتم‌های اخیر و نشان دادن دقت بالاتر

سایر مطالب این گزارش، بدین صورت سازمان‌دهی شده است: ابتدا در بخش ۲ پیشینه پژوهش بررسی می‌شود. در بخش ۳ جزئیات الگوریتم پیشنهادی شرح داده می‌شود. در بخش ۴ عملکرد الگوریتم به‌صورت تجربی ارزیابی می‌شود. درنهایت در بخش ۵ جمع‌بندی و نتیجه‌گیری انجام می‌شود.

۲- مرور پیشینه پژوهش

تعداد زیادی از پژوهش‌ها، یک سری فرض‌های محدودکننده درباره نحوه تحول ویژگی‌ها درنظر گرفته‌اند. در پژوهشی [7] از مفهوم جویبارداده ذوزنقه‌ای[11] رونمایی می‌شود که در آن هر نمونه از جویبارداده باید همه ویژگی‌هایی که قبلاً مشاهده شده‌اند را دربر داشته باشد؛ یعنی اندازه هر نمونه جدید باید بزرگتر یا برابر با نمونه قبلی باشد.

پژوهشی دیگر [8] با فرض وجود دوره همپوشانی انجام شده است. در این دوره کوتاه، نمونه‌ها باید همه ویژگی‌های قدیمی و جدید را دربر داشته باشند تا رابطه بین آنها یاد گرفته شود. پس از اتمام دوره همپوشانی، همه ویژگی‌های قدیمی باید همزمان ناپدید و همه ویژگی‌های جدید همزمان پدیدار شوند. در نسخه‌ای دیگر از این پژوهش، چالش رانش مفهوم نیز وارد می‌شود [9]. نسخه‌ای دیگر [10] در حالت نیمه‌نظارتی ارائه شده است. در پژوهشی دیگر نیز که با فرض وجود دوره همپوشانی انجام شده است، از یادگیری عمیق برای کشف رابطه بین ویژگی‌ها استفاده می‌شود [11].

دسته‌ای دیگر از پژوهش‌ها، هیچ فرض محدودکننده‌ای درباره نحوه تغییر ویژگی‌ها ندارند. در این پژوهش‌ها که به واقعیت نزدیک‌ترند فرض می‌شود که هر نمونه از جویبارداده بتواند ویژگی‌های متفاوتی در مقایسه با نمونه قبل یا بعد از خود داشته باشد. پژوهش فعلی نیز چنین فرضی در تعریف مسئله دارد.

در پژوهشی [6]، الگوریتم OLVF ارائه می‌شود که یاد می‌گیرد فضاهای ویژگی و نمونه‌ها را به‌طور همزمان طبقه‌بندی کند. برای طبقه‌بندی یک نمونه، الگوریتم به‌طور پویا طبقه‌بند نمونه و نمونه آموزشی را به زیرفضای ویژگی مشترک آنها تصویر می‌کند. طبقه‌بند فضای ویژگی، اطمینان تصویرسازی را برای فضای ویژگی داده شده پیش‌بینی می‌کند. طبقه‌بند نمونه، با پیروی از اصل کمینه‌سازی ریسک تجربی به‌روزرسانی شده و قدرت محدودیت‌ها با اطمینان تصویرسازی مقیاس‌بندی می‌شود. تفاوت OLVF با الگوریتم پیشنهادی در این است که عمومی نیست (نمی‌تواند با هر طبقه‌بندی کار کند) و همچنین از تکنیک نگاشت ویژگی‌ها برای تخمین مقادیر ناموجود استفاده نمی‌کند.

در پژوهشی دیگر [5]، الگوریتم GLSC ارائه می‌شود. این الگوریتم نیز مانند الگوریتم پیشنهادی، با کشف روابط بین ویژگی‌ها مقادیر ناموجود را تخمین می‌زند. نسخه‌ای دیگر از الگوریتم در حالت نیمه‌نظارتی ارائه می‌شود [12]. نسخه‌ای دیگر نیز در شرایطی ارائه می‌شود که ویژگی‌ها بتوانند انواع مختلفی داشته باشند و تنها منحصر به مقادیر عددی نباشند [13]. تفاوت GLSC با الگوریتم پیشنهادی در این است که عمومی نیست و همچنین در آن هیچ حد بالایی برای خطای تخمین ویژگی‌های ناموجود تعیین نشده است. ممکن است برخی از ویژگی‌ها هیچ رابطه خطی با یکدیگر نداشته باشند که در این الگوریتم برای حل مشکل مذکور اقدامی انجام نشده است.

۳- الگوریتم پیشنهادی

در این بخش، یک الگوریتم برای طبقه‌بندی جویبارداده در فضای ویژگی پویا ارائه می‌شود. برای سادگی، طبقه‌بندی دودویی درنظر گرفته می‌شود؛ اما به‌راحتی می‌توان با به‌کارگیری روش‌هایی مثل OvR آن را برای استفاده در مسائلی با تعداد طبقه بیشتر، تعمیم داد [14]. در هر لحظه t یک نمونه $x_t \subset \mathbb{F} \times \mathbb{R}$ دریافت می‌شود که $\mathbb{F} = \{f_1, f_2, f_3, \cdots\}$ مجموعه همه ویژگی‌های ممکن و $\mathbb{R}$ مجموعه اعداد حقیقی را نشان می‌دهد. پس از پیش‌بینی برچسب $\hat{y}_t = \mathcal{H}(x_t)$ برچسب صحیح $y_t \in \{+1, -1\}$ آشکار می‌شود. طبقه‌بند $\mathcal{H}$ با استفاده از نتیجه پیش‌بینی و برچسب صحیح به‌روزرسانی می‌شود. در ادامه نشان داده می‌شود که چگونه می‌توان با به‌کارگیری تکنیک نگاشت ویژگی‌ها و تخمین مقادیر ویژگی‌های ناموجود x'_t پیش‌بینی بهتری ارائه نمود ($\hat{y}_t = \mathcal{H}(x_t \cup x'_t)$).

۳-۱- ساختمان‌داده

برای ذخیره‌سازی اطلاعات و مدل‌سازی روابط بین ویژگی‌ها، لازم است یک ساختمان‌داده مناسب انتخاب شود. در مسئله طبقه‌بندی جویبارداده در فضای ویژگی پویا، با دریافت هر نمونه ممکن است ویژگی‌های جدیدی اضافه یا برخی از ویژگی‌های قدیمی حذف شود. همچنین ممکن است روابط تازه‌ای بین ویژگی‌ها کشف شود یا حتی برخی از روابط ناکارآمد گذشته حذف شود. بنابراین به‌دلیل طبیعت پویای مسئله، استفاده از ساختمان‌داده گراف و بازنمایی آن به‌وسیله لیست مجاورت می‌تواند مناسب باشد.

هر گراف متشکل از تعدادی رأس و یال است. در این مسئله، به‌ازای هر ویژگی یک رأس ایجاد می‌شود و در صورتی که بین هر دو ویژگی ارتباطی وجود داشته باشد یک یال (نگاشت) بین آنها ایجاد خواهد شد. گراف می‌تواند جهت‌دار یا غیرجهت‌دار باشد. زمانی که رابطه بین ویژگی‌ها خطی باشد، می‌توان از گراف غیرجهت‌دار استفاده نمود. درصورتی که رابطه بین ویژگی‌ها

چندجمله‌ای و غیرخطی باشد، لازم است از گراف جهت‌دار استفاده شود؛ زیرا در چنین شرایطی نگاشت دوطرفه ویژگی‌ها با استفاده از تنها یک مدل رگرسیون امکان‌پذیر نیست. بنابراین با درنظر گرفتن شرایط عمومی، حالت جهت‌دار انتخاب می‌شود.

هر ویژگی (رأس) f یک لیست از نگاشت‌های ورودی به آن را نگهداری می‌کند که با $\mathcal{I}_f$ نمایش داده می‌شود. همچنین تعدادی مشخصه دیگر نیز که برای فرآیند نرمال‌سازی لازم است، نگهداری می‌کند. هر نگاشت (یال) جهت‌دار $m_{i,j}$ که f_i را به f_j متصل می‌کند دارای پنج مشخصه است:

- $\mathcal{N}_{m_{i,j}}$: تعداد نمونه‌هایی که در آنها هر دو ویژگی f_i و f_j وجود دارد
- $\mu_{m_{i,j}}$: میانگین مقادیر ویژگی f_j
- $\mathcal{M}_{m_{i,j}}$: تابع نگاشت
- $\mathcal{E}_{m_{i,j}}$: مجموع مربع‌های خطای نگاشت
- $\bar{\mathcal{E}}_{m_{i,j}}$: مجموع مربع‌های خطای میانگین

۳-۲- نرمال‌سازی

در اغلب مجموعه‌داده‌ها، مقادیر خام مربوط به ویژگی‌ها دارای بازه‌ها و مقیاس‌های متفاوتی هستند. بر همین اساس، در برخی از الگوریتم‌های یادگیری ماشین برای اینکه توابع بهینه‌سازی به‌درستی عمل کنند، لازم است نرمال‌سازی انجام شود. همچنین این اقدام تأثیر بسزایی در تسریع همگرایی گرادیان کاهشی دارد [15].

یکی از روش‌های رایجی که در بسیاری از پژوهش‌های یادگیری ماشین مورد استفاده قرار گرفته و کارایی خوبی داشته است، نرمال‌سازی براساس توزیع نرمال استاندارد است [16]. در این روش برای تبدیل هر مقدار، تفاضل آن از میانگین ویژگی مربوطه محاسبه شده و حاصل بر انحراف معیار تقسیم می‌شود. پس از نرمال‌سازی با استفاده از این روش، میانگین و واریانس مقادیر مربوط به هر ویژگی، به‌ترتیب به صفر و یک تغییر پیدا می‌کند. اما هنگام کار با جویبارداده، شرایط کمی متفاوت است؛ زیرا برای محاسبه میانگین و انحراف معیار، برخلاف پردازش دسته‌ای به همه نمونه‌ها دسترسی نداریم. برای حل این مشکل می‌توان از آمار متحرک استفاده نمود که در آن مقادیر مربوط به میانگین و انحراف معیار تخمین زده شده و با ارائه هر نمونه جدید، به‌روزرسانی می‌شود [17]. بنابراین فرآیند نرمال‌سازی به‌صورت فرمول (۱) انجام خواهد شد:

$$
\begin{aligned}
&\forall f \in \mathcal{S}_{x_t}: \\
&\quad \mathcal{N}_{t,f} = \mathcal{N}_{t-1,f} + 1 \\
&\quad \mu_{t,f} = \mu_{t-1,f} + \frac{x_{t,f} - \mu_{t-1,f}}{\mathcal{N}_{t,f}} \\
&\quad s_{t,f} = s_{t-1,f} + (x_{t,f} - \mu_{t-1,f}) \times (x_{t,f} - \mu_{t,f}) \\
&\quad \sigma_{t,f} = \sqrt{\frac{s_{t,f}}{\mathcal{N}_{t,f}}} \\
&\quad x_{t,f} = \frac{x_{t,f} - \mu_{t,f}}{\sigma_{t,f}}
\end{aligned}
\qquad (۱)
$$

که $\mu_{t,f}$ میانگین، $s_{t,f}$ مجموع مربع‌ها و $\sigma_{t,f}$ انحراف معیار ویژگی f در لحظه t را نشان می‌دهد. مجموعه ویژگی‌های نمونه x_t نیز با $\mathcal{S}_{x_t}$ نمایش داده شده است.

۳-۳- نگاشت ویژگی‌ها

نگاشت ویژگی‌ها یک مسئله رگرسیون است که در آن رابطه بین هر جفت از ویژگی‌ها یاد گرفته می‌شود. رابطه بین دو ویژگی ممکن است خیلی ساده به‌صورت خطی یا اینکه به‌صورت عمومی با درجه بالاتر و چندجمله‌ای درنظر گرفته شود [18]. در این مورد، حالت عمومی که رگرسیون چندجمله‌ای است را درنظر می‌گیریم. در این شرایط، تابع نگاشت تک‌متغیره به‌صورت فرمول (۲) تعریف می‌شود:

$$\mathcal{M}(x_i) = \sum_{d=0}^{\mathcal{D}} \theta_d x_i^d = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \cdots + \theta_{\mathcal{D}} x_i^{\mathcal{D}} \qquad (۲)$$

که $\mathcal{D}$ ابرمؤلفه[۱۲] درجه و x_i مقدار مربوط به ویژگی f_i از نمونه x را نشان می‌دهد.

هدف از مسئله تحلیل رگرسیون، یادگیری مؤلفه‌های ناشناخته θ است. یکی از روش‌های یادگیری یا تخمین مؤلفه‌ها، تعریف یک تابع زیان و کمینه‌سازی مقدار آن در تکرارهای متوالی است. یکی از توابع زیان رایج در این زمینه، زیان مربع (L2) است. که به صورت فرمول (۳) تعریف می‌شود:

$$J(\theta) = (\mathcal{M}(x_i) - x_j)^2 \qquad (۳)$$

برای کمینه‌سازی مقدار تابع زیان، از تکنیک معروف گرادیان کاهشی تصادفی (SGD)[۱۳] که یک الگوریتم بهینه‌سازی تکرارشونده مرتبه اول است استفاده می‌شود. از این الگوریتم برای یافتن کمینه محلی یک تابع مشتق‌پذیر استفاده می‌شود. در این الگوریتم، گام‌هایی در جهت مخالف گرادیان تابع در نقطه فعلی برداشته می‌شود تا به‌مقدار کمینه نزدیک شود [19]. البته هنگامی که با جویبارداده سروکار داریم، بهتر است به‌جای واژه تصادفی، از واژه برخط استفاده شود؛ زیرا در این حالت نمونه‌ها به‌ترتیب و به‌صورت متوالی در هر لحظه دریافت می‌شوند و برخلاف پردازش دسته‌ای، هیچ عامل تصادفی در انتخاب نمونه‌ها نقشی ندارد. به‌منظور به‌کارگیری این تکنیک، لازم است بردار گرادیان محاسبه شود:

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \frac{\partial J(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_{\mathcal{D}}} \end{bmatrix} = \begin{bmatrix} 2(\mathcal{M}(x_i) - x_j) \\ 2x_i(\mathcal{M}(x_i) - x_j) \\ 2x_i^2(\mathcal{M}(x_i) - x_j) \\ \vdots \\ 2x_i^{\mathcal{D}}(\mathcal{M}(x_i) - x_j) \end{bmatrix} \qquad (۴)$$

در هر لحظه t که یک نمونه x_t از جویبارداده دریافت می‌شود، لازم است نگاشت‌های بین هر جفت از ویژگی‌های موجود در آن نمونه به‌روزرسانی شود. با این کار به‌تدریج رابطه بین ویژگی‌ها یاد گرفته می‌شود. به‌روزرسانی هر

نگاشت، با انتساب مقادیر جدید به هر پنج مشخصه آن انجام می‌شود. مقدار جدید هر مشخصه، به مقدار آن در لحظه قبل وابسته است.

فرآیند به‌روزرسانی نگاشت‌ها در فرمول (۵) بیان شده است. در این فرآیند، یک واحد به شمارنده‌ای که تعداد دفعات مشاهده همزمان ویژگی‌های دو طرف نگاشت را نگهداری می‌کند اضافه می‌شود. میانگین ویژگی که نگاشت به آن انجام می‌شود و خطای میانگین، براساس الگوریتم برخط ولفورد[۱۴] تخمین زده می‌شود [17]. خطای نگاشت نیز همان خطای باقی‌مانده است که از محاسبه تفاضل مقدار تخمین‌زده شده توسط تابع نگاشت و مقدار حقیقی ویژگی مربوطه به‌دست می‌آید. درنهایت مؤلفه‌های تابع نگاشت به‌روزرسانی می‌شود.

$$\forall f_i, f_j \in \mathcal{S}_{x_t}:$$

$$\begin{aligned}
\mathcal{N}_{t,m_{i,j}} &= \mathcal{N}_{t-1,m_{i,j}} + 1 \\
\mu_{t,m_{i,j}} &= \mu_{t-1,m_{i,j}} + \frac{x_{t,j} - \mu_{t-1,m_{i,j}}}{\mathcal{N}_{t,m_{i,j}}} \\
\bar{\mathcal{E}}_{t,m_{i,j}} &= \bar{\mathcal{E}}_{t-1,m_{i,j}} + (x_{t,j} - \mu_{t-1,m_{i,j}}) \times (x_{t,j} - \mu_{t,m_{i,j}}) \\
\mathcal{E}_{t,m_{i,j}} &= \mathcal{E}_{t-1,m_{i,j}} + (x_{t,j} - \mathcal{M}_{m_{i,j}}(x_{t,i}))^2 \\
\theta_{t,\mathcal{M}_{m_{i,j}}} &= \theta_{t-1,\mathcal{M}_{m_{i,j}}} - \eta \nabla_\theta J(\mathcal{M}_{m_{i,j}}(x_{t,i}))
\end{aligned} \qquad (۵)$$

پس از دریافت نمونه x_t و به‌روزرسانی نگاشت‌ها، می‌توان مقادیر ویژگی‌های ناموجود در نمونه را با استفاده از ویژگی‌های موجود در آن نمونه، تخمین زد. این دسته از ویژگی‌ها که نمونه x_t فاقد آنها است، قبلاً در نمونه‌های گذشته مشاهده شده و به فضای ویژگی اضافه شده‌اند. ممکن است بین هرکدام از این ویژگی‌ها و ویژگی‌هایی که در نمونه فعلی وجود دارد، رابطه‌ای وجود داشته باشد که قبلاً یاد گرفته شده است. بنابراین با استفاده از نگاشت‌های ایجاد شده در لحظه‌های قبل از t، می‌توان مقادیر ویژگی‌های ناموجود را تخمین زد و از پتانسیل کامل طبقه‌بند برای پیش‌بینی برچسب مناسب بهره‌مند شد.

مجموعه ویژگی‌های موجود در نمونه x_t با نماد $\mathcal{S}_{x_t}$ نمایش داده می‌شود و مجموعه تمام ویژگی‌هایی که تا لحظه t در نمونه‌های گذشته مشاهده شده‌اند با F_t نمایش داده می‌شود. بنابراین مجموعه ویژگی‌های ناموجود در نمونه x_t با محاسبه تفاضل این دو مجموعه $(F_t - \mathcal{S}_{x_t})$ حاصل می‌شود. برای تخمین هر یک از ویژگی‌های ناموجود، باید بهترین نگاشت‌هایی که از ویژگی‌های $\mathcal{S}_{x_t}$ به آن ویژگی وجود دارد، انتخاب شود. برای انتخاب یک نگاشت، از معیار ضریب تعیین[۱۵] که یک معیار شناخته شده در مسئله تحلیل رگرسیون است استفاده می‌شود [20]. این معیار، مطابق فرمول (۶) برای هر نگاشت محاسبه می‌شود:

$$R^2_{m_{i,j}} = 1 - \frac{\mathcal{E}_{m_{i,j}}}{\bar{\mathcal{E}}_{m_{i,j}}} \qquad R^2_{m_{i,j}} \in (-\infty, 1] \qquad (۶)$$

در واقع ضریب تعیین، کارایی مدل رگرسیون یاد گرفته شده را نسبت به مدل پایه‌ای که مستقل از هر متغیر دیگری تنها میانگین متغیر هدف را گزارش می‌کند، اندازه‌گیری می‌کند. اگر خطای نگاشت کمتر از خطای میانگین باشد، مقدار ضریب تعیین بیشتر از صفر می‌شود و نشان‌دهنده عملکرد بهتر مدل یادگرفته شده نسبت به مدل پایه است. اگر خطای نگاشت بیشتر از خطای میانگین باشد، ضریب تعیین کمتر از صفر می‌شود. در چنین شرایطی، مدل پایه از مدل یادگرفته شده بهتر عمل کرده است و استفاده از آن نگاشت برای تخمین مناسب نیست. مقدار منفی ضریب تعیین می‌تواند نشان‌دهنده عدم وجود رابطه چندجمله‌ای بین ویژگی‌های دو طرف نگاشت باشد. بنابراین یکی از شروط لازم برای انتخاب یک نگاشت، به‌صورت فرمول (۷) خواهد بود:

$$R^2_{m_{i,j}} > \mathcal{R} \qquad \mathcal{R} \in [0, 1) \qquad (۷)$$

که $\mathcal{R}$ یک ابرمؤلفه است که آستانه ضریب تعیین برای انتخاب یک نگاشت را مشخص می‌کند. مقدار آن اغلب برابر با صفر درنظر گرفته می‌شود؛ اما امکان اختصاص مقادیر بالاتر نیز به آن وجود دارد که در ادامه توضیحاتی پیرامون آن ارائه خواهد شد.

نکته دیگری که لازم است در انتخاب یک نگاشت مورد توجه قرار گیرد، مشخص کردن حداقل تعداد دفعات مشاهده همزمان ویژگی‌های دو طرف نگاشت است. گاهی اوقات ممکن است یک نگاشت صرفاً با یک یا دو بار به‌روزرسانی، به مقدار بالایی از ضریب تعیین دست یابد؛ اما این احتمال وجود دارد که نقاط ارائه شده نویز باشند و ویژگی‌های دو طرف نگاشت، دیگر در هیچ‌یک از نمونه‌های بعدی به‌طور همزمان مشاهده نشوند. در چنین شرایطی برای تخمین هر یک از آن دو ویژگی که در نمونه‌های بعدی ناموجود باشند، از این نگاشت که بیشترین مقدار ضریب تعیین را دارد استفاده می‌شود و مقادیر نامناسبی برای ویژگی‌های ناموجود درنظر گرفته می‌شود. بنابراین به‌منظور حل این مشکل، شرط لازم دیگری برای انتخاب یک نگاشت فرمول (۸) تعریف می‌شود:

$$\mathcal{N}_{m_{i,j}} > \mathcal{C} \qquad \mathcal{C} \in \mathbb{W} \qquad (۸)$$

که $\mathcal{C}$ یک ابرمؤلفه است که آستانه تعداد دفعات به‌روزرسانی یک نگاشت را برای انتخاب، مشخص می‌کند. $\mathbb{W}$ نشان‌دهنده مجموعه اعداد حسابی است.

درنهایت تخمین ویژگی‌های ناموجود با رعایت همه شروط مذکور، مطابق فرمول (۹) انجام می‌شود. در واقع برای تخمین هر ویژگی ناموجود، میانگین وزن‌دار مقادیر تخمین نگاشت‌های انتخابی محاسبه می‌شود. وزن‌ها همان ضریب تعیین مربوط به هر نگاشت هستند. هرچه یک نگاشت ضریب تعیین بیشتری داشته باشد، تأثیر آن در محاسبه مقدار تخمین بیشتر می‌شود.

$$\begin{aligned}
x'_t = \bigcup_{f_j \in F_t - \mathcal{S}_{x_t}} \left(f_j, \frac{\sum_{m_{i,j} \in \mathcal{I}_{f_j}} R^2_{m_{i,j}} \times \mathcal{M}_{m_{i,j}}(x_{t,i})}{\sum_{m_{i,j} \in \mathcal{I}_{f_j}} R^2_{m_{i,j}}}\right) \\
s.t. \quad f_i \in \mathcal{S}_{x_t}, \quad R^2_{m_{i,j}} > \mathcal{R}, \quad \mathcal{N}_{m_{i,j}} > \mathcal{C}
\end{aligned} \qquad (۹)$$

۳-۴- تحلیل تئوری

همان‌طور که ملاحظه شد، تخمین ویژگی‌های ناموجود مطابق رابطه (۹) انجام می‌شود. اگر برای تخمین یک ویژگی ناموجود هیچ نگاشت مناسبی یافت نشود، به‌طور ضمنی مقدار صفر برای آن درنظر گرفته می‌شود. از آنجا که نرمال‌سازی براساس توزیع نرمال استاندارد انجام شده است، در واقع صفر همان میانگین مقادیر مربوط به آن ویژگی تا لحظه t است.

بنابراین اگر $\mathcal{C}$ به‌اندازه کافی بزرگ باشد، طبق فرمول (۱۰) خطای میانگین یک حد بالا برای خطای تخمین محسوب می‌شود و عملکرد الگوریتم پیشنهادی در تخمین ویژگی‌ها هیچ‌گاه از عملکرد مدل پایه بدتر نخواهد بود.

$$\sum_{t=1}^{\infty}\sum_{f_j\in F_t-\mathcal{S}_{x_t}}\sum_{m_{i,j}\in\mathcal{I}_{f_j}} min(\mathcal{E}_{t,m_{i,j}},\bar{\mathcal{E}}_{t,m_{i,j}}) \leq \sum_{t=1}^{\infty}\sum_{f_j\in F_t-\mathcal{S}_{x_t}}\sum_{m_{i,j}\in\mathcal{I}_{f_j}} \bar{\mathcal{E}}_{t,m_{i,j}} \qquad (۱۰)$$

۳-۵- پیچیدگی زمانی و حافظه‌ای

شبه کد DCDF2M در الگوریتم (۱) بیان شده است. در هر تکرار با دریافت هر نمونه x_t اضافه شدن ویژگی‌های جدید به فضای ویژگی و نرمال‌سازی که در خطوط ۴ و ۵ مشخص شده است، در زمان $O(|x_t|)$ انجام می‌شود. به‌روزرسانی نگاشت‌ها در خط ۶ در زمان $O(|x_t|^2)$ انجام می‌شود. تخمین ویژگی‌های ناموجود در خط ۷ در زمان $O(|F_t|^2)$ انجام می‌شود. پیچیدگی زمانی پیش‌بینی برچسب و به‌روزرسانی طبقه‌بند در خطوط ۸ تا ۱۰ بستگی به طبقه‌بند انتخابی $\mathcal{H}$ دارد. درصورتی که $\mathcal{H}$ یک طبقه‌بند خطی مثل رگرسیون لجستیک[۱۶] باشد، این فرآیند در زمان $O(|F_{t+1}|)$ انجام می‌شود. بنابراین پیچیدگی زمانی الگوریتم، $O(|F_t|^2+|x_t|^2)$ خواهد بود؛ به‌شرط اینکه پیچیدگی زمانی طبقه‌بند انتخابی $\mathcal{H}$ از آن فراتر نرود.

یادگیری روابط بین ویژگی‌ها و ذخیره‌سازی نگاشت‌ها به $O(|F_t|^2)$ حافظه نیاز دارد. میزان حافظه مصرفی طبقه‌بند $\mathcal{H}$ نیز متفاوت است. درصورتی که $\mathcal{H}$ را رگرسیون لجستیک درنظر بگیریم، حافظه مصرفی آن $O(|F_t|)$ می‌شود. بنابراین پیچیدگی حافظه‌ای الگوریتم، $O(|F_t|^2)$ خواهد بود؛ به‌شرط اینکه پیچیدگی حافظه‌ای طبقه‌بند انتخابی $\mathcal{H}$ از آن فراتر نرود.

۳-۶- حذف نگاشت‌های ضعیف

همان‌طور که قبلاً اشاره شد، برخی از جفت ویژگی‌ها ممکن است هیچ رابطه خطی یا چندجمله‌ای با یکدیگر نداشته باشند. در چنین شرایطی اگر عملکرد یک نگاشت از عملکرد مدل پایه ضعیف‌تر باشد، آن نگاشت هیچ‌گاه مورد استفاده قرار نمی‌گیرد و ذخیره‌سازی و به‌روزرسانی آن در تکرارهای متخلف نه‌تنها فایده‌ای ندارد، بلکه منجربه افزایش مصرف حافظه و زمان اجرا می‌شود. برای حل این مشکل می‌توان با بررسی یک شرط در هنگام به‌روزرسانی هر نگاشت، این دسته از نگاشت‌های ناکارآمد را شناسایی و حذف نمود. بنابراین شرط فرمول (۱۱) تعریف می‌شود:

$$R^2_{m_{i,j}} \leq \mathcal{R} \quad and \quad \mathcal{N}_{m_{i,j}} > \bar{\mathcal{C}} \qquad \bar{\mathcal{C}} \in \mathbb{W} : \bar{\mathcal{C}} \geq \mathcal{C} \qquad (۱۱)$$

که $\bar{\mathcal{C}}$ یک ابرمؤلفه است که آستانه تعداد دفعات به‌روزرسانی یک نگاشت ضعیف را برای حذف، مشخص می‌کند. به‌طور ساده این شرط بیان می‌کند که اگر یک نگاشت پس از $\bar{\mathcal{C}}$ مرتبه به‌روزرسانی، شرط (۷) را ارضا نکرد، می‌توان آن را حذف نمود.

گاهی اوقات ممکن است در ابعاد بالا، منابع کافی برای به‌روزرسانی و ذخیره‌سازی همه نگاشت‌ها دردسترس نباشد. در چنین شرایطی می‌توان با

Algorithm 1 DCDF2M

Input: Classifier $\mathcal{H}$, Hyperparameters $\mathcal{D}, \mathcal{R}, \mathcal{C}$.

1: Initialize feature set: $F_t = \varnothing$
2: **for** $t = 1$ **to** $t = \infty$ **do**
3: Receive instance: x_t
4: Add new observed features: $F_{t+1} = F_t \cup \mathcal{S}_{x_t}$
5: Perform normalization using Eq. (1)
6: Update maps using Eq. (5)
7: Estimate unavailable feature values using Eq. (9): $x_t^{'}$
8: Predict class label: $\hat{y}_t = \mathcal{H}(x_t \cup x_t^{'})$
9: Receive true class label: y_t
10: Update classifier $\mathcal{H}$ using: $y_t, x_t \cup x_t^{'}$

اختصاص مقادیر بالاتر به ابرمؤلفه $\mathcal{R}$ تنها نگاشت‌های مهم‌تر را درنظر گرفت. این امر در تسریع اجرای الگوریتم و کاهش مصرف حافظه بسیار مؤثر است.

۴- ارزیابی

در این بخش عملکرد الگوریتم DCDF2M در مقایسه با دو الگوریتم GLSC و OLVF ارزیابی می‌شود. معیار اصلی برای ارزیابی عملکرد الگوریتم‌ها دقت است؛ هرچند زمان اجرا و میزان حافظه مصرفی آنها نیز با یکدیگر مقایسه خواهد شد. برای شبیه‌سازی فضای ویژگی پویا، مطابق سناریوی طراحی شده در دو پژوهش اخیر عمل شده است. یک مؤلفه تحت عنوان نرخ حذف[۱۷] تعریف می‌شود و به‌ازای مقادیر 0.25، 0.5 و 0.75 آزمایش‌ها انجام می‌شود. نرخ حذف تعیین می‌کند که از هر نمونه، چه تعداد از ویژگی‌ها حذف شود. مثلاً وقتی نرخ حذف برابر با 0.5 است، از هر نمونه مجموعه‌داده که دریافت می‌شود به‌صورت تصادفی یکنواخت نصف ویژگی‌های آن حذف می‌شود. آزمایش‌ها به‌ازای هر مجموعه‌داده و هر نرخ حذف، ۲۰ بار تکرار شده و میانگین نتایج گزارش می‌شود. در هر تکرار، ترتیب نمونه‌ها در مجموعه‌داده به‌صورت تصادفی عوض می‌شود.

از آنجا که الگوریتم پیشنهادی عمومی است، برای ابرمؤلفه $\mathcal{H}$، طبقه‌بندهای رگرسیون لجستیک (LR)، بیز ساده گوسی (GNB)[۱۸] و درخت هافدینگ (HT)[۱۹] انتخاب شده است. با این کار سه گونه متفاوت از الگوریتم DCDF2M ایجاد شده است. سایر ابرمؤلفه‌ها به‌صورت $\mathcal{D} = 1$، $\mathcal{R} = 0.1$، $\mathcal{C} = 5$ و $\bar{\mathcal{C}} = 20$ مقداردهی شده‌اند.

۴-۱- مجموعه‌داده‌ها

برای اینکه یک مقایسه عادلانه ارائه شود، از ۱۰ مجموعه‌داده که در دو پژوهش اخیر انتخاب شده‌اند، استفاده شده است. این مجموعه‌داده‌ها با کاربردهای مختلف برگرفته از مخزن UCI هستند که معمولاً برای انجام آزمایش‌های مربوط به طبقه‌بندی دودویی استفاده می‌شوند [21]. تعداد ویژگی‌ها و تعداد نمونه‌ها در هر مجموعه‌داده، در جدول (۱) ذکر شده است.

۴-۲- نتایج

در جدول (۲) میزان دقت الگوریتم‌ها در حالتی که نرخ حذف برابر با 0.5 است گزارش شده است. نتایج کامل برای هر سه نرخ حذف، در بخش ضمایم موجود است. به‌طور میانگین، مقادیر دقت برای دو الگوریتم GLSC و

جدول (۱) : مشخصات مجموعه‌داده‌ها

Dataset	Features	Instances
Diabetes	8	768
Magic	10	19020
German	20	1000
Svmguide3	22	1243
WDBC	30	569
Ionosphere	33	351
Spambase	57	4601
Splice	60	3190
A8A	123	22696
DNA	180	3186

جدول (۲) : دقت الگوریتم‌ها با نرخ حذف 0.5

Ds. / Alg.	DCDF2M (LR)	DCDF2M (GNB)	DCDF2M (HT)	GLSC	OLVF
Diabetes	0.698 ± 0.007	**0.700 ± 0.009**	0.691 ± 0.009	0.661 ± 0.015	0.677 ± 0.010
Magic	**0.735 ± 0.003**	0.710 ± 0.003	0.705 ± 0.005	0.699 ± 0.003	0.696 ± 0.003
German	**0.708 ± 0.010**	0.638 ± 0.032	0.697 ± 0.003	0.611 ± 0.013	0.616 ± 0.015
Svmguide3	**0.771 ± 0.005**	0.761 ± 0.020	0.767 ± 0.005	0.594 ± 0.020	0.612 ± 0.009
WDBC	**0.931 ± 0.008**	0.914 ± 0.006	0.913 ± 0.008	0.925 ± 0.009	0.916 ± 0.012
Ionosphere	0.780 ± 0.021	**0.803 ± 0.020**	0.783 ± 0.027	0.704 ± 0.018	0.685 ± 0.016
Spambase	**0.853 ± 0.007**	0.669 ± 0.013	0.671 ± 0.013	0.846 ± 0.005	0.806 ± 0.007
Splice	0.695 ± 0.007	**0.761 ± 0.007**	0.760 ± 0.008	0.677 ± 0.013	0.697 ± 0.006
A8A	**0.811 ± 0.002**	0.375 ± 0.019	0.758 ± 0.003	0.699 ± 0.003	0.686 ± 0.002
DNA	**0.827 ± 0.005**	0.814 ± 0.006	0.782 ± 0.034	0.782 ± 0.009	0.821 ± 0.004

OLVF تفاوت چندانی نداشته است. درعوض برتری الگوریتم DCDF2M نسبت به این دو الگوریتم کاملاً مشهود است؛ به‌طوری در تمام مجموعه‌داده‌ها حداقل یکی از گونه‌های الگوریتم پیشنهادی عملکرد بهتری داشته است. به‌طور میانگین با درنظر گرفتن نتایج همه مجموعه‌داده‌ها و مقادیر مختلف برای نرخ حذف، دقت الگوریتم DCDF2M نسبت به دو الگوریتم مذکور به‌میزان 7.1 درصد بهبود داشته است.

همان‌طور که قبلاً بررسی شد، پیچیدگی زمانی الگوریتم DCDF2M برابر با $O(|F_t|^2 + |x_t|^2)$ است. پیچیدگی زمانی الگوریتم GLSC و OLVF نیز به‌ترتیب برابر با $O(|x_t|^2 \times |F_t|)$ و $O(|F_t| + |x_t|)$ است. زمان اجرای الگوریتم‌ها در آزمایش تجربی که در جدول (۳) گزارش شده نیز مطابق انتظار است. OLVF از همه الگوریتم‌ها سریع‌تر اجرا شده است. به‌طور میانگین زمان اجرای DCDF2M نیز از GLSC کمتر بوده است.

میزان حافظه مصرفی الگوریتم‌ها در جدول (۴) گزارش شده است. OLVF کمترین مقدار را داشته است؛ زیرا پیچیدگی حافظه‌ای آن برابر با $O(|F_t|)$ است. پیچیدگی حافظه‌ای DCDF2M و GLSC برابر با $O(|F_t|^2)$ است. در آزمایش تجربی حافظه مصرفی GLSC از DCDF2M کمتر بوده است؛ زیرا DCDF2M برای هر جفت ویژگی ۵ مشخصه را ذخیره می‌کند اما GLSC برای هر جفت ویژگی تنها یک وزن ذخیره می‌کند.

خلاصه نتایج ارزیابی با استفاده از نمودار جعبه‌ای در شکل (۱) نمایش داده شده است.

۵- نتیجه‌گیری

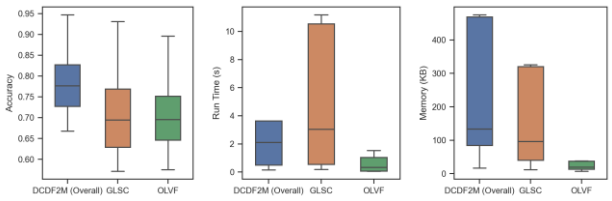
در این پژوهش، مسئله یادگیری برخط در فضای ویژگی پویا بررسی شد که در آن ویژگی‌ها می‌توانند با هر ترتیب دلخواهی تغییر کنند. یک الگوریتم عمومی جدید با استفاده از نگاشت ویژگی‌ها ارائه شد که خطای آن در تخمین ویژگی‌های ناموجود از یک حد بالا پیروی می‌کند. با انجام آزمایش‌های تجربی، عملکرد الگوریتم پیشنهادی با دو مورد از آخرین الگوریتم‌هایی که

جدول (۳) : زمان اجرای الگوریتم‌ها برحسب ثانیه

Ds. / Alg.	DCDF2M (LR)	DCDF2M (GNB)	DCDF2M (HT)	GLSC	OLVF
Diabetes	0.15	0.23	0.21	0.19	0.05
Magic	3.52	5.54	5.52	5.4	1.13
German	0.49	0.66	0.63	0.59	0.1
Svmguide3	0.71	0.94	0.89	0.7	0.11
WDBC	0.5	0.59	0.62	0.54	0.07
Ionosphere	0.47	0.57	0.6	0.41	0.05
Spambase	3.64	4.48	4.81	11.18	0.8
Splice	3.62	4.42	4.75	8.61	0.57
A8A	35.11	44.81	46.33	233.4	7.71
DNA	32.34	34.34	36.0	69.45	1.53

جدول (۴) : حافظه مصرفی الگوریتم‌ها برحسب کیلوبایت

Ds. / Alg.	DCDF2M (LR)	DCDF2M (GNB)	DCDF2M (HT)	GLSC	OLVF
Diabetes	17	67	79	12	7
Magic	42	104	772	14	8
German	77	202	224	40	13
Svmguide3	107	238	335	43	14
WDBC	127	315	345	90	20
Ionosphere	141	347	380	103	20
Spambase	450	808	1682	303	37
Splice	475	852	909	326	38
A8A	1950	2723	9837	1319	75
DNA	5516	6653	6823	3234	119



شکل (۱) : مقایسه عملکرد الگوریتم‌ها

فرض‌های یکسانی در تعریف مسئله داشته‌اند مقایسه شد. نتایج نشان دادند که به‌طور میانگین دقت الگوریتم ارائه شده، 7.1 درصد بیشتر است.

ضمایم

دقت الگوریتم‌ها با درنظر گرفتن مقادیر مختلف برای نرخ حذف

Alg. / Ds.	Diabetes (0.25)	Diabetes (0.50)	Diabetes (0.75)
DCDF2M (LR)	**0.727 ± 0.011**	0.698 ± 0.007	0.661 ± 0.007
DCDF2M (GNB)	0.724 ± 0.012	**0.700 ± 0.009**	**0.668 ± 0.014**
DCDF2M (HT)	0.719 ± 0.012	0.691 ± 0.009	0.654 ± 0.012
GLSC	0.690 ± 0.013	0.661 ± 0.015	0.619 ± 0.015
OLVF	0.699 ± 0.014	0.677 ± 0.010	0.621 ± 0.016
Alg. / Ds.	Magic (0.25)	Magic (0.50)	Magic (0.75)
DCDF2M (LR)	**0.765 ± 0.002**	**0.735 ± 0.003**	**0.707 ± 0.002**
DCDF2M (GNB)	0.722 ± 0.003	0.710 ± 0.003	0.698 ± 0.005
DCDF2M (HT)	0.749 ± 0.007	0.705 ± 0.005	0.691 ± 0.006
GLSC	0.730 ± 0.013	0.699 ± 0.003	0.649 ± 0.003
OLVF	0.745 ± 0.002	0.696 ± 0.003	0.649 ± 0.003
Alg. / Ds.	German (0.25)	German (0.50)	German (0.75)
DCDF2M (LR)	**0.724 ± 0.009**	**0.708 ± 0.010**	**0.699 ± 0.002**
DCDF2M (GNB)	0.674 ± 0.035	0.638 ± 0.032	0.638 ± 0.022
DCDF2M (HT)	0.698 ± 0.004	0.697 ± 0.003	0.697 ± 0.004
GLSC	0.622 ± 0.016	0.611 ± 0.013	0.571 ± 0.015
OLVF	0.645 ± 0.009	0.616 ± 0.015	0.575 ± 0.015
Alg. / Ds.	Svmguide3 (0.25)	Svmguide3 (0.50)	Svmguide3 (0.75)
DCDF2M (LR)	**0.782 ± 0.007**	**0.771 ± 0.005**	**0.767 ± 0.005**
DCDF2M (GNB)	0.781 ± 0.014	0.761 ± 0.020	0.733 ± 0.026
DCDF2M (HT)	0.777 ± 0.007	0.767 ± 0.005	0.763 ± 0.003
GLSC	0.590 ± 0.020	0.594 ± 0.020	0.592 ± 0.015
OLVF	0.620 ± 0.013	0.612 ± 0.009	0.610 ± 0.012
Alg. / Ds.	WDBC (0.25)	WDBC (0.50)	WDBC (0.75)
DCDF2M (LR)	**0.947 ± 0.006**	**0.931 ± 0.008**	0.889 ± 0.010
DCDF2M (GNB)	0.923 ± 0.007	0.914 ± 0.006	0.890 ± 0.009
DCDF2M (HT)	0.921 ± 0.008	0.913 ± 0.008	0.883 ± 0.017
GLSC	0.931 ± 0.007	0.925 ± 0.009	**0.897 ± 0.013**
OLVF	0.928 ± 0.010	0.916 ± 0.012	0.896 ± 0.013

Alg. / Ds.	Ionosphere (0.25)	Ionosphere (0.50)	Ionosphere (0.75)
DCDF2M (LR)	0.815 ± 0.010	0.780 ± 0.021	0.712 ± 0.018
DCDF2M (GNB)	**0.828 ± 0.014**	**0.803 ± 0.020**	**0.754 ± 0.019**
DCDF2M (HT)	0.818 ± 0.018	0.783 ± 0.027	0.736 ± 0.020
GLSC	0.673 ± 0.027	0.704 ± 0.018	0.671 ± 0.027
OLVF	0.698 ± 0.011	0.685 ± 0.016	0.668 ± 0.021
Alg. / Ds.	Spambase (0.25)	Spambase (0.50)	Spambase (0.75)
DCDF2M (LR)	**0.886 ± 0.003**	**0.853 ± 0.007**	0.787 ± 0.005
DCDF2M (GNB)	0.675 ± 0.020	0.669 ± 0.013	0.644 ± 0.015
DCDF2M (HT)	0.694 ± 0.013	0.671 ± 0.013	0.632 ± 0.014
GLSC	0.873 ± 0.006	0.846 ± 0.005	**0.797 ± 0.007**
OLVF	0.843 ± 0.006	0.806 ± 0.007	0.752 ± 0.009
Alg. / Ds.	Splice (0.25)	Splice (0.50)	Splice (0.75)
DCDF2M (LR)	0.747 ± 0.006	0.695 ± 0.007	0.621 ± 0.009
DCDF2M (GNB)	**0.813 ± 0.007**	**0.761 ± 0.007**	**0.669 ± 0.008**
DCDF2M (HT)	0.812 ± 0.007	0.760 ± 0.008	0.664 ± 0.014
GLSC	0.708 ± 0.011	0.677 ± 0.013	0.618 ± 0.009
OLVF	0.750 ± 0.004	0.697 ± 0.006	0.626 ± 0.008
Alg. / Ds.	A8A (0.25)	A8A (0.50)	A8A (0.75)
DCDF2M (LR)	**0.825 ± 0.001**	**0.811 ± 0.002**	**0.787 ± 0.001**
DCDF2M (GNB)	0.347 ± 0.021	0.375 ± 0.019	0.415 ± 0.026
DCDF2M (HT)	0.759 ± 0.003	0.758 ± 0.003	0.758 ± 0.002
GLSC	0.707 ± 0.002	0.699 ± 0.003	0.675 ± 0.003
OLVF	0.695 ± 0.001	0.686 ± 0.002	0.652 ± 0.003
Alg. / Ds.	DNA (0.25)	DNA (0.50)	DNA (0.75)
DCDF2M (LR)	**0.878 ± 0.004**	**0.827 ± 0.005**	0.735 ± 0.005
DCDF2M (GNB)	0.860 ± 0.005	0.814 ± 0.006	0.721 ± 0.007
DCDF2M (HT)	0.836 ± 0.020	0.782 ± 0.034	0.713 ± 0.019
GLSC	0.803 ± 0.012	0.782 ± 0.009	0.718 ± 0.005
OLVF	0.865 ± 0.003	0.821 ± 0.004	**0.740 ± 0.006**

مراجع

[1] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, p. 1–37, 2014.

[2] M. M. Masud, Q. Chen, L. Khan, C. Aggarwal, J. Gao, J. Han and B. Thuraisingham, "Addressing concept-evolution in concept-drifting data streams," in *2010 IEEE International Conference on Data Mining*, 2010.

[3] M. M. Masud, Q. Chen, J. Gao, L. Khan, J. Han and B. Thuraisingham, "Classification and novel class detection of data streams in a dynamic feature space," in *Joint European conference on machine learning and knowledge discovery in databases*, 2010.

[4] J. P. Barddal, H. M. Gomes and F. Enembreck, "A survey on feature drift adaptation," in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2015.

[5] Y. He, B. Wu, D. Wu, E. Beyazit, S. Chen and X. Wu, "Toward mining capricious data streams: A generative approach," *IEEE transactions on neural networks and learning systems*, vol. 32, p. 1228–1240, 2020.

[6] E. Beyazit, J. Alagurajah and X. Wu, "Online learning from data streams with varying feature spaces," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[7] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang and X. Wu, "Online learning from trapezoidal data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, p. 2709–2723, 2016.

[8] B.-J. Hou, L. Zhang and Z.-H. Zhou, "Learning with feature evolvable streams," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] Z.-Y. Zhang, P. Zhao, Y. Jiang and Z.-H. Zhou, "Learning with feature and distribution evolvable streams," in *International Conference on Machine Learning*, 2020.

[10] B.-J. Hou, Y.-H. Yan, P. Zhao and Z.-H. Zhou, "Storage fit learning with feature evolvable streams," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[11] H. Lian, J. Atwood, B.-J. Hou, J. Wu and Y. He, "Online Deep Learning from Doubly-Streaming Data," in *Proc. of the 30th ACM International Conference on Multimedia (MM)*, 2022.

[12] Y. He, X. Yuan, S. Chen and X. Wu, "Online learning in variable feature spaces under incomplete supervision," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[13] D. Wu, S. Zhuo, Y. Wang, Z. Chen and Y. He, "Online Semi-Supervised Learning with Mix-Typed Streaming Features," in *Proc. of the 37th AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[14] C. M. Bishop and N. M. Nasrabadi, Pattern recognition and machine learning, vol. 4, Springer, 2006.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015.

[16] G. Joel, "Data science from scratch," *O'Reilly Media*, 2015.

[17] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, p. 419–420, 1962.

[18] H. O. Hartley, "The modified Gauss-Newton method for the fitting of non-linear regression functions by least squares," *Technometrics*, vol. 3, p. 269–280, 1961.

[19] . Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, 462–466, 1952.

[20] N. R. Draper and H. Smith, Applied regression analysis, vol. 326, John Wiley & Sons, 1998.

[21] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2017.

زیرنویس‌ها

1 Offline
2 Batch processing
3 Data Stream
4 Concept Drift
5 Concept Evolution
6 Feature Drift
7 Feature Evolution
8 Varying Feature Space
9 Data Stream Classification in Dynamic Feature Space Using Feature Mapping
10 Classifier
11 Trapezoidal Data Stream
12 Hyperparameter
13 Stochastic Gradient Descent
14 Welford
15 Coefficient of Determination
16 Logistic Regression
17 Removing Ratio
18 Gaussian Naive Bayes
19 Hoeffding Tree