# تشخیص داده‌های پَرت در داده‌های جریانی با استفاده از مدل مبتنی بر QLattice و یادگیری برخط

**سحر فردین[۱و۲] و مهدی هاشم‌زاده[۱و۲]**

[۱]دانشکده فناوری اطلاعات و مهندسی کامپیوتر دانشگاه شهید مدنی آذربایجان، تبریز، ایران

[۲]آزمایشگاه تحقیقاتی هوش مصنوعی و یادگیری ماشین دانشگاه شهید مدنی آذربایجان، تبریز، ایران

## چکیده

**تشخیص داده‌های پَرت در جریان داده (داده‌های جریانی)، که ویژگی‌های خاصی نظیر نامحدودبودن و گذرابودن را دارند، چالش‌های زیادی دارد. برای این منظور، در این پژوهش، یک رویکرد مبتنی بر مدل رده‌بندی QLattice، که بر مبنای محاسبات کوانتوم کار می‌کند و در کاربرد مورد هدف عملکرد بهتری نسبت به دیگر روش‌های رده‌بندی دارد، معرفی می‌کنیم. با توجه به امکان تغییر توزیع داده‌ها در طول زمان در داده‌های جریانی، طرحی برای بهره‌گیری از یادگیری افزایشی برخط نیز در روش پیشنهادی ارائه می‌شود. با توجه به نامحدودبودن جریان داده‌ها و حافظهٔ پردازشی محدود، فرآیند تشخیص بر روی پنجره‌ای از داده‌ها که همواره با داده‌های نمونه‌برداری شده از پنجره‌های قبلی به‌روزرسانی می‌شود، اعمال می‌گردد. تابعی نیز برای حل مشکل نامتوازن‌بودن داده‌ها طراحی شده که از روش نمونه‌برداری برای حل این مشکل بهره می‌گیرد. نتایج آزمایش‌ها نشان می‌دهد که رویکرد پیشنهادی دقت عملکرد بهتری نسبت به روش‌های دیگر دارد.**

**واژگان کلیدی: تشخیص داده پرت، جریان داده، یادگیری برخط، یادگیری افزایشی و داده‌کاوی.**

# Outlier Detection on Data Streams Using a QLattice-based Model and Online Learning

**Sahar Fardin[1,2] & Mahdi Hashemzadeh[1,2]**

[1]Faculty of Information Technology and Computer Engineering,
Azarbaijan Shahid Madani University, Tabriz, Iran

[2]Artificial Intelligence and Machine Learning Research Laboratory,
Azarbaijan Shahid Madani University, Tabriz, Iran

## Abstract

**With the advancement of computer science, the dramatic developments in data mining area and their increasing applications, the identification of outlier or anomaly data has also become one of the most important research topics. In most applications, the outlier data contain beneficial information that can be used to gain useful knowledge. Today, there are a large number of applications on data streams, in the vast majority of which the discovery of outlier/anomaly data is very important and in some cases vital. Detection of anomalies is an important way for detecting frauds, network intrusion detection, detection of abnormal behaviors in monitoring systems, and other rare events that are always of great importance; but they are often difficult to identify. Most of the existing efficient outlier detection algorithms have been designed for the static data. While outlier detection is more challenging in data streams, where data are generating continuously and has especial properties such as infinity and transience. In this research, we introduce an approach based on the QLattice classification model, which works based on the quantum computing and performs better in the intended application than other classification methods. Given the possibility of changing the distribution of data over time in streaming data, a scheme to take advantage of online incremental learning is also applied in the proposed method. Considering the unlimited data flow and limited processing memory, the detection process is applied to a window of data that is constantly updated with data sampled from previous windows. A function is**

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

• تاریخ ارسال مقاله: ۱۴۰۰/۱/۲۹ • تاریخ پذیرش: ۱۴۰۱/۲/۲۱ • تاریخ انتشار: ۱۴۰۲/۷/۳۰ • نوع مطالعه: پژوهشی





[ DOI: 10.61186/jsdp.20.2.81 ]

**also designed to solve the problem of data imbalance, which uses the random sampling technique to solve this issue. The results of experiments obtained on benchmark datasets show that the proposed approach has better performance than other methods.**

**Keywords: Outlier detection – Data streams – Online learning – Incremental learning – Data mining**

## ۱- مقدمه

با پیشرفت روزافزون فناوری اطلاعات، تعداد پایگاه‌های داده روزبه‌روز بیشتر شده و نیاز به هوشمندسازی سامانه‌های اطلاعاتی نیز افزایش یافته است. داده‌های زیادی در حوزه‌های کاری مختلف در رایانه‌ها ذخیره می‌شوند؛ اما به‌طور عمومی این داده‌های خام، به‌تنهایی مفهوم خاصی ندارند و ما باید دانش مفید را از این داده‌ها استخراج کنیم. در کنار حجم انبوهی از داده‌های ایستا که در پایگاه‌های داده ذخیره می‌شوند، امروزه جریان سیل‌آسایی از داده‌های جریانی (جریان داده)[1] نیز همواره در حال تولیدشدن هستند. داده‌های تولیدشده توسط کاربردهایی نظیر شبکه‌های حسگر، سامانه‌های کارت‌های اعتباری، سامانه‌های نظارت بر ترافیک شبکه، دوربین‌های نظارتی و ده‌ها کاربرد حیاتی دیگر در قالب جریان‌های داده هستند [1-7]. یک جریان داده دنباله پیوسته و نامحدودی از داده‌ها است که با برچسب‌های زمانی مرتب شده‌اند. این نوع از داده‌ها، غالبا نرخ ورود بالایی دارند و توزیع آن‌ها به‌مرورزمان تغییر می‌کند [8]. داده‌های جریانی تفاوت زیادی با داده‌های ایستا دارند؛ که ما را در استفاده از الگوریتم‌هایی که برای داده‌های ایستا استفاده می‌شود، محدود و ناتوان می‌کنند.

علم داده‌کاوی کاربردهای زیادی در حوزه‌های مختلف دارد که یکی از این کاربردها، شناسایی داده‌های پرت[2] و الگوهای نامتعارف[3] است. داده‌کاوی فرآیند کشف الگوهای جالب و دانش سودمند از میان حجم انبوهِ از داده‌ها است. منابع داده‌ای می‌توانند شامل پایگاه‌های داده، انباره داده‌ها و یا داده‌هایی باشند که به‌صورت پویا و جریانی به سامانه تزریق می‌شوند [9]. در میان شاخه‌های مختلف علم داده‌کاوی، روش‌های خوشه‌بندی، رده‌بندی، کاوش قوانین انجمنی، خلاصه‌سازی و رگرسیون و تشخیص داده‌های پرت به‌شکل گسترده‌تری در کاربردهای دنیای واقعی به‌کار گرفته می‌شوند. تشخیص داده‌های پرت فرآیندی برای یافتن اشیای داده‌ای است که رفتار آن‌ها متفاوت‌تر از انتظار است. این اشیا داده‌های پرت یا ناهنجاری‌ها نامیده می‌شوند. بر اساس تعریف *هاوکینز*، دادهٔ پرت، داده‌ای است که انحراف زیادی نسبت به دیگر داده‌های موجود در مجموعه‌داده دارد و این انحراف به‌قدری است که به نظر می‌رسد این داده با سازوکار متفاوتی تولید شده است [10]؛ اما تشخیص داده‌های پرت در داده‌های جریانی، چالش‌های مهمی دارد که باید راه‌حل‌های کارآمد برای رفع آن‌ها داشته باشیم. یکی از چالش‌های مهم، امکان تغییر توزیع این داده‌ها در طول زمان است. با توجه به این‌که داده‌های جریانی، دارای تغییرات پویای قابل توجهی هستند، توزیع و الگوی رفتاری داده‌های ورودی به سامانه در طول زمان تغییر می‌یابد. به این تغییر در توزیع و رفتار داده‌ها که به‌مرورزمان به وجود می‌آید پدیدهٔ «رانش مفهوم[4]» گفته می‌شود [11]. این پدیده باعث می‌شود، به‌کارگیری روش‌های تشخیص داده‌های پرتِ سنتی، که اغلب برای داده‌های ایستا طراحی شده‌اند، با مشکلات عدیده‌ای مواجه شود؛ چون در قریب به اتفاق این روش‌ها فرض بر این است که کل داده‌ها به‌صورت یک‌جا و مستقل از زمان در اختیار ما قرار دارد [12, 13].

دومین چالش برای تشخیص ناهنجاری در جریان داده‌ها، حجم بالای داده‌های ورودی است. ازاین‌رو، پردازش چندین‌باره هر داده به‌دلیل مقدار غیر قابل کنترل داده‌ها که با سرعت بالایی در حال تولیدشدن هستند، به‌طورمعمول امکان‌پذیر نیست [14, 15]. ازاین‌رو، با توجه به نیاز روزافزون برای تجزیه و تحلیل داده‌های پرسرعت، تشخیص ناهنجاری‌ها چالش برانگیزتر نیز می‌شود؛ زیرا مانند روش‌های سنتیِ تشخیص ناهنجاری، دیگر نمی‌توان تمام داده‌ها را برای پردازش ذخیره کرد. درواقع علاوه‌بر حجم پردازش بالا، مشکل محدودیت حافظه پردازشی نیز وجود دارد که سومین چالش بزرگ ما در تشخیص ناهنجاری‌ها در جریان‌های داده است [14, 16].

از دیدگاه علم یادگیری ماشین، الگوریتم‌های تشخیص داده پرت را می‌توان به سه دستهٔ نظارت‌شده، نیمه‌نظارت‌شده و بدون‌نظارت رده‌بندی کرد [17]. تفاوت آن‌ها اغلب به‌خاطر وجود داده‌های برچسب‌گذاری

[1] Data streams
[2] Outlier
[3] Anomaly patterns
[4] Concept drift





[ DOI: 10.61186/jsdp.20.2.81 ]

شده‌است. روش‌های مبتنی بر یادگیری تحت نظارت، که مورد پژوهش ما نیز در این مقاله است، به‌طور معمول مستلزم آموزش و استخراج یک مدل پیش‌بینی بر اساس داده‌های دارای برچسب (مجموعه آموزشی) و استفاده از آن برای رده‌بندی داده‌های جدید است [18]. به عبارت دیگر، مسئلهٔ تشخیص داده پرت یک مسئله رده‌بندی[1] خواهد بود که ما در آن به دنبال ردهٔ اقلیت هستیم؛ یعنی داده‌هایی که با داده‌های دیگر متفاوت هستند. برخی از الگوریتم‌های یادگیری که در این زمینه استفاده شده‌اند، عبارتند از: ماشین‌های بردار پشتیبان[2]، شبکه‌های عصبی مصنوعی و **KNN**[3]. بعنوان مثال یک نوع خاصی از شبکه‌های عصبی مصنوعی توسط *هاوکینز و همکاران* [19] معرفی شده است که از **RNN**[4] برای تشخیص داده پرت استفاده می‌کند. قریب به اتفاق این الگوریتم‌ها و روش‌های بهبود یافته آنها نتایج مناسبی در تشخیص داده‌های پرت بر روی داده‌های ایستا از خود نشان داده‌اند؛ با این‌حال به‌دلیل ماهیت پویای جریان داده‌ها و چالش‌های یادشده، اغلب الگوریتم‌های موجود در تحلیل داده‌های جریانی با مشکلات جدی مواجه می‌شوند [21, 20]؛ بنابراین لازم است روش‌های تحلیل مناسب و مختص داده‌های جریانی را طراحی کنیم.

تاکنون برخی روش‌های تشخیص داده‌های پرت برای جریان داده ارائه شده‌اند که به‌طورکلی می‌توان آنها را به سه دسته دسته‌بندی کرد: ۱) روش‌های مبتنی بر آمار، ۲) روش‌های مبتنی بر خوشه‌بندی و نزدیک‌ترین همسایه، و ۳) روش‌های مبتنی بر ایزوله‌سازی [22].

**روش‌های مبتنی برآمار:** رویکردهای مبتنی بر آمار به‌طورکلی مدلی ایجاد می‌کنند که رفتار طبیعی مجموعه‌داده‌ها را مشخص می‌کند. داده‌های ورودی جدیدی که تناسبی با مدل ندارند یا تناسب خیلی کمی با آن دارند، ناهنجاری تلقی می‌شوند. بعضی از این روش‌ها بر اساس درجه انحراف از مدل، به داده‌ها امتیاز می‌دهند [23]. روش‌های مبتنی بر آمار می‌توانند پارامتری باشند، که دراین‌صورت آن‌ها به دانش قبلی در مورد توزیع مجموعه‌داده نیاز دارند. آن‌ها می‌توانند به‌صورت غیر پارامتری نیز باشند که در این صورت باید توزیع پایه را از مجموعه‌داده یاد گرفته و استنباط کنند. البته در داده‌های جریانی، چنین دانش قبلی همیشه در دسترس نیست.

**روش‌های مبتنی بر خوشه‌بندی و نزدیک‌ترین همسایه:** روش‌های خوشه‌بندی و نزدیک‌ترین همسایه، بر اساس نزدیکی و شباهت بین داده‌ها کار می‌کنند. الگوریتم‌های این گروه، یا روش‌های مبتنی بر فاصله هستند و یا روش‌های مبتنی بر تراکم. روش‌های خوشه‌بندی با توجه به شباهت بین داده‌ها، مجموعه‌داده را به خوشه‌های مختلف تقسیم می‌کنند و دورترین خوشه یا خوشه‌ای که دارای کمترین چگالی است را به‌عنوان یک خوشه پرت در نظر می‌گیرند [25, 24]. روش نزدیک‌ترین همسایه، با محاسبه فاصلهٔ بین همهٔ داده‌ها، همسایه‌های یک داده را پیدا می‌کند. داده‌ای که از **K**-اُمین نزدیک‌ترین همسایه دورتر باشد، به‌عنوان داده پرت در نظر گرفته می‌شود [26]. در روش‌های مبتنی بر چگالی، با محاسبه یک ضریب ناهنجاری محلی (LOF[5])، چگالی داده‌ها در مجموعه‌داده نسبت به چگالی همسایه‌های آن مورد بررسی قرار می‌گیرد و درصورتی‌که چگالی داده نسبت به همسایه‌ها کم‌تر باشد، احتمال پرت‌بودن آن داده زیاد خواهد شد [27]. این رویکردها نیاز به محاسبه فاصله یا تراکم بین همهٔ داده‌ها دارند و یا باید دانش قبلی در مورد مجموعه‌داده داشته باشند؛ بنابراین مشکل مصرف **CPU** و حافظه دارند. در مراجع [29, 28] پیشنهادهایی برای کاهش پیچیدگی محاسبات و کاهش مصرف حافظه در روش‌های مبتنی بر **LOF** ارائه شده است. در یکی از پژوهش‌های اخیری که مبتنی بر **LOF** انجام شده است، فضای داده به مکعب‌های متعددی تقسیم می‌شود و داده‌ها در این مکعب‌ها قرار گرفته و عمل تشخیص، که همان محاسبه ضریب ناهنجاری محلی (**LOF**) است، روی مکعب‌ها انجام می‌شود و بدین صورت پیچیدگی زمانی و مصرف حافظه کاهش می‌یابد [30]. در مرجع [31] نیز یک روش مبتنی بر کرنل برای تشخیص داده‌های پرت محلی ارائه شده است. در این روش، امتیاز **KOF**[6]، که نشان‌دهنده میزان پرت‌بودن داده‌ها است، برای هر داده محاسبه می‌شود.

**روش‌های مبتنی بر ایزوله‌سازی:** اساس کار این روش‌ها که در مرجع [32] معرفی شده‌اند، ایزوله کردن و جداسازی داده‌های غیر عادی از مجموعه‌داده ها است. فرض بر این است که داده‌های پرت بسیار متفاوت‌تر از داده‌های عادی هستند. بنابراین، آن‌ها خیلی سریع‌تر ایزوله می‌شوند. اولین الگوریتم ارائه شده بر اساس این

[1] Classification
[2] Support vector machines
[3] K Nearest Neighbors
[4] Replicator Neural Network
[5] Local Outlier Factor
[6] KDE-based Outlier Factor




[ DOI: 10.61186/jsdp.20.2.81 ]

روش، **IForest**[1] نام دارد که در مرجع [32] معرفی شده است؛ که البته محدودیت‌های مختلفی در داده‌های جریانی متفاوت از خود نشان داده است. در مرجع [33] نویسندگان برای سازگارکردن این روش با محیط‌های جریانی، از پنجره‌های لغزان استفاده کرده‌اند.

در مرجع [34]، روش **HS-Trees**[2] برای تشخیص داده‌های پرت ارائه شده است که یک مدل مبتنی بر درخت تک‌ردهه برای جریان داده‌ها است. این مدل، برای آموزش فقط به داده‌های نرمال نیاز دارد و به‌طورمعمول قادر است داده‌های پرتی را که تعداد کمی دارند، تشخیص دهد. از دیگر روش‌هایی که درهمین اواخر برای تشخیص ناهنجاری‌ها در محیط‌های جریانی ارائه شده‌اند، استفاده از یک الگوریتم یادگیری بدون‌ناظر، به نام **HTM**[3] است [35]. شبکه‌های **HTM** به‌طور مداوم و برخط، مشخصات مکانی و زمانی داده‌های ورودی را یاد می‌گیرند؛ ازاین‌رو، برای محیط‌های جریانی مناسب هستند. از الگوریتم‌های یادگیری عمیق نیز برای داده‌های جریانی استفاده شده است؛ اما به‌دلیل مشکلاتی که حین به‌روزرسانی مدل به‌وجود می‌آید، به‌طورمعمول یادگیری به‌صورت برون‌خط انجام شده و مرحلهٔ آزمون به‌صورت برخط انجام می‌شود. همچنین روش دیگری مبتنی بر یادگیری عمیق با آموزش برخط برای شناسایی تهدیدهای داخلی سامانه در مرجع [36] ارائه شده که برای کاربرد تک‌منظوره طراحی شده‌است.

در این مقاله، ما یک روش نظارت‌شده مبتنی بر یک مدل جدید یادگیری با نام **QLattice**[4] [37] را برای تشخیص داده‌های پرت در جریان داده‌ها معرفی می‌کنیم. این مدل یک رویکرد نوین در دنیای هوش مصنوعی و یادگیری ماشین است که در سال ۲۰۲۰ توسط یک شرکت پژوهشی در زمینه یادگیری ماشین به نام **Abzu**[5] به‌صورت یک ایدهٔ استارت‌آپی معرفی شده است. با این روش بهترین مدل یادگیری ماشین را برای کاربرد مورد نظر خویش می‌توانیم استخراج کنیم. این روش علاوه‌بر ساختن بهترین مدل یادگیری، از آنجایی‌که بر مبنای محاسبات کوانتوم کار می‌کند، سرعت عمل بالایی نیز دارد. با توجه به ماهیت داده‌های جریانی، که امکان تغییر توزیع داده‌ها با گذشت زمان وجود دارد، رویکرد پیشنهادی را به صورت یک مدل یادگیری افزایشی برخط[6] طراحی می‌کنیم. این مدل یادگیری دو مزیت اساسی دارد: ۱) با این روش می‌توان داده‌هایی با حجم بسیار بالا را آموزش داد. برای مثال داده‌های جریانی که به‌دلیل حجم بالای خود در حافظه جا نمی‌شوند؛ و ۲) با این روش تغییراتی که ممکن است در ماهیت و توزیع داده‌ها در طی زمان به‌وجود آید پوشش داده می‌شود. با توجه به نامحدودبودن جریان داده‌ها و داشتن حافظهٔ پردازشی محدود، فرآیند تشخیص بر روی پنجره‌ای از داده‌ها که همواره با داده‌های نمونه‌برداری‌شده از پنجره‌های قبلی به‌روزرسانی می‌شود، اعمال می‌گردد. تابعی نیز برای حل مشکل نامتوازن‌بودن داده‌ها (در آموزش رده‌بند) طراحی شده که از روش نمونه‌برداری برای حل این مشکل بهره می‌گیرد. نتایج آزمایش‌های به‌دست‌آمده بر روی مجموعه‌داده‌های استاندارد نشان می‌دهد که رویکرد پیشنهادی دقت عملکرد بهتری نسبت به روش‌های دیگر دارد. درکل نوآوری‌های رویکرد پیشنهادی از سه جنبه قابل بیان هستند:

۱) استفاده از مدل یادگیری جدید **QLattice** به‌عنوان یک استارت‌آپ در کاربرد تشخیص داده‌های پرت؛
۲) استفاده از یادگیری افزایشی برخط برای غلبه بر پدیدهٔ *رانش مفهوم* در جریان داده و
۳) بهره‌گیری از فرایند نمونه‌برداری از داده‌های قبلی در جریان داده‌ها برای حفظ تاریخچهٔ داده‌ها و حذف داده‌های گذشته جهت رفع مشکل محدودیت حافظه.

در ادامه، در بخش ۲، پیش‌زمینه لازم درخصوص مدل **QLattice** ارائه می‌شود. بخش ۳ روش پیشنهادی را توضیح می‌دهد. بخش ۴ آزمایش‌ها و نتایج به‌دست‌آمده را ارائه می‌کند. بخش ۵ مربوط به نتیجه‌گیری و کارهای آینده است.

## ۲- مدل QLattice

در این بخش، فلسفهٔ پیدایش **Qlattice**، مراحل مختلف الگوریتم آن و نحوهٔ کارکرد آن شرح داده می‌شود.

### ۲-۱- فلسفهٔ پیدایش QLattice

زمانی که راجع به یادگیری ماشین و کاربردهای آن، مانند اتومبیل‌های خودران، تشخیص چهره و یا پردازش زبان‌های طبیعی فکر می‌کنیم، به این نتیجه می‌رسیم که وجود ماشین‌های خلاقی که خودشان به‌تنهایی راجع به محیط خود آموزش دیده و خود را با محیط سازگار کنند، دور از ذهن نیست. متخصصان شرکت **Abzu**، استفاده از

[1] Isolation Forest
[2] Half-space trees
[3] Hierarchical Temporal Memory
[4] Quantum Lattice
[5] https://abzu.ai/
[6] Online incremental learning




[ DOI: 10.61186/jsdp.20.2.81 ]

اصطلاح «هوش مصنوعی» را در یادگیری ماشین رضایت‌بخش نمی‌دانند؛ زیرا تعاملات انسانی زیادی در آن وجود دارد؛ به‌طوری‌که ما انسان‌ها آبَر پارامترها[1] را تنظیم کرده و معماری را طراحی و تابع‌های فعال‌سازی[2] مختلف را تا حصول نتیجه بهتر آزمایش می‌کنیم. البته نمی‌توان انکار کرد که روش‌ها و الگوریتم‌های مختلف یادگیری ماشین در دهه‌های گذشته بسیار موفق بوده‌اند و به‌احتمال زیاد در دهه‌های آینده شاهد پیشرفت‌های بیشتری در این زمینه‌ها خواهیم بود. با این حال، این، روشِ صحیحِ ایجاد هوش مصنوعی نیست [38].

اما چگونه می‌توانیم در مسیر درست حرکت کنیم؟ یکی از نخستین اقدامات، کاهش تعاملات انسانی است. باید به ماشین اجازه داده شود تا خودش مدل صحیح را پیدا، بدون این‌که انسان معماری آن را طراحی کند. این ایده مشابه یک فرآیند تکاملی است. به‌این‌صورت که مدل‌ها نمایان‌گر انواع جمعیت و داده‌ها نمایان‌گر محیطی هستند که مدل‌ها برای زنده‌ماندن باید با آن محیط سازگار شوند. این ایده به این صورت کار می‌کند که ابتدا جمعیت اولیه مدل‌ها به‌صورت تصادفی تولید می‌شود؛ سپس داده‌ها از طریق هر مدل اجرا می‌شوند و فقط مدل‌هایی که از معیارهای انتخابی مانند معیار کمترین خطا پیروی می‌کنند، می‌توانند به نسل بعدی منتقل شوند؛ سپس نمونه جدیدی از مدل‌ها ترسیم‌شده و معماری آن به‌احتمال زیاد به نمونه‌های مناسب نسل قبلی شباهت خواهد داشت. در مرحلهٔ بعد، دور جدید انتخاب‌ها آغاز می‌شود و این روند ادامه می‌یابد [38].

پس از اجرای مراحلی که در بالا توضیح داده شد، مدل‌های انتخاب‌شده باید بهترین تطابق را با داده‌ها داشته باشند. به‌اختصار مدل‌های نهایی، نتیجهٔ فرایندی با شرایط اولیه تصادفی بودند که تکامل آنها توسط خود داده‌ها شکل گرفته است. مدلی که نه‌تنها پارامترهای خود را متناسب با داده‌ها تغییر می‌دهد، بلکه کل معماری آن را نیز تغییر می‌دهد. این الگوریتم که تحول جدیدی در مسیر هوش مصنوعی است، به‌عنوان مدل **QLattice** شناخته می‌شود که توسط تیم دانش‌بنیان **Abzu** طراحی شده است [39].

## ۲-۲- الگوریتم QLattice

**QLattice** یک رویکرد یادگیری ماشین تکاملی است که به کاربر این امکان را می‌دهد که از میان هزاران مدل بهترین مدل را متناسب با مسئله خود پیدا کند. این کار با به‌روزرسانی‌های مداوم **QLattice** با بهترین ساختارهای مدل داده انجام می‌شود. این به‌روزرسانی‌ها به الگوریتم یاد می‌دهد که کدام مدل مسئلهٔ کاربر را بهتر توصیف می‌کند و به این ترتیب فضای جستجو کاهش پیدا می‌کند و به الگوریتم اجازه می‌دهد پیشنهادهای بهتری برای مرحله بعد ارائه دهد.

**QLattice** به صورت یادگیری نظارت شده کار می‌کند. برای شروع ابتدا باید داده‌های ورودی و خروجی مشخص شوند. متغیرهای ورودی و خروجی به ترتیب نقاط ورود و خروج **QLattice** هستند که به آن‌ها رجیستر[3] گفته می‌شود. پس از تنظیم این موارد می‌توان مجموعه‌ای از مدل‌های ممکن را استخراج کرد. این مدل-ها، از تمام مسیرهایی که متغیرهای مستقل می‌توانند برای رسیدن به هدف نهایی طی کنند، ناشی می‌شوند. به مجموعهٔ این مدل‌ها **QGraph** گفته می‌شود. اصلی‌ترین کاری که **QLattice** انجام می‌دهد تولید **QGraph**ها است. یک **QGraph** مجموعه‌ای از تمام مدل‌های ممکن است که ورودی را به خروجی متصل می‌کند. همانطور که از نام آن مشخص است، مدل‌های مربوط به **QGraph**، گراف هستند. هر مدل مجموعه‌ای از گره‌ها است که توسط یال‌ها به هم متصل می‌شوند. پس از محاسبات انجام شده، مناسب‌ترین گراف‌ها که دارای کم‌ترین مقدار خطا هستند، از میان سایر نمودارهای **QGraph** انتخاب می‌شوند. اکنون **QLattice** می‌تواند با بهترین نمودار انتخاب شده به‌روز شود؛ و به این ترتیب نسل بعدی **QGraph** متشکل از نمودارهایی خواهد بود که از معماری مشابهٔ پیروی می‌کنند. سپس مراحل انتخاب مدل که در بالا توضیح داده شد تکرار می‌شوند و این حلقهٔ تکرارِ شامل دریافت **QGraph**، برازش[4] و به‌روزرسانی، تا رسیدن به نتیجهٔ دلخواه ادامه دارد. نهایتاً مدل‌ها به صورت خودکار ایجاد می‌شوند و ما در مسیر رسیدن به ماشینی که خود با محیط سازگار شده و تکامل می‌یابد، قرار می‌گیریم.

نمونه‌ای از یک گراف **QLattice** در شکل (۱) نشان داده شده است. در این شکل، مستطیل‌های با رنگ سبز نشان‌دهنده ورودی‌ها و خروجی هستند و مستطیل‌های با رنگ صورتی نشان‌دهنده تعاملات مدل هستند که اصطلاحاً هر یک از آنها را یک «تعامل[5]» می‌نامیم. هر تعامل یک مقدار ورودی دریافت می‌کند و

[1] Hyperparameters
[2] Activation Functions
[3] Register
[4] Fit
[5] Interaction

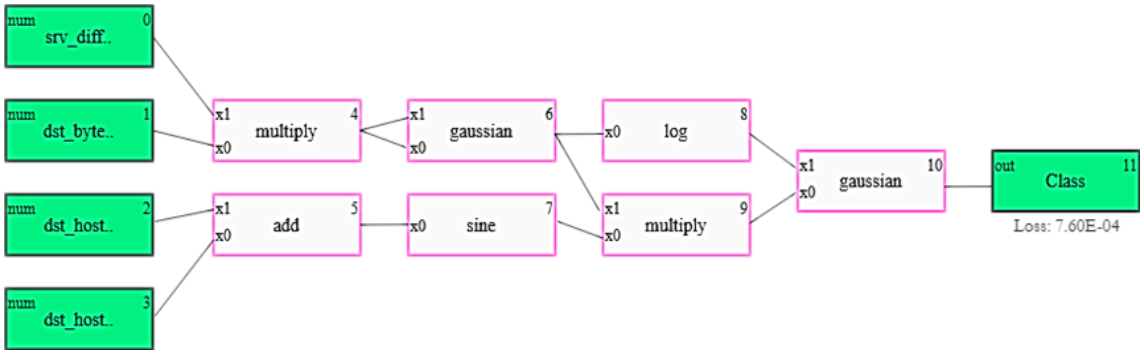



[ DOI: 10.61186/jsdp.20.2.81 ]

سپس با استفاده از یک تابع، آن مقدار را ارزیابی کرده و به تعامل بعدی منتقل می‌کند. این مدل به نوعی شبیه یک شبکه عصبی مصنوعی است که تعداد گره کمتری دارد و تابع فعال‌سازی آن روی گره‌ها یکسان نیست. هر مدل دارای یک جریان طبیعی از چپ به راست است. انواع تعاملات موجود در الگوریتم **QLattice** که برای تبدیل داده‌ها مورد استفاده قرار می‌گیرند، عبارتند از: **Add**، **Gaussian**، **Tanh**، **Sine**، **Multiply**، **Linear** و **Log**. این توابع تقریبا تمام وابستگی‌های طبیعی بین داده‌ها را پوشش می‌دهند.

گراف شکل (۱) مربوط به تشخیص داده‌های پَرت در یک مجموعه‌داده نمونه (تشخیص حمله در مجموعه دادهٔ **HTTP** که در بخش ۴ معرفی شده) است. این مجموعه‌داده دارای ۴۲ ویژگی است و در این گراف ۴ ویژگی از بین آن‌ها به‌عنوان ورودی برای مدل انتخاب شده است. همچنین برچسب داده‌ها یا ویژگی هدف به‌عنوان خروجی مدل در نظر گرفته می‌شود که ردهٔ یک وقوع حمله و ردهٔ صفر عدم وقوع حمله را نشان می‌دهد. توابعی که برای تبدیل داده‌ها استفاده می‌شوند، توسط مستطیل‌های به‌رنگ صورتی مشخص شده‌اند. برای مثال تابع **Add** دو ورودی دارد که این ورودی‌ها توسط $x_1$ و $x_0$ و وزن یال‌های آن‌ها توسط $w_1$ و $w_0$ نشان داده شده‌اند. خروجی این تابع به‌صورت $\mathrm{add}(w_0 * x_0 + w_1 * x_1 + bias)$ محاسبه می‌شود [40]. به‌طورکلی، در هر جریان، مجموعه‌داده به‌عنوان ورودی وارد شده و توسط تعامل‌ها ارزیابی و درنهایت یک پیش‌بینی به‌عنوان خروجی تولید می‌شود.

**QLattice** از میان هزاران مدل ممکن برای داده، در جستجوی گرافی است که مجموعه‌ای از ویژگی‌ها و ترکیب تعاملات درست را انتخاب کرده باشد. همچنین هنگام تولید **Qgraph**ها با استفاده از الگوریتم **QLattice**، عمق گراف‌ها را در مجموعه **Qgraph** می‌توان کنترل کرد. برای مثال عمق گراف در شکل (۱) برابر با چهار در نظر گرفته شده است. مزیتی که کاهش عمق در گراف دارد، این است که تعداد ویژگی‌های مورد استفاده را کاهش می‌دهد و مدل ساده‌تری تولید می‌کند؛ درحالی‌که افزایش عمق امکان به‌وجودآمدن معماری‌های پیچیده‌تری از مدل فراهم می‌کند. داشتن عمق کمتر به کاربر این امکان را می‌دهد تا ساده‌ترین مدلی که بتواند رابطهٔ بین متغیرهای ورودی و خروجی را توصیف کند، پیدا کند [37].




**(شکل-۱): یک مدل استخراج شده توسط الگوریتم QLattice بر روی یک داده نمونه**

**(Figure-1): A model extracted by the QLattice algorithm on a sample data**

## ۳- روش پیشنهادی

### ۳-۱- ساختار کلی

رویکرد پیشنهادی برای تشخیص داده‌های پرت یک رویکرد مبتنی بر یادگیری ماشین است که بر سه ایده اصلی استوار است: ۱) استفاده از الگوریتم **QLattice** برای استخراج بهترین مدل یادگیری، ۲) یادگیری از نوع افزایشی برخط و ۳) نمونه‌برداری از داده‌های قبلی و حفظ تاریخچه داده‌ها. شکل (۲) ساختار کلی روش پیشنهادی را نشان می‌دهد که از چهار بخش اصلی تشکیل شده است:

۱) **پیش‌پردازش داده‌ها:** در این مرحله، برخی عملیات پیش‌پردازشی برای آماده‌سازی داده‌ها جهت تحلیل و بررسی آن‌ها صورت می‌گیرد.





[ DOI: 10.61186/jsdp.20.2.81 ]

۲) **مدل یادگیری مبتنی بر QLattice**: در این مرحله، یک مدل یادگیری (رده‌بند)، که از روی داده‌های آموزشی ساخته می‌شود، برای شناسایی داده‌های پرت به‌کار گرفته می‌شود. این مرحله درواقع مرحلهٔ برون‌خط رویکرد پیشنهادی است که مبتنی بر روش **QLattice** [37] ساخته می‌شود. فرضیه ما بر این است که الگوریتم **QLattice**، در کاربرد مورد هدف، سرعت و دقت عمل بیشتری نسبت به دیگر الگوریتم‌های رده‌بندی دارد.

۳) **تشخیص داده‌های پرت:** در این مرحله، عمل تشخیص داده‌های پرت در داخل پنجره مشخصی از داده‌ها با به‌کارگیری رده‌بند آموزش دیده شده (مدل مبتنی بر **QLattice**) انجام می‌شود؛ ضمن این‌که داده‌های مورد بررسی به بخش به‌روزرسانی مدل نیز ارسال می‌شود.

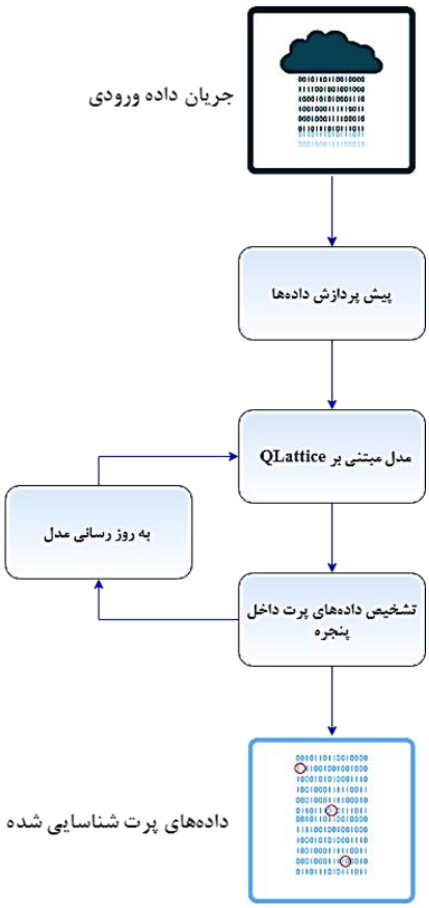
۴) **به‌روزرسانی مدل:** این مرحله از ایده یادگیری برخط الهام گرفته شده‌است. با توجه به این‌که داده‌های جریانی، دارای تغییرات پویای قابل‌توجه هستند، این امکان وجود دارد که با گذشت زمان، توزیع و الگوی رفتاری داده‌هایی که وارد سامانه می‌شوند، به‌طرز چشم‌گیری تغییر کند؛ بنابراین مدل ساخته‌شده براساس مجموعه‌داده‌های گذشته، باید مرتب به‌روز شود.

۵) درواقع، رویکرد پیشنهادی از دو مرحلهٔ یادگیری برون‌خط و یادگیری برخط تشکیل شده است. ابتدا در مرحلهٔ برون‌خط، مدل اولیهٔ رده‌بندی، با استفاده از داده‌های آموزشی ساخته، سپس در مرحلهٔ برخط، داده‌های جریانی که وارد سامانه می‌شود، در داخل یک پنجره مشخصی بررسی شده و توسط رده‌بند داده‌های پرت تشخیص داده می‌شود. این داده‌های جدید و همچنین داده‌های نمونه‌برداری‌شده از پنجرهٔ قبل برای به‌روزرسانی مدل رده‌بندی استفاده می‌شوند. در ادامه، جزئیات هر یک از مراحل یادشده به‌صورت جداگانه تشریح می‌شود.

## ۲-۳- پیش‌پردازش داده‌ها

در رویکرد پیشنهادی، چهار عمل پیش‌پردازشی به‌شرح زیر انجام می‌شود:

۱) **نرمال‌سازی داده‌ها:** انواع داده‌های ورودی به دو دستهٔ عددی[1] و اسمی[2] تقسیم می‌شوند. پیش‌فرض الگوریتم **QLattice** در نسخه موجود آن [37]، نوع عددی برای همهٔ ویژگی‌ها است. ازاین‌رو، ویژگی‌های اسمی در یک فهرست جداگانه ذخیره می‌شوند. الگوریتم **QLattice** به‌طور خودکار، عمل نرمال‌سازی داده‌های عددی را با تنظیم مقیاسی بر اساس کمترین و بیشترین مقادیر مشاهده‌شده در هر ویژگی انجام می‌دهد و همهٔ داده‌ها را در یک محدوده مشخص، بین ۱- و ۱ قرار می‌دهد. برای کدگذاری داده‌های اسمی، روش‌های مختلفی وجود دارد که یکی از این روش‌ها، روش **One-Hot Encoding** [41] است. الگوریتم **QLattice** از این روش برای کدگذاری داده‌های اسمی استفاده می‌کند. برای این منظور، فهرست ویژگی‌های اسمی در اختیار الگوریتم قرار می‌گیرد تا به‌صورت خودکار عملیات کدگذاری و تبدیل داده‌ها را انجام دهد. نمونه‌ای از این روش کدگذاری در شکل (۳) نشان داده شده است.

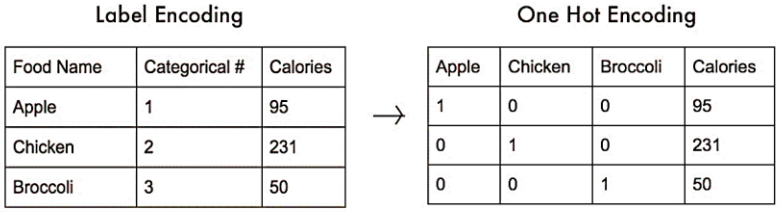



**(شکل-۲): ساختار کلی رویکرد پیشنهادی**
**(Figure-2): Structure of the proposed approach**

[1] Numerical data
[2] Nominal




[ DOI: 10.61186/jsdp.20.2.81 ]




(شکل-۳): نمونه‌ای از تکنیک One-Hot Encoding برای رمزگذاری داده‌های اسمی [42]

**(Figure-3): An example of the One-Hot Encoding technique for encrypting nominal data [42]**

۲) **انتخاب ویژگی**[1]: عمل انتخاب ویژگی را نیز می‌توانیم هم به‌صورت دستی و هم به‌وسیلهٔ خود الگوریتم **QLattice** به‌صورت خودکار انجام دهیم. در نسخه موجود از پیاده‌سازی الگوریتم **QLattice** [37]، بر اساس عملیات بهینه‌سازی که انجام می‌شود، ویژگی‌های مناسب مدل به‌صورت خودکار انتخاب می‌شود. ازاین‌رو در رویکرد پیشنهادی هر دو حالت انتخاب ویژگیِ دستی و خودکار را می‌توانیم داشته باشیم.

۳) **حذف ویژگی‌های با مقدار ثابت در بین رکوردها:** در این مرحله، از بین ویژگی‌های انتخاب‌شده، آن‌هایی که مقدار یکسانی بین همهٔ رکوردهای داده‌ای دارند را حذف می‌کنیم. ازآنجایی‌که در روش پیشنهادی، عمل جستجو بین ویژگی‌ها و انتخاب آن‌ها توسط خود الگوریتم انجام می‌گیرد و ویژگی‌هایی که دارای مقدار ثابتی بین همهٔ رکوردها هستند (یعنی برای همه رکوردها مقدار یکسانی دارند)، طبیعتاً تأثیری در عمل تشخیص داده‌های پرت و عملکرد الگوریتم نخواهند داشت، ازاین‌رو، برای سریع‌ترشدن انتخاب‌های الگوریتم، این نوع از ویژگی‌ها حذف می‌شوند.

۴) **حذف داده‌های حاوی مقادیر گم‌شده**[2]: در مرحله آخر از عملیات پیش‌پردازشی، داده‌هایی را که حاوی ویژگی‌های فاقد مقدار یا گم‌شده هستند، حذف و پاک‌سازی می‌کنیم.

## ۳-۳- مدل رده‌بندی مبتنی بر QLattice

در این بخش، نحوه به‌کارگیری الگوریتم **QLattic** برای مسئلهٔ تشخیص داده‌های پرت در جریان داده‌ها و همچنین متوازن‌سازی داده‌ها برای یادگیری در مسئلهٔ تشخیص داده پرت تشریح می‌شود.

### ۳-۳-۱- نحوه بکارگیری و استخراج مدل QLattice برای تشخیص داده‌های پرت

برای استفاده از مدل **QLattice**، ابتدا باید نوع مسئلهٔ خود را به‌عنوان یک مسئلهٔ رگرسیون و یا یک مسئلهٔ رده‌بندی مشخص کنیم. موضوع تشخیص داده‌های پرت یک مسئلهٔ رده‌بندی است که حاوی دو رده داده‌های نرمال و داده‌های پرت است. داده‌های نرمال را به‌عنوان رده صفر و داده‌های پرت را ردهٔ یک در نظر می‌گیریم. ویژگی‌های داده‌ها به‌عنوان ورودی و برچسب داده‌ها به‌عنوان خروجی به **QLattice** داده می‌شود. به این ترتیب **QLattice** شروع به تولید گراف‌هایی می‌کند که متناسب با داده‌های ما باشند. درواقع رده‌بند ما **QGraph**های تولیدشده توسط **QLattice** خواهد بود.

با توجه به این‌که **QLattice** روی ردهتر **Abzu** اجرا می‌شود (برای کسب اطلاعات بیشتر به تارنمای **Abzu**[3] مراجعه نمایید)، برای شروع کار و استفاده از آن، کاربر نیاز به یک توکن دسترسی به سامانه و نشانی **URL** اختصاصی دارد. پس از واردکردن این موارد و احراز هویت کاربر، عملیات یادگیری و استفاده از **QLattice** آغاز می‌شود. این الگوریتم، با استفاده از زبان برنامه‌نویسی پایتون نسخهٔ ۳ پیاده‌سازی شده‌است و از کتابخانهٔ **Feyn** برای ساخت مدل یادگیری **QLattice** استفاده می‌شود. در ادامه مراحل آموزش مدل توضیح داده می‌شود. این مراحل عبارتند از سه مرحله به‌شرح زیر:

۱) **برازش**[4]: برای آموزش مدل‌ها، ابتدا وزن یال‌های روی هر تعامل به‌صورت تصادفی انتخاب می‌شود؛ بنابراین مدل اولیه هیچ تناسبی با داده‌ها نخواهد داشت.

[1] Feature selection
[2] Missing value
[3] https://docs.abzu.ai/docs/guides/setup/accessing.html
[4] Fit




[ DOI: 10.61186/jsdp.20.2.81 ]

فرایند برازش در **QGraph** نیاز به دو مؤلفه دارد: ۱) داده‌های آموزشی که **QGraph** باید روی آن‌ها آموزش ببیند و ۲) تابع خطایی که برای بهینه‌سازی استفاده می‌شود. برای این منظور، تابع میانگین مجموع خطاها را استفاده می‌کنیم.

۲) **مرتب‌سازی**[1]: در این قسمت، ما باید بهترین نموداری (گرافی) را که با داده‌ها تناسب بیشتری دارد، پیدا کرده و در یک متغیر نگهداری کنیم. این کار با استفاده از مرتب‌سازی نمودارها بر اساس تابع خطای آن‌ها انجام می‌شود. به‌این‌صورت که در هر تکرار حلقه در هر دوره، تمام **QGraph**های به‌دست‌آمده براساس تابع خطایی که دارند به‌صورت صعودی مرتب شده و نخستین عنصر فهرست که دارای کمترین خطا است، به‌عنوان بهترین **QGraph** به‌دست‌آمده نگهداری می‌شود.

۳) **به‌روزرسانی**[2]: اگر هیچ یک از مدل‌های استخراج‌شده و حتی بهترین آن‌ها با داده‌های ما مطابقت نداشته باشد، **QLattice** را با بهترین مدل مرحلهٔ قبل به‌روز می‌کنیم. با این کار ما بهترین یادگیری و بهترین معماری انتخاب‌شده را به مرحله بعد منتقل می‌کنیم تا بر انتخاب نمودارهای **QGraph** در مرحله بعد تأثیر بگذارد؛ و نمودارها را به سمت ویژگی‌های نموداری که منجر به کمترین خطا در مجموعه نمودارهای قبل شد، سوق می‌دهیم. اکنون الگوریتم ما برای طراحی مرحله بعدی نمودارهای **QGraph** آماده است. این کار را تا زمانی که به نتیجه مطلوب مورد نظر نرسیده‌ایم و یا این‌که دیگر تغییراتی در تابع خطا مشاهده نشود، ادامه می‌دهیم.

با توجه به این‌که الگوریتم **QLattice** نوعی الگوریتم تکاملی است و به‌مرورزمان به حالت ایده‌آل خود نزدیک می‌شود، می‌توان از آن برای یادگیری افزایشی در محیط‌های جریانی استفاده کرد؛ همچنین با توجه به این‌که مهندسی ویژگی‌ها[3]، و اغلب عملیات پیش‌پردازشی توسط خود الگوریتم انجام می‌شود، سرعت پردازش داده‌های جریانی را بسیار بیشتر می‌کند؛ همچنین این الگوریتم برای داده‌هایی با ابعاد بزرگ مناسب است و نیازی به کاهش ابعاد در داده‌ها نداریم.

[1] Sort
[2] Update
[3] Feature engineering

### ۳-۳-۲- متوازن‌سازی داده‌ها برای یادگیری در مسئلهٔ تشخیص داده پرت

در اغلب مسائل مربوط به یادگیری ماشین، با مشکل توزیع نامتوازن رده داده‌ها مواجه هستیم. این مساله زمانی رخ می‌دهد که تعداد مشاهدات مربوط به یک رده به طور چشم‌گیری کم‌تر از مشاهداتی باشد که به رده دیگر تعلق دارند [44, 43]. این داده‌ها، اصطلاحاً داده‌های نامتوازن نامیده می‌شوند. الگوریتم‌های یادگیری ماشین در مواجهه با این نوع از داده‌ها، رده‌بندی‌های نامناسبی را انجام می‌دهند. چراکه بعلت کم بودن نمونه‌های آموزشی رده اقلیت، نمی‌توانند رده آنها را به خوبی یاد بگیرند. این مشکل در رده‌بندی داده‌های پرت، که ماهیتاً داده‌های نادر هستند، حادتر نیز می‌شود.

رویکردهای متنوعی برای حل مشکل داده‌های نامتوازن وجود دارد که تکنیک‌های نمونه‌برداری مختلفی را به کار می‌گیرند این رویکردها عبارتند از: زیر نمونه-برداری تصادفی، بیش نمونه‌برداری تصادفی و یا الگوریتم‌های پیچیده‌تر نظیر روش **SMOTE** [44, 43]. در این پژوهش، رویکردهای مختلف، نظیر موارد مذکور، مورد استفاده قرار گرفتند. اغلب رویکردها کارایی یکسانی در عملکرد کلی روش پیشنهادی از خود نشان دادند و روش پیشنهادی حساسیت خاصی به این مرحله از فرایند خود ندارد. ازاین‌رو، برای پرهیز از پیچیدگی غیر ضروری الگوریتم، از ساده‌ترین رویکرد یعنی روش زیر نمونه برداری تصادفی[4] برای متوازن سازی داده‌ها استفاده می‌کنیم. در این روش، به طور تصادفی از داده‌هایی که جزء ردهٔ اکثریت مجموعه‌داده هستند، به تعداد مشخصی نمونه‌برداری انجام می‌شود و در کنار نمونه‌های اقلیت به رده‌بند داده می‌شود. بدین ترتیب مشکل عدم توازن رده‌های داده هنگام آموزش مدل اولیه و همچنین در فرایند تکرار الگوریتم **QLattice** را برطرف می‌کنیم. با انجام مراحل فوق بر روی ده درصد داده‌های مجموعه‌داده، مدل اولیهٔ رده‌بندی (که آن را مدل $h_0$ می‌نامیم) برای تشخیص داده‌های پرت ساخته می‌شود.

### ۴-۳- تشخیص داده‌های پرت داخل پنجره

همانطور که درقبل اشاره شد، روش پیشنهادی دارای دو مرحلهٔ برون‌خط و برخط است. مرحله برون‌خط روش پیشنهادی، ساخت مدل اولیهٔ $h_0$ توسط **QLattice** بود

[4] Random under-sampling

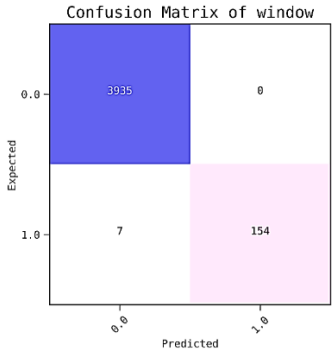



[ DOI: 10.61186/jsdp.20.2.81 ]

که در بخش قبل توضیح داده شد. در این بخش و بخش ۳-۵، مرحله برخط الگوریتم، یعنی هنگامی که داده‌های جریانی وارد سامانه می‌شوند، توضیح داده می‌شود.

برای پردازش داده‌های جریانی، با توجه به این‌که داده‌ها به‌مرورزمان وارد سامانه می‌شوند و نامحدود هستند و ممکن است فضای کافی برای ذخیرهٔ آن‌ها موجود نباشد، پنجره‌ای با اندازه مشخص برای تحلیل داده‌ها در نظر می‌گیریم. اندازه این پنجره بر اساس یک بازه زمانی مشخص و همچنین بر اساس تعداد داده‌های ورودی قابل تنظیم است. اندازه پنجره باید به‌گونه‌ای تنظیم شود که زمان کافی برای تحلیل داده‌ها داشته باشیم. در روش پیشنهادی، اندازه پنجره را بر اساس تعداد داده‌های ورودی تنظیم می‌کنیم ( آزمایش‌های لازم در این خصوص در بخش ۵-۴ ارائه شده است).

پس از این‌که تعداد داده‌های مشخص وارد پنجره می‌شوند، بهترین مدل آموزش دیده در مرحلهٔ قبل یعنی گرافی که به‌عنوان بهترین گراف را از مجموعه نمودارهای **QGraph** انتخاب شده است، به‌عنوان رده‌بند در نظر می‌گیریم و عمل پیش‌بینی روی داده‌های فعلی را انجام می‌دهیم. نمونه‌ای از نتایج پیش‌بینی مدل (در قالب ماتریس درهم‌ریختگی[1]) بر روی نمونه‌ای از داده‌هایی که وارد پنجره شده‌اند، در شکل (۴) نشان داده شده است. در این مثال، تعداد داده‌های داخل پنجره ۴۰۹۶ رکورد است؛ که در بین این داده‌ها ۱۶۱ داده پرت وجود داشته و مدل توانسته ۱۵۴ مورد از آنها را به‌درستی تشخیص دهد.




**(شکل-۴): نمونه‌ای از نتایج (ماتریس درهم ریختگی) پیش‌بینی به‌وسیله مدل QLattice روی داده‌های داخل پنجره**

**(Figure-4): Example result (confusion matrix) of prediction by QLattice model on data inside the window**

## ۳-۵- به‌روزرسانی مدل

همان‌طور که در بخش ۱ توضیح داده شد، در مسائل مربوط به رده‌بندی در جریان داده‌ها، یک مشکل اساسی مواجه‌شدن با پدیده «رانش مفهوم» است که کارایی طبقه‌بند را به‌شدت تحت تاثیر قرار می‌دهد. راه حلی که برای غلبه بر این مشکل در روش پیشنهادی در نظر گرفته‌ایم، استفاده از یادگیریِ افزایشیِ برخط است. در ادامه، ابتدا چارچوب کلی این روش یادگیری را توضیح می‌دهیم و سپس نحوه به‌کارگیری آن در رویکرد پیشنهادی را تشریح می‌کنیم.

### ۳-۵-۱- یادگیری برخط

امروزه روش‌های یادگیری ماشینِ افزایشی و برخط، در زمینهٔ یادگیری در کاربردهای جریان داده‌ای و بلادرنگ در مقابل روش‌های یادگیری دسته‌ای[2]، که کل داده‌ها در اختیار ما قرار دارد، بسیار مورد توجه قرار گرفته‌اند [45]. یادگیری برخط به حالتی گفته می‌شود که همه داده‌ها در حال حاضر موجود نیستند و به مرور وارد سامانه می‌شوند. در این نوع یادگیری، یک مدل ساخته و سپس با رسیدن داده‌های جدیدتر، این مدل به‌روزرسانی می‌شود [45]. مدل یادگیری برخط دو مزیت اساسی دارد: ۱) با این روش می‌توان داده‌هایی با حجم بسیار بالا را آموزش داد. برای مثال داده‌هایی که به‌دلیل حجم بالای خود در حافظه جا نمی‌شوند. ۲) تغییراتی که ممکن است در رفتار و توزیع داده‌ها به‌وجود آیند، با این روش پوشش داده می‌شود.

چارچوب یادگیری افزایشی برخط که در رویکرد پیشنهادی استفاده شده، در شکل (۵) نمایش داده شده است. در این چارچوب، دنبالهٔ نامحدودی از داده‌ها $S = (s_1, s_2, \dots, s_t, \dots)$ را در نظر بگیرید که به‌صورت تاپل‌های $s_i = (x_i, y_i)$ یکی پس از دیگری وارد سامانه می‌شوند. هدفِ آموزش، پیش‌بینی برچسب داده در زمان $t$ یعنی $y_t$ برای ورودی $x_t$ است. پیش‌بینی توسط مدل آموزش‌دیده در مرحله قبل یعنی $h_{t-1}$ انجام می‌شود $(y'_t = h_{t-1}(x_t))$.

در این چارچوب، همهٔ مدل‌های میانی برای ارزیابی عملکرد در نظر گرفته می‌شوند. هر یک از این مدل‌ها فقط برای پیش‌بینی داده‌های بعدی خود مورد استفاده قرار می‌گیرند و هر نمونه در ابتدا به‌عنوان داده آزمون برای مدل و سپس برای سازگار شدن مدل بعنوان داده آموزش می‌تواند استفاده شود. این چارچوب مناسب‌ترین طریق برای یادگیری داده‌های جریانی می‌تواند باشد.

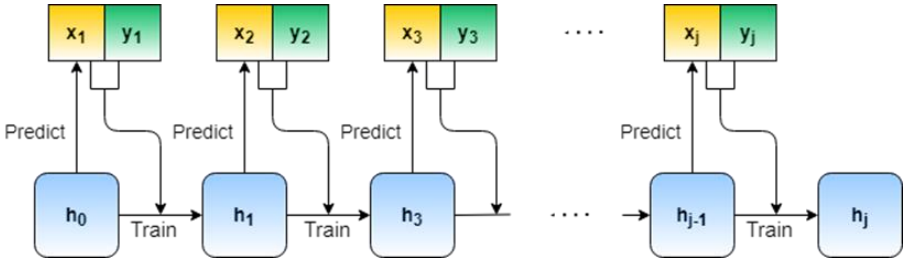
[1] Confusion matrix

[2] Batch learning





[ DOI: 10.61186/jsdp.20.2.81 ]




**(شکل-۵): چارچوب کلی الگوریتم‌های یادگیری افزایشی برخط**
**(Figure-5): General framework of online incremental learning algorithms**

### ۳-۵-۲- استفاده از یادگیری افزایشی برخط در تشخیص داده‌های پرت

برای تشخیص داده‌های پرت در جریان داده‌ها، چارچوب یادگیری افزایشی برخط مبتنی بر مدل **QLattice** را طراحی می‌کنیم. به این صورت که ابتدا مدل $h_0$ توسط یک سِری داده‌های اولیه که دارای برچسب هستند، ساخته می‌شود و پس از آنکه جریان داده‌ها وارد سامانه شد، ابتدا عمل پیش‌بینی توسط مدلِ ساخته‌شده در مرحله قبل روی داده‌های داخل پنجره انجام، سپس جهت حفظ دقت الگوریتم و داشتن تاریخچه‌ای از داده‌های قدیمی، عمل نمونه‌برداری از داده‌های قبل انجام می‌شود. الگوریتم **QLattice** روی داده‌های جدید و داده‌های نمونه‌برداری‌شده اعمال شده و با اعمال فرایند مرتب‌سازی، **QGraph** بهینه از بین **QGraph**های تولیدشده انتخاب و عمل به‌روزرسانی مدل با این **QGraph** انجام می‌شود. در تکرارهای بعد، هنگامی که داده‌ها وارد پنجره می‌شوند از آخرین مدل (مدل به‌روزرسانی‌شده) برای پیش‌بینی استفاده می‌شود. یعنی در هر مرحله، مدل یادگیری با استفاده از داده‌های پنجره فعلی و داده‌های نمونه-برداری‌شده از داده‌های قبلی به‌روزرسانی می‌شود. تابعی که برای نمونه‌برداری از داده‌های قبلی در نظر گرفته شده است، در هر بار فراخوانی تعداد متوازنی از هر دو رده غیر نرمال (پرت) و نرمال را برای حفظ دقت الگوریتم و متوازن‌سازی داده‌ها انتخاب می‌کند. داده‌های نمونه‌برداری شده به همراه داده‌های پیش‌بینی‌شده در پنجره فعلی بصورت افزایشی برای آموزش مدل و به‌روزرسانی آن استفاده می‌شود.

## ۴- آزمایش‌ها و نتایج به‌دست‌آمده

در این بخش، نتایج عملکرد رویکرد پیشنهادی را مورد ارزیابی قرار می‌دهیم. ابتدا، مجموعه‌داده‌های استفاده شده در این پژوهش و معیارهای ارزیابی را معرفی می‌کنیم. سپس نتایج آزمایش‌های صورت‌گرفته برای ارزیابی کارایی رویکرد پیشنهادی و مقایسه با دیگر روش‌های موجود را ارائه و بحث می‌کنیم.

کلیه نتایج گزارش‌شده با استفاده از یک رایانه با پردازنده اینتل **Core (TM) i5** با فرکانس ۲/۶۷ گیگا هرتز و ۳ گیگا بایت حافظه **RAM** به دست آمده است که نسخه نهایی سیستم عامل ویندوز ۱۰ را اجرا می‌کند. همچنین فرایندهای مربوط به روش پیشنهادی با زبان برنامه‌نویسی پایتون ورژن ۳ پیاده‌سازی شده‌اند.

### ۴-۱- مجموعه‌داده های ارزیابی

برای ارزیابی روش پیشنهادی از دو مجموعه‌دادهٔ استاندارد از کتابخانه یادگیری ماشین **UCI**[1] استفاده شده است؛ مجموعه‌دادهٔ نخست **KDDCup99** است که یک مجموعه‌داده بزرگ و پرکاربرد در زمینهٔ ارزیابی سامانه‌های تشخیص نفوذ در شبکه‌های رایانه‌ای است [46]. مجموعه‌دادهٔ دوم، مجموعه **Covertype** است که به‌طورمعمول در پژوهش‌های مربوط به جریان داده‌ها استفاده می‌شود که برای کاربرد پیش‌بینی نوع پوشش جنگل‌ها طراحی شده است [33]. هر دو مجموعه‌داده برچسب‌گذاری شده‌اند و برای کارهای رده‌بندی مورد استفاده قرار می‌گیرد. در ادامه، توضیحات این مجموعه‌های داده و چگونگی استفاده از آن‌ها برای مسئلهٔ تشخیص داده‌های پرت ارائه می‌شود.

**مجموعه‌داده KDDCup99**: یک مجموعه‌داده برای تشخیص نفوذ در شبکه است که به‌طور گسترده در مطالعات مربوط به تشخیص داده پرت استفاده می‌شود [34, 33, 28]. این مجموعه‌داده ، شامل ۴۸۹۸۴۳۱ رکورد داده‌ای و ۴۱ ویژگی است. به منظور امکان مقایسه

[1] https://archive.ics.uci.edu/




[ DOI: 10.61186/jsdp.20.2.81 ]

نتایج رویکرد پیشنهادی با دیگر روش‌های موجود و قابل مقایسه، نظیر مراجع [34، 33، 28]، ما نیز از سرویس‌های (داده‌های) **HTTP** و **SMTP** این مجموعه‌داده استفاده می‌کنیم. در این مجموعه‌داده، رکوردها شامل چهار دستهٔ کلی حملات شبکه‌های رایانه‌ای به‌شرح زیر هستند:

۱) **حملات منع سرویس DOS:** این حملات شامل حملات رایجی نظیر **Ping of death**، **Tear drop**، **SYN flood**،**Smurf** هستند.

۲) **حملات R2L:** این حملات نشان‌دهندهٔ دسترسی غیر مجاز به سامانه از راه دور هستند. به‌عنوان مثال حدس زدن رمز عبور.

۳) **حملات U2R:** این حملات نشان‌دهندهٔ دست‌یابی غیر مجاز به سطح دسترسی مدیر سیستم از طریق سطح دسترسی کاربر است. برای مثال: انواع حملات سرریز بافر.

۴) **حملات وارسی و تجسس:** نمونه‌هایی از این حملات عبارتند از **port-scan** و **ping-sweep**.

در این مجموعه‌داده، همهٔ رکوردها با برچسب حملاتی که در بالا نام برده شدند، به‌عنوان داده پرت در نظر گرفته می‌شوند. با توجه به این‌که تعداد داده‌ها و ابعاد این مجموعه‌داده برای تحلیل زیاد است، در بیش‌تر کارهای پژوهشی پیشین، نظیر [34، 33، 28]، فقط از چهار بُعد این مجموعه‌داده استفاده شده است؛ اما با توجه به این‌که، ما در روش پیشنهادی محدودیتی در انتخاب ابعاد داده نداریم و مدل **QLattice** برای داده‌هایی که ابعاد بیشتری دارند، بهتر هم جواب می‌دهد، می‌توانیم همه ۴۱ ویژگی این مجموعه‌داده را استفاده کنیم. با این‌حال در مقایسه نتایج روش‌ها، شرایط مساوی و عادلانه را لحاظ می‌کنیم.

**مجموعه‌داده Covertype:** این مجموعه‌داده که برای پیش‌بینی نوع پوشش جنگل‌ها به‌کار می‌رود، شامل چهار نوع منطقه جنگلی است که این مناطق جنگل‌هایی با حداقل میزان خرابی توسط انسان‌ها را نشان می‌دهند. انواع پوشش‌های جنگلی موجود، بر اساس نتیجهٔ فرآیندهای اکولوژیکی تعریف شده است. این مجموعه‌داده، شامل ۵۸۱۰۱۲ رکورد و ۵۴ ویژگی است که به‌طورمعمول فقط ۱۰ ویژگی نخست آن برای تحلیل استفاده می‌شود. این مجموعه‌داده چهار رده دارد، که نمونه داده‌های ردهٔ ۲ به‌عنوان داده نرمال و داده‌های رده ۴ به‌عنوان داده پرت در نظر گرفته می‌شوند. نمونه داده‌های رده‌های دیگر حذف شده‌اند. با در نظر گرفتن این رده‌ها تعداد داده‌های مورد بررسی در مجموعه‌داده ۲۸۶۰۴۸ رکورد می‌شود.

خلاصه‌ای از مشخصات هر سه مجموعه دادهٔ **HTTP**، **SMTP** و **Covertype** در جدول (۱) ارائه شده است.

**(جدول-۱): مشخصات مجموعه‌داده های ارزیابی**
**(Table-1): Characteristics of evaluation datasets**

| عنوان مجموعه‌داده | تعداد رکورد | تعداد ویژگی |
|---|---|---|
| **HTTP** | 623091 | 41 |
| **SMTP** | 96554 | 41 |
| **Covertype** | 581012 | 54 |

## ۲-۴- معیارهای ارزیابی و تنظیمات اولیه

در مسئلهٔ تشخیص داده‌های پرت، چهار حالت ممکن *مثبتِ درست*[1]، *منفیِ درست*[2]، *مثبتِ کاذب*[3] و *منفیِ کاذب*[4] به‌شرح زیر می‌تواند به‌وجود آید:

۱) **مثبت درست:** یک تشخیص درصورتی مثبت درست است که دادهٔ مورد نظر پرت بوده و الگوریتم به درستی آن را پرت و یا حمله تشخیص داده باشد.

۲) **منفی درست:** یک تشخیص درصورتی منفی درست است که دادهٔ مورد نظر دادهٔ نرمال بوده و الگوریتم به‌درستی آن را نرمال تشخیص داده باشد.

۳) **مثبت کاذب:** یک تشخیص درصورتی مثبت کاذب است که دادهٔ مورد نظر دادهٔ نرمال بوده و الگوریتم به اشتباه آن را داده پرت یا حمله تشخیص داده باشد.

۴) **منفی کاذب:** یک تشخیص درصورتی منفی کاذب است که دادهٔ مورد نظر دادهٔ پرت بوده و الگوریتم به اشتباه آن را نرمال تشخیص داده باشد.

به‌طورمعمول الگوریتم‌های تشخیص داده‌های پرت به‌وسیله معیارهای **AUC** (مساحت زیر نمودار **ROC**)، صحت[5] و بازیابی[6] ارزیابی می‌شوند. برای رسم نمودار **ROC** و محاسبهٔ معیار **AUC**، باید معیارهای نرخ مثبت درست[7] و نرخ مثبت کاذب محاسبه شوند. این معیارها و معیارهای دقت[8] و بازیابی با استفاده از چهار حالت تشخیصی که در بالا نام برده شد، محاسبه می‌شوند. در ادامه، نحوه محاسبه این معیارهای ارزیابی ارائه می‌شوند.

[1] True positive (TP)
[2] True negative (TN)
[3] False positive (FP)
[4] False negative (FN)
[5] Precision
[6] Recall
[7] True positive rate
[8] Accuracy




[ DOI: 10.61186/jsdp.20.2.81 ]

معیار نرخ مثبت درست از نسبت مقدار پارامتر مثبت درست به مجموع مقادیر پارامترهای مثبت درست و منفی کاذب محاسبه می‌شود. رابطهٔ (۱) نحوه محاسبه این معیار را نشان می‌دهد:

$$TPR = \frac{TP}{TP + FN} \quad (۱)$$

معیار نرخ مثبت کاذب از نسبت مقدار پارامتر مثبت کاذب به مجموع مقادیر پارامترهای مثبت کاذب و منفی درست حاصل می‌شود. رابطهٔ (۲) معادلهٔ این معیار را نشان می‌دهد:

$$FPR = \frac{FP}{FP + TN} \quad (۲)$$

درنهایت معیارهای دقت، صحت و بازیابی طبق روابط (۳ تا ۵) تعریف می‌شوند:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (۳)$$

$$Precision = \frac{TP}{TP + FP} \quad (۴)$$

$$Recall = \frac{TP}{TP + FN} \quad (۵)$$

## ۳-۴- آزمایش ۱: ارزیابی کارایی روش پیشنهادی و مقایسه با دیگر روش‌ها

در این بخش، کارایی الگوریتم پیشنهادی را ارزیابی و با روش‌های موجود و قابل مقایسه که در این حوزه ارائه شده‌اند مقایسه می‌کنیم. روش نخست قابل مقایسه، روش معرفی‌شده در مرجع [28] است. این الگوریتم که بر مبنای الگوریتم **LOF** [27] کار می‌کند، الگوریتم **DILOF** نامیده می‌شود و جزء دسته روش‌های مبتنی بر نزدیک‌ترین همسایهٔ معرفی‌شده در بخش ۱ است که محاسبات پیچیده‌ای دارد و پس از هر بار ورود و خروج داده‌ها اطلاعات مربوط به هر داده، از جمله نزدیک‌ترین همسایه‌ها، به‌روز می‌شود. الگوریتم بعدی که برای مقایسه در نظر گرفته شده است، الگوریتم **HS-trees** [34] است. این الگوریتم یک مدل یادگیری تک ردهه و یک درخت باینری است که برای تشخیص داده پرت در جریان داده‌ها استفاده شده است. روش سوم، الگوریتم **iForestASD** [33] است، که بر مبنای الگوریتم **iForest** طراحی شده است.

برای همه مجموعه‌داده‌های استفاده شده در آزمایش‌ها، از ۱۰٪ رکوردها برای ساخت مدل اولیه استفاده می‌کنیم و مابقی داده‌ها برای آزمون و ارزیابی استفاده می‌شوند. داده‌های آموزشی به‌صورت تصادفیِ متوازن[1] (به‌گونه‌ای که هر دو رده صفر و یک به‌صورت متوازن برای آموزش موجود باشند) انتخاب می‌شوند. داده‌های منتخب از مجموعه‌داده آزمون حذف می‌شوند. همچنین از اندازه پنجرهٔ یکسان و برابر با ۱۰۲۴ برای همه داده‌ها استفاده می‌کنیم (جزئیات بیشتر درخصوص انتخاب اندازهٔ پنجره در بخش ۵-۴ ارائه شده است). همچنین عمق درنظرگرفته‌شده برای استخراج مدل آموزشی **QLattice** را برابر با ۴ در نظر گرفتیم.

نتایج حاصل از مقایسهٔ رویکرد پیشنهادی با دیگر روش‌ها بر روی سه مجموعه‌داده ، در جدول (۲) ارائه شده است. نتایج مربوط به روش‌های دیگر از مراجع مربوطه به‌طور مستقیم گزارش شده‌اند. برای آن دسته از معیارهای ارزیابی که روش‌های دیگر آن‌ها را گزارش نکرده‌اند، از نماد «-» استفاده شده است.

همان‌طور که در جدول (۲) مشاهده می‌شود، روش پیشنهادی در هر سه مجموعه‌داده ، مقدار **AUC** بهتری نسبت به الگوریتم **DILOF** و **iForestASD** دارد. همچنین روی مجموعه‌داده **SMTP**، روش پیشنهادی در مقایسه با الگوریتم **HS-trees**، که از درخت تصمیم برای یادگیری استفاده می‌کند، بهتر عمل کرده و نتایج روی مجموعه‌داده **HTTP** به‌طورتقریبی مشابه با این الگوریتم،ذولی دقت روش پیشنهادی روی مجموعه‌داده **Covertype** ، به مقدار ناچیزی (۰/۰۱) پایین‌تر از روش **HS-trees** است.

**(جدول-۲): مقایسه کارایی روش پیشنهادی با دیگر روش‌ها بر روی سه مجموعه‌داده ارزیابی**

**(Table-2): Comparison of the performance of the proposed method with other methods on three evaluation datasets**

| مجموعه‌داده HTTP | | | | |
|---|---|---|---|---|
| روش | *Accuracy* | *Precision* | *Recall* | *AUC* |
| **DILOF** | - | - | - | 0.87 |
| **HS-trees** | - | - | - | **0.99** |
| **iForestASD** | - | - | - | 0.94 |
| **پیشنهادی** | 0.99 | 0.83 | 0.98 | **0.99** |
| **مجموعه‌داده SMTP** | | | | |
| **DILOF** | - | - | - | 0.80 |
| **HS-trees** | - | - | - | 0.87 |
| **iForestASD** | - | - | - | 0.86 |
| **پیشنهادی** | 0.99 | 0.81 | 0.99 | **0.99** |
| **مجموعه‌داده Covertype** | | | | |
| **DILOF** | - | - | - | - |
| **HS-trees** | - | - | - | **0.99** |
| **iForestASD** | - | - | - | 0.83 |
| **پیشنهادی** | 0.99 | 0.63 | 0.96 | 0.98 |

[1] Stratified random sampling



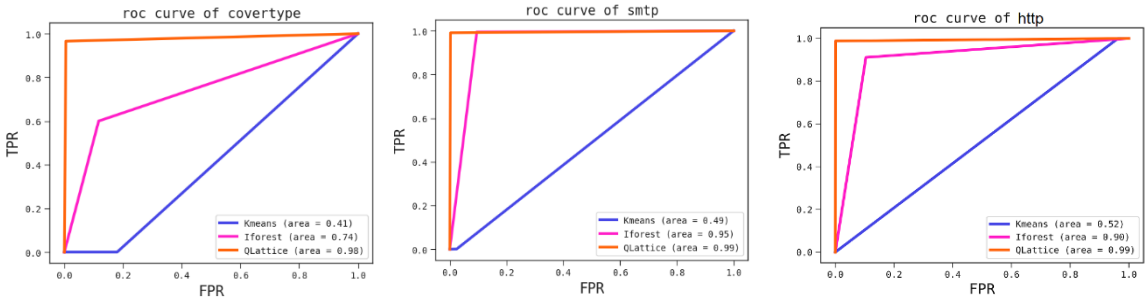


[ DOI: 10.61186/jsdp.20.2.81 ]

## ۴-۴- آزمایش۲: مقایسهٔ کارایی الگوریتم QLattice با سایر الگوریتم‌های یادگیری

به‌طورمعمول فرایند یادگیری در کاربردهایی که با داده‌های حجیم سر و کار دارند و تعداد داده‌ها و ویژگی‌های آن‌ها زیاد است، به‌صورت افزایشی انجام می‌شود. در کتابخانه **Scikit-Learn** [47]، که ما در پیاده‌سازی رویکرد پیشنهادی از آن استفاده کردیم، تابعی به نام **partial_fit()** برای یادگیری برخط در این نوع داده‌ها وجود دارد. این تابع قابلیت یادگیری تدریجی در داده‌ها را برای ما میسر می‌سازد و در هر زمانِ خاص تعداد دادهٔ مشخصی برای یادگیری انتخاب می‌شوند. بدین ترتیب مشکل کمبود حافظه در این موارد برطرف می‌شود. همهٔ الگوریتم‌های یادگیری نمی‌توانند به‌صورت تدریجی یاد بگیرند. در این بخش ما دو الگوریتم **K-means** و **Iforest** را که بیشتر برای کارهای تشخیص داده پرت استفاده شده‌اند، مورد ارزیابی قرار می‌دهیم و در شرایط یکسان با الگوریتم استفاده‌شده در روش پیشنهادی یعنی مدل مبتنی بر **QLattice** مقایسه می‌کنیم.

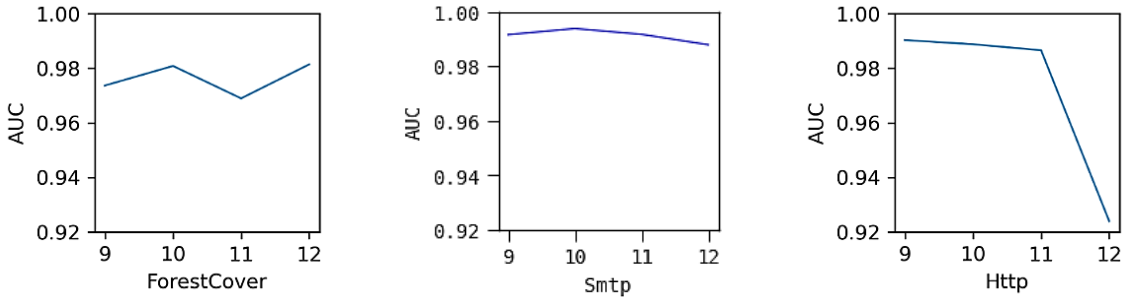
منحنی ROC و معیار AUC به‌دست‌آمده در این آزمایش روی هر سه مجموعه‌داده ، در شکل (۶) نشان داده شده است. همان‌طور که در این شکل مشخص است، الگوریتم QLattice در هر سه مجموعه‌داده نسبت به الگوریتم‌های دیگر بهتر عمل می‌کند. میانگین مقدار AUC الگوریتم QLattice بر روی سه مجموعه‌داده نزدیک ۰/۹۹ است که این عملکرد بهتر QLattice در فرایند یادگیری، نشان از مناسب‌بودن آن برای کاربرد در تحلیل داده‌های جریانی دارد؛ ازاین‌رو، این مدل یادگیری نه‌تنها در کاربرد مورد هدف بلکه در سایر موارد کار با داده‌های جریانی می‌تواند گزینه مناسبی باشد.




(c) (ج) (b) (ب) (a) (الف)

(شکل-۶): مقایسه عملکرد روش پیشنهادی با سایر الگوریتم‌های یادگیری بر روی (الف) HTTP، (ب) SMTP و (ج) Covertype

**(Figure-6): Comparison of the proposed method with other learning algorithms on (a) HTTP, (b) SMTP and (c) Covertype**




(c) (ج) (b) (ب) (a) (الف)

(شکل ۷): مقدار AUC حاصل‌شده به‌وسیلی روش پیشنهادی با استفاده از اندازه‌های مختلف پنجره در مقیاس لگاریتمی (محور افقی) بر روی سه مجموعه‌داده (الف) HTTP، (ب) SMTP، و (ج) Covertype

**(Figure-7): AUC values obtained by the proposed method using different window sizes in logarithmic scale (horizontal axis) on three datasets (a) HTTP, (b) SMTP and (c) Covertype**

## ۴-۵- آزمایش۳: تأثیر اندازهٔ پنجره در عملکرد الگوریتم

در پردازش داده‌های جریانی، انتخاب اندازهٔ پنجرهٔ تحلیل داده‌ها حائز اهمیت است و در کاربردهای مختلف با توجه به سرعت ورود داده‌ها می‌تواند تأثیرگذار باشد. تعیین اندازهٔ پنجره در مصرف حافظه مورد نیاز نیز تأثیرگذار است؛ به‌طوری‌که هر چقدر اندازه پنجره بیشتر باشد، میزان مصرف حافظه و همچنین زمان تحلیل داده‌ها





[ DOI: 10.61186/jsdp.20.2.81 ]

بیشتر می‌شود. در این بخش، به‌منظور بررسی تأثیر اندازه پنجره استفاده‌شده در رویکرد پیشنهادی، نتایج حاصل از اجرای الگوریتم با اندازه‌های مختلف پنجره بر روی مجموعه‌های مختلف داده بررسی می‌شود.

مقدار **AUC** حاصل شده توسط روش پیشنهادی با استفاده از اندازههای مختلف پنجره در مقیاس لگاریتمی بر روی سه مجموعه‌داده مورد ارزیابی در شکل (۷) نشان داده شده است. همانطور که در این شکل قابل مشاهده است، بهترین اندازه پنجره برای هر سه مجموعه‌داده، مقدار ۱۰۲۴ است. از این رو در آزمایش‌های مختلف از این اندازه پنجره استفاده شده است. در این شکل، محور افقی اندازه پنجره را در مقیاس لگاریتمی و محور عمودی دقت الگوریتم را در اندازه‌های مختلف پنجره نشان می‌دهد.

**(جدول-۳): مقایسه کارایی رویکرد پیشنهادی در دو حالت استفاده و عدم استفاده از یادگیری برخط**

**(Table-3): Comparison of the performance of the proposed approach in two modes of use and non-use of online learning**

| روش پیشنهادی | $TP$ | $FP$ | $FN$ | $TN$ | $FNR$ | $FPR$ |
|---|---|---|---|---|---|---|
| **مجموعه‌داده SMTP** | | | | | | |
| **با استفاده از یادگیری برخط** | 1055 | 235 | 10 | 85599 | 0.009 | 0.002 |
| **بدون استفاده از یادگیری برخط** | 1054 | 1226 | 11 | 84608 | 0.010 | 0.014 |
| **مجموعه‌داده HTTP** | | | | | | |
| **با استفاده از یادگیری برخط** | 3318 | 58064 | 323 | 499078 | 0.088 | 0.104 |
| **بدون استفاده از یادگیری برخط** | 2050 | 61070 | 1591 | 496072 | 0.436 | 0.109 |
| **مجموعه‌داده Covertype** | | | | | | |
| **با استفاده از یادگیری برخط** | 2038 | 648 | 435 | 254323 | 0.175 | 0.002 |
| **بدون استفاده از یادگیری برخط** | 2080 | 1151 | 393 | 253820 | 0.158 | 0.004 |

### ۶-۴- آزمایش۴: ارزیابی الگوریتم QLattice با استفاده از یادگیری برخط

در این بخش، عملکرد الگوریتم پیشنهادی را نسبت به حالتی که از یادگیری برخط استفاده نمی‌کند، ارزیابی می‌کنیم. نتایج به دست آمده برای تعداد چهار حالت تشخیصی $TP$، $FP$، $FN$، و $TN$، و همچنین مقادیر $FNR$ و $FPR$ را در هر دو حالت بررسی و مقایسه می‌کنیم. هدف از انجام این آزمایش، نشان دادن کاهش نرخ مثبت کاذب و نرخ منفی کاذب به هنگام بهره‌گیری از یادگیری برخط است. نتایج به‌دست‌آمده در جدول (۳) ارائه شده است. همان‌طور که در این جدول مشخص است، در حالت استفاده از یادگیری برخط، عملکرد الگوریتم به بر روی هر سه مجموعه‌داده بهتر شده است. گفتنی است نتایج و بهبودهای حاصل‌شده بر روی مجموعه‌ای محدود از داده‌ها به‌دست آمده‌اند و بدیهی است که در کاربردهای دنیای واقعی که حجم وسیعی از داده‌ها همواره در حال جریان خواهند بود، این مقدار از بهبود در تشخیص دادهٔ پرت بسیار مؤثر می‌تواند باشد.

## ۶- نتیجه‌گیری و محورهای توسعه و مطالعه بیشتر

در این مقاله، به مسئله تشخیص داده‌های پرت در جریان داده‌ها و اهمیت آنها، که در حوزه‌های مختلف مانند شبکه‌های رایانه‌ای، سامانه‌های هوشمند تشخیصی و نظارتی مورد استفاده قرار می‌گیرند، پرداختیم. برخی روش‌های تشخیص داده پرت در جریان داده‌ها و ماهیت این داده‌ها و چگونگی تجزیه و تحلیل آن‌ها، معرفی شد. همچنین به انواع رویکردهای مختلف برای تشخیص این نوع از داده‌ها در مجموعه‌داده های ایستا و به‌خصوص جریان داده‌ها اشاره کردیم. یک رویکرد جدید نیز برای تشخیص داده‌های پرت در جریان داده‌ها ارائه شد.

رویکرد پیشنهادی، یک روش مبتنی بر رده‌بندی برای تشخیص داده‌های پرت بر اساس یادگیری افزایشی برخط بود. اساس و پایه روش پیشنهادی، استفاده از یک الگوریتم جدید یادگیری به نام مدل یادگیری **QLattice** برای رده‌بندی داده‌ها بود. ازآنجایی‌که این الگوریتم برای رده‌بندی داده‌ها، دقت و سرعت عمل بالایی دارد، برای تحلیل داده‌های جریانی و تشخیص داده پرت از آن بهره گرفتیم. همچنین برای کاهش نرخ تشخیص‌های مثبت کاذب و منفی کاذب استفاده از یادگیری افزایشی برخط را پیشنهاد دادیم. در روش پیشنهادی از پنجرهٔ ثابت برای تحلیل داده‌های جریانی استفاده شد؛ به این صورت که ابتدا مدل اولیهٔ **QLattice**، به‌وسیلهٔ بخشی از داده‌ها که دارای برچسب هستند، ساخته و سپس در مرحله برخط، عمل پیش‌بینی روی داده‌های پنجره انجام و داده‌های پرت شناسایی شده و سپس مدل یادگیری با استفاده از داده‌های جدید به‌روزرسانی می‌شود. آزمایش‌های انجام‌شده بر روی مجموعه‌داده‌های استاندارد دنیای واقعی و نتایج به‌دست‌آمده از آن، نشان داد که رویکرد پیشنهادی عملکرد خوبی در تشخیص داده‌های پرت دارد و همچنین





[ DOI: 10.61186/jsdp.20.2.81 ]

با توجه به این‌که در داده‌های جریانی، احتمال تغییر رفتار داده‌ها با گذشت زمان وجود دارد، استفاده از یادگیری برخط و به‌روزرسانی مدل، باعث می‌شود، دقت الگوریتم در این شرایط کاهش پیدا نکند.

با توجه به این‌که در این پژوهش از یک الگوریتم جدید یادگیری برای تشخیص داده‌های پرت استفاده شده است، توسعه بیشتر بر روی این الگوریتم و رویکرد پیشنهادی می‌تواند در ابعاد مختلفی ادامه داشته باشد. طبق تجربیاتی که در این پژوهش به‌دست آمد، محورهای توسعه و مطالعه بیشتر می‌تواند شامل موارد زیر باشد:

- تمرکز بیشتر بر روی نمونه‌برداری‌هایی که از داده‌های قبلی برای به‌روزرسانی مدل انجام می‌شود؛ به‌طوری‌که دقت عملکرد الگوریتم افزایش یابد.
- سعی در موازی‌سازی دو مرحلهٔ تشخیص و یادگیری برخط، جهت سرعت بخشیدن بیشتر به الگوریتم.
- استفاده ترکیبی از الگوریتم **QLattice** و دیگر الگوریتم‌های مبتنی بر درخت به‌عنوان روش‌های مبتنی بر گروه که به‌صورت موازی برای بهبود عملکرد رده‌بندی و حل مشکل یادگیری در داده‌های نامتوازن به‌کار می‌روند.
- افزایش سرعت الگوریتم پیشنهادی با استفاده از یک معیار اندازه‌گیری برای تعیین مواقعی که نیاز به به‌روزرسانی مدل وجود دارد؛ به‌گونه‌ای که فقط در صورت مشاهدهٔ تغییر در رفتار داده‌های جدید، عمل به‌روزرسانی انجام شود. درغیراین‌صورت عمل به‌روزرسانی انجام نشود.

## ۷- مراجع 7-Refrence


[1] Y. Djenouri, D. Djenouri, and J. C.-W. Lin, "Trajectory Outlier Detection: New Problems and Solutions for Smart Cities," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 2, pp. 1-28, 2021.

[2] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, A. Cano, and J. C.-W. Lin, "A Two-Phase Anomaly Detection Model for Secure Intelligent Transportation Ride-Hailing Trajectories," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[3] M. Hashemzadeh and A. Zademehdi, "Fire detection for video surveillance applications using ICA K-medoids-based color model and efficient spatio-temporal visual features," *Expert Systems with Applications*, vol. 130, pp. 60-78, 2019.

[4] M. Hashemzadeh, G. Pan, and M. Yao, "Counting moving people in crowds using motion statistics of feature-points," *Multimedia tools and applications*, vol. 72, no. 1, pp. 453-487, 2014.

[5] M. Hashemzadeh, G. Pan, Y. Wang, M. Yao, and J. Wu, "Combining velocity and location-specific spatial clues in trajectories for counting crowded moving objects," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 02, p. 1354003, 2013.

[6] M. Hashemzadeh and N. Farajzadeh, "Combining keypoint-based and segment-based features for counting people in crowded scenes," *Information Sciences*, vol. 345, pp. 199-216, 2016.

[7] N. Farajzadeh, A. Karamiani, and M. Hashemzadeh, "A fast and accurate moving object tracker in active camera model," *Multimedia Tools and Applications*, vol. 77, no. 6, pp. 6775-6797, 2018.

[8] S. Sadik and L. Gruenwald, "Research issues in outlier detection for data streams," *Acm Sigkdd Explorations Newsletter*, vol. 15, no. 1, pp. 33-40, 2014.

[9] J. Han, M. Kamber, and J. Pei, "Data mining: concepts and techniques, Waltham, MA," *Morgan Kaufman Publishers*, vol. 10, pp. 978-1, 2012.

[10] D. M. Hawkins, *Identification of outliers*. Springer, 1980.

[11] S. Mehta, "Concept drift in streaming data classification: Algorithms, Platforms and issues," *Procedia computer science*, vol. 122, pp. 804-811, 2017.

[12] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85-126, 2004.

[13] M. Singh and R. Pamula, "ADINOF: adaptive density summarizing incremental natural outlier detection in data stream," *Neural Computing and Applications*, pp. 1-17, 2021.

[14] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250-2267, 2013.

[15] Y. Yang, L. Chen, and C. Fan, "ELOF: fast and memory-efficient anomaly detection algorithm in data streams," *Soft Computing*, pp. 1-12, 2020.

[16] L. Chen, W. Wang, and Y. Yang, "CELOF: Effective and fast memory efficient local outlier detection in high-dimensional data streams," *Applied Soft Computing*, vol. 102, p. 107079, 2021.

[17] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1-30, 2020.

[18] M. V. Joshi, R. C. Agarwal, and V. Kumar, "Mining needle in a haystack: classifying rare classes via two-phase rule induction," in *Proceedings of the 2001 ACM SIGMOD*




[ DOI: 10.61186/jsdp.20.2.81 ]

Estimation-based Local Outlier Detection over Large Data Streams," in *EDBT*, 2019, pp. 421-432.

[32] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008: IEEE, pp. 413-422.

[33] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12-17, 2013.

[34] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[35] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134-147, 2017.

[36] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," *arXiv preprint arXiv:1710.00811*, 2017.

[37] B. V. Ashok, "QLattice Environment and Feyn QGraph Models - A new Perspective towards Deep Learning," Zenodo, 2020.

[38] M. Machado. "A new kind of AI." https://medium.com/abzuai/a-new-kind-of-ai-7665f8198877 (accessed.

[39] K. B. T. Jelen. https://docs.abzu.ai/docs/guides/qlattice.html (accessed.

[40] C. Cave. "Opening the black box." https://medium.com/abzuai/opening-the-black-box-247a63ce553e (accessed.

[41] J. Brownlee, "Why one-hot encode data in machine learning," *Machine Learning Mastery*, 2017.

[42] M. DelSole. "What is One Hot Encoding and How to Do It." https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179 (accessed.

[43] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of artificial intelligence research*, vol. 61, pp. 863-905, 2018.

[44] P. Soltanzadeh and M. Hashemzadeh, "RCSMOTE: Range-Controlled synthetic minority over-sampling technique for handling the class imbalance problem," *Information Sciences*, vol. 542, pp. 92-111, 2021.

[45] "Overview of Online Machine Learning in Big Data Streams," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Y. Zomaya Eds. Cham: Springer International Publishing, 2019, pp. 1239-1239.

[46] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD *international conference on Management of data*, 2001, pp. 91-102.

[19] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *International Conference on Data Warehousing and Knowledge Discovery*, 2002: Springer, pp. 170-180.

[20] M. U. Togbe *et al.*, "Anomaly Detection for Data Streams Based on Isolation Forest Using Scikit-Multiflow," in *International Conference on Computational Science and Its Applications*, 2020: Springer, pp. 15-30.

[21] G. Han, J. Tu, L. Liu, M. Martínez-García, and Y. Peng, "Anomaly Detection Based on Multidimensional Data Processing for Protecting Vital Devices in 6G-Enabled Massive IIoT," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5219-5229, 2021.

[22] N. M. R. SURI and G. Athithan, *Outlier detection: techniques and applications*. Springer, 2019.

[23] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 275-300, 2004.

[24] C. C. Aggarwal, S. Y. Philip, J. Han, and J. Wang, "A framework for clustering evolving data streams," in *Proceedings 2003 VLDB conference*, 2003: Elsevier, pp. 81-92.

[25] I. Assent, P. Kranen, C. Baldauf, and T. Seidl, "Anyout: Anytime outlier detection on streaming data," in *International Conference on Database Systems for Advanced Applications*, 2012: Springer, pp. 228-242.

[26] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 811-820.

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *ACM sigmod record*, 2000, vol. 29, no. 2: ACM, pp. 93-104.

[28] G. S. Na, D. Kim, and H. Yu, "DILOF: Effective and memory efficient local outlier detection in data streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1993-2002.

[29] M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan, and X. Zhang, "Fast memory efficient local outlier detection in data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3246-3260, 2016.

[30] J. Gao, W. Ji, L. Zhang, A. Li, Y. Wang, and Z. Zhang, "Cube-based incremental outlier detection for streaming computing," *Information Sciences*, vol. 517, pp. 361-376, 2020.

[31] X. Qin, L. Cao, E. A. Rundensteiner, and S. Madden, "Scalable Kernel Density



[ DOI: 10.61186/jsdp.20.2.81 ]

CUP 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009: IEEE, pp. 1-6.
[47] "Strategies to scale computationally: bigger data." https://scikit-learn.org/0.15/modules/scaling_strategies.html (accessed 2018).

**سحر فردین** مدرک کارشناسی و کارشناسی ارشد خود را در رشته مهندسی فناوری اطلاعات گرایش مدیریت سامانه‌های اطلاعاتی بهترتیب از دانشگاه تبریز و دانشگاه شهید مدنی آذربایجان دریافت کرده‌اند . ایشان از سال۱۳۹۷ تاکنون عضو آزمایشگاه هوش مصنوعی و یادگیری ماشین دانشکدهٔ فناوری اطلاعات و مهندسی کامپیوتر دانشگاه شهید مدنی آذربایجان هستند. زمینه‌های پژوهشی نام‌برده عبارتند از: داده‌کاوی، بازشناخت الگو و یادگیری ماشین. نشانی رایانامهٔ ایشان عبارت است از:

s.fardin@azaruniv.ac.ir

**مهدی هاشم‌زاده** متولد ۱۳۵۸ در شهر تبریز است. ایشان درجهٔ کارشناسی و کارشناسی ارشد مهندسی کامپیوتر گرایش نرم‌افزار را به‌ترتیب در سال‌های ۱۳۸۰ و ۱۳۸۵ دریافت کرد و در سال ۱۳۹۲ دکترای تخصصی علوم کامپیوتر گرایش هوش مصنوعی و بینایی ماشین را از دانشگاه Zhejiang کشور چین دریافت کردند. ایشان از مهرماه سال ۱۳۹۲ تاکنون عضو هیئت علمی دانشکدهٔ فناوری اطلاعات و مهندسی کامپیوتر دانشگاه شهید مدنی آذربایجان است که در حال حاضر دارای مرتبه علمی دانشیاری هستند. همچنین ایشان مؤسس و مدیر آزمایشگاه هوش مصنوعی و یادگیری ماشین این دانشکده هستند. زمینه‌های پژوهشی نام‌برده عبارتند از: نظارت ویدئویی هوشمند، داده‌کاوی، بینایی ماشین، بازشناخت الگو، و یادگیری ماشین. نشانی رایانامهٔ ایشان عبارت است از:

hashemzadeh@azaruniv.ac.ir



[ DOI: 10.61186/jsdp.20.2.81 ]