JIPET

Journal of Intelligent Procedures in Electrical Technology
Vol. 16/ No. 64/ Winter 2026
P-ISSN: 2322-3871, E-ISSN: 2345-5594, http://jipet.iaun.ac.ir/

https://dorl.net/dor/20.1001.....
Research Article

# Providing a Method Based on Data Mining and Semantic Web Techniques to Prioritize Software Requirements

**Sahar Ghazizadeh[1], *Ms.C.*, Seyed Ebrahim Dashti[2], *Assistant Professor***

[1]Department of Electrical Engineering- Larestan Branch, Islamic Azad University, Larestan, Iran
[2]Department of Computer Engineering- Jahrom Branch, Islamic Azad University, Jahrom, Iran
ghazizadeh_s@yahoo.com, seyedebrahim.dashti@iau.ac.ir

**Abstract**
Requirements engineering is one of the important and main phases in the software development process. In this phase, various activities are performed by software engineers. In this paper, the problem of prioritizing requirements in the iterative software development process has been investigated. The main goal is to automatically find priority groups of requirements so that each group of requirements can be implemented in one iteration of the development process. For this purpose, data mining, text processing and semantic similarity techniques have been used. The proposed methods have been applied to the data set related to the requirements of different software. The proposed methods have different phases. In the clustering phase, various clustering algorithms such as K-Means, hierarchical clustering, and Density-based spatial clustering of applications with noise (DBSCAN) have been used. In order to check the effectiveness of clustering methods, the results obtained for the prioritization of requirements have been compared with the priority groups determined by experts. The results of the comparison show that the presented methods have good and acceptable performance for classifying requirements and clustering methods. Proposed method provides an average of 3% better performance.

**Keywords**: clustering algorithms, requirements prioritization, semantic similarity, text processing

**Received:** 11 April 2023
**Revised:** 19 July 2023
**Accepted:** 11 August 2023

**Corresponding Author:** Dr. Seyed Ebrahim Dashti

Citation: S. Ghazizadeh, S.E. Dashti, "Providing a method based on data mining and semantic web techniques to prioritize software requirements", Journal of Intelligent Procedures in Electrical Technology, vol. 16, no. 64, pp. 57-70, March 2026 (in Persian).

https://dorl.net/dor/20.1001.....
مقاله پژوهشی

# ارائه روشی مبتنی بر تکنیک‌های داده‌کاوی و وب‌معنایی جهت اولویت‌بندی نیازمندی‌های نرم افزار

**سحر قاضی‌زاده[۱]، دانشجوی کارشناسی‌ارشد، سید ابراهیم دشتی[۲]، استادیار**

۱- دانشکده فنی مهندسی- واحد لارستان، دانشگاه آزاد اسلامی، لارستان، ایران
۲- دانشکده برق و کامپیوتر- واحد جهرم، دانشگاه آزاد اسلامی، جهرم، ایران
ghazizadeh_s@yahoo.com, seyedebrahim.dashti@iau.ac.ir

**چکیده:** مهندسی نیازمندی‌ها یکی از فازهای مهم و اصلی در فرآیند توسعه نرم‌افزار است. در این فاز فعالیت‌های مختلفی توسط مهندسین نرم‌افزار انجام می‌شود. در این مقاله مسئله اولویت‌بندی نیازمندی‌ها در فرآیند توسعه تکراری نرم‌افزار مورد بررسی قرار گرفته است. هدف اصلی پیداکردن گروه‌های اولویت نیازمندی‌ها به‌صورت خودکار بوده، به‌گونه‌ای که بتوان هر گروه از نیازمندی‌ها را در یک تکرار از فرآیند توسعه پیاده‌سازی کرد. برای این منظور از تکنیک‌های داده‌کاوی، پردازش متن و شباهت معنایی استفاده شده است. روش پیشنهادی بر روی مجموعه داده‌های مربوط به نیازمندی‌های نرم‌افزارهای مختلف اعمال شده است. این روش دارای فازهای پیش-پردازش، استخراج ویژگی، خوشه‌بندی و اولویت‌بندی است. در فاز اول آماده‌سازی داده‌ها انجام می‌شود و در فاز دوم ویژگی‌های نیازمندی‌ها استخراج می‌شود. در فاز خوشه‌بندی از الگوریتم‌های مختلف خوشه‌بندی مانند میانگین k، سلسله‌مراتبی و خوشه‌بندی فضایی مبتنی بر چگالی در کاربردهای دارای نویز (DBSCAN) استفاده شده و در نهایت اولویت‌بندی انجام می‌شود. نتایج مقایسه نشان می‌دهد روش ارائه شده دارای کارایی خوب و قابل قبولی نسبت به روش‌های قبل در دسته‌بندی و خوشه‌بندی بوده و به‌طور میانگین دارای ۳ درصد عملکرد بهتری است.

**کلمات کلیدی:** اولویت‌بندی نیازمندی‌ها، پردازش متن، شباهت معنایی، الگوریتم‌های خوشه‌بندی

**تاریخ ارسال مقاله:** ۱۴۰۲/۱/۲۲
**تاریخ بازنگری مقاله:** ۱۴۰۲/۴/۲۸
**تاریخ پذیرش مقاله:** ۱۴۰۲/۵/۲۰

**نام نویسنده‌ی مسئول:** دکتر سیدابراهیم دشتی
**نشانی نویسنده‌ی مسئول:** جهرم- دانشگاه آزاد اسلامی واحد جهرم- دانشکده برق و کامپیوتر

## ۱- مقدمه

فرآیند توسعه نرم‌افزار دارای فازهای مختلفی مانند امکان سنجی، استخراج نیازمندی‌ها، تحلیل، طراحی، پیاده‌سازی، تست، تحویل و نگهداری سیستم است. فاز نیازمندی‌ها یکی از کلیدی‌ترین فازها در توسعه نرم‌افزار بوده و شناخت دقیق نیازمندی‌ها، استخراج آن‌ها و ردیابی آن‌ها در فرآیند تولید نرم‌افزار مهم است. عموماً بیشترین مشکلاتی که در توسعه نرم‌افزارها وجود دارد منشاء آن به فاز نیازمندی‌ها برمی‌گردد. نیازمندی‌های نرم‌افزار ابتدا به‌صورت زبان طبیعی بیان شده و مستندسازی آن‌ها نیز به زبان طبیعی است. زبان طبیعی برای بیان نیازمندی‌ها ممکن است دقیق نبوده و یا نیازمندی‌های بیان شده مبهم یا برخی از نیازمندی‌ها با یکدیگر تضاد داشته باشند. از طرفی نیازمندی‌ها ممکن است فراموش شده یا یک نیازمندی شناسایی شده باشد اما در محصول نهایی این نیازمندی پیاده‌سازی نشده باشد.

موارد ذکر شده نشان دهنده طیف گسترده مشکلاتی است که در مدیریت نیازمندی‌ها وجود دارد. برای حل این مشکلات در سال‌های اخیر روش‌های مختلفی توسط محققین ارائه شده که هر یک از روش‌های ارائه شده صرفا به بخش خاصی از مشکلات مربوط به مهندسی نیازمندی‌ها پرداخته‌اند. در این مقاله روشی برای اولویت‌بندی نیازمندی‌ها ارائه شده است. برای این منظور از روش‌های داده‌کاوی و همچنین وب‌معنایی استفاده شده و با استفاده از تکنیک‌های داده‌کاوی نیازمندی‌های نرم‌افزار بر اساس درجه اهمیت در خوشه‌های مختلف قرار داده می‌شوند. این خوشه‌ها رتبه‌های کلی نیازمندی‌ها را مشخص و سپس در داخل هر خوشه از روش‌های رتبه‌بندی کلمات استفاده شده است. چندین روش خوشه‌بندی مانند میانگین k[1] استفاده شده است. الگوریتم-های خوشه‌بندی[2] نیازمند استفاده از ویژگی‌ها هستند. با استفاده از تکنیک‌های تبدیل کلمه به بردار برخی از ویژگی‌های مورد نیاز از متن نیازمندی‌ها استخراج شده و در اختیار الگوریتم قرار داده می‌شوند. در نهایت بر اساس خوشه‌های به‌دست آمده اولویت‌بندی نیازمندی‌ها انجام می‌شود. نوآوری‌های مقاله حاضر به شرح زیر است:

۱) ارائه یک روش چند مرحله‌ای جهت اولویت‌بندی موثر نیازمندی‌های نرم‌افزار به‌گونه‌ای که در یک فرآیند توسعه تکراری یا چابک قابل استفاده باشند.

۲) استفاده از چندین روش خوشه‌بندی در مرحله گروه‌بندی نیازمندی‌ها و شناسایی بهترین الگوریتم خوشه‌بندی.

۳) استفاده از شباهت‌های معنایی کلمات جهت کمک به کارایی الگوریتم‌های خوشه‌بندی.

۴) استفاده از یک مجموعه داده تقریبا جدید در حوزه نیازمندی‌های نرم‌افزار که طیف قابل توجهی از نرم‌افزار را پوشش می‌دهد.

۵) استفاده از دانش افراد خبره جهت ارزیابی روش‌های ارائه شده.

در ادامه ساختار مقاله به این شرح است. در بخش دوم پیشینه تحقیق ارائه شده و در بخش سوم روش پیشنهادی جهت اولویت‌بندی نیازمندی‌ها بیان شده است. در بخش چهارم نیز نتایج به‌دست آمده از اعمال روش پیشنهادی برای اولویت‌بندی مجموعه‌ای از نیازمندی‌های نرم‌افزار ارائه و در نهایت بخش پنجم به نتیجه‌گیری و ارائه پیشنهادات اختصاص داده شده است.

## ۲- پیشینه تحقیق

در سال‌های اخیر روش‌های متعددی برای اولویت‌بندی نیازمندی‌ها ارائه شده است. در مرجع [۱] مسئله اولویت‌بندی نیازمندی‌ها مدل شده و مجموعه‌ای از تحقیقات انجام شده در سال‌های اخیر در این زمینه، شناسایی و تحلیل شده‌اند. نتایج این تحقیق نشان می‌دهد که اولویت‌بندی نیازمندی‌ها به‌طور گسترده و جدی در زمینه مهندسی نرم‌افزار مورد بحث و تحقیق قرار گرفته است. ارزیابی نویسندگان این مقاله نشان می‌دهد که مجموعه روش‌های موجود برای اولویت‌بندی نیازمندی‌ها دارای نقاط ضعف مختلفی مانند ضعف در مقیاس‌پذیری، عدم پشتیبانی از به‌روزرسانی اولویت‌ها، عدم هماهنگی بین ذینفعان، وابستگی بین نیازمندی‌ها و همچنین عدم امکان به‌کارگیری تکنیک‌های موجود در محیط‌های پیچیده و واقعی است.

در تحقیق انجام شده در مرجع [۲]، روش جدیدی برای اولویت‌بندی نیازمندی‌ها با استفاده از ارتباطات بین نیازمندی‌های ذینفعان و مشخصات استخراج شده از آن‌ها به شکل موارد کاربرد و نیازمندی‌های غیرعملکردی ارائه شده است. در مرجع [۳] مروری بر روش‌های اولویت‌بندی نیازمندی‌ها در نرم‌افزارهای موجود در بازار که توسط کاربران زیادی ممکن است استفاده شود ارائه شده که در این پژوهش جهت جمع‌آوری نتایج از پرسش‌نامه اینترنتی استفاده شده است. در تحقیق دیگری [۴]، روشی

جهت اولویت‌بندی نیازمندی‌ها بر اساس فرآیند شبکه تحلیل ارائه شده که در این مقاله مدلی ارائه شده که بر اساس آن نیازمندی‌های مستقل و وابسته قابل اولویت‌بندی هستند. نتایج این پژوهش نشان می‌دهد که روش پیشنهادی می‌تواند به‌طور موفق نیازمندی‌ها را اولویت‌بندی کند. در پژوهش [۵] تکنیک‌های اولویت‌بندی نیازمندی‌ها مانند رتبه‌بندی ساده، تکنیک انتساب عددی، رای گیری تجمعی، فرآیند سلسله مراتبی تحلیلی[۳] (AHP)، اولویت‌بندی ارزش‌گرا[۴] (VOP)، درخت جستجوی دودویی[۵] (BST) و مرتب‌سازی حبابی[۶] مورد بررسی قرار گرفته و بر اساس این بررسی، شش فاکتور برای اولویت‌بندی نیازمندی‌ها انتخاب شده است. در تحقیق دیگری چارچوب هزینه‌یابی مبتنی بر فعالیت[۷] (ABC) با برخی از روش‌های ارائه شده در مقاله‌ها مانند روش تحلیل سلسله مراتبی[۸] (AHP)، روش ارزش-هزینه، روش ویگر[۹] و گروه‌های اولویت مقایسه شده است. همچنین معیارهایی مانند مقیاس‌پذیری، نزدیکی به توسعه کاربردی، مصونیت در مقابل مدیریت تغییرات و امکان‌پذیری توصیف بصری برای مقایسه استفاده شده است. در مرجع [۶] روش جدیدی برای اولویت‌بندی نیازمندی‌های خانه‌های هوشمند ارائه شده است. مسأله اولویت‌بندی نیازمندی‌های جدید در مرجع [۷] مورد بررسی قرار گرفته و در این مقاله یک روش جدید برای اولویت بندی نیازمندی‌ها ارائه شده که شامل تجزیه و تحلیل تمام فاکتورها و نتایج ناشی از پیاده‌سازی نیازمندی‌های انتخاب شده در محصول نرم‌افزاری است. روش ارائه شده شامل معیارهای مالی، قوانین کسب و کار شرکت‌ها و معیارهای کیفیت تعریف شده توسط مدیران است. در پژوهش دیگری [۸] به بررسی توابع برازش برای اولویت بندی نیازها بر اساس جستجو پرداخته شده است. در این پژوهش بیان شده که انتخاب افراد مورد نیاز در بین چندین گزینه توسعه سیستم نرم‌افزاری می‌تواند چالش برانگیز باشد و اولویت‌بندی نیازمندی‌ها، ترتیب اجرای نیازمندی‌ها را بر اساس اولویت یا اهمیت آنها در مورد دیدگاه‌های سهام‌داران تعریف می‌کند که یک کار پیچیده است. بر اساس این مشکل، هدف این مطالعه ارائه نیازمندی‌ها با استفاده از یک الگوریتم ژنتیک برای یافتن راه‌حل‌های بهینه بوده که می‌تواند در اولویت‌بندی نیازمندی‌ها در طول فرایند توسعه نرم افزار کمک کند.

در مرجع [۹] چارچوبی را برای فرایند اولویت‌بندی نیازمندی‌ها در توسعه نرم‌افزار چابک ارائه شده است. در مرجع [۱۰] اولویت-بندی نیازمندی‌های نرم‌افزار راه‌حل‌گرا را با استفاده از تکنیک اولویت‌بندی دیدگاه چندگانه بررسی شده است. در این تحقیق یک الگوریتم برای افزایش فعالیت اولویت‌بندی نیازمندی‌های نرم‌افزار به نام تکنیک اولویت‌بندی دیدگاه چندگانه[۱۰] ارائه شده است. این تکنیک پیشنهادی سعی می‌کند سه دیدگاه را نشان دهد؛ دیدگاه مشتری، دیدگاه‌های تجاری و فنی. همچنین این تکنیک برای تعداد متوسط تا زیاد نیازمندی‌ها طراحی شده و اثربخشی و کارایی روش اولویت‌بندی دیدگاه چندگانه پیشنهادی به‌صورت تجربی مورد بررسی قرار گرفته تا نشان داده شود که ایا ارزش استفاده از آن در محیط کار واقعی را دارد یا خیر.

در مرجع [۱۱] یک روش مبتنی بر AHP[۱۱] را برای اولویت‌بندی نیازمندی‌ها برای یک سیستم کامپیوتری تشخیص چهره ارائه شده که در این پژوهش از تکنیک سلسله مراتب تحلیلی نوتروسفیک[۱۲] برای تجزیه و تحلیل اولویت‌بندی نیازمندی‌های لازم برای پیاده‌سازی سیستم تشخیص چهره در تعاونی استفاده شده است. در پژوهش دیگری [۱۲]، اولویت‌بندی نیازمندی‌ها و استفاده از مدل تکراری برای اجرای موفقیت آمیز نیازمندی‌ها مورد مطالعه قرار گرفته که در این مطالعه، اولویت‌بندی نیازمندی‌ها به‌عنوان درجه‌بندی نیازهای نرم‌افزاری به‌طور خاص مورد بررسی قرار گرفته است. اولویت‌بندی نیازمندی‌ها برای مدیریت و پیاده‌سازی آسان بوده، در حالی که نیازمندی‌های بدون اولویت پر هزینه هستند و زمان زیادی را صرف می‌کنند تا زمان تخمین کل پروژه بتواند فراتر رود. از آنجا که همه نیازمندی‌ها به یکدیگر وابسته هستند، بنابراین کل زمان تخمین بیش از زمانی است که نیازمندی‌ها منتظر نیازمندهای پیش‌نیاز هستند. اولویت نیازمندی‌ها نیز زمانی افزایش می‌یابد که سایر نیازمندی‌ها منتظر بمانند اما تعیین اولویت پایین برای نیازمندی‌های مورد نیاز پروژه را به تاخیر می‌اندازد.

در مرجع [۱۳] به بررسی ادبیات سیستماتیک در مورد اولویت‌بندی نیازمندی‌ها و ارزیابی تجربی آنها اشاره و در این مطالعه بیان شده که بسیاری از روش‌های اولویت‌بندی نیازمندی‌های پیشنهاد شده در محیط‌های واقعی مورد بررسی قرار نگرفته‌اند.

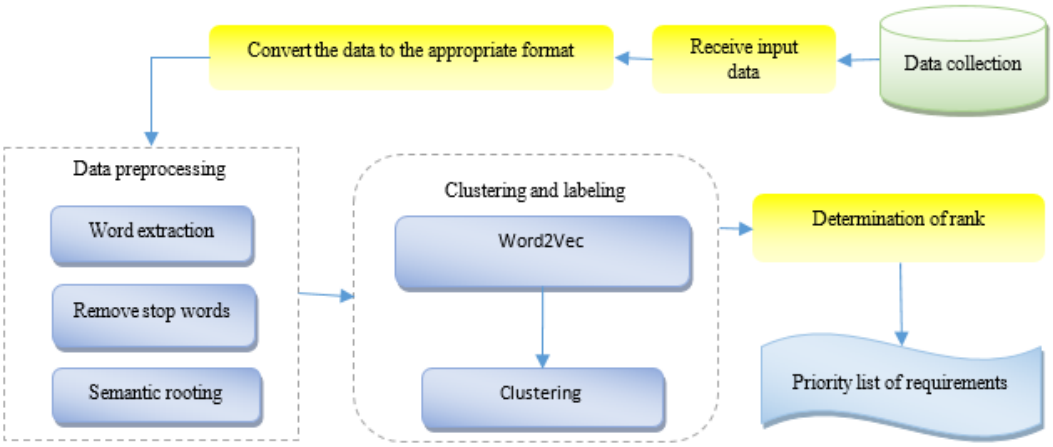
در مطالعه‌های دیگری اولویت‌بندی نیازمندی‌ها برای بهبود کیفیت سیستم‌هایی با قابل پیکربندی بالا مورد بررسی قرار گرفته است [۱۴،۱۵]. در یک کار دیگر تحقیقاتی به مساله جمع‌آوری اطلاعات جهت روش‌های اولویت‌بندی نیازمندی‌ها در فرآیندهای توسعه نرم‌افزار چابک در مرجع [۱۶] پرداخته شده که در آن بیان شده در روش‌های توسعه نرم‌فزار چابک مانند روش‌های توسعه کلاسیک روش سرراستی برای اولویت‌بندی نیازمندی‌ها وجود ندارد زیرا در این روش‌ها همه نیازمندی‌ها در ابتدای فرآیند توسعه

نرم‌افزار در اختیار تیم توسعه قرار ندارد و نیازمندی‌ها ممکن است در زمان‌های مختلف توسعه به‌دست تیم توسعه برسد. به این منظور در این مقاله به استخراج اطلاعت جهت روش‌های اولویت‌بندی نیازمندی‌ها در روش‌های توسعه چابک پرداخته شده و نقاط ضعف و قوت آن‌ها مطرح شده است. این تحقیق به توسعه‌دهندگان کمک می‌کند تا بر اساس شرایط موجود، بهترین روش را برای اولویت‌بندی نیازمندی‌ها انتخاب کنند. در مرجع [۱۷] به مساله اولویت‌بندی نیازمندی‌های نرم‌افزار بر اساس روش‌های بهینه‌سازی منطق فازی بهبود یافته پرداخته شده است. در این روش ابتدا از الگوریتم بهینه‌سازی جهت انتخاب ویژگی‌ها استفاده شده و سپس از منطق فازی بهبود یافته برای اولویت‌بندی نیازمندی‌ها استفاده شده است. در این پژوهش هدف از اولویت‌بندی نیازمندی‌ها این است که بهینه‌سازی لازم جهت پارامترهای مدیریت پروژه به‌دست آید.

در مرجع [۱۸] مساله اولویت‌بندی نیازمندی‌های در فرآیند توسعه مبتنی بر مولفه مورد مطالعه قرار گرفته است. در روش‌های توسعه مبتنی بر مولفه، تاکید بر استفاده مجدد از مولفه‌هایی که دارای ویژگی‌های یکسان هستند، است. به عبارت دیگر قابلیت استفاده مجدد اصل مهم در فرآیند توسعه نرم‌افزار مبتنی بر مولفه است. همچنین در این تحقیق از روش‌های متن کاوی و استنتاج مبتنی بر نمونه استفاده شده است. رویکرد استفاده شده در نظر گرفتن ویژگی‌ها و شرایطی است که در فرآیند توسعه مبتنی بر مولفه وجود دارد. نویسندگان این مقاله بر اساس این روش‌ها چارچوبی را ارائه داده‌اند و روش پیشنهادی آنها با برخی روش‌های دیگر مقایسه شده است. نتایج این آزمایش‌ها نشان می‌دهد روش پیشنهادی می‌تواند جواب‌های قابل رقابت تولید کند. در کار تحقیقاتی دیگر از الگوریتم بهینه‌سازی کولونی زنبورهای مصنوعی جهت اولویت‌بندی نیازمندی‌های نرم‌افزار استفاده شده است [۱۹]. در این تحقیق به اهمیت اولویت‌بندی نیازمندی‌های نرم‌افزار به منظور توسعه یک نرم‌افزار با کیفیت اشاره شده است. همچنین بیان شده زمانی‌که محدودیت‌های زمان و هزینه وجود دارد مساله اولویت‌بندی نیازمندی‌ها اهمیت بیشتری خواهد داشت و همچنین با استفاده از آن می‌توان رضایتمندی بالاتر کاربران را به‌دست آورد. به همین منظور از یک روش بهینه‌سازی برای اولویت‌بندی نیازمندی‌ها استفاده شده است. این روش نیازمندی‌هایی که دارای اولویت بالاتری هستند را شناسایی می‌کند. در مرجع [۲۰] از روش الویت‌بندی سلسله مراتبی تحلیلی برای اولویت‌بندی نیازمندی‌های نرم افزار استفاده کرده‌اند. آنها از محدودیت‌هایی مانند زمان، بودجه و تخصیص منابع که در فرآیند اولویت‌بندی تاثیرگذار هستند نام برده‌اند. در کار تحقیقاتی دیگری [۲۱] اولویت‌بندی گروهی چند معیاره نیازمندی‌های نرم‌افزار در فرآیند توسعه نرم‌افزار مورد توجه قرار گرفته و از برچسب‌های زبانی فازی استفاده شده است. در مطالعه‌ای مساله اولویت‌بندی نیازمندی‌ها مورد توجه قرار گرفته و در مورد نقاط قوت و ضعف روش‌های اولویت‌بندی نیازمندی‌های موجود بحث شده است [۲۲]. در این مطالعه به توسعه‌دهندگان نرم‌افزار کمک می‌کند تا با استفاده از ویژگی‌های موجود در توسعه نرم‌افزار روش مناسب اولویت‌بندی نیازمندی‌ها را انتخاب کنند. همچنین پیشنهادات ارائه شده که کمک خواهد کرد تا روش‌های اولویت‌بندی موجود بهبود یابد. همچنین در این مقاله کارایی روش‌های اولویت‌بندی نرم‌افزار در توسعه سیستم‌های نرم‌افزاری در محیط‌های توسعه چابک مورد توجه قرار گرفته و چالش‌هایی که در مسیر استفاده از روش‌های اولویت‌بندی نرم‌افزار وجود دارد را بیان می‌کند.

## ۳- روش پیشنهادی

در این بخش جزئیات روش پیشنهادی ارائه و تصویری کلی از روش پیشنهادی در شکل (۱) ارائه شده است. روش پیشنهادی دارای فازهای زیر است؛ ۱) دریافت داده ورودی، ۲) تبدیل داده ورودی به فرمت مناسب، ۳) پیش‌پردازش شامل استخراج کلمات، حذف کلمات توقف و ریشه‌یابی کلمات، ۴) خوشه‌بندی با استفاده از روش‌های مختلف، ۵) تعیین رتبه خوشه‌ها و نیازمندی‌ها و ۶) استخراج لیست اولویت نیازمندی‌ها. در فاز اول نیازمندی‌ها در قالب مناسب سازماندهی و پیش پردازش لازم روی آنها انجام شده تا به فرمت مناسب تبدیل شوند. سپس روش پیشنهادی بر روی آن‌ها اعمال و اولویت‌بندی نیازمندی‌ها انجام می‌شود. در مرحله بعد، ویژگی‌های لازم استخراج شده و سپس از تکنیک‌های خوشه‌بندی برای تعیین گروه هر یک از نیازمندی‌ها استفاده خواهد شد و اولویت‌بندی بر اساس خوشه‌های به‌دست آمده انجام می‌شود.




**شکل (۱): ساختار کلی روش پیشنهادی برای اولویت‌بندی نیازمندی‌ها**

Figure (1): The general structure of the proposed method for prioritizing requirements

### ۳-۱- دریافت ورودی و تبدیل داده

دیتاست مورد استفاده دربرگیرنده اطلاعات کامل فاز مهندسی نیازمندی‌های نرم‌افزار است. در این مطالعه از دیتاست مجموعه داده‌های نیازمندیهای عمومی[۱۳] (PURE) استفاده شده است. دیتاست حاوی اطلاعات مربوط به سند مشخصات نیازمندی‌های نرم‌افزارهای مختلف است. این دیتاست اطلاعات مربوط به نرم‌افزارهایی با ویژگی‌های مختلف را جمع‌آوری نموده و بخش‌هایی از این اطلاعات که مورد نیاز هستند شناسایی و به‌صورت دستی استخراج می‌شوند. اطلاعات مورد استفاده در این پژوهش صرفاً نیازمندی‌های عملکردی هستند. این اطلاعات پس از استخراج در قالب یک فایل متنی آماده شده‌اند تا عملیات پیش‌پردازش بر روی آن‌ها انجام شود. نمونه‌ای از فایل ورودی در شکل (۲) نشان داده شده است. همان‌گونه که مشاهده می‌شود مجموعه داده‌ها دربرگیرنده نیازمندی‌های نرم‌افزار بوده که هر کدام از آن‌ها به زبان طبیعی و در قالب یک جمله یا پاراگراف ارائه شده‌اند.

### ۳-۲- پیش پردازش

نیازمندی‌هایی که به‌عنوان ورودی فاز پیش‌پردازش استفاده شده‌اند به‌صورت جملات یا پاراگرف‌های انگلیسی هستند. بنابراین جهت استفاده باید پیش‌پردازش شوند. یکی از اولین کارهایی که در پردازش متن انجام می‌شود شناسایی توکن‌های موجود در یک جمله است. در این مرحله، کلمات موجود در جمله شناسایی شده و آن‌ها از یکدیگر جدا می‌شوند. برای مثال به نیازمندی شکل (۳) توجه کنید. در این مرحله هر نیازمندی به مجموعه‌ای از کلمات تبدیل می‌شود. خروجی حاصل از مرحله شناسایی کلمات برای مثال بالا در شکل (۴) نشان داده شده است.

```
Req1: ETCS shall provide the driver with information to allow him to drive the train safely.
Req2: ETCS shall be able to supervise train and shunting movements.
Req3: If the supervision is performed by a RBC it shall be possible to prevent movements of a traction unit in its area if not authorised by the RBC.
Req4: ETCS is required to be functional to a maximum train speed of 500 km/h.
```

**شکل (۲): نمونه‌ای از فایل ورودی به برنامه جهت پیش-پردازش**

Figure (2): An example of the input file to the pre-processing program

```
ETCS shall provide the driver with information to allow him to drive the train safely.
```

**شکل (۳): نمونه‌ای از نیازمندی‌های استخراج شده برای یک نرم‌افزار**

Figure (3): An example of extracted requirements for a software

```
['ETCS', 'shall', 'provide', 'the', 'driver', 'with', 'information', 'to', 'allow', 'him', 'to', 'drive', 'the', 'train', 'safely']
```

**شکل (۴): کلمات استخراج شده از متن نیازمندی**

Figure (4): Words extracted from the requirement text

پس از تفکیک کلمات، لازم است کلمات توقف شناسایی و حذف شوند. دلیل حذف کلمات توقف این است که این‌گونه کلمات تاثیری در معنای کلی جمله ندارند. برای حذف کلمات توقف در این پژوهش از کتابخانه جعبه ابزار زبان طبیعی[۱۴] (NLTK) استفاده شده است. مجموعه کلمات توقف در زبان انگلیسی که در این کتابخانه تعریف شده در شکل (۵) نشان داده شده است.
با اعمال این کتابخانه روی کلمات به‌دست آمده از روی نیازمندی‌های نرم‌افزار، لیستی از کلمات موثر به‌دست می‌آید. در شکل (۶) نتیجه حاصل از حذف کلمات توقف نشان داده شده است. پس از این مرحله ریشه کلمات به‌دست می‌آید. برای این منظور از کتابخانه وب‌معنایی[۱۵] ورد-نت[۱۶] استفاده می‌شود که یک پایگاه داده از واژگان است که رابطه‌های معنایی بین کلمات را فراهم می‌کند و کلمات را با تعیین مترادف و هجی کلمات به یکدیگر مرتبط می‌کند. این کتابخانه، کلمات مترادف و هم معنی را در قالب مجموعه هم معنی ارائه داده و بنابراین می‌توان به‌صورت ترکیب و توسعه‌ای از فرهنگ لغات و اصلاحات در نظر گرفت. پس از انجام مرحله‌های بالا، مجموعه‌ای از کلمات موثر برای هر نیازمندی موجود بوده که با استفاده از این کلمات می‌توان نیازمندی-های نرم‌افزار را خوشه‌بندی کرد. خوشه‌بندی نرم‌افزار به این علت انجام می‌شود تا نیازمندی‌های مرتبط به هم در یک خوشه قرار بگیرند. امروزه برای توسعه بسیاری از سیستم‌های نرم‌افزاری از روش‌های توسعه تکراری استفاده می‌شود. در این روش توسعه تولید نرم‌افزار در قالب تکرارهای مختلفی انجام می‌شود. در هر تکرار بخشی از نیازمندی‌ها پیاده‌سازی شده و بنابراین ضروری است که نیازمندی‌های مرتبط به هم شناسایی شوند. نیازمندی‌هایی که مربوط به یک مجموعه هستند، در یک تکرار پیاده‌سازی شده و با خوشه‌بندی نیازمندی‌ها، رتبه هر خوشه شناسایی و رتبه کلی خوشه‌ها به‌دست می‌آید. تعداد خوشه‌ها نشان‌دهنده تعداد تکرارهای مورد نیاز در توسعه نرم‌افزار است. سپس رتبه هر یک از نیازمندی‌ها در هر خوشه محاسبه شده تا اولویت دقیق هر نیازمندی شناسایی شود.

### ۳-۳- خوشه‌بندی و برچسب‌زنی

برای خوشه‌بندی نیازمندی‌ها باید ویژگی‌های لازم استخراج شوند. روش‌های مختلفی برای استخراج ویژگی از روی متن وجود دارد. در این پژوهش برای استخراج ویژگی‌ها از روش تبدیل جملات به بردار استفاده شده است. در ادامه توضیح مختصری در مورد این روش ارائه می‌شود.
الف- تبدیل جملات به بردار: روش تبدیل جملات به بردار ویژگی‌های لازم را در اختیار الگوریتم خوشه‌بندی قرار می‌دهد. این روش به دو شیوه قابل انجام است. روش ساخت کوله کلمات[۱۷] مشابه شبکه‌های عصبی پیشخور بوده که در آن لایه پنهان غیرخطی برداشته و لایه طرحی برای همه کلمات به اشتراک گذاشته می‌شود. شیوه دوم اسکیپگرام[۱۸] است که شبیه به روش قبل بوده، اما به جای پیش‌بینی کلمه فعلی بر اساس متن، سعی می‌کند طبقه‌بندی یک کلمه را بر اساس یک کلمه دیگر در همان جمله به حداکثر برساند. به‌طور دقیق‌تر، از هر کلمه فعلی به عنوان ورودی به طبقه‌بندی‌کننده خطی-لگاریتمی[۱۹] با لایه طرح‌ریزی مداوم استفاده شده و کلمات را در یک محدوده خاص پیش و بعد از کلمه فعلی پیش‌بینی می‌شود. از آنجا که معمولاً کلمات دورتر با کلمه فعلی ارتباط کمتری با کلمات نزدیک دارند، با نمونه‌گیری کمتر از آن کلمات در مثال‌های آموزش، وزن کمتری به کلمات داده خواهد شد. پیچیدگی آموزش این معماری از رابطه (۱) تعیین می‌شود که در آن c حداکثر فاصله کلمات است.

```
["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself",
"she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this",
"that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing",
"a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into",
"through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further",
"then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such",
"no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now"]
```

**شکل (۵): لیست کلمات توقف در کتابخانه جعبه ابزار زبان طبیعی**

Figure (5): List of stop words in NLTK library

```
['ETCS', 'shall', 'provide', 'driver', 'information', 'allow', 'drive,' 'train', 'safely']
```

**شکل (۶): نتیجه حاصل از حذف کلمات توقف در مثال بالا**

Figure (6): The result of removing stop words in the example above

$$Q=C\times(D+D\times\log_2^{\gamma}) \quad (۱)$$

ب- تعیین تعداد خوشه‌های بهینه: در الگوریتم میانگین k یکی از مواردی که باید مشخص شود تعیین تعداد خوشه‌هاست. برای تعیین تعداد موثر خوشه‌ها در این پژوهش از روش آرنج[۲۰] استفاده می‌شود. این روش کمک خواهد کرد تا تعداد بهینه مراکز خوشه‌ها را پیدا شود. در این روش مجموع مربعات خطا برای مقادیر مختلف K محاسبه می‌شود. برای این منظور، این الگوریتم به K مقادیر مختلف را انتساب می‌دهد و سپس برای هر مقدار K عمل خوشه‌بندی را انجام می‌دهد و هر مقدار K که کمترین مجموع مربعات خطا را ارائه بدهد به عنوان K بهینه انتخاب می‌شود.

ج- خوشه بندی با استفاده از الگوریتم میانگین k: این الگوریتم خوشه‌بندی مبتنی بر پارتیشن[۲۱] است. با مقداردهی اولیه تصادفی مراکز خوشه‌ها شروع و سپس مراحل زیر تکرار می‌شود:

۱) اختصاص دادن هر نقطه داده به نزدیکترین مرکز خوشه

۲) به‌روز رسانی مرکزهای خوشه بر اساس میانگین نقاط داده اختصاص داده شده که این روند تا زمان همگرایی ادامه دارد.

خروجی به‌دست آمده حاصل از اجرای این الگوریتم در شکل (۷) نشان داده شده است. در این شکل، نیازمندی‌های ۱ و ۳ در خوشه شماره صفر و نیازمندی‌های ۲ و ۴ در خوشه شماره یک قرار گرفته‌اند.

د- خوشه‌بندی سلسله مراتبی: خوشه‌بندی سلسله مراتبی، سلسله مراتبی از خوشه‌ها را با ادغام (انباشتگی) یا تقسیم (تقسیم-کننده) خوشه‌ها به‌صورت مکرر ایجاد می‌کند. در روش اول، هر نقطه داده به‌عنوان یک خوشه جداگانه در نظر گرفته شده و شبیه‌ترین خوشه‌ها را شناسایی می‌کند تا زمانی که یک خوشه باقی بماند. خوشه‌بندی تقسیمی، با در نظر گرفتن تمام نقاط داده به‌عنوان یک خوشه شروع می‌شود و به‌صورت بازگشتی آن‌ها را تا زمانی که هر نقطه داده در خوشه خود قرار می‌گیرد، تقسیم می‌کند. در خوشه‌بندی سلسله مراتبی، خوشه‌ها با استفاده از یک ترتیب از پیش تعیین شده، از بالا به پایین ساخته می‌شوند. در این پژوهش از روش تجمیعی برای خوشه‌بندی سلسله مراتبی استفاده شده است. در این روش که یک روش پایین به بالا هست، هر نمونه داده در ابتدا به‌عنوان یک خوشه در نظر گرفته شده و فاصله بین کلاسترها[۲۲] با استفاده از روش وارد[۲۳] محاسبه می‌شود. در این روش دو خوشه مشابه با یکدیگر ترکیب می‌شوند. این روش ادامه پیدا کرده تا زمانی‌که تنها یک خوشه باقی بماند. سپس دندوگرام[۲۴] ترسیم می‌شود. با تحلیل دندوگرام می‌توان تعداد بهینه خوشه‌ها را تعیین نمود. نمونه‌ای از خروجی الگوریتم خوشه‌بندی سلسله مراتبی در شکل (۸) نشان داده شده است.

ه- خوشه‌بندی انتظار بیشینه‌سازی (EM): سومین الگوریتم خوشه‌بندی مورد استفاده در این فاز الگوریتم EM است. این الگوریتم یک الگوریتم بهینه‌سازی تکراری بوده که هدف آن به حداکثر رساندن احتمال تولید داده از یک مدل مخلوط است.

```
Cluster 0: ['ETCS', 'shall', 'provide', 'the', 'driver', 'with', 'information', 'to', 'allow', 'him', 'to', 'drive', 'the', 'train', 'safely']
Cluster 1: ['ETCS', 'shall', 'be', 'able', 'to', 'supervise', 'train', 'and', 'shunting', 'movements]
Cluster 0: ['If', 'the', 'supervision', 'is', 'performed', 'by', 'a', 'RBC', 'it', 'shall', 'be', 'possible', 'to', 'prevent', 'movements', 'of', 'a', 'traction',
'unit', 'in', 'its', 'area', 'if', 'not', 'authorised', 'by', 'the', 'RBC']
Cluster 1: ['ETCS', 'is', 'required', 'to', 'be', 'functional', 'to', 'a', 'maximum', 'train', 'speed', 'of', '500', 'km/h']
```

**شکل (۷): نتیجه حاصل از خوشه‌بندی با استفاده از الگوریتم میانگین کا**

Figure (7): The result of clustering using K-means algorithm

```
Cluster 0: ['ETCS', 'shall', 'provide', 'the', 'driver', 'with', 'information', 'to', 'allow', 'him', 'to', 'drive', 'the', 'train', 'safely['
Cluster 1: ['ETCS', 'shall', 'be', 'able', 'to', 'supervise', 'train', 'and', 'shunting', 'movements']
Cluster 2: ['If', 'the', 'supervision', 'is', 'performed', 'by', 'a', 'RBC', 'it', 'shall', 'be', 'possible', 'to', 'prevent', 'movements', 'of', 'a', 'traction',
'unit', 'in', 'its', 'area', 'if', 'not', 'authorised', 'by', 'the', 'RBC']
Cluster 0: ['ETCS', 'is', 'required', 'to', 'be', 'functional', 'to', 'a', 'maximum', 'train', 'speed', 'of', '500', 'km/h]
```

**شکل (۸): نتیجه حاصل از خوشه‌بندی با استفاده از الگوریتم خوشه‌بندی سلسله مراتبی**

Figure (8): The result of clustering using hierarchical clustering algorithm

```
Cluster 0: ['ETCS', 'shall', 'provide', 'the', 'driver', 'with', 'information', 'to', 'allow', 'him', 'to', 'drive', 'the', 'train', 'safely]
Cluster 1: ['ETCS', 'shall', 'be', 'able', 'to', 'supervise', 'train', 'and', 'shunting', 'movements]
Cluster 2: ['If', 'the', 'supervision', 'is', 'performed', 'by', 'a', 'RBC', 'it', 'shall', 'be', 'possible', 'to', 'prevent', 'movements', 'of', 'a', 'traction', 'unit',
'in', 'its', 'area', 'if', 'not', 'authorised', 'by', 'the', 'RBC']
Cluster 0: ['ETCS', 'is', 'required', 'to', 'be', 'functional', 'to', 'a', 'maximum', 'train', 'speed', 'of', '500', 'km/h]
```

**شکل (۹): نتیجه حاصل از خوشه‌بندی با استفاده از الگوریتم خوشه‌بندی انتظار بیشینه‌سازی**

Figure (9): The result of clustering using the EM clustering algorithm

```
Cluster 0: ['ETCS', 'shall', 'provide', 'the', 'driver', 'with', 'information', 'to', 'allow', 'him', 'to', 'drive', 'the', 'train', 'safely']
Cluster 1: ['ETCS', 'shall', 'be', 'able', 'to', 'supervise', 'train', 'and', 'shunting', 'movements]
Cluster 2: ['If', 'the', 'supervision', 'is', 'performed', 'by', 'a', 'RBC', 'it', 'shall', 'be', 'possible', 'to', 'prevent', 'movements', 'of', 'a', 'traction', 'unit', 'in', 'its', 'area', 'if', 'not', 'authorised', 'by', 'the', 'RBC]
Cluster 0: ['ETCS', 'is', 'required', 'to', 'be', 'functional', 'to', 'a', 'maximum', 'train', 'speed', 'of', '500', 'km/h']
```

**شکل (۱۰): نتیجه حاصل از خوشه‌بندی با استفاده از الگوریتم خوشه‌بندی فضایی مبتنی بر چگالی در کاربردهای دارای نویز**

Figure (10): The result of clustering using the DBSCAN clustering algorithm

```
Cluster 0: ['ETCS', 'shall', 'provide', 'the', 'driver', 'with', 'information', 'to', 'allow', 'him', 'to', 'drive', 'the', 'train', 'safely]
Cluster 1: ['ETCS', 'shall', 'be', 'able', 'to', 'supervise', 'train', 'and', 'shunting', 'movements']
Cluster 2: ['If', 'the', 'supervision', 'is', 'performed', 'by', 'a', 'RBC', 'it', 'shall', 'be', 'possible', 'to', 'prevent', 'movements', 'of', 'a', 'traction', 'unit', 'in', 'its', 'area', 'if', 'not', 'authorised', 'by', 'the', 'RBC']
Cluster 0: ['ETCS', 'is', 'required', 'to', 'be', 'functional', 'to', 'a', 'maximum', 'train', 'speed', 'of', '500', 'km/h]
```

**شکل (۱۱): نتیجه حاصل از خوشه‌بندی با استفاده از الگوریتم خوشه‌بندی** MeanShift

Figure (11): The result of clustering using MeanShift clustering algorithm

در زمینه خوشه‌بندی، اغلب از مدل مخلوط گاوسی[۲۵] استفاده می‌شود. به‌طور تکراری، یک مرحله انتظار (E-step)، که در آن احتمال هر نقطه داده متعلق به هر خوشه را محاسبه می‌کند و یک مرحله حداکثرسازی (M-step)، که در آن پارامترهای توزیع‌های گاوسی را به‌روز رسانی می‌کند انجام می‌شود. نمونه‌ای از خروجی این الگوریتم برای خوشه‌بندی چهار نیازمندی در شکل (۹) نشان داده شده است.

ی- خوشه‌بندی فضایی مبتنی بر چگالی در کاربردهای دارای نویز[۲۶]: چهارمین الگوریتم خوشه‌بندی مورد استفاده در این فاز الگوریتم خوشه‌بندی فضایی مبتنی بر چگالی در کاربردهای دارای نویز (DBSCAN) است. این الگوریتم خوشه‌بندی مبتنی بر چگالی است که نقاط داده‌ای را که در فضای ویژگی نزدیک به یکدیگر هستند را کنار هم قرار می‌دهد و مناطق با چگالی کمتر را از هم جدا می‌کند. این خوشه‌ها را به عنوان مناطق متراکم که توسط مناطق با تراکم کمتر از هم جدا شده‌اند تعریف می‌کند. الگوریتم با یک نقطه داده دلخواه شروع می‌شود و با اتصال نقاط همسایه در یک آستانه فاصله مشخص، خوشه را گسترش می‌دهد. نمونه‌ای از خروجی این الگوریتم برای خوشه‌بندی چهار نیازمندی در شکل (۱۰) نشان داده شده است.

ط- خوشه‌بندی میانگین شیفت[۲۷]: پنجمین الگوریتم خوشه‌بندی مورد استفاده در این فاز الگوریتم میانگین شیفت (MS) است. این الگوریتم خوشه‌بندی مبتنی بر چگالی است که هدف آن کشف حالت‌ها یا مناطق متراکم در توزیع داده است. با مجموعه‌ای از نقاط داده شروع می‌شود و به‌طور مکرر نقاط را به سمت مناطق با تراکم بالاتر با به‌روزرسانی موقعیت آن‌ها بر اساس میانگین نقاط در یک پنجره فاصله مشخص تغییر می‌دهد. نمونه‌ای از خروجی این الگوریتم برای خوشه‌بندی چهار نیازمندی در شکل (۱۱) نشان داده شده است.

### ۴-۳- اولویت‌بندی

برای رتبه‌دهی به هر یک از خوشه‌های به‌دست آمده از مرحله قبل، ابتدا رتبه هر یک از نیازمندی‌های موجود در یک خوشه محاسبه شده و سپس مجموع امتیازهای به‌دست آمده محاسبه می‌شود. پس از محاسبه مجموع امتیازها، این امتیازها نرمال‌سازی می‌شوند. پس از محاسبه امتیازها برای هر یک از خوشه‌ها می‌توان خوشه‌ها را رتبه‌بندی کرد و تکرارهای مختلف فرآیند توسعه را بر اساس آن‌ها مرتب کرد. برای تعیین رتبه هر یک از نیازمندی‌ها بر اساس درجه اهمیت قابل استخراج از کلمات ذکر شده در آن نیازمندی محاسبه می‌شود. عموماً در زمان استخراج نیازمندی‌های کاربران از کلماتی استفاده می‌کنند که نشان‌دهنده درجه اهمیت نیازمندی برای آن‌هاست. برای مثال لیست زیر برخی از کلمات مورد استفاده را ارائه می‌دهد که سطوح مختلف نیازمندی‌ها را نشان می‌دهند: ۱) باید، ۲) می‌تواند، ۳) بهتر است و موارد مشابه.

طبیعی است وقتی کلمه "باید" توسط یک کاربر گفته می‌شود در مقایسه با زمانی که کلمه "می‌تواند" را استفاده می‌کند، درجه اولویت بالاتری را مشخص می‌کند و نشان دهنده اهمیت بیشتر آن نیازمندی توسط کاربر است. در توسعه تکراری نرم‌افزار عموما نیازمندی‌های کلیدی که نشان‌دهنده اولویت بالاتری هستند در یک مجموعه قرار می‌گیرند و نیازمندی‌های دارای اولویت کمتر در گروه بعدی قرار گرفته و این فرآیند را بر اساس تعداد تکرارهای مورد نظر می‌توان انجام داد. بنابراین به نظر می‌رسد این روش بتواند به‌طور موثری فرآیند تعیین تکرارهای نرم‌افزار را تسهیل کند.

## ۴- بررسی کارایی

### ۴-۱- پیکربندی الگوریتم‌ها

الگوریتم‌های خوشه‌بندی جهت اجرا نیازمند تعیین برخی از پارامترها به‌صورت دستی است. در ادامه مقادیر پارامترها برای هر یک از الگوریتم‌ها ارائه شده است. مقادیر انتساب داده شده به پارامترها جهت انجام آزمایش‌ها به‌صورت تجربی تعیین شده است. در الگوریتم میانگین k تعداد خوشه‌ها برابر با پنج قرار داده شده است. الگوریتم خوشه‌بندی سلسله مراتبی نیازمند تنظیم پارامترها توسط کاربر نیست. در الگوریتم EM نیز تعداد خوشه‌ها برابر با پنج قرار داده شده است. مقدار اپسیلون در الگوریتم DBSCAN برابر ۰/۱۵ انتخاب شده و همچنین تعداد مینیمم نقاط برابر شش قرار داده شده است. در الگوریتم MS نیاز به تنظیم پارامترها به‌صورت دستی نیست و به‌صورت خودکار محاسبه می‌شود. برای الگوریتم ABC اندازه جمعیت برابر صد و تعداد تکرار برابر هزار در نظر گرفته شده است. الگوریتم‌های مطالعه شده بر روی یک کامپیوتر Corei5 با ۸ گیگابایت حافظه اصلی اجرا شده‌اند.

### ۴-۲- ارزیابی عملکرد

در این بخش کارایی روش پیشنهادی برای اولویت‌بندی نیازمندی‌ها ارائه می‌شود. روش پیشنهادی بر روی نیازمندی‌های چهار نرم‌افزار مختلف اعمال شده و نتایج به‌دست آمده با روش رتبه‌بندی دستی که توسط فرد متخصص انجام شده مقایسه شده‌اند. در جدول (۱) دقت گروه‌بندی هر یک از روش‌های خوشه‌بندی در مقایسه با امتیازبندی و گروه‌بندی افراد متخصص ارائه شده است. رتبه‌های به‌دست آمده برای هر یک از نیازمندی‌ها در مقایسه با رتبه‌های تعیین شده توسط فرد خبره مقایسه می‌شود. برای محاسبه دقت الگوریتم در این روش رتبه هر نیازمندی که توسط یک الگوریتم محاسبه شده است با رتبه تعیین شده توسط فرد خبره مقایسه شده و میزان اختلاف رتبه‌ها به‌دست می‌آید. سپس مجموع اختلاف‌های رتبه‌ها جمع شده و میانگین اختلاف رتبه محاسبه می‌شود. طبیعی است در این محاسبه هر چه میزان اختلاف‌ها بیشتر باشد نشان‌دهنده خطای بیشتری در رتبه‌بندی است و هر الگوریتمی که خطای کمتری را تولید کند الگوریتم بهتری است. رابطه محاسبه خطا به‌صورت زیر است:

$$\text{Error}=\sum_{i=1}^{n}\frac{|r(i.alg)-r(i.opt)|}{n} \qquad (۲)$$

در این رابطه n نشان دهنده تعداد نیازمندی‌ها، r(i,alg) نشان دهنده رتبه به‌دست آمده برای نیازمندی i توسط الگوریتم خوشه‌بندی است و r(i.opt) نشان دهنده رتبه پیشنهادی توسط فرد خبره برای نیازمندی i است. کارایی روش‌های خوشه‌بندی در اولویت‌بندی نیازمندی‌های نرم‌افزارهای مدیریت ترافیک راه‌آهن اروپا، فروشگاه الکترونیکی گاما-ج[۲۸]، سیستم کنترلی جمینی[۲۹] و سیستم ارتباطات بی‌سیم در جدول (۱) نشان داده شده است. در مورد نرم‌افزار مدیریت ترافیک راه‌آهن اروپا، نتایج نشان می‌دهد الگوریتم DBSCAN کارایی بهتری نسبت به سایر الگوریتم‌ها داشته و خطای کمتری را تولید می‌کند و در رتبه بعد الگوریتم EM قرار می‌گیرد. روش‌های پیشنهادی با مطالعه ارائه شده در مرجع‌های [۱۹] و [۲۳] مقایسه شده است.

نتایج نشان می‌دهد الگوریتم DBSCAN کارایی بهتری در اولویت‌بندی نیازمندی‌های فروشگاه الکترونیکی گاما-ج نسبت به سایر الگوریتم‌ها داشته و خطای کمتری را تولید می‌کند و در رتبه بعد الگوریتم EM قرار می‌گیرد. این الگوریتم کارایی بهتری در اولویت‌بندی نیازمندی‌های سیستم کنترلی جمینی نسبت به سایر الگوریتم‌ها داشته و خطای کمتری را تولید می‌کند و در رتبه بعد الگوریتم DBSCAN قرار می‌گیرد. الگوریتم‌های دیگر در رتبه‌های بعدی قرار می‌گیرند. الگوریتم DBSCAN کارایی بهتری در اولویت‌بندی نیازمندی‌های سیستم ارتباطات بی‌سیم نسبت به سایر الگوریتم‌ها داشته و خطای کمتری را تولید می‌کند و در رتبه بعد الگوریتم EM قرار می گیرد. این دو الگوریتم به‌طور موثر از توزیع بردار کلمات که توسط الگوریتم Word2Vec تولید می‌شود استفاده می‌کنند. همچنین شباهت‌های معنایی که در کلمات وجود دارد منجر به تولید بردارهای مشابه برای کلمات دارای شباهت معنایی می‌شود. این دو الگوریتم نسبت به بقیه الگوریتم‌ها قابلیت بهتری جهت استفاده از این ویژگی دارند.

در شکل (۱۲) بهترین خوشه‌های به‌دست آمده توسط الگوریتم‌ها ترسیم شده است. به علت این‌که تعداد پارامترها برای خوشه-بندی بیشتر از دو است لذا جهت ارائه مناسب خوشه‌ها از روش تجزیه و تحلیل مؤلفه‌های اصلی[۳۰] (PCA) یا تحلیل مولفه‌های

اصلی استفاده شده تا تعداد ابعاد به دو کاهش یابند و خوشه‌ها ترسیم شوند. محور افقی مولفه اول و محور عمودی مولفه دوم را به ترتیب نشان می‌دهند. لذا احتمال از دست رفتن برخی از اطلاعات اولیه نیز در این روش وجود دارد. با توجه به کاهش تعداد ابعاد و توجه به این نکته که خوشه‌بندی‌های اولیه توسط PCA حفظ شده‌اند، طبیعی است که در این اشکال خوشه‌های این‌چنین مشاهده شود.

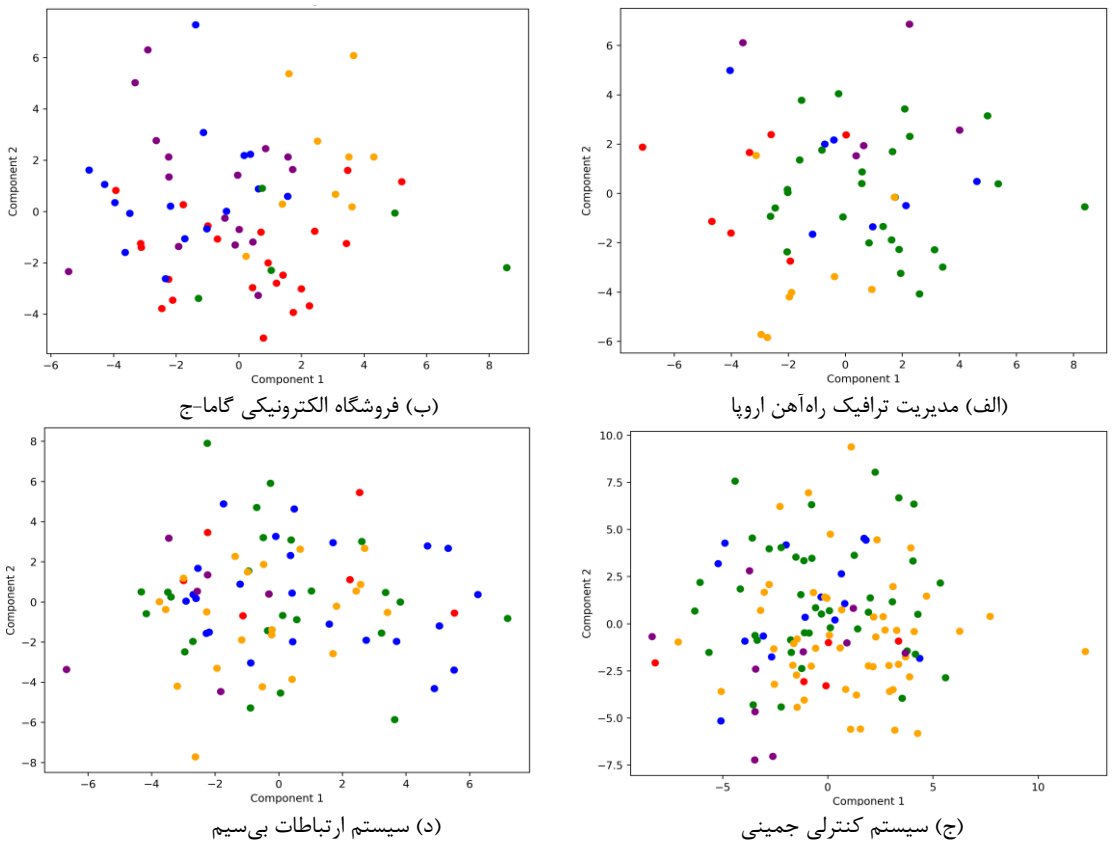
### ۳-۴- پیچیدگی زمانی

پیچیدگی زمانی الگوریتم‌های خوشه‌بندی در جدول (۲) نشان داده شده است. با توجه به اینکه این الگوریتم‌ها دارای ساختارهای متفاوت و روش‌های خوشه‌بندی متفاوت هستند، به همین دلیل از پارامترهای مختلفی نیز استفاده می‌کنند. تعداد نقاط داده در همه الگوریتم‌ها به عنوان معیار پیچیدگی زمانی وجود دارد.

Table (1): Efficiency of clustering methods on software requirements

**جدول (۱): کارایی روش‌های خوشه‌بندی بر روی نیازمندی‌های نرم‌افزار**

| الگوریتم | مدیریت ترافیک راه آهن اروپا | فروشگاه الکترونیکی GAMMA-J | سیستم کنترلی جمینی | سیستم ارتباطات بی‌سیم |
|---|---|---|---|---|
| میانگین کا | ۰/۱۹ | ۰/۲ | ۰/۱۶ | ۰/۱۶ |
| سلسله مراتبی | ۰/۲۱ | ۰/۱۸ | ۰/۱۴ | ۰/۱۶ |
| انتظار بیشینه‌سازی | ۰/۱۴ | ۰/۱۴ | ۰/۱ | ۰/۱۲ |
| دی-بی اسکن | ۰/۱۲ | ۰/۱ | ۰/۱۱ | ۰/۰۹ |
| متوسط شیفت | ۰/۱۷ | ۰/۱۵ | ۰/۱۷ | ۰/۱۴ |
| الگوریتم کلونی زنبورهای مصنوعی [۱۸] | ۰/۲ | ۰/۲۲ | ۰/۱۶ | ۰/۱۸ |
| وابستگی اجرا مشترک نیمه خودکار [۲۲] | ۰/۱۶ | ۰/۱۷ | ۰/۱۲ | ۰/۱۳ |




(الف) مدیریت ترافیک راه‌آهن اروپا (ب) فروشگاه الکترونیکی گاما-ج

(ج) سیستم کنترلی جمینی (د) سیستم ارتباطات بی‌سیم

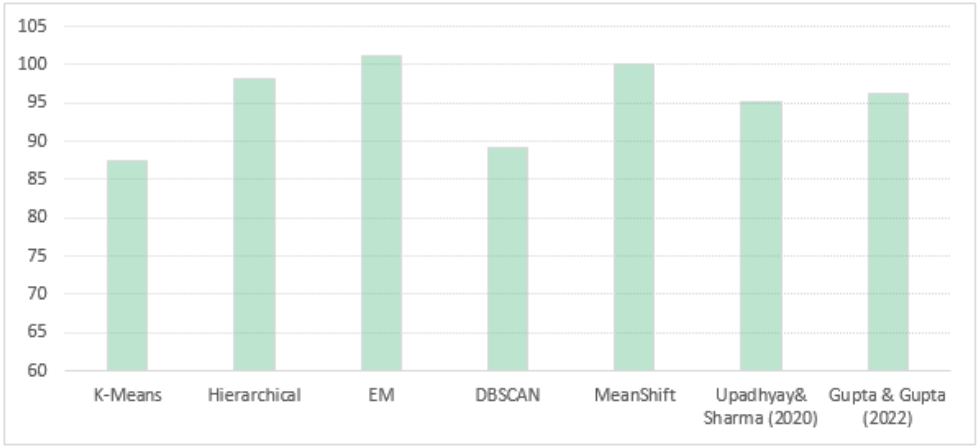
**شکل (۱۲): بهترین خوشه‌های به دست آمده پس از اعمال روش تحلیل مؤلفه‌های اصلی**

Figure (12): The best clusters obtained after applying the PCA method, a) European railway traffic management, b) GAMMA-J electronic store, c) Gemini control system, d) Wireless communication system

Table (2): Comparison of time complexity of clustering algorithms

**جدول (۲): مقایسه پیچیدگی زمانی الگوریتم‌های خوشه‌بندی**

| الگوریتم | پیچیدگی زمانی | پارامترها |
|---|---|---|
| میانگین k | $O(t\times n\times k\times d)$ | t: تعداد تکرار، n: تعداد نقاط داده، k: تعداد خوشه‌ها، d: ابعاد داده |
| سلسله مراتبی | $O(n^2 \log n)$ | n: تعداد نقاط داده |
| انتظار بیشینه‌سازی | $O(t\times n\times k\times d)$ | t: تعداد تکرار، n: تعداد نقاط داده، k: تعداد خوشه‌ها، d: ابعاد داده |
| دی-بی اسکن | $O(n^2)$ | n: تعداد نقاط داده |
| متوسط شیفت | $O(n^2\times d)$ | n: تعداد نقاط داده، d: ابعاد داده |
| الگوریتم کلونی زنبورهای مصنوعی [۱۸] | $O(t\times n\times k\times d)$ | t: تعداد تکرار، n: تعداد نقاط داده، k: تعداد خوشه‌ها، d: ابعاد داده |
| وابستگی اجرا مشترک نیمه خودکار [۲۲] | $O(t\times n\times k\times d)$ | t: تعداد تکرار، n: تعداد نقاط داده، k: تعداد خوشه‌ها، d: ابعاد داده |




**شکل (۱۳): مقایسه زمان اجرای الگوریتم‌های خوشه‌بندی**

Figure (13): Comparison of execution time of clustering algorithms

تعداد تکرار، تعداد خوشه و ابعاد داده در چهار الگوریتم به‌طور مشترک وجود دارد. اما این سه معیار جزو پیچیدگی زمانی الگوریتم‌های خوشه‌بندی سلسله مراتبی و DBSCAN نیستند. پیچیدگی زمانی DBSCAN از پیچیدگی زمانی الگوریتم‌های سلسله مراتبی و میانگین شیفت کمتر است. زمان اجرای الگوریتم‌ها بر حسب ثانیه در نمودار شکل (۱۳) نشان داده شده است. محور افقی نشان‌دهنده نام الگوریتم‌ها و محور عمودی نشان‌دهنده زمان اجراست. اکثر الگوریتم‌ها دارای زمان اجرای نزدیک به هم هستند. الگوریتم‌های میانگین k و DBSCAN دارای زمان‌های اجرای کمتری هستند. اگرچه اختلاف زمان اجرای آن‌ها با سایر الگوریتم‌ها چندان قابل توجه نیست. الگوریتم میانگین k دارای زمان اجرای کمتری نسبت به الگوریتم DBSCAN است اما این اختلاف زمان چندان قابل توجه نیست.

### ۴-۴- ارزیابی کلی کارایی

با بررسی کارایی کلی روش‌های مطالعه شده در این تحقیق، به نظر می‌رسد الگوریتم‌های DBSCAN و EM کارایی بهتری را در مقایسه با الگوریتم‌های دیگر ارائه می‌دهند. به هر حال به نظر می‌رسد تمامی الگوریتم‌ها قابلیت خوبی جهت رتبه‌بندی خودکار نیازمندی‌ها دارا می‌باشند. میانگین خطای کلی بر روی مجموعه داده‌ها در جدول (۳) نشان داده شده است. الگوریتم DBSCAN نسبت به سایر الگوریتم‌های مورد مطالعه کارایی بهتری را نشان می‌دهد. این الگوریتم بر خلاف برخی از الگوریتم‌های خوشه‌بندی مانند میانگین k نیازی به تعیین تعداد خوشه‌ها در داده‌ها ندارد. در این مطالعه برای تعیین تعداد خوشه‌ها از روش آرنج استفاده شده است. اگرچه این روش معمولا برای تعیین تعداد خوشه‌ها استفاده می‌شود اما به هر حال دارای محدودیت‌هایی هم هست و ممکن است در تعیین تعداد دقیق خوشه‌ها دارای خطا باشد و همین مساله روی کارایی کلی الگوریتم‌هایی که از این روش استفاده می‌کنند تاثیرگذار است.

Table (3): Average error on all data

**جدول (۳): میانگین خطا بر روی همه داده‌ها**

| الگوریتم | کارایی کلی (برحسب درصد) |
|---|---|
| میانگین کا | ۱۷٫۷۵ |
| سلسله مراتبی | ۱۷٫۲۵ |
| انتظار بیشینه‌سازی | ۱۲٫۵ |
| دی-بی‌اسکن | ۱۰٫۵ |
| متوسط شیفت | ۱۵٫۷۵ |
| الگوریتم کلونی زنبورهای مصنوعی [۱۸] | ۱۹ |
| وابستگی اجرا مشترک نیمه خودکار [۲۲] | ۱۴٫۵ |

یکی از دلایلی که روی کارایی الگوریتم‌های خوشه‌بندی تاثیرگذار است، شکل کلی خوشه‌هایی است که از روی داده‌ها قابل استخراج هستند. یکی از نقاط قوت الگوریتم DBSCAN این است که می‌تواند خوشه‌هایی با شکل دلخواه پیدا کند. حتی می‌تواند خوشه‌ای را پیدا کند که به طور کامل توسط یک خوشه (اما متصل به) متفاوت احاطه شده است. با توجه به پارامتر MinPts، اثر به اصطلاح تک پیوند یا خوشه‌های مختلف که توسط یک خط نازک از نقاط به هم متصل می‌شوند کاهش می‌یابد و این مساله روی بهبود کارایی الگوریتم تاثیرگذار است. یکی از دلایل دیگری که الگوریتم DBSCAN کارایی بهتری را ارائه می‌دهد این است که این الگوریتم دارای مفهوم نویز است و نسبت به نمونه‌های پرت مقاومت بیشتری از خود نشان می‌دهد. به دلیل اینکه متن ورودی به زبان طبیعی است، مستعد وجود داده‌هایی است که نسبت به داده‌های دیگر ممکن است دارای فاصله قابل توجه باشند. همچنین این الگوریتم فقط به دو پارامتر نیاز دارد و عمدتاً به ترتیب نقاط در مجموعه داده حساس نیست. این در حالی است که در برخی از الگوریتم‌های خوشه‌بندی ترتیب ارائه نقاط در مجموعه داده ممکن است بر روی کارایی آن‌ها تاثیرگذار باشد. بعلاوه، نقاطی که در لبه دو خوشه مختلف قرار دارند، ممکن است در صورت تغییر ترتیب نقاط، عضویت خوشه را با هم عوض‌کنند، و انتساب خوشه فقط تا هم شکلی منحصر به فرد است.

## ۵- نتیجه‌گیری

در این مقاله روش جدیدی برای اولویت‌بندی نیازمندی‌های نرم‌افزار با استفاده از الگوریتم‌های خوشه‌بندی و شباهت معنایی ارائه شده است. روش پیشنهادی دارای چند مرحله مختلف است که در هر مرحله پردازش‌های متفاوتی بر روی داده‌ها انجام می‌شود. روش پیشنهادی با دریافت مجموعه از نیازمندی‌ها کار خود را شروع می‌کند و سپس با انجام پردازش‌های مختلف در فازهای پیش رو، فرآیند برچسب‌زنی و تعیین اولویت نیازمندی‌ها را انجام می‌دهد. در نهایت رتبه هر یک از نیازمندی‌ها را به عنوان خروجی ارائه می‌دهد. روش پیشنهادی دارای کارایی مناسب جهت تعیین اولویت نیازمندی‌ها است و اولویت‌بندی نیازمندی‌ها را به شیوه موثری انجام می‌دهد. از بین روش‌های پیشنهادی الگوریتم خوشه‌بندی DBSCAN دارای کارایی بهتری نسبت به سایر الگوریتم‌ها بوده و خطای کمتری دارد. این الگوریتم قابلیت بهره‌برداری از توزیع کلمات و ارتباط معنایی کلمات که توسط مراحل قبلی تهیه شده‌اند را دارد. همچنین این الگوریتم می‌تواند خوشه‌ها را با هر شکلی مانند خطی، دایره‌ای یا نامنظم شناسایی کند. این انعطاف‌پذیری به DBSCAN اجازه می‌دهد تا ساختارهای پیچیده‌ای را در داده‌ها ثبت کند که ممکن است به راحتی توسط الگوریتم‌هایی که شکل خوشه‌ای خاص را در نظر می‌گیرند، مانند میانگین k یا خوشه‌بندی سلسله مراتبی، به‌راحتی ثبت نشوند. همچنین این الگوریتم کارایی خوبی برای کنترل موثر نویز و موارد پرت بوده و بین نقاط اصلی (مناطق متراکم درون خوشه‌ها)، نقاط مرزی (نقاط در حاشیه خوشه‌ها) و نقاط نویز (نقاط جدا شده) تمایز قائل می‌شود. با در نظر گرفتن تراکم نقاط، DBSCAN می‌تواند نقاط داده پرت را شناسایی و در نتیجه انتساب‌های خوشه‌ای دقیق‌تر را انجام دهد. همچنین این الگوریتم دارای پارامترهایی است که با تنظیم آن‌ها می‌توان نتایج خوشه‌بندی را بهینه کرد.

**سپاسگزاری**

این مقاله مستخرج از پایان‌نامه دوره کارشناسی‌ارشد در دانشگاه آزاد اسلامی واحد لارستان است. نویسندگان بر خود لازم می‌دانند مراتب تشکر صمیمانه خود را از همکاران حوزه پژوهشی دانشگاه آزاد اسلامی و داوران محترم که ما را در انجام و ارتقای کیفی این مقاله یاری نموده‌اند، اعلام نمایند.

## References

## مراجع

[1] P. Achimugu, A. Selamat, R. Ibrahim, M.N.R. Mahrin, "A systematic literature review of software requirements prioritization research", Information and Software Technology, vol. 56, no. 6, pp. 568-585, June 2014 (doi: 10.1016/j.infsof.2014.02.001).

[2] L. Alawneh, "Requirements prioritization using hierarchical dependencies", Information Technology - New Generations, vol. 558, pp. 459-464, 2018 (doi: 10.1007/978-3-319-54978-1_59).

[3] J.R.F.D. Santos, A.B. Albuquerque, P.R. Pinheiro, "Requirements prioritization in market-driven software: A survey based on large numbers of stakeholders and requirements", Proceeding of the IEEE/QUATIC, pp. 67-72, Lisbon, Portugal, Sept. 2016 (doi**:** 10.1109/QUATIC.2016.020).

[4] J. Khan, I. U. Rehman, L. Ali, S. Khan, I.J. Khan, "Requirements prioritization using analytic network process (anp)", International Journal of Scientific and Engineering Research, vol. 7, no. 11, Nov. 2016.

[5] E. Serral, P. Sernani, A.F. Dragoni, F. Dalpiaz, "Contextual requirements prioritization and its application to smart homes", Ambient Intelligence, vol. 13, pp. 94-109, April 2017 (doi: 10.1007/978-3-319-56997-0_7).

[6] S.D. Federico, S. Gonnet, "New requirements prioritization based on customer historical profiles", Proceeding of the IEEE/CLEI, pp. 1-8, Valparaiso, Chile, Oct. 2016 (doi**:** 10.1109/CLEI.2016.7833344).

[7] A. Corezolla, L. Costa, F.C. Souza, A.C. Souza, "Investigating fitness functions for search-based requirements prioritization", Anais do Computer on the Beach, vol 12, pp. 451-458, April 2021 (doi: 10.14210/cotb.v12.p-451-458).

[8] K. AbdElazim, R. Moawad, E. Elfakharany, "A framework for requirements prioritization process in agile software development", Journal of Physics: Conference Series, vol. 1454, no. 1, Article Number: 012001, Feb. 2020 (doi: 10.1088/1742-6596/1454/1/012001).

[9] I. Ibriwesh, S.B. Ho, I. Chai, C.H. Tan, "Prioritizing solution-oriented software requirements using the multiple perspective prioritization technique algorithm: An empirical investigation", Concurrent Engineering, vol. 27, no. 1, pp. 68-79, March 2019 (doi: 10.1177/1063293X188085).

[10] N.P.B. Arévalo, M.F.C. Carrasco, J.L.T. Espinoza, M.V. Córdova, "Neutrosophic AHP for the prioritization of requirements for a computerized facial recognition system", Neutrosophic Sets and Systems, vol. 34, no. 1, pp. 21, June 2020.

[11] M. Yaseen, N. Ibrahim, A. Mustapha, "Requirements prioritization and using iteration model for successful implementation of requirements", International Journal of Advanced Computer Science and Applications, vol. 10, no. 1, Jan. 2019 (doi: 10.14569/IJACSA.2019.0100115).

[12] F.A. Bukhsh, Z.A. Bukhsh, M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation", Computer Standards & Interfaces, vol 69, Article Number: 103389, March 2020 (doi: 10.1016/j.csi.2019.103389).

[13] H. Tufail, I. Qasim, M.F. Masood, S. Tanvir, W.H. Butt, "Towards the selection of optimum requirements prioritization technique: a comparative analysis", Proceeding of the IEEE/ICIM, pp. 227-231, Cambridge, UK, March 2019 (doi: 10.1109/INFOMAN.2019.8714709).

[14] A. Ali, Y. Hafeez, S. Hussain, S. Yang, "Role of requirement prioritization technique to improve the quality of highly-configurable systems", IEEE Access, vol 8, pp. 27549-27573, Feb. 2020 (doi: 10.1109/ACCESS.2-020.2971382).

[15] N. Govil, A. Sharma "Information extraction on requirement prioritization approaches in agile software development processes", Proceeding of the IEEE/ICCMC, pp. 1097-1100, Erode, India, April 2021 (doi: 10.1109/ICCMC51019.2021.9418285).

[16] K.J. Kumar, N. Rajkumar, "Improving energy-efficient management for identifying software requirement prioritization based on optimized fuzzy logic social spider optimization", Personal and Ubiquitous Computing, vol. 27, pp. 1419-1428, June 2023 (doi: 10.1007/s00779-021-01617-1).

[17] S. Ali, Y. Hafeez, M. Humayun, N.Z. Jhanjhi, D.N. Le, "Towards aspect based requirements mining for trace retrieval of component-based software management process in globally distributed environment", Information Technology and Management, vol 23 no. 3, pp. 151-165, Nov. 2022 (doi: 10.1007/s10799-021-00343-7).

[18] N. Upadhyay, A. Sharma, "Requirement prioritization based on cost using artificial bee colony algorithm", Proceeding of the IEEE/ICRITO, pp. 426-430, Noida, India, June 2020 (doi: 10.1109/ICRITO488-77.2020.9197941).
[19] T.Z. Win, R. Mohamed, J. Sallim, "Requirement prioritization based on non-functional requirement classification using hierarchy AHP", IOP Conference Series: Materials Science and Engineering, vol. 769, no. 1, Article Number: 012060, Feb. 2020 (doi**:** 10.1088/1757-899X/769/1/012060).
[20] G.D. Rottoli, C. Casanova, "Multi-criteria group requirement prioritization in software engineering using fuzzy linguistic labels", Proceeding of the ICAIW, pp. 16-28, Buenos Aires, Argentina, Oct. 2021.
[21] A. Ejnioui, CE Otero, AA Qureshi, "Software requirement prioritization using fuzzy multi-attribute decision making", Proceeding of the IEEE/COS, pp. 1-6, Kuala Lumpur, Malaysia, Oct. 2012 (doi: 10.1109/ICOS.2-012.6417646).
[22] Y. Kuengjai, L. Ramingwong, "A pilot study of requirement prioritization techniques in agile software development", Proceedings of the CSSE, pp. 146-151, Singapore, Oct. 2021 (doi: 10.1145/3494885.3494912).
[23] A. Gupta, C. Gupta, "CDBR: A semi-automated collaborative execute-before-after dependency-based requirement prioritization approach", Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 2, pp. 421-432, Feb. 2022 (doi: 10.1016/j.jksuci.2018.10.004).

**زیرنویس‌ها**

1. k-means clustering
2. Clustering algorithms
3. Analytical hierarchy process (AHP)
4. Value oriented prioritization
5. Binary search tree (BST)
6. Bubble sort
7. Activity based costing
8. Analytical hierarchy process
9. Viger's method
10. Prioritizing multiple perspectives
11. AHP-based method
12. Neutrosphic analytical hierarchy technique
13. PUblic REquirements dataset (PURE)
14. Natural language toolkit
15. Semantic web library
16. Wordnet
17. Bag of words
18. Skipgram
19. Log-linear
20. Elbow method
21. Partition based clustering algorithm
22. Distance between clusters
23. Ward
24. Dendogram
25. Gaussian mixture model
26. Density based spatial clustering of applications with noise
27. Mean shift
28. Gamma-J
29. Gemini
30. Principal component analysis