La Ingeniería de Requisitos potenciada por Inteligencia Artificial: Un enfoque para la detección temprana de errores de clasificación en los Requisitos No Funcionales en el desarrollo de software.

Gonzalez Carlos Ezequiel[1].

[1]Escuela de Informática, Universidad Nacional del Oeste, Argentina
ezequieleg831785@gmail.com

Resumen. La Ingeniería de Requisitos se erige como un pilar fundamental en el desarrollo de software, responsable de la definición, análisis y validación de las necesidades de los stakeholders. En este contexto, la Inteligencia Artificial emerge como una herramienta transformadora, ofreciendo soluciones innovadoras para potenciar el análisis y la viabilidad de sistemas de software. En tal sentido, los sistemas impulsados por la Inteligencia Artificial se traducen en una reducción de costos y riesgos en el desarrollo temprano del software. La identificación temprana de problemas potenciales permite tomar medidas preventivas y mitigar su impacto negativo en las etapas posteriores del desarrollo. De este modo, este artículo se plantea demostrar mediante un estudio de caso práctico la implementación de clasificadores, es decir, la alta efectividad que posee la aplicación de la Inteligencia Artificial para la clasificación de Requisitos No Funcionales en la construcción de sistemas de software, en comparación con estudios realizados previamente en otras investigaciones con algoritmos estándares. Por consiguiente, se propone la incorporación de Inteligencia Artificial para mitigar el impacto de la mala clasificación y evitando así ambigüedades y errores en la clasificación de Requisitos No Funcionales en una etapa temprana de desarrollo, lo que hace a la Inteligencia Artificial una herramienta valiosa para el desarrollo de software de alta calidad.

Palabras Clave: Ingeniería de Requisitos, Inteligencia Artificial, Requisitos No funcionales, Eficiencia en Requisitos, Viabilidad en Requisitos.

1. Introducción.

El presente artículo se desarrolla en el marco de la propuesta sobre la aplicación de la Inteligencia Artificial en el proceso de requisitos, de la materia Ingeniería de Software II de la carrera Licenciatura en Informática de la Universidad Nacional del Oeste. El proceso de especificación de Requisitos No Funcionales, exige un análisis exhaustivo y una alta precisión en la información para garantizar el correcto funcionamiento del sistema, donde en otras palabras menciona Bohem como un proceso lento y propenso a errores [11]. En el contexto actual, los sistemas demandan una mayor seguridad dado que son sistemas cada vez más complejos, en el cual los Requisitos No Funcionales son de vital importancia, ya que la especificación de estos, además de garantizar seguridad y fiabilidad en el software, puede generar nuevos requerimientos funcionales vitales en

el servicio de un sistema [1]. De este modo, la detección temprana de los Requisitos No Funcionales ayuda a reducir la refactorización posterior [1], ocasionando que las especificaciones con ambigüedades, errores y la perdida de la trazabilidad se reduzcan significativamente y evitando defectos en los proyectos de software [13], por otro lado, aumenten la fiabilidad y la seguridad. Estos serían los principales desafíos de la Inteligencia Artificial en el desempeño de la especificación y clasificación de Requisitos [9,10]. De este modo, en el presente trabajo expone la utilización de la inteligencia Artificial, como una herramienta potencial y eficaz para automatizar, optimizar y mejorar la precisión en la obtención de los Requisitos No Funcionales. Asimismo, Se presentan estudios que muestran la efectividad de la identificación automática de los Requisitos No Funcionales, donde se demostró mediante un clasificador la recuperación 87% de los posibles Requisitos No Funcionales en un conjunto de datos, además de evidenciar que la Inteligencia Artificial puede entrenarse y reentrenarse para adaptarse en la clasificación y detección de estos [1], por último se realizó el correspondiente estudio de caso para, posteriormente hacer la respectiva comparación de los resultados obtenidos. La sección 2 describe las técnicas de detección y obtención de los Requisitos No Funcionales. La sección 3 Aplicación de la Inteligencia Artificial en la automatización de la clasificación de los Requisitos No Funcionales. La sección 4 Análisis de factibilidad comparativa. La sección 5 expone conclusiones y trabajos futuros.

2. Técnicas de detección y obtención de los Requisitos No Funcionales.

Principalmente, los métodos de obtención y clasificación se basan en dos enfoques principales. Estos incluyen la obtención de la información de los stakeholders y la detección semiautomática o manual de los Requisitos No Funcionales [1]. Pero existen otras técnicas orientadas a objetivos, que proporcionan una notación y un entorno en el que los analistas pueden analizar los Requisitos No Funcionales [1]. Sin embargo, su proceso de especificación puede no estar completos, de este modo la Inteligencia Artificial puede brindar apoyo gestionando las omisiones importantes. A su vez, se tiene técnicas de detección que han si desarrolladas por un grupo de investigadores, donde se aplicó la Inteligencia Artificial en la programación orientada a aspectos para el diseño y código de requisitos a un bajo nivel, extrayendo clones con la comparación de patrones de las relaciones entre comportamientos [1], en el cual es ampliado en el artículo [3]. Estas técnicas, se tienen en cuenta un esquema general de un sistema de detección y diagnóstico de fallas, en el siguiente esquema se detalla (ver Fig-1).

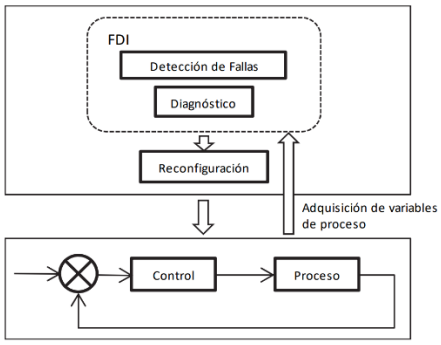

Fig. 1 sistema de detección y diagnóstico de fallas.
Fuente: [2].

De este modo, se tiene una supervisión en las entradas y salidas de los parámetros. Múltiples modelos fueron aplicados, Algunos de estos ejemplos fueron, modelos matemáticos, modelos obtenidos de la recolección de grandes volúmenes de datos, entre otros. Pero dado la naturaleza de los sistemas actuales, se requiere mucho coste humano para recuperar y diagnosticar errores en los Requisitos No Funcionales [1], en la que describió Rosenhainer como "punto de partida", para encontrar buenos requisitos iniciales [1,4]. En tales casos, la Inteligencia Artificial aplicada en la obtención de los Requisito No funcionales han demostrado una gran reducción de ambigüedades, coste y tiempo en su proceso de especificación.

3. Aplicación de la Inteligencia Artificial en la automatización de la clasificación de los Requisitos No Funcionales.

La aplicación de la Inteligencia Artificial se implementó en clasificadores de Requisitos No Funcionales para facilitar la recuperación de palabras claves estos. Este enfoque se explica en detalle en el artículo [1]. En esta investigación, se contó con dos etapas, en el cual la primera hace hincapié en la identificación de un conjunto de indicadores para cada categoría Requisitos No Funcionales, en la que se estableció requisitos previamente preclasificados y de esta manera poder calcular la ponderación de los indicadores. Se identificaron el reconocimiento de términos "autentica" y "acceder", que frecuentemente aparecen en la clasificación de requisitos de seguridad [1]. Para el segundo paso, calcularon la probabilidad de que un Requisito No funcional pertenezca a un tipo, en funciones de la aparición de los términos indicadores, en el que según esta investigación [1], recibirían puntuaciones de clasificación por encima de cierto tope. En la cual, tuvieron en cuenta las expresiones para el valor de las probabilidades (ver Fig-2,3).

$$Pr_Q(t) = \frac{1}{N_Q} \sum_{d_Q \in S_Q} \frac{\text{freq}(d_Q, t)}{|d_Q|} \cdot \frac{N_Q(t)}{N(t)} \cdot \frac{NP_Q(t)}{NP_Q}$$

Fig. 2 Expresión de probabilidad para la disminución de indicadores.
Fuente: [1,7].

$$Pr_Q(R) = \frac{\sum_{t \in R \cap I_Q} Pr_Q(t)}{\sum_{t \in I_Q} Pr_Q(t)}$$

Fig. 3 Expresión de probabilidad para la asignación de mayor puntuación a un Requisitos No Funcional.
Fuente: [1,7].

En la figura (2), el primer factor representa el termino de frecuencia del estándar de recuperación de información y muestra que la puntuación de ponderada en la que aumenta si el termino T aparece con frecuencia media de términos en los documentos de Requisito No funcional de tipo Q y el componente restante de la expresión, mide la frecuencia inversa del documento y penaliza la puntuación de peso si el termino ocurre en varios tipos de calidad [1,8]. Por otra parte, en la figura (3), los términos indicadores de tipo Q contenido en R, y el denominador es la suma de las ponderaciones de todos los términos indicadores de tipo Q.
De esta manera, se logró dividir en Requisitos Funcionales y los Requisitos No Funcionales, mediante las sumas de sus pesos de los términos, pudiendo así clasificar los tipos determinados de requisitos.

4. Análisis de factibilidad comparativa.

Se realizaron diversos estudios de profesionales, aplicando métodos para el proceso de recuperación de Requisito No funcionales, para la identificación de palabras claves de seguridad o rendimiento [1]. Se utilizaron métricas del estándar de recuperación, como la de precisión [1,5] y la de especificidad [1,6]. Se midieron la precisión y la especificidad del número total de Requisitos No Funcionales con la clasificación binaria, definida como verdaderos negativos con respecto a todos los negativos [1,5] (ver Fig.4,5).

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}.$$

Fig. 4 Cálculo de clasificación binaria para la precisión.
Fuente: [1].

$$\text{specificity} = \frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}.$$

Fig. 5 Cálculo de clasificación binaria para la especificación.
Fuente: [1].

Los resultados mostraron una recuperación del 70% y una precisión de 31,5%. Para las palabras claves relacionadas con el rendimiento, se obtuvo una recuperación del 43,8% y una precisión del 21,4%. Además, se identificaron obstáculos en la dificultad de encontrar catálogos aceptados y estandarizados para los otros tipos de Requisitos No Funcionales en el conjunto de datos [1]. Otros estudios aplicaron técnicas de clasificación estándar, mediante el algoritmo de árbol de decisión estándar (J48) [13], incluyendo la combinación de selección de subconjuntos de características. Estas técnicas pudieron mejorar el porcentaje de recuperación y precisión entre las técnicas estándar, aunque fueron inferiores a los resultados del clasificador, explicado continuación [1].

Se evaluó el funcionamiento del clasificador con 15 especificaciones de requisitos como proyectos trimestrales de una universidad [1], en la que las especificaciones contenían un total de 326 Requisitos No funcionales y 358 Requisitos Funcionales, en la que eran del tipo legal, mantenibilidad, operativa, rendimiento, apariencia y disponibilidad [1]. Se obtuvo el siguiente esquema de los principales indicadores (ver Tabla 1).

Rank	Availability	Legal	Look and feel	Maintainability	Operational	Performance	Scalability	Security	Usability
1	avail	compli	appear	updat	interfac	second	simultan	onli	us
2	achiev	regul	interfac	mainten	environ	respons	handl	access	easi
3	dai	standard	profession	releas	server	time	year	author	user
4	time	sarban	appeal	new	oper	longer	capabl	user	train
5	hour	oxlei	colour	chang	product	fast	support	inform	product
6	pm	php	look	dure	system	minut	expect	ensur	abl
7	year	pear	simul	promot	databas	take	concurr	data	understand
8	technic	legal	product	product	browser	process	abl	authent	successfulli
9	downtim	law	compli	addit	window	user	number	secur	intuit
10	long	estimat	scheme	everi	web	system	user	system	learn
11	system	regard	logo	budget	comput	let	launch	malici	system
12	product	complianc	sound	develop	applic	maximum	process	prevent	click
13	seven	rule	brand	season	us	complet	next	incorrect	minut
14	defect	requir	feel	integr	internet	flow	product	product	self
15	asid	izognmovi	sea	oper	abl	everi	connect	ar	explanatori

Tabla 1 de indicadores principales.
Fuente: [1].

Se realizaron 15 iteraciones para evaluar la eficacia del clasificador de Requisitos No Funcionales. Además, se aplicaron dos métodos alternativos para seleccionar los términos indicadores:

1. Términos K principales, donde K es un numero positivo. Para los tipos de Requisito No Funcional. Los términos de ponderaciones más altas fueron seleccionados como términos indicadores [1].

2. Todos los términos, donde cada termino con pesos distintos de cero con respecto a un tipo de Requisito No Funcional se seleccionó como termino indicador para ese tipo [1].

Los resultados de diferentes tipos de requisitos mostraron diferencias significativas. Se obtuvo un 98,39%, mostrando solo 1 de los 62 del tipo de Requisito No Funcional de usabilidad estaba mal clasificado o no clasificado. Por otra parte, se obtuvo una especificidad del 69,07%, indicando que el clasificador detecto eficazmente varios tipos de Requisito No Funcionales [1]. Basándose en estos resultados, se realizó un tercer estudio de entrenamiento y reentrenamiento del clasificador, con el objetivo de capacitar en la extracción nuevos términos indicadores que luego se iban a utilizar de nuevo para la clasificación de los Requisito No Funcionales [1]. Los resultados fueron aun mejores, obteniendo así resultados en términos de especificidad entre el 73, 87%, mejorando significativamente los resultados anteriores.

A partir de lo mencionado, se puso en práctica el correspondiente estudio de caso, el cual se basó en la imitación de la investigación antes expuesta, con el objetivo de comprobar la tasa de efectividad real que posee un clasificador de este tipo o similar en la clasificación de Requisitos No Funcionales y, de este modo, comprobar que tales resultados validan que la aplicación de la Inteligencia Artificial en este ámbito es de real importancia para el futuro del desarrollo del software. A continuación, se detallan el tipo de algoritmo utilizado, así como los criterios y métodos implementados para la obtención de los resultados.

Se propuso implementar una simulación a través del lenguaje de programación Python, utilizando sus librerías de algoritmos de Inteligencia Artificial empleados en clasificadores. De esta manera, se buscó simular, mediante un nuevo conjunto de datos, la capacidad de estos algoritmos en detectar correctamente las palabras claves. Dicho de este modo, se utilizó el conjunto de términos relacionado según la clasificación de Sommerville “Seguridad”, en el cual el programa implementado debería clasificar correctamente las palabras “contraseña” y “producto”, separándolas en Requisito Funcional y No Funcional, respectivamente. Por otro lado, se utilizó un conjunto de datos con oraciones cortas relacionada a un sistema de software, en el que se mezclaban Requisitos Funcionales junto con Requisitos No Funcionales. El algoritmo debería ser capaz de detectar solo estos últimos.

Para las pruebas, se utilizó las librerías Pandas, NLTK (Natural Language ToolKit), utilizado para el procesamiento de lenguaje natural, incluyendo tokenizacion, lematización y manejo de stopwords. Por último, la librería scikit-learn, utilizado para el vectorizado de texto (TF-IDF), división de datos en entrenamiento y prueba, junto con la implementación del clasificador Naive Bayes Multinomial (ver Fig.6,7).

```
vectorizador = TfidfVectorizer()

vectores_palabras = vectorizador.fit_transform(palabras)
```

Fig 6 Fragmento de código de la vectorización con TF-IDF.

```
X_train, X_test, y_train, y_test = train_test_split(vectores_palabras, etiquetas, test_size=0.2,
random_state=42)


clasificador = MultinomialNB()

clasificador.fit(X_train, y_train)


predicciones = clasificador.predict(X_test)

print(classification_report(y_test, predicciones))
```

Fig 7 Fragmento de código de la utilización de train_test_split

El algoritmo Naive Bayes Multinomial utilizado principalmente para datos discretos, como el conteo de palabras en la clasificación de texto. Proporciona una forma de calcular la probabilidad posterior P(y|X) a partir de la probabilidad previa P(y) la verosimilitud P(X|y) y la probabilidad

de la evidencia P(X). Naive Bayes asume que las características son independientes entre sí dado el valor de la variable de clase, lo que simplificaría el cálculo de la probabilidad conjunta:

$$P(X|y) = P(x1|y) \cdot P(x2|y) \cdot \ldots \cdot P(xn|y)$$

En la figura (6), se utilizó "TfidfVectorizer" de scikit-learn, en la que convierten las palabras preprocesadas en vectores numéricos que pueden ser entendidos por el modelo de aprendizaje automático. Este proceso asigna pesos a cada palabra según su importancia en el documento y en el corpus general, similar a la investigación antes mencionada y descrita en el artículo [1].
En la figura (7), se dividió el conjunto de datos en entrenamiento y prueba con "train_test_split", utilizando el clasificador de Naive Bayes Multinomial para entrenar el modelo con los datos de entrenamiento, en la que se evaluó el modelo utilizando métricas como precisión, recall y F1 con "classsification_report".
A continuación, se expone los resultados de la primera prueba realizada en la que el algoritmo debería clasificar correctamente las palabras "contraseña" y "producto" (ver Tabla.2,3).

	No_Seguridad	Seguridad
1	venta	contraseña
2	marketing	privacidad
3	producto	cifrado
4	sitio web	usuario
5	publicidad	virus
6	finanzas	spam
7	comercio electrónico	dato
8	redes sociales	Vpn
9	motor de Búsquedas	malware
10	noticias	spyware

Tabla 2 de especificación de palabras claves utilizada.

Predicción para "**contraseña**":	seguridad
Predicción para "**producto**":	no seguridad

Tabla 3 resultado obtenido del clasificador.

En la tabla (2), se simplifico la cantidad de palabras, ya que para esta prueba se tomó un total de 360 palabras claves dividida en Requisitos Funcionales Y Requisitos No Funcionales como se observa en la tabla (2). Se concluye que para esta primera prueba el clasificador pudo identificar

y clasificar correctamente las palabras mostrada en la tabla (3), teniendo así una efectividad del 100% en términos de precisión en un conjunto de datos 360 palabras claves.
Para la segunda prueba realizada el clasificador debería de poder clasificar todo el conjunto de datos completo, al igual que la tabla anterior se simplifico y seleccionó las palabras más importantes. Los resultados se muestran a continuación (ver Tabla.4,5).

TIPO	REQUISITO
Funcional	El sistema debe permitir al usuario iniciar sesión con su nombre de usuario y contraseña.
Funcional	El usuario debe poder recuperar su contraseña mediante su correo electrónico.
No_Funcional	El sistema debe responder en menos de 2 segundos.
Funcional	El sistema debe enviar notificaciones por correo electrónico al usuario.
No_Funcional	El sistema debe estar disponible el 99.9% del tiempo.
No_Funcional	El sistema debe ser escalable para soportar hasta 10,000 usuarios concurrentes.
Funcional	El sistema debe permitir la exportación de datos en formato CSV.

Tabla 4 Especificación de oraciones de Requisitos utilizada.

Requisitos no Funcionales	
Requisito no Funcional:	El sistema debe responder en menos de 2 segundos.
Requisito no Funcional:	El sistema debe estar disponible el 99.9% del tiempo.
Requisito no Funcional:	El sistema debe acceder al menú en menos de 1 segundos.
Requisito no Funcional:	El sistema debe generar pdf en menos de 4 segundos.
Requisito no Funcional:	El sistema debe permitir la exportación de datos en formato CSV.
Requisito no Funcional:	El sistema debe ser escalable para soportar hasta 10,000 usuarios concurrentes.
Requisito no Funcional:	El sistema debe realizar copias de seguridad automáticas cada 24 horas.

Tabla 5 Resultado de la especificación de oraciones de Requisitos utilizada.

En la tabla (4), se seleccionó las oraciones correspondientes a un sistema de software mezcladas en Requisitos Funcionales y Requisitos No Funcionales, en que el clasificador debería seleccionar y clasificar aquellas oraciones que pertenezcan a los Requisitos No Funcionales.
En la tabla (5), se observa los resultados que se obtuvo del clasificador en la clasificación de Requisitos No Funcionales. Se observa que pudo reconocer y clasificar con alta tasa efectividad, es decir, un 94% en términos generales Solo no pudo reconocer la oración “El sistema debe permitir la exportación de datos en formato CSV”, en la fila 5 de la tabla (5), ya que esta oración estaba clasificada como Requisito Funcional, como se observa en la tabla (4). Se concluye la comparación de los resultados obtenidos por los investigadores con los resultados del caso de estudio descripto, se pone en evidencia una notable mejora en la efectividad del clasificador para identificar y clasificar Requisitos No Funcionales. Los investigadores lograron una especificidad del 69,07%, que mejoro al 73,87% y con respecto al estudio practico se alcanzó un 94% en la clasificación. Ambos estudios demuestran que cada vez se hace más evidente el potencial de la Inteligencia Artificial aplicada a la clasificación de Requisitos en el ámbito del desarrollo de software.

5. Conclusiones y Trabajos Futuros

En este marco teórico se ha demostrado que la aplicación de la Inteligencia Artificial para la especificación, validación y recuperación de los Requisitos No Funcionales posee una alta efectividad.

El uso de métricas de recuperación y especificación, permitió una evaluación del desempeño aplicados, obteniendo así una taza de recuperación máxima de 94% y en especificación y clasificación de un 98,41%. Se pudo mejorar los resultados obtenidos, alcanzando resultados aun superiores de los obtenidos de los algoritmos de formato estándar. De esta manera, se confirma que la robustez y adaptabilidad de los métodos de Inteligencia Artificial en la especificación y recuperación de los Requisitos No Funcionales aumentan la probabilidad de una mayor integración de algoritmos de aprendizaje automático más avanzados en un corto periodo de tiempo. Con la implementación como redes neuronales profundas y modelos de aprendizaje no supervisados, en la que podrán ser capaces de identificar patrones y relaciones complejas aplicadas en la obtención de Requisitos Funcionales y Requisitos No Funcionales de manera más eficiente, permitiendo así la obtención, especificación y recuperación aún más precisa y automatizadas.

Según lo expuesto anteriormente, se espera que futuros trabajos e investigaciones profundicen y mejoren algoritmos y métodos matemáticos orientados en la inteligencia artificial para la obtención de requisitos.

Existen varias pruebas y técnicas más potentes en la que se podrían ser replicados con experimentos similares, para evaluar y mejorar la efectividad de un modelo de clasificación como el expuesto. Por un lado, se tiene la validación cruzada (Cross-Validation), para evaluar el rendimiento de in modelo al dividir los datos en múltiples subconjuntos (folds) y entrenar el modelo en diferente combinación de estos subconjuntos. La validación cruzada k-fold divide los datos en k partes y el modelo se entrena y evalúa k veces. Por otro, se tiene Grid Search combinado con (Cross-Validation), en la que se utilizaría para mejorar los hiperparametros para un modelo. Esta técnica busca a través de un conjunto especificado, la validación cruzada para el rendimiento de cada combinación de un conjunto de datos.

El objetivo es continuar optimizando los procesos de entrenamiento y reentrenamiento de clasificadores, explorando técnicas y métodos de aprendizaje continuo o no supervisados.

A partir de ello, se podrá mejorar la taza de efectividad cada vez más cerca al total de conjuntos y subconjuntos de requisitos de un sistema completo, contribuyendo así al correcto desarrollo del software.

Referencias

1. Cleland-Huang, Jane & Settimi, Raffaella & Zou, Xuchang & Solc, Peter. (2007). Automated classification of non-functional requirements. Requir. Eng. 12. 103-120. 10.1007/s00766-007-0045-1.
2. Hurtado-Cortés, Luini Leonardo, Villarreal-López, Edwin, Villarreal-López Luís. Detección y diagnóstico de fallas mediante técnicas de inteligencia arti
3. ficial, un estado del arte. Dyna [en linea]. 2016, 83(199), 19-28[fecha de Consulta 30 de mayo de 2024]. ISSN: 0012-7353. Disponible en: https://www.redalyc.org/articulo.oa?id=49648868002.
4. Kellens A, Mens K (2005) A survey of aspect mining tools and techniques. INGI Technical Report, 2005–08, UCL, Belgium, Deliverable 6.2a for the workpackage 6 of the IWT project 040116 ‘‘AspectLab’’
5. Rosenhainer L (2004) Identifying crosscutting concerns in requirements specifications. In: Workshop on early aspects: aspect-oriented requirements engineering and architecture design, Vancouver, Canada, 2004. Available at http://www.trese.cs.utwente.nl/Docs/workshops/oopsla-early-aspects-2004/.

6. Frakes WB, Baeza-Yates R (1992) Information retrieval: data structures and algorithms. Prentice-Hall, Englewood Cliffs.
7. Altman DG, Bland JM (1994) Statistics notes: diagnostic tests 1: sensitivity and specificity. Br Med J 308(1552).
8. Salton G, McGill MJ (1983) Introduction to modern information retrieval. McGraw-Hill, New York.
9. Khlood Ahmad, Mohamed Abdelrazek, Chetan Arora, Muneera Bano, John Grundy,Requirements engineering for artificial intelligence systems: A systematic mapping study, Information and Software Technology,Volume 158.
10. H. Kuwajima, H. Yasuoka, T. Nakae, Engineering problems in machine learning systems, Machine Learning (2020) 1–24.
11. Bohem, B.W.: Software Engineering Economics. IEEE transaction on software engineering, 4-21. (1984).
12. Hussain,I., Ormandjieva,O., Kosseim,L.: Automatic Quality Assessment of SRS Text by Means of a Decision-Tree- Based Text Classifier. Seventh International Conference on Quality Software(Qsic),IEEE (2007).doi: 10.1109/QSIC.2007.4385497
13. Mulla,N., Girase,S.:A New Approach to Requirement Elicitation Based On Stakeholder Recommandation and Collaborative Filtering. International Journal of Software Engineering and Application.(2012).doi:10.5121/ijsea.2012.3305