Intelligent Monitoring of Software Test Automation of Web Sites

Maria Talakh [1], Serhii Holub [2] and Yurii Lazarenko [3]

[1] *Yuriy Fedkovych Chernivtsi National University, Rivnenska str., 16, Chernivtsi, 58000, Ukraine*
[2] *Cherkasy State Technological University, Shevchenko Boulevard 460, Cherkasy, 18006, Ukraine*
[3] *Test automation engineer Softserve, Nebesnoyi Sotni st. 16a, Chernivtsi, 58018, Ukraine*

Abstract

The active use of test automation approaches in software development causes further automation of this process, in particular, in the field of results analysis. This paper presents the results of applying the methodology for creating information technologies of multi-level intelligent monitoring to provide data for decision-making processes by a testing engineer. Methods of text mining and machine learning are combined to build a methodology for creating intelligent multi-level monitoring systems. On the example of a set of reports, which were divided by an expert into 4 classes, the processes of coordination of interactions of different types of methods for forming an array of informative features, typical aggregates for the synthesis of classifier models at each stage of monitoring were investigated. The task of classifying the results of text research was solved to obtain the reason for the failure of the tests. A set of n-grams for each class of tests formed an array of input data for the synthesizer of models of the monitoring information system. The classification of these data was provided using an ensemble of models, where the resulting value was obtained combining predictions by meta-learning using stacking. The effectiveness of the described methodology has been experimentally proven.

Keywords

Intellectual monitoring, test automation, ensemble training.

1. Introduction

Intelligent monitoring of automatic software testing processes provides identification of the causes of errors to speed up their processing by the tester. Based on the testing results, the monitoring information system (MIS) [1] solves the problems of classifying errors and their causes. This paper describes the process of using intelligent monitoring information technology to support decision-making by a QA engineer on the choice of methods and means of eliminating errors. The work of the method is described using the example of developing a website.

The introduction of test automation into the software development process is now a must, the number of available test results is increasing significantly.

Jobs, or agents who have the role of a QA engineer and perform tests, can work around the clock, seven days a week, and in addition, the number of test cases accumulates during each sprint. Thus, more results are obtained for management and analysis. A significant amount of data requires approaches to their processing [2].

If the time spent examining the test results exceeds the time saved by running automated tests, then the very meaning of its implementation is lost. To reap the benefits of automation, it is important to know how to properly handle the growing number of test results. Continuous Delivery requires constant testing that slows down the pace of work. In the

ISIT 2021: II International Scientific and Practical Conference «Intellectual Systems and Information Technologies», September 13–19, 2021, Odesa, Ukraine
EMAIL: m.talah@chnu.edu.ua (A.1); s.holub@chdtu.edu.ua (A.2); lazarenko199705@gmail.com (A. 3)
ORCID: 0000-0002-5067-6848 (A. 1); 0000-0002-5523-6120 (A. 2); 0000-0002-0592-2202 (A. 3)

fast pace of product development and testing, optimizing (both failed and successful) test cases is critical [3].

At the same time, there are not so many applications for tracking the history of automated tests, they are limited in functionality. And standard reports are always hard to read since the appearance is not informative enough. The most common test frameworks are NUnit and TestNG, the latter of which was used in [4].

Machine learning of classifier models is based on the analysis of previous runs of auto-tests. The main stages are [1]:

1) coordination of the form and content of information about the properties of objects of observation;

2) determination of the list of diagnostic signs;

3) the formation of an array of input data (MVD);

4) construction and training of models by the MIS synthesizer;

5) testing and using models to identify the causes of errors. The set of the obtained classifier models is entered into the model knowledge base, forming its hierarchical structure.

The task of ensuring the information content of the Ministry of Internal Affairs, sufficient for constructing useful models by the available methods and means, implemented in the synthesizer of the MIS models, is always relevant. The effective organization of observation of research objects provides for the attraction of the latest scientific achievements in the subject area, in which the monitoring technology is implemented [1].

Now there are not many implemented ready-made products for classifying the results of automated testing. Most of them are test developments and student projects [5].

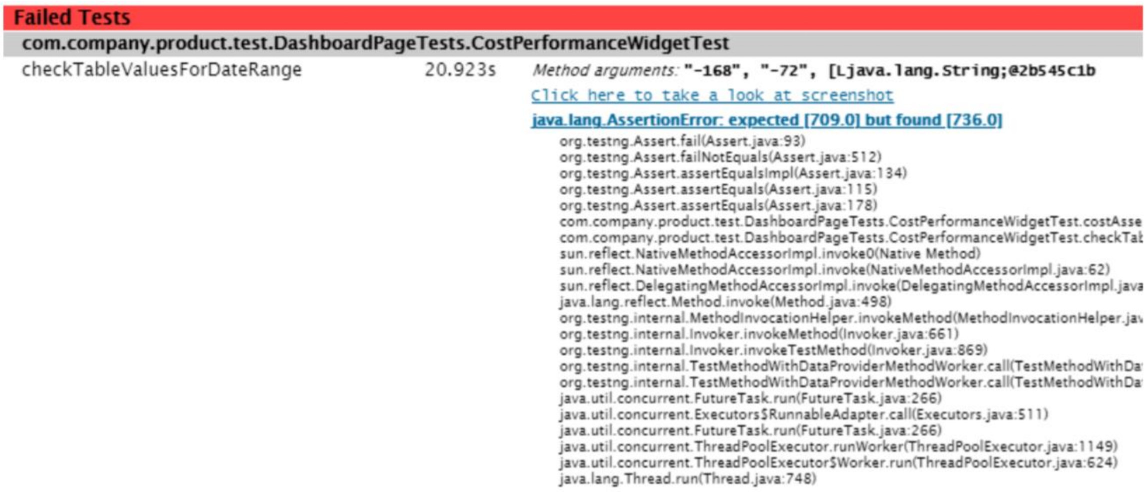
Thus, the use of machine learning technology of models through MIS for analyzing the results of Automation tests is relevant and allows you to increase the efficiency of QA engineers, allows you to increase the coverage of web applications with new tests, and improves the quality of applications. It also reduces the time it takes to make decisions to improve on failed tests.

2. Results

A hypothesis was formulated that the list of classification features for identifying the causes of errors in programs should be obtained by using Text Mining methods [6-8]. This work examines the processes of forming a list of features and forming an array of input data by combining expert methods, Text Mining methods, and ensemble classification methods. For this, the results of the classification of samples were studied after the formation of an array of input data based on the features obtained by using text mining reports of the fall of automatic tests. The results were evaluated by the number of correctly classified samples.

Thus, at the first stage of building a monitoring information system, it is necessary to determine the format of the input and output data.

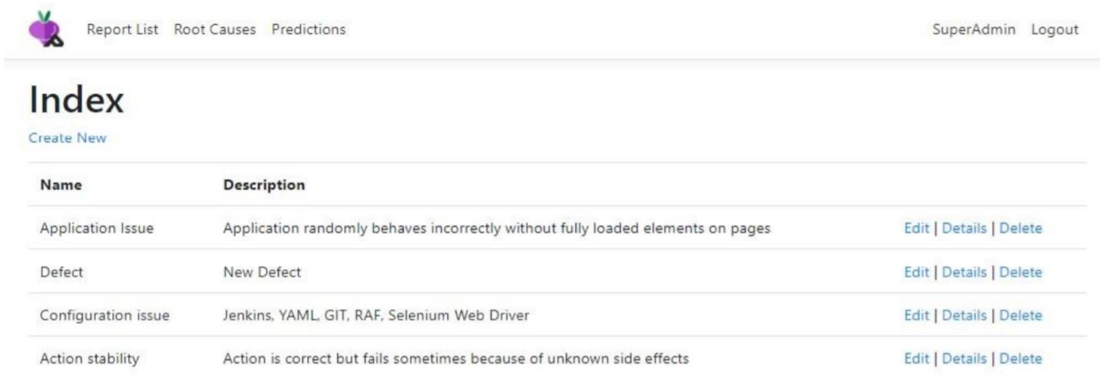
Input data for the system - are error messages, and their example was shown on Figure 1.



```
Failed Tests
com.company.product.test.DashboardPageTests.CostPerformanceWidgetTest
checkTableValuesForDateRange          20.923s   Method arguments: "-168", "-72", [Ljava.lang.String;@2b545c1b
                                                Click here to take a look at screenshot
                                                java.lang.AssertionError: expected [709.0] but found [736.0]
                                                    org.testng.Assert.fail(Assert.java:93)
                                                    org.testng.Assert.failNotEquals(Assert.java:512)
                                                    org.testng.Assert.assertEqualsImpl(Assert.java:134)
                                                    org.testng.Assert.assertEquals(Assert.java:115)
                                                    org.testng.Assert.assertEquals(Assert.java:178)
                                                    com.company.product.test.DashboardPageTests.CostPerformanceWidgetTest.costAsse
                                                    com.company.product.test.DashboardPageTests.CostPerformanceWidgetTest.checkTal
                                                    sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
                                                    sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
                                                    sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java
                                                    java.lang.reflect.Method.invoke(Method.java:498)
                                                    org.testng.internal.MethodInvocationHelper.invokeMethod(MethodInvocationHelper.jav
                                                    org.testng.internal.Invoker.invokeMethod(Invoker.java:661)
                                                    org.testng.internal.Invoker.invokeTestMethod(Invoker.java:869)
                                                    org.testng.internal.TestMethodWithDataProviderMethodWorker.call(TestMethodWithDa
                                                    org.testng.internal.TestMethodWithDataProviderMethodWorker.call(TestMethodWithDa
                                                    java.util.concurrent.FutureTask.run(FutureTask.java:266)
                                                    java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
                                                    java.util.concurrent.FutureTask.run(FutureTask.java:266)
                                                    java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
                                                    java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
                                                    java.lang.Thread.run(Thread.java:748)
```

Figure 1: Example of input data. Error message

The initial data will be a table with the results of automatic testing, the application of machine learning algorithms, and detailed analysis will be carried out directly by the testing engineer. They establish the reason for the fall of the test. That is, the input data was analyzed by an expert, who allocated 4 classes of system responses to the results of automation testing. However, if necessary, this list can be expanded, that was shown on Figure 2.



Index

Create New

Name	Description	
Application Issue	Application randomly behaves incorrectly without fully loaded elements on pages	Edit | Details | Delete
Defect	New Defect	Edit | Details | Delete
Configuration issue	Jenkins, YAML, GIT, RAF, Selenium Web Driver	Edit | Details | Delete
Action stability	Action is correct but fails sometimes because of unknown side effects	Edit | Details | Delete

Figure 2: Page view with the root causes of test failures, added by the administrator

For the analysis of the reports, the text Mining approaches were used. Tokenization was carried out (with the selection of the optimal length and mixture of n-grams. In addition, from the preprocessing approaches, the text cleaning was used. The main purpose was to clean the text from specific characters (primarily specific punctuation and symbols). Also, the most frequently and rarely used words were removed, do not allow define the specifics of the text and classify it.

Algorithms for the synthesis of classifier models were constructed using artificial intelligence and machine learning [9-11].

In particular, the CountVectorizer provided by the scikit-learn library [12] was used to vectorize sentences. The method takes the words of each sentence and builds the vocabulary of all unique words in the sentences. This vocabulary can then be used to create a vector of word count features. Now, from each sentence, we can get occurrences of words (or n-grams) in sentences based on the previous dictionary. The dictionary consists of all words with Stake trace, each of which represents one word or phrase in the vocabulary. If we take the previous sentences and implement the CountVectorizer, we will get a vector representing the number of occurrences of each word in the sentences.

After that, you can see the resulting function vectors for each sentence based on the previous dictionary, that is, Bag-of-words. Each document is represented as an n-gram vector. You can then use these vectors as function vectors for the machine learning model.

Table 1
An example of the type of input

Body of the test	n-grams	Root cause
<i>Method arguments: </i><span class="arguments">"-229", "-57", [Ljava.lang.String;@3358cf5e</span> <div class="testOutput"> <a href="checkTableValuesForDateRange.png"> Click here to take a look at screenshot </a> </div> … java.lang.Thread.run(Thread.java:748) </div>	'test dashboardpagetests' 'invoke nativemethodaccessorimpl' 'costperformancewidgettest sun' 'dashboardpagetests costperformancewidgettest'	Defect
…	…	…

Thus, the list of informative signs contains a mix of n-gram and the reason of the route cause, determined by the expert. The resulting value for the entire test case was determined based on the Bayes' formula:

$$P\left(\frac{H_i}{A}\right) = \frac{P(H_i)P\left(\frac{A}{H_i}\right)}{\sum_{k=1}^{n} P(H_k)P\left(\frac{A}{H_k}\right)}, \qquad (1)$$

where A and B are events.

- p (A) and p (B) are the probabilities of A and B without relation to each other;
- p (A / B), the conditional probability, is the probability of A if B. is true.
- p (B / A) is the probability of observing event B, provided that A.

The general architecture of the ensemble is shown on Figure 3.

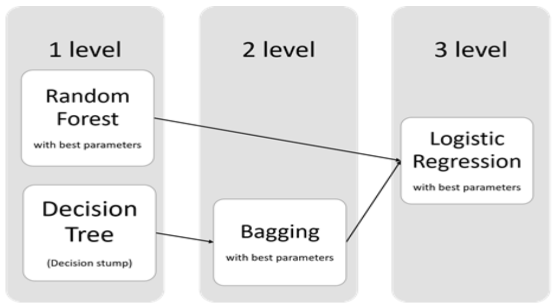


Figure 3: The architecture of the ensemble classification model

Thus, at the first level of the MIS, a random forest classifier with optimal parameters is created by creating samples based on the bootstrap from the verified elements. Another first-level model was decision tree "stumps", which were used as a baseline model for running with optimal parameters. The generalization of the results obtained at the previous levels was due to the logistic regression model using the Softmax function.

The idea behind the MIS is to automate the process of analyzing test reports by an engineer and provide support for making the final decision. To do this, you will need to use two roles: the admin side and the engineer side. The administrator has access to all points and the ability to edit root causes, create and train the model. The engineer has access to all reports, can view and analyze tests. All the functionality that is on the Report list page, open to the engineer. The functional diagram of the algorithm is shown on Figure 4.

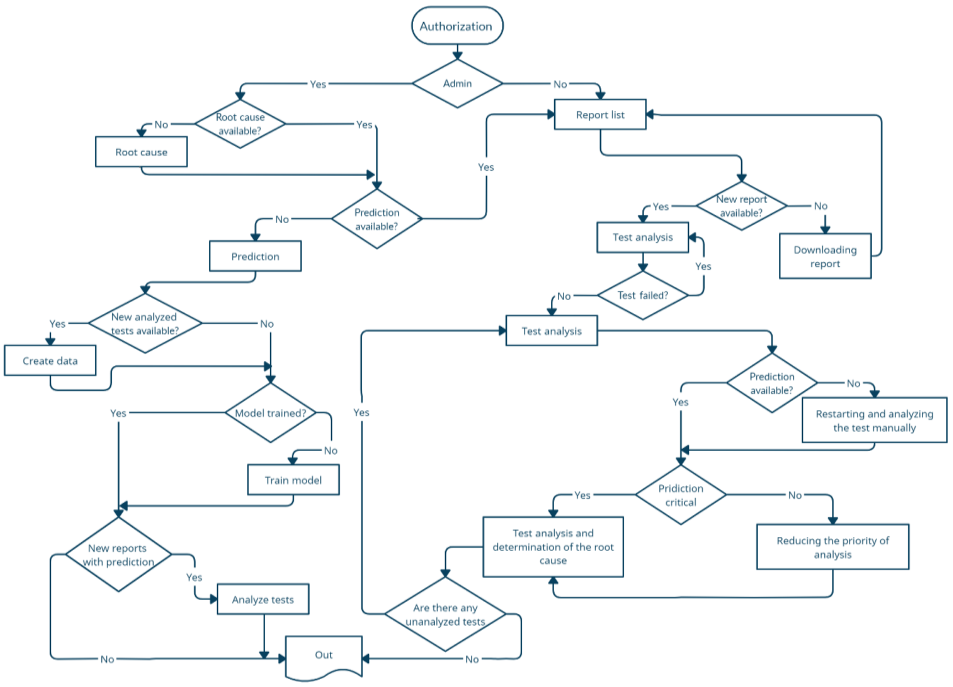
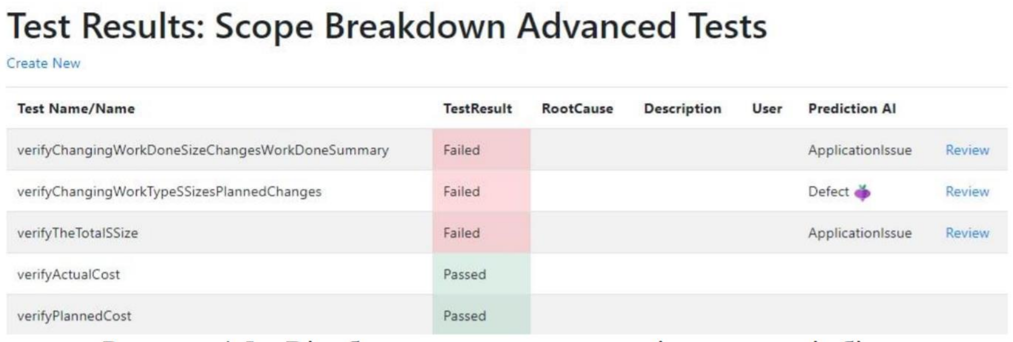


Figure 4: Functional diagram of the automated test analysis algorithm

So, the system, after analyzing the results of automatic testing, displays predictions for unanalyzed tests in a special column, inside the test assembly. The engineer, in turn, when analyzing, will pay attention to those tests that will be with the most likely critical root causes (Defect, Framework issue, etc.). The final decision rests with the engineer.

Each test has its name and status, which are displayed on Figure 5. If the test has not yet been analyzed using the classification algorithm, the "Prediction AI" column will be empty. To view more detailed information about the test, you must click on it.

For unanalyzed tests, predictions will be displayed in a special column. When analyzing, an engineer will need to first of all pay attention to those tests that will be with the most likely critical root causes (Defect, Framework issue, etc.). The final decision rests with the engineer.



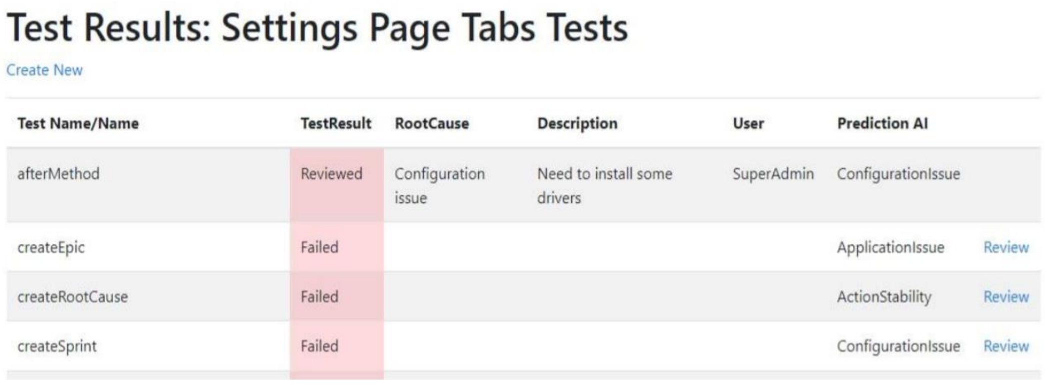
Test Results: Scope Breakdown Advanced Tests

Create New

Test Name/Name	TestResult	RootCause	Description	User	Prediction AI	
verifyChangingWorkDoneSizeChangesWorkDoneSummary	Failed				ApplicationIssue	Review
verifyChangingWorkTypeSSizesPlannedChanges	Failed				Defect	Review
verifyTheTotalSSize	Failed				ApplicationIssue	Review
verifyActualCost	Passed					
verifyPlannedCost	Passed					

Figure 5: View of tests with the results of auto-analysis based on the proposed algorithm

Thus, the engineer pays attention, first, to those tests that have critical root causes and, in more detail, examines the causes of their occurrence and enters the results into the database. These detailed reports will also be the basis for training the model on subsequent launches. The results of auto-analysis and engineer's analysis can be seen as a result in the table with tests. An example of these results is on Figure 6.



Test Results: Settings Page Tabs Tests

Create New

Test Name/Name	TestResult	RootCause	Description	User	Prediction AI	
afterMethod	Reviewed	Configuration issue	Need to install some drivers	SuperAdmin	ConfigurationIssue	
createEpic	Failed				ApplicationIssue	Review
createRootCause	Failed				ActionStability	Review
createSprint	Failed				ConfigurationIssue	Review

Figure 6: The final version of the analysis of one of the tests.

The developed system was tested based on reports of automatic testing of the website development process. The model test results are demonstrated using an error matrix and are shown in Figure 7. It is worth noting that the result c6an be influenced by different class sizes within the studied array of input data.

From the 62 studied reports, the reasons were correctly identified in 55, that is, the number of correctly identified causes of errors was 89%.

3. Conclusions

The use of MIS to provide information to decision-making processes in the field of automation testing allows you to successfully solve the problems of classifying the analysis of the input standard reports of automated tests based on the data of the expert assessment of automation engineers.

The developed system can have a wide practical use for solving problems in the process of developing and testing software products. To quickly identify critical root causes of test failures, chronological analysis and help the engineer make decisions. The introduction of this system can lead to an increase in the ability of engineers to cover web applications with new tests and will lead to a higher quality of applications. It will also reduce the time it takes to make decisions about failed tests. The obtained accuracy allows us to assert that the proposed approach for the formation of the Ministry of Internal Affairs makes it possible to form it sufficiently informative due to its application in conjunction with the proposed architecture of the ensemble.

The effectiveness of using the proposed architecture of the ensemble classifier for the synthesis of models-classifiers based on an array has been experimentally confirmed.

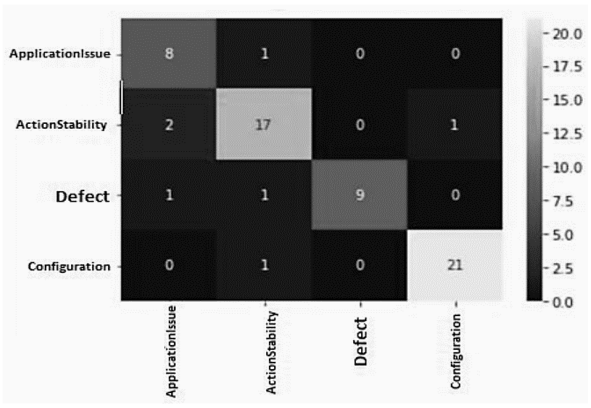


Figure 7: Confusion matrix of the classification model of automation test reports

4. References

[1] S.V. Golub Bagatorivneve modelyuvannya v tehnologiyah monitoryngu otochuyuchogo seredovyshha (Multilevel modeling in environmental monitoring technologies), Cherkasy: Vyd. ChNU imeni Bohdan Khmelnytsky, 2007, 220 p.

[2] T. Cser Why test automation needs machine learning, 2020/ URL: https://www.functionize.com/gated/eBook-Why-test-automation-needs-ML.pdf

[3] Rajesh Mathur, Scott Miles, Miao Du Adaptive Automation: Leveraging Machine Learning to Support Uninterrupted Automated Testing of Software Applications, arXiv:1508.00671v1 [cs.SE] 4 Aug 2015 URL:https://arxiv.org/pdf/1508.00671.pdf

[4] A. Trudova, M. Dolezel A. Buchalcevova Artificial Intelligence in Software Test Automation: A Systematic Literature Review, Conference: 15th International Conference on Evaluation of Novel Approaches to Software Engineering, Portugal 2020, pp. 181-192, DOI:10.5220/0009417801810192.

[5] H. Tran, M. Chechik Test Generation using Model Checking, University of Toronto, Collection of Reports from CSC2108 Automated Verification, 2018, pp. 13. URL:http://www.cs.toronto.edu/~chechik/courses00/csc2108/projects/4.pdf

[6] R. Talib M. Kashif, A. Shaeela, F. Fakeeha Text Mining: Techniques, Applications and Issues, International Journal of Advanced Computer Science and Applications 7(11), November, 2016, pp. 414-418. DOI:10.14569/IJACSA.2016.071153 URL: https://www.researchgate.net/publication/311394659_Text_Mining_Techniques_Applications_and_Issues

[7] A. M. Jadhav, D. P. Gadekar A survey on text mining and its techniques, Survey Paper, Computer Science & Engineering, India, 3(11), 2014, pp. 2110 – 2113. URL:https://www.semanticscholar.org/paper/A-Survey-on-Text-Mining-and-Its-Techniques-Jadhav-Gadekar/b46c04ad14d021ad444a71103d5c87297df3b976

[8] D. Antons, E. Grünwald, P. Cichy, T. O. Salge The application of text mining methods in innovation research: current state, evolution patterns, and development priorities, R&D management 50 (2020), pp. 329-351. doi: 10.1111/radm.12408

[9] S.V. Golub, A.S. Avramenko Udoskonalennia klasyfikatoriv v monitorynhovykh intelektualnykh systemakh (Improvement of classifiers in monitoring intelligent systems), Bulletin of the Academy of Engineering Sciences of Ukraine, 1 (2019), pp. 67-71.

[10] A. Avramenko, S. Holub Classification models in information systems for social and environmental crisis monitoring. In: Inzynier XXI wieku. Monografia: Wydawnictwo naukowe Akademii Techniczno-Humanistycznej w Bielsku-Bialej, 2016, pp. 43-46.

[11] A.S. Avramenko, S.V. Golub Decrease of time of model synthesis in intellectual monitoring systems. Mathematical machines and systems, 3 (2019), pp. 129-134. DOI: 10.34121/1028-9763-2019-3-129-134.

[12] Library scikit-learn. Machine Learning in Python, URL: https://scikit-learn.org/stable.