Research Article
Journal of Intelligent Systems: Theory and Applications 4(1) (2021) 37-42
DOI: 10.38016/jista.755419

Classifier Selection in Resource Limited Hardware: Decision Analysis and Resolution Approach

Atilla Özgür [1]

[1]Jacobs University Logistics and Mathematics, Bremen/Germany

a.oezguer@jacobs-university.de

Abstract
Digitalization, Industry 4.0 and Internet of things (IoT) have become more popular in the recent years. Most of these systems depend on micro-controllers and sensors. These micro-controllers and sensors are mostly cheap, low RAM and low CPU systems; thus, they are resource constrained environments. In this study, a supervised learning classifier comparison technique suitable for resource constrained environments is proposed. This technique, Decision Analysis and Resolution (DAR), is originated in the domain of Software Engineering. First, DAR is explained using an example of car buying scenario. Then 11 off-the-shelf classifiers are compared using DAR for low RAM and less powerful CPU environments in an intrusion detection scenario. This scenario simulated on well-known KDD99 intrusion detection dataset. All the experiments are realized using python scikit-learn package. According to the experiments, Decision Tree classifier is the most suitable to implement for resource constrained environments with a small lead. Results for the other three classifiers (Bagging, Multi Layer Perceptron, Random Forest) are also very similar. To aid the reproducibility of the experiments, the whole source code of the study is provided in the popular open source repository https://github.com/ati-ozgur/classifier-comparison-using-DAR.

Keywords: Classifier selection, Decision analysis and resolution, Machine learning, Performance metrics, Resource limited environment

Kısıtlı Kaynaklı Donanımlarda Sınıflandırıcı Seçimi: Karar Analizi ve Çözüm Yaklaşımı

Öz
Dijitalleşme, Endüstri 4.0 ve Nesnelerin İnterneti (IoT) son yıllarda daha popüler hale gelmiştir. Bu sistemlerin çoğu mikro denetleyicilere ve sensörlere bağlıdır. Bu mikro denetleyiciler ve sensörler çoğunlukla ucuz, düşük RAM ve düşük CPU sistemleridir; bu nedenle, kaynak kısıtlı ortamlardır. Bu çalışmada, kaynak kısıtlı ortamlara uygun, denetimli bir öğrenme sınıflandırıcı karşılaştırma tekniği önerilmiştir. Bu teknik, Karar Analizi ve Çözümü (DAR), Yazılım Mühendisliği alanında ortaya çıkmıştır. İlk olarak DAR, örnek bir araba satın alma senaryosu ile açıklanmıştır. Ardından, 11 hazır sınıflandırıcı, bir saldırı tespit senaryosunda düşük RAM ve düşük CPU ortamları için DAR kullanılarak karşılaştırılmıştır. Bu senaryo, iyi bilinen KDD99 saldırı tespit veri setinde gerçekleştirilmiştir. Tüm deneyler python scikit-learn paketi kullanılarak gerçekleştirilmiştir. Deneylere göre, Karar Ağacı sınıflandırıcısı, diğer sınıflandırıcılara göre küçük bir fark ile kaynak kısıtlı ortamlara uygulanmak için en uygun sınıflandırıcıdır. Diğer üç sınıflandırıcı (Boosting, Çok Katmanlı Algılayıcı, Rastgele Orman) sonuçları da çok benzerdir. Deneylerin tekrarlanabilirliğine yardımcı olmak için, tüm kaynak kod popüler açık kaynak kod deposu https://github.com/ati-ozgur/classifier-comparison-using-DAR'da verilmiştir.

Anahtar Kelimeler: Kaynak kısıtlı ortam sınıflandırıcı seçimi, Karar analizi ve çözümü, Makine öğrenmesi, Performans metrikleri

* Corresponding Author.
E-mail: a.oezguer@jacobs-university.de

Recieved	: 20 June 2020
Revision	: 28 January 2021
Accepted	: 08 February 2021

1. Introduction

Industry 4.0 is first termed by Germany Government to increase digitalization and automation in manufacturing sector (Lasi et al. 2014, Uygun & Ilie 2018). Internet of things (IoT), sensors and mobile systems are very important topics in Industry 4.0 and digitalization (Ileri & Furat 2020). These systems are depended on computers. Even though some computers they work on are powerful, most of the computers they work on are resource constrained environments such as micro-controllers and sensors (Karahan & Hökelek 2020). This can be seen from increasing market size of micro controllers (Research 2019), which was 18.6 billion USD in 2018. Most of the market still belongs to 8- and 16-bit micro-controllers (Research 2019). Most prevalent reasons for this phenomenon are cost and size. Resource-limited hardware products, for example 8-16-bit micro-controllers, are cheaper and smaller. Algorithms and systems that work on in these resource constrained environments are becoming more valuable.

Supervised learning also known as classification is a widely used technique for variety of problems, such as face recognition (Yavuz et al. 2016), bioinformatics (Yılmaz 2020), medical (Saleh & Hussein 2019), and intrusion detection (Özgür & Erdem 2012). Widely different classification algorithms are proposed in the literature (Taşcı 2019, Özgür et al. 2018, Özgür & Erdem 2018). Proposed algorithms mostly assume that necessary computing capabilities exist for algorithms to work and rarely address low computing requirements. But computing capabilities of micro-controllers are lower than general-purpose computers. Even though some algorithms are suitable for working with limited resources, how to choose among such algorithms in a limited hardware situation is rarely addressed.

In contrast, choosing between alternative solutions is a widely researched topic in other domains. For example, Basheleishvili et al. (Basheleishvili et al. 2019) proposed fuzzy-logic based model for selection of university staff. Similarly, Çınar and Uygun (Çınar & Özer Uygun 2019) proposed another fuzzy based model for supplier selection. On the other hand, Faydalı and Erkan (Faydalı & Erkan 2020) proposed VIKOR model for selection of factory machines.

This article proposes a decision theoretic approach for a practitioner to make an informed decision between classification algorithms in resource-limited hardware, such as a micro-controller with low RAM and low CPU power. This approach, called Decision Analysis and Resolution (DAR) (Team 2006), is originated in the domain of software engineering.

Remainder of the paper is as follows. Section 2 introduces different classifier comparison metrics and explains why using only accuracy is not a good choice. Section 3 gives basic introduction to DAR with car example. Section 4 introduces a classification example using a micro-controller in the intrusion detection domain with a well-known dataset. Finally, section 6 concludes the paper.

2. Classifier Comparison Metrics

Most of the time, classifier evaluation is made with single metric: accuracy. For example, Özgür and Erdem reviewed 149 articles in intrusion detection domain that was published in SCI-index journals (Özgür & Erdem 2016). Among these reviewed articles, accuracy was used by 130 articles, making the accuracy metric the most used metric. On the other hand, second most used metric, False positive (False alarm) was used only 70 times. But using only accuracy to compare classifiers is simplistic at best and may be plain wrong in some cases.

Netflix (Amatriain & Basilico 2012) did not use their competition winner algorithm that increased their previous accuracy more than 10%. They had considered engineering efforts vs improved accuracy and had chosen another algorithm for their systems. Engineering effort is one example about considering other metrics for classifier evaluation among many others.

According to different domains and purposes, different base metrics may be more important. For example, according to Axelsson's seminal paper (Axelsson 1999), one of the most important performance metric for an intrusion detection system is false positive rate (false alarm rate). Axelsson proposed that all alarms should be investigated, and this investigation is a costly endeavor.

For screening diseases, ideally a classifier should be highly sensitive (high true positive rate) and should miss very few persons with the disease (Wilson & Jungner 1968). Missing a sick person will be more costly for the society in most situations. If disease is a contagious one, missing one could lead to more patients. If disease is expensive to treat in further levels, such as cancer, it is again helpful to detect it very early. For the wrong results (false positive), more qualified doctors or more expensive tests could check the results again; therefore, false positive results will be corrected in the second checks.

For low resource-environments, computer resource related metrics are more important. In such environments, resources like RAM and CPU may become more important parameters than accuracy. Therefore, low model size for low RAM and low training/testing time for better usage of CPU becomes important.

Table 1 shows other performance metrics that are used in different situations. Using different performance metrics may lead to choosing different classifiers. Instead of trying to choose a classifier based on a single metric, incorporating different criteria to decision process would be more reliable. Here, we propose using a decision technique from software engineering domain, Decision Analysis and Resolution (Team 2006) for this purpose.

Table 1. Classifier Performance Metrics

Metric
Detection Rate
False Negative
False Positive
Training Memory
Training Time
Testing Time
F-1 Measure
Model Size in RAM
Model Size in Storage
Others

3. Decision Analysis and Resolution (DAR)

Decision Analysis and Resolution (DAR) technique is originated in software engineering domain (Team 2006). It is a technique to make more informed decisions, for example buying versus building a new tool, subcontracting versus building a new software in house. An example helps to understand this technique better: we start with a common example of buying a car. Suppose that a company needs to buy a car and wants to make an informed decision. There are a lot of metrics which can influence the decision. Some of them are provided below.

- Cost
- Secondhand Worth
- Fuel Consumption
- Baggage Capacity

More metrics can be added to this list but to make the example simpler, count of metrics is intentionally restricted.

A decision between two brands (Brand F and Brand H) should be made. Table 2 shows DAR comparison between these two brands. A DAR process starts with 100 points and distribute these points to metrics, (weight column). In Table 2, Initial Cost is given 40 points; Secondhand Worth is given 20 points; Fuel Usage is given 30 points; and Baggage Capacity is given 10 points for a total of 100 points.

Initial Cost, Secondhand Worth, Fuel Usage and Baggage Capacity could not be compared because to the different range of values of these variables. Initial Cost and Secondhand Worth has values in dollars, while Fuel Usage has liters/100 km and Baggage capacity is cm^2. To be able to use mathematical operations meaningfully (multiplication and addition), we need to normalize these values to same range. Min-max normalization, Equation 1 and Equation 2 could be used for this purpose. Equation 1 is used when higher values are better, and Equation 2 is used when higher values are worse. Since higher values for second hand worth and baggage size are better, these features are normalized using Equation 1. Similarly, since higher values for cost and fuel consumption are worse, these features are normalized using Equation 2.

$$x_{norm} = \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad (1)$$

$$x_{norm} = 1 - \frac{x_i - \min(x)}{\max(x) - \min(x)} \qquad (2)$$

Summarized version of DAR car buying scenario can be seen in Table 2. The Min-Max column values are found using other brands. For example, 10 different cars from similar segments are used to find minimum and maximum values for these features. F and H columns are real values for these cars, while normalized F and normalized H values are min-max normalized (Equation 1 and Equation 2) values. Since we normalize values using Equation 1 and Equation 2), total column shows how good this car it is. Excel version of the Table 2 is provided in the source code repository.

After calculation, Brand H is found to be a better choice according to given weights, even though initial cost of Brand H is higher. Here, Brand H having better values in the metrics of fuel consumption and baggage capacity are deciding factors for this decision, even though its initial cost is more expensive. DAR is a valuable technique to use when making a decision which has more than two criteria.

3.1. How to decide weights

Decision weights change from user to user and domain to domain. Those who can decide these weights called differently according to domain, for example users, evaluators, stakeholders (Shukla & Auriol 2013), decision makers (Faydalı & Erkan 2020) and experts (Çınar & Özer Uygun 2019). Thus, weights chosen by these experts are often very subjective (Phillips & Polen 2002). Even though, weights chosen by single user is subjective, a number methods to find common ground between different users are proposed such as Rank ordinal method (Danielson & Ekenberg 2017) and Utility Rank Order Weighting (UROW) (Shukla & Auriol 2013).

Table 2. Decision Analysis and Resolution - Car Buying Scenario

	Weights	Min	Max	Brand F	Brand H	Normalized F	Normalized H	Weight*Normalized F	Weight*Normalized H
Initial Cost	60	75000	100000	80000	95000	0,6	0,2	24	8
2nd-Hand Worth	20	20000	40000	30000	37000	0,75	0,85	15	17
Fuel Usage	30	4	6	5	4,4	0,5	0,8	15	24
Baggage Capacity	10	350	500	400	500	0,34	1	3,34	10
							Total	**57,34**	**59**

4. Decision Analysis and Resolution (DAR): Intrusion Detection Domain

Intrusion detection is a widely researched subject in literature (Özgür & Erdem 2012, Özgür et al. 2018, Özgür & Erdem 2018, Sahingoz 2019). In our example problem, an intrusion detection classifier should be decided for a resource constrained environment, such as a micro-controller with a low RAM and less powerful CPU environment. An example for this situation is given by Karahan & Kaya 2020. Different metrics can be used for this purpose, see Table 1.

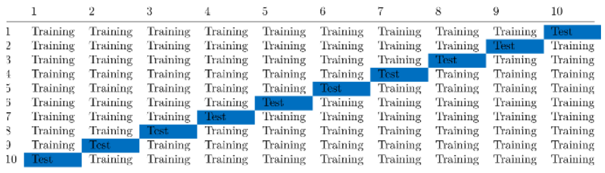
KDD99 is the most used data set in the intrusion detection domain, see (Özgür & Erdem 2016). According to Özgür and Erdem (Özgür & Erdem 2016), KDD99 dataset is used by 149 articles indexed in SCI-index between 2010-2015. KDD99 has 41 features to 23 classes. KDD99 is a suite of different datasets, one of these datasets is 10% of KDD99. KDD99 10% is widely used since training and testing time issues with full dataset. In our experiments, KDD99 10% (494021 instances) have been used due to training and testing time problems with full dataset. Table 3 shows the 10 fold cross validation results for different classifier comparison metrics using KDD99 10% dataset. As can be seen from Table 3, making a comparison between different classifiers using all these metrics is not easy.

Table 3. All Metrics for KDD99 dataset

Classifier name	**Mean training time (sc)**	**Mean testing time(sc)**	**Mean accuracy score**	**Mean precision score**	**Mean recall score**	**Mean F1 score**	**Mean model size (k bytes)**
AdaBoost	52.4111	1.8633	0.7866	0.7866	0.7866	0.7866	63810
Bagging	15.0923	0.2439	0.9997	0.9997	0.9997	0.9997	1027874
Decision Tree (CART)	2.3330	0.0084	0.9995	0.9995	0.9995	0.9995	126409
K Neighbors	0.0486	441.6670	0.9986	0.9986	0.9986	0.9986	149392857
Logistic Regression	154.8023	0.0195	0.9574	0.9574	0.9574	0.9574	8642
Multi Layer Perceptron	298.5620	0.2062	0.9963	0.9963	0.9963	0.9963	214557
Naive Bayes	0.4724	0.3285	0.9484	0.9484	0.9484	0.9484	16251
One Rule	0.0269	0.0547	0.4098	0.4098	0.4098	0.4098	836
Random Forest	19.5079	0.7065	0.9998	0.9998	0.9998	0.9998	17025372
Support Vector Machines	14453.5621	1306.2801	0.6352	0.6352	0.6352	0.6352	179564859
Zero Rule	0.0222	0.0004	0.5684	0.5684	0.5684	0.5684	839

Since our working environment is a resource constrained environment, following metrics are chosen for DAR comparison. These are model-size for low-memory constraint, training and testing time for less powerful CPU constraint and accuracy for overall performance. For these metrics 100 points are distributed. Accuracy is given 40 points, model size is given 30 points, and training time is given 10 points, and testing time is given 20 points. These metrics are chosen by the authors subjectively according to their knowledge in intrusion detection domain.

Using the same approach in section 3, Table 4 DAR Results is obtained. All the results are obtained using 10-fold cross validation. In 10-fold cross validation, dataset divided into 10 folds. Among these 10 folds, 9 of them are used for training, while remaining 1-fold is used for testing, see Figure 1. This means that for every cross validation 444619 instances are used for training and 49402 instances are used for testing. This procedure is repeated 10 times and results are averaged. Full metric results for every run can be seen in github repository.



	1	2	3	4	5	6	7	8	9	10
1	Training	Training	Training	Training	Training	Training	Training	Training	Training	Test
2	Training	Training	Training	Training	Training	Training	Training	Training	Test	Training
3	Training	Training	Training	Training	Training	Training	Training	Test	Training	Training
4	Training	Training	Training	Training	Training	Training	Test	Training	Training	Training
5	Training	Training	Training	Training	Training	Test	Training	Training	Training	Training
6	Training	Training	Training	Training	Test	Training	Training	Training	Training	Training
7	Training	Training	Training	Test	Training	Training	Training	Training	Training	Training
8	Training	Training	Test	Training	Training	Training	Training	Training	Training	Training
9	Training	Test	Training	Training	Training	Training	Training	Training	Training	Training
10	Test	Training	Training	Training	Training	Training	Training	Training	Training	Training

Figure 1. Cross Validation 10-fold

Experiments are realized using python (3.8) and scikit learn toolkit (0.23.2) (Pedregosa et al. 2011). Experiment computer has an operating system of Linux Mint 20 Ulyana with 32G of RAM. Its CPU is an Intel(R) Core(TM) i77700K CPU @ 4.20GHz with 8 cores. All classifiers are used with default parameters. Software code for the experiments is provided in https://github.com/ati-ozgur/classifier-comparison-using-DAR.

In the results, best possible total score is 100 points of 100 points. Total column is found using the DAR process outlined in the section 3. Here it is weighted normalized summation of accuracy, model size, training and testing time. Accuracy is normalized using Equation 1 since high values are good, while other three metrics are normalized using Equation 2 since high values are worse. Then normalized values of these metrics are multiplied with weights and summed to get the total column in Table 4.

Table 4. DAR Results for Intrusion Detection in KDD99 dataset time.

Classifier Name	**Accuracy (%)**	**Model Size(Mb)**	**Training Time (sec)**	**Testing Time (sec)**	**Total**
Weights	40	30	10	20	100
Decision Tree (CART)	99.95	123.51	3.21	0.01	99.97
Bagging	99.97	1008.04	10.53	0.20	99.89
Multi Layer Perceptron	99.62	210.02	1054.49	0.46	99.42
Random Forest	99.98	16642.80	129.21	3.45	98.29
Logistic Regression	95.98	8.54	504.06	0.03	97.14
Naive Bayes	94.84	15.91	0.21	0.17	96.52
AdaBoost	78.66	64.39	24.32	0.94	85.55
Zero Rule	56.84	0.88	0.01	0.01	70.79
One Rule	40.90	0.88	0.01	0.02	60.00
K Neighbors	99.86	302790.12	2858.42	1517.63	57.78
Support Vector Machines	63.52	175356.42	34064.04	2686.32	27.94

According to results, best classifier with very small lead is the Decision Tree classifier. Other three classifiers —Bagging, Multi Layer Perceptron, Random Forest — get very similar scores in DAR table. Firstly, the DAR process effectively removes outliers from consideration. Support Vector Machines and K Neighbors classifiers are outliers in model size and testing time categories and accordingly their total scores becomes lower. Secondly, if we have used only accuracy as metric, K Neighbors with accuracy of 99.86% will be among the best classifiers. But its total score is below 60 points since its other 3 metrics are very bad compared to other classifiers.

In short, DAR approach helps to choose among different classifiers according to given constraints.

4. Discussion

Novelty of the proposed method is its simplicity. Compared to previously proposed decision methods (Basheleishvili et al. 2019), (Çınar & Özer Uygun 2019), (Faydalı & Erkan 2020), DAR is easier to understand and implement. An example Excel file for car example and python code for the IDS example is provided in the software source code site. The main model complexity is O(n) since only 1 loop exists in the solution. All other calculations are simple arithmetic operations, again showing simplicity of the method compared to alternatives.

Even though, DAR is a simple method, it is helpful to implement it for comparing alternatives. In the machine learning domain using only accuracy to rank classifiers is rampant. But, in real world implementations, other considerations should be considered.

Limitation of the current study is that weights are subjective and taken from only one expert. In a real-world scenario, other methods such as given in section 3.1 should be implemented to use information given by multiple experts.

5. Conclusions

A method to choose among different classifiers named Decision Analysis and Resolution (DAR) is proposed. This method is originated in software engineering domain. According to given constraints, this method ranks the classifiers. A resource constrained environment has been chosen for demonstration purposes. For this environment, accuracy, model size, training time and testing time have been chosen for comparison metrics. Using scikit-learn toolbox, 11 supervised learning classifiers have been applied to well-known intrusion detection dataset KDD99. According to our results, Decision Tree is most suitable classifier for this resource constrained environment, even though Random Forest is the best accuracy classifier.

A natural progression of this work is to test the current code on real micro controllers like Raspberry Pi or Arduino. Using different datasets on the resource constrained environments may be another application. For example, deep learning systems are very popular for image processing tasks like face detection and car plate detection. But these deep learning systems are also resource hungry. Comparing them would be an interesting application.

References

Amatriain, X. & Basilico, J. (2012), 'Netflix recommendations: Beyond the 5 stars (part 1)', https://netflixtechblog.com/ netflix-recommendations-beyond-the-5-stars-part-1-55838468f429.

Axelsson, S. (1999), The base-rate fallacy and its implications for the difficulty of intrusion detection, in 'In Proceedings of the 6th ACM Conference on Computer and Communications Security', pp. 1–7.

Basheleishvili, I., Bardavelidze, A. & Tsiramua, S. (2019), 'The development of a model for decision support system of assessment and selection of university academic staff', Journal of Intelligent Systems: Theory and Applications 2(2), 18–23.

Çınar, A. & Özer Uygun (2019), 'Selecting green supplier using intuitionistic fuzzy AHP', Journal of Intelligent Systems: Theory and Applications 2(2), 24– 31.

Danielson, M. & Ekenberg, L. (2017), Trade-offs for ordinal ranking methods in multi-criteria decisions, in 'Lecture Notes in Business Information Processing', Springer International Publishing, pp. 16–27.

Faydalı, R. & Erkan, E. F. (2020), 'A fuzzy VIKOR method for machine selection', Journal of Intelligent Systems: Theory and Applications 3(1), 7–12.

Ileri, Y. & Furat, M. (2020), 'A roadmap for digitalization of industrial processes', European Journal of Science and Technology pp. 349 – 357.

Karahan, O. & Hökelek, H. (2020), 'Mobile robot position controlling system based on IoT through Raspberry Pi', Journal of Intelligent Systems: Theory and Applications 3, 25 – 30.

Karahan, O. & Kaya, B. (2020), 'Raspberry Pi firewall and intrusion detection system', Journal of Intelligent Systems: Theory and Applications 3, 21 – 24.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T. & Hoffmann, M. (2014), 'Industry 4.0', Business & Information Systems Engineering 6(4), 239–242.

Özgür, A. & Erdem, H. (2012), 'Saldırı tespit sistemlerinde kullanılan kolay erişilen makine öğrenme algoritmalarının karşılaştırılması', Bilişim Teknolojileri Dergisi 5, 41–48.

Özgür, A. & Erdem, H. (2016), 'A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015', PeerJ Preprints .

Özgür, A. & Erdem, H. (2018), 'Feature selection and multiple classifier fusion using genetic algorithms in intrusion detection systems', Journal of the Faculty of Engineering and Architecture of Gazi University 33, 0 – 0.

Özgür, A., Nar, F. & Erdem, H. (2018), 'Sparsity-driven weighted ensemble classifier', International Journal of Computational Intelligence Systems 11, 962– 978.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Édouard Duchesnay (2011), ‘Scikit-learn: Machine learning in python’, Journal of Machine Learning Research 12(85), 2825–2830.

Phillips, B. C. & Polen, S. M. (2002), ‘Add decision analysis to your cots selection process’, Software Engineering Technology .

Research, G. V. (2019), ‘Microcontroller market size, share & trends analysis report’, https://www.grandviewresearch.com/industry-analysis/ microcontroller-market. Last Accessed August 2019.

Sahingoz, O. K. (2019), ‘A clustering approach for intrusion detection with big data processing on parallel computing platform’, Balkan Journal of Electrical and Computer Engineering 7, 286 – 293.

Saleh, N. & Hussein, N. (2019), ‘Artificial intelligence in corneal topography’, Journal of Intelligent Systems: Theory and Applications 2(1), 1–6.

Shukla, V. & Auriol, G. (2013), Methodology for determining stakeholders’ criteria weights in systems engineering, in ‘Proceedings of the Posters Workshop at CSDM’.

Taşcı, E. (2019), ‘A meta-ensemble classifier approach: Random rotation forest’, Balkan Journal of Electrical and Computer Engineering 7, 182 – 187.

Team, C. P. (2006), Cmmi for development, version 1.2, Technical report.

URL: http://btd.gazi.edu.tr/dergi/sayi/volume5-2-5.pdf

Uygun, Y. & Ilie, M. (2018), Autonomous Manufacturing-related Procurement in the Era of Industry 4.0, Springer Fachmedien Wiesbaden, pp. 81–97.

Wilson, J. & Jungner, G. (1968), Principles and practice of screening for disease, Technical report, World Health Organization.

Yavuz, H. S., Çevikalp, H. & Edizkan, R. (2016), ‘A comprehensive comparison of features and embedding methods for face recognition’, Turkish Journal of Electrical Engineering and Computer Science 24, 313 – 340.

Yılmaz, A. (2020), ‘Assessment of mutation susceptibility in DNA sequences with word vectors’, Journal of Intelligent Systems: Theory and Applications 3(1), 1–6.