Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 11, No. 4, 2023, 517-524.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Original paper

# Auto-UFSTool: An Automatic Unsupervised Feature Selection Toolbox for MATLAB

Farhad Abedinzadeh Torghabeh[1], Yeganeh Modaresnia[1], and Seyyed Abed Hosseini[2*]

*1. Department of Biomedical Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran.*
*2. Department of Electrical Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran.*

**Article Info**

**Article History:**
*Received 11 March 2023*
*Revised 31 July 2023*
*Accepted 19 September 2023*

*DOI:10.22044/jadm.2023.12820.2434*

**Keywords:**
*Unsupervised Feature Selection, MATLAB, Automatic Toolbox, Dimension Reduction, Unsupervised Learning.*

**Corresponding author: abed_hosseyni@yahoo.com (S. A. Hosseini).*

**Abstract**

Various data analysis research has recently become necessary in finding and selecting the relevant features without class labels using Unsupervised Feature Selection (UFS) approaches. Despite the fact that several open-source toolboxes provide feature selection techniques to reduce redundant features, data dimensionality, and computation costs, these approaches require programming knowledge, which limits their popularity, and has not adequately addressed unlabeled real-world data. Automatic UFS Toolbox (Auto-UFSTool) for MATLAB, proposed in this study, is a user-friendly and fully-automatic toolbox that utilizes several UFS approaches from the most recent research. It is a collection of 25 robust UFS approaches, most of which were developed within the last five years. Therefore, a clear and systematic comparison of competing methods is feasible without requiring a single line of code. Even users without any previous programming experience may utilize the actual implementation by the Graphical User Interface (GUI). It also provides the opportunity to evaluate the feature selection results and generate graphs that facilitate the comparison of subsets of varying sizes. It is freely accessible in the MATLAB file exchange repository, and includes scripts and source code for each technique. The link to this toolbox is freely available to the general public on: bit.ly/AutoUFSTool.

## 1. Introduction

In several applications with high dimensional data, feature selection [1] (also called variable subset selection) has been found to provide a greater accuracy while using less data.

In these fields, the observations or samples under examination frequently have useless and redundant information in their descriptions [2]. This might substantially impact data processing, leading to biases or even inaccurate models. Feature selection refers to methods that select essential features for developing predictive models and evaluating their variables in tasks such as classification, regression, and clustering. Furthermore, feature selection not only reduces the dimensionality of the data, making it easier to visualize and to understand, but also increases the generalization of models [3]. Feature selection is an intriguing research topic due to its obvious advantages, where numerous feature selection approaches have been proposed in the recent decades. Feature selection approaches can be categorized as supervised, semi-supervised, and unsupervised based on the information in the datasets. For the supervised approach, the data must be labeled to identify and select significant features [4,5]. Semi-supervised approaches simply need the labeling of particular objects. On the other hand, Unsupervised Feature Selection (UFS) approaches do not require a supervised dataset. Over the last few decades, various methods have been developed for selecting features; most of them were created for supervised classification problems [6]. However, due to the recent technological

advancements and the large volume of data without labels generated in many applications including text mining, bioinformatics, image retrieval, social media, and intrusion in network security, as examples, UFS approaches have attracted the scientific community's attention. UFS approaches have two significant benefits: a) they are impartial, and perform well when previous information is not provided, as opposed to supervised feature selection approaches, which may not be able to handle data from new classes. b) they can lessen the danger of data overfitting [7].

Similar to the other feature selection methods, UFS may be classified into four primary ways based on the strategy of feature selection [8]:

- Filter methods: assess the data directly to identify the most important features, and no clustering technique is used to direct the search for relevant features. Primarily, filter techniques are characterized by their fast and scalable solution.
- Wrapper methods: assess subsets of features utilizing the clustering algorithm's outcomes. This methodology is defined by identifying subsets of features that increase the clustering quality. They are nevertheless computationally expensive algorithms that can only be applied to clustering techniques that follow a specific methodology.
- Hybrid methods: utilize the strengths of previous methods, while seeking a balance between efficiency and effectiveness.
- Embedding methods: it is possible to view embedded methods as a sub-category within the three main above-mentioned methods.

The rest of the study is structured as what follows. Auto-UFSTool is given in Section 2. The method is presented in Section 2.1. Section 2.2 provides an example, and Section 2.3 describes the implementation, and finally, Section 3 discusses the evaluation metrics of the paper.

## 2. Auto-UFSTool

### 2.1. Method

Several open-source toolboxes such as sklearn in python [9], the caret package in R [10], the Feature Selection Library (FSLib) in MATLAB [11], weka in Java [12], and MATLAB toolbox for Feature Ranking (MatFR) [13] have published a few approaches for feature selection. However, a few freshly found strategies are included in these toolboxes. Furthermore, these toolboxes require programming skills, which limit their wider adoption, and they have not adequately addressed the underlying issue of unlabeled real-world data.

We intend to develop a user-friendly toolbox with various UFS approaches. Automatic UFS Toolbox (Auto-UFSTool) is a library for feature selection that is extensively used in MATLAB, a language that is frequently used in many scientific fields. It is publicly accessible through the MATLAB file exchange repository.

The toolbox provides users with 25 efficient UFS techniques on the subject. It includes an example script and the source code for each technique as well as Graphical User Interface (GUI) provides users with an automated user-friendly environment. This article presents an overview of the toolbox's UFS techniques, which include filter, embedding, hybrid, and wrapper approaches.

**Table 1. Summary of different works pertaining to unsupervised feature selection.**

| Row | Articles | Acronym | Type | Complexity |
|---|---|---|---|---|
| 1 | [14] | CFS | f | $O((n^2/2)T)$ |
| 2 | [15] | LS | f | N/A |
| 3 | [16] | SPEC | f | N/A |
| 4 | [17] | MCFS | f | N/A |
| 5 | [18] | UDFS | f | $O(n^2)+O(T^2)$ |
| 6 | [19] | LLCFS | f | N/A |
| 7 | [20] | NDFS | f | $O(n^3)+O(cT^2)$ |
| 8 | [21] | RUFS | f | $O(T^2)+O(Tn)$ |
| 9 | [22] | FSASL | w | $O(n^3+Tn^2)$ |
| 10 | [23] | SOCFS | f | N/A |
| 11 | [24] | SOGFS | e | N/A |
| 12 | [25] | UFSOL | w | $O(iTcn^3)$ |
| 13 | [26] | Inf-FS | f | $O(n^{2.37}+ (1+T))$ |
| 14 | [27] | DGUFS | w | N/A |
| 15 | [28] | SRCFS | f | N/A |
| 16 | [29] | CNAFS | e | $O(Tn^2 + T^2n + T^3)$ |
| 17 | [30] | EGCFS | e | N/A |
| 18 | [31] | RNE | e | N/A |
| 19 | [32] | Inf-FS2020 | f | $O(n^3(1+T))$ |
| 20 | [33] | UAR-HKCMI | h | N/A |
| 21 | [34] | FMIUFS | h | N/A |
| 22 | [35] | FRUAR | h | N/A |
| 23 | [36] | U2FS | e | $O(T^3 + n^3)$ |
| 24 | [37] | JMVFG | f | N/A |
| 25 | [38] | NNSE | e | N/A |

All of the original resources and codes are available online, as well as in the original articles. Auto-UFSTool is a list of UFS methodologies. Table 1 summarizes their type, which is f = filters, w = wrappers, h = hybrid, and e = embedding methods, the abbreviation of their names, and complexity. In terms of complexity, T represents the number of samples, n represents the number of initial observations, i represents the number of iterations in the case of iterative methods, C represents the number of classes, and in some articles, the complexity is Not Available (N/A).

This work's significance arises from three key factors. Firstly, there are 25 UFS methods implemented, and 11 of them have been introduced

in the past five years. This ensures a comprehensive coverage of recent advancements. Secondly, the toolbox is designed to be user-friendly, simplifying the feature selection process. Thirdly, the users can effortlessly load their data, apply specific techniques, evaluate results, and conduct comparisons without the need to write a single line of code. As a result, a systematic comparison of different methodologies from diverse viewpoints becomes practical and straightforward.

Figure 1 shows the GUI window of the Auto-UFSTool. Four main blocks have been designed to facilitate user interaction with the GUI. The first block contains a push button that lets users load various data formats such as Excel or Mat files. If structured data is loaded into the program, the user will be asked to separate it by type (i.e. feature vectors and their labels).

The following block contains a drop-down list with 25 UFS methods, allowing the users to select one easily. Once the user has chosen a method from the drop-down list, the selected method is displayed in a pop-up window. To implement the preferred method, the user is prompted to initialize parameters manually or by default. The options window will be displayed if the user wishes to select the options manually. An in-depth understanding of the original UFS technique's paper is required to select options and parameters [14]-[38].

Once the feature selection has been completed, the workspace will be saved with the results. A button in the following block allows user to cluster their selected features into subsets based on the number of baseline features or manually select the size and the number of subsets using the K-means clustering method. In the next block, clustering results will be evaluated using the metrics explained in Section 3. In the final block, the users can view the obtained results as a graphical representation.

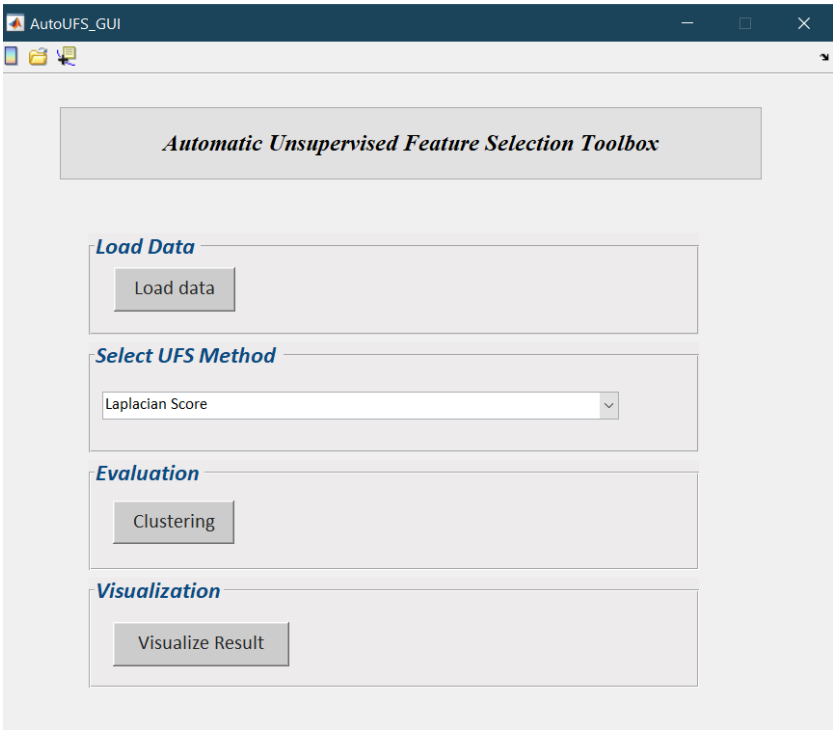


**Figure 1. Auto-UFSTool graphical user interface environment.**

## 2.2. Example

In the presence of an input matrix $X_{m\times n}$ ($m$ samples and $n$ features per sample), the process for utilizing one of the UFS methods in the toolbox is as follows:

$$\text{Result} = \text{Auto\_UFSTool}(\text{X}, \text{Method}). \quad (1)$$

where Result represents the output rank indices of features in descending order of their relative importance or a subset of the feature. The method should be replaced with the acronym of the method's name given in Table 1.

As illustrated in Figure 1, a user can utilize any UFS method from the drop-down list that is automatically displayed. An alternative approach for implementing the UFS procedure involves expressing Eq. (1) as a script.

An example is presented using the Columbia Object Image Library (COIL20) dataset, a

collection of images from Columbia University, featuring 20 distinct objects [39]. As each object was rotated on a turntable, 72 images were captured at 5 degrees apart, and each object contained 72 images. Each image is 32 by 32 pixels and contains 256 gray levels per pixel. As a result, with the input $X, m = 1440\ and\ n = 1024$.

After loading the data from the first block, one line of code to utilize the Robust UFS (RUFS) [21] algorithm is presented below.

$$\text{Result} = \text{Auto\_UFSTool}(X', \text{'RUFS'}). \quad (2)$$

It is crucial to recognize that users will either directly provide all method options and parameters, or the algorithm will utilize default values initialized with the input data for necessary parameters. The file "UFS_Names.mat" contains the names of all UFS methods. Please refer to the original articles and algorithm implementations for further information cited in Table 1.

This toolbox allows the user to generate graphs for comparing their results. The process involves first prompting the user to indicate their preference for generating a plot. Subsequently, based on either default settings, determined by the size of the primary feature space, or the user's specific selection, a line chart is generated accordingly. It generates graphs that facilitate the user's comparison of findings based on the selected criteria and charts' size. This will demonstrate the evaluation criteria presented in Section 3.

## 2.3. Implementation

This toolbox is written using the GUI Development Environment (GUIDE) in MATLAB 2018b, which is a prominent programming language for machine learning and pattern recognition research. The Auto-UFSTool GUI was tested on 64-bit Windows 8/10/11 PCs with MATLAB R2019b/R2022a on a range of publicly available datasets based on original articles.

## 3. Evaluation

Evaluation of the results of a clustering algorithm is a crucial step in the data clustering process. In supervised learning, "the evaluation of the resultant classification model is a vital element of the classification model development process, and there are widely acknowledged evaluation techniques and procedures" [40].

Due to the nature of unsupervised learning, cluster validation is not well-developed, resulting in difficulty in assessing clustering algorithm quality. This challenge gives birth to numerous evaluation methodologies. Several considerations must be discussed for validating clustering algorithm results, while evaluating clustering results [40].

- Determining the clustering tendency of the data (i.e. the existence of a non-random structure).
- Finding the proper number of clusters.
- Evaluating the quality of clustering results without using metadata.
- Comparing derived results with external information.
- Comparing two cluster sets to decide which is superior.

The first three problems are handled through internal or unsupervised validation, as no external information is utilized. External or overseen validation resolves the fourth problem. Supervised and unsupervised validation methods can address the last issue.

Auto-UFSTool provides two internal and six external validation methods to evaluate the performance of UFS algorithms. External validation proceeds by introducing extra information into the clustering validation procedure, namely external class labels for the training instances. External validation methods are not used on most clustering issues since unsupervised learning approaches are typically employed when such information is unavailable. However, they can still be used when external information is accessible and synthesizing data from an existing data collection.

The K-means clustering algorithm, a widespread and fundamental clustering technique with several applications, is utilized to evaluate the efficacy of feature selection techniques.

The evaluation result will be generated under two conditions:

- Baseline features

Using K-means, all original features will be clustered.

- Variable subsets

If the number of original features is fewer than 100, the toolbox will offer five potential subset sizes: 5, 10, 20, 30, and 40.

If the number of features is more significant than 100 but less than 1000, the toolbox will recommend 50, 100, 150, 200, and 300 as alternative subset sizes.

It is also possible to skip this option and manually select the subset size and number.

To prove the validity of the selected technique, the Auto-UFSTool provides the opportunity to manually or automatically compare the result with subsets of varying sizes.

The Auto-UFSTool offers eight commonly used evaluation measures, including redundancy, Jaccard, purity, NMI, accuracy, precision, recall,

and F-measure, for assessing the performance of UFS algorithms. Additionally, the users can employ other metrics of their choice if desired.

### 3.1. Redundancy

Assume $\boldsymbol{F}$ is the set of selected features, and $X_F$ solely consists of $\boldsymbol{F}$ features, where $\rho_{i,j}$ yields the Pearson correlation between $fi$ and $fj$ and $m$ is the number of selected features.

$$RED(F) = \frac{1}{m(m-1)} \sum_{f_i, f_j \in F, i>j} \rho_{i,j}, \quad (3)$$

This metric evaluates the average correlation between all feature pairs; a high value implies that a significant number of selected features are strongly correlated. Hence, redundancy is anticipated in $\boldsymbol{F}$.

### 3.2. Jaccard

The Jaccard score derived from

$$JAC(K_F, K, k) = \frac{1}{n} \sum_{i=1}^{n} \frac{NB(i,k,K_F) \cap NB(i,k,K)}{NB(i,k,K_F) \cup NB(i,k,K)}, \quad (4)$$

where $K_F = X_F X_F^t$; and $K_F$ and $K$ are the similarity matrix computed from the selected features and the input similarity matrix, respectively. Furthermore, $NB(i,k,K)$ yields the $k$ nearest neighbors of the $i$th instance based on the specified pairwise similarity $K$. The Jaccard score estimates the average overlap between the $K_F$ and $K$ neighbourhoods. A high Jaccard score shows consistency between the pairwise similarities described by the two similarity matrices.

The last two metrics are used to evaluate an algorithm's ability to preserve sample similarity in continuous and discrete ways, respectively [41].

### 3.3. Purity

Purity determines if each cluster includes only instances of the same class.

$$U = \sum_i p_i (\max_j \frac{p_{ij}}{p_i}). \quad (5)$$

In Eq. (5), $p_i = n_i/n$, $p_j = n_j/n$, and $p_{ij} = n_{ij}/n$, where $n_{ij}$ represents the number of instances in the class $i$ discovered in the cluster $j$ and $n_i(n_j)$ is the number of instances in the cluster $i(j)$.

### 3.4. NMI

Normalized Mutual Information (NMI) indicates how much uncertainty regarding class labels is reduced when cluster labels are known. One of the advantages of NMI is that it is normalized, which permits the evaluation of various clustering models with different numbers of clusters.

$$NMI(P,Q) = \frac{I(P,Q)}{\sqrt{H(P)H(Q)}}. \quad (6)$$

Figure 2 shows a comparison of the above-mentioned evaluation metrics with their baselines.

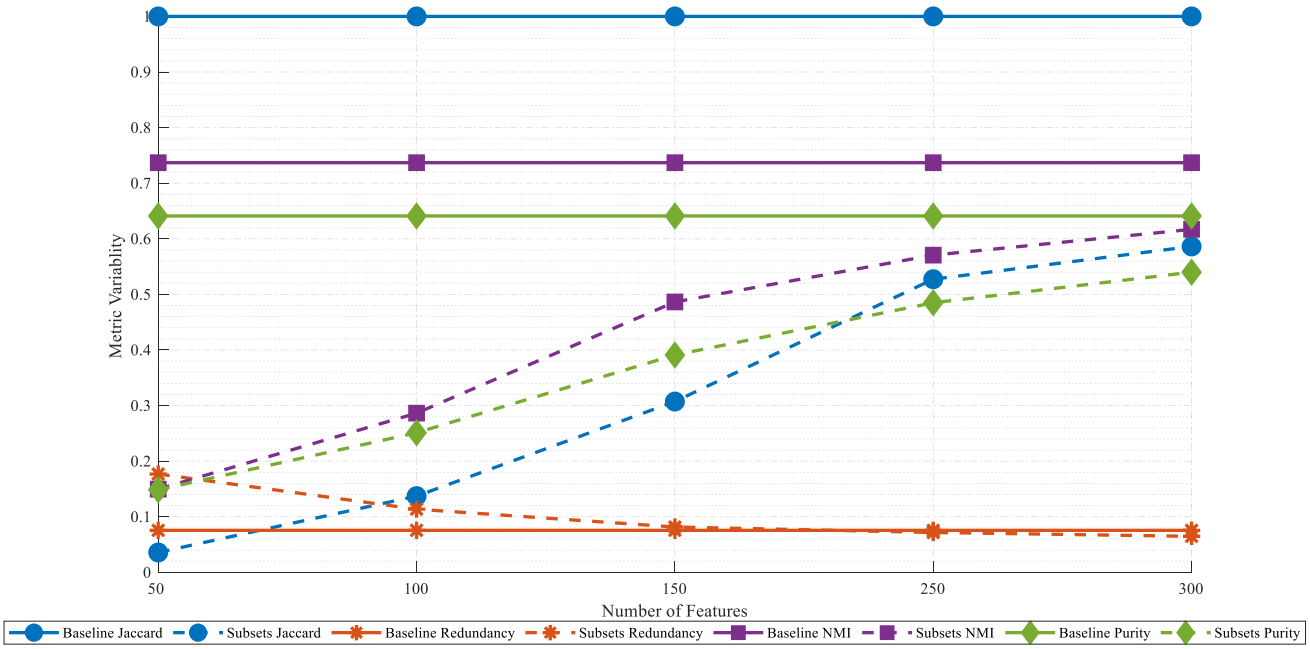


**Figure 2. Variability of the Redundancy, Jaccard, Purity, and NMI subsets and their baselines.**

In order to compare the result of a clustering algorithm $C = \{C1, C2, \ldots, Cm\}$ to a potentially different partition $P = \{P1, P2, \ldots, Pm\}$, which may indicate the analyst's specialist knowledge,

the results obtained by another clustering algorithm, or simply a grouping considered to be "correct" [42].

A contingency matrix must be constructed to analyze the clusters identified by the algorithm to conduct this analysis. This contingency matrix has four terms:

- $TP$: The number of data pairings in the same cluster in $C$ and $P$.
- $FP$: The number of data pairings discovered inside the same cluster in $C$ but distinct clusters in $P$.
- $FN$: The number of data pairings that were discovered in distinct clusters in $C$ but the same cluster in $P$.
- $TN$: The number of data pairings discovered in distinct clusters in $C$ and $P$.

From these four criteria, it is simple to conclude:

External validation methods that may be used to compare two data partitions identify the relation between each cluster found in $C$ and its natural correlation to the classes in the reference result specified in $P$. Several metrics may be constructed to quantify the similarity between the clusters in $C$, which were generated by the clustering algorithm, and the clusters in $P$.

### 3.5. Accuracy

Accuracy measures either true positive or true negative against the total dataset.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}. \quad (7)$$

### 3.6. Precision

Precision measures the true positives or the number of correctly classified instances inside the same cluster.

$$PRE = \frac{TP}{TP + FP} = \frac{TP}{P} = \frac{P_{i,j}}{P_i}. \quad (8)$$

### 3.7. Recall

Recall measures the proportion of components that are appropriately grouped inside the same cluster.

$$REC = \frac{TP}{TP + FN} = \frac{P_{i,j}}{P_j}. \quad (9)$$

### 3.8. $F_1$ Score

$F_1$ score combines accuracy and recall into a single metric, the weighted harmonic mean of the variables:

$$F_1 = \frac{2TP}{2TP + FP + FN} = \frac{2P_{i,j}}{P_i + P_j}. \quad (10)$$

## 4. Future Works

Four factors come into play for future work. First, incorporate current UFS approaches into the toolbox and follow up on newly developed techniques. Secondly, to evaluate the validity of selected features and combine them with clustering assessment criteria. In unsupervised classifications, for example, a variety of criteria may be used to estimate the appropriate number of clusters. Thirdly, explore various clustering strategies. Last but not least, the toolbox might be built in Python and R to hasten the implementation of these UFS techniques.

## References

[1] H. Liu and H. Motoda, Feature selection for knowledge discovery and data mining. in *The Springer International Series in Engineering and Computer Science*, Springer New York: NY, 1998.

[2] G. Ritter, Robust cluster analysis and variable selection. CRC Press, 2014.

[3] R. J. Curts and D. E. Campbell, Pattern recognition algorithms for data mining. Chapman & Hall CRC Press, 2004.

[4] S. Hosseini and M. Khorashadizade, "Efficient Feature Selection Method using Binary Teaching-learning-based Optimization Algorithm," *J. AI Data Min.*, vol. 11, no. 1, pp. 29-37, January 2023.

[5] Z. Shojaee, S. A. Shahzadeh Fazeli, E. Abbasi, and F. Adibnia, "Feature Selection based on Particle Swarm Optimization and Mutual Information," *J. AI Data Min*., vol. 9, no. 1, pp. 39-44, January 2021.

[6] C.C. Aggarwal. Data classification: algorithms and applications. Chapman & Hall CRC, 1st ed. Taylor & Francis, 2014.

[7] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 907-948, 2020.

[8] H. Liu, Feature engineering for machine learning and data analytics. CRC Press, 2018.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, *et al*., "Scikit-learn: machine learning in python," *J. of Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, January 2011.

[10] M. Kuhn, "Building predictive models in R using the caret package," *J. Stat. Softw.*, vol. 28, no. 5, pp. 1-26, 2008.

[11] G. Roffo, "Feature selection library (MATLAB Toolbox)," *Arxiv.org*, August 2018. [Online] Available: https://arxiv.org/abs/1607.01327. [Accessed Sept. 10, 2022]

[12] I. H. Witten, E. Frank, and J. Geller, "Data mining: practical machine learning tools and techniques with Java implementations," *ACM SIGMOD Record*, vol. 31, no. 1, pp. 76–77, March 2002.

[13] Z. Zhang, X. Liang, W. Qin, S. Yu, and Y. Xie, "matFR: a MATLAB toolbox for feature ranking," *Bioinformatics*, vol. 36, no. 19, pp. 4968–4969, December 2020.

[14] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 389–422, 2002.

[15] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," *Advances in neural information processing systems*, vol. 18, 2005.

[16] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proceedings of the 24th international conference on Machine learning*, Corvallis, 2007, pp.1151-1157.

[17] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for multi-cluster data," in *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining*, 2010, pp. 333–342.

[18] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "ℓ2,1-Norm regularized discriminative feature selection for unsupervised learning," *International joint conference on artificial intelligence, IJCAI*. 2011, pp. 1589–1594.

[19] H. Zeng and Y. M. Cheung, "Feature selection and kernel learning for local learning-based clustering," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1532-1547, August 2011.

[20] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Proceedings of the association for the advancement of artificial intelligence conference on artificial intelligence, AAAI*, vol. 26, no. 1, 2012 pp. 1026–1032.

[21] M. Qian and C. Zhai, "Robust unsupervised feature selection," in *Proceedings of the twenty-third international joint conference on artificial intelligence, IJCAI*, 2013, pp. 1621-1627.

[22] L. Du and Y. D. Shen, "Unsupervised feature selection with adaptive structure learning," in *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining*, August 2015, pp. 209-218.

[23] D. Han and J. Kim, "Unsupervised simultaneous orthogonal basis clustering feature selection," in *IEEE conference on computer vision and pattern recognition, CVPR*, 2015, pp. 5016–5023.

[24] F. Nie, W. Zhu, and X. Li, "Unsupervised feature selection with structured graph optimization," in *Proceedings of the association for the advancement of artificial intelligence conference on artificial intelligence, AAAI*, vol. 30, no. 1, 2016, pp. 1302-1308.

[25] J. Guo, Y. Quo, X. Kong, and R. He, "Unsupervised feature selection with ordinal locality," in *IEEE international conference on multimedia and expo ,ICME*, 2017, pp. 1213–1218.

[26] G. Roffo, S. Melzi, and M. Cristani, "Infinite feature selection," in *IEEE international conference on computer vision, ICCV*, 2015, pp. 4202–4210.

[27] J. Guo and W. Zhu, "Dependence guided unsupervised feature selection," in *Proceedings of the association for the advancement of artificial intelligence conference on artificial intelligence, AAAI*, vol. 32, no. 1, 2018, pp. 2232-2239.

[28] D. Huang, X. Cai, and C. D. Wang, "Unsupervised feature selection with multi-subspace randomization and collaboration," *Knowledge-based systems*, vol. 182, p. 104856, October 2019.

[29] A. Yuan, M. You, D. He, and X. Li, "Convex non-negative matrix factorization with adaptive graph for unsupervised feature selection," *IEEE Transactions on cybernetics*, vol. 52, no. 6, pp. 5522-5534, 2022.

[30] R. Zhang, Y. Zhang, and X. Li, "Unsupervised feature selection via adaptive graph learning and constraint," *IEEE Transactions on neural networks and learning systems*, vol. 33, no. 3, pp. 1355–1362, March 2022.

[31] Y. Liu, D. Ye, W. Li, H. Wang, and Y. Gao, "Robust neighborhood embedding for unsupervised feature selection," *Knowledge-based systems*, vol. 193, p. 105462, April 2020.

[32] G. Roffo, S. Melzi, U. Castellani, A. Vinciarelli, and M. Cristani, "Infinite feature selection: a graph-based feature filtering approach," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4396–4410, 2020.

[33] Z. Yuan, H. Chen, X. Yang, T. Li, and K. Liu, "Fuzzy complementary entropy using hybrid-kernel function and its unsupervised attribute reduction," *Knowledge-based systems*, vol. 231, pp. 107398, November 2021.

[34] Z. Yuan, H. Chen, P. Zhang, J. Wan, and T. Li, "A novel unsupervised approach to heterogeneous feature selection based on fuzzy mutual information," *IEEE Transactions on fuzzy systems*, vol. 30, no. 9, pp. 3395 – 3409, September 2022.

[35] Z. Yuan, H. Chen, T. Li, Z. Yu, B. Sang, and C. Luo, "Unsupervised attribute reduction for mixed data based on fuzzy rough sets," *Information sciences* (*Ny*), vol. 572, pp. 67–87, September 2021.

[36] A. Villa, A. M. Narayanan, S. Van Huffel, A. Bertrand, and C. Varon, "Utility metric for unsupervised feature selection," *Peer J computer science*, vol. 7, pp. 1–26, April 2021.

[37] S.G. Fang, D. Huang, C.D. Wang, and Y Tang, "Joint multi-view unsupervised feature selection and graph learning" *Arxiv.org*, August 2023. [Online] Avilable: https://arxiv.org/abs/2204.08247. [Accessed Sept. 18, 2023]

[38] M. You, A. Yuan, D. He, and X. Li, "Unsupervised feature selection via neural networks and self-expression with adaptive graph constraint," *Pattern Recognition*, vol. 135, pp. 109173, Mar. 2023.

[39] S. A. Nene, S. K. Nayar and H. Murase, "CAVE Software: Columbia Object Image Library (COIL-20)," Technical Report CUCS-005-96, February 1996.

[40] P.N. Tan, M. Steinbach, V. Kumar, Introduction to data mining. *Pearson Addison-Wesley*, 1st ed., 2014.

[41] Z. Zhao, L. Wang, H. Liu, and J. Ye, "On similarity preserving feature selection," *IEEE Transactions on knowledge and data engineering*, vol. 25, no. 3, pp. 619-632, March 2013.

[42] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of intelligent information systems*, vol. 17, pp. 107–145, 2001.

# Auto-UFSTool: جعبه‌ابزار انتخاب ویژگی بدون نظارت برای نرم‌افزار متلب

**فرهاد عابدین‌زاده طرقبه[1]، یگانه مدرس‌نیا[1] و سیدعابد حسینی[2,*]**

[1] گروه مهندسی پزشکی، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران.

[2] گروه مهندسی برق، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران.

ارسال ۲۰۲۳/۰۳/۱۱؛ بازنگری ۲۰۲۳/۰۷/۳۱؛ پذیرش ۲۰۲۳/۰۹/۱۹

**چکیده:**

اخیراً انجام پژوهش‌های مختلف تحلیل داده برای یافتن و انتخاب ویژگی‌های منا سب بدون دا شتن برچ سب د سته به کمک رویکردهای انتخاب ویژگی بدون نظارت ضروری شده است. علیرغم وجود چندین جعبه‌ابزار در دسترس که روش‌های انتخاب ویژگی را برای کاهش ویژگی‌های اضافی، ابعاد داده و هزینه‌های محاسباتی ارائه می‌دهند، نیاز به دانش برنامه‌نویسی و نپرداختن به داده‌های بدون برچسب دنیای واقعی، محبوبیت آن‌ها را کاهش داده است. در این مطالعه جعبه‌ابزار خودکار انتخاب ویژگی بدون نظارت Auto-UFSTool برای نرم‌افزار متلب پیشنهاد شده که کاربرپسند و کاملاً خودکار است و از رویکردهای انتخاب ویژگی بدون نظارت مختلف مشتق شده از جدیدترین پژوهش‌ها استفاده می‌کند. این جعبه‌ابزار مجموعه‌ای از ۲۵ رویکرد انتخاب ویژگی بدون نظارت قوی است که بیشتر آن‌ها در پنج سال گذشته توسعه یافته‌اند. بنابراین مقایسه واضح و سازمان‌یافته با روش‌های متفاوت را بدون نیاز به برنامه‌نویسی امکان‌پذیر می‌کند و حتی کاربران بدون تجربه قبلی برنامه‌نویسی، می‌توانند از پیاده‌سازی واقعی توسط رابط کاربری گرافیکی استفاده نمایند. همچنین این جعبه‌ابزار فرصت را برای ارزیابی نتایج انتخاب ویژگی و ایجاد نمودارها جهت مقایسه زیرمجموعه‌ها با اندازه‌های مختلف فراهم می‌کند. این جعبه‌ابزار در پایگاه تبادل فایل نرم‌افزار متلب به صورت رایگان قابل‌د سترس ا ست و شامل ا سکریپت‌ها و برنامه منبع برای هر روش است. این جعبه‌ابزار به‌صورت رایگان برای عموم در دسترس است: bit.ly/AutoUFSTool.

**کلمات کلیدی:** انتخاب ویژگی بدون نظارت، نرم‌افزار متلب، جعبه‌ابزار خودکار، کاهش ابعاد، یادگیری بدون نظارت.