Original scientific paper

Received: December 10, 2024.
Revised: March 21, 2025.
Accepted: April 11, 2025.

UDC:
004.42:004.738.5
004.855.6
10.23947/2334-8496-2025-13-1-175-190

Check for updates

A Web Application for Learning Support Vector Machine Algorithms in Computer Engineering

Nenad Jovanović[1], *Stefan Jovanović*[2], *Srećko Stamenković*[3], *Dražen Marinković*[4], *Negovan Stamenković*[5]

[1]Faculty of Technical Sciences, University of Pristina, Kosovska Mitrovica, Serbia, e-mail: nenad.jovanovic@pr.ac.rs
[2]Faculty of Electronic Engineering, University of Nis, Serbia, e-mail: stefan.n.jovanovic@elfak.rs
[3]Toplica Academy of Applied Studies, Department of Business Studies Blace, Serbia, e-mail: stamenkovic.srecko@gmail.com
[4]University MB, Belgrade, Serbia, e-mail: drazenmarinkovic@gmail.com
[5]Faculty of Sciences and Mathematics, University of Pristina, Kosovska Mitrovica, Serbia, e-mail: negovan.stamenkovic@pr.ac.rs

Abstract: In this paper, we present a web application designed for learning and visualizing Support Vector Machine (SVM) algorithms, which are key components in the fields of machine learning and data processing. The application was developed as an interactive tool that allows students and researchers to experiment with SVM models, providing insight into their structure and functionality. By using modern web technologies, the application offers a user environment that is accessible, intuitive, and adaptable for learning and research. In addition to implementing a web tool for learning the SVM algorithm, this study proposes a method for its application in teaching and analyzes the impact of applying the new interactive method on final learning outcomes. To assess the effectiveness of this tool, an experiment was conducted consisting of three phases: pre-testing, training, and post-testing. To evaluate students' experiences with the applied alternative learning method using the auxiliary tool and their perception of the software system's effectiveness, the standardized System Usability Scale (SUS) was used.

Keywords: *Support Vector Machine, educational technology, algorithm visualization, simulation systems, educational tools.*

Introduction

With the development of machine learning, intelligent software systems are increasingly being developed today, which, based on input data, use one or a group of algorithms to provide appropriate output data as a result of execution. Machine learning algorithms are used in a wide range of different fields, including medical diagnostics, control of robotic systems, management of industrial systems, telecommunications, finance and stock trading, the computer game industry, the music industry, and many others. However, in many cases, machine learning models are considered black boxes because the inner functionality of the underlying algorithms is not entirely understandable to analysts (Mühlbacher et al., 2014), and even for experts, tuning and parameterizing certain models can be challenging (Zeiler and Fergus, 2014). Due to the complexity of machine learning algorithms, there is a need for systems that enable their application in everyday work as well as in the education process. Well-known software systems, environments, and libraries for research and concrete application of machine learning algorithms are Weka (Hall et al., 2009), TensorFlow (Abadi et al., 2015), PyTorch (Paszke et al., 2019), Scikit-learn (Pedregosa et al., 2011), Keras (Chollet et al., 2015), and Shogun (Sonnenburg et al., 2010). The focus of this paper is on systems used as support in education to help students understand the dynamic behaviors of machine learning algorithms. The analysis of existing educational tools for visualization and simulation of machine learning was conducted in the second section.

*Corresponding author: stamenkovic.srecko@gmail.com

© 2025 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

The goal of machine learning is to construct algorithms capable of learning to predict specific target outputs. To achieve this, the learning algorithm is given specific training examples that demonstrate the target relationship between input and output values. The algorithm should generate approximately correct output, including examples that it did not encounter during training. For a good understanding of these topics, undergraduate students need to master other areas of computer engineering, and it is desirable to have some prior knowledge, which is often not acquired at previous levels of education (Çağlayan, 2019). Therefore, it is very important to properly motivate students to learn these topics. Some of the pedagogical methods recommended for teaching machine learning are hands-on tasks, collaborative learning, and problem-based learning (Hazzan and Mike, 2023). Hands-on tasks involve engaging students in real hands-on projects that include data collection, model training, and performance evaluation. This encourages practical application and solving the set tasks. Through the analysis of real problems, Problem-Based Learning helps students develop critical thinking and practical problem-solving skills. Collaborative learning allows students to solve problems together, share ideas and learn from each other. Computer engineering continuously strives for innovations in pedagogical methods and tools, especially in the field of machine learning. Auxiliary software tools should enable students and future engineers to visualize and simulate different algorithms for educational purposes and to gain insight into the possible effects of certain algorithms, thus more easily selecting the appropriate algorithm for specific needs and optimally determining the parameters with which to use the appropriate algorithms through simulation.

Rutten et al. (2012) considered a large number of experimental studies on the effects of software tools in teaching published over a decade. Many studies reviewed in that work compare working conditions with and without auxiliary tools, showing positive results in favor of using software tools to improve traditional learning methods. Today, it is common to use different learning support tools in computer engineering courses during laboratory exercises. In courses more oriented towards studying hardware and abstract systems, such tools are often software simulators (Djordjevic et al., 2005; Jovanovic et al., 2012; Stamenkovic and Jovanovic, 2024) whose task is to simulate the system being studied. In software courses, tools for visual representation of algorithms are often used (Thakur et al., 2011; Stamenkovic et al., 2023). The field of designing tools for the visual representation of algorithms has a long tradition, and many such software systems have been realized so far. Interpreting machine learning models is currently a popular topic in the information visualization community, and the results demonstrate that insights from machine learning models can lead to better predictions and improved reliability of results (Chatzimparmpas et al., 2020). With the growing popularity of machine learning, there is also a greater need for systems for the visual representation of algorithms in this field.

Support Vector Machine (SVM) is one of the fundamental algorithms in the field of machine learning, and understanding it is crucial for a deep comprehension of the theoretical and practical aspects of this field. Support Vector Machines (SVM) are powerful machine learning algorithms used for classification, regression, and anomaly detection (Cortes and Vapnik, 1995; Bennett and Campbell, 2000; Schölkopf and Smola, 2002). The idea of the method is to find a separating hyperplane in the vector space where the data are represented so that all data from the same class are on the same side of the plane, as shown in Figure 1. SVM finds the optimal separating hyperplane, which is the plane with the maximum margin. The margin represents the width of the separation between classes that should be maximized. SVM algorithms are characterized by their ability to effectively work in high-dimensional spaces and with data that are not linearly separable due to the use of kernel functions. Since SVM effectively classifies multidimensional data of various types, it is successfully applied in various fields, including biosciences for analyzing biological processes, medicine for analyzing medical images, DNA analysis, and predicting population structure. It is also widely used for classifying hate speech on social networks. Due to their wide application and significance, SVM algorithms are an integral part of university course literature on machine learning.

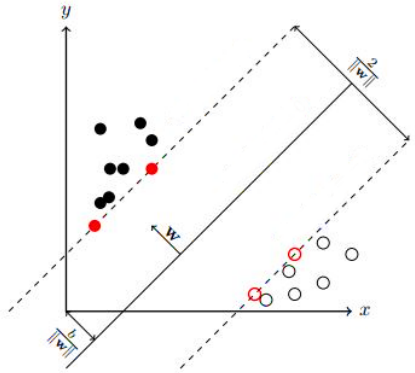

Figure 1. An example showing the optimal hyperplane with the maximum margin that separates data into two classes

The aim of this study is the development of a web application that serves as an educational tool for learning and visualizing the principles and applications of SVM algorithms. The application is developed using modern web technologies such as HTML-CSS, JavaScript, and Java EE technologies, enabling interactive and visual learning. The use of these technologies contributes to the flexibility and accessibility of the application, making it available to a wide range of users, including students and researchers.

The paper is systematized into seven sections. Following the introduction, the second section analyzes relevant literature to present existing solutions in this field. The third section describes the system architecture with an emphasis on modular design, integration with databases, and how SVM processing is performed within the application. A detailed description of the user interface, with examples of how users interact with the application, including the visualization of SVM models, is presented in the fourth section. The fifth section presents a case study that precisely describes the strategy for learning the SVM algorithm using the developed auxiliary tool. The sixth section provides a detailed description of the conducted experiment and the results of the tool evaluation, along with a discussion. Conclusions and suggestions for future work are presented in the seventh chapter.

Literature Review

Learning support tools are part of the modern educational process. The main task of these systems is to enable users to master the material more easily. Nowadays, there are a large number of diverse learning support systems. Depending on the field of application and the level of knowledge required to use the system, the technology applied for system implementation and the didactic methods that the systems implement differ. For supporting the learning of algorithms in the fields of machine learning and artificial intelligence, visualization and simulation tools for algorithm operation are successfully used. The perception of reality is fundamentally visual, and in many cases, humans use symbolic processing, visual diagrams, and other forms of imaginative processes to gain intuition about what, in its formal aspect, has an abstract structure (Díaz, Dormido, and Rivera, 2015). Today, it is widely accepted that visualizing information about the internal workings of complex algorithms can contribute to better understanding and offer solutions for better and more interpretable machine learning models (Sacha et al., 2017). Below, some of the systems developed as tools to support learning algorithms in the fields of machine learning and artificial intelligence will be presented.

Principal Component Analysis (PCA) is the most commonly used procedure in exploratory data analysis and in machine learning for predictive models. However, due to its complexity, the mechanisms and results of this operation are difficult for users to understand. To help students better understand and use PCA, Jeong et al. (2009) developed a system called iPCA (interactive PCA), which visualizes the results of principal component analysis using multiple different views and a rich set of user interactions. The interactive iPCA system with an example is shown in Figure 2. As can be seen, the visualization is realized at a high level through multiple different views, and there are options for configuring the parameters of the operation.

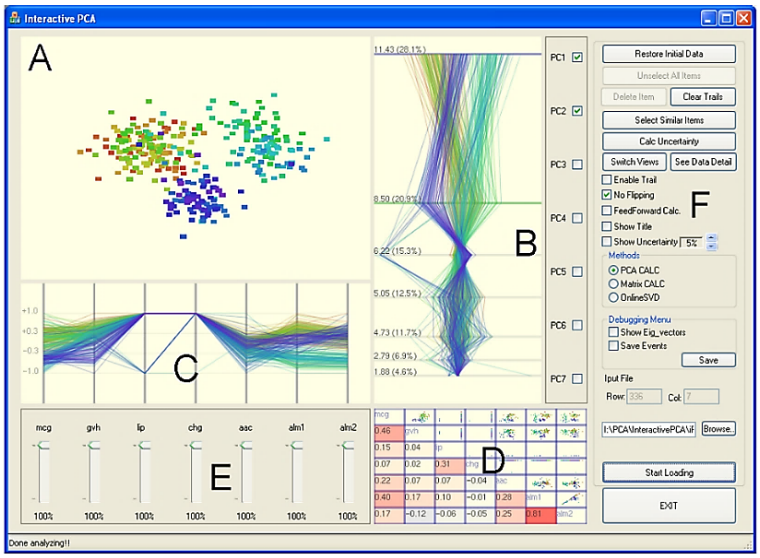

Figure 2. Interactive iPCA System: (A) Projection View; (B) Characteristic Vectors Overview; (C) Data View; (D) Correlation Overview; (E) Dimension Control; (F) Adjustment Options.

Mayfield and Rosé (2010) focus on the development of an interactive tool that aids in the analysis of errors in the text retrieval process. The proposed tool is designed to enable users, especially students and researchers, to more easily identify and understand errors that occur during text classification. The paper emphasizes the importance of understanding errors in machine learning models and presents a user interface (Figure 3) that allows for visual exploration and analysis of errors. The proposed tool is useful for education because it provides deeper insight into the decision-making processes of machine learning models.

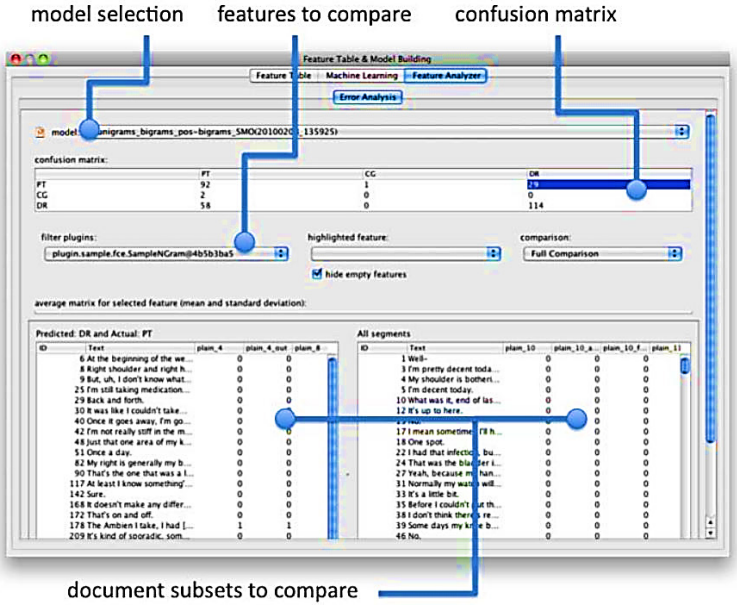

Figure 3. Interface of the error analysis tool in the text retrieval process

Kim et al. (2021) presented an interactive framework for demonstrating symbolic regression that allows users not only to perform general configuration but also to control the system during training. The demonstration system for symbolic regression enables user interaction with the DSR (deep symbolic regression) algorithm. The interface provides real-time visualization and diagnostics to help the user monitor and control the algorithm. The interactive platform includes visualization of top-performing mathematical expressions and real-time algorithm diagnostics. This software system consists of a core algorithm (DSR) running in the background, a framework for algorithm visualization, and an adaptive user interface for data

loading and real-time configuration (Figure 4). The interactive platform is web-based, meaning that the user interacts with the algorithm through a web browser while execution is performed on the server side.

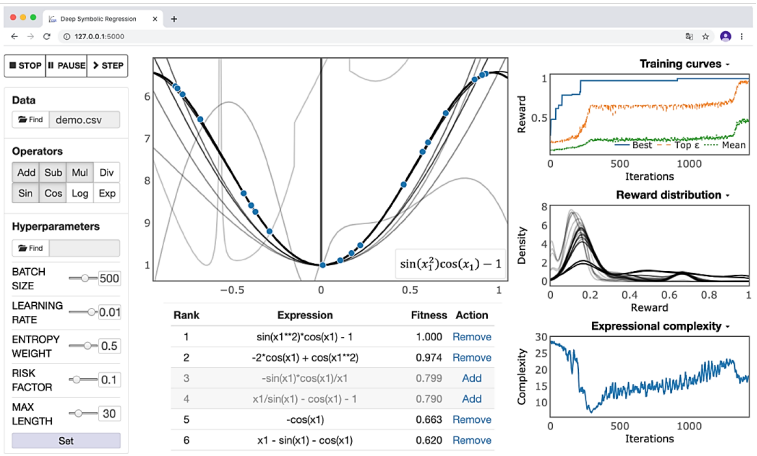

Figure 4. User interface of the symbolic regression demonstration tool

NNeduca (Jovanovic et al., 2023) is a software environment designed for teaching and learning the principles of artificial neural networks. The tool was created with the intention of improving both teaching methods and learning outcomes. The proposed system can be used to design a neural network with any number of layers and an arbitrary number of neurons in each of those layers. The development process of the neural network is flexible and configurable, the technical features and appearance of the environment are customizable, and the choice of transfer function for each layer is optional. All processes in neural networks are presented visually, and the results are illustrated using appropriate graphs. Neural networks can be applied to both regression and classification tasks, making this tool applicable for various calculations in many fields. The system focuses on students for whom studying neural networks is a challenging task, providing them with the opportunity to experiment with different types of neural networks, adjust parameters, train networks on various datasets, and visually analyze the results through an intuitive user interface.

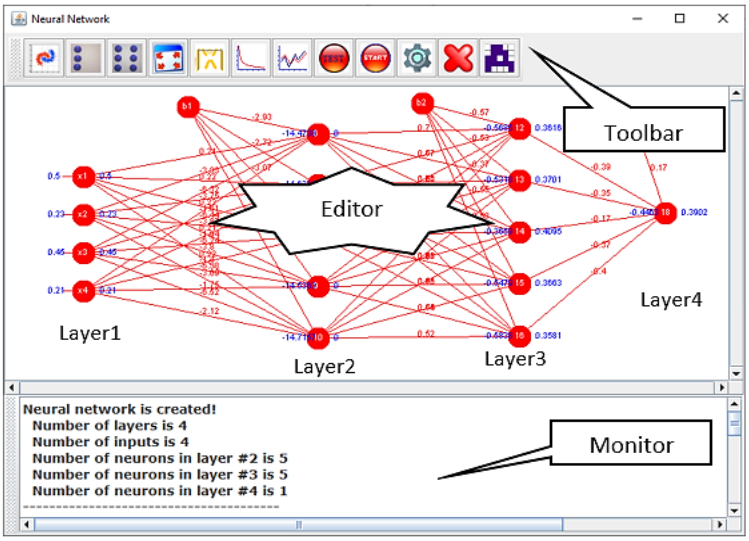

Figure 5. User interface of the NNeduca software environment

No tool developed at any university for assisting in the learning of SVM algorithms was found in the literature search. This fact served as an additional motivation for the development of such a system.

System architecture

Figure 6 shows the architecture of the software system. The basic components of the system are:

1. Frontend (HTML, CSS, JavaScript):
   - HTML, CSS, JavaScript: The user interface allows the input of parameters for the SVM model.
   - AJAX: Enables asynchronous data sending and receiving of results without refreshing the page.
   - Visualization System.
2. Backend:
   - Java EE**:** Enterprise application for handling user requests and business logic.
   - SvmServlet: Receives AJAX requests with parameters, forwards them to SvmProcessorBean, and returns results as JSON.
   - SvmProcessorBean: Processes data, trains the SVM model using libsvm, and returns results to the servlet.
3. Libraries:
   - libsvm: Library for implementing SVM algorithms for model training.
   - D3.js: Visualizes results through interactive graphs.

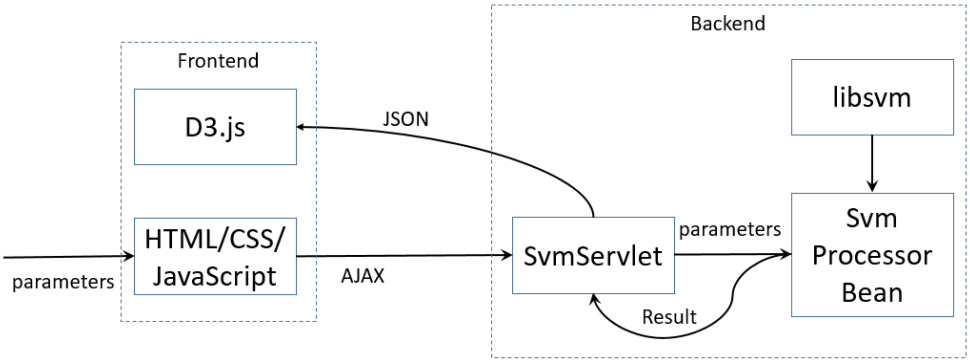

Figure 6. System architecture

The user configures parameters for the SVM model via the web interface (frontend) by entering values such as the type of SVM, kernel function, C, gamma, and other parameters. Once the parameters are set, an asynchronous HTTP request is sent to the SvmServlet (backend) using AJAX, which provides a smooth user experience without requiring page reloading. The SvmServlet receives these parameters and forwards them to the SvmProcessorBean component. The SvmProcessorBean uses the libsvm library to train the SVM model with the given parameters. After processing, the SvmProcessorBean returns the results to the SvmServlet, which then sends a JSON response with the results back to the frontend via AJAX. The results are then displayed on the frontend using D3.js to create interactive graphs, which allow the user to visualize the performance of the trained model.

This architecture allows users to interactively experiment with SVM models using the flexibility of AJAX requests for quick processing and result visualization. All layers of the system are clearly defined and interconnected, allowing for efficient and clear handling of user requests and result visualization.

Practical work with the system

The page at https://ml.pr.ac.rs/svm.html provides an interactive environment for working with Support Vector Machine (SVM) models (Figure 7). It allows users to experiment with different types of SVMs (C-SVC, nu-SVC, one-class SVM, epsilon-SVR, nu-SVR) and kernel functions (Linear, Polynomial, Radial Basis Function, Sigmoid).

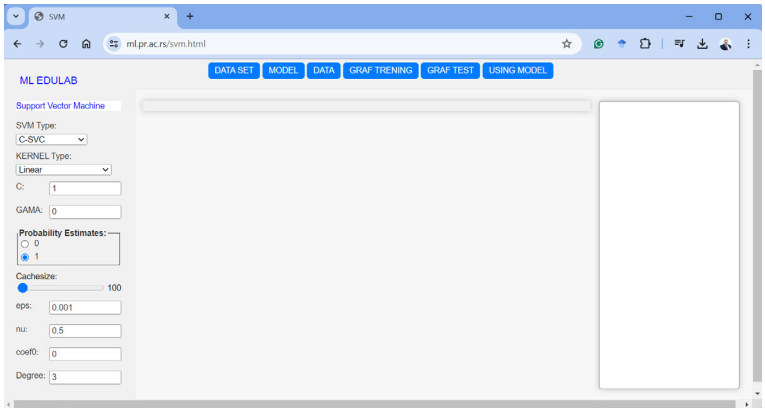

Figure 7. The main window of the application

Users can configure parameters for the SVM model via the web interface. The data are then loaded into the DATA SET module. The data is in Excel format. It is necessary to specify which columns in the Excel table represent the input data (in this case, 0,1), while the remaining columns represent the output data. In addition, the percentage of data to be used for testing must be specified (in this case, 10%). After that, by clicking the LOAD DATA button, the data will be loaded (Figure 8).

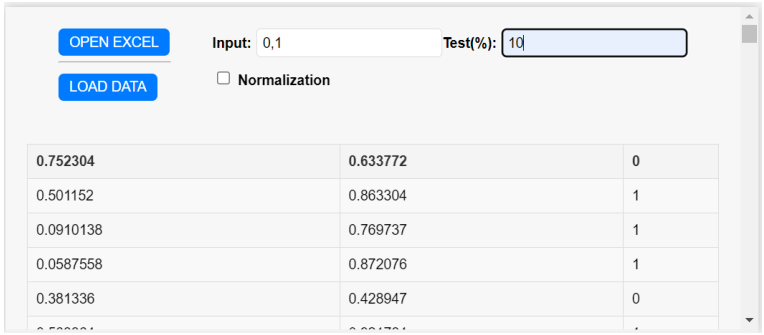

Figure 8. Data set module

The data and parameters are then sent asynchronously via an HTTP request to the backend (SvmServlet) using AJAX. On the backend, the request is processed. SvmServlet receives the parameters and forwards them to SvmProcessorBean.

SvmProcessorBean is an Enterprise JavaBean (EJB) component used to implement business logic in the SVM visualization application. Its main functions include processing parameters, training the SVM model, and generating results. SvmProcessorBean returns the results to SvmServlet.

SvmServlet is a Java Servlet component that plays a key role in the communication between the frontend and the backend of the application. Its main functions include receiving parameters, forwarding parameters, processing results, and sending the response back to the frontend. SvmServlet sends a JSON response with the results back to the frontend via AJAX. The results are displayed on the frontend using D3.js to create interactive graphs (Figure 9).

Different types of SVMs allow the algorithm to be applied to a wide range of tasks, from classification to regression and anomaly detection, providing flexibility and efficiency in solving various machine learning problems.

The user selects one of several types of SVMs. C-SVC (C-Support Vector Classification) is a classic SVM for classification. It uses regularization with the parameter C, which controls the balance between maximizing the margin and minimizing classification errors. An alternative is nu-SVC (nu-Support Vector Classification) with the parameter v (nu), which controls the upper bound on the fraction of errors and support vectors, providing a more flexible approach to controlling model complexity. A one-class SVM is used for anomaly detection. Training is performed only on positive examples, and the model identifies whether new data belongs to the same class. Epsilon-SVR (Epsilon-Support Vector Regression) is a regression SVM for predicting continuous values. The parameter epsilon controls the width of the region around the

function within which errors are not penalized. Nu-SVR (nu-Support Vector Regression) is a regression SVM with parameters v (nu) and epsilon for better control over the number of support vectors and error tolerance.

Kernel functions are mathematical functions that transform data into a higher dimension so that the SVM algorithm can more easily find a separating hyperplane for classification or regression. They enable the SVM to model nonlinear boundaries between classes. In the application, the following kernel functions can be used:

1. **Linear**: Uses a linear kernel, the simplest function for separating data. It is efficient for high-dimensional data.
2. **Polynomial**: Uses a polynomial kernel that can model nonlinear relationships. Parameters include the degree of the polynomial and the coefficient.
3. **Radial Basis Function (RBF)**: The most commonly used kernel for SVM. The parameter gamma controls the width of the RBF and adjusts the flexibility of the model.
4. **Sigmoid**: Uses a sigmoid kernel, which is similar to the neuron activation function. It can model nonlinear relationships and is used in support vector networks.

Within the SVM (Support Vector Machine) model, there are several parameters that can be adjusted to optimize performance. Understanding these parameters is crucial for effectively using SVM algorithms in machine learning. Below is a detailed description of the most important adjustable parameters for SVM models, explaining their function and impact on the model:

1. **C (Regularization Parameter)**: Controls the trade-off between achieving a low training error and a low testing error, which is achieved by maximizing the margin.
2. **Gamma (Parameter for RBF, Polynomial, and Sigmoid Kernels)**: Controls the width of the Gaussian curve in the RBF kernel. Lower gamma values create wider curves (smoother boundaries), while higher values create narrower curves (more precise boundaries).
3. **Cachesize**: Determines the amount of memory allocated for caching the kernel matrix. Larger values can speed up the computation of the kernel matrix but require more memory.
4. **Epsilon (Parameter for Epsilon-SVR)**: Controls the width of the epsilon-tube within which errors are not penalized. Larger values of epsilon make the model more tolerant of errors within the epsilon-tube.
5. **Nu (Parameter for nu-SVC and nu-SVR)**: Controls the upper bound on the fraction of errors and support vectors. It allows direct control of model complexity and error tolerance.
6. **Coef0 (Parameter for Polynomial and Sigmoid Kernels)**: A constant added in the polynomial and sigmoid kernels. It affects the shape and complexity of the model when using nonlinear kernel functions.
7. **Degree (Degree of Polynomial for Polynomial Kernel)**: Determines the degree of the polynomial in the polynomial kernel. Higher values make the model more complex and allow it to capture nonlinear patterns in the data.

Users can adjust these parameters to optimize the performance of their SVM models for specific problems. By experimenting with different values of these parameters, it is possible to achieve a balance between model accuracy and its generalization (Figure 9).

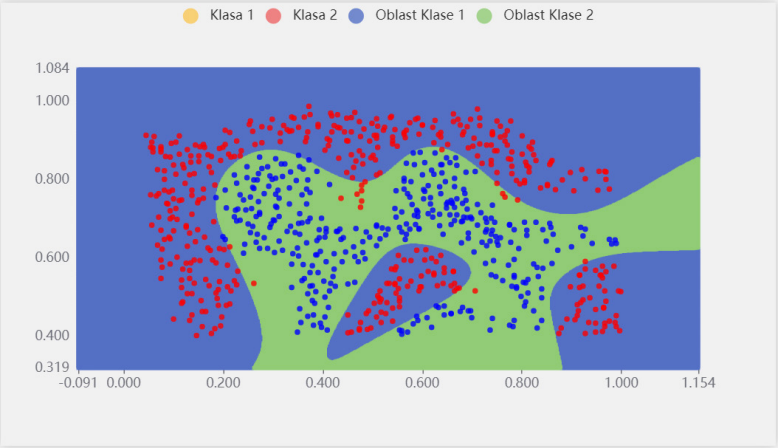

Figure 9. SVM model results visualization

The obtained model can be used for classification by clicking the USING MODEL button (Figure 10).

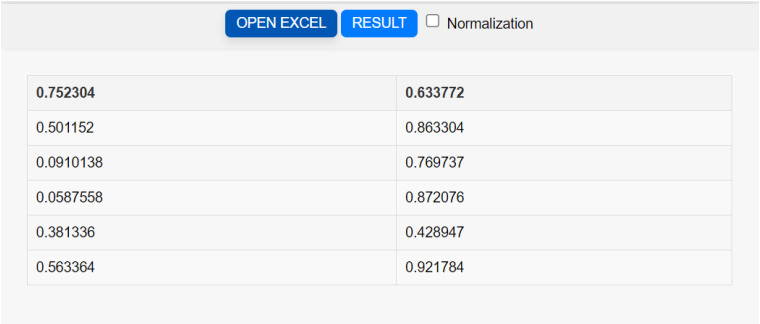

Figure 10. Test data

The data should be in Excel format, and the classification results should be obtained in graphical form (Figure 11).

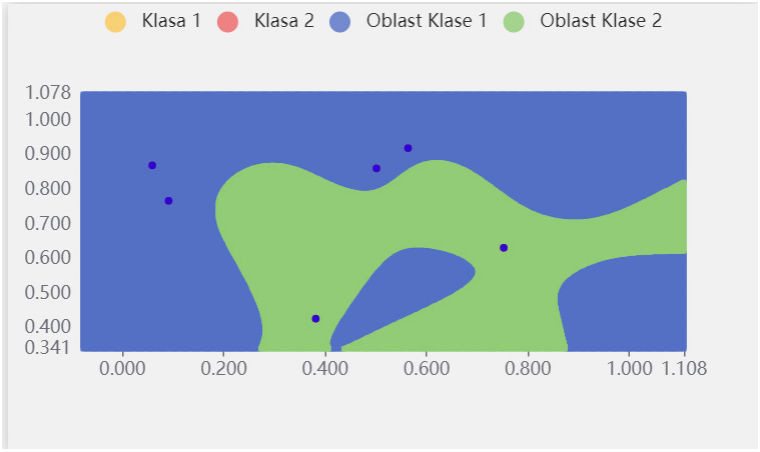

Figure 11. Classification results

This application significantly contributes to education through its interactive and practical approach. By providing users with the ability to directly experiment with different SVM models and parameters, the application enables a deeper understanding and efficient application of theoretical knowledge in practice. With its rich visualizations and flexibility, the application is an invaluable tool for students, researchers, and professionals in the field of machine learning.

Case study

The presented web-based tool is suitable for supporting the learning/teaching of SVM algorithms in all undergraduate courses that cover machine learning algorithms. The alignment of this software system with the literature used in theoretical teaching is of exceptional importance due to its usability and easier integration within laboratory exercises. Given that the tool covers different types of SVMs as well as all kernel functions, it will be suitable for many undergraduate curricula. For learning complex theoretical constructs, the active learning method is recommended (Silva et al., 2019; Pradono et al., 2013), and this method is also applied when learning SVM through the auxiliary tool. Collaboration among students also has a significant impact on learning outcomes (Guo et al., 2020). The learning outcomes of applying the realized educational tool include: (1) understanding how the SVM algorithm works; (2) identifying its advantages and disadvantages compared to other classification algorithms; (3) the ability to recognize problems for which the SVM algorithm is most suitable; (4) acquiring the skill to correctly select the appropriate type of SVM, kernel function, and properly configure it for the most efficient solution to the given problem.

For learning this algorithm, 4 hours of theoretical and 4 hours of practical instruction are recommended. The teaching should be based on the traditional paradigm of theory-examples-exercises. Thus, the strategy for learning the SVM algorithm by applying the developed auxiliary tool is as follows:

- Theoretical concepts are first presented for each topic during the theoretical lectures. (Auxiliary tools for visualization and simulation are powerful teaching aids only when used in conjunction with traditional teaching (Taher and Khan, 2014).
- Laboratory exercises involve a more detailed explanation of the topics covered in lectures using tools to visually represent the presented concepts. To prepare for the exercise, students should read the relevant lecture material and textbooks, as well as the accompanying laboratory material. Each exercise consists of two components:
    - Demonstration of Examples: At the beginning of the laboratory exercise, the instructor demonstrates examples using the educational tool. The instructor should utilize the system's potential to engage students, as otherwise, the exercises would merely represent a different approach to theoretical presentation. In line with the principles of active learning, an interactive approach to the demonstration involves engaging students so that they are not just passive observers. The easiest way to include students during the demonstration is by frequently asking "what if" questions and prompting them to predict the results.
    - Laboratory Assignment: After the demonstration is completed, students receive appropriate laboratory tasks that they complete independently.

By applying this strategy, students show a high level of motivation and actively participate in the entire process and discussion. Another advantage of the presented web tool is that once students become familiar with it, they can use it independently after class as an additional resource for solving homework assignments and seminar papers.

Experiments and evaluation

This study employs a descriptive research design that relies on quantitative data. The aim of the study is to conduct a quantitative assessment of the contribution to learning through the use of the developed web-based tool. The research instruments include a controlled experiment to test the efficiency of the software tool and a survey to gather user feedback on satisfaction. The objective of the survey is an objective assessment of the tool's usability.

For the experiment, a purposive (nonprobabilistic) sampling technique was applied (Vehovar et al., 2016). The selected sample consisted of students who had not yet attended the Machine Learning course and who had no experience with using the SVM algorithm but had general knowledge about the machine learning concept. A total of 38 undergraduate students from the Faculty of Technical Sciences in Kosovska Mitrovica and the Toplica Academy of Vocational Studies in Prokuplje were selected. For the experiment, the selected sample of students was divided into two groups: control and experimental. The allocation of students to the groups was random. Both the experimental and control groups consisted of 19 students each.

Experiment methodology

The experiment consisted of three phases: pre-testing, training, and post-testing (Dugard and Todman, 1995). The pre-test questionnaire was designed with questions related to general information about classification algorithms and the SVM algorithm. This questionnaire assesses the prior knowledge of all participants on the given topic. It contained eight multiple-choice quiz questions and six questions requiring standard written answers. Quiz questions were scored with 5 points each, while the standard questions were scored with 10 points each (or less if the answer was partial).

The post-test questionnaire was similar in structure to the pre-test questionnaire, but the questions focused solely on the SVM algorithm. The post-test was used to assess the knowledge gained after the training phase. The training process for the control group involved traditional teaching, while the learning strategy for the experimental group is described in detail in the Case Study section. Traditional teaching was also based on the theory-examples-exercises paradigm. For each topic, theoretical concepts were first presented during lecture sessions. Laboratory exercises were designed to provide a more detailed

explanation of the topics covered in lectures, without the use of an educational web tool, but rather through practical examples applying the hands-on tasks pedagogical method. The instructors and examiners were the same for both groups.

Survey methodology

In addition to the controlled experiment, it was necessary to conduct a survey on the advantages and significance of the auxiliary learning system. This process aimed to assess students' experiences with the alternative learning method using the auxiliary tool and their perception of the software system's effectiveness. For this purpose, the standardized System Usability Scale (SUS) was used. SUS is a quick and reliable technique for measuring system usability (Brooke, 1996). It consists of a standard questionnaire with 10 questions. For each question, respondents select a statement indicating their level of agreement or disagreement with the statement on a 5-point Likert scale. SUS is generally administered after the respondent has had the opportunity to use the system being evaluated but before any other testing or discussion takes place. If a respondent feels unable to answer a particular item, they should mark the central point on the scale.

Experiment results

For this experiment, quantitative data were collected based on the established methodology, and descriptive statistics were used for their analysis. To reliably examine the existence of statistically significant differences in the achieved results between students in the control and experimental groups, the ANCOVA statistical method was used. The analyses were conducted using SPSS software (version 25). ANCOVA (Analysis of Covariance) is used to determine the presence of significant differences between two or more independent groups concerning a dependent variable. By isolating the effect of a categorical independent variable on the dependent variable, researchers can draw more accurate and reliable conclusions from their data. ANCOVA seeks differences in adjusted mean values.

Individual differences between students in academic abilities can significantly impact learning outcomes. Even within the same group, there can be significant variations in students' prior knowledge. This variation can obscure the true impact of the applied method on the learning outcomes or results. By including pre-test results as a covariate in the ANCOVA model, it is possible to more clearly and precisely understand whether students' success in the post-test was due to the applied teaching method. In this statistical model, the post-test result is taken as the dependent variable, the teaching method as the categorical factor, and the pre-test result as the covariate.

- H_0 - Null Hypothesis: There is no significant difference in the adjusted mean values of the post-test results between the control and experimental groups.
- H_1 - Alternative Hypothesis: There is a significant difference in the adjusted mean values of the post-test results between the control and experimental groups.

The achieved post-test results of the students are shown in Figure 12.

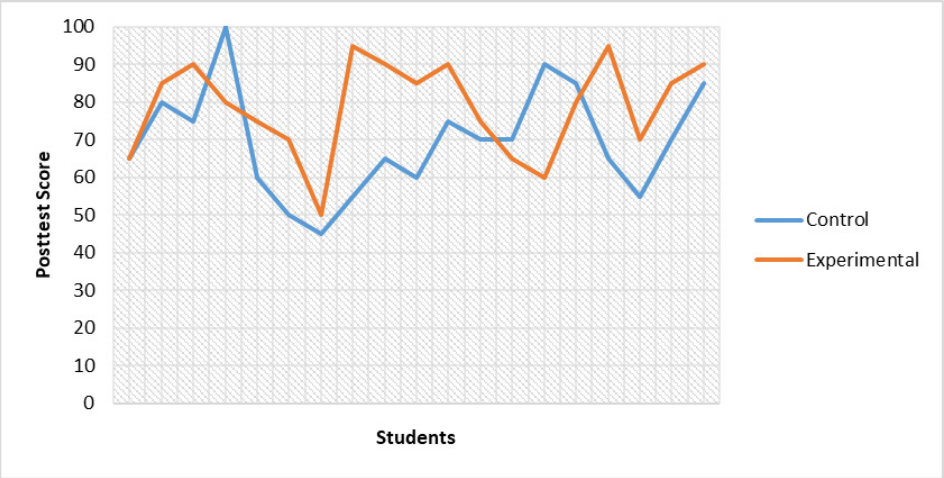

Figure 12. Achieved post-test results

Descriptive statistics for the dependent variable (post-test results) are presented in Table 1. Here, the mean values and standard deviations of the results for each group are shown.

Table 1. *Descriptive Statistics for Dependent Variable: Posttest*

Group	Mean	Std. Deviation	N
control	69,4737	14,22974	19
experimental	78,6842	12,67567	19
Total	74,0789	14,08734	38

The results of Levene's Test for Homogeneity of Variances are shown in Table 2. This test evaluates whether the variances between groups are approximately equal.

Table 2. *Levene's Test of Equality of Error Variances*[a]

Dependent Variable: Posttest

F	df1	df2	Sig.
,130	1	36	,721

a. Design: Intercept + Pretest + Group

The main part of the ANCOVA test results is shown in Table 3. This table presents the effects of the test between subjects, focusing on the level of significance concerning the independent variable.

Table 3. *Tests of Between-Subjects Effects*

Dependent Variable: Posttest

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Corrected Model	892,054[a]	2	446,027	2,420	,104	,121
Intercept	3024,956	1	3024,956	16,413	,000	,319
Pretest	86,133	1	86,133	,467	,499	,013
Group	868,456	1	868,456	4,712	,037	,119
Error	6450,710	35	184,306			
Total	215875,000	38				
Corrected Total	7342,763	37				

a. R Squared = ,121 (Adjusted R Squared = ,071)

A comparison of the mean values of post-test results between the control and experimental groups is presented in Table 4.

Table 4. *Pairwise Comparisons*

Dependent Variable: Post

(I) Group	(J) Group	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
control	experimental	-9,674*	4,456	,037	-18,721	-,627
experimental	control	9,674*	4,456	,037	,627	18,721

Based on estimated marginal means

*. The mean difference is significant at the ,05 level.

b. Adjustment for multiple comparisons: Bonferroni.

Survey results

The SUS provides a score that represents a composite measure of the overall usability of the system under study. To calculate the score, first sum the contributions from each item. Contributions range from 0 to 4. For questions 1, 3, 5, 7, and 9, the contribution is the scale position minus 1. For items 2, 4, 6, 8, and 10, the contribution is 5 minus the scale position. The sum of contributions is then multiplied by 2.5 to obtain the overall score. SUS scores range from 0 to 100. Bangor et al. (2008) experimentally demonstrate that software with SUS scores below 50 is "unacceptable," while scores above 70 indicate "acceptable" usability. Scores between 50 and 70 are considered "marginally acceptable."

Following the completion of the post-test, students in the experimental group who had the opportunity to use the educational tool received the SUS questionnaire. The results of this survey, processed using the described technique, are presented in Table 5.

Table 5. *Responses to Individual System Usability Scale Statements*

	Statement	M	SD
1	I think that I would like to use this system frequently.	3,47	0,84
2	I found the system unnecessarily complex.	2,63	0,90
3	I thought the system was easy to use.	3,37	0,68
4	I think that I would need the support of a technical person to be able to use this system.	2,89	0,99
5	I found the various functions in this system were well integrated.	3,63	0,68
6	I thought there was too much inconsistency in this system.	2,74	0,81
7	I would imagine that most people would learn to use this system very quickly.	3,53	0,70
8	I found the system very cumbersome to use.	2,58	1,02
9	I felt very confident using the system.	3,42	0,84
10	I needed to learn a lot of things before I could get going with this system.	2,32	1,06

Discussion

The results of the post-test show that the average score of students in the experimental group was higher than that of the control group. This is clearly seen in Table 1, which displays the descriptive statistics, revealing an average score of 78.68 for the experimental group compared to 69.47 for the control group. However, it was crucial to prove that this improvement in the experimental group's scores was statistically significant. Therefore, Levene's test was conducted initially to test for equality of variances. The results of this test, which are shown in Table 2, with Sig. = 0.721, indicate that there was no significant difference in variance between the control and experimental groups because the significance level exceeded the threshold of 0.05. Thus, based on the pre-test results, it was established that the formed groups did not statistically differ before the start of the training.

Finally, the results of the ANCOVA analysis (Table 3) demonstrated a statistically significant impact of using the instructional method with the supplementary software tool on the post-test results. With F (1, 35) = 4.712 and a p-value of 0.037 < 0.05, the null hypothesis (H0) is rejected, supporting the alternative hypothesis (H1) that there is a significant difference in adjusted mean post-test scores between the control and experimental groups.

Table 4 compares the mean post-test scores, showing a difference of 9.67% between the mean scores of the experimental and control groups.

The usability assessment of the system was conducted using the SUS technique, and the results of this survey are presented in Table 5. This table displays the mean scores for each question. By summing these mean scores, a total contribution of 30.58 was obtained. As described, this contribution is multiplied by 2.5 to yield a total SUS score of 76.45. Therefore, since the final score is >70, the implemented tool is considered acceptable on the SUS usability scale.

Finally, it is important to highlight potential threats to the validity of this empirical assessment.

These threats can be classified as internal, construct, and external validity.

Internal validity threats primarily concern causal questions regarding the presented results. It's worth noting that none of the students who participated in the experiment had prior knowledge of the study preparation. However, if anyone somehow obtained information about the experiment, it could have led to bias in evaluation. Since collaboration among students is important for the learning process, it cannot be accurately assessed whether one group of students has better or worse communication and teamwork skills. To mitigate this threat, both groups had the same instructors.

Construct validity threats usually relate to potential errors in assessment. This type of threat can be influenced by the evaluation method used. For evaluating this tool, a pretest-posttest method was applied. Different results may have been obtained using different assessment methods. To reduce the impact of this threat on the assessment of tool effectiveness, the standard SUS technique, which is widely accepted for evaluating usability levels, was applied.

External validity threats concern the extent to which findings from the experiment can be generalized and how relevant they are to other students beyond this study. To mitigate this threat, students from two different higher education institutions participated in the experiment. However, for generalization, one should consider the sample size of students, geographic distance between participating universities, socio-cultural differences, and the complexity of tests assigned to students. Thus, the results presented in this study cannot be generalized until more data are obtained from other empirical evaluations involving different universities and students.

Conclusion

Acquiring skills and knowledge in machine learning is a crucial aspect of academic education for IT engineers. Teaching complex algorithms in this field should consider the use of innovative tools and methodologies developed to enhance and supplement the learning and teaching processes. The web application for learning Support Vector Machines (SVM) represents a significant advancement in computer engineering education, providing intuitive and interactive ways to understand and apply SVM algorithms. Developed using state-of-the-art web technologies, the proposed application promises to become a valuable resource for students, educators, and researchers in the field of machine learning. This study analyzes the impact of implementing a new interactive learning method based on the developed web application on learning outcomes.

As a result of using the visual software tool in machine learning classes, student engagement levels have increased. Visualization of the complex SVM algorithm has enabled students to better understand the workings of the algorithms, down to finer details. These claims are substantiated by a conducted experiment aimed at quantitatively assessing the learning contributions using the developed web-based tool. Research instruments included a controlled experiment to verify the efficiency of the software tool and a survey to collect user satisfaction feedback. The experiment results show that students who used the tool in their learning (experimental group) have a better understanding of the taught topics compared to students whose learning was based solely on traditional methods (control group). Through the application of the ANCOVA statistical method, the study reliably demonstrated a statistically significant difference in outcomes between the control and experimental groups.

The usability assessment of the proposed system was evaluated using the SUS technique. Based on the final survey results, it can be stated that the implemented tool was rated as acceptable on the SUS usability scale.

In future work and research, a broader assessment of the tool with a greater number of students from various universities should be considered. In addition, future work will be valuable to analyze the advantages and limitations of the developed web application compared to similar existing tools. Further enhancements to the proposed system could involve the addition of modules for learning other machine learning algorithms.

Acknowledgments

The authors would like to express their gratitude to the respondents who participated in the research and the reviewers whose constructive suggestions significantly enhanced the quality of this work.

Author Contributions

Conceptualization, N.J., S.J. and S.S.; methodology, N.J., S.J. and S.S.; software, N.J. and S.J.; formal analysis, S.S., D.M. and N.S.; writing—original draft preparation, N.J. and S.S.; writing—review and editing, S.J., D.M. and N.S. All authors have read and agreed to the published version of the manuscript.

Conflict of interests

The authors declare no conflict of interest.

References

Abadi, M., Agarwal, A., Barham, P. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://doi.org/10.48550/arXiv.1603.04467

Bangor, A., Kortum, P.T. and Miller, J.T., (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, *24*(6), 574-594. https://doi.org/10.1080/10447310802205776

Bennett, K.P. and Campbell, C., (2000). Support vector machines: hype or hallelujah?. *ACM SIGKDD explorations newsletter*, 2(2), 1-13. https://doi.org/10.1145/380995.380999

Brooke, J. (1996). SUS: A "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability evaluation in industry,* 189–194. London: Taylor & Francis.

Çağlayan, C., (2019). Comparison of the Code-based or Tool-based Teaching of the Machine Learning Algorithm for the First-Time Learners. *In 2019 1st International Informatics and Software Engineering Conference* (UBMYK), 1-3. IEEE. https://doi.org/10.1109/UBMYK48245.2019.8965519

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 1-27. https://doi.org/10.1145/1961189.1961199

Chatzimparmpas, A., Martins, R. M., Jusufi, I., & Kerren, A. (2020). A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization*, *19*(3), 207-233. https://doi.org/10.1177/1473871620904671

Chollet F., et al. (2015). Keras. https://keras.io

Cortes, C. and Vapnik, V., (1995). Support-vector networks. *Machine learning*, *20*, 273-297. https://doi.org/10.1007/BF00994018

DÍaz, J.M., Dormido, S. and Rivera, D.E., (2015). *Interactive Education for Time-Domain Time Series Analysis using ITTSAE*. IFAC-PapersOnLine, 48(28), 751-756. https://doi.org/10.1016/j.ifacol.2015.12.220

Djordjevic, J., Nikolic, B. and Milenkovic, A., (2005). FlexibleWeb-Based Educational System for Teaching Computer Architecture and Organization. *IEEE Transactions on Education*, 48(2), 264-273. https://doi.org/10.1109/TE.2004.842918

Dugard, P. and Todman, J., (1995). Analysis of pre□test□post□test control group designs in educational research. *Educational Psychology*, *15*(2), 181-198. https://doi.org/10.1080/0144341950150207

Guo, P., Saab, N., Post, L.S. and Admiraal, W., 2020. A review of project-based learning in higher education: Student outcomes and measures. *International journal of educational research*, *102*, 101586. https://doi.org/10.1016/j.ijer.2020.101586

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), 10-18. https://doi.org/10.1145/1656274.1656278

Hazzan, O., and Mike, K. (2023). *Guide to Teaching Data Science: An Interdisciplinary Approach*. Springer Nature.

Jeong, D. H., Ziemkiewicz, C., Fisher, B., Ribarsky, W., & Chang, R. (2009). iPCA: An interactive system for pca□based visual analytics. *In Computer Graphics Forum* (Vol. 28, No. 3, pp. 767-774). Oxford, UK: Blackwell Publishing Ltd. https://doi.org/10.1111/j.1467-8659.2009.01475.x

Jovanović, N., Popović, R., Marković, S. and Jovanovic, Z., (2012). Web laboratory for computer network. *Computer Applications in Engineering Education*, 20(3), 493-502. https://doi.org/10.1002/cae.20417

Jovanović, N., Stamenković, S., and Jovanović, S. (2023). NNeduca: a software environment to teach artificial neural networks, *Comput. Appl. Eng. Educ*. Volume 31, Issue 5, 1447–1464. https://doi.org/10.1002/cae.22655

Kim, J. T., Kim, S., & Petersen, B. K. (2021). An interactive visualization platform for deep symbolic regression. *In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence,* 5261-5263. https://doi.org/10.24963/ijcai.2020/763

Mayfield, E., & Rosé, C. (2010). An interactive tool for supporting error analysis for text mining. In *Proceedings of the NAACL HLT 2010 Demonstration Session,* 25-28.

Mühlbacher, T., Piringer, H., Gratzl, S., Sedlmair, M. and Streit, M. (2014). Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE transactions on visualization and computer graphics*, 20(12), 1643-1652. https://doi.org/10.1109/TVCG.2014.2346578

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information*

processing systems, 32. https://doi.org/10.48550/arXiv.1912.01703

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., (2011). Scikit-learn: Machine learning in Python. *The Journal of machine Learning research*, *12*, 2825-2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page

Pradono, S., Astriani, M.S. and Moniaga, J., (2013). A method for interactive learning. *International Journal of Communication & Information Technology*, 7(2), 46-48. https://doi.org/10.21512/commit.v7i2.583

Rutten, N., Van Joolingen, W. R., and Van der Veen, J. T., (2012). The learning effects of computer simulations in science education. *Computers & Education, Elsevier*, 58(2012), 136-153. https://doi.org/10.1016/j.compedu.2011.07.017

Sacha, D., Sedlmair, M., Zhang, L., Lee, J.A., Peltonen, J., Weiskopf, D., North, S.C. and Keim, D.A. (2017). What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, *268*, 164-175. https://doi.org/10.1016/j.neucom.2017.01.105

Schölkopf, B. and Smola, A.J., (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond. *MIT press*.

Silva, W., Steinmacher, I. and Conte, T. (2019). Students' and instructors' perceptions of five different active learning strategies used to teach software modeling. *IEEE Access*, *7*, 184063-184077. https://doi.org/10.1109/ACCESS.2019.2929507

Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., Bona, F.D., Binder, A., Gehl, C. and Franc, V., (2010). The SHOGUN machine learning toolbox. *The Journal of Machine Learning Research*, *11*, 1799-1802. https://doi.org/10.5446/19980

Stamenković, S., Jovanović, N. (2024). A Web-based Educational System for Teaching Compilers, *IEEE Transactions on Learning Technologies*. Vol. 17, 143-156. https://doi.org/10.1109/TLT.2023.3297626

Stamenković, S., Jovanović, N., Vasović, B., Cvjetković, M. and Jovanović, Z., (2023). Software tools for learning artificial intelligence algorithms. *Artificial Intelligence Review*, 1-30. https://doi.org/10.1007/s10462-023-10436-0

Taher, M., and Khan, A., (2014). Impact of Simulation-based and Hands-on Teaching Methodologies on Students' Learning in an Engineering Technology Program. *Proceedings of the ASEE Annual Conference & Exposition*, 1-22. https://doi.org/10.18260/1-2--20593

Thakur, J., and Kumar, N. (2011). DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis. *International journal of emerging technology and advanced engineering*, 1(2), 6-12.

Vehovar, V., Toepoel, V. and Steinmetz, S., (2016). *Non-probability sampling*. The Sage handbook of survey methods. https://doi.org/10.4135/9781473957893.n22

Zeiler, M.D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13,* 818-833. Springer International Publishing. https://doi.org/10.48550/arXiv.1311.2901