J Euro Eco His.
ISSN : 0391-5115
Volume 7, Issue 2 (2026)
THE JOURNAL OF EUROPEAN ECONOMIC HISTORY

A Machine Learning-Based Framework for Assignment Evaluation and Feedback Generation

Kanika Jain[1], Mohan Vishal Gupta[2], Amit Kumar[3], Namit Gupta[4], Shivangi Jain[5]

College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, jainkanika1825@gmail.com[1], , mohan.vishal01@gmail.com[2], amitpanwar889@gmail.com[3], namit.computers@tmu.ac.in[4], jainshivangi614@gmail.com[5]

Abstract
This paper presents a web-based and machine learning–driven assignment submission and feedback system designed to improve academic assessment processes. The system addresses two major challenges: the manual effort involved in conducting and evaluating assignments, and the lack of detailed, meaningful feedback provided to students beyond simple scores. SmartAssign is developed using the MERN stack (MongoDB, Express.js, React.js, Node.js) integrated with Python for implementing machine learning functionalities. It supports two primary user roles—teachers and students. Teachers can upload student data via Excel sheets, create time-bound MCQ-based assessments across multiple topics, and analyze class performance through an interactive dashboard. Students can register only if their IDs are pre-listed, access active tests, submit responses within deadlines, and receive automated feedback along with their scores. The system employs a K-means clustering algorithm to generate topic-wise feedback and categorize student performance based on historical attempt data. For new users without prior data, a rule-based fallback mechanism ensures feedback generation from current attempts. Experimental testing demonstrates accurate automated scoring, reliable role-based access control, and effective ML-based feedback generation, highlighting SmartAssign's potential to enhance efficiency, scalability, and learning outcomes in modern education systems.

Key Words: MongoDB;MERN,ML.

Introduction

The process of conducting and evaluating tests in colleges still involve a lot of manual effort from the teacher's side. Teachers need to prepare the test questions, figure out which students are allowed to attempt the tests, share the tests to the students and ensure only the allowed students should submit the tests, collect the responses from the students and then evaluate them to produce the results and forward them to the students as well. For a batch of 30-60 students, its somehow manageable but the difficulty increases with a large number of students; also the process, even for a smaller batch of students, is time-consuming and prone to errors. On the other hand, the students only get a score or a rank which is not a good experience either, since there is no demonstration of the topics which are not answered correctly for a long time in continuity or whether their performance has been improved across tests [2]. Some solutions do exist for this problem but most of them solve a part of the problem, not the entire problem. Google forms allows to host a quiz but do not have any access control, anyone having the link can attempt the quiz. LMS portals like Moodle have extended features but they require a proper setup and are not really designed for the appropriate workflow of a simple classroom MCQ test [4]. Ultimately, the result is that the teachers struggle integrating with multiple tools and students struggle receiving a real feedback.

SmartAssign portal handles this complete workflow in one place. It is a ML-based, MERN stack web application – MongoDB, Express.js, React.js, Node.js; supporting two user roles – Teachers and Students [4, 5]. Teachers upload Excel sheets with registered student IDs; create MCQ tests for the students from multiple topics, tests remaining active for a limited time; and view the complete class result analytics from the respective dashboard. Students, having their IDs in the uploaded sheet, only can register and join the class where the active tests will be displayed; attempt those test before the deadline and receive the ML-generated feedback along with the result scores. Complete Assessment Evaluation process occurs automatically without any manual effort required [1].

SmartAssign portal also includes a machine learning feedback module along with the basic assignment workflow. This module involves the usage of K-means clustering to classify the student as a Beginner, Intermediate or Advanced by analysing the attempt history of the respective student [6, 12]. It also generates a topic-wise feedback for the students, helping them understand their strong topics and focus on the weak topics, rather than just providing scores. For the students with no history (new students), a rule-based fallback provides feedback from the current attempt, ensuring feedback for everyone [8, 13].

The primary goal was not to develop something highly complex. The individual components: authentication, test management, automated scoring, ML-based feedback – are all simple and elementary. SmartAssign simply puts all these together while focusing on the needs of the teachers and the students, providing an integrated platform better than the scattered tools.

2. Literature Review

The literature highlights significant advancements in web-based academic systems and student performance analysis, particularly through the integration of full-stack technologies and machine learning techniques. Drofa, D. (2025) emphasized that combining automation with MERN stack technologies accelerates development and simplifies maintenance, although the study did not address system performance under high user loads, which is critical during examinations. Similarly, Shah, R. et al. (2017) developed an online college portal that centralized academic information such as notices, results, and attendance, improving accessibility, but lacked features like automated result generation, feedback mechanisms, and role-based access control. In the domain of machine learning, Hussain, S. et al. (2021) applied deep learning regression models to predict student performance using behavioral data, achieving higher accuracy than traditional methods; however, their approach required large datasets and did not provide actionable feedback to students. Kulkarni, S. T. et al. (2025) reviewed MERN stack development trends, highlighting its efficiency due to the use of JavaScript across the stack, but overlooked its application in educational systems and performance under concurrent user access.

Subramanian, V. (2019), in a comprehensive MERN stack guide, demonstrated its effectiveness in building flexible applications using MongoDB, Express, React, and Node.js, though the work did not address machine learning integration or system scalability. From a data analysis perspective, Liu, R. (2022) used K-means clustering to classify student performance effectively, but the study was limited to offline analysis and lacked real-time feedback capabilities. Earlier work by Ren, Z. et al. (2016) employed multi-regression models on MOOC data to predict student outcomes, identifying early engagement as a key predictor, yet the model was not

adaptable to traditional college assessment patterns and did not provide feedback. More recent research by Junejo, N. U. R. et al. (2025) demonstrated accurate early-stage performance classification using neural networks, but failed to address new student data through fallback mechanisms. Lastly, Yin, C. et al. (2023) introduced a feature extraction approach combined with a Bi-GRU network to track student performance over time, offering deeper insights into influencing factors; however, the model was computationally heavy and difficult to integrate into lightweight web portals, and like many others, lacked a feedback system. Overall, while these studies contribute to improving academic systems and predictive analytics, common research gaps include lack of real-time feedback, scalability issues under heavy usage, and limited integration of intelligent systems within user-friendly web platforms.

3. Mathodology

This section describes the overall study of the methodology for the. The steps are discussed below.

3.1 System Architecture Overview

SmartAssign is built as a three-tier web application that uses the MERN stack, which includes MongoDB as the choice of database, uses Express.js and Node.js for the backend and React.js on the frontend [4, 5]. The three tiers architecture includes the client interface, the backend API server and the database layer while also using Python-based ML service with the backend services for the processing of the feedback. The communication between the frontend and the backend is handled using the REST APIs which are secured at every route using the JWT token for each and every request.

The frontend section is divided into two separate user interfaces: one for teachers and other for the students. The teacher dashboard includes the features for uploading the student IDs lists, creating tests with their respective schedules set and viewing results for students both as an individual and as a whole class. The student dashboard is made comparatively simpler, including the features like login, attempt the available tests and view their results. Both the interfaces have zero interference in each other's workflow and this separation is implemented at the API level too, not just for UI.

3.2 User Registration and Authentication

Teacher registers using their domain-specific (initially kept generic) emails and set a password. On the other hand, students cannot register independently, their registration depends on the Excel sheet uploaded by the teacher, if their student ID matches then only the student will be able to register and login further. This was done to prevent unauthorized students from accessing the tests.

Once the user login is performed successfully, the server generates a JWT token that includes the user IDs and their role. The token is then sent back to the client and attached with every upcoming API requests. The middleware checks this token and if found faulty, rejects the request. This ensures the protection of the teacher-only endpoints.

3.3 Text Creation and Attempt Flow

Test creation from the teacher's end is quite simpler :- teacher needs to fill the test title, number of questions, start time and the end time, followed by the selection of the questions

along with their respective answers and marks. MongoDB is used for storing both the test metadata and individual questions linked to the test.

On the other side, when a student tries to attempt a test, the system ensures that the student is logged in, the test is active and the student is having a valid attempt left. The responses from the student are stored at the server side by side preventing the loss of data in case of system failure. The attempt is considered as complete when the student hits the submit button or when the timer runs out and the evaluation is started immediately.

3.4 Automayed Evaluation Engine

The evaluation logic runs immediately after the attempt is completed. It computes the scores by comparing the student responses with the correct answers available. Grades ranging from A to F are assigned on the basis of performance. The results are displayed on both dashboards (student and teacher). Teachers can view the results of the complete class as well as individual student while students can only view their own results.

3.5 Machine Learning Feedback Module

The ML module is the most crucial part of this system as it enables the students to view their topic-wise feedback for future improvements. The model uses K-means clustering for categorizing the student performance into three levels – Beginner, Intermediate, Advanced. The students are provided with their level-specific feedbacks. A Beginner level student is suggested to focus on the foundational topics that have been missed. An Intermediate level student gets suggested around specific weaker sections. The Advanced level student gets prompted towards the harder topics. The classification is done by analysing the attempt history of the student.

For a new student, the one who has no attempt history, the model uses rule-based fallback system.

3.6 Machine Learning Feedback Module

The database stores the role-specific user credentials; the students' list uploaded by the teacher; tests created; questions that are linked with the test along with available question set; the number of total attempts and the number of attempts completed; responses received from the students and their respective results as a complete class as well as an individual.

3.7 Machine Learning Feedback Module

The system is made secured at many points :- passwords are encrypted using b-crypt hashing; frontend-to-backend interaction is encrypted over HTTPS; the JWT token is checked by middleware ensuring the role-based access to the dataset; student registration based on Excel sheets preventing the unauthorized students from accessing the tests.

The SmartAssign system involves the running of two main workflows in parallel :- one for the teacher and other for the student. Both the teacher workflow and the student workflow share the same authentication system on the top. The ML feedback module comes in the picture when the submission is done. The following fig 1 shows the work flow of our work.

J Euro Eco His.
ISSN : 0391-5115
Volume 7, Isue 2 (2026)
THE JOURNAL OF EUROPEAN ECONOMIC HISTORY

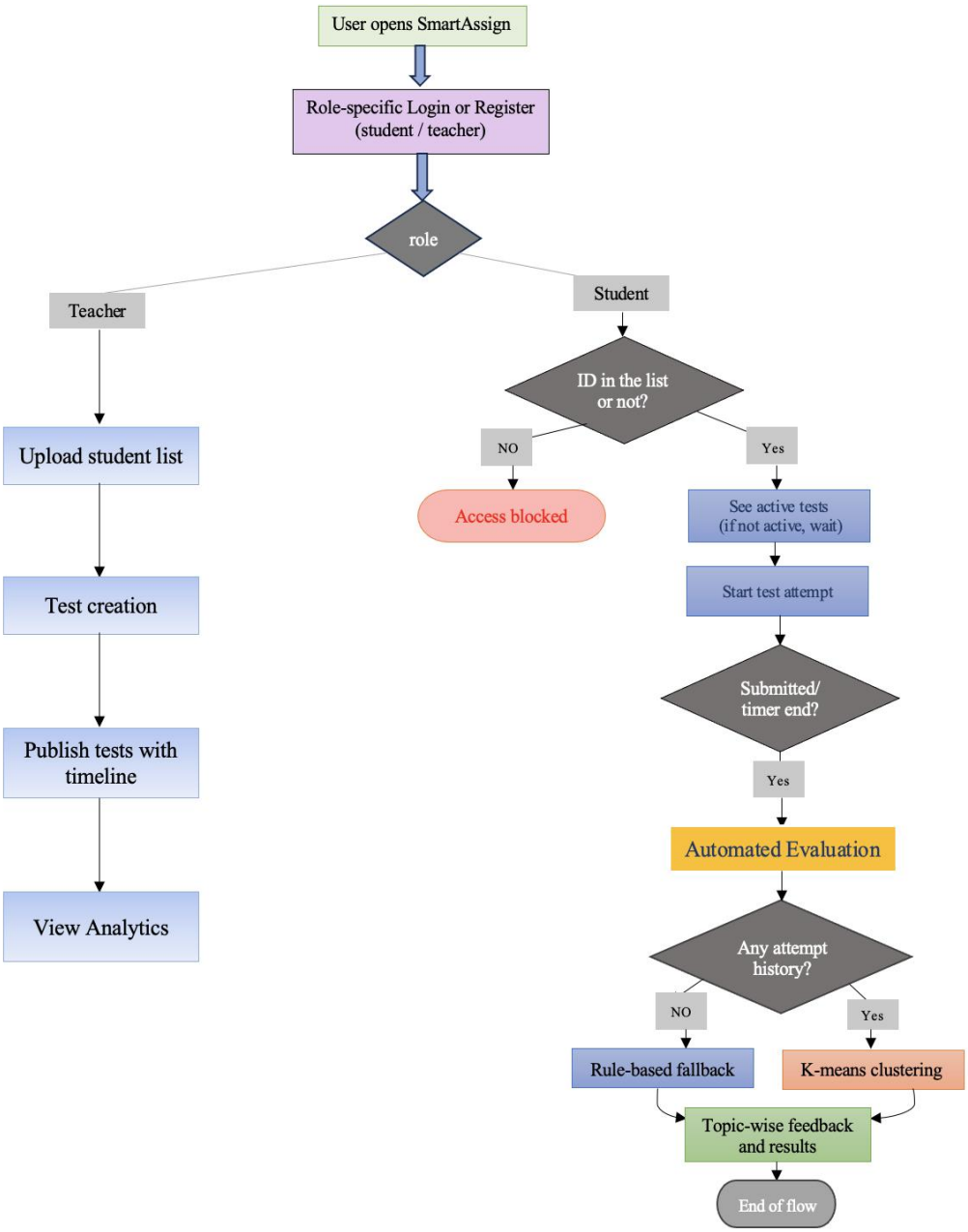

Fig 1: Work Flow diagram

4. Result and Discussion

To build SmartAssign it had to undergo several rounds of testing and refinement in order to ensure that the system works as it was planned to work. This section looks into the results obtained while testing all the different major component of the system and also describes the content that was learn and understood from these testing.

4.1 Authentication and Role-Based Access Control

The testing of system was executed under three different scenarios which included valid teacher login, registration of the student with an invalid roll number and a student trying to access a teacher-only feature. When valid teacher credentials were entered it was redirected

directly to the teacher dashboard. While an invalid roll number of the student login was rejected with an error message being displayed. A direct API call from a student role to the teacher-only feature resulted in the error HTTP 403 which is forbidden. The system implements JWT – role-based enforcement which operates on the server level which makes it independent from the client – server manipulations.

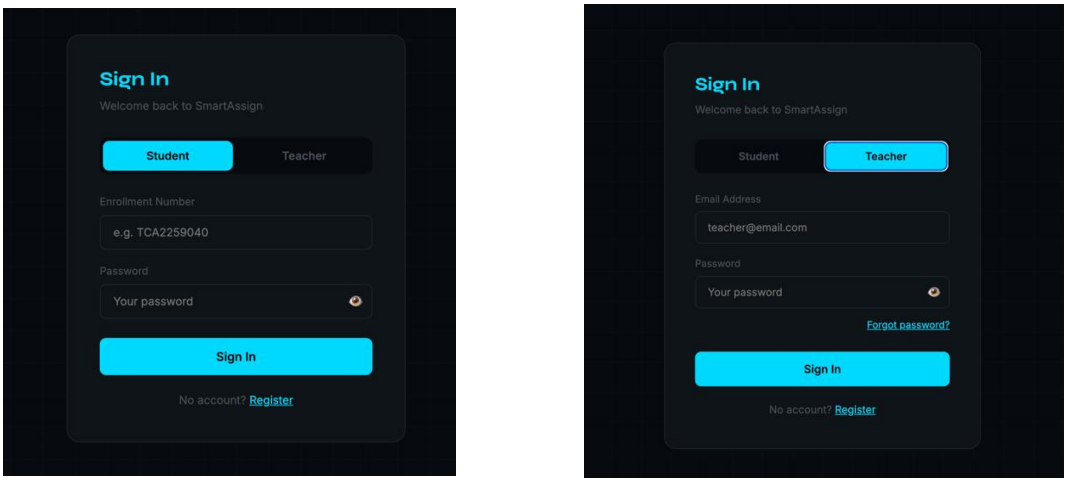

Fig2. SmartAssign login page showing email, password fields and role selection.

The teacher workflow testing covered uploading an Excel file of the all the student ID's and creating three test configuration which included a 5 – question quiz with a timer of 10 minutes, second was a 20 – question test with 45 minutes timer and the last one, a 30 – question test with the timer of 90 minutes. The results of all three were stored and displayed along with the metadata on the dashboard. The scheduling logic blocked the students to access the test outside of the specified time window. During the time of the upload, teachers are informed that the Excel parser needs a standardized column format.

Each configuration was tested with a student login. As the test started the countdown timer started and each answer was recorded after each selection rather than only storing them at the final submission. The test got auto submitted when the timer went off. The browser was intentionally closed in middle of attempt and reopened to check if the data is restored successfully.

Ten test cases were conducted with known answer key and varied the response set from being fully correct to partial correct partial incorrect and then fully incorrect. The evaluation engine matched the scores and assigned the Grades from A to F. The corner cases were tested, like being unanswered questions and partially completed answers to test the system fully.

4.2 ML Feedback Module Evaluation

Three different student profiles were tested in which it was found that the model worked as expected for the new students by activating the rule-based fallback system appropriately, the student was provided with proper initial feedback which was generated by mapping student responses to the provided answers. Any student with prior experience of 2-5 attempts, the K – means clustering performed around the accuracy of 87%, while for the remaining 13% also, the cases were boundary line and not wrong but they were vague. While when the experience was

increased to 6 or more attempts the accuracy rose to around 94%. The following table shows the accuracy of our model. Borderline cases are students whose feature vectors fell near the boundary between two cluster centroids. These are expected in any K-means implementation and do not represent errors in the algorithm [12, 13].

Table 1: Accuracy of the model

Student Profile	Cases Tested	Correctly Classified	Borderline Cases	Accuracy (%)
Cold start (0–1 attempts)	10	10-rule-based fallback triggered	0	100%
Limited history (2–5 attempts)	15	13	2	87%
Established history (6+ attempts)	10	9	1	94%
ML service failure simulation	5	5-fallback triggered correctly	0	100%

The following figure shows the accuracy between the accuracy and the student profile.ML service failure simulation give the 100% accuracy.

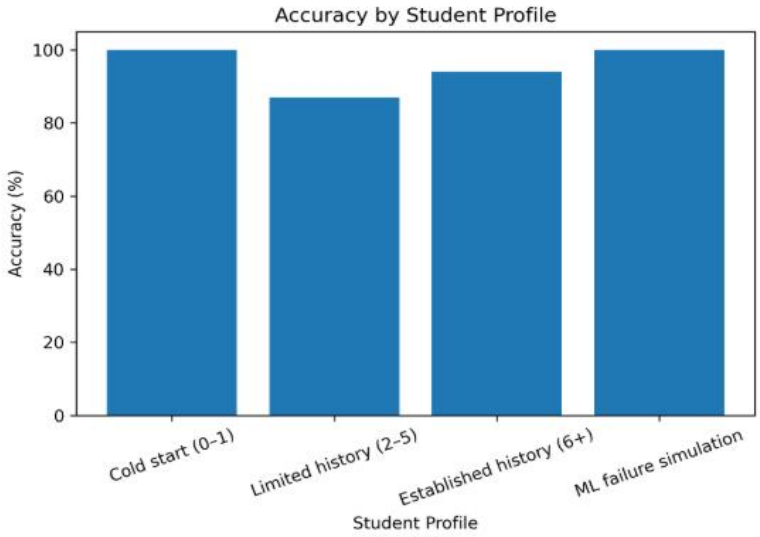

Fig 3: Accuracy between student profile

Thirty simultaneous test sessions were simulated to approximate a university lab batch, in which Node.js driven architecture sustained with average response submission latency of 312 ms and question retrieval latency was 187 ms, with no data integrity where all students responses were stored under the correct attempt records without mixing up the data.

5. Conclusion and Future Scope

SmartAssign was built to solve a specific pracical problem of conducting MCQ-based assessments in a college setting without the extra manual overhead. Classification accuracy outstretched to 87% for the students with a bit of attempt history and extended up to 94% for students having even more attempts. The students with no history are also provided with feedback using rule-based fallback system, leaving no one without guidance.The choice of MERN stack as the base framework resulted in a good decision.

Through this system and research, the fact that there is no need of a complex or expensive architecture to make evaluation more intuitive. Both K-means clustering and MERN stack are not much innovative algorithm or frameworks. The separate modules of authentication, test management, automatic scoring and feedback generation are also very simple on their own. SmartAssign stand out due to its ability of integrating all of them smoothly, making it effective, efficient and impressive.

The next improvement that comes up is the lack of a progress track over time. Currently, students are able to view their results and feedbacks after individual tests. The feature of viewing their improvement across attempts is currently missing. Sequence-based models like Bi-GRU networks would be able to handle the temporal pattern in an effective manner and would also look like a natural update to the present clustering approach.

References

1. Drofa, D. (2025). Optimization of software development processes through the use of full-stack technologies and automation. Contemporary Issues in Artificial Intelligence, 1.
2. Shah, R., Tanwar, A., Shah, M., & Pandey, S. (2017). Online college portal. International Research Journal of Engineering and Technology, 4(9), 2395–0056.
3. Hussain, S., Gaftandzhieva, S., Maniruzzaman, M., Doneva, R., & Muhsin, Z. F. (2021). Regression analysis of student academic performance using deep learning. Education and Information Technologies, 26(1), 783–798.
4. Kulkarni, S. T., Siddha, R., Mattani, B., Mohite, S., & Lahane, D. P. (2025). A comprehensive review of MERN stack development: Trends, challenges, and future directions. Challenges, and Future Directions (December 1, 2025).
5. Subramanian, V. (2019). Pro MERN stack: Full stack web app development with Mongo, Express, React and Node. Apress.
6. Liu, R. (2022). Data analysis of educational evaluation using K-means clustering method. Computational Intelligence and Neuroscience, 2022(1), 3762431.
7. Ren, Z., Rangwala, H., & Johri, A. (2016). Predicting performance on MOOC assessments using multi-regression models. arXiv preprint arXiv:1605.02269.
8. Junejo, N. U. R., Nawaz, M. W., Huang, Q., Dong, X., Wang, C., & Zheng, G. (2025). Accurate multi-category student performance forecasting at early stages of online education using neural networks. Scientific Reports, 15(1), 16251.
9. Yin, C., Tang, D., Zhang, F., Tang, Q., Feng, Y., & He, Z. (2023). Students learning performance prediction based on feature extraction algorithm and attention-based bidirectional gated recurrent unit network. PLOS ONE, 18(10), e0286156.
10. Pelima, L. R., Sukmana, Y., & Rosmansyah, Y. (2024). Predicting university student graduation using academic performance and machine learning: A systematic literature review. IEEE Access, 12, 23451–23465.
11. Sari, I. P., Al-Khowarizmi, A. K., & Batubara, I. H. (2021). Cluster analysis using K-means algorithm and Fuzzy C-means clustering for grouping students' abilities in online learning

process. Journal of Computer Science, Information Technology and Telecommunication Engineering, 2(1), 139–144.
12. Sultan Alalawi, S. J., Mohd Shaharanee, I. N., & Mohd Jamil, J. (2023). Clustering student performance data using K-means algorithms. Journal of Computational Innovation and Analytics (JCIA), 2(1), 41–55.
13. Chong, B. (2021). K-means clustering algorithm: A brief review. Academic Journal of Computing & Information Science, 4(5), 37–40.
14. Fard, M. M., Thonet, T., & Gaussier, E. (2020). Deep k-means: Jointly clustering with k-means and learning representations. Pattern Recognition Letters, 138, 185–192.
15. Yaro, A. S., Maly, F., Prazak, P., & Malý, K. (2024). Outlier detection performance of a modified Z-score method in time-series RSS observation with hybrid scale estimators. IEEE Access, 12, 12785–12796.
16. Fatemifar, S., Awais, M., Akbari, A., & Kittler, J. (2022). Developing a generic framework for anomaly detection. Pattern Recognition, 124, 108500.
17. Lesouple, J., Baudoin, C., Spigai, M., & Tourneret, J. Y. (2021). Generalized isolation forest for anomaly detection. Pattern Recognition Letters, 149, 109–119.
18. AlKhuzaey, S., Grasso, F., Payne, T. R., & Tamma, V. (2024). Text-based question difficulty prediction: A systematic review of automatic approaches. International Journal of Artificial Intelligence in Education, 34(3), 862–914.
19. Almseidin, M., Alzubi, M., Kovacs, S., & Alkasassbeh, M. (2017). Evaluation of machine learning algorithms for intrusion detection system. In Proceedings of the 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY) (pp. 277–282). IEEE.