Journal of Modern Technology
Vol. 02, No. 02, July 2025, pp. 317-326.
DOI: https://doi.org/10.71426/jmt.v2.i2.pp317-326 , E-ISSN: 3049-3102
Homepage: https://review.journal-of-moden-technology.com/index.php/jmt/index

Smart Detection of Low Engagement in Students using Artificial Intelligence and Behavioral Data

Ortencia Laci[1], Keti Dervishi[2], Klea Vreto[3*]

[1]Department of Computer Science, Faculty of Engineering and Architecture, University of New York Tirana, Albania, Email: ortencialaci@unyt.edu.al , ORCID: https://orcid.org/0009-0002-7454-8346
[2]Department of Computer Science, Faculty of Engineering and Architecture, University of New York Tirana, Albania, Email: ketidervishi@unyt.edu.al, ORCID: https://orcid.org/0009-0000-9742-4034
[3*]Department of Computer Science, Faculty of Engineering and Architecture, University of New York Tirana, Albania, Email: kleavreto@unyt.edu.al, ORCID: https://orcid.org/0009-0001-6371-046X

Article Info

Article history:

Received: May 20, 2025
Revised:
Accepted:
First Online:

Keywords:

Artificial Intelligence
Data analysis
Student evaluation
Boolean logic

ABSTRACT

This work aims to develop a basic artificial intelligence program that can identify the class's most indolent student. The purpose of the program is to examine specific behaviors that demonstrate a lack of effort. These include missing a lot of classes, arriving late, especially by more than twenty minutes, turning in assignments late, and receiving poor grades. The program evaluates each of these factors for each student and determines which ones fit all the criteria for being lazy. We start by entering each student's information. Their name, number of absences, tardiness on various days, final grade, and whether or not they turned in their work on time are all included in this. This makes it easier to evaluate each student's performance in relation to the class average. It determines whether the student meets each condition using straightforward checks, also known as Boolean logic, such as "true" or "false." A student is labeled lazy if they have missed more than four classes, been late more than twenty minutes at least once, failed to turn in assignments on time, and received a grade below the class average. The program creates a list of the lazy students after evaluating every one of them. It selects the student with the lowest score from this list. The term "laziest" is then applied to that student. The program indicates that no lazy student was found if no student satisfies every requirement. This prevents students from being unfairly judged if they only missed one or two things.

Corresponding Author:

Email address of corresponding author: kleavreto@unyt.edu.al (Klea Vreto)

1. INTRODUCTION

In many classrooms, certain students consistently exhibit behaviors such as frequent absences, late arrivals, and missed assignments. These patterns are often indicative of low engagement or motivation. Recognizing such behavior manually can be subjective or delayed [1]-[2]. Hence, this study proposes an AI-based framework that uses objective behavioral data to identify the least engaged student. The motivation behind this work is to move beyond assumptions and instead use data-driven methods for identifying disengagement. While explainable AI has been widely used in security, finance, and healthcare, its application in behavioral analysis within education remains underexplored [15]-[17]. This research aims to bridge that gap by offering a

transparent and interpretable method for assessing student inactivity. Some of the key activities of AI are discussed below in detail.

1. 1 Scanning the sensitive data

AI can scan documents and figure out which ones contain private or important information, like health records or credit card numbers. It uses smart techniques like pattern matching and natural language processing to tag this data so it's handled more carefully [3]-[4].

A) Instance: In a hospital, AI can read patient files and mark them as sensitive. This ensures only certain people can view them, and strong encryption is applied automatically [5], [13]-[14].

B) Code:

```
import re
def detect_sensitive_patient_data(text):
sensitive_keywords = ["diagnosis", "patient ID", "insurance", "HIV", "cancer", "blood type"]
flagged = []
for keyword in sensitive_keywords:
if re.search(rf"\b{keyword}\b", text, re.IGNORECASE):
flagged.append(keyword)
if flagged:
return f"Sensitive data detected: {', '.join(flagged)}"
return "No sensitive data detected."
# Example usage
file_text = "Patient ID: 12345\nDiagnosis: Diabetes\nPrescribed medication: Metformin"
print(detect_sensitive_patient_data(file_text))
```

C) Output: Sensitive data detected: diagnosis, patient ID.

D) Flowchart: A flowchart for the A) instance and B) coding of section 1.1 is shown in Figure 1.

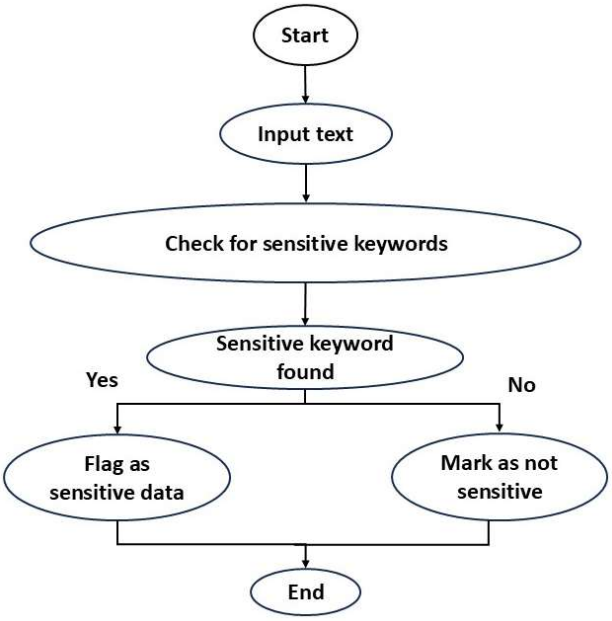

Figure 1. Scanning the sensitive data.

1. 2 Catching suspicious actions

AI can keep an eye on how people use systems and spot strange behavior, like someone accessing files at odd hours or downloading too much at once. These could be signs of hacking [6][7].

A) Instance: At a bank or Apple company, if an employee tries to open hundreds of customer files late at night, AI quickly detects it and alerts the security team.

B) Code:

```
def detect_suspicious_behavior(activity_log):
threshold = 100  # Example: if more than 100 files are accessed in a short period
late_hours = range(0, 6)  # Midnight to 6 AM
alerts = []
for log in activity_log:
if log['files_accessed'] > threshold or log['hour'] in late_hours:
alerts.append(f"ALERT: Suspicious activity by {log['user']} at {log['hour']}h")
return alerts if alerts else ["No suspicious activity detected."]
# Example usage
log_data = [
{'user': 'john_doe', 'files_accessed': 150, 'hour': 2},
{'user': 'jane_smith', 'files_accessed': 20, 'hour': 14},
]
for alert in detect_suspicious_behavior(log_data):
print(alert)
```

C) Output: ALERT: Suspicious activity by john_doe at 2h

D) Flowchart: A flowchart for the A) instance and B) coding of section 1.2 is shown in Figure 2.

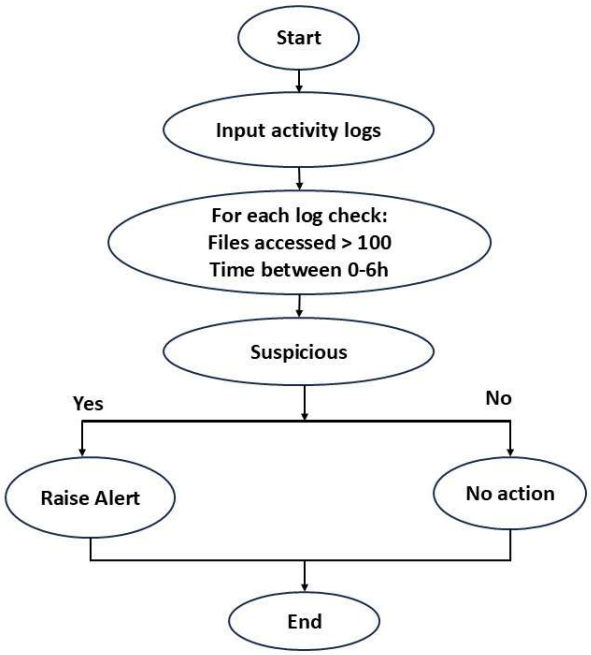

Figure 2. Scanning the sensitive data.

1. 3 Confidentiality of data viewing

AI helps decide who should be allowed to see certain data based on their job, behavior, or location. It can change permissions in real-time if something seems off [8].

1. 3. 1 Task 1:

A) Instance: In a company's human resource system, AI lets people see salary data but blocks junior staff from accessing it.

B) Code:

```
# Simulated user data
users = [
{"username": "alice", "role": "HR_Manager", "location": "Office", "behavior": "Normal"},
{"username": "bob", "role": "Junior_Staff", "location": "Office", "behavior": "Normal"},
```

```
{"username": "charlie", "role": "HR_Assistant", "location": "Remote", "behavior": "Suspicious"},
]
# Function to check access permission
def can_view_salary(user):
if user["role"] not in ["HR_Manager", "HR_Assistant"]:
return False
if user["behavior"] == "Suspicious":
return False
if user["location"] == "Remote" and user["role"] != "HR_Manager":
return False
return True
# Simulated check
for user in users:
print(f"User: {user['username']}")
if can_view_salary(user):
print("Access to salary data granted.\n")
else:
print("Access denied to salary data.\n")
```

C) Output:

```
User: alice
Access to salary data granted.

User: bob
 Access denied to salary data.

User: charlie
 Access denied to salary data.
```

D) Flowchart: A Flowchart for the instance as discussed in section 1.3 is illustrated by Figure 3.

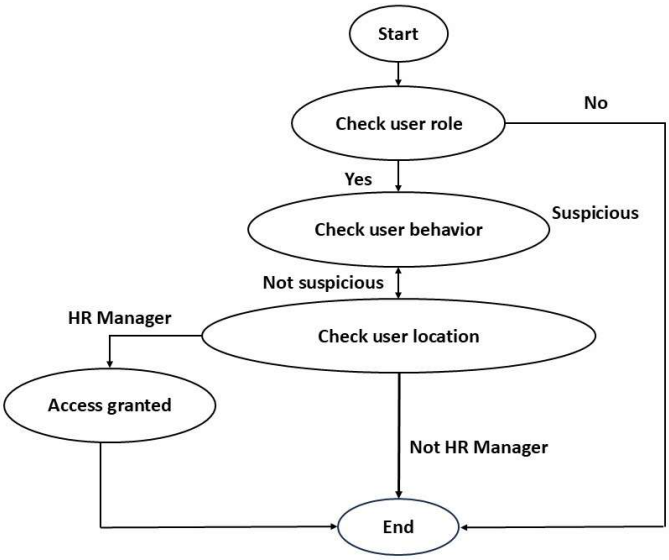

Figure 3. Confidentiality of data viewing.

Another real-world example is detecting spam emails in any platform supported by Google. AI scans the content, email sender and any other information related with this email. It detects suspicious components and flags it to the Google, and after that it blocks the email or send it to the spam folder. The system constantly learns from new threats and from user feedback like marking a message as "Not Spam".

1. 3. 1 Task 2:

Why data sensitivity is required for data security with diagram? to answer this, let's start by understanding what "data sensitivity" means. In simple words, it's about how important or private a piece of information is. Think about your own life. Some things you're okay with everyone knowing, like your favourite movie or the city you live in. But other things you'd want to keep private — like your bank account details, your passwords, or your medical history. These private things are what we call sensitive data. It's data that needs extra care because, if it falls into the wrong hands, it could cause you harm. Sensitive data can include things like personal information, financial details, business secrets, or anything that could be used to hurt someone or break their trust. Now, we can talk about data security [9]-[10]. This is the process of protecting data from unauthorized access, damage, or theft. Just like we lock our homes to keep out burglars, we need to 'lock' our data to keep it safe. Data security involves many tools and practices like using passwords, encrypting information, setting up firewalls, and making sure only the right people have access to certain files or systems. The goal of data security is to make sure that data stays safe, private, and correct and that it's available when the right person needs it. Imagine you're guarding a house. If you don't know which rooms have important things, you might waste your time guarding the wrong door or leaving the safe wide open. It's the same with data. Not all information is equally important. If we treat all data the same, we might end up protecting unimportant files and accidentally exposing critical ones. So, the first step in data security is classifying data based on its sensitivity. Once we know which data is most sensitive, we can apply the strongest protection to it.

Sensitive data can look different depending on the person or organization. For individuals, it can be things like Social Security numbers, medical records, or private messages. For businesses, it could include trade secrets, financial data, or customer information. For governments, it could include classified documents or national security files. In all cases, if sensitive data gets stolen, the consequences can be serious: identity theft, fraud, financial loss, lawsuits, and damage to reputation. That's why identifying and protecting sensitive data is so important.

If an organization doesn't take data sensitivity seriously, they can easily make mistakes, like sending confidential files to the wrong person, or not encrypting sensitive customer data. These mistakes can lead to data violation, where hackers steal private information [11]-[12]. And once the data is out, it's nearly impossible to take it back. People can lose trust, businesses can lose customers, and in some cases, people can lose their jobs. So, understanding and respecting data sensitivity it's a must.

A) Code 1:

Is a user input sensitive or not?

```
def classify_data_sensitivity(data_input):
sensitive_terms = ['SSN', 'credit card', 'password', 'medical', 'salary']
for term in sensitive_terms:
if term.lower() in data_input.lower():
return "High Sensitivity: Apply strong encryption and access control."
return "Low Sensitivity: Basic protection is sufficient."
# Example usage
data = "This file contains the employee salary and bonus information."
print(classify_data_sensitivity(data))
```

B) Output: High Sensitivity: Apply strong encryption and access control.

C) Flowchart: Figure 4 represents the instances of sensitive data for strong encryption and access control.

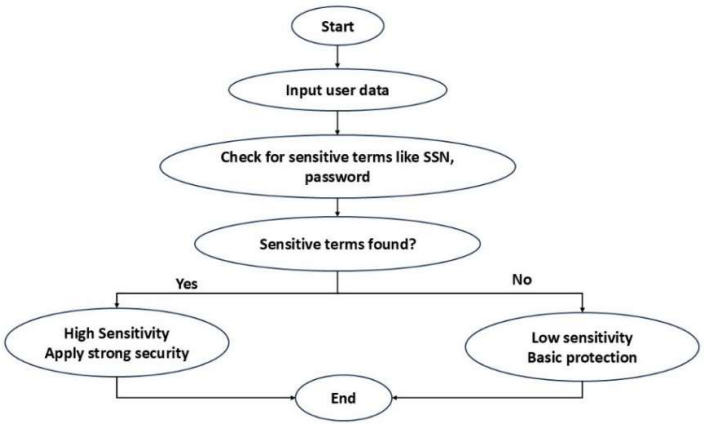

Figure 4. Instances of sensitive data for strong encryption and access control.

D) Code 2:

```
Scanning government's sensitivity data
def government_doc_classifier(doc):
levels = {
"Top Secret": ["nuclear", "spy", "weapons"],
"Confidential": ["passport", "immigration", "embassy"],
"Public": ["weather", "tourism", "education policy"]
}
for level, keywords in levels.items():
for keyword in keywords:
if keyword in doc.lower():
return f"{level} Document: Needs {'maximum' if level=='Top Secret' else 'moderate'} security."
return "Unclassified Document: No special protection needed."
# Example usage
document = "This document contains nuclear launch codes and spy identities."
print(government_doc_classifier(document))
```

E) Output: Top Secret Document: Needs maximum security.

F) Flowchart: Figure 5 represents instances of sensitive data for spy identities.

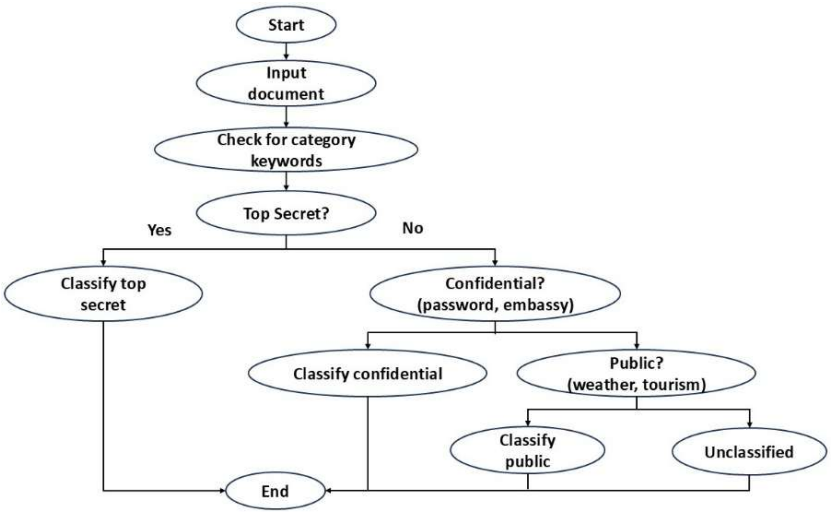

Figure 5. Instances of sensitive data for spy identities.

G) Graph: A graphical representation of the classification of data sensitivity levels is shown in Figure 6.

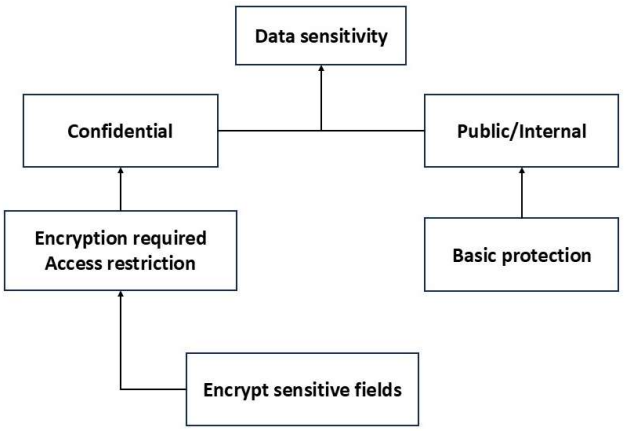

Figure 6. Classification of data sensitivity levels.

As shown in Figure 6, it consists of the following classification of data sensitivity levels.

1. *Data sensitivity:* Importance of the data.
2. *Confidential:* Very important and secret data.
3. *Public/Internal:* Data than can be shared with everyone.
4. *Basic protection:* No need for any security action.
5. *Encryption required access restriction:* Data is locked and can be unlocked only with permission.
6. *Encrypt sensitive fields:* Only the important data are being transformed into a secret code(encrypted).

The main concept behind the Figure 6 is to show us that how sensitivity decides about data protection. Starting by the first box is about deciding data importance. 'Confidential' means that the data we are dealing with is sensitive and important and needs protection. 'Public/Internal' means that this data is not that important and private, so it doesn't need special protection, it can be shared or not doesn't really matter. Below these boxes, it written the action needed to take, as we said above for confidential data, we need strong protection and for public or internal data we need a basic protection or not at all. And in the end, for the confidential data need to be encrypted which is the highest phase of security.

2. METHODOLOGY

To get the laziest one on the class we put some conditions, so based on those later we can decide if it is or not lazy. These are the specific things the AI agent looks at in a student's data:

A) Absences:

If the student has missed more than 4 classes.

This condition determines the number of missed classes for a student. A student is not regularly attending school if they miss more than four days of class. Regular attendance is crucial because it increases the likelihood that students will understand the material, stay on task, and finish their assignments. A student who misses a lot of classes is usually falling behind or not taking their studies seriously. We use this condition to identify students who might not be working hard enough by just failing to show up. It's among the first indications of indolence or disinterest.

B) Minutes being late:

If they have ever been late for more than 20 minutes.

This condition checks if the student has ever been late to class for more than 20 minutes. Being a little late sometimes is normal, but arriving very late shows a lack of discipline and responsibility. A student who regularly comes late, especially by a large amount of time, might miss important parts of the lesson. This behavior can affect their performance and show that they are not managing their time well. That's why we include this rule — it helps us find students who are not only absent but also not punctual when they do attend.

C) Result:

If their score is less than the class average.

This condition compares the student's final grade or result to the class average. A student may be struggling or not trying hard enough if their score is lower than that of the majority of the class. Although a low score by itself does not always indicate laziness, it takes on greater significance when paired with other behaviors (such as tardiness or missing class). The score helps determine whether a person's bad habits are also affecting their academic performance, so we use this condition to ensure that the program doesn't label someone lazy based on a single minor problem.

D) Submitted on time:

If they did not submit their assignment on time.

This requirement determines whether the student submitted their classwork or assignment on time. Time management and responsibility are demonstrated by timely submission of work. Failing to do this frequently indicates that a student lacks organization or is not serious about their academic work. This is an obvious indication of indolence or lack of drive. Since turning in work late—especially without a valid excuse—is a clear sign that a student is not putting in enough effort, we included this rule. Only if all of these conditions are true, the student is considered as lazy by the code.

3. FLOW OF CODE EXECUTION OF THE PROPOSED WORK

3.1 Creating the data for students

We create a student's class and attach attributes to each of them. This attributes are:

- Name
- Number of absences in class
- A list of how many minutes late they were each day
- Their final result of grades
- A boolean saying whether they submitted the assignment on time or not.

A) Calculate class average

One of the conditions needed to be filled is to check whether the student's average is below the class average, so for this we need to find and calculate class average. This can be done by formula.

B) Check if it's lazy or not

After this, based on the conditions provided, if someone fills those means it's lazy, if not means not lazy.

C) Create list of lazy students

If we have more than one person that has these conditions, we create a list and put all the list with the students that meet this criteria.

D) Find the laziest student

Now, the student added to the 'lazy list' are being compared which one has the lowest points. The one found , is being outputted as the laziest student among all.

3.2 Flowchart

The Figure 7 illustrates the execution of the code as discussed in the previous section. The flowchart starts by collecting the data of all students — their names, absences, how late they've been, if they submitted their work on time, and their grades. Then, it calculates the class average. Next, it checks each student one by one:

- *Did they miss more than 4 classes?*
- *Were they late for more than 20 minutes at least once?*
- *Is their grade below the class average?*
- *Did they submit assignments late?*

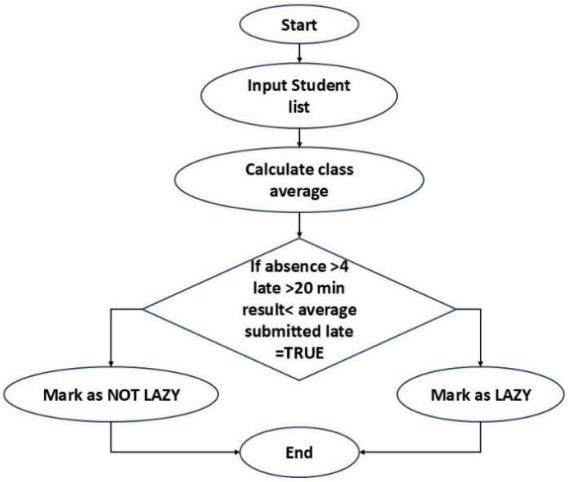

Figure 7. Flow of execution of the code.

If a student matches all these, they are added to the "lazy list." From that list, the student with the lowest score is chosen as the "laziest." If no one matches all conditions, it prints that no lazy student was found. Output summary of lazy student from the propose AI strategy is presented in Table 1.

Table 1. Output summary of lazy student detection code.

Student name	Absences	Late minutes	Submitted on time	Final grade	Is Lazy?
Almesa	6	(10, 15, 25)	No	60	Yes
John	2	(5, 10)	Yes	78	No

Table 2. Output summary of lazy student and their scores.

Name of the laziest student	Score	Class average
Almesa	45.00%	58.20%
Kles	34.00%	56.00%
Eljo	48.00%	58.80%

Table 2 illustates the that out of all students who meet our conditions, Almesa had the lowest score and filled all the lazy conditions.

4. CONCLUSION

This work presnets an effective way to look at student performance using basic logic. Instead of guessing, we used actual data to support our decisions. It was interesting to see how a few simple checks could lead to clear results. We learned that even small tools can make a difference in how teachers understand their students. With a few improvements, this idea could really be useful in real classrooms.

In the time coming, the cureent work presented in this paper can be better by just adding some few changes. One first idea can be by adding more conditions, which can check better which is the laziest person in class. One attribute can be teacher feedback or comments, like what are their idea of this specific student, how is his behavior, what is his active participation in class. This can differentiate if we have a same of very near result between every student, and teacher's idea is more important than ever in this case. Another attribute added can be to check if they ever have been gotten any ethic penalty, but this may be the last thing to be checked on. We can transform this simple program or code into a website or school program, that may help professors to learn more about these students.

DECLARATIONS

Conflict of Interest: The authors declare that there is no conflict of interest.
Funding: This research received no external funding.
Availability of data and materials: No data is available in this article.
Publisher's note: The Journal and Publisher remain neutral about jurisdictional claims in published maps and institutional affiliations.
Acknowledgments: The authors would like to formally acknowledge Dr. Debabrata Samanta, CIT Program Head & Assistant Professor, Rochester Institute of Technology, Kosovo, Email: dsamanta@auk.org for his exceptional guidance.

REFERENCES

[1] Chen JW, Tsai MS, Hung CL. Towards an Effective Tool Wear Monitoring System with an AI Model Management Platform. In2024 IEEE 22nd International Conference on Industrial Informatics (INDIN) 2024 Aug 18 (pp. 1-6). IEEE. doi: 10.1109/INDIN58382.2024.10774399.

[2] Ashwitha M, Abinaya S. AI-Powered Disease Prediction Tool. In2025 4th OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 5.0 2025 Apr 9 (pp. 1-6). IEEE. doi: 10.1109/OTCON65728.2025.11070635.

[3] Neeraj M, Daniel E, Durga S, Seetha S. Explainable Multi-Stage Churn Prediction using Graph Neural Network in Telecom Sector. In2025 8th International Conference on Trends in Electronics and Informatics (ICOEI) 2025 Apr 24 (pp. 1100-1105). IEEE. doi: 10.1109/ICOEI65986.2025.11013438.

[4] Bhuvaneswari R, Kumar P, Kaviya S. Explainable AI-Driven Heart Disease Prediction. In2025 International Conference on Visual Analytics and Data Visualization (ICVADV) 2025 Mar 4 (pp. 959-964). IEEE. doi: 10.1109/ICVADV63329.2025.10961035.

[5] Dwivedi R, Dave D, Naik H, Singhal S, Omer R, Patel P, Qian B, Wen Z, Shah T, Morgan G, Ranjan R. Explainable AI (XAI): Core ideas, techniques, and solutions. ACM computing surveys. 2023 Jan 13;55(9):1-33. https://doi.org/10.1145/3561048

[6] Akther F, Begum M, Mahmud T, Boltayev A, Hanip A, Hossain MS. Streamlit-Based AI for Multi-Disease Prediction. In2025 3rd International Conference on Inventive Computing and Informatics (ICICI) 2025 Jun 4 (pp. 1622-1628). IEEE. doi: 10.1109/ICICI65870.2025.11069667.

[7] Asal B, Demir MÖ. Enhancing Software Defect Prediction through Explainable AI: Integrating SHAP and LIME in a Voting Classifier Framework. In2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP) 2024 Sep 21 (pp. 1-7). IEEE. doi: 10.1109/IDAP64064.2024.10710700.

[8] Singh AP, Gupta G. Evaluation of the Explainable AI-NLP Framework for Text Categorization. In2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE) 2024 May 9 (pp. 181-185). IEEE. doi: 10.1109/IC3SE62002.2024.10593585.

[9] Olu-Ajayi R, Alaka H, Sunmola F, Ajayi S, Mporas I. Statistical and artificial intelligence-based tools for building energy prediction: A systematic literature review. IEEE Transactions on Engineering Management. 2024 Jul 15;71:14733-53. doi: 10.1109/TEM.2024.3422821.

[10] Lu J, Wu R, Li P. Multiple Explanations for Neural Network Based Dropout Prediction. In2024 4th International Conference on Computer Communication and Artificial Intelligence (CCAI) 2024 May 24 (pp. 247-252). IEEE. doi: 10.1109/CCAI61966.2024.10603374.

[11] Mustaqeem M, Alam M, Mustajab S, Alshanketi F, Alam S, Shuaib M. Comprehensive bibliographic survey and forward-looking recommendations for software defect prediction: datasets, validation methodologies, prediction approaches, and tools. IEEE Access. 2024 Dec 13. doi: 10.1109/ACCESS.2024.3517419.

[12] Pandey H, Pandey P, Gupta D. Airfoil Self Noise Prediction Using Machine Learning and Explainable AI. In2024 IEEE Recent Advances in Intelligent Computational Systems (RAICS) 2024 May 16 (pp. 1-6). IEEE. doi: 10.1109/RAICS61201.2024.10689885.

[13] Baker RS, Yacef K. The state of educational data mining in 2009: A review and future visions. Journal of educational data mining. 2009 Oct 1;1(1):3-17.

[14] Thai-Nghe N, Horváth T, Schmidt-Thieme L. Factorization Models for Forecasting Student Performance. InEDM 2011 Jul 6 (pp. 11-20).

[15] Hajdini A, Fazliu L, Gjoshi D. Comparative Study of Join Algorithms in MySQL: Cross, Inner, Outer, and Self Joins. Journal of Computing and Data Technology. 2025 Jun. 9;1(1):1-9. https://review.journal-of-modern-technology.com/index.php/jcdt/article/view/57

[16] Dintakurthy Y, Innmuri RK, Vanteru A, Thotakuri A. Emerging Applications of Artificial Intelligence in Edge Computing: A Comprehensive Review. Journal of Modern Technology. 2025 Jan. 25;1(2):175-8. https://review.journal-of-modern-technology.com/index.php/jmt/article/view/31

[17] Soma AK. Weighted Graph Clustering with PaCCo. In2025 International Conference on Emerging Systems and Intelligent Computing (ESIC) 2025 Feb 8 (pp. 815-818). IEEE.