ENHANCING MECHANICAL ENGINEERING EDUCATION THROUGH A VIRTUAL INSTRUCTOR IN AN AI-DRIVEN VIRTUAL REALITY FATIGUE TEST LAB

by

Amir Abbas Yahyaeian

A Thesis

Submitted to the Faculty of Purdue University
In Partial Fulfillment of the Requirements for the degree of

Master of Science in Mechanical Engineering

Department of Mechanical and Energy Engineering at IUPUI
Indianapolis, Indiana
August 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Alan Jones, Co-Chair
Department of Mechanical and Energy Engineering

Dr. Jing Zhang, Co-Chair
Department of Mechanical and Energy Engineering

Dr. Xiaoping Du
Department of Mechanical and Energy Engineering

Approved by:
Dr. Likun Zhu

Dedicated to my loving family,
whose unwavering support and encouragement
have been the cornerstone of my academic journey.
Your belief in me has fueled my determination.
and I am forever grateful for your presence in my life.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my advising committee at IUPUI, Dr. Alan Jones, Dr. Jing Zhang, and Dr. Xiaoping Du and the graduate program coordinator, Lisa Hastings, for their invaluable guidance and support throughout this research. I extend my deepest appreciation to my advisor at UT, Prof. Mohsen Hamedi. Special thanks go to Jason Piercy, the VR lab technician at IUPUI, for his technical expertise, assistance, and invaluable contributions to the experimental tests conducted for this thesis. Additionally, I extend my thanks to the volunteer students, Amin Akbari, Amir R. Marjani, Arash Behforooz, Ehsan Malekipoor, Marzie M. Bonekhater, and Shokooh Sh. Ahmadi, who are all engineering students at IUPUI. Their involvement in providing opportunities for experimental tests and their invaluable assistance has significantly contributed to the completion of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis demonstrates the combination of virtual reality (VR) and artificial intelligence (AI) specifically exploring the practical application of Natural Language Processing (NLP) and GPT-based models in educational VR laboratories. The objective is to design a comprehensive learning environment where users can independently engage in laboratory experiments, deriving similar educational outcomes as they would from a traditional, physical laboratory setup, particularly within the realms of Science, Technology, Engineering, and Mathematics (STEM) disciplines.

Using machine learning techniques and authentic virtual reality simulating educational experiments, we propose an advanced learning platform—Virtual Reality Instructional Laboratory Environment (VRILE). A key feature of the VRILE is an AI-powered instructor capable of not only guiding the learners through the tasks but also responding intelligently to their actions in real-time.

The AI constituent of the VRILE uses the GPT-2 model for text generation in the field of Natural Language Processing (NLP). To ensure the generated instructions were contextually relevant and meaningful to lab participants, the model was trained on a dataset derived from an augmentation over user interactions within the VR environment.

By pushing the boundaries of how AI can be utilized in educational VR environments, this research paves the way for broader adoption across other domains of engineering education. Furthermore, it provides a solid foundation for future research in this interdisciplinary field. It marks a significant stride in the integration of technology and education, encouraging more ventures into this promising frontier.

1. INTRODUCTION

1.1 Background

Laboratory exercises are essential components of engineering education, offering a bridge between theoretical concepts and their practical application. These exercises provide a hands-on approach to education. While many possible experiments are possible, the first experiment implemented in the VRILE (Virtual Reality Instructional Laboratory Environment) project was a rotary fatigue test. Fatigue testing is a fundamental concept in material science and mechanical engineering that allows students to understand the behavior of materials under varying degrees of load and stress. It's a complex yet critical concept that exposes students to the realities of material properties, enhancing their comprehension of the subject matter. However, traditional laboratory environments are not without their shortcomings. Resource constraints often limit the capacity for students to experiment freely. Physical laboratories demand extensive resources, including space, equipment, and the associated costs of both. Additionally, safety is a perennial concern in these environments, given the nature of the materials and machinery involved. Ensuring that students can safely carry out their experiments necessitates considerable investment in safety equipment and training. There is also the issue of student engagement. The conventional lab setup does not always provide an environment conducive to active learning, making it challenging for some students to fully grasp complex procedures. VR technology has the potential to simulate a physical lab environment, thereby overcoming limitations associated with space and resource constraints. It allows for a more immersive learning experience that could enhance student engagement. In addition, AI, especially in the form of a virtual instructor, can provide personalized guidance during lab sessions, supplementing the learning experience. The combination of VR and AI not only revolutionizes the way lab sessions are conducted but also introduces a novel approach to experiential learning in engineering education.

1.2 Objectives

The principal aim of this study is to create a learning platform that leverages the power of both VR and AI for enhanced laboratory instruction in the field of mechanical engineering, specifically for fatigue testing. The envisaged learning platform, referred to as a Virtual Reality Instructional Learning Environment (VRILE), has two essential components: a simulated VR environment that authentically replicates the traditional fatigue testing laboratory, and an AI-enabled virtual instructor that delivers personalized, contextually relevant instruction to the students as they navigate the lab. The VR environment component of the VRILE plays a crucial role in delivering an immersive, engaging, and interactive learning experience that is safe and resource efficient. By mimicking the real-life laboratory setup in detail, the VR environment provides students with a practical, hands-on experience of conducting fatigue tests, without the restrictions associated with traditional labs. Students can interact with virtual machinery, tools, and equipment in a manner that simulates physical interaction, their understanding of the procedural aspects of fatigue testing. The virtual instructor component of the VRILE, powered by AI, aims to augment the student's learning experience in the VR environment. The virtual instructor is designed to provide just-in-time, personalized instruction to students based on their current task, location within the lab, and their learning progression. The instruction is not pre-determined or static but is dynamically generated using machine learning models that interpret the student's current context and learning needs. To accomplish these objectives, the research employs a two-stage machine learning approach. First, a machine learning model is utilized to accurately interpret the student's current task and location within the VR environment. The outputs from this model are then transferred to an NLP model, specifically a GPT-based model. The NLP model generates the appropriate instructional prompt that is then delivered to the student. This approach ensures that the instructional prompts are not only contextually relevant but also meaningful and beneficial to the student's learning process.

By integrating VR and AI in this manner, this research aims to create a comprehensive, immersive, and intelligent virtual learning platform that overcomes the limitations of traditional labs and sets a new benchmark for experiential learning in engineering education.

1.3 Scope of this thesis

This thesis focuses on the design, development, and assessment of the VRILE for fatigue testing. The research delves into the technological foundations, choice of machine learning models, data collection, augmentation and processing, model training, and implementation within a VR environment. The selection of suitable machine learning models, which form the backbone of the AI-powered virtual instructor is one of the first requirements in the development of an Intelligent Virtual Instructor (IVI). The process not only encompasses identifying models that demonstrate high predictive accuracy and efficiency but also involves an understanding of their underlying mathematical intricacies. The chosen models are then put through the rigors of extensive training and testing, leveraging curated datasets. This aids in optimizing the models' performance in the distinct context of VRILE. Further, this study elucidates the process of data collection, management, and preprocessing. This work does not rely only on theoretical or pre-existing datasets. Instead, it builds upon the empirical data collected from experiments with students, all volunteers, from the School of Engineering and Technology. Given the highly specialized nature of the data used for training the models, the methodologies employed for gathering and processing this data carry significant implications for the success of the AI components. The research, therefore, delves into the specifics of how the data is gathered and curated, and the strategies employed to ensure it is in the optimal format for model training. This research does not solely represent an exploration of AI or VR technologies as separate entities, both of which have made considerable strides in multiple sectors. Rather, the unique contribution of this thesis lies in the innovative synthesis of these two cutting-edge technologies, creating an integrated tool designed to provide experiential learning for STEM based laboratories.

This thesis is a master’s thesis that builds on a previous project which laid the groundwork by designing the VR lab. The aim of this project is to enhance this VR environment by integrating an AI-powered instructor. This research focused on deploying the best-suited machine learning model through comparative evaluation and employed transfer learning for the NLP component. This allows for the application of pre-existing models, thereby optimizing available resources.

1.4 The structure of the thesis

The structure of this thesis is organized into six chapters. Each chapter is designed to build upon the information and context established in the preceding sections, providing a coherent and logical exploration of our work. Following the introduction, Chapter 2 presents an overview of the pertinent topics, such as the importance and challenges of educational labs, experiential learning, and the role of VR and AI in education. This chapter also delves into the details of AI, such as machine learning classification, natural language processing, and text generation using GPT-2. Chapter 3 provides a brief overview of the development of a VR environment for a laboratory experience. Chapter 4 details the Proposed VRILE, discussing its features, benefits, and the need for an intelligent virtual instructor within this environment and describe the choice of the machine learning model for the intelligent instructor. Chapter 5 presents a comprehensive picture of the Model Implementation and Deployment. It investigates the AI model, comparing and selecting suitable models for the task, data collection, and preprocessing as well as model training and fine-tuning. Along with a description of how the machine learning model is utilized within the Unity game engine. The Conclusion and Future Work in Chapter 6 encapsulates the outcomes of this research and suggests potential future directions for this work.

2. LITERATURE REVIEW

2.1 Laboratories in STEM education

Developmental and research labs primarily focus on guiding product development and advancing fundamental knowledge, while instructional labs are designed to educate students rather than generate new discoveries [1]. The objectives of engineering laboratories, as identified in the literature, encompass several aspects. These include aiding students in connecting theoretical concepts to practical applications [1]–[6], equipping them with research and design skills, granting access to contemporary technology [7]–[9], fostering motivation and retention [3], [6], [8], integrating new technology into the curriculum, and promoting self-learning and lifelong education. Lab experiences are widely recognized as crucial in STEM disciplines like engineering, chemistry, physics, and life sciences [10], with accreditation bodies such as ABET emphasizing the significance of sound lab practices [11].

Nevertheless, state-of-the-art labs present challenges to modern engineering and science curricula. The increasing range of experimental topics and the requirement for precise equipment have led to higher costs associated with acquiring, maintaining, and supporting lab equipment by technicians [12]–[15]. Consequently, many curricula struggle to provide comprehensive education across various STEM subjects due to insufficiently developed labs. To address these limitations, Virtual Reality Integrated Lab Experiences (VRILES) have been proposed. VRILES not only enable students to conduct predefined experiments but also facilitate exploration of experimental apparatus and techniques without the risks of equipment damage or personal harm. Leveraging the visualization capabilities of VR environments, precise 3D models of the system and its components offer deeper insights into the internal structure, experiment physics, and operation and interfaces of modern experimental equipment [16]–[18].

In the context of distance learning, providing lab experiences encounters significant obstacles. These include issues related to equipment accessibility, costs, licensing, human resources, maintenance, development of off-site lab facilities, equipment setup, and student training [1]. Current approaches to tackle these concerns comprise video demonstrations, desktop virtual reality, and remotely accessible labs (tele-operated). Video demonstrations involve presenting students with experimental videos and providing representative datasets but lack interactivity. Desktop virtual reality allows students to interact with 3D graphical representations of equipment using a computer monitor, mouse, and keyboard. Remotely accessible labs enable students to operate lab equipment through web interfaces, transmitting sensor, visual, and audio data. However, these methods may still possess certain limitations. Video demonstrations merely familiarize students with the equipment and are often utilized as a preliminary step prior to conducting real-life labs [19]–[23]. Remote labs lack a true sense of presence in the lab environment despite offering additional sensory input [20]. Desktop virtual labs and tele-operated equipment may leave students feeling isolated, thereby reducing their motivation [6], [20]. Desktop virtual labs have been implemented in various disciplines, including chemical engineering, physics, mathematics, chemistry, biology, and electronic circuits [24]–[40].

During the development of the first VRILEs, just about any experiment could have been implemented. However, the first laboratory that was developed was a rotary fatigue laboratory. While the development of the fatigue laboratory occurred before the work in this thesis, it allows for comparison to real-life laboratories and allows for accelerated fatigue testing not feasible in real-life conditions. Fatigue testing is a crucial aspect of mechanical engineering that involves subjecting materials and components to cyclic loading conditions to evaluate their durability, reliability, safety, and longevity. T Fatigue tests provide valuable insights into the fatigue behavior of materials and components, allowing engineers to enhance their designs by making them more resistant to fatigue failure. By identifying potential failure modes and stress concentrations, fatigue testing aids in the development of structures that are durable and reliable over time [40]. This information contributes to optimizing the design of materials and components, ultimately improving their performance under cyclic loading conditions. Understanding fatigue strength through testing enables accurate predictions of performance over time, leading to enhanced reliability [41].

2.2 Challenges in real-life labs

Ensuring adequate training for students and researchers is vital to prevent accidents. Beaumont et al. [41]highlight additional challenges. Long and expensive test campaigns are common, especially for high-cycle endurance tests, which can be time-consuming and costly. Reducing the number of required specimens and shortening testing procedures are addressed through Accelerated Life Tests (AFT) but ensuring high-reliability assessment remains a challenge.

Statistical estimation of material properties from fatigue tests can be difficult due to multiple factors affecting the results. Simulation and modeling, while beneficial, also present challenges due to the complexity of models and the need for accurate input data. In Le et al. [42]research the challenges related to teaching fatigue theory are discussed. Traditional lecture-based approaches may limit students' exposure to the practical aspects of fatigue testing, including specimen preparation, testing equipment, and data analysis. Safety concerns pose challenges in providing hands-on experience. The complex nature of fatigue theory can be challenging for students unfamiliar with materials science or mechanical engineering. Additionally, the lack of prior knowledge in fatigue theory can impede the understanding of more advanced concepts related to fatigue testing. To address these challenges, proposed solutions include integrating active learning approaches in teaching fatigue theory, which expose students to multiple aspects of fatigue testing. This includes hands-on fatigue testing, FEA simulation, theoretical calculations, and practical assignments. By adopting an integrated approach, students can enhance their understanding of fatigue theory and its practical applications in mechanical design [42].

The VRILE project offers a practical solution to these challenges. By transitioning to a virtual platform, resource limitations can be circumvented, offering students the freedom to access the lab irrespective of their geographical location and schedule. Safety hazards, a pivotal concern in physical labs, become a non-issue in a virtual setup. Students are afforded the liberty to experiment, learn, and even make mistakes without facing real-world repercussions. Moreover, the integration of AI offers a level of instruction consistency, ensuring a uniform learning experience. Many labs are performed in group settings, commonly one or two students perform most of the tasks associated with the lab while others may only watch. With the VRILE, each student must fully perform the entire lab procedures. Finally, the virtual environment can replicate real-world scenarios, providing an unlimited number of opportunities for practice and learning. These features position the VRILE as an innovative and effective solution to the prevalent issues in traditional educational labs.

2.3 Experiential learning

Experiential learning is a powerful approach to education that involves learning through experience, reflection, and conceptualization. Gentry et al. [43] highlight the key components of experiential learning, emphasizing the importance of feedback, the role of emotions, and the various learning styles involved. It emphasizes that experiential learning goes beyond passive learning by actively engaging learners with their environment and encouraging reflection. Examples of experiential learning activities, such as live cases, where students work on real-world problems or cases provided by organizations. This hands-on approach allows students to apply their knowledge and skills, receive feedback from professionals, and develop problem-solving and decision-making abilities. Furthermore, Gentry et al. [4]underscores the benefits of experiential learning, including increased motivation, engagement, and retention of knowledge. It concludes by providing criteria for evaluating teaching methodologies that facilitate experiential learning, promoting the adoption of effective approaches. Experiential learning techniques, such as lab simulations, case studies, and group projects, are employed to internalize concepts and principles as [44]. The p significance of reflection activities, including concept mapping, aids in connecting experiences to abstract concepts. By visualizing relationships between key terms and concepts, concept mapping enhances understanding and serves as a study guide [44].

Experiential learning offers a transformative approach to education. The integration of hands-on experiences, feedback, reflection, and conceptualization fosters deeper engagement, enhances knowledge retention, and develops practical skills. Experiential learning activities, such as live cases and case studies, provide students with authentic contexts to apply theoretical knowledge and receive feedback. Moreover, the inclusion of deliberate reflection and concept mapping enhances the transfer of experiential learning to abstract conceptualization, strengthening students' grasp of complex concepts and principles. Experiential learning is a dynamic field with numerous approaches and applications beyond those discussed. Other examples include internships, field trips, service-learning projects, and cooperative education programs. These activities further enhance experiential learning by connecting students to real-world contexts, fostering the development of professional skills, and nurturing a sense of social responsibility. The interdisciplinary nature of experiential learning allows it to be adaptable across various disciplines and educational levels.

The VRILE project exhibits a well-rounded implementation of the key components of experiential learning in its design and function. It creates a virtual environment that mimics a real-world lab, providing a practical context for students to apply their theoretical knowledge. This authentic setting facilitates active engagement, allowing students to understand the implications of their actions in a safe and controllable setting, akin to live case studies, but with added advantages of repeatability and zero physical risk. The integration of AI, particularly the GPT-2 model, in the VRILE project amplifies the experiential learning experience. It acts as a virtual instructor that delivers contextually relevant, real-time feedback to students. This feature mirrors the feedback loop essential in experiential learning, fostering immediate reflection and improvement in students' approaches. The AI component thus enhances the learning experience by providing immediate, personalized feedback, something often not feasible in traditional lab settings due to resource constraints. Moreover, the project encourages abstract conceptualization and active experimentation, the two pivotal stages in experiential learning. By providing a realistic 3D laboratory environment and a step-by-step procedural guide, students can actively test their ideas and witness the outcomes of their actions, subsequently forming or modifying their abstract understanding of fatigue testing principles. This process contributes to their grasp of complex concepts and principles, fostering deeper learning.

2.4 Benefits of VR in educational environments

Virtual reality (VR) has gained significant attention in the field of education, offering immersive and interactive learning experiences. VR has emerged as a promising technology for enhancing education. [45], [46]studies recognize its potential to transform the learning experience by offering immersive, interactive, and engaging activities. VR's ability to improve learning outcomes and communication skills warrants further research and collaboration. As educators and researchers continue to explore the possibilities of VR in education, its implementation holds significant promise for shaping the future of learning. Freina et al. [45] present an extensive exploration of VR in education. They categorize VR into non-immersive and immersive types, with immersive VR showing potential as devices become more user-friendly and economically accessible. The authors underscore the advantages of VR, including enhanced engagement, motivation, and the ability to provide unique experiences difficult to replicate in the real world. Building upon Freina et al., McGovern et al. [46] delve into a comprehensive review on VR in education. The authors examine existing studies that demonstrate the effectiveness of VR in enhancing learning outcomes and student engagement. They discuss different types of VR technology and their applications across educational settings. The results reveal the positive impact of VR on participants' communication skills, presentation performance, and engagement. The authors assert that VR has the potential to significantly enhance learning outcomes and communication skills while providing a safe environment for training and enhances learner involvement and motivation through a game-like approach. It supports diverse learning styles and offers experiences that are difficult to replicate in the real world.

2.5 Benefits of AI in educational environments

Artificial intelligence (AI) has the potential to revolutionize education, personalizing learning experiences, improving student learning outcomes, and increasing the efficiency of educators in managing classrooms and administrative tasks. Tahiru [47] examines the challenges, opportunities, and benefits of Artificial Intelligence in Education (AIED). The author highlights the potential of AI to improve education through personalized learning experiences, automation of administrative tasks, real-time feedback, collaboration support, and increased accessibility. However, concerns such as technology dependence, bias in AI algorithms, privacy issues, cost implications, and potential job loss are also identified. The authors emphasize the need for further research in areas such as personalized and collaborative learning and ethical and socio-economic implications. Studies focusing on the integration of AI and machine learning (ML) in education have shown that AI-assisted systems can be developed for personalized learning, adaptive testing, intelligent tutoring systems, learning analytics, and content creation, which use AI to adapt the learning experience to the individual needs of each student [48]–[50] The use of AI in education thus has the potential to help teachers identify areas where students need additional support and to provide students with the tailored feedback and support they need to succeed [48]. Furthermore, AI can be used to track student progress and to identify areas where students need additional support, enabling educators to better target instructional strategies to meet the needs of individual students. AI can also be used to facilitate language learning and communication skills through the use of natural language processing (NLP). NLP can help teachers identify areas where students are struggling with language and can facilitate communication between students from different cultures and language backgrounds, promoting greater cultural awareness and understanding [50]. In higher education, AI can be used to extend and enhance teaching and learning. For example, AI can promote efficient delivery of course materials, facilitate student progress tracking, and increase the accuracy of evaluations [51]. Additionally, the use of AI can create more engaging and interactive learning environments [52]. A recent study by Sumo and Bah [53] concluded that the application of AI in education can improve the effectiveness of institutional teaching, learning and research.

While artificial intelligence has been extensively employed in a multitude of educational contexts and virtual reality scenarios, its application in the context of a VR-driven, interactive, mechanical engineering laboratory is a novel frontier. In particular, the AI-driven virtual instructor element of the VRILE project, which provides in-situ personalized responses and guidance, represents a unique intersection of AI and education that is uncharted in the current landscape of mechanical engineering education. AI in education has been lauded for its capacity to tailor educational experiences to the individual learner, and for the efficiency it brings to administrative tasks, among other benefits. However, the VRILE project leverages these strengths in a distinctive and innovative way. The AI-driven instructor operates in a reactive and dynamic manner, responding to the learner's immediate context within the virtual laboratory, and guiding them through the complexities of performing a fatigue test, a feature that has not been previously implemented in this specific domain. This nuanced application of AI transcends conventional uses by adding a layer of interactivity and realism that mirrors the unpredictability and immediacy of real-world engineering lab scenarios. As such, this approach does not simply replicate the existing lab experience in a virtual format, but enhances it by offering personalized, real-time instruction that is tailored to the learner's precise stage in the process and their specific location within the virtual lab environment. Given this, the VRILE project emerges as an exciting and novel application of AI in the domain of STEM education. It encapsulates the immense potential that AI technologies hold for enriching learning experiences and highlights the significance of ongoing collaboration between educators, AI experts, and developers in achieving ethical and effective integration of AI in education. As such, the VRILE project can be considered a pioneering effort in harnessing AI to create more immersive, interactive, and effective educational experiences in STEM laboratories.

2.5.1 AI instructor

In conducting a comprehensive literature review on the topic of AI-instructors, it became apparent that the availability of directly related works is relatively limited. Despite employing meticulous search strategies and exploring various scholarly databases, only a limited number of studies specifically addressing the concept of AI-instructors were identified. This scarcity of existing literature underscores the nascent nature of the field and highlights the need for further investigation and original contributions. Nevertheless, the studies that were discovered offer valuable insights into the potential applications and implications of AI-instructors, serving as a foundation for this research endeavor. Shekhar et al. [54] propose the use of an AI-driven Contextual Virtual Teaching Assistant (CVTA) using the RASA open-source conversational AI framework. This assistant aims to provide round-the-clock support to students, automating and refining the quality of the learning experience. Through the implementation of AI and natural language processing (NLP), the CVTA can offer contextual help, tailored answers, and alerts about deadlines. The paper discusses the benefits of AI and NLP in educational environments, including reduced dependence on human interaction and enhanced availability. The CVTA demonstrates its effectiveness in addressing delays, uncertainty, and miscommunication in remote online environments. Limitations are acknowledged, such as the reliance on instructor materials and the accuracy of the NLP model, and future research is suggested to explore alternative AI models and integration with other educational technologies.

2.5.2 Classification in machine learning

Machine learning classification is among the methods most frequently used in intelligent systems. Seven machine learning algorithms can be used for classification tasks, namely Decision Table, Random Forest, Naïve Bayes, Support Vector Machine, Neural Networks, JRip, and Decision Tree [55]. For this project four powerful classification algorithms in machine learning were considered, which are: decision tree, random forest, naive Bayes, and support vector machine (SVM).

Bayesian network

A Bayesian Network, also referred to as a Bayesian Belief Network, provides a straightforward and intuitive method for applying Bayes Theorem to solve complex problems. While it may not strictly adhere to the traditional Bayesian framework, the design of Bayesian Networks allows for the subjective specification of probability distributions for the random variables (nodes) and their relationships (edges). This subjective specification enables the model to capture and represent a "belief" or subjective understanding of the complexities within a given domain [56]. Bayesian probability focuses on subjective probabilities or beliefs associated with different outcomes. It recognizes that our beliefs about the likelihood of an event can be influenced by a variety of factors, such as prior knowledge, personal experiences, or expert opinions. By incorporating these subjective beliefs into the modeling process, Bayesian Networks offer a more flexible and versatile framework for reasoning under uncertainty. One of the fundamental strengths of a Bayesian Network lies in its ability to capture the joint probabilities of the events represented within the model. By explicitly defining the relationships between variables, the network provides a structured representation of the dependencies and interactions among different factors. This allows for efficient probabilistic inference, enabling us to reason about the probabilities of specific events or make predictions based on available evidence. Furthermore, Bayesian Networks facilitate the exploration of causal relationships between variables. The network structure can reflect our understanding of cause-and-effect relationships, providing insights into the underlying mechanisms at play within a system. This causal reasoning capability makes Bayesian Networks particularly valuable in various domains, such as healthcare, finance, and decision-making systems, where understanding the causal relationships between factors is critical [56], [57].

Bayesian Networks can be made more tangible through a small example:

Let's consider a problem involving three random variables [56]: A, B, and C. A's dependence on B and C's dependence on B can be described as follows:

- A exhibits conditional dependence on B, denoted as $P(A|B)$.
- C demonstrates conditional dependence on B, denoted as $P(C|B)$.

It is worth noting that A and C have no direct influence on each other. Furthermore, the conditional independencies can be expressed as:

- A is conditionally independent of C given the presence of B: $P(A|B, C)$.
- C is conditionally independent of A given the presence of B: $P(C|B, A)$.

It is important to observe conditional dependence in the context of conditional independence, indicating that A is conditionally dependent on B when C is taken into consideration. The conditional independence of A given C can also be stated as the conditional dependence of A given B. As A remains unaffected by C, its probabilities can be calculated solely based on B:

- $P(A|C, B) = P(A|B)$

Notably, B remains unaffected by both A and C and has no parent nodes. Consequently, the conditional independence of B from A and C can be expressed as P (B, P(A|B), P (C|B)) or simply as P (B).

Additionally, the joint probability of A and C given B, or conditioned on B, can be expressed as the product of their respective conditional probabilities:

- P (A, C | B) = P(A|B) * P(C|B)

The model effectively summarizes the joint probability, P(A, B, C), which can be calculated as:

- P (A, B, C) = P(A|B) * P(C|B) * P(B)

Finally, the graphical structure of the Bayesian Network can be depicted as Figure 2-1:

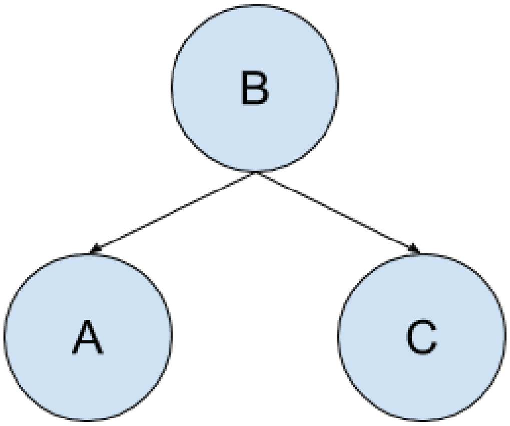


Figure 2-1: A simple Bayesian network with nodes and edges (from ref. [56])

Benefits of Bayesian networks [57]–[61]:

- Probabilistic modeling: Bayesian networks provide a principled framework for representing and reasoning about uncertainty using probability theory. They allow the modeling of complex relationships among variables and the quantification of uncertainty in predictions.

- Causal reasoning: Bayesian networks can capture causal relationships between variables, enabling the understanding of cause-and-effect relationships in the data.

This can be particularly useful in domains where causal relationships are important, such as healthcare or decision-making systems.

- Interpretability: Bayesian networks provide a graphical representation that facilitates the visualization and interpretation of the dependencies and interactions among variables. This can help in understanding the underlying structure of the problem domain and aid in decision-making processes.

- Incremental learning and updating: Bayesian networks support incremental learning, where the model can be updated with new data or additional evidence easily. This makes them adaptable to changing environments and allows for continuous learning.

Disadvantages of Bayesian networks:

- Computational complexity: Inference and learning in Bayesian networks can be computationally expensive, especially when dealing with large datasets or complex networks. As the number of variables and dependencies increases, the complexity of computations grows, making them less suitable for real-time or resource-constrained applications.

- Data requirements: Bayesian networks often require a substantial amount of data to accurately estimate the model parameters and learn the structure. Insufficient data can lead to inaccurate or unreliable models.

- Model assumptions: Bayesian networks rely on certain assumptions, such as conditional independence between variables, which may not always hold in practice. Violations of these assumptions can affect the accuracy and reliability of the model.

- Knowledge engineering: Constructing a Bayesian network typically involves domain expertise and knowledge engineering efforts. Determining the appropriate structure and specifying the conditional probability distributions can be challenging and time-consuming, requiring input from experts in the field.

In summary, Bayesian Networks offer a powerful and intuitive approach to modeling complex problems. By incorporating subjective beliefs, capturing joint probabilities, and facilitating causal reasoning, they provide a versatile framework for reasoning under uncertainty and making informed decisions based on available evidence. Additionally, constructing a Bayesian network often requires expert knowledge or domain expertise to accurately specify the network structure and prior probabilities [62]. Bayes' theorem is the fundamental equation in Bayesian networks that describes how to update the probability of a hypothesis (H) given new evidence (E) based on prior knowledge (P(H)) and the likelihood of observing the evidence given the hypothesis.

Decision tree

A decision tree is a hierarchical model employed in decision support systems to represent decisions and their potential outcomes, encompassing chance events, resource expenses, and utility considerations. This algorithmic approach employs conditional control statements and falls under the category of non-parametric, supervised learning, serving purposes in both classification and regression tasks. The tree structure comprises a root node, branches, internal nodes, and leaf nodes, forming a hierarchical and tree-like configuration. The name, decision tree, reflects its usage of a tree-like flowchart structure that illustrates predictions resulting from a series of feature-based divisions. The process commences with a root node and concludes with a decision derived from the leaves [63]. Figure 2-2 depicts an outline of a root node following with leaves.

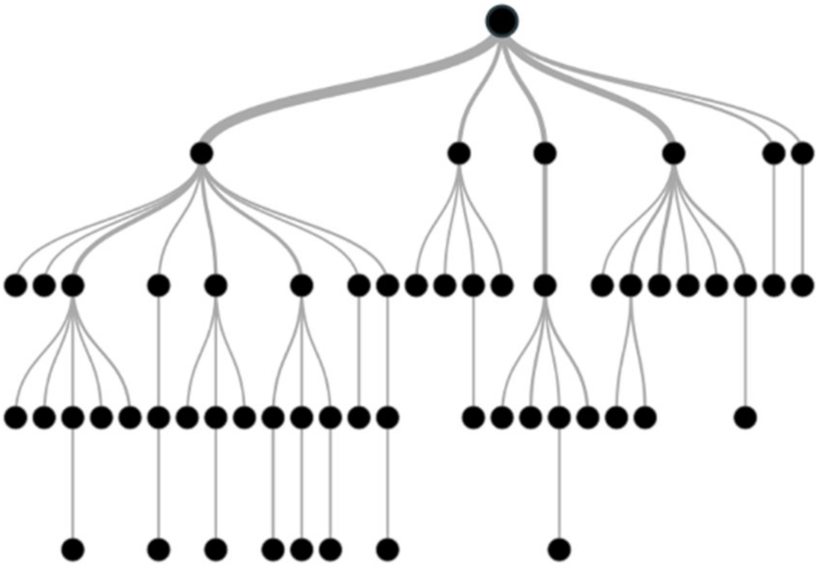


Figure 2-2: Root node, decision nodes, leaves, etc. in a Decision tree algorithm (from ref. [63])

In a decision tree, branching and nodes play essential roles in representing the decision-making process and capturing relationships between variables [64]. Figure 2-3 shows branching process in this algorithm. Some of the essential terminology used for Decision trees are as follows:

- Root Nodes: These nodes mark the initial stage of the decision tree, where the population divides based on different features.
- Decision Nodes: After splitting the root nodes, the resulting nodes are referred to as decision nodes.
- Leaf Nodes: These nodes signify the termination of further divisions and are often known as leaf nodes or terminal nodes.
- Sub-tree: Similar to a small portion of a graph being labeled as a sub-graph, a section of the decision tree is referred to as a sub-tree.
- Pruning: Pruning involves the removal of certain nodes to prevent overfitting, ensuring the tree's generalizability.

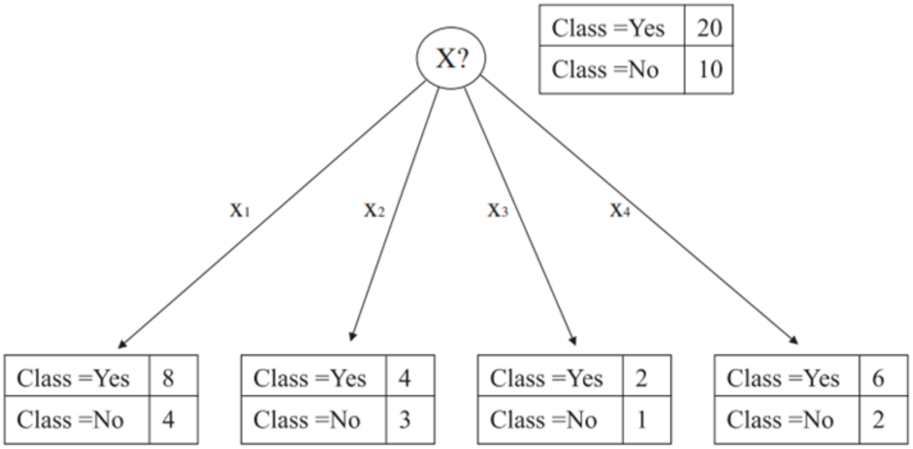


Figure 2-3: Branching of a node in a decision tree (from Ref. [63])

Advantages of Decision Trees [63]–[72]:

- Interpretability: Decision trees offer a highly interpretable model, as the tree structure provides a clear visual representation of the decision-making process. The rules and splits in the tree can be easily understood and communicated to stakeholders.

- Feature Importance: Decision trees provide insights into the importance of different features or variables in the decision-making process. By examining the top-level splits, we can identify which features have the most significant impact on the outcome.

- Handling Non-linear Relationships: Decision trees can handle non-linear relationships between features and the target variable. They are capable of capturing complex interactions and can model relationships that may be missed by other linear models.

- Handling Missing Values: Decision trees can handle missing values in the dataset by considering alternative paths based on available features. This makes decision trees robust in handling datasets with incomplete information.

Disadvantages of Decision Trees:

- Overfitting: Decision trees are prone to overfitting, especially when the tree grows too deep and captures noise or irrelevant patterns in the training data. Overfitting can lead to poor generalization and reduced performance on unseen data.

- Lack of Robustness: Decision trees can be sensitive to small changes in the training data. A slight variation in the data or the addition of new data points can lead to a completely different tree structure, which may affect the stability and consistency of the model.

- High Variance: Decision trees can exhibit high variance, meaning that small changes in the training data can result in significantly different tree structures and predictions. This variability can reduce the reliability of the model.

- Bias towards Features with More Levels: Decision trees tend to favor features with more levels or categories since they can create more splits and potentially improve the model's performance. This bias may neglect other features that could be important but have fewer levels.

- Difficulty in Capturing Certain Relationships: Decision trees may struggle to capture certain complex relationships that require a deeper understanding of the data. They can be limited in modeling interactions between multiple variables or handling situations where the relationships are not easily separable by simple splits.

Random Forest

Random Forest is an ensemble training algorithm that utilizes the technique of bagging to construct a collection of decision trees, as illustrated in Figure 2-4. By employing random subsets of the training data, Random Forest effectively addresses overfitting concerns and creates a classifier that is more resistant to noise and generalizes better. This ensemble approach introduces the concept of random feature selection during the construction of each individual tree, allowing for efficient training even in high-dimensional feature spaces [68]. The ensemble of decision trees in Random Forest collectively combines their predictions to make accurate and robust classifications. The aggregation process, whether through majority voting or probability averaging, leverages the diversity of the trees to improve the overall predictive performance. This diversity arises from the random sampling of both training instances and features, resulting in trees that capture different aspects and patterns of the data.

Advantages of Random Forest [68]–[72]:

- Robustness to Noise: Random Forest is known for its ability to handle noisy data. By constructing an ensemble of decision trees and aggregating their predictions, it can reduce the impact of outliers and noisy instances, leading to more reliable results.

- Mitigation of Overfitting: Random Forest's use of bagging and random feature selection helps to mitigate overfitting. By building multiple trees on different subsets of the data and features, it reduces the risk of capturing spurious patterns specific to the training set.

- Fast Training: Despite using multiple decision trees, Random Forest can be efficiently trained. The parallelizability of tree construction and prediction enables faster processing, making it suitable for large datasets.

- Feature Importance: Random Forest provides a measure of feature importance, allowing for insights into the relevance and impact of different features on the prediction. This information can aid in feature selection and feature engineering tasks.

Disadvantages of Random Forest:

- Lack of Interpretability: The interpretability of Random Forest is often lower compared to individual decision trees. While the feature importance can be informative, understanding the complex interactions and decision-making process of the ensemble as a whole can be challenging.

- Increased Memory Usage: Random Forest requires more memory compared to individual decision trees, as it needs to store multiple trees along with their associated data structures. This can be a limitation when dealing with limited memory resources or extremely large datasets.

- Slower Prediction: Although Random Forest can be trained efficiently, its prediction phase can be relatively slower compared to simpler models like single decision trees. This is because predictions involve evaluating each instance across multiple trees and combining their results.

Random Forest's ability to handle noise, mitigate overfitting, and provide efficient training has made it a popular choice in various domains where accuracy and interpretability are crucial. It stands as a powerful and reliable ensemble learning method, leveraging the collective strength of multiple decision trees to achieve superior predictive capabilities.

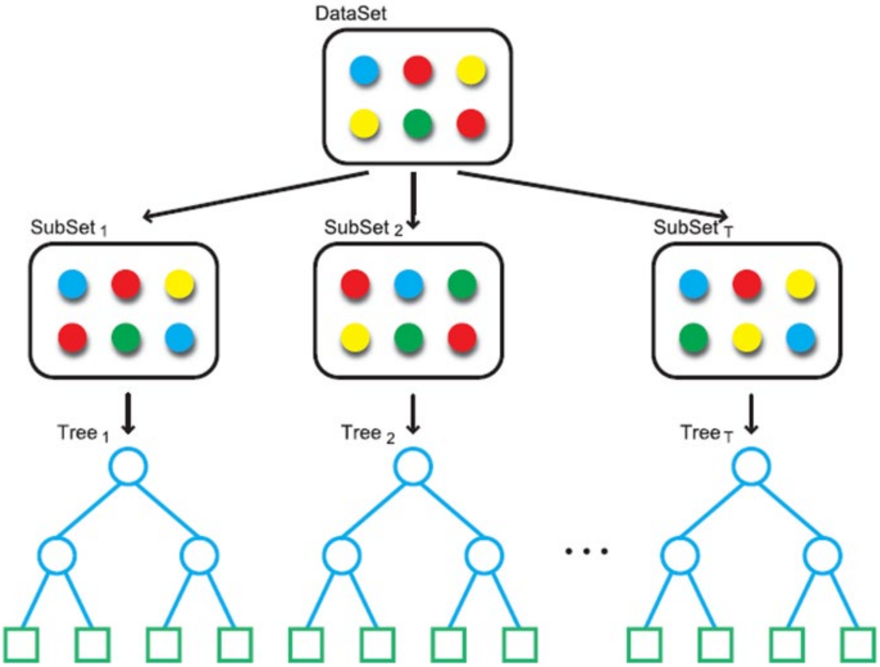


Figure 2-4: : Random Forest structure (from Ref. [68])

Support Vector Machines

Support Vector Machines (SVM) are powerful algorithms that excel in handling high-dimensional data and nonlinear decision boundaries. SVM finds the optimal hyperplane that maximizes the margin between classes, leading to good generalization performance. SVMs also have a strong theoretical foundation and are effective even in cases where the number of features is larger than the number of instances. However, SVMs can be sensitive to the choice of kernel function, which is used to transform the input data into a higher-dimensional feature space. The selection of the kernel and its associated parameters requires careful consideration to avoid overfitting or underfitting the data. Additionally, SVMs can be computationally expensive, especially when dealing with large datasets [73]. In the SVM, nonlinear separation is realized by using the nonlinear vector function Φ(x) that maps the m-dimensional input vector x into the l-dimensional feature space.

$$D(x) = w^t \times \emptyset(x) + b \quad (2)$$

where w is the l-dimensional vector and b is the bias term. Then the L1 SVM is formulated in the primal form as follows:

$$minimize\ Q(w,b,\delta) = \frac{1}{2} w^t \times w + C \sum_{i=1}^{M} \delta_i \quad (3)$$

$$subject\ to \quad y_i D(x_i) \geq 1 - \delta \quad for \quad i = 1, \ldots, M \quad (4)$$

where C is the margin parameter that controls the trade-off between the training error and the generalization ability, xi are M m-dimensional training inputs and belong to Class 1 or 2 and the associated labels are yi = 1 for Class 1 and -1 for Class 2, and δi (>0) are the slack variables for xi [74].

Advantages of SVMs [74]–[76]:

- Effective in High-Dimensional Spaces: SVMs perform well in high-dimensional feature spaces, making them suitable for tasks involving a large number of features. They can handle complex relationships and capture non-linear decision boundaries through the use of kernel functions.

- Robust against Overfitting: SVMs are less prone to overfitting, especially when the margin is properly regularized. The use of a margin maximization objective helps to generalize well to unseen data, reducing the risk of overfitting.

- Versatile Kernels: SVMs offer flexibility by allowing the use of different kernel functions, such as linear, polynomial, and radial basis function (RBF). This enables modeling of complex patterns and non-linear relationships in the data.

- Support for Outliers: SVMs are effective in handling outliers as they focus on the support vectors, which are the data points nearest to the decision boundary. Outliers have less influence on the model's overall decision, improving its robustness.

Disadvantages of SVMs:

- Computationally Intensive: SVMs can be computationally intensive, especially when dealing with large datasets. Training an SVM requires solving a quadratic optimization problem, which can become time-consuming and memory-intensive for datasets with a large number of samples.

- Sensitivity to Parameter Tuning: SVMs have various parameters, such as the choice of the kernel, regularization parameter (C), and kernel-specific parameters. Proper parameter tuning is crucial for achieving optimal performance. However, selecting appropriate values can be challenging and may require expertise or extensive experimentation.

- Lack of Transparency: SVMs do not provide direct interpretability or intuitive explanations for their decisions. The resulting hyperplane may be difficult to interpret, particularly in high-dimensional spaces. This can limit the interpretability of the model, making it harder to understand the underlying relationships between features and the target variable.

- Imbalanced Class Handling: SVMs may struggle with imbalanced datasets, where one class has significantly more instances than the other. The algorithm tends to prioritize accuracy on the majority class, potentially leading to poor performance on the minority class. Additional techniques, such as class weighting or resampling methods, may be required to address this issue.

2.5.3 Introduction to natural language processing (NLP)

Natural Language Processing (NLP) is a dynamic and rapidly advancing field that focuses on developing algorithms and techniques to enable computers to understand, interpret, and generate human language. The complexity and variability of human language pose significant challenges in NLP. Language is rich in ambiguity, context-dependency, figurative expressions, and cultural nuances, making it difficult for machines to comprehend and accurately process. NLP strives to overcome these challenges by employing a combination of statistical models, machine learning algorithms, and linguistic rules to extract meaning from textual data. In recent years, the field of NLP has seen remarkable advancements due to the advent of deep learning techniques. Neural networks, particularly recurrent neural networks (RNNs) and transformer models, and generative models: Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) have shown impressive capabilities in language understanding, text generation, and machine translation tasks. These models can capture long-range dependencies in language, learn hierarchical representations, and generate coherent and contextually relevant text. However, they also present challenges such as the need for large amounts of annotated training data, computational resources, and the interpretation of their internal mechanisms [77]. RNNs and LSTMs are popular architectures in NLP for sequential data processing. A simple comparison is shown in Figure 2-5. They are particularly useful for tasks such as language modeling, sentiment analysis, named entity recognition, and machine translation. RNNs are designed to capture sequential dependencies in text, making them suitable for tasks where the context and order of words matter. LSTMs, a variant of RNNs, address the vanishing gradient problem and can effectively model long-range dependencies in text [78].

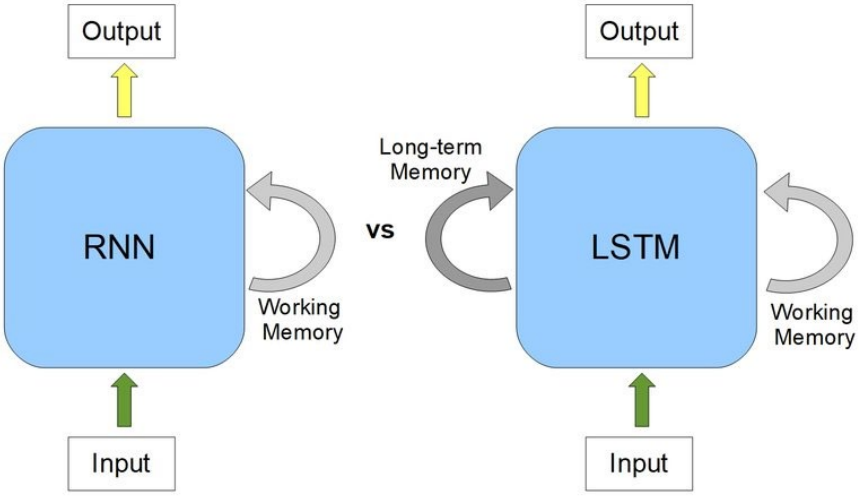


Figure 2-5: RNN vs. LSTM (from Ref.[79])

VAEs, shown in Figure 2-6, on the other hand, are generative models used in NLP for text generation and representation learning. They aim to learn a low-dimensional representation of the input data by encoding it into a latent space. VAEs have been successfully applied to tasks such as text generation, text summarization, and style transfer in NLP. GANs, known for their generative capabilities, have also found applications in NLP. The general idea of autoencoders is pretty simple and consists in setting an encoder and a decoder as neural networks and learning the best encoding-decoding scheme using an iterative optimization process. At each iteration the autoencoder architecture gets fed (the encoder followed by the decoder) with some data, then the model compares the encoded-decoded output with the initial data and backpropagate the error through the architecture to update the weights of the networks [80].

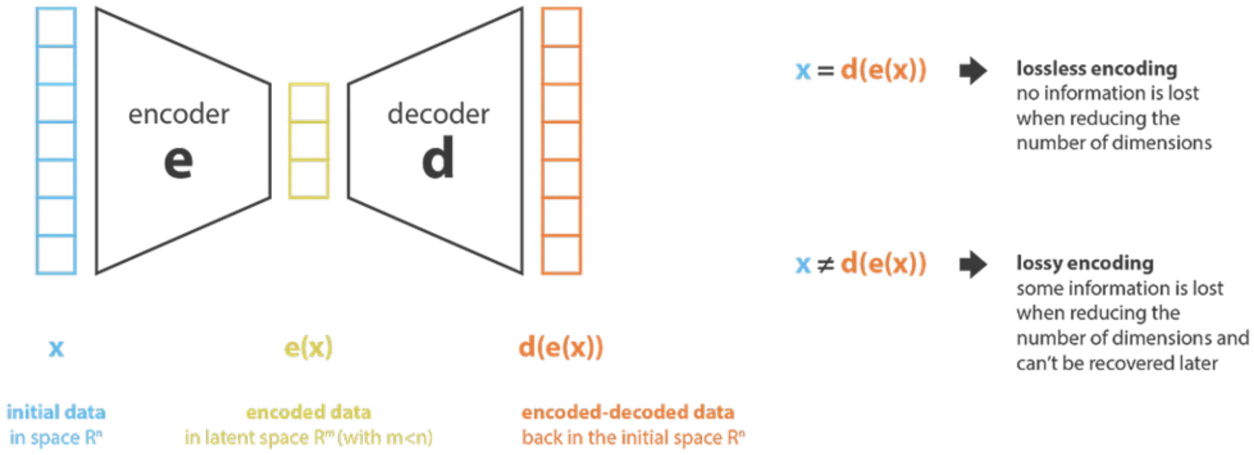

encoder
e
decoder
d
x
e(x)
d(e(x))
initial data
in space R^n
encoded data
in latent space R^m (with m<n)
encoded-decoded data
back in the initial space R^n
$x = d(e(x))$
lossless encoding
no information is lost when reducing the number of dimensions
$x \neq d(e(x))$
lossy encoding
some information is lost when reducing the number of dimensions and can't be recovered later

Figure 2-6: Dimensionality reduction principle with encoder and decoder in a VAE (from Ref.[81])

GANs consist of a generator and a discriminator network that compete against each other with an example shown in Figure 2-7. In NLP, GANs have been used for text generation tasks such as creating realistic text samples, improving language models, and generating adversarial examples for robustness testing [82]. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples [81].

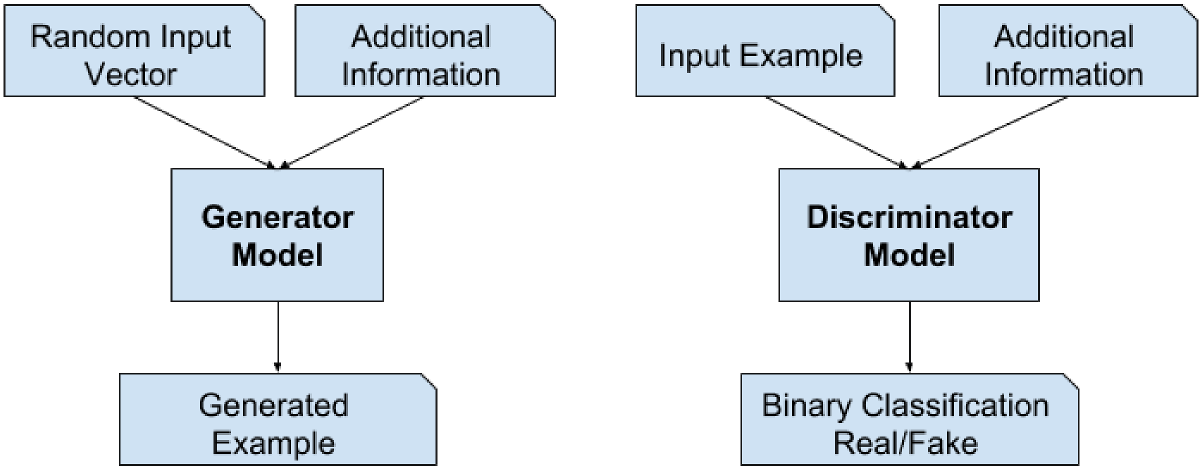


Figure 2-7: Architecture of a GAN model (from Ref. [81])

2.5.4 Role of NLP in lab guiding system

Lab guiding systems play a crucial role in facilitating efficient and accurate laboratory procedures and experiments. With the rapid advancements in NLP technologies, there has been a growing interest in exploring the integration of NLP into various applications, however there are not many related projects elaborating NLP in virtual guiding systems. These systems aim to streamline laboratory workflows, improve experimental accuracy, and enhance the overall user experience. In recent years, there has been a growing interest in leveraging Natural Language Processing (NLP) techniques to enhance the capabilities of lab guiding systems. NLP, as a technology that enables human-computer interaction through natural language, holds significant potential for revolutionizing the way users interact with lab guiding systems [83].

In the context of the VRILE project, NLP is employed to train the virtual instructor to comprehend and generate text in a meaningful and contextual manner. This is done through the use of a transformer-based language model, the GPT-2 model. This model has been trained on a vast amount of web text and can generate human-like text based on the input it receives. For the VRILE project, a dataset was created containing all possible combinations of lab locations and stages of the fatigue test process, along with corresponding prompts. This dataset was used to fine-tune the GPT-2 model to generate appropriate prompts for the specific context of the fatigue test process. Therefore, the role of NLP in VRILE is to provide the AI-driven instructor with the capability to communicate in a way that's easy to understand and natural for the students, generating instructional prompts that are contextually relevant based on the user's current location and stage in the fatigue test process. The adoption of NLP makes the virtual instructor interactive and responsive, thereby contributing to a more immersive and realistic VR learning experience.

2.5.5 Role of GPT-2 in text generation

The Generative Pre-Trained Transformer (GPT) series, developed by Open AI, epitomizes the progression in the domain of transformer-based language models. It features several iterations, each improving on its predecessor's architecture and pre-training objectives. Herein, Zhang et al. [84] present an overview and comparative analysis of these models' cardinal characteristics. The initial iteration, GPT-1, set the foundational architecture for the series, integrating 117 million parameters and undergoing training on a broad corpus of web text under a language modeling objective. This model achieved an unprecedented performance on multiple natural language processing (NLP) tasks, ranging from language modeling to text classification, and extending to question answering. Building upon the initial success, GPT-2 brought increased power and scope, incorporating 1.5 billion parameters and undergoing extensive training on a significantly expanded corpus of web text. The model manifested notable enhancements over GPT-1 in diverse language tasks, particularly with its exceptional capacity to generate coherent and diverse text. The latest iteration, GPT-3, represents a leap in scale and sophistication, boasting over 175 billion parameters, and using an even larger corpus for training. GPT-3 has reported unparalleled achievements on a variety of NLP tasks, including language translation, question answering, and text completion. Particularly noteworthy is its capability for few-shot learning, allowing it to learn new tasks from minimal examples. Figure 2-8 demonstrates the architecture of GPT-1, 2, and 3, including number and types of layers, tokens, and skip connections.

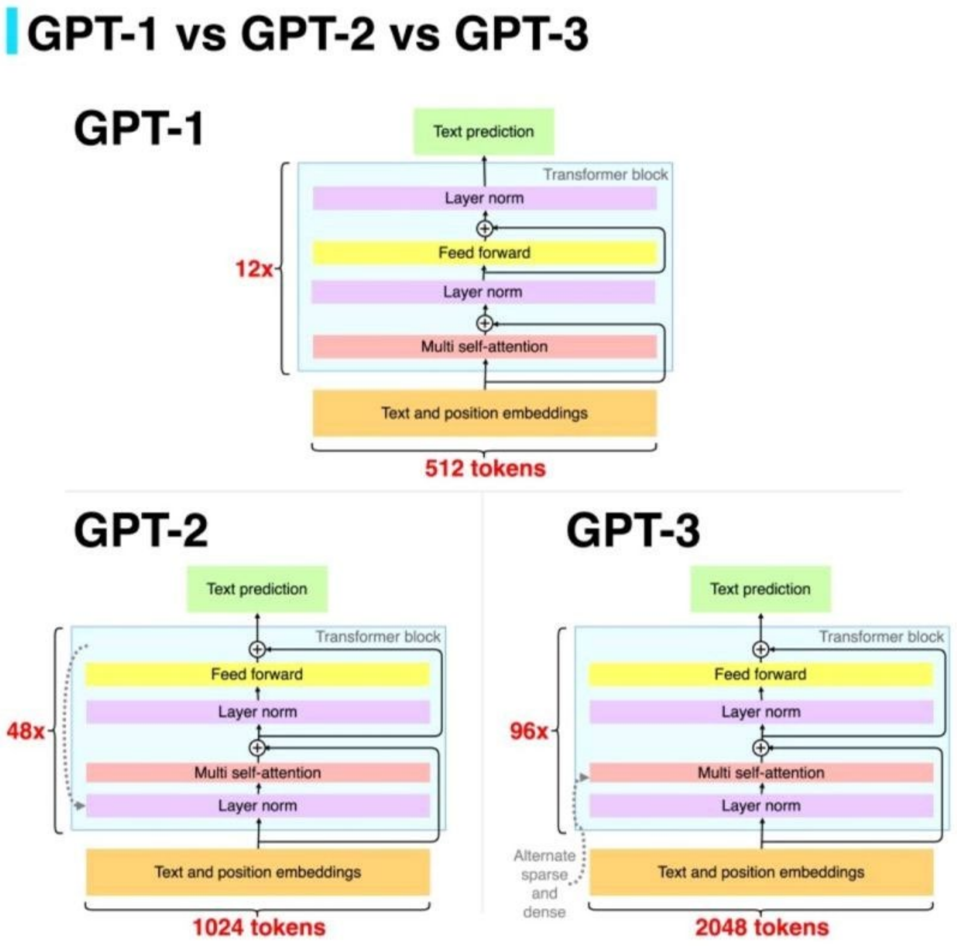


Figure 2-8: Architecture of GPT variants (from Ref. [85])

The progression in GPT series shows consistent improvements in performance across multiple NLP tasks with each new iteration. GPT-3, in particular, has demonstrated significant advancements, especially in few-shot learning. However, an escalating demand for computational resources and training data accompanies each iteration, increasing the difficulty and expense involved in their training. The larger models, specifically GPT-2 and GPT-3, have further stirred discussions about their environmental footprint and the ethical considerations linked with their use [84].

The VRILE project employs the GPT-2 model for a few key reasons. First, the size and complexity of GPT-2 strikes a balance between computational feasibility and linguistic capability. While GPT-2 brings a significant boost in language generation performance compared to GPT-1, it is less computationally demanding and data-intensive than GPT-3. This balance is essential for a project like VRILE, where resources may be limited, and efficiency is important. Second, GPT-2's capacity for generating coherent, diverse, and contextually appropriate text aligns well with the project's goal to provide in-situ personalized guidance to students performing mechanical fatigue tests. This requirement goes beyond simple keyword recognition or fixed responses; the AI needs to generate relevant prompts based on the students' locations in the lab, stages in the testing process, and possibly even past interactions. GPT-2's capabilities make it a suitable choice for this task.

3. TECHNOLOGICAL FOUNDATION

3.1 Unity game engine

Unity allows for the creation of dynamic applications across an array of platforms, encompassing desktop, mobile, and notably, virtual reality. Unity's strength lies in its adaptability and comprehensive toolset, enabling the crafting of deeply immersive experiences distinguished by breathtaking visuals and lifelike physics simulations. It offers support for a broad spectrum of programming languages such as C#, JavaScript, and Boo, thereby providing developers with the flexibility to use the language they are most comfortable with. One of Unity's most salient features is its cross-platform compatibility. Unity empowers developers to construct applications that effortlessly transition across numerous platforms, necessitating minimal adjustments. This versatility renders it an optimal choice for developers seeking to deploy applications across diverse platforms - from conventional desktop and mobile devices to the rapidly evolving domain of virtual reality [86]. Unity extends a robust suite of features and tools that function to assist developers in building top-tier games and applications by combining various assets such as audio files, scripts and 3D models (Figure 3-1). This game engine includes a visual editor for intuitive design and modification, a physics engine for realistic motion and collision simulations, and an array of assets and resources to facilitate the creation of visually striking graphics and hyper-realistic environmental design.

Figure 3-1: 3D models that could be used in the application (from Ref. [86])

Though its roots are in game development, Unity's application has permeated various sectors including architecture, engineering, and education. The software's versatility allows it to be employed in creating a broad scope of applications, from elementary 2D games to intricate virtual reality simulations (Figure 3-2). These simulations offer the potential to create hyper-realistic training scenarios or help in visualizing complex architectural or engineering concepts. In summary, Unity emerges as a potent and adaptable tool that furnishes developers with all the necessary resources to generate superior games and applications. This wide-ranging utility extends across various platforms and industries, testifying to Unity's comprehensive capacity as a premier tool for 3D game development and beyond.

Figure 3-2: An example of 3D modeling with Unity (from Ref. [87])

3.2 VR hardware

VR hardware refers to the physical devices and equipment used to experience virtual reality environments. These hardware components are designed to immerse users in a simulated three-dimensional world and provide a sense of presence and interaction within the virtual environment. For interacting with VRILE a headset or head mounted display and a pair of hand-held controllers are essential.

3.2.1 Head-Mount Display (HMD)

The head-mounted display (HMD), shown in Figure 3-3 is a key component of virtual reality hardware, as it provides the visual display and immersion necessary for the VR experience. Some important aspects of HMDs include:

- Display Technology: HMDs use high-resolution displays, typically OLED or LCD panels, to present the virtual world to the user. These displays are divided into two sections, one for each eye, to create stereoscopic 3D imagery. The resolution and pixel density of the displays impact the level of detail and clarity in the virtual environment.

- Field of View (FoV): The field of view refers to the extent of the virtual world that is visible to the user. A wider field of view provides a more immersive experience by filling the user's peripheral vision. FoV varies among different HMDs, with typical ranges falling between 90 to 120 degrees.

- Tracking Sensors: HMDs incorporate various sensors, such as accelerometers, gyroscopes, and magnetometers, to track the orientation and movement of the user's head in real-time. This allows the user to look around and explore the virtual environment naturally. Some advanced HMDs also include inside-out tracking systems, using built-in cameras and sensors to track the position of the HMD within the physical space.

- Optics and Lenses: HMDs employ specialized lenses to project the virtual images onto the user's eyes properly. These lenses help optimize the field of view, reduce distortion, and minimize the screen door effect (visible gaps between pixels). Different HMDs may offer adjustable lens settings to accommodate users with varying interpupillary distances.

Figure 3-3: Oculus Go standalone VR headset (from Amazon)

3.2.2 Controllers

VR controllers are handheld input devices that enable users to interact with the virtual environment, example controllers are shown in Figure 3-4. They provide a way to manipulate objects, navigate menus, and perform various actions within the VR space. Some key features of VR controllers follow:

- Tracking Technology: Similar to HMDs, VR controllers incorporate tracking technology to capture their position and movement in 3D space. This tracking can be achieved through external sensors, inside-out tracking systems, or a combination of both. Accurate tracking ensures precise control and a natural user experience.

- Buttons and Triggers: VR controllers typically feature buttons, triggers, touchpads, or thumb sticks to provide input commands. These inputs allow users to navigate menus, select options, perform gestures, or activate specific actions within the virtual environment.

- Haptic Feedback: Haptic feedback technology is used in VR controllers to provide tactile sensations to users' hands. Vibrations or force feedback can simulate the sense of touch when interacting with virtual objects, enhancing the immersion and realism of the VR experience.

- Ergonomics: VR controllers are designed with ergonomics in mind to provide comfort during prolonged use. They are shaped to fit naturally in the user's hands and may include straps or grips to secure the controllers during active movements.

- Hand Tracking: Some VR systems are equipped with hand tracking capabilities, allowing users to interact with the virtual environment using their natural hand movements without the need for handheld controllers. This technology recognizes and translates hand gestures and movements into virtual actions.

Figure 3-4: Meta Quest 2 controller (from Amazon)

The specific features and functionalities of VR headsets and controllers vary across different brands and models. Each VR platform may have its own unique design, tracking technology, and input methods. When selecting VR hardware, it's important to consider factors such as compatibility with your computer or gaming system, tracking accuracy, comfort, and the availability of software and content for the specific platform. However, all current controller hardware include the capabilities listed above.

The combination of the HMD and controllers allow for navigating and interacting with the VR environment. Users can move around the environment, pick up items, flip switches, load machines and do all of the actions typically necessary for completing an educational lab with the same, or similar, motions that would be used in the real-life laboratory.

4. VR LAB ENVIRONMENT

4.1 Overview

The Virtual Reality (VR) lab environment, described here integrates VR and Artificial Intelligence (AI) to provide learners with an immersive, interactive, and personalized educational laboratory experience. The blending of VR and AI in education can be leveraged to not only enhance learning outcomes but also to reshape the landscape of experiential learning. Subsequent sections in this chapter provide a comprehensive examination of the VRILE project's features, culminating with a discussion on the pivotal role of the GPT-2 machine learning model in driving the AI-enabled instructional process.

The VRILE project (Virtual Reality Interactive Learning Environment) represents an innovative approach to STEM education, leveraging advancements in VR and NLP. This initiative originated from the recognized need to address the challenges faced in physical lab settings, particularly those concerning safety, resource allocation, and the in-situ guidance. The VRILEs seek to offer an immersive and interactive learning environment, enhancing the laboratory experience and ensuring the mastery of lab protocols among students. The system at its core consists of two key elements. The first is the virtual environment created with the Unity game engine, which serves as the digital representation of the lab. The virtual lab is designed to represent a real-world space in terms of layout, equipment, and tasks. The second key element is an AI-driven virtual instructor, which is based on the GPT-2 model from Hugging Face's transformers library [88]. This AI instructor delivers personalized, step-by-step instructions to guide the students through lab procedures. The VRILE system's objectives extend beyond enhancing the student's familiarity with lab procedures. It is also designed to foster active learning and experiential learning by student engagement through simulating real-world conditions and responses, thereby allowing students to experience and learn from their mistakes in a safe and controlled environment. The application of the GPT-2 model further aids in this by providing dynamic, context-aware prompts based on the student's progress and the virtual lab's state. In summary, the VRILE project combines the immersive capabilities of VR and the predictive power of AI to revolutionize the way mechanical engineering education is delivered. The following sections will describe the features of the VRILE system. Figure 4-1 shows the main desk of the lab, the area where most of the experimental tasks will be performed.

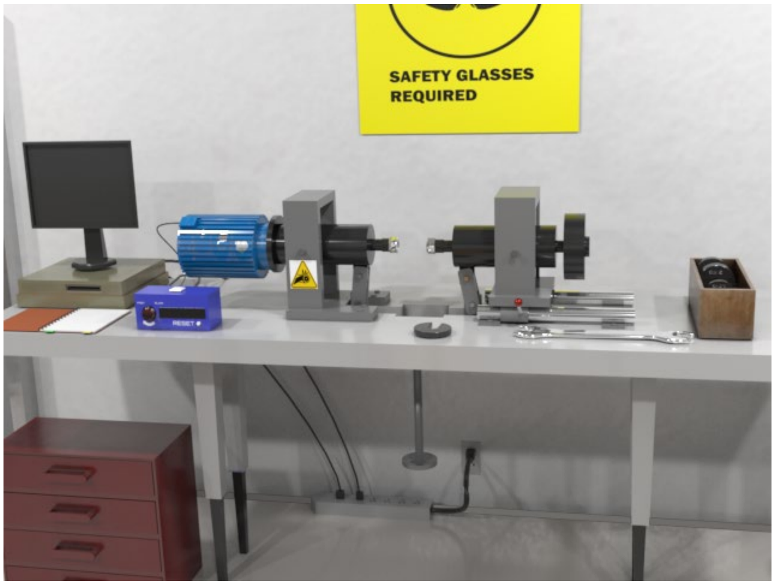


Figure 4-1: The main desk of the VRILE fatigue lab containing the test machine

4.2 Features

The features of the VRILE, each contributing to enhancing student engagement, ensuring safety, increasing accessibility, and providing personalized learning pathways are described in this section. Together, these features build a platform that is not only capable of simulating real-world lab scenarios but also provides an interactive and adaptive learning environment. Figure 4-2 shows a student conducting the fatigue test in VRILE.

- **Enhanced Immersive Learning Environment**: The VRILE project creates a comprehensive immersive learning environment, replicating the physical lab with a high degree of fidelity. This environment not only captures the spatial and visual aspects but also mimics the operations and responses of real-world equipment, offering students a genuine learning experience.

- **Interactivity and Engagement**: The virtual lab equipment is designed to provide a highly interactive experience. The use of VR hardware like headsets and handsets allows for natural and intuitive interaction with the virtual tools and equipment, fostering engagement, active learning, and experiential learning.

- **Safety and Risk Mitigation**: One of the principal advantages of VRILE is the safety it offers. With the virtual environment, students can learn and practice potentially dangerous procedures without any actual risk. Mistakes become opportunities for learning, rather than sources of potential harm or damage.

- **Accessibility and Convenience**: The VRILE system offers unparalleled convenience. It enables learning from any location, at any time, eliminating logistical constraints associated with traditional labs. This feature ensures wider accessibility, enabling more students to gain practical experience.

- **AI-Powered Virtual Instructor**: The AI-driven virtual instructor, built using the GPT-2 model, provides context-aware, personalized instructions. This ensures every student receives in-situ guidance, enhancing the learning process.

- **Scalability**: VRILE is highly scalable. The virtual environment can be readily expanded or modified to include new equipment or procedures, making it a future-proof solution for practical education.

- **Adaptive Learning Paths**: The integration of AI allows for the implementation of adaptive learning paths. The AI instructor can adjust its guidance based on each student's progress and interaction with the virtual environment, offering a personalized and effective learning experience.

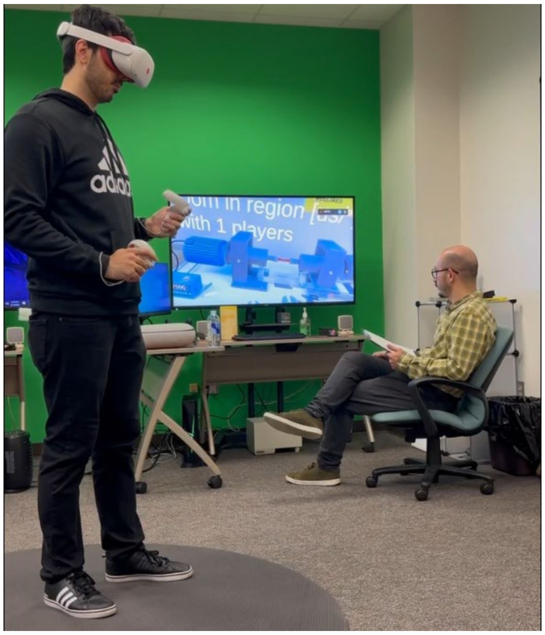


Figure 4-2: A student interacting with VRILE while his activities are streamed for data gathering purposes

4.3 The need for an instructor in the VR lab

An AI instructor in the VRILE allows for an enhanced learning experience, improving accessibility, performing the lab independently and ensuring the optimal use of the virtual environment.

An AI instructor serves as an interactive guide, facilitating the navigation and utilization of the virtual lab. By prompting step-by-step instructions, the AI instructor ensures that users perform the necessary actions accurately and efficiently. Given the complex nature of mechanical tests and lab operations, the assistance of an AI instructor is crucial in ensuring users follow the right procedures, reducing the potential for errors. The AI instructor adjusts its instructions based on the individual's progress and location within the lab. This level of adaptation caters to varying learning paces and styles, fostering a more engaging and effective learning experience. Furthermore, the AI instructor addresses accessibility issues by being available at any time and in any place, providing instruction and guidance whenever the user needs it. This continuous availability mitigates the logistical constraints of having a human instructor and enhances the flexibility of learning, particularly beneficial for remote or asynchronous learning scenarios. Finally, the incorporation of an AI instructor can significantly augment the realism of the VRILE lab, simulating the presence of an actual human instructor and replicating the interaction and communication that typically occur in a real-life lab environment. This can greatly enrich the immersive quality of the VRILE lab, contributing to a more authentic and impactful learning experience. Overall, the inclusion of an AI instructor in the VRILE lab is not merely a value-added feature; it is a critical component that amplifies the learning potential of the virtual lab environment.

4.4 Intelligent virtual instructor

The Intelligent Virtual Instructor (IVI) in a VRILE is an AI-powered entity designed to guide students through their experimental learning journeys in the virtual lab environment. The IVI leverages AI, ML, NLP, and VR, amalgamating these sophisticated technologies to create an intuitive, adaptive, and highly interactive virtual guide.

The IVI is designed around a robust conversational AI model, specifically a variation of the GPT-2 model. This model has been pre-trained on a vast corpus of text data and fine-tuned with domain-specific data from mechanical engineering and education, enabling it to provide accurate, context-aware responses to the students' inquiries and instructions. The model's underlying mechanism is based on the Transformer architecture, a deep learning model architecture renowned for its effectiveness in handling sequential data such as natural language. By leveraging this architecture, the IVI is capable of understanding and generating human-like text, thereby enabling seamless interaction between the students and the virtual lab environment. The creation of IVI involved several steps, starting from the selection and training of the AI model, incorporating it into the Unity game engine, and finally integrating it with the VR hardware. The process also involved continuous testing and fine-tuning to ensure that the IVI can understand and respond to a wide range of queries and commands related to the virtual lab exercises. Importantly, the IVI's construction was guided by principles of instructional design and pedagogy to ensure its effectiveness as a teaching and learning tool. The IVI does not merely provide information; it guides the learner through the process, prompting reflection, encouraging questions, and facilitating deeper understanding. The IVI is designed to adapt to an individual’s needs, allowing for a personalized and immersive learning experience. Figure 4-3 represents the primary stages of the IVI:

- User Query: This is the stage where the student interacts with the IVI, asking questions or giving instructions.
- Stage Understanding: The VRILE leverages classification models to interpret the student's location and test’s stage. The model processes the understood query, retrieves the required information which will be the input to the NLP model.
- Response Generation: The NLP model generates a human-like text response based on the processed query.
- Output Response: The generated response is delivered back to the student, completing the interaction cycle.

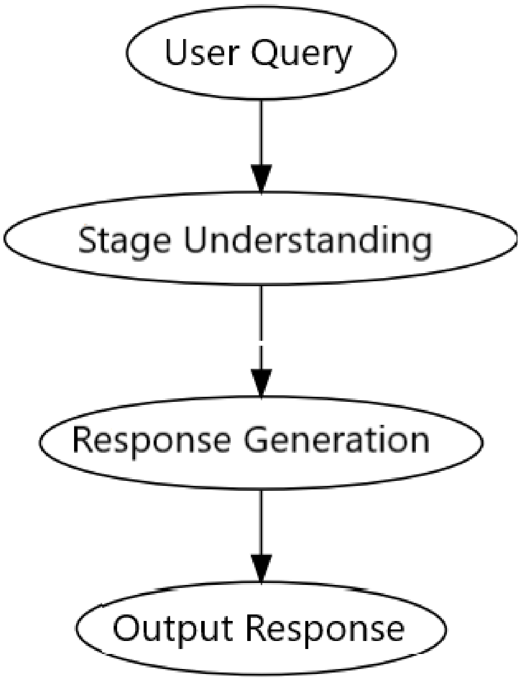


Figure 4-3: Workflow of the IVI in VRILE

In summary, the IVI in VRILE is a novel approach to experiential learning. It leverages advanced AI and VR technologies to create an interactive, adaptive, and highly immersive learning environment, setting the stage for the next generation of education.

4.5 Machine learning model

In the context of the fatigue test VRILE project, employing machine learning serves to intelligently automate the identification of both the user's position within the VR environment and their progression through the stages of the fatigue test. This alleviates the need for labor-intensive, manually programmed logic paths for each possible scenario. Instead, the system learns from the data generated by users' interactions and movements in the VR lab, enabling it to predict subsequent actions, and subsequently trigger the appropriate responses from the Intelligent Virtual Instructor (IVI).

The foundation of the machine learning model in the VRILE system, is a multi-class classification model. Given the discrete and defined nature of both the areas and the stages of the fatigue test, a decision tree model can serve as an effective and interpretably simple solution. A decision tree algorithm would construct a tree-like model of decisions based on the input variables

- in this case, the user's position coordinates and their interactions with different objects. Each node in the decision tree represents a feature (an attribute), each link (branch) represents a decision rule, and each leaf represents an outcome. The topmost node in a decision tree is known as the root node. In Figure 4-4, a combined decision tree approach is illustrated using the VRILE dataset. The figure provides an overview of the decision tree model's structure and its application to the dataset. Figure 4-5, provides a closer examination of the output and the final prompt is depicted. This figure showcases the specific output generated by the decision tree model, highlighting the prompt that is ultimately presented to the user.

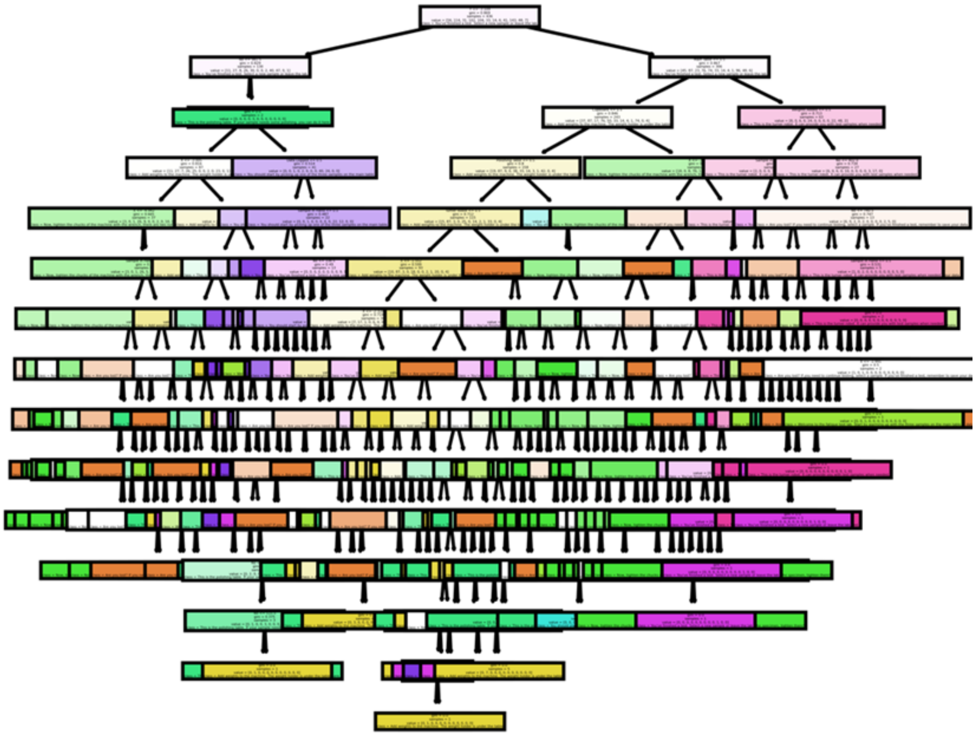


Figure 4-4: Combination of several decision trees which forms a random forest

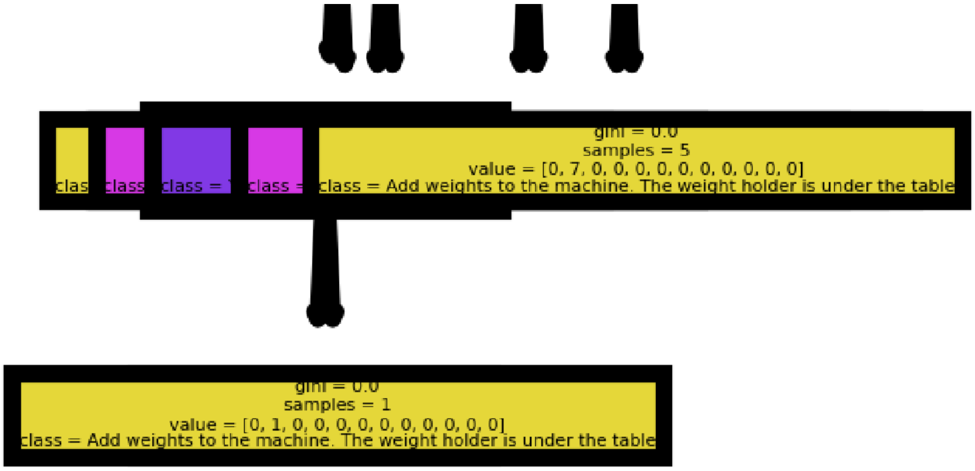

gini = 0.0
samples = 5
value = [0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
clas
class
class =
class =
class = Add weights to the machine. The weight holder is under the table
gini = 0.0
samples = 1
value = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
class = Add weights to the machine. The weight holder is under the table

Figure 4-5: Zoomed-in on the final prediction of the model provide in Figure 16. The identified class (prompt) is: “Add weights to the machine. The weight holder is under the table.”

As the complexity and interactivity of the VRILE project expands, it may become necessary to employ more complex models, such as neural networks, to better capture the nuances in user behavior. However, the benefit of these more sophisticated models must be weighed against their computational cost, as well as the interpretability and simplicity of decision tree models. It is always advisable to start with simpler models and gradually increase the complexity as required by the use case. In the next chapter, different architectures will be employed and compared to figure out the most accurate and efficient model for VRILE.

In terms of learning and pedagogy, such a machine learning approach dovetails nicely with experiential and constructivist learning theories. The system's ability to adapt to each user's unique interaction pattern in the VR lab offers a level of personalization that mirrors the personalized learning experiences that are often advocated in these theories.

5. MODEL IMPLEMENTATION AND DEPLOYMENT

This chapter examines each task involved in constructing the IVI , including understanding the problem space, the selection of appropriate machine learning models, the acquisition and preprocessing of data, and the process of training and fine-tuning these models. Furthermore, it elucidates the crucial step of integrating the trained models into the Unity game engine for real-time interaction within the VR environment. This represents the confluence of numerous machine-learning disciplines, including classification, natural language processing, and model deployment techniques in the context of an immersive VR learning experience.

5.1 Understanding the task

This section focuses on three primary components of the VRILE system's operational model. These tasks constitute the core functionalities that our proposed system must accomplish for effective and engaging virtual lab instructions.

5.1.1 Stage identification

The first task, "Stage Identification," pertains to discerning the current stage of the fatigue test in the virtual lab. As the fatigue test is a multi-step process, it is essential to accurately recognize the stage at which the student is currently working. This identification will help the system understand the student's progress and offer appropriate guidance or correction as needed. The proposed stages for successfully performing a fatigue test within the VRILE are depicted in Figure 5-1.

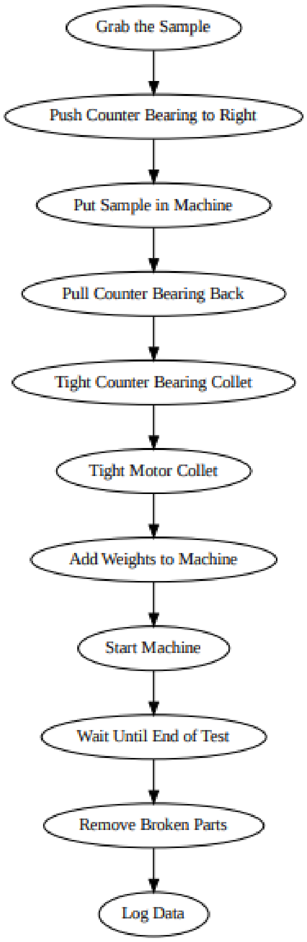
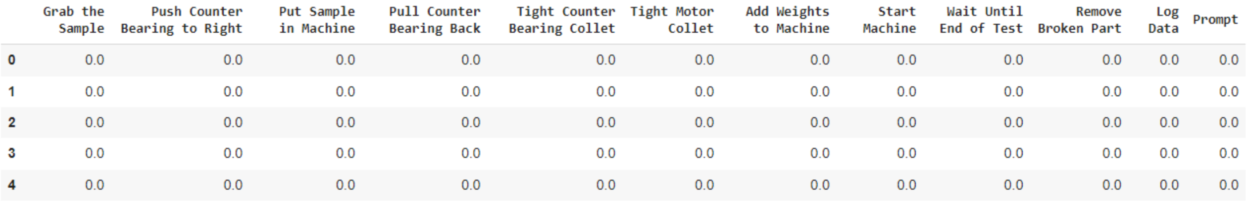


Figure 5-1: Proposed stages for a successful fatigue test

These eleven stages are: “Grab the sample”, “Push counter bearing to right”, “put sample in machine”, “pull counter bearing back”, “Tight the counter bearing collet”, “Tight the motor collet”, “Add weights to the machine”, “Start the machine”, “Wait until end of the test”, “Remove broken parts”, and “Log data.” The simple structure of the data for this task is summarized in Table 5-1. Table 5-1 shows a dataset including 5 samples (the number of rows) and 11 features (the number of columns up to the last column), and the target which is the “Prompt” column.

Table 5-1: A simple example of the data structure for the stage identification level

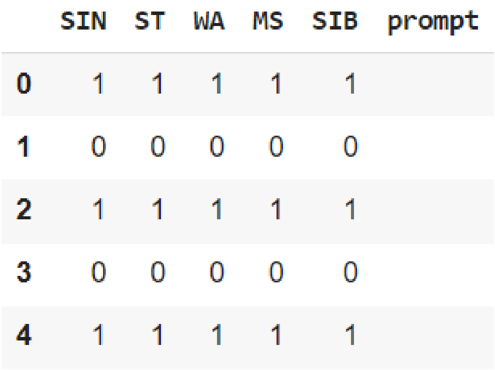


	Grab the Sample	Push Counter Bearing to Right	Put Sample in Machine	Pull Counter Bearing Back	Tight Counter Bearing Collet	Tight Motor Collet	Add Weights to Machine	Start Machine	Wait Until End of Test	Remove Broken Part	Log Data	Prompt
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The dataset consists of binary and text variables. Each feature can be either 0; representing False, deactivated, or off, or 1; representing True, activated, or on. The prompt column consists of different guiding prompts. Each row represents a specific scenario that a student can experience. In the following sections of this research this raw format of the data will be initialized by real data and data augmentation.

However, the large number of features resulted in a feature space with 211 possible combinations, making it computationally intensive to analyze. To address this challenge and streamline the analysis, the feature space reduced by selecting a subset of five key features: SIN, ST, WA, MS, and SIB. These abbreviations represent the following stages in the fatigue test: "Sample in Machine," "Sample Tightened," "Weights Added," "Machine Started," and "Sample Is Broken," respectively. Condensing the original 11 features into this smaller set of five, the resulting feature space decreased 25 possible combinations. This reduction significantly simplified the dataset and enabled more efficient analysis, as it substantially reduced the computational burden associated with exploring the feature space. The new dataset is shown in Table 5-2.

Table 5-2: An example of the data structure with 5 features



	SIN	ST	WA	MS	SIB	prompt
0	1	1	1	1	1	
1	0	0	0	0	0	
2	1	1	1	1	1	
3	0	0	0	0	0	
4	1	1	1	1	1	

The streamlined feature set enabled faster calculations and facilitated more efficient analysis of the fatigue test data. This improvement is particularly relevant in scenarios where the rendering process needs to update values at a specific rate, such as in virtual reality (VR) applications. Excessive calculations during the execution of machine learning models can overwhelm the rendering process, potentially leading to screen stutters or freezing. Any visual stuttering can lead to disorientation or motion sickness for the user.

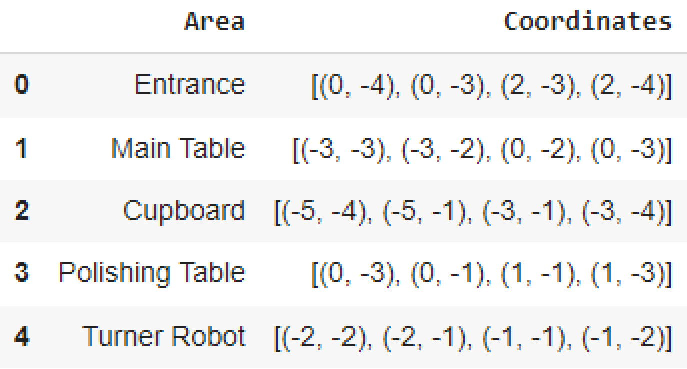
Motion sickness is a common issue experienced by users when their visual perception does not align with their physical sensations, leading to discomfort and nausea [94], [95]. By reducing the computational demands through feature reduction, the risk of inducing motion sickness in VR applications during the rendering process is minimized. This is particularly crucial as the reduction in computational complexity allows for smoother rendering and a more immersive experience for the user.

Consequently, utilizing the reduced feature set improved the speed and feasibility of analyzing the fatigue test data, allowing for faster calculations and streamlined processes. The smaller feature space retained the essential information required for understanding and interpreting the test results while minimizing the complexity and computational requirements. By explaining the rationale behind this feature reduction approach, the practical benefits of simplifying the feature space in the context of fatigue testing would be faster lunch of the program. The reduced feature set not only enables more efficient analysis but also highlights the key stages of the test process, aiding in the interpretation of the results obtained from the dataset.

5.1.2 Location identification

The next task is "Location Identification," which deals with determining the student's location in the virtual lab environment. Since different tasks and pieces of equipment are associated with different areas within the lab, pinpointing the student's location is key to understanding their actions and providing relevant instruction. It will enable the IVI to provide spatially aware guidance, leading to a more immersive and intuitive learning experience. Within the fatigue test room, and considering the current version of VRILE, localization can be introduced as Table 5-3.

Table 5-3: Localization for location identification level



	Area	Coordinates
0	Entrance	[(0, -4), (0, -3), (2, -3), (2, -4)]
1	Main Table	[(-3, -3), (-3, -2), (0, -2), (0, -3)]
2	Cupboard	[(-5, -4), (-5, -1), (-3, -1), (-3, -4)]
3	Polishing Table	[(0, -3), (0, -1), (1, -1), (1, -3)]
4	Turner Robot	[(-2, -2), (-2, -1), (-1, -1), (-1, -2)]

As depicted in Table 3, The VRILE fatigue lab consists of five key areas that simulate a real-life fatigue test lab, aiming to provide an authentic learning experience. These areas, presented in Table 3, are designed to replicate the essential components of a fatigue lab and enable students to engage with the virtual environment. Additionally, there are other objects and areas present within the lab, enhancing the realism without directly affecting the testing process. The first area is the "Entrance," serving as the starting point for students to initiate their experiments. From this location, students can navigate and interact with various elements within the virtual lab. The "Main table" holds the initial test samples and serves as a central hub for preliminary actions. Students can examine and manipulate the samples, conduct equipment setups, and perform necessary tasks related to the fatigue testing process. The "Cupboard" is a designated storage area within the virtual lab, providing students with the option to access additional test samples if required. This area simulates the practicality of having a repository of samples readily available. The "Polishing table" offers students the opportunity to polish their samples before conducting the fatigue tests. This space replicates real-world procedures commonly followed in fatigue testing labs. The "Turner robot" is an interactive element that mimics a sophisticated piece of equipment. It can provide students with test samples tailored to their specific requirements, introducing automation and technology typically found in real fatigue testing labs.

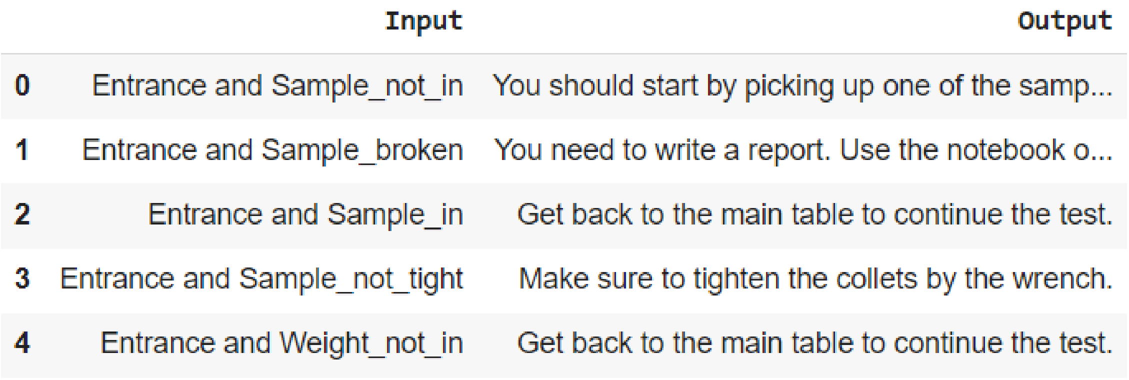
In addition there is also a state referred to as "Wandering." This term captures instances where students explore outside the specific regions, resulting in unpredictable changes in their coordinates. Including this state acknowledges the unstructured and exploratory nature of human behavior within a laboratory setting, contributing to a more realistic learning environment.

By recognizing and incorporating these six distinct stages - "Entrance," "Main table," "Cupboard," "Polishing table," "Turner robot," and "Wandering" - the VRILE offers an immersive and lifelike experience for students. This comprehensive model allows students to engage with the virtual environment, replicate real-world fatigue testing scenarios, and gain a deeper understanding of the subject matter through practical exploration and interaction.

5.1.3 Instruction generation

In the "Instruction Generation" section, the focus is on the process of generating suitable instructional prompts based on the identified stage of the test and the location of the student within the virtual lab. This task requires the implementation of machine learning models capable of recognizing the stage of the test and the student's location as described before. The results obtained from these models contribute to a comprehensive understanding of the fatigue process and the student's interaction with the virtual environment, providing valuable input for the subsequent NLP model. To generate instructional prompts, a dataset consisting of two columns, namely the input and the output, is constructed. The input column describes the student's location within the virtual lab and the stage of the test they are currently at. The output column contains the appropriate guiding prompt corresponding to the given input, shown in Table 5-4 is the first five lines of the generated dataset. By utilizing classification models, accurate identification of the stage and location is achieved, resulting in the creation of a reliable dataset.

Table 5-4: An example of the dataset used for fine-tuning the NPL model



	Input	Output
0	Entrance and Sample_not_in	You should start by picking up one of the samp...
1	Entrance and Sample_broken	You need to write a report. Use the notebook o...
2	Entrance and Sample_in	Get back to the main table to continue the test.
3	Entrance and Sample_not_tight	Make sure to tighten the collets by the wrench.
4	Entrance and Weight_not_in	Get back to the main table to continue the test.

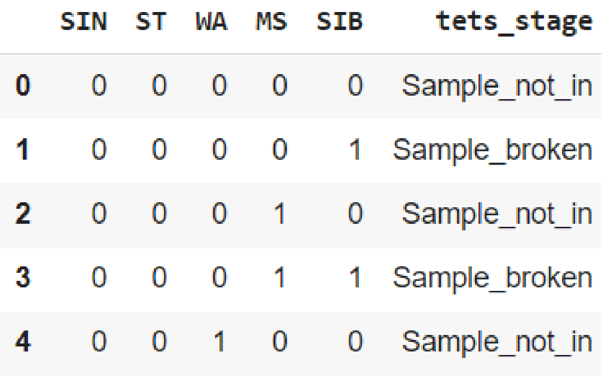
To enhance the quality and naturalness of the generated prompts, fine-tuning techniques are applied to the GPT-2 model. By fine-tuning this language model on the generated dataset, the goal is to generate more coherent, informative, and contextually appropriate instructional prompts. This step directly impacts the effectiveness of the learning experience within the virtual lab environment, ensuring that students receive accurate and helpful guidance throughout the fatigue testing process.

5.2 Choices of model

5.2.1 Classification models

Based on the factors outlined in Section 2.4.2, decision trees and potentially random forests are identified as suitable options for scenarios with constrained hardware resources, where computation speed is crucial, and access to powerful CPUs and GPUs is limited. However, all the four different algorithms introduced in chapter, will be compared in this chapter. To evaluate their performance, a comparison is conducted using a synthetic dataset capable of generating distinct X and Y positions, as well as varied values for the test process. The dataset employed in this analysis is derived from empirical data collected through an experiment involving volunteer students, which will be extensively discussed in subsequent sections. The data structure for stage identification is depicted in Table 5-5 and the summary of the dataset used for location identification is shown in Table 5-6. The complete python code is provided in appendix 1.

Table 5-5: The 5 first lines (head) of the dataset used for stage identification (full dataset in appendix 2)



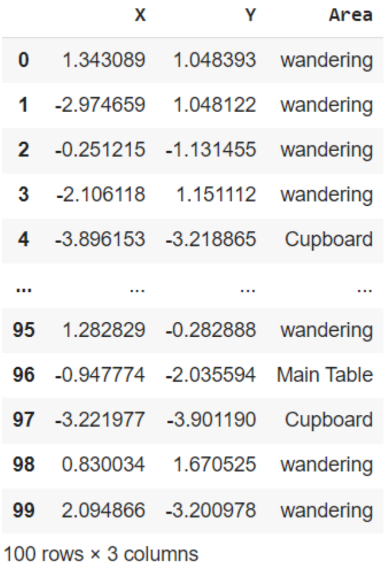
	SIN	ST	WA	MS	SIB	tets_stage
0	0	0	0	0	0	Sample_not_in
1	0	0	0	0	1	Sample_broken
2	0	0	0	1	0	Sample_not_in
3	0	0	0	1	1	Sample_broken
4	0	0	1	0	0	Sample_not_in

The dataset is comma-separated and consists of six columns: SIN, ST, WA, MS, SIB, and test_stage and 49 rows. Each row represents a sample or observation, and the columns represent different features and the corresponding test stage. Here's a breakdown of the columns and their meanings:

- SIM: Represents the value for the feature "SIM" (Sample in Machine). It takes binary values (0 or 1), indicating whether the sample is in the machine or not.
- ST: Represents the value for the feature "ST" (Sample Tightened). It also takes binary values (0 or 1), indicating whether the sample is tightened or not.
- WA: Represents the value for the feature "WA" (Weights Added). It takes binary values (0 or 1), indicating whether weights are added or not.
- MS: Represents the value for the feature "MS" (Machine Started). It takes binary values (0 or 1), indicating whether the machine is started or not.
- SIB: Represents the value for the feature "SIB" (Sample is Broken). It takes binary values (0 or 1), indicating whether the sample is broken or not.
- test_stage: Represents the test stage associated with the sample. It describes the specific stage of the fatigue test process, such as "Sample_not_in," "Sample_broken," "Sample_in," "Sample_not_tight," "Weight_not_in," "Machine_not_on," or "Sample_not_broken."

Each row in the dataset corresponds to a particular scenario that could happen in the test process.

Table 5-6: the dataset used for location identification



	X	Y	Area
0	1.343089	1.048393	wandering
1	-2.974659	1.048122	wandering
2	-0.251215	-1.131455	wandering
3	-2.106118	1.151112	wandering
4	-3.896153	-3.218865	Cupboard
...	...	...	...
95	1.282829	-0.282888	wandering
96	-0.947774	-2.035594	Main Table
97	-3.221977	-3.901190	Cupboard
98	0.830034	1.670525	wandering
99	2.094866	-3.200978	wandering

100 rows × 3 columns

Recalling the areas defined in section 5.1.2 this dataset which is a fake generation of different coordinates, consists of three columns: X, Y, and Area. Here's an explanation of each column:

- X: This column represents the X-coordinate of a location. It ranges from negative values to positive values, indicating positions on a horizontal axis.

- Y: This column represents the Y-coordinate of a location. It ranges from negative values to positive values, indicating positions on a vertical axis.

- Area: This column indicates the area or region associated with the corresponding X and Y coordinates. The values in this column can be one of the following:

  - "wandering": This label is assigned when the X and Y coordinates do not fall within any specific defined area.

  - "Entrance", "Main Table", "Cupboard", "Polishing Table", and "Turner Robot" have defined based on the coordinates provided in Table 5-3.

Note that the dataset contains 100 rows, with each row representing a different location specified by the X and Y coordinates and assigned to a particular area.

Best model for stage identification

After applying the four algorithms to the dataset, their performance was evaluated based on two key metrics: accuracy and the confusion matrix.

- Accuracy: Accuracy measures the overall correctness of the model's predictions. It is calculated by dividing the number of correct predictions by the total number of predictions. A higher accuracy indicates that the model is making more correct predictions [89].

- Confusion Matrix: The confusion matrix provides a detailed breakdown of the model's predictions by showing the number of true positives, true negatives, false positives, and false negatives. It is particularly useful in evaluating the performance of classification models [90].

The accuracy of a model gives an indication of its overall predictive power. A higher accuracy implies that the model is better at correctly classifying instances in the dataset. It is an important metric as it provides a summary of the model's performance in terms of correctness. The confusion matrix, on the other hand, provides more detailed information about the model's predictions. It helps identify the types of errors the model is making. The four components of the confusion matrix are:

- True Positives (TP): The number of instances correctly predicted as positive.
- True Negatives (TN): The number of instances correctly predicted as negative.
- False Positives (FP): The number of instances incorrectly predicted as positive.
- False Negatives (FN): The number of instances incorrectly predicted as negative.

The confusion matrix allows for a deeper analysis of the model's performance, specifically in terms of false positives and false negatives. These can be crucial in certain applications where misclassification of certain classes is more costly or has different implications. By considering both accuracy and the confusion matrix, we gain a comprehensive understanding of the model's performance. High accuracy coupled with a well-balanced confusion matrix, where the model has a low number of false positives and false negatives, indicates a robust and reliable model.

In summary, accuracy provides a summary measure of the model's overall correctness, while the confusion matrix offers detailed insights into the model's performance by breaking down the predictions into true positives, true negatives, false positives, and false negatives. Evaluating both metrics helps in assessing the effectiveness of the models and making informed decisions about their performance in classifying the dataset [89]–[93]. Table 5-7 provide the accuracies of different models implemented on the dataset.

Table 5-7: Comparing classification models on stage identification dataset

Model	Accuracy
Classification	93%
Decision tree	93%
SVM	93%
Bayesian	93%

The similarity in accuracy among the models indicates that they were all successful in learning and generalizing from the dataset. It implies that they were able to identify and utilize the relevant features and patterns to make accurate predictions without overfitting or underfitting the data. The equality in accuracy among the four models could be also influenced by the limited amount of data in the dataset. When the dataset is relatively small, it may contain fewer complex patterns or variations, making it easier for the models to achieve similar levels of accuracy. Figure 5-2 - Figure 5-5 depicts the confusion matrix for each model:

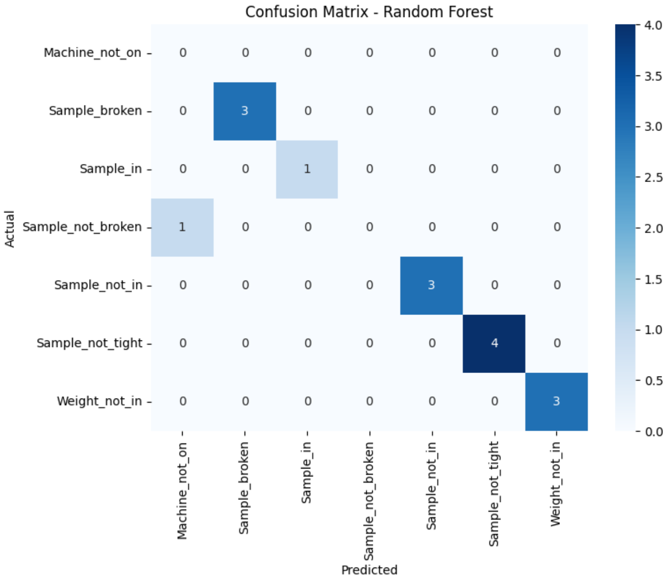


Figure 5-2: Confusion matrix for Random forest algorithm trained over stage identification dataset

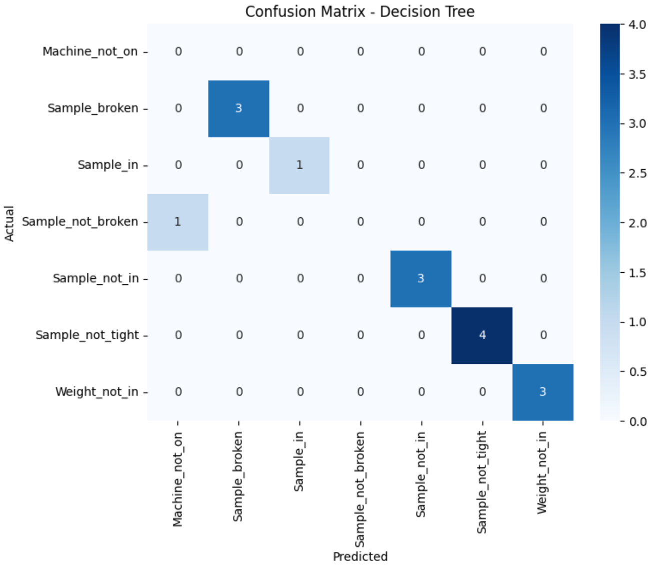


Figure 5-3: Confusion matrix for Decision tree algorithm trained over stage identification dataset

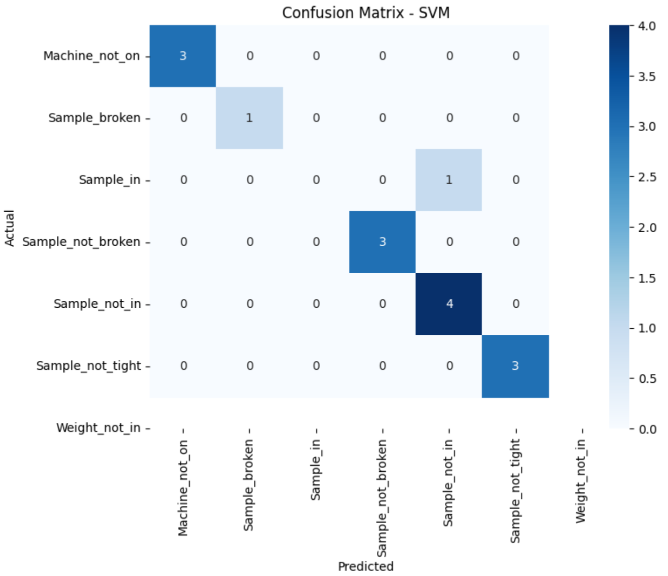


Figure 5-4: Confusion matrix for SVM algorithm trained over stage identification dataset

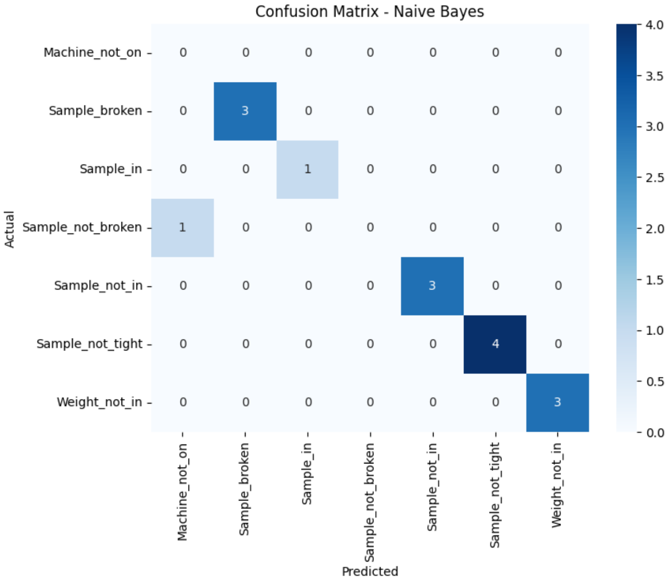


Figure 5-5: Confusion matrix for Naïve Bayes algorithm trained over stage identification dataset

In the SVM confusion matrix, we can observe that:

- Class 0 (Machine_not_on) is correctly predicted 3 times.
- Class 1 (Sample_broken) is correctly predicted 1 time.
- Class 2 (Sample_in) is never correctly predicted.
- Class 3 (Sample_not_broken) is correctly predicted 3 times.

- Class 4 (Sample_not_in) is correctly predicted 4 times.
- Class 5 (Sample_not_tight) is correctly predicted 3 times.
- Class 6 (Weight_not_in) is never correctly predicted.

In the Naive Bayes, Decision Tree, and Random Forest confusion matrices, we can observe that:

- Class 0 (Machine_not_on) has never correctly predicted.
- Class 1 (Sample_broken) is correctly predicted 3 times.
- Class 2 (Sample_in) is correctly predicted 1 time.
- Class 3 (Sample_not_broken) has never correctly predicted.
- Class 4 (Sample_not_in) is correctly predicted 3 times.
- Class 5 (Sample_not_tight) is correctly predicted 4 times.
- Class 6 (Weight_not_in) is correctly predicted 3 times.

Based on the confusion matrices, we can compare the performance of SVM and other models as follows:

- SVM model has higher accuracy in predicting Class 0, Class 3, Class 4.
- Other models have higher accuracy in predicting Class 1, Class 2, Class 5, and Class 6.

Based on the results obtained from the dataset, it can be concluded that SVM is the weakest performing model among the four models evaluated. However, when comparing the performance of the remaining three models (Decision Tree, Random Forest, and Naive Bayes), they exhibit similar accuracy and confusion matrices, indicating similar behavior in making predictions. Given the similarity in accuracy and confusion matrices, opting for the simplest model, which is the Decision Tree, can offer advantages in terms of computational efficiency and time-saving. The Decision Tree model is known for its simplicity and interpretability, making it easier to understand and analyze the underlying decision-making process. Additionally, the Decision Tree model requires less computational resources compared to more complex models like Random Forest.

Therefore, in the context of this dataset, choosing the Decision Tree model can be a favorable choice due to its simplicity and computational efficiency, without compromising on the overall performance.

Best model for location identification

Table 8 shows accuracies for each model after training and evaluation on the dataset. From the accuracy scores, we can see that both Decision Tree and Random Forest models have the highest accuracy of 83.33%, followed by Naive Bayes with 80% accuracy, and SVM with 70% accuracy.

Table 5-8: Comparing classification models on location identification dataset

Model	Accuracy
Random Forest	83.33%
Decision Tree	83.33%
SVM	70%
Bayesian	80%

By analyzing the confusion matrices shown in Figure 5-6 – Figure 5-9, it can be concluded that:

- Decision Tree: It has some misclassifications, particularly in predicting the second class.
- Random Forest: It also has misclassifications, especially in predicting the first and fourth classes.
- SVM: It has misclassifications in predicting the first, fourth, and fifth classes.
- Naive Bayes: It shows misclassifications in predicting the first, third, and fifth classes.

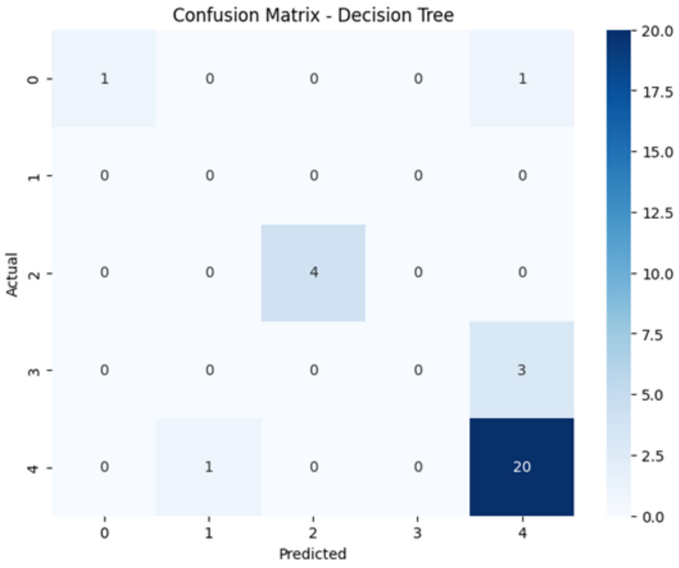


Figure 5-6: Confusion matrix for Decision tree algorithm trained over location identification dataset

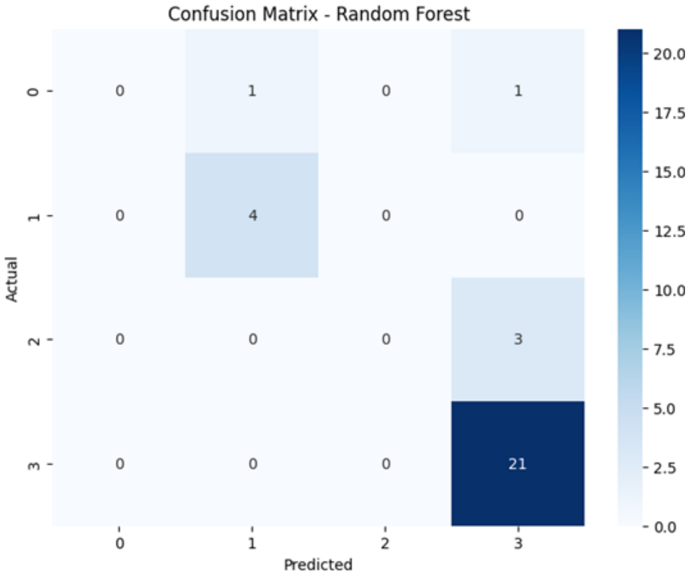


Figure 5-7: Confusion matrix for Random Forest algorithm trained over location identification dataset

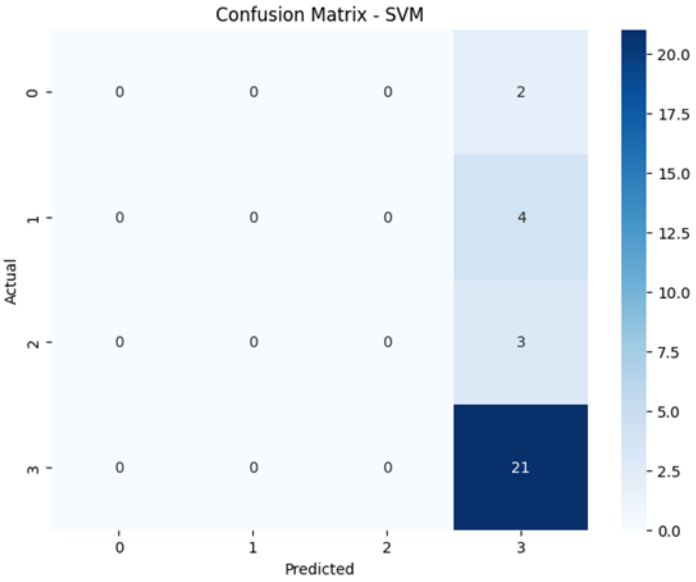


Figure 5-8: Confusion matrix for SVM algorithm trained over location identification dataset

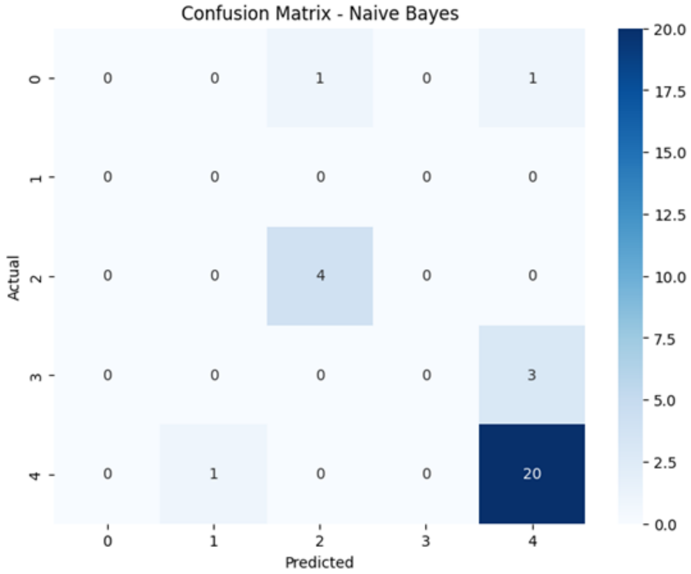


Figure 5-9: Confusion matrix for Naïve Bayes algorithm trained over location identification dataset

Overall, Decision Tree and Random Forest models have similar accuracy and comparable performance, but Decision Tree seems to have slightly better performance as it has fewer misclassifications compared to Random Forest. SVM and Naive Bayes models show lower accuracy and higher misclassifications in comparison. Recalling the results achieved in the previous section, the model used for classification task for this study is chosen to be the Decision Tree.

5.2.2 Text generator model

The Generative Pre-trained Transformer (GPT) model is a deep learning model that utilizes the power of transfer learning to generate coherent and contextually relevant text. In this chapter, we will delve into the world of GPT models, their capabilities, and compare different versions of GPT to select the most suitable model for the project. GPT takes advantage of transfer learning by pre-training a language model on a massive amount of text data and then fine-tuning it on specific downstream tasks such as text generation, translation, or sentiment analysis. This pre-training enables GPT to learn the underlying patterns and structures of language, making it proficient at generating coherent and contextually relevant text [94]. Transfer learning is a technique where a model pre-trained on a large corpus of data is fine-tuned on a specific task [84], [94].

GPT-1

GPT-1, the first version of the model, demonstrated impressive language generation capabilities. It had 117 million parameters and showcased the potential of large-scale language models in generating coherent text. However, as subsequent versions were released, GPT-1 became less prominent due to its comparatively smaller size and limitations in capturing complex language nuances.

GPT-2

GPT-2 marked a significant breakthrough in text generation. With 1.5 billion parameters, GPT-2 showcased exceptional proficiency in generating high-quality, contextually relevant text. It captured long-range dependencies in language, resulting in coherent and human-like responses. GPT-2 was widely acclaimed for its ability to generate paragraphs of text that were often indistinguishable from human-written content.

GPT-3

GPT-3 took text generation to unprecedented heights. With a staggering 175 billion parameters, GPT-3 surpassed its predecessors in both size and performance. It exhibited remarkable language understanding and generation capabilities, capable of producing contextually relevant and coherent responses across various domains. GPT-3 showcased its versatility by excelling in tasks such as text completion, translation, and even creative writing. However, it is important to note that GPT-3 is not freely available and requires API access for utilization.

GPT-4

GPT-4, the most recent version at the time of writing, further pushed the boundaries of text generation. With even larger model size and enhanced capabilities, GPT-4 promises advancements in various NLP tasks. However, similar to GPT-3, GPT-4 is not freely available and requires API access.

Considering the various versions of GPT, it is important to choose a model that aligns with the project's requirements. While GPT-3 and GPT-4 offer exceptional performance, their limited accessibility due to API usage restrictions may hinder their applicability in certain scenarios. Moreover, considering the computational needs, GPT-2 stands out as a favorable choice for many projects. It strikes a balance between model size, performance, and computational requirements. GPT-2's 1.5 billion parameters allow for generating high-quality text while still being manageable in terms of computational resources. Its training can be achieved on high-end GPUs, which are relatively accessible compared to specialized hardware setups required for GPT-3. GPT-2 has become widely adopted due to its efficient resource utilization and impressive text generation capabilities [95]. Therefore, for this project, GPT-2 was selected.

5.3 Data

The data for this project was gathered through an experimental test conducted in a real-world setting. The test involved collecting key points and observations related to the fatigue test by the students. These key points formed the main backbone of the data structure, providing the initial dataset for analysis.

5.3.1 Experimental test

The experimental test focused on investigating the behavior of six volunteer students from the engineering school at IUPUI University. Prior to the test, the students were provided with a standardized explanation of how to use the virtual reality (VR) tools and interact with objects within the VR environment. During the experiment, each student was tasked with performing a single fatigue test using one of the samples located on the main desk within the VR environment. Their activities and interactions within the VR environment were visually streamed on a screen for the real-life instructor . The primary objective of this experiment was twofold: first, to identify instances when the students required assistance, and second, to determine appropriate guidance that could be provided to the students throughout the remainder of the test. Figure 5-10 shows 3 of the volunteer students performing the fatigue test in VRILE while their activities are streamed and recorded for later investigation.

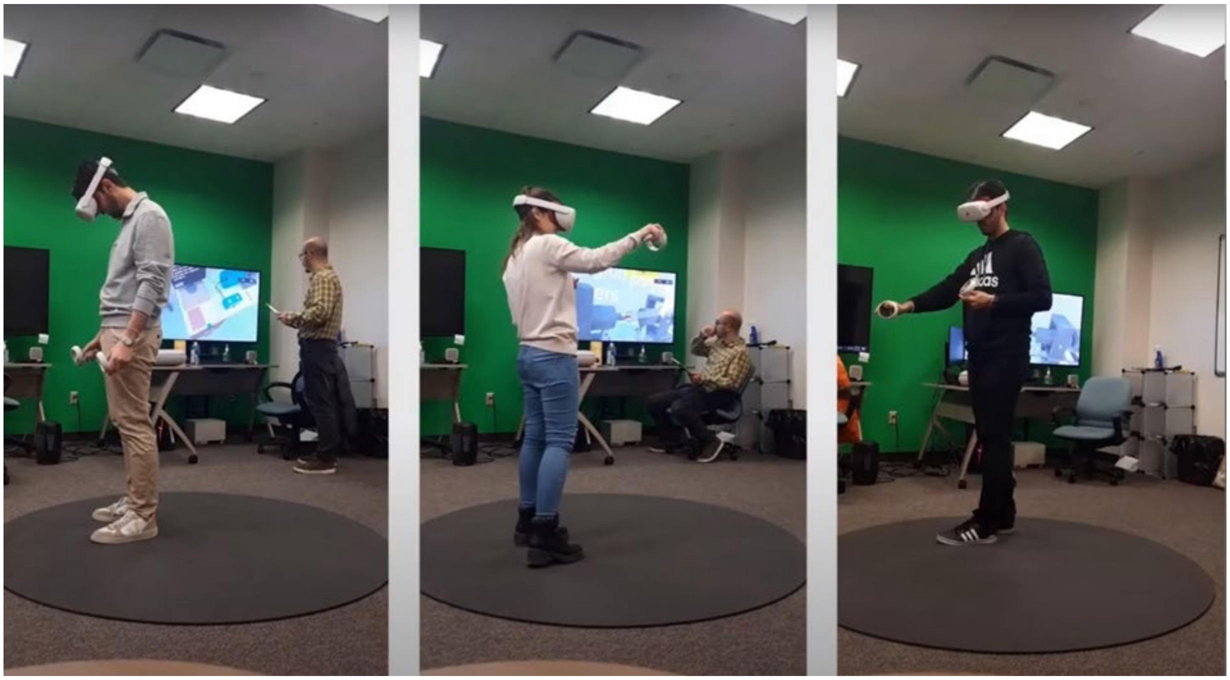


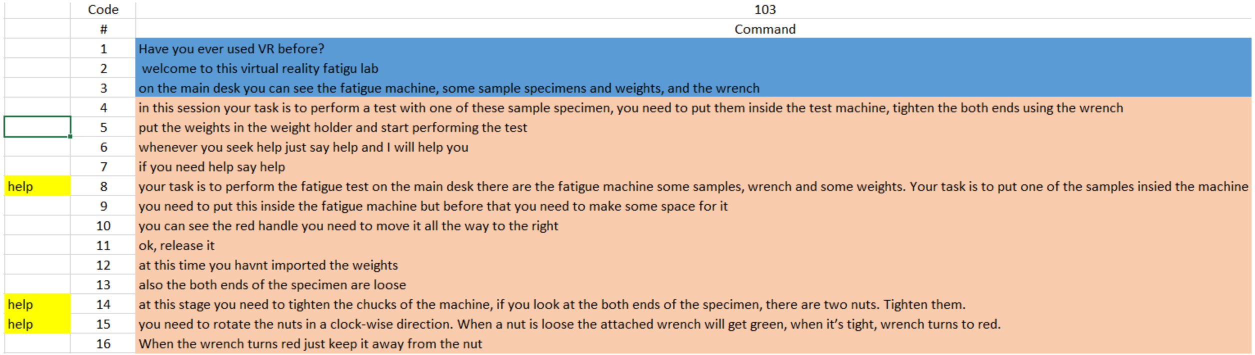
Figure 5-10: Volunteer students during the experiment at IUPUI University’s VR lab

Whenever a student requested assistance, a real-world instructor, observed the student's current stage within the test and delivered suitable prompts to guide the student accordingly. This interactive process continued until the student successfully completed the test and saved the corresponding data using a notebook.

Throughout the experiment, all the activities taking place within the VR environment were streamed and recorded. Additionally, all conversations and interactions between the students and the real-life teacher were recorded for further analysis. Subsequently, a thorough investigation of the recorded conversations and interactions pertaining to the students' positions and test stages formed the fundamental structure of the dataset required for this project.

By conducting this experimental test, valuable insights were obtained regarding the students' behavior and performance during the fatigue test in the VR environment. The recorded data and interactions provided essential information for the subsequent stages of data analysis and model development in the project. The conversation between the student and teacher divided into several stages including introduction to VR environment, introduction to the fatigue test lab, stage-based and area-based questions. An example of one of the initial datasets obtained from the experiment is shown in Table 5-9.

Table 5-9: An example of the data obtained from the experimental test



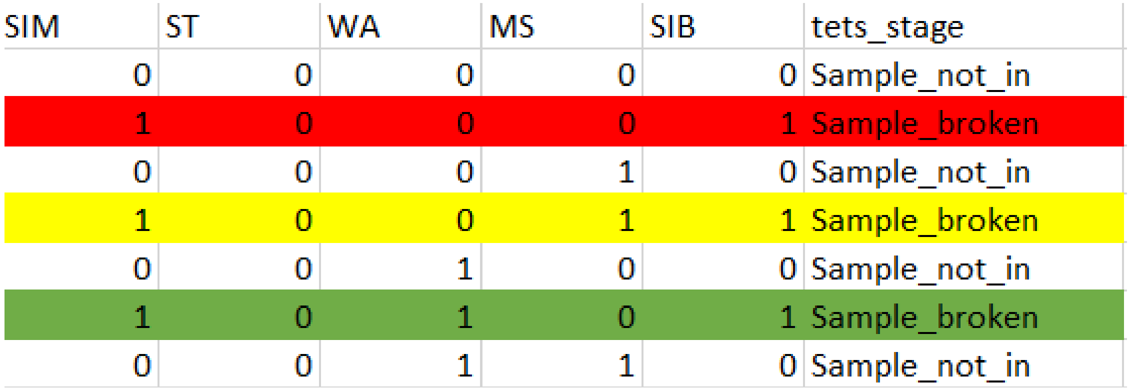
	Code	103
	#	Command
	1	Have you ever used VR before?
	2	welcome to this virtual reality fatigu lab
	3	on the main desk you can see the fatigue machine, some sample specimens and weights, and the wrench
	4	in this session your task is to perform a test with one of these sample specimen, you need to put them inside the test machine, tighten the both ends using the wrench
	5	put the weights in the weight holder and start performing the test
	6	whenever you seek help just say help and I will help you
	7	if you need help say help
help	8	your task is to perform the fatigue test on the main desk there are the fatigue machine some samples, wrench and some weights. Your task is to put one of the samples insied the machine
	9	you need to put this inside the fatigue machine but before that you need to make some space for it
	10	you can see the red handle you need to move it all the way to the right
	11	ok, release it
	12	at this time you havnt imported the weights
	13	also the both ends of the specimen are loose
help	14	at this stage you need to tighten the chucks of the machine, if you look at the both ends of the specimen, there are two nuts. Tighten them.
help	15	you need to rotate the nuts in a clock-wise direction. When a nut is loose the attached wrench will get green, when it’s tight, wrench turns to red.
	16	When the wrench turns red just keep it away from the nut

These observations classified based on the stage of the test and area which the student stays in and eventually led into the data structure former described in section 5.2.1. Another important achievement of the experimental test was to figure out the relative superiority of the features. Analyzing the performance of the students, it was figured out that the determiner features in stage identification do not have same importance. For example, in Table 5-10 the highlighted samples describe different scenarios:

- In Red, sample is in machine and sample is broken.
- In Yellow, sample is in machine, machine is started, and sample is broken.
- In green, sample is in machine, weights are added, and sample is broken.

For this specific example as the SIB feature has the most important among other activated features, the test stage for all the above scenarios is affected by the status of this feature and is “sample is broken”. In other scenarios, priority of the features is considered as well.

Table 5-10: Different probable scenarios and their corresponding stages



SIM	ST	WA	MS	SIB	tets_stage
0	0	0	0	0	Sample_not_in
1	0	0	0	1	Sample_broken
0	0	0	1	0	Sample_not_in
1	0	0	1	1	Sample_broken
0	0	1	0	0	Sample_not_in
1	0	1	0	1	Sample_broken
0	0	1	1	0	Sample_not_in

5.3.2 Data augmentation

As the observation through the experimental data was limited due to the constraints of the real-world experiment, additional steps were taken to enhance the dataset through data augmentation techniques. In order to expand the dataset and capture a wider range of scenarios and stages, Python was utilized to generate synthetic or "fake" data based on the underlying logic of the problem. By leveraging the knowledge gained from the experimental test, a set of rules and guidelines was established to generate simulated data that emulates different stages and scenarios encountered during the fatigue test in the VR environment. These rules were implemented in the form of algorithms and scripts, allowing for the generation of a diverse range of synthetic data points. Python code is provided in Appendix 3.

The generated fake data complemented the original experimental data by providing additional samples that cover various stages of the test. This augmentation process served to enrich the dataset, enabling a more comprehensive analysis and training of the models. The synthetic data, while not obtained directly from real-world observations, was carefully designed to capture the essence of the problem and mimic realistic patterns and behaviors. It is important to note that data augmentation through synthetic data generation is a common practice in machine learning and data science. It helps overcome limitations in the available data and allows for a more robust analysis and modeling process. By incorporating the generated fake data alongside the experimental data, the dataset became more representative of different scenarios and stages, thus enhancing the overall validity and reliability of the findings and model performance.

5.3.3 Data preprocessing

The preprocessing process in the classification aspect plays a crucial role in preparing the dataset for the decision tree classifier. It involves handling missing values and encoding categorical variables. Here is a detailed explanation of the preprocessing steps:

- Handling Missing Values: Before applying any further analysis, it is important to address missing values that might exist in the dataset. In this research, a simple imputation technique called "most_frequent" strategy is employed. This technique identifies any missing values within the features and replaces them with the most frequently occurring value for each respective column. By doing so, the dataset becomes completer and more suitable for subsequent analysis.

- Encoding Categorical Variables: Categorical variables, such as the target variable in this research denoted as "y," are encoded into numerical labels to facilitate effective processing by the decision tree classifier. To achieve this, a label encoder is employed. The label encoder maps each unique category in the categorical variable to a corresponding numerical label. This transformation allows the decision tree classifier to interpret and analyze the categorical data during the training and prediction stages.

5.3.4 Data postprocessing

The post-processing technique employed in this research involves reversing the label encoding process to obtain the predicted class labels in their original categorical form.

During the preprocessing phase, the categorical target variable (y) was encoded into numerical labels using the LabelEncoder. This encoding allows the decision tree classifier to process the data effectively. However, to interpret and present the predicted class labels in their original categorical format, the label encoding needs to be reversed. After making predictions on the test data using the decision tree classifier, the predicted class labels (y_pred) are obtained. To transform these numerical labels back to their original categorical values, the inverse_transform method of the LabelEncoder is applied. This process maps the numerical labels back to their corresponding original categories, enabling the interpretation and presentation of the predicted class labels in their familiar form. Figure 5-11 simply shows the encoding and decoding process.

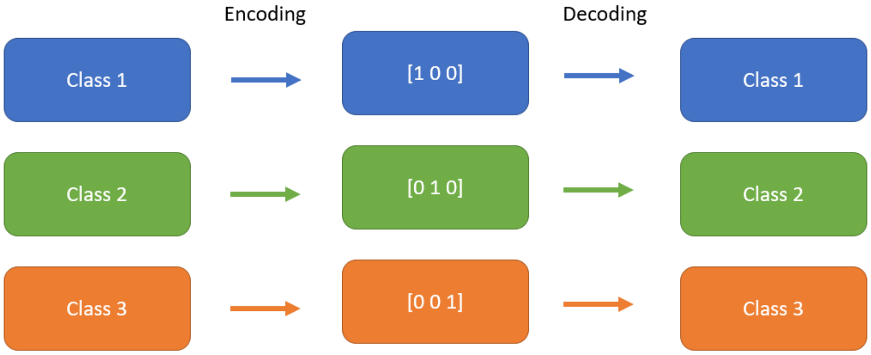


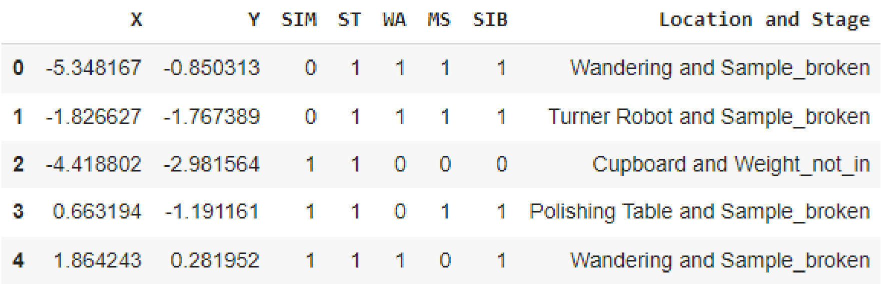
Figure 5-11: Label encoding-decoding process

By reversing the label encoding, the program can provide meaningful and understandable results, allowing for the NLP model to receive these values as inputs and process them. This post-processing technique ensures that the predicted class labels are presented in a format that aligns with the original representation of the target variable. It facilitates the interpretation and communication of the research findings in a manner that is consistent with the domain-specific understanding of the categorical labels.

5.3.5 Data structure

The dataset utilized in this research exhibits a structure with 1000 samples generated through data augmentation techniques. Initially, the dataset consisted of 37 samples, and by leveraging data augmentation, it was expanded to achieve a larger and more diverse collection of samples. The dataset incorporates two classification algorithms that were previously discussed, resulting in an expansion of the feature space to include 7 distinct features. These features capture essential characteristics and attributes relevant to the classification task. Table 5-11 shows the first five rows of the dataset.

Table 5-11: The first five rows (head) of the dataset



	X	Y	SIM	ST	WA	MS	SIB	Location and Stage
0	-5.348167	-0.850313	0	1	1	1	1	Wandering and Sample_broken
1	-1.826627	-1.767389	0	1	1	1	1	Turner Robot and Sample_broken
2	-4.418802	-2.981564	1	1	0	0	0	Cupboard and Weight_not_in
3	0.663194	-1.191161	1	1	0	1	1	Polishing Table and Sample_broken
4	1.864243	0.281952	1	1	1	0	1	Wandering and Sample_broken

The final column of the dataset represents two crucial aspects: the area or location within the test room and the stage of the test. The test room is divided into various areas, each associated with a unique location. Simultaneously, the test stage encompasses different stages or conditions observed during the test. To capture the diversity present in the test room and the test stages, the dataset defines 35 distinct classes. Each class is specified by combining the area and the corresponding test condition. The classes are as follows:

1. Entrance and Sample_not_in
2. Entrance and Sample_broken
3. Entrance and Sample_in
4. Entrance and Sample_not_tight
5. Entrance and Weight_not_in
6. Entrance and Machine_not_on
7. Entrance and Sample_not_broken
8. Main Table and Sample_not_in
9. Main Table and Sample_broken
10. Main Table and Sample_in
11. Main Table and Sample_not_tight
12. Main Table and Weight_not_in
13. Main Table and Machine_not_on

14. Main Table and Sample_not_broken
15. Cupboard and Sample_not_in
16. Cupboard and Sample_broken
17. Cupboard and Sample_in
18. Cupboard and Sample_not_tight
19. Cupboard and Weight_not_in
20. Cupboard and Machine_not_on
21. Cupboard and Sample_not_broken
22. Polishing Table and Sample_not_in
23. Polishing Table and Sample_broken
24. Polishing Table and Sample_in
25. Polishing Table and Sample_not_tight
26. Polishing Table and Weight_not_in
27. Polishing Table and Machine_not_on
28. Polishing Table and Sample_not_broken
29. Turner Robot and Sample_not_in
30. Turner Robot and Sample_broken
31. Turner Robot and Sample_in
32. Turner Robot and Sample_not_tight
33. Turner Robot and Weight_not_in
34. Turner Robot and Machine_not_on
35. Turner Robot and Sample_not_broken

The dataset's class definition captures the intricate relationship between the location and the stage of the test, providing distinct labels for each combination. This enables the classification algorithms to learn and predict the appropriate class label based on the provided features and the associated area and test stage. The target values or the last column which contains the 35 classes will be the input to the NLP model which will be discussed in following sections.

5.4 Model training

The model training process involves training a decision tree classifier using the training data. Here's a description of the model training steps:

- Splitting the Dataset: The dataset is split into training and testing sets using the train_test_split function from sklearn.model_selection. The split is performed with a 70% training and 30% testing ratio, indicated by test_size=0.3.

- Creating the Decision Tree Classifier: A decision tree classifier is created using the DecisionTreeClassifier class from sklearn.tree. This classifier is a popular machine learning algorithm that builds a tree-like model of decisions and their possible consequences.

- Training the Classifier: The decision tree classifier is trained on the training data using the fit method. This process involves analyzing the features (X_train) and their corresponding target labels (y_train) to learn patterns and build the tree model. The classifier uses an algorithm to determine the optimal splitting criteria for each node in the tree.

- Completing the Model Training: After the fit method is called, the decision tree classifier adjusts its internal parameters and structure based on the provided training data. The classifier learns from the features and their associated target labels to make accurate predictions on unseen data.

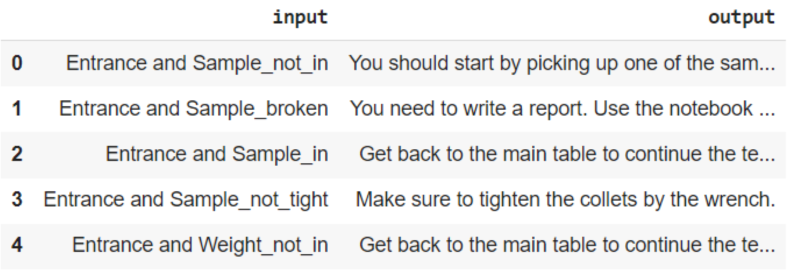
The model training process achieved accuracy of 92% over the augmented dataset. This accuracy indicates that the model is performing well and making accurate predictions, showcasing its effectiveness in classifying the samples. The achieved accuracy of 92% indicates that the model is successfully capturing the underlying patterns and relationships within the augmented dataset, providing a solid foundation for accurate predictions. This level of accuracy is considered acceptable at this stage and demonstrates the potential of the model for accurate classification in real-world scenarios.

The complete Python code used for the model training can be found in Appendix 4, enabling readers to review and replicate the implementation. The code includes the necessary steps for loading the dataset, preprocessing the data to handle missing values, encoding categorical variables, splitting the dataset into training and testing sets, creating and training the decision tree classifier, and evaluating the accuracy of the model.

5.5 NLP dataset

The provided dataset is designed for training an NLP (Natural Language Processing) model which is briefly shown in Table 5-12. It consists of input-output pairs, where the input represents a specific area and a condition, and the corresponding output provides instructions or guidance for the given situation.

Table 5-12: The head of the NLP dataset



	input	output
0	Entrance and Sample_not_in	You should start by picking up one of the sam...
1	Entrance and Sample_broken	You need to write a report. Use the notebook ...
2	Entrance and Sample_in	Get back to the main table to continue the te...
3	Entrance and Sample_not_tight	Make sure to tighten the collets by the wrench.
4	Entrance and Weight_not_in	Get back to the main table to continue the te...

The dataset comprises input-output pairs that cover various scenarios within a lab environment. Each input consists of two components, the area and the condition. The areas represent different locations in the lab, such as the entrance, main table, cupboard, polishing table, and Turner robot. The conditions describe specific circumstances related to the test or equipment, including sample presence, sample condition, collet tightness, weight presence, and machine status. The corresponding outputs are instructional or informative sentences that guide the user in addressing the particular scenario described by the input. The instructions cover a range of actions to be taken, such as picking up samples, writing reports, tightening collets, adding weights, turning on machines, and returning to the main table for further testing.

The dataset provides a comprehensive set of scenarios and corresponding instructions, allowing the NLP model to learn and generate appropriate responses based on the given input conditions. It is intended to enable the model to understand and respond to various situations within the lab environment accurately.

By processing the input descriptions, the trained model can generate relevant instructions or responses based on the provided scenarios, contributing to enhanced operational efficiency and accuracy within the lab setting.

5.6 NLP preprocessing

The preprocessing aspect in the provided transfer learning code involves tokenization and preparing the data for language modeling.

- Tokenization: Tokenization is a fundamental step in NLP preprocessing, where the input text is split into individual units called tokens. In this code, the tokenizer from the GPT-2 model is utilized to perform tokenization. The tokenizer processes the input text and converts it into a sequence of tokens. These tokens can represent individual words, subwords, or characters, depending on the underlying tokenization algorithm.

- Language Model Input Preparation: Once the input text is tokenized, it needs to be prepared as input for the language model. In this code, the input text is transformed into language modeling format, where the model predicts the next token based on the preceding tokens. The code uses the TextDataset class from the Transformers library to create a training dataset. The file_path argument specifies the path to the training dataset file, and the block_size argument determines the maximum length of each sequence. The dataset takes care of creating sequences from the tokenized input text, ensuring they are of the specified block size or shorter.

- Data Collation: Data collation is performed using the DataCollatorForLanguageModeling class. This data collator assists in preparing the input data for the language model during training. It performs tasks such as padding sequences to a uniform length, creating attention masks, and organizing the data into batches. In this code, mlm=False is set, indicating that masked language modeling is not used in this particular task.

By incorporating these preprocessing techniques, the code which is provided in Appendix 5, ensures that the input text is properly tokenized, transformed into language modeling format, and prepared for training the GPT-2 model. This preprocessing step is crucial for effective language modeling, as it enables the model to understand and generate coherent and contextually relevant text based on the given input.

5.7 Transfer learning and fine-tuning GPT-2

Transfer learning involves utilizing knowledge gained from one task or dataset to improve performance on another related task or dataset. GPT-2, being a pre-trained language model, can be effectively used for transfer learning. The idea is to take advantage of the pre-trained weights and linguistic knowledge encoded in GPT-2 and apply them to a new task or domain.
Here's a general approach to transfer learning with GPT-2[101], [103]–[105]:

1. Pre-training: GPT-2 is initially trained on a large corpus of publicly available text data. During pre-training, the model learns to predict the next word in a sentence, thereby capturing contextual relationships and semantic representations.

2. Fine-tuning: After pre-training, the next step is fine-tuning. Fine-tuning involves training the pre-trained GPT-2 model on a specific downstream task or domain-specific dataset. The model is adapted to the new task by exposing it to examples from the target dataset and updating the model parameters accordingly.

3. Task-specific architecture: Depending on the specific task, you may need to modify the architecture of GPT-2 by adding task-specific layers or modifying the output layer to suit the target task.

4. Training process: During fine-tuning, the model is trained on the target dataset using standard supervised learning techniques. The model is typically fine-tuned for fewer iterations compared to pre-training, as the initial knowledge from GPT-2 helps accelerate the learning process.

Benefits of Transfer Learning with GPT-2:

- Reduced data requirements: By leveraging the pre-trained knowledge, transfer learning with GPT-2 allows you to achieve good performance even with limited labeled data.
- Faster convergence: Pre-training on a large corpus helps GPT-2 learn useful representations, which speeds up the fine-tuning process and enables quicker convergence on the target task.
- Improved performance: GPT-2's ability to capture contextual relationships and semantic representations from pre-training often leads to improved performance on downstream tasks.

Limitations and Considerations:

- Domain mismatch: If the target task or domain is significantly different from the data on which GPT-2 was pre-trained, there might be a domain mismatch, and fine-tuning alone may not yield optimal results. In such cases, additional techniques like domain adaptation or more extensive modifications may be required.
- Data biases: GPT-2, like any language model, can inherit biases present in the pre-training data. Fine-tuning should be done with caution to ensure the biases are not amplified or perpetuated in the target task.
- Ethical considerations: It's important to be mindful of ethical implications when fine-tuning GPT-2, such as potential misuse of the model for generating harmful or misleading content.

6. CONCLUSION AND FUTURE WORK

6.1 Results and discussion

The implementation of the project resulted in the development of an intelligent virtual instructor (IVI) capable of providing guidance within the VR fatigue lab environment. The virtual instructor recognizes the lab's current state and generates relevant, natural-language instructions, significantly enhancing the educational effectiveness of the VR lab. The system creates a more engaging and productive learning environment. Throughout the study, a Decision Tree model achieved an accuracy of 93% in classifying the stage and location of users within the VR fatigue lab. The integration of the GPT-2 model showcased its capabilities in generating context-relevant instructional prompts. This integration enhanced the interactivity and user engagement of the VRILE system, enriching the educational experience.

6.2 Case study: An example of a student performing in the lab with the required physical space and equipment

In order to demonstrate the functionality of the intelligent virtual instructor (IVI) within the virtual environment for fatigue tests, a case study was conducted to illustrate how a novice student would experience the most probable scenarios and the corresponding IVI responses at each stage. Figure 6-1 shows the student wearing the headset and performing the test. The choice of a novice student allows to showcase how the IVI can effectively assist and guide someone who is new to the subject matter. Novice students often require more guidance and instructions compared to experienced individuals, making them an ideal target audience to highlight the IVI's capabilities in supporting learners at various stages of the test. In this case study the student is experiencing several stages in which he needs to be guided by the IVI.

Figure 6-1: A student performing the fatigue and guided by IVI

The following is a case study of how the IVI will work with a novice student. After the student has entered the VR space, they will be standing in the south-east corner of the lab near the lab entrance door at coordinates (1,-4). The top-down view of the lab can be seen in Figure 6-2 for better understanding the coordinates. The student is free to walk around and interact with the environment, but in this case, the student does not move and simply looks around the environment by only moving their head. Since the student did not read the pre-lab, they have no idea what to do, so they request help by pushing on the help button on the right controller. Note: the introduction to the equipment, along with the help button was introduced in the introductory environment which is required before entering the fatigue lab.

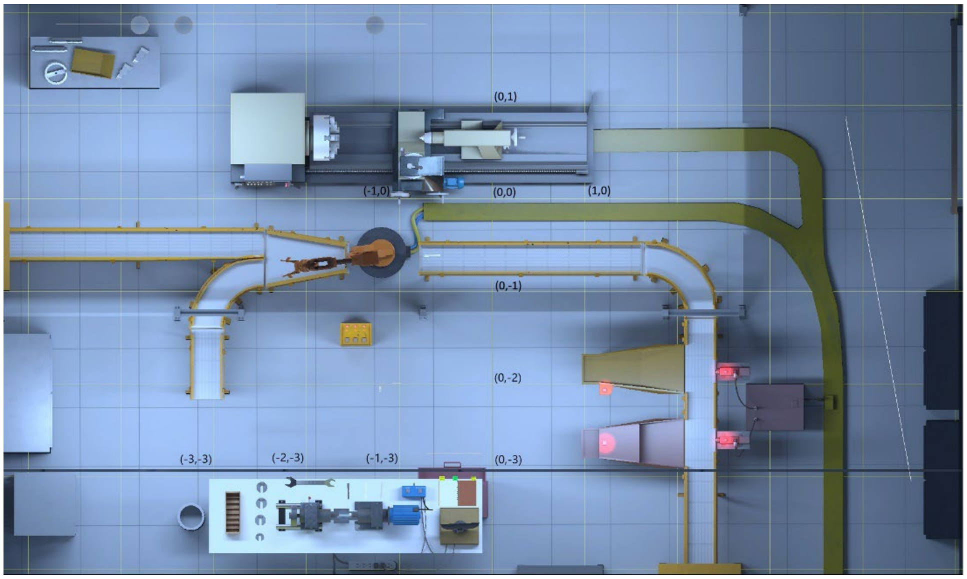
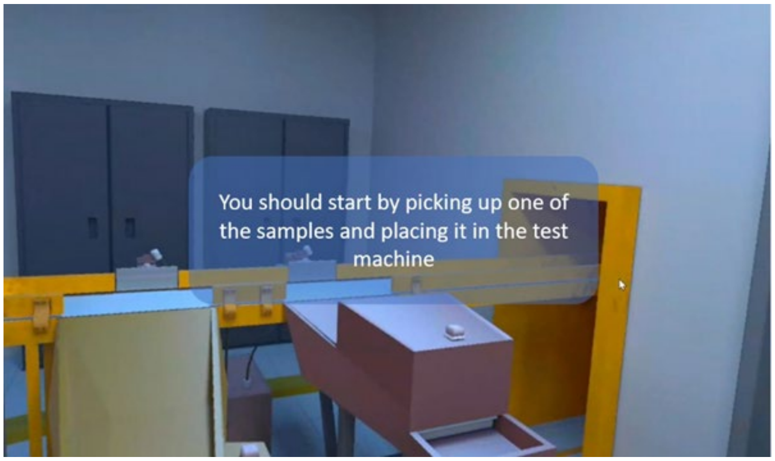


Figure 6-2: Top-down view of the lab

At this point the student is at coordinates (1, -3) and all of the states of the lab are 0. When this state is provided to the IVI, the prompt "You should start by picking up one of the samples and placing it in the test machine" is created, which is then converted to a script shown in the eye-view of the student as depicted in Figure 6-3.

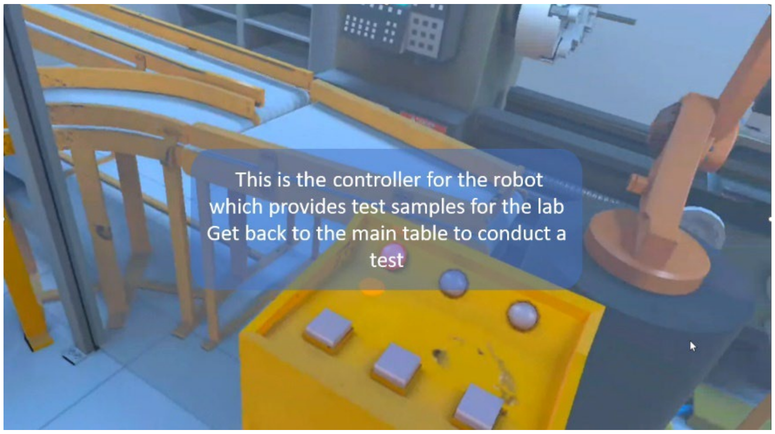


You should start by picking up one of the samples and placing it in the test machine

Figure 6-3: Case study, position of the student: Polishing table, state of the test: sample not in the machine

Upon following the initial prompt, the student proceeds to the center of the lab space, reaching coordinates (-1, 3), where they locate the fatigue machine as instructed. However, the student's curiosity is piqued by the robot in the vicinity, leading them to walk over to the conveyor controller box, positioned at coordinates (-1, 2). Believing that the IVI will now provide information about the controller box, the student presses the help button once again.

Despite the lab state remaining unchanged, the IVI accurately recognizes the new coordinates and responds with the appropriate prompt: "This is the controller for the robot, which provides test samples for the lab. Please return to the main table to conduct the test." The prompt is delivered to the student as depicted in Figure 6-4, offering guidance and redirection to ensure the student stays on track with the intended task of conducting the test at the main table.



This is the controller for the robot
which provides test samples for the lab
Get back to the main table to conduct a
test

Figure 6-4: Case study, position of the student: Turner robot controller, state of the test: sample not in the machine, counter bearing left

Upon realizing that the IVI is primarily focused on guiding the experiment, the student decides to interact with the conveyor controller, leading to the conveyor stopping as a result of the button presses. However, after a brief moment of experimentation, the student becomes disinterested and decides to return to the fatigue machine at coordinates (-1, 3). Once again, the student presses the help button, seeking further instructions from the IVI.

In response to the student's query, the IVI provides the appropriate output: "Commence the process by selecting one of the samples and placing it into the test machine." This guidance is displayed in Figure 6-5, directing the student to continue the experiment by selecting a sample and inserting it into the fatigue machine for testing. The IVI remains focused on facilitating the smooth execution of the experiment, maintaining its instructional role for the student.

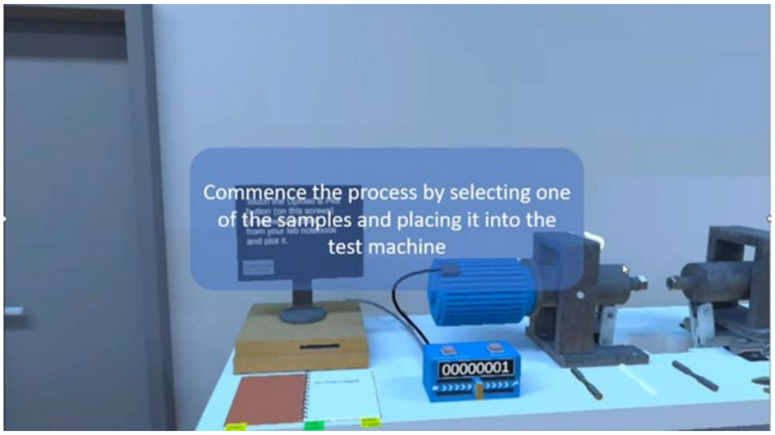

Commence the process by selecting one of the samples and placing it into the test machine

Figure 6-5: Case study, position of the student: Main table, state of the test: sample not in the machine

As the student attempts to insert the specimen into the counter bearing collet and encounters difficulties due to the collet not being slid back to accommodate the specimen, they press the help button once more to seek assistance. The IVI promptly provides the necessary prompt: "Use the red handle and push the counter bearing to the right to create more space for the specimen to fit into the machine." This instruction is displayed to the student as shown in Figure 6-6. The IVI continues to fulfill its role as an informative and supportive guide, helping the student overcome the specific challenge encountered during the experiment. By following the IVI's instructions, the student can proceed with placing the specimen appropriately into the fatigue machine.

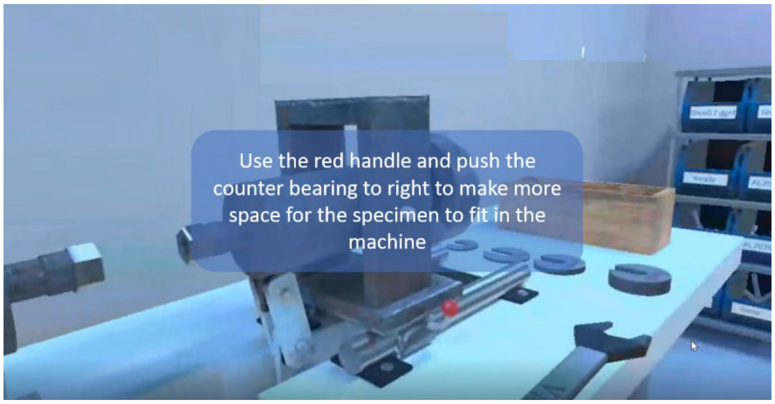
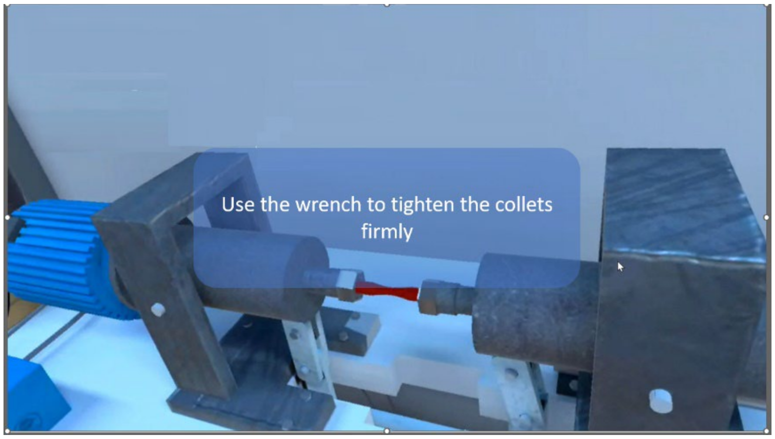

Use the red handle and push the counter bearing to right to make more space for the specimen to fit in the machine

Figure 6-6: Case study, position of the student: Main table, state of the test: sample in hand

The student proceeds to resolve the issue by grabbing the counter bearing slider knob and sliding it to the right, allowing space for the specimen to be inserted successfully. Once the specimen is in place, the student slides the counter bearing back into position and proceeds to initiate the fatigue test by pressing the start button on the fatigue control box. Despite the motor bearing spinning and the counter increasing, the specimen fails to rotate. After waiting for a few minutes and experiencing a sense of boredom, the student decides to seek further guidance by pressing the help button. As a result, the state of the lab changes, and the student finds themselves at coordinates (-2, -3).

In response to the student's request for assistance, the IVI generates the appropriate prompt: "Use the wrench to tighten the collets firmly." The prompt is displayed as shown in Figure 6-7, providing clear instructions to address the issue encountered during the fatigue test. By following the IVI's guidance and using the wrench to secure the collets, the student can potentially resolve the problem and proceed with the experiment successfully.



Use the wrench to tighten the collets firmly

Figure 6-7: Case study, position of the student: Main table, state of the test: sample not tightened

The student stops the machine, grabs the wrench, and tightens both collets. Once again, the student starts the machine and now the specimen is spinning correctly. However, since no weight was added to the fatigue machine, the specimen is not being fatigued. Once again, after a short time, the student hits the help button. Generated prompt = “Place additional weights by accessing the weight holder located beneath the table”, screenshot from the VR environment is provided in Figure 6-8.

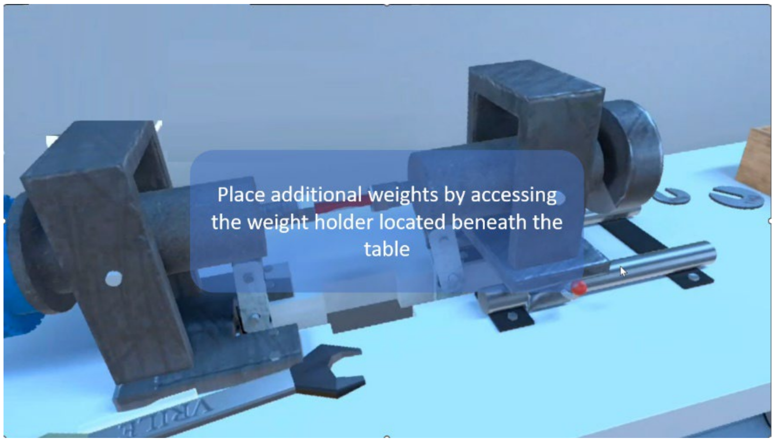

Place additional weights by accessing the weight holder located beneath the table

Figure 6-8: Case study, position of the student: Main table, state of the test: weights not added

The student once again stops the machine, finds the weight, adds the weight and restarts the machine. After about a minute, the specimen breaks. While the experiment has been completed, the data has not been logged. Not sure what to do next, the student once again asks for help. With the lab in this state, the generated prompt becomes “To prepare a report, utilize the notebook located in the left corner of the main table” shows in Figure 6-9.

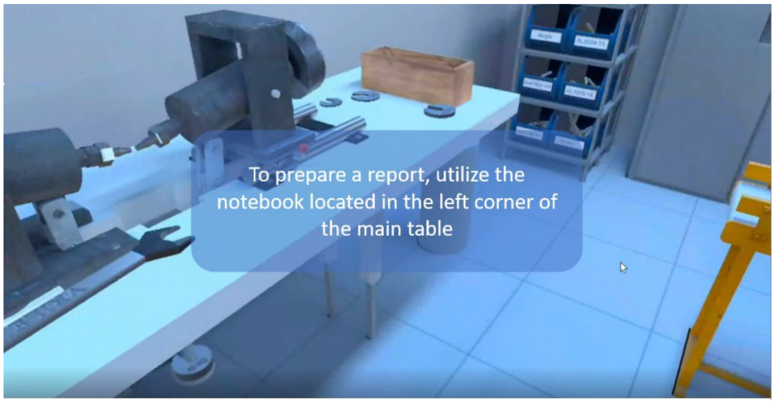


Figure 6-9: Case study, position of the student: Main table, state of the test: sample is broken

The student then activates the log-experiment button in the notebook and the results of the test will be saved for later reports. The student then loosens the collets, removes the broken specimen, and is ready to repeat the experiment with a different specimen.

It is important to note that these scenarios are just a few examples of the probable experiences a novice student might encounter in the virtual environment. The IVI's responses vary based on the student's location within the environment and the stage of the test. The dynamic and adaptive nature of the IVI enables it to provide context-specific instructions tailored to the student's progress and needs.

The case study highlights IVI's role in assisting and supporting students throughout the fatigue tests. By offering step-by-step instructions and reinforcing important actions, the IVI ensures that students perform the tasks correctly and gain a comprehensive understanding of the procedures.

6.3 Overview of the IVI

Figure 6-10 shows the flow chart outlines the key steps and components involved in the project, starting from the "student needs help" stage and culminating in the generation of suitable help prompts by the intelligent virtual instructor (IVI). Let's go through each step:

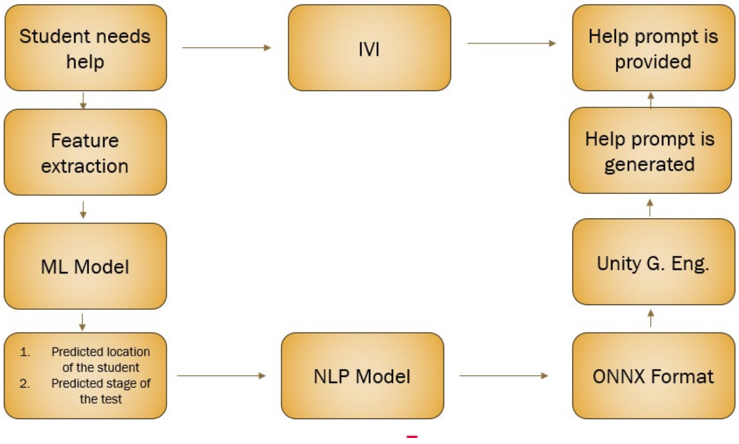


Figure 6-10: Flow chart; summarizing the process from student need for help to the suitable help prompt.

- Student Needs Help:

This initial stage represents a student's request for assistance within the virtual environment during the fatigue tests. It serves as the trigger point for the IVI to provide guidance and support.

- Feature Extraction:

After the student requests help, the system performs feature extraction. This step involves capturing and analyzing dynamic location information, such as X and Y coordinates, as well as binary data from the test stages. These features provide relevant contextual information for the subsequent stages.

- Machine Learning Model:

The extracted features serves as the input for a machine learning model. The model is trained to process the extracted features and predict the student's location within the virtual environment and the current stage of the fatigue test. These predictions serve as valuable outputs for subsequent steps.

- NLP Model:

The outputs from the machine learning model, which represent the student's location and test stage, serve as inputs for a natural language processing (NLP) model. The NLP model is trained to generate suitable guiding prompts based on the specific location and test stage. These prompts will assist the student in overcoming challenges or seeking further instructions.

- ONNX Format:

The NLP model, which is a deep learning model, is converted into the open neural network exchange (ONNX) format. This format allows for compatibility with the Unity game engine, which is the platform used for the virtual environment and IVI implementation.

- Unity Game Engine:

The ONNX format model is integrated into the Unity game engine. This engine serves as the environment for the IVI, allowing the model to operate within the virtual environment seamlessly. The Unity game engine enables the IVI to interact with the student and provide the necessary guidance based on their location and stage of the test.

- Help Prompt Is Generated:

With the IVI implemented in the virtual environment, the system is now capable of generating help prompts in real-time. When a student requests assistance, the IVI utilizes the location and stage predictions from the machine learning model to provide contextually appropriate guidance prompts. These prompts help the student navigate through the fatigue tests, overcome challenges, and proceed with the test procedures effectively.

- Help Prompt Is Delivered:

The ultimate objective of the study is successful generation and delivery of help prompts by the IVI. Through the feature extraction, machine learning models, NLP model, and the Unity game engine, the system accomplishes the goal of providing tailored assistance to students within the virtual environment.

The flow chart demonstrates the systematic process involved in the project, where features are extracted, models are employed for prediction and generation of suitable prompts, and the Unity game engine delivers the IVI within the virtual environment.

6.4 Future work

The implications of this project extend beyond its current scope and hold potential for future applications in educational settings, particularly for complex lab-based subjects. The AI-driven VRILE system developed in this study offers students an improved understanding and engagement with laboratory procedures, leading to enhanced learning outcomes. Further research can explore advanced language models to generate instructional text, enhancing accuracy and contextual relevance. Alternative AI models for classifying user stages and locations can be investigated to provide additional insights and improve classification accuracy. Conducting larger-scale user trials and collecting more extensive data would facilitate a comprehensive evaluation of the system's performance across diverse user groups. Future studies can also investigate the long-term impact of such technologies on student learning outcomes, providing valuable insights into sustained benefits of AI and VR integration in education.

This work focused on one single experimental lab (fatigue lab), but this should be extended to other laboratory environments, such as fluid dynamics laboratories, thermodynamics laboratories, and material science laboratories, to name a few. Developing a series of interconnected laboratory experiences that would match with a typical curriculum and allow students to progress from basic laboratory practices to more advanced experiments, including experimental design using virtual equipment. Finally, expanding the IVI to operate in a less rigid environment, such as the introductory environment shown in Figure 6-11and Figure 6-12, where students are taught how to use the VR equipment, before moving on to the experimental labs, would create a unified user experience in the VRILEs, from the fundamental beginning to the advanced, sophisticated experimental procedures.

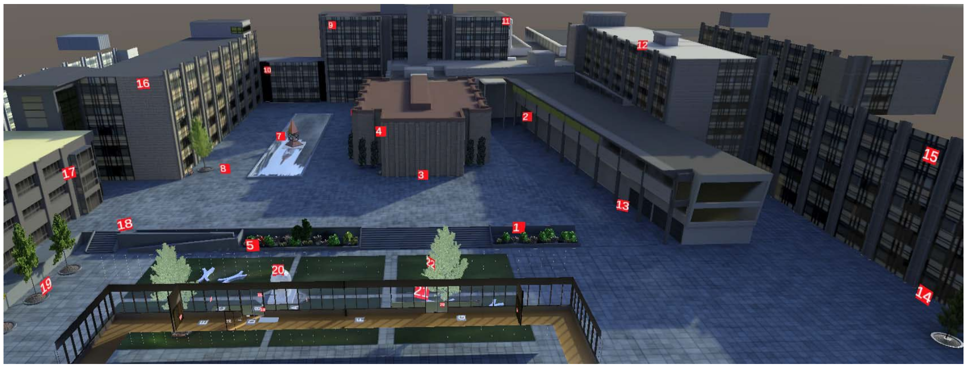


Figure 6-11: Proposed outside environment of the VRILE project in the future.

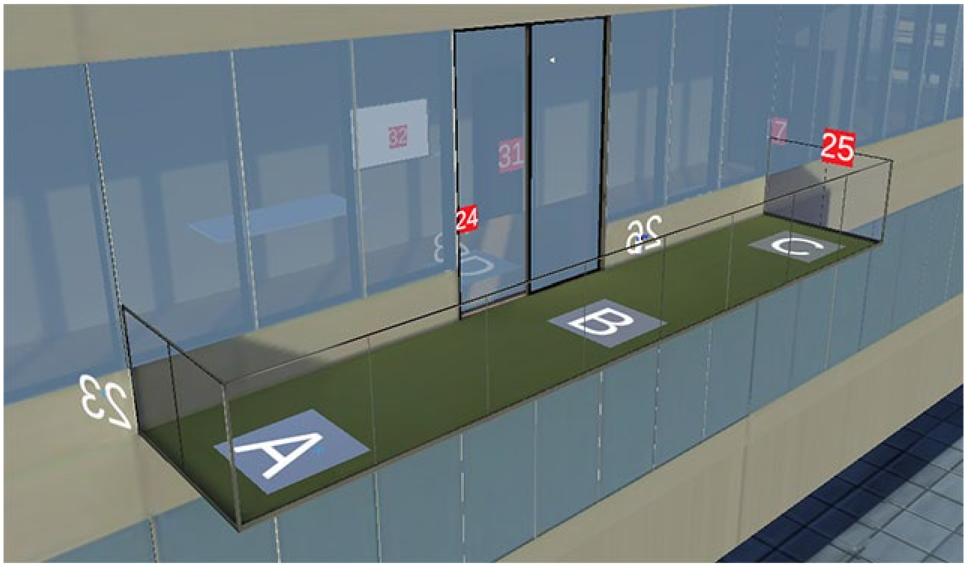


Figure 6-12: A balcony and automatic glass doors intended for the next generation VRILE

By providing a true-to-life representation of the campus and incorporating more mechanical labs, VRILE aspires to offer an immersive educational experience. It aims to become a comprehensive tool for mechanical engineering education, preparing students for a wide range of lab operations in a risk-free and engaging virtual environment.

This project demonstrates the potential of combining AI and VR technologies to enhance educational experiences. Continued advancements in these technologies have the potential to revolutionize the way students learn and engage with complex subjects in immersive virtual environments. Ongoing research and development in this area present promising possibilities for the future, with the potential to positively impact learning outcomes and contribute to student success.

REFERENCES

[1] A. Striegel, “Distance education and its impact on computer engineering laboratories,” in *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193)*, Oct. 2001, pp. F2D-4. doi: 10.1109/FIE.2001.963707.

[2] S. H. Johnson, W. L. Luyben, and D. L. Talhelm, “Undergraduate Interdisciplinary Controls Laboratory,” *Journal of Engineering Education*, vol. 84, no. 2, pp. 133–136, 1995, doi: 10.1002/j.2168-9830.1995.tb00160.x.

[3] D. J. Olinger and J. C. Hermanson, “Integrated Thermal-Fluid Experiments in WPI’s Discovery Classroom,” *Journal of Engineering Education*, vol. 91, no. 2, pp. 239–243, 2002, doi: 10.1002/j.2168-9830.2002.tb00697.x.

[4] “Feeling is Believing: Using a Force-Feedback Joystick to Teach Dynamic Systems - Okamura - 2002 - Journal of Engineering Education - Wiley Online Library.” https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2002.tb00713.x (accessed Jun. 21, 2023).

[5] R. Robinson, “Improving Design Of Experiment Skills Through A Project Based Fluids Laboratory,” presented at the 2002 Annual Conference, Jun. 2002, p. 7.642.1-7.642.13. Accessed: Jun. 20, 2023. [Online]. Available: https://peer.asee.org/improving-design-of-experiment-skills-through-a-project-based-fluids-laboratory

[6] J. E. Ashby, “The effectiveness of collaborative technologies in remote lab delivery systems,” in *2008 38th Annual Frontiers in Education Conference*, Oct. 2008, pp. F4E-7-F4E-12. doi: 10.1109/FIE.2008.4720394.

[7] A. M. Bisantz and V. L. Paquet, “Implementation and Evaluation of a Multi-course Case Study for Framing Laboratory Exercises,” *Journal of Engineering Education*, vol. 91, no. 3, pp. 299–307, 2002, doi: 10.1002/j.2168-9830.2002.tb00707.x.

[8] R. Robinson and J. Wellin, “Introducing Data Acquisition And Experimental Techniques To Mechanical Engineering Students In The Freshmen Year,” presented at the 2002 Annual Conference, Jun. 2002, p. 7.744.1-7.744.14. Accessed: Jun. 20, 2023. [Online]. Available: https://peer.asee.org/introducing-data-acquisition-and-experimental-techniques-to-mechanical-engineering-students-in-the-freshmen-year

[9] A. Mendoza and N. Buyurgan, “A Hands-on Experimental Laboratory for RFID Education,” *IIE Annual Conference. Proceedings*, pp. 1–5, 2006.

[10] A. D. Grant, “The Effective Use of Laboratories in Undergraduate Courses,” *International Journal of Mechanical Engineering Education*, vol. 23, no. 2, pp. 95–101, Apr. 1995, doi: 10.1177/030641909502300202.

[11] M. Iqbal Khan, S. M. Mourad, and W. M. Zahid, “Developing and qualifying Civil Engineering Programs for ABET accreditation,” *Journal of King Saud University - Engineering Sciences*, vol. 28, no. 1, pp. 1–11, Jan. 2016, doi: 10.1016/j.jksues.2014.09.001.

[12] L. D. Feisel and A. J. Rosa, “The Role of the Laboratory in Undergraduate Engineering Education,” *Journal of Engineering Education*, vol. 94, no. 1, pp. 121–130, 2005, doi: 10.1002/j.2168-9830.2005.tb00833.x.

[13] B. Bidanda and R. E. Billo, “On the Use of Students for Developing Engineering Laboratories,” *Journal of Engineering Education*, vol. 84, no. 2, pp. 205–213, 1995, doi: 10.1002/j.2168-9830.1995.tb00167.x.

[14] D. Magin and S. Kanapathipillai, “Engineering students’ understanding of the role of experimentation,” *European Journal of Engineering Education*, vol. 25, no. 4, pp. 351–358, Dec. 2000, doi: 10.1080/03043790050200395.

[15] M. Ogot, G. Elliott, and N. Glumac, “An Assessment of In-Person and Remotely Operated Laboratories,” *Journal of Engineering Education*, vol. 92, no. 1, pp. 57–64, 2003, doi: 10.1002/j.2168-9830.2003.tb00738.x.

[16] P. C.-H. Liu, A. Trivedi, and B. P. Lee, “Implementing Automatic Data Collection and Identification Laboratory,” in *Southcon/96 Conference Record*, Jun. 1996, pp. 222–226. doi: 10.1109/SOUTHC.1996.535068.

[17] H. A. El-Mounayri, E. Fernandez, and T. Wasfy, “Development of an Online Virtual Reality-Based Advanced Manufacturing Curriculum for Use in Professional Certification,” presented at the ASME 2008 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers Digital Collection, Aug. 2009, pp. 307–315. doi: 10.1115/IMECE2008-69147.

[18] C. B. Rogers, H. El-Mounayri, T. Wasfy, and J. Satterwhite, “Assessment of STEM e-Learning in an Immersive Virtual Reality (VR) Environment,” *Author*, 2018, Accessed: Jun. 21, 2023. [Online]. Available: https://scholarworks.iupui.edu/handle/1805/17939

[19] J. V. Nickerson, J. E. Corter, S. K. Esche, and C. Chassapis, “A model for evaluating the effectiveness of remote engineering laboratories and simulations in education,” *Computers & Education*, vol. 49, no. 3, pp. 708–725, Nov. 2007, doi: 10.1016/j.compedu.2005.11.019.

[20] B. Balamuralithara and P. C. Woods, “Virtual laboratories in engineering education: The simulation lab and remote lab,” *Computer Applications in Engineering Education*, vol. 17, no. 1, pp. 108–118, 2009, doi: 10.1002/cae.20186.

[21] D. Grimaldi and S. Rapuano, “Hardware and software to design virtual laboratory for education in instrumentation and measurement,” *Measurement*, vol. 42, no. 4, pp. 485–493, May 2009, doi: 10.1016/j.measurement.2008.09.003.

[22] V. Chang and J. A. Del Alamo, “Collaborative webLab: Enabling collaboration in an online laboratory,” in *World Congress on Networked Learning in a Global Environment*, 2002.

[23] N. Ertugrul, “New Era in Engineering Experiments: an Integrated and Interactive Teaching/Learning Approach, and Real-Time Visualisations,” *Int J Eng Educ*, vol. 14, Jan. 1998.

[24] C. Martin-Villalba, A. Urquia, and S. Dormido, “Development of virtual-labs for education in chemical process control using Modelica,” *Computers & Chemical Engineering*, vol. 39, pp. 170–178, Apr. 2012, doi: 10.1016/j.compchemeng.2011.10.010.

[25] A. C. Rafael, F. Bernardo, L. M. Ferreira, M. G. Rasteiro, and J. C. Teixeira, “Virtual Applications Using a Web Platform to Teach Chemical Engineering: The Distillation Case,” *Education for Chemical Engineers*, vol. 2, no. 1, pp. 20–28, Jan. 2007, doi: 10.1205/ece06007.

[26] C.-H. Su and T.-W. Cheng, “A Sustainability Innovation Experiential Learning Model for Virtual Reality Chemistry Laboratory: An Empirical Study with PLS-SEM and IPMA,” *Sustainability*, vol. 11, no. 4, Art. no. 4, Jan. 2019, doi: 10.3390/su11041027.

[27] K. D. Forbus *et al.*, “CyclePad: An articulate virtual laboratory for engineering thermodynamics,” *Artificial Intelligence*, vol. 114, no. 1, pp. 297–347, Oct. 1999, doi: 10.1016/S0004-3702(99)00080-6.

[28] S.-A. Ángel, “Real and virtual bioreactor laboratory sessions by STSE–CLIL WebQuest,” *Education for Chemical Engineers*, vol. 13, pp. 1–8, Oct. 2015, doi: 10.1016/j.ece.2015.06.004.

[29] J. O. Campbell, J. R. Bourne, P. J. Mosterman, and A. J. Brodersen, “The Effectiveness of Learning Simulations for Electronic Laboratories,” *Journal of Engineering Education*, vol. 91, no. 1, pp. 81–87, 2002, doi: 10.1002/j.2168-9830.2002.tb00675.x.

[30] M. E. Macias, V. M. Cazares, and E. E. Ramos, “A virtual laboratory for introductory electrical engineering courses to increase the student performance,” in *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193)*, Oct. 2001, pp. S2C-S13 vol.3. doi: 10.1109/FIE.2001.964022.

[31] D. Gillet, A. V. N. Ngoc, and Y. Rekik, “Collaborative web-based experimentation in flexible engineering education,” *IEEE Transactions on Education*, vol. 48, no. 4, pp. 696–704, Nov. 2005, doi: 10.1109/TE.2005.852592.

[32] C. Schmid, “The virtual control lab VCLab for education on the Web,” in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, Jun. 1998, pp. 1314–1318 vol.2. doi: 10.1109/ACC.1998.703627.

[33] B. Aktan, C. A. Bohus, L. A. Crowl, and M. H. Shor, "Distance learning applied to control engineering laboratories," *IEEE Transactions on Education*, vol. 39, no. 3, pp. 320–326, Aug. 1996, doi: 10.1109/13.538754.

[34] F. A. Candelas, F. Torres, F. G. Ortiz, P. Gil, J. Pomares, and S. T. Puente, "Teaching and learning robotics with internet teleoperation," in *Proc. Second International Conference on Multimedia and Information & Communication Technologies in Education*, 2003, pp. 1827–1831.

[35] S. T. Puente, F. Torres, F. Ortiz, and F. A. Candelas, "Remote robot execution through WWW simulation," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, Sep. 2000, pp. 503–506 vol.4. doi: 10.1109/ICPR.2000.902967.

[36] R. Safaric, M. Truntic, D. Hercog, and P. Gregor, "Control and robotics remote laboratory for engineering education," *International Journal of Online Engineering*, vol. 1, Jun. 2005.

[37] E. Guimaraes *et al.*, "REAL: a virtual laboratory for mobile robot experiments," *IEEE Transactions on Education*, vol. 46, no. 1, pp. 37–42, Feb. 2003, doi: 10.1109/TE.2002.804404.

[38] F. Torres, F. A. Candelas, S. T. Puente, J. Pomares, P. Gil, and F. G. Ortiz, "Experiences with virtual environment and remote laboratory for teaching and learning robotics at the University of Alicante," *International Journal of Engineering Education*, vol. 22, no. 4, p. 766, 2006.

[39] C. S. Tzafestas, N. Palaiologou, and M. Alifragis, "Virtual and remote robotic laboratory: comparative experimental evaluation," *IEEE Transactions on Education*, vol. 49, no. 3, pp. 360–369, Aug. 2006, doi: 10.1109/TE.2006.879255.

[40] A. Pipinato, M. Molinari, C. Pellegrino, O. S. Bursi, and C. Modena, "Fatigue tests on riveted steel elements taken from a railway bridge," *Structure and Infrastructure Engineering*, vol. 7, no. 12, pp. 907–920, Dec. 2011, doi: 10.1080/15732470903099776.

[41] P. Beaumont, F. Guérin, P. Lantieri, M. L. Facchinetti, and G. M. Borret, "Accelerated fatigue test for automotive chassis parts design: An overview," in *2012 Proceedings Annual Reliability and Maintainability Symposium*, Jan. 2012, pp. 1–6. doi: 10.1109/RAMS.2012.6175513.

[42] X. Le, R. L. Roberts, A. W. D. P.e, and H. Connors, "An Integrated Active Learning Approach for Understanding Fatigue Theory," presented at the 2018 ASEE Annual Conference & Exposition, Jun. 2018. Accessed: Jun. 14, 2023. [Online]. Available: https://peer.asee.org/an-integrated-active-learning-approach-for-understanding-fatigue-theory

[43] J. W. Gentry, "What is experiential learning," *Guide to business gaming and experiential learning*, vol. 9, p. 20, 1990.

[44] M. R. Young, “Experiential Learning=Hands-On+Minds-On,” *Marketing Education Review*, vol. 12, no. 1, pp. 43–51, Mar. 2002, doi: 10.1080/10528008.2002.11488770.

[45] L. Freina and M. Ott, *A Literature Review on Immersive Virtual Reality in Education: State Of The Art and Perspectives*. 2015. doi: 10.12753/2066-026X-15-020.

[46] E. McGovern, G. Moreira, and C. Luna-Nevarez, “An application of virtual reality in education: Can this technology enhance the quality of students’ learning experience?,” *Journal of Education for Business*, vol. 95, no. 7, pp. 490–496, Oct. 2020, doi: 10.1080/08832323.2019.1703096.

[47] F. Tahiru, “AI in Education: A Systematic Literature Review,” *JCIT*, vol. 23, no. 1, pp. 1–20, Jan. 2021, doi: 10.4018/JCIT.2021010101.

[48] B. Pendy, “Artificial Intelligence: The Future of Education,” *Jurnal Indonesia Sosial Sains*, vol. 2, no. 11, 2021, doi: 10.59141/jiss.v2i11.801.

[49] M. Tanjga, “E-learning and the Use of AI: A Review of Current Practices and Future Directions,” *Qeios*, May 2023, doi: 10.32388/AP0208.

[50] T. Wang *et al.*, “Exploring the Potential Impact of Artificial Intelligence (AI) on International Students in Higher Education: Generative AI, Chatbots, Analytics, and International Student Success,” May 2023, doi: 10.20944/preprints202305.0808.v1.

[51] H. Crompton and D. Song, “The Potential of Artificial Intelligence in Higher Education,” *Revista Virtual Universidad Católica del Norte*, no. 62, Art. no. 62, 2021, doi: 10.35575/rvucn.n62a1.

[52] Q. Chang, X. Pan, N. Manikandan, and S. Ramesh, “Artificial Intelligence Technologies for Teaching and Learning in Higher Education,” *Int. J. Rel. Qual. Saf. Eng.*, vol. 29, no. 05, p. 2240006, Oct. 2022, doi: 10.1142/S021853932240006X.

[53] D. S. Sumo and M. L. Bah, “Chinese Language Education in the Era of Artificial Intelligence; Innovation Development, Pedagogy & the Smart Classroom.” OSF Preprints, Nov. 16, 2021. doi: 10.31219/osf.io/axr27.

[54] G. Shekhar, R. D’Souza, and K. Fernandes, “AI-Driven Contextual Virtual Teaching Assistant Using RASA,” in *Proceedings of the 21st Annual Conference on Information Technology Education*, in SIGITE ’20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, p. 346. doi: 10.1145/3368308.3415442.

[55] proofeditor, “Comprehensive review of supervised machine learning algorithms to identify the best and error free,” *International Journal of Scholarly Research in Engineering and Technology*, Jan. 16, 2023. https://srrjournals.com/ijsret/content/comprehensive-review-supervised-machine-learning-algorithms-identify-best-and-error-free (accessed Jun. 14, 2023).

[56] J. Brownlee, “Curve Fitting With Python,” *MachineLearningMastery.com*, Nov. 03, 2020. https://machinelearningmastery.com/curve-fitting-with-python/ (accessed Mar. 27, 2023).

[57] S. Taheri and M. Mammadov, “Structure learning of Bayesian Networks using global optimization with applications in data classification,” *Optim Lett*, vol. 9, no. 5, pp. 931–948, Jun. 2015, doi: 10.1007/s11590-014-0803-1.

[58] J. Cheng and R. Greiner, “Comparing Bayesian Network Classifiers.” arXiv, Jan. 23, 2013. doi: 10.48550/arXiv.1301.6684.

[59] D. Grossman and P. Domingos, “Learning Bayesian network classifiers by maximizing conditional likelihood,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 46.

[60] B. G. Marcot and T. D. Penman, “Advances in Bayesian network modelling: Integration of modelling technologies,” *Environmental modelling & software*, vol. 111, pp. 386–393, 2019.

[61] B. Mihaljević, C. Bielza, and P. Larrañaga, “Bayesian networks for interpretable machine learning and optimization,” *Neurocomputing*, vol. 456, pp. 648–665, 2021.

[62] S. Wang, R. Gao, and L. Wang, “Bayesian network classifiers based on Gaussian kernel density,” *Expert Systems with Applications*, vol. 51, Jan. 2016, doi: 10.1016/j.eswa.2015.12.031.

[63] A. Saini, “Decision Tree Algorithm - A Complete Guide,” *Analytics Vidhya*, Aug. 29, 2021. https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/ (accessed Jun. 21, 2023).

[64] B. Charbuty and A. Abdulazeez, “Classification based on decision tree algorithm for machine learning,” *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021.

[65] Y. Freund and L. Mason, “The alternating decision tree learning algorithm,” in *icml*, 1999, pp. 124–133.

[66] C. J. Mantas and J. Abellán, “Credal-C4.5: Decision tree based on imprecise probabilities to classify noisy data,” *Expert Systems with Applications*, vol. 41, no. 10, pp. 4625–4637, Aug. 2014, doi: 10.1016/j.eswa.2014.01.017.

[67] A. Priyam, G. R. Abhijeeta, A. Rathee, and S. Srivastava, “Comparative analysis of decision tree classification algorithms,” *International Journal of current engineering and technology*, vol. 3, no. 2, pp. 334–337, 2013.

[68] Y. Mishina, R. Murata, Y. Yamauchi, T. Yamashita, and H. Fujiyoshi, “Boosted Random Forest,” *IEICE Transactions on Information and Systems*, vol. E98.D, no. 9, pp. 1630–1636, 2015, doi: 10.1587/transinf.2014OPP0004.

[69] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS journal of photogrammetry and remote sensing*, vol. 114, pp. 24–31, 2016.

[70] G. Biau, "Analysis of a random forests model," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 1063–1095, 2012.

[71] S. J. Rigatti, "Random forest," *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.

[72] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?," in *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8*, Springer, 2012, pp. 154–168.

[73] A. A. Soofi and A. Awan, "Classification techniques in machine learning: applications and issues," *J. Basic Appl. Sci*, vol. 13, pp. 459–465, 2017.

[74] S. Abe, "Fuzzy support vector machines for multilabel classification," *Pattern Recognition*, vol. 48, no. 6, pp. 2110–2117, Jun. 2015, doi: 10.1016/j.patcog.2015.01.009.

[75] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[76] D. Meyer and F. T. Wien, "Support vector machines," *The Interface to libsvm in package e1071*, vol. 28, no. 20, p. 597, 2015.

[77] T. Iqbal and S. Qureshi, "The survey: Text generation models in deep learning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, Part A, pp. 2515–2528, Jun. 2022, doi: 10.1016/j.jksuci.2020.04.001.

[78] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306.

[79] R. Yasrab and M. Pound, *PhenomNet: Bridging Phenotype-Genotype Gap: A CNN-LSTM Based Automatic Plant Root Anatomization System*. 2020. doi: 10.1101/2020.05.03.075184.

[80] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating Sentences from a Continuous Space." arXiv, May 12, 2016. doi: 10.48550/arXiv.1511.06349.

[81] J. Brownlee, "A Gentle Introduction to Generative Adversarial Networks (GANs)," *MachineLearningMastery.com*, Jun. 16, 2019. https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (accessed Jun. 14, 2023).

[82] G. Tevet, G. Habib, V. Shwartz, and J. Berant, “Evaluating Text GANs as Language Models.” arXiv, Mar. 24, 2019. doi: 10.48550/arXiv.1810.12686.

[83] C. I. Guinn and R. J. Montoya, “Natural language processing in Virtual Reality training environments,” *Research Triangle Institute Research Triangle Park*, 2009.

[84] H. Zhang *et al.*, “Text Feature Adversarial Learning for Text Generation With Knowledge Transfer From GPT2,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2022, doi: 10.1109/TNNLS.2022.3210975.

[85] G. Liguori, “Post | LinkedIn.” https://www.linkedin.com/posts/ingliguori_gpt1-gpt2-gpt3-activity-7028774382193774592-xdoj/?originalSubdomain=py (accessed Jun. 14, 2023).

[86] V. T. Nguyen and T. Dang, “Setting up Virtual Reality and Augmented Reality Learning Environment in Unity,” in *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, Oct. 2017, pp. 315–320. doi: 10.1109/ISMAR-Adjunct.2017.97.

[87] “Room - VR Interior Environment [Unity], Indhranath Sri Ram,” *ArtStation*, May 09, 2022. https://www.artstation.com/artwork/r9Ba2J (accessed Jun. 14, 2023).

[88] “🤗 Transformers.” https://huggingface.co/docs/transformers/index (accessed Jun. 14, 2023).

[89] Y. Singh, P. K. Bhatia, and O. Sangwan, “A review of studies on machine learning techniques,” *International Journal of Computer Science and Security*, vol. 1, no. 1, pp. 70–84, 2007.

[90] M. Heydarian, T. E. Doyle, and R. Samavi, “MLCM: Multi-label confusion matrix,” *IEEE Access*, vol. 10, pp. 19083–19095, 2022.

[91] M. Yin, J. Wortman Vaughan, and H. Wallach, “Understanding the effect of accuracy on trust in machine learning models,” in *Proceedings of the 2019 chi conference on human factors in computing systems*, 2019, pp. 1–12.

[92] S. Haghighi, M. Jasemi, S. Hessabi, and A. Zolanvari, “PyCM: Multiclass confusion matrix library in Python,” *Journal of Open Source Software*, vol. 3, no. 25, p. 729, 2018.

[93] J. Liang, “Confusion matrix: Machine learning,” *POGIL Activity Clearinghouse*, vol. 3, no. 4, 2022.

[94] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, “Transfer Learning in Natural Language Processing,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 15–18. doi: 10.18653/v1/N19-5004.

[95] “gpt2 · Hugging Face.” https://huggingface.co/gpt2 (accessed Jun. 21, 2023).

APPENDIX A. CLASSIFICATION ALGORITHMS

Comparing Decision tree, Random Forest, SVM, and Bayesian over the stage identification dataset.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

# Read the CSV file
df = pd.read_csv('/content/FINAL_stage_combinations_5-Aug2_06-21-23.csv')


X = df.iloc[:,:-1]
y = df.iloc[:,-1]

label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
                                                    test_size=0.3,
                                                    random_state=10)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

# Create a Random Forest classifier
```

```python
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

rf_predictions = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

print("Random Forest Accuracy:", rf_accuracy)

# Create a Decision Tree classifier
dt_classifier = DecisionTreeClassifier(random_state=10)
dt_classifier.fit(X_train, y_train)

dt_predictions = dt_classifier.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_predictions)

print("Decision Tree Accuracy:", dt_accuracy)

# Confusion matrix for Random Forest
rf_cm = confusion_matrix(y_test, rf_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(rf_cm, annot=True, cmap="Blues", fmt="d",
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Random Forest")
plt.show()

# Confusion matrix for Decision Tree
dt_cm = confusion_matrix(y_test, dt_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(dt_cm, annot=True, cmap="Blues", fmt="d",
            xticklabels=label_encoder.classes_,
            yticklabels=label_encoder.classes_)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Decision Tree")
plt.show()

# Classification report for Random Forest
rf_report = classification_report(y_test,
                                  rf_predictions,
                                  labels=label_encoder.transform(label_encoder.classes_),
                                      target_names=label_encoder.classes_)
    print("Random Forest Classification Report:")
    print(rf_report)

    # Classification report for Decision Tree
    dt_report = classification_report(y_test,
                                      dt_predictions,
                                      labels=label_encoder.transform(label_encoder.classes_),
                                      target_names=label_encoder.classes_)
    print("Decision Tree Classification Report:")
    print(dt_report)

    # Create SVM classifier
    svm_classifier = SVC(random_state=42)
    svm_classifier.fit(X_train, y_train)
    svm_predictions = svm_classifier.predict(X_test)

    # Create Naive Bayes classifier
    nb_classifier = GaussianNB()
    nb_classifier.fit(X_train, y_train)
    nb_predictions = nb_classifier.predict(X_test)

    # Evaluate SVM classifier
    svm_accuracy = accuracy_score(y_test, svm_predictions)
    svm_cm = confusion_matrix(y_test, svm_predictions)
    svm_report = classification_report(y_test,
                                       svm_predictions,
                                       labels=label_encoder.transform(label_encoder.classes_),
                                       target_names=label_encoder.classes_)

    print("SVM Accuracy:", svm_accuracy)
    print("SVM Confusion Matrix:")
    print(svm_cm)
    print("SVM Classification Report:")
    print(svm_report)

    # Evaluate Naive Bayes classifier
    nb_accuracy = accuracy_score(y_test, nb_predictions)
    nb_cm = confusion_matrix(y_test, nb_predictions)
    nb_report = classification_report(y_test, nb_predictions,
                                      labels=label_encoder.transform(label_encoder.classes_),
                                      target_names=label_encoder.classes_)

    print("Naive Bayes Accuracy:", nb_accuracy)
    print("Naive Bayes Confusion Matrix:")
    print(nb_cm)
    print("Naive Bayes Classification Report:")
    print(nb_report)

    # Confusion matrix for SVM
    svm_cm = confusion_matrix(y_test, svm_predictions)
    plt.figure(figsize=(8, 6))
    sns.heatmap(svm_cm, annot=True, cmap="Blues",
                fmt="d", xticklabels=label_encoder.classes_,
                yticklabels=label_encoder.classes_)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title("Confusion Matrix - SVM")
    plt.show()

    # Confusion matrix for Naive Bayes
    nb_cm = confusion_matrix(y_test, nb_predictions)
    plt.figure(figsize=(8, 6))
    sns.heatmap(nb_cm, annot=True, cmap="Blues", fmt="d",
                xticklabels=label_encoder.classes_,
                yticklabels=label_encoder.classes_)

    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title("Confusion Matrix - Naive Bayes")
    plt.show()

```

The provided code performs the following tasks:

- Reading the CSV File: The code imports the pandas library and reads a CSV file called 'FINAL_stage_combinations_5-Aug2_06-21-23.csv' using the read_csv function. The data is stored in a DataFrame called df.

- Data Preprocessing:

1. Separating Features and Target: The code separates the features (X) and the target variable (y) from the DataFrame df.
2. Encoding the Target Variable: The code uses LabelEncoder from scikit-learn to encode the categorical target variable y into numerical labels.
3. Splitting the Dataset: The code splits the dataset into training and testing sets using the train_test_split function. It assigns 70% of the data for training (X_train, y_train) and 30% for testing (X_test, y_test).

- Random Forest Classification:

1. Creating the Classifier: The code creates a Random Forest classifier using RandomForestClassifier from scikit-learn.
2. Training the Classifier: The code trains the Random Forest classifier on the training data using the fit method.
3. Making Predictions: The code makes predictions on the testing data using the trained Random Forest classifier.
4. Evaluating the Classifier: The code calculates the accuracy of the Random Forest classifier using the accuracy_score function.

- Decision Tree Classification:

1. Creating the Classifier: The code creates a Decision Tree classifier using DecisionTreeClassifier from scikit-learn.
2. Training the Classifier: The code trains the Decision Tree classifier on the training data using the fit method.
3. Making Predictions: The code makes predictions on the testing data using the trained Decision Tree classifier.
4. Evaluating the Classifier: The code calculates the accuracy of the Decision Tree classifier using the accuracy_score function.

- Confusion Matrix and Classification Report:

1. Confusion Matrix: The code creates confusion matrices for both the Random Forest and Decision Tree classifiers using confusion_matrix from scikit-learn. It visualizes the confusion matrices using heatmaps with the help of matplotlib and seaborn.
2. Classification Report: The code generates classification reports for both classifiers using classification_report from scikit-learn. The classification reports provide precision, recall, F1-score, and support for each class.

- SVM and Naive Bayes Classification:

  1. Creating the Classifiers: The code creates a Support Vector Machine (SVM) classifier using SVC and a Naive Bayes classifier using GaussianNB from scikit-learn.
  2. Training the Classifiers: The code trains both classifiers on the training data using the fit method.
  3. Making Predictions: The code makes predictions on the testing data using the trained classifiers.
  4. Evaluating the Classifiers: The code calculates the accuracy, confusion matrix, and classification report for both classifiers.

- Confusion Matrix and Classification Report (SVM and Naive Bayes):

  1. The code generates confusion matrices and visualizes them using heatmaps for both SVM and Naive Bayes classifiers.
  2. The code also generates classification reports for both classifiers, providing precision, recall, F1-score, and support for each class.

Overall, this code performs classification tasks using Random Forest, Decision Tree, SVM, and Naive Bayes classifiers. It evaluates the classifiers' performance through accuracy, confusion matrix, and classification report, providing insights into the effectiveness of each model for the given dataset. The result of the code are discussed in section 5.2.1.

Comparing Decision tree, Random forest, SVM, and Bayesian over the location identification dataset.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix

```

```
# Read the CSV file
df = pd.read_csv("/content/FINAL_coordinates_dataset_06-21-23.csv")

# Separate the features (X) and the target variable (y)
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Decision Tree
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)
dt_predictions = dt_classifier.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_predictions)

# Random Forest
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)
rf_predictions = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

# SVM
svm_classifier = SVC()
svm_classifier.fit(X_train, y_train)
svm_predictions = svm_classifier.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)

# Bayesian Network (Naive Bayes)
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
nb_predictions = nb_classifier.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_predictions)

# Confusion Matrix - Decision Tree
dt_cm = confusion_matrix(y_test, dt_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(dt_cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Decision Tree")
plt.show()

```

```
# Confusion Matrix - Random Forest
rf_cm = confusion_matrix(y_test, rf_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(rf_cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Random Forest")
plt.show()

# Confusion Matrix - SVM
svm_cm = confusion_matrix(y_test, svm_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(svm_cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - SVM")
plt.show()

# Confusion Matrix - Naive Bayes
nb_cm = confusion_matrix(y_test, nb_predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(nb_cm, annot=True, cmap="Blues", fmt="d")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Naive Bayes")
plt.show()

# Print accuracy scores
print("Decision Tree Accuracy:", dt_accuracy)
print("Random Forest Accuracy:", rf_accuracy)
print("SVM Accuracy:", svm_accuracy)
print("Naive Bayes Accuracy:", nb_accuracy)

```

APPENDIX B. FAKE DATA AUGMENTATION

```
#Fake data for different stages and areas:

import random
import pandas as pd

# Define the areas and their coordinates
areas = {
    'Entrance': [(0, -4), (0, -3), (2, -3), (2, -4)],
    'Main Table': [(-3, -3), (-3, -2), (0, -2), (0, -3)],
    'Cupboard': [(-5, -4), (-5, -1), (-3, -1), (-3, -4)],
    'Polishing Table': [(0, -3), (0, -1), (1, -1), (1, -3)],
    'Turner Robot': [(-2, -2), (-2, -1), (-1, -1), (-1, -2)]
}

# Define the mappings for the rest of the features
feature_mappings = {
    (0, 0, 0, 0, 0): 'Sample_not_in',
    (0, 0, 0, 0, 1): 'Sample_broken',
    (0, 0, 0, 1, 0): 'Sample_not_in',
    (0, 0, 0, 1, 1): 'Sample_broken',
    (0, 0, 1, 0, 0): 'Sample_not_in',
    (0, 0, 1, 0, 1): 'Sample_broken',
    (0, 0, 1, 1, 0): 'Sample_not_in',
    (0, 0, 1, 1, 1): 'Sample_broken',
    (0, 1, 0, 0, 0): 'Sample_not_in',
    (0, 1, 0, 0, 1): 'Sample_broken',
    (0, 1, 0, 1, 0): 'Sample_not_in',
    (0, 1, 0, 1, 1): 'Sample_broken',
    (0, 1, 1, 0, 0): 'Sample_not_in',
    (0, 1, 1, 0, 1): 'Sample_broken',
    (0, 1, 1, 1, 0): 'Sample_not_in',
    (0, 1, 1, 1, 1): 'Sample_broken',
    (1, 0, 0, 0, 0): 'Sample_in',
    (1, 0, 0, 0, 1): 'Sample_broken',
    (1, 0, 0, 1, 0): 'Sample_not_tight',
    (1, 0, 0, 1, 1): 'Sample_broken',
    (1, 0, 1, 0, 0): 'Sample_not_tight',
    (1, 0, 1, 0, 1): 'Sample_broken',
    (1, 0, 1, 1, 0): 'Sample_not_tight',
    (1, 0, 1, 1, 1): 'Sample_broken',
    (1, 1, 0, 0, 0): 'Weight_not_in',
    (1, 1, 0, 0, 1): 'Sample_broken',
    (1, 1, 0, 1, 0): 'Weight_not_in',
    (1, 1, 0, 1, 1): 'Sample_broken',
    (1, 1, 1, 0, 0): 'Machine_not_on',
    (1, 1, 1, 0, 1): 'Sample_broken',
    (1, 1, 1, 1, 0): 'Sample_not_broken',
    (1, 1, 1, 1, 1): 'Sample_broken'
}

# Generate fake data for the dataset
num_samples = 1000  # Number of samples in the dataset
dataset = []

for _ in range(num_samples):
    x = random.uniform(-6, 6)  # Generate a random X coordinate
    y = random.uniform(-5, 1)  # Generate a random Y coordinate

    # Determine the area based on X and Y coordinates
    area = 'Wandering'
    for area_name, coords in areas.items():
        if coords[0][0] <= x <= coords[2][0] and coords[0][1] <= y <=
coords[2][1]:
            area = area_name
            break

    # Generate random values for the features
    sim, st, wa, ms, sib = random.choices([0, 1], k=5)

    # Determine the stage based on the feature values
    stage = feature_mappings[(sim, st, wa, ms, sib)]

    # Append the generated data to the dataset
    dataset.append([x, y, sim, st, wa, ms, sib, stage, area])

# Create a DataFrame from the generated dataset
columns = ['X', 'Y', 'SIM', 'ST', 'WA', 'MS', 'SIB', 'Stage', 'Location']
df = pd.DataFrame(dataset, columns=columns)

# Add a new column that combines "Location" and "Stage" with "and" in a
text format
df['Location and Stage'] = df['Location'] + ' and ' + df['Stage']

# Remove the "Stage" and "Location" columns
df.drop(['Stage', 'Location'], axis=1, inplace=True)
```

```
# Save the DataFrame to a CSV file
df.to_csv('fake_dataset.csv', index=False)
```

The code begins by importing the necessary libraries, including random for generating random values and pandas for data manipulation. Next, the code defines the areas and their coordinates using a dictionary called areas. Each area is associated with a list of coordinates that define its boundaries. Following that, the code defines a mapping for the rest of the features using a dictionary called feature_mappings. This mapping associates specific combinations of feature values with corresponding stage labels.

The code then proceeds to generate the fake data for the dataset. It specifies the number of samples (num_samples) to be generated. Inside the loop, random X and Y coordinates are generated within specified ranges. The code determines the area corresponding to the generated coordinates by iterating through the areas dictionary and comparing the coordinates to the defined boundaries. Random values are generated for the remaining features: SIM, ST, WA, MS, and SIB. These values are used to determine the stage based on the feature_mappings dictionary.

The generated data is appended to the dataset list as a list of values representing each sample. After the loop, a DataFrame is created from the dataset using the defined column names. A new column called "Location and Stage" is added to the DataFrame by combining the "Location" and "Stage" columns using the "+" operator and the word "and" in a text format. Finally, the "Stage" and "Location" columns are dropped from the DataFrame, and the resulting DataFrame is saved to a CSV file named "fake_dataset.csv".

This code allows for the generation of a synthetic dataset with different stages, areas, and associated features, which can be useful for testing and experimentation purposes.

APPENDIX C. MODEL TRAINING

```python
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer

# Load the dataset
df =
pd.read_csv("/content/Final_fake_dataset_for_2_in_1_classification_06-21-
23.csv")  # Replace "your_dataset.csv" with the actual file name

# Split the dataset into features (X) and target (y)
X = df.drop("Location and Stage", axis=1)
y = df["Location and Stage"]

# Preprocessing
# Handle missing values
imputer = SimpleImputer(strategy="most_frequent")
X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

# Encode categorical variables
encoder = LabelEncoder()
y = encoder.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Create a decision tree classifier
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
# Create a confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
# print(cm)
# Plot the confusion matrix graphically
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d",
xticklabels=encoder.classes_, yticklabels=encoder.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()

# Post-processing
# Reverse label encoding
predicted_labels = encoder.inverse_transform(y_pred)
# print("Predicted Labels:", predicted_labels)
```

The provided code is responsible for performing a classification task using a decision tree classifier on a given dataset. It involves loading the dataset, preprocessing the data, training the classifier, making predictions on the test data, evaluating the accuracy of the classifier, and generating a confusion matrix. The steps involved in the code are as follows:

- The dataset is loaded from a CSV file using pandas.
- The dataset is split into features (X) and the target variable (y).
- Preprocessing is performed to handle missing values using the "most_frequent" strategy with the SimpleImputer class. Categorical variables are encoded using LabelEncoder.
- The data is then split into training and testing sets using train_test_split.
- A decision tree classifier is created using the DecisionTreeClassifier class.
- The classifier is trained on the training data using the fit method.
- Predictions are made on the test data using the predict method.
- The accuracy of the classifier is evaluated using the accuracy_score function.
- A confusion matrix is created using the confusion_matrix function to provide insights into the classifier's performance.

APPENDIX D. NLP TRANSFER LEARNING

```
import pandas as pd
from sklearn.model_selection import train_test_split
from transformers import GPT2LMHeadModel, GPT2Tokenizer, TextDataset,
DataCollatorForLanguageModeling
from transformers import Trainer, TrainingArguments

# Load the dataset
df = pd.read_csv('/content/input_to_NLP_06-16-23.csv')

# Shuffle and split the dataset into training and test sets
train_dataset, test_dataset = train_test_split(df, test_size=0.2,
random_state=42)

# Write the training and test sets to csv files
train_dataset.to_csv('train.csv', index=False)
test_dataset.to_csv('test.csv', index=False)

# Load pre-trained model and tokenizer
model_name = 'gpt2'
model = GPT2LMHeadModel.from_pretrained(model_name)
tokenizer = GPT2Tokenizer.from_pretrained(model_name)

# Define your training datasets
train_dataset = TextDataset(
    tokenizer=tokenizer,
    file_path="/content/input_to_NLP_06-16-23.csv",  # path to your
training file
    block_size=128
)

# Use the same dataset for training and evaluation
eval_dataset = train_dataset

# Use a data collator for language modeling
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False
)

# Define training arguments
training_args = TrainingArguments(
    output_dir="output",  # output directory
    overwrite_output_dir=True,  # overwrite the output directory
    num_train_epochs=50,  # number of training epochs
    per_device_train_batch_size=16,  # batch size per device during
training
    per_device_eval_batch_size=64,  # batch size for evaluation
    eval_steps=400,  # Number of update steps between two evaluations.
    save_steps=800,  # after # steps model is saved
    warmup_steps=500  # number of warmup steps for learning rate scheduler
)

# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,  # this is the same as the training dataset
)

# Fine-tune the model
trainer.train()

# Generate text
prompt_text = "Turner Robot and Sample_not_broken"
encoded_prompt = tokenizer.encode(prompt_text, add_special_tokens=False,
return_tensors="pt")

# Generate a batch of sequences
output_sequences = model.generate(
    input_ids=encoded_prompt,
    max_length=100,
    temperature=0.7,
    top_k=0,
    top_p=0.9,
)

# Decode the output sequences
for generated_sequence in output_sequences:
    generated_sequence = generated_sequence.tolist()
    text = tokenizer.decode(generated_sequence,
clean_up_tokenization_spaces=True)
    print(text)
```

The code demonstrates the process of training and utilizing a language model using the GPT-2 (Generative Pre-trained Transformer 2) architecture.

- Loading and Preparing the Dataset: The code begins by loading a dataset from a CSV file using pandas. The dataset is then shuffled and split into training and test sets using the train_test_split function from sklearn.model_selection. The training and test sets are subsequently saved as separate CSV files.

- Initializing the Model and Tokenizer: The pre-trained GPT-2 model and tokenizer are loaded using the GPT2LMHeadModel and GPT2Tokenizer classes from the Transformers library, respectively. The from_pretrained method is used to load the pre-trained GPT-2 model and tokenizer, based on the specified model_name.

- Creating the Training Dataset: A TextDataset is created to define the training dataset. The tokenizer is utilized to tokenize the text data, and the file_path argument specifies the path to the training file. The block_size parameter determines the maximum length of each sequence.

- Defining Training Arguments: Training arguments are defined using the TrainingArguments class. Parameters such as the output directory, number of training epochs, batch sizes, evaluation steps, save steps, and warmup steps are set.

- Initializing the Trainer: The Trainer class is initialized with the provided model, training arguments, data collator, and the training and evaluation datasets. The trainer manages the training process, including model checkpoints, evaluation, and training iterations.

- Fine-tuning the Model: The trainer.train() method is called to initiate the fine-tuning process. The model is trained on the training dataset, and the specified training arguments are used to control the training process.

- Generating Text: After training the model, text generation is performed. A prompt text is specified, encoded using the tokenizer, and passed to the model's generate method. The generated output sequences are then decoded using the tokenizer and printed as text.

The code showcases the complete pipeline for training a language model using GPT-2, including dataset preparation, model initialization, fine-tuning, and text generation. It provides a framework for utilizing pre-trained language models and adapting them to specific text generation tasks.