Integrating Machine Learning into Engineering Education through an Expanded Design of Experiments Laboratory

Dr. Yan Wu, University of Wisconsin - Platteville

Yan Wu earned her bachelor's degree in Precision Instruments, with a minor in Electronics and Computer Technology, from Tsinghua University in 1996, followed by an M.S. in Mechanical Engineering from the University of Alabama in 1998. She completed her Ph.D. in Electrical Engineering at the University of Illinois Urbana-Champaign (UIUC) in 2005, focusing on MEMS and electrostatic actuation. Subsequently, she served as a Postdoctoral Research Associate in the Department of Mechanical Science and Engineering at UIUC, engaging in scanning probe microscopy research. In 2009, she joined the Department of Engineering Physics at the University of Wisconsin–Platteville. Her sabbaticals include visiting scholar positions in the Department of Biomedical Engineering at the University of Wisconsin–Madison (2015–2016) and the Department of Precision Instruments at Tsinghua University (2024–2025).

Dr. Harold T. Evensen, University of Wisconsin - Platteville

Hal Evensen earned his doctorate in Engineering Physics from the University of Wisconsin-Madison, where he performed research in the area of plasma nuclear fusion. Before joining UW-Platteville in 1999, he was a post-doctoral researcher at the University

Integrating Machine Learning into Engineering Education through an Expanded Design of Experiments Laboratory

Abstract

Machine learning (ML) is a cornerstone of artificial intelligence, yet its instruction in engineering curricula often remains confined to standalone courses designed for computer science (CS) majors, typically requiring substantial programming experience. This paper presents a practical approach to integrating ML into existing engineering courses for non-CS majors by expanding a traditional Design of Experiments (DoE) laboratory module. In this expanded DoE lab, students first conduct a full factorial DoE using a catapult experiment. In the second part of the lab, the collective data sets from all groups are shared with every student, serving as the foundation for constructing ML regression and classification models, introducing students to core ML concepts through hands-on labs.

This approach introduces several innovations to engineering education. Students engage in experiential learning by interacting with a catapult and collecting original data, rather than relying on generic online sources. ML is integrated by connecting it to DoE, a standard curriculum topic, allowing seamless inclusion without major course changes. DoE is highlighted for situations with limited or costly data, while ML is employed for large datasets, providing students with a thorough overview of statistical and predictive methods in engineering.

The objective of this expanded DoE lab is to introduce fundamental machine learning concepts to engineering majors, enabling them to collaborate effectively with data scientists on interdisciplinary projects. The instructional materials, including laboratory guides, source code, and assessment tools, are designed for straightforward adaptation into any course covering experimental methods and data analysis. Student learning outcomes were evaluated through project presentations, quizzes, and reflective surveys over two semesters. Results indicate increased engagement and improved comprehension of both experimental design and machine learning principles.

1. Introduction

Machine learning (ML) has emerged as a pivotal technology in today's Artificial Intelligence (AI)-powered world, transcending its origins as a specialized skill within data science to an essential component of modern engineering practice. Contemporary engineered systems generate vast volumes of data and operate in increasingly complex, nonlinear environments where traditional analytical models often prove inadequate. In contrast, ML offers robust solutions for challenges such as pattern recognition, forecasting, and optimization [1]. These tasks are central to innovation across engineering sectors, including automotive manufacturing, construction, and advanced electronics. This broad applicability has intensified the job market demand for engineers capable of integrating domain expertise with data-driven methodologies. Incorporating ML into engineering curricula is therefore not merely advantageous but imperative for preparing students to engage thoughtfully with AI technologies, navigate the ethical dimensions of their use, and make well-informed decisions in professional practice. Ultimately, a solid understanding of ML methodologies fosters more effective engineering solutions and equips engineering graduates to excel in a dynamic technological landscape.

Despite this growing need, ML instruction in higher education remains concentrated primarily within computer science and data science programs, where students typically possess strong programming skills and a solid foundation in statistics. Efforts to extend ML education to non-CS majors, such as those in mechanical engineering, are ongoing and characterized by significant curricular experimentation. One proposed strategy is program-level curriculum redesign, in which short ML modules are embedded across multiple semesters and integrated into core engineering courses [2]. This distributed approach provides repeated exposure to ML concepts, ranging from foundational statistics to domain-specific applications, such as robotics. However, its adoption has been limited due to the high degree of faculty coordination and infrastructure support required. A more common alternative is the standalone ML course, designed either for broad applicability for all engineering programs [3] or tailored to specific engineering domains using project-based learning and discipline-relevant datasets [4]. For non-CS majors, such courses are typically electives for junior or senior students. Although they offer deeper skill development, their accessibility is constrained by already crowded curricula and variability in faculty availability. An increasingly popular approach is the embedded modular model [5], [6], in which 1–3 week ML units are incorporated into existing engineering courses and supported by labs or mini-projects. These modules maximize accessibility and minimize curricular disruption, and surveys [5], [6] indicate they effectively enhance novice students' awareness and confidence in ML. Their success depends on careful design, appropriate instructional support, and alignment with broader program goals [7], [8].

Building on the growing interest in expanding ML instruction for non-CS majors, this work presents a practical modular approach for integrating ML into existing engineering courses through the expansion of a traditional Design of Experiments (DoE) laboratory. DoE, which originated in the 1920s, is a foundational statistical methodology for systematically examining cause-and-effect relationships and remains central to process optimization and product improvement across mechanical, industrial, chemical, materials, electrical/electronic, and energy engineering. Consequently, many engineering programs have established well-designed courses or lab modules to cover the topic of DoE [9]. A widely used instructional tool for introducing DoE concepts is the tabletop catapult experiment, in which students investigate how multiple controllable factors influence a measurable outcome, typically the launch distance of a projectile.

We position DoE and ML as complementary analytical frameworks: DoE is especially effective when data are limited or costly to collect, whereas ML excels in extracting patterns and predictive insights from the large, complex datasets increasingly generated by modern engineering systems, particularly those enabled by embedded sensing and Internet of Things (IoT) technologies. Integrating these perspectives enables students to understand how classical statistical reasoning and contemporary data-driven approaches can jointly support decision-making in engineering applications. In the expanded laboratory module described in this work, students first conduct a full factorial DoE using the catapult experiment. Subsequently, the aggregated data from all laboratory groups are shared and used as the basis for developing ML regression and classification models. This two-part structure, which we refer to as the DoE to ML module in the following, provides a coherent, hands-on introduction to both DoE and ML, reinforcing key concepts such as variable selection, data uncertainty, regression analysis, and predictive modeling while situating ML naturally within an existing engineering curriculum.

An additional strength of this instructional approach is that the dataset used for the ML component is generated directly by students and curated by instructors across multiple semesters.

Each iteration of the expanded DoE laboratory contributes new measurements to a cumulative data repository, growing a dataset that reflects authentic variability in student-collected data. The dataset is hosted on GitHub, a cloud-based platform where users can store, share, and work together with others to write code. While ML activities typically benefit from large, high-quality datasets, reliance on student-generated data introduces challenges, including measurement noise, inconsistencies, and unexpected model behavior. Many existing ML courses or instruction modules mitigate these issues by using pre-existing, well-documented datasets from benchmark repositories [10] or published research [4], which allow instructors to focus more on algorithmic instruction. However, when integrating ML into an existing domain-specific course, it can be difficult to align such pre-existing datasets meaningfully with the rest of the course content. By involving students directly in the data collection process, they experience firsthand the importance of cleaning data and preserving data authenticity, exposing them to the full ML workflow. Moreover, students tend to exhibit higher engagement and ownership when working with data they have collected themselves, reinforcing experiential learning principles [11],[12]. Although this study focuses on the catapult-based DoE experiment, the same data collection framework can be adapted to any complex system or laboratory context, enabling the development of discipline-specific datasets tailored to the needs of individual engineering programs.

In the following sections, we detail the design and implementation of the DoE to ML laboratory module. The module aims to provide students with sufficient background and hands-on experience in ML so they can collaborate effectively with data scientists on interdisciplinary projects. The instructional materials are embedded within a junior-level course on experimental design and metrology in Engineering Physics at the University of Wisconsin-Platteville, leveraging students' prior exposure to experimental statistics gained through standard physics laboratory sequences. Although an introductory Python programming course is recommended, it is not required to successfully engage with the module. In the first half of the semester, students gain familiarity with statistical tools such as measurement uncertainty analysis, hypothesis testing, and regression through several experiments. These activities form a natural foundation for placing the module near the end of the course. While modern physics is a formal prerequisite, the DoE to ML module does not depend on specialized physics content, making the instructional materials readily adaptable to a wide range of courses emphasizing experimental methods and data analysis.

To evaluate the effectiveness of the module in introducing machine learning, we assessed student engagement and learning across two semesters using project presentations, quizzes, and reflective surveys. These evaluations consistently showed active engagement and evident learning gain of both DoE principles and introductory ML concepts, demonstrating the effectiveness of integrating ML instruction into an established DoE laboratory structure.

2. Module Design

2.1 Existing structures of the course

The Engineering Physics Program is an ABET-accredited program at the University of Wisconsin-Platteville. Its curriculum combines elements of mechanical and electrical engineering with advanced coursework in physics. At the junior level, students enroll in EP4010 Engineering Physics Lab, a required course for Engineering Physics majors and typically their

first course after completing the standard physics sequence. This intensive laboratory course emphasizes experimental design, experimental statistics, metrology, and quality measurements. EP4010 typically enrolls about 10 students per semester and meets twice a week for two-hour lab sessions. Working in small groups of two or three students, they complete six experiments in total: two short, one-week experiments; four full, two-week experiments; and a final design project. Throughout the semester, lectures on measurement theory and experimental design are interwoven between the lab sessions.

Design of Experiments (DoE) in EP4010 is taught as a two-week module centered on a catapult experiment. In this activity, the catapult shown in Figure 1is treated as a "black box" system that students must analyze and model. The system output is the launch distance (labeled y in Figure 1), and it has seven possible input variables (labeled A through G in Figure 1). Students select three of these variables as factors and design a 2-level, 3-factor full-factorial experiment to determine within the three selected variables which factors and interactions most significantly affect launch distance. They primarily use Excel for data analysis and develop prediction models by calculating the effects of each factor manually. At the end of the module, students test their DoE model by either identifying the settings needed to achieve a specific target distance or adjusting the catapult to exceed a specified threshold. The resulting prediction model generally performs well when the input ranges are kept small, where the catapult behaves approximately as a linear system.

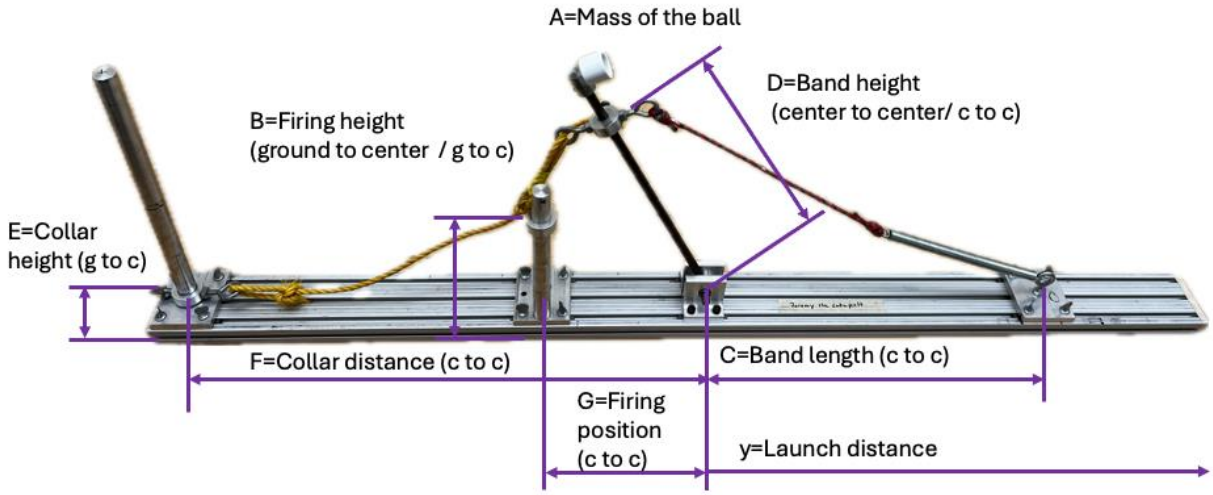

Figure 1: *Experimental setup of the catapult and definitions of all input variables and output launch distance. The measuring protocol clarifies whether distance variables are recorded from center-to-center or from ground-to-center.*

Building on the existing course structure, we designed the DoE-to-ML module as a two-part series. The first part is the existing two-week DoE catapult experiment. The second part is a two-week ML extension, separated from the first by a lecture session. This lecture session allows the instructor to introduce key ML concepts and facilitates class discussion and presentations that conclude the earlier experiment.

This break also enables the instructor to pool DoE data from all student groups into a cumulative repository hosted on GitHub. Because the catapults used by different groups are identically

constructed and the variables are consistently defined, the combined data can be treated as coming from a single theoretical catapult that represents the behavior of all experimental units. Traditional DoE models typically perform well only within the specific experimental range and can fail when previously fixed parameters change. By pooling data from groups that use different settings and input variables, we can create an ML model that better predicts behavior across a broader range of conditions and captures nonlinear responses. Unlike static DoE models, ML models are inherently iterative: each new catapult launch can be recorded and used to further train and refine the models, highlighting the learning nature of the process.

2.2 Content of ML module and learning objectives

Given the two-week time constraint for the ML component, our focus is limited to supervised learning within the broader field of machine learning. Over these two weeks, students use data from the cumulative repository to build two types of models: polynomial regression models that predict numerical launch distances and logistic regression models that classify shots as “long” or “short.”

To introduce classification, we convert the continuous launch distance into a binary label using a physical threshold (200 cm): shots beyond the line are coded as “1” (long) and the rest as “0” (short). This mirrors real engineering contexts where only categorical outcomes are available, though it necessarily discards useful numerical information. Moving the decision boundary also allows us to create balanced or imbalanced datasets. For instance, most throws will be labeled "0” (short), producing a dataset with many more "short" labels if the physical threshold is 300cm. Engaging with imbalanced data helps students see why accuracy can be misleading and why metrics such as precision and recall are essential for classification tasks. Students also experiment with shifting the logistic-regression probability cutoff to explore the trade-offs among these metrics. Although this simplified approach is intentionally helpful for learning the mechanics of classification, we caution students that when numerical data are available, the correct workflow is to build a regression model first and then derive classifications from its predictions rather than collapsing the data prematurely.

By conducting both regression and classification on the same physical system, students gain a tangible understanding of how machine-learning methods extend beyond traditional numerical regression techniques. In the final phase of the module, students deploy their trained models during live testing, allowing them to compare predicted distances or short/long classifications with real-time experimental outcomes and reflect on the strengths and limitations of data-driven models.

We define the learning objectives of the ML part as follows:

- **Understanding key concepts and terminology in machine learning:**
    - Describe core ideas of supervised learning, including training vs. testing data, labels, features, and learning algorithms based on minimizing the cost function.
    - Distinguish classification from regression in supervised learning and evaluate each using appropriate metrics.
    - Identify sources of bias in machine learning. Explain the bias-variance trade-off by analyzing how higher polynomial degrees of a model affect its performance. cost function.
- **Developing skills to train and evaluate the supervised learning model:**

- Preprocess data by cleaning and scaling raw data, choosing justified data-splitting strategies to reduce bias, and transforming labels as needed.
- Implement models using algorithms such as polynomial regression and logistic regression.
- Use cross-validation techniques and model evaluation metrics to select and optimize model performance.

Since the course does not require prior programming experience, we begin with Excel, an accessible, low-barrier platform, to build intuition about data manipulation, regression concepts, and the logic behind model construction. Students first revisit their own DoE data, applying new analytical tools such as the LINEST function and Excel Solver in regression analysis. This also gives them an opportunity to clean the data for missing points or inconsistencies, a final check before their data permanently joins the data repository.

Student next work with example Excel spreadsheets containing a small but representative subset of the cumulative dataset hosted on GitHub. The dataset is intentionally small enough to be handled in Excel yet large enough to illustrate the need for an ML model. The Excel activities are deliberately hands-on: by configuring loss functions, objectives, and constraints in Excel Solver, students internalize key ideas about modeling and error metrics before encountering automated ML workflows. This phase aims to provide conceptual transparency and nurture a deeper understanding of why models behave as they do.

Once students are comfortable with these concepts, we transition to Python in Google Colab, a cloud-based Jupyter environment that supports free open-source libraries such as Scikit-learn for ML model building. Students can scale their analyses to larger datasets and explore more sophisticated tools such as feature scaling, pipelines, and polynomial models. The step-by-step Colab notebooks, combined with AI-supported debugging and explanation, are designed to lower programming barriers while preserving conceptual rigor. Students apply the same workflow first to the small dataset they previously used in Excel and then to the full repository, highlighting how ML extends traditional DoE thinking to nonlinear and high-variability systems. Optional extensions, such as creating interactive sliders for deploying models, reinforce the connection between digital modeling and real-world engineering applications.

Throughout the module, the instructional design leverages a use–modify–create progression: students first operate pre-built tools, then modify partially structured code, and ultimately construct or adapt their own ML models. This approach reduces programming anxiety for non-CS majors while ensuring that conceptual understanding, not technical syntax, anchors learning. Together, the tiered software progression and scaffolded analytical tasks cultivate a coherent bridge from DoE to ML, enabling students to appreciate both the methodological continuity and the enhanced predictive power of ML afforded by modern computational modeling.

2.3 Assessment tools

The direct assessment of the ML part of the module integrates documentation, conceptual evaluation, and an interactive oral presentation to measure both procedural skills and conceptual understanding.

Students maintain a structured lab notebook throughout the activities, responding to targeted prompts that guide their analysis and reflection. This notebook is submitted along with several technical artifacts, including Excel spreadsheets demonstrating regression and ML simulation results, and Jupyter notebooks containing student-modified Python code for both regression and classification tasks. These artifacts are evaluated for completeness and evidence of methodological understanding.

To assess conceptual learning, students complete an open-book quiz on terminology and concepts in ML. Administered after students finish the module but before their small group presentation, the quiz requires them to define key concepts in their own words and demonstrate comprehension beyond procedural execution. The details of the quiz questions are in Appendix I.

The final assessment is a 10 to 15-minute small group oral presentation held in the instructor's office. Rather than delivering a formal slide presentation, students engage in an informal discussion of their work, focusing on the problem statement, experimental approach, key results, and interpretation of their findings. The instructor uses targeted questions to probe student reasoning, clarify misconceptions, and evaluate depth of understanding. Students need to submit a handout with their key findings before the meeting. Typical questions asked during the small group presentations are in Appendix II.

In addition to these direct measures, students complete pre- and post-module surveys that serve as an indirect assessment of learning gains, confidence, and engagement with the DoE to ML module. The questions of the pre- and post-module surveys are in Appendix III.

3. Module Implementation

In Fall 2024 and Fall 2025, the DoE-to-ML module was delivered by different faculty members each semester, with enrollments of 12 and 9 students, respectively. The mode of instructional delivery differed across the two semesters: in Fall 2024, the lecture connecting DoE and ML was provided asynchronously through recorded videos hosted on the learning management system (Canvas), while in Fall 2025, this material was delivered during an in-person class session.

To support students' engagement with the full DoE to ML learning sequence, a comprehensive set of instructional materials was provided [13]. These included a 16-page written primer introducing core DoE concepts; a detailed laboratory manual for the catapult-based DoE experiment; slide decks used in the ML introductory lecture; a standalone ML lab manual; example Excel spreadsheets illustrating polynomial regression and classification workflows; sample Python code for regression and classification tasks hosted on GitHub; and a shared repository of class-generated catapult data maintained online. There were only minor updates in the slide decks, concept quiz, and the sample Python coding from Fall 2024 to Fall 2025. Notably, the Fall 2025 cohort worked with a substantially expanded dataset that incorporated both Fall 2024 data and newly collected experimental data, allowing students to explore ML models under more diverse and realistic data conditions.

When implementing the catapult-based DoE experiment, we adopted several practices that helped position students for the subsequent ML extension of the module. Students were organized into teams of at least two people, which increased the total number of experimental runs and thereby enriched the dataset used for later modeling. To ensure comparability across

teams, we established a shared measurement protocol as illustrated in Figure 1; for example, all teams measured launch distance from the catapult arm's axle using centimeters. We also informed students early on that their data would be pooled for class-wide analysis in the ML portion of the module, which encouraged careful documentation. Teams were guided to select diverse input settings so that the aggregated dataset would span a broad range of factor values. Students were further reminded to record the values of all fixed inputs, both for within-team consistency and for later data sharing across the class.

After completing their experimental runs, each team constructed a DoE model based on a full factorial design. Using this model, teams predicted how different factor settings would influence launch distance. Model validation was structured around two tasks: first, teams attempted to reach a specified target distance using predictions from their model (a numerical outcome), and second, they attempted to surpass a predetermined threshold (a binary outcome). The threshold challenge provided a natural bridge to the classification task introduced in the ML module. During validation, students regularly encountered the limitations of their models. Particularly, when testing conditions outside their design space or altering factors previously held constant, their DoE quickly falls apart. This prompted discussions about linearity assumptions and the need for transformations when dealing with nonlinear responses. These observations prepared students for the transition to ML techniques, which can accommodate larger datasets and more complex relationships but require increased computational resources. At this stage, data from all teams were combined and shared with the full class, allowing students to examine data quality, participate in cleanup, and prepare the dataset for ML modeling.

Upon completing the DoE portion, students were given a pre-module survey and then proceeded to the ML portion of the module described in Section 2.2. The post-module survey was collected after the students finished their small group oral presentation on their ML models.

4. Assessment Results and Discussion

4.1 Direct assessment

To evaluate the effectiveness of the DoE-to-ML module in meeting its ML learning objectives, we applied a four-point rubric that aligns assessment tools with the intended outcomes as detailed in Appendix IV. The rubric includes four performance levels: Exemplary (4 points), Accomplished (3 points), Developing (2 points), and Inadequate (0 points). Figure 2 presents the average student scored over two semesters (21 students in total).

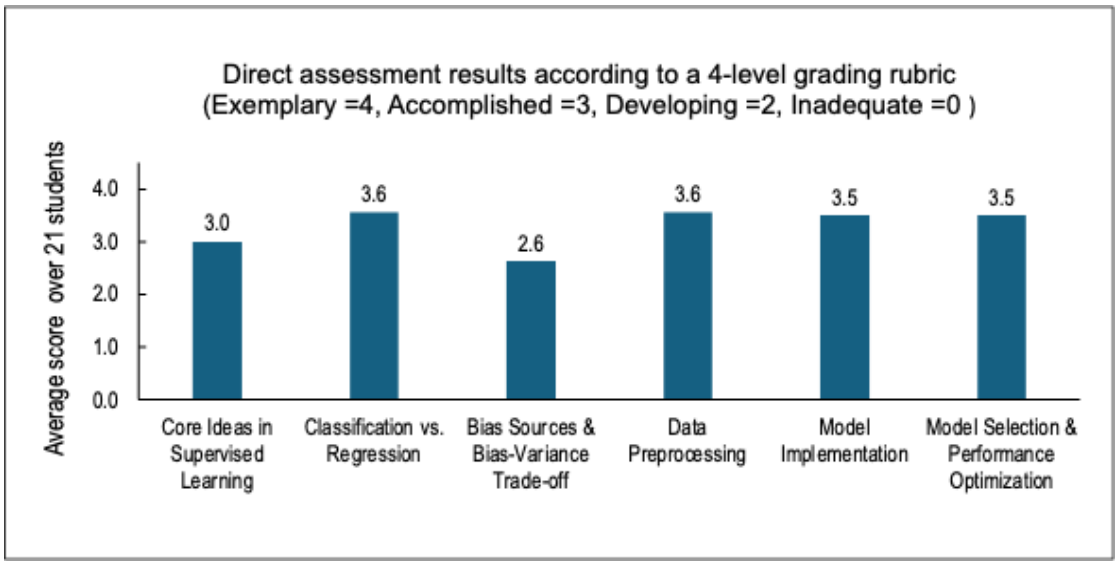

Figure 2: *Direct assessment results showing average performance score of 21 students in two semesters according to the 4-point rubric in Appendix IV.*

Students performed between the Accomplished and Exemplary levels on the three skill-related learning outcomes, with average scores of 3.6 for Data Preprocessing, 3.5 for Model Implementation, and 3.5 for Model Selection and Performance Optimization. The small gap from the maximum score of 4.0 suggests that, while students were generally able to execute the tasks, they often lacked deeper, data-driven justification for their decisions. This pattern suggests that the module effectively supported the development of core supervised-learning skills, yet students' ability to transfer these skills to unfamiliar contexts remains limited.

Performance on concept-based outcomes was more variable. Students earned average scores of 3.0 for Core Ideas in Supervised Learning, 3.6 for Distinguishing Classification from Regression, and 2.6 for Bias and the Bias–Variance Trade-off. The relatively strong performance on distinguishing classification from regression reflects repeated emphasis in the module, although gaps persist in students' understanding of evaluation metrics, particularly for classification models, as the nuances among accuracy, precision, and recall can be challenging.

Students performed at the Accomplished level for the core ideas in supervised learning. Most students clearly understand the purpose of splitting data into training, validation, and test sets. They demonstrated an inconsistent understanding of the role of the cost function in ML. Although Excel Solver makes the mechanism visible, Python-based work obscures underlying computations with a single function in the Scikit-learn library. This inconsistency indicates a need for more explicit instructional guidance to help students connect procedural steps with underlying concepts.

The lower performance on Bias Sources and the Bias–Variance Trade-off points to a conceptual gap. Students were able to identify data-driven sources of bias and apply mitigation strategies but did not connect the bias-variance trade-off to model-driven bias. They tended to view the trade-off purely as a technical step for model optimization, without recognizing that it is fundamentally about bias as well. This pattern suggests that students require clearer support in recognizing how the structure of the ML model itself introduces bias. For example, using a first-order linear model

to represent a second-order physical system consistently leads to systematic error, regardless of the amount of data provided.

Overall, students demonstrated strong performance in achieving the learning objectives for supervised learning, particularly on skill-based outcomes where average scores approached the Exemplary range. These results indicate that the module successfully supported students in executing key supervised-learning tasks and provided a solid foundation for further development. Although concept-based outcomes showed greater variability, the patterns suggest clear opportunities for targeted instructional refinement rather than deficiencies in student effort or engagement.

4.2 Indirect Assessment

We next examine evidence from pre- and post-module surveys in Appendix III. This additional perspective allows us to assess how students' self-reported understanding evolved and how well these perceptions align with their demonstrated performance.

Pre-module survey results showed that students began with moderate familiarity with machine learning (mean 2.7) but a strong sense of its importance (mean 4.1). Their initial ideas of ML applications ranged from accurate, engineering-relevant intuitions about prediction, classification, and automation to more advanced examples referencing neural networks, encryption, and industrial automation. At the same time, several misconceptions were evident: some students assumed ML could "automate all labor jobs," others viewed ML as inherently less accurate than human judgment, or misunderstood models as fixed sets of instructions or physical machines. Some representative quotes are

- *"It can be applied to almost any application. Most commonly in applications where some sort of predictive test is helpful or when there is a lot of data or easy/cheap data available."*
- *"It can be really good at making educated guesses and estimates about various topics in the real-world if trained correctly."*

Students' explanations of what constitutes an ML model reflected a wide range of sophistication. Many correctly described models as systems that learn patterns from data to make predictions, sometimes referencing input/output structure, parameters, or training mechanics like the quotes below. Others, however, held persistent misconceptions: some described the model as a set of human written instructions, the testing procedure, the physical machine itself, or assumed that models continually update without re-training.

- *"A program/algorithm that takes in data and processes it in some sort of way that might identify patterns or trends and uses what it knows to answer a question."*
- *"A machine learning model is a system that can track patterns and learn through collecting data."*

The overlapping self-assessment questions administered before and after the module provide a complementary perspective on students' perceived learning. As shown in Figure 3, average self-reported understanding increased across all concepts, with pre-module means ranging from 2.0 to 3.4 and post-module means rising to 3.4–4.0 on the 5-point scale. The corresponding normalized learning gains, summarized in Table 1, further illustrate these patterns. The highest gains occurred for the concepts of classifier and model training (both 57%), aligning with

students' strong performance on related skill-based assessment tasks. More moderate gains were observed for supervised learning (35%) and recall (48%), while precision showed the lowest gain (28%). The term "precision" is commonly used in early content of this course in the context of measurement uncertainty. The low normalized gain could be a result influenced by students' relatively high pre-module familiarity with the term in non-ML contexts. Overall, the pattern of raw and normalized gains is consistent with the direct-assessment findings: students made meaningful progress in their conceptual understanding of core supervised-learning ideas, though areas involving classification metrics and distinctions in terminology continue to present challenges.

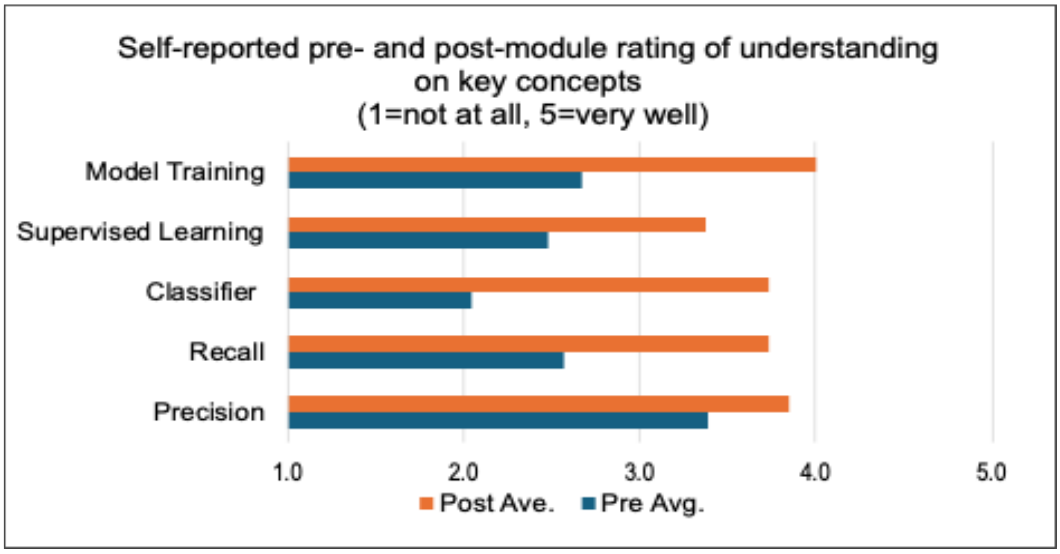

Figure 3: *Unpaired pre- and post- module averages showing students' self-assessed understanding level on key machine learning concepts.*

Table 1 *Average self-reported understanding before and after instruction (1–5 scale), with raw and normalized learning gains*

Concept	**Pre Avg.**	**Maximum Possible Gain (5-Pre Avg.)**	**Post Ave.**	**Raw Gain**	**Normalized Gain (Raw gain/ Maximum Possible Gain)**
Model Training	2.7	2.3	4.0	1.3	57%
Supervised Learning	2.5	2.5	3.4	0.9	35%
Classifier	2.0	3.0	3.7	1.7	57%
Recall	2.6	2.4	3.7	1.2	48%
Precision	3.4	1.6	3.8	0.5	28%

The post-module survey shows students' overall interest in machine learning slightly increased (mean 3.63), and they expressed moderate willingness to learn more in a future course (3.26). Their confidence in building ML models was moderate (2.8), which is reasonable given the module's introductory scope and the non-CS background of the audience. Satisfaction varied widely (1–5), with an overall average of 3.2, indicating a slightly positive level of satisfaction. Several factors likely contributed to the widespread responses. Students entered the module with diverse prior exposure to machine learning. For those already familiar with advanced concepts such as neural networks and weighted connections, the introductory focus may have felt insufficient. Conversely, students with limited prior knowledge sometimes found the pace fast and expressed a desire for additional guidance, particularly during the Excel-based activities, where the lack of built-in explanatory space required the instructor to provide extensive verbal clarification. Students in the Fall 2024 cohort reported that having access to a recorded walk-through helped address these challenges. Overall, the satisfaction data suggest that while the module was generally well-received, differences in students' backgrounds and the instructional affordances of the tools shaped their perceptions of the implementation.

Students' responses to the open-ended question, "What did you like most about the module?" highlighted several consistent strengths of the instructional design.

1) Authenticity: working with real, class-generated data made the learning feel meaningful and relevant.
    - *"I thought it was cool to see all of the class's data compiled to train an ML model and how it can be setup and used."*
    - *"I liked that we were able to look at our previous collected data and make inferences based on it."*
2) Continuity: connecting ML activities to earlier DoE labs reinforced understanding and highlighted curricular coherence.
    - *"I enjoyed that this lab built off the DOE data, allowing us to further dive into DoE analysis and machine learning"*
3) Novelty: being introduced to ML concepts was exciting for many students, especially those with little prior exposure.
    - *"Seeing applications of machine learning & what it's actually used for"*
    - *"Being introduced to ML in general. I was entirely unfamiliar with how it functions and what principles it was based on"*

Analysis of students' responses to the open-ended question "What improvements would you suggest for future iterations of this module?" reveals three dominant themes regarding areas for improvement in future iterations.

1) Clearer instructional materials, especially terminology, annotated code, and more coherent structure across Excel and Python components.
    - *"Tweaking of the Python code to make it easier to find/understand certain functions. Same for Excel modules, just more/better annotation"*
2) More step-by-step guidance, rather than relying on premade spreadsheets or scripts.
    - *"Maybe having it split up more. More time for each model and less of following the lab code given and running it (more of a guided how we would code this)."*

3) Reduced cognitive load, including more time to absorb ML concepts and fewer document switches during activities.
    - *"Slower explanation of the principles behind Machine Learning. I felt that I had a grasp on the fundamentals but just needed more time to work with the information"*
    - *"The instructions felt confusing at times, especially when I had to constantly switch between multiple documents at a time to make progress."*

Taken together, the indirect assessment data show that students entered the module with varied prior understanding and several misconceptions but made meaningful conceptual gains and maintained or increased their interest in machine learning. They valued the authenticity and relevance of using their own experimental data and saw clear connections to earlier course material. At the same time, their feedback highlights a need for clearer scaffolding, more guided practice, and a streamlined workflow to reduce cognitive load and support deeper learning.

5. Conclusion

This work demonstrates a practical, scalable, and effective approach for integrating machine learning into engineering curricula by expanding an existing Design of Experiments (DoE) laboratory. By anchoring ML instruction in a familiar, hands-on experimental context, the DoE to ML module provides students with a coherent learning progression that connects foundational statistical reasoning with modern data-driven modeling techniques.

Assessment results show that it produced strong student skills in preprocessing, model implementation, and optimization, and improved conceptual understanding of regression, classification, and supervised learning; however, gaps remain in areas such as the bias–variance tradeoff and classification metrics. Students valued the authenticity of class-generated data, the seamless connection between DoE and ML, and the novelty of applying ML in engineering, but requested clearer scaffolding, more guided practice, and reduced cognitive load when juggling multiple tools and documents.

Overall, the results show that embedding ML instruction within an established DoE framework is both pedagogically effective and logistically feasible for engineering programs. Future improvements will focus on refining instructional materials, streamlining workflows, and deepening conceptual support, enabling the module to serve as a robust model for integrating machine learning into domain-specific engineering courses

Acknowledgement

The authors would like to thank the Kern Family Foundation for their support through the KEEN (Kern Entrepreneurial Engineering Network) Engineering Unleashed Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Kern Family Foundation.

References

[1] D. M. Dimiduk, E. A. Holm, and S. R. Niezgoda, "Perspectives on the Impact of Machine Learning, Deep Learning, and Artificial Intelligence on Materials, Processes, and Structures

Engineering," Integrating Mater. Manuf. Innov., vol. 7, no. 3, pp. 157–172, Sep. 2018, doi: 10.1007/s40192-018-0117-8.

[2] J. Frochte, M. Lemmen, and M. Schmidt, "Seamless Integration of Machine Learning Contents in Mechatronics Curricula," in 2018 19th International Conference on Research and Education in Mechatronics (REM), Delft: IEEE, Jun. 2018, pp. 75–80. doi: 10.1109/REM.2018.8421794.

[3] J. Wanliss, "Leveraging Novel Machine Learning in Engineering Education," in 2024 ASEE Annual Conference & Exposition Proceedings, Portland, Oregon: ASEE Conferences, Jun. 2024, p. 47741. doi: 10.18260/1-2--47741.

[4] J. Niemirowski, K. Cruse, and D. Hall, "Implementation and Evaluation of a Predictive Maintenance Course Utilizing Machine Learning," in 2023 ASEE Annual Conference & Exposition Proceedings, Baltimore , Maryland: ASEE Conferences, Jun. 2023, p. 43514. doi: 10.18260/1-2--43514.

[5] Y. Hao, Y. Liu, B. Liu, G. Amarantidis, and R. Ghannam, "Integrating AI in Engineering Education: A Comprehensive Review and Student-Informed Module Design for U.K. Students," IEEE Trans. Educ., vol. 68, no. 2, pp. 173–185, Apr. 2025, doi: 10.1109/TE.2025.3536105.

[6] Y. Xu, B. Zhao, S. Tung, and H. Hu, "Infusing Data Science into Mechanical Engineering Curriculum with Course-Specific Machine Learning Modules," in 2023 ASEE Annual Conference & Exposition Proceedings, Baltimore, Maryland: ASEE Conferences, Jun. 2023, p. 43958. doi: 10.18260/1-2--43958.

[7] K. Xia et al., "A modular approach for integrating data science concepts into multiple undergraduate STEM+C courses," in 2022 ASEE Annual Conference & Exposition Proceedings, Minneapolis, MN: ASEE Conferences, Aug. 2022, p. 42010. doi: 10.18260/1-2--42010.

[8] K. Mike and O. Hazzan, "MACHINE LEARNING FOR NON-MAJORS: A WHITE BOX APPROACH," Stat. Educ. Res. J., vol. 21, no. 2, p. 10, Jul. 2022, doi: 10.52041/serj.v21i2.45.

[9] A. R. Vazquez and X. Xuan, "A review of Design of Experiments courses offered to undergraduate students at American universities," 2023, doi: 10.48550/ARXIV.2309.16961.

[10] V. Lavor, F. De Come, M. T. Dos Santos, and A. S. Vianna, "Machine learning in chemical engineering: Hands-on activities," Educ. Chem. Eng., vol. 46, pp. 10–21, Jan. 2024, doi: 10.1016/j.ece.2023.09.005.

[11] D. Obada et al., "Teaching Basic Concepts in Machine Learning to Engineering Students: A Hands-on Approach," in 2024 ASEE Annual Conference & Exposition Proceedings, Portland, Oregon: ASEE Conferences, Jun. 2024, p. 48058. doi: 10.18260/1-2--48058.

[12] Lao, Natalie, "Reorienting machine learning education towards tinkerers and ML-engaged citizens," Ph.D. thesis, Massachusetts Institute of Technology, 2020. [Online]. Available: http://dspace.mit.edu/handle/1721.1/7582

[13] Y. Wu and H. T. Evensen, "From Design of Experiments to Machine Learning," *KEEN Card* [Online]. Available: https://engineeringunleashed.com/card/4365 [Accessed April 3, 2026].

Appendix I: Machine Learning Concept and Terminology Quiz

a. Training, Cross-Validation, and Testing data sets.
    i. What do these mean?
    ii. What are the relative sizes of these?
    iii. How are these used?
    iv. Why are these used?
b. Bias in ML models
    i. Explain what bias is in ML, using the catapult as an example
    ii. What are some techniques you used in this experiment to minimize bias?
c. The data scaling used in the demo Excel sheet is common practice in ML. What does it mean to "scale" the data?
    i. What does "scaling" mean?
    ii. Why would one scale their data?
d. What is the difference between a cost function and a loss function? Why are they important / how are they used?
e. In your own words, how does building a classification model differ from building a regression model? What changes/stays the same / is similar?
f. How do we evaluate a classification model with no MSE score?
g. What is an imbalanced dataset, in your own words?
h. What does it mean to change the "threshold" value from its 50% default?
i. What is a dummy classifier? When are they useful?

Appendix II: Typical Questions during Small Group Presentation

1. Bonus: Were you able to complete the "bonus" section?
    a. How did predictions align with reality (actual throw distances)?
    b. How could you improve your model if you had more time?
    c. Show me your interactive interface!
2. Classification model in Python
    a. What were your cutoffs for imbalanced sets?
        i. Excess "long"
        ii. Excess "short"
        iii. Original "200"
    b. Did you build a model with the expanded data set (_ABCDEFGy_raw.csv)
        i. What type of model was best?
        For example, the Linear model, threshold = 0.6, gave perfect test results for me
3. Regression model in Python
    a. What part of the Python code accomplishes the same function when you use Excel Solver?
    b. Did transforming the output variable have an effect? (Did it improve it? How do you know?)
    c. Did you retrain with the larger data set?
        i. What type of model was best?
        ii. How do you know your model was "better"? *MSE scores?*

4. Excel regression (DoE)
    a. Tabs showing LINEST, Toolpak, and Solver results
        i. LINEST & Toolpak:
            1. What is Std. Error of the Experiment?
            2. Show which coefficients meet the 95% confidence (TDIST test)
        ii. Solver: identify cost & loss functions
5. Excel Regression "ML"
    a. How are the input data scaled (tab #2)? Give and describe the equation for scaling raw data to the data for regression
    b. Excel: Which order of polynomial (1^{st}, 2^{nd}, 3^{rd}) was the "better" model, and how do you know / why would you say so?
    c. Data selection for training and cross-validation: comparing the two different ways to split data for training and validation, which way will reduce bias in ML models and build a robust model? Justify your choice.

6. Excel Classification "ML"
    a. Summarize the key differences and similarities between classification and regression model building
    b. What was the effect of increasing the order of your model?
        i. Accuracy:
        ii. Precision
        iii. Recall:
    c. Which cut-off value gave the best F1 score?
    d. Dummy classifier: How did the dummy classifier do in terms of accuracy, precision, and sensitivity? How does the performance of the ML models compare with the dummy classifier?

Appendix III: Pre- and Post-Module Survey Questions

Pre-Module Survey

Section 1: Initial Impressions of Machine Learning

1. **How familiar are you with the concept of machine learning?**

- Not familiar at all
- Slightly familiar
- Moderately familiar
- Very familiar
- Extremely familiar

2. **How important do you think machine learning is in today's world?**

- Not important
- Slightly important
- Moderately important
- Very important
- Extremely important

3. **What are your initial thoughts on how machine learning can be applied in real-world scenarios? (Open-ended)**

Section 2: Understanding Key Aspects of Machine Learning

4. **How well do you understand the following terms? (Rate on a scale of 1 to 5, where 1 is "Not at all" and 5 is "Very well")**
   - Supervised learning
   - Precision
   - Recall
   - Classifier
   - Model training
5. **Can you briefly explain what a machine learning model is? (Open-ended)**

Post-Module Survey

Section 1: Interest in Machine Learning

1. **How has your interest in machine learning changed after completing the module?**
   - Decreased significantly
   - Decreased slightly
   - Stayed the same
   - Increased slightly
   - Increased significantly
2. **Would you be interested in learning more about machine learning in future courses?**
   - Not interested
   - Slightly interested
   - Moderately interested
   - Very interested
   - Extremely interested

Section 2: Understanding Key Aspects of Machine Learning

3. **How well do you understand the following terms after completing the module? (Rate on a scale of 1 to 5, where 1 is "Not at all" and 5 is "Very well")**
   - Supervised learning
   - Precision
   - Recall
   - Classifier
   - Model training
4. **How confident are you in your ability to build a machine learning model?**
   - Not confident at all
   - Slightly confident

- Moderately confident
- Very confident
- Extremely confident

Section 3: Satisfaction with the Module

5. **How satisfied are you with the implementation of this module?**

- Very dissatisfied
- Dissatisfied
- Neutral
- Satisfied
- Very satisfied

6. **What did you like most about the module? (Open-ended)**
7. **What improvements would you suggest for future iterations of this module? (Open-ended)**

Appendix IV Machine Learning Catapult Lab Grading Rubric (continued on next page)

Learning Outcome	Exemplary (4)	Accomplished (3)	Developing (2)	Inadequate (0)	Specific Assignments / Evidence
Core Ideas in Supervised Learning	Accurately articulate the specific purpose, relative sizes, and roles of data sets and explains how ML algorithms minimize the cost function to find parameters.	Defines terms correctly; basic explanation the roles of cost functions, training, cross-validation and testing sets in ML	Defines most terms but cannot explain the role of the cost function, the purpose of data splitting	Fails to define fundamental machine learning building blocks or distinguish between input features and target labels.	**Quiz:** Appx I (a, d). **Presentation:** Experimental approach, Appx II Q3a, Q4
Classification vs. Regression	Differentiates numerical vs. categorical outputs; justifies metrics like MSE and F1-score.	Correctly identifies both but provides weak justification for specific metrics.	Can distinguish tasks but confuses metrics (e.g., MSE for classification).	Unable to distinguish the two tasks.	**Quiz:** Appx I (e, i). **Presentation:** Experimental approach, Appx II Q6a
Bias Sources & Bias-Variance Trade-off	Correctly mix data for model training; Uses MSE data to show how increasing complexity reduces bias but eventually increases variance.	Understands the general concept of bias-variance and justifies data mixing to reduce model bias.	Provides basic definition of bias but cannot explain how data selection or model complexity contributes to a biased model.	Cannot identify any sources of bias in the experiment	**Quiz:** Appx I (b). **Presentation:** Results and interpretation, Appx II Q3c, Q5b, Q5c
Data Preprocessing	Use correct tools for data mixing and splitting. Implements scaling and ln(y) transformation with justification.	Performs scaling/splitting correctly; implements transformation with weak justification.	Implements basic scaling but fails to justify splitting or transformation logic.	Fails to preprocess data or explain its necessity.	**Quiz:** Appx I (c). **Presentation:** Q3b, Q5a.

Learning Outcome	Exemplary (4)	Accomplished (3)	Developing (2)	Inadequate (0)	Specific Assignments / Evidence
Model Implementation	Builds functioning models in Excel and Python; saves final models as pickle modules in Python.	Implements models in both with minor errors; results are consistent across platforms.	Implements in only one platform (Excel or Python) or has significant logic errors.	Unable to produce a functioning model in either Excel or Python.	**Files:** Excel and Python source files **Presentation:** Appx II Q1
Model Selection & Performance Optimization	Employs cross-validation (CV) to select optimal model complexity and adjusts parameters—such as polynomial degree, probability thresholds.	Uses CV sets; attempts optimization but provides limited justification for specific parameter choices, such as polynomial order or cutoff values.	Reports standard performance metrics but lacks a systematic approach for model selection, often relying on training data or default settings.	Fails to apply evaluation metrics or validation sets to select, refine, or optimize model performance in either Excel or Python environments	**Quiz**: Appx I (f, g, h). **Presentation:** Appx II Q2, Q3, Q5b. Q6b, Q6c, Q6d