sustainability

MDPI

Article

Multi-Agent Systems and Machine Learning for Wind Turbine Power Prediction from an Educational Perspective

Fatih Soygazi

Department of Computer Engineering, Aydın Adnan Menderes University, Aydın 09100, Türkiye; fatih.soygazi@adu.edu.tr

Abstract: Artificial intelligence (AI) is an umbrella term that encompasses different fields of study, and topics related to these fields are addressed separately or within the scope of AI. Multi-agent systems (MASs) and machine learning (ML) are the core concepts of AI that are taught during AI courses. The separate explanation of these core research areas is common, but the emergence of federated learning has triggered their combined usage. This paper describes a practical scenario in the energy domain where these technologies can be used together to provide a sustainable energy solution for predicting wind turbine active power production. The projects in the AI course were assigned prior to the step-by-step learning of MASs and ML. These concepts were applied using a wind turbine energy dataset collected in Turkey to predict the power production of wind turbines. The observed performance improvements, achieved by applying various agent architectures and data partitioning scenarios, indicate that boosting methods such as LightGBM yield better results even when the settings are modified. Additionally, a questionnaire about the assignments was filled out by the student groups to assess the impact of learning MASs and ML through project-based education. The application of MASs and ML in a hybrid way proves valuable for learning core concepts related to AI education, as evidenced by feedback from students.

Keywords: machine learning; multi-agent systems; artificial intelligence; education; federated learning; sustainability; wind turbine; energy

Citation: Soygazi, F. Multi-Agent Systems and Machine Learning for Wind Turbine Power Prediction from an Educational Perspective. *Sustainability* **2023**, *15*, 16291. https://doi.org/10.3390/su152316291

Academic Editors: Derya Birant and Zerrin Isik

Received: 6 October 2023
Revised: 13 November 2023
Accepted: 15 November 2023
Published: 24 November 2023

1. Introduction

AI is increasingly emerging as a stronger method for achieving sustainability goals and supporting interdisciplinary studies day by day. At this point, it is far more beneficial to take an example from any field of study rather than simply explaining course content from a traditional educational perspective, and to develop a practical application that can work in that field. With this approach, the most current topics in AI, such as software agents and machine learning, are considered, and the aim is to develop and simulate a real-world system.

Wind energy generated by wind turbines is a sustainable method of electricity generation. Wind energy is environmentally neutral and an essential part of reducing climate change since, unlike fossil fuels, it does not release greenhouse gases or contribute to air pollution. The overall sustainability of wind energy continues to grow as technology progresses and wind turbine efficiency rises, making it a crucial role in the shift to a more sustainable and clean energy future.

AI courses are generally taught in universities using Russell and Norvig's book [1,2]. While this book covers all current topics related to AI, due to the breadth of the subject, not all of its content can be conveyed to students. Hence, it is crucial for a deeper understanding of these subjects at an expert level to focus on specific topics and learn them through implementation. Kong et al. [3] design an AI course involving the topics of AI, ML, supervised learning, and unsupervised learning. The course materials were crafted to facilitate participants' understanding of these key ideas, omitting mathematical formulas

and the coding specifics used in AI algorithm implementation. The pre-course and post-course surveys showed that participants, regardless of their backgrounds and gender, made significant progress in grasping AI concepts. Although this study follows the course with theoretical knowledge, computer science courses might be given by programming assignments related to the topics as they are applied in this paper. Ergezer et al. [4] introduce a new AI course encompassing subjects such as embedded systems, AI, digital signal processing, linear algebra, and probability and statistics. This course comprises theoretical lectures on these topics and practical laboratory assignments for hands-on experience. At the semester's conclusion, students create their own embedded AI projects that could be executed on embedded devices. Palanca et al. [5] design a course focusing on an MAS as a collaborative learning environment where the aim is to tackle real-world challenges. During the entire course, students are tasked with addressing various typical problems within the realm of an MAS, all focused on the same domain. In this case, the domain pertains to the improvement of urban mobility through taxis. To support this educational approach, a specialized MAS tool, known as SimFleet, was developed. SimFleet empowers students to simulate diverse scenarios related to these challenges and devise effective solutions where they can easily implement their own strategies for each simulation agent. This approach encourages students to engage in problem-solving through negotiation strategies, simulating the operation of urban taxi fleets, and through cooperative strategies where various agents collaborate to achieve common objectives. The courses in [4,5] are similar types of practical courses to the ones that are applied in this paper.

MASs like JADE [6], SPADE (Smart Python Agent Development Environment) [7–9], and Redis [10] are at the forefront of modern distributed computing and AI. These systems enable the creation of networks of intelligent agents that can communicate, collaborate, make autonomous decisions, and mirroring complex real-world scenarios. JADE, for instance, is a robust Java-based framework that provides a comprehensive platform for developing and deploying MAS applications. JADE provides support for different message transport protocols, including HTTP, IIOP (Internet Inter-ORB Protocol), and JADE's own native transport protocol. Foundation for Intelligent Physical Agents (FIPA) Agent Communication Language (FIPA-ACL) [11] is the primary communication language used, which defines the message structure and semantics for agent communication. JADE agents use FIPA-ACL to exchange messages. SPADE, on the other hand, specializes in creating agent-based solutions for intelligent and adaptable systems. SPADE agents also use the FIPA-ACL as the standard communication language, although its message transport mechanism is based on XMPP (Extensible Messaging and Presence Protocol) [12]. Redis, while primarily known as an in-memory data store, also plays a pivotal role in facilitating agent communication and data sharing within distributed environments [13]. Actually, Redis is not typically associated with being an MAS or a platform for building MASs. But its messaging infrastructure assists users to act like an MAS [14].

There are various studies where MASs can be used together with AI for different purposes. Grzonka et al. [15] introduce an innovative model known as the Multi-Agent-System-based Cloud Monitoring (MAS-CM) model. This model is tailored to enhance the efficiency and security of tasks related to collection, scheduling, and execution in large-scale service-oriented environments. Specifically, the combined usage of an MAS and ML might be applied in various use cases. Drogoul and Zucker [16] propose an agent-oriented method, named Cassiopeia, which was designed to address a challenge in software engineering which involves having MAS techniques. Cassiopeia is characterized by its independence from implementation techniques, the definition of agents in terms of three levels of roles, and a methodological process that combines bottom-up and top-down approaches. This paper highlights how Cassiopeia enables the integration of ML techniques into MAS design. It classifies and incorporates ML techniques based on their compatibility with different levels of behaviors and suitability for specific cases, offering a comprehensive perspective on the potential use of various ML techniques within MAS

design methodologies. Examples drawn from the RoboCup challenge demonstrate these techniques, and the applied MAS and ML integration has been implemented following Cassiopeia. When considering the adoption of an MAS methodology as a comprehensive approach to utilizing both an MAS and ML, it is important to recognize that mastering such a methodology through a step-by-step course may not be feasible within the coverage of a single semester. Therefore, I have chosen not to follow this complex methodology but instead have opted for a more straightforward approach to learning MAS and ML. As an application that includes MAS and ML integration developed with a similar perspective, Razek et al. [17] propose an educational application that discusses an ML approach, the ID3 Decision Tree Induction Algorithm, for analyzing and predicting learners' online learning styles. The aim is to enhance interaction by selecting appropriate learning content. The paper presents the Confidence Intelligent Tutoring System (CITS), an MAS designed to facilitate interactions in a community of learners. The CITS extracts knowledge about learners to adapt distance learning environments for more cooperative interactions. The approach is demonstrated to determine learning styles with 78% accuracy, offering an alternative to lengthy questionnaires for predicting learning styles.

Federated learning has become a new approach for training machine learning models in distributed systems. In this way, a centralized collective model is constructed from the contributions of all participating nodes. Similar to our perspective, Rincon et al. [18] address this issue by introducing an MAS that forms a flexible and dynamic federated learning framework, allowing for the easy addition of nodes to the system, namely FlaMAS (Federated Learning Based on a SPADE MAS). This proposal has been experimented on in the SPADE platform with the well-known MNIST dataset. The goal here is to train the MNIST dataset via Convolutional Neural Networks (CNNs) and decrease the training time, as well as the number of epochs needed to obtain a model with an accuracy higher than 90%. In order to ascertain the potential reduction of these two variables, they made the decision to distribute the identical dataset among three SPADE Client agents. The fourth SPADE agent served as a Server agent responsible for receiving messages from the three agents. The responsibility of a Server agent is to compute the average of the weights, which represent the network parameters responsible for transforming the input data within the network's hidden layers. These weight parameters are transmitted by each of the Client agents at the conclusion of a training epoch. Each of the following experiments employs the same network as the first one, maintaining the majority of the hyperparameters consistent with the previous experiment. However, there is one notable exception: the number of epochs. In the next experiment, two new hyperparameters are introduced. These novel hyperparameters include the number of local epochs (N-local epochs) and the number of global epochs (N-global epochs). N-local epochs represent the count of training epochs performed by each agent, while N-global epochs denote the number of global iterations executed by the central agent before halting the training process. The parameter N-global epochs specifically pertains to the centralized learning epochs. Once the system is trained in this way, the best models obtained are stored for evaluation.

MASs and ML are two popular research areas that might be applied for industrial purposes. Hanga and Kovalchuk [19] review the use of AI techniques, including MASs and ML, in the oil and gas industry. They analyze the factors contributing to the limited and gradual adoption of AI within the oil and gas sector. They provide recommendations for future AI research to promote wider integration of AI technologies in the oil and gas industry, where these sectors are improving gradually. Applying these topics in the power and energy industry is another hot topic that should be considered. McArthur et al. [20] mention a review of power engineering applications for which an MAS is being studied and identify technical challenges that need to be resolved for wider adoption in the energy sector. Ma et al. [21] conduct a scoping review of the literature regarding the use of ontology in MASs within the energy sector. The paper outlines five key aspects: the definition of agents and MASs, the application of an MAS in the energy field, defined ontologies in the energy domain, MAS design methodologies and architectures, and the role of ontology in

MAS development. Yao et al. [22] discuss the growing utilization of agent-based methods in energy system research, specifically focusing on two trends: multi-energy system transition simulation through agent-based modeling (ABM) and multi-energy system management via MAS modeling. The review primarily covers MAS application papers in various energy system domains, including building energy systems, district energy systems, and regional energy systems, considering aspects such as energy carriers, agent control architecture, optimization algorithms, and agent development environments.

Wind energy is a commonly used form of renewable energy today. As Turkey deploys numerous wind turbines across the country, the data collected by these turbines [23] become a valuable resource to develop an application for predicting power production. This paper specifically emphasizes this aspect from the ML perspective, similar to [24,25].

To explore the application of MAS and ML in the energy domain, Toquica et al. [26] implement an MAS environment where agent planning plays a crucial role in maintaining the dynamic equilibrium between energy demand and generation. Because planning is dependent on future demand, agents' anticipation of consumption becomes a vital element in the process. They develop procedures for automating specific tasks in a broad manner, catering to residential prosumers and consumers operating at the distribution level. This proposed approach leads to the creation of a Trading Environment (TE) configuration for multi-stage single-side auctions, offering practical utility for the management of future Smart Energy Markets. The paper conducts various experiments with different numbers of SPADE agents, involving both controllable and uncontrollable electric loads, to assess the system's performance in household settings. Power generation is predicted using input variables such as time of day, solar irradiance, wind speed, wind direction, cloud coverage, and external temperature. The prediction model is trained with a feed-forward neural network with 100 neurons in five layers, employing a hyperbolic tangent as the activation function. Additionally, the consumption of controllable loads is forecasted via Support Vector Machine (SVM) with a radial basis function as the kernel.

As explained in previous papers, ML and MAS applications have been separately implemented in various domains. It is assumed that addressing these topics in an educational context represents an achievement in teaching students. Hence, this paper is organized around this main idea.

In this paper, the goal is to develop an ML application using the shared data in an MAS environment. Wind turbines are commonly used in wind power production and are highly desirable for the sustainable development of energy solutions, as is widely known. The prediction of wind power production has been considered one of the most suitable areas for achieving this goal, and a dataset related to this field has been used. Sensors on wind turbines can measure wind speed and direction, allowing data to be collected. In this study, the focus is on the step-by-step development of an application that predicts the power that can be generated based on these data. For this purpose, the operation of an MAS is first explained, and two different MAS environments, SPADE [7–9] and Redis [10] (as a messaging framework like an MAS), are used to develop agents. The use of two different methods aims to compare the study, identify the pros and cons of these environments, and simulate the project in a realistic manner to determine the most suitable method.

It is an objective of this study to provide students with some challenges related to software agents and ML-based tasks during education. When developing a system, the components are independently implemented, whereas the integrated execution of them is so critical. Hence, the educational perspective in computer science must be given to students for the integration of various research areas. This paper focuses on achieving integration through various methods using software agents, ML, and teaching students to address practical challenges. Theoretical knowledge and practical implementation of the federated learning approach were assessed by the students and the results demonstrate that this type of AI education helps students comprehend the subjects more easily.

This study meets the Sustainable Development Goals (SDGs) of the United Nations in two aspects, namely for Goal 4—"Quality education"—and Goal 7—"Affordable and clean

energy". To address these aspects, this study initially covered the theoretical concepts of software agents and ML to establish a shared understanding of these subjects (the concept of software agents, common ML algorithms, etc.). Then, students implemented these concepts in three phases. The first phase involves deploying the software agent frameworks (SPADE and Redis) and gaining an understanding of their advantages and disadvantages from the communicational, executional, and performance perspectives. Secondly, the wind turbine dataset is split into multiple columns, aligning with the number of sensors to capture distinct sets of sensing data elements, each of which is conveyed through a software agent. A Manager Agent gathers the data from these sensing agents and consolidates them as a central point for utilization in the ML process. This phase assists students in learning how an MAS works in a real problem domain. The third and last phase adds a new agent, called an ML agent, to the system and this agent develops ML models through the data transferred from the Manager Agent and makes a prediction about the power production. These experiments provide a suitable solution for connecting the theoretical knowledge gained from the course with real-time application.

The contributions of this paper are as follows:

1. Theoretical knowledge and practical experiences were combined in an AI course by handling various topics, such as software agents and ML, at the same time.
2. The implementation of the project via various ML algorithms assisted us in obtaining a simulation environment to predict the active power production of wind turbines. Additionally, it paved the way for the application of this project in a sensor-based environment as a digital twin of a wind turbine.
3. The paper confirms that the predicted wind turbine active power production closely aligns with the validation and test data values. These results increase sustainability awareness by demonstrating that this system can successfully function as a simulation.
4. A course evaluation was conducted to assess the performance of this integrated AI learning experience. The general idea of the students is positive regarding the educational approach of the course.

The remainder of this paper is organized as follows: Section 2, titled "Methodology", presents the project implementation phases step by step, demonstrating the outputs as well as highlighting certain challenges faced during the project execution. Section 3, "Evaluation of the Educational Content and Discussion", encompasses both student feedback and a discussion of these insights which aimed to enhance the contribution of the course to students in the future. Finally, Section 4, "Conclusions", wraps up the outlined procedures in the paper and delves into forthcoming projects, building upon wind turbine power prediction within an MAS environment.

2. Methodology

The incorporation of MAS and ML technology has considerably improved the sustainability of wind turbine power production. Wind farms can maximize energy output while avoiding wear and tear by using MASs to coordinate their actions and optimize the operation of individual turbines. ML algorithms are employed to continuously analyze vast amounts of data, including weather patterns, turbine performance, and grid demand, enabling predictive maintenance and real-time adjustments for optimal power production. The relationship between MAS and ML contributes to a more sustainable and environmentally friendly approach to renewable energy generation.

Before developing the application for predicting wind turbine power generation, the theoretical part of the course was completed. It initially begins with an explanation of the MAS concept, rational agents, environment types, and agent types, following [1]. Software agents share common properties such as autonomy, cooperation, and learning [1]. During the theoretical session, these properties were explained in detail, with a connection made to different agent types. The chosen project aimed to apply these properties to gain a better understanding of the execution mechanism of software agents and MASs.

The methodology of this paper is structured into three phases, guiding the application of MASs and ML and systematically teaching them through the development of an application. This project is actually simulating the behavior of wind energy harvesting, even though it could be applied in the same domain in a real environment. Therefore, I plan to conduct research that applies this kind of scenario in an IoT environment and aims to follow a similar approach for using the application in a sensor-based agent environment in the future. Palanca et al. [27] discuss the application of MAS technology in the context of IoT. They introduce the concept of the IoT artifact as a new interface abstraction for developing MASs based on IoT devices and adopt the programming model of the SPADE MAS platform, providing both a theoretical framework and a practical model for real-world applications. Following the idea behind [27], this work aimed to develop an MAS using the SPADE platform. The SPADE instant messaging model benefits from XMPP standards, and as such, the project in this paper is implemented via XMPP, utilizing an online XMPP server [28].

2.1. Phase#1: Message Communication

The first phase introduces MASs and involves the development of a message transfer application. In this application, the objective is to transmit a message from one agent to multiple agents that can receive it instantly. Initially, the idea was to transmit messages via publish–subscribe (PubSub) protocol in the SPADE platform. Publish–subscribe is a messaging pattern in which message senders, known as 'publishers', do not designate specific recipients, known as 'subscribers', for their messages. Instead, they classify these messages into categories without knowing which subscribers, if any, will receive them. Conversely, subscribers indicate their interest in one or more categories and only receive messages that match their interests, without knowing which publishers, if any, are responsible for those messages. In this phase of the project, the objective is to transmit information regarding various wind turbine features such as wind speed, theoretical power curve, and wind direction [23]. Each of these sensed features is sent to a central agent, which collects and analyzes the data to make predictions. Initially, the project was designed to be implemented using PubSub [8]. However, the students encountered difficulties in executing the agents via PubSub within the SPADE platform, and managing message synchronization among the agents proved challenging. As a result, I made the decision to change the messaging mechanism by having the feature-oriented agents send messages directly to the central agent.

There are multiple behaviors for message communication in SPADE for various purposes. In this phase, the goal is just to send a message from one agent to another agent; that is why OneShotBehavior is used for SenderAgent. Two messages are sent one time and the messaging process is finished. PeriodicBehavior makes it continue messaging until a stopping condition. PeriodicBehavior is mostly suitable for message sending, whereas the receiver side handles these messages via CyclicBehavior [7].

You will find the relevant code for a simplified messaging approach in Appendix A. This code fragment is a straightforward example of sending a message from a sender, with multiple subscribers capable of reading the transmitted message from the sender agent. The educational objective of this assignment is to emphasize the concept of one sender with multiple receivers. Subsequently, the next phase will shift the focus to one receiver with multiple senders to demonstrate the practical implementation of the system. The design aims to provide comprehensive learning experiences for each situation by exposing students to various scenarios in MASs. The monitoring of the messages can also be tracked through the online XMPP server, and the successful completion of the messaging process can be verified. The same operations as those in this assignment are implemented using Redis. The final version of the Redis implementation will be presented in Section 2.3.

2.2. Phase#2: Implementation of the MAS Protocol

The second phase is the core of the MAS messaging platform for wind turbine active power prediction. The goal here is wind turbine monitoring using an MAS. There are three publisher agents and one subscriber agent. The data in [23] are split into these agents. The inputs of the system are streamed by publisher agents. The simulation assumes that these values are gathered from each respective sensor and transmitted to a central point. The subscriber agent processes the data in this stream and outputs them to a file. The ultimate output of this phase is the same dataset as that presented in [23], which is stored in the central agent's storage. Therefore, the simulation periodically streams data from the wind turbine. The data, originally given for a 10 min period in [23], are streamed at a 1 s interval for simulation purposes. The objective of this phase is to enhance the understanding of agent communication within various scenarios, specifically focusing on multiple senders to one receiver, while the first phase of this assignment deals with the scenario of one sender and multiple receivers. Appendix B pertains only to the WindDirectionAgent, intended for simulation purposes. The implementation for other features follows a similar pattern. Agents associated with different features transmit messages to the receiver using the XMPP protocol. Initially, the server account name must be specified within the message. The PeriodicBehavior facilitates the transmission of messages at regular intervals until all data collected from the sensor (the dataset file) are exhausted. The *self_counter* variable in the ReceiverAgent represents the number of lines read for that particular feature. For the sake of simplification, *self_counter* is set to 25, which means only 25 lines of elements are read, even though the complete dataset is utilized in the application. The ReceiverAgent listens to each agent, and all the data collected from them are combined and written to an output file in the central agent.

2.3. Phase#3: Integration of the MAS and ML

The implementation of this system was conducted via SPADE and Redis by different students to capture the challenges in various MASs. The validation of the model in the SPADE version, as depicted in Figure 1, is considered. However, in the Redis version, which aims to partition the data across various configurations among different agents based on the number of ML models, the validation operation is omitted. Instead, the models are directly tested for educational purposes, as shown in Figure 2.

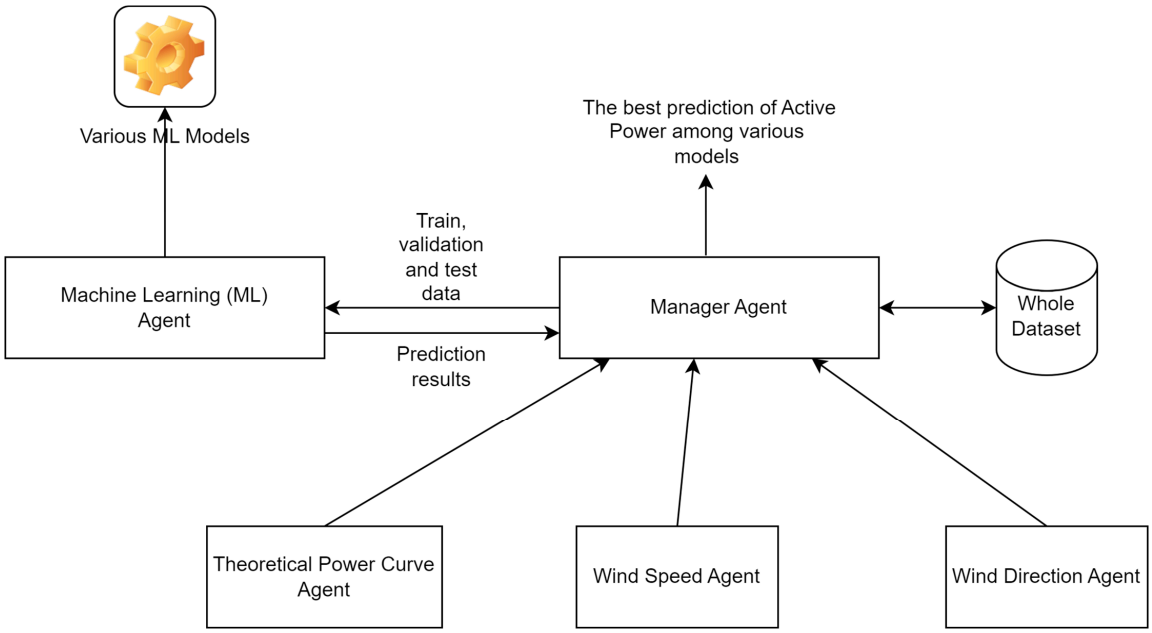

Figure 1. The general architecture of a sustainable MAS (SPADE version).

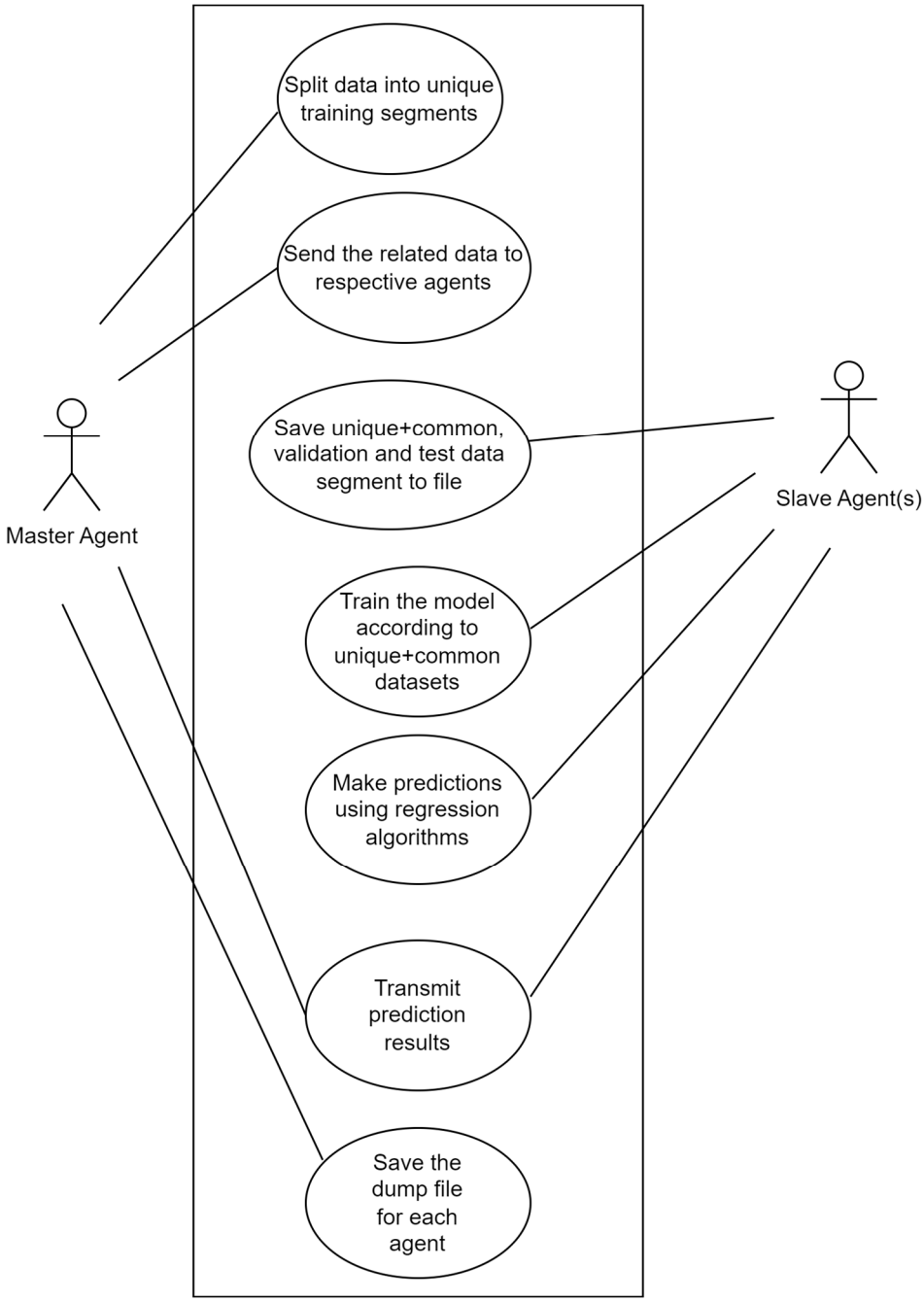

Figure 2. The use case diagram of a sustainable MAS (Redis version).

The final phase of the project involves the application of regression algorithms to create a wind turbine active power prediction model within an MAS. The three inputs (Wind Speed, Wind Direction, and Theoretical Power Curve) are collected in Manager Agent and the prediction value of the Manager Agent (the central agent is now referred to as the Manager Agent from this phase because the entire process is managed by this agent) is the Active Power that is produced by the wind turbine. As explained in Section 2.2, there are three agents for data gathering and one agent for management. From now on, all the data are kept in a common data storage area where each ML agent can access them for training data that are split by the Manager Agent. The streaming of training data is a process that affects the overall communication performance of the system; however, it is thought to be unnecessary to measure this performance where it is not the focus of this paper.

Initially, the wind turbine dataset is stored in the local storage of the Manager Agent. The dataset is split into three sets: 80% for training, 10% for validation, and 10% for testing. A new agent, called the ML Agent, is introduced to train the model and validate it using the validation dataset.

The ML Agent(s) has two primary objectives: reporting the performance of the regression algorithms and responding to predictions of Active Power using the required features (Theoretical Power Curve, Wind Speed, Wind Direction). Reports on Active Power values are provided for the comparison of various regression algorithms, using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics.

For each algorithm, a new ML agent can be used, or just one agent can generate all ML models. For educational purposes and to struggle with the challenges in a more complex MAS environment, one agent is used in the SPADE version of the system, while the Redis version assigned each model to various agents.

The ML agent directly accesses the train and validation datasets from a local file and trains various regression models. MAE and RMSE values are taken into consideration, and the best model is selected for the prediction process. While the selection of the best model by LGBMRegressor (LightGBM algorithm) is detailed in Figures 3 and 4, the performance of the chosen model has not been assessed in the SPADE version, where a traditional approach in ML is experimented with. Then, the ML Agent initially receives the 10% test dataset streamed from the Manager Agent after generating the ML models. The model produced by the LGBMRegressor makes predictions as the data are streamed and sends the predicted Active Power values back to the Manager Agent. When the data streaming stops (e.g., no new data received for 5 s), the Manager Agent calculates the MAE and RMSE by comparing actual and predicted Active Power values, and then displays these values on the screen. In a real-time environment, the Manager Agent will respond with just the predicted Active Power value. However, the MAE and RMSE outputs from Manager Agent were requested by the students to gain a broader familiarity with ML concepts. The overall communication architecture of the MAS is shown in Figure 1.

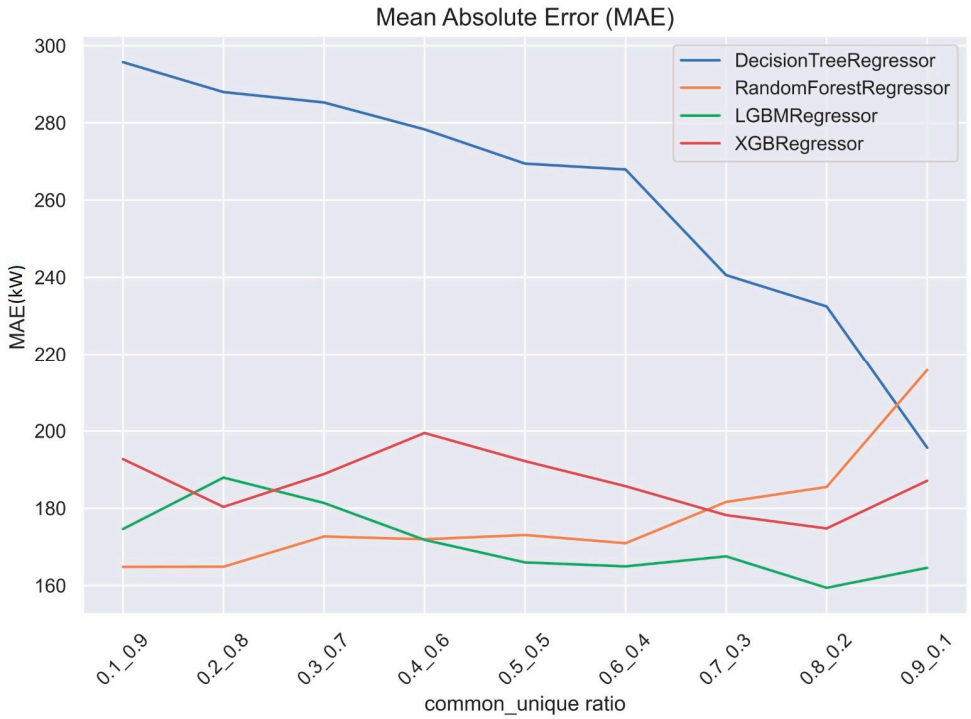

Figure 3. The MAE values training the model with various algorithms in separate agents to predict generated Active Power (kW) with various common_unique training data ratios.

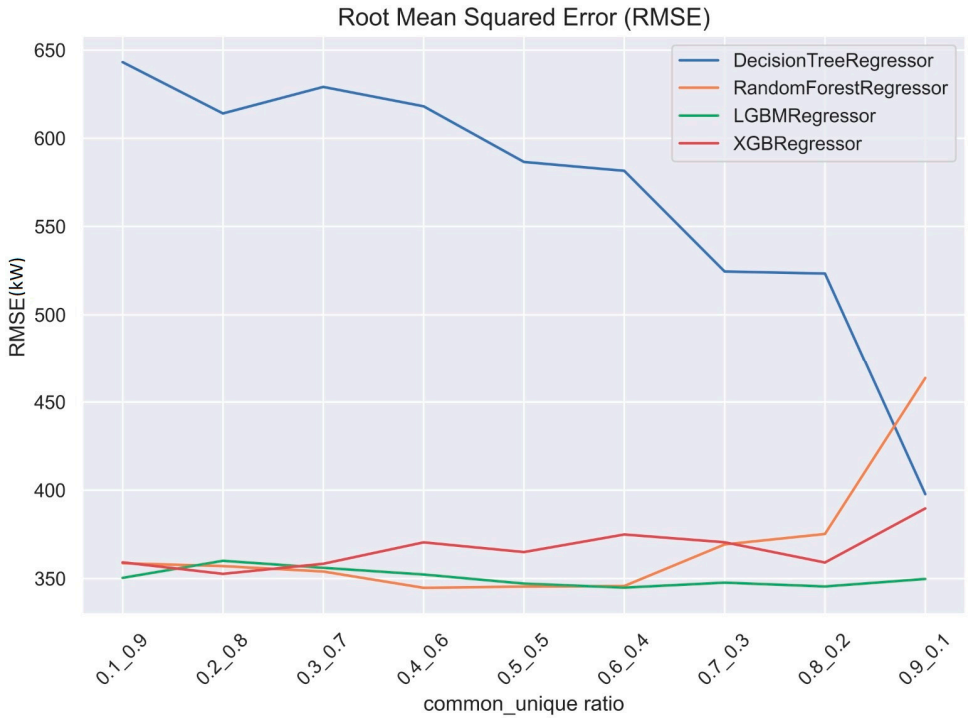

Figure 4. The RMSE values training the model with various algorithms in separate agents to predict generated Active Power (kW) with various common_unique training data ratios.

This setup of the MAS is designed primarily for educational purposes. In a real-time environment, dense communication among all these agents is not always necessary. In practice, the decision-making process among the ML models and the execution of these models after the training process in the ML agent could be handled by the Manager Agent. The goal of this setup is to emphasize and enhance agent communication, even though it may lead to increased communication costs.

Redis version of the system behaves differently, as illustrated in Figure 2. Redis version involves Master and Slave agents capable of communicating with each other. In this setup, the additional objective is to partition the training dataset into unique segments and train various models using these datasets. The data are transmitted to each Slave Agent with a unique portion, while the remaining data are shared for common usage. Consequently, a common dataset is consumed within each Slave Agent, with a separate, unique dataset segment handled by each individual Slave Agent.

The behavior of the Master Agent is similar to that of the central agent depicted in Figure 1, but in Redis version, Slave Agents are associated with ML algorithms rather than specific features (Theoretical Power Curve, Wind Speed, Wind Direction). Consequently, the focus of communication shifts towards the models, in contrast to the communication architecture centered around the sensed features of the wind turbine in Figure 1. Initially, the Master Agent splits the data into training, validation, and test sets. However, a distinct approach is developed for the training set, which is divided into common and unique datasets based on defined and adjustable split ratios. This configuration facilitates the comparison of model performance across different data sizes. In this setup, the Master Agent stores the complete dataset and sends the feature values to each Slave Agent asynchronously. The Slave Agents receive both common and unique training data records from the Master Agent and store these agent-specific data in their local storage. Subsequently, the Slave Agents individually train the model using the assigned regression algorithms and agent-specific data. As the test data are initially distributed to all Slave Agents, they can make predictions and transmit the prediction results, along with the MAE and RMSE metrics, which can be easily calculated by processing the prediction and test results, back to

the Master Agent. The Master Agent saves the metric values for each Slave Agent in dump files, which can be analyzed to compare performance results across various configurations.

As indicated earlier, MAE and RMSE metrics are used to evaluate the performances of the models for predicting active power generation of wind turbines. MAE, an evaluation metric applied to continuous variables, quantifies the prediction error by calculating the difference between the actual value and the predicted value for a given instance. MAE measures the average magnitude of all absolute errors and is computed using the following formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{1}$$

where n represents the number of data points, y_i is the true value, and x_i is the predicted value. Unlike MAE, RMSE involves squaring the difference between true and predicted observations and, subsequently, computing the average of these differences. RMSE formula is

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \tag{2}$$

While MAE provides an overall sense of the average error magnitude, RMSE offers insights into how much individual errors deviate from the mean. This becomes especially crucial in scenarios where large errors are particularly undesirable. In the initialization of a real-time scenario for the wind turbine power prediction simulation, the interpretability of large errors is significant, not only considering the average error magnitude. Therefore, these metrics jointly serve as the main success criteria for application from both MAS and ML perspectives where each ML model is trained in separate agents.

The outputs of different configurations by using DecisionTreeRegressor, RandomForestRegressor, XGBRegressor, and LGBMRegressor are shown in Figure 3 for MAE and in Figure 4 for RMSE results.

In Figures 3 and 4, common data signifies that all agents share the same data elements for training, while unique data denotes the specific portion of the dataset assigned uniquely to each agent. The formula indicating the number of data points in each agent is provided below:

$$\text{d} = \text{w}*\text{cr} + \text{w}*((1 - \text{cr})/\#\text{ of agents}) \tag{3}$$

where d represents the total number of data points in each agent, w is the entire dataset, and cr is the common ratio (the ratio of common data to entire data). An increase in the unique ratio (the ratio of unique data to entire data) results in the multiplication of agent number by unique data points, leading to a decrease in the total number of data points in each agent. The alteration of the common and unique data ratio in the training dataset significantly influences the results of each model. While an increase in the common ratio increases the overall training data for each agent and an increase in the unique ratio causes a reduction in the total number of data elements in the training set.

From Figures 3 and 4, it is seen that boosting methods such as XGBoost and LightGBM outperform Decision Tree and Random Forest regressors. Figure 4 illustrates that an increase in data points in the training dataset leads to a decrease in the RMSE value for DecisionTreeRegressor. This suggests that the augmentation of training data gradually influences performance, with the common data ratio having a more substantial impact than unique data. LGBMRegressor (LightGBM) stands out by yielding superior results in terms of both MAE and RMSE values, showing robustness across varying data ratios. Using a gradient-based learning approach for decision tree construction, LightGBM adopts a leaf-wise tree growth strategy rather than the conventional level-wise approach. Consequently, LightGBM prioritizes nodes contributing the most to the reduction of the loss value during training, resulting in faster convergence and potentially more accurate models. The faster convergence property also enhances the model's robustness across different dataset split ratios.

3. Evaluation of the Educational Content and Discussion

At the end of the semester, an electronic survey was conducted to objectively evaluate the impact of the educational program on the learning process mentioned above and gain concrete insights into student satisfaction. There were 27 students attending this survey. The survey is categorized into "Learning Value", "Value Added", "Design and Other Issues", and the important open-ended questions about taking the perspective of the students follows the survey perspective of [29]. "Learning Value", "Value Added", and "Design and Other Issues" are created as a five-level Likert-type scale survey, ranging from 'strongly disagree' to 'strongly agree'.

"Learning Value" questions involve inquiries designed to examine students' views on the effectiveness of the educational program of AI, especially teaching MASs and ML. "Value Added" aims to evaluate the capacity for learning and preparing similar systems, as well as the extent to which this educational content contributes to that capability. "Design and Other Issues" is related to the students' assessment of the course's clarity and their level of engagement with the course. The open-ended questions assist students in expressing their thoughts about the course openly and anonymously. Table 1 represents the questions in the survey.

Table 1. The students survey questions on the Likert scale.

No.	Question
	Learning Value
Q1	Phase#1 enhanced my ability to understand messaging in MAS.
Q2	Phase#1 enhanced my ability to understand how MASs work.
Q3	Phase#1 enhanced my ability to run a MAS.
Q4	Phase#2 enhanced my ability to apply the messaging protocol in a real-world scenario of a MAS.
Q5	Phase#2 enhanced my ability to understand how MAS work in a real-world scenario.
Q6	Phase#3 enhanced my ability to understand traditional ML algorithms.
Q7	Phase#3 enhanced my ability to understand the evaluation metrics in ML.
Q8	Phase#3 enhanced my ability to use MAS and ML in combination.
	Value Added
Q9	The plan of the projects helped me to integrate different technologies into working systems solutions.
Q10	I can learn similar topics in this way without such a project plan.
	Design and Other Issues
Q11	The ideas and concepts within the project phases were presented clearly and easy to follow.
Q12	The time allocated to the course was enough to learn each concept in the projects comprehensively.
Q13	I was interested in MAS before I started to study it.
Q14	I was interested in ML before I started to study it.
Q15	I am interested in this topic after having studied MAS.
Q16	I am interested in this topic after having studied ML.
Q17	I recommend this kind of project based educational strategy to a fellow student.

"Learning Value" is assessed phase by phase to gauge students' comprehension of each assignment's perspective. For instance, in Phase#1, there were challenges in implementing message communication in SPADE, leading to certain adjustments in the plans for students. In Phase#3, the integration of MASs and ML encourages students to develop real-world applications. Hence, understanding the concepts behind each phase is crucial for future educational decisions. "Value Added" evaluates the enhancement of students' engineering capabilities in solving unforeseen problems using optimal procedures. Student feedback is essential for maximizing the development of this skill within the realm of engineering. "Design and Other Issues" gathers student feedback to provide insights into MASs and ML expertise, and to assess whether the course has aroused student interest in these research areas.

The "Learning Value" section of the survey reveals that the majority of students either agree or strongly agree that they followed the project instructions and comprehended the learning objectives of the course as shown in Figure 5. However, there are some

dissatisfactions evident in Q6, stemming from the fact that the ML course is separate, preventing students from obtaining a more comprehensive knowledge of ML. In the "Value Added" section of the survey, it is clearly seen that the integration of various research areas is appealing to students, as seen in Figure 6. The problematic point is that the students do not have full confidence in themselves, as indicated in Q10. This suggests courses of this kind should be integrated into the educational curriculum from the early years. In the "Design and Other Issues" section, it is demonstrated in Figure 7 that the students are highly satisfied with the course and would recommend it to their colleagues for the next year (Q17). However, Q13 and Q15 indicate that MASs may not be very appealing to some students, possibly due to their academic nature and limited application in companies in Turkey. Overall, it appears that the course content and teaching methodology have left a positive impression on students. However, the criticisms mentioned here should be taken into consideration for future courses.

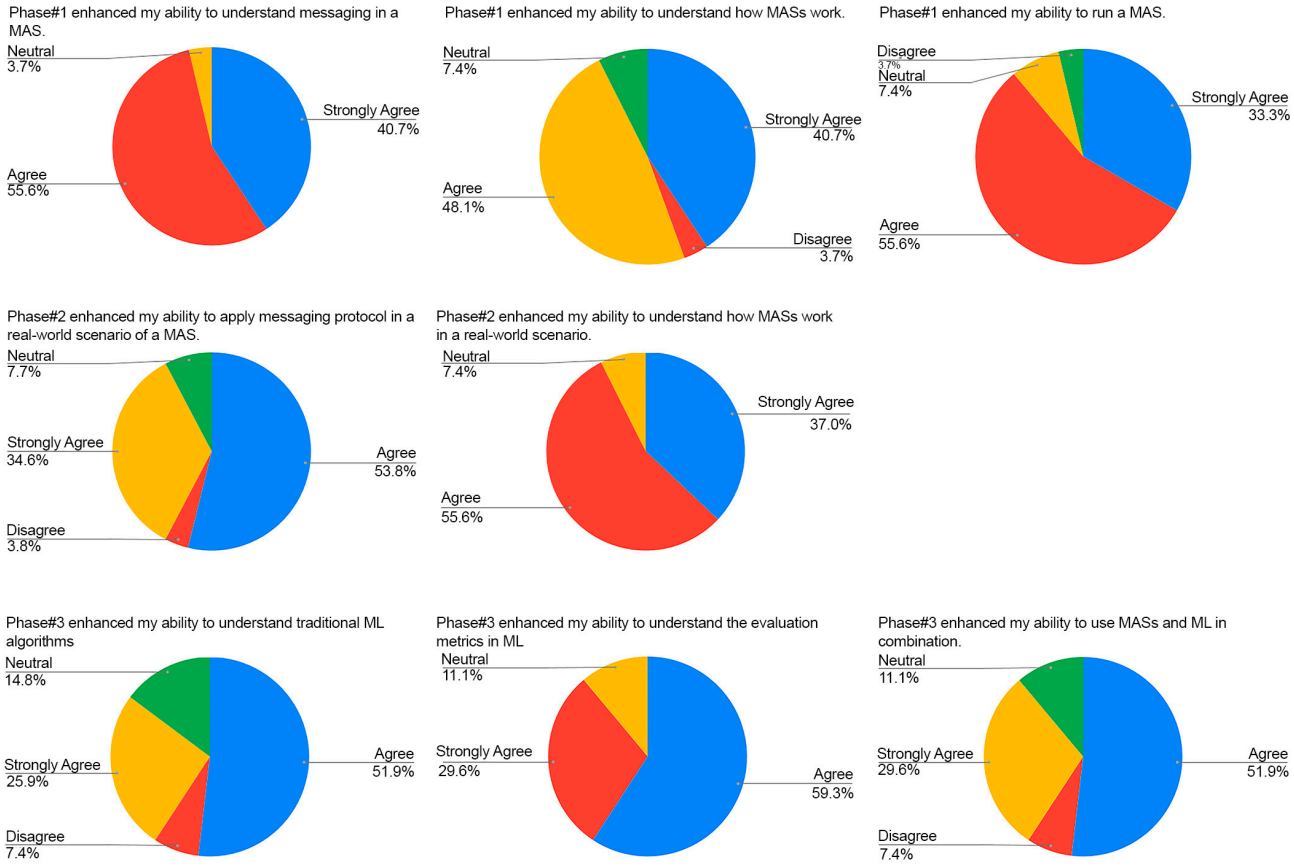

Figure 5. Student answers for the "Learning Value" part of the survey.

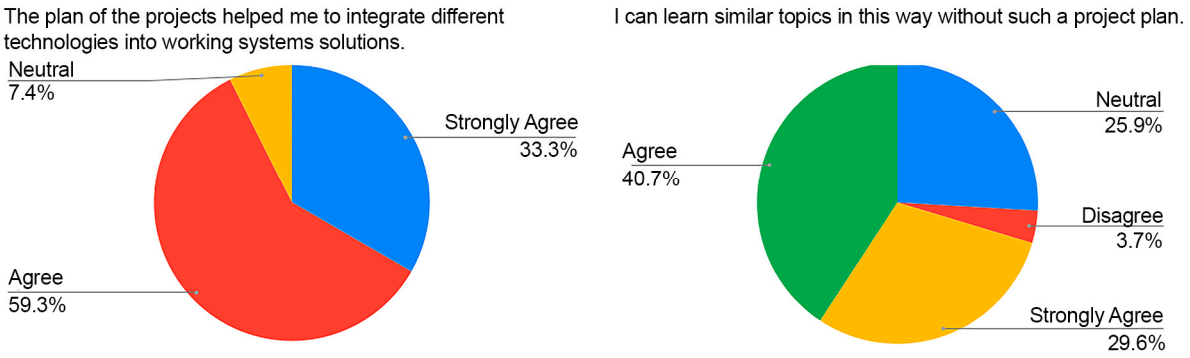

Figure 6. Student answers for the "Value Added" part of the survey.

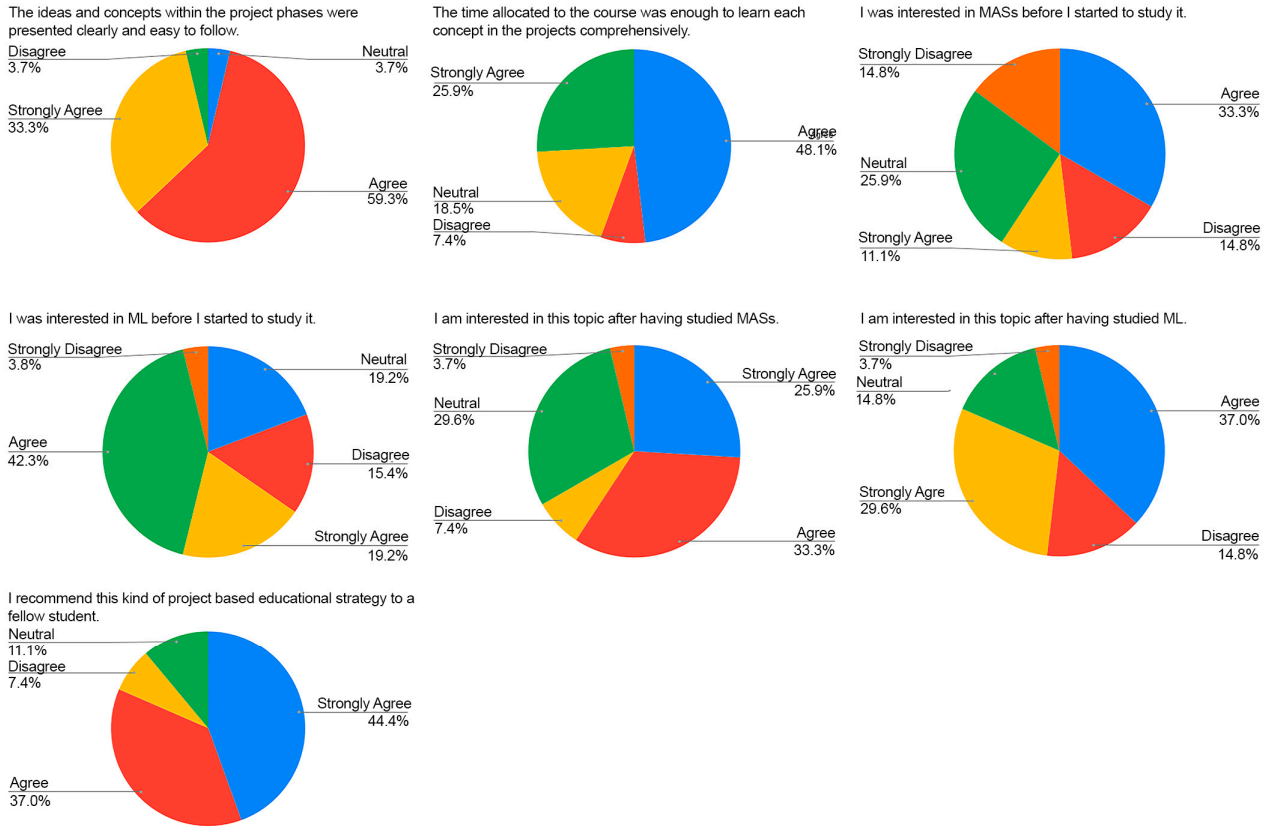

Figure 7. Student answers for the "Design and Other Issues" part of the survey.

In Table 2, the common answer related to Q18 pertains to understanding the functioning of the MAS and its integration into real-time applications. The essence of responses to Q19 is that the course content assists students in comprehending message communication and raises their awareness of how theoretical knowledge can translate into practical experience. Comments regarding Q20 largely revolve around the suggestion to conduct additional laboratory sessions for each specific phase. However, during that particular semester, the course was conducted online, making it challenging to plan lab sessions. Nevertheless, recommendations suggest weekly practical guidance would be beneficial in motivating students.

Table 2. Open-ended student survey questions.

No.	Question
Q18	What were the most useful or valuable parts of these term projects?
Q19	What were the most useful or valuable parts of this course?
Q20	How do you think this course could be improved?

As seen in Table 3, there are various research studies from both educational perspectives and industrial applications of MASs and ML. However, there is a lack of research addressing the integration of all these areas, which is proposed in this paper. When proposing a combination of these areas, the implementation of the MAS architecture was carried out using two different MAS architectures, and the challenges/variations in different phases were explained in Section 2.

Table 3. The comparison of research in MASs and ML from an educational perspective and in the energy domain.

Reference	The Educational Perspective of AI	Integrated MAS and ML Work	MASs in Energy Domain	ML in Energy Domain
[3]	Yes	No	No	No
[4]	Yes	No	No	No
[5]	Yes	No	No	No
[15]	No	Yes	No	No
[16]	No	Yes	No	No
[17]	No	Yes	No	No
[18]	No	Yes	No	No
[20]	No	No	Yes	No
[21]	No	No	Yes	No
[22]	No	No	Yes	No
[23]	No	No	No	Yes
[24]	No	No	No	Yes
[25]	No	No	No	Yes
[26]	No	Yes	Yes	Yes
Proposed Method	Yes	Yes	Yes	Yes

4. Conclusions

Decision-makers can benefit from MASs and ML models for sustainable development in energy systems. There are two objectives in this manuscript following considering these systems. The first objective of this paper is to integrate theoretical knowledge and practical experience related to both MASs and ML during an AI course. The second objective is to create a simulation of a sustainable environment by implementing MASs and ML for wind turbines. Hence, a practical scenario involving the prediction of wind turbine active power generation is adopted to teach the concepts within these research domains. The simulation of a wind turbine was accomplished using SPADE and Redis platforms, employing various MAS architectures. The efficacy of this step-by-step teaching approach was assessed by the students. The implementation of the project yields educational and practical outcomes that can assist in the development of a more practical educational plan and the simulation of a real-time wind power generation system.

The proposed solution is implemented in various agent architectures with different perspectives by adjusting the settings of training datasets, using common and unique parts of the datasets to evaluate the models' performance. These adjustments provide insights into the students' understanding of challenges in various scenarios and highlight performance variations. The observed performance improvements in boosting methods, such as LightGBM, indicate that these methods yield better results when the settings are modified.

The successful outcomes of the experiments demonstrate that IoT devices can be integrated into MAS with flexibility and simplicity. Consequently, a unified architecture that combines an MAS and ML can be implemented in a dynamic real-time environment, accommodating changing parameters such as various sensors or ML algorithms. Ongoing data collection from specific sensors plays a pivotal role in augmenting our knowledge of wind turbine lifecycles. These data also provide valuable insights into factors like maintenance requirements and the early detection of sensor malfunctions, particularly when predicted values deviate from the actual data. Such information is essential for ensuring the long-term sustainability and reliability of wind turbine infrastructure.

The future direction of this paper involves implementing this system with real-time sensors for each agent. In this configuration, the system will function based on the regression model deployed on the Manager Agent, effectively generating a digital twin of a wind turbine. Consequently, the wind turbine simulation will operate as a "Phantom IoT" twin through an MAS and ML, enabling feasible power generation predictions in different windy regions of Turkey.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. The Implementation of Messaging among Agents in SPADE

```
class SenderAgent(Agent):
    class InformBehav(OneShotBehaviour):
        async def run(self):
            print("InformBehav running")
            msg1 = Message(to = "subs1@anonym.im")
            msg2 = Message(to = "subs2@anonym.im") # Instantiate the message

            msg1.set_metadata("performative", "inform")
            msg2.set_metadata("performative", "inform")# Set the "inform" FIPA performative

            msg1.body = sentences_tokens[0] + " " + sentences_tokens[1]
            msg2.body = sentences_tokens[2] + " " + sentences_tokens[3] # Set the message content

            await self.send(msg1)
            print("Message 1 sent!")
            await self.send(msg2)
            print("Message 2 sent!")

            # stop agent from behaviour
            await self.agent.stop()
    async def setup(self):
        print("SenderAgent started")
        b = self.InformBehav()
        self.add_behaviour(b)

class ReceiverAgent(Agent):
    class RecvBehav(OneShotBehaviour):
     async def run(self):
        print("SubsAgent running")

        msg = await self.receive(timeout = 10) # wait for a message for 10 s
        if msg:
            print("Message received with content: {}".format(msg.body))
        else:
print("Did not received any message after 10 s 1")

        # stop agent from behaviour
        await self.agent.stop()

    async def setup(self):
      print("ReceiverAgent started")
      b = self.RecvBehav()
      template = Template()
      template.set_metadata("performative", "inform")
      self.add_behaviour(b, template)
```

Appendix B. The Implementation of Wind Turbine Sensed Data Fusion among Agents in SPADE (Code Fragment related to Wind Direction Agent)

```
class WindDirectionAgent(Agent):
    class InformBehav(PeriodicBehaviour):
     async def run(self):
       msg = Message(to="subs1@anonym.im")
       msg.body = str(wind_direction[self.counter][0]) + "+" + str(date_time[self.counter][0])
       await self.send(msg)
       print("Wind Direction message sent!")

       if self.counter == 25:
          self.kill()
       self.counter += 1

    async def on_end(self):
       # stop agent from behaviour
       await self.agent.stop()

    async def on_start(self):
       self.counter = 0

    async def setup(self):
       print(f"WindDirectionAgent started at {datetime.datetime.now().time()}")
       start_at = datetime.datetime.now() + datetime.timedelta(seconds = 5)
       b = self.InformBehav(period = 1, start_at = start_at)
      self.add_behaviour(b)

class ReceiverAgent(Agent):
    class RecvBehav(CyclicBehaviour):
     async def run(self):
       msg = await self.receive(timeout = 10) # wait for a message for 10 s
       total_msg = msg.body
       msg_parts = total_msg.split("+")
       if msg:
       if str(msg.sender) == "theoreticalpowercurve@anonym.im":
          self.x_tpc = msg_parts[0]
          self.x_tpc_dt = msg_parts[1]
       elif str(msg.sender) == "windspeed@anonym.im":
          self.x_ws = msg_parts[0]
          self.x_ws_dt = msg_parts[1]
       elif str(msg.sender) == "winddirection@anonym.im":
          self.x_wd = msg_parts[0]
          self.x_wd_dt = msg_parts[1]
       if self.x_tpc_dt == self.x_ws_dt and self.x_ws_dt == self.x_wd_dt:
       self.data_received = self.data_received.append({"DateTime": self.x_tpc_dt, "Active
       Power": active_power[self.counter][0], "Theoretical Power Curve": self.x_tpc,
                  "Wind Speed": self.x_ws,
                   "Wind Direction": self.x_wd}, ignore_index = True)
      self.counter += 1
       else:
       if self.counter == 25:
          print(self.data_received)
          self.data_received.to_csv("data_received.csv")
          self.kill()
          quit()
```

```
async def on_start(self):
        self.data_received = pd.DataFrame(columns=[“DateTime”, “Active Power”, “Wind
        Speed”,”Theoretical Power Curve”, “Wind Direction”])
        self.x_tpc = None//Theoretical Power Curve
        self.x_ws = None//Wind Speed
        self.x_wd = None//Wind Direction
        self.x_tpc_dt = “”
        self.x_ws_dt = “”
        self.x_wd_dt = “”
        self.counter = 0

    async def on_end(self):
        await self.agent.stop()

    async def setup(self):
     print(“ReceiverAgent started”)
     b = self.RecvBehav()
     self.add_behaviour(b)
```

References

1. Russell, S.J. *Artificial Intelligence a Modern Approach (AIMA)*, 3rd ed.; Pearson Education, Inc.: Upper Saddle River, NJ, USA, 2010.
2. Schools Worldwide That Have Adopted AIMA. Available online: https://aima.cs.berkeley.edu/adoptions.html (accessed on 6 October 2023).
3. Kong, S.-C.; Man-Yin Cheung, W.; Zhang, G. Evaluation of an artificial intelligence literacy course for university students with diverse study backgrounds. *Comput. Educ. Artif. Intell.* **2021**, *2*, 100026. [CrossRef]
4. Ergezer, M.; Kucharski, B.; Carpenter, A. Curriculum design for a multidisciplinary embedded artificial intelligence course: (abstract only). In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, 21–24 February 2018; Association for Computing Machinery: New York, NY, USA, 2018; p. 1087.
5. Palanca Cámara, J.; Jordán, J.; Julian Inglada, V.J. Using learning by doing methodology for teaching multi-agent systems. In Proceedings of the 15th International Technology, Education and Development Conference, INTED2021 Proceedings, Online Conference, 8–9 March 2021; pp. 3866–3871.
6. Bellifemine, F.; Bergenti, F.; Caire, G.; Poggi, A. JADE—A java agent development framework. In *Multi-Agent Programming: Languages, Platforms and Applications*; Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A., Eds.; Multiagent Systems, Artificial Societies, and Simulated Organizations; Springer: Boston, MA, USA, 2005; pp. 125–147.
7. Palanca, J.; Terrasa, A.; Julian, V.; Carrascosa, C. Spade 3: Supporting the new generation of multi-agent systems. *IEEE Access* **2020**, *8*, 182537–182549. [CrossRef]
8. SPADE. Available online: https://spade-mas.readthedocs.io/en/latest/readme.html (accessed on 6 October 2023).
9. Palanca, J.; Rincon, J.A.; Carrascosa, C.; Julian, V.; Terrasa, A. A flexible agent architecture in SPADE. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection, Proceedings of the 20th International Conference, PAAMS 2022, L'Aquila, Italy, 13–15 July 2022, Proceedings*; Dignum, F., Mathieu, P., Corchado, J.M., De La Prieta, F., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 320–331.
10. Gutierrez, F. Messaging with Redis. In *Spring Boot Messaging: Messaging APIs for Enterprise and Integration Solutions*; Gutierrez, F., Ed.; Apress: Berkeley, CA, USA, 2017; pp. 81–92.
11. Fipa ACL. Fipa ACL Message Structure Specification. Foundation for Intelligent Physical Agents. Available online: http://www.fipa.org/specs/fipa00061/SC00061G.html (accessed on 6 October 2023).
12. Extensible Messaging and Presence Protocol (XMPP): Core (No. rfc6120). Available online: https://xmpp.org/rfcs/rfc6120.html (accessed on 6 October 2023).
13. Redis. Available online: https://redis.com/solutions/use-cases/messaging/ (accessed on 6 October 2023).
14. Costantini, S.; De Gasperis, G.; Pitoni, V.; Salutari, A. DALI: A multi agent system framework for the web, Cognitive Robotic and Complex Event Processing. In Proceedings of the 18th Italian Conference on Theoretical Computer Science and the 32nd Italian Conference on Computational Logic co-located with the 2017 IEEE International Workshop on Measurements and Networking (2017 IEEE M&N), Naples, Italy, 26–28 September2017; pp. 286–300.
15. Grzonka, D.; Jakóbik, A.; Kołodziej, J.; Pllana, S. Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security. *Future Gener. Comput. Syst.* **2018**, *86*, 1106–1117. [CrossRef]
16. Drogoul, A.; Zucker, J.-D. Methodological Issues for Designing Multi-Agent Systems with Machine Learning Techniques: Capitalizing Experiences from the Robocup Challenge. Research Report, lip6.1998.041, LIP6, 1998. 1998. Available online: https://hal.science/hal-02547805/ (accessed on 5 October 2023).

17. Razek, M.A.; Frasson, C.; Kaltenbach, M. Using a machine learning approach to support an intelligent cooperative multi-agent system. In *Technologies de l'Information et de la Communication dans les Enseignements d'Ingénieurs et dans l'Industrie*; Institut National des Sciences Appliquées de Lyon: Villeurbanne, France, 2002; pp. 119–123.
18. Rincon, J.; Julian, V.; Carrascosa, C. FLaMAS: Federated learning based on a SPADE MAS. *Appl. Sci.* **2022**, *12*, 3701. [CrossRef]
19. Hanga, K.M.; Kovalchuk, Y. Machine learning and multi-agent systems in oil and gas industry applications: A survey. *Comput. Sci. Rev.* **2019**, *34*, 100191. [CrossRef]
20. McArthur, S.D.; Davidson, E.M.; Catterson, V.M.; Dimeas, A.L.; Hatziargyriou, N.D.; Ponci, F.; Funabashi, T. Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges. *IEEE Trans. Power Syst.* **2007**, *22*, 1743–1752. [CrossRef]
21. Ma, Z.; Schultz, M.J.; Christensen, K.; Værbak, M.; Demazeau, Y.; Jørgensen, B.N. The application of ontologies in multi-agent systems in the energy sector: A scoping review. *Energies* **2019**, *12*, 3200. [CrossRef]
22. Yao, R.; Hu, Y.; Varga, L. Applications of agent-based methods in multi-energy Systems—A systematic literature review. *Energies* **2023**, *16*, 2456. [CrossRef]
23. Wind Turbine Scada Dataset. Available online: https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset (accessed on 6 October 2023).
24. Das, M.; Balpetek, N.; Akpinar, E.; Akpinar, S. Investigation of wind energy potential of different provinces found in Turkey and establishment of predictive model using support vector machine regression with the obtained results. *J. Fac. Eng. Arch. Gazi Univ.* **2019**, *34*, 2203–2213.
25. Demirsoy, G.; Özsoy, N.; Aytar, D.B.; Kaya, B.B.; Tugrul, B. An analysis of Antalya's wind energy potential utilizing machine learning algorithms. In Proceedings of the 7th International Artificial Intelligence and Data Processing Symposium, Harbin, China, 3–5 March 2023; pp. 1–10.
26. Toquica, D.; Agbossou, K.; Henao, N.; Malhamé, R.; Kelouwani, S.; Amara, F. Prevision and planning for residential agents in a transactive energy environment. *Smart Energy* **2021**, *2*, 100019. [CrossRef]
27. Palanca, J.; Rincon, J.; Julian, V.; Carrascosa, C.; Terrasa, A. Developing IoT artifacts in a MAS platform. *Electronics* **2022**, *11*, 655. [CrossRef]
28. Privacy Focused XMPP/Jabber Server. Available online: https://anonym.im/ (accessed on 6 October 2023).
29. Eken, S.; Şara, M.; Satılmış, Y.; Karslı, M.; Tufan, M.F.; Menhour, H.; Sayar, A. A reproducible educational plan to teach mini autonomous race car programming. *Int. J. Electr. Eng. Educ.* **2020**, *57*, 340–360. [CrossRef]