T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

MACHINE LEARNING EDUCATION ENVIRONMENT

AHMET ÖZDEMIR

SUPERVISOR
DR. ALP ARSLAN BAYRAKÇI

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

MACHINE LEARNING EDUCATION ENVIRONMENT

AHMET ÖZDEMIR

SUPERVISOR
DR. ALP ARSLAN BAYRAKÇI

2025
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 25/06/2025 by the following jury.

JURY

Member (Supervisor)	:	Dr. Alp Arslan Bayrakçı
Member	:	Prof. Erkan Zergeroğlu

ABSTRACT

This project, the ”ML Education Environment,” addresses the challenges of machine learning experimentation by providing an interactive, web-based platform designed to demystify the algorithm training and evaluation process. It targets the common barriers faced by students and practitioners, such as complex environment setups and the need for extensive boilerplate code, which can obscure the nuanced impact of different models, datasets, and hyperparameters.

The system is architected with a decoupled frontend and backend, containerized using Docker for easy deployment and accessibility. The backend is built with Python and the FastAPI framework, serving a RESTful API to handle user requests. A model factory design pattern dynamically dispatches tasks to the appropriate machine learning model implementation, which includes a suite of classification algorithms like Decision Tree, Logistic Regression, SVM, K-Nearest Neighbor, and Artificial Neural Networks. A key feature is the caching mechanism that stores the results of completed experiments in a JSON file, avoiding redundant computations by retrieving past results for identical configurations, thus speeding up the user workflow.

The frontend is a React-based single-page application that provides a user-friendly interface for selecting models, configuring hyperparameters through interactive pop-ups, choosing from a variety of datasets (including subsets and full versions like MNIST), and defining global run settings such as train/test split ratios and feature scaling. Upon running an experiment, the platform processes the request and presents a comprehensive analysis of the results. This includes performance metrics such as accuracy, precision, recall, and F1-score, visualized through interactive charts and comparison tables. To further enhance the educational value, the system integrates with the Gemini API to provide AI-driven, natural-language explanations of the generated charts and metrics, helping users better interpret the outcomes.

In essence, the ML Education Environment successfully creates a sandboxed laboratory for machine learning. It lowers the barrier to entry by abstracting away the complexities of coding and setup, allowing users to focus entirely on the conceptual and practical aspects of model behavior. By enabling rapid, code-free experimentation and visual comparison, the platform serves as an effective educational tool for fostering a deeper, more intuitive understanding of machine learning principles.

Keywords: Machine Learning, Educational Technology, Web Application, Decoupled Architecture, React, FastAPI, Docker, Classification Algorithms, Hyperparameter Tuning, Visual Experimentation, Performance Metrics, Feature Scaling, Cross-Validation, Caching, Sandboxed Environment, AI-Powered Analysis.

ÖZET

Bu proje, makine öğrenmesi deneylerini eğitim amaçlı basitleştirmek için tasarlanmış, ”ML Education Environment” adlı etkileşimli ve web tabanlı bir platformu sunmaktadır. Karmaşık ortam kurulumu ve kodlama yükü gibi yaygın zorlukları ele alan bu sistem, kullanıcıların Karar Ağacı, Destek Vektör Makineleri ve Yapay Sinir Ağları gibi çeşitli sınıflandırma algoritmalarını görsel bir arayüz üzerinden yapılandırmasına ve karşılaştırmasına olanak tanır. Docker ile konteynerize edilmiş olan platform, React tabanlı bir kullanıcı arayüzü ve Python/FastAPI ile geliştirilmiş bir sunucu mimarisine sahiptir. Tekrarlayan hesaplamaları önleyen bir önbellekleme (caching) mekanizması ve sonuçları yorumlamaya yardımcı olan Gemini API entegrasyonu gibi temel özellikler içermektedir. Bu platform, öğrencilerin ve uygulayıcıların farklı parametrelerin model performansı üzerindeki etkisini sezgisel olarak anlamaları için etkili bir sanal deney alanı (sandbox) oluşturmaktadır.

Anahtar Kelimeler: Makine Öğrenmesi Eğitimi, Etkileşimli Web Platformu, Görsel Yapılandırma, Kodsuz Deney, React, FastAPI, Docker, Önbellekleme, Yapay Zeka Destekli Analiz.

ACKNOWLEDGEMENT

I would like to express my gratitude to my project advisor Dr. Alp Arslan Bayrakçı for the guidance, support and insightful feedback he gave me throughout this graduation project.

I am also grateful to Gebze Technical University Computer Engineering Department for providing the academic foundation and resources necessary to undertake this study.

Ahmet Özdemir

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation	**Explanation**
ML	Machine Learning
UI	User Interface
API	Application Programming Interface
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
CORS	Cross-Origin Resource Sharing
SVM	Support Vector Machine
KNN	K-Nearest Neighbors
ANN	Artificial Neural Network
DT	Decision Tree
LR	Logistic Regression
CV	Cross-Validation
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
LLM	Large Language Model
Symbol	**Explanation**
X	Feature matrix (Input variables)
y	Target vector (Output variable)
C	Regularization parameter in SVM and Logistic Regression
K	Number of neighbors in KNN algorithm
R^2	R-squared (Coefficient of determination)

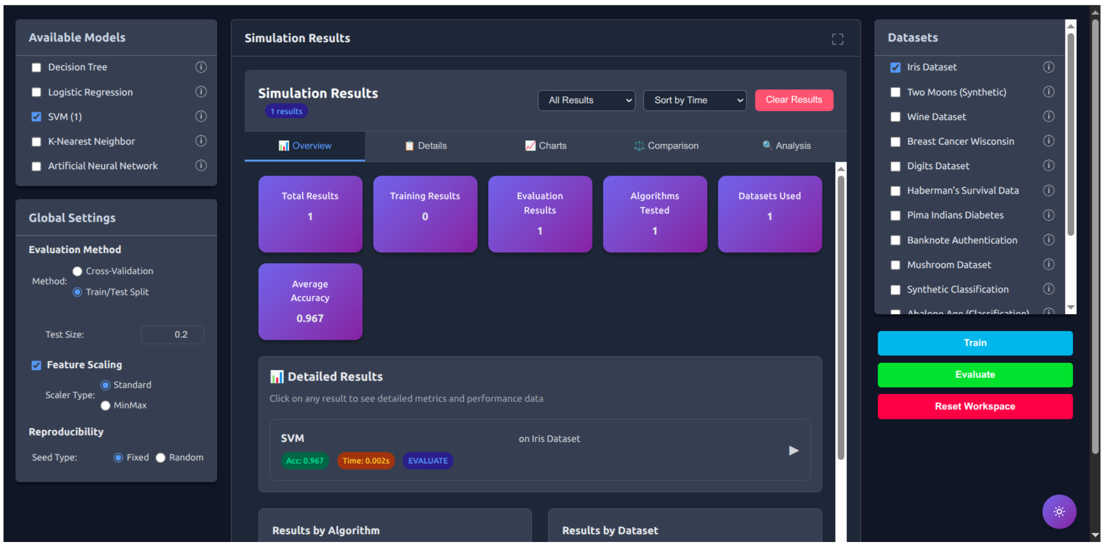
CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. 1.1.



Models: [✓ SVM (1)]

Method: [✓ Train/Test Split]

Test Size: 0.2

[✓ Feature Scaling]

Scaler Type: [✓ Standard]

Seed Type: [✓ Fixed]

Datasets: [✓ Iris Dataset]

Simulation Results — 1 results

Total Results	Training Results	Evaluation Results	Algorithms Tested	Datasets Used	Average Accuracy
1	0	1	1	1	0.967

Click on any result to see detailed metrics and performance data

SVM on Iris Dataset — Acc: 0.967 — Time: 0.002s — EVALUATE

Figure 1.1: A general view of the interface.

1.1. Project Description

The ML Education Environment is a web-based virtual laboratory that allows users to interactively train, evaluate, and compare machine learning classification models. The system is designed with a modern software architecture, featuring a decoupled frontend built with React and a backend built with Python and FastAPI.

The user workflow is designed to be simple and intuitive, following these key steps:

- **Model Selection:** Users can choose from a suite of classic classification algorithms, including Decision Tree, Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Artificial Neural Networks (ANN).

- **Hyperparameter Configuration:** Through interactive pop-ups, users can configure key hyperparameters for each model, such as `Criterion`, `Max Depth`, or `Kernel`.
- **Dataset Selection:** The project includes a variety of standard datasets of differing sizes and complexity. These datasets, which range from small, classic examples like Iris and Wine to larger ones like MNIST, allow for comprehensive testing of the algorithms. A complete list of the datasets and their respective properties is presented in Table 1.1. Additionally, the ”Bike Sharing Hourly” dataset, which is fundamentally a regression problem, was intentionally included to demonstrate how the classification-focused platform handles different problem types and to serve as an example for non-classification tasks.
- **Global Settings:** Users can control global settings for experiment consistency, such as the train/test split ratio, the choice between train/test split and cross-validation, and the application of feature scaling.

Once an experiment is run, the backend processes this configuration, trains the model, evaluates it against specified metrics (e.g., Accuracy, Precision, F1-Score), and returns the results to the frontend. The frontend then presents these outcomes in clear tables and graphs for the user.

Table 1.1: A full summary of the datasets used in the project.

Dataset Name	**Number of Sample**	**Number of Features**	**Number of Class**
Iris	150	4	3
Two Moons (Synthetic)	400	2	2
Wine	178	13	3
Breast Cancer Wisconsin	569	30	2
Digits	1797	64	10
Haberman’s Survival	306	3	2
Pima Indians Diabetes	768	8	2
Banknote Authentication	1372	4	2
Mushroom	8124	22	2
Synthetic Classification	1000	20	2
Abalone	4177	8	28
Bike Sharing Hourly	17379	16	Regression Task
MNIST (Subset)	7,000	784	10
MNIST (Full)	70,000	784	10

1.2. Success Criteria

The success of this project is not measured by the performance of the machine learning models themselves, but rather by the platform's effectiveness as an educational and experimental tool. The goal is to provide a robust and transparent simulation environment where the outcomes—whether high or low in performance—serve as valuable learning moments.

1.2.1. Functional and Technical Criteria

- **System Implementation:** The platform must successfully integrate the full suite of planned classification algorithms (Decision Tree, Logistic Regression, SVM, KNN, ANN) and allow users to run training and evaluation tasks without system errors.
- **Core Feature - Experimentation Engine:** The platform must correctly process user-defined configurations and execute the ML pipeline, including data processing, model training, and evaluation, as specified by the user's global settings.
- **Core Feature - Caching:** The caching mechanism must be operational, correctly identifying identical experiments and serving stored results to prevent redundant computations, thereby ensuring an efficient user experience.
- **Core Feature - Deployment:** The entire system must be containerized with Docker and easily launchable via a single `docker-compose` command, fulfilling the goal of accessibility and straightforward setup.

1.2.2. User Experience and Educational Criteria

- **Clarity and Usability:** The user interface must be intuitive, allowing users with minimal ML or coding experience to conduct experiments. Tooltips and other informational guides should clearly explain the purpose of different models and datasets.
- **Comprehensive Result Visualization:** The platform must display results using a variety of standard metrics and visualizations, such as comparative charts and confusion matrices, in a format that is clear and easy to understand.
- **Fostering Intuitive Learning:** The primary success is measured by the platform's ability to act as a "sandbox." It should empower users to observe the

direct cause-and-effect relationship between their choices (e.g., adjusting a hyperparameter, enabling feature scaling) and the resulting model performance.

- **Enhanced Interpretation with AI:** The integration with the Gemini API must successfully provide users with natural-language explanations of their results, helping them to better interpret the data visualizations and metrics, thus deepening their learning experience.

It is important to note that achieving a specific performance threshold (e.g., 95% accuracy) is not a success criterion for this project. Instead, the platform's success lies in its ability to transparently run the simulation and report the outcome, whether good or bad, as this is integral to the learning process.

1.3. Challenges and Goals

The development of the ML Education Environment was guided by a set of core goals and shaped by several key challenges:

1.3.1. Goals

- **Educational Impact:** The primary goal was to create an accessible, interactive tool that transforms the abstract theory of machine learning into a tangible, hands-on experience.
- **Accessibility:** To lower the barrier to entry for machine learning experimentation by providing a completely code-free, visual platform.
- **Technical Robustness:** To build a stable and scalable system with a modern, decoupled architecture capable of handling various experimental configurations.
- **Efficiency:** To design an efficient workflow for the user, where a caching mechanism minimizes redundant computations and provides rapid feedback.

1.3.2. Challenges

- **UI/UX Design:** A significant challenge was designing an interface that is simple enough for beginners to use without feeling overwhelmed, yet powerful enough to allow for meaningful and complex experiments.
- **System Integration:** Managing the complex data flow and state synchronization between the React frontend, the asynchronous FastAPI backend, the ML model

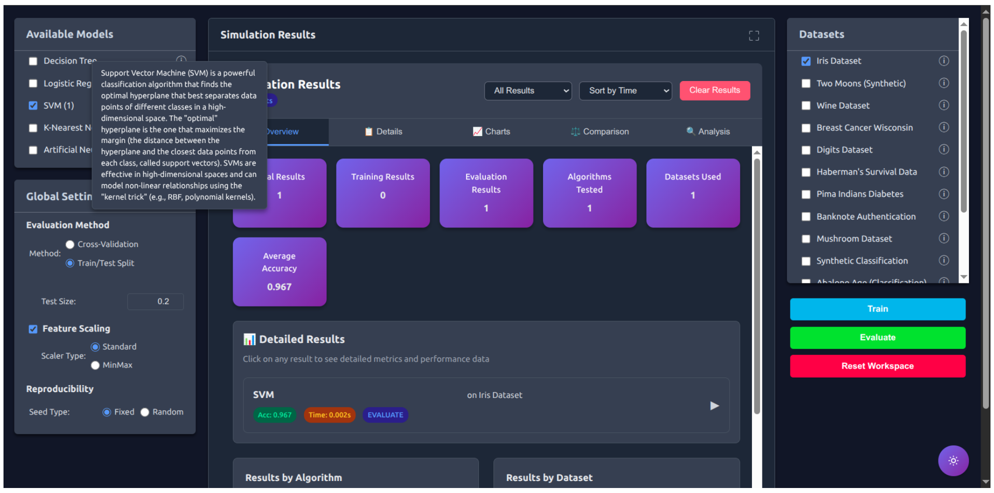
execution pipeline, and the caching layer required careful architectural planning.

- **Performance and Scalability:** Ensuring that the backend could efficiently handle computationally intensive tasks, especially with larger datasets like the full MNIST dataset, was a key consideration that directly influenced the implementation of the caching strategy.

- **Effective Communication of Results:** Presenting complex machine learning metrics and visualizations in a manner that is not just accurate but also easily digestible for a non-expert audience was a core challenge addressed by the UI design and the Gemini API integration.

2. IMPLEMENTATION OF THE EDUCATIONAL WORKFLOW AND SYSTEM PIPELINE

This chapter delves into the technical implementation of the ML Education Environment, focusing on the end-to-end pipeline that powers the user's experimental workflow. It further explores the design strategies employed to create an effective educational interface and the integration methods that ensure the system's robustness and portability. A core philosophy of the user interface is to present information in an accessible and non-intrusive manner. For instance, instead of overwhelming the user with lengthy descriptions, key information about models or datasets is revealed in clean, informative tooltips that appear on-demand when the user hovers over an information icon, as demonstrated in Figure 2.1. This approach keeps the interface uncluttered while ensuring educational content is readily available, forming a key part of the platform's user-centric design.



Support Vector Machine (SVM) is a powerful classification algorithm that finds the optimal hyperplane that best separates data points of different classes in a high-dimensional space. The "optimal" hyperplane is the one that maximizes the margin (the distance between the hyperplane and the closest data points from each class, called support vectors). SVMs are effective in high-dimensional spaces and can model non-linear relationships using the "kernel trick" (e.g., RBF, polynomial kernels).

Evaluation Method
Method: Cross-Validation [✓ Train/Test Split]
Test Size: 0.2
Feature Scaling ✓
Scaler Type: [✓ Standard] MinMax
Reproducibility
Seed Type: [✓ Fixed] Random

Click on any result to see detailed metrics and performance data
SVM on Iris Dataset
Acc: 0.967 Time: 0.002s EVALUATE

Figure 2.1: Informational tooltip providing on-demand details about the SVM algorithm without cluttering the main interface.

2.1. The Machine Learning Pipeline: From Request to Result

The core of the ML Education Environment is a robust, end-to-end pipeline that translates a user's selections on the web interface into a tangible machine learning experiment and returns the results. This entire process, from the initial API call to the final cached response, is designed to be modular and efficient. The workflow is composed of several distinct stages, each handled by a specialized component in the backend.

2.1.1. API Request Handling and Validation

The process begins when the user initiates a "Train" or "Evaluate" action from the frontend. This action dispatches a POST request to the corresponding endpoint (`/train` or `/evaluate`) on the FastAPI backend. Upon receipt, FastAPI utilizes Pydantic models to automatically parse and validate the incoming JSON payload. This ensures that the request contains all necessary information—such as the algorithm name, hyperparameters, selected dataset, and global settings—in the correct format before any computational work begins. If any data is missing or malformed, FastAPI returns an informative HTTP error, preventing invalid requests from entering the pipeline.

2.1.2. Data Processing and Preparation

Once the request is validated, it is passed to the `data_processor` module. This component is responsible for all data-related preliminary steps. It first locates and loads the selected dataset (e.g., `iris.csv`) from the directory specified by the `DATA_DIR` variable. It then separates the data into a feature matrix (X) and a target vector (y). Based on the user's global settings, the processor either partitions the data into training and testing sets using `train_test_split` or prepares the full dataset for cross-validation. Crucially, if feature scaling is enabled, it is applied correctly within this stage to prevent data leakage, for example by fitting the scaler only on the training data in a train/test split scenario.

2.1.3. The Model Factory and Dynamic Instantiation

The prepared data dictionaries are then forwarded to the `model_factory`. This component serves as a central dispatcher that decouples the main application logic from the specific model implementations. It reads the `algorithm` name from the user's request (e.g., "SVM") and calls the corresponding function (e.g., `train_and_evaluate_svm`) from the `ml_models` package. The user-defined hyperparameters are passed along to this function, which then instantiates the appropriate Scikit-learn model with the specified configuration. This factory pattern makes the system highly modular and extensible, as new algorithms can be integrated by simply adding a new file to the ml_models directory and a corresponding case in the factory.

2.1.4. Caching and Response Generation

After a model is trained and/or evaluated, the results payload is sent to the cache_manager. A unique SHA-256 hash key is generated from the experiment's precise configuration, including the model name, parameters, dataset, and global settings. The cache_manager first checks if a result for this key already exists. If so, the cached result is returned immediately. If not, the new result is saved to a central JSON file (training_cache.json), ensuring that any future identical requests can be served instantly. This caching strategy is vital for improving performance and providing a responsive user experience. Finally, the complete results object is serialized into a JSON format and returned to the frontend.

2.2. Designing an Effective Educational Interface

The primary goal of this project is to create an educational tool, and therefore, the design of the user interface (UI) was paramount. The interface was developed not just as a means to trigger backend processes, but as an integral part of the learning experience itself. The core design philosophy was based on three principles: simplicity, direct manipulation, and immediate feedback, all intended to lower the cognitive load on the user and allow them to focus on understanding machine learning concepts.

2.2.1. Core Principles of the User Interface

To achieve its educational objectives, the UI was built around the following principles:

- **Clarity and Simplicity:** The interface utilizes a clean, three-column layout to logically separate model selection, global settings, and dataset selection. This structure intuitively guides the user through the necessary steps of building an experiment without causing information overload.
- **Direct Manipulation:** Instead of requiring users to write code or edit configuration files, the platform allows for direct manipulation of all experimental variables. Users interact with checkboxes, sliders, and dropdown menus, making abstract concepts like hyperparameter tuning feel tangible and accessible.
- **Immediate and Contextual Feedback:** The interface provides immediate feedback for user actions. For example, configuring a model instance instantly adds

it to the ”Configured Model Instances” list. Furthermore, as shown in Figure 2.1, informational tooltips provide on-demand explanations for models and datasets, enriching the user’s understanding without cluttering the main view.

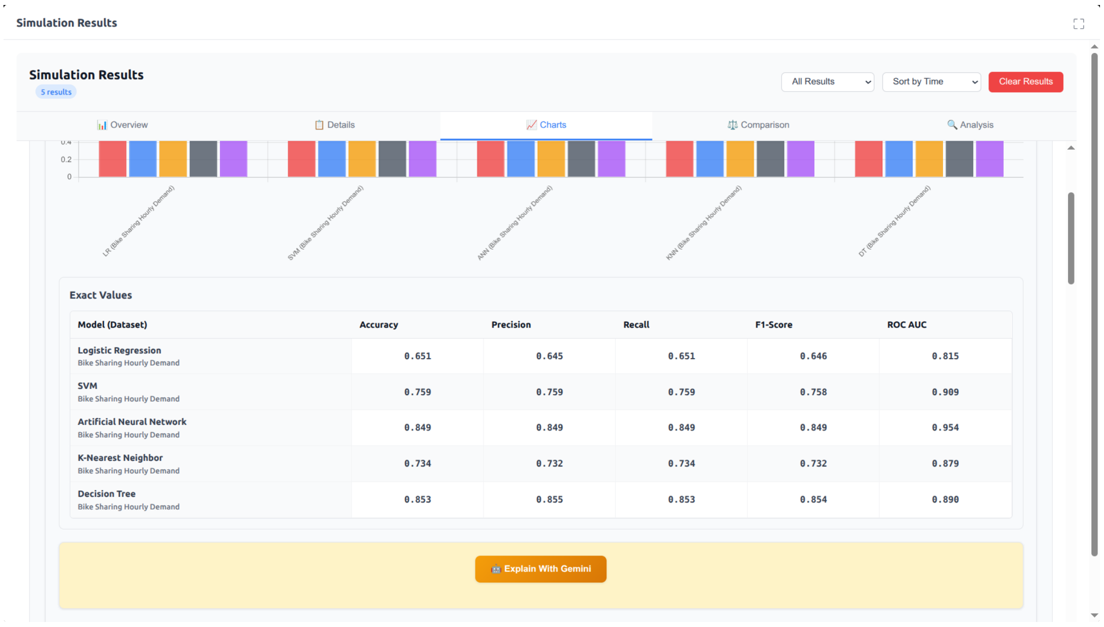
2.2.2. Interactive Components for Conceptual Learning

Several key components were specifically designed to transform abstract machine learning concepts into interactive learning modules.

2.2.2.0.1 Global Settings Panel. This panel is a crucial educational component, allowing users to control high-level experimental conditions. By toggling between `Train/Test Split` and `Cross-Validation`, a user can directly run experiments and compare how these fundamental evaluation strategies affect model performance and result stability. Similarly, the `Feature Scaling` option provides a practical demonstration of why scaling is critical for distance-based algorithms like SVM and KNN, but less so for tree-based models.

2.2.2.0.2 Hyperparameter Configuration. When a user selects a model, a modal pop-up appears, providing a dedicated and user-friendly interface for hyperparameter configuration as shown in Figure 2.3. This design abstracts away the complexity of a model’s API and code-based configuration. The use of specific input types—such as sliders for numerical ranges (e.g., K in KNN), dropdowns for categorical choices (e.g., `Criterion` in Decision Trees), and text fields for flexible inputs (e.g., hidden layer sizes in ANNs)—makes the process of tuning a model more intuitive and exploratory.

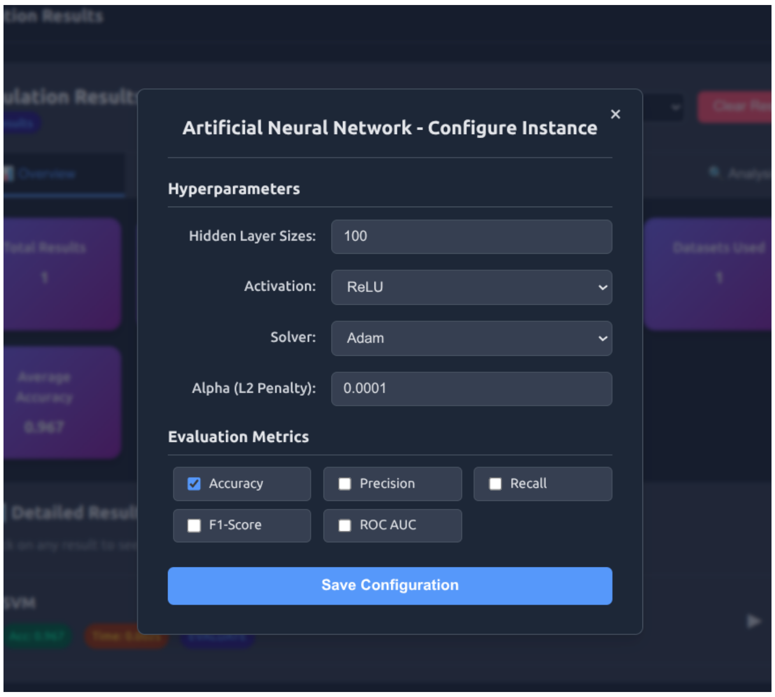
2.2.2.0.3 Results Visualization. The presentation of results is perhaps the most critical part of the learning loop. The `ResultsViewer.js` component utilizes a multi-tab layout (”Overview”, ”Charts”, ”Comparison”, etc.) to structure the output. This allows a user to begin with a high-level summary and progressively delve into more complex analyses. The ”Charts” tab, powered by `MetricsChart.js`, provides multiple perspectives on performance, including comparative bar charts for direct metric comparison, radar charts for a holistic view of model strengths, and confusion matrices for detailed error analysis. This multi-faceted approach enables users to interpret the outcomes of their experiments from various angles. A critical example demonstrating this educational role of the platform is presented in Figure 2.2. In this scenario, the ’Bike Sharing Hourly’ dataset, which is fundamentally a regression problem, was intentionally run with classification algorithms. The resulting low metric values, displayed on the interface, visually demonstrate the consequences of a mismatch between the problem type and the applied model family, showcasing a core educational objective of the platform.


Simulation Results
Simulation Results
5 results
All Results
Sort by Time
Clear Results
Overview
Details
Charts
Comparison
Analysis
LR (Bike Sharing Hourly Demand)
SVM (Bike Sharing Hourly Demand)
ANN (Bike Sharing Hourly Demand)
KNN (Bike Sharing Hourly Demand)
DT (Bike Sharing Hourly Demand)
Explain With Gemini

Exact Values

Model (Dataset)	Accuracy	Precision	Recall	F1-Score	ROC AUC
Logistic Regression Bike Sharing Hourly Demand	0.651	0.645	0.651	0.646	0.815
SVM Bike Sharing Hourly Demand	0.759	0.759	0.759	0.758	0.909
Artificial Neural Network Bike Sharing Hourly Demand	0.849	0.849	0.849	0.849	0.954
K-Nearest Neighbor Bike Sharing Hourly Demand	0.734	0.732	0.734	0.732	0.879
Decision Tree Bike Sharing Hourly Demand	0.853	0.855	0.853	0.854	0.890

Figure 2.2: Interface output showing the expected low performance metrics obtained by running classification models on the regression-based 'Bike Sharing Hourly' dataset. This scenario is used as an educational example to highlight the importance of the model-problem type compatibility.



Artificial Neural Network - Configure Instance

Hyperparameters

Hidden Layer Sizes: 100

Activation: ReLU

Solver: Adam

Alpha (L2 Penalty): 0.0001

Evaluation Metrics

[✓ Accuracy] Precision Recall F1-Score ROC AUC

Figure 2.3: Interactive pop-up for configuring Artificial Neural Network.

2.3. System Integration and Containerization

This section covers how the disparate parts of the project—the frontend web application and the backend processing engine—are connected and deployed to work as a single, cohesive unit. The use of containerization with Docker was a foundational decision to ensure portability, consistency, and ease of setup.

2.3.1. Service Orchestration with Docker Compose

The entire application stack is managed by a single `docker-compose.yml` file. This file defines the project's two main services: a `frontend` service for the React application and a `backend` service for the Python/FastAPI application, as illustrated in Figure 2.4. Docker Compose handles the building of images for each service from their respective Dockerfiles and orchestrates the creation of an isolated network where they can communicate. This approach ensures that the application runs in a consistent environment, eliminating the "it works on my machine" problem and fulfilling the key goal of accessibility for any user.

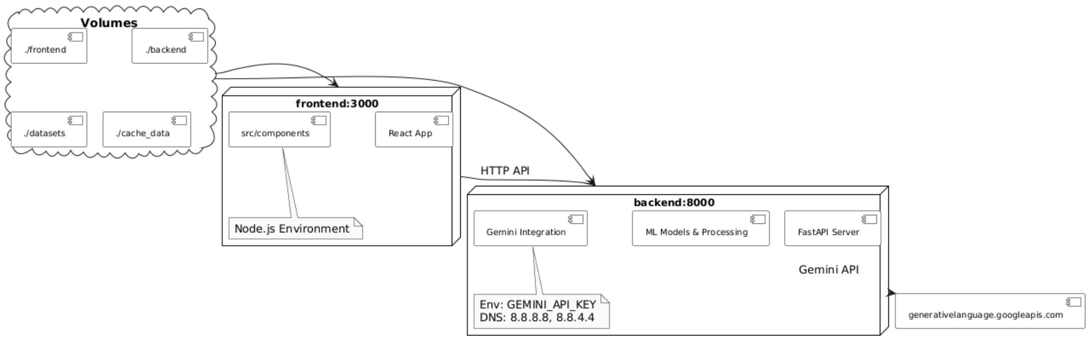


Figure 2.4: Service architecture as defined in Docker Compose, showing the interconnected frontend and backend services.

2.3.2. Inter-Service Communication

The frontend and backend services communicate over the virtual network created by Docker Compose. The React development server, running inside the `frontend` container, is configured to proxy API requests to the backend service. This is achieved by using the service name defined in `docker-compose.yml` as the hostname (e.g., sending requests to `http://backend:8000`). Docker's internal DNS resolves this hostname to the correct container's IP address on the shared network. This method provides a stable and robust way for the services to communicate without needing to hardcode IP addresses.

2.3.3. Data Persistence with Volumes

To manage data and ensure its persistence, Docker volumes are used strategically. The source code for both services is mounted into the containers, which allows for live-reloading during development, speeding up the implementation cycle. More importantly, two key directories are mounted as volumes into the backend container:

- `./project_datasets/datasets`: This volume provides the backend with access to the datasets located on the host machine, allowing users to easily add or modify datasets.
- `./cache_data`: This volume maps to the location where `training_cache.json` is stored, ensuring that the cache persists even if the backend container is stopped or rebuilt. This is crucial for the efficiency and stateful nature of the application.

3. CONCLUSION AND FUTURE WORK

3.1. Conclusion

This project set out to address a fundamental challenge in machine learning education: the significant barrier to entry presented by complex coding requirements and environment configurations. The primary objective was to develop an interactive, code-free, and web-based educational platform, the ”ML Education Environment,” to demystify the experimental process for students and practitioners. Based on the outcomes and the final state of the application, it can be confidently stated that all initial project goals and success criteria have been fully met and, in some areas, exceeded.

The platform successfully integrates a suite of core classification algorithms and provides an intuitive user interface for configuring experiments with various datasets and global settings. The technical implementation, featuring a decoupled architecture containerized with Docker, has proven to be robust, portable, and efficient. The caching mechanism, in particular, was highly effective, drastically reducing computation times for repeated experiments and enhancing the user experience by providing near-instantaneous feedback.

Beyond the initial scope, several key features were added to increase the platform's utility and educational value. A dark/light mode option was implemented to improve user comfort, and more significantly, an integration with the Gemini API was established. This allows users to receive AI-powered, natural-language explanations of complex charts and results, transforming raw data into understandable insights. Furthermore, an ”Analysis” section was developed to allow users to inspect the underlying source code of the models they select, providing a bridge between visual experimentation and the code that powers it.

In terms of personal development, this project provided deep insights into building highly modular systems and managing the complexities of frontend-backend communication. It deepened my own understanding of machine learning models and the nuances of the training pipeline. Moreover, it offered significant experience in user-centered design, focusing on how to present complex technical information in a way that is both visually appealing and conceptually clear—a critical skill for any computer engineer tasked with communicating technical outputs. The final platform stands as a proof of concept that with a well-designed interface, even individuals with minimal knowledge in the field can explore and grasp the fundamental dynamics of machine learning.

3.2. Future Work

While the ”ML Education Environment” successfully meets its objectives, it also establishes a strong foundation for numerous potential enhancements. The following areas represent promising directions for future development to further expand its scope and educational impact:

- **Expansion of Model Diversity:** The most immediate enhancement would be to broaden the scope of supported algorithms. Integrating regression models (e.g., Linear Regression, Ridge, Lasso) and unsupervised learning algorithms (e.g., K-Means, DBSCAN) would provide a more comprehensive educational experience covering a wider range of machine learning tasks.
- **User-Uploaded Content:** A significant step towards greater flexibility would be to implement functionality allowing users to upload their own datasets (e.g., via CSV files). In a more advanced version, a framework could be developed for users to upload their own custom model scripts that adhere to a specific API, enabling them to test their own creations within the platform's environment.
- **Automated Report Generation:** A powerful feature would be to add an automated reporting system. This could run as a background thread, generating a user-friendly, well-formatted document (e.g., PDF or Markdown) for each completed experiment. This report could summarize the configuration, results, and metrics, and could even leverage an LLM to generate descriptive text, providing users with a shareable and persistent record of their findings.
- **Global Deployment and Accessibility:** To maximize its educational reach, the platform could be deployed to a global cloud server, making it a publicly accessible web tool for students and educators worldwide, moving beyond its current Docker-based local deployment model.

CV

Personal Information

Name Surname	Ahmet Özdemir
Date of Birth	03.01.2001
E-mail	ahmetozdemiir.ao@gmail.com
GitHub	https://github.com/ahmetozdemirrr

Education

- **B.Sc. in Computer Engineering**, Gebze Technical University, Kocaeli, [Expected Graduation Year, e.g., 2025]

BIBLIOGRAPHY

[1] E. Alpaydın, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[2] FastAPI Team, ”FastAPI,” [Online]. Available: https://fastapi.tiangolo.com/. [Accessed: May, 2025].

[3] Docker Inc., ”Docker Documentation,” [Online]. Available: https://docs.docker.com/. [Accessed: May, 2025].