AN INTELLIGENT MACHINE LEARNING MODEL FOR FIRE WEATHER INDEX PREDICTION

AN INTERNSHIP PROJECT REPORT

Submitted by

BALAGURU. K

in partial fulfilment of the requirements for the completion of

INFOSYS SPRINGBOARD INTERNSHIP

in the domain of

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Conducted at

INFOSYS SPRINGBOARD

OCTOBER 2025

ABSTRACT

The increasing frequency and intensity of forest fires have become a major concern worldwide, posing significant threats to ecosystems, biodiversity, human life, and property. Early prediction of fire risk plays a crucial role in mitigating these dangers by enabling authorities and local agencies to take timely preventive and control measures. This project aims to develop a predictive system for determining the Fire Weather Index (FWI) a key indicator used to assess fire danger levels using advanced machine learning techniques. The system leverages historical weather and environmental data, including variables such as temperature, relative humidity, wind speed, and rainfall. The project begins with data collection from reliable meteorological sources, followed by data preprocessing steps such as handling missing values, feature encoding, and normalization to ensure data consistency and accuracy. Exploratory Data Analysis (EDA) is conducted to identify patterns, correlations, and trends within the dataset, which helps in selecting the most influential features for model training. Feature scaling is applied to optimize model convergence and stability. Among various regression algorithms evaluated, Ridge Regression is chosen due to its capability to handle multicollinearity and provide robust predictions. The model's performance is quantitatively assessed using error metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE), offering insights into prediction accuracy. To enhance accessibility and real-world applicability, the trained model is integrated into a user-friendly Flask web application, enabling users to input real-time weather parameters and instantly receive fire risk predictions categorized as low, moderate, or high. This project not only highlights the effectiveness of machine learning in environmental risk assessment but also demonstrates practical skills in data science workflows, model optimization, and full-stack deployment acquired during the Infosys Springboard internship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

SNO	WORD	ABBREVATIONS
1	FWI	Fire weather index
2	AI	Artificial Intelligence
3	ML	Machine Learning
4	MSE	Mean Squared Error
5	MAE	Mean Absolute Error
6	RMSE	Root Mean Squared Error
7	API	Application Programming Interface
8	IDE	Integrated Development Environment

CHAPTER 1

INTRODUCTION OF THE PROJECT

1.1 Introduction

Forest fires are increasingly frequent due to the combined impacts of climate change, prolonged droughts, and human activities such as deforestation and uncontrolled land use. These fires cause extensive ecological damage, loss of wildlife habitats, and severe economic consequences for affected regions. Predicting fire occurrence in advance can significantly reduce damage by enabling early warning systems, efficient allocation of firefighting resources, and timely evacuation measures. In recent years, data-driven approaches have gained prominence, with machine learning offering powerful tools to analyze complex weather patterns and predict fire risk effectively. By leveraging large-scale meteorological and environmental datasets including parameters like temperature, humidity, wind speed, and precipitation machine learning models can uncover hidden patterns that traditional statistical methods often overlook. This project applies supervised learning techniques to predict the Fire Weather Index (FWI), a globally recognized indicator used to quantify fire danger levels. The approach integrates data preprocessing, feature engineering, and model optimization to ensure high prediction accuracy. Through this work, the project aims to contribute to proactive disaster management by providing accurate, real-time insights into potential fire risks.

1.2 Problem statement

Traditional fire risk assessment methods rely heavily on manual analysis, empirical observations, and basic statistical models that often fail to capture the complex, nonlinear relationships between environmental variables influencing fire behavior. Such methods are not only time-consuming and labor-intensive but also prone to human error and limited scalability when dealing with large datasets

or varying geographical regions. Moreover, they struggle to adapt to dynamic climate conditions and rapid weather fluctuations, leading to delayed or inaccurate predictions. Therefore, there is a growing need for an automated, data-driven, and scalable system that can utilize both real-time and historical data to accurately predict fire risk. By integrating advanced machine learning techniques, such a system can provide faster, more reliable forecasting, supporting proactive decision-making and efficient resource allocation for fire prevention and management.

1.3 Objectives of the project

- To analyze historical weather and fire data
- To build a machine learning model for Fire Weather Index prediction
- To evaluate model performance using error metrics
- To deploy the model using a web-based interface
- To provide early fire risk classification for preventive action

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing system

Existing fire prediction systems primarily depend on rule-based approaches and traditional statistical techniques. These systems use manually defined rules, threshold values, or linear models to estimate fire risk based on environmental factors such as temperature, humidity, and wind speed. While these methods provide basic insights into fire danger levels, they often fail to capture the complex, nonlinear interactions among multiple climatic variables that influence fire occurrences. Furthermore, most existing systems rely on static datasets, making them less adaptive to real-time weather fluctuations and modern data streams from sensors and satellites.

2.1.1 Drawbacks:

- Low prediction accuracy: Rule-based and traditional methods oversimplify relationships among variables, leading to inconsistent and imprecise results.
- Poor handling of large datasets: These systems are not optimized for processing or learning from massive, high-dimensional datasets commonly available today.
- Lack of real-time prediction: Many existing models operate on historical data and cannot update predictions dynamically as new data arrives.
- Limited scalability: Traditional systems often struggle to adapt when extended to larger regions or integrated with modern data infrastructure such as IoT or cloud-based systems.

2.2 Proposed system

The proposed system employs machine learning techniques to accurately predict the Fire Weather Index (FWI), a key metric used to evaluate fire danger levels based on meteorological and environmental factors. The system leverages historical weather data, including temperature, humidity, wind speed, and rainfall, to train a predictive model capable of identifying complex relationships among these parameters. Data preprocessing steps such as normalization, feature selection, and outlier removal are performed to ensure model reliability and consistency. A supervised learning approach specifically Ridge Regression is used to enhance prediction accuracy by minimizing overfitting and handling multicollinearity effectively. The model's performance is evaluated using standard regression metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).

To ensure usability and accessibility, the trained model is deployed through a Flask-based web application, allowing users to input real-time weather conditions and instantly receive fire risk predictions categorized as low, moderate, or high. This integration of machine learning and web deployment enhances the system's practicality and real-world applicability.

2.2.1 Advantages:

- Higher prediction accuracy: Machine learning algorithms capture complex dependencies among environmental factors for precise forecasting.
- Automated data processing: The system automatically preprocesses and analyzes data, reducing manual intervention.
- Real-time prediction capability: Live weather inputs enable up-to-date fire risk predictions.

CHAPTER 3

SYSTEM SPECIFICATIONS

3.1 Hardware specifications

- Processor: Intel i7
- RAM: Minimum 8 GB
- Storage: 256 GB HDD

3.2 Software specifications

- Operating System: Windows
- Programming Language: Python
- Libraries: NumPy, Pandas, Scikit-learn, Matplotlib
- Framework: Flask
- IDE: VS Code
- Algorithm : Ridge Regression

3.3 Technology used

3.3.1 Python :

Python is a versatile, high-level programming language widely used in data science and machine learning applications due to its simplicity, readability, and extensive library support. It provides strong capabilities for numerical computation, data visualization, and model development. In this project, Python serves as the core language for data preprocessing, model training, evaluation, and integration with the web application. Its object-oriented nature and modular design make it ideal for building scalable machine learning solutions.

3.3.2 Libraries :

- NumPy: NumPy (Numerical Python) is a fundamental library for numerical computations. It supports multi-dimensional arrays,

mathematical operations, and linear algebra functions. In this project, NumPy is used for array manipulations, matrix operations, and handling numerical data efficiently.

- Pandas: Pandas is a data manipulation and analysis library that offers powerful tools such as DataFrames for handling structured datasets. It is used for importing, cleaning, transforming, and managing weather and environmental data before feeding it into the machine learning model.
- Scikit-learn: Scikit-learn is a robust machine learning library that provides a wide range of tools for data preprocessing, model selection, training, and evaluation. In this project, it is used to implement the Ridge Regression algorithm, perform feature scaling, and compute performance metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).
- Matplotlib: Matplotlib is a visualization library used to create static, interactive, and animated plots. It helps visualize data trends, feature distributions, and model performance through charts and graphs, aiding better understanding during exploratory data analysis (EDA).

3.3.3 Flask :

Flask is a lightweight and flexible web framework for Python that facilitates the development and deployment of web applications. It provides the tools needed to connect the trained machine learning model with a web interface. In this project, Flask enables users to input real-time weather parameters via a simple web form and receive fire risk predictions instantly. Its minimalistic design allows easy integration with HTML, CSS, and JavaScript for creating a user-friendly front end.

3.3.4 Visual studio code (vs code) :

Visual Studio Code is a powerful, open-source Integrated Development Environment (IDE) developed by Microsoft. It supports multiple programming languages and provides essential features such as code linting, debugging, version

control (Git), and integrated terminal support. For this project, VS Code serves as the primary environment for writing and testing Python scripts, managing virtual environments, and deploying the Flask application efficiently.

3.3.5 Ridge regression :

Ridge Regression is a type of linear regression that includes a regularization term (L2 penalty) to prevent overfitting by shrinking the coefficients of less significant features. It is particularly effective in scenarios where independent variables are highly correlated (multicollinearity). In this project, Ridge Regression helps improve the accuracy and generalization capability of the Fire Weather Index prediction model. By minimizing both the error and the magnitude of coefficients, the model achieves a stable and reliable fit even with complex weather datasets.

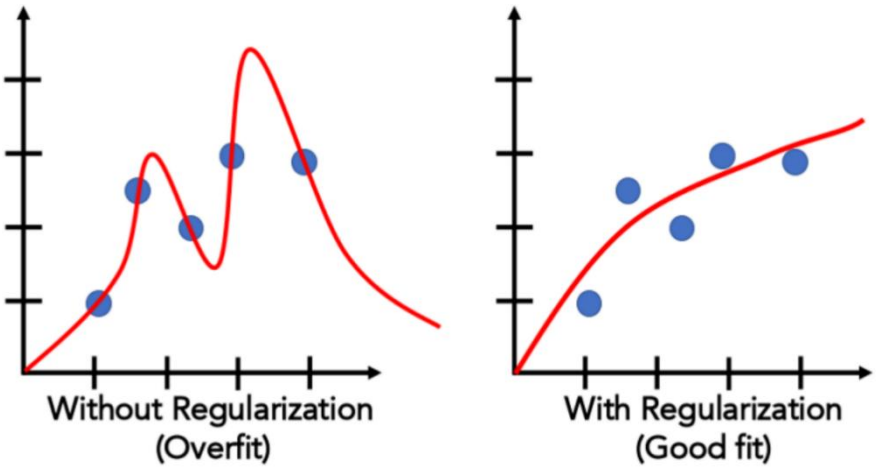
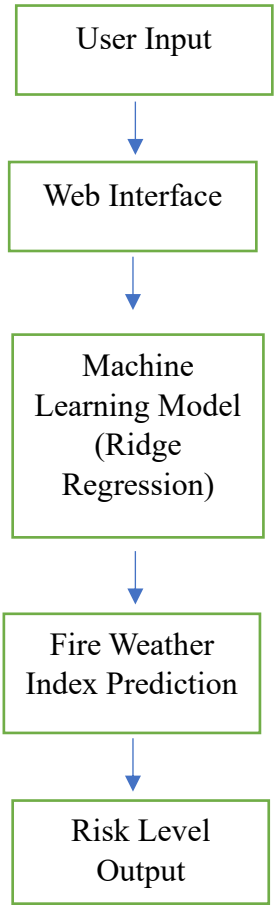


Fig 3.1 Ridge regression

CHAPTER 4

SYSTEM DESIGN

4.1 System architecture



4.1 System architecture

The user enters weather details like temperature, humidity, wind speed, and rainfall through a Flask web app. The system processes this data to make it suitable for prediction, then uses a Ridge Regression model to calculate the Fire Weather Index. Based on the predicted value, the system classifies the fire risk as low, moderate, or high and displays the result to the user on the web page.

CHAPTER 5

SYSTEM IMPLEMENTATIONS

5.1 Modules

The system is divided into multiple functional modules that work together to detect FWI accurately and shows output in web app. Each module plays a vital role in ensuring seamless communication, high reliability, and effective decision making . The modules are listed below

- 5.1.1 Dataset Overview & Preprocessing
- 5.1.2 Dataset Statistics & Visualization
- 5.1.3 Model Building using Ridge Regression
- 5.1.4 Model Deployment using Flask

5.1.1 Dataset overview & preprocessing

In this section, the dataset used for the project was explored and understood. Initial inspection was done using basic statistical measures to identify missing values and data distribution. Preprocessing steps such as feature scaling were applied to improve model performance. Standardization was used to ensure all features have a mean of zero, and normalization was applied where required to bring values into a common range.

Pseudo code :

- Import pandas and numpy.
- Load dataset from CSV.
- Display basic information of the dataset.
- Remove duplicate rows.
- Handle missing values.
- Clean text data by removing spaces and converting to lowercase.
- Fix incorrect data types.

- Rename columns for consistency.
- Remove outliers using IQR method.
- Display final cleaned data shape.
- Save cleaned dataset to CSV.

5.1.2 Dataset statistics & visualization

Statistical analysis was performed to understand the relationship between features.

- Correlation analysis helped identify dependencies between variables.
- Histograms were plotted to visualize data distribution and detect skewness or outliers. This step improved understanding of the dataset before applying machine learning models.

Pseudo code :

- Import pandas, seaborn, and matplotlib libraries.
- Load dataset from CSV.
- Create a copy of the dataset for correlation analysis.
- Select only numeric columns.
- Plot histograms for all numeric features.
- Compute correlation of all numeric variables with FWI.
- Display correlation values with FWI.
- Plot a correlation heatmap of all numeric variables.

5.1.3 Model building using ridge regression

Ridge Regression was implemented to handle multicollinearity and reduce overfitting. The model was trained and evaluated using performance metrics such as:

- Mean Squared Error (MSE)

- Mean Absolute Error (MAE) and Hyperparameter α (alpha) was tuned to control model complexity and improve prediction accuracy.

Pseudo code :

- Import required libraries (numpy, pandas, matplotlib, sklearn, joblib).
- Load cleaned dataset from CSV.
- Select feature columns and target column (FWI).
- Split data into training and testing sets.
- Scale features using StandardScaler.
- Define a list of alpha values for Ridge Regression.
- Loop through alpha values to train Ridge models.
- Predict on training and testing sets for each alpha.
- Compute performance metrics: MSE, RMSE, MAE, R^2.
- Identify best alpha based on lowest test MSE.
- Train Ridge model using the best alpha.
- Predict FWI using the best model.
- Plot Predicted vs Actual FWI.
- Plot MSE, RMSE, and MAE vs Alpha.
- Print best alpha and corresponding performance metrics.
- Print results table for all alpha values.
- Check for overfitting or underfitting.
- Save the best Ridge model and scaler using joblib.

5.1.4 Model deployment using flask

A Flask web application was developed to deploy the trained machine learning model. The backend processes user inputs and returns predictions in real time. This integration demonstrates how machine learning models can be connected with web applications for practical use.

Pseudo code :

- Import required libraries: Flask, numpy, joblib.
- Initialize Flask app and set secret key.
- Define a dictionary of valid users and passwords.
- Load trained Ridge regression model and scaler using joblib.
- Create /login route to handle GET and POST requests.
- Validate username and password on login and create session.
- Display login page with error messages for invalid credentials.
- Create /logout route to clear session and redirect to login.
- Create /predict route to handle FWI prediction.
- Check if user is logged in; otherwise redirect to login.
- Get meteorological and FWI sub-index inputs from POST request.
- Convert inputs to numpy array and reshape for prediction.
- Scale input data using the loaded scaler.
- Predict FWI using the trained Ridge model.
- Round prediction and display on the FWI prediction page.
- Handle exceptions and show error messages if prediction fails.
- Create / route to redirect to login page.
- Run the Flask app in debug mode.
- Use HTML templates for login and prediction pages with CSS styling and optional background video.

5.2 User testing

User Testing was performed to ensure that the application is intuitive, user-friendly, and functional for end-users. The main goal was to verify whether the FWI prediction system, along with the GUI, works as expected and provides correct results.

Process:

- Multiple users, including students and technical interns, were asked to interact with the application.
- Users provided various soil data inputs or other required parameters to test the FWI prediction functionality.
- Users observed how the interface handled data entry, prediction output, and error messages.

Key observations:

Users were able to navigate the GUI easily and input data without confusion.

Prediction results were displayed correctly and quickly after input submission.

Conclusion:

User testing confirmed that the application is usable, functional, and meets user expectations. Feedback helped in refining the interface and improving overall user experience.

5.3 Unit testing

- Unit Testing was performed to ensure that individual components of the project work correctly. In the internship project:
- Each Python function, including the FWI prediction algorithm, was tested separately with sample inputs to validate the correctness of outputs.
- Flask backend functions were tested independently to confirm they return accurate predictions and responses.

Outcome:

Unit testing helped in identifying small bugs early, such as incorrect data handling in lists or minor calculation errors in the prediction model.

5.4 Integration testing

Integration Testing ensures that all modules of the project work together correctly. In this project:

- The interaction between the Web Interface and Flask backend was tested to verify proper data flow.
- The prediction module was tested in combination with the user interface to ensure smooth operation.

Outcome:

Integration testing confirmed that all modules are well-connected, and data flows seamlessly from input to prediction output.

5.5 Acceptance testing

Acceptance Testing was performed to ensure that the project meets the objectives and requirements of the internship:

- The FWI prediction was verified against sample datasets to ensure accurate results.
- All frontend and backend functionalities were reviewed against the internship project guidelines.
- Users and mentors provided feedback confirming that the application fulfills the intended purpose.

Outcome:

Acceptance testing confirmed that the project is ready for deployment, meets functional requirements, and aligns with the learning objectives of the internship.

CHAPTER 6
CONCLUSION

6.1 Conclusion

This project successfully demonstrates the use of machine learning for predicting the Fire Weather Index. By integrating data analysis, model training, and web deployment, the system provides an effective solution for early fire risk assessment. The project highlights the practical skills gained during the Infosys Springboard internship and shows how AI and ML can be applied to real-world environmental problems

CHAPTER 7

OUTPUTS

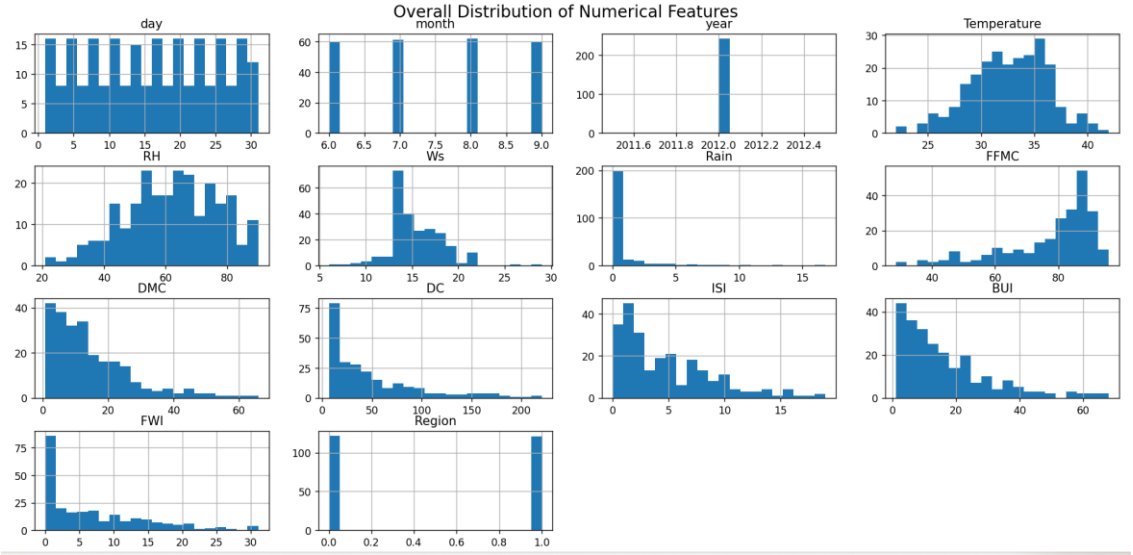


Fig 7.1 Histogram

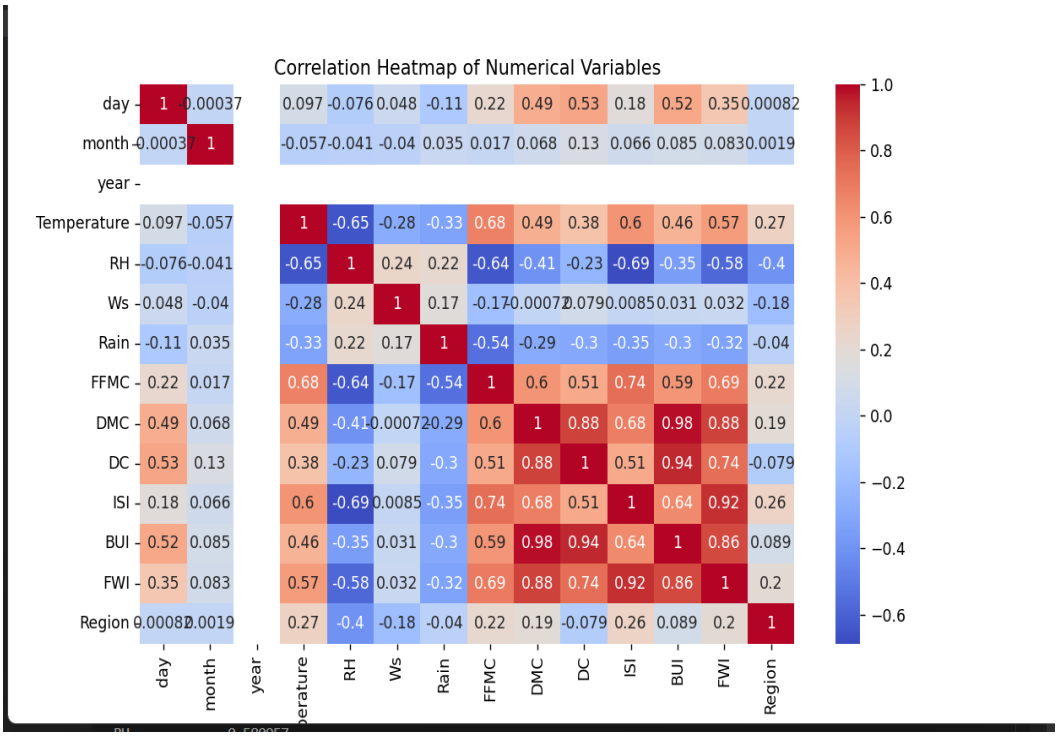


Fig 7.2 Heat Map

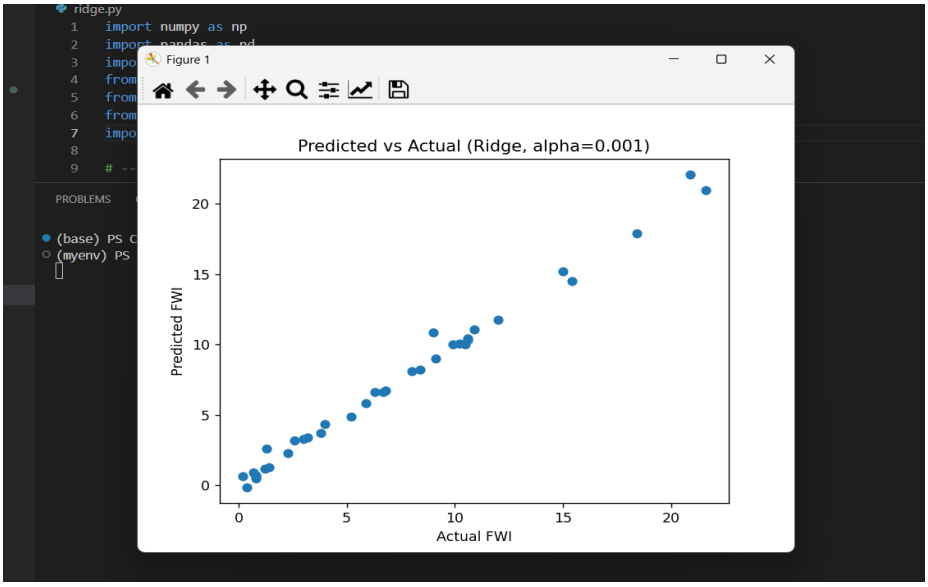


Fig 7.3 Actual vs Predicted

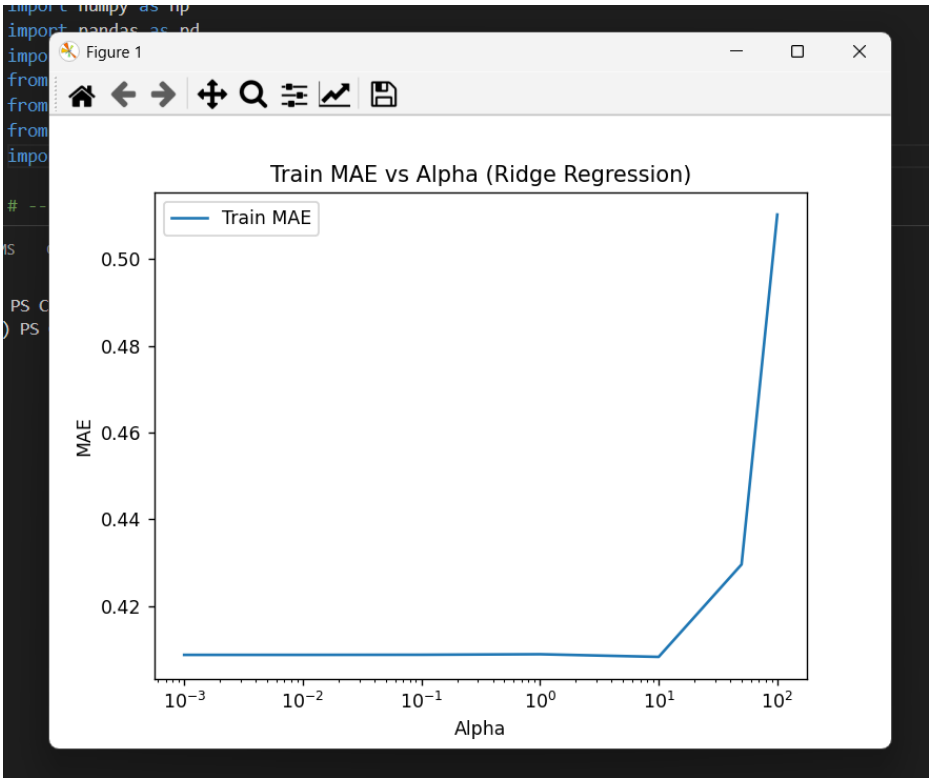


Fig 7.4 Train mae vs Alpha

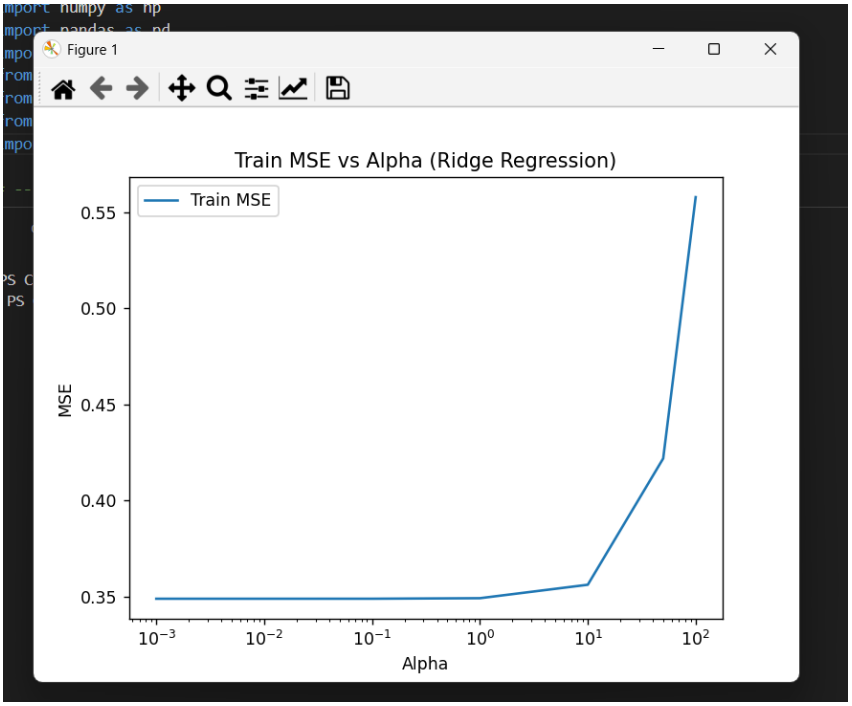


Fig 7.5 Train mse vs Alpha

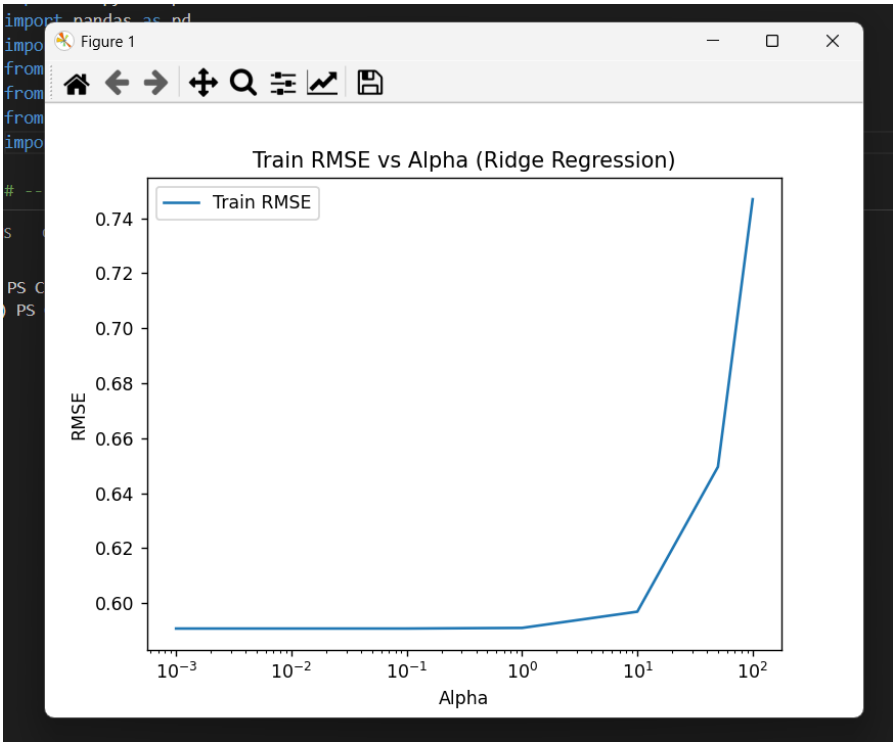


Fig 7.6 Train rmse vs Alpha

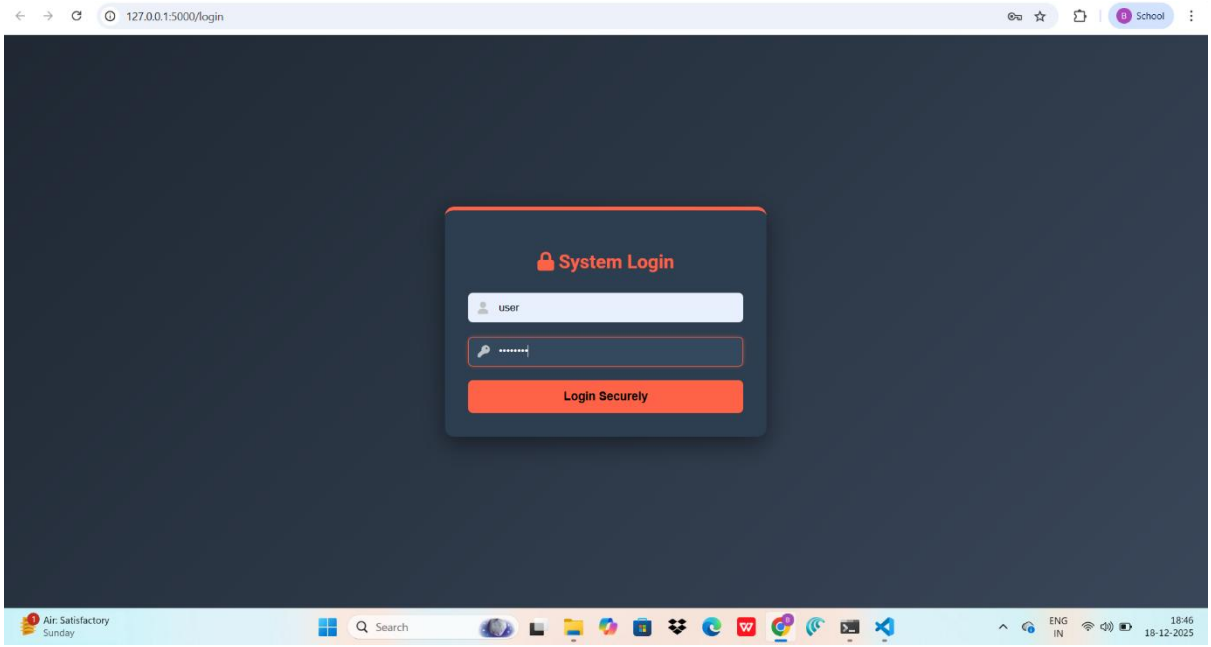


Fig 7.7 Login page

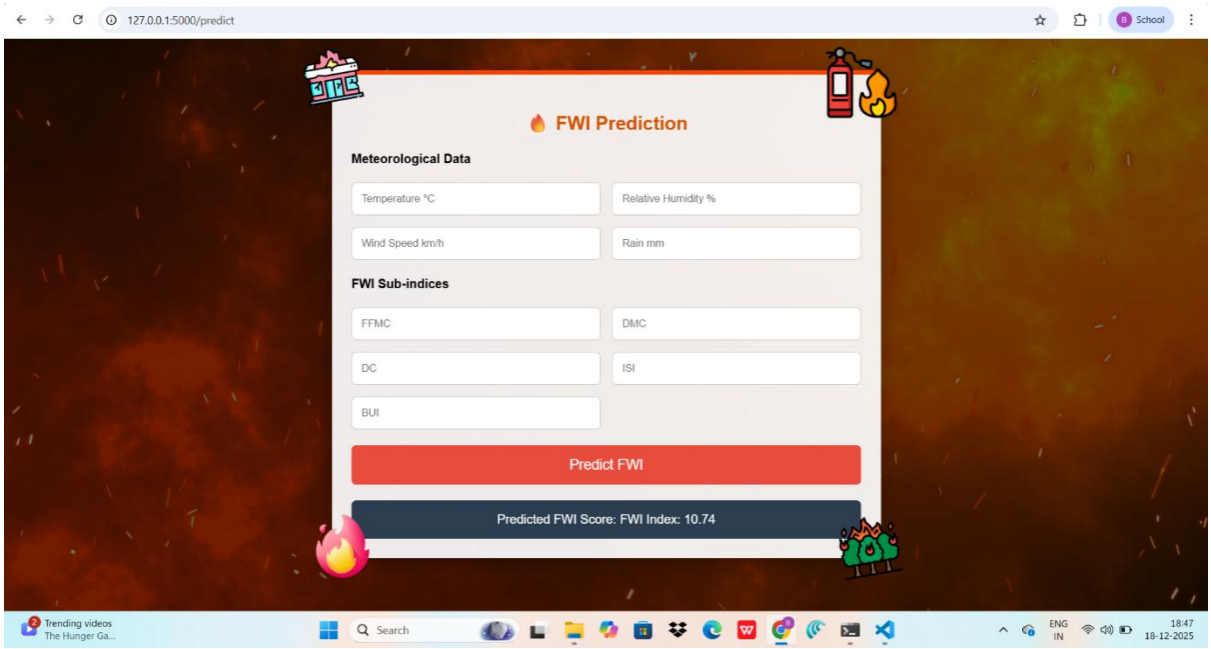


Fig 7.8 FWI prediction