Tempest FWI Predictor – A Machine Learning Model to Predict Fire Weather Index

Submitted by: Mishti Kalra

Date: 14 December 2025

Learnings Summary

I focused on deploying the trained Fire Weather Index (FWI) machine learning model using the Flask web framework. This phase emphasized backend–frontend integration, model loading, form handling, and dynamic result visualization, ensuring the solution is user-interactive and deployment-ready.

Module 6: Flask Application Development and Deployment

- Designed a clean and modular Flask application structure with **app.py** as the main execution file.
- Configured Flask routes to handle both GET and POST requests efficiently.
- Understood the importance of separating application logic, templates, and static resources.

1. Frontend Development Using HTML Templates

- Developed **index.html** to collect all required meteorological input features from users.
- Ensured proper form validation and structured input mapping for backend processing.
- Created **home.html** to display the predicted Fire Weather Index (FWI) in a clear and user-friendly format.
- Learned dynamic content rendering using Jinja2 templating.

2. Model Integration and Backend Processing

- Loaded the trained Ridge Regression model (**ridge.pkl**) during application runtime.
- Loaded the saved StandardScaler (**scaler.pkl**) to maintain consistency between training and inference.
- Implemented preprocessing logic to scale user inputs before prediction.
- Ensured seamless data flow from frontend form inputs to model inference.

3. End-to-End Prediction Workflow

- Handled user form submissions using Flask request objects.
- Converted input values into NumPy arrays compatible with the trained model.
- Generated real-time FWI predictions based on user inputs.
- Displayed results dynamically on the output page without manual refresh.

Conclusion

By completing the Flask deployment phase, I successfully transformed a trained machine learning model into a fully functional web application. This experience enhanced my understanding of Flask-based deployment, frontend–backend communication, model serialization, and real-time prediction delivery, making the FWI predictor suitable for practical and real-world usage.