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

Project Statement:

Wildfires pose a significant threat to ecosystems, human life, and property. The Fire Weather Index (FWI) is a crucial tool used by meteorological and environmental agencies worldwide to estimate wildfire potential. This project aims to build a machine learning model that predicts FWI based on real-time environmental data, enabling proactive wildfire risk management. The model is trained using Ridge Regression, deployed via a Flask web application, and supports early warning systems for wildfire hazards.

Outcomes:

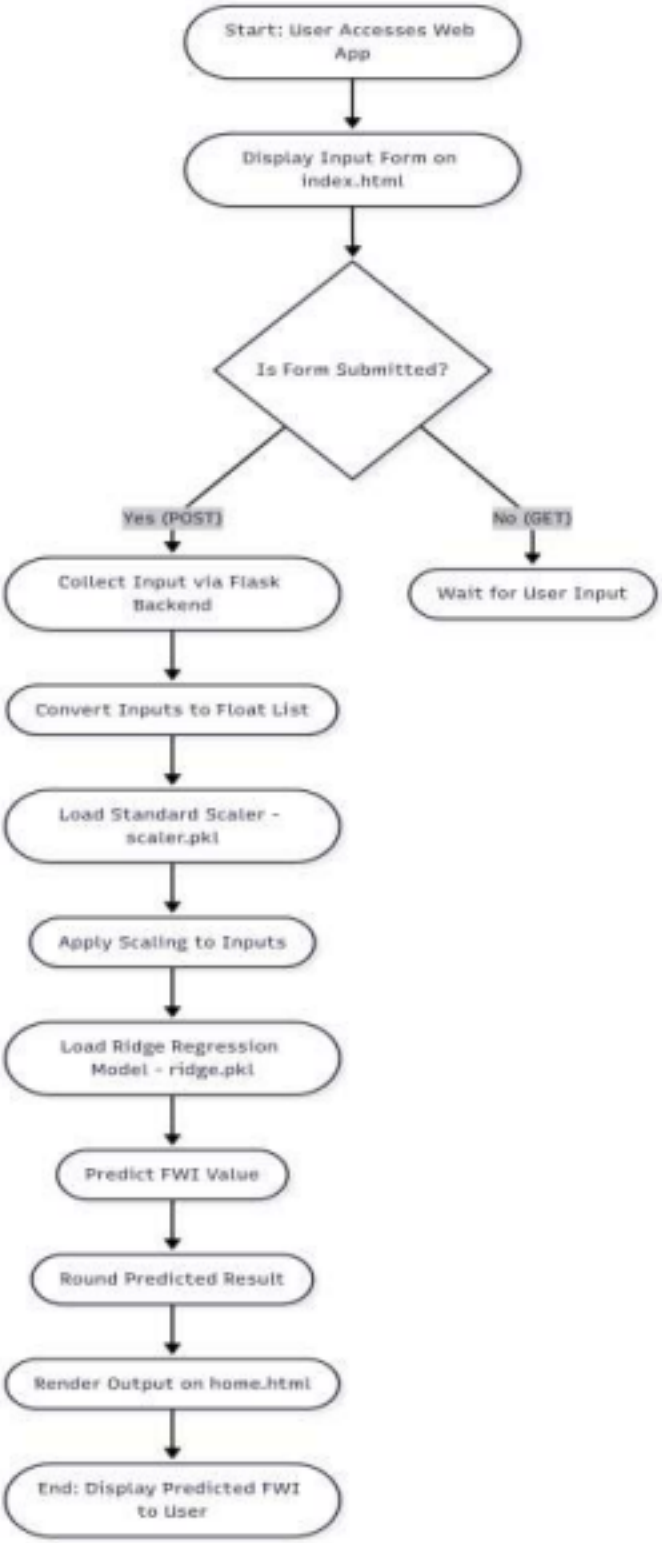
- A predictive ML model trained using Ridge Regression to forecast FWI.
- A pre-processing pipeline using StandardScaler for normalization.
- A Flask-based web app where users can input environmental values and get FWI predictions. • A system that can help forest departments, emergency planners, and climate researchers make data driven decisions.

Modules to be implemented

- Data Collection
- Data Exploration (EDA) and Data Preprocessing
- Feature Engineering and Scaling
- Model Training using Ridge Regression
- Evaluation and Optimization
- Deployment via Flask App
- Presentation and Documentation

SpringBoard Internship Program

End-to-End Workflow of the FWI Predictor



Week-wise Module Implementation and High-Level Requirements with Expected Outputs Milestone 1: Week 1-2 Module 1: Data Collection

• Collected a structured dataset containing relevant environmental features and the FWI target variable. • Ensured the dataset includes Temperature, Relative Humidity, Wind Speed, Rain, FFMC, DMC, ISI, and Region.

• Verified the data types, consistency, and proper formatting of the dataset.

• Loaded the dataset into a Pandas DataFrame for further analysis.

- Conducted initial inspection to understand feature distributions and data quality.

Module 2: Data Exploration and Data Preprocessing

- Checked for missing or null values and handled them appropriately.
- Performed outlier detection using boxplots and statistical thresholds.
- Visualized data distributions using histograms and density plots.
- Explored feature relationships using correlation matrix and scatterplots.
- Encoded categorical values like Region using label encoding or mapping.
- Saved the cleaned dataset for use in modeling.

Milestone 2: Week 3-4

Module 3: Feature Engineering and Scaling

- Selected key input features most correlated with the FWI target variable.
- Normalized numerical features using StandardScaler for consistent scale.
- Split the dataset into input features (X) and target variable (y).
- Separated data into training and testing sets using train_test_split.
- Ensured the scaler was saved as a .pkl file for deployment consistency.

Module 4: Model Training using Ridge Regression

- Trained a Ridge Regression model to handle multicollinearity in input data.
- Tuned the alpha parameter to balance bias-variance tradeoff.
- Evaluated training and validation performance during training.
- Saved the trained model using pickle as ridge.pkl.
- Documented the training process and parameters used.

Milestone 3: Week 5-6

Module 5: Evaluation and Optimization

- Evaluated the model using Mean Absolute Error (MAE).
- Computed Root Mean Squared Error (RMSE) to penalize large errors.
- Calculated R^2 Score to assess variance explanation.
- Plotted predicted vs actual values to visualize performance.
- Tuned model parameters (alpha) and retrained if needed to improve metrics.

Milestone 4: Week 7-8

Module 6: Deployment via Flask App

- Created a Flask app structure with app.py as the main application.
- Developed index.html for the user input form with all feature fields.
- Created home.html to display the predicted FWI value to the user.
- Loaded ridge.pkl and scaler.pkl during app runtime for predictions.
- Handled form input collection, preprocessing, prediction, and output display.

Module 7: Presentation and Documentation

- Prepared complete documentation including project summary, modules, and architecture. •

Created system architecture and workflow diagrams using draw.io or mermaid. • Captured screenshots of the web app interface and output pages.

• Described the end-to-end pipeline from input to prediction.

• Organized all files for submission, review, and GitHub publishing.

Evaluation Criteria:

Milestone 1 Evaluation (Week 2):

• Approval of the collected FWI dataset, ensuring appropriate feature variety and data quality. • Approval of exploratory data analysis outputs (e.g., histograms, correlations, feature distributions). • Approval of preprocessing steps including handling of missing values, outlier treatment, and encoding. • Approval of region encoding strategy and feature consistency across the datase

Milestone 2 Evaluation (Week 3-5):

• Approval of feature selection rationale based on correlation and domain relevance. • Approval of normalization and scaling techniques (e.g., StandardScaler) with documented justification. • Approval of train-test split and data partitioning strategy ensuring generalizability. • Approval of Ridge Regression model selection to address multicollinearity in weather features. • Approval of saved model (ridge.pkl) and scaler (scaler.pkl) with validation performance.

Milestone 3 Evaluation (Week 6-7):

• Approval of model performance metrics (MAE, RMSE, R^2 Score) and residual analysis. • Approval of model tuning (e.g., Ridge alpha parameter) and its impact on results. • Approval of visualization and interpretation of model predictions vs actual values. • Approval of model robustness on test data and documentation of evaluation findings.

Milestone 4 Evaluation (Week 7–8):

• Approval of fully functional Flask web application for real-time FWI prediction.

• Approval of form-based input design, user experience, and clarity of output results. • Approval of integration of model and scaler into the Flask backend for live inference. • Approval of thorough deployment documentation covering file structure, execution, and setup. • Approval of final project documentation, system architecture, and workflow diagrams. • Approval of final GitHub submission or designated code repository ensuring reproducibility and clarity.