A DOCUMENTATION ON

FIRE WEATHER INDEX PREDICTOR

Project Duration

DEC 2025 – JAN 2026

Project Mentor

PRAVEEN

PROJECT SUBMITTED BY

Pranjali Kolawale

Infosys Springboard Internship Program

INDEX

FIGURES

FIRE WEATHER INDEX PREDICTOR

INTRODUCTION

Wildfires have become more common in recent years and can cause serious damage to forests, animals, houses, and even people. Changes in climate, such as higher temperatures and uneven rainfall, make it easier for fires to start and spread. Because of this, it is important to know the fire danger level in advance so that forest officials and emergency teams can prepare and take action on time.

The Fire Weather Index (FWI) is a measure used to understand how risky the weather conditions are for a wildfire to occur. It depends on factors like temperature, humidity, wind speed, rainfall, and the dryness of forest fuel. Checking all these conditions manually can be slow and may not always give accurate predictions.

To solve this problem, this project aims to build a machine learning model that can predict the FWI using environmental data. By analysing the dataset and creating a predictive system, the project helps in giving early warnings and improving wildfire risk management.

PROBLEM STATEMENT

Wildfires can spread quickly and cause heavy damage, so it is important to know the fire risk in advance. The Fire Weather Index (FWI) helps measure how likely a fire is to start, but calculating it manually is not always accurate or fast. To solve this, the aim of this project is to use environmental factors like temperature, humidity, wind speed, and rainfall to build a machine learning model that can predict the FWI automatically. This will help in giving early warnings and improving wildfire preparedness.

OBJECTIVES

- To study the weather factors that affect the Fire Weather Index (FWI).
- To clean and prepare the dataset for accurate analysis.
- To explore the data and understand its patterns.
- To build a machine learning model that can predict FWI.
- To test the model and improve its accuracy.
- To create a simple web app for making FWI predictions.

MODULE 1: DATA COLLECTION AND INITIAL DATA PREPARATION

1.1 Loading Dataset and Explored Structure

Collected the Fire Weather Index (FWI) Dataset from the Kaggle. Then dataset was imported into a Pandas DataFrame. Basic exploration was done by checking the first few rows, the shape of the dataset, column names, and data types. This helped in understanding the kind of information available, such as weather parameters (*Temperature*, *RH*, *Ws*, *Rain*) and FWI-related indices (*FFMC*, *DMC*, *DC*, *ISI*, *BUI*, *FWI*), along with the *Classes* column representing fire risk levels.

1.2 Identifying Missing Values and Incorrect Values

The dataset was checked for missing entries and inconsistencies. Some columns, particularly *Classes*, *FWI*, and *DC*, contained missing or misformatted values, like spaces within numbers or text-based numeric entries. Detecting these issues early is important to ensure accurate analysis and modeling later.

```
missing_row = df[df.isnull().any(axis=1)]

print(missing_row)
```

1.3 Cleaning Column Names and Formats

Column names were standardized by removing extra spaces. Numeric columns that had text or unwanted spaces, such as *DC*, were cleaned by stripping spaces and converting them into proper numeric types. This step ensures that all calculations and analyses will be accurate and avoids errors during preprocessing.

```
df.columns = df.columns.str.strip()

df['DC'] = df['DC'].astype(str).str.strip()

df['DC'] = df['DC'].str.replace(' ', '')

df['DC'] = df['DC'].astype(float)
```

1.4 Handling Missing Values

Different strategies were applied based on column type. For the categorical *Classes* column, missing values were filled with the mode, as it represents the most common fire risk category. For numeric columns like *FWI*, invalid entries were replaced with the mean of the column after conversion to numeric. This ensures that the dataset remains complete and ready for exploratory analysis.

```
df['Classes'] = df['Classes'].fillna(df['Classes'].mode()[0])
```

1.5 Validating Data After Cleaning

After performing cleaning and handling missing values, the dataset was reviewed again using info() and sample rows. This confirmed that all corrections were applied properly, column types were consistent, and the dataset was ready for further analysis. This step helps prevent issues in downstream tasks like EDA or modeling.

```
print(df.info())
```

MODULE 2: DATA EXPLORATION AND PREPROCESSING

After collecting and cleaning the FWI dataset in Module 1, the next step was to explore the data and perform preprocessing to ensure it is ready for modeling. This module focused on understanding the data distribution, identifying patterns, handling categorical features, and preparing the dataset for machine learning.

2.1 Checking for Duplicates and Data Overview

The dataset was first examined for duplicate records, which could skew the analysis. Fortunately, no duplicates were found, confirming that each record was unique. Descriptive statistics were generated for numerical features to understand their distribution, central tendency, and variability. This helped in identifying potential anomalies, extreme values, or inconsistencies.

2.2 Handling Categorical Features

The dataset contained the categorical feature **Region**, which was converted into numerical values using label encoding. This transformation allowed the model to interpret region information effectively while maintaining the distinction between different regions.

```
df['Region'] = LabelEncoder().fit_transform(df['Region'])
```

2.3 Visualizing Data Distribution

Visualization played an important role in understanding the dataset:

- **Boxplots** provided insights into variations and outliers, particularly across regions, helping identify extreme values in feature like Temperature.

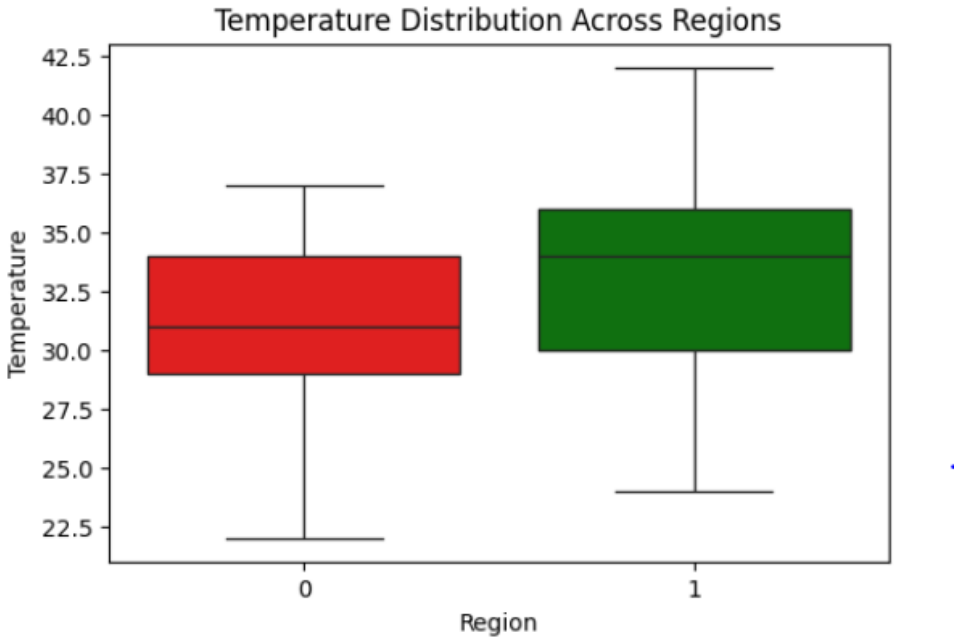

Figure 1

To understand how temperature values were spread across the dataset, I created a boxplot, which helped me quickly spot any unusual readings. After this, I applied the IQR method to calculate outliers. The first quartile (Q1) was 30.0 and the third quartile (Q3) was 35.0, giving an IQR of 5.0. Using these values, I calculated the lower limit as 22.5°C and the upper limit as 42.5°C. Any temperature outside this range is considered an outlier. Based on this, I found two outliers in the dataset, both with a temperature of 22°C at index numbers 93 and 105, which fall below the lower limit and indicate unusually low temperature values.

- **Histograms** were used to examine the distribution of numerical features in the data.

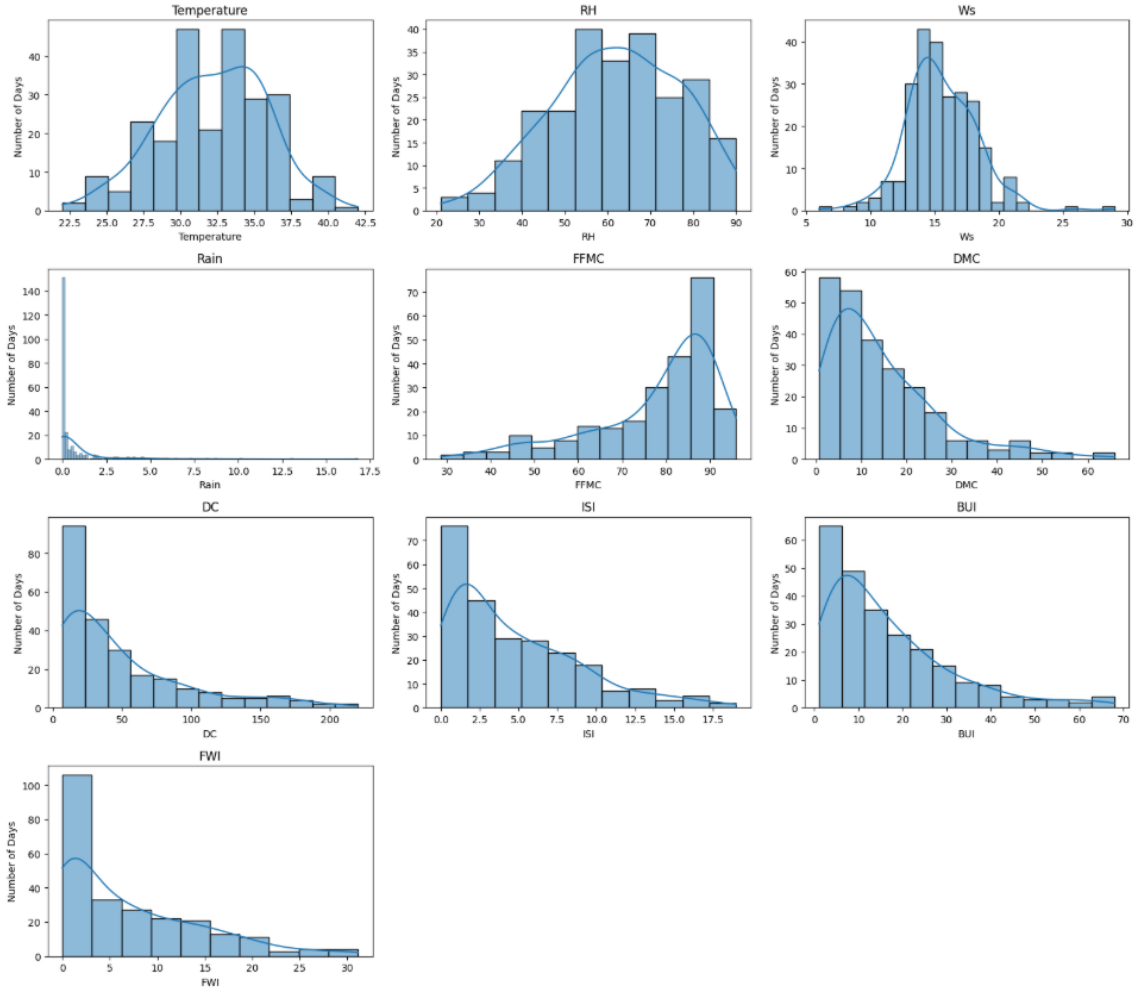

Figure 2

This figure shows histograms with smooth curves for all the main variables in the dataset, such as Temperature, Humidity, Wind Speed, Rainfall, and different fire-risk indexes (FFMC, DMC, DC, ISI, BUI, FWI). These plots help us understand how the data is spread out. Most variables like Rain, DC, ISI, BUI, and FWI are right-skewed, which means high values occur only on a few days. Temperature and RH appear more balanced, showing regular day-to-day patterns. Overall, the figure gives a clear idea of the typical value ranges, variations, and the occasional extreme conditions that can affect fire weather.

- **Scatter plots** give a visual idea of how different weather factors relate to the Fire Weather Index (FWI). They help us understand which variables have stronger or weaker influence on fire risk.

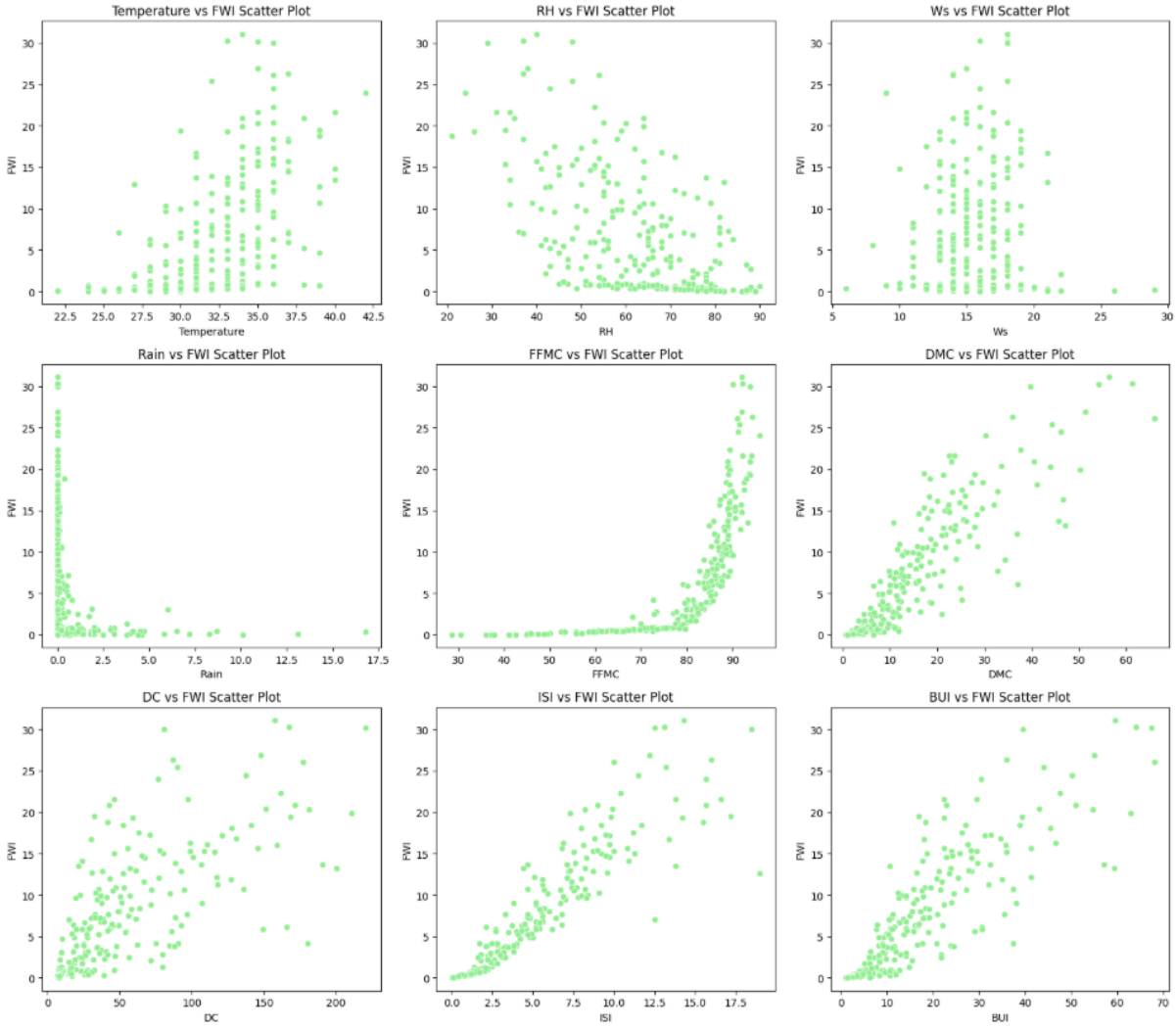

Figure 3

The scatter plots clearly show that temperature has a direct connection with FWI, as higher temperatures usually lead to an increase in fire risk. Relative humidity mostly stays linked with low FWI values, meaning more moisture in the air helps reduce the possibility of fire. Wind speed is spread out but still shows that stronger winds can slightly raise FWI. Rain has the opposite effect, as days with rainfall almost always show very low FWI. The fuel-related indices like FFMC, DMC, DC, ISI, and BUI display a strong upward trend, indicating that when these dryness and fuel buildup values rise, FWI also increases. Overall, the plots make it clear that dry weather, high temperature, low humidity, and accumulated dry fuel together lead to a higher chance of fire.

- **Correlation heatmaps** were generated to study relationships between features and the target variable FWI. Features showing strong correlation with FWI were noted as important for model input.

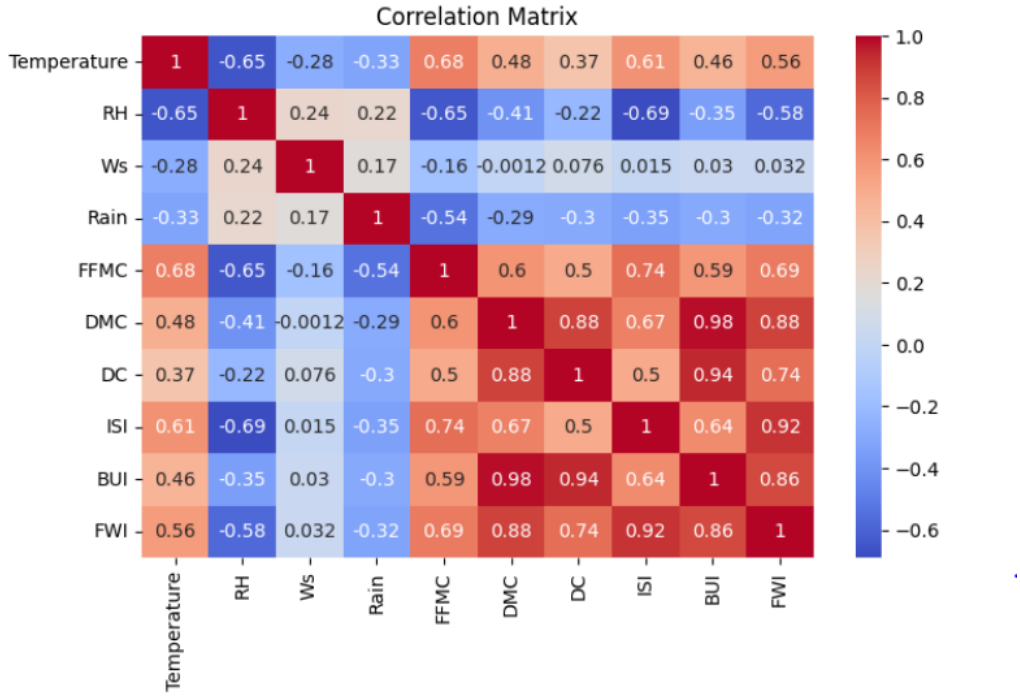

Figure 4

This correlation matrix shows how the main weather and fire-risk variables are related to each other. Fire-related indexes like DMC, DC, ISI, BUI, and FWI have strong positive correlations, meaning they increase together. Temperature also shows positive links with some fire indexes. In contrast, Relative Humidity has strong negative correlations with Temperature and several fire-risk variables. Rainfall has weak negative relationships overall. This figure helps identify which factors influence fire-weather conditions the most.

2.4 Preparing the Dataset

After visual exploration, unnecessary columns, such as **Classes**, were dropped since they were not relevant for FWI prediction. All numerical features were checked for consistency and corrected if needed, ensuring that the dataset was clean and formatted correctly for subsequent modeling.

```
df = df.drop('Classes', axis=1)
```

2.5 Saving the Processed Dataset

Finally, the fully cleaned and preprocessed dataset was saved as a new file. This ensured that the dataset could be easily used in the next modules for feature engineering, scaling, and model training.

```
df.to_csv("./datasets/FWI_Dataset_Cleaned.csv", index=False)
```

MODULE 3: FEATURE SELECTION AND SCALING

In this module, we prepared the FWI dataset for model training. Important features like temperature, humidity, wind, rain, and fire-fuel indices were selected as inputs, and FWI was set as the target. The data was split into training and testing sets, and all features were scaled to the same range to make the model training more effective. The scaling method was saved to use later during model deployment.

3.1 Importing Libraries and Loading Cleaned Dataset

First, we imported the necessary Python libraries for data handling, preprocessing, and saving objects. The cleaned Fire Weather Index (FWI) dataset was then loaded into a Pandas DataFrame from a CSV file, making it ready for further processing.

3.2 Feature Selection and Target Variable

In this step, we selected the input features that influence fire risk and the target variable for prediction. The input features include meteorological parameters such as Temperature, Relative Humidity (RH), Wind Speed (Ws), Rain, and fire-fuel indices like FFMC, DMC, DC, ISI, and BUI. The Fire Weather Index (FWI) was set as the target variable.

```
X = df[['Temperature', 'RH', 'Ws', 'Rain',
        'FFMC', 'DMC', 'DC', 'ISI', 'BUI']]
y = df['FWI']
```

3.3 Splitting the Dataset

The dataset was split into training and testing sets to evaluate the model's performance. 80% of the data was used for training, and 20% was kept aside for testing. A fixed random state was used to ensure that the split can be reproduced in future runs.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

3.4 Feature Scaling

The dataset was split into training and testing sets to evaluate the model's performance. 80% of the data was used for training, and 20% was kept aside for testing. A fixed random state was used to ensure that the split can be reproduced in future runs.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3.5 Saving the Scalar

The fitted StandardScaler was saved using pickle to ensure that the same scaling can be applied to new data during model deployment or future predictions. This helps maintain consistency in the input data for the trained model.

```
with open("scaler.pkl", "wb") as f:
    pickle.dump(scaler, f)
```

MODULE 4: MODEL TRAINING AND EVALUATION

Since the objective of this project is to **predict the Fire Weather Index (FWI)**, which is a continuous numerical value, regression-based machine learning models were selected. The following models were used to compare prediction performance and best model will choose for final predictions.

4.1 Selecting Regression Models

- **Linear Regression**
  Linear Regression was used to model the direct relationship between meteorological features and the Fire Weather Index. It serves as a baseline model for understanding how well simple linear relationships can predict fire risk.
- **Ridge Regression**
  Ridge Regression was chosen to handle multicollinearity among weather and fire-fuel features. By applying regularization, it helps improve prediction accuracy and reduces overfitting.
- **Lasso Regression**
  Lasso Regression was used to identify the most important features affecting FWI by shrinking less important feature coefficients to zero. This helps in simplifying the model while maintaining good performance.
- **ElasticNet Regression**
  ElasticNet Regression combines the strengths of both Ridge and Lasso regression. It is effective when multiple input features are correlated and provides a balanced regularization approach.

```
models = {
    "Linear Regression": LinearRegression(),
    "Ridge Regression": Ridge(alpha=1.0),
    "Lasso Regression": Lasso(alpha=0.01),
    "ElasticNet Regression":ElasticNet(alpha=0.01,l1_ratio=0.5)
}
```

4.2 Model Training

In this step, all selected regression models were trained using the scaled training data. After training, each model was used to predict the Fire Weather Index (FWI) values on the test dataset. This helped in understanding how well each model learns the relationship between the input features and the target variable.

```
results = []
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
```

4.3 Model Evaluation

After generating predictions, the performance of each regression model was evaluated using standard evaluation metrics. Mean Absolute Error (MAE) was used to measure the average prediction error, Root Mean Squared Error (RMSE) was used to penalize larger errors, and the R^2 score was used to understand how well the model explains the variation in Fire Weather Index values. These metrics helped in comparing the accuracy and reliability of different models.

```
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
```

Model	MAE	RMSE	R2 Score
Linear Regression	0.500693	0.673129	0.988375
Ridge Regression	0.507991	0.685902	0.987929
Lasso Regression	0.498853	0.678988	0.988172
ElasticNet Regression	0.510224	0.688238	0.987847

4.4 Final Model Selection and Training

All the regression models were evaluated using MAE, RMSE, and R^2 score to compare their prediction performance. The results showed that Linear Regression, Ridge, Lasso, and ElasticNet achieved very similar accuracy with high R^2 values, indicating strong prediction capability for Fire Weather Index (FWI). Even though some models showed slightly lower error values, Ridge Regression was chosen as the final model because it provides more stable and reliable predictions. It handles correlated input features effectively through regularization, which reduces the chances of overfitting and improves generalization on unseen data.

```
final_model = Ridge(alpha=1.0)
final_model.fit(X_train_scaled, y_train)
```

4.5 Saving the Final Model

After selecting Ridge Regression as the final model, it was trained on the scaled training data and saved using the pickle library. Saving the model allows it to be reused later for deployment or future predictions without retraining. This ensures consistent and efficient prediction of Fire Weather Index values.

```
with open("ridge.pkl", "wb") as f:
    pickle.dump(final_model, f)
```