

# **Fire Weather Index Predictor**

(A Machine Learning Model to predict Fire Weather index)



*Submitted by*

**Devinandhana M S**

Under the Guidance of Mentor **Praveen**

## **1. Abstract**

The Tempest FWI Predictor is a machine learning model that estimates the Fire Weather Index, FWI, using environmental features such as temperature, humidity, wind speed, rainfall, and FWI sub-indices: FFMC, DMC, and ISI. The purpose of the project is to support Wildfire Early Detection Systems used by Forest Departments and Environmental Agencies. The model uses Supervised Learning Regression and is deployed through a Flask web application. This document covers the workflow of the project, including data collection, preprocessing, model training, evaluation, and deployment.

## **2. Introduction**

Forest fires have become increasingly common due to rising temperatures and changing climatic conditions. Predicting the Fire Weather Index (FWI) is essential for assessing fire danger and enabling early preventive actions. The FWI is calculated using key weather factors such as temperature, humidity, wind speed, rainfall, and fire danger codes.

This project aims to develop a machine learning-based system that predicts the FWI using historical fire weather data. The work includes cleaning the dataset, analyzing variable relationships, encoding features, and training a prediction model. By leveraging data-driven analysis, the project supports more accurate and efficient fire risk assessment compared to traditional manual methods.

## **3. Problem Statement**

The goal of this project is to build an accurate machine learning model that predicts the Fire Weather Index (FWI) using meteorological and fire-related variables such as temperature, humidity, wind speed, rainfall, and various danger codes. This includes cleaning the dataset, encoding categorical features, analyzing correlations, and training a prediction model that can support forest fire risk assessment and early warning systems.

## **4. Objective**

- Build an ML model to predict Fire Weather Index (FWI)
- Train Regression model using environmental features
- Build a Flask web app for real-time prediction
- Enable early wildfire detection and risk assessment

## **5. Dataset**

### **Description**

Dataset Name: **FWI\_Dataset.csv**

Source: kaggle

Rows: 245

Columns: 15

Features:

- Temperature – Celsius
- RH – Relative Humidity
- Ws – Wind Speed (km/h)
- Rain – mm
- FFMC – Fine Fuel Moisture Code
- DMC – Duff Moisture Code
- DC – Drought Code
- ISI – Initial Spread Index
- BUI – Buildup Index
- FWI – Target variable
- Classes – Fire intensity category
- Region – geographical area

## **6.Module 1 — Data Collection & Exploration**

Module 1 focuses on collecting the forest fire dataset and preparing it for analysis. This includes loading the data, inspecting its structure, cleaning column names, handling missing values, converting data types, and removing unnecessary attributes. The goal of this module is to ensure that the dataset is clean, consistent, and ready for further exploration and modeling. All preprocessing tasks such as handling categorical values, fixing numeric columns, and saving the cleaned dataset are completed in this stage.

### **Code:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
df = pd.read_csv("data/FWI_Dataset.csv")
print("First 5 rows:")
display(df.head())
print("\nDataset Info:")
display(df.info())
```

```

print("\nMissing Values:")
print(df.isnull().sum())
df = df.dropna()

```

### **Output:**

First 5 rows:																
	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes	Region	
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	not fire	Bejaia	
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	not fire	Bejaia	
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	not fire	Bejaia	
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0	not fire	Bejaia	
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	not fire	Bejaia	

```

None

Missing Values:
day          0
month        0
year          0
Temperature   0
RH            0
Ws            0
Rain          0
FFMC          0
DMC           0
DC            0
ISI           0
BUI           0
FWI           0
Classes       1
Region        0
dtype: int64

```

### **Code:**

```

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

print("Cleaned Columns:", df.columns.tolist())

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

df.drop(columns=['Classes'], inplace=True)

print("Missing values after cleaning:")

```

```
print(df.isnull().sum())
df.head()
```

## Output:

Cleaned Columns: ['day', 'month', 'year', 'Temperature', 'RH', 'Ws', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI', 'FWI', 'Classes', 'Region']

Missing values after cleaning:

day	0
month	0
year	0
Temperature	0
RH	0
Ws	0
Rain	0
FFMC	0
DMC	0
DC	0
ISI	0
BUI	0
FWI	0
Region	0

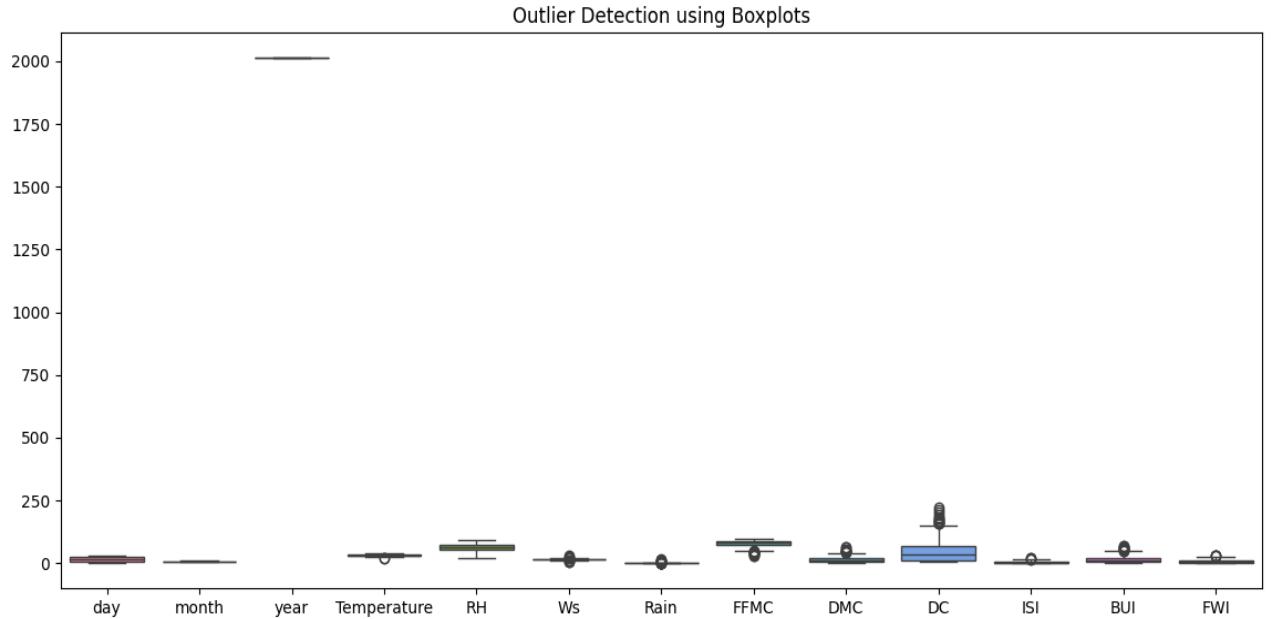
*dtype: int64*

## 7. Module 2 — Data Preprocessing

Module 2 involves analyzing the cleaned dataset to understand patterns, distributions, and relationships among variables. This includes generating statistical summaries, correlation matrices, scatter plots, density plots, boxplots, and pair plots. EDA helps identify trends, outliers, and feature interactions that influence the Fire Weather Index (FWI). The insights gained in this module guide the selection of relevant features and support informed decisions for building the machine learning model.

## I. Box Plot (Outlier Detection)

Box plots were used to detect outliers and understand the distribution of numerical features. They highlight the spread of data and identify extreme values that could impact model performance.



### #Removal of the outliers

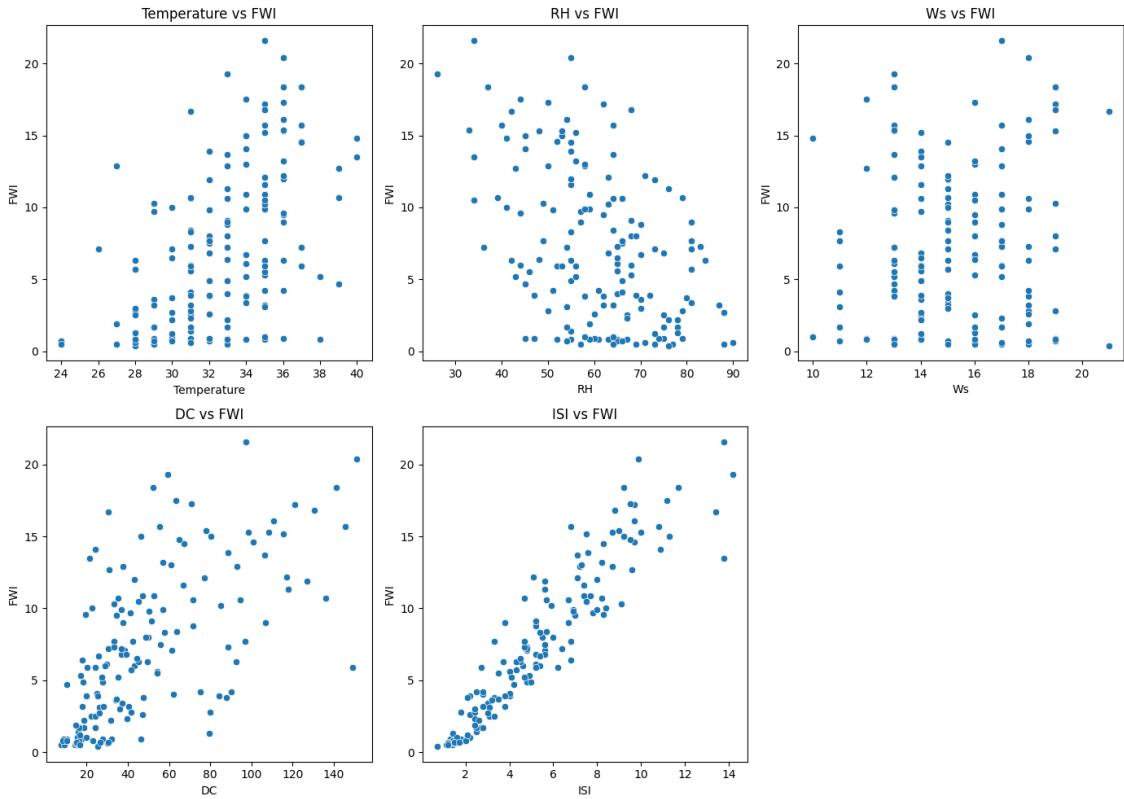
```
numeric_cols = df.select_dtypes(include=[np.number]).columns
for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower) & (df[col] <= upper)]
df = df.reset_index(drop=True)
print("Shape after outlier removal:", df.shape)
```

### Output:

Shape after outlier removal: (155, 14)

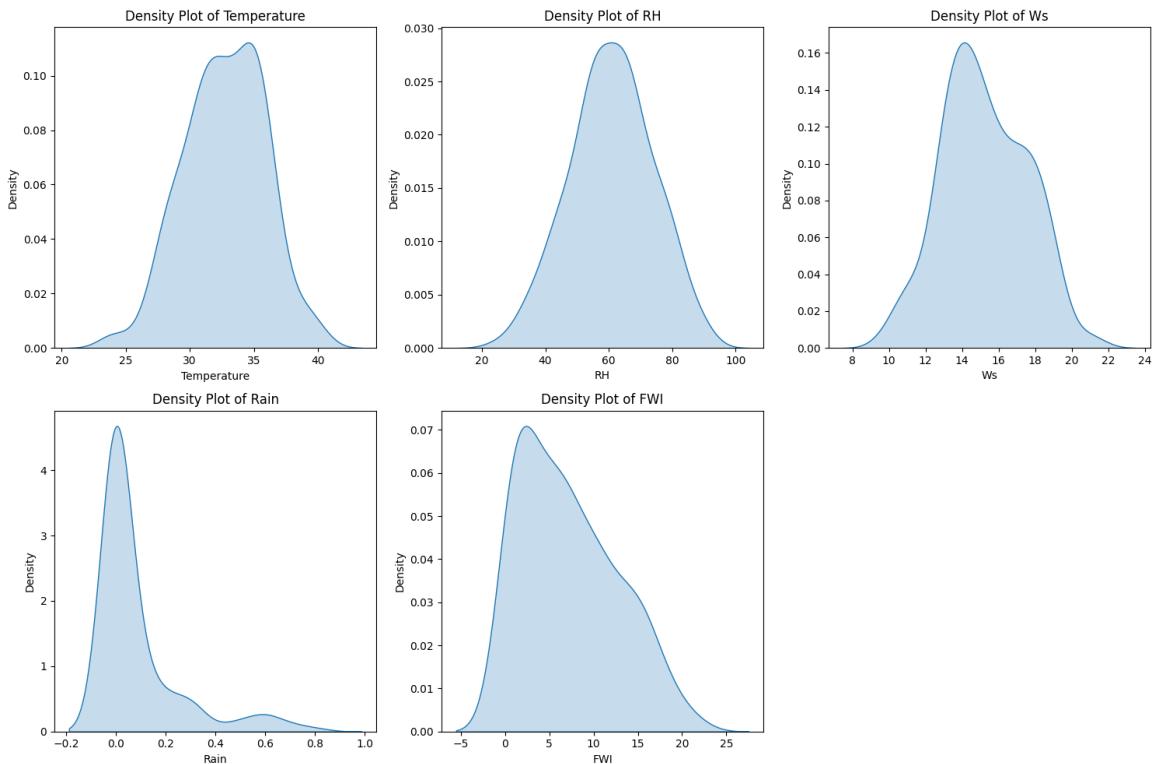
## II. Scatter Plot

Scatter plots illustrate the relationship between individual meteorological features and the Fire Weather Index. They help analyze how changes in weather conditions influence fire risk levels.



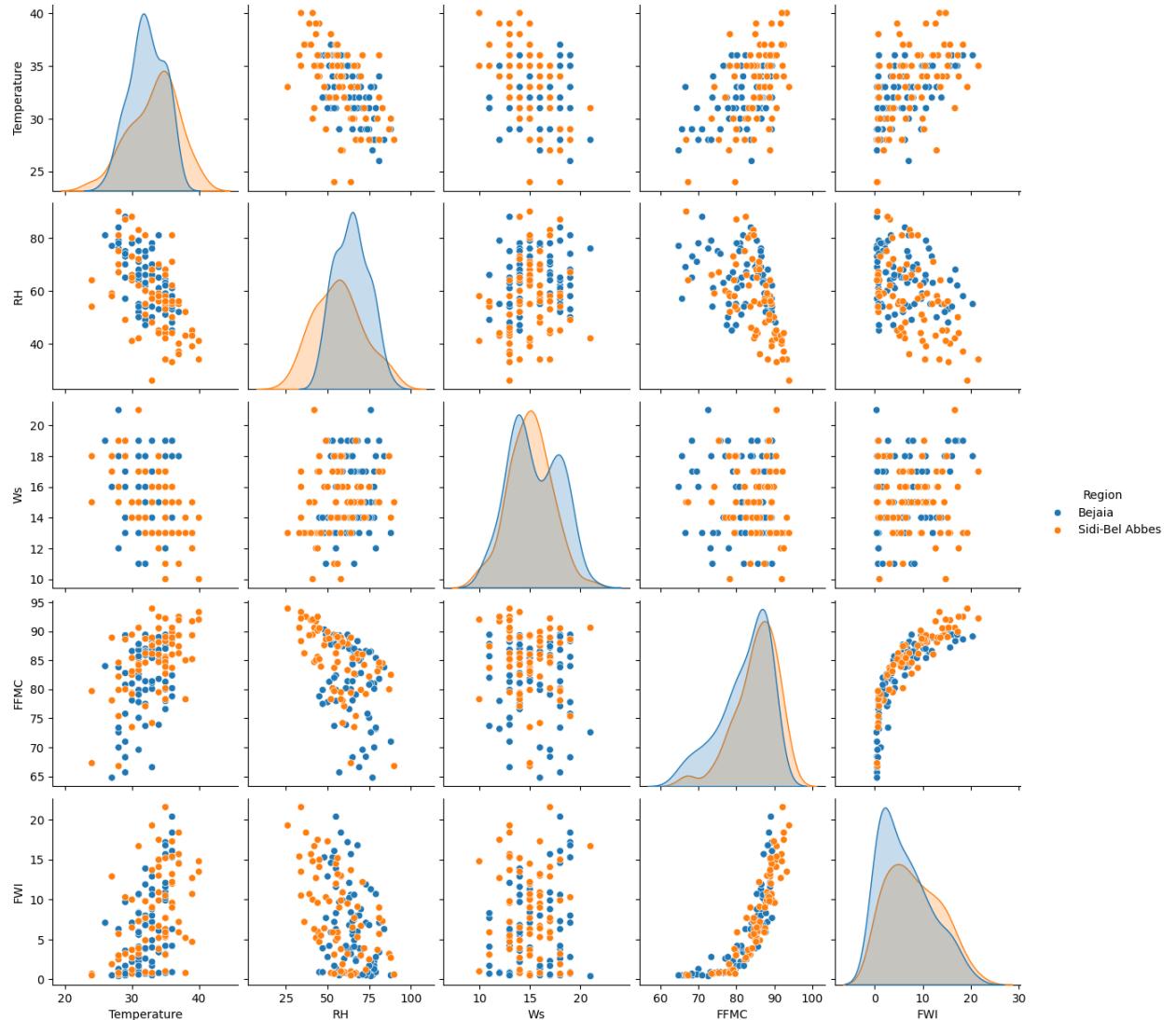
### III. Density Plot

Density plots show the probability distribution of continuous variables. They help visualize how frequently different values occur and whether the data follows a normal distribution.



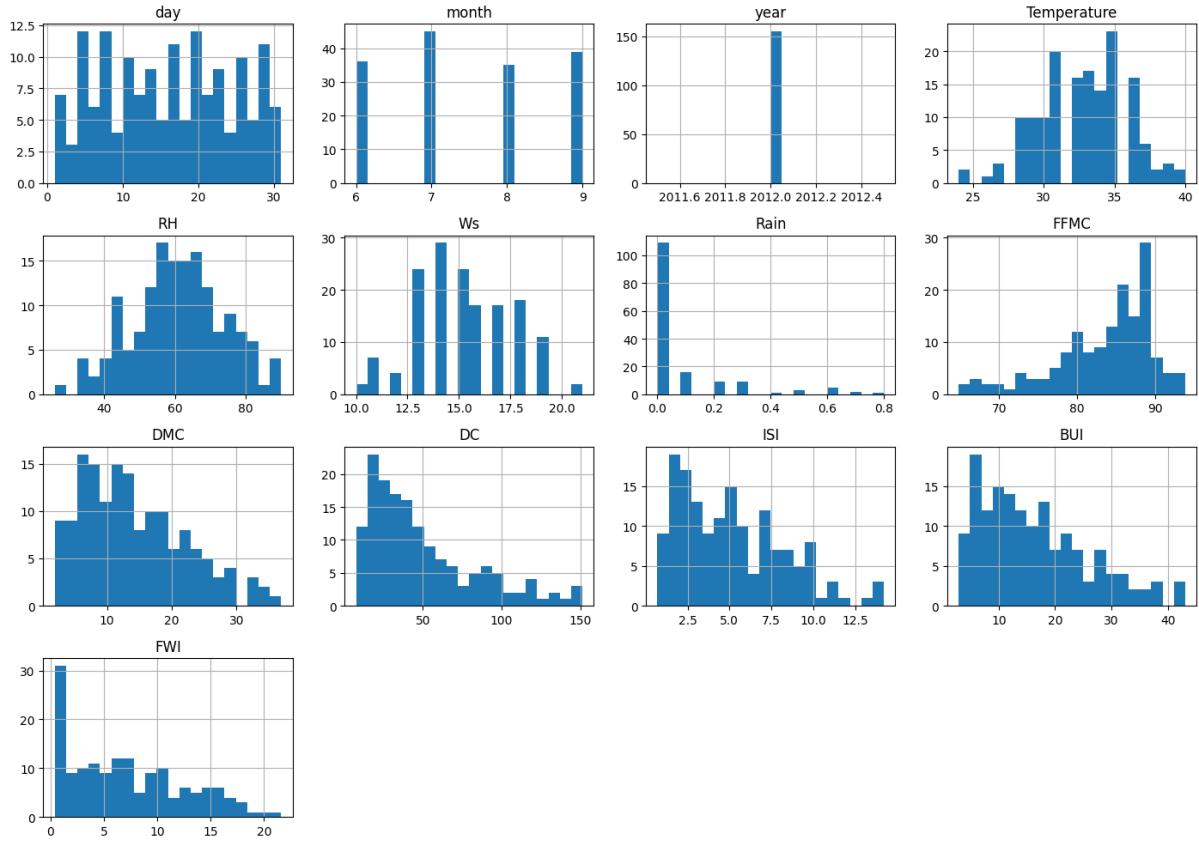
## IV. Pair plot

The pair plot shows pairwise relationships between selected numerical features and the target variable. It provides insights into linear trends, distributions, and potential interactions between variables affecting the Fire Weather Index.



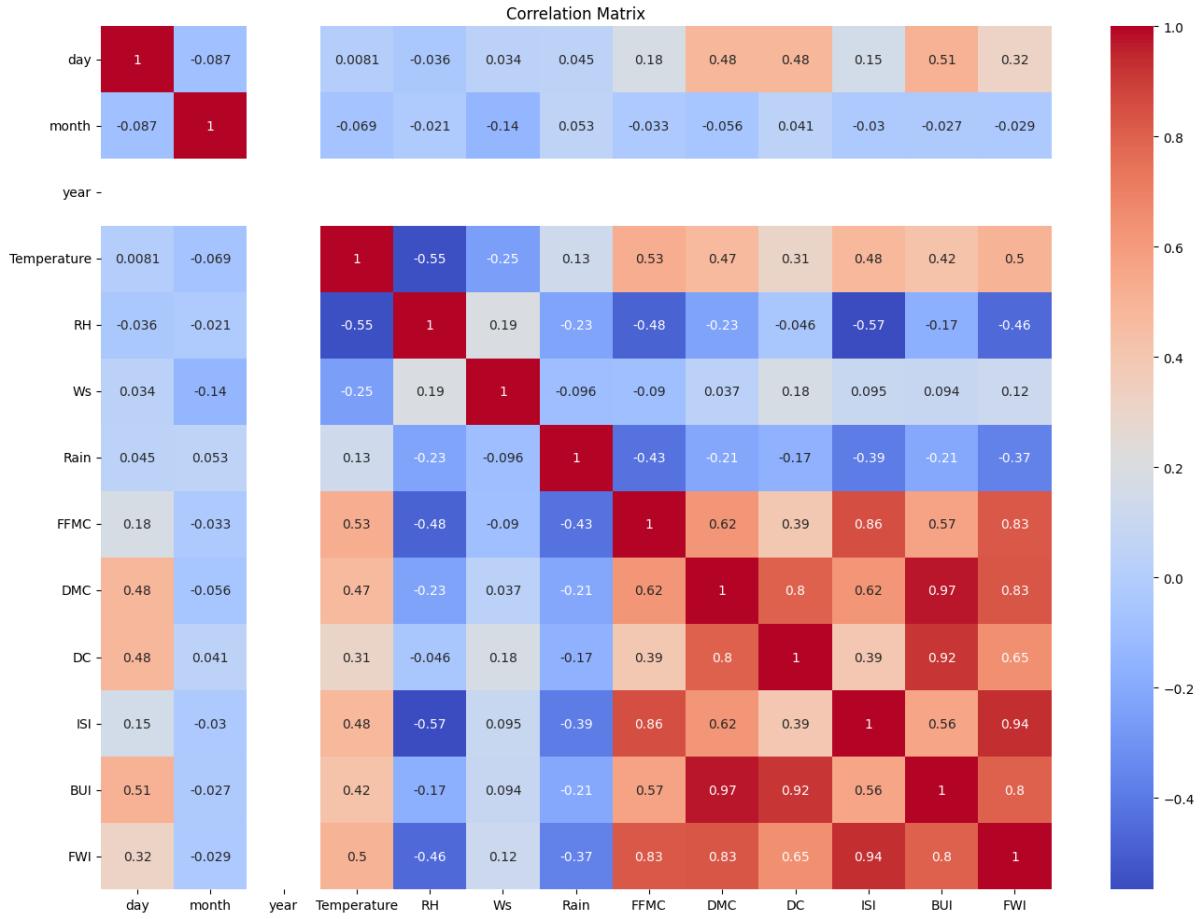
## V. Histogram

A histogram represents the frequency distribution of numerical features in the dataset. It shows how values are spread across different ranges and helps understand the overall distribution pattern of meteorological variables and the Fire Weather Index.



## VI. Correlation Matrix

The correlation matrix visualizes the strength and direction of relationships between numerical features in the dataset. It helps identify which meteorological variables have a strong influence on the Fire Weather Index (FWI). Features such as FFMC, DMC, DC, ISI, and BUI show strong positive correlation with FWI, indicating their importance in fire risk prediction.



## Code: (Region Encoding)

```

if 'Region' in df.columns:
    le_region = LabelEncoder()
    df['Region'] =
    le_region.fit_transform(df['Region'].astype(str))
    print("Region column encoded!")

else:
    print("Region column not found!")

df.to_csv("data/preprocessed_fwi.csv", index=False)
print("Saved as preprocessed_fwi.csv")

```

## Output:

Region column encoded!

Saved as preprocessed\_fwi.csv

## **8. Module 3-Feature Engineering & Scaling**

In this module, relevant features were selected based on domain relevance. The dataset was divided into input features and target variable (FWI). Data was split into training and testing sets to ensure generalization. Numerical features were normalized using StandardScaler to bring them to a common scale. The trained scaler was saved to maintain consistency during model deployment.

### **Code:**

```
corr = df.corr(numeric_only=True) ['FWI'].abs()

selected_features = corr[corr > 0.3].index.tolist()

selected_features.remove('FWI')

if 'day' in selected_features:

    selected_features.remove('day')

if 'Ws' not in selected_features:

    selected_features.append('Ws')

X = df[selected_features]

y = df['FWI']

print("selected features:")

print(X.columns.tolist())

selected features:
['Temperature', 'RH', 'Rain', 'FFMC', 'DMC', 'DC', 'ISI', 'BUI', 'Ws']

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

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

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

    pickle.dump(scaler, f)

print("Training data shape:", X_train_scaled.shape)
print("Testing data shape:", X_test_scaled.shape)
```

```
Training data shape: (124, 9)
Testing data shape: (31, 9)
```

## **9.Module 4- Model Training using Ridge Regression**

In this module, a Ridge Regression model was trained using standardized environmental features to predict the Fire Weather Index (FWI). Ridge Regression was selected to address multicollinearity among weather variables by applying L2 regularization. The model was trained on scaled training data, evaluated using MAE, RMSE, and R<sup>2</sup> metrics, and saved as a pickle file (ridge.pkl) for deployment in the Flask web application

### **Code:**

```
ridge_model = Ridge(alpha=1.0)

ridge_model.fit(X_train_scaled, y_train)

y_pred_ridge= ridge_model.predict(X_test_scaled)

ridge_mae = mean_absolute_error(y_test, y_pred_ridge)
ridge_rmse = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
ridge_r2 = r2_score(y_test, y_pred_ridge)

print("Mean Absolute Error (MAE) :", ridge_mae)
print("Root Mean Squared Error (RMSE) :", ridge_rmse)
print("R2 Score:", ridge_r2)

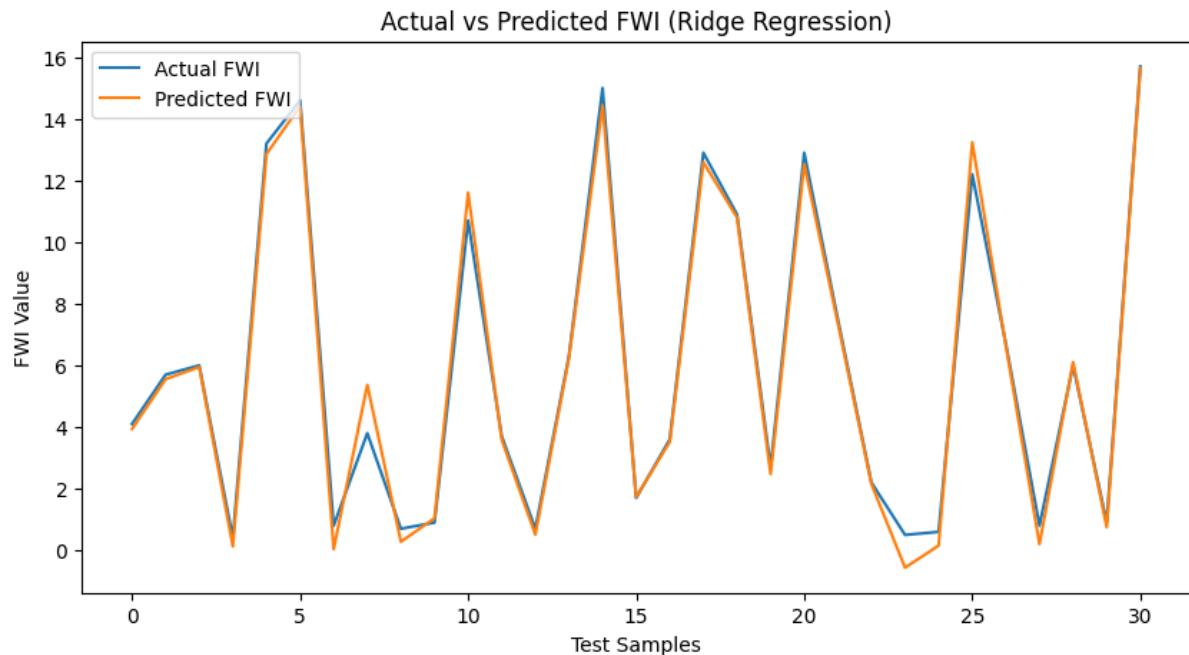
with open("ridge.pkl", "wb") as file:
    pickle.dump(ridge_model, file)

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))

plt.plot(y_test.values, label="Actual FWI")
plt.plot(y_pred_ridge, label="Predicted FWI")
plt.xlabel("Test Samples")
plt.ylabel("FWI Value")
plt.title("Actual vs Predicted FWI (Ridge Regression)")
plt.legend()
```

```
plt.show()
```



## **10.Module 5 – Evaluation & Optimization**

In this module, the trained Ridge Regression model was evaluated using MAE, RMSE, and R<sup>2</sup> score. Both training and testing R<sup>2</sup> scores were computed to analyze generalization and detect overfitting. An Actual vs Predicted plot was used to visually assess performance. Hyperparameter tuning of the alpha value was performed to optimize the model. The results show that Ridge Regression provides stable and reliable predictions for Fire Weather Index forecasting.

```
from sklearn.metrics import r2_score
y_train_pred = ridge_model.predict(X_train_scaled)
y_test_pred = ridge_model.predict(X_test_scaled)

train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)

print("Train R2 Score:", train_r2)
print("Test R2 Score:", test_r2)
```

**output:**

```
Train R2 Score: 0.9880281948713446
```

```
Test R2 Score: 0.9902192469980509
```

```
from sklearn.metrics import mean_absolute_error,  
mean_squared_error, r2_score  
import numpy as np  
  
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)  
  
print("MAE:", mae)  
print("RMSE:", rmse)  
print("R2 Score:", r2)
```

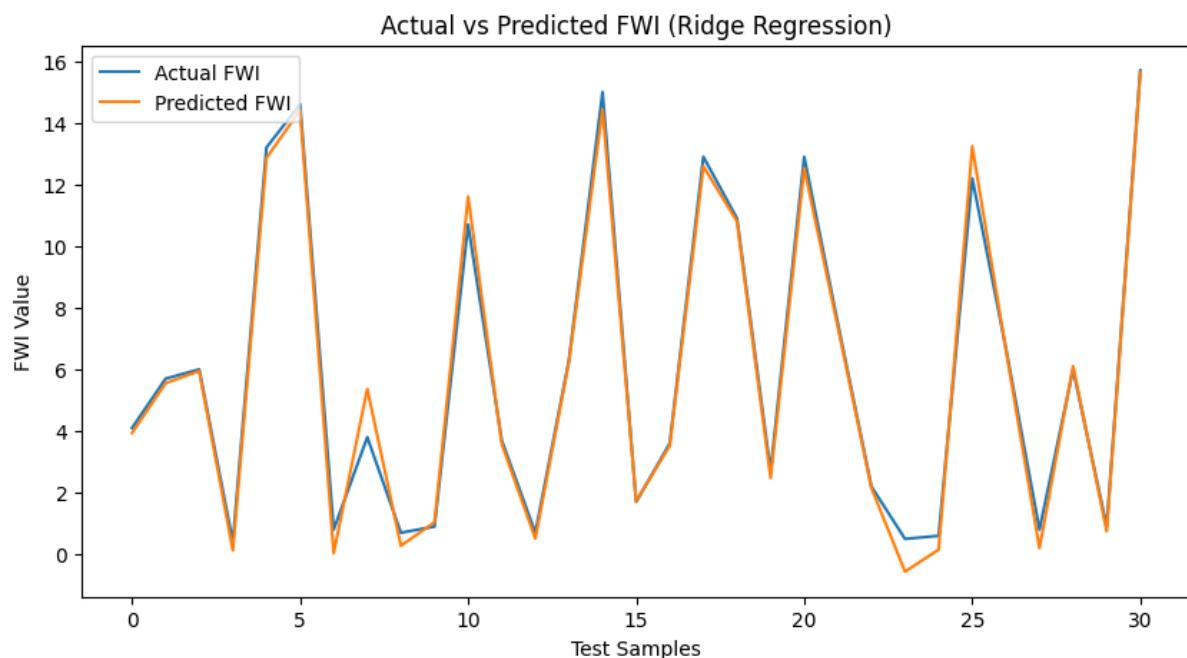
### output:

```
MAE: 0.3449895951634342
```

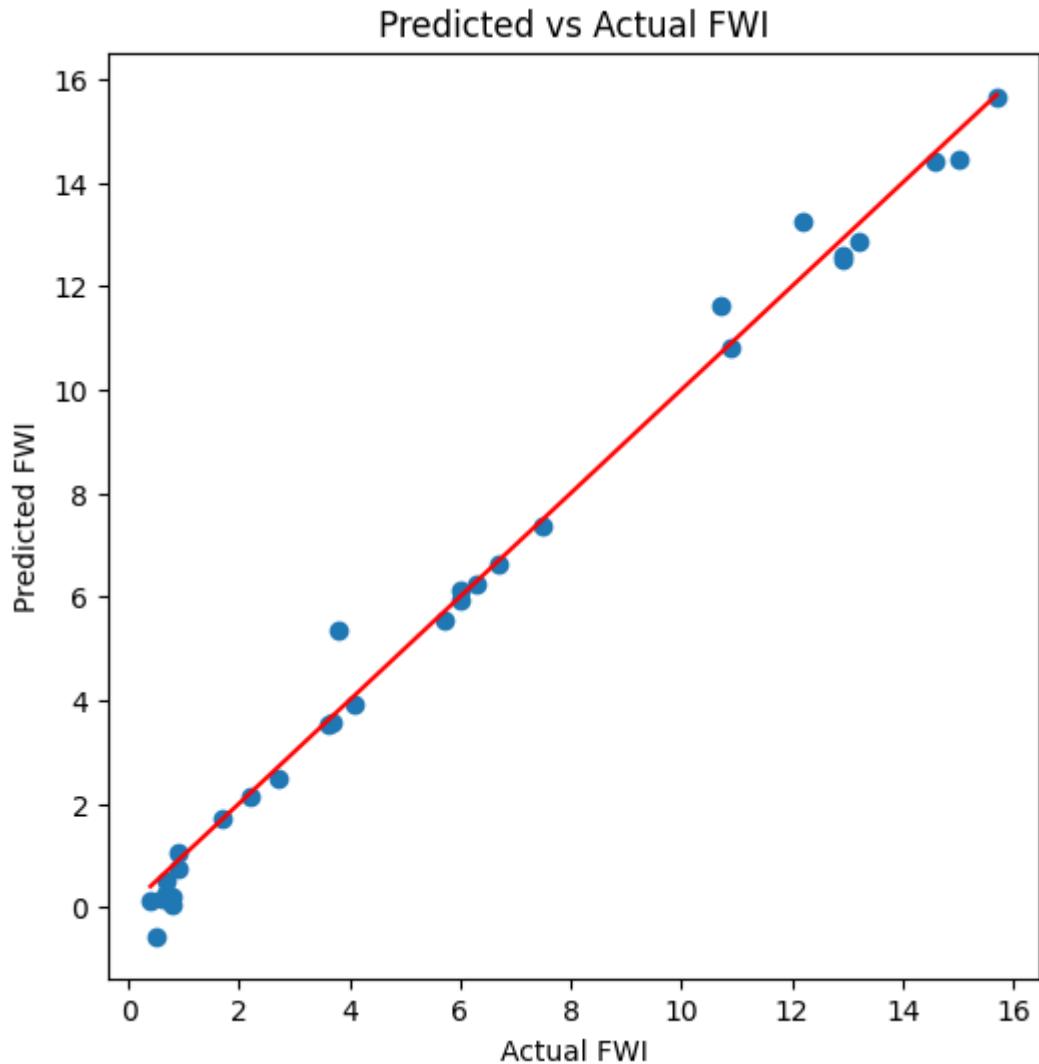
```
RMSE: 0.5023559930222369
```

```
R2 Score: 0.9902192469980509
```

This plot compares actual Fire Weather Index values with model-predicted values across test samples. A close overlap between the two lines indicates good model performance.



The scatter plot compares predicted FWI values against actual values. Points closer to the diagonal line represent accurate predictions, while deviations indicate prediction errors.



```
from sklearn.linear_model import Ridge
alphas = [0.1, 1, 10, 50, 100]
r2_scores = []
for a in alphas:
    model = Ridge(alpha=a)
    model.fit(X_train_scaled, y_train)
    pred = model.predict(X_test_scaled)
    r2_scores.append(r2_score(y_test, pred))
for a, score in zip(alphas, r2_scores):
    print(f"Alpha: {a}, R2 Score: {score}")
```

**output:**

```
Alpha: 0.1, R2 Score: 0.9911186973335182
Alpha: 1, R2 Score: 0.9902192469980509
Alpha: 10, R2 Score: 0.9750408333445757
Alpha: 50, R2 Score: 0.9546157659943422
Alpha: 100, R2 Score: 0.9402603241986112
```

```
best_alpha = alphas[r2_scores.index(max(r2_scores))]

print("Best Alpha Value:", best_alpha)
```

```
Best Alpha Value: 0.1
```

## **11.Module 6: Deployment Using Flask**

The optimized Ridge Regression model and scaler were deployed using a Flask web application. Users can input meteorological parameters through a web interface and receive real-time FWI predictions.

**code:**

```
from flask import Flask, render_template, request
import numpy as np
import pickle

from flask import Flask, send_from_directory
app = Flask(__name__)
model = pickle.load(open("ridge.pkl", "rb"))
scaler = pickle.load(open("scaler.pkl", "rb"))

@app.route("/")
def home():

    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():

    features = [
```

```

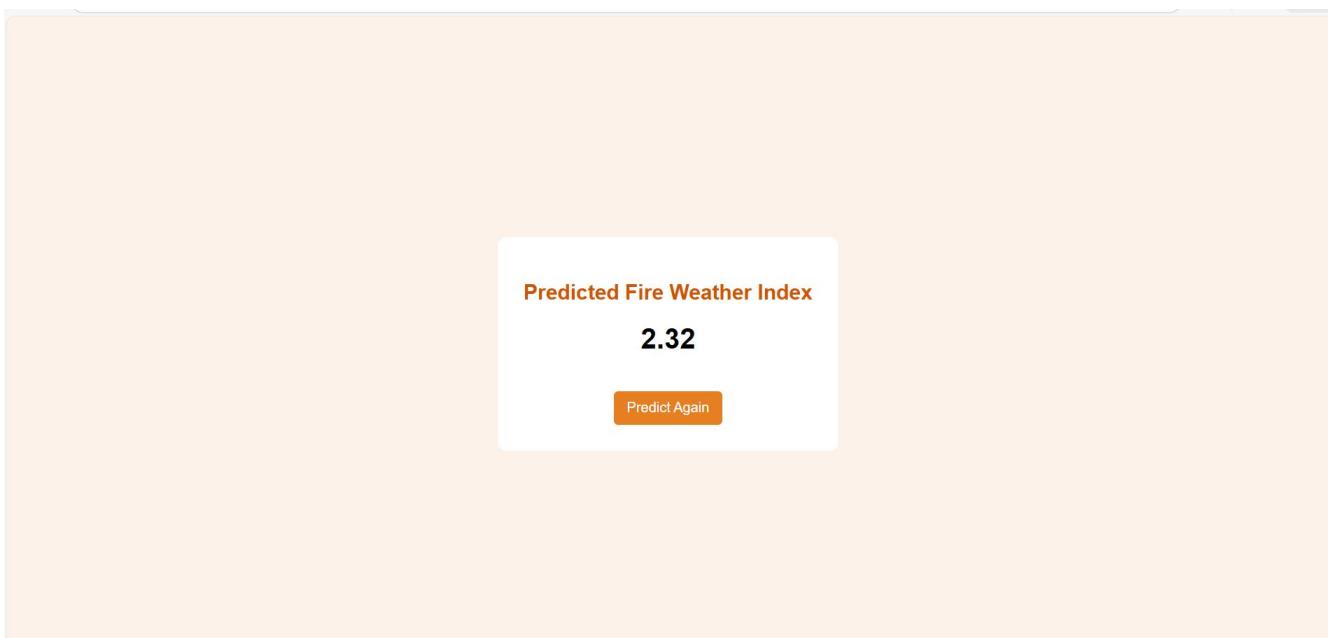
        float(request.form['Temperature']),
        float(request.form['RH']),
        float(request.form['Ws']),
        float(request.form['Rain']),
        float(request.form['FFMC']),
        float(request.form['DMC']),
        float(request.form['DC']),
        float(request.form['ISI']),
        float(request.form['BUI']),
    ]
scaled_features = scaler.transform([features])
prediction = model.predict(scaled_features)[0]
return render_template(
    "output.html",
    prediction=round(prediction, 2)
)
@app.route('/fwi_image.jpg')
def bg_image():
    return send_from_directory('.', 'fwi_image.jpg')
if __name__ == "__main__":
    app.run(debug=True)

```

## index.html

The screenshot shows a web-based form titled "Fire Weather Index Prediction". The form consists of nine input fields, each labeled with a weather variable: Temperature, Relative Humidity, Wind Speed, Rain, FFMC, DMC, DC, ISI, and BUI. Below these input fields is an orange "Predict FWI" button. To the right of the form is a decorative graphic featuring a yellow sun, white clouds, and blue waves against a blue background.

## Output.html



## Conclusion

This project successfully developed a machine learning-based Fire Weather Index prediction system. The use of Ridge Regression ensured model stability and robustness. The deployment of the model through a Flask application demonstrates the practical applicability of the system in real-world fire risk assessment.

## Future Scope

- Integration with real-time weather APIs

- Use of advanced models such as Gradient Boosting
- Mobile application deployment
- Incorporation of geographical and satellite data
- Automated alert systems for high-risk zones