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**

Sourse: 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

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          244 non-null    int64
 1   month        244 non-null    int64
 2   year         244 non-null    int64
 3   Temperature  244 non-null    int64
 4    RH          244 non-null    int64
 5    Ws          244 non-null    int64
 6   Rain         244 non-null    float64
 7   FFMC         244 non-null    float64
 8   DMC          244 non-null    float64
 9   DC           244 non-null    object
 10  ISI          244 non-null    float64
 11  BUI          244 non-null    float64
 12  FWI          244 non-null    object
 13  Classes      243 non-null    object
 14  Region       244 non-null    object
dtypes: float64(5), int64(6), object(4)
```

```
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

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Region
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	Bejaia
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	Bejaia
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	Bejaia
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0	Bejaia
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	Bejaia

Code:

```
df['FWI'] = pd.to_numeric(df['FWI'], errors='coerce')
df['DC']  = pd.to_numeric(df['DC'], errors='coerce')
print(df[['FWI', 'DC']].dtypes)
```

Output:

```
FWI     float64
DC      float64
dtype: object
```

Code:

```
df.to_csv("data/cleaned_fwi.csv", index=False)
print("Cleaned dataset saved as cleaned_fwi.csv")
```

7. Module 2 — Data Preprocessing

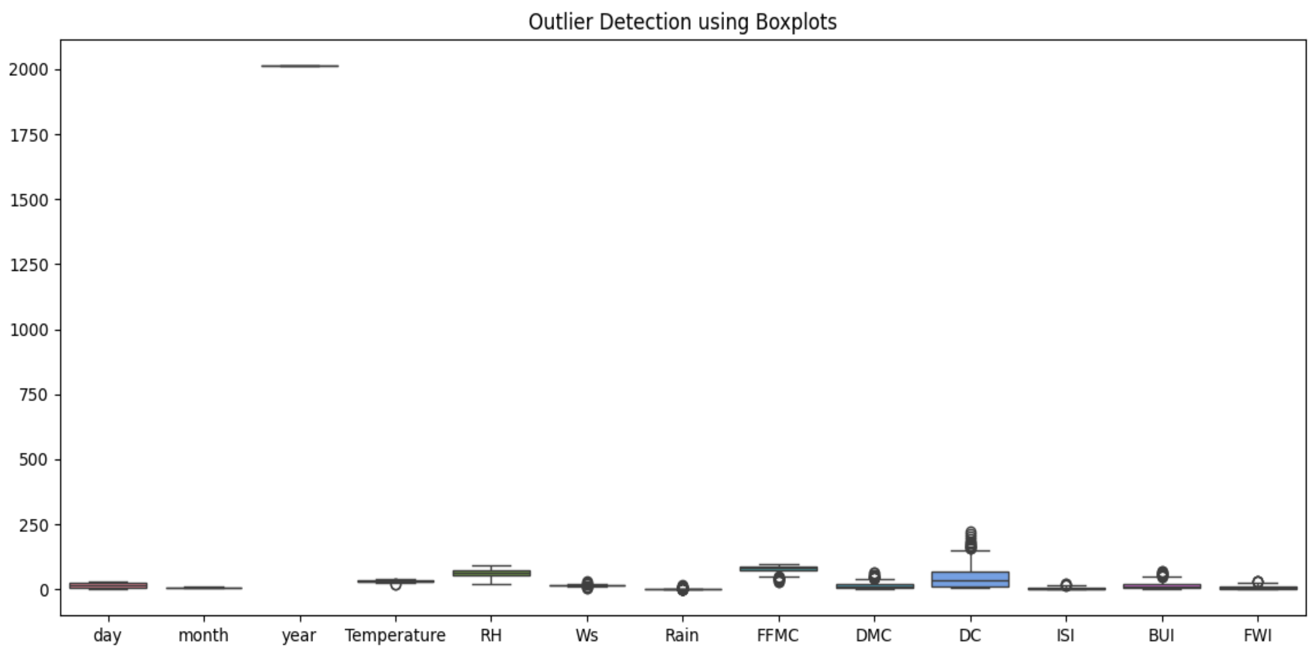
Module 2 involves analyzing the cleaned dataset to understand patterns, distributions, and relationships among variables. This includes generating statistical summaries, correlation matrices, scatter plots, density plots, boxplots, and pair plots. EDA helps identify trends, outliers, and feature interactions 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.

Code:

```
df = pd.read_csv("data/cleaned_fwi.csv")

print("Loaded Cleaned Dataset")

df.head()
```

	day	month	year	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Region
0	1	6	2012	29	57	18	0.0	65.7	3.4	7.6	1.3	3.4	0.5	Bejaia
1	2	6	2012	29	61	13	1.3	64.4	4.1	7.6	1.0	3.9	0.4	Bejaia
2	3	6	2012	26	82	22	13.1	47.1	2.5	7.1	0.3	2.7	0.1	Bejaia
3	4	6	2012	25	89	13	2.5	28.6	1.3	6.9	0.0	1.7	0.0	Bejaia
4	5	6	2012	27	77	16	0.0	64.8	3.0	14.2	1.2	3.9	0.5	Bejaia

```
plt.figure(figsize=(14, 6))

sns.boxplot(data=df.select_dtypes(include=['int64',
'float64']), orient='v')

plt.title("Outlier Detection using Boxplots")

plt.show()
```



#Removal of the outliers

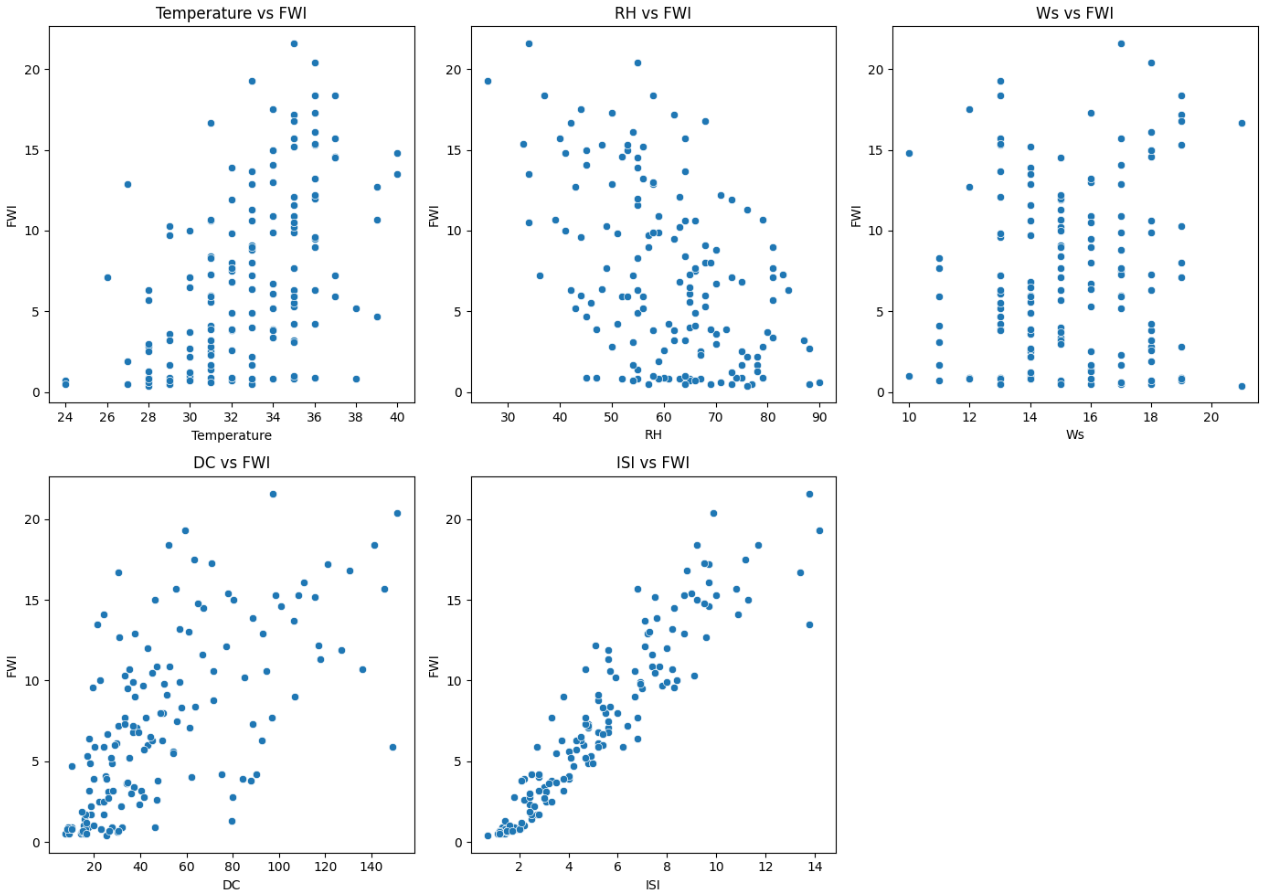
```
numeric_cols = df.select_dtypes(include=[np.number]).columns
for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower) & (df[col] <= upper)]
df = df.reset_index(drop=True)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

Output:

```
Shape after outlier removal: (155, 14)
```

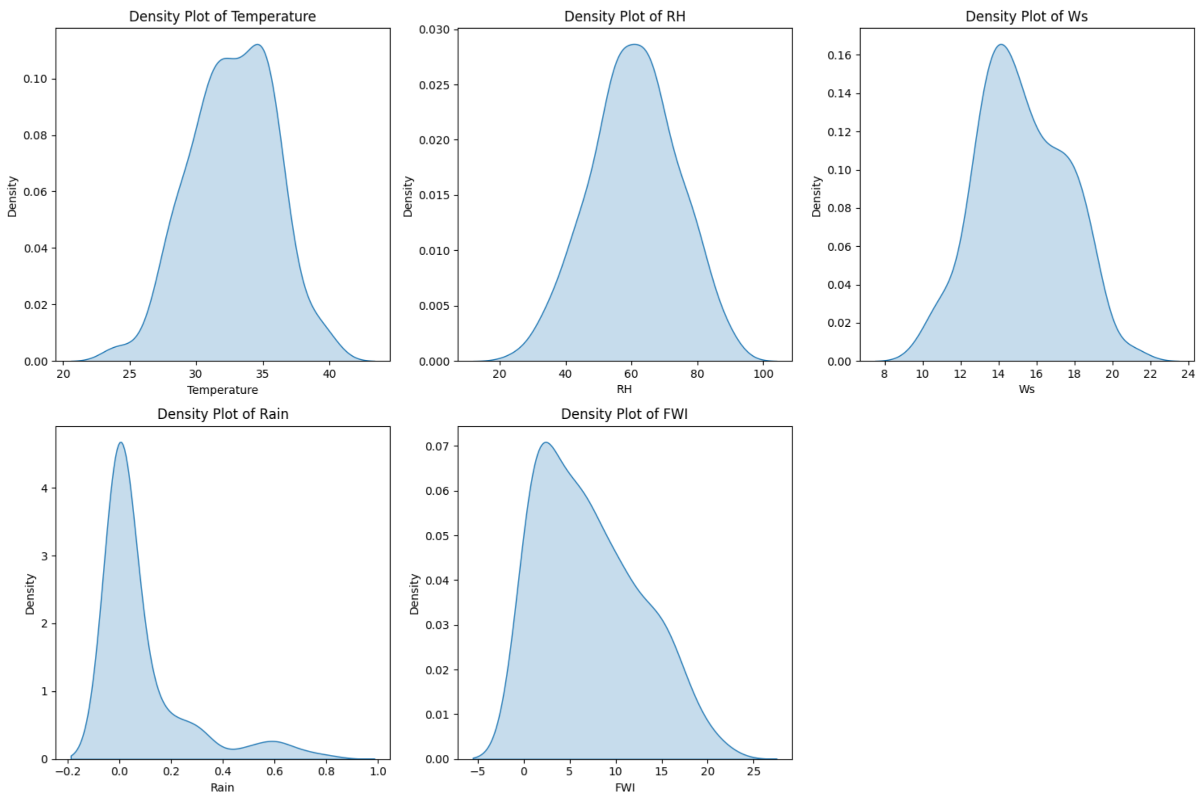
Code: (Scatterplot)

```
plt.figure(figsize=(14, 10))
features = ['Temperature', 'RH', 'Ws', 'DC', 'ISI']
for i, col in enumerate(features, 1):
    plt.subplot(2, 3, i)
    sns.scatterplot(x=df[col], y=df['FWI'])
    plt.title(f"{col} vs FWI")
    plt.xlabel(col)
    plt.ylabel("FWI")

plt.tight_layout()
plt.show()
```



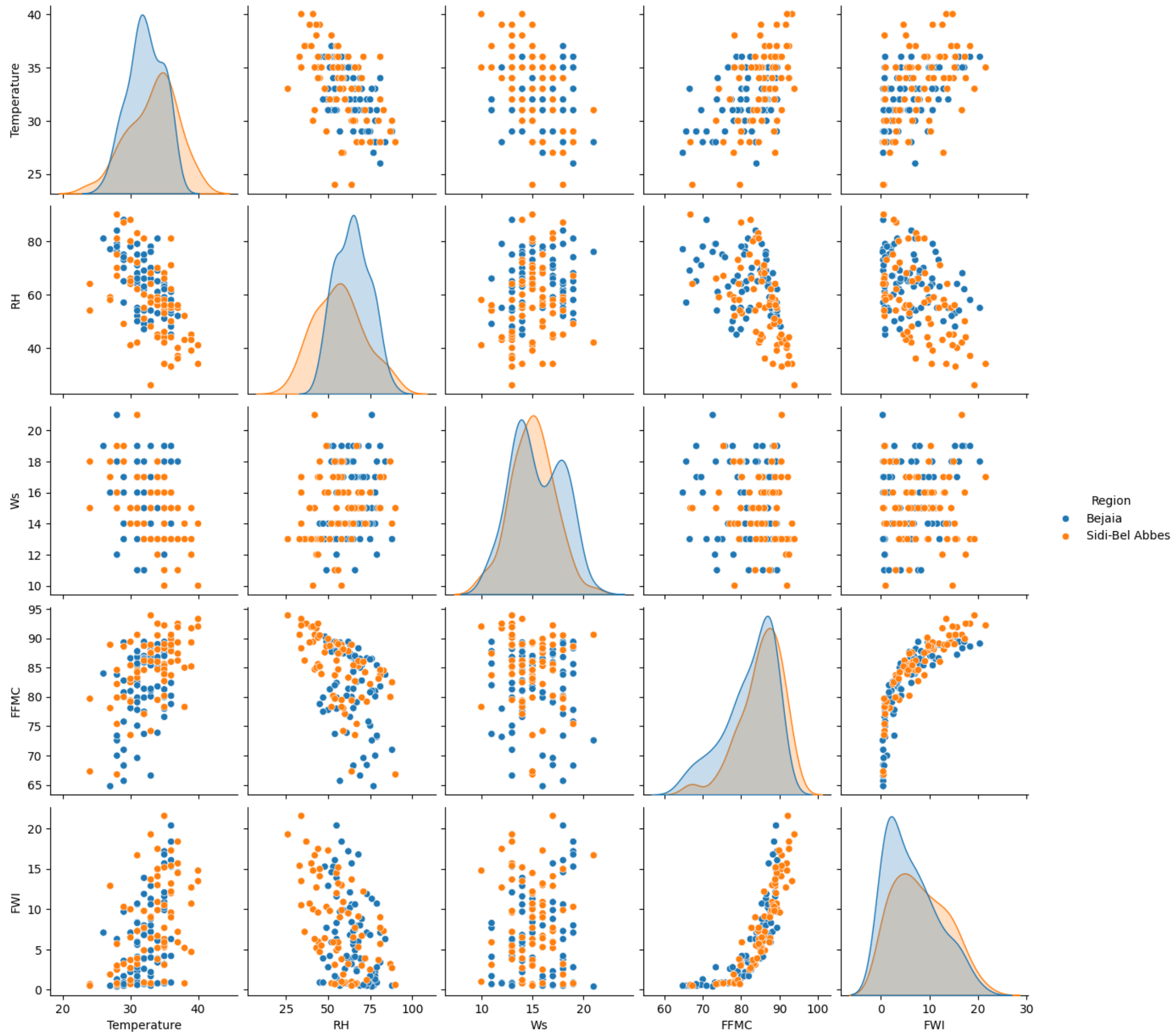
Code: (Density plot)

```
plt.figure(figsize=(15, 10))
num_cols = ['Temperature', 'RH', 'Ws', 'Rain', 'FWI']
for i, col in enumerate(num_cols, 1):
    plt.subplot(2, 3, i)
    sns.kdeplot(df[col], fill=True)
    plt.title(f"Density Plot of {col}")
plt.tight_layout()
plt.show()
```



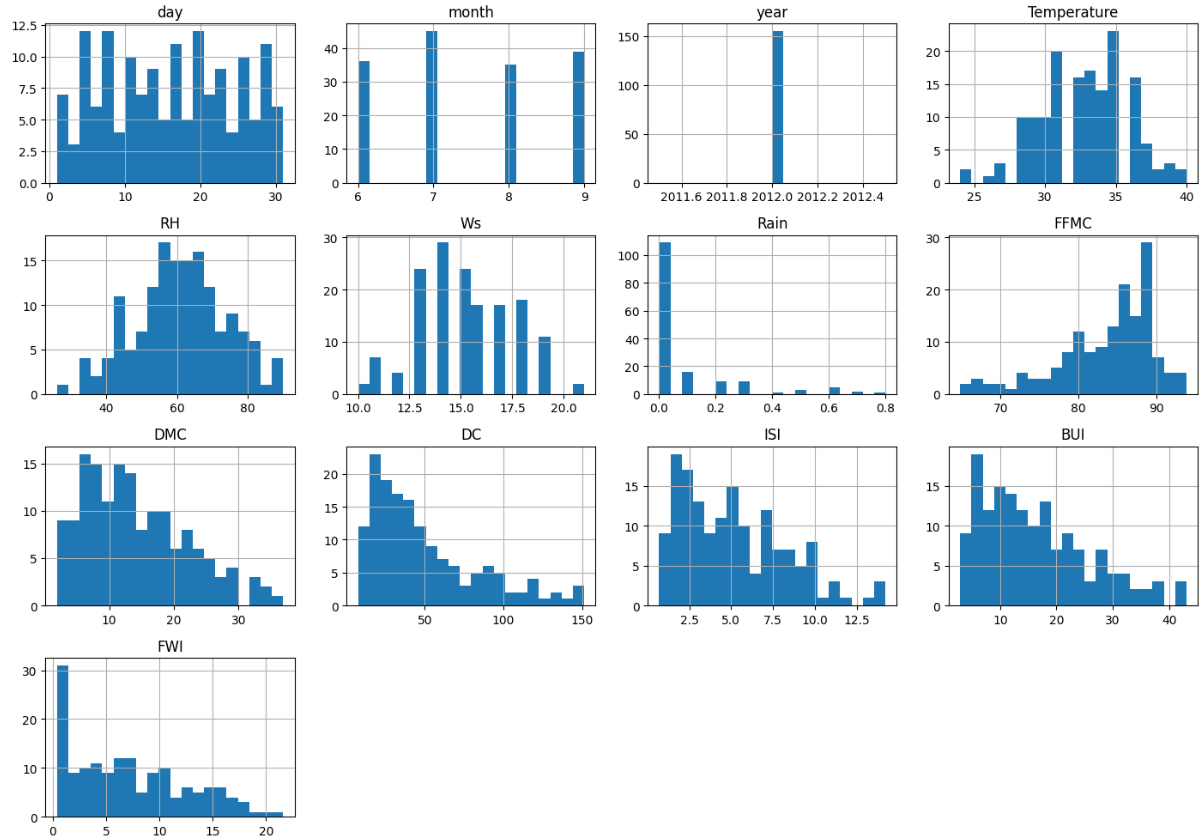
Code: (Pair plot)

```
sns.pairplot(df[['Temperature', 'RH', 'Ws', 'FFMC', 'FWI',
'Region']],
             hue='Region')
plt.show()
```



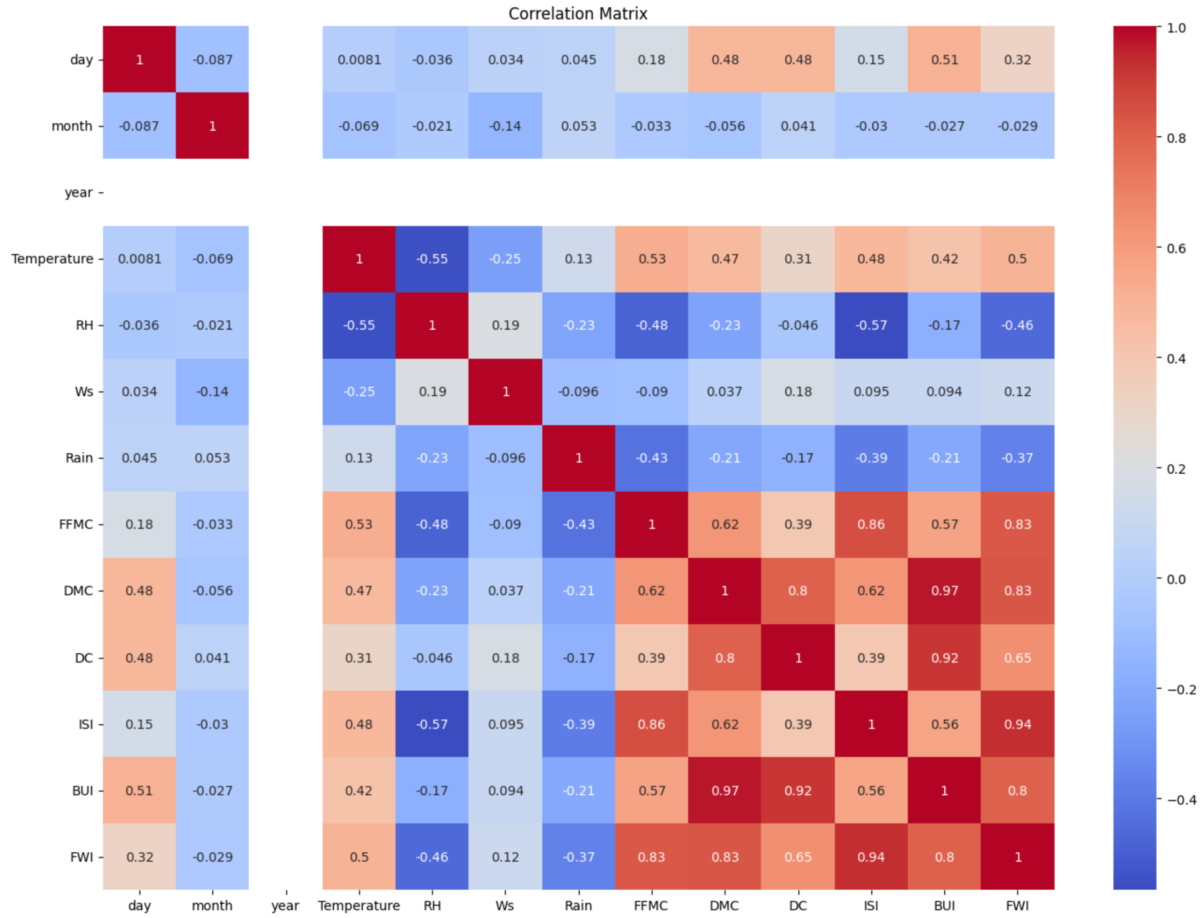
Code: (Histogram)

```
df.hist(figsize=(14, 10), bins=20)

plt.tight_layout()

plt.show()
```



Code: (Correlation Matrix)

```
plt.figure(figsize=(14, 10))
numeric_df = df.select_dtypes(include=['int64', 'float64'])
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.tight_layout()
plt.show()
```



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
```