Fire Weather Index Predictor

(A Machine Learning Model to Predict Fire Weather Index)

Infosys Springboard

Infosys SpringBoard Virtual Internship Program

Submitted

by

Gudipati Naga Venkata Sravanthi

Under the guidance of Mentor **Praveen**

PROBLEM STATEMENT:

We have developed a Fire Weather Index (FWI) Prediction System that will evaluate the potential risk of wildfires based on several key meteorological and fire behavior factors. The FWI Prediction System will utilize temperature, humidity, wind speed, precipitation levels, as well as several important fire danger indices (FFMC, DMC, DC, ISI). The data has been thoroughly pre-processed, cleaned and explored to validate its quality and visually identify the most relevant trends in the data.

These advanced models allow us to model the relationships between multiple variables, both environmental and fire risk-related, for the creation of accurate and reliable Fire Weather Index Predictions (FWI). In addition to this input, we include both localized and regional-level information to help assess patterns of wildfires to improve the FWI Prediction System's applicability to local fire patterns. The ultimate objective of the FWI Prediction System is to assist both government agencies as well as decision-makers in finding fire locations before a major wildfire outbreak, taking preventative action and effectively allocating resources toward those areas at greatest risk.

EXPECTED OUTCOMES:

Deliverables are expected, but this list does not include everything. Some examples of deliverables are: a Ridge Regression model that can give accurate Fire Weather Index (FWI) predictions in different weather conditions; a data-preparation process that uses StandardScaler to normalize the data so the model works better; a simple Flask web app where users can enter weather values and get FWI predictions instantly; and an analytical tool that helps forest departments, disaster management teams, and research groups plan for wildfires by using information from past data.

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

Requirements:

- pandas==2.3.3
- numpy==2.3.5
- matplotlib>=3.0
- seaborn>=0.11
- scikit-learn>=1.0

Milestone 1

Module 1: Data Collection

Initially, the data to be used in making accurate predictions of the FWI was collected from different sources on the internet. The final choice was based on the environmental and fire danger parameter factors that were necessary or relevant to make an accurate prediction. The chosen dataset contained the following parameters: Temperature, Humidity, Wind Speed, Rainfall, FFMC, DMC, ISI, Regional Identification. All of these parameters became part of the dataset. Once the dataset was acquired, it was placed in a Pandas DataFrame. The first thing that was done was to check that the data was ready for analysis by reviewing its integrity.

The integrity review included assessing the types of data, identifying which components contained missing or inconsistent values, assessing how much memory the dataset consumed, generating statistical summaries for all variables, understanding the distribution of each feature within the dataset, confirming the number or number of rows and columns, and identifying and removing any duplicated rows. After confirming that the dataset had met all of the needed requirements for quality, the dataset was validated to ensure that it was accurate enough for the future phases of development and modelling.

1. loading the dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
# Load the dataset
df= pd.read_csv("E:\FWI\myenv\FWI Dataset.csv")
df.head(4)
```

	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

FIGURE 1

```
# Data types
df['DC'] = df['DC'].astype(str).str.replace(' ', '', regex=False)
df['FWI'] = df['FWI'].astype(str).str.replace(' ', '', regex=False)
df['DC'] = pd.to_numeric(df['DC'], errors='coerce')
df['FWI'] = pd.to_numeric(df['FWI'], errors='coerce')
df.dtypes
```

```
day              int64
month            int64
year             int64
Temperature      int64
 RH              int64
 Ws              int64
Rain           float64
FFMC           float64
DMC            float64
DC             float64
ISI            float64
BUI            float64
FWI            float64
Classes         object
Region          object
dtype: object
```

FIGURE 2

```
df.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    float64
 10  ISI          244 non-null    float64
 11  BUI          244 non-null    float64
 12  FWI          243 non-null    float64
 13  Classes      243 non-null    object 
 14  Region       244 non-null    object 
dtypes: float64(7), int64(6), object(2)
memory usage: 28.7+ KB
```

FIGURE 3

```
#step 1: Check for missing values
print("Missing values in each column:")
df.isnull().sum()
rows_with_missing = df[df.isnull().any(axis=1)]
print("Rows with missing values: \n",rows_with_missing)
```

```
Rows with missing values:
      day  month  year  Temperature   RH   Ws  Rain   FFMC   DMC     DC   ISI  \
165   14      7  2012           37   37   18   0.2  88.9  12.9  14.69  12.5

      BUI  FWI Classes           Region
165  10.4  NaN      NaN  Sidi-Bel Abbes
```

FIGURE 4

MODULE 2: Data Exploration (EDA) and Data Preprocessing

During the preprocessing and exploratory analysis phase, the dataset was thoroughly examined to ensure its accuracy, completeness, and suitability for machine learning model development. This stage aimed to identify and resolve data quality issues while gaining a deeper understanding of the underlying patterns within the dataset.

```
#step 2: Outlier Detection using Boxplots
plt.figure(figsize=(12, 6))
df.boxplot(rot=45)
plt.title("Boxplot for Outlier Detection")
plt.show()
```

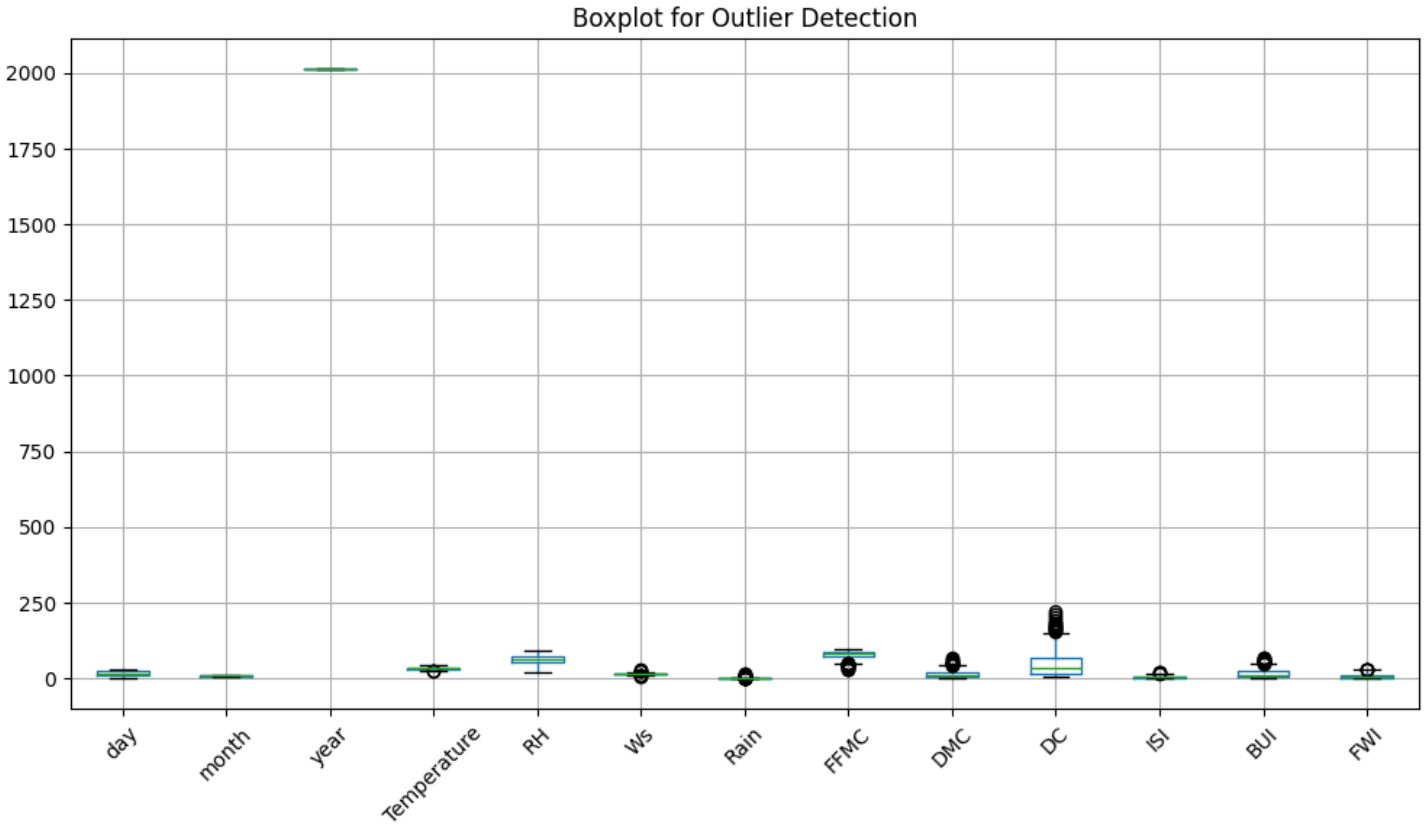

FIGURE 5

Outliers in the dataset were analyzed using both visual methods (boxplots) and statistical techniques based on the Interquartile Range (IQR). This combined approach helped identify extreme values that could negatively impact model performance.

```
#statistical threshold
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
def detect_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower_bound) | (data[column] >
upper_bound)]
    return outliers

# Check number of outliers in each numerical column
for col in num_cols:
    outliers = detect_outliers_iqr(df, col)
    print(f"Column: {col} → Number of outliers: {len(outliers)}")
```

```
Column: day → Number of outliers: 0
Column: month → Number of outliers: 0
Column: year → Number of outliers: 0
Column: Temperature → Number of outliers: 2
Column: RH → Number of outliers: 0
Column: Ws → Number of outliers: 8
Column: Rain → Number of outliers: 35
Column: FFMC → Number of outliers: 16
Column: DMC → Number of outliers: 12
Column: DC → Number of outliers: 15
Column: ISI → Number of outliers: 4
Column: BUI → Number of outliers: 12
Column: FWI → Number of outliers: 4
```

FIGURE 6

To identify extreme or abnormal values, the **Interquartile Range (IQR) method** was applied to all numerical features in the dataset. This statistical technique detects outliers by measuring how far a value lies from the central spread of the data.

```
#step 3: Visualizations
#Histograms
df.hist(figsize=(14, 10), bins=20, edgecolor='black')
plt.suptitle("Histograms of Numerical Features")
plt.show()
```

```
# Density Plots
plt.figure(figsize=(12, 8))

for col in num_cols:
    sns.kdeplot(df[col], label=col, fill=True)

plt.title("Density Plots of Numerical Features")
plt.xlabel("Value")
plt.ylabel("Density")
plt.legend()
plt.show()
```

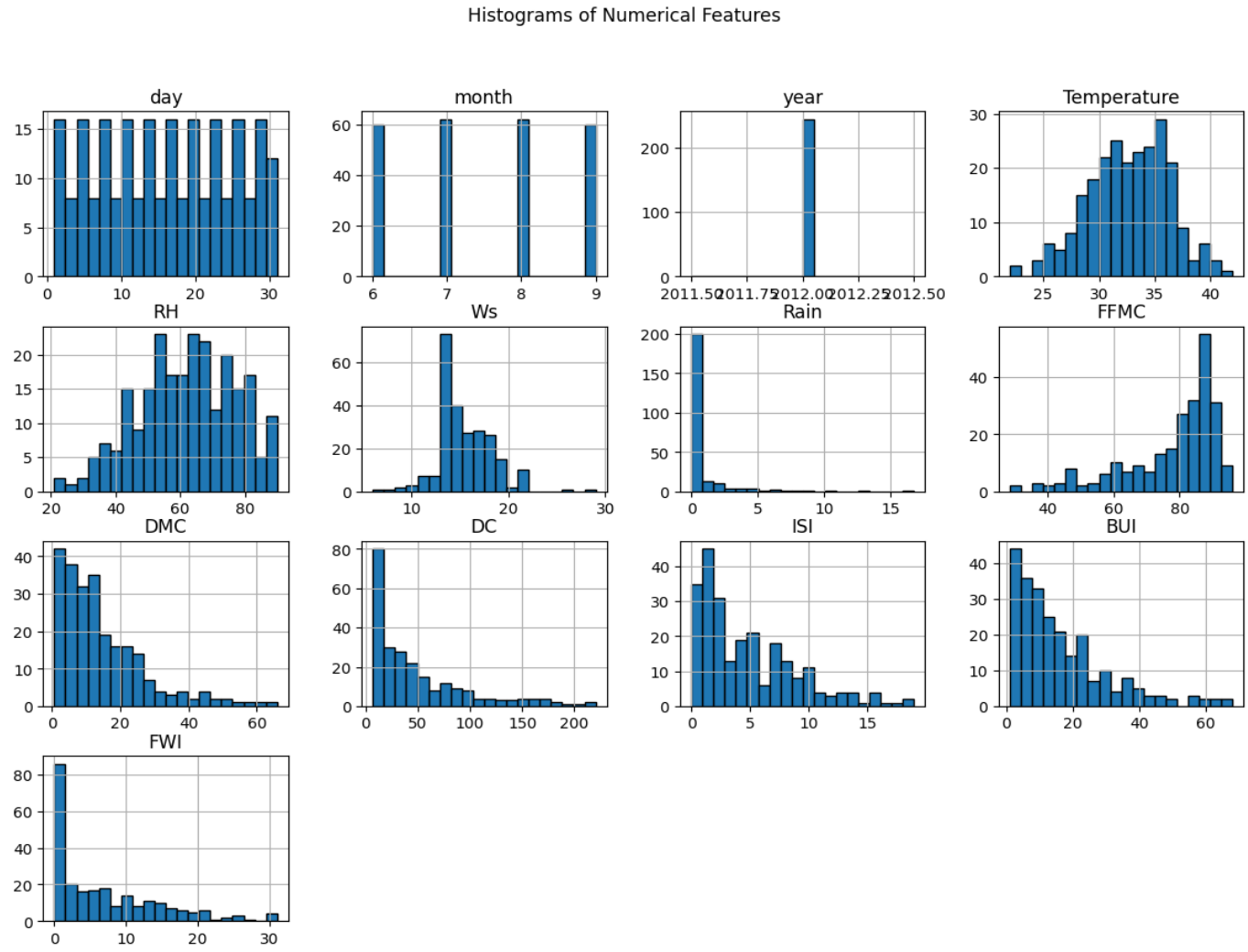

FIGURE 7

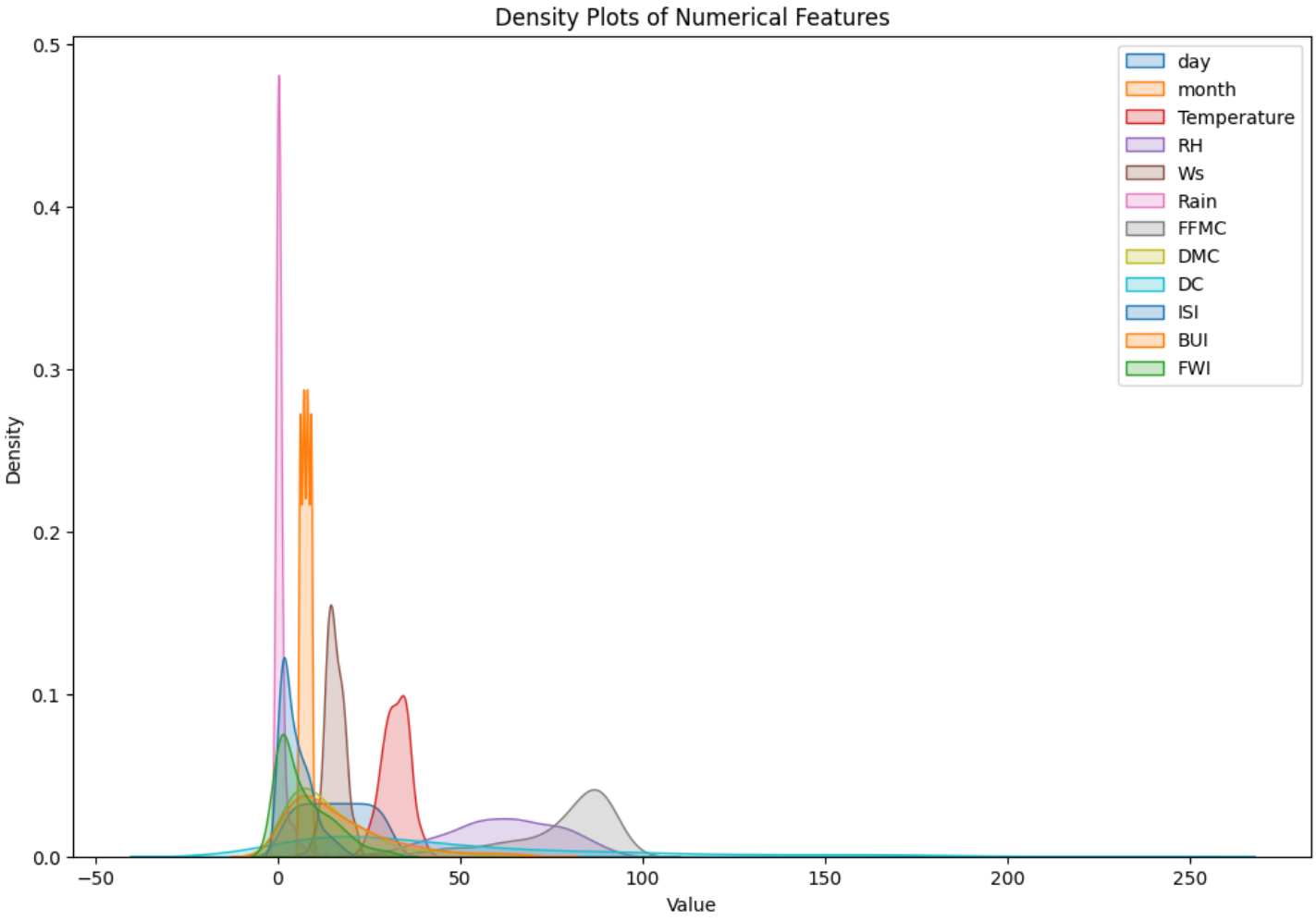

FIGURE 8

Histograms combined with KDE (Kernel Density Estimate) curves were generated for all numerical features to understand the overall distribution and behavior of the data. These visualizations help reveal important patterns that may influence the performance of the machine learning model.

Key Observations from the Graphs:

- **Temperature, Relative Humidity (RH), and Wind Speed** show smoother and more balanced distributions, indicating that these variables are spread fairly evenly across the dataset.
- **Rain, DMC, ISI, and BUI** exhibit strong right-skewness. This means most values are low, with only a few extreme high values present. Such skewness is common in environmental datasets where rainfall and fire indices vary significantly over time.

- Features like **FFMC and DC** display moderate variation with visible peaks, suggesting consistent seasonal or environmental patterns.
- The KDE curves helped identify whether each feature follows a normal distribution or deviates from it. Identifying non-normal distributions is crucial for selecting appropriate scaling methods and machine learning algorithms.

```
#step 4: Correlation Matrix
plt.figure(figsize=(12, 8))
correlation_matrix = df[num_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix Heatmap")
plt.show()
```

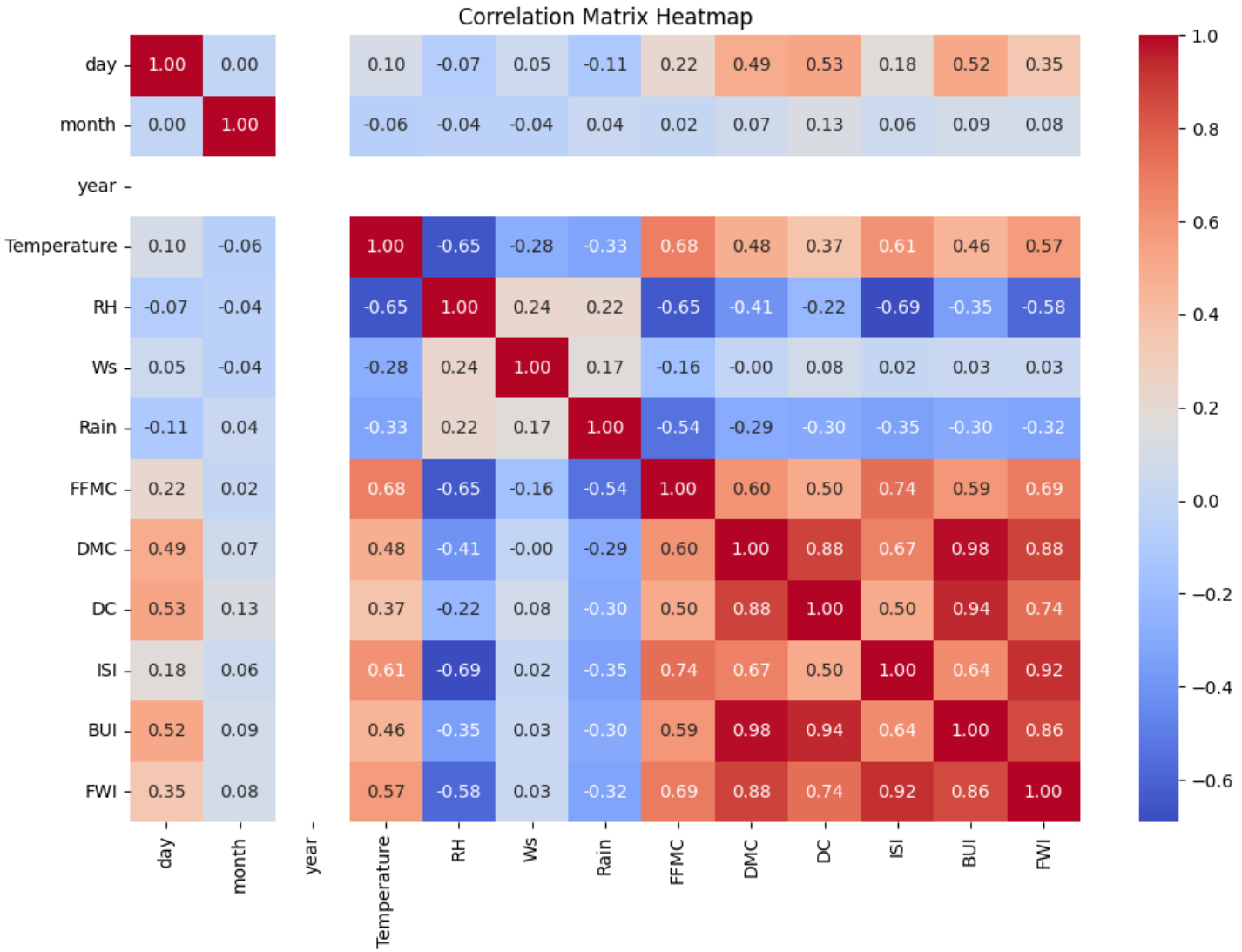

FIGURE 9

A correlation heatmap was generated to study relationships among numerical features.

Key findings:

- Strong positive correlations were observed among *BUI, DMC, FFMC, ISI,* and *Temperature*, suggesting shared patterns in fire-danger conditions.
- *Relative Humidity (RH)* showed strong negative correlations with several fire indices, reflecting its natural dampening effect on fire risk.

```
# Scatterplots of Selected Features
sns.pairplot(df[num_cols].dropna())
plt.suptitle("Scatterplots of Selected Features", y=1.02)
plt.show()
```

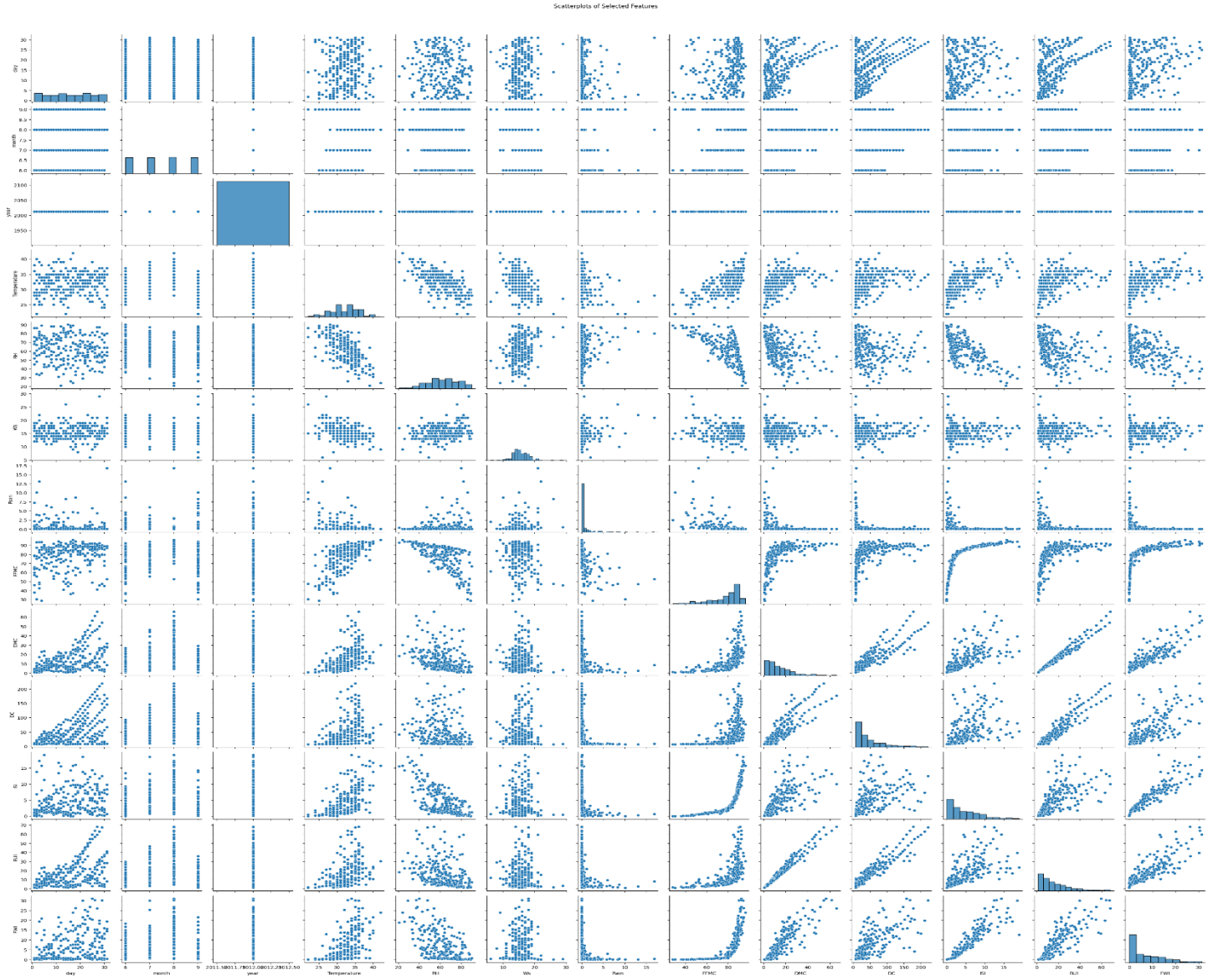

FIGURE 10

Pair plots were generated to visually analyze the relationships between multiple numerical features in the dataset. These plots display scatterplots for every pair of variables along with individual feature distributions on the diagonal. This makes it easier to observe how different environmental and fire-danger features interact with each other.

- Variables such as **FFMC, DMC, ISI, and Temperature** show noticeable upward trends when paired with one another, indicating strong positive relationships linked to fire-prone conditions.
- **Relative Humidity (RH)** demonstrates an inverse pattern with several fire indices, reflecting the natural decrease in fire risk when humidity levels are higher.
- Some features, such as **Rain and Wind Speed**, show more scattered relationships, suggesting weaker correlations with the Fire Weather Index (FWI).
- The diagonal plots reveal each feature's distribution, helping identify skewness, spread, and potential outliers.

```
#step 5: Encoding Categorical Variables
df.columns = df.columns.str.strip()

print("Unique values in Region:", df['Region'].unique())
le = LabelEncoder()

df['Region_encoded'] = le.fit_transform(df['Region'])
df[['Region', 'Region_encoded']].head()
```

	Region	Region_encoded
0	Bejaia	0
1	Bejaia	0
2	Bejaia	0
3	Bejaia	0
4	Bejaia	0

FIGURE 11

SUMMARY

In **Module 1**, a structured dataset containing all relevant environmental features required for Fire Weather Index (FWI) analysis was collected and loaded into a Pandas DataFrame. The dataset included key attributes such as Temperature, Relative Humidity, Wind Speed, Rain, FFMC, DMC, ISI, BUI, Region, and the target variable FWI. Initial inspection was performed to verify data types, ensure consistency, and confirm the completeness and proper formatting of all features. In **Module 2**, the dataset underwent thorough exploration and preprocessing to prepare it for modeling. Missing values were handled appropriately, and outliers were detected using both visual (boxplots) and statistical (IQR-based) methods, revealing that some features, such as Temperature, Wind Speed, Rain, FFMC, DMC, ISI, and BUI, contained outliers. Feature distributions were analyzed through histograms and density plots, relationships between variables were examined using correlation matrices and scatterplots, and categorical attributes like Region were encoded numerically. Finally, unnecessary columns were removed, and the cleaned, transformed dataset was saved, ensuring it is accurate, consistent, and ready for subsequent machine learning modeling.