FIRE WEATHER INDEX PREDICTOR

A Project Report Submitted to

Infosys Springboard

By

Mallisetti Meghana

Under the guidance of Mentor ***Praveen***

Problem 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.

Expected 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

Milestone 1: Data Collection and Data Preprocessing

Module 1: Data Collection

My Data Collection is done by browsing the **Kaggle** for the datasets. The dataset which I found was the **Algerian_forest_fires_dataset.csv** which as all the required features like the

- Temperature
- Relative Humidity
- Wind Speed
- Rain
- FFMC
- DMC
- ISI
- Region

I have downloaded the dataset as the FWI Dataset.csv and started with the basic inspection of the data.

At First, I have done is loaded the dataset into the pandas.

Python Code Snippet

```
#Step-1:Loading the dataset

import pandas as pd

df = pd.read_csv("FWI Dataset.csv")

#Step-2: Basic information about the dataset

print(df.info())
```

After loading the dataset into the pandas, I have conducted the initial inspection by checking the head() and tail() of the dataset to understand the datatypes and feature distribution. I have stripped the whitespaces from the column names for extra clarity and I have also observed that the 'DC' and 'FWI' columns have the object datatype so I have converted the datatype to float64.

```
#Since the 'FWI' and 'DC' columns contain non-numeric values

df['FWI'] = pd.to_numeric(df['FWI'], errors='coerce')

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

```
#Converting the 'FWI' and 'DC' columns to numeric data types
df['FWI'] = pd.to_numeric(df['FWI'])
df['DC'] = pd.to_numeric(df['DC'])
#printing the data types of each column after conversion
print(df.info())
```

So, after converting the datatypes of **FWI** and **DC** I have checked If the changes have achieved or not by using the **info().**

#	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	243 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

Table 1: Displaying the changes of the datatypes after conversion of DC and FWI columns

Key Learnings:

Module 1 taught me the importance of gathering and cleaning a dataset before any kind of analysis. I was able to understand how to check for the structure of data, verify the data types, and load the dataset into Pandas for initial inspection. This module helped me become confident in handling raw data and prepare it further for preprocessing and exploration.

Module 2: Data Exploration and Data Preprocessing

In this module, I have checked for the **missing values** at first in the dataset and removed the NaN values in the '**FWI**' and '**DC**' columns and also checked for the null values using the **isnull().**

Python Code Snippet

```
#Handling missing values by removing rows with NaN values in 'FWI' and 'DC' columns

df = df.dropna(subset=['FWI', 'DC'])

print("Data after removing rows with missing 'FWI' and 'DC' values:")

print(df.info())
```

Outliners using Boxplot:

A boxplot is a technique that assists in the detection of data values that are significantly higher or lower than the normal range. Such exceptional values are referred to as outliers. Outliers in a boxplot are represented by points that are beyond the whiskers, indicating that they do not belong to the usual distribution of the data.

```
import matplotlib.pyplot as plt

import numpy as np

plt.figure(figsize=(12,6))

df.select_dtypes(include=np.number).boxplot()

plt.xticks(rotation=90)

plt.title("Boxplot for Outliner Detection")

plt.show()
```

Removing the Outliners using the IQR method:

The IQR (Interquartile Range) technique is used to eliminate values that deviate excessively from the normal range. We start by computing the IQR = Q3- Q1, where Q1 and Q3 correspond to the 25th and 75th percentiles, respectively. Any observation that lies below Q1- 1.5×IQR or above Q3 + 1.5×IQR is labeled as an outlier. To ensure that the dataset is clean and more reliable for analysis, we exclude these points.

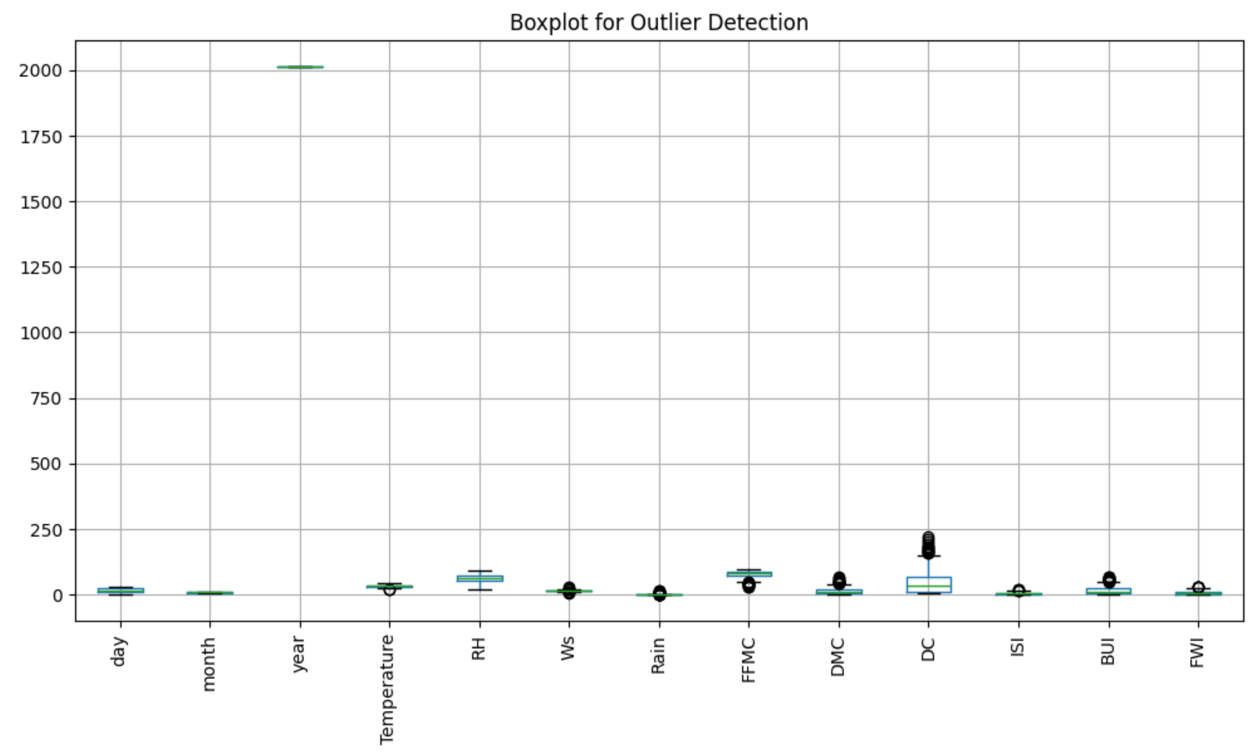

Figure 1: Displaying the Boxplot for Outliner Detection

Data Visualization using the Histograms:

Histograms were employed to comprehend the distribution of each attribute in the dataset. They represent the occurrence of various values and assist in recognizing characteristics like skewness, spread, and whether the distribution of data is even or uneven.

Correlation Matrix:

The correlation matrix is a graphical representation that indicates the strength of the relationship between different features thereby facilitating the identification of positive, negative, and weak correlations in the dataset.

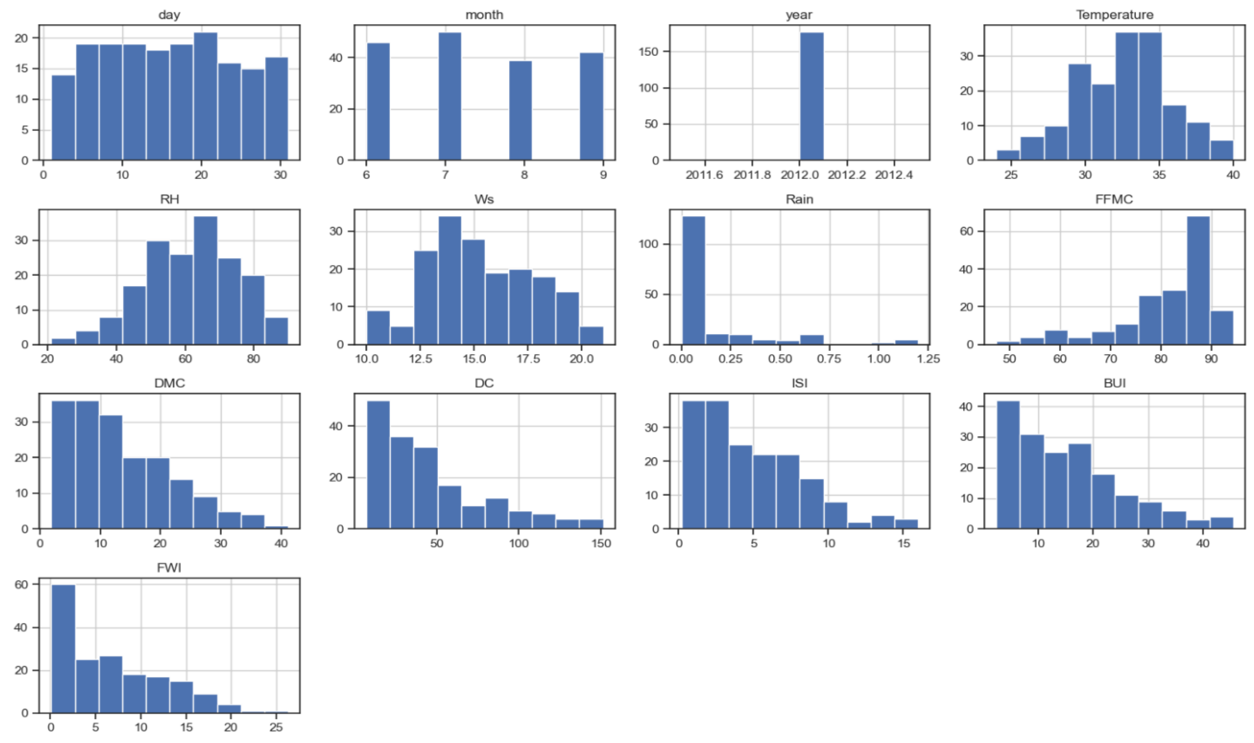

Figure 2: Visualizing Data Distribution using Histograms

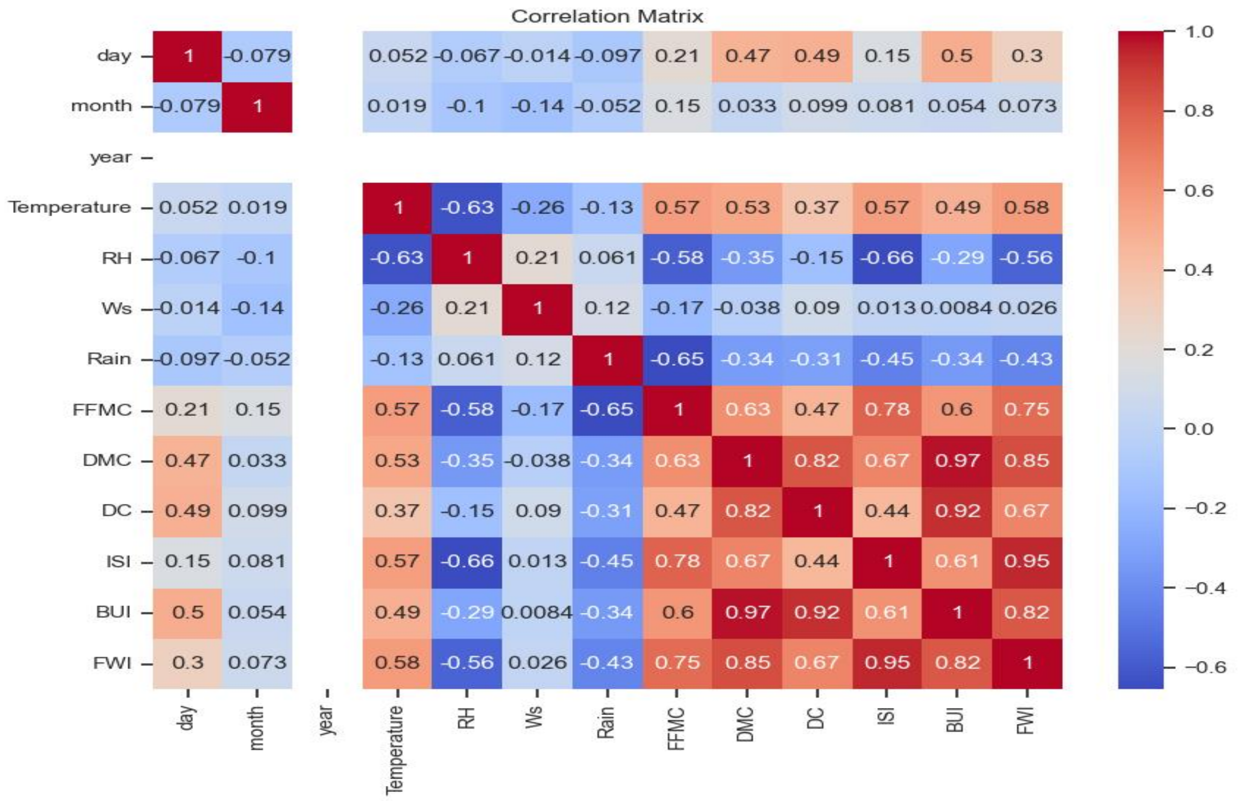

Figure 3: Correlation Matrix shows the relationships between features like Temperature, RH, Wind Speed, Rain, FFMC, DMC, DC, ISI, BUI, and FWI

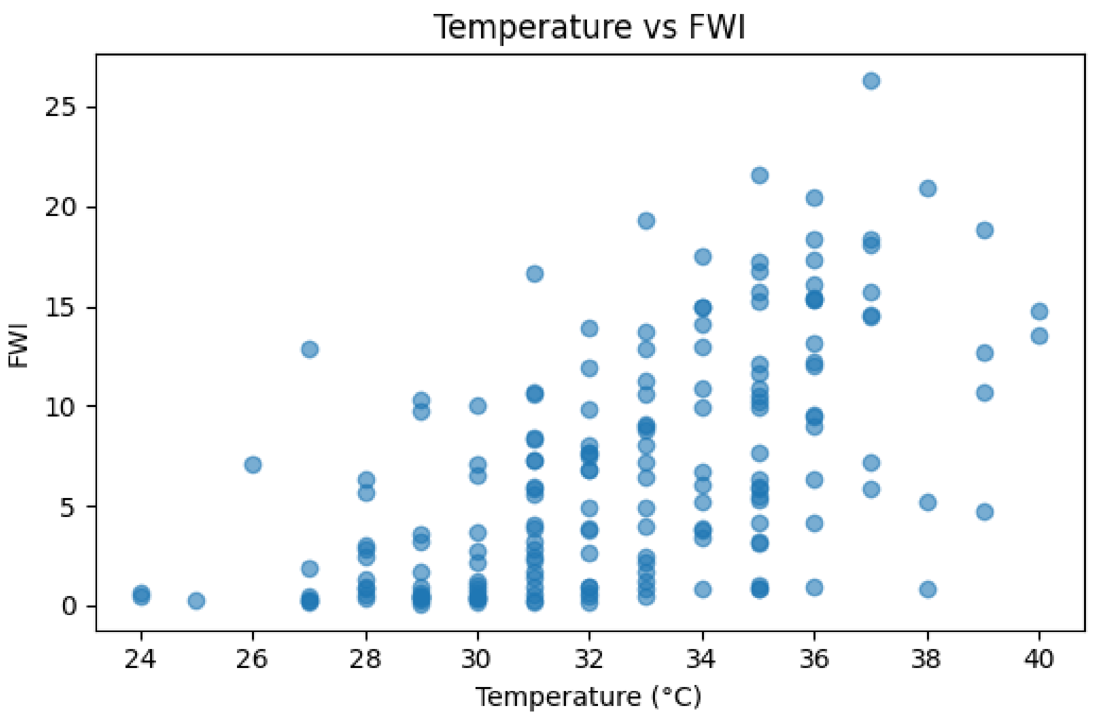

Figure 4: Scatter Plot of the Temperature and the FWI features

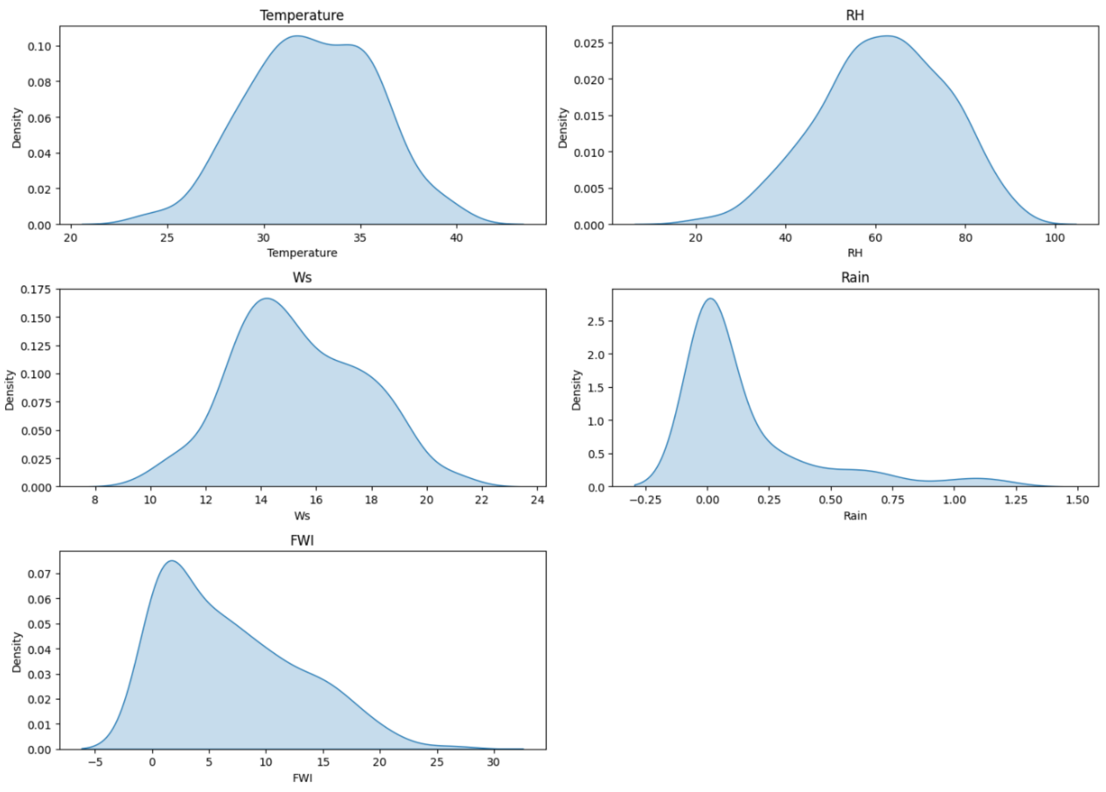

Figure 5: Density Plot diagrams of Temperature, RH, WS, Rain and FWI

Pair Plot:

A pair plot is a visualization that shows the relationships between multiple features in a dataset by creating scatterplots for every pair of variables and histograms for their individual distributions.

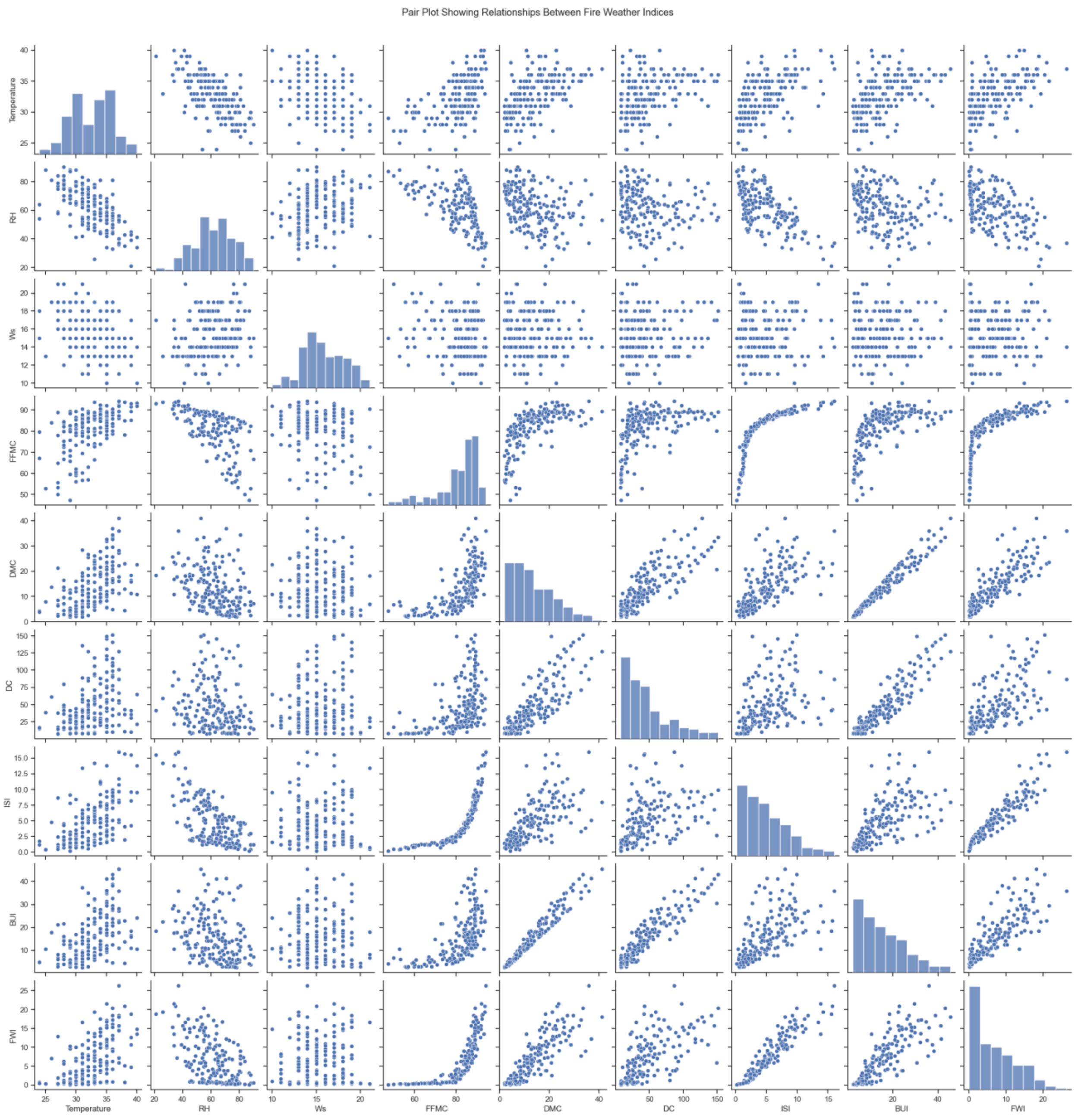

Figure 4: The pair plot shows how the key features like Temperature, RH, Wind Speed, FFMC, DMC, DC, ISI, BUI, and FWI are related to each other using the scatterplots and histograms

Encoding the Categorical Variables:

In the encoding of the categorical variables the Region and the Classes variables are to be encoded using the **LabelEncoder**.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

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

After encoded the Region variable I have checked the dataset if the changes I have made affected accurately or not for that, I have used the **is.null()** for checking the null values and used **info()** for seeing the changes affected.

Saving the Clean Dataset:

After completing preprocessing, the cleaned dataset with name **FWI_Cleaned_Dataset.csv** was saved for further use in the modeling phase. This ensures consistency and avoids repeating preprocessing steps during model training.

```
#Atep 12: Saving the cleaned dataset to a new CSV file

df.to_csv("FWI_Cleaned_Dataset.csv", index=False)
```

Key Learning:

Module 2, I learned how to explore and clean data using Python libraries like Pandas, NumPy, Matplotlib, and Seaborn by checking for missing values, handling outliers, visualizing patterns, and encoding categorical features in order to prepare the dataset for model training.