Project Title - Fire Weather Index Prediction(FWI) Using Machine Learning

Project 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.

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 prediction.
• 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

Submitted by: *Gayatri Autade*

Milestone 1 – Data Collection & Preprocessing

1. Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
import math
```

2. Load the Dataset

```
data=pd.read_csv("C:/Users/autad/Downloads/FWI
Dataset.csv")
print(data.head())
```

Output -

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

    FWI      Classes    Region
0  0.5  not fire     Bejaia
1  0.4  not fire     Bejaia
2  0.1  not fire     Bejaia
3    0  not fire     Bejaia
4  0.5  not fire     Bejaia
```

3. Initial Inspection of the Dataset

```
data.info()
data.describe()
data.isnull().sum()
```

4. Fix Data Type Issues

```
# Handling Missing Values

data['DC'] = data['DC'].astype(str).str.replace(' ', '',
regex=False)
data['FWI'] = data['FWI'].astype(str).str.replace(' ',
'', regex=False)

data['DC'] = pd.to_numeric(data['DC'], errors='coerce')
data['FWI'] = pd.to_numeric(data['FWI'], errors='coerce')
data.info()
```

Output -

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	244 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

5. Identify rows with missing values

```
null_rows = data[data.isnull().any(axis=1)]
print("Rows with missing values:\n", null_rows)
```

Output -

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  0.4      NaN  Sidi-Bel Abbes
```

6. Fill missing values using mode

```
for col in data.columns:
    data.loc[:, col] =
data[col].fillna(data[col].mode()[0])

# Drop remaining missing values
data.dropna(inplace=True)

data.isnull().sum()
```

Output -

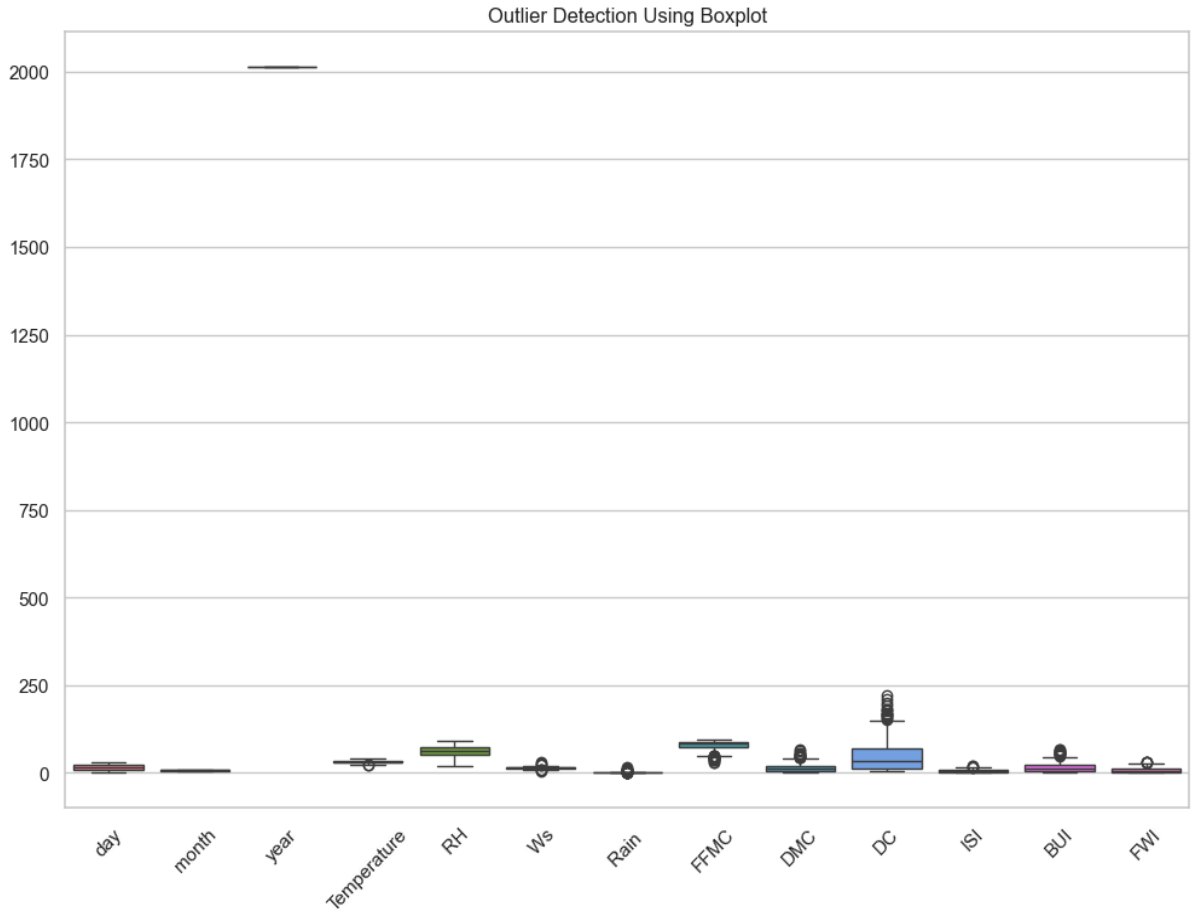
```
day            0
month          0
year           0
Temperature    0
 RH            0
 Ws            0
Rain           0
FFMC           0
DMC            0
DC             0
ISI            0
BUI            0
FWI            0
Classes        0
Region         0
dtype: int64
```

7. Outlier Detection Using Boxplots

```
plt.figure(figsize=(12, 8))
sns.boxplot(data=data)
plt.xticks(rotation=45)
```

```
plt.title("Outlier Detection Using Boxplot")
plt.show()
```

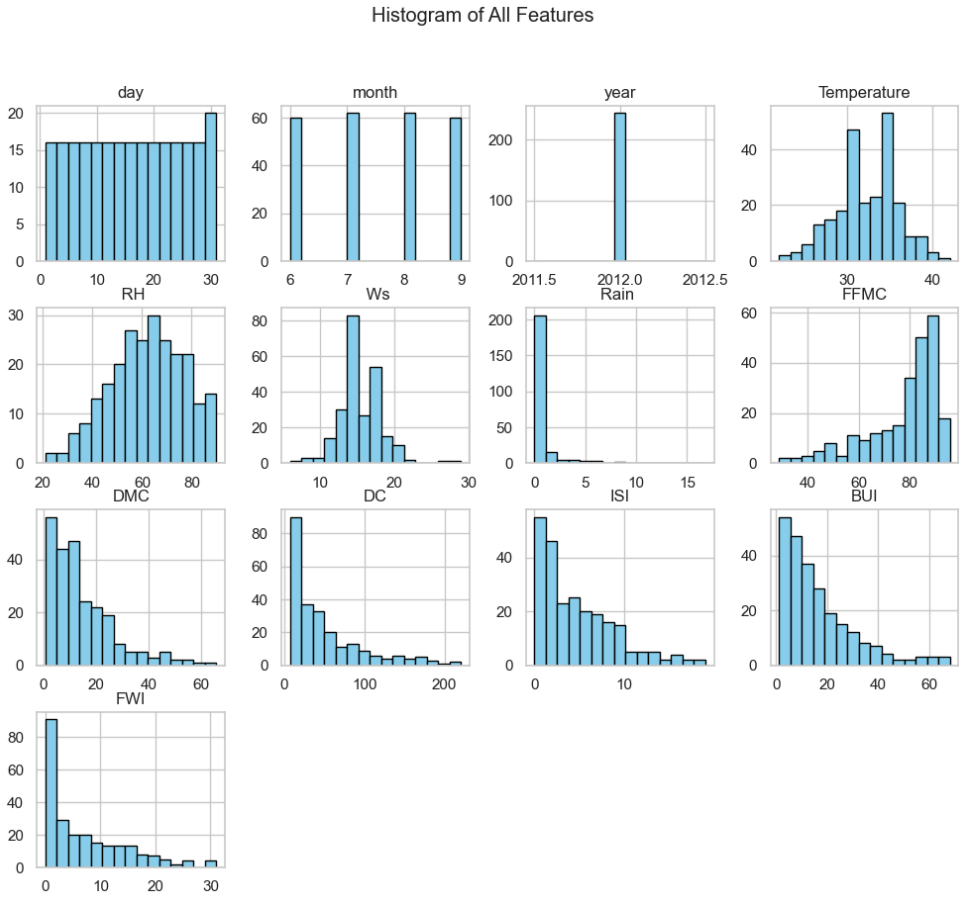
Output -



8. Histograms for All Features

```
data.hist(figsize=(12, 10), bins=15,color='skyblue',
edgecolor='black')
plt.suptitle("Histogram of All Features")
plt.show()
```

Output -

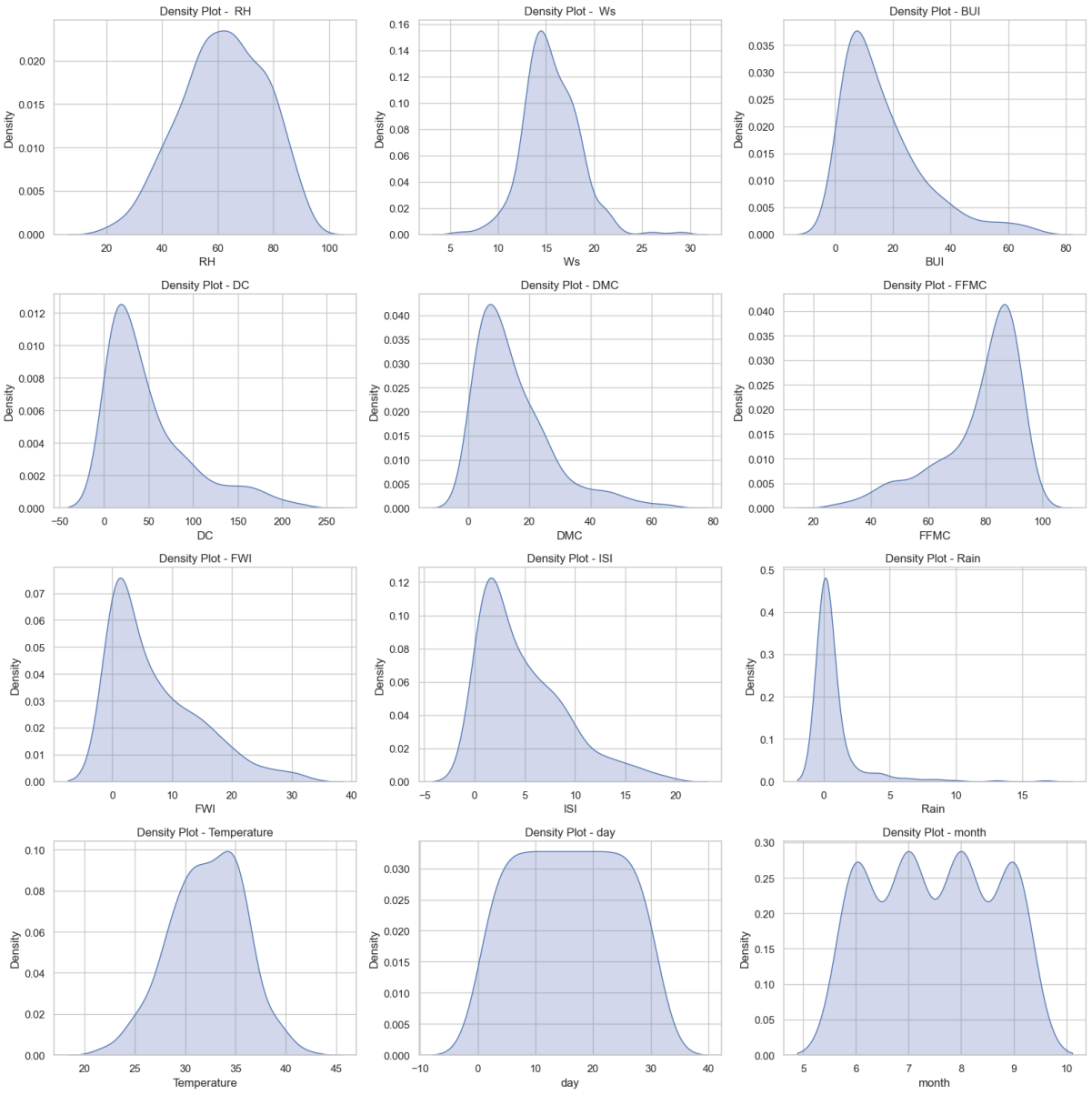


9. density plots for numerical features

```
# Select numerical columns
num_cols = sorted([col for col in data.columns if
data[col].dtype != "object"])

valid_cols = []

# Filter out columns with zero variance
for col in num_cols:
    if data[col].nunique() > 1:
        valid_cols.append(col)

cols = 3
rows = math.ceil(len(valid_cols) / cols)

plt.figure(figsize=(16, 4 * rows))

for i, col in enumerate(valid_cols, start=1):
    plt.subplot(rows, cols, i)
```

```
    sns.kdeplot(data[col], fill=True)
    plt.title(f"Density Plot - {col}")
    plt.xlabel(col)
    plt.ylabel("Density")

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

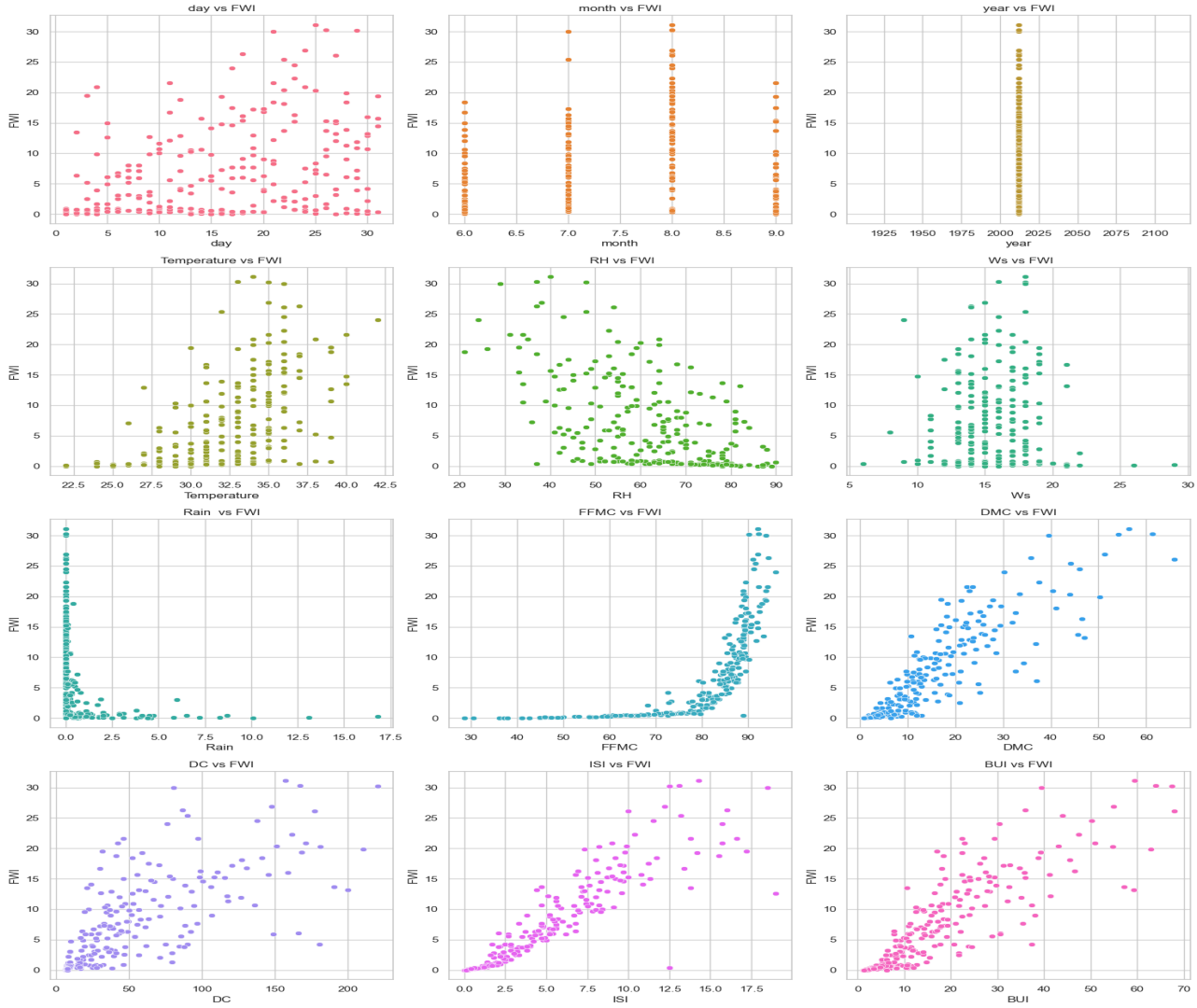
Output -



10.Scatter Plots of Numerical Features vs FWI

```
um_cols = [col for col in
data.select_dtypes(include=['float64', 'int64']).columns
if col != "FWI"]

palette = sns.color_palette("husl", len(num_cols))  #
colorful palette
cols = 3
rows = math.ceil(len(num_cols) / cols)

plt.figure(figsize=(15, 5 * rows))

for i, col in enumerate(num_cols, start=1):
    plt.subplot(rows, cols, i)
    sns.scatterplot(x=data[col], y=data["FWI"],
color=palette[i-1])
    plt.xlabel(col)
    plt.ylabel("FWI")
    plt.title(f"{col} vs FWI")

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

Output -

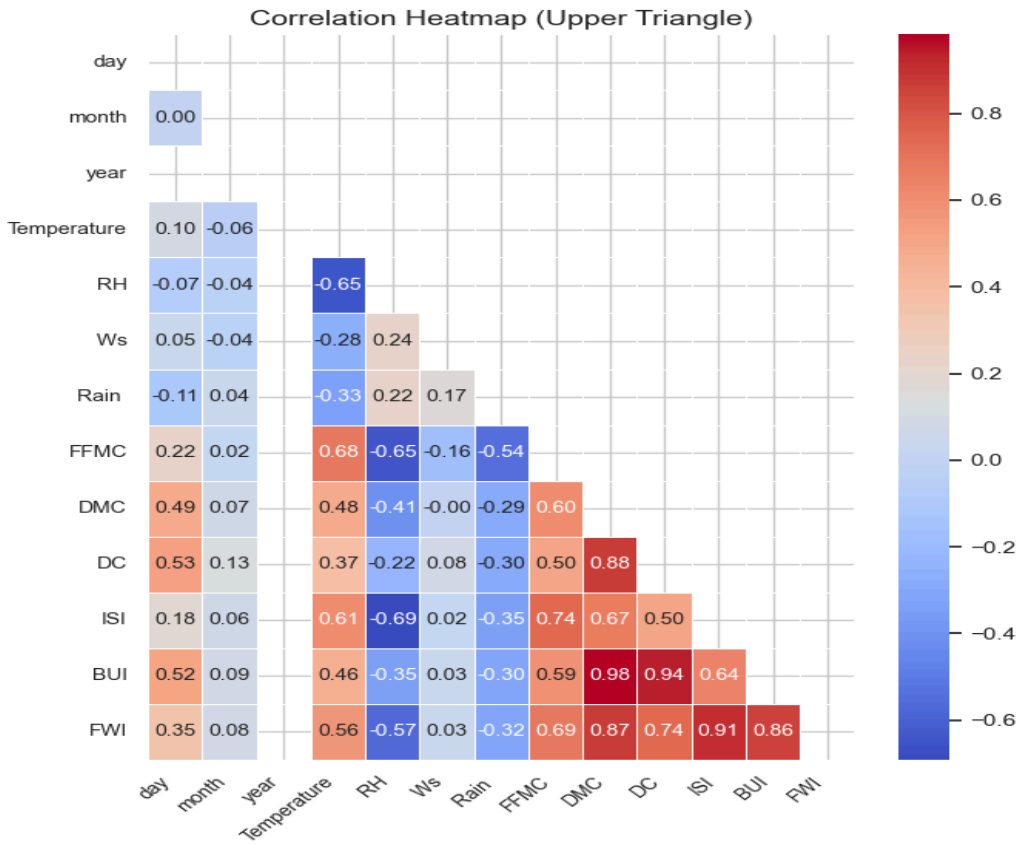


11. Correlation Heatmap

```
plt.figure(figsize=(7, 8))
corr =
data.select_dtypes(include=['float64','int64']).corr()


mask = np.triu(np.ones_like(corr, dtype=bool))

ax = sns.heatmap(
    corr,
    annot=True,
    cmap="coolwarm",
    fmt=".2f",
```

```
    mask=mask,
    linewidths=0.5,
    annot_kws={"size": 10}
)

plt.xticks(rotation=45, ha="right", fontsize=10)
plt.yticks(fontsize=10)

plt.title("Correlation Heatmap (Upper Triangle)",
fontsize=14)
plt.tight_layout()
plt.show()
```

Output -



12.Encoding Categorical Variables

```
if "Region" in data.columns:
    le = LabelEncoder()
    data["Region"] = le.fit_transform(data["Region"])
```

13.Save the Dataset

```
data.to_csv("cleaned_FWI_dataset.csv", index=False)
print("Cleaned dataset saved successfully!")
```

Output -

Cleaned dataset saved successfully!