Algerian Forest Fires: Forecasting Fire Risk Using FWI and Regression Models

Objective

The goal of this project was to analyze the Algerian Forest Fires dataset and develop regression models to predict the Fire Weather Index (FWI), a numerical indicator of fire danger, based on various weather and fire-related features.

Why This Project?

Forest fires are a growing concern worldwide, and being able to predict fire risk can aid in timely preventive measures. This dataset covers two regions in Algeria: **Bejaia** and **Sidi-Bel Abbes**, providing environmental measurements from June to September. This made it a great candidate for regression-based prediction.

Dataset Summary

Source: Algerian Forest Fires Dataset (June–September 2012)
Regions: Bejaia (Northeast Algeria) and Sidi Bel-Abbes (Northwest Algeria)
Instances: 244 (122 per region)
Classes: 138 'fire', 106 'not fire'
Features: 11 input features + 1 output class

Feature Dictionary & Business Relevance

Feature	Description	Why It Matters
day/month/year	Date of observation	Temporal features (removed before modeling)
Temperature (Temp)	Maximum daily temperature in Celsius (22 to 42)	Higher temps dry vegetation → higher fire risk
Relative Humidity (RH)	Humidity % (21 to 90)	Lower RH → more dryness → higher fire probability
Wind Speed (Ws)	Wind in km/h (6 to 29)	Wind spreads fires more rapidly
Rain	Precipitation in mm (0 to 16.8)	Rain reduces ignition probability
FFMC	Fine Fuel Moisture Code (28.6 to 92.5)	Dry surface fuel ignition potential
DMC	Duff Moisture Code (1.1 to 65.9)	Moisture in shallow organic layers
DC	Drought Code (7 to 220.4)	Deep fuel layer dryness — long-term fire potential

ISI	Initial Spread Index (0 to 18.5)	Combines wind & FFMC to estimate spread rate
BUI	Build-Up Index (1.1 to 68)	Fuel availability from DMC & DC
FWI	Fire Weather Index (0 to 31.1)	Final calculated index for fire danger — our target variable
Classes	'fire' or 'not fire'	Used for class distribution and correlation reference

Data Understanding and Cleaning

- **Identified structural issues**: One row (index 122) was a textual header separating the regions. It was removed for clean processing.
- **Added a 'Region' column**: Rows 0–121 were labeled Region 0 (Bejaia), and rows 123–end as Region 1 (Sidi-Bel).
- **Converted all numeric-looking object columns** (e.g., Temperature, RH, Wind Speed) to proper numeric data types.
- **Dropped irrelevant features**: Columns like day, month, and year were removed since they added little predictive value.
- **Mapped class labels**: Converted Classes into binary values — fire = 1, not fire = 0.

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246 entries, 0 to 245
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          246 non-null    object
 1   month        245 non-null    object
 2   year         245 non-null    object
 3   Temperature  245 non-null    object
 4    RH          245 non-null    object
 5    Ws          245 non-null    object
 6   Rain         245 non-null    object
 7   FFMC         245 non-null    object
 8   DMC          245 non-null    object
 9   DC           245 non-null    object
 10  ISI          245 non-null    object
 11  BUI          245 non-null    object
 12  FWI          245 non-null    object
 13  Classes      244 non-null    object
dtypes: object(14)
memory usage: 27.0+ KB
```

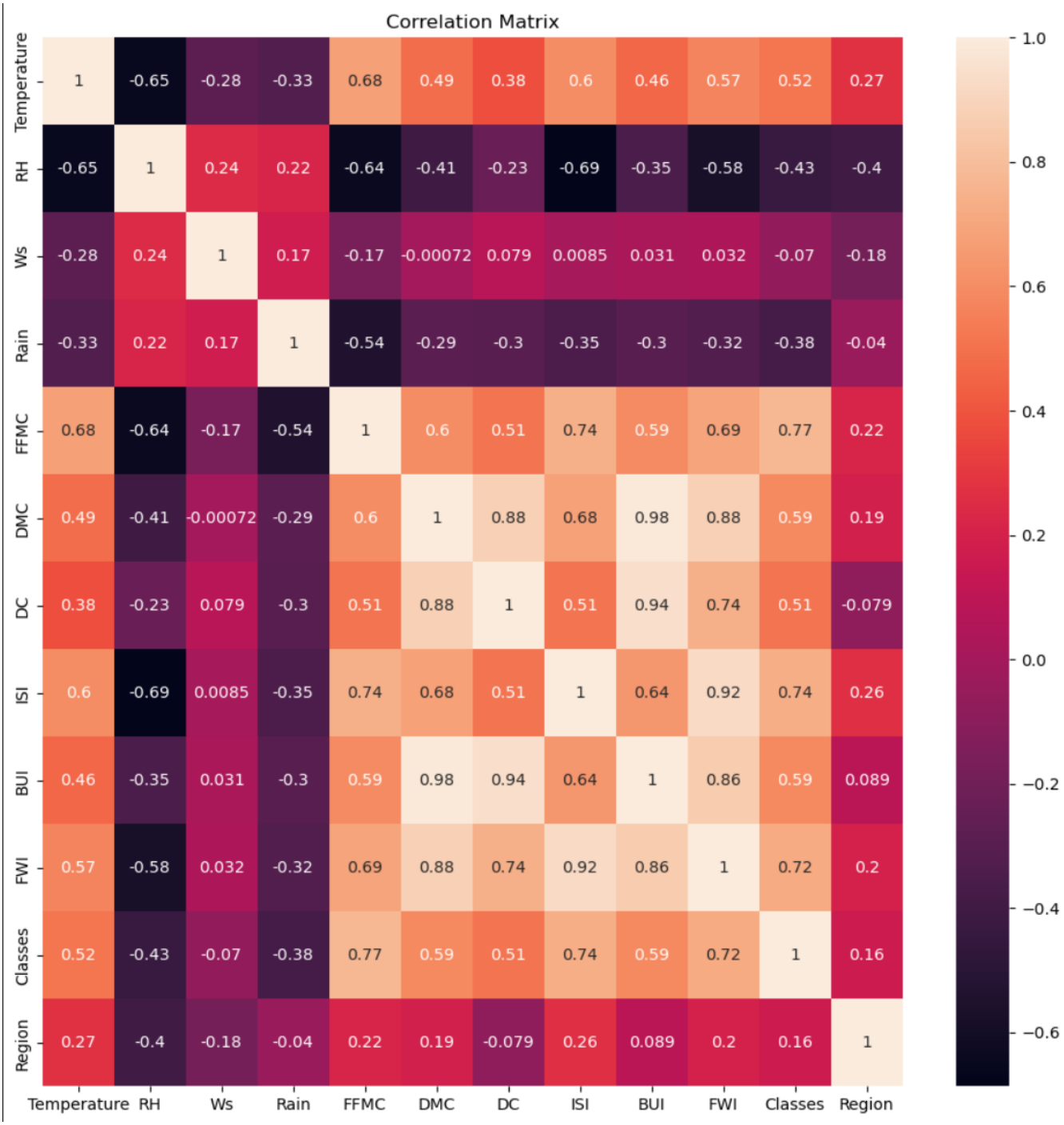
Before

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   day          243 non-null    int32
 1   month        243 non-null    int32
 2   year         243 non-null    int32
 3   Temperature  243 non-null    int32
 4   RH           243 non-null    int32
 5   Ws           243 non-null    int32
 6   Rain         243 non-null    float64
 7   FFMC         243 non-null    float64
 8   DMC          243 non-null    float64
 9   DC           243 non-null    float64
 10  ISI          243 non-null    float64
 11  BUI          243 non-null    float64
 12  FWI          243 non-null    float64
 13  Classes      243 non-null    object
 14  Region       243 non-null    int32
dtypes: float64(7), int32(7), object(1)
memory usage: 22.0+ KB
```

After

Preprocessing Steps

- **Handled missing values** by removing affected rows entirely.
- **Converted all columns** to appropriate numeric types.
- **Created df2**: a numeric dataset for modeling.
- **Dropped highly correlated feature BUI** to reduce multicollinearity. This was identified via a correlation heatmap.

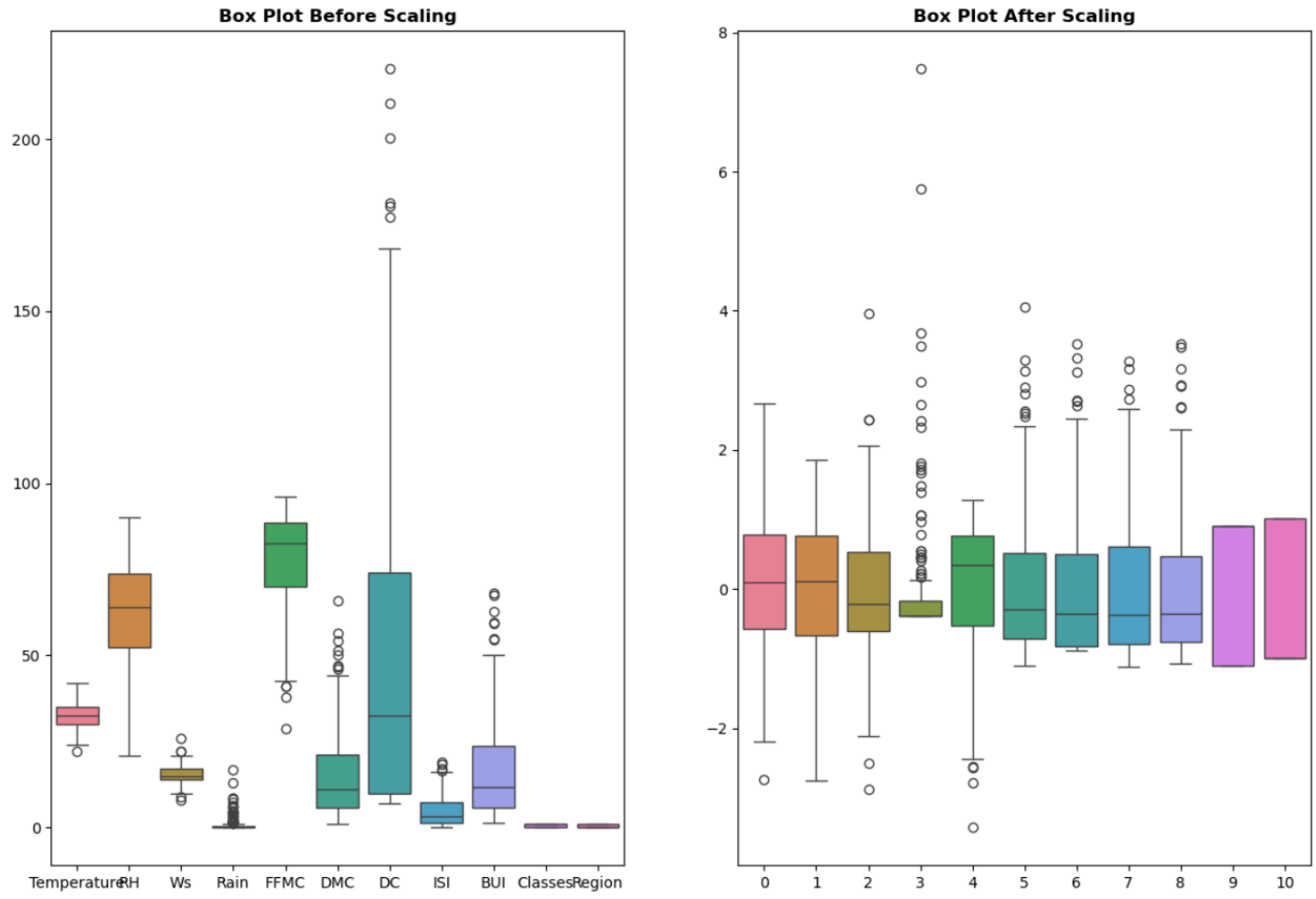


Data Scaling

Applied **StandardScaler** to normalize features.

Why - To ensure features are on the same scale so models like Ridge and Lasso can apply regularization correctly.

Impact - Boxplots showed all features centered similarly which improves training stability.

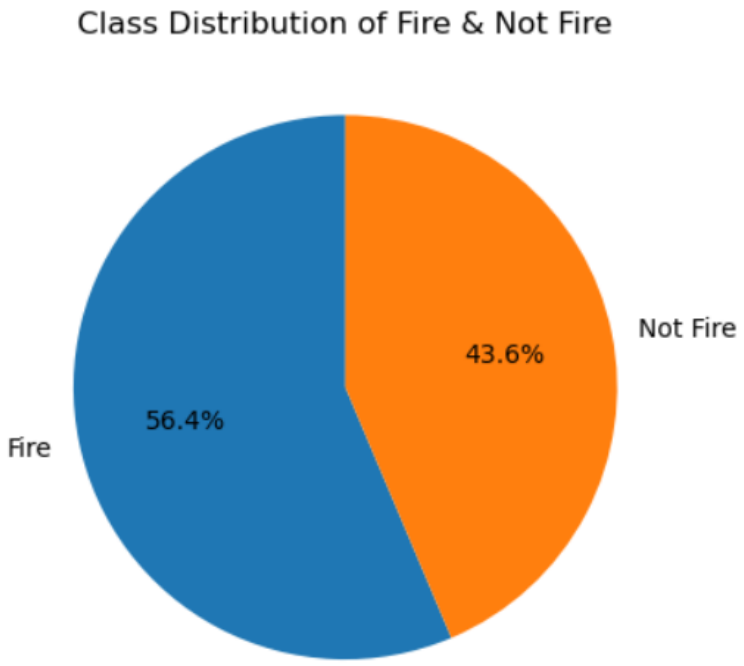


Impact of Scaling on Different Features

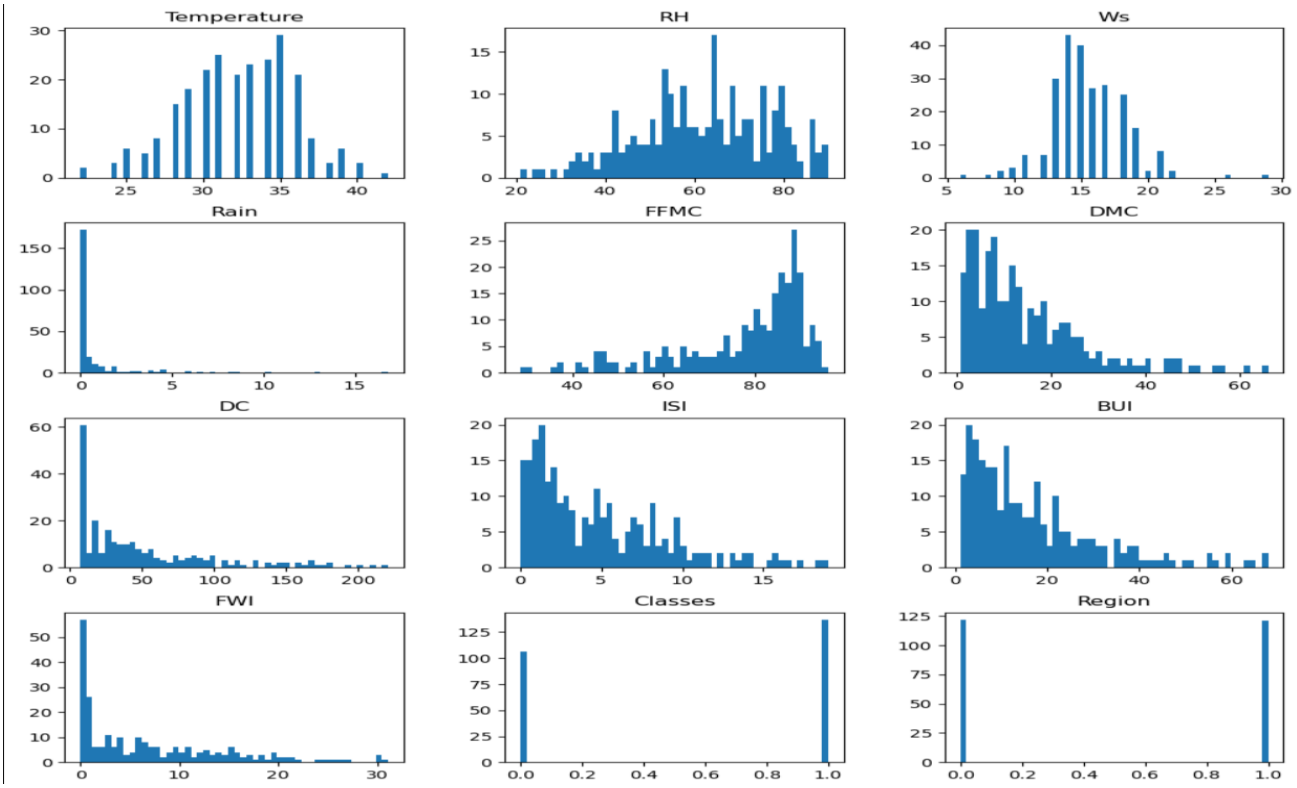
Exploratory Data Analysis

Class Distribution:

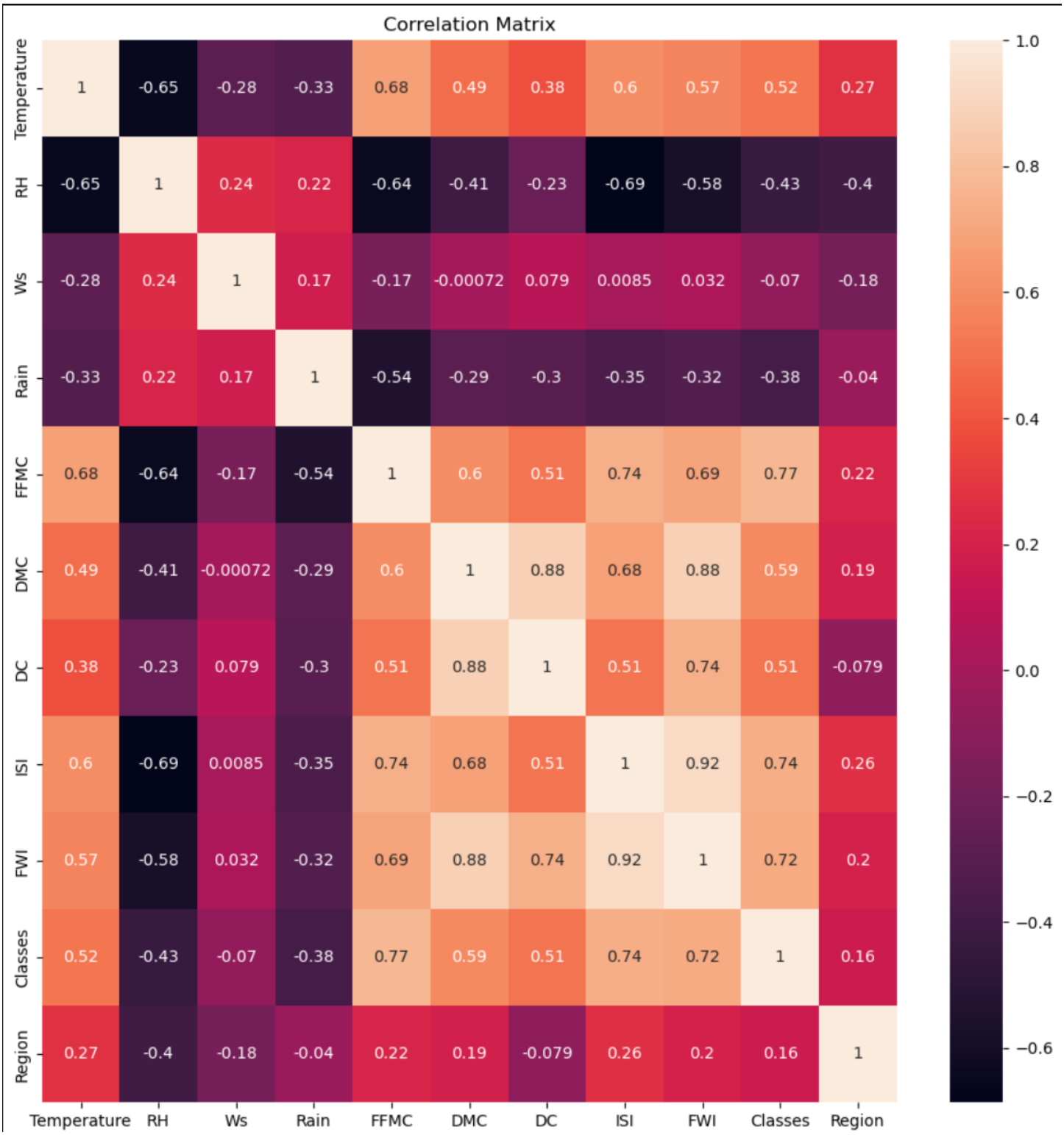
Pie chart revealed class imbalance: ~56% fire, ~44% not fire.



Histograms: Showed feature distributions and skewness.



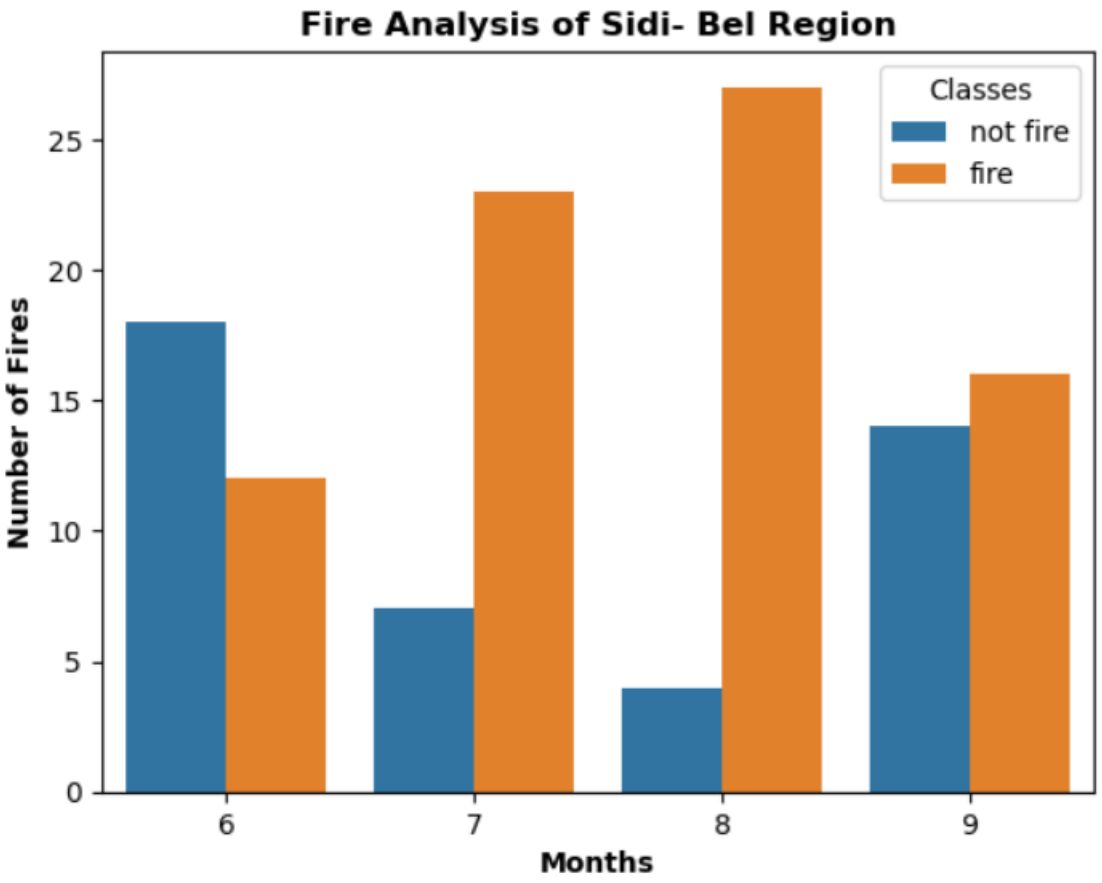
Heatmap: Helped visualize correlations, guiding feature selection and model design.



Monthly Trends:

Bar chart for Region 1 showed peak fire months (e.g., June–September).

Why : Helps validate seasonal fire patterns.

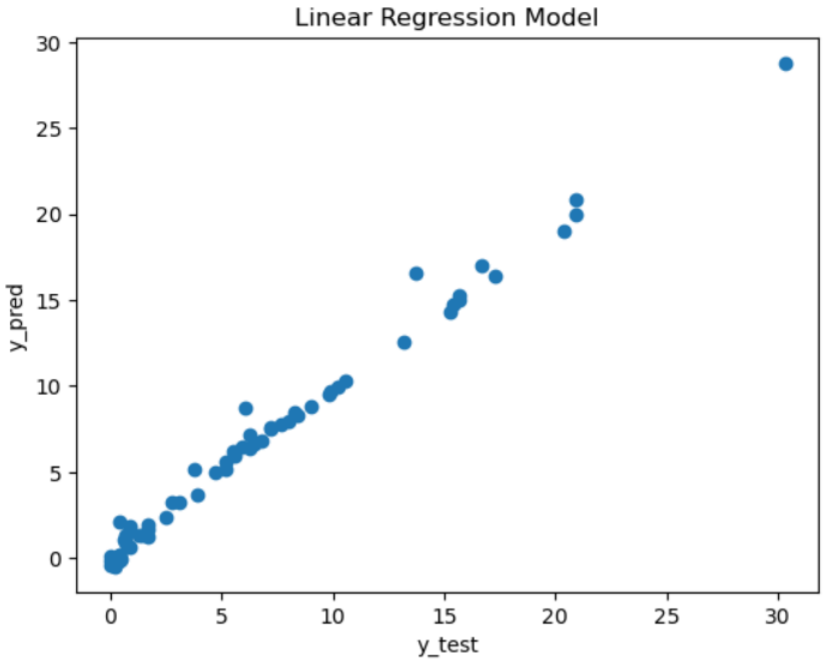


Modeling Approach

- **Target variable**: FWI (Fire Weather Index)
- **Features**: Included Temperature, RH, Wind Speed, Rain, FFMC, DMC, DC, ISI, Class, and Region
- **Train/Test Split**: 75% training, 25% testing

The models used in this project -

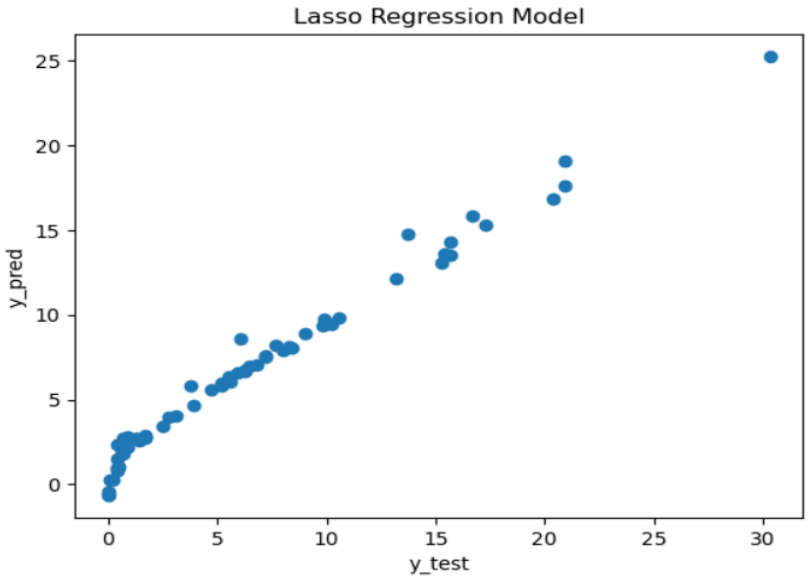
1. **Linear Regression**
   Linear regression was used as a baseline model. It attempts to model the relationship between the input features and the FWI value by fitting a straight line. Since the dataset was clean and mostly linearly related, this model performed exceptionally well.
   - **Output:** Continuous predicted FWI values (e.g., 3.24, 15.9, 29.2)
   - **Interpretation:** These values represent the estimated severity of fire risk given the environmental conditions.
   - **Performance:** R^2 = 0.989, MAE = 0.465
   - **Why**: Baseline model to measure how well basic assumptions fit.
   - **Impact**: Delivered an excellent R^2 of **0.989**, indicating strong predictive power.



2. Lasso Regression

Lasso Regression uses **L1 regularization**, which not only reduces the magnitude of coefficients but can also shrink some of them to **exactly zero** effectively performing **feature selection**.

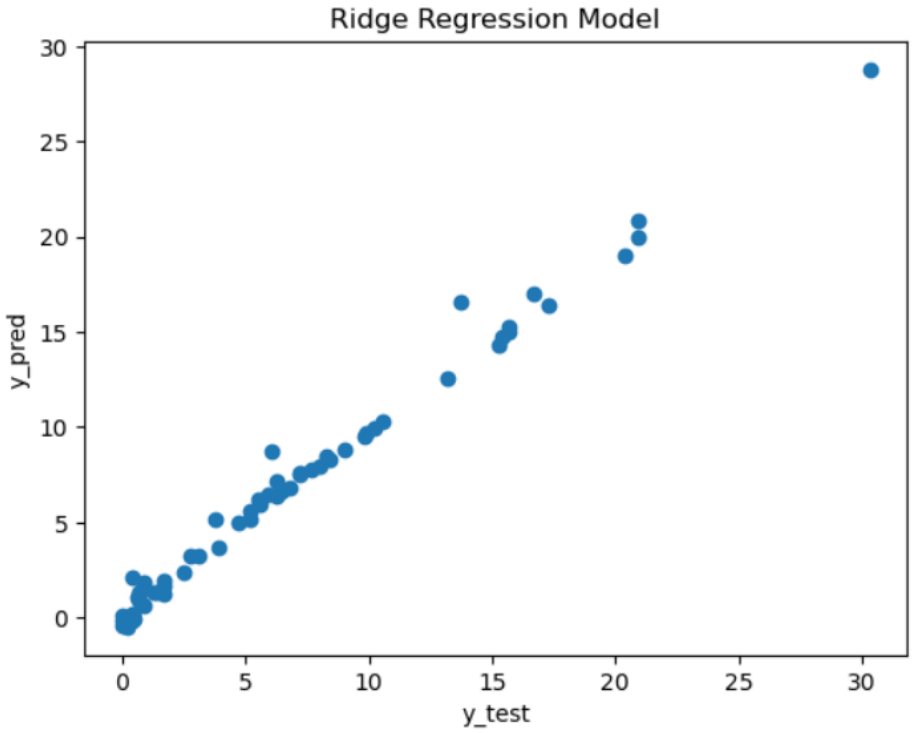
- **Output**: Similar continuous FWI predictions.
- **Interpretation**: Useful if we want a simpler model that uses fewer features.
- **Performance**: R^2 = 0.954, MAE = 1.08
- **Trade-off**: Slightly worse performance due to possible elimination of relevant variables.
- **Why?**: To perform feature selection and reduce overfitting.
- **Impact**: R^2 dropped to **0.954**, and MAE increased, indicating it may have dropped useful features.



3. Ridge Regression

Ridge Regression applies L2 regularization, which penalizes large coefficients but does not force any coefficient to zero. It is useful when multicollinearity exists or we want a more stable model.

- **Output:** Predicted FWI values across the same continuous range.
- **Performance:** R^2 = 0.987, MAE = 0.503
- **Benefit:** Prevents overfitting while still using all available features.
- **Why:** Adds penalty to coefficients without removing features.
- **Impact:** Balanced performance with R^2 of 0.987 which is a good trade-off between bias and variance.



Sample Prediction

```
sample_input =  np.array([[32, 38, 18, 0.0, 88.0, 40.0, 120.0, 12.0, 52.3, 1, 1]])
scaled_input = scaler.transform(sample_input)
predicted_fwi = ridge_model.predict(scaled_input)
```

```
C:\Users\mrsal\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but
StandardScaler was fitted with feature names
  warnings.warn(
```

```
predicted_fwi
```

```
array([23.24502485])
```

Prediction of FWI based on sample values

Model Export for Reusability

Exported the scaler and final Ridge model using pickle for later deployment:

Import pickle

pickle.dump(scaler, open('scaler.pickle', 'wb'))

pickle.dump(ridge_model, open('ridge.pickle', 'wb'))

Key Learnings and Reflections

- **Cleaning matters**: Structural fixes + proper typing = smooth modeling
- **Standardization is essential**: Models performed better after scaling
- **Lasso can be overaggressive**: Regularization is powerful but needs tuning
- **Ridge regression worked best** in this case due to stable and interpretable coefficients

Model	Mean Absolute Error (MAE)	R^2 Score 0.989	Reason for Accuracy Difference
Linear Regression	0.465	0.989	Captured linear relationships well with clean, preprocessed data. No penalty terms.
Ridge Regression	0.503	0.987	Slight regularization helped prevent overfitting without reducing relevant feature impact.
Lasso Regression	1.08	0.954	Some important features may have been penalized to zero, reducing prediction power.

Model Comparison with Key Metrics