National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Shiyamal Kumar
x23253142

School of Computing
National College of Ireland

Supervisor: Dr Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

StudentName	Shiyamal Kumar		
Student ID:	x23253142		
Programme:	MSc Data Analytics	**Year:**	2024 - 2025
Module:	Research Practicum		
Lecturer:	Dr Christian Horn		
Submission Due Date:	15-09-2025		
Project Title:	Enhancing Forest Fire Severity Prediction using Machine Learning: Evaluation of Linear Regression, Random Forest and Gradient Boosting Models		
Word Count:	1967	**Page Count:**	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shiyamal Kumar

Date: 15-09-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shiyamal Kumar
x23253142

1 Software and Environment Preparation

This section describes the main tools and technologies that have been employed in the project. The project has been implemented with the help of the Python programming language in the Jupyter Notebook interface. The requirements are Python 3.12+, JupyterLab, and libraries such as NumPy 1.26+, Pandas 2.2+, Scikit-learn 1.5+, Matplotlib 3.9+, and Seaborn 0.13+. The implementation environment is a system with Windows 11 or later, or a Unix-based system, with at least 8 GB of RAM and an Intel i5 (or equivalent) processor, running the Anaconda distribution of packages for simplified package and environment management.

2 System Specifications

This part explains the hardware and system configuration used to conduct the experiments for model development. The technical specifications of the machine included a 64-bit processor, 8 GB RAM, and a 1.6 GHz Intel Core i5 10th Generation processor. Jupyter Notebook and the required Python libraries were utilized for implementation. Each experiment and visualization was carried out in a repeatable and consistent environment.

3 Data Preparation, Transformation, Feature Engineering

```
# Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from scipy.stats import boxcox
```

Figure 1: Required Libraries

These are some of the Python libraries that are needed when it comes to the implementation and evaluation of the deep learning and machine learning models to predict forest fire risks. Pandas and NumPy handle data manipulation and Number crunching, whereas Matplotlib and Seaborn assist data visualization. The modules scikit-learn, which as preprocessing and model-building train_test_split, StandardScaler, regression models such as LinearRegression, RandomForestRegressor, GradientBoostingRegressor, are used. The performance of the model can be evaluated through such parameters as MAE, MSE, and R2. Skewed data

distribution can be made normal through the Box-Cox transformation of SciPy, which enhances the accuracy of a model.

```
# Load Dataset
df = pd.read_csv('forestfires.csv')
```

```
# Display first few rows
print("Head of Dataset:")
print(df.head())
```

```
Head of Dataset:
   X  Y month  day  FFMC   DMC     DC  ISI  temp  RH  wind  rain  area
0  7  5   mar  fri  86.2  26.2   94.3  5.1   8.2  51   6.7   0.0   0.0
1  7  4   oct  tue  90.6  35.4  669.1  6.7  18.0  33   0.9   0.0   0.0
2  7  4   oct  sat  90.6  43.7  686.9  6.7  14.6  33   1.3   0.0   0.0
3  8  6   mar  fri  91.7  33.3   77.5  9.0   8.3  97   4.0   0.2   0.0
4  8  6   mar  sun  89.3  51.3  102.2  9.6  11.4  99   1.8   0.0   0.0
```

Figure 2: Loading and Previewing the Dataset

The importation of the forest fire dataset is done by the use of pandas.read_csv(), which allows further analysis and modeling of the dataset. The data uses forestfires.csv as a name and has several features that include temperature, humidity, wind, rainfall, and fire weather index (FFMC, DMC, DC, ISI) among other location (X, Y) and time (month, day) properties. The most straightforward recommendations are executing df.head(), exploring the first several rows, and getting an idea about the structure and nature of the data. This is one of the most important steps in order to check the successful loading and be ready to preprocess, visualise, and perform predictive models of the forest fire risk.

```
# Dataset Information
print("\n Dataset Info:")
print(df.info())


# Descriptive Statistics
print("\n Descriptive Statistics:")
print(df.describe())


# Check Missing Values
print("\n Missing Values:")
print(df.isnull().sum())


# Column data types
print("\n Data Types:")
print(df.dtypes)


# Check and remove duplicates
duplicate_count = df.duplicated().sum()
print("\n Duplicate Records:", duplicate_count)

if duplicate_count > 0:
    df = df.drop_duplicates()
    print(" Duplicates removed. New shape:", df.shape)


# Unique values in categorical columns
print("\n Unique Values in 'month' and 'day':")
print("month:", df['month'].unique())
print("day:", df['day'].unique())
```

```
 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X       517 non-null    int64
 1   Y       517 non-null    int64
 2   month   517 non-null    object
 3   day     517 non-null    object
 4   FFMC    517 non-null    float64
 5   DMC     517 non-null    float64
 6   DC      517 non-null    float64
 7   ISI     517 non-null    float64
 8   temp    517 non-null    float64
 9   RH      517 non-null    int64
 10  wind    517 non-null    float64
 11  rain    517 non-null    float64
 12  area    517 non-null    float64
dtypes: float64(8), int64(3), object(2)
memory usage: 52.6+ KB
None

 Descriptive Statistics:
                X           Y        FFMC         DMC          DC         ISI  \
count  517.000000  517.000000  517.000000  517.000000  517.000000  517.000000
mean     4.669246    4.299807   90.644681  110.872340  547.940039    9.021663
std      2.313778    1.229900    5.520111   64.046482  248.066192    4.559477
min      1.000000    2.000000   18.700000    1.100000    7.900000    0.000000
25%      3.000000    4.000000   90.200000   68.600000  437.700000    6.500000
50%      4.000000    4.000000   91.600000  108.300000  664.200000    8.400000
75%      7.000000    5.000000   92.900000  142.400000  713.900000   10.800000
max      9.000000    9.000000   96.200000  291.300000  860.600000   56.100000

             temp          RH        wind        rain         area
count  517.000000  517.000000  517.000000  517.000000   517.000000
mean    18.889168   44.288201    4.017602    0.021663    12.847292
std      5.806625   16.317469    1.791653    0.295959    63.655818
min      2.200000   15.000000    0.400000    0.000000     0.000000
25%     15.500000   33.000000    2.700000    0.000000     0.000000
50%     19.300000   42.000000    4.000000    0.000000     0.520000
75%     22.800000   53.000000    4.900000    0.000000     6.570000
max     33.300000  100.000000    9.400000    6.400000  1090.840000
```

Feature	Missing Values	Data Type	Unique Values (if applicable)
X	0	int64	—
Y	0	int64	—
month	0	object	mar, oct, aug, sep, apr, jun, jul, feb, jan, dec, may, nov
day	0	object	fri, tue, sat, sun, mon, wed, thu
FFMC	0	float64	—
DMC	0	float64	—
DC	0	float64	—
ISI	0	float64	—
temp	0	float64	—
RH	0	int64	—
wind	0	float64	—
rain	0	float64	—
area	0	float64	—

Figure 3: Initial Dataset Exploration and Cleaning

The current dataset, named forest fire, has 517 records composed of 13 variables, which cover meteorological, spatial, and temporal ones. The df.info() confirms that there are no missing data and that the data is well-organized with proper data type: integers, spatial coordinates, float, fire index, and weather conditions, object, and categorical data (month and day).
Descriptive statistics demonstrate valuable information. Relative humidity averages 44.3 percent with the average temperature of 18.9 o C. There is a huge skew here in the area burned with the fire, with a mean of 12.85, with the highest being 1090.84, indicating the existence of outliers.

It has been checked to prove that there were no duplicates, including 4 dupped entries which were deleted, leaving 513 rows in the data. The fact that categorical variables are one-of-a-kind was also proven: there are 12 months in the column of month, and it also includes all 7 weekdays in the day column, which is why the time range is really comprehensive.
Transactions such as the review of the structure, analysis of descriptive statistics, identification of data types, and cleaning of duplicates are the preconditions of effective modeling. They make sure that the data is correct, clean, and can be used to preprocess, feature engineering, and fire risk prediction using deep learning.

```
# Convert categorical features to numeric using one-hot encoding
df_encoded = pd.get_dummies(df, columns=['month', 'day'], drop_first=True)

# Log transform of skewed 'area' variable (add 1 to avoid log(0))
df_encoded['log_area'] = np.log1p(df_encoded['area'])

# Drop the original 'area' column
df_encoded.drop('area', axis=1, inplace=True)
```

Figure 4: Data Transformation and Encoding

In order to subject this dataset to modeling, categorical columns month and day were encoded numerically, and since to prevent collinearity, the drop_first argument must be set to True. Its skewed variable, area, was log-transformed as log 1p () to normalize its distribution to guarantee better performance of the model. The original column with the name area was thus removed and replaced by another column, which has a new attribute named log_area to have it be further analyzed.

The correlation heatmap gives an illustrative view of correlation patterns of all numeric and coded variables or features of the data. It demonstrates that the variables such as DMC, DC and ISI that are important fire weather indices are positively correlated with one another at a high level. The variables will most probably affect one another since, in general, the drier environment (indicated by larger DMC and DC values) tends to contribute to the quicker spread of the fire, which is reflected in the ISI. But in the case of the target variable log_area, the results indicate that it has only weak positive relationships with some predictors, those being temperature, DMC, and ISI. [***Refer to appendix -1***].

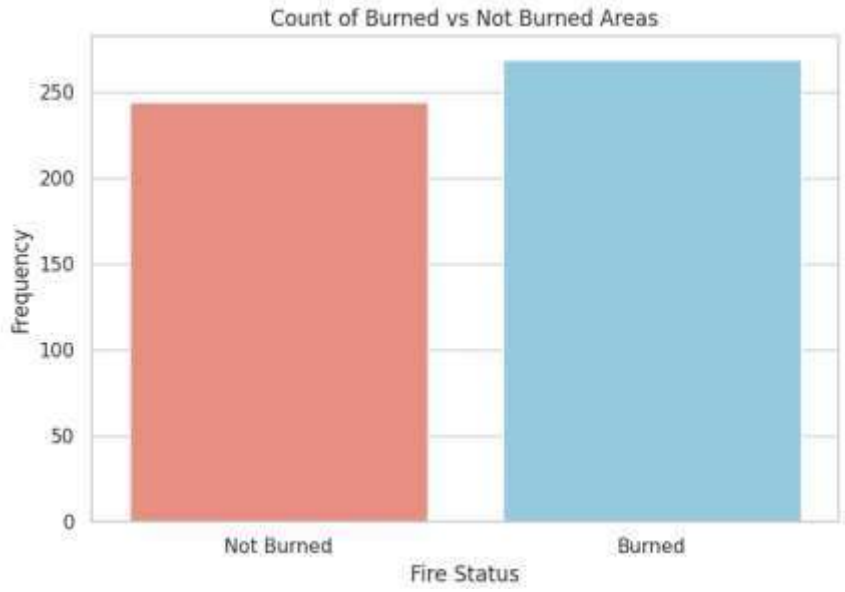
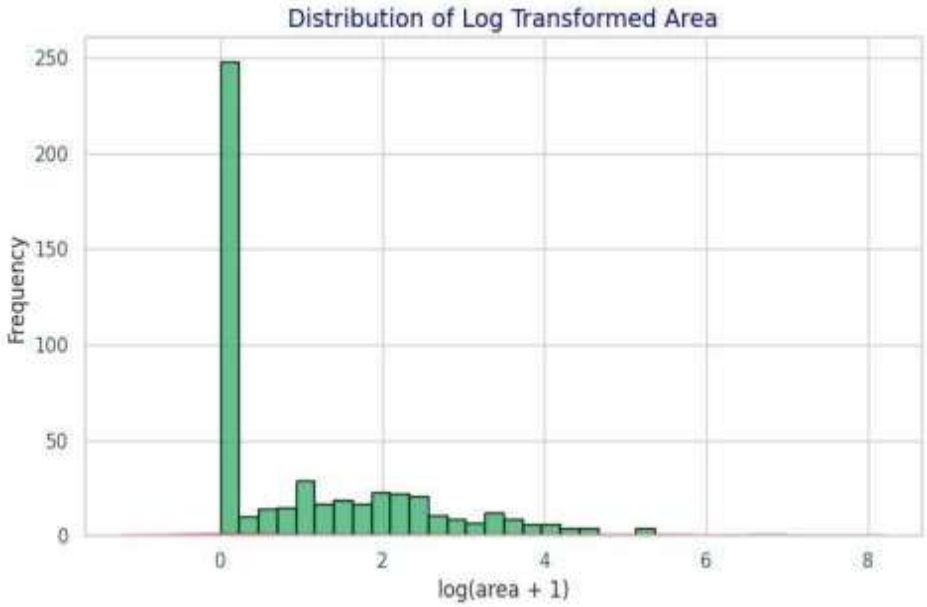


Figure 5: Count of Burned vs Not Burned Area

The bar graph compares burned and burned areas. It divides the dataset into two classes depending on whether some area had been burnt or not. This plot denotes that this data is moderately balanced with a bit more burned values than the non-burned ones. The balance provides high importance when classification-based models are used, so that it will not result in the model existing in other biases to the data set. It also validates the importance of modeling effort that aims at foretelling the incidence and the scale of fire outbreaks.



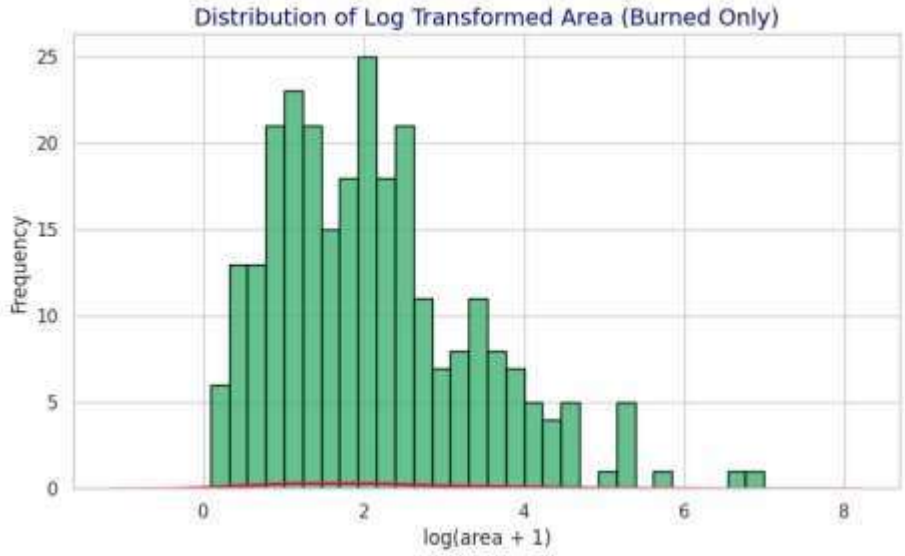


Figure 6: Comparing burned versus not burned areas.

The distribution of the burned area in log form is shown in the plots. This upper histogram of burned and non-burned incidents has a very sharp peak at zero, which means a very high number of times that there was an occurrence of fire, but no area was burnt. This was a very skewed distribution requiring a log transformation. The lower plot isolates just burned cases and has a smoother and bell shaped distribution.

The pairplot provides an overall visual correlation of the relationship between selected features, that is, temperature, relative humidity (RH), wind, rain, and log-transformed burned area, by using fire status (Burned or Not Burned). It shows that there is a substantial negative correlation between temperature and RH whereby high temperatures are associated with low humidity. This can be seen in the burn areas where the orange (Burned) indicators are more saturated in the warmer and drier places. Also, the distribution of log_area shows that more burned observations occur in the middle and high temperature intervals[***Refer to appendix 2***].

Boxplots (both on the full and on the burned-only datasets) aid in such feature distributions as well as outliers. As an example, the median temperature and speed of winds are just a bit increased in burned locations. Compared to non-burnt cases, the RH is also lower in burned cases, an observation that supports the fact that dryness serves as a fire spread. Moreover, FFMC (Fine Fuel Moisture Code), DMC (Duff Moisture Code), and ISI (Initial Spread Index) are, in general, more in burned areas, a fact that proves these fire indices as a reliable predictor. ***Refer to appendix -3***].

The single scatterplots associating each feature with log_area give clear indications of non-linear or weak relationships. FFMC and DMC are examples of variables that have a positive pattern with log_area, that is, the results are higher in cases of greater burned areas. Conversely, other features such as rain and RH have a wider distribution compared to log_ area, which shows that there is no or little strong linear relationship[***Refer to appendix -4***].

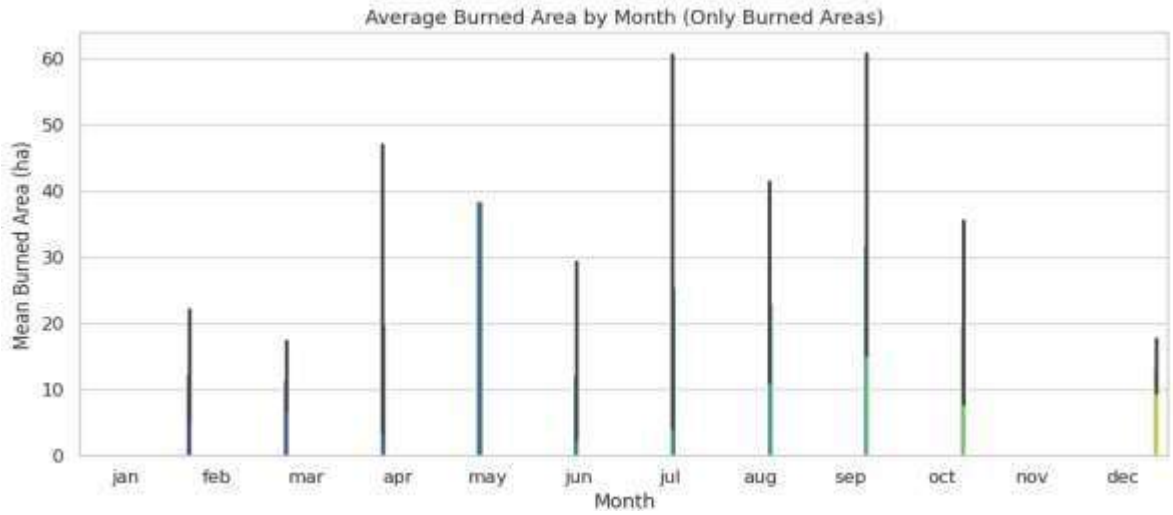


Figure 7: Average Burned Area by Month (Only Burned Areas)

The plot only indicates the average monthly burned area of burnt area. The statistics indicate that July and September have the highest averages of burned areas, with over 60-hectare averages per month. This is in line with the dry summer period, when the weather is hot and the humidity is low, which encourages the spread of fires. On the other hand, the average burned areas during months such as February, March, and December are low, indicating that they are not as fire-prone since there is more moisture as well as cooler temperatures.

The relationship has been plotted between rain and the log of the burned area. It reveals a particular contrasting trend that most fires occur at times when there is no rain or it is scarce. The Log (area+1) has its maximum at rain levels approaching the origin of zero, and nearly every data point is associated with this origin. This reinstates the fact that rainfall plays a significant role of inhibiting the spread of fire. Scanty rainfall figures indicate that once the rain falls, the likelihood of big ground fires is almost zero [***Refer to appendix 5***].

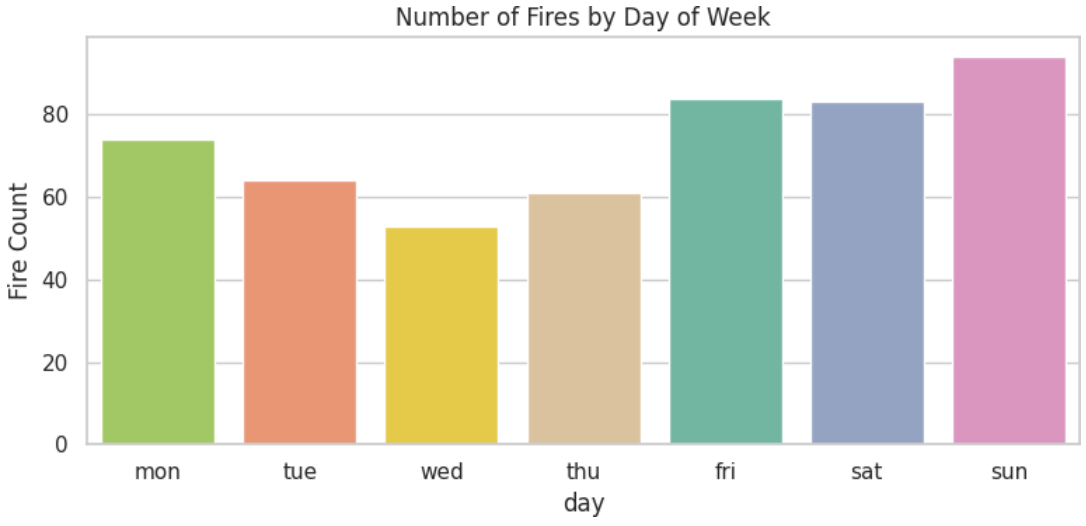


Figure 8: Number of Fires by Day of Week

The plot portrays how many fires occur on various days of the week. The Sunday fires are highest, and the others follow, namely, Friday and Saturday. This may create a cause to correlate with more human activity or careless activity during the weekend, including campfires or open burning. Business days in the middle of the week, such as Wednesday and Thursday, record the lowest cases perhaps because the routine weeks restrict such events. The trend is an indication that citizen education and precautionary steps, especially on weekends, would help curtail the instances of fire outbreaks due to human interference.

```
# In[63]:

# Convert categorical features to numeric using one-hot encoding
df_encoded = pd.get_dummies(df, columns=['month', 'day'], drop_first=True)

# Log transform of skewed 'area' variable (add 1 to avoid log(0))
df_encoded['log_area'] = np.log1p(df_encoded['area'])

# Drop the original 'area' column
df_encoded.drop('area', axis=1, inplace=True)

# Create a status column to label burned vs non-burned
df['status'] = np.where(df['area'] > 0, 'Burned', 'Not Burned')

# Recreate log_area to make sure 0 area stays visible at 0
df['log_area'] = np.log1p(df['area'])

# Add target column
df_encoded['status'] = np.where(df_encoded['log_area'] > 0, 'Burned', 'Not Burned')

# Define numerical features
numerical_features = ['temp', 'RH', 'wind', 'rain', 'FFMC', 'DMC', 'DC', 'ISI']
```

Figure 9: Data Preprocessing and Feature Engineering

In order to analyze the dataset, categorical variables, including month and day variables, were transformed into numerical one-hot encoding since it would be a failure to analyze the first category of the category since it is the one that would produce a multicollinearity error. Log-transformation of the skewed 'area' variable with the log1p function to deal with zeroes was applied and the original column with the 'area' column removed. One more column of binary type, called status, was inserted to differentiate between the Burned and the Not Burned areas. In the original DataFrame, the area transformed using log was also kept. Finally, some important numerical characteristics were specified, i.e. temperature, relative humidity, wind, rainfall, and fire indices (Mambile *et al*. 2024).

```python
# Outlier report function
def cap_outliers_report(df, features):
    outlier_report = {}
    for feature in features:
        Q1 = df[feature].quantile(0.25)
        Q3 = df[feature].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR

        outliers = df[(df[feature] < lower) | (df[feature] > upper)]
        outlier_count = outliers.shape[0]
        total = df.shape[0]
        percent = (outlier_count / total) * 100

        outlier_report[feature] = {
            'outliers': outlier_count,
            'total': total,
            'percent_outliers': round(percent, 2)
        }
    return pd.DataFrame(outlier_report).T
```

```python
# Apply to all data and burned-only
df_burned = df[df['area'] > 0].copy()
print(" Outlier Report: All Data")
print(cap_outliers_report(df, numerical_features))

print("\n Outlier Report: Burned Only")
print(cap_outliers_report(df_burned, numerical_features))

# Apply Box-Cox transformation on FFMC
if (df['FFMC'] > 0).all():
    df['FFMC_boxcox'], fitted_lambda = boxcox(df['FFMC'])
    print(f" Skewness of FFMC after Box-Cox: {pd.Series(df['FFMC_boxcox']).skew():.3f}")
    print(f" Box-Cox lambda: {fitted_lambda:.3f}")
    df_encoded['FFMC'] = df['FFMC_boxcox']
    print(" FFMC has zero or negative values; Box-Cox not applicable.")

# Outlier capping function
def cap_outliers(df, features):
    df_capped = df.copy()
    for feature in features:
        Q1 = df_capped[feature].quantile(0.25)
        Q3 = df_capped[feature].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Cap values
        df_capped[feature] = np.where(df_capped[feature] < lower_bound, lower_bound,
                                      np.where(df_capped[feature] > upper_bound, upper_bound,
                                               df_capped[feature]))
    return df_capped

# Apply it
df_capped = cap_outliers(df, numerical_features)

# Check new shape
print("Original shape:", df.shape)
print("Shape after capping:", df_capped.shape)

# Cap outliers in encoded dataset
numerical_features_encoded = ['temp', 'RH', 'wind', 'rain', 'FFMC', 'DMC', 'DC', 'ISI']
df_encoded_no_outliers = cap_outliers(df_encoded, numerical_features_encoded)
```

```
 Outlier Report: All Data
      outliers  total  percent_outliers
temp       2.0  513.0              0.39
RH        12.0  513.0              2.34
wind      13.0  513.0              2.53
rain       8.0  513.0              1.56
FFMC      53.0  513.0             10.33
DMC       17.0  513.0              3.31
DC        17.0  513.0              3.31
ISI       14.0  513.0              2.73

 Outlier Report: Burned Only
      outliers  total  percent_outliers
temp      13.0  269.0              4.83
RH         3.0  269.0              1.12
wind      12.0  269.0              4.46
rain       2.0  269.0              0.74
FFMC      29.0  269.0             10.78
DMC       16.0  269.0              5.95
DC        33.0  269.0             12.27
ISI        6.0  269.0              2.23
 Skewness of FFMC after Box-Cox: -0.262
 Box-Cox lambda: 12.665
Original shape: (513, 16)
Shape after capping: (513, 16)
```

Figure 10: Outlier Detection and Treatment

A report of outliers was prepared to the entire dataset as well as a reduced one containing only the burned parts. It is the work of the function to detect outliers based on the method of Interquartile Range (IQR) and to report the number and the percentage of outliers per feature. Both datasets had the largest number of outliers (about 10 percent) in FFMC. FFMC was also transformed to straighten its skew with a Box-Cox transformation that attained skewness -0.262 using the transformation value of 12.665. Outlier capping was applied that restricted outliers to values more than 1.5 * IQR. The shape of dataset was not changed in the post-capping so that the consistency of data could also be maintained with a reduced effect of extreme values (Soualah *et al*. 2024).

```python
# Prepare features and target
X = df_encoded_no_outliers.drop('log_area', axis=1)
y = df_encoded_no_outliers['log_area']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X.drop('status', axis=1, errors='ignore'),
    y,
    test_size=0.2,
    random_state=42
)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```python
# Model dictionary
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(random_state=42)
}
```

```python
# Function to evaluate and print results
def evaluate_model(name, model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    preds = model.predict(X_test)

    r2 = r2_score(y_test, preds)
    mae = mean_absolute_error(y_test, preds)
    rmse = np.sqrt(mean_squared_error(y_test, preds))

    print(f"\n{name} Performance:")
    print(f"R² Score: {r2:.4f}")
    print(f"MAE: {mae:.4f}")
    print(f"RMSE: {rmse:.4f}")
    return model
```

```python
# Train and evaluate all models
trained_models = {}
for name, model in models.items():
    trained_models[name] = evaluate_model(name, model, X_train_scaled, y_train, X_test_scaled, y_test)
```

Figure 11: Model Training

The data being used was split into features (X) and target variable (log_area). Categorical variables had already been encoded, and the status column was not a part of modeling. Data was divided in the training and testing ratios 80-20. The application of StandardScaler transformed the inputs so that the model can perform better by normalizing the features. Linear Regression, Random Forest Regressor and Gradient Boosting Regressor models were trained. The evaluation methodology was determined to evaluate models based on R^2 score, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). All the models were trained on the training set and tested on the scaled test set and the results were printed out to compare the performance of the models.

References

Mambile, C., Kaijage, S. and Leo, J., 2024. Application of deep learning in forest fire prediction: a systematic review. IEEE Access.

Soualah, L., Bouzekri, A. and Chenchouni, H., 2024. Hoping the best, expecting the worst: Forecasting forest fire risk in Algeria using fuzzy logic and GIS. Trees, Forests and People, 17, p.100614.

Appendix

Appendix -1

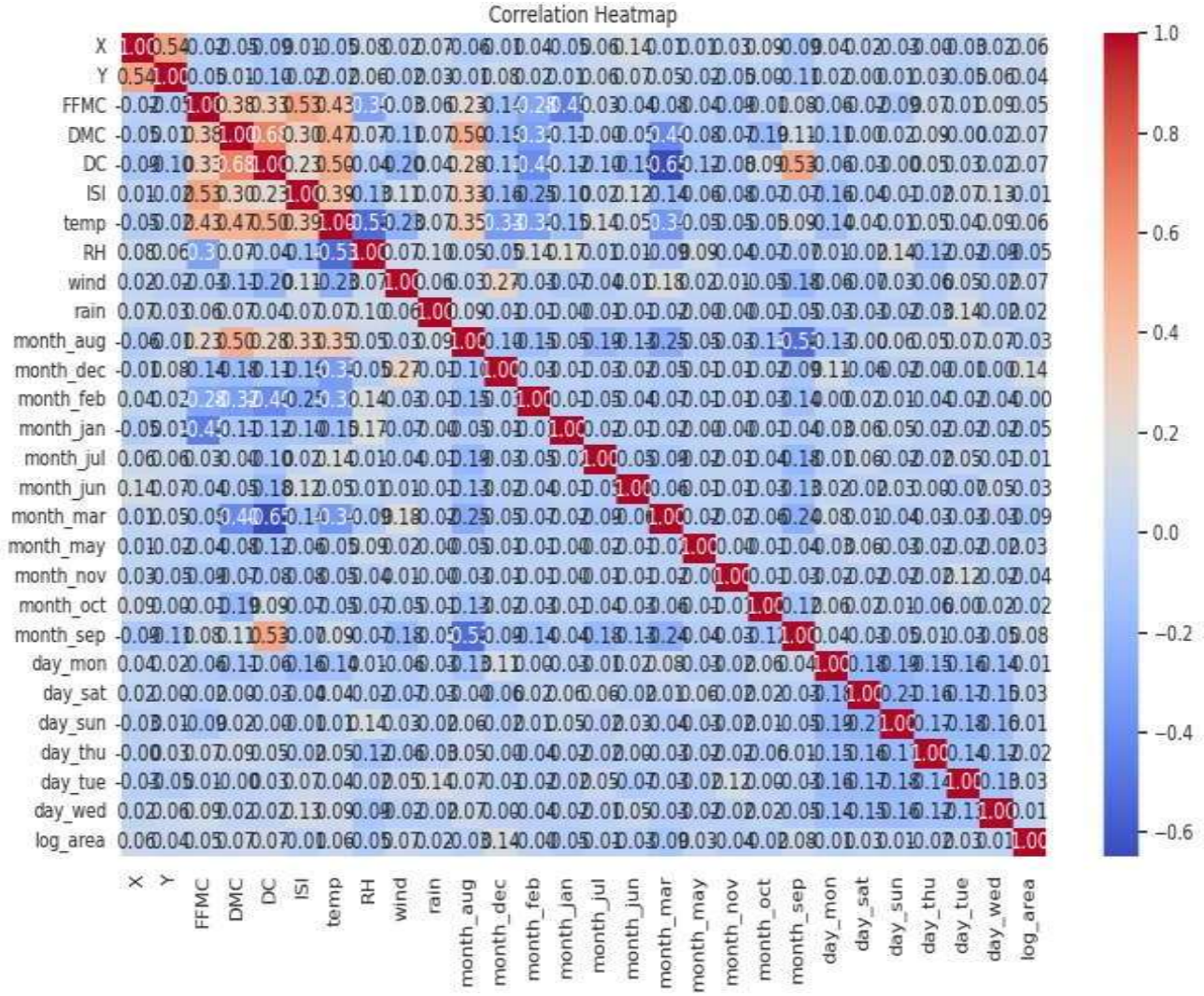
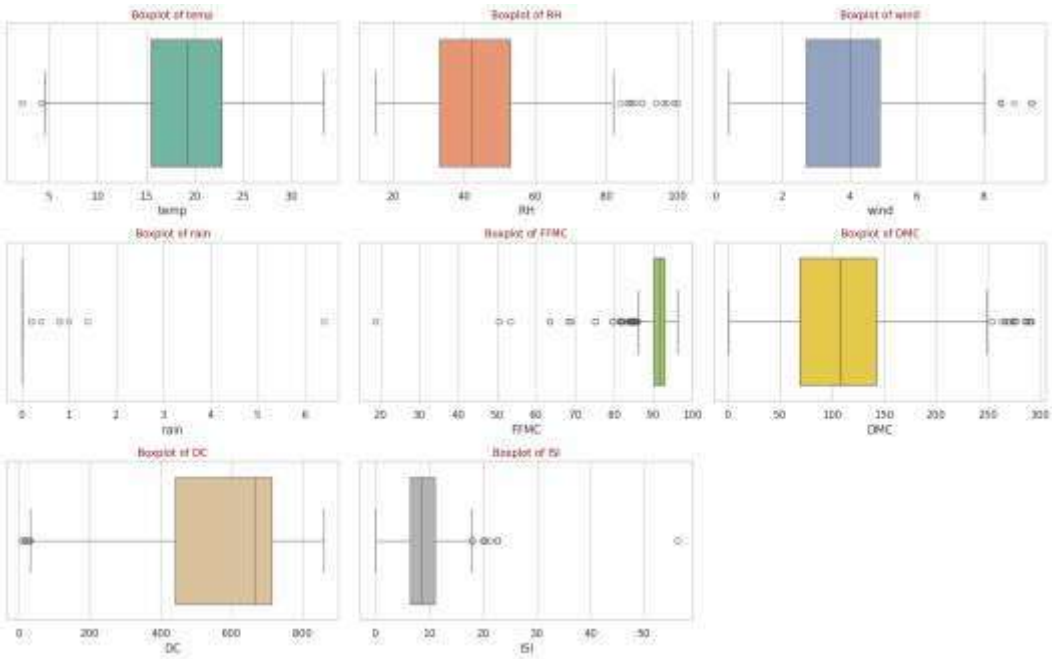


Figure 1: Correlation Heatmap

Appendix -2

Figure 2: Pairplot of Selected Features with Burn Status

Appendix -3



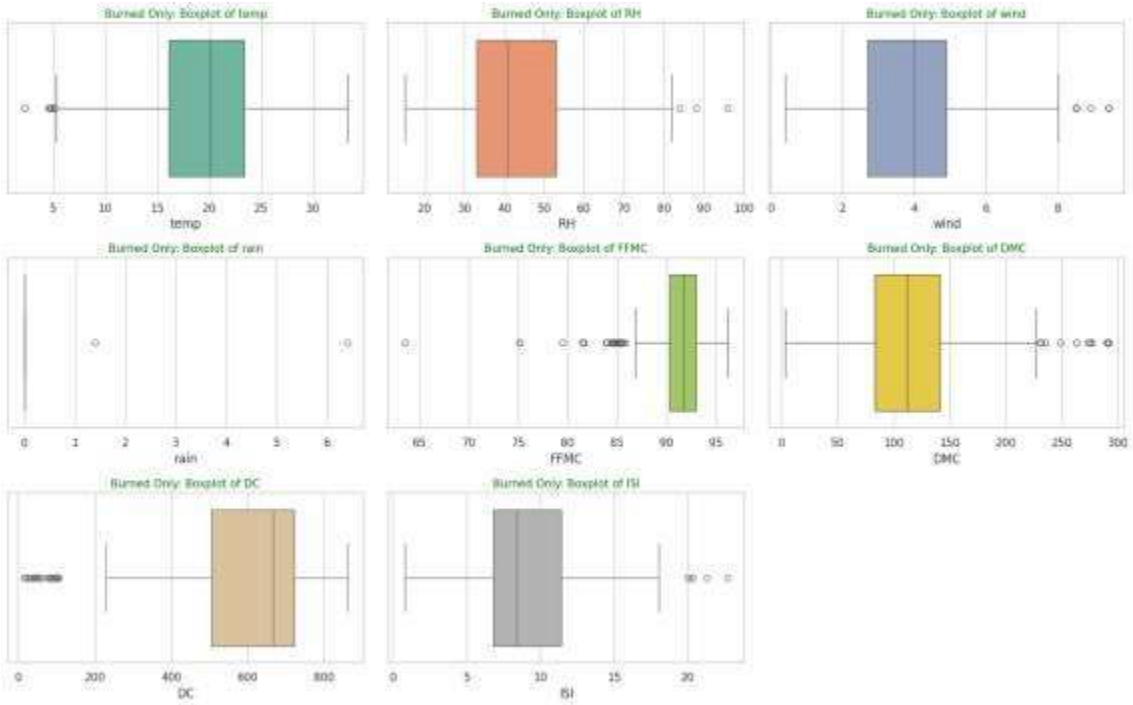


Figure 3: Boxplots- both for the entire dataset and for burned-only

Appendix -4

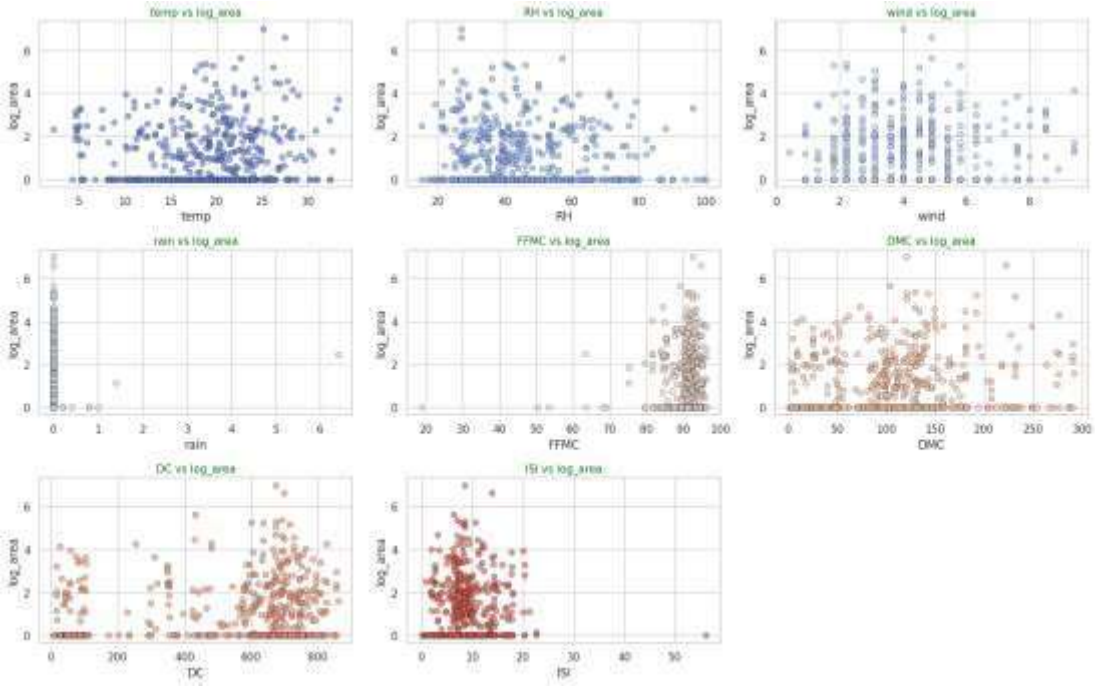


Figure 4: Scatterplots comparing each feature to log_area

Appendix -5

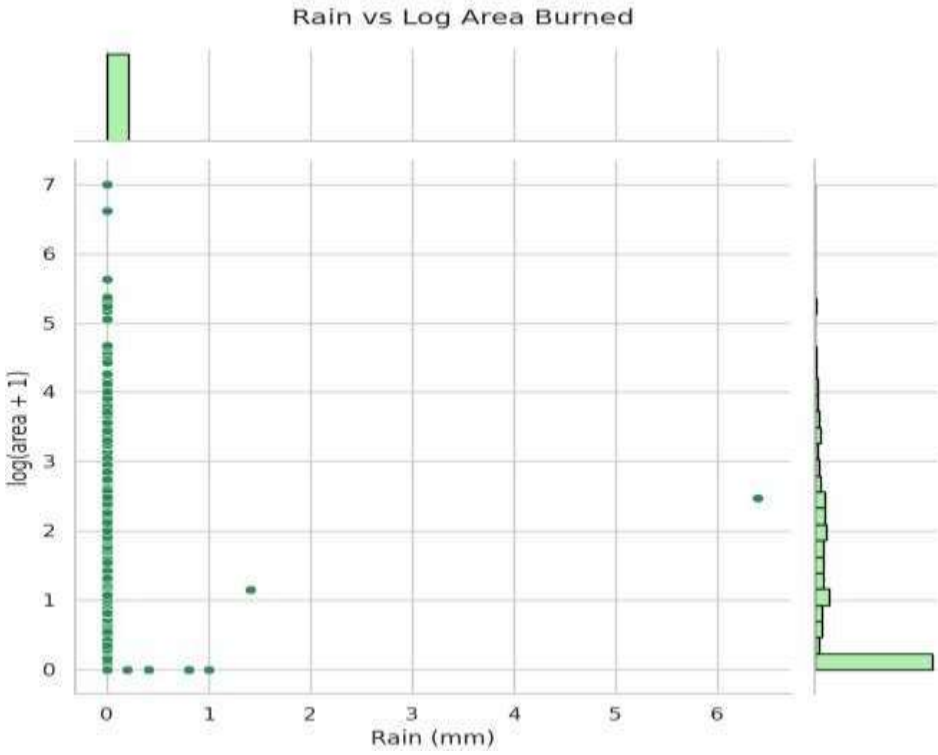


Figure 5: Rain vs Log Area Burned