Key Findings Summary

1. **Business Objectives:** The project aims to create prediction models for forest fires based on weather data to help forest management organisations allocate resources and avert fire damage. We utilised both classification and regression models:
    - Regression models are used to predict the burnt area and optimise resource utilisation.
    - Classification models categorise flames depending on their severity.
2. **Data Insights:**
    - The target variable (burned area) is significantly skewed, with many tiny fires and few major ones. The log transformation was used to normalise the distribution.
    - Predictors such as temperature, wind speed, and fine fuel moisture code (FFMC) have substantial connections with fire severity.
3. **Model Performance:** Decision Tree, Random Forest, Support Vector Machine (SVM), k-Nearest Neighbors (kNN), and Artificial Neural Network (ANN).
    - Random Forest outscored others with the lowest RMSE (0.188) and greatest accuracy for regression tasks.
    - For classification, the Artificial Neural Network (ANN) obtained the highest accuracy (44.74%); however, it still battled with class imbalances.
4. **Key Predictors:** Temperature and Relative Humidity Interaction (Temp_RH_Interaction), Drought Code (DC), and Wind Speed were the best predictors for regression and classification tasks.
5. **Challenges:**
    - A significant class imbalance in fire severity categories exists, with most occurrences falling into the “No Damage” category, hindering accurate forecasts for more severe fire events.
    - Outliers and skewness in variables such as rain and FFMC necessitated log modifications to enhance model performance.

Table of Contents

I. Business Objectives

The main goal of this project is to create prediction models that use weather data and fire indications to group forest fires and measure how impact they are. Preventing forest fires is very important for protecting the environment, saving lives, and keeping the economy stable. The goal is to give environmental and forest management agencies useful information for predicting the levels of fire damage and making more accurate fire intensity classifications by using both classification and regression models.

- **Classification Models:** These models group fires into groups based on how bad they are (no damage, low, moderate, high, very high). Classification models can help make the best use of resources by figuring out how destructive a fire might be, which can reduce reaction times and keep things from worsening (Jain et al., 2020).

- **Regression Models:** These models put a number on the burned area and give accurate predictions that help spread resources and planning for the future (Rodrigues & de la Riva, 2014). This data helps people choose where to put their emergency resources.

This two-model method works with key performance indicators (KPIs) including:

- Shortening reaction time: Finding high-risk situations early lets resources be used more quickly.
- Optimising wildfire resources: Accurate estimates of how destructive the fire is and how much area it has burned to ensure that firefighters put their efforts where they are needed most.
- Minimising fire damage: Data-based predictions let forest management teams move ahead of time, reducing the area that fires can damage.

These tools provide useful information for more than just forest management teams. These predictions can help insurance companies determine the extent of risk and help states make policies that have the least negative effects on society, the economy, and the environment (Alcasena et al., 2016). Additionally, adding these models to early warning systems improves preparedness and helps protect weak spots, eventually lowering the overall effects of forest fires on the environment and society.

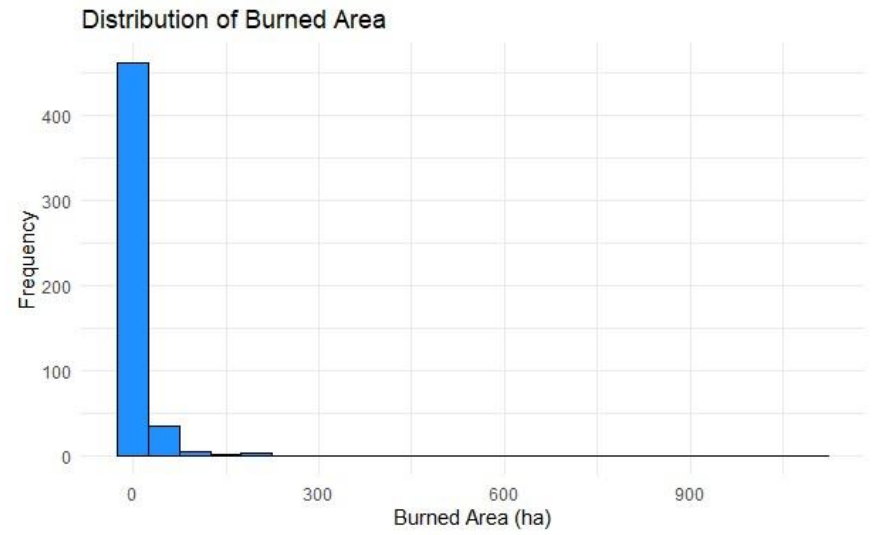
II. Data Exploration

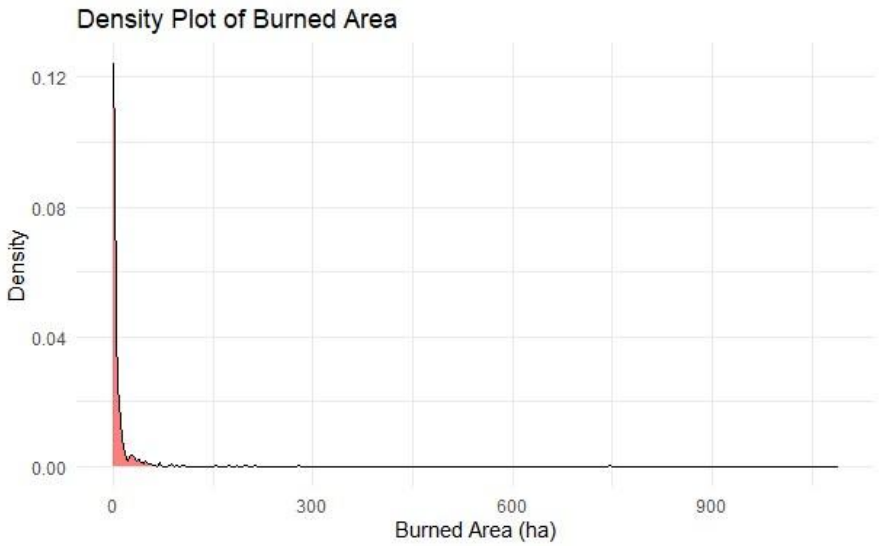
2.1 Target Variable Exploration

Analysing the raw and transformed distributions helps one grasp the distribution and features of the target variable, Burned Area. Understanding this will enable one to choose suitable transformations to prepare the data for modelling.

2.1.1 Histogram and Density Plots of Burned Area (Raw)

The raw value histogram and density displayed to determine the distribution of the burned area and to look for any skewness or outliers.





The histogram of the burned area (in hectares) has a distinct right skew, with most observations clustered at 0 and very few values dispersed over the upper range. This indicates that the dataset is dominated by tiny fires, with just a few instances involving significant burnt areas. Most of the burned areas are less than 200 hectares; as the burned area rises, the abrupt drop-off indicates the unusual occurrence of substantial fires.

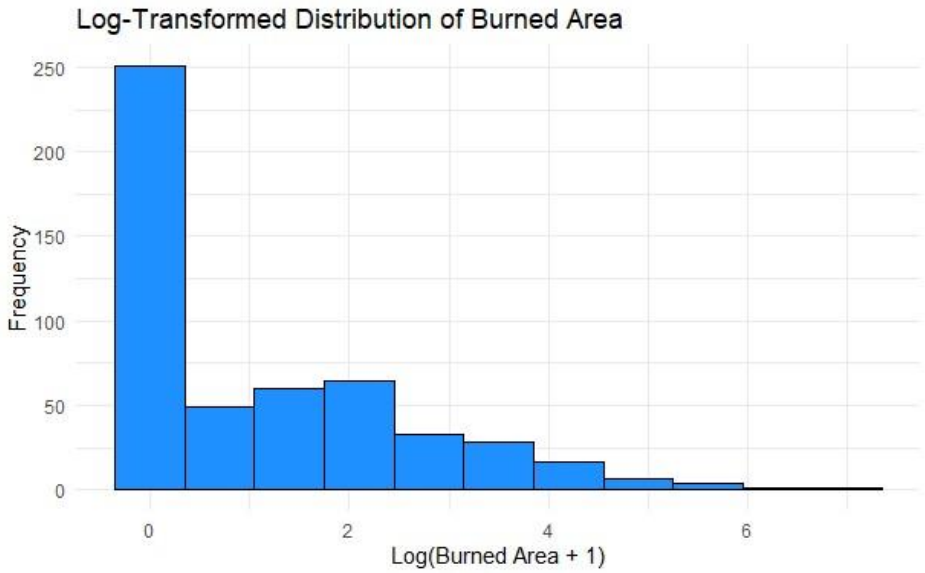
Statistical Interpretation:

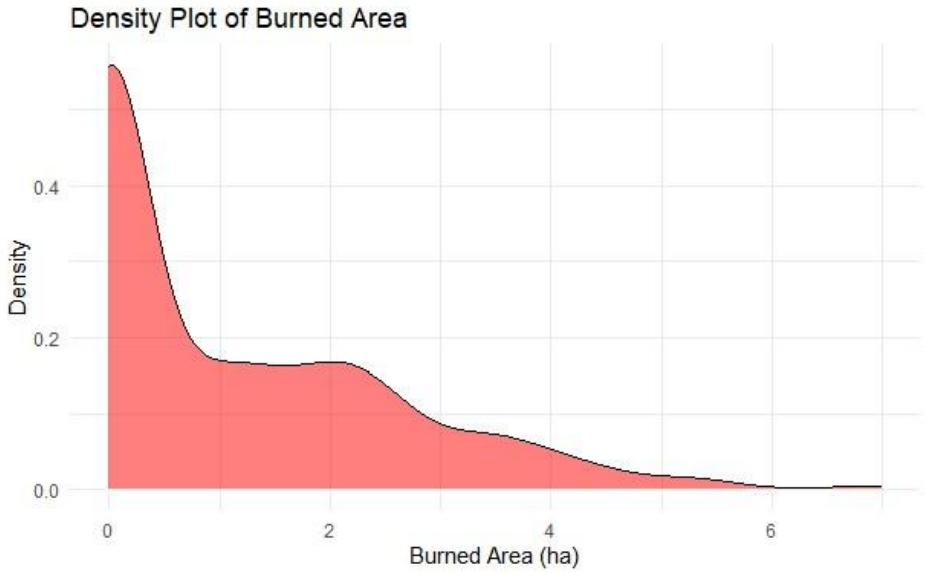
- Mean: 12.89 ha (hectares)
- Median: 0.54 ha
- Skewness: 12.73 (extremely right-skewed)
- Kurtosis: 190.09 (significantly peaked distribution)

The burnt area is significantly skewed, with several flames burning zero or tiny land. A few vast fires significantly impact the mean value, as seen by the distribution's tail. This significant skewness implies that prediction models might benefit from putting the burnt area into a more normal distribution. The median (0.54 hectares) is much lower than the mean, showing a significant effect from outliers.

2.1.2 Histogram and Density Plots of Log-Transformed Burned Area

In its raw form, the burned area distribution is severely skewed, which might negatively impact model performance. To remedy this, we logarithmically alter the burned area variable. By lowering skewness, the log transformation improves the data's modelling suitability.





In contrast to the raw burned area, the log-transformed histogram displays a more dispersed and normalised distribution. The change lessens the effect of extreme outliers, such as big fires, but leaves the peak at the left side, representing minor burned areas (around 0), visible. Consequently, the distribution becomes more balanced and more straightforward to predict.

Statistical Interpretation:

- Mean: 1.11 (log-transformed units)
- Median: 0.43 (log-transformed units)
- Skewness: 1.21 (moderately skewed)

- Kurtosis: 0.93 (approaching normal distribution)

The log-transformed burned area lowered the skewness from 12.73 to 1.21, making the distribution more controllable and closer to the standard shape. This adjustment greatly enhances the dataset's applicability for predictive modelling since most machine learning algorithms usually need to distribute target variables to make correct predictions.

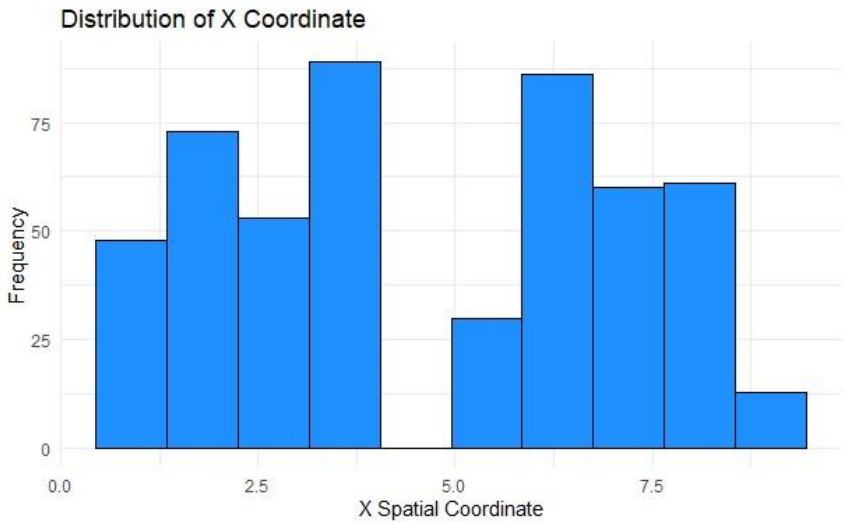
Despite the transition, a few significant fire episodes (more than 50 hectares) remain, which will be critical in determining the accuracy of the model in forecasting severe fire events.

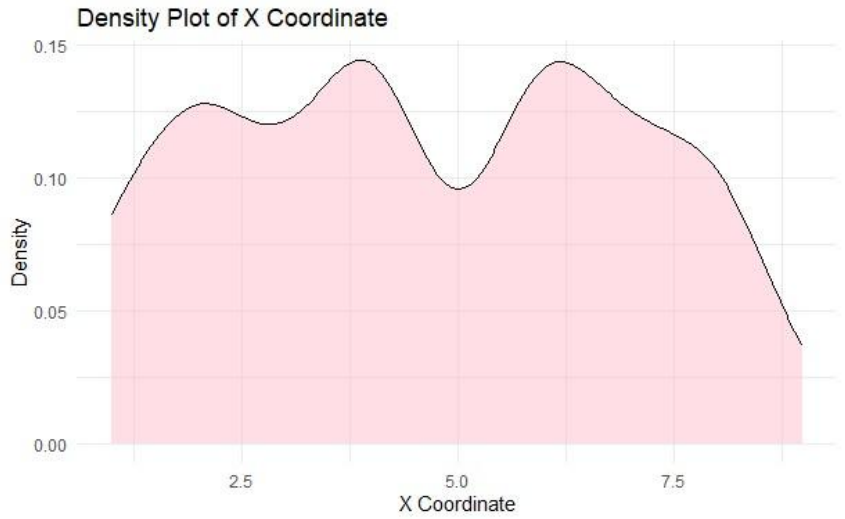
2.2 Exploratory Predictors

This part emphasises the distribution and significance of essential predictors, mostly numerical variables, for the forecasted burned area. Exploring the distribution of these variables is crucial for finding patterns, outliers, skewness, and possible adjustments to raise model performance.

2.2.1 Histogram and Density plots of X (Spatial Coordinate)

On a horizontal axis, the X coordinate shows the geographical location of the flames. This numerical variable is categorical and has the potential to provide light on the geographical patterns of fire incidents.





The histogram shows the frequency of fires at different X locations and gives a simple count of the number of times each section occurs. The fairly balanced distribution suggests that fires are not overly skewed towards one X value. This means that fires are spread out fairly evenly across this coordinate's area.

The density plot adds to this by showing areas with higher concentrations more clearly through a smoother graph. The highest points on the density plot show the X coordinate values that are most likely to cause fires. This helps find "hotspots," or places that might be a problem and need more attention when it comes to preventing fires and allocating resources.

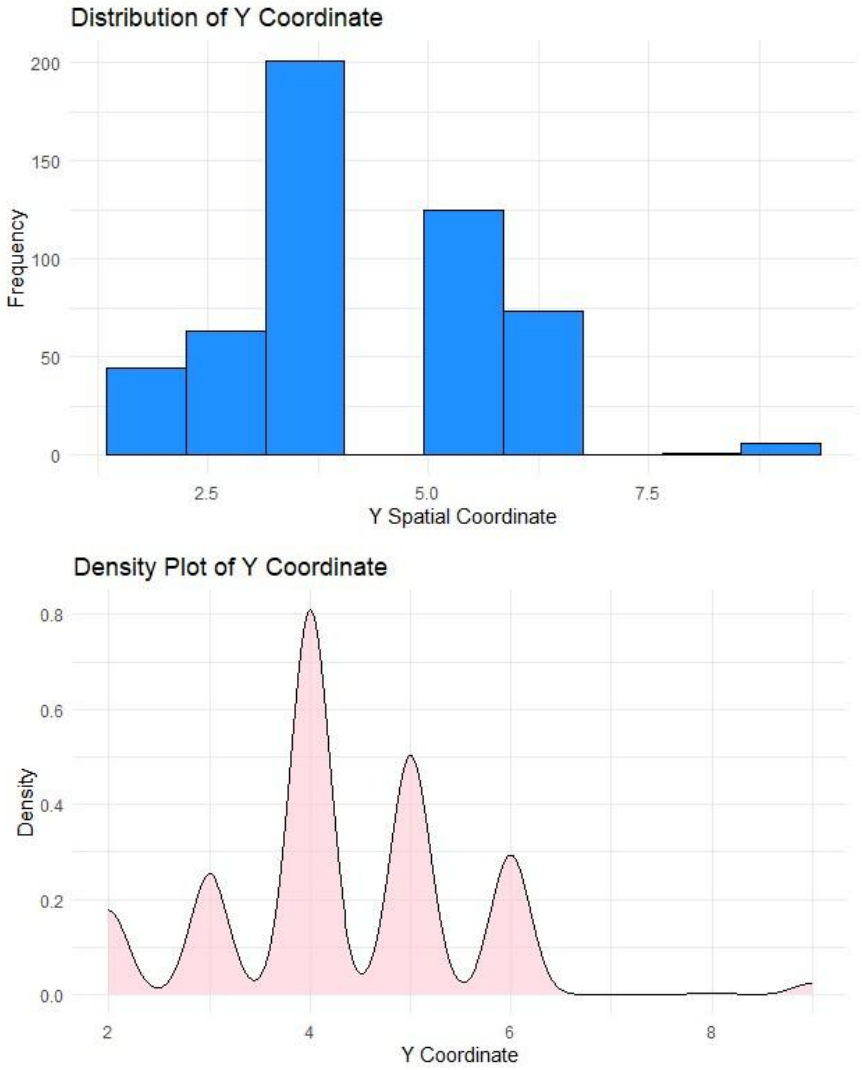
Statistical Interpretation

- Mean: 4.67
- Skewness: 0.03 (nearly normal)
- Kurtosis: -1.19 (flat distribution

The X-coordinate variable may not be as predictive by itself, given the roughly uniform distribution. It may be used with other geographical factors, such as Y. Because the distribution is approximately normal, no transformation is required to identify possible fire-prone locations.

2.2.2 Histogram and Density plots of Y (Spatial Coordinate)

The Y coordinate indicates the geographical position of flames on a vertical axis. It complements the X coordinate in terms of analysing geographical trends.



The histogram shows the frequency distribution, showing where fires happen more often. Peaks in the histogram help us determine which Y locations have had the most fires, which is helpful for planning strategies for focused tracking and fire control. This knowledge can help set priorities for places where fire safety and reaction are needed.

The density plot adds to this by better showing the underlying distribution pattern. The various points in the density plot show groups of fires, pointing out places where fires are more likely to happen. This visualisation is useful for finding patterns and areas that are close together. It helps people make smart choices about where to put resources, how to stop fires, and how to respond, which improves risk management in these important areas.

Statistical Interpretation

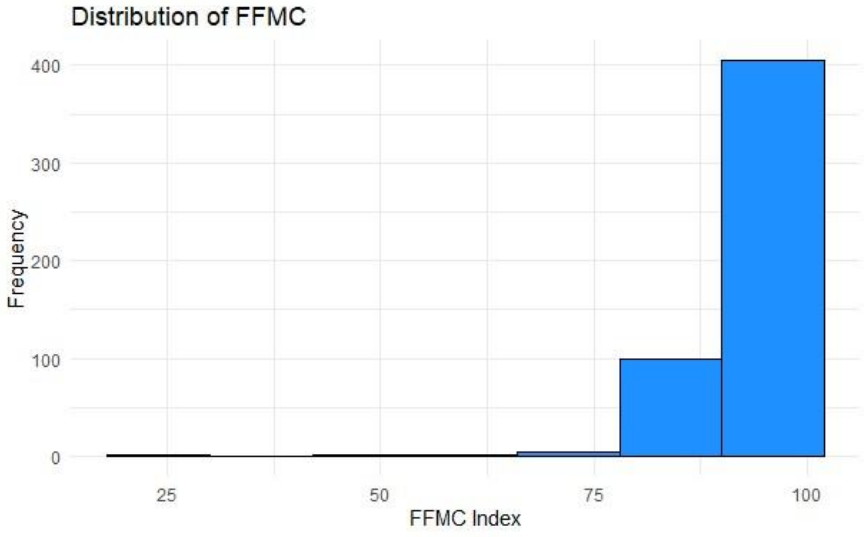
- Mean: 4.39

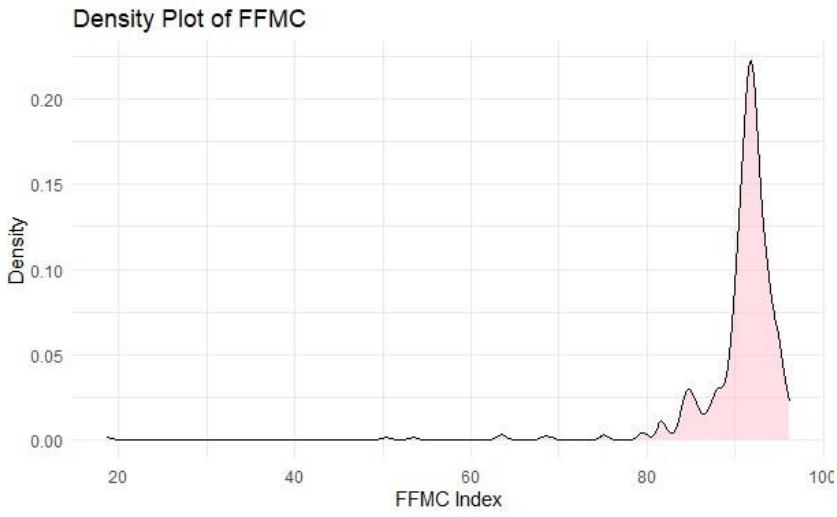
- Skewness: 0.41 (mild right skew)
- Kurtosis: 1.38 (slightly peaked)

Unlike the X coordinate, the Y coordinate has a slight skew, indicating that particular vertical portions of the study area may be more susceptible to flames. Although no modification is required, integrating this variable with X in a heatmap might assist in revealing spatial patterns.

2.2.3 Histogram and Density plots of Fine Fuel Moisture Code (FFMC)

The FFMC measures the moisture content of refined fuels, such as leaves and twigs. Higher numbers indicate dry fuels, which raise fire danger, whereas lower values suggest wetter conditions.





The FFMC histogram shows an intense concentration towards higher values, especially between 90 and 96. Most FFMC readings are high, so the conditions are dry and suitable for starting and spreading fires. Not many readings below 80 on the left side of the graph show that most records show dry conditions instead of wet ones.

The density plot makes the distribution easier to see by drawing attention to the peak between 90 and 96. It is clear from the density graph that the FFMC numbers are not spread out equally, with

many being very high. Because FFMC is a crucial indicator of flammable surface fuels, these pictures help us understand the conditions that lead to fires.

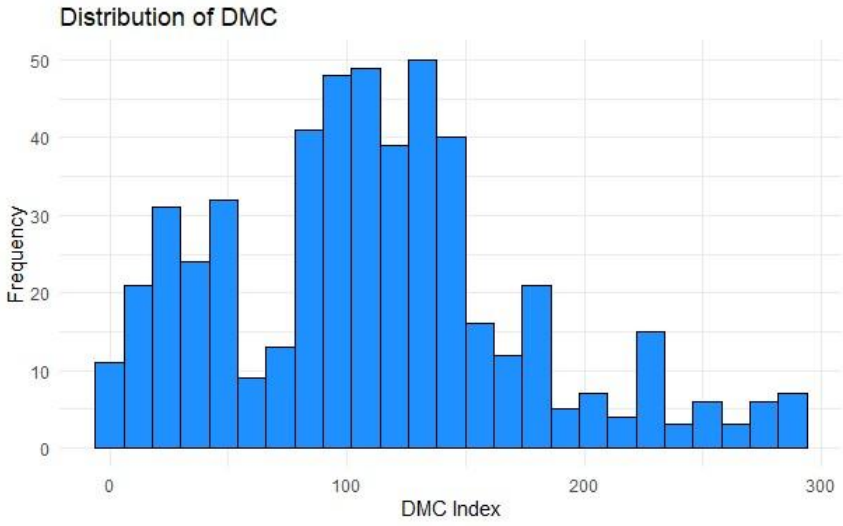
Statistical Interpretation:

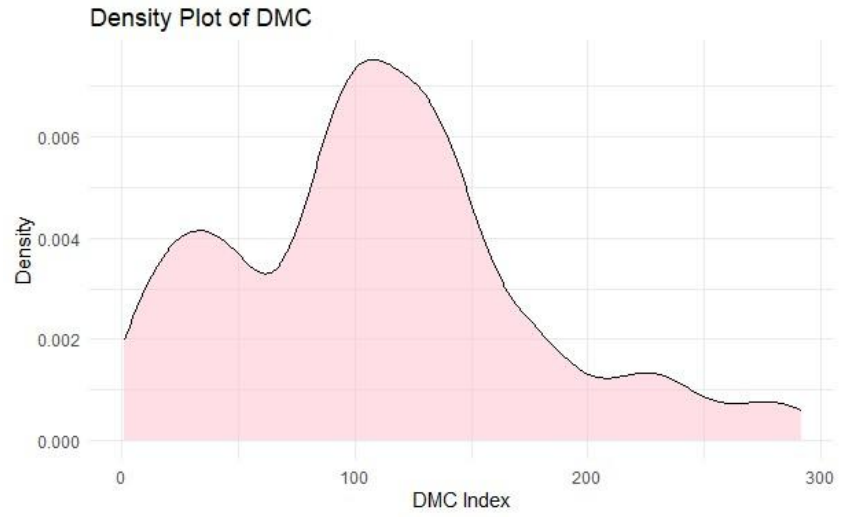
- Mean: 90.64
- Skewness: -6.51 (extremely left-skewed)
- Kurtosis: 65.60 (significantly peaked)

The significant kurtosis and skewness of FFMC expose that most fire-prone scenarios arise in relatively dry environments. Although the significant skewness of this variable makes predicting fire initiation difficult, it is nonetheless important and calls for further research.

2.2.4 Histogram and Density plots of Duff Moisture Code (DMC)

The DMC indicates fire hazards and gauges the moisture content of deeper organic layers. Higher readings indicate drier weather, which raises the possibility of flames that burn deeper.





The histogram and density plots for the DMC Index show how this variable is spread out in the dataset and how it changes over time. The histogram shows a fairly bimodal distribution with groups around

values close to 100, which helps you figure out how often different DMC values happen. On the other hand, the density plot gives a more refined view of the same data and a better idea of how the data is distributed overall.

These charts are necessary because they help us figure out how the DMC, which counts the amount of water in things breaking down on the forest floor, changes over time and its primary trend. High numbers mean that the forest floor is dry, which makes it easier for fires to start. By showing this data in a picture, we can better understand the natural factors that raise the risk of forest fires, which helps the model make predictions and explain data.

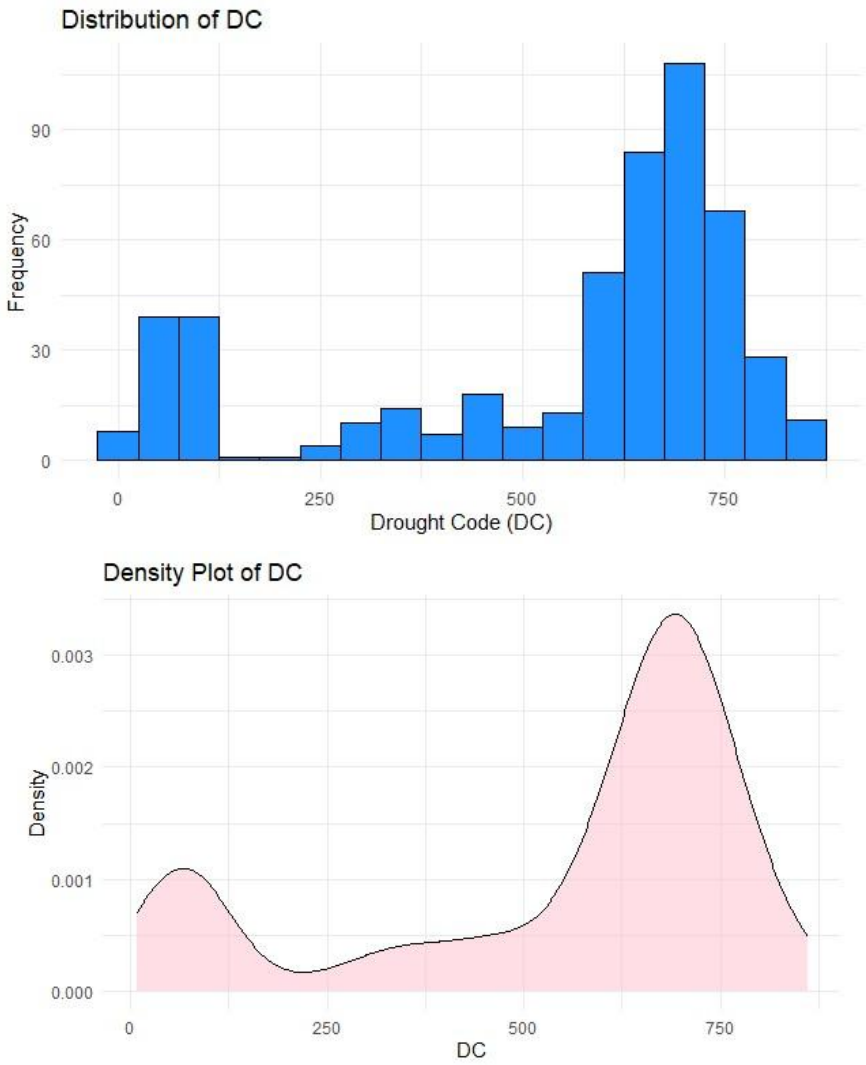
Statistical Interpretation:

- Mean: 110.89
- Skewness: 0.54 (moderate right skew)
- Kurtosis: 0.16 (normal distribution)

DMC's modest skewness points to dry circumstances ruling the dataset. The low kurtosis, therefore, suggests that extreme values are rare. No transformation is immediately required. The DMC distribution reveals a slight right skew, meaning most fire occurrences happen under dry organic layer circumstances.

2.2.5 Histogram and Density plots of Drought Code (DC)

To evaluate the fire danger over time, it is essential to quantify the impact of long-term drying conditions using the Drought Code.



The graph shows the frequency of different DC values. There is a clear peak near the upper end of the range (above 700). This means that dry conditions, shown by the DC, were common and severe across the sample. The graph shows how many observations are concentrated in different amounts of drought. This makes it easy to see where most of the data points are, which can help us figure out how drought affects the fire risk.

In addition, the density map shows the DC index's highs and lows as well as the values' smooth distribution. This plot does a better job of showing the most usual numbers than the histogram. Understanding these trends is important to determining how changes in the severity of drought across the dataset might affect the number of fires. This will help find high-risk areas that may need specific safety measures.

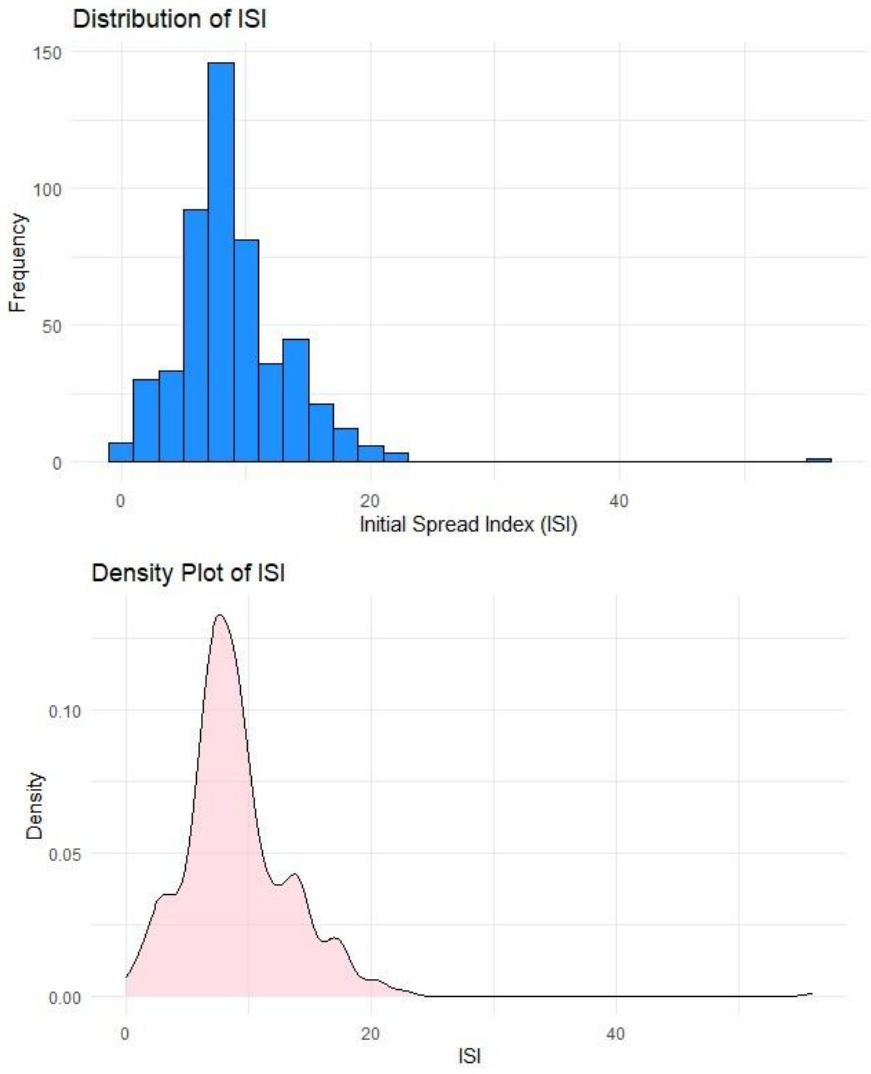
Statistical Interpretation:

- Mean: 546.89
- Skewness: -1.10 (moderately left-skewed)
- Kurtosis: -0.23 (flat distribution)

DC is significantly left-skewed, indicating that most fires occur during protracted dry conditions. Given the nature of the drought code, no change is required. It is likely to have an essential role in forecasting more extensive and protracted flames, particularly during prolonged dry spells.

2.2.6 Histogram and Density plot of Initial Spread Index (ISI)

The Initial Spread Index is a composite metric used to predict the pace of fire spread, taking into account both wind speed and the moisture content of particulate fuel.



To understand how forest fires tend to spread, you need to look at the histogram and density plot for the Initial Spread Index (ISI). The graph shows how often different ISI values are seen, which tells us how often specific amounts of beginning fire spread are seen. Forest management teams need this knowledge because it helps them figure out how often they might have to deal with diseases that spread quickly versus situations where things are easier to control.

The density plot adds to the histogram by showing the data distribution more smoothly. This lets us see the underlying trends and highest rates. These visualisations help us understand how fires usually behave and what situations are more likely to be dangerous. This helps us make smarter decisions about how to use resources and stop fires.

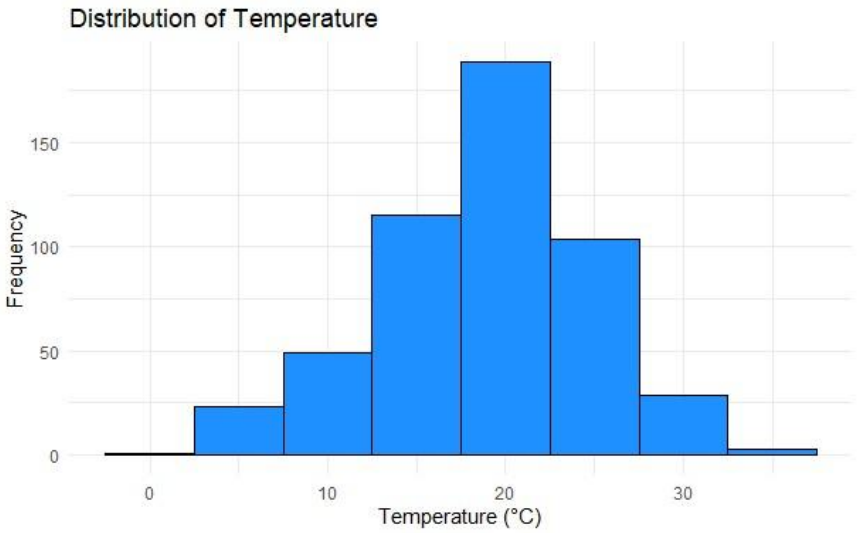
Statistical Interpretation:

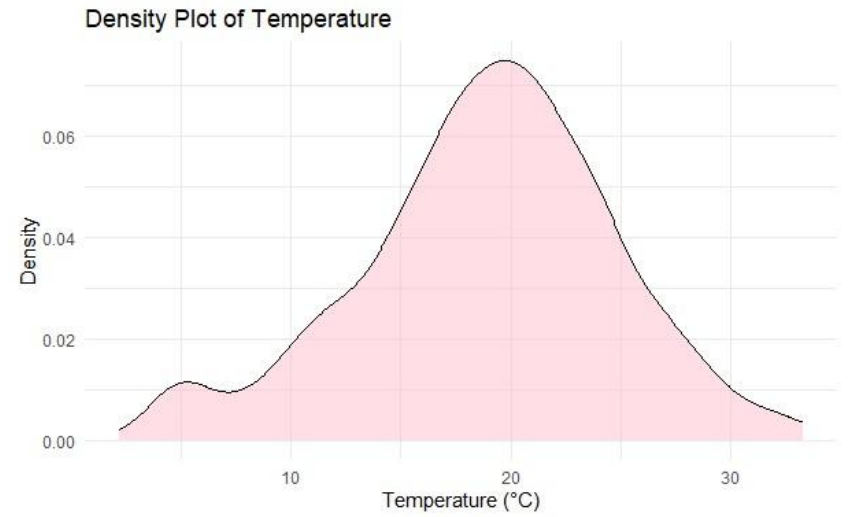
- Mean: 9.0
- Skewness: 2.51 (highly right-skewed)
- Kurtosis: 20.98 (highly peaked)

ISI is highly skewed to the right. Hence, most flames start out at a slow pace and only occasionally show a fast spread of fire. This skewness implies that before adding information to the prediction model, transformation (logarithmic) is probably needed. The significantly peaked kurtosis indicates a concentration of values around the mean, meaning that very few occurrences had extreme beginning spread conditions.

2.2.7 Histogram and Density Plots of Temperature

Temperature is a very important indicator of fire strength because higher temperatures make fires more likely to start and spread. Analysing temperature distributions helps us understand their function in fire dynamics.





The histogram displays how often each temperature event occurs, showing that temperatures are most often around 20°C, with only a few cases being very high or very low. This helps determine whether the temperature usually stays in a certain range, which is important for learning about the factors that can cause forest fires.

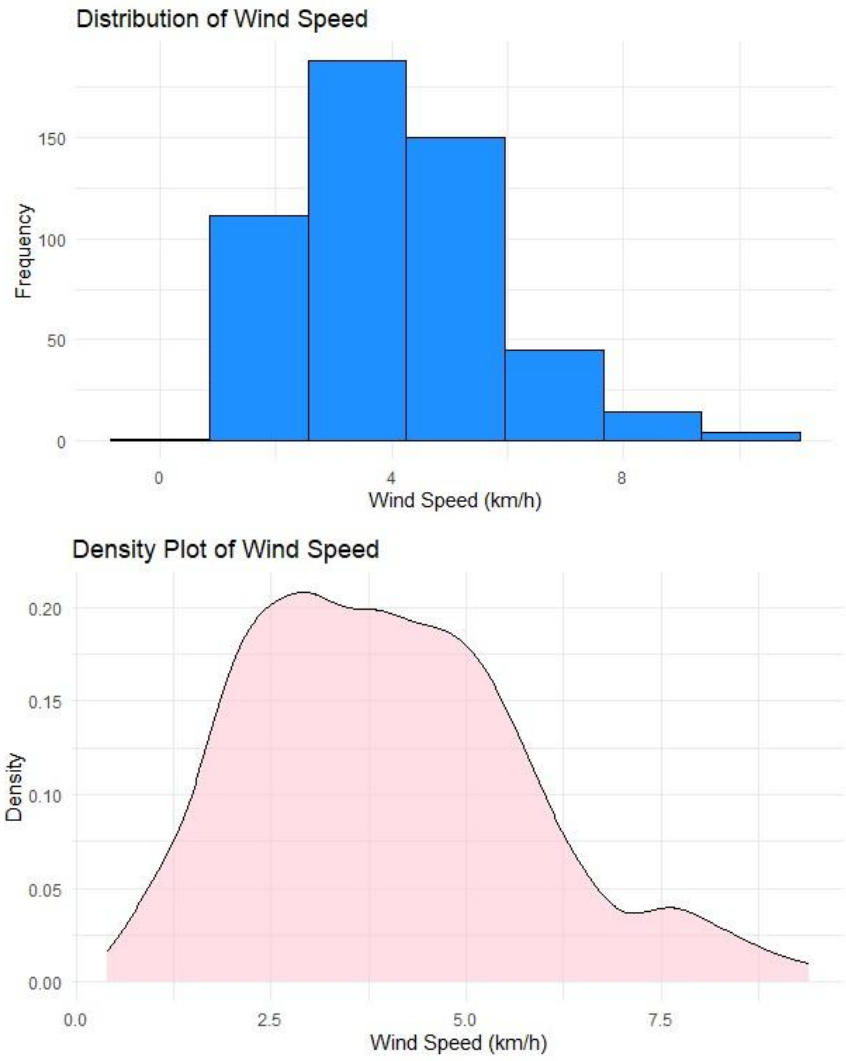
The density plot adds to this study by showing the chance distribution of temperatures on a smooth slope. It lets us see the general shape of the temperature distribution, which looks like a normal distribution with a small negative skew in this case. These pictures are useful for finding any changes or trends in temperature that might impact how a fire behaves and how dangerous it is expected to be.

Statistical Interpretation:

- Mean: 18.89°C
- Skewness: -0.33 (slightly left-skewed)
- Kurtosis: 0.09 (normal distribution)

The temperature displays an almost symmetric distribution with no extreme tails, with the skewness at -0.33 and kurtosis at 0.09. No transformation is required.

2.2.8 Histogram and Density Plots of Wind Speed



The density plot and wind speed histogram show how the wind is spread across the dataset. The graph shows that most wind speeds are between 2.5 and 5 km/h, meaning mild wind is typical during forest fires. The density plot, which shows a one-modal distribution with a high density of around 4 km/h, backs up this finding even more.

These charts help us understand how the wind blows during fires, which is important because the wind dramatically affects how the fire spreads. Looking at this variable, we can see patterns or trends that do not make sense. This helps us make predictions or figure out how fires behave in different wind situations.

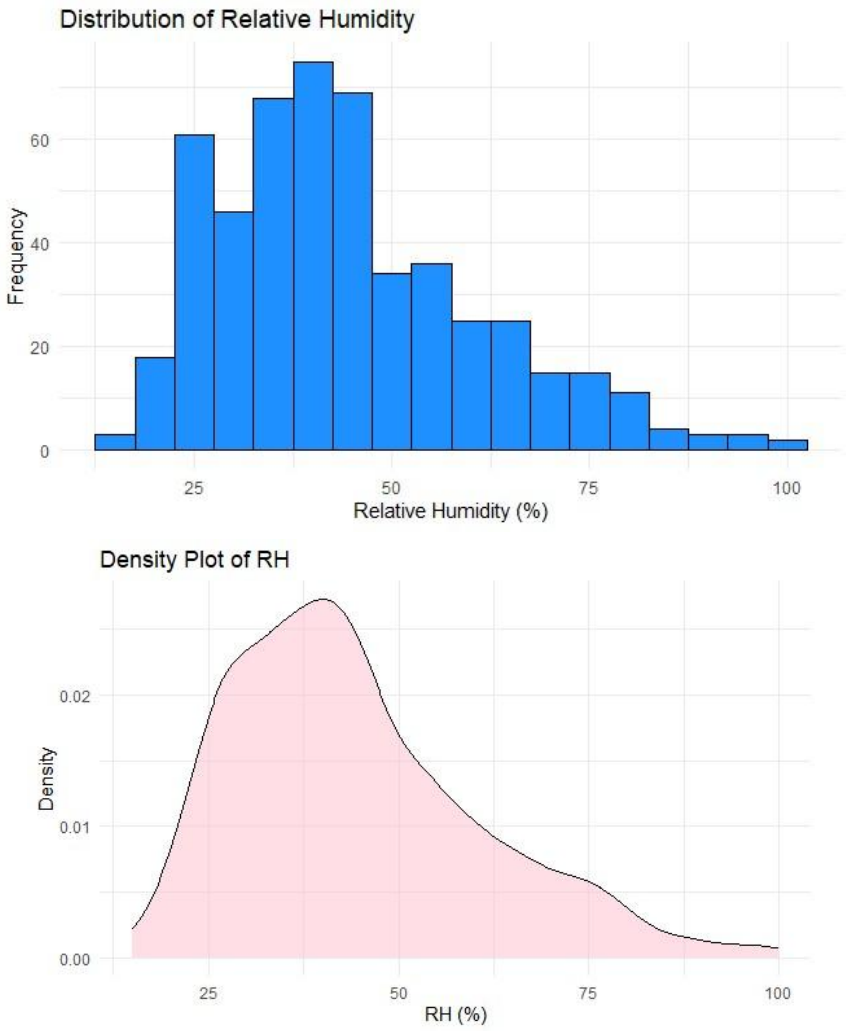
Statistical Interpretation:

- Mean: 4.01 km/h
- Skewness: 0.58 (moderate right skew)
- Kurtosis: 0.03 (normal distribution)

The distribution exhibits few outliers (high wind speeds) with a modest skewness of 0.58 and an almost normal kurtosis of 0.03. These anomalies could affect the propagation of fire. Although handling outliers might help, based on existing data, no change is required.

2.2.9 Histogram and Density plots of Relative Humidity (RH)

The moisture content of a plant is influenced by relative humidity, which in turn impacts how quickly it burns.



The histogram and density plots for Relative Humidity (RH) show how relative humidity is distributed and concentrated in the dataset. The graph shows how often different amounts of relative humidity occur, which helps you figure out what numbers happen most often. We see that most of the data are grouped around 25-50% humidity, which means that mild humidity is more common. This could affect fire conditions.

The density plot adds to the histogram by showing the overall direction of the RH data distribution through a smooth slope that draws attention to the fact that the data is skewed towards lower humidity levels. The fact that the peak is between 25 and 50 per cent says that relative humidity is usually modest during forest fires. This is important for understanding how humidity affects the start and spread of fires.

Statistical Interpretation:

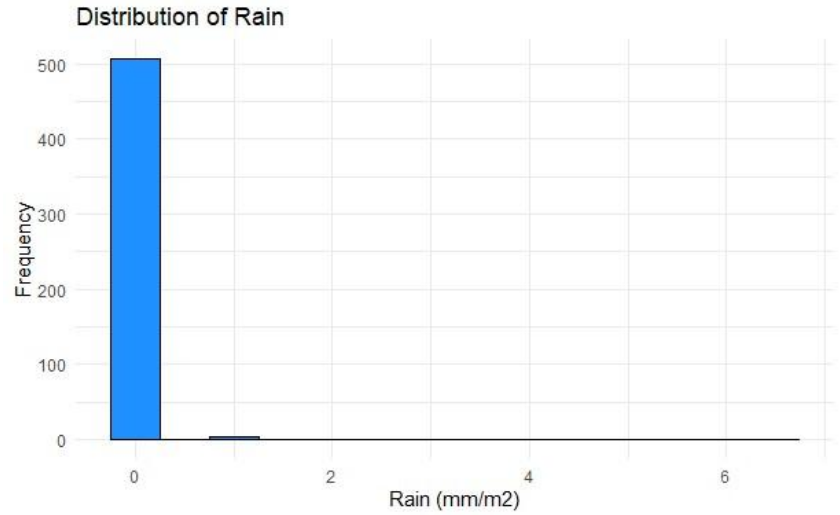
- Mean: 44%

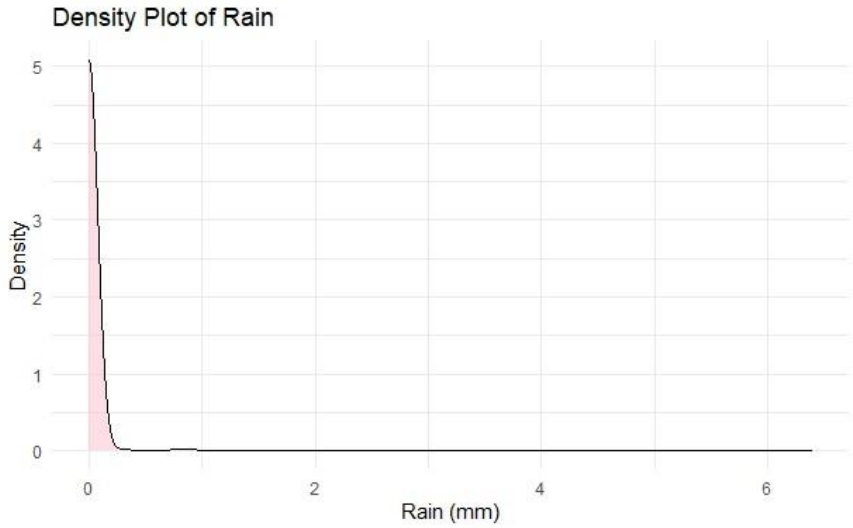
- Skewness: 0.85 (moderate right skew)
- Kurtosis: 0.38 (normal distribution)

Though severe low-humidity circumstances are unusual, relative humidity is skewed to the right, suggesting that fires often occur in dry settings. Given that increased humidity lowers fire probability, RH is predicted to have a modest inverse connection with the burnt area. The mild skewness is fine; transformation may not be required.

2.2.10 Histogram and Density Plots of Rain

Rainfall lessens the chance that a fire will start and spread. This variable aids in determining how wet circumstances affect fire behaviour.





The histogram and density plot make it easy to see how the "Rain" variable is spread out. The histogram shows that most data points have values close to 0 mm, meaning it did not rain much during the readings. The density plot, which shows a skewed distribution, supports this idea. It shows that rain was very uncommon and did not have much of an effect on most of the data points.

These graphics are very helpful for understanding why there isn't much rainfall data and figuring out how that might affect forest fires. Since rain does not happen very often, it might not affect the model

much. However, knowing how it is spread out can help you figure out its role in combination with other weather factors.

Statistical Interpretation:

- Mean: 0.02 mm
- Skewness: 19.62 (extremely right-skewed)
- Kurtosis: 412.34 (significantly peaked)

The severe skewness and kurtosis imply that most fire incidents occur under dry circumstances, with rare cases of rainfall during fire seasons. Unless combined with other weather-related variables, the dataset's low rainfall could make this variable less predictive. A logarithmic transformation might reduce the excessive skewness and enhance model performance. The right-skewed distribution suggests that most fire occurrences happen with little to no rain.

2.2.11 Summary Table

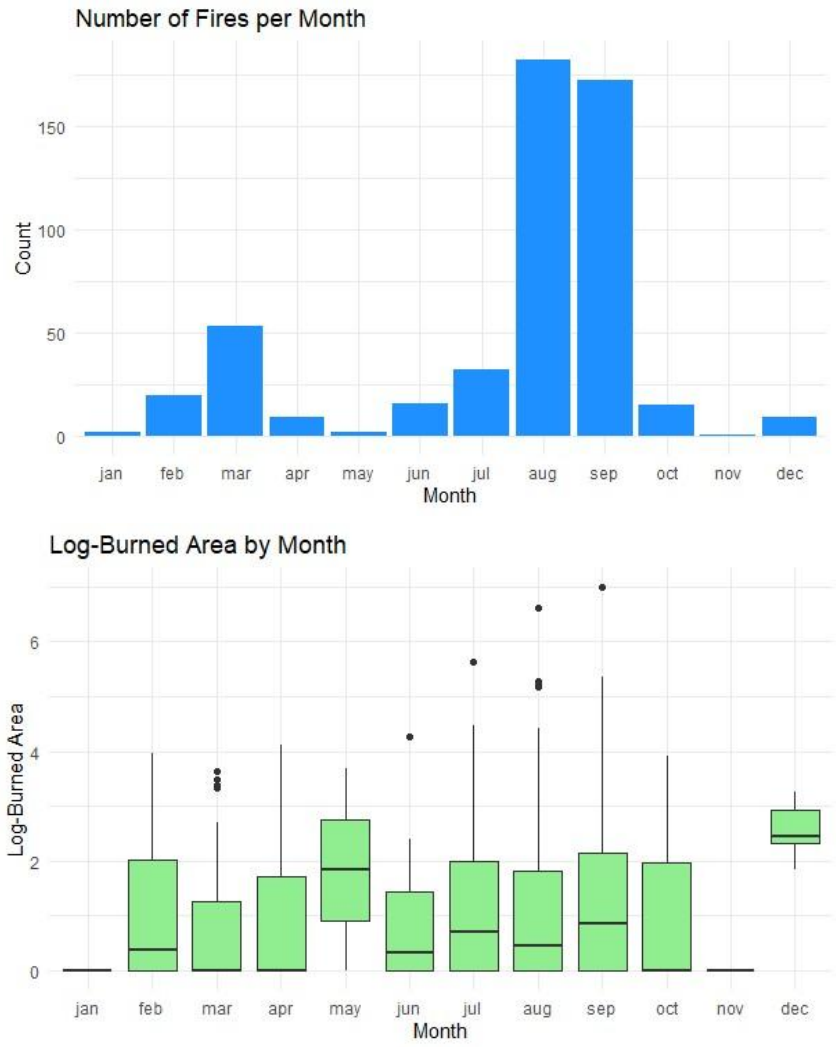
Predictors	Skewness	Kurtosis
X	0.03	-1.19
Y	0.41	1.38
FFMC	-6.51	65.60
DMC	0.54	0.16
DC	-1.10	-0.23
ISI	2.51	20.98
Temp	-0.33	0.09
Wind	0.58	0.03
RH	0.85	0.38
Rain	19.62	412.34

FFMC, **ISI**, and **Rain** may benefit from log transformations because of severe skewness and kurtosis. This will assist to mitigate the effects of outliers and stabilise model performance.

2.3 Categorical Variables Exploration

Critical new perspectives on the temporal distribution of forest fires come from categorical factors like month and day. Through the analysis of these factors, we may identify trends or patterns linked to seasonality, which, therefore, affect the frequency and severity of fires. Understanding possible fire danger periods made possible by this study helps forest management organisations allocate resources and prepare appropriately. **2.3.1 Bar and Box Plots for Month**

Seasonal weather fluctuations substantially impact the frequency and intensity of forest fires (Flannigan et al., 2009). Fire occurrences must be studied regularly to identify essential fire seasons and execute effective control methods (Westerling et al., 2006).

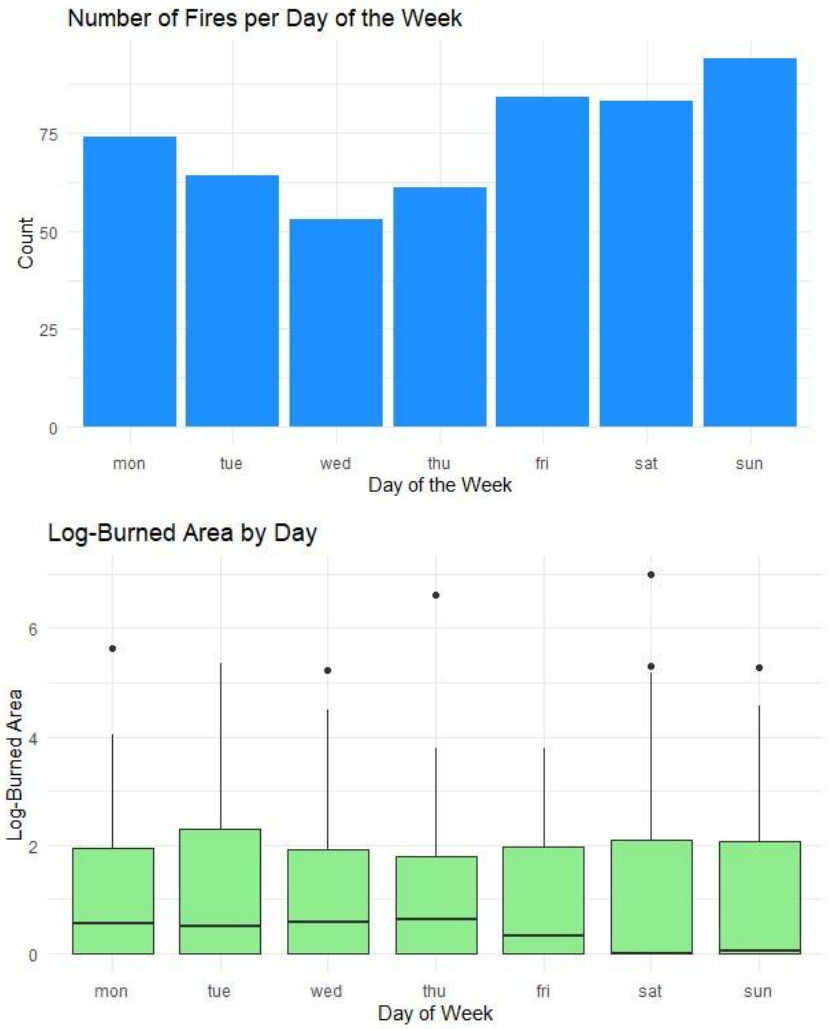


A bar plot of monthly fire frequencies shows temporal dispersion trends. While winter months show fewer fires due to increased rainfall and humidity, summer months usually see higher temperatures and drier conditions, which allow ignition and spread and hence influence fire activity (Flannigan et al., 2009).

A box plot of the log-transformed burned area by month depicts the yearly fire size and severity fluctuations. Significant increases in burned areas, especially in August and September, point to times when environmental circumstances support the spread of fire (Bradstock, 2010). Extreme fire incidents indicated by outliers could call for further research on contributing causes.

2.3.2 Bar and Box Plots for Day

The number of fires may vary depending on the day of the week, especially if human activity is involved in the ignition events. Examining fire incidents day by day yields useful information for fire control by pointing out patterns in human behaviour.

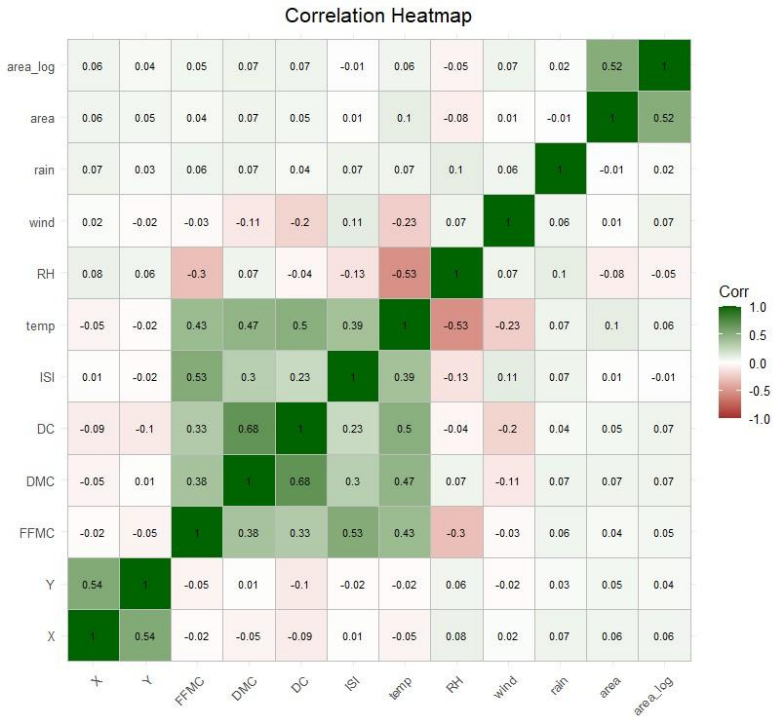


The bar plot clearly illustrates a surge in fire incidences on weekends (Saturday and Sunday), most likely owing to increased human activity. Fusco et al. (2016) have consistently correlated these activities with unintentional fire starts in arid environments. Given that human presence increases the danger of fire, understanding this pattern might inform targeted fire prevention initiatives and enhanced weekend patrols (Johnston et al., 2020).

The box plot shows that more significant burned areas occur more often on weekends, implying a causal relationship between human activity and more catastrophic fires. Particularly when human ignitions take place, delayed reactions and easily accessible fuel help flames to spread (Keeley & Syphard, 2018).

2.4 Correlation Analysis

A correlation heat map visually shows the correlations among many numerical variables in the dataset. Emphasising the strength and direction of the linear correlations between variables helps one rapidly identify important predictors and multicollinearity.



Correlation with Log-Transformed Area

Weak relationships between the log-transformed burned area and most meteorological predictors. With a slight positive correlation (0.06), temperature hardly affects fire size, consistent with actual findings. A weak positive correlation (0.07) for wind speed indicates a limited impact on bigger flames. With a very small association (0.05), the Fine Fuel Moisture Code (FFMC) suggests that drier fine fuels marginally influence fire size. Analogously, the modest positive correlation (0.07) of Duff Moisture Code (DMC) indicates that reduced duff moisture only somewhat increases fire size. Overall, these predictions, taken together, are not very powerful influences.

Multicollinearity between Predictors

Several predictor variables have moderate to high correlations, suggesting possible multicollinearity. DC and DMC have a high positive correlation (0.68), indicating that they assess similar moisture conditions. This multicollinearity may influence regression models and must be addressed by regularisation or dimensionality reduction (Dormann et al., 2013). DC and temperature (0.50) and DMC and temperature (0.47) have modest associations, indicating that higher temperatures lead to drier conditions. FFMC and DMC have a lesser connection (0.33), suggesting they measure distinct characteristics of fuel moisture. These connections demonstrate the interdependency of fire-related factors.

Negative Correlations

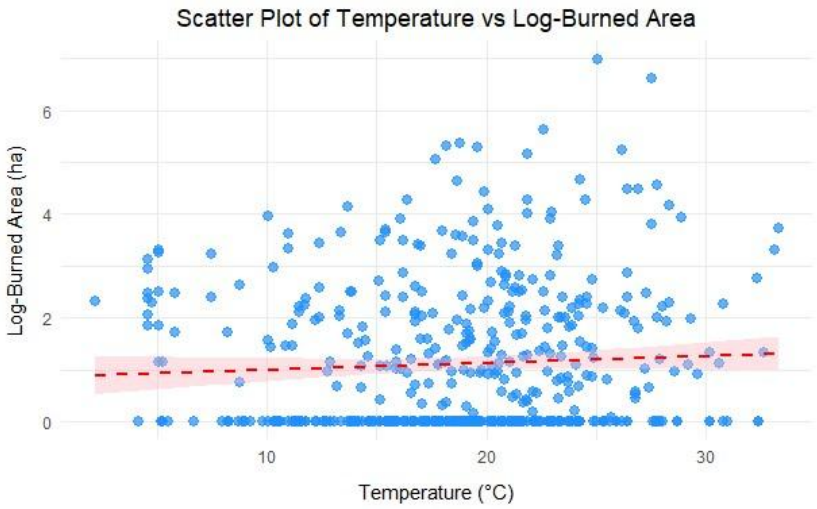
A small negative relationship (-0.05) exists between relative humidity (RH) and the log-transformed burnt area. This means that higher humidity makes the fire smaller. This fits with what Jolly et al. (2015) found: higher humidity stops fires from spreading by keeping wetness in fuels, which lowers the chance of starting and the strength of the fire. Also, the Initial Spread Index (ISI) has a slight

negative correlation (-0.01), which means that it doesn't have a significant effect on the size of the fires in this set of data.

2.5 Scatter Plots (Lower Triangle)

Temperature (temp), wind speed (wind), the Fine Fuel Moisture Code (FFMC), and the Duff Moisture Code (DMC) have been chosen as important indicators. Each of these factors is important for predictive models because they significantly affect how fires behave and how much land is burned.

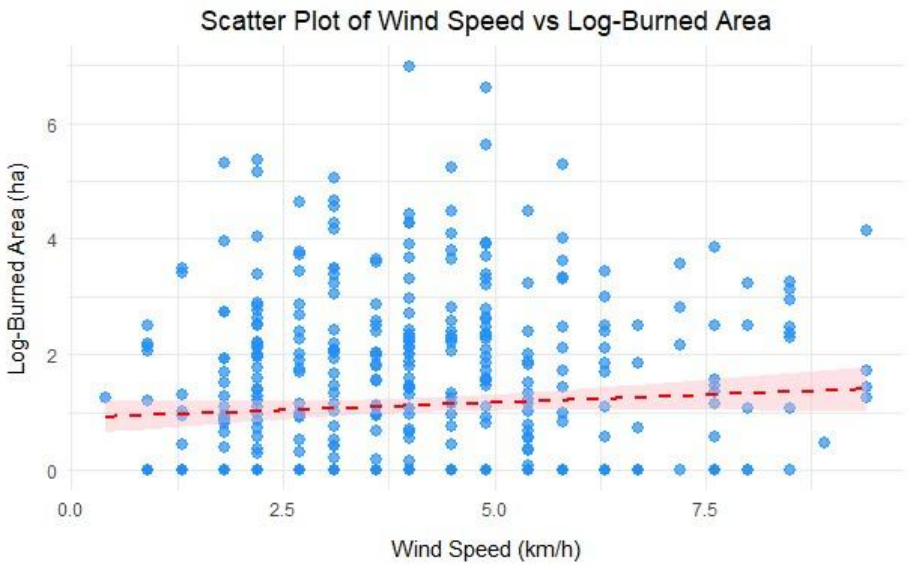
2.5.1 Temperature vs. Log-Burned Area



Positive Relationship: The red dashed trend line on the scatter plot shows a positive linear trend between temperature and the log-transformed burnt area. This implies that greater burnt areas are often linked to higher temperatures. According to the trend line's positive slope, more giant flames will likely occur as temperatures rise.

Linear Trend with Confidence: The confidence interval, highlighted in light pink around the trend line, shows the level of uncertainty in the connection between temperature and the log-burned area. The relatively narrow band supports the trend's dependability, indicating that the linear model fits the data in this range rather well.

No Clear Non-Linear Pattern: There is no evidence of a non-linear relationship between temperature and log-burned area; the linear trend line fits very well. Though further research using non-linear models might confirm this assumption, this implies that linear models may be sufficient in capturing the influence of temperature on fire size. **2.5.2 Wind Speed and Log-Burned Area**

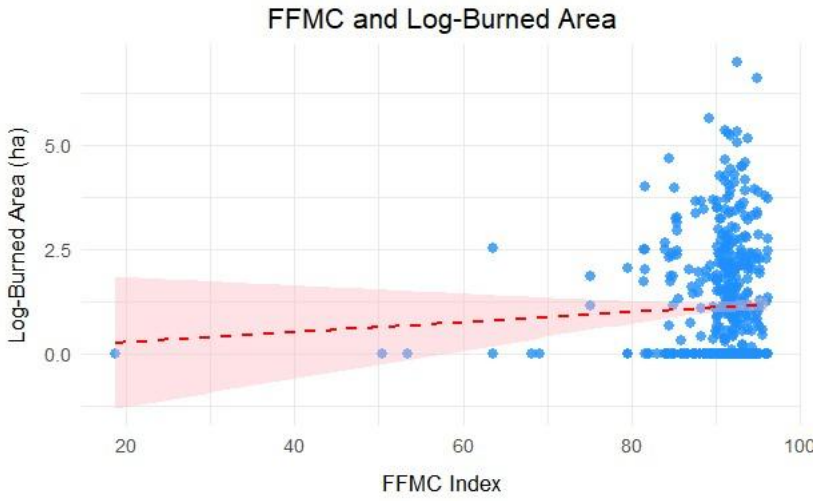


Positive Linear Relationship: As shown by the red dotted trend line, there is a positive linear relationship between wind speed and log-transformed burned area. The burning area starts to get bigger as the wind speed goes up. This makes sense because stronger winds tend to feed fires by spreading embers and providing oxygen, which can speed up the spread of the fire and affect more land.

Moderate Confidence in Prediction: The trend line's confidence interval, which is coloured in light pink, is quite small, indicating that the correlation between wind speed and burnt area is generally constant between data points. This suggests that, while there is still considerable fluctuation, particularly at higher wind speeds, wind speed is a good predictor of the amount of burnt regions.

No Strong Non-linear Trends: The scatter plot does not clearly show non-linear patterns, and the linear trend line provides a reasonable fit to the data. This implies that a linear model might capture much of the link between wind speed and fire size. Nevertheless, more research using non-linear models could still be helpful, especially at very high wind speeds when the fire's behaviour may vary.

2.5.3 FFMC vs. Log-Burned Area



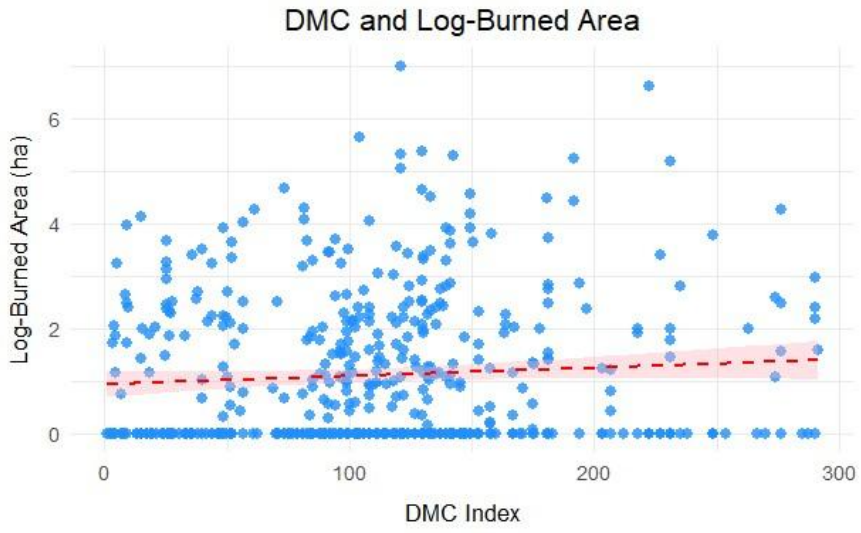
Negative Linear Relationship: FFMC (Fine Fuel Moisture Code) and the log-transformed burned area exhibit a negative linear connection, as seen by the red dotted trend line. This implies that the logburned area falls as FFMC rises. Drier fine fuels indicated by a higher FFMC score correspond with bigger flames generally. This negative connection fits the idea that drier fuels produce more burned areas.

Consistent Predictive Power: The very small light pink confidence interval around the trend line indicates that the link between FFMC and the log-burned area is constant and dependable across the dataset. The linear trend line matches well, demonstrating that FFMC effectively predicts fire size, particularly when modelling bigger fire occurrences**.**

Absence of Non-linear Behaviour: Based on the solid, smooth trend line, the two variables seem to have a linear relationship. The lack of notable deviations or non-linear behaviour suggests that a linear model might represent the connection between FFMC and the log-burned area.

Variability at Lower FFMC Values: Although greater burned areas are more often seen at higher FFMC values, the figure also reveals considerable fluctuation at lower FFMC values, where burned areas vary from small to large.

2.5.4 DMC vs. Log-Burned Area



Positive Linear Relationship: The dashed red trend line shows a positive linear relationship between DMC and the log-transformed burned area. Thus, the log-burned area usually increases as DMC rises. This implies that bigger flames are associated with drier conditions in the duff layer, that is, with greater DMC.

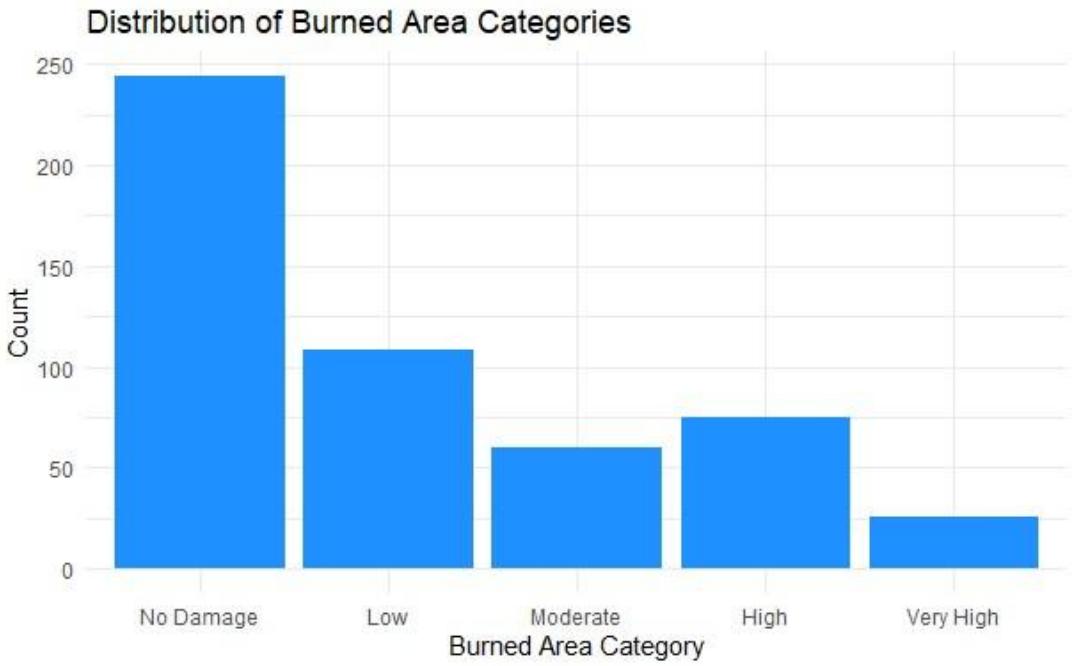
Moderate Confidence in Trend: The light pink confidence interval around the trend line is quite tight over the majority of the DMC value range, demonstrating that the relationship between DMC and the log-burned area is stable and dependable. This implies that DMC is a reliable predictor of burned area size, with considerable fluctuation at higher DMC levels.

Linear Fit is Appropriate: There is no clear indication of non-linear behaviour; the smooth trend line matches the data really well. This implies that linear approaches allow one to efficiently describe the

link between DMC and log-burned area. Still, further research on non-linear effects at very high DMC values would be advisable, particularly for a more accurate depiction of major flames.

2.6 Distribution of Burned Area Categories

Looking at the burning area types as a bar plot shows how often each severity class happens, from "No Damage" to "Very High". This picture is significant for understanding the imbalance in information in the class. As the chart shows, the mismatch means that most of the findings are in the "No Damage" group, while the number of incidents in the higher intensity levels, like "High" and "Very High", is much lower. This is essential information for understanding the data structure because the uneven spread of classes can directly affect training and evaluating the model.



A big problem in machine learning is class misbalance, especially when the job is to classify things. Models often lean towards the class with the most members ("No Damage"). This can make it hard to guess the lower classes, like "High" or "Very High." Regarding risk management, these minority groups are often the most important because they appear as high-risk fires that need immediate attention (Buda et al., 2018). The model can better predict high-risk fire groups after class imbalance is fixed. This makes it easier to save people and less environmentally and economically damaged by forest fires.

III. Data Preparation

3.1 Data Cleaning

3.1.1 Handling Missing Values

The below analysis shows that there are no missing values in any of the dataset's columns.

```
> print(missing_values)
   X     Y month   day  FFMC   DMC    DC   ISI  temp    RH  wind  rain  area
   0     0     0     0     0     0     0     0     0     0     0     0     0
```

3.1.2 Handling Duplicate Records

Four duplicate rows were found, and they correspond to the same values for every feature including the burned area.

```
> # View the duplicates to evaluate them
> head(duplicates)
    X Y month day FFMC   DMC    DC  ISI temp RH wind rain  area
54  4 3   aug wed 92.1 111.2 654.1  9.6 20.4 42  4.9    0  0.00
101 3 4   aug sun 91.4 142.4 601.4 10.6 19.8 39  5.4    0  0.00
216 4 4   mar sat 91.7  35.8  80.8  7.8 17.0 27  4.9    0 28.66
304 3 6   jun fri 91.1  94.1 232.1  7.1 19.2 38  4.5    0  0.00
>
```

The above duplicaes must be deleted. Because retaining duplicates may cause model overfit, in which case repeated entries disproportionately affect the model, therefore lowering its generalizability (Han et al., 2011). Eliminating duplicates also guarantees equal weighting of data and increases computing efficiency, hence strengthening the model (Kotsiantis et al., 2006). Furthermore, many duplicates have a burned area of zero, adding no meaningful information.

The total number of rows now has changed to 513 after deleting those duplicates.

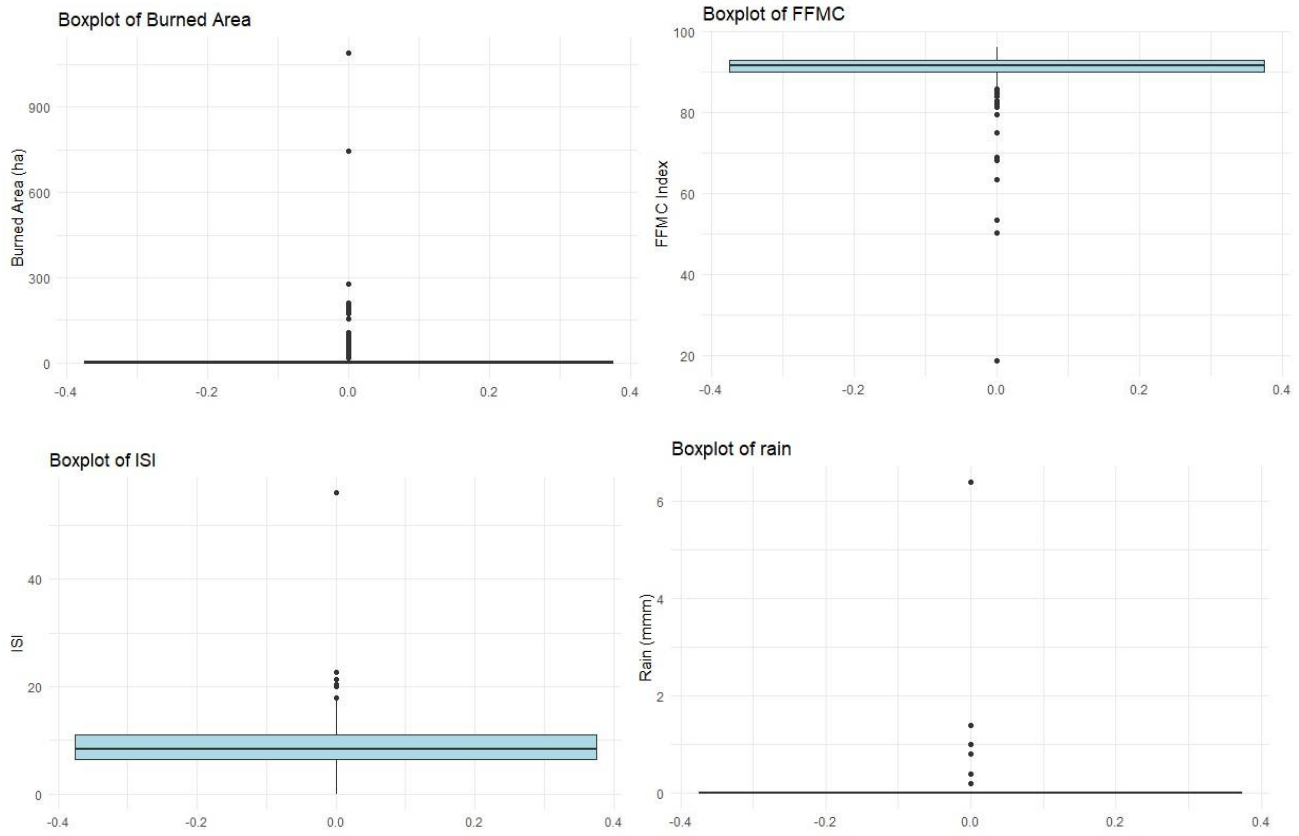
```
> print(paste("Number of rows after removing duplicates:", nrow(forestfires_clean)))
[1] "Number of rows after removing duplicates: 513"
```

3.1.3 Outlier Detection

Outliers may considerably impact prediction model accuracy and dependability, mainly when they disproportionately influence statistical parameters like mean and standard deviation. Finding and removing outliers is essential for ensuring that models work well because too high or too low numbers can change the results of regression and machine learning methods (Aggarwal, 2017). This study carefully found and dealt with outliers for essential factors like Burned Area, FFMC, ISI, and Rain.

Boxplot Analysis

Boxplots give a visual depiction of the data distribution that highlights possible outliers. Data points beyond the “whiskers”, which usually indicate 1.5 times the Interquartile Range (IQR) above the third quartile or below the first quartile, are referred to as outliers in a boxplot.



- Burned Area: The most excellent burned area exceeds 1000 hectares, and the raw burned Area boxplot exposes multiple extreme outliers. While outliers go well beyond the top range, most of the observations fall below 200 hectares.
- FFMC: The FFMC boxplot also showed numerous lower outliers, most likely resulting from relatively uncommon wet circumstances throughout the fire seasons.
- ISI and Rain: Both ISI and Rain have significantly right-skewed distributions with multiple high outliers, as verified visually by boxplots.

Interquartile Range (IQR) Method

The IQR method was used programmatically to identify outliers for every variable exactly. To determine the IQR for each predictor, the Q1 (25th percentile) and Q3 (75th percentile) were computed:

- Lower Bound = Q1 - 1.5 x IQR
- Upper Bound = Q3 + 1.5 x IQR

Any value that fell outside of these boundaries was labelled as an outlier.

The following summarises the number of outliers along with a list of those outliers discovered in each crucial variable:

- Number of outliers detected in ‘area’: 63
- Number of outliers detected in ‘FFMC’: 53
- Number of outliers detected in ‘ISI’: 23
- Number of outliers detected in ‘rain’: 8

3.2 Transformation

3.2.1 Log Transformation

To successfully handle outliers and stabilise the data distribution, 'area', 'FFMC', 'ISI' and 'rain' were transformed using logarithms. This change narrows the range of high values, lowers skewness, and lessens the effect of prominent peaks, which makes the data better for models. Log transformation is a well-known way to fix skewed data, especially when working with factors that have significant differences in their magnitudes (Osborne, 2010). This method lessens the impact of outliers, creating a more normal distribution that makes regression models and other statistical studies work better.

- **Area, ISI:** The log transformation worked for area and ISI, significantly decreasing skewness and enhancing data symmetry.
- **FFMC:** A reflection and log transformation were used to address left-skewed. The distribution became right-skewed due to this reflection, making it appropriate for log transformation. The table below shows the skewness values of each predictor before and after applying log transformation:

Predictors	Skewness (before)	Skewness (after Log-transformed)
area	12.73	1.21
FFMC	-6.51	0.17
ISI	2.51	-0.93
rain	19.62	14.07

Area: Burned Area's log transformation effectively reduced the excessive skewness and brought the distribution closer to normal.

FFMC: Skewness of FFMC was decreased to 0.17 by the transformation, suggesting near-normality, which is advantageous for most models.

ISI: The log transformation of ISI effectively decreased skewness from 2.52 to -0.93, considerably improving the distribution's symmetry.

Rain: The Rain variable offered a special difficulty as strong right-skewness (skewness of 14.07 after log transformation) dominated zero values.

3.2.2 Encoding Categorical Variables

Day and month are nominal categorical variables in the dataset that lack a significant numerical order. Treating them as ordinal (e.g., December as "greater" than January) would be erroneous. To prevent this, each category was transformed into binary columns (such as month_jan, day_mon), where the existence of each category is indicated by 0 or 1. This process is known as one-hot encoding.

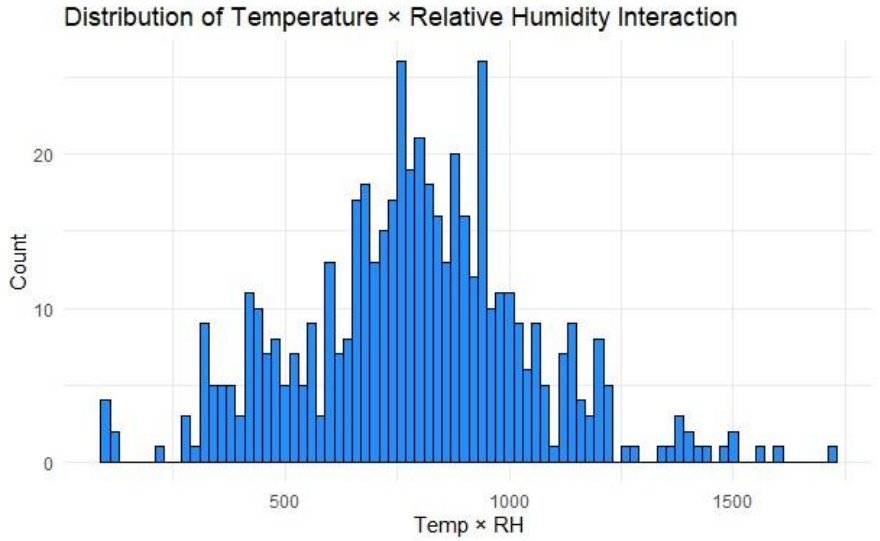
This strategy is critical for models that use distance calculations, such as Support Vector Machines

(SVM), k-Nearest Neighbors (kNN), Artificial Neural Networks (ANN), and tree-based models. Onehot encoding prevents these models from misinterpreting categorical variables as numerical, reducing bias and enhancing model performance (Pedregosa et al., 2011).

3.2.3 Feature Engineering

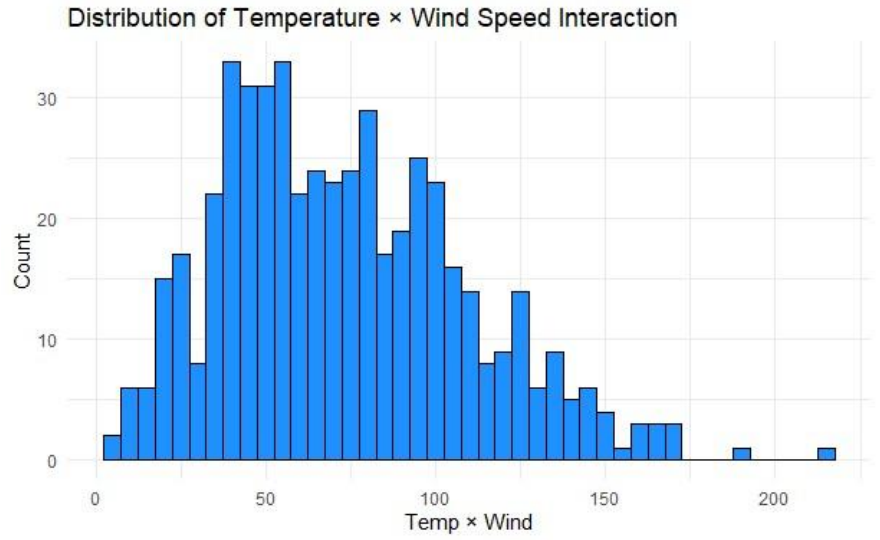
Temperature and Relative Humidity Interaction

This depicts the inverse connection in which high temperatures diminish fuel moisture, increasing fire danger, but high humidity lessens it. Division by zero is avoided by adding one to relative humidity. The histogram distribution is right-skewed, suggesting that high-risk circumstances are uncommon yet critical.



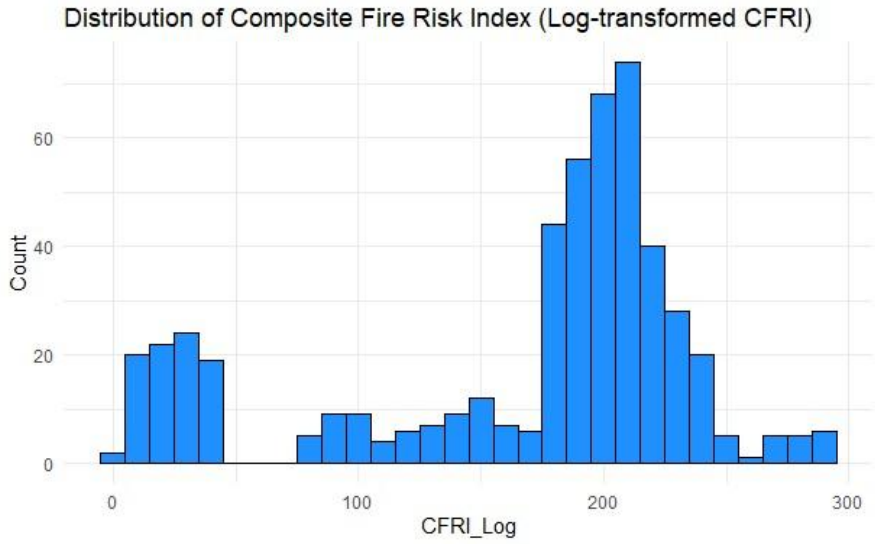
Temperature and Wind Speed Interaction

Wind accelerates fire propagation by providing oxygen and transporting embers. This interaction phrase describes conditions including high temperatures and strong winds. The histogram's rightskewed distribution indicates key times of high fire danger.



Composite Fire Risk Index (CFRI)

Combines standardised indicators (FFMC, DMC, DC, and ISI) to provide a complete fire risk measure. The log-transformed CFRI balances the effect of extreme values, resulting in a more regularly distributed risk representation and improved model prediction.



Data Dictionary of Feature Engineering

Variable Name	Data Type	Description	Reason for Inclusion
Temp_RH_Interaction	Numeric	The product of temperature and relative humidity.	Captures the combined effect of temperature and humidity on fire danger, focusing on dry, hot circumstances.
Temp_Wind_Interaction	Numeric	Product of temperature and wind speed.	Describes the interplay impact of temperature and wind on fire spread velocity.
CFRI	Numeric	The Composite Fire Risk Index combines many fire indexes.	Combines many indicators to deliver a comprehensive fire risk assessment, increasing model predictability.

3.3 Normalisation

Min-Max Normalisation was used to translate the characteristics into a range of [0, 1]. This approach is very successful for models sensitive to feature sizes, such as k-Nearest Neighbours (kNN), Support Vector Machines (SVM), and Artificial Neural Networks (ANN).

Normalisation guarantees that all features are on a similar scale, avoiding characteristics with more comprehensive ranges, such as those of the Drought Code, from controlling the model training process. This stage was vital for models using gradient-based optimisation as it allowed quicker convergence to an optimum solution. Even though changing outlier sensitivity was an option, the pros

of Min-Max scaling, like avoiding overfitting and ensuring that the model was trained moderately, were more excellent than the cons (Han et al., 2011). Because some factors, like area_log, aren't evenly distributed, the previous step's stratified sampling ensured that big and small fires were reflected.

IV. Data Sampling

Splitting Methodology

To build a good performance machine-learning model, splitting the data into training and testing groups after cleaning it is essential. This separation allows for generalisation, stops overfitting, and provides an excellent way to judge model success.

With stratified sampling, this dataset was split into 30% for testing and 70% for training. This method keeps the log-transformed burned area (area_log) distribution of the goal variable across both groups. Because of the uneven distribution of fire sizes, with fewer big fires than small ones, the stratified selection ensures that both sets include both types of fires, removing any bias (Kuhn and Johnson, 2013). This method is essential for models that were learned on unknown data.

Stratification on area_log helps balance the goal variable's extremes so that the model can learn and predict the full range of burning areas. On the other hand, random selection might not include enough outliers, which could lead to wrong predictions for big fires.

Challenges with Imbalanced Target Variables

One issue is that the goal variable isn't balanced, and many cases show small or unburned areas. In other words, models may learn too much from the ruling class. By ensuring that both big and small fires are in the training and testing sets, stratified sampling helps fix this problem and makes it easier for the model to learn from and predict across the whole target distribution (Fernández et al., 2018).

Potential Data Leakage Concerns

Data leakage is a typical issue when preparing data for machine learning, particularly when performing transformation such as scaling. Early data separation helps prevent biases that can compromise the model assessment (Kaufman et al., 2012). Data leakage often results in unduly optimistic performance indicators, skew model assessment and generalizability.

V. Building the Model

5.1 Regression Models

As you can see, this part is all about carefully building several regression models to guess the area_log (log-transformed burned area) variable. To find the best way, we try a number of them. Because of the kind of data, we have and how complex the problem is, we use Decision Trees, Random Forest, Support Vector Regression (SVR), k-Nearest Neighbours (kNN), and Artificial Neural Networks (ANN).

We want to use various methods to find out more and determine which models best match our standards for making predictions.

More than 70% of the data set is used as training. This ensures that models only learn patterns that make sense and don't add any bias. Models can find links between predictions and the goal variable (area_log) by training on 70% of the data and testing on the other 30%. Cross-validation on the training set also helps to find the best hyperparameters, which prevents overfitting and ensures generalisation (Hastie et al., 2009).

5.1.2 Decision Tree Regression

Model Building

The “rpart” package in R was used to create the decision tree model. This package makes it simple to use decision trees. The model was trained using all the features from the training dataset on the goal variable, which was the burned area turned into logarithms (area_log).

We used an ANOVA splitting rule that lowers the number of squares at each split to make the model. When the goal variable is continuous, this measure works well for regression tasks (Hastie et al., 2009). To find the best mix between model complexity and forecast accuracy, the complexity parameter (cp) and other hyperparameters were picked by lowering the cross-validation error as much as possible.

To create the tree structure, the information is broken up into groups repeatedly. The quality that narrows the range in the goal variable the most determines which split exists. The process of breaking keeps going until it reaches a stopping point, such as a certain level in the tree or the node with the fewest data.

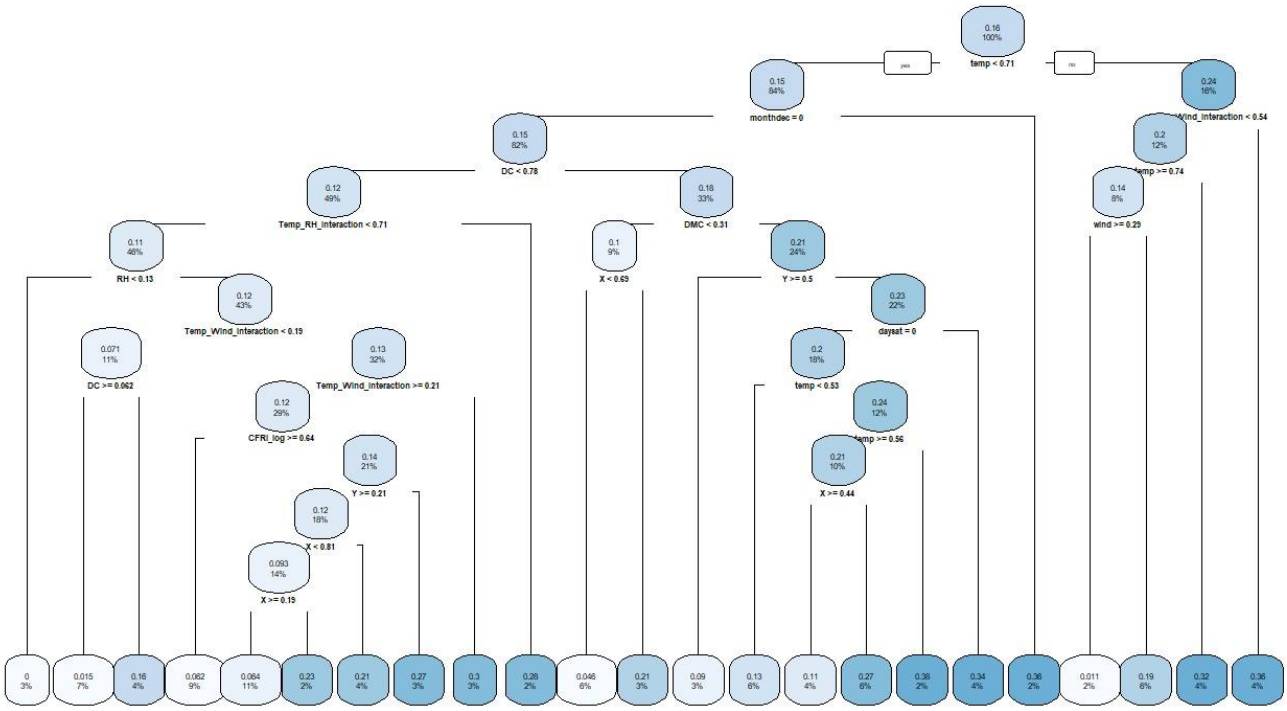
Model Summary

Key Splits: The root node split when temp < 0.709, which shows that temperature is an important factor in determining where the fire will happen.

Important Features: More important splits were seen on Temp_RH_Interaction < 0.714 and DC < 0.78, which shows that the Drought Code (DC) measure and the interaction between temperature and relative humidity are essential.

Terminal Nodes: There were nodes like Node 32 (RH < 0.125) and Node 268 (CFRI_log >= 0.637) that did not split any further and were the endpoints of predictions.

Visual Presentation



This figure shows some of the situations in which the expectations for the burned area are different. Each node shows a key split along with its related deviance number and prediction value.

Key Metrics:

- Number of terminal nodes (leaf nodes): 25.
- Temperature (temp), the relationship between temperature and relative humidity (Temp_RH_Interaction), and the drought code (DC) are the most important factors for lowering deviations.

Summary Table of Key Nodes

To provide clarity, the important values are summarised in a table, with an emphasis on the top ten attributes.

Node ID	Split Condition	Number of Observations	Deviance Reduction	Prediction Value (yval)
Root	temp < 0.709	360	15.51	0.1649
2	monthdec < 0.5	302	11.17	0.1500
8	DC < 0.78	175	5.31	0.1201
16	Temp_RH_Interaction < 0.714	166	4.34	0.1114
66	Temp_Wind_Interaction < 0.189	39	0.68	0.0715
134	Temp_Wind_Interaction >= 0.211	106	2.76	0.1167

538	Y >= 0.214	64	1.53	0.1196
17	Terminal Node	9	0.73	0.2817
18	DMC < 0.314	32	0.90	0.1022
315	Terminal Node (temp < 0.558)	7	0.31	0.3849

5.1.2 Random Forest Regression

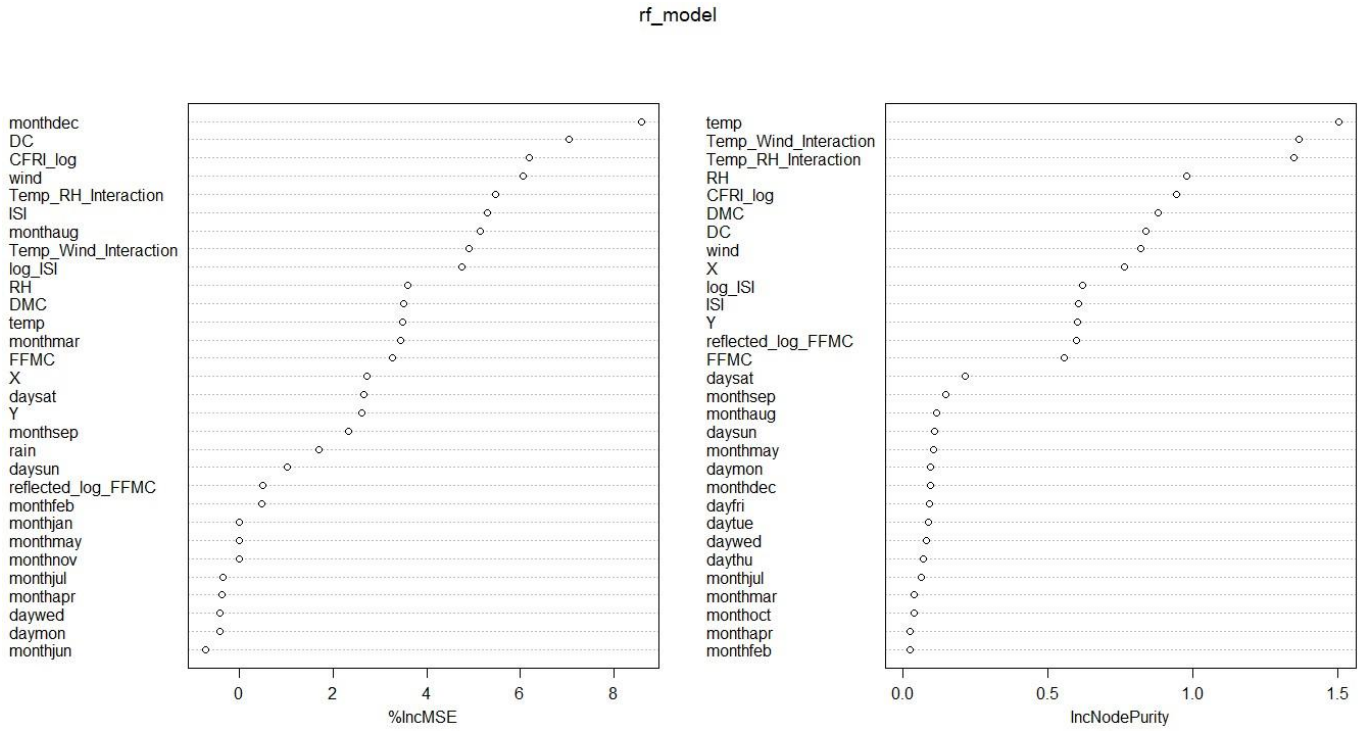
Model Building

The randomForest function was used to build the Random Forest model, which used the response variable area_log and all the other features as indicators. We set the number of trees (ntree) to 500 to get a good mix between how fast the model works and how accurate it is. Feature value was also estimated to understand the model and determine which factors make the most important contributions to the results.

Feature Importance Analysis

Random Forest uses two main measures to show how important each feature is:

- % Increase in Mean Squared Error (%IncMSE): This shows how changing a feature randomly affects the accuracy of a prediction. Higher values indicate more significant importance.
- Increase in Node Purity (IncNodePurity): This shows how much residual error has been reduced when the feature splits nodes across trees. **Visual Presentation**



The feature importance plot highlights the top predictors influencing fire severity:

- Key Predictors: The factors that made the most difference were monthdec (December), Temp_RH_Interaction, wind, and DC. It is essential to know about these things to understand how forest fires spread, with yearly effects and weather relations playing a significant role.
- Interaction Features: Temp_RH_Interaction and Temp_Wind_Interaction made significant contributions, emphasising the combined effect of temperature, humidity, and wind.
- Less Relevant Features: Certain categorical variables, like monthjun (June) and daytue, had negative or zero importance, which means they do not have much of an effect on how accurate the model is.

Summary table of the top ten features

Rank	Feature Name	%IncMSE
1	monthdec	8.59
2	DC	7.05
3	Temp_RH_Interaction	5.48
4	ISI	5.30
5	Temp_Wind_Interaction	4.90
6	log_ISI	4.75
7	monthaug	5.15
8	RH	3.59
9	temp	3.48
10	DMC	3.50

5.1.3 Support Vector Regression

Model Building

The SVR model was made with epsilon-regression and an RBF kernel set up as follows:

- SVM-Type: epsilon-regression means that the model attempts to match the data points within an epsilon range.
- Kernel Type: radial basis function (RBF), which lets features be mapped into higher dimensions in a way that is not linear.
- Cost (C): This is set to 1, which strikes a balance between lowering the training error and controlling the model's complexity.
- Gamma: is equal to 0.0294, tells the model how much each data point changes it. There is a decision limit close to the data points when gamma is high. There is a smoother decision function when gamma is low.
- Epsilon: set to 0.1, which tells the model to ignore small mistakes within a specific range.

The model used 331 support vectors, which shows that many data points went into making the border that the SVR used. The fact that there are so many support vectors in the dataset suggests that it needs complicated decision boundaries. This is common in models that try to predict forest fires because of the non-linear relationships between environmental predictors and fire size.

5.1.4 k-Nearest Neighbors Regression

Model Building

The caret package in R was used to build the kNN model. Here are the steps:

- Cross-Validation: Five rounds of cross-validation were conducted to ensure the model worked well with various data sets. Cross-validation prevented models from fitting too well and gave a more true picture of their performance (Altman, 1992).
- A grid search was used to tune the hyperparameters to find the best number of neighbours (k) for the model. The value of k was set between 3 and 15, with each 2 representing a different amount of closeness for the forecast. This tuning method aimed to find the best balance of bias and variation. The model might fit too well if k is too small and not enough if k is too large (Hastie et al., 2009).

The grid search showed that k = 15 was the most stable and easy-to-understand model. An area with 15 was less affected by noise and didn't overfit. k = 15 was chosen because it had the right amount of bias and variance. This lets the model find important trends without getting too complicated.

5.1.5 Artificial Neural Network

Model Building

The ANN model was constructed using the following approach:

- Neural Network Structure: Using a 34-5-1 design, the model had 34 input features, 5 hidden nodes in a single hidden layer, and 1 output node for predicting area_log. The choice of 5 hidden nodes struck an excellent mix between model complexity and the risk of overfitting (Hornik et al., 1989).
- Tuning Parameters:
    - Set the max parameter to 500 to specify the most iterations the model may run to minimise the loss function. After the final iteration, the model reached a final loss value of 9.066548, which means that the training successfully reduced prediction mistakes on the training set.
    - The decay value was set at 0.01, which provided regularisation to avoid overfitting by mitigating the influence of high weights.

5.2 Classification Models

5.2.1 Decision Tree

Model Building

A decision tree classification model was made with the rpart package in R to put the burned area (area_category) into groups. The model used the Gini impurity criterion to find the best splits, which reduced the amount of error in classification. At each node, the tree split based on the trait that made the impurities less noticeable the most. The tree was grown in steps, limiting how deep it could go and how many samples needed from each node to keep it from fitting too well. This method ensured a balance between model complexity and speed while still letting the model be understood (Breiman et al., 1984).

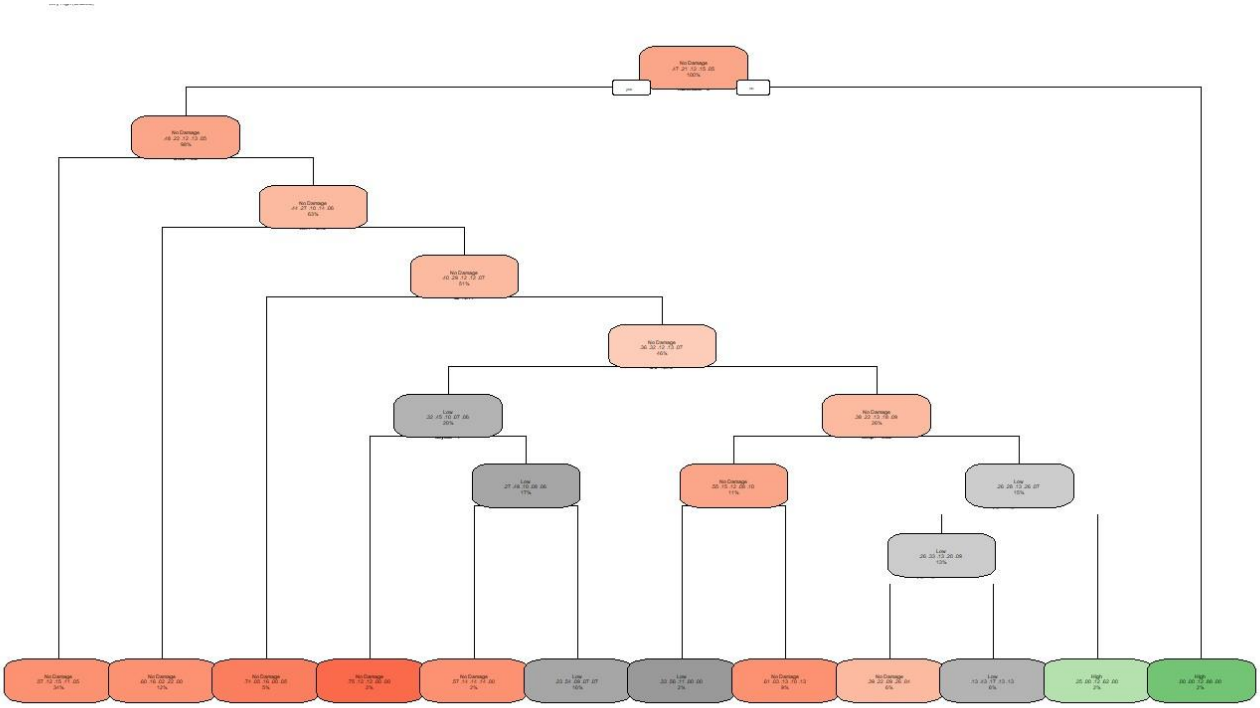
Model Summary

Key Splits: The first split was at monthdec < 0.5, which meant that December was a good indicator of how destructive the fire would be. DMC < 0.3018 and RH < 0.4940 were two more critical breaks.

Important Features: Relative Humidity (RH), Drought Code (DC), and temperature were essential for sorting, as shown by how often they were used in splits.

Terminal Nodes: The model ended with 25 terminal nodes, which showed the different paths used to predict the intensity levels.

Visual Presentation



The narrative clearly shows the splits and the decision-making process. By showing the splitting condition, number of data, and projected group for each point in the plot, it is easier to see how the model works.

Summary Table of Key Nodes

To provide clarity, the important values are summarised in a table, with an emphasis on the top ten attributes.

Node ID	Split Condition	Number of Observations	Gini Reduction	Prediction Value (yval)
Root	monthdec < 0.5	361	0.47	No Damage
4	DMC < 0.3018	124	0.57	No Damage
11	RH < 0.4940	184	0.40	No Damage
22	ISI < 0.1068	19	0.74	No Damage
46	DC < 0.7782	71	0.32	Low
94	temp >= 0.6495	40	0.55	No Damage
95	temp < 0.6495	54	0.26	Low
380	Temp_RH_Interaction < 0.4487	23	0.39	No Damage
381	Temp_RH_Interaction >= 0.4487	23	0.13	Low
3	monthdec >= 0.5	8	0.88	High

5.2.2 Random Forest

Model Building

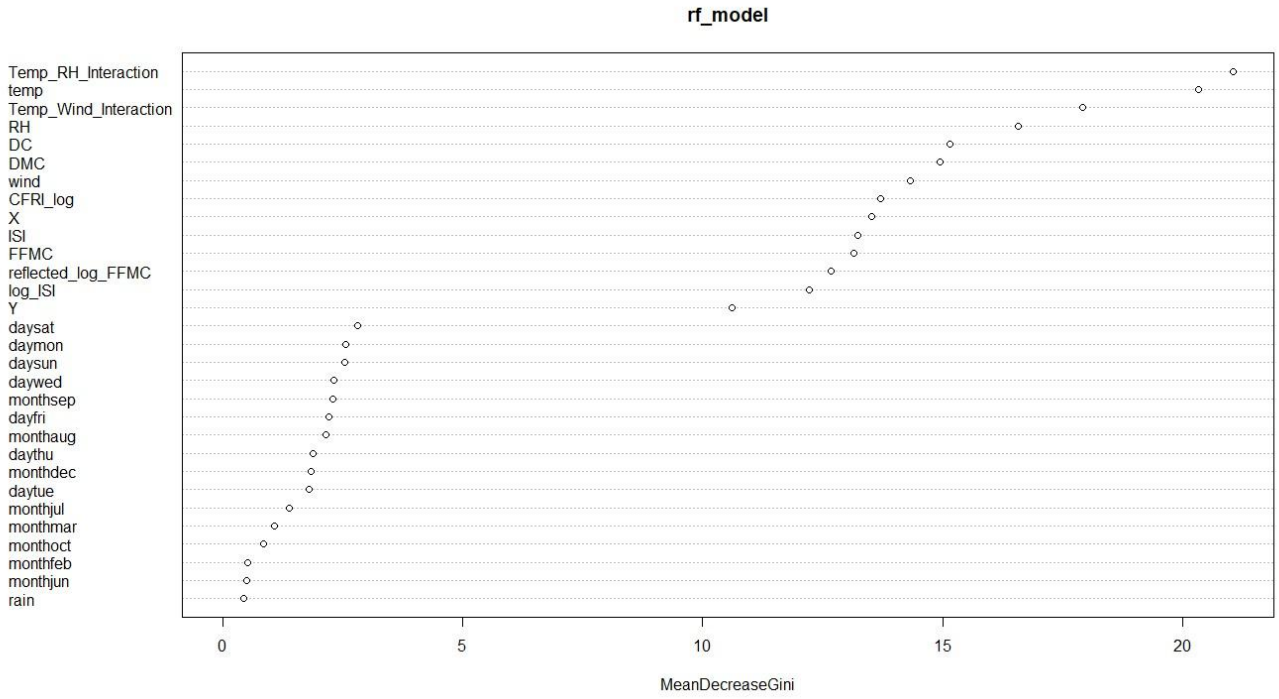
To find a good mix between speed and processing efficiency, the model was built with 500 trees (ntree = 500). Using 500 trees ensures the model's estimates are stable while lowering variation and overfitting. This is one of the main benefits of ensemble learning methods like Random Forest (Breiman, 2001).

Feature Importance Analysis

The Mean Decrease Gini measure was used to determine each feature's importance. The most important parts were:

- Temp_RH_Interaction and Temperature (temp): These factors were critical because they showed how temperature and its relationship with relative humidity affected the intensity of the fire.
- Relative Humidity (RH) and Drought Code (DC): These indicators were also critical because they showed how the fire damage was caused.
- Less Influential Features: Categorical features, such as rain_category, did not have much effect.

Visual Presentation



The varImpPlot shows clearly how important each feature is, highlighting the critical factors that affect how severe a fire is classified. Temp_RH_Interaction and temperature strongly affect the model, as shown by their high Gini values.

Summary Table of The Top Ten Features

Rank	Feature Name	Mean Decrease Gini
1	Temp_RH_Interaction	21.06
2	temp	20.33
3	Temp_Wind_Interaction	17.92
4	RH	16.56
5	DC	15.14
6	DMC	14.94
7	wind	14.33
8	FFMC	13.15
9	CFRI_log	13.71
10	ISI	13.23

5.2.3 Support Vector Machine

Model Building

The svmRadial method was used to evaluate the model's success after being trained using crossvalidation (10-fold CV). It was a hyperparameter tuned for the cost parameter (C), which manages the trade-off between the margin width and the training error. This changes the model's complexity and the risk of overfitting.

Key Insights from Cross-Validation

- Tuning Parameter (C): The cost parameter (C) was set to work best at C = 1, giving a 49% accuracy. The Kappa number was 0.044, which means it was hard to tell the difference between the fire intensity levels.
- Optimal Model: The final model had sigma = 0.29 and C = 1 to find the best mix between complexity and efficiency.

5.2.4 k-Nearest Neighbours

Model Building

The caret package in R was used to build the k-Nearest Neighbors (kNN) classification model, and 5fold cross-validation was used to test its effectiveness. From 3 to 15, the constant k (number of neighbours) was tuned, and k = 9 was found to be the best value.

5.2.5 Artificial Neural Network

Model Building

With the nnet package in R, the Artificial Neural Network (ANN) model was made. The framework of the neural network was made with:

- Hidden Layer Size: 5 neurons in a single hidden layer.
- Maximum Iterations: The model was trained 500 times in total to ensure it learned enough without becoming too perfect.
- Weight Decay: Using decay = 0.01, a regularisation constant, to stop overfitting by punishing big weights helped keep the model generalizable.

VI. Model Evaluation

6.1 Regression Metrics

In this section, we assess the effectiveness of each regression model designed to predict the logtransformed burned area of forest fires (area_log). Model evaluation is vital to find out how well a model fits new data. This tells how well it can predict, how accurate, and how useful it is. Based on the features of the information, this step helps determine which method gives the most correct results.

Evaluation Approach

To assess model performance, we used the following key metrics:

- **Root Mean Squared Error (RMSE):** Is a cost function that finds the square root of the mean square error and scales mistakes to match goal values (James et al., 2013).
- **Mean Absolute Error (MAE):** Finds the mean size of the mistakes in a set of predictions, regardless of how they were made. The mean absolute difference between the expected and actual numbers is worked out (James et al., 2013).
- **R-Squared:** The R-Squared value shows how much of the variation in the dependent variable can be accounted for by the factors that are not dependent. It also shows how well the guesses fit the actual data points (James et al., 2013).

Test Data Evaluation

The evaluation was done on the 30% test dataset that was kept from the model-building stage to ensure that the models were tested on data they had not seen before. This method objectively measures the model's ability to generalise, which is very important for avoiding overfitting (Hastie et al., 2009). When a model learns noise or particular patterns from the training set that do not generalise well to new data, overfitting results.

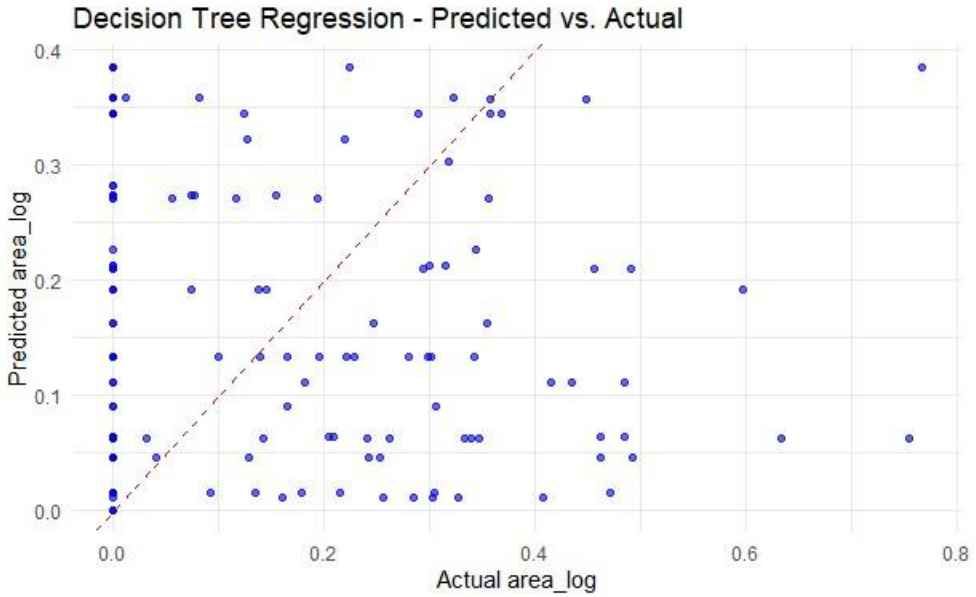
6.1.1 Decision Tree Regression

Evaluation Metrics

Three important metrics were calculated in order to assess the Decision Tree regression model:

- Root Mean Squared Error (RMSE = 0.217): A number of 0.217 means that the model's results are mostly close to the real values, but there are still some errors that need to be fixed, which means that the model could be better.
- Mean Absolute Error (MAE = 0.172): The MAE shows that, on average, the model's results are pretty close to the real numbers, with only small differences.
- R-Squared (R^2 = 0.00035): The model doesn't seem to be able to explain much of the variation in the data, as shown by the very low R^2 number. In this case, the model is too simple, and important connections in the data are being missed, which is called underfitting.

Visual Presentation



The plot demonstrates a notable departure from the actual values and a significant clustering of expected values around zero. Because the model finds it challenging to adequately represent the connections between features and the target variable, this lack of alignment with the red dashed line points to underfitting and low prediction accuracy. Especially for values over 0.2, the figure shows a significant prediction error and a broad dispersion of points away from the diagonal, indicating that the model lacks the required complexity or data representation to generalise beyond the training set.

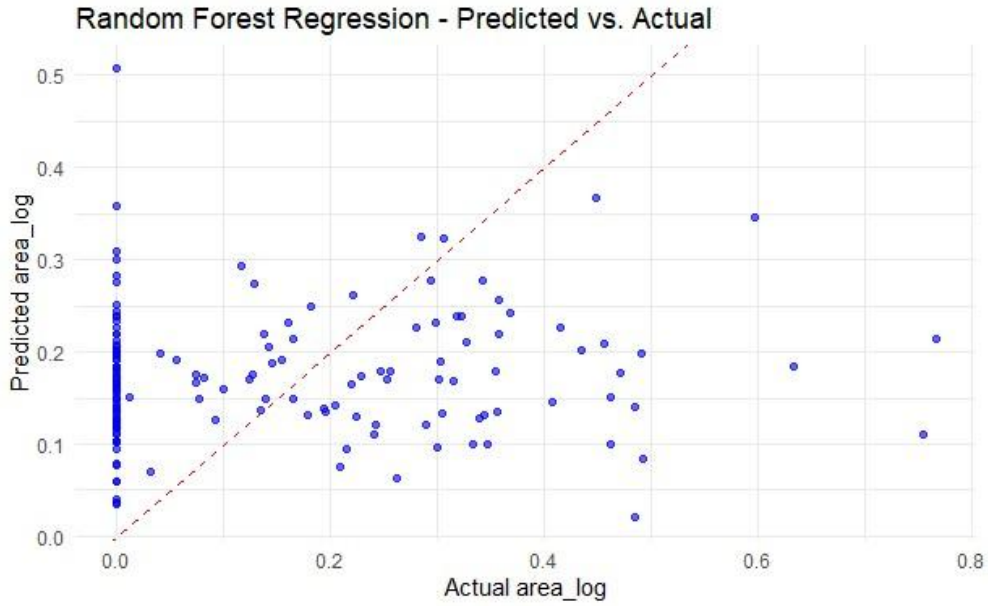
6.1.2 Random Forest Regression

Evaluation Metrics

Three important metrics were calculated in order to assess the Random Forest regression model:

- RMSE = 0.188: The result indicates that the Random Forest model may provide more accurate forecasts than the Decision Tree model, with fewer prediction mistakes.
- MAE = 0.15: It shows that the average difference between anticipated and actual values is less, emphasising the model's enhanced predictive accuracy.
- R^2 = 0.0084: While still low, this R^2 result is somewhat better than the Decision Tree model.

Visual Presentation



With a few predicted values lining closer to the red dashed line, the plot demonstrates modest improvement over the Decision Tree. Still, many anticipated locations remain dispersed, particularly in the upper ranges; predictions around zero still cluster. This implies that Random Forest still finds it difficult to sufficiently capture the complexity of the data even if it gains from ensemble learning, maybe owing to overfitting or constraints in the data used for training.

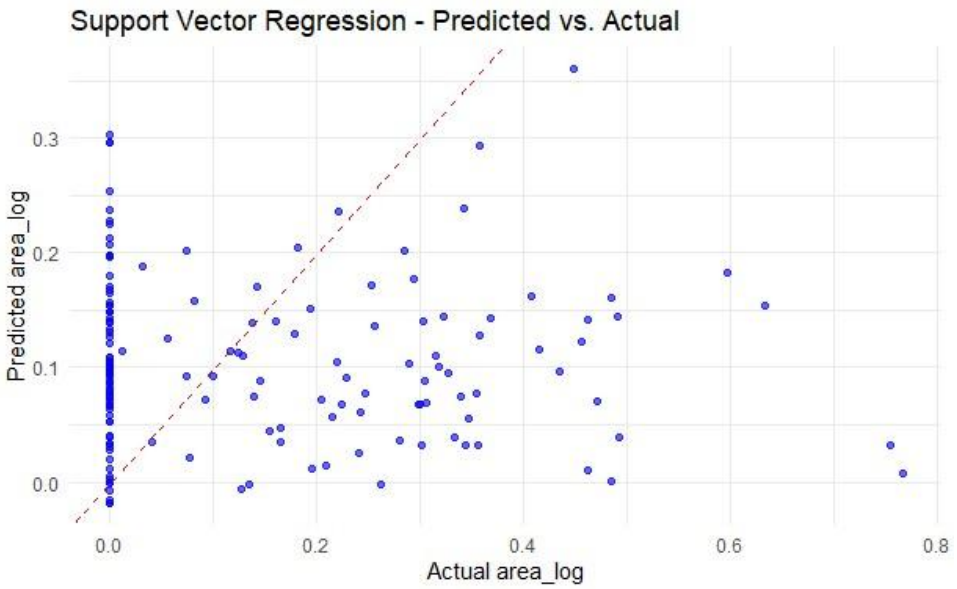
6.1.3 Support Vector Regression

Evaluation Metrics

Three important metrics were calculated in order to assess the Support Vector Regression model:

- RMSE = 0.199: An RMSE of 0.199 implies a moderate prediction error, which is somewhat worse than a Random Forest but better than a Decision Tree.
- MAE = 0.153: The MAE value of 0.153 indicates that, on average, the SVR model's predictions are closer to actual values than the Decision Tree but comparable to the Random Forest.
- R^2 = 0.00077. The SVR model's low R^2 value indicates that it cannot fully explain the variation in the target variable, highlighting its inadequate capacity to grasp the data's underlying complexity.

Visual Presentation



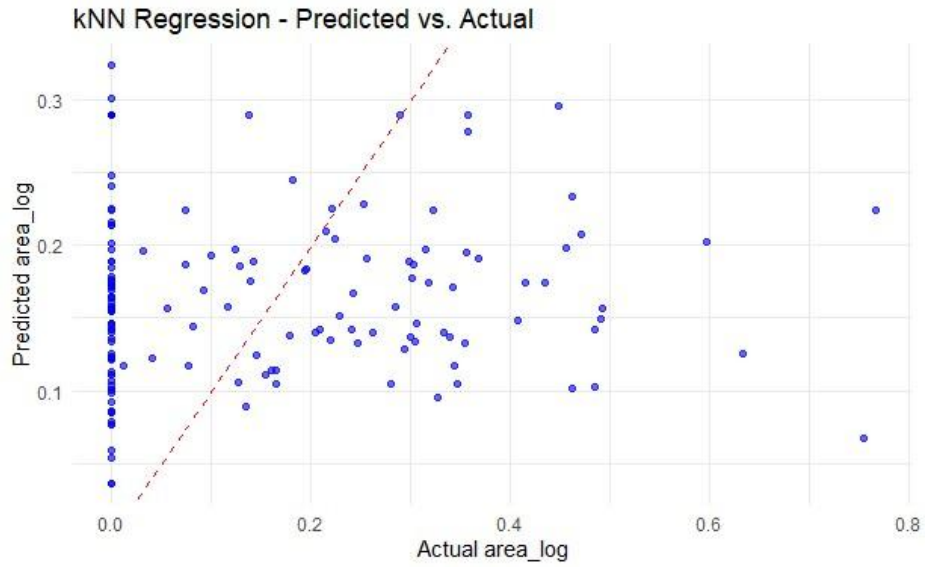
With predictions mainly concentrating on lower values and deviating from the actual values along the diagonal, the SVR model exhibits a similar pattern to the Decision Tree and Random Forest. Although SVR uses a kernel-based method, it does not considerably improve alignment, which might point to problems with parameter tuning or RBF kernel limits in the relation-based modelling of this dataset.

6.1.4 kNN Regression

Evaluation Metrics

Three important metrics were calculated in order to assess the kNN regression model:

- RMSE = 0.185: It indicates that kNN has the lowest prediction error of the tested models, suggesting a good match.
- MAE = 0.154: It shows that the kNN model's average prediction error is somewhat greater than Random Forest, but lower than the Decision Tree and SVR models.
- R^2 = 0.0039: Still low, indicating difficulty interpreting data variance. **Visual Presentation**



With numerous predictions focused around zero, like previous models, the kNN model finds it challenging to match projected values with actual ones. This may be ascribed to the bias-variance tradeoff and the sparsity of data points in certain areas, restricting kNN's capacity to provide correct predictions. For this dataset, the non-parametric character of kNN is less successful, suggesting that it may not be suitable for catching the associations required for correct regression.

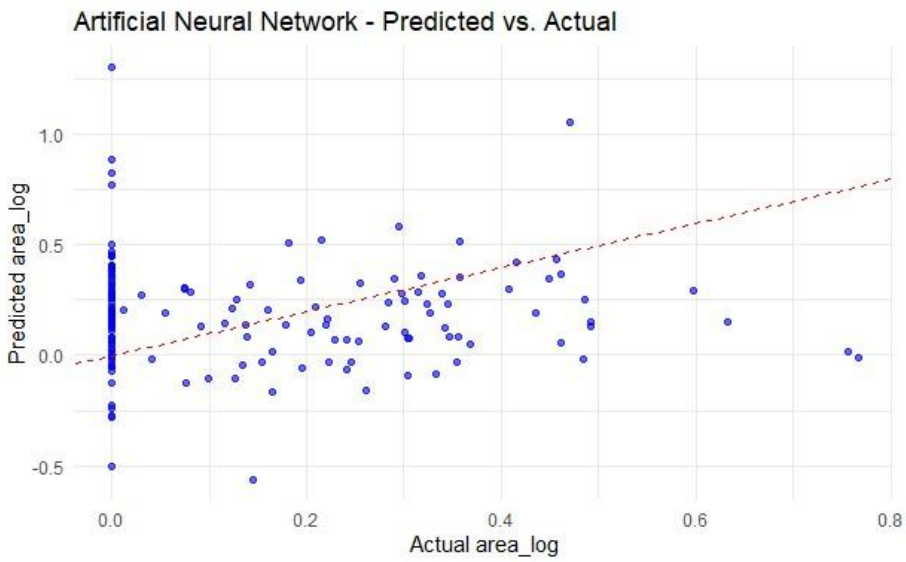
6.1.5 Artificial Neural Network

Evaluation Metrics

Three important metrics were calculated in order to assess the ANN regression model:

- RMSE = 0.305: The ANN model had the highest RMSE of the models tested, suggesting considerable prediction errors and a difficulty to match the data properly.
- MAE = 0.235: The result demonstrates the ANN model's poor performance, since it has a greater average prediction error than other models.
- R^2 = 0.00011: Extremely low, indicating poor model performance and lack of complexity.

Visual Presentation



Though it still has difficulty matching the actual values, the ANN model exhibits a more significant forecast dispersion than other models. Although the trend is generally upward, the significant volatility and broad dispersion of points indicate a possible underfitting of the model. Inappropriate hyperparameter tuning, insufficient training data, or poor model complexity might all contribute to the performance failing to capture the underlying connections adequately.

6.1.6 Model Comparison Table

Regression Models	RMSE	MAE	R-Squared
Decision Tree	0.217	0.172	0.00035
Random Forest	0.188	0.156	0.0084
Support Vector Regression	0.199	0.153	0.00077
kNN Regression	0.185	0.154	0.0039
Artificial Neural Network	0.305	0.235	0.00011

Best Performing Model: Along with the most excellent R-Squared value (0.0084), the Random Forest model showed the lowest RMSE (0.188) and MAE (0.156), therefore suggesting it was the most successful in projecting the log-transformed burnt area.

Least Performing Model: The Artificial Neural Network (ANN) performed the worst. It had the biggest RMSE (0.305), MAE (0.235), and R-Squared value (0.00011), which meant it was the worst model for regression in this case.

6.2 Classification Metrics

We used the following key measures to identify how well each classification model worked:

Confusion Matrix: This shows how many true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) there were in each class.

Accuracy: Accuracy quantifies the overall correctness of the model's predictions; however, it may be deceptive in unbalanced datasets (James et al., 2013).

Kappa Score: Kappa score checks how well-expected values match up with real values, taking into account the chance of picking at random. It gives a more complex picture of how well a classification system works, especially when the data isn't fair (James et al., 2013).

Precision: Precision calculates the number of true positives out of all the expected positives, to avoid false positives (James et al., 2013).

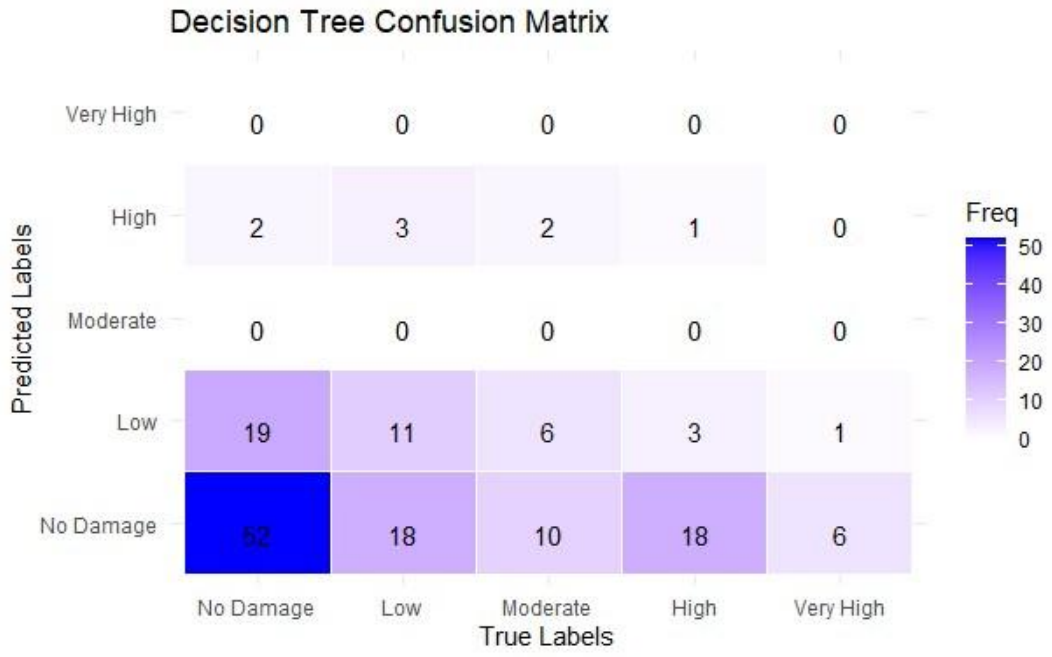
Recall: Recall assesses the capacity to detect all true positives while minimising false negatives (James et al., 2013).

F1 Score: The F1 score is the harmonic mean of accuracy and recall, establishing a balance between these two measures, particularly valuable when both false positives and false negatives are expensive (James et al., 2013).

6.2.1 Decision Trees

The confusion matrix and various metrics were computed to assess the decision tree's classification performance:

- Confusion Matrix and Statistics: The confusion matrix demonstrates that the decision tree accurately identified a large chunk of the "No Damage" class, while other classes had lower accuracy rates.



- Accuracy = 42.11%: Given the intricacy of the challenge and the class imbalance, this performance shows opportunity for improvement.

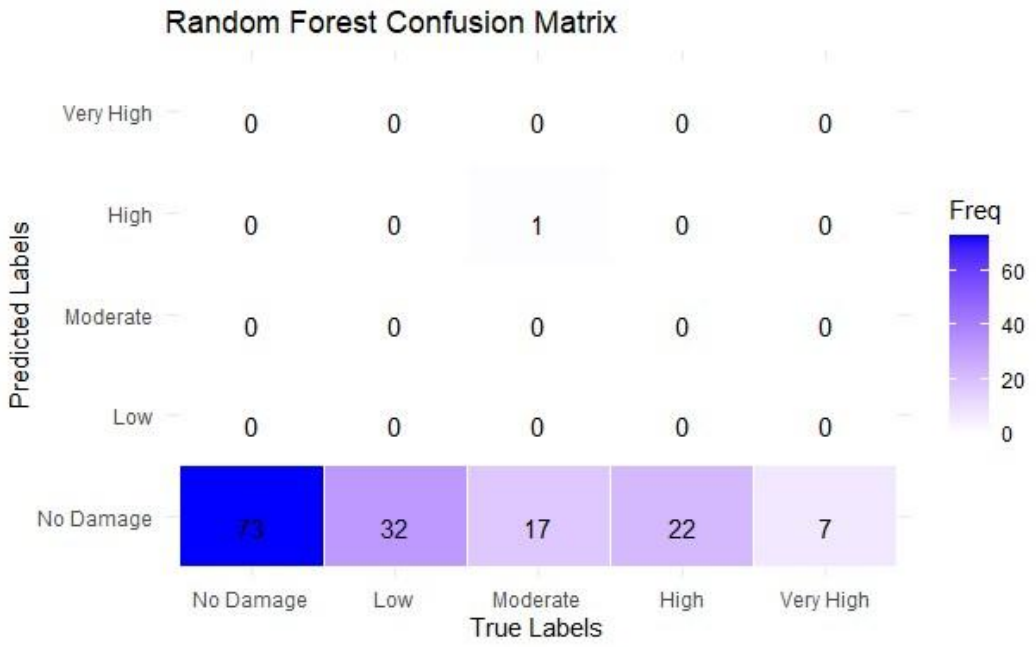
- Class-wise Performance:

- Sensitivity (Recall): The model has a high recall for the “No Damage” class (0.7123), which means it could correctly identify many cases. However, the recall values for other classes, especially Moderate and Very High, were 0, which means that these classes were not captured well.
- Specificity: The model was very good at identifying the Very High class.
- Balanced Accuracy: The adjusted accuracy values for the different classes were between 0.4958 and 0.5510. This shows how hard it is to describe some classes, especially those with few cases correctly.

6.2.2 Random Forest

The confusion matrix and various metrics were computed to assess the random forest’s classification performance:

- Confusion Matrix and Statistics: Only a few or none of the other groups were correctly forecast by the model. It mostly predicted the No Damage class. Based on this, the Random Forest model may favour the majority class (No Damage).



- Accuracy = 48.03%: This shows that the model is having a hard time getting better at classifying things than just guessing.
- Kappa = 0.0042: There seems to be very little agreement between the predictions and the real classes.
- Precision: The No Damage class has a precision of 0.48, which means that 48% of the occurrences predicted as “No Damage” were right.
- Recall:

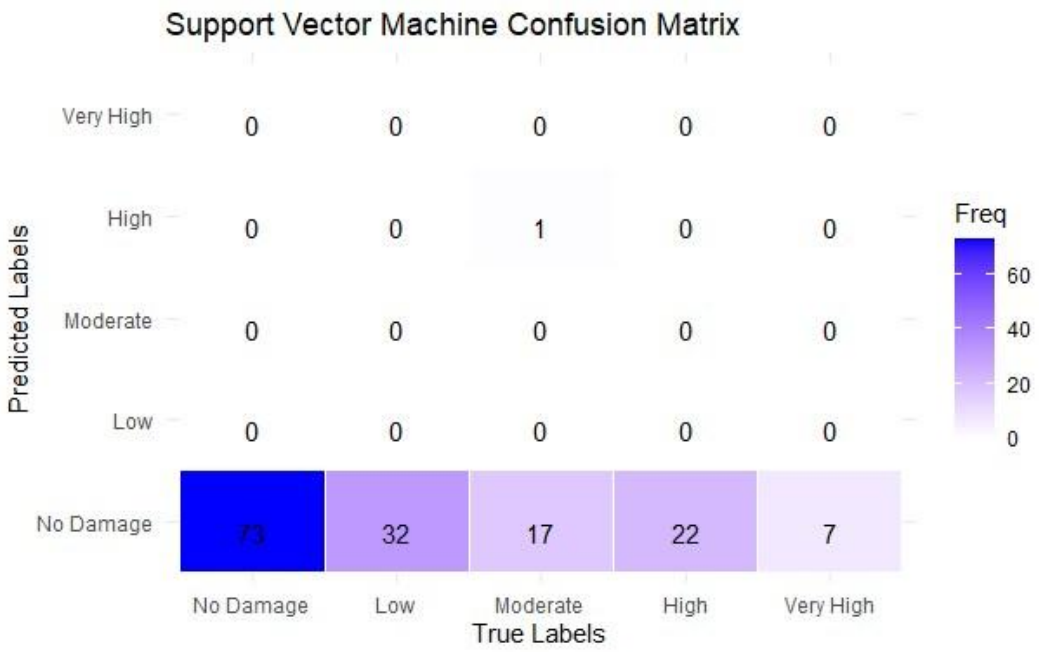
-
  - Class recall for the “No Damage” class was 1.00, which means that all instances of that class in the test set were correctly forecast.

    Other classes, like Low, Moderate, High, and Very High, had a recall of 0, which means the model didn't get those classes right.
- Balanced Accuracy:
  - No Damage class: 0.5063
  - Other groups, such as Low, Moderate, High, and Very High, had an average accuracy of about 0.50, which showed that the model wasn't good at distinguishing between them.

6.2.3 Support Vector Machine

The confusion matrix and various metrics were computed to assess the support vector machine’s classification performance:

- Confusion Matrix and Statistics: Same issue happens with Support Vector Machine as shown in the previous models



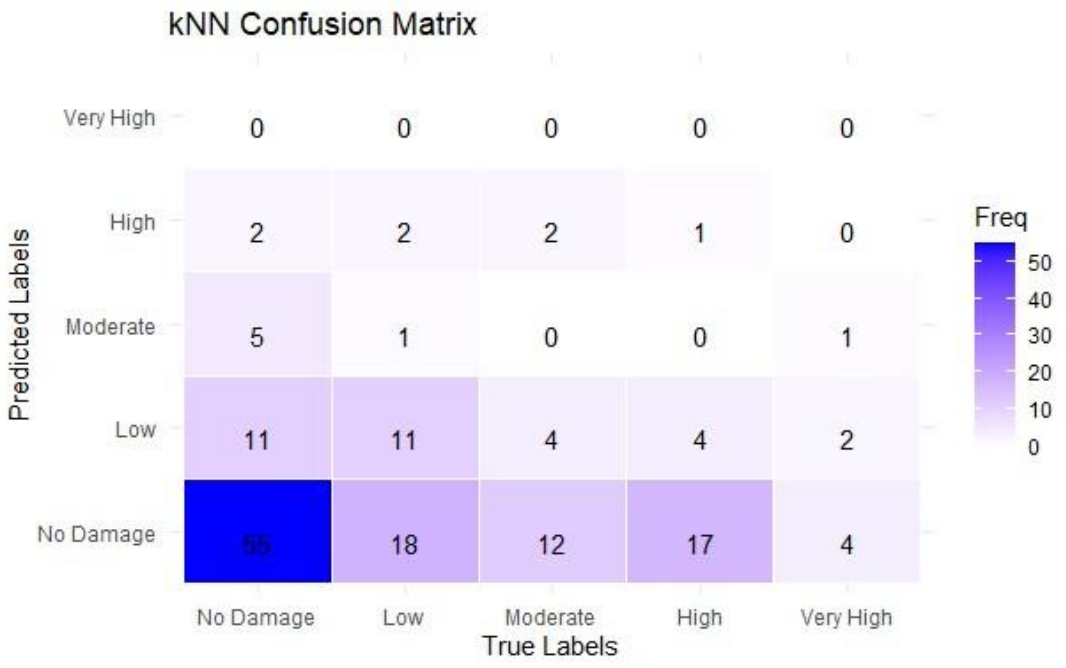
- Accuracy = 48.03%: showing that the model did not do better than random guessing.
- Kappa = 0.0042: was very low, which means that there wasn’t much agreement between the projected and real classes, just enough to be expected by chance.
- Precision:
  - The precision for the No Damage class is 0.48, which means that 48% of the times that “No Damage” was correctly forecast, it would have been true.

-
  - It was impossible to determine the accuracy for classes like Low, Moderate, High, and Very High because the forecasts were all zero, giving NaN numbers.
- Recall:
  - All instances of the “No Damage” class in the test set were correctly forecasted. This is called a recall of 1.00.
  - Low, Moderate, High, and Very High were the only classes with a recall of 0. This means the model did not predict any of these classes.
- Balanced Accuracy:
  - No Damage class: 0.5063
  - Other classes, such as Low, Moderate, High, and Very High, had an average accuracy of about 0.50, which showed that the model was not good at telling the difference between these groups.

6.2.4 k-Nearest Neighbours

The confusion matrix and various metrics were computed to assess the k-Nearest Neighbours’ classification performance:

- Confusion Matrix and Statistics: Same issue happens with kNN as shown in the previous models



- Accuracy = 44.08%: It was less accurate than the majority class average (48.03%). This shows that the model did not do better than random guessing.
- Kappa = 0.0812: was low, which means that forecasts and real classes did not match very well, which was not just a coincidence.

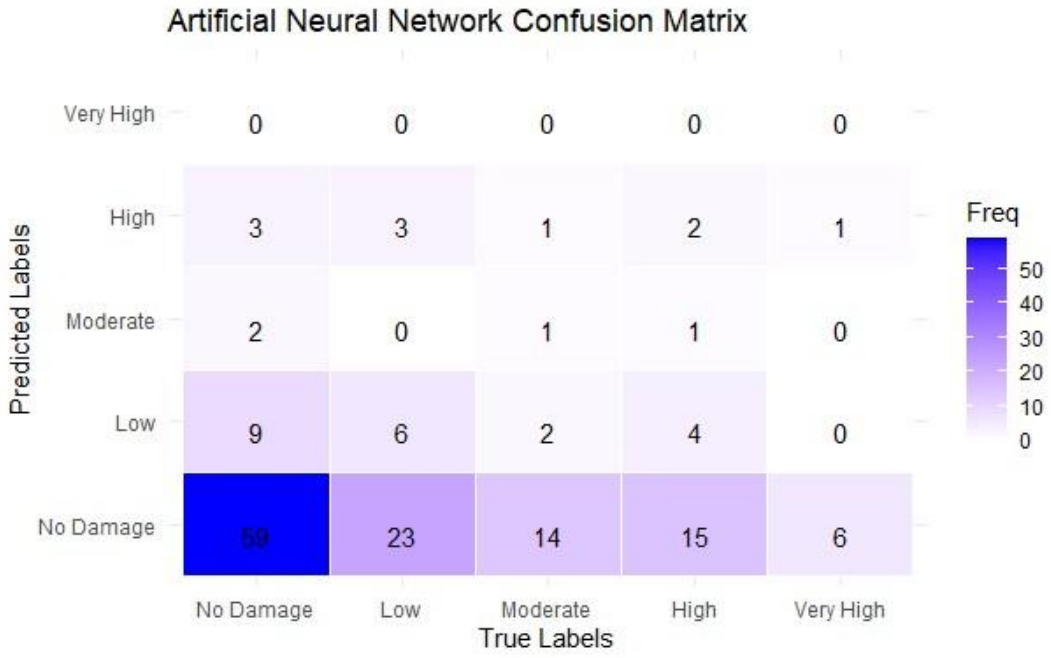
- Precision:
    - No Damage: 0.519, which means that 51.9% of the times that “No Damage” was forecast, it happened.
    - Low: 0.344, which means that 34.4% of guesses for the “Low” class were right.

      The accuracy numbers for the Moderate, High, and Very High classes were 0, which means that the model had a hard time figuring out what these classes were.

6.2.5 Artificial Neural Network

The confusion matrix and various metrics were computed to assess the Artificial Neural Network’s classification performance:

- Confusion Matrix and Statistics: Same issue happens with ANN as shown in the previous models



- Accuracy: 44.74% • Kappa: 0.0611
- Precision:
    - No Damage: 0.504, which means that 50.4% of the times that “No Damage” was forecast
    - Low: 0.286, which means that 28.6% of guesses were right for the “Low” class.
    - The accuracy numbers for the Moderate and High classes were 0.250 and 0.200, respectively, showing that the model did not predict these classes correctly.
- Recall:
    - With a recall of 0.808, the “No Damage” class performed best, correctly identifying 80.8% of the times it was used.

-
  - The Low class had a memory value of 0.188, while the Moderate, High, and Very High classes had recall values of 0.056, 0.091, and 0, respectively. This means that the model did not do well in these groups correctly predict these classes.
- Balanced Accuracy:

  The No Damage class had an adjusted accuracy of 0.537, which could tell the difference between this class's presence and absence.
  - An accurate score of 0.531 was given to the low class.
  - The accuracy scores for the Moderate, High, and Very High classes were all close to 0.5, which means that the model wasn't very good at telling the difference between these groups.

6.2.6 Model Comparison Table

Model	Accuracy	Kappa	Precision	Recall	F1 Score
Decision Tree	0.4211	0.0484	0.1800	0.2203	0.1920
Random Forest	0.4803	0.0042	0.0967	0.2000	0.1304
Support Vector Machine	0.4803	0.0042	0.0967	0.2000	0.1304
k-Nearest Neighbors (kNN)	0.4408	0.0812	0.2011	0.2285	0.2054
Artificial Neural Network	0.4474	0.0611	0.2480	0.2284	0.2127

Best Performing Model: The Artificial Neural Network (ANN) did the best, with an accuracy of 0.4474 and the highest F1 Score (0.2127), which means it did a better job of finding true positives and reducing false positives.

Least Performing Model: The Random Forest and Support Vector Machine (SVM) models did the worst, with low Kappa (0.0042) and Precision (0.0967), which means they weren't very good at correctly classifying the data.

VII. References

Aggarwal, C. C. (2016). *Outlier Analysis* (2nd ed. 2017.). Springer Nature. https://doi.org/10.1007/978-3-319-47578-3

Alcasena, F. J., Salis, M., Nauslar, N. J., Aguinaga, A. E., & Vega-García, C. (2016). Quantifying economic losses from wildfires in black pine afforestations of northern Spain. *Forest Policy and Economics*, *73*, 153–167. https://doi.org/10.1016/j.forpol.2016.09.005

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, *46*(3), 175–185. https://doi.org/10.1080/00031305.1992.10475879

○

Bradstock, R. A. (2010). biogeographic model of fire regimes in Australia: current and future implications. *Global Ecology and Biogeography*, *19*(2), 145–158. https://doi.org/10.1111/j.1466-8238.2009.00512.x

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. https://doi.org/10.1016/j.neunet.2018.07.011

Dormann, C. F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., Marquéz, J. R. G., Gruber, B., Lafourcade, B., Leitão, P. J., Münkemüller, T., McClean, C., Osborne, P. E., Reineking, B., Schröder, B., Skidmore, A. K., Zurell, D., & Lautenbach, S. (2013). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography (Copenhagen)*, *36*(1), 27–46. https://doi.org/10.1111/j.1600-0587.2012.07348.x

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). *Learning from Imbalanced Data Sets* (1st ed.). Springer International Publishing AG. https://doi.org/10.1007/978-3-319-98074-4

Flannigan, M. D., Krawchuk, M. A., de Groot, W. J., Wotton, B. M., & Gowman, L. M. (2009). Implications of changing climate for global wildland fire. *International Journal of Wildland Fire*, *18*(5), 483–507. https://doi.org/10.1071/WF08187

Fusco, E. J., Abatzoglou, J. T., Balch, J. K., Finn, J. T., & Bradley, B. A. (2016). Quantifying the human influence on fire ignition across the western USA. *Ecological Applications*, *26*(8), 2390–2401. https://doi.org/10.1002/eap.1395

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Second Edition). Springer. https://doi.org/10.1007/978-0-38784858-7

Jain, P., Coogan, S. C. P., Subramanian, S. G., Crowley, M., Taylor, S., & Flannigan, M. D. (2020). A review of machine learning applications in wildfire science and management. *Environmental Reviews*, *28*(4), 478–505. https://doi.org/10.1139/er-2020-0019

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R* (1st ed., Vol. 103). Springer Nature. https://doi.org/10.1007/978-1-46147138-7

Jiawei Han, Micheline Kamber, & Jian Pei. (2011). *Data Mining: Concepts and Techniques, 3rd Edition*. Morgan Kaufmann.

Johnston, L. M., Wang, X., Erni, S., Taylor, S. W., McFayden, C. B., Oliver, J. A., Stockdale, C., Christianson, A., Boulanger, Y., Gauthier, S., Arseneault, D., Wotton, B. M., Parisien, M.-A., & Flannigan, M. D. (2020). Wildland fire risk research in Canada. *Environmental Reviews*, *28*(2), 164–186. https://doi.org/10.1139/er-2019-0046

Jolly, W. M., Cochrane, M. A., Freeborn, P. H., Holden, Z. A., Brown, T. J., Williamson, G. J., & Bowman, D. M. J. S. (2015). Climate-induced variations in global wildfire danger from 1979 to 2013. *Nature Communications*, *6*(1), 7537–7537. https://doi.org/10.1038/ncomms8537

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining:

Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data*, *6*(4), 1–21. https://doi.org/10.1145/2382577.2382579

Keeley, J. E., & Syphard, A. D. (2018). Historical patterns of wildfire ignition sources in California ecosystems. *International Journal of Wildland Fire*, *27*(12), 781-. https://doi.org/10.1071/WF18026

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. *The Artificial Intelligence Review*, *26*(3), 159–190. https://doi.org/10.1007/s10462-007-9052-3

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling* (1st ed.). Springer Nature. https://doi.org/10.1007/978-1-4614-6849-3

Lange, N. (1997). Neural Networks for Pattern Recognition [Review of *Neural Networks for Pattern Recognition*]. *Journal of the American Statistical Association*, *92*(440), 1642–1645. American Statistical Association. https://doi.org/10.2307/2965437

Osborne, J. W. (2010). Improving Your Data Transformations: Applying the Box-Cox Transformation. *Practical Assessment, Research & Evaluation*, *15*(12), 12-.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. https://doi.org/10.5555/1953048.2078195

Rodrigues, M., & de la Riva, J. (2014). An insight into machine-learning algorithms to model humancaused wildfire occurrence. *Environmental Modelling & Software : With Environment Data News*, *57*, 192–201. https://doi.org/10.1016/j.envsoft.2014.03.003

Westerling, A. L., Hidalgo, H. G., Cayan, D. R., & Swetnam, T. W. (2006). Warming and Earlier Spring Increase Western U.S. Forest Wildfire Activity. *Science (American Association for the Advancement of Science)*, *313*(5789), 940–943. https://doi.org/10.1126/science.1128834

VIII. Appendices

Appendix A: Regression Models – Building and Evaluating

```
install.packages("dplyr")
install.packages("tidyverse")
install.packages("caret")
install.packages("randomForest")
install.packages("Metrics")
install.packages("e1071")  install.packages("rpart")
install.packages("nnet")
install.packages("Metrics")  library(dplyr)
library(tidyverse) library(VIM)
library(caret)       library(randomForest)
library(e1071) library(rpart) library(nnet)
```

```r
library(Metrics) library(rpart.plot)
library(ggplot2)


# Load file or Import Dataset
forestfires <- read.csv("forestfires.csv")


# Data Preparation ------------------------------------------------------
# Handle missing values and duplicates
na_data <- apply(forestfires, 2, function(x) sum(is.na(x))) print(na_data)
 dup_data <- forestfires[duplicated(forestfires), ]
print(paste("Number of duplicate rows:", nrow(dup_data)))
forestfires_clean <- distinct(forestfires)

print(paste("Number of rows after removing duplicates:",
nrow(forestfires_clean)))


# Transformation ------------------------------------------------------
# Log transformation and categorical encoding for area
forestfires_clean$area_log <- log1p(forestfires_clean$area)


# FFMC and ISI
max_FFMC <- max(forestfires_clean$FFMC)
forestfires_clean$reflected_log_FFMC <- log1p(max_FFMC - forestfires_clean$FFMC)

forestfires_clean$log_ISI <- log1p(forestfires_clean$ISI)

# One-hot encoding for 'month' and 'day'
dummies <- dummyVars(~ month + day, data = forestfires_clean)
encoded_data <- predict(dummies, newdata = forestfires_clean) %>% as.data.frame()

forestfires_clean <- bind_cols(forestfires_clean, encoded_data) %>%   select(-
month, -day)

# Feature engineering
forestfires_clean <- forestfires_clean %>%
mutate(Temp_RH_Interaction = temp * RH,
         Temp_Wind_Interaction = temp * wind,
         CFRI_log = (reflected_log_FFMC + DMC + DC + log_ISI) / 4)
# Data Sampling ------------------------------------------------------
set.seed(123)
train_index <- createDataPartition(forestfires_clean$area_log, p = 0.7, list =
FALSE)
train_data <- forestfires_clean[train_index, ] test_data
<- forestfires_clean[-train_index, ]

# Normalization ------------------------------------------------------
normalizer <- preProcess(train_data, method = c("range"))
```

```r
# Normalizing train and test datasets train_norm
<- predict(normalizer, train_data) test_norm <-
predict(normalizer, test_data)



# Exclude target variable from features
features <- setdiff(names(train_norm), c("area", "area_log"))

# Prepare training and testing datasets
X_train <- train_norm[, features] y_train
<- train_norm$area_log

X_test <- test_norm[, features] y_test
<- test_norm$area_log



# Evaluation
eva_modle <- function(true_values, predictions)
{   r2_va <- cor(true_values, predictions)^2
mae_va <- mae(true_values, predictions)   rmse_va
<- rmse(true_values, predictions)
  return(list(MAE = mae_va, RMSE = rmse_va, R2 = r2_va)) }



# Building Model ------------------------------------------------------------
# Decision Tree  ------------------------------------------- set.seed(123)

dt_model <- rpart(area_log ~ ., data = train_norm[, c(features, "area_log")],
method = "anova")

print(dt_model)
 ##
Plot
rpart.plot(dt_model, type = 2, fallen.leaves = TRUE, cex = 0.5)
# Predict Test dataset
dt_predict <- predict(dt_model, newdata = X_test)

# Evaluation
dt_eva <- eva_modle(y_test, dt_predict)
print("Decision Tree Regression Evaluation Metrics are shown:") print(dt_eva)



# Random Forest -----------------------------------------------
set.seed(123)
rf_model <- randomForest(area_log ~ ., data = train_norm[, c(features,
"area_log")], ntree = 500, importance = TRUE)
# Check Important Variables importance(rf_model)
varImpPlot(rf_model)

## Evaluating the Model #
Predict Test dataset
rf_predict <- predict(rf_model, newdata = X_test)
```

```r
# Evaluate
rf_eva <- eva_modle(y_test, rf_predict)
print("Random Forest Regression Evaluation Metrics are shown:") print(rf_eva)



# Support Vector Regression (SVR) ----------------------------------------
set.seed(123)
svr_model <- svm(area_log ~ ., data = train_norm[, c(features, "area_log")], type
= 'eps-regression', kernel = 'radial')

print(svr_model)

# Predict Test dataset
svr_predict <- predict(svr_model, newdata = X_test)

# Evaluate
svr_eva <- eva_modle(y_test, svr_predict)
print("Support Vector Regression Evaluation Metrics are shown:")
print(svr_eva)



# kNN ---------------------------------------------------------------------
set.seed(123)
train_control <- trainControl(method = "cv", number = 5)

knn_turn <- expand.grid(k = seq(3, 15, by = 2))

# Train model
knn_model <- train(area_log ~ ., data = train_norm[, c(features, "area_log")],
method = "knn",                    trControl = train_control,
tuneGrid = knn_turn)

# Pick Best performance
print(paste("Best k:", knn_model$bestTune$k))

# Predict Test dataset
knn_predict <- predict(knn_model, newdata = X_test)

# View model results print(knn_model$results)

# Evaluate
knn_eva <- eva_modle(y_test, knn_predict)
print("k-Nearest Neighbors Evaluation Metrics are shown:") print(knn_eva)



# ANN ---------------------------------------------------------------------
set.seed(123)
ann_model <- nnet(area_log ~ ., data = train_norm[, c(features, "area_log")],
size = 5, linout = TRUE, maxit = 500, decay = 0.01)  print(ann_model)

# Predict Test dataset
```

```r
ann_predict <- predict(ann_model, newdata = X_test)

# Evaluate
ann_eva <- eva_modle(y_test, ann_predict)
print("Artificial Neural Network Evaluation Metrics are shown:")
print(ann_eva)


# Compare All Model  ------------------------------------------------
model_performance <- data.frame(
  Model = c("Decision Tree", "Random Forest", "SVM", "kNN", "ANN"),
  MAE = c(dt_eva$MAE, rf_eva$MAE, svr_eva$MAE, knn_eva$MAE, ann_eva$MAE),
  RMSE = c(dt_eva$RMSE, rf_eva$RMSE, svr_eva$RMSE, knn_eva$RMSE, ann_eva$RMSE),
  R_Squared = c(dt_eva$R2, rf_eva$R2, svr_eva$R2, knn_eva$R2, ann_eva$R2)
)
print("Regression Model Performance Comparison is:") print(model_performance)


# Visulisation ------------------------------------------------------------
# Plotting Function
plotpredict <- function(true_values, predictions, model_name)
{   plotdf <- data.frame(True = true_values, Predicted = predictions)

  ggplot(plotdf, aes(x = True, y = Predicted)) +
geom_point(color = "blue", alpha = 0.5) +
    geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
ggtitle(paste(model_name, "- Actual vs. Predicted")) +     xlab("Actual
area_log") +     ylab("Predicted area_log") +     theme_minimal()
}


# Decision Tree
plotpredict(y_test, dt_predict, "Decision Tree")

# Random Forest
plotpredict(y_test, rf_predict, "Random Forest")
# SVR
plotpredict(y_test, svr_predict, "Support Vector Regression")
 #
kNN
plotpredict(y_test, knn_predict, "kNN")
 #
ANN
plotpredict(y_test, ann_predict, "Artificial Neural Network")
```

Appendix B: Classification Models – Building and Evaluating

```r
library(dplyr)
library(tidyverse)
library(caret)
library(randomForest)
library(e1071) library(rpart)
library(nnet)
library(rpart.plot)
```

```r
# Load file or Import Dataset
forestfires <- read.csv("forestfires.csv")


# Data Preparation ------------------------------------------------------------
# Handle missing values and duplicates
na_data <- apply(forestfires, 2, function(x) sum(is.na(x))) print(na_data)
 dup_data <- forestfires[duplicated(forestfires), ]
print(paste("Number of duplicate rows:", nrow(dup_data)))
forestfires_clean <- distinct(forestfires)

print(paste("Number of rows after removing duplicates:",
nrow(forestfires_clean)))



# Transformation --------------------------------------------------------------
# Log transformation and categorical encoding for area
forestfires_clean$area_log <- log1p(forestfires_clean$area)

epsilon <- 1e-5
forestfires_clean$area_cat       <-       cut(forestfires_clean$area_log,
breaks = c(0, log(1 + epsilon), log(5), log(10), log(50), Inf),
                                    labels = c("No Damage", "Low", "Moderate",
"High", "Very High"),
include.lowest = TRUE)

table(forestfires_clean$area_cat)

# FFMC and ISI
max_FFMC <- max(forestfires_clean$FFMC)
forestfires_clean$reflected_log_FFMC <- log1p(max_FFMC - forestfires_clean$FFMC)

forestfires_clean$log_ISI <- log1p(forestfires_clean$ISI)

# One-hot encoding for 'month' and 'day'
dummies <- dummyVars(~ month + day, data = forestfires_clean)
encoded_data <- predict(dummies, newdata = forestfires_clean) %>% as.data.frame()

forestfires_clean <- bind_cols(forestfires_clean, encoded_data) %>%   select(-
month, -day)

# Feature engineering
forestfires_clean <- forestfires_clean %>%
mutate(Temp_RH_Interaction = temp * RH,
         Temp_Wind_Interaction = temp * wind,
         CFRI_log = (reflected_log_FFMC + DMC + DC + log_ISI) / 4)
# Data Sampling ---------------------------------------------------------------
# Data splitting set.seed(123)
train_index <-
createDataPartition(forestfire
s_clean$area_cat, p = 0.7,
list = FALSE)
```

```r
train_data <- forestfires_clean[train_index, ] test_data
<- forestfires_clean[-train_index, ]
 table(train_data$area_cat)
table(test_data$area_cat)

# Normalisation ------------------------------------------------------------
# Normalization
features <- setdiff(names(train_data), c("area", "area_log", "area_cat"))
normalizer <- preProcess(train_data[, features], method = "range")
train_norm <- predict(normalizer, train_data[, features]) test_norm <-
predict(normalizer, test_data[, features])
 train_norm <- bind_cols(train_norm, area_cat =
train_data$area_cat) test_norm <- bind_cols(test_norm, area_cat =
test_data$area_cat)
 train_norm$area_cat <-
as.factor(train_norm$area_cat) test_norm$area_cat <-
as.factor(test_norm$area_cat)

# Prepare 2 datasets for modeling X_train
<- train_norm[, features] y_train <-
train_norm$area_cat

X_test <- test_norm[, features] y_test
<- test_norm$area_cat

# Evaluation function
evaluate_class <- function(true_values, predictions)
{   confusion <- confusionMatrix(predictions,
true_values)   accuracy <- confusion$overall["Accuracy"]
kappa <- confusion$overall["Kappa"]

  recall_per_class <- confusion$byClass["Sensitivity"]
precision_per_class <- confusion$byClass["Pos Pred Value"]

  recall_per_class[is.na(recall_per_class)] <- 0
precision_per_class[is.na(precision_per_class)] <- 0

  f1s_per_class <- 2 * (precision_per_class * recall_per_class) /
    (precision_per_class + recall_per_class)
f1s_per_class[is.na(f1s_per_class)] <- 0

  f1_score <- mean(f1s_per_class)
recall <- mean(recall_per_class)
precision <- mean(precision_per_class)

  return(list(ConfusionMatrix = confusion,
              Accuracy = accuracy,
Kappa = kappa,
              F1_Score = f1_score,
              Recall = recall,
              Precision = precision))
}


# Building Classification Models -----------------------------------------
```

```r
# Decision tree model
set.seed(123)
dt_model <- rpart(area_cat ~ ., data = train_norm, method = "class")
rpart.plot(dt_model, type = 2, fallen.leaves = TRUE, cex = 0.3)

dt_predict <- predict(dt_model, X_test, type = "class") dt_eva
<- evaluate_class(y_test, dt_predict)

print("Decision Tree Evaluation Metrics are shown:")
print(dt_eva$ConfusionMatrix)

# Random forest model set.seed(123)
rf_model <- train(area_cat ~ ., data = train_norm, method = "rf", trControl =
trainControl(method = "cv"), tuneLength = 10)
 importance(rf_model$finalModel)
varImpPlot(rf_model$finalModel)
 rf_predict <- predict(rf_model, X_test)
rf_eva <- evaluate_class(y_test, rf_predict)

print("Random Forest Evaluation Metrics are shown:")
print(rf_eva$ConfusionMatrix)

# SVM model set.seed(123)
svm_model <- train(area_cat ~ ., data = train_norm, method = "svmRadial",
trControl = trainControl(method = "cv"), tuneLength = 10)
 svm_predict <- predict(svm_model, X_test)
svm_eva <- evaluate_class(y_test, svm_predict)

print("SVM Classification Evaluation Metrics are shown:")
print(svm_eva$ConfusionMatrix)

# kNN model set.seed(123)
train_control <- trainControl(method = "cv", number = 5) knn_turn
<- expand.grid(k = seq(3, 15, by = 2))
 knn_model <- train(area_cat ~ ., data = train_norm, method =
"knn",                    trControl = train_control, tuneGrid =
knn_turn)
 print(paste("Best k:",
knn_model$bestTune$k)) plot(knn_model)
 knn_predict <- predict(knn_model, X_test)
knn_eva <- evaluate_class(y_test, knn_predict)
 print("kNN Classification Evaluation Metrics are
shown:") print(knn_eva$ConfusionMatrix)

# ANN model set.seed(123)
ann_model <- nnet(area_cat ~ ., data = train_norm, size = 5, maxit = 500, decay =
0.01, trace = FALSE)
ann_predict <- predict(ann_model, X_test, type = "class") ann_predict
<- factor(ann_predict, levels = levels(y_test))

ann_eva <- evaluate_class(y_test, ann_predict)

print("Artificial Neural Network Evaluation Metrics are shown:")
print(ann_eva$ConfusionMatrix)
```

```r
# Model performance comparison model_performance
<- data.frame(
  Model = c("Decision Tree", "Random Forest", "SVM", "kNN", "ANN"),
Kappa = c(dt_eva$Kappa, rf_eva$Kappa, svm_eva$Kappa, knn_eva$Kappa,
ann_eva$Kappa),
  Recall = c(dt_eva$Recall, rf_eva$Recall, svm_eva$Recall, knn_eva$Recall,
ann_eva$Recall),
  Accuracy = c(dt_eva$Accuracy, rf_eva$Accuracy, svm_eva$Accuracy,
knn_eva$Accuracy, ann_eva$Accuracy),
  F1_Score = c(dt_eva$F1_Score, rf_eva$F1_Score, svm_eva$F1_Score,
knn_eva$F1_Score, ann_eva$F1_Score),
  Precision = c(dt_eva$Precision, rf_eva$Precision, svm_eva$Precision,
knn_eva$Precision, ann_eva$Precision)
)
print("Shuffled Classification Model Performance Comparison:")
print(model_performance)

# Visualization ----------------------------------------------------------
# Confusion Matrix Visualization
plotmatrix <- function(true_values, predicted_values, model_name)
{   confmatrix <- table(True = true_values, Predicted = predicted_values)
confmatrix_df <- as.data.frame(confmatrix)

  ggplot(data = confmatrix_df, aes(x = True, y = Predicted, fill = Freq)) +
geom_tile(color = "white") +
geom_text(aes(label = Freq), vjust = 0.5) +
scale_fill_gradient(name = "Frequency",
low = "white", high = "blue") +       labs(title = paste("Confusion Matrix
for", model_name),            x = "True Class", y = "Predicted Class") +
theme_minimal()
}

# Decision Tree
plotmatrix(y_test, dt_predict, "Decision Tree")
# Random Forest
plotmatrix(y_test, rf_predict, "Random Forest")
 #
SVM
plotmatrix(y_test, svm_predict, "Support Vector Machine")
 #
kNN
plotmatrix(y_test, knn_predict, "kNN")
 #
ANN
plotmatrix(y_test, ann_predict, "Artificial Neural Network")
```