Forest Fire Classification Using Random Forest

D214 Data Analytics Capstone

Scott Babcock WGU - MS, Data Analytics

Created: August 9 2025 *Last Edited: September 10 2025*

A, Research Question

This capstone project will focus on the research question “Can environmental conditions be used to accurately predict when a wildfire will occur?” In the Western United States, wildfires are a growing problem that has a far-reaching impact. Wildfires have a devastating impact on citizens and communities and have the potential to damage critical infrastructure and ecosystem services. The U.S. Department of the Interior reported that as of 2023, the cost of wildfires had elevated to $424 billion annually (U.S. Department of the Interior, 2023). There are unseen consequences as well. Growing evidence shows that exposure to wildfire smoke, commonly called particle pollution, is associated with increased cardiovascular and respiratory problems in humans. These effects appear to increase significantly as the smoke density increases (U.S. Environmental Protection Agency, n.d.). While flames alone threaten wildlife, particle pollution is nearly as dangerous. According to Braelei Hardt, an Evolutionary Biologist with the National Wildlife Federation, it is estimated that nearly 250 elk died due to asphyxiation or carbon monoxide poisoning during the 1988 Yellowstone National Park fires (Hardt, 2023). Hardt (2023) goes on to explain that birds are susceptible to smoke as well. Smoke inhalation from Colorado wildfires has been linked to the mass avian die-off in 2020, where an estimated one million birds perished. Taken together, these findings highlight that the impact of wildfires is multifaceted, encompassing economic loss, ecological disruption, and serious public health risks. Historically, wildfire detection has been reactive. Satellites, aircraft, and drones help spot wildfires that have already started, though they can struggle to identify smaller fires. Cameras placed in remote areas can help identify smoke and fire in real-time. Of course, relying on the human eye is also key (U.S. Government Accountability Office, 2025). Technological advancements have recently improved fire detection and modeling, enabling fires to be caught and controlled earlier. The use of artificial intelligence (AI) in fire detection is still in its initial stages. AI has helped process information faster and more efficiently, leading to decreased response times. However, the optimal way to mitigate fire damage would be to prevent wildfires from happening in the first place (Western Fire Chiefs Association, 2023). Using a dataset that contains fire incident history from the California Department of Forestry and Fire Protection (CAL FIRE) with environmental conditions from the National Oceanic and Atmospheric Administration (NOAA) from 1984 to 2025, this analysis will aim to predict when a fire could start based on weather conditions. Understanding how these variables influence ignition probability is critical for resource allocation, early warning systems, and risk mitigation. Prior studies have shown correlations between weather variables and fire frequency, but comprehensive predictive models using long-term data remain underdeveloped. The belief is that the weather features will predict when a fire ignition occurs, thus establishing the following null and alternative hypotheses:

Null hypothesis (H0): Weather features have no predictive power regarding the timing of fire ignition events.
Alternative hypothesis (H1): Weather features have predictive power regarding the timing of fire ignition events.

B, Data Collection

The data for the project was hosted on Zenodo, a general-purpose open repository built by CERN and operated by European OpenAIRE (CERN, n.d.). The dataset is licensed under Creative Commons Attribution 4.0 International, allowing redistribution and reuse if the author or authors are credited. Yavas, Kadlec, Kim, and Chen developed the California Weather and Fire Prediction Dataset. The dataset can be found at this location: (https://zenodo.org/records/14712845). The data was downloaded in CSV format with a file size of 1,482KB. The dataset contains daily weather conditions in California from 1984 to 2025, indicating whether a fire ignition occurred on that day. There are 14,988 rows and 14 columns, which include the following descriptors from the authors (Yavas, Kadlec, Kim, & Chen, 2025):

- **DATE**: The date of the observation.
- **PRECIPITATION**: Daily precipitation in inches.
- **MAX_TEMP**: Maximum daily temperature in degrees Fahrenheit.

- **MIN_TEMP**: Minimum daily temperature in degrees Fahrenheit.
- **AVG_WIND_SPEED**: Average daily wind speed in mph.
- **FIRE_START_DAY**: A binary indicator (True/False) showing whether a wildfire started on that date.
- **YEAR**: The year of the observation.
- **TEMP_RANGE**: The difference between maximum and minimum temperatures, indicating daily temperature variability.
- **WIND_TEMP_RATIO**: The ratio of average wind speed to maximum temperature, capturing wind-temperature dynamics.
- **MONTH**: The calendar month of the observation (1–12).
- **SEASON**: The season of the observation (Winter, Spring, Summer, Fall).
- **LAGGED_PRECIPITATION**: Cumulative precipitation over the preceding 7 days, reflecting recent moisture conditions.
- **LAGGED_AVG_WIND_SPEED**: Average wind speed over the preceding 7 days, indicating sustained wind patterns.
- **DAY_OF_YEAR**: The numeric day within the year (1–365/366).

The primary advantage of this data collection technique is that the authors had already retrieved the weather data for the time span and matched it up with the fire events from CAL FIRE. The data download presented a considerable time savings. A CSV download was also flexible and efficient, requiring only a read.csv call in R to load it. The read.csv function is part of the base R package, meaning anyone could load the data without additional libraries needing to be called.

The primary disadvantage of the data collection technique was that the analysis was limited to what the authors provided. While there are many good indicators included in the dataset, there are additional metrics that would have proven useful. Some could be calculated within R, such as additional lag metrics, but others would require supplementation with additional datasets. A question that initially came to mind was what part of California the weather metrics were from – were they from a specific park, averages from all of California, and so on.

Some of these data challenges were addressed by creating additional metrics within R. The dataset contained seven-day lags for total precipitation and average wind speed, meaning it calculated the metrics for seven days leading up to a given date. However, there were no metrics for lagging temperatures. Seven-day lags were created for average maximum and minimum temperatures and average temperature range to assess whether the areas were experiencing consistent high temperatures or low temperature variability leading up to fire ignition. While day-of temperature is important, consistent high temperatures leading up to a fire event dry out the land and could play a critical role in ignition. The dataset also had no variables that would account for a human element in fire ignition. According to Camper Champ (2024), a camper rental company, nearly 88 million Americans went camping in 2023, and the camping participation rate has been rising yearly. This means a significant amount of traffic goes in and out of forests. Since holidays are a popular time to plan trips, a binary true/false field was created to indicate whether a given date was within plus or minus six days of a holiday. The assumption for this metric was that the days surrounding a holiday would likely drive more traffic into forests, potentially increasing the risk of fire ignition caused by humans.

C, Data Extraction and Preparation

Programming Language and Environment

The R programming language was chosen for this analysis, and R Studio was the environment that enabled the work. R was designed specifically for statistical computing and analysis, and is popular among statisticians, data scientists, and researchers. There are an extensive number of packages that allow complex analysis to be performed with minimal coding. Potential downsides of using R include the learning curve and complexity of the language, as well as performance and memory constraints. Some view the R syntax and data structures as more complex than those of other languages, such as Python, which can be intimidating

for inexperienced users. R can be slower than other languages and requires substantial memory, so code optimization is key (GeeksforGeeks, 2025b).

The analysis and accompanying paper use a file type called R Markdown, which allows for a combination of text, code execution, and results display. Therefore, the data extraction and preparation steps are all provided below. An advantage to this method is the seamless way it blends the report text and analysis. There is no need to copy and paste pictures of code, graphs, or charts like a word processor would require. A disadvantage is that R Markdown requires more knowledge beyond the R programming language. There is a specific way the file needs to be set up in order for it to perform the work and display correctly.

Packages Used in Analysis

The following packages are used in the analysis. There are multiple ways to load packages when using R, the most common being the library function. The downside to the library method is that if a package is not installed, the end user would receive an error. This is not ideal for sharing scripts. The method below is two-fold and will load all packages and install them, if necessary, without user intervention. First, the if function will look for the pacman package, install it if needed, and load it. Next, the p_load function from pacman will load or install all other packages with minimal coding.

```
# install/load required packages
if (!require("pacman", character.only = TRUE)) {
  install.packages("pacman", dependencies = TRUE)
  library("pacman", character.only = TRUE)
}

p_load(
  tidyverse,      # data wrangling & plotting
  patchwork,      # plotting manipulation
  gridExtra,      # plotting manipulation
  scales,         # axis value manipulation
  lubridate,      # date handling
  timeDate,       # lookup holidays
  zoo,            # date lags
  janitor,        # clean names
  skimr,          # quick data summaries
  ranger,         # fast random-forest
  doParallel,     # model training efficiency
  caret,          # resampling & metrics
  pROC,           # ROC curves & AUC
  precrec,        # plotting ROC/precision-recall
  naniar          # identifying, visualizing, and handling missing data
)
```

Data Load

The wildfires dataset is loaded with the read_csv function, part of the base R package. The base R package comes as part of the R installation, so any R user can access the functions that are part of it. Following the read_csv call, the clean_names function from the janitor package is piped in to ensure that all column names are unique. Piping, represented by %>%, is a way to chain functions together and is enabled through the dplyr package.

```r
# load dataset
df_raw <- read.csv("Data/CA_Weather_Fire_Dataset_1984-2025.csv") %>%
  clean_names()
```

Data Structure

With the data loaded, the skim function from the skimr package is used on the data frame. This function thoroughly summarizes the number of rows and columns, types of data classes, and column-specific information. There are 14,988 rows and 14 columns in the data frame. Three of the columns are categorical variables, and 11 are numeric. The categorical variable summary shows that the three fields are: date, fire_start_day, and season. These columns are fully populated. The date field has 14,988 unique values, corresponding to the total number of rows in the dataset, meaning there are no duplicate dates. While date was treated as categorical in the data load, it was transformed into a date/time object during data cleaning for more flexibility. The fire_start_day field was a nominal binary categorical variable, which meant it had no inherent order and contained two unique values. The season field is an ordinal categorical variable that has four unique values. Season is considered ordinal because the four seasons have a natural cyclical order. The numeric variable summary showed that the 11 numeric fields were: precipitation, max_temp, min_temp, avg_wind_speed, year, temp_range, wind_temp_ratio, month, lagged_precipitation, lagged_avg_wind_speed, and day_of_year. Precipitation is continuous/ratio because it can be a fractional value with a true zero. Maximum and minimum temperatures are continuous/interval since they can be fractional values and do not have true zeros. Average wind speed was continuous/ratio since it could be a fractional value with a true zero. The year variable was discrete/interval because years are whole numbers and have meaningful differences, but cannot be looked at in terms of ratios. Temperature ranges are continuous/interval since they can be fractional values and do not have a true zero. The month was being stored as numeric, which made it discrete, but in reality, it was categorical and ordinal. The two lag fields, lagged_precipitation and lagged_avg_wind_speed, were both continuous/ratio, like their non-lagged counterparts. They can both be expressed as fractional values and have true zeros. The day of the year is a discrete/interval because it can only be a whole number and does not have a true zero.

```r
# check structure of data and view summaries
skim(df_raw)
```

Table 1: Data summary

Name	df_raw
Number of rows	14988
Number of columns	14
Column type frequency:	
character	3
numeric	11
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
date	0	1	10	10	0	14988	0
fire_start_day	0	1	4	5	0	2	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
season	0	1	4	6	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
precipitation	1	1	0.03	0.18	0.00	0.00	0.00	0.00	4.53	□□□□□
max_temp	1	1	70.53	7.26	50.00	65.00	70.00	75.00	106.00	□□□□□
min_temp	1	1	56.49	6.77	33.00	51.00	57.00	62.00	77.00	□□□□□
avg_wind_speed	12	1	7.44	2.13	1.79	6.04	7.16	8.50	26.17	□□□□□
year	0	1	2004.02	11.84	1984.00	1994.00	2004.00	2014.00	2025.00	□□□□□
temp_range	1	1	14.04	6.00	2.00	10.00	12.00	17.00	41.00	□□□□□
wind_temp_ratio	12	1	0.11	0.04	0.02	0.09	0.10	0.12	0.46	□□□□□
month	0	1	6.52	3.45	1.00	4.00	7.00	10.00	12.00	□□□□□
lagged_precipitation	0	1	0.23	0.65	0.00	0.00	0.00	0.06	8.18	□□□□□
lagged_avg_wind_speed	0	1	7.43	1.39	3.23	6.52	7.48	8.28	13.93	□□□□□
day_of_year	0	1	182.99	105.52	1.00	92.00	183.00	274.00	366.00	□□□□□

```
# examine first 10 records of dataset
df_raw %>% head(10)
```

```
##          date precipitation max_temp min_temp avg_wind_speed fire_start_day
## 1  1984-01-01             0       79       51           4.70          False
## 2  1984-01-02             0       71       46           5.59          False
## 3  1984-01-03             0       70       47           5.37          False
## 4  1984-01-04             0       76       45           4.70          False
## 5  1984-01-05             0       74       49           5.14          False
## 6  1984-01-06             0       65       49           7.38          False
## 7  1984-01-07             0       59       54           5.82          False
## 8  1984-01-08             0       59       55           3.36          False
## 9  1984-01-09             0       61       54           6.71          False
## 10 1984-01-10             0       70       47           4.70          False
##    year temp_range wind_temp_ratio month season lagged_precipitation
## 1  1984         28      0.05949367     1 Winter                    0
## 2  1984         25      0.07873239     1 Winter                    0
## 3  1984         23      0.07671429     1 Winter                    0
## 4  1984         31      0.06184211     1 Winter                    0
## 5  1984         25      0.06945946     1 Winter                    0
## 6  1984         16      0.11353846     1 Winter                    0
## 7  1984          5      0.09864407     1 Winter                    0
## 8  1984          4      0.05694915     1 Winter                    0
## 9  1984          7      0.11000000     1 Winter                    0
## 10 1984         23      0.06714286     1 Winter                    0
##    lagged_avg_wind_speed day_of_year
## 1               4.700000           1
## 2               5.145000           2
## 3               5.220000           3
## 4               5.090000           4
## 5               5.100000           5
## 6               5.480000           6
```

```
## 7               5.528571           7
## 8               5.337143           8
## 9               5.497143           9
## 10              5.401429          10
```

```
#examine last 10 records of dataset
df_raw %>% tail(10)
```

```
##             date precipitation max_temp min_temp avg_wind_speed fire_start_day
## 14979 2025-01-03             0       61       49           5.14          False
## 14980 2025-01-04             0       62       48           4.92          False
## 14981 2025-01-05             0       75       45           4.03          False
## 14982 2025-01-06             0       71       47           2.91          False
## 14983 2025-01-07             0       73       49          13.42          False
## 14984 2025-01-08             0       73       53          10.51          False
## 14985 2025-01-09             0       68       46           4.92          False
## 14986 2025-01-10             0       70       46           3.58          False
## 14987 2025-01-11             0       66       46             NA          False
## 14988 2025-01-12             0       69       46             NA          False
##       year temp_range wind_temp_ratio month season lagged_precipitation
## 14979 2025         12      0.08426230     1 Winter                    0
## 14980 2025         14      0.07935484     1 Winter                    0
## 14981 2025         30      0.05373333     1 Winter                    0
## 14982 2025         24      0.04098592     1 Winter                    0
## 14983 2025         24      0.18383562     1 Winter                    0
## 14984 2025         20      0.14397260     1 Winter                    0
## 14985 2025         22      0.07235294     1 Winter                    0
## 14986 2025         24      0.05114286     1 Winter                    0
## 14987 2025         20              NA     1 Winter                    0
## 14988 2025         23              NA     1 Winter                    0
##       lagged_avg_wind_speed day_of_year
## 14979              4.805000           3
## 14980              4.843333           4
## 14981              4.640000           5
## 14982              4.294000           6
## 14983              5.815000           7
## 14984              6.485714           8
## 14985              6.550000           9
## 14986              6.327143          10
## 14987              6.561667          11
## 14988              7.068000          12
```

Missing Values

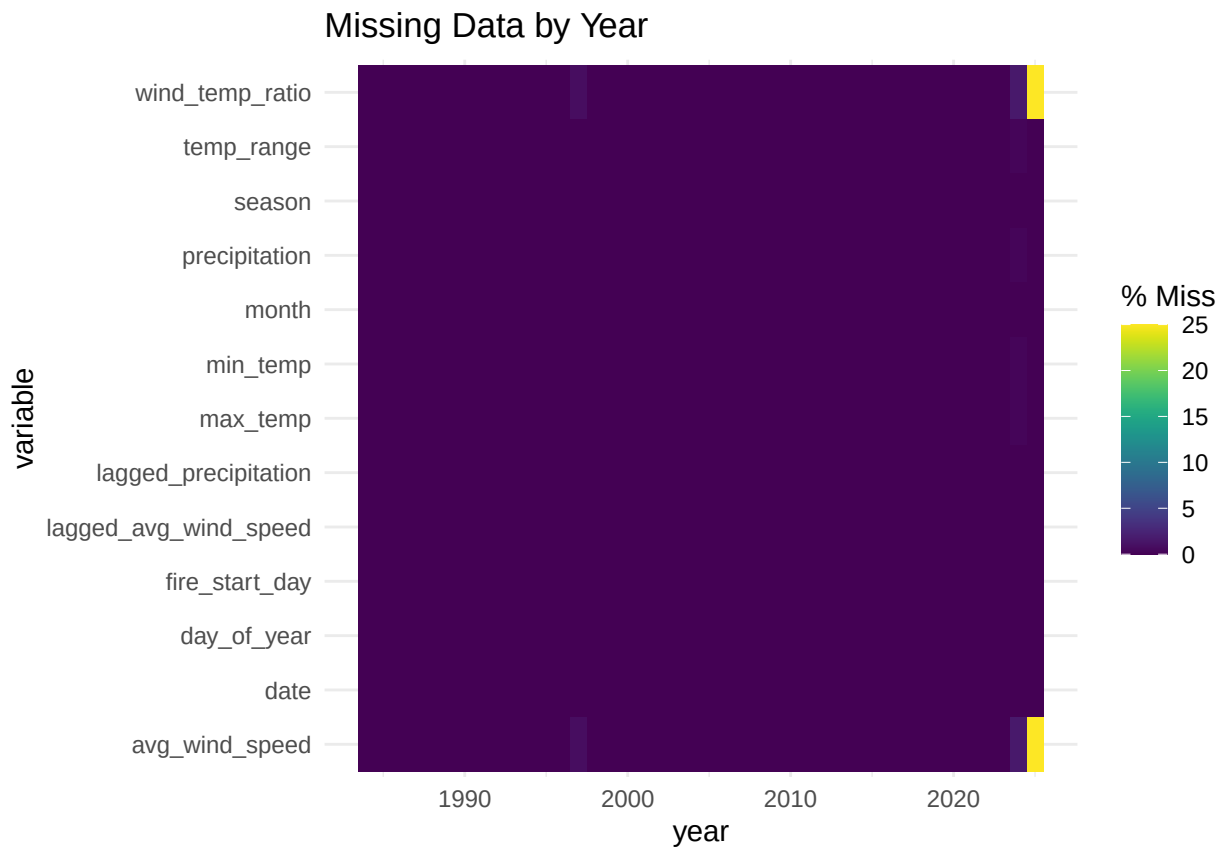
After assessing the structure of the data, it was checked for missing values. The miss_var_summary function from the naniar package provided a simple summary table of missing values. It showed the variable, the number of rows with missing values, and the percentage of the dataset that the missing values comprised. The avg_wind_speed and wind_temp_ratio fields each had 12 missing values, representing 0.08% of the dataset. The precipitation, max_temp, min_temp, and temp_range fields all had one missing value, representing 0.006% of the dataset. The presence of missing values provided the first decision point for the analysis. The rows with missing values could be dropped, and given that they represented a small fraction of the total dataset, it would have no impact. However, this would result in a date gap, hindering future

time series analysis. The missing values could also be imputed with a mean or median to keep the dataset intact. The downside of this approach is that the actual values could differ from the imputed values, potentially skewing results. Again, given that there are so few missing values, imputation would have a negligible impact. Prior to making a decision, further exploration was warranted. The gg_miss_fct function from the naniar package was used to visualize the relationship between the missing values and a specified variable from the dataset. The year variable was chosen to see if there is a pattern across time. The resulting plot showed that the missing values primarily occurred in the most recent year (2025), indicated by the yellow coloring. A subsequent count of records by year from 2020 to 2025 showed that there were only 12 records in 2025, which corresponded to the number of missing values in two of the columns. It would be safe to drop all records from 2025 and have the analysis timeframe range from 1984 to 2024. The remaining missing values occurred in 1997 and 2024. The missing values were interpolated using the zoo package's 'na.approx' function. This method estimated the values based on surrounding data points using linear interpolation. This choice was ideal for ordered data because it would preserve trends. A new data frame called df_clean was created to house the clean data. Once the values had been imputed, miss_var_summary was called again to ensure no missing values remained.

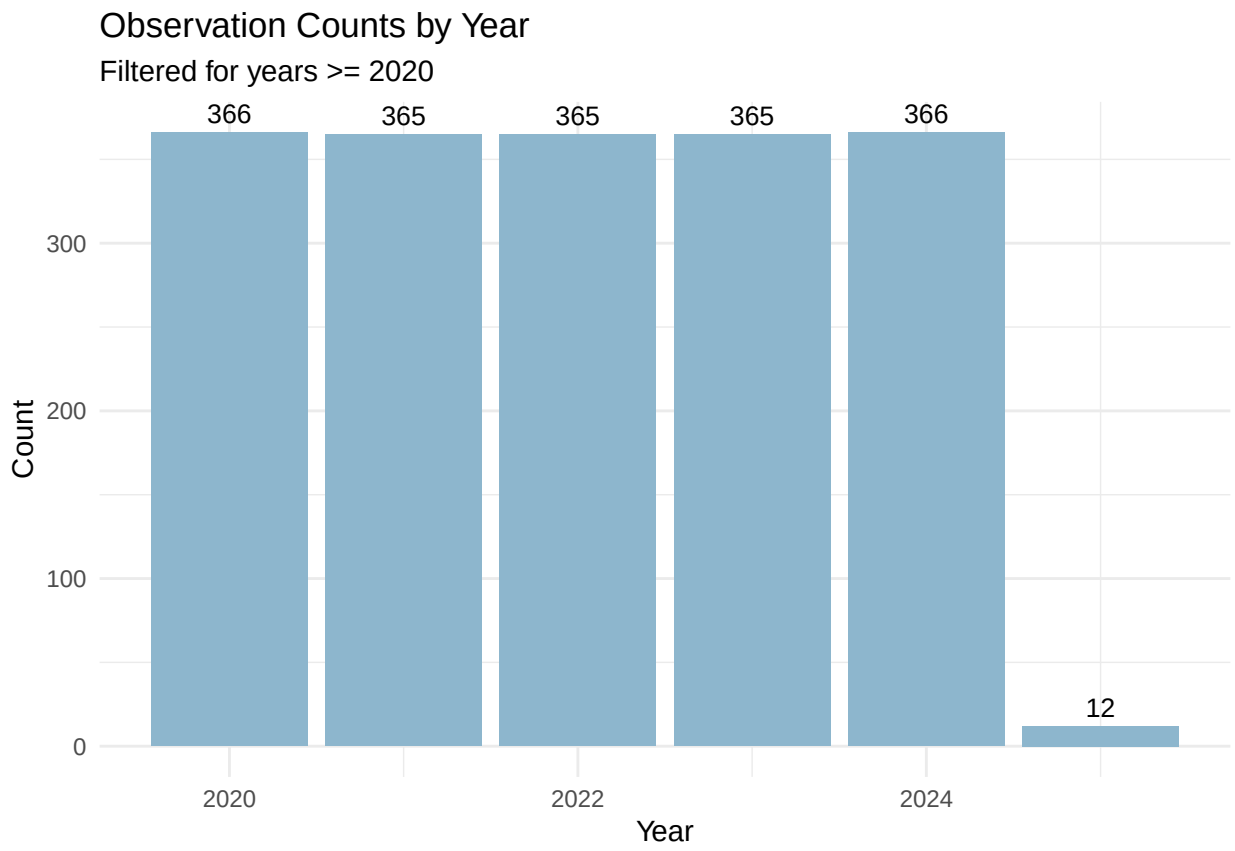
```
# check for missing values
miss_var_summary(df_raw)
```

```
## # A tibble: 14 x 3
##    variable              n_miss pct_miss
##    <chr>                  <int>    <num>
##  1 avg_wind_speed            12  0.0801
##  2 wind_temp_ratio           12  0.0801
##  3 precipitation              1  0.00667
##  4 max_temp                   1  0.00667
##  5 min_temp                   1  0.00667
##  6 temp_range                 1  0.00667
##  7 date                       0  0
##  8 fire_start_day             0  0
##  9 year                       0  0
## 10 month                      0  0
## 11 season                     0  0
## 12 lagged_precipitation       0  0
## 13 lagged_avg_wind_speed      0  0
## 14 day_of_year                0  0
```

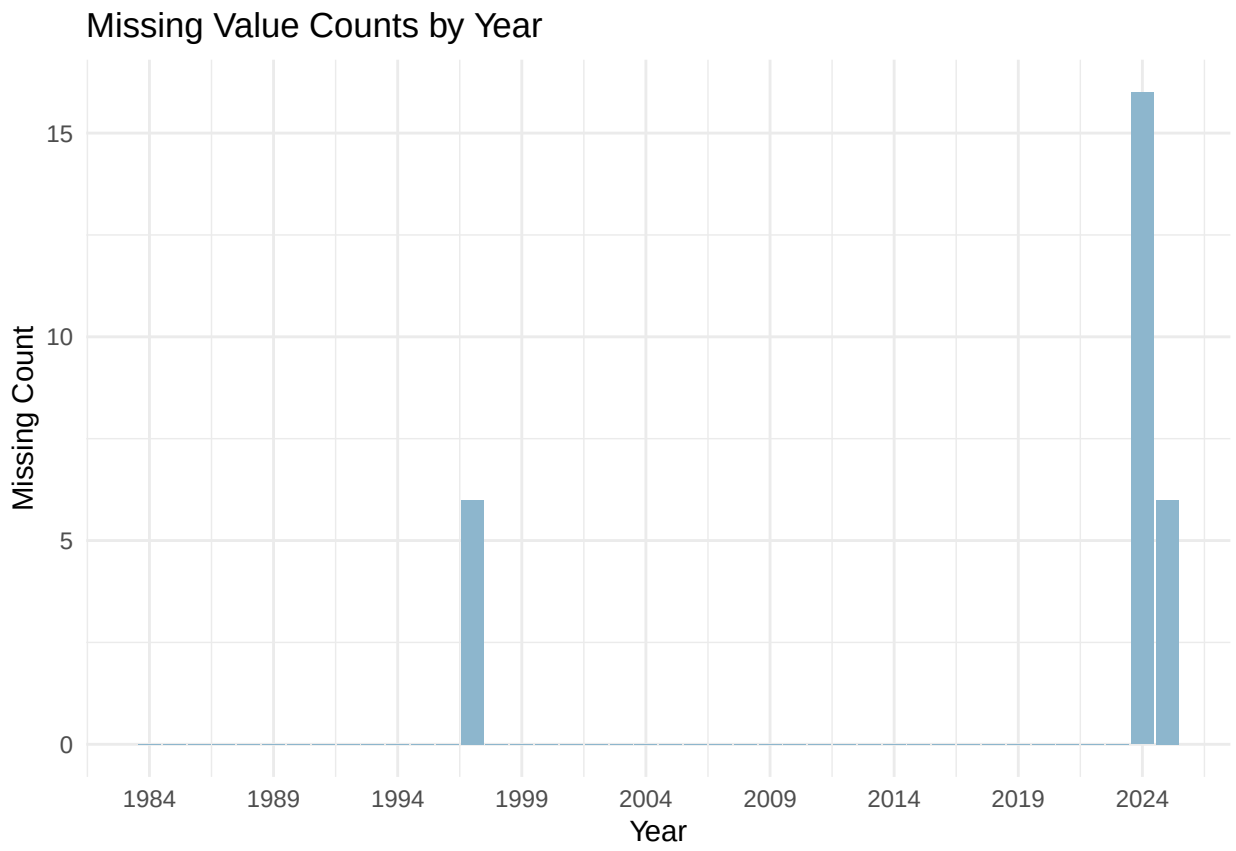
```
# visualize missing data by year
gg_miss_fct(df_raw, fct = year)+
  labs(title = "Missing Data by Year") +
  theme_minimal()
```




```r
# count number of records in each year (2020+)
df_raw %>%
  filter(year >= 2020) %>%
  group_by(year) %>%
  count() %>%
  ggplot(aes(x = year, y = n))+
  geom_col(fill = "lightskyblue3")+
  geom_text(aes(label = n), vjust = -0.5, size = 3.5) +
  theme_minimal()+
  labs(
    title = "Observation Counts by Year",
    subtitle = "Filtered for years >= 2020",
    x = "Year",
    y = "Count"
  )
```




```r
# count missing values by year
df_raw %>%
  group_by(year) %>%
  summarize(missing = sum(is.na(across(everything())))) %>%
  ggplot(aes(x = year, y = missing))+
  geom_col(fill = "lightskyblue3")+
  scale_x_continuous(breaks = seq(1984,2024,5))+
  theme_minimal()+
  labs(
    title = "Missing Value Counts by Year",
    x = "Year",
    y = "Missing Count"
  )
```




```
# filter out records from 2025
df_clean <- df_raw %>%
  filter(year != 2025)
# account for remaining missing values - median imputation
df_clean <- df_clean %>%
  mutate(across(where(is.numeric), ~ na.approx(., rule = 2)))
# check for missing values following imputation
miss_var_summary(df_clean)
```

```
## # A tibble: 14 x 3
##    variable              n_miss pct_miss
##    <chr>                  <int>    <num>
##  1 date                       0        0
##  2 precipitation              0        0
##  3 max_temp                   0        0
##  4 min_temp                   0        0
##  5 avg_wind_speed             0        0
##  6 fire_start_day             0        0
##  7 year                       0        0
##  8 temp_range                 0        0
##  9 wind_temp_ratio            0        0
## 10 month                      0        0
## 11 season                     0        0
## 12 lagged_precipitation       0        0
## 13 lagged_avg_wind_speed      0        0
## 14 day_of_year                0        0
```

Duplication

The data was next checked for duplicate rows. The duplicated function is part of the base R package and checks each row against all other rows in the dataset. It returned a binary true/false field indicating whether a row is a duplicate of any other row. Because true/false represents one and zero, wrapping it in a sum function would total the number of duplicate rows. There were no duplicate rows present in the dataset.

```
# check for duplicate rows
sum(duplicated(df_raw))
```

```
## [1] 0
```

Data Cleaning and Manipulation

The dataset contained three categorical columns of character type: date, fire_start_day, and season. The date field was converted to a date-time object. The season field was converted to a factor with four levels. Factors store categorical values as numbers, making them more memory-efficient than character strings, particularly when many values exist. They also allow for improved performance in modeling, ensuring that modeling functions treat the data correctly (Spector, n.d.). For clarity purposes, a new field was created for fire_start_day called fire_occurred, which was a factor with two levels: No_fire and Fire, represented by 0 and 1, respectively. The year and month fields were numeric integers. To ensure they were treated properly, the year and month functions from the lubridate package were used on the date field to retrieve the year and month components. The month field was changed to an ordered factor with month names as the 12 levels, represented by 1-12. After manipulating these fields, the classes were again checked to ensure the changes occurred.

```
# check class of fields
class(df_clean$date)
```

```
## [1] "character"
```

```
class(df_clean$year)
```

```
## [1] "numeric"
```

```
class(df_clean$month)
```

```
## [1] "numeric"
```

```
class(df_clean$season)
```

```
## [1] "character"
```

```
# change data types and create new target variable field
df_clean <- df_clean %>%
  mutate(date = as_date(date),
         year = year(date),
         month = month(date, label = TRUE),
         season = factor(season, levels = c("Spring", "Summer", "Fall", "Winter"), ordered = TRUE),
```

```
        fire_occurred = ifelse(fire_start_day=='True',1,0)) %>%
  select(-fire_start_day)

# change target variable into factor
df_clean$fire_occurred <- factor(df_clean$fire_occurred,
                           levels = c(1,0),
                           labels = c("Fire","No_Fire"))

#re-check class of fields
class(df_clean$date)
```

```
## [1] "Date"
```

```
class(df_clean$year)
```

```
## [1] "numeric"
```

```
class(df_clean$month)
```

```
## [1] "ordered" "factor"
```

```
class(df_clean$season)
```

```
## [1] "ordered" "factor"
```

```
class(df_clean$fire_occurred)
```

```
## [1] "factor"
```

Data Enrichment

One goal of data enrichment was to account for a human element in fire events. To accomplish this, the holidayNYSE function from the timeDate package was used to pull the holiday dates for the unique years in the data frame. A binary true/false column called is_holiday was created in the data frame where a date matches a holiday date. Identifying a holiday in and of itself does not accomplish the goal, as camping trips typically span six nights on average (CamperChamp, 2024). A six-day buffer around is_holiday would help capture the average camping trip length.

```
# get holiday dates and flag in data
yrs <- unique(df_clean$year)
us_holidays <- as.Date(holidayNYSE(yrs))
df_clean$is_holiday <- df_clean$date %in% us_holidays
# create 6-day buffer around holiday
df_clean <- df_clean %>%
  mutate(near_holiday = map_lgl(date, ~ any(abs(.x - date[is_holiday]) <= days(6))))
```

The dataset contained seven-day lags for precipitation and wind speed, but none for temperature metrics. Three additional lags were created to calculate the average maximum temperature, average minimum temperature, and average daily temperature range for the seven preceding days for every given day. These are

important metrics to consider because consistently elevated temperatures or consistently high values for daily low affect fire ignition and combustion processes (Smith, 2022). The lag function from dplyr was used in conjunction with rollapply from the zoo package to achieve this. The lags at the start of the dataset in 1984 returned NA values because there were not a full seven days for the calculation. Using linear interpolation, the 'na.approx' function was again used to estimate the missing values based on surrounding data points.

```
# create additional lag variables
df_clean <- df_clean %>%
  arrange(date) %>%
  mutate(lagged_avg_max_temp = round(lag(rollapply(max_temp, 7, mean, align = "right", fill = NA)),2),
         lagged_avg_min_temp = round(lag(rollapply(min_temp, 7, mean, align = "right", fill = NA)),2),
         lagged_avg_temp_range = round(lag(rollapply(temp_range, 7, mean, align = "right", fill = NA)),
         )
# impute missing lag values
df_clean <- df_clean %>%
  mutate(
    lagged_avg_max_temp = na.approx(lagged_avg_max_temp, rule = 2),
    lagged_avg_min_temp = na.approx(lagged_avg_min_temp, rule = 2),
    lagged_avg_temp_range = na.approx(lagged_avg_temp_range, rule = 2)
  )
```

Exploratory Data Analysis

With the data cleaned, exploratory analysis could begin. The target variable, fire_occurred, was checked for class imbalance and to see how fire occurrences varied across different metrics. Univariate analysis would then be performed on the predictor variables to understand the summary statistics and distributions. Finally, bivariate analysis would be performed with the target variable across the predictor variables. These types of analyses would help identify potential influential variables for modeling.

Class Imbalance

The target variable, what the analysis aims to predict, contained two classes: fire and no fire. The data was checked to see how many days fall under each categorical value using the table function from the base R package. There were 10,005 records with no fires and 4,971 records with fire occurrences. The 'prop.table' function was called to get the proportion of values, which is also from the base R package. Records with no fires comprised 66% of the dataset versus 33% of records containing fire occurrences. The no-fire class clearly outnumbered the fire class, referred to as a class imbalance. Class imbalances pose unique challenges for machine learning models. They often show bias towards the majority class, which would be problematic if the goal were to predict the minority class. Specific metrics, such as accuracy, could be misleading, as they could just show that the model excels at predicting the majority class. Because the minority class had fewer samples, models could not pick up patterns and learn as easily (Singh, 2024). Class imbalance can be mitigated in a variety of ways. When random sampling, two of the most common ways are oversampling and undersampling. With oversampling, the size of the minority class would be increased by duplicating sample records or generating synthetic data samples. An advantage of this would be that it increases the size of the minority class and retains the size of the majority class. However, oversampling can lead to overfitting if the duplication method is used because the model could learn patterns that are too specific to the repeated records. Undersampling, on the other hand, would be the opposite. The number of samples from the majority class would be reduced to align with the minority class, which could reduce training time and simplify the model. The downsides to undersampling would be the loss of data from the majority class and a potential reduction in accuracy (Singh, 2024). Another potential mitigation strategy would be to change the model training metric to sensitivity instead of the default accuracy metric. Sensitivity is the rate of true positives, which in the analysis would represent the minority class. By changing the value to sensitivity,

the model would optimize for the true positive rate, which would help negate some class imbalance issues. Now that it was clear there was a class imbalance in the data, a strategy would be developed for modeling.

```
# check for class imbalance
(imbalance <- table(df_clean$fire_occurred))
```

```
##
##    Fire No_Fire
##    4971   10005
```

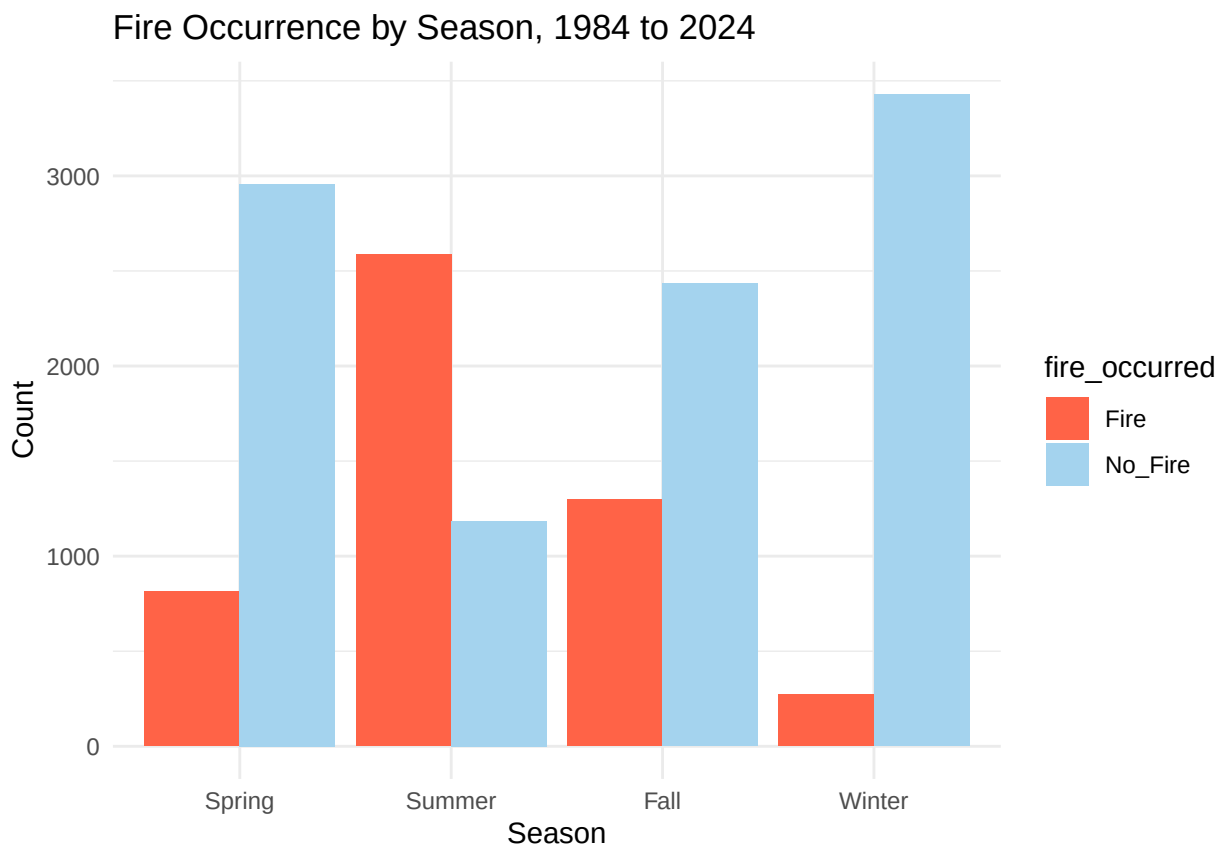
```
# create proportion table
prop.table(imbalance)
```

```
##
##      Fire   No_Fire
## 0.3319311 0.6680689
```

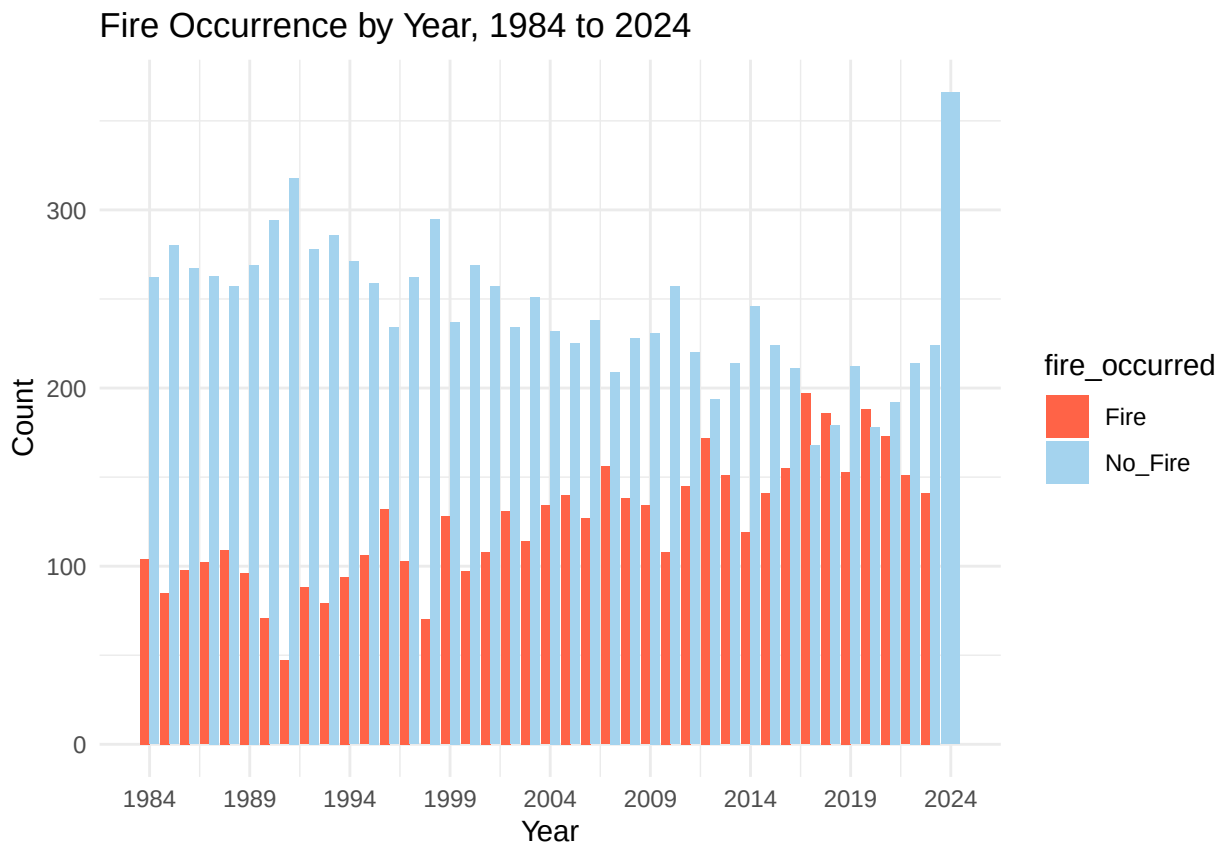
Target Variable Over Time

The target variable would be examined across various time-related variables to identify trends. The charts would be created using ggplot2, a very flexible visualization package. It enables users to create several different chart types with consistent syntax.

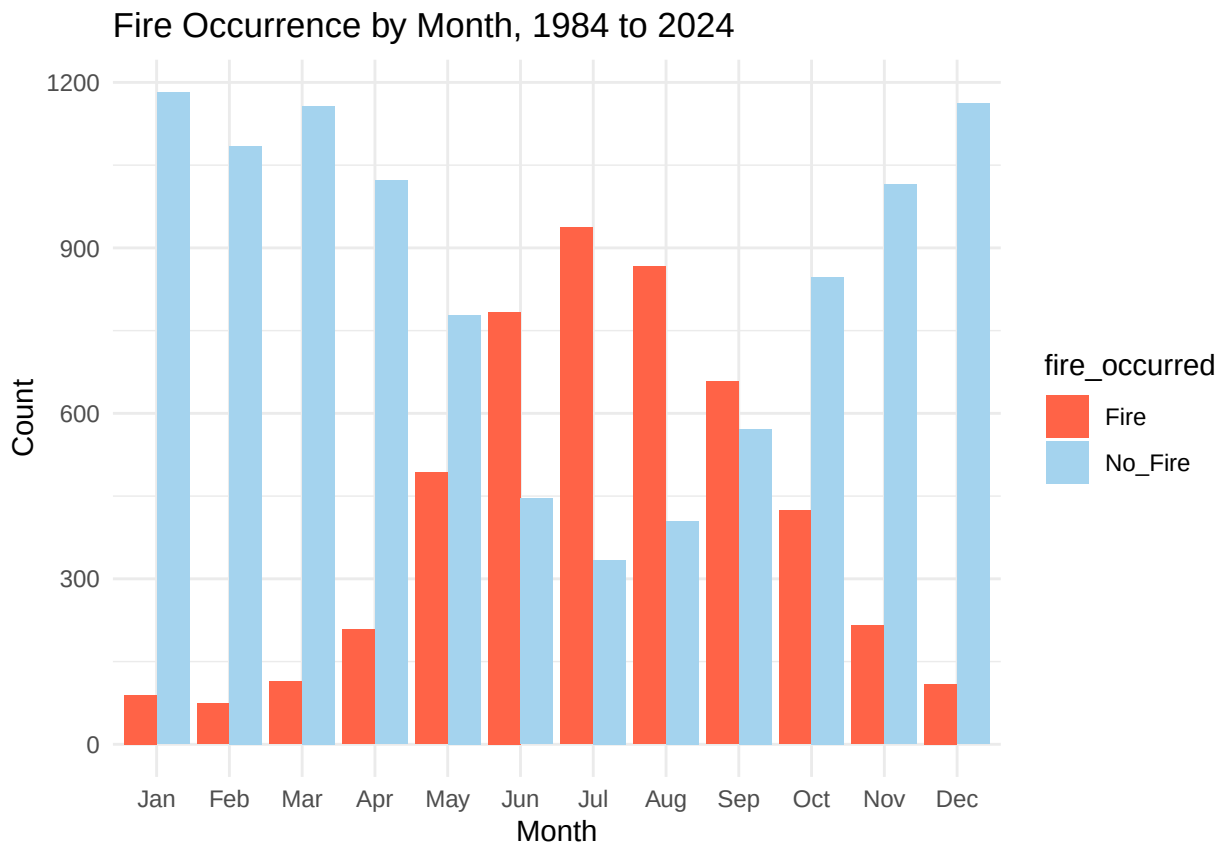
```
# view fire occurrences by season
df_clean %>%
  count(season, fire_occurred) %>%
  ggplot(aes(x = season, y = n, fill = fire_occurred)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
  labs(title = "Fire Occurrence by Season, 1984 to 2024", y = "Count", x = "Season") +
  theme_minimal()
```



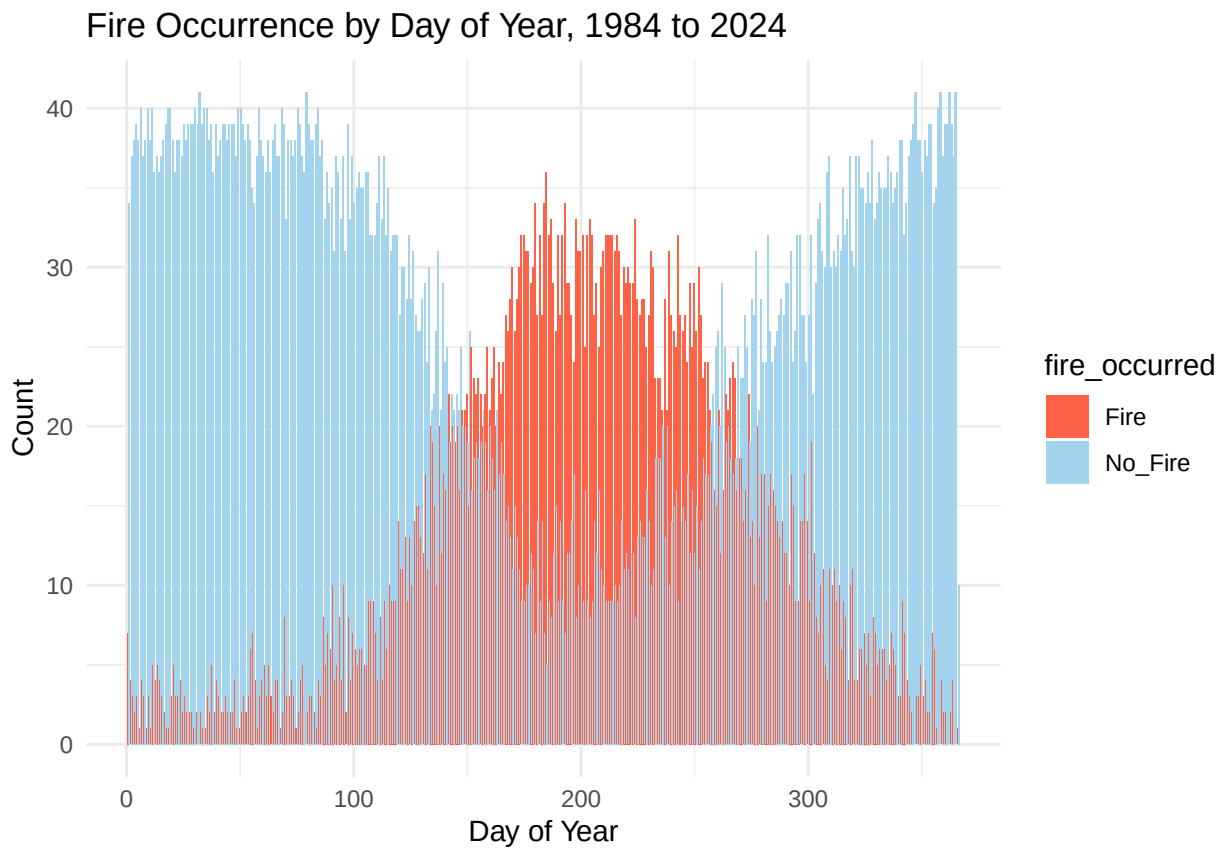

```r
# view fire occurrences by year
df_clean %>%
  count(year, fire_occurred) %>%
  ggplot(aes(x = year, y = n, fill = fire_occurred)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_x_continuous(breaks = seq(1984,2024, by = 5)) +
  scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
  labs(title = "Fire Occurrence by Year, 1984 to 2024", y = "Count", x = "Year") +
  theme_minimal()
```



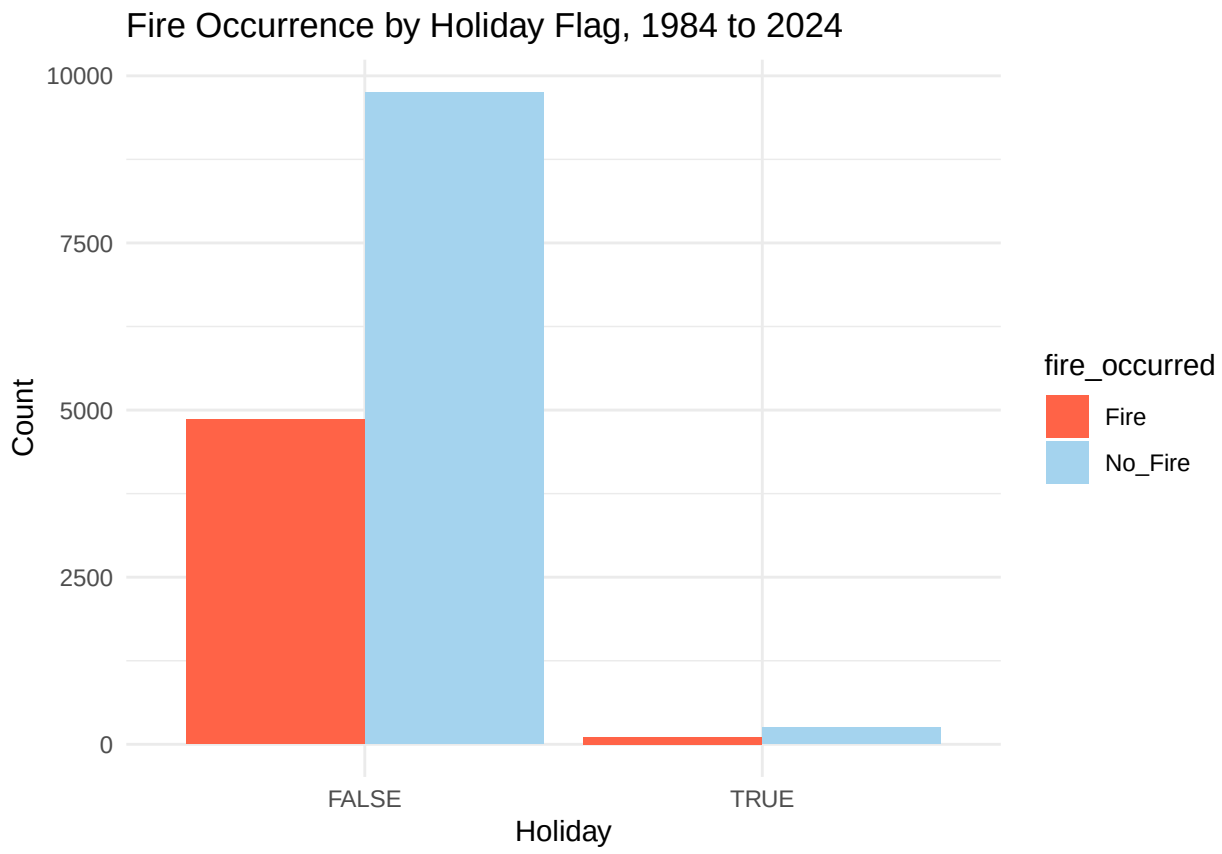

```
# view fire occurrences by month
df_clean %>%
  count(month, fire_occurred) %>%
  ggplot(aes(x = month, y = n, fill = fire_occurred)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
  labs(title = "Fire Occurrence by Month, 1984 to 2024", y = "Count", x = "Month") +
  theme_minimal()
```



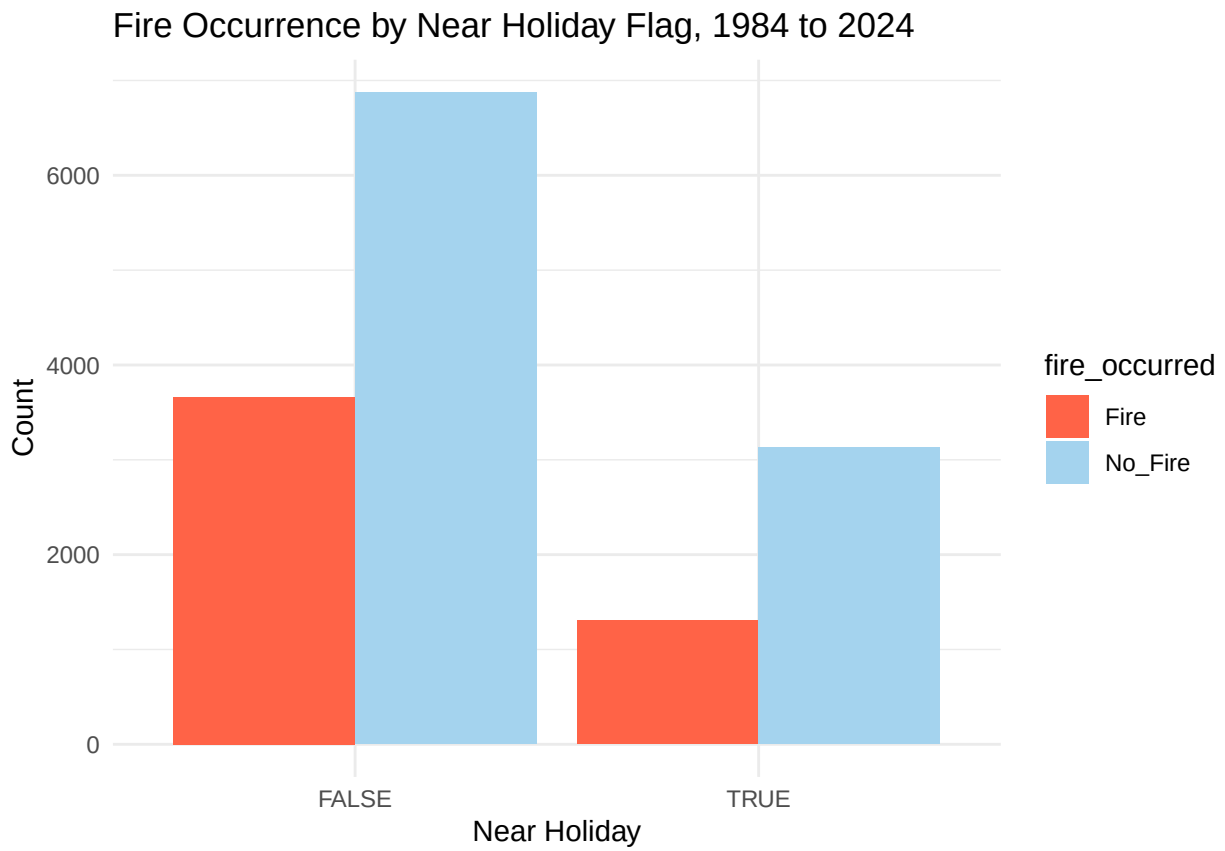

```
# view fire occurrences by day of year
df_clean %>%
  count(day_of_year, fire_occurred) %>%
  ggplot(aes(x = day_of_year, y = n, fill = fire_occurred)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
  labs(title = "Fire Occurrence by Day of Year, 1984 to 2024", y = "Count", x = "Day of Year") +
  theme_minimal()
```




```r
# view fire occurrences on holidays
df_clean %>%
  count(is_holiday, fire_occurred) %>%
  ggplot(aes(x = is_holiday, y = n, fill = fire_occurred)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
  labs(title = "Fire Occurrence by Holiday Flag, 1984 to 2024", y = "Count", x = "Holiday") +
  theme_minimal()
```



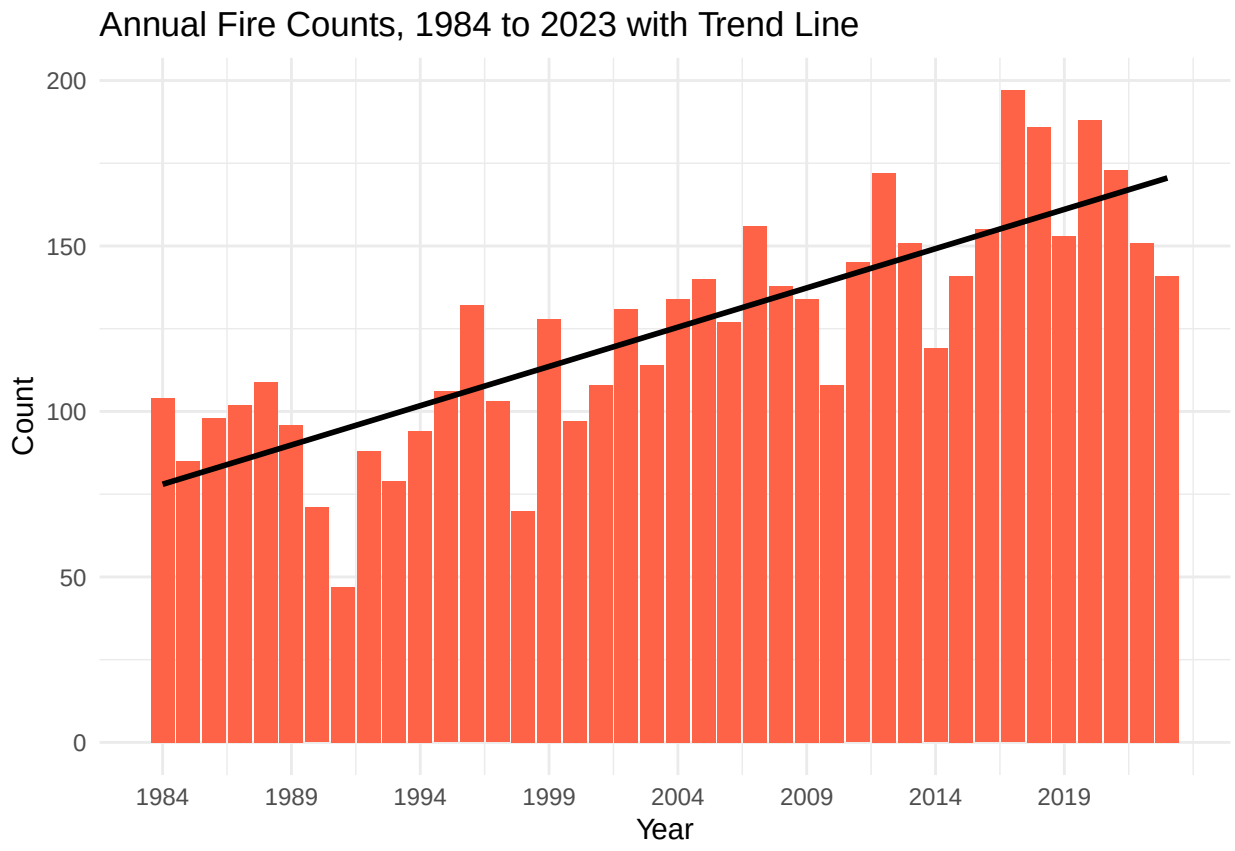

```
# view fire occurrences on near holidays
df_clean %>%
  count(near_holiday, fire_occurred) %>%
  ggplot(aes(x = near_holiday, y = n, fill = fire_occurred)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
  labs(title = "Fire Occurrence by Near Holiday Flag, 1984 to 2024", y = "Count", x = "Near Holiday") +
  theme_minimal()
```



In the first chart, fire occurrence by season, there were significantly more fires in Summer compared to other seasons, with Fall having the second most. Few fires seem to have occurred in Winter and Spring. The second chart, which looked at fire occurrence by year, showed that fire numbers had increased over the years. Given rising temperatures in recent years, an upward trend would be anticipated. An unexpected finding from the chart was zero fire occurrences in 2024, which signaled a problem with the dataset. This issue did not appear in previous cleaning steps because the column was populated, but did not have accurate information. The entire year would need to be dropped to avoid skewing modeling results. When looking at fire occurrences by month, June through August were the peak fire months, with July having the most. There were rarely fires in November through April. Drilling down even further, when looking at fire occurrence by day of year, there were spikes between days 150 and 250. There were no clear patterns when looking at fires across holidays. Fires happened on holidays, but no signal indicated they were more common. The same went for the near_holiday variable, which looked at a 6-day buffer before and after holidays. The fire occurrences near holidays looked proportionally aligned with those not near holidays.

```
# filter out 2024 from the dataset due to no fire data
df_clean <- df_clean %>%
  filter(year != 2024)
# filter for fire occurrence and view over years
df_clean %>%
  filter(fire_occurred == "Fire") %>%
  count(year, fire_occurred) %>%

  ggplot(aes(x = year, y = n)) +
  geom_col(fill = "tomato1") +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  scale_x_continuous(breaks = seq(1984,2023, by = 5)) +
  labs(title = "Annual Fire Counts, 1984 to 2023 with Trend Line", y = "Count", x = "Year") +
```

```
  theme_minimal()
```




```
# determine if number of fires pre/post 2004 is significant
df_clean$period <- ifelse(df_clean$year >= 2004,"2004 and After","Before 2004")
df_clean %>%
    filter(fire_occurred == "Fire") %>%
    count(period)
```

```
##           period    n
## 1 2004 and After 3009
## 2    Before 2004 1962
```

```
# test for significance
t.test(as.numeric(fire_occurred) ~ period, data = df_clean)
```

```
##
##  Welch Two Sample t-test
##
## data:  as.numeric(fire_occurred) by period
## t = -18.494, df = 14451, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 2004 and After and group Before 2004 i
## 95 percent confidence interval:
##  -0.1585173 -0.1281357
## sample estimates:
```

```
## mean in group 2004 and After    mean in group Before 2004
##                      1.588090                      1.731417
```

After filtering out the records from 2024, another chart of fire occurrences by year was created, this time isolating the fire occurrences. It was much easier to see that fire counts had increased significantly over the years. There also seemed to be less variance in the year-to-year fire count beginning in the middle of the graph, around 2004. To test this theory, a new variable period was created to segment the data before and after 2004. This segment resulted in 19 years' worth of data pre-2004, and 20 years' worth post-2004. The count function was used to sum up the fire occurrences, broken down by the new period variable. Pre-2004, there were 1,962 fire occurrences, while post-2004, the number had grown to 3,009. A t-test was then used to determine if there was a significant difference in the means between the two groups. The null hypothesis (H0) was that the mean fire occurrence was the same before and after 2004, while the alternative hypothesis (H1) was a significant difference in the mean fire occurrence between the two periods. The null hypothesis could be rejected if the p-value is < 0.05. The resulting p-value was 0.0000, which meant that there was strong statistical evidence that mean fire occurrence increased after 2004; thus, the null hypothesis was rejected.

Univariate Analysis

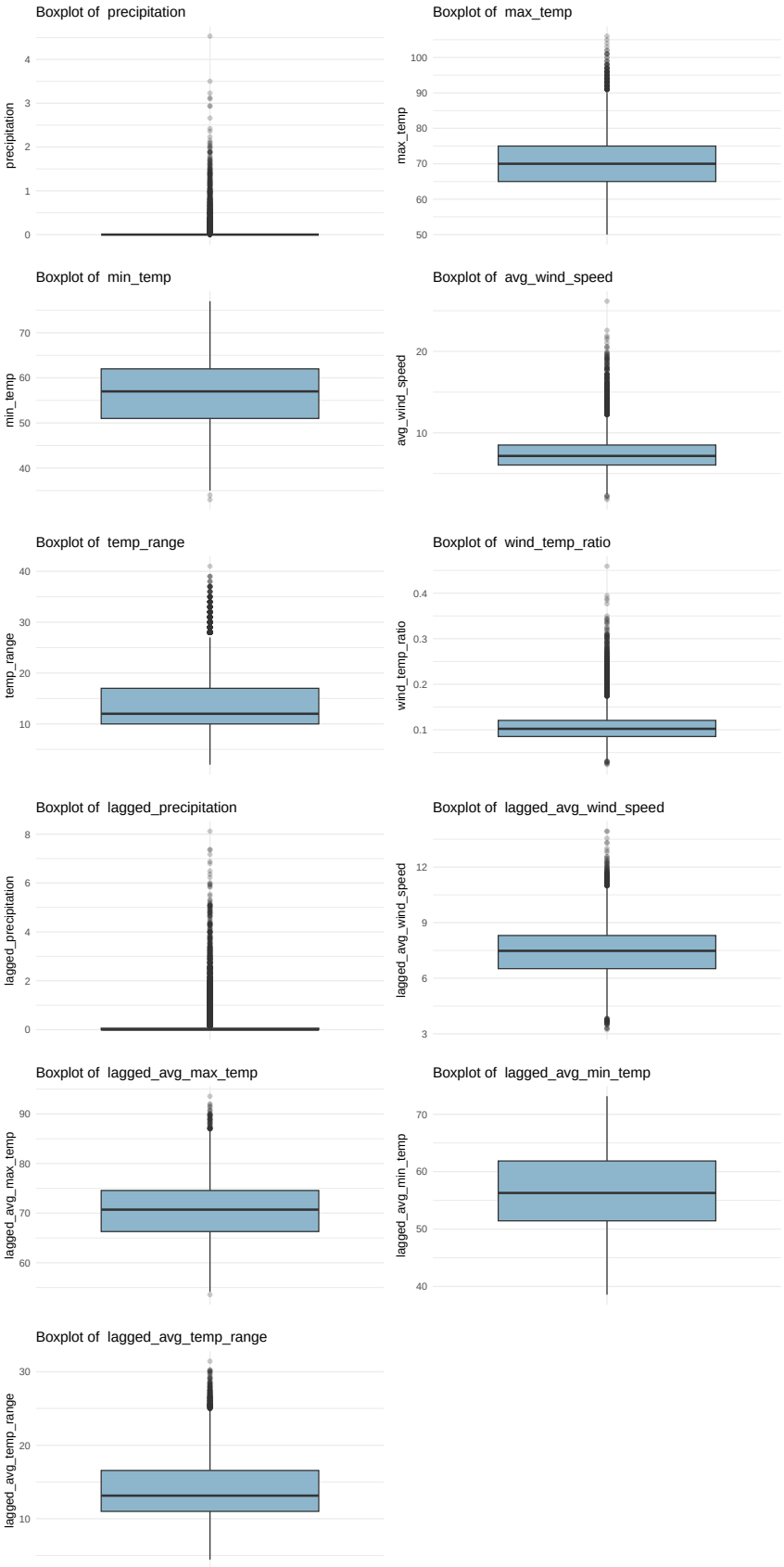
Summary statistics and boxplots were created for each numeric variable. The summary statistics table revealed that there was rarely any precipitation. Quartiles one through three each had a value of zero inches, and a non-zero value was not present until the maximum value of 4.53, meaning any non-zero values appeared in quartile four. The lagged_precipitation variable was the cumulative precipitation over the previous seven days, explaining why the maximum value of 8.12 was greater than the individual day maximum of 4.53. Nearly all weather-related variables contained outliers, except for the two minimum temperature metrics. Outliers were considered acceptable for the analysis because the weather is unpredictable, and extremes are common. In fact, outliers may be crucial in providing fire ignition signals, which would be explored during bivariate analysis.

```
# isolate numeric predictor variables
cont_vars <- c("precipitation","max_temp","min_temp","avg_wind_speed","temp_range",
    "wind_temp_ratio","lagged_precipitation","lagged_avg_wind_speed",
    "lagged_avg_max_temp","lagged_avg_min_temp","lagged_avg_temp_range")

# create dataframe with numeric predictors
num_vars <- df_clean %>%
  select(where(is.numeric))

# Compute quartiles for each variable
(quartile_summary <- map_dfr(names(num_vars), function(var) {
  quantiles <- quantile(num_vars[[var]], probs = c(0, 0.25, 0.5, 0.75, 1), na.rm = TRUE)
  tibble(
    Variable = var,
    Min = round(quantiles[1],2),
    `Q1 (25%)` = round(quantiles[2],2),
    `Median (50%)` = round(quantiles[3],2),
    `Q3 (75%)` = round(quantiles[4],2),
    Max = round(quantiles[5],2))
})
)
```

```
## # A tibble: 13 x 6
```

```
##    Variable                 Min `Q1 (25%)` `Median (50%)` `Q3 (75%)`     Max
##    <chr>                  <dbl>      <dbl>          <dbl>      <dbl>   <dbl>
##  1 precipitation           0          0              0          0       4.53
##  2 max_temp               50         65             70         75     106
##  3 min_temp               33         51             57         62      77
##  4 avg_wind_speed          1.79       6.04           7.16       8.5    26.2
##  5 year                 1984       1993.          2004.      2013    2023
##  6 temp_range              2         10             12         17      41
##  7 wind_temp_ratio         0.02       0.09           0.1        0.12    0.46
##  8 lagged_precipitation    0          0              0          0.06    8.12
##  9 lagged_avg_wind_speed   3.23       6.52           7.48       8.31   13.9
## 10 day_of_year             1         92            183        274     366
## 11 lagged_avg_max_temp    53.6       66.3           70.7       74.6    93.6
## 12 lagged_avg_min_temp    38.6       51.4           56.3       61.9    73.1
## 13 lagged_avg_temp_range   4.43      11             13.1       16.6    31.4
```

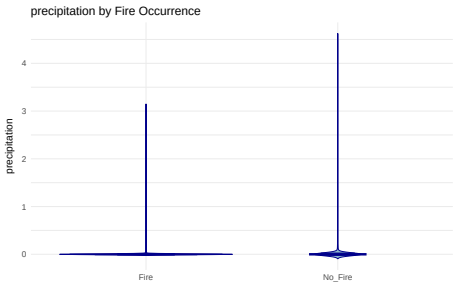
```r
# create loop function that will create boxplots for all numeric predictors
u_plot_list <- list()

for (v in cont_vars) {
  u_box <-
  ggplot(df_clean, aes(x = "", y = .data[[v]])) +
    geom_boxplot(outlier.alpha = 0.25, fill = "lightskyblue3") +
    labs(title = paste("Boxplot of ",v), x = NULL, y = v) +
    theme(legend.position = "none")+
    theme_minimal()

  u_plot_list[[length(u_plot_list) + 1]] <- u_box
}
wrap_plots(u_plot_list, ncol = 2)
```

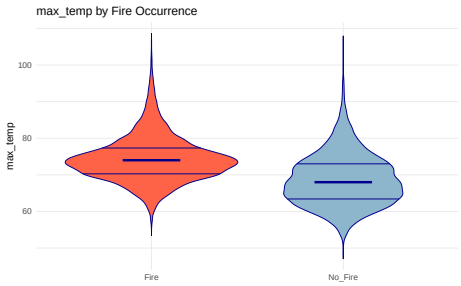

Boxplot of precipitation
precipitation
Boxplot of max_temp
max_temp
Boxplot of min_temp
min_temp
Boxplot of avg_wind_speed
avg_wind_speed
Boxplot of temp_range
temp_range
Boxplot of wind_temp_ratio
wind_temp_ratio
Boxplot of lagged_precipitation
lagged_precipitation
Boxplot of lagged_avg_wind_speed
lagged_avg_wind_speed
Boxplot of lagged_avg_max_temp
lagged_avg_max_temp
Boxplot of lagged_avg_min_temp
lagged_avg_min_temp
Boxplot of lagged_avg_temp_range
lagged_avg_temp_range

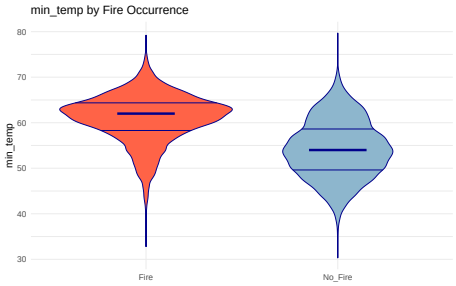
Bivariate Analysis

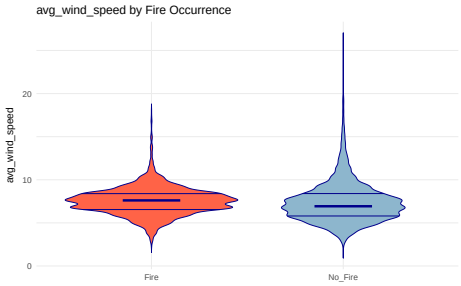
Violin plots, with lines for quartiles one through three, were created for each numeric variable set against the fire_occurrence target variable. Violin plots are similar to boxplots but go one step further by showing the density of data across the values (Ferrer, 2025). Precipitation and lagged precipitation showed that fires rarely occur on days with rain. This suggested that dry or drought conditions may be highly predictive. Fires are clearly associated with high individual day and lagged maximum temperatures, meaning hot days and sustained high temperatures would be predictive. High minimum temperatures show the most distinct difference between the fire and no fire groups. When assessing variable importance, there would be a strong likelihood that minimum temperature would be the most predictive variable. The wind metrics did not appear to differ much between the two groups. Fire occurrence is slightly higher on windy days, but the difference was much less pronounced than the temperature metrics. At this point in the analysis, the exploratory work supported the research question posed.

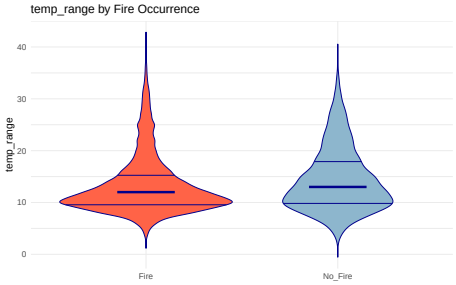
```
# create loop function that will create boxplots for all numeric predictors by target
b_plot_list <- list()

for (v in cont_vars) {
  b_vio <-
  ggplot(df_clean, aes(x = fire_occurred, y = .data[[v]], fill = fire_occurred)) +
    geom_violin(trim = FALSE,
                draw_quantiles = c(0.25, 0.75),
                color = "darkblue",
                linewidth = 0.5)+
    stat_summary(fun = median, geom = "crossbar", width = 0.3, color = "darkblue") +
    scale_fill_manual(values = c("No_Fire" = "lightskyblue3", "Fire" = "tomato1"))+
    guides(fill = "none")+
    labs(title = paste(v, "by Fire Occurrence"), x = NULL, y = v) +
    theme(legend.position = "none")+
    theme_minimal()

  b_plot_list[[length(b_plot_list) + 1]] <- b_vio
}
wrap_plots(b_plot_list, ncol = 2)
```

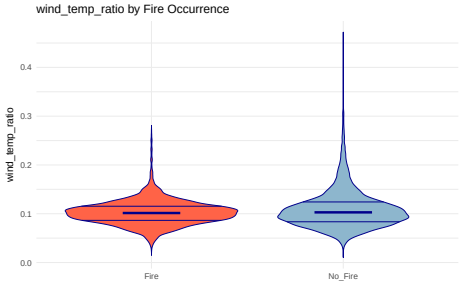

precipitation by Fire Occurrence
precipitation
0
1
2
3
4
Fire
No_Fire

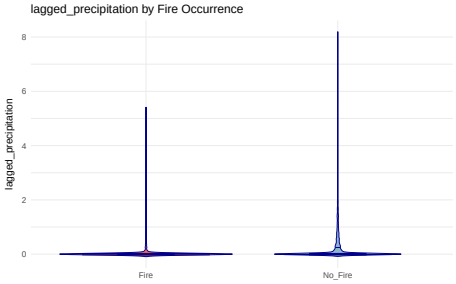

max_temp by Fire Occurrence
max_temp
60
80
100
Fire
No_Fire

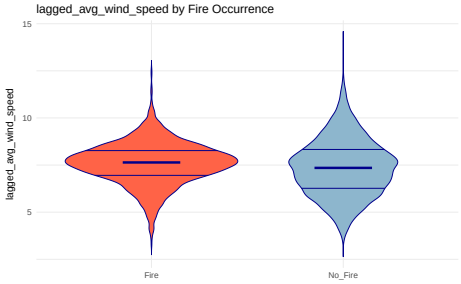

min_temp by Fire Occurrence
min_temp
30
40
50
60
70
80
Fire
No_Fire

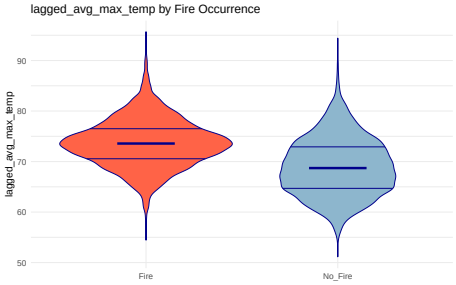

avg_wind_speed by Fire Occurrence
avg_wind_speed
0
10
20
Fire
No_Fire

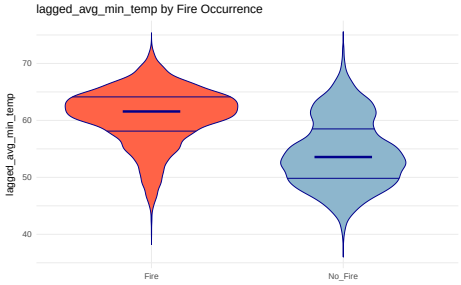

temp_range by Fire Occurrence
temp_range
0
10
20
30
40
Fire
No_Fire

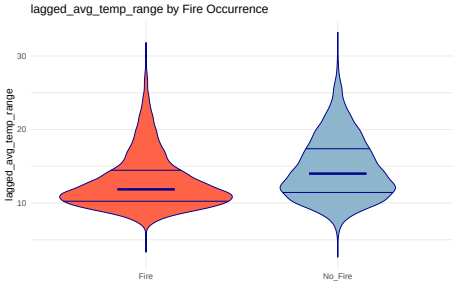

wind_temp_ratio by Fire Occurrence
wind_temp_ratio
0.0
0.1
0.2
0.3
0.4
Fire
No_Fire


lagged_precipitation by Fire Occurrence
lagged_precipitation
0
2
4
6
8
Fire
No_Fire


lagged_avg_wind_speed by Fire Occurrence
lagged_avg_wind_speed
5
10
15
Fire
No_Fire

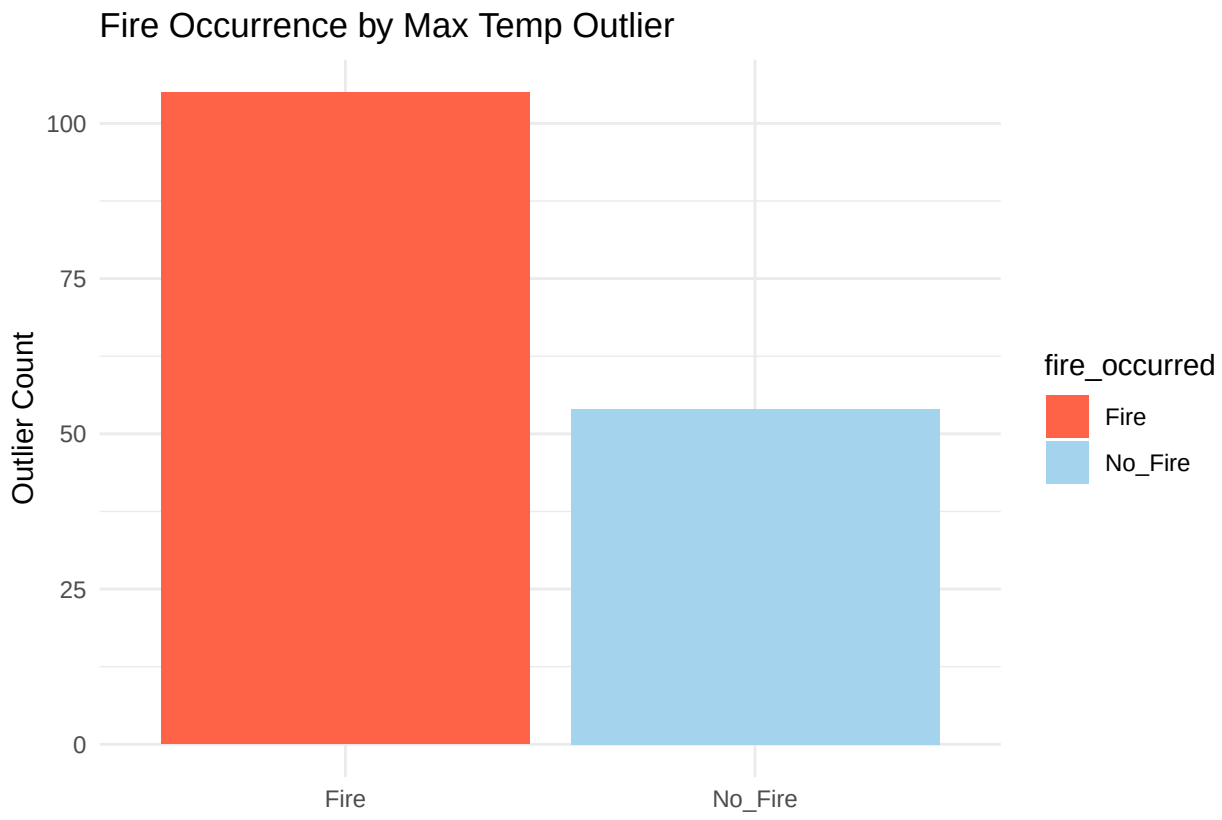

lagged_avg_max_temp by Fire Occurrence
lagged_avg_max_temp
50
60
70
80
90
Fire
No_Fire


lagged_avg_min_temp by Fire Occurrence
lagged_avg_min_temp
40
50
60
70
Fire
No_Fire

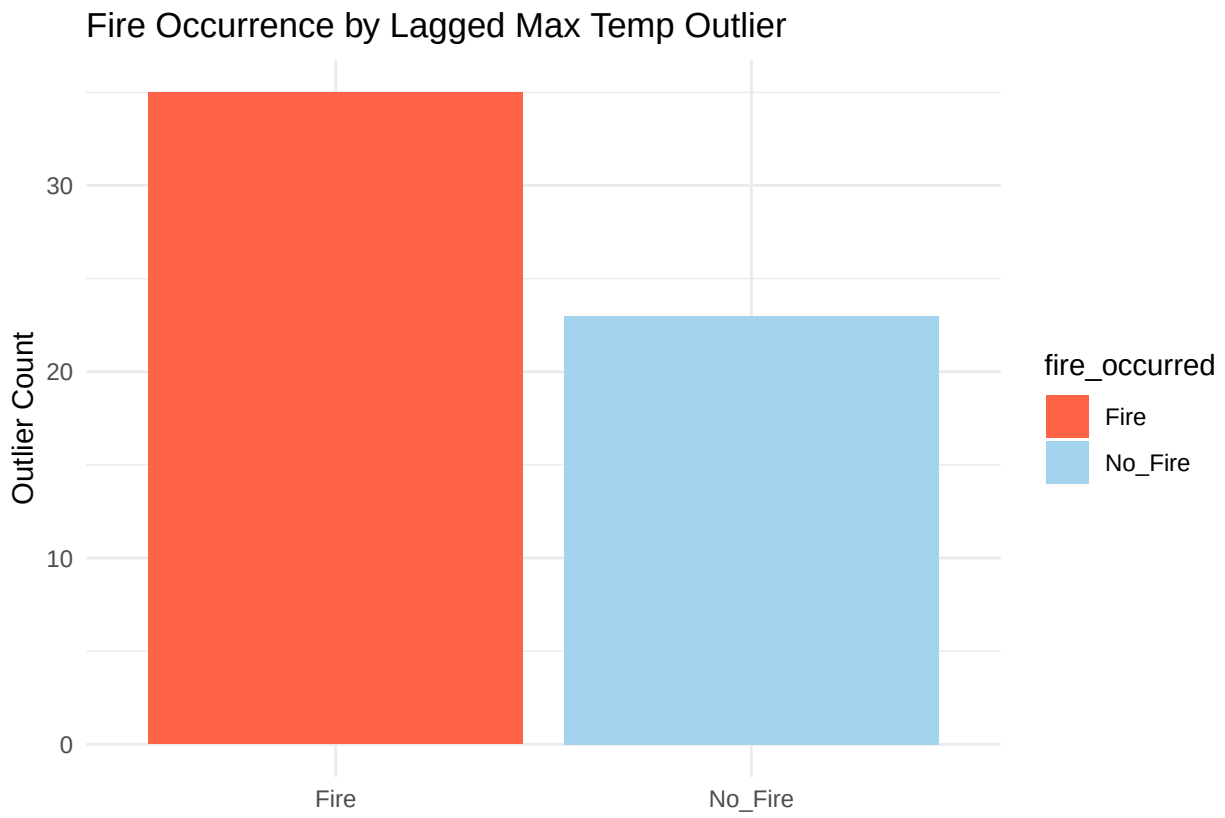

lagged_avg_temp_range by Fire Occurrence
lagged_avg_temp_range
10
20
30
Fire
No_Fire

In the bivariate analysis above, it was clear that the maximum temperature metrics would be an important predictor for the random forest modeling. It was also discovered during the univariate analysis that these metrics contained high-end outliers. Some additional work around the maximum temperature outliers was performed below. The interquartile range (IQR) method flagged outliers in the dataset. In the IQR method, the quartile one value was subtracted from the quartile three value to get the interquartile range, which was then multiplied by 1.5 and added back to the quartile three value. If an individual data point exceeded the calculated value, it would be considered an outlier. In the charts below, fires were quite common on outlier days. There were 159 days flagged as outliers for the maximum temperature variable. Fires occurred on 105, or 66%, of these outlier days. Similarly, there were a total of 58 days flagged as outliers for the lagged average maximum temperature variable. Fires occurred on 35, or 60% of the outlier days. When the temperature is excessively hot on an individual day or a seven-day basis, there is a greater chance of a fire occurring.

```
# calculate outliers for max temp metrics using IQR method
q1_max_temp <- quartile_summary %>%
  filter(Variable == "max_temp") %>%
  pull(`Q1 (25%)`)
q1_lagged_max_temp <- quartile_summary %>%
  filter(Variable == "lagged_avg_max_temp") %>%
  pull(`Q1 (25%)`)

q3_max_temp <- quartile_summary %>%
  filter(Variable == "max_temp") %>%
  pull(`Q3 (75%)`)
q3_lagged_max_temp <- quartile_summary %>%
  filter(Variable == "lagged_avg_max_temp") %>%
  pull(`Q3 (75%)`)

iqr_max_temp <- q3_max_temp - q1_max_temp
iqr_lagged_max_temp <- q3_lagged_max_temp - q1_lagged_max_temp

# flag high-end outliers in df_clean
df_clean$outlier_max_temp <- df_clean$max_temp > (q3_max_temp + (1.5 * iqr_max_temp))
df_clean$outlier_lagged_max_temp <- df_clean$lagged_avg_max_temp > (q3_lagged_max_temp + (1.5 * iqr_lagg

# explore fire occurence among max temp outliers
df_clean %>%
  filter(outlier_max_temp == TRUE) %>%
  count(fire_occurred) %>%

  ggplot(aes(x = fire_occurred, y = n, fill = fire_occurred))+
    geom_bar(stat="identity")+
    scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
    labs(title = "Fire Occurrence by Max Temp Outlier", y = "Outlier Count", x = "") +
    theme_minimal()
```




```
df_clean %>%
  filter(outlier_lagged_max_temp == TRUE) %>%
  count(fire_occurred) %>%

  ggplot(aes(x = fire_occurred, y = n, fill = fire_occurred))+
    geom_bar(stat="identity")+
    scale_fill_manual(values = c("No_Fire" = "lightskyblue2", "Fire" = "tomato1"))+
    labs(title = "Fire Occurrence by Lagged Max Temp Outlier", y = "Outlier Count", x = "") +
    theme_minimal()
```



D, Analysis

The primary analysis would be performed using a Random Forest Classifier, a supervised machine learning algorithm. Supervised machine learning involves training a model with labeled data, where each input comes with a corresponding correct output (GeeksforGeeks, 2025a). The models learn the patterns and intricacies of what leads to the correct output and apply those learnings to unseen data to make predictions. Supervised learning is often highly accurate and has strong predictive power. Supervised models can also be easier to interpret than many unsupervised methods and offer straightforward evaluation metrics. However, supervised learning models require quality, labeled data to achieve these impressive results, which is not always available or practical. Supervised learning can also be prone to overfitting, which means the models learn the patterns of the training data too well and, at times, do not perform well on unseen data. All in all, supervised learning algorithms are the most widely used approaches in machine learning (Polamuri, n.d.).

Supervised learning algorithms can be broken into two main categories: classification and regression. Classification algorithms predict a categorical target value, while regression algorithms predict a continuous numeric variable. Random Forest can be used for either; however, in this case, it is well-suited to classify a binary outcome like whether a fire will occur. The Random Forest algorithm creates multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. Each tree is trained on a subset of data, or certain features, and the final prediction is made by aggregating the predictions from all trees using a majority voting system (Anwla, n.d.).

Random Forest was chosen for this analysis for the aforementioned reasons, as well as its ability to handle large, complex datasets. Random Forest can handle data with many features, whereas other methods can lead to overfitting as the number of features increases. The original dataset contained 11 predictors, and with additional feature engineering, the dataset expanded to 17 predictors. Random Forest also handles outliers

well, which is critical since the exploratory analysis showed that most weather metrics contain outliers. The success with outliers is because individual decision trees split data on feature thresholds, not distance measurements, so the magnitude of values does not impact the tree splits. Each tree split also selects a random subset of features, which prevents any single outlier from dominating the model (Anwla, n.d.).

Potential drawbacks to using Random Forest include reduced interpretability and higher computational cost. A single decision tree can be easy to understand because the splits are visible and the path to a prediction can be traced. However, since Random Forest creates multiple decision trees and performs majority voting, understanding why a prediction was made can be more challenging. There is also a balance trade-off between the number of trees created and computational cost. Having the model create more trees can increase accuracy, but at the cost of consuming more memory, processing power, and time (GeeksforGeeks, 2025a). During the analysis, when a larger number of trees were used, the accuracy did increase marginally, but the models took much longer to run.

Model Setup

To begin the model-building process, the 'set.seed' function was used to create reproducible results. This function would apply more when using random samples, guaranteeing that the same random samples appear in the train and test sets. Random samples were not used in the analysis; however, it was best practice to implement. Next, a new data frame was created that housed some minor changes to variables causing issues when running Random Forest. The target variable fire_occurred was reordered to ensure the positive class appeared first. The positive class represented the outcome of interest that the model would predict (Ebner, 2023). The logical fields from the dataset, near_holiday, outlier_max_temp, and outlier_lagged_max_temp were explicitly converted to factors to avoid unexpected behavior.

Next, a cutoff date was defined for the training and test groups. The cutoff value used was January 1, 2018, which ensured that six years of data, or 2,191 days/records, appeared in the test subset. Two training subsets were used in the analysis, the first of which contained data from 1984 through 2017. This training set had 12,419 days, or 34 years' worth of data. This model approach represented a train/test split of approximately 85%/15% which would be acceptable given the data size of 14,610 records. The second training subset was created after observing the trend of fire occurrence over the years. Recall that fire occurrence picked up significantly after 2004, which was proven statistically significant by a t-test. Therefore, the second training set included data from 2004 through the end of 2017, which equated to 5,114 days/records or 14 years' worth of data. The same test holdout was used in both scenarios, so the second approach represented a 72%/28% train/test split, which again was an acceptable ratio. The two training sets were created because there were concerns that, given fires were less common from 1984 to 2003, the model training would be unduly influenced by years with fewer fire occurrences. Using a second training set, which included years when fires were more prevalent, the idea was that the model could train on data more similar to the test set.

A time-based split was chosen over a random split in the analysis. Implementing a time-based split where the unseen test data was in the future would simulate real-world conditions. The model would learn from the past and predict the future. Constrastingly, a random split would shuffle the data and randomly assign rows to the training and test sets. It would be possible for future records to leak into the training set and cause the model to learn from data that should not have been available to it, which is not realistic. The time-based split works by training on an initial fixed time window of data and validating on a defined future horizon. The model then rolls forward by the time specified in the horizon and repeats the process until it chronologically reaches the end of the dataset. Each time slice is called a fold (Kuhn, n.d.). Another advantage to this method is that by respecting the time order, the model can pick up on seasonal trends, such as the fact that fires were more prevalent in summer months. With a random split, there would be no guarantee that the model would learn these seasonal impacts. The primary disadvantage of the time-based split is the computational strain. It took significantly longer to train the two models using this method.

For the training set that contained data from 1984 to 2017, a time window of 10 years was chosen with a horizon of 365 days. The parameters led to 24 training folds. The calculation for this was as follows: 34

years equated to approximately 12,410 total days, less the initial window size of 10 years (approximately 3,650 days), divided by the horizon of 365 days ((12,410 - 3,650) / 365 = 24). For the smaller training set representing 2004 to 2017, a time window of 4 years was used with the same 365-day horizon. These parameters led to 10 training folds. Using the same formula, 14 years of data was approximately 5,110 days, less a time window of 4 years (1,460 days), divided by a horizon of 365 days ((5,110 - 1,460) / 365 = 10). Additional tuning parameters in the model setup further impacted the number of training folds. A tuneLength set to 5 meant the model would try selecting a random number of predictors for each tree split, 5 times. The rule of thumb for a tuning length would be the square root of the number of predictors (Kuhn & Vaughan, n.d.). There are 18 predictors in the dataset, so this meant a minimum tuning length of four. A value of 5 was selected to be more thorough. The 'num.trees' argument specifies how many distinct trees would be created. The first model created 6,000 trees: 24 training folds * 5 tuning values * 50 trees. The second model created 2,500 trees: 10 training folds * 5 tuning values * 50 trees.

The remaining model parameters dealt with how the model would evaluate the data. The method argument within the train function specified the classification or regression model. The ranger model was chosen, which, according to Wright & Ziegler (n.d.), is a fast implementation of Random Forests suited for high-dimensional data. Sensitivity was chosen for the evaluation metric, which differed from the default accuracy value for a typical classification problem. Choosing accuracy for the evaluation metric would assume that the classes were balanced and that all errors were equally important. This would not be the case when trying to predict when a fire will occur. Missing positive cases, when a fire occurs, would be exceptionally costly.

On the other hand, sensitivity is related to the true positive rate, which is the true positives divided by the sum of true positives and false negatives (TP / (TP + FN)). By setting the evaluation metric to Sens, the model would try to maximize the detection of the positive class. These values and calculations can be found later in the confusion matrix. Setting the model importance to permutation can help offer insight into how distinctive features contribute to the model's predictive power. Permutation evaluates how much the performance drops when a feature's values are shuffled. The main disadvantage of this method is computational cost. It performs multiple re-evaluations of the model for each variable, which can be slow (Gulen, 2025). Many model choices had a high computational cost to achieve better model results. To speed up the training process, parallel processing was enabled with the doParallel package. The number of cores on the machine was detected, and R used the number of cores minus one core while running tasks.

```
# pre-modeling manipulation
df_model <- df_clean
df_model$fire_occurred <- factor(df_model$fire_occurred,
                                 levels = c("Fire","No_Fire"))
df_model <- df_model %>%
  mutate(
    near_holiday = factor(near_holiday),
    outlier_max_temp = factor(outlier_max_temp),
    outlier_lagged_max_temp = factor(outlier_lagged_max_temp)
  )

# establish cutoff date for training/test sets
cutoff_date <- as_date("2018-01-01")

# identify predictor variables to be used in analysis
predictor_vars <- c("precipitation","max_temp","min_temp", "avg_wind_speed", "temp_range", "wind_temp_ra
                    "lagged_precipitation","lagged_avg_wind_speed", "lagged_avg_max_temp", "lagged_avg_
                    "season","year", "month", "day_of_year", "near_holiday", "outlier_max_temp","outlier

# train controls
init_yrs1 <- 10
init_yrs2 <- 4
```

```r
init_window1 <- init_yrs1 * 365
init_window2 <- init_yrs2 * 365
horizon <- 365

ctrl <- trainControl(method = "timeslice",
                     initialWindow = init_window1,
                     horizon = horizon,
                     fixedWindow = TRUE,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = "final",
                     allowParallel = TRUE)

ctrl2 <- trainControl(method = "timeslice",
                     initialWindow = init_window2,
                     horizon = horizon,
                     fixedWindow = TRUE,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = "final",
                     allowParallel = TRUE)

model_metric <- "Sens"
model_imp <- "permutation"
model_tuneLength <- 5
model_numTrees <- 50

m1_folds <- ((34*365)-init_window1) / horizon
m2_folds <- ((14*365)-init_window2) / horizon

m1_trees <- m1_folds * model_tuneLength * model_numTrees
m2_trees <- m2_folds * model_tuneLength * model_numTrees

# train/test splits for all models
train <- df_model %>% filter(date <  cutoff_date)
train_2 <- train %>% filter(year >= 2004)
test <- df_model %>% filter(date >= cutoff_date)

# train
train_x <- train %>% select(all_of(predictor_vars))
train_x2 <- train_2 %>% select(all_of(predictor_vars))

train_y <- train$fire_occurred
train_y2 <- train_2$fire_occurred

# test
test_x  <- test %>% select(all_of(predictor_vars))
test_y <- test$fire_occurred

# establish parallel processing
num_cores <- detectCores()
cl <- makeCluster(num_cores-1)
registerDoParallel(cl)
```

```
# model 1 - all years
if (file.exists("model_1.rds")) {
  rf_model_1 <- readRDS("model_1.rds")
} else {

set.seed(444)
rf_model_1 <- train(
  x          = train_x,
  y          = train_y,
  method     = "ranger",
  trControl  = ctrl,
  metric     = model_metric,
  importance = model_imp,
  tuneLength = model_tuneLength,
  num.trees  = model_numTrees
)
saveRDS(rf_model_1, "model_1.rds")
}

# model 2 - 2004+
if (file.exists("model_2.rds")) {
  rf_model_2 <- readRDS("model_2.rds")
} else {

set.seed(444)
rf_model_2 <- train(
  x          = train_x2,
  y          = train_y2,
  method     = "ranger",
  trControl  = ctrl2,
  metric     = model_metric,
  importance = model_imp,
  tuneLength = model_tuneLength,
  num.trees  = model_numTrees
)
saveRDS(rf_model_2, "model_2.rds")
}
stopCluster(cl)
```

Model Results

Once the models were trained, the following steps were to use the predict function for each model against the test holdout. This applied the model learnings to the unseen data. The two prediction variables below (pred_1/pred_2) assigned a Fire/No Fire designation for each observation in the test set. The predictions were needed to set up the confusion matrices. The probability variables (probs_1/probs_2) returned a probability value between 0 and 1 that each observation belonged to the positive class. The probabilities were needed to set up the ROC curve and AUC calculation.

Confusion matrices gave valuable information about model performance. The main element of the output was a table that compared the predicted and actual outcomes. It identified true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In the analysis context, a true positive was when the model predicted a fire and there actually was a fire. A true negative was the opposite; the model predicted

no fire, and there actually was no fire. A false positive would be where the model predicted a fire, but there was actually no fire. Finally, a false negative occurred when the model predicted no fire, and there actually was a fire. A false negative would be the worst-case scenario in the model. Model 1, the all-year model, correctly predicted 720 fire occurrences and 954 no fire occurrences. There were 245 false positives, where the model predicted fire and there was no fire, and 272 false negatives, where there was a fire and the model predicted no fire.

The model accuracy was determined by taking the true positives and true negatives and dividing by the entire population ((TP+TN) / (TP+TN+FP+FN)), which resulted in an accuracy of 0.7640347 for model 1. However, as previously stated, accuracy was not the best barometer for an unbalanced class like this. Fire occurrence was relatively rare, and there would be costly ramifications for not correctly predicting the positive class. Sensitivity would be the better judgment of the model, which could be calculated by taking the true positives and dividing by the sum of true positives and false negatives (TP / (TP+FN)). Model 1 achieved a 0.7258065 sensitivity rate. Model 2, the 2004-2017 model, correctly predicted 714 fire occurrences and 954 no fire occurrences. There were 245 false positives, where the model predicted fire and there was no fire, and 278 false negatives, where the model predicted no fire and there actually was a fire. This resulted in a slightly worse accuracy for model 2 of 0.7612962 and a slightly worse sensitivity rate of 0.7197581. Both models significantly outperformed the no information rate of 0.547, which was the resulting accuracy achieved by always picking the majority class, or in this case, always predicting no fire. The accuracy difference was confirmed as statistically significant for both models by the p-value of 0.00. There was a clear signal that the models could learn meaningful patterns in the data.

A summary table was created to combine accuracy, sensitivity, and precision from the confusion matrix and two additional model comparison metrics. The F1 score was calculated for each model, which struck a balance between sensitivity and precision (Ebner, 2023). Precision is the proportion of correct positive predictions (TP / (TP+FP)). The ROC-AUC score, also called the area under the curve, represented the overall ability of the model to distinguish between positive and negative classes according to Google Developers (n.d.). A ROC-AUC score of 1.0 would mean the model was perfect at distinguishing between positive and negative classes. In contrast, a score of 0.5 would mean the model could not distinguish between the positive and negative classes, so effectively the model would be no better than random chance. Both models performed similarly, with Model 1 having an F-score of 0.735 versus 0.732 for model 2. Model 1 had an ROC-AUC score of 0.828 versus 0.832 for model 2. Generally, ROC-AUC scores of 0.9 and above are considered excellent, and scores between 0.8 and 0.9 are considered good. Both models fall into the good category.

The most important feature in both models was day_of_year. From previous charts, it was clear that fires occurred most often in the summer, specifically in July and August. The models could pick up on trends at the most granular level. Season and month were important variables in both models, though season ranked fourth most important in model 1. The engineered feature lagged_avg_min_temp was the third most important variable in model 1, and nearly all other temperature variables were of moderate importance in both models. There seemed to be a stark drop-off in importance after the temperature metrics, meaning there was little predictive value in the precipitation, wind, holiday, and outlier metrics. The variable importance would be a good sign overall for the analysis hypothesis.

```
# model predictions
pred_1 <- predict(rf_model_1, test_x)
pred_2 <- predict(rf_model_2, test_x)

# model probabilities
probs_1 <- predict(rf_model_1, test_x, type = "prob")[, "Fire"]
probs_2 <- predict(rf_model_2, test_x, type = "prob")[, "Fire"]

# confusion matrices
cm_1 <- confusionMatrix(pred_1, test_y, positive = "Fire")
cm_2 <- confusionMatrix(pred_2, test_y, positive = "Fire")
```

```r
model1_acc <- cm_1$overall["Accuracy"]
model1_sens <- cm_1$byClass["Sensitivity"]
model1_tp <- cm_1$table["Fire", "Fire"]
model1_tn <- cm_1$table["No_Fire", "No_Fire"]
model1_fp <- cm_1$table["Fire", "No_Fire"]
model1_fn <- cm_1$table["No_Fire", "Fire"]

model2_acc <- cm_2$overall["Accuracy"]
model2_sens <- cm_2$byClass["Sensitivity"]
model2_tp <- cm_2$table["Fire", "Fire"]
model2_tn <- cm_2$table["No_Fire", "No_Fire"]
model2_fp <- cm_2$table["Fire", "No_Fire"]
model2_fn <- cm_2$table["No_Fire", "Fire"]

# view confusion matrix for model 1
cm_1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Fire No_Fire
##    Fire     720     245
##    No_Fire  272     954
##
##                Accuracy : 0.764
##                  95% CI : (0.7457, 0.7817)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5227
##
##  Mcnemar's Test P-Value : 0.2528
##
##             Sensitivity : 0.7258
##             Specificity : 0.7957
##          Pos Pred Value : 0.7461
##          Neg Pred Value : 0.7781
##              Prevalence : 0.4528
##          Detection Rate : 0.3286
##    Detection Prevalence : 0.4404
##       Balanced Accuracy : 0.7607
##
##        'Positive' Class : Fire
##
```

```r
# view confusion matrix for model 2
cm_2
```

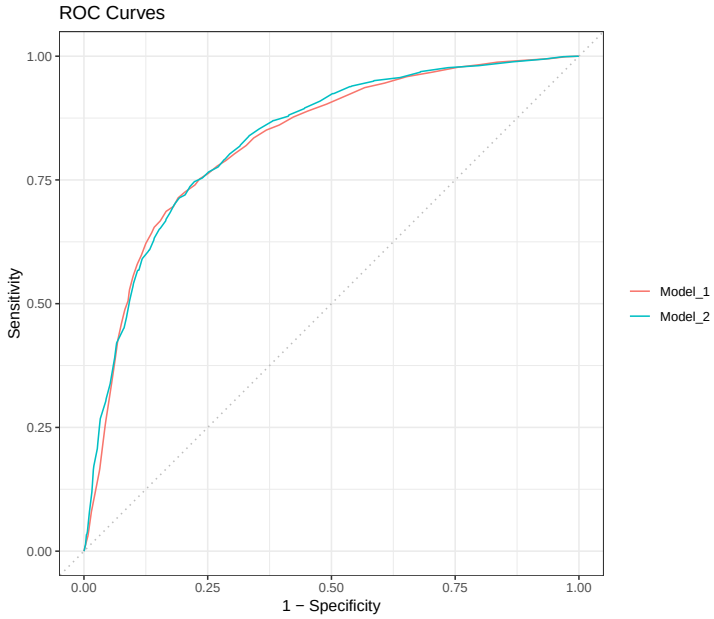
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Fire No_Fire
##    Fire     714     245
```

```
##    No_Fire  278     954
##
##                Accuracy : 0.7613
##                  95% CI : (0.7429, 0.779)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.5169
##
##  Mcnemar's Test P-Value : 0.1617
##
##             Sensitivity : 0.7198
##             Specificity : 0.7957
##          Pos Pred Value : 0.7445
##          Neg Pred Value : 0.7744
##              Prevalence : 0.4528
##          Detection Rate : 0.3259
##    Detection Prevalence : 0.4377
##       Balanced Accuracy : 0.7577
##
##        'Positive' Class : Fire
##
```

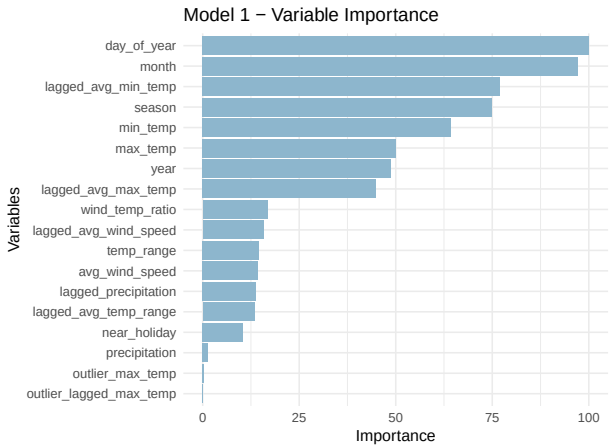
```r
# ROC-AUC
auc_1 <- pROC::auc(roc(response = test_y, predictor = probs_1, levels = rev(levels(test_y))))
auc_2 <- pROC::auc(roc(response = test_y, predictor = probs_2, levels = rev(levels(test_y))))

# gather metrics in summary table
data.frame(
  Model       = c("Model_1","Model_2"),
  Accuracy    = c(cm_1$overall["Accuracy"], cm_2$overall["Accuracy"]),
  Sensitivity = c(cm_1$byClass["Sensitivity"], cm_2$byClass["Sensitivity"]),
  Precision   = c(cm_1$byClass["Pos Pred Value"], cm_2$byClass["Pos Pred Value"]),
  F1          = c(2 * (cm_1$byClass["Pos Pred Value"]*cm_1$byClass["Sensitivity"]) /
                    (cm_1$byClass["Pos Pred Value"]+cm_1$byClass["Sensitivity"]),

                  2 * (cm_2$byClass["Pos Pred Value"]*cm_2$byClass["Sensitivity"]) /
                    (cm_2$byClass["Pos Pred Value"]+cm_2$byClass["Sensitivity"])),
  ROC_AUC     = c(as.numeric(auc_1), as.numeric(auc_2))
)
```

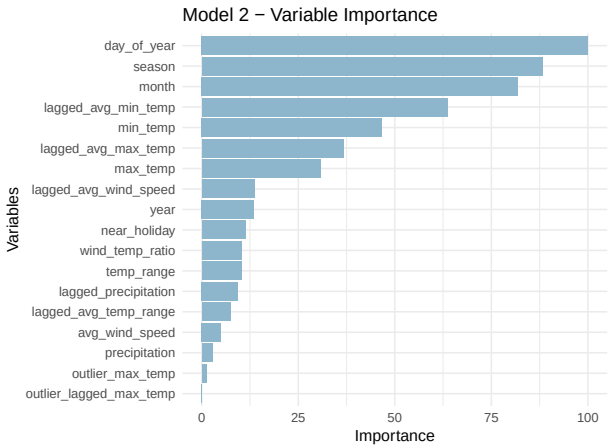
```
##     Model  Accuracy Sensitivity Precision        F1   ROC_AUC
## 1 Model_1 0.7640347   0.7258065 0.7461140 0.7358201 0.8276765
## 2 Model_2 0.7612962   0.7197581 0.7445255 0.7319323 0.8321514
```

```r
# create ROC curve to compare two models
scores <- list(probs_1, probs_2)
labels <- as.integer(test_y == "Fire")
mm <- mmdata(scores = scores, labels = labels, modnames = c("Model_1","Model_2"))
ev <- evalmod(mm)

autoplot(ev, "ROC") + ggtitle("ROC Curves")
```




```
# establish variable importance
imp1 <- varImp(rf_model_1)
imp2 <- varImp(rf_model_2)

df_imp1 <- imp1$importance
df_imp2 <- imp2$importance

df_imp1$Variable <- rownames(df_imp1)
df_imp2$Variable <- rownames(df_imp2)

# variable importance for model 1
gg_imp1 <- ggplot(df_imp1, aes(x = reorder(Variable, Overall), y = Overall)) +
  geom_bar(stat = "identity", fill = "lightskyblue3") +
  coord_flip() +
  labs(title = "Model 1 - Variable Importance", x = "Variables", y = "Importance") +
  theme_minimal()

# variable importance for model 2
gg_imp2 <- ggplot(df_imp2, aes(x = reorder(Variable, Overall), y = Overall)) +
  geom_bar(stat = "identity", fill = "lightskyblue3") +
  coord_flip() +
  labs(title = "Model 2 - Variable Importance", x = "Variables", y = "Importance") +
  theme_minimal()

wrap_plots(gg_imp1, gg_imp2, ncol = 2)
```





Model Selection & Conclusion

Model Selection & Conclusion The confusion matrices and summary table showed that both models were strong performers. Model 1, the 1984 to 2017 model, outperformed Model 2, the 2004 to 2017 model, in all metrics except for ROC-AUC, giving it a slight edge. Considering this was a classification problem where the true positive rate would be critical, model 1 was considered the best choice, given it also had the highest sensitivity rate at 0.726. The final step in the analysis was calculating a confidence interval for ROC-AUC, which would determine whether to accept or reject the null hypothesis (H0). If the lower bound of the confidence interval exceeded 0.5, the analysis would reject (H0).

The best model, model 1, achieved a ROC-AUC score of 0.828 (95% CI: 0.8103-0.845 (DeLong)), significantly above the 0.5 threshold. Therefore, the null hypothesis that weather features have no predictive power was rejected, and the alternative hypothesis was supported, indicating that weather conditions predict fire ignition events better than chance.

```
# create confidence interval for AUC
roc_obj <- roc(test_y, probs_1)

ci <- ci.auc(roc_obj)

cat("Hypothesis Testing: Reject H  if Lower Bound > 0.5\n")
```

```
## Hypothesis Testing: Reject H  if Lower Bound > 0.5
```

```
ci
```

```
## 95% CI: 0.8103-0.845 (DeLong)
```

E, Data Summary and Implications

The analysis began with the question “Can environmental conditions be used to accurately predict when a wildfire will occur?” Two Random Forest models were trained and tested; ultimately, one was deemed the best fit. The final model produced a 95% ROC-AUC confidence interval between 0.8103 and 0.845, which supported the alternate hypothesis that weather conditions predict fire ignition events better than chance.

These findings implied that environmental factors such as minimum and maximum temperature and lagging 7-day temperature averages contain meaningful predictive signals, and that machine learning approaches could leverage these signals to anticipate wildfire risk. By predicting when these conditions were likely to occur, fire agencies could pre-position crews, aircraft, drones, and equipment in areas likely to ignite, thereby reducing response times. If conditions crossed a high-risk threshold, Utility companies could be given more advanced warning to shut down power lines in these impact areas. With the cost of wildfires at $424 billion annually and rising (U.S. Department of the Interior, 2023), more anticipatory moves can save a significant amount of money for the government and citizens alike.

A key limitation of this analysis was that the data reflected historical conditions, and changing climate patterns may alter relationships over time, potentially reducing the stability of the model when applied to future years. However, the model chosen as the best fit contained data from 1984 to 2017, where exploratory analysis revealed the rate of fire occurrences had already been increasing dramatically over the years. The model seemed to be learning from these changing conditions. Another limitation of the analysis was the vague description of where the data was from geographically. Yavas, Kadlec, Kim, and Chen (2025) indicated that the data came from weather observations and wildfire data in California. However, there was no indication of where the fires occurred within the state. Having coordinates would help improve the dataset, or if the data represented averages for the entire state, it should be specified.

Based on these results, the recommended course of action would be integrating weather-based predictive models into wildfire risk monitoring systems, allowing resource managers to better allocate prevention and response efforts during periods of elevated risk. For future study, one approach would be to incorporate additional predictors such as vegetation indices, moisture levels, or additional human activity data to improve predictive accuracy. Another direction to take would be to build more localized models. Local models could include more fine-tuned variations in predictors that may get obscured when looking at larger areas, such as if the data represented all of California. Local models could integrate community-level factors such as proximity to power lines, housing density, or recent land use changes, influencing ignition risk and potential damage. By tailoring models to the conditions of a given landscape and community, fire managers could allocate resources more efficiently, issue more precise warnings, and take preventive actions that directly address the vulnerabilities of high-risk areas.

F, Sources

Report

Anwla, P.K. (n.d.). *What is Random Forest and how it works*. TowardsMachineLearning. (https://towardsmachinelearning.org/random-forest/)

CamperChamp. (2024). *Camping statistics 2024 – USA & North America*. (https://camperchamp.com/camping-statistics/north-america/)

CERN. (n.d.). *Zenodo*. (https://zenodo.org/)

Ebner, J. (2023, December 6). *Positive and negative classes, explained*. SharpSight. (https://sharpsight.ai/blog/positive-and-negative-classes/)

Ferrer, J. (2025, April 2). *Understanding violin plots vs. box plots*. Statology. (https://www.statology.org/understanding-violin-plots-vs-box-plots/)

GeeksforGeeks. (2025a, July 11). *Supervised machine learning*. (https://www.geeksforgeeks.org/machine-learning/supervised-machine-learning/)

GeeksforGeeks. (2025b, July 23). *Pros and cons of R programming language*. (https://www.geeksforgeeks.org/r-language/pros-and-cons-of-r-programming-language/)

Google Developers. (n.d.). *Classification: ROC and AUC. Google Machine Learning Crash Course*. (https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc)

Gulen, K. (2025, March 25). *What is permutation importance*? Dataconomy. (https://dataconomy.com/2025/03/25/what-is-permutation-importance/)

Hardt, B. (2023, July 21). *How does wildfire smoke affect wildlife*? National Wildlife Federation. (https://blog.nwf.org/2023/07/how-does-wildfire-smoke-affect-wildlife/)

Kuhn, M. (n.d.). *Data splitting*. The caret Package. (https://topepo.github.io/caret/data-splitting.html)

Kuhn, M., & Vaughan, D. (n.d.). *Random forests via ranger*. Tidymodels. (https://parsnip.tidymodels.org/reference/details_rand_forest_ranger.html)

Penn State Department of Statistics. (n.d.). *Lesson 3.2: Describing data, part 2. STAT 200 - Statistics Online*. The Pennsylvania State University. (https://online.stat.psu.edu/stat200/lesson/3/3.2)

Polamuri, S. (n.d.). *10 most popular supervised learning algorithms in machine learning*. Data Aspirant. (https://dataaspirant.com/supervised-learning-algorithms/)

Singh, D. (2024, December 11). Class imbalance techniques: Challenges and solutions for machine learning models. Medium. https://medium.com/ai-enthusiast/class-imbalance-techniques-challenges-and-solutions-for-machine-learning-models-49929d54f31f

Smith, A. (2022, August 6). *Weather factors that influence fire danger*. OpenSnow. (https://opensnow.com/news/post/weather-factors-that-influence-fire-danger)

Spector, P. (n.d.). *Factors in R*. University of California, Berkeley. (https://www.stat.berkeley.edu/~spector/s133/factors.html)

U.S. Department of the Interior, Office of Policy Analysis. (2023, May 25). *Increasing damages from wildfires warrant investment in wildland fire management* [Brief]. U.S. Department of the Interior. (https://www.doi.gov/sites/doi.gov/files/ppa-brief-wildland-fire-econ-review-2023-05-25.pdf)

U.S. Environmental Protection Agency. (n.d.). *Why is wildfire smoke a health concern?* (https://www.epa.gov/wildfire-smoke-course/why-wildfire-smoke-health-concern)

U.S. Government Accountability Office. (2025, June 3). *As wildfire trends change, so should the use of technology meant to detect them*. (https://www.gao.gov/blog/wildfire-trends-change-so-should-use-technology-meant-detect-them)

Western Fire Chiefs Association. (2023, May 30). *What technology is used to predict wildfires*? WFCA. (https://wfca.com/wildfire-articles/fire-prediction-technology/)

Wright, M. N., & Ziegler, A. (n.d.). *Ranger: A fast implementation of random forests*. RDocumentation. (https://www.rdocumentation.org/packages/ranger/versions/0.16.0/topics/ranger)

Yavas, C. E., Kadlec, C., Kim, J., & Chen, L. (2025). *California Weather and Fire Prediction Dataset (1984–2025) with Engineered Features* [Data set]. Zenodo. (https://doi.org/10.5281/zenodo.14712845)

Code

Chiu, W. (2021, November 5). *Spatial analysis and modeling in R* [RPubs document]. (https://rstudio-pubs-static.s3.amazonaws.com/832120_2121fc5a4a8548ceab2b294ae2ea205f.html)

Finnstats (May 6, 2021). *Class imbalance – handling imbalanced data in R. R-bloggers*. (https://www.r-bloggers.com/2021/05/class-imbalance-handling-imbalanced-data-in-r/)

GeeksforGeeks (2025, August 7). *How to calculate AUC (Area Under Curve) in R*. (https://www.geeksforgeeks.org/r-language/how-to-calculate-auc-area-under-curve-in-r/)

Global Health with Greg Martin. (2017, June 8). *R programming for beginners – statistic with R (t-test and linear regression) and dplyr and ggplot* [video]. YouTube. https://www.youtube.com/watch?v=ANMuuq502rE

Hornik, K., & Leisch, F. (2022). *TimeDate: Rmetrics - chronological and calendar objects*. R package version 4041.110. (https://www.rdocumentation.org/packages/timeDate/versions/4041.110)

Kuhn, M. (n.d.). *Machine learning with caret in R* [MOOC]. DataCamp. (https://app.datacamp.com/learn/courses/machine-learning-with-caret-in-r)

Nowosad, J. (2025, April 29). *Spatial machine learning with R: caret, tidymodels, and mlr3*. R-Bloggers. (https://www.r-bloggers.com/2025/04/spatial-machine-learning-with-r-caret-tidymodels-and-mlr3/)

Pedersen, T. L. (2025). patchwork: The Composer of Plots (Version 1.3.1.9000) [R package]. https://patchwork.data-imaginist.com/reference/wrap_plots.html

ProgrammingR. (n.d.). *Quartile in R: How to calculate and interpret quartiles*. (https://www.programmingr.com/statistics/quartile/)

Radecic, D. (2024, January 18). *R doParallel: A brain-friendly introduction to parallelism in R*. R-Bloggers. (https://www.r-bloggers.com/2024/01/r-doparallel-a-brain-friendly-introduction-to-parallelism-in-r/)

Saito, T. (n.d.). *Introduction to precrec* [Documentation article]. (https://evalclass.r-universe.dev/articles/precrec/introduction.html)

Statistics Globe. (n.d.). *For loop in R (3 examples) | How to write, run & use for-loops*. (https://statisticsglobe.com/for-loop-in-r)

Tierney, N. (n.d.). *Dealing with Missing Data in R* [MOOC]. DataCamp. (https://app.datacamp.com/learn/courses/dealing-with-missing-data-in-r)

Zeileis, A., & Grothendieck, G. (2020). *Zoo: S3 infrastructure for regular and irregular time series* (Version 1.8-14) [R package]. (https://www.rdocumentation.org/packages/zoo/versions/1.8-14)