HarvardX Data Science Capstone: Dengue Analysis and Modeling

Jennifer Cheng

November 21, 2024

Executive Summary

Dengue fever remains a significant global health challenge, with outbreaks causing substantial morbidity and mortality, particularly in tropical and subtropical regions. Understanding the factors influencing dengue outbreaks is critical for effective prevention and control strategies. This report examines the Dengue dataset, which includes features such as climatic variables, vegetation indices, and temporal data, alongside target variables representing the number of dengue cases (total_cases) in San Juan (SJ) and Iquitos (IQ). These features are essential for analyzing the factors influencing dengue outbreaks and building predictive models.

Methods/Analysis

The analysis involved a structured methodology beginning with **data cleaning**, where missing values and irrelevant features were addressed to ensure dataset integrity. This was followed by **feature engineering**, where additional features were derived to enhance the predictive power of the models. Next, **exploratory data analysis (EDA)** was conducted to analyze patterns and distributions, providing valuable insights into the dataset and potential predictors of outcomes. In the **modeling** phase, multiple machine learning models, including Random Forest (RF) and XGBoost, were implemented, with a particular focus on handling class imbalances. Finally, the **evaluation** phase assessed model performance using Mean Absolute Error (MAE) to measure predictive accuracy.

Features in the Dengue Dataset

1. **City**: Indicates the location of the data, either San Juan (SJ) or Iquitos (IQ).
2. **Year**: The year in which the data was recorded.
3. **Week of the Year**: The specific week within the year when the data was recorded.
4. **Station Maximum Temperature (Kelvin)**: Maximum temperature recorded at the meteorological station.
5. **Station Minimum Temperature (Kelvin)**: Minimum temperature recorded at the meteorological station.
6. **Station Average Temperature (Kelvin)**: Mean temperature recorded at the meteorological station.
7. **Precipitation Amount (mm)**: Total rainfall during the specified period.
8. **NDVI Variables (Normalized Difference Vegetation Index)**:
    - **NDVI_NE**: NDVI for the northeast region.

- **NDVI_NW**: NDVI for the northwest region.
- **NDVI_SE**: NDVI for the southeast region.
- **NDVI_SW**: NDVI for the southwest region.

9. **Reanalysis Data**:
    - **Reanalysis Maximum Air Temperature (Kelvin)**: Highest air temperature observed during the period.
    - **Reanalysis Minimum Air Temperature (Kelvin)**: Lowest air temperature observed during the period.
    - **Reanalysis Dew Point Temperature (Kelvin)**: Dew point temperature, indicative of atmospheric moisture.
    - **Reanalysis Precipitation Amount (mm)**: Modeled estimate of precipitation based on reanalysis data.
10. **Sea-Level Pressure (millibars)**: Pressure measured at sea level.
11. **Relative Humidity (%)**: Measure of moisture content in the air.
12. **Wind Speed (m/s)**:
    - **Station Average Wind Speed**: Average wind speed recorded at the station.
    - **Reanalysis Wind Speed**: Modeled estimate of wind speed based on reanalysis data.
13. **Total Cases**: The target variable representing the number of dengue cases reported.

Mean Absolute Error (MAE)

MAE is a common metric used to evaluate the accuracy of regression models. It measures the average magnitude of errors between the predicted and actual values, without considering their direction (it treats all errors as positive).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Characteristics:

- MAE is always non-negative and a lower value indicates better model performance
- MAE tells us the average amount by which the model's predictions deviate from the actual values.
- Since it is measured in the same units as the target variable, MAE is directly interprebale in terms of the variable being predicted.

Download required packages

The required libraries were loaded, and the datasets were successfully imported. The `dengue_labels_train` dataset contains the target variable, while `dengue_features_train` provides the predictors. Both datasets were checked for dimensions and basic structure, ensuring alignment between features and labels.

```r
# Install all needed libraries if it is not present

if(!require(tidyverse)) install.packages("tidyverse")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(dplyr)) install.packages("dplyr")
if(!require(caret)) install.packages("caret")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(xgboost)) install.packages("xgboost")
```

```
## Loading required package: xgboost
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
if(!require(randomForest)) install.packages("randomForest")
```

```
## Loading required package: randomForest
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
```

```
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
if(!require(kableExtra)) install.packages("kableExtra")
```

```
## Loading required package: kableExtra
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
# Loading all needed libraries

library(tidyverse)
library(ggplot2)
library(dplyr)
library(caret)
library(xgboost)
library(randomForest)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggcorrplot)
library(knitr)
library(kableExtra)
```

Load the Dengue Labels and Features for Training Dataset

```
## Loading the dataset

dengue_labels_train <- read.csv("dengue_labels_train.csv")
dengue_features_train <- read.csv("dengue_features_train.csv")
```

Check Dimensions and Summary of Data

The dataset consists of two main files: **dengue_labels_train** and **dengue_features_train**.

- dengue_labels_train: This file contains 1,456 rows and 4 columns, including the city, year, week of the year, and the total reported dengue cases. This dataset serves as the target variable for predictive analysis.
- dengue_features_train: This file contains 1,456 rows and 24 columns, detailing various environmental and climatic features like temperature, precipitation, and vegetation indices (NDVI) for the same time periods and locations. These features will be used to predict dengue cases.

```
# Check dimensions

dengue_labels_summary <- data.frame("Length" = nrow(dengue_labels_train), "Columns" = ncol(dengue_labels
print(dengue_labels_summary)
```

```
##   Length Columns
## 1   1456       4
```

```
dengue_features_summary <- data.frame("Length" = nrow(dengue_features_train), "Columns" = ncol(dengue_fe
print(dengue_features_summary)
```

```
##   Length Columns
## 1   1456      24
```

View the first few rows of the datasets:

These datasets provide a combination of climatic, temporal, and geographical data that makes this suitable for regression or time-series analysis.

```
head(dengue_features_train) %>% kable(caption="Dengue Features Train")%>% kable_styling(bootstrap_optio
```

city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm
sj	1990	18	1990-04-30	0.1226000	0.1037250	0.1984833	0.1776167	12.42
sj	1990	19	1990-05-07	0.1699000	0.1421750	0.1623571	0.1554857	22.82
sj	1990	20	1990-05-14	0.0322500	0.1729667	0.1572000	0.1708429	34.54
sj	1990	21	1990-05-21	0.1286333	0.2450667	0.2275571	0.2358857	15.36
sj	1990	22	1990-05-28	0.1962000	0.2622000	0.2512000	0.2473400	7.52
sj	1990	23	1990-06-04	NA	0.1748500	0.2543143	0.1817429	9.58

```
head(dengue_labels_train) %>% kable(caption="dengue labels train")%>% kable_styling(bootstrap_options =
```

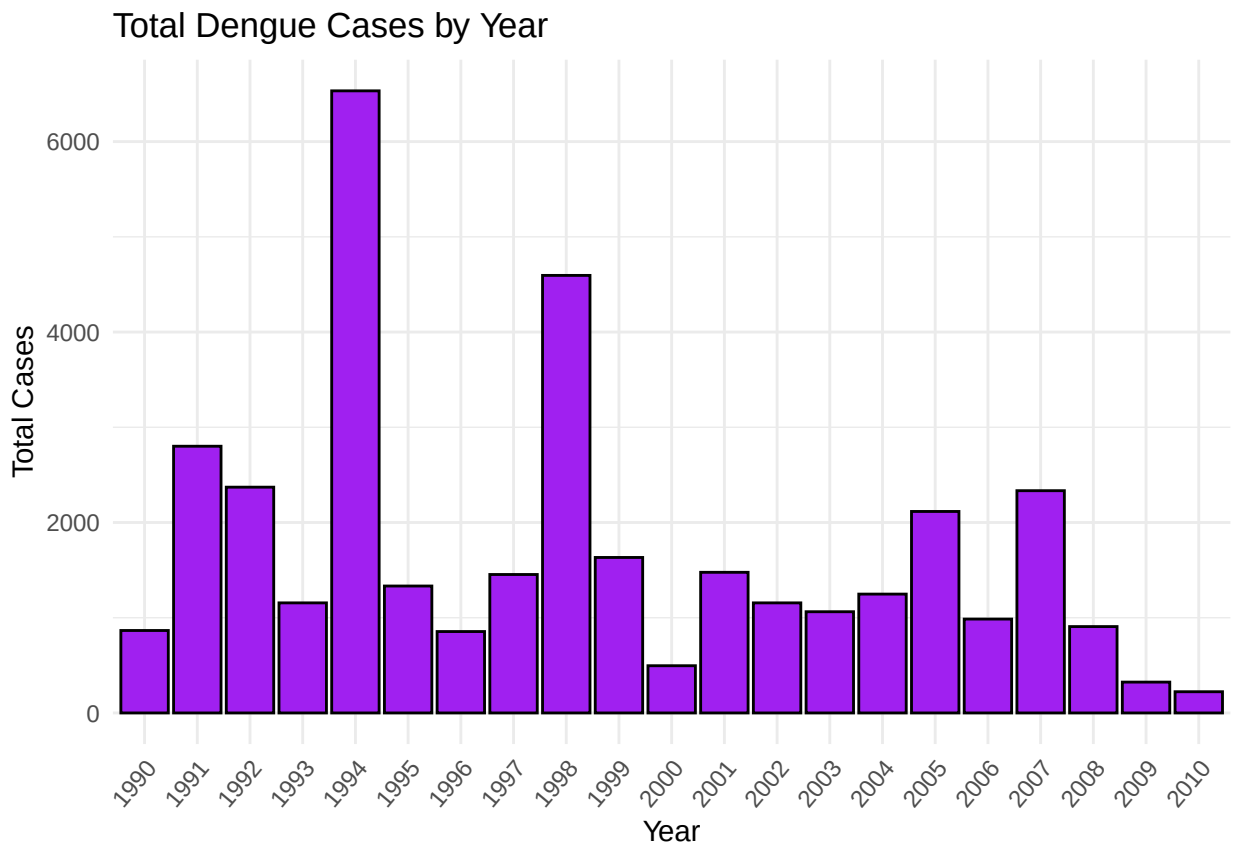
Table 2: dengue labels train

city	year	weekofyear	total_cases
sj	1990	18	4
sj	1990	19	5
sj	1990	20	4
sj	1990	21	3
sj	1990	22	6
sj	1990	23	2

Dengue Cases by Year

This bar chart shows the total dengue cases by year from 1990 to 2010. Certain years, such as those with sharp peaks in 1995 and 1998, indicate major outbreaks. With no consistent upward or downward trend, this variability points to potential environmental or seasonal factors.

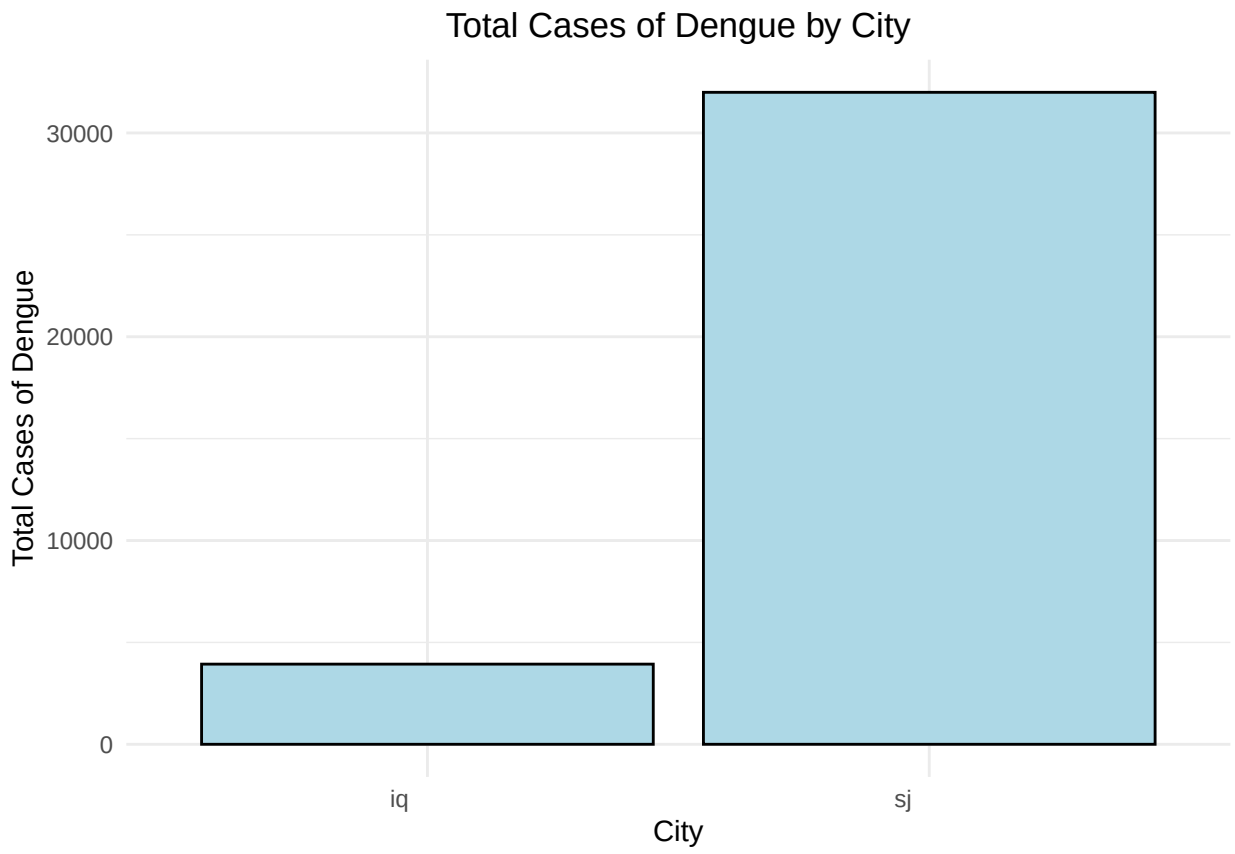
```
# Group by year and aggregate total cases
year_max_cases <- dengue_labels_train %>%
  group_by(year) %>%
  summarise(total_cases = sum(total_cases, na.rm = TRUE))

# Create the bar plot

ggplot(year_max_cases, aes(x = as.factor(year), y = total_cases)) +
  geom_bar(stat = "identity", fill = "purple", color = "black") +
  labs(title = "Total Dengue Cases by Year", x = "Year", y = "Total Cases") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 50, hjust = 1))
```



Dengue Cases by City

The bar chart illustrates the total dengue cases across two cities, "iq" (Iquitos) and "sj" (San Juan). It is evident that San Juan has significantly more dengue cases compared to Iquitos, with totals exceeding 30,000 cases in San Juan versus a substantially lower number in Iquitos. This stark disparity suggests that San Juan might face higher risk factors such as climate, population density, or public health conditions conducive to dengue transmission.

```
# Group by city and aggregate total cases
city_affected <- dengue_labels_train %>%
  group_by(city) %>%
  summarise(total_cases = sum(total_cases, na.rm = TRUE))

# Create the bar plot

ggplot(city_affected, aes(x = city, y = total_cases)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "black") +
  labs(title = "Total Cases of Dengue by City", x = "City", y = "Total Cases of Dengue") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 0, hjust = 2)) +
  theme(plot.title = element_text(hjust = 0.5))  # Center the title
```



Data Preprocessing

Handling Missing Values

The table highlights the number of missing values for specific columns in the dataset. The column `ndvi_ne` has the highest count of missing values with 194, followed by `ndvi_nw` with 52, and both `ndvi_se` and `ndvi_sw` with 22 missing values each.

```
# Missing values by column
missing_values_by_column <- dengue_features_train %>%
  summarise(across(everything(), ~ sum(is.na(.)))) %>%
  pivot_longer(cols = everything(), names_to = "Column", values_to = "MissingCount") %>%
  filter(MissingCount > 0)


head(missing_values_by_column) %>% kable(caption="Missing values by column")%>% kable_styling(bootstrap
```

Table 3: Missing values by column

Column	MissingCount
ndvi_ne	194
ndvi_nw	52
ndvi_se	22

ndvi_sw	22
precipitation_amt_mm	13
reanalysis_air_temp_k	10

```
# Rows with missing values
rows_with_missing <- dengue_features_train %>%
  filter(if_any(everything(), is.na)) %>%
  summarise(Count = n())

print(rows_with_missing)
```

```
##   Count
## 1   257
```

Fill the missing values with mean

The missing values were replaced with the mean of the respective columns using the `mutate` and `across` functions in R. By imputing the missing values with the column-wise mean, the dataset's completeness and suitability for analysis are improved without introducing significant bias. The table shows that no missing values remain.

```
# Fill the missing values with mean
dengue_features_train <- dengue_features_train %>%
  mutate(across(everything(), ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)))

# This shows that the missing values are resolved

head(dengue_features_train) %>% kable(caption="Dengue Features Train") %>% kable_styling(bootstrap_opti
```

city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm
sj	1990	18	1990-04-30	0.1226000	0.1037250	0.1984833	0.1776167	12.42
sj	1990	19	1990-05-07	0.1699000	0.1421750	0.1623571	0.1554857	22.82
sj	1990	20	1990-05-14	0.0322500	0.1729667	0.1572000	0.1708429	34.54
sj	1990	21	1990-05-21	0.1286333	0.2450667	0.2275571	0.2358857	15.36
sj	1990	22	1990-05-28	0.1962000	0.2622000	0.2512000	0.2473400	7.52
sj	1990	23	1990-06-04	0.1422935	0.1748500	0.2543143	0.1817429	9.58

Temperature Conversion

Temperature features were converted from Kelvin to Celsius for better interpretability. These variables are crucial for understanding climatic influences on dengue outbreaks. The updated column names also improve clarity.

```
# Temperature
# Finding columns with "_temp_k"
temperature_kelvin_columns <- grep("_temp_k", names(dengue_features_train), value = TRUE)
print(temperature_kelvin_columns)
```

```
## [1] "reanalysis_air_temp_k"       "reanalysis_avg_temp_k"
## [3] "reanalysis_dew_point_temp_k" "reanalysis_max_air_temp_k"
## [5] "reanalysis_min_air_temp_k"
```

```
# Convert Kelvin to Celsius for the identified columns
dengue_features_train[temperature_kelvin_columns] <- dengue_features_train[temperature_kelvin_columns] -

# Rename columns by replacing '_temp_k' with '_temp_c'
names(dengue_features_train) <- gsub("_temp_k", "_temp_c", names(dengue_features_train))

# View updated column names
print(names(dengue_features_train))
```

```
##  [1] "city"
##  [2] "year"
##  [3] "weekofyear"
##  [4] "week_start_date"
##  [5] "ndvi_ne"
##  [6] "ndvi_nw"
##  [7] "ndvi_se"
##  [8] "ndvi_sw"
##  [9] "precipitation_amt_mm"
## [10] "reanalysis_air_temp_c"
## [11] "reanalysis_avg_temp_c"
## [12] "reanalysis_dew_point_temp_c"
## [13] "reanalysis_max_air_temp_c"
## [14] "reanalysis_min_air_temp_c"
## [15] "reanalysis_precip_amt_kg_per_m2"
## [16] "reanalysis_relative_humidity_percent"
## [17] "reanalysis_sat_precip_amt_mm"
## [18] "reanalysis_specific_humidity_g_per_kg"
## [19] "reanalysis_tdtr_k"
## [20] "station_avg_temp_c"
## [21] "station_diur_temp_rng_c"
## [22] "station_max_temp_c"
## [23] "station_min_temp_c"
## [24] "station_precip_mm"
```

Round to 3 decimal places

Rounding improves data consistency and readability, especially when dealing with continuous variables. This step is essential for visualizations, summaries, and reports, where overly precise values may be unnecessary.

```
# Round all numerical columns to 3 decimal places
dengue_features_train <- dengue_features_train %>%
  mutate(across(where(is.numeric), ~ round(., 3)))

# Display the first few rows of the dataset

head(dengue_features_train) %>% kable(caption="Dengue Features Train") %>% kable_styling(bootstrap_optio
```

city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	rear
sj	1990	18	1990-04-30	0.123	0.104	0.198	0.178	12.42	
sj	1990	19	1990-05-07	0.170	0.142	0.162	0.155	22.82	
sj	1990	20	1990-05-14	0.032	0.173	0.157	0.171	34.54	
sj	1990	21	1990-05-21	0.129	0.245	0.228	0.236	15.36	
sj	1990	22	1990-05-28	0.196	0.262	0.251	0.247	7.52	
sj	1990	23	1990-06-04	0.142	0.175	0.254	0.182	9.58	

Create a column for average for station and analysis

These columns represent the average of maximum and minimum temperatures derived from two different sources: station recordings and reanalysis data. These new variables can be useful for identifying correlations between temperature trends and dengue case patterns.

- **avg_station_max_min**: Average of station_max_temp_c and station_min_temp_c.
- **avg_analysis_max_min**: Average of reanalysis_max_air_temp_c and reanalysis_min_air_temp_c.

```
#Average (Max and min)
# Create a new column 'avg_station_max_min'
dengue_features_train <- dengue_features_train %>%
  mutate(avg_station_max_min = (station_max_temp_c + station_min_temp_c) / 2)

# Display the first few values of the new column
head(dengue_features_train$avg_station_max_min)
```

```
## [1] 24.70 26.95 27.50 28.30 29.45 29.15
```

```
# Create a new column 'avg_analysis_max_min'
dengue_features_train <- dengue_features_train %>%
  mutate(avg_analysis_max_min = (reanalysis_max_air_temp_c + reanalysis_min_air_temp_c) / 2)

# Display the first few values of the new column
head(dengue_features_train$avg_analysis_max_min)
```

```
## [1] 24.70 25.50 25.75 26.05 26.55 27.10
```

Add 'total_cases' column

Adding the **total_cases** column from the dengue_labels_train dataset to the dengue_features_train dataset ensures that the target variable is available within the both datasets. This streamlines analysis and model training processes.

```
# Add the 'total_cases' column from dengue_labels to dengue_features
dengue_features_train$total_cases <- dengue_labels_train$total_cases

# View the first few rows to verify

head(dengue_features_train) %>% kable(caption="Dengue Features Train") %>% kable_styling(bootstrap_opti
```

city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	rear
sj	1990	18	1990-04-30	0.123	0.104	0.198	0.178	12.42	
sj	1990	19	1990-05-07	0.170	0.142	0.162	0.155	22.82	
sj	1990	20	1990-05-14	0.032	0.173	0.157	0.171	34.54	
sj	1990	21	1990-05-21	0.129	0.245	0.228	0.236	15.36	
sj	1990	22	1990-05-28	0.196	0.262	0.251	0.247	7.52	
sj	1990	23	1990-06-04	0.142	0.175	0.254	0.182	9.58	

```
summary_table <- as.list(summary(dengue_features_train$total_cases))

summary_df <- data.frame(
  Statistic = names(summary_table),
  Value = as.numeric(summary_table)
)


# Summary of Total Cases

head(summary_df) %>% kable(caption="Summary of Total Cases") %>% kable_styling(bootstrap_options = c("s
```

Table 7: Summary of Total Cases

Statistic	Value
Min.	0.00000
1st Qu.	5.00000
Median	12.00000
Mean	24.67514
3rd Qu.	28.00000
Max.	461.00000

Correlation Matrix

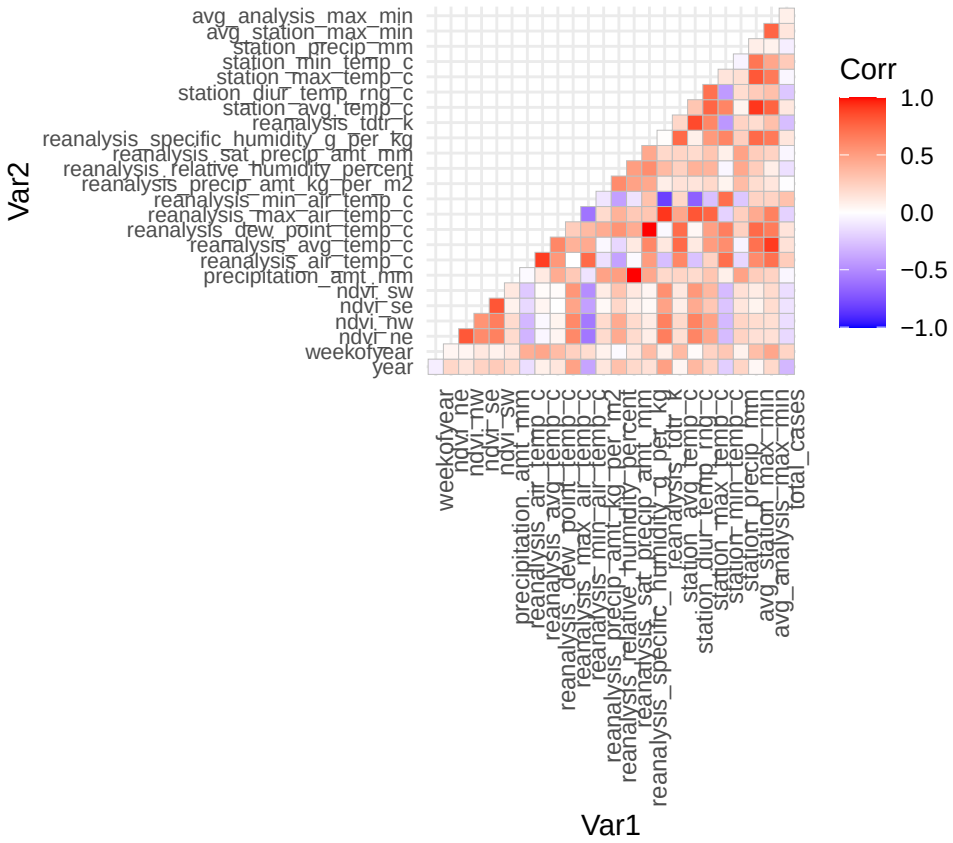
The correlation matrix revealed relationships between various numerical features in the dengue dataset. Strong correlations were observed between certain temperature features and dengue cases, whereas NDVI values showed weaker relationships. This analysis helps identify the features most likely to improve model performance.

```
# CORRELATION MATRIX

# Select only numerical columns
numeric_columns <- dengue_features_train %>%
  select(where(is.numeric))

# Create the correlation matrix
correlation_matrix <- cor(numeric_columns, use = "complete.obs")

# Print the correlation matrix
```

```
head(correlation_matrix) %>% kable(caption="Correlation Matrix") %>% kable_styling(bootstrap_options = c
```

	year	weekofyear	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reana
year	1.0000000	-0.0716490	0.2060195	0.1404616	0.2316619	0.2747632	0.2038558	
weekofyear	-0.0716490	1.0000000	0.0493891	0.0477166	0.1197071	0.0677387	0.1166278	
ndvi_ne	0.2060195	0.0493891	1.0000000	0.8069591	0.5864105	0.6441809	0.1901420	
ndvi_nw	0.1404616	0.0477166	0.8069591	1.0000000	0.5513386	0.6473126	0.1888090	
ndvi_se	0.2316619	0.1197071	0.5864105	0.5513386	1.0000000	0.8211246	0.0742664	
ndvi_sw	0.2747632	0.0677387	0.6441809	0.6473126	0.8211246	1.0000000	0.1234938	

```
# Visualize the correlation matrix
library(ggcorrplot)

# Plot the correlation matrix
ggcorrplot(correlation_matrix,
           method = "square",
           type = "lower",
           lab = TRUE,
           tl.cex = 8,
           tl.srt = 45,
           lab_size = 0, # Too many labels block the visual
           colors = c("blue", "white", "red")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1), # Tilt x-axis text
    axis.text.y = element_text(size = 8) # Adjust y-axis text size

  )
```



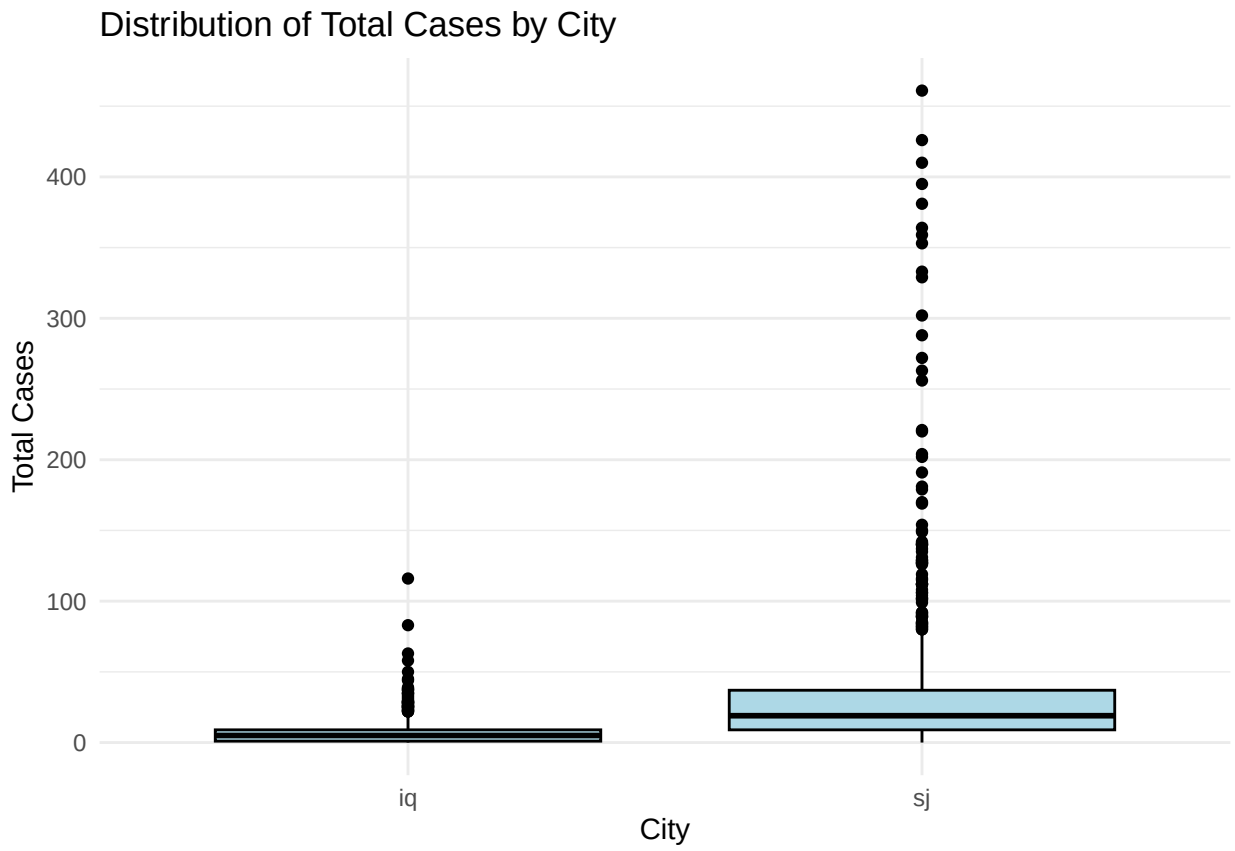
Key Observations:

- **Highly Correlated Variables**:
  - `ndvi_ne` and `ndvi_nw`: **0.81** (Strong positive correlation).
  - `ndvi_se` and `ndvi_sw`: **0.82** (Strong positive correlation).
  - `ndvi_sw` and `ndvi_ne`: **0.64** (Moderate positive correlation).
- **Weak or Negative Correlations**:
  - `station_min_temp_c` shows weak or negative correlations with most variables.
  - Example: `station_min_temp_c` and `ndvi_sw`: **-0.29** (Weak negative correlation).
- **Temperature and Precipitation**:
  - `station_max_temp_c` is moderately correlated with `ndvi_ne` (**0.45**) and `ndvi_nw` (**0.48**), suggesting that vegetation and temperature may share some relationship.
  - `precipitation_amt_mm` shows very weak correlations with other variables, indicating it might have less direct impact on these NDVI values.

Exploratory Data Analysis

Boxplot: Dengue Cases by City

```
# Boxplot of total_cases by city
ggplot(dengue_features_train, aes(x = city, y = total_cases)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(
    title = "Distribution of Total Cases by City",
    x = "City",
    y = "Total Cases"
  ) +
  theme_minimal()
```



General Observations:

1. **Higher Case Volume in SJ:**
   - The distribution of dengue cases in "sj" is significantly larger than in "iq," with a higher median and a broader interquartile range. This indicates that "sj" consistently experiences more frequent and severe outbreaks.
2. **Outliers in SJ:**

- "sj" exhibits numerous extreme outliers, with total cases exceeding 400 in some instances. This suggests that occasional severe outbreaks disproportionately impact "sj."

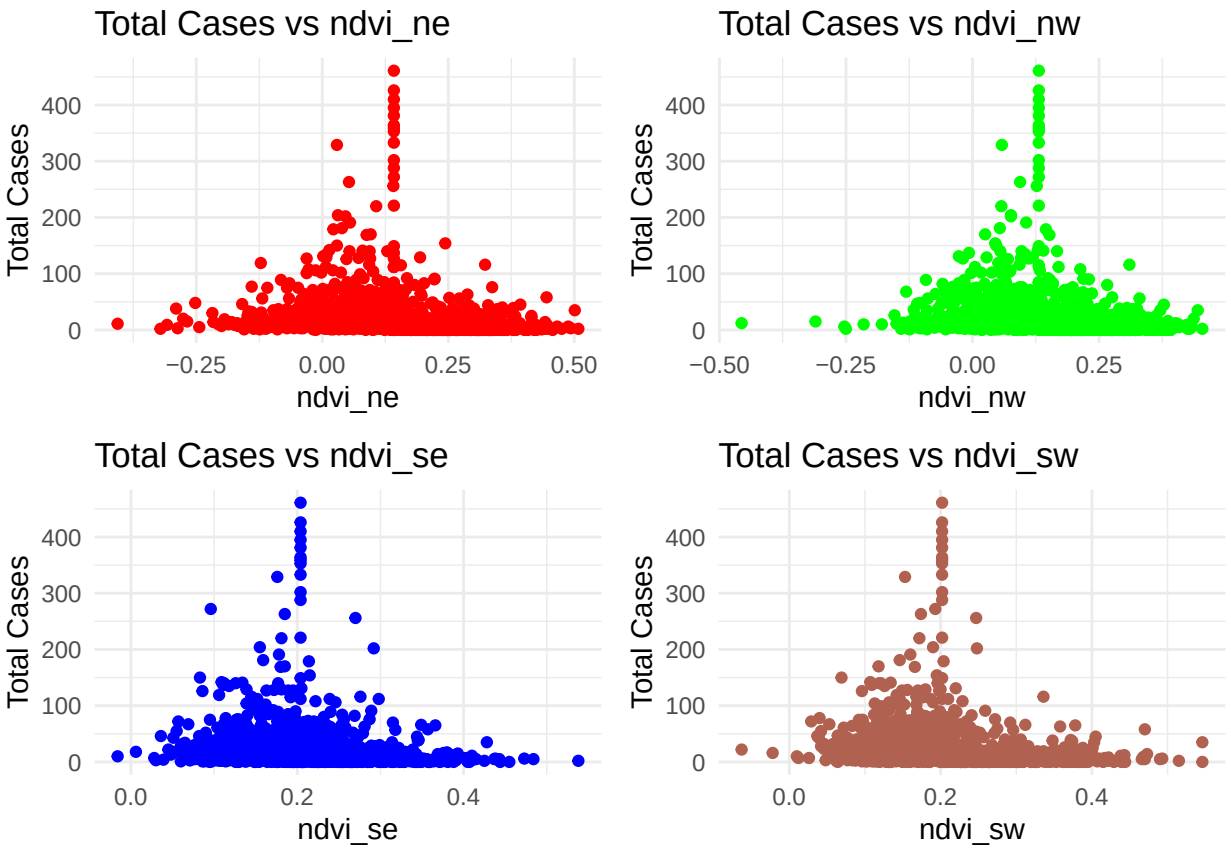
3. **Limited Variability in IQ:**
   - In contrast, "iq" displays a tightly packed distribution with fewer cases and less pronounced variability. The smaller spread suggests a relatively lower and more predictable risk profile for dengue outbreaks.
4. **Implications:**
   - The disparity highlights "sj" as a high-risk area requiring focused public health interventions to manage and mitigate severe dengue outbreaks effectively.

Scatter Plots: Total Cases vs NDVI features

```
# Assuming 'x' is a vector of column names for NDVI features
x <- c("ndvi_ne", "ndvi_nw", "ndvi_se", "ndvi_sw")

# Create individual scatter plots
plot1 <- ggplot(dengue_features_train, aes_string(x = x[1], y = "total_cases")) +
  geom_point(color = "red") +
  labs(title = paste("Total Cases vs", x[1]), x = x[1], y = "Total Cases") +
  theme_minimal()
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
plot2 <- ggplot(dengue_features_train, aes_string(x = x[2], y = "total_cases")) +
  geom_point(color = "green") +
  labs(title = paste("Total Cases vs", x[2]), x = x[2], y = "Total Cases") +
  theme_minimal()

plot3 <- ggplot(dengue_features_train, aes_string(x = x[3], y = "total_cases")) +
  geom_point(color = "blue") +
  labs(title = paste("Total Cases vs", x[3]), x = x[3], y = "Total Cases") +
  theme_minimal()

plot4 <- ggplot(dengue_features_train, aes_string(x = x[4], y = "total_cases")) +
  geom_point(color = "#B16150") +
  labs(title = paste("Total Cases vs", x[4]), x = x[4], y = "Total Cases") +
  theme_minimal()

# Arrange the plots in a 2x2 grid
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```



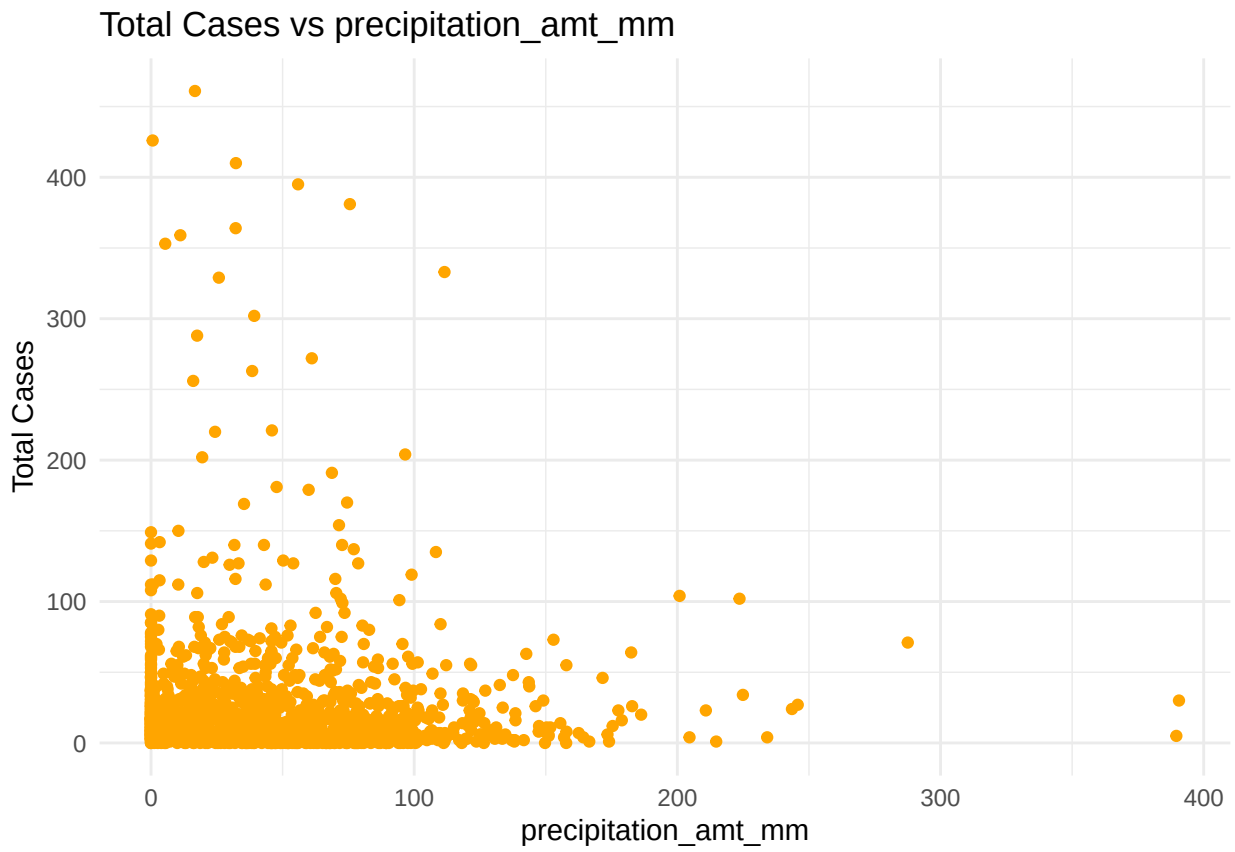
The scatter plots show the relationships between different NDVI indices (plot1: `ndvi_ne`, plot2: `ndvi_nw`, plot3: `ndvi_se`, plot4: `ndvi_sw`) and the total dengue cases.

General Observations:

1. **Centralized Clustering**:
   - Across all plots, most points are clustered around the center of the x-axis (NDVI values near 0), suggesting that areas with moderate vegetation indices are where dengue cases are most concentrated.
2. **Vertical Spread (Total Cases)**:
   - The total cases spread significantly in all plots, with a few points reaching very high values (outliers). This suggests that while NDVI might have some relationship with dengue cases, other factors likely play a significant role.
3. **Symmetry**:
   - The spread of cases seems symmetric around 0 in most NDVI plots, implying no significant difference in total cases between positive and negative NDVI values.

Scatter Plot: Total Cases vs Precipitation Amount

```r
# total cases vs precipitation
# Assuming 'x' is a vector containing column names

x <- c("ndvi_ne", "ndvi_nw", "ndvi_se", "ndvi_sw", "precipitation_amt_mm")

# Create a scatter plot
ggplot(dengue_features_train, aes_string(x = x[5], y = "total_cases")) +
  geom_point(color = "orange") +
  labs(title = paste("Total Cases vs", x[5]),
       x = x[5],
       y = "Total Cases") +
  theme_minimal() +
  theme(legend.position = "none")
```



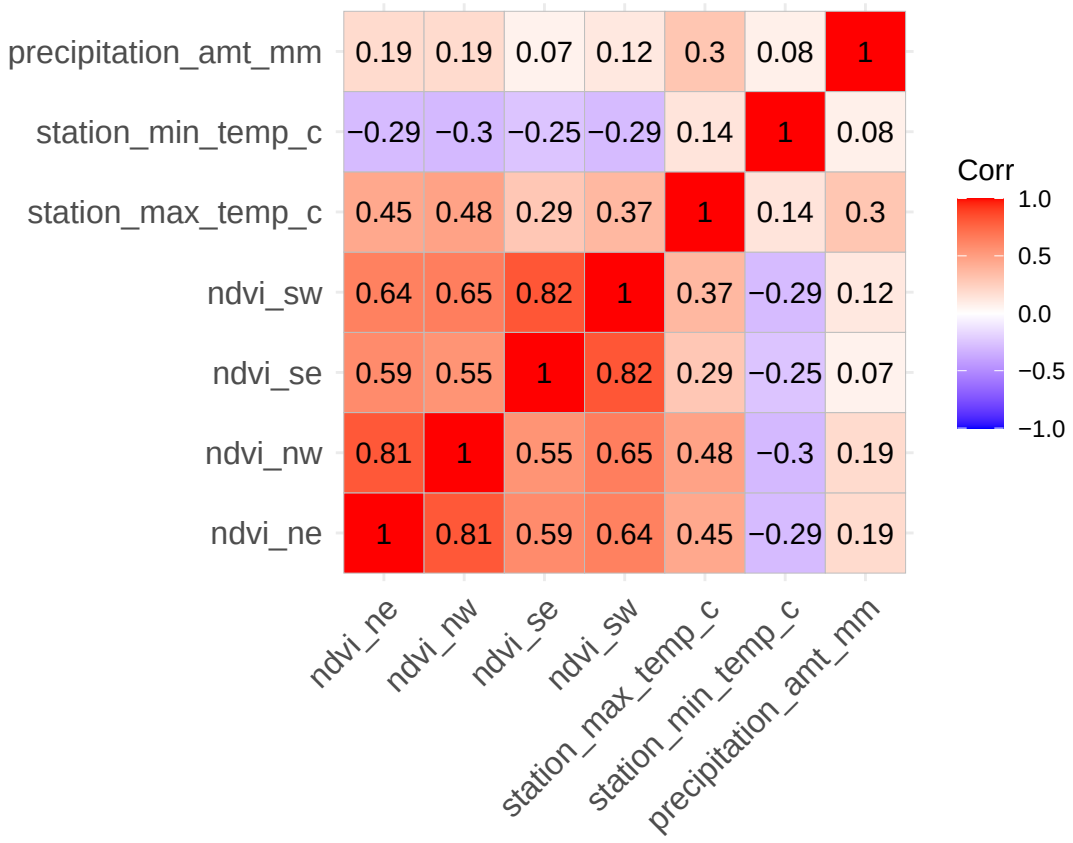
Observations:

1. **High Density Near Zero Precipitation**:
   - Most data points are clustered near the lower end of the x-axis (precipitation close to 0), with a wide spread of dengue cases. This suggests that dengue cases occur even with minimal rainfall.
2. **Decrease in Density with Higher Precipitation**:
   - As precipitation increases, the number of data points decreases. This implies that areas or time periods with high rainfall are less common in the dataset.

3. **Extreme Cases**:
    - A few points with very high dengue cases (e.g., above 300) are scattered across the precipitation range. These could be outliers or special cases, such as extreme weather conditions.
4. **Lack of a Clear Trend**:
    - The plot doesn't show a strong linear or non-linear relationship between precipitation and dengue cases. While some cases are associated with moderate rainfall, others occur at both low and high precipitation levels.

Correlation Matrix for selected features

```
# Select numerical variables
numeric_vars <- dengue_features_train %>%
  select(ndvi_ne, ndvi_nw, ndvi_se, ndvi_sw, station_max_temp_c, station_min_temp_c, precipitation_amt_r

# Compute correlation matrix
cor_matrix <- cor(numeric_vars, use = "complete.obs")

# Visualize the correlation matrix
library(ggcorrplot)
ggcorrplot(cor_matrix, method = "square", lab = TRUE)
```



Key Observations:

- **Strong Positive Correlations Among NDVI Features:**
  - NDVI (Normalized Difference Vegetation Index) variables (e.g., `ndvi_ne`, `ndvi_nw`, `ndvi_se`, and `ndvi_sw`) exhibit strong positive correlations with one another (above 0.8). This indicates that vegetation indices across different regions tend to move together, likely due to shared environmental conditions.
- **Moderate Correlation Between Station Maximum Temperature and NDVI:**
  - There is a moderate positive correlation (around 0.4–0.5) between `station_max_temp_c` and NDVI variables. This suggests that higher maximum temperatures are associated with greener vegetation, which aligns with the idea that warmer climates often foster vegetation growth.
- **Weak Relationships with Precipitation and Minimum Temperature:**
  - Precipitation (`precipitation_amt_mm`) and minimum temperature (`station_min_temp_c`) show weak or negative correlations with most variables. This implies that these factors are less directly linked to vegetation or maximum temperature patterns in this dataset.
- **Implications for Modeling:**
  - The strong correlations among NDVI variables might lead to multicollinearity issues in regression models. Dimensionality reduction techniques, like PCA, could help. Additionally, the weak link between precipitation and NDVI suggests that precipitation may not be a primary driver of vegetation or dengue cases, which could guide feature selection in predictive models.

Feature Engineering

The dengue cases of San Juan sj and Iquitos iq are not dependent, so we need to split them into different data frames for individual analysis.

Key Steps:

1. Splitting the dataset: The dataset is divided into X_sj for San Juan and X_iq for Iquitos
2. Dropping irrelevant columns: Columns like city, weekofyear, and week_start_date are removed because they are not predictive for the analysis and to reduce redundancy.
3. One-hot encoding for the Year Column: The year column is transformed into multiple binary columns, representing each year as a separate feature.
4. Transformed datasets: Ensure they are independent and better suited for ML algorithms.

This process ensures that the feature sets for the two cities are tailored to their respective data, which improves model accuracy and interpretability. Handling collinearity and removing irrelevant columns help models better capture patterns without being influenced by redundant or non-informative data.

```
# FEATURE ENGINEERING
# Split the data by city
X_sj <- dengue_features_train[dengue_features_train$city == 'sj', ]
X_iq <- dengue_features_train[dengue_features_train$city == 'iq', ]

# Drop specific columns
```

```r
X_sj <- X_sj[, !(names(X_sj) %in% c('city', 'weekofyear', 'week_start_date'))]
X_iq <- X_iq[, !(names(X_iq) %in% c('city', 'weekofyear', 'week_start_date'))]

# One-hot encode the 'year' column
one_hot_sj <- model.matrix(~ year - 1, data = X_sj)
X_sj <- X_sj[, !names(X_sj) %in% 'year']
X_sj <- cbind(X_sj, one_hot_sj)

one_hot_iq <- model.matrix(~ year - 1, data = X_iq)
X_iq <- X_iq[, !names(X_iq) %in% 'year']
X_iq <- cbind(X_iq, one_hot_iq)

# View the transformed datasets

head(X_sj) %>% kable(caption="X San Juan") %>% kable_styling(bootstrap_options = c("striped", "hover",
```

ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_c	reanalysis_avg_temp_c
0.123	0.104	0.198	0.178	12.42	24.423	24.593
0.170	0.142	0.162	0.155	22.82	25.061	25.293
0.032	0.173	0.157	0.171	34.54	25.631	25.729
0.129	0.245	0.228	0.236	15.36	25.837	26.079
0.196	0.262	0.251	0.247	7.52	26.369	26.514
0.142	0.175	0.254	0.182	9.58	26.480	26.614

```r
head(X_iq) %>% kable(caption="X Iquitos") %>% kable_styling(bootstrap_options = c("striped", "hover", "
```

	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_c	reanalysis_avg_te
937	0.193	0.132	0.341	0.247	25.41	23.590	
938	0.217	0.276	0.289	0.242	60.61	23.484	
939	0.177	0.173	0.204	0.128	55.52	23.266	
940	0.228	0.145	0.254	0.200	5.60	22.207	
941	0.329	0.322	0.254	0.361	62.76	23.283	
942	0.206	0.191	0.232	0.255	16.24	24.041	

Select columns and rearrange dataset

Key Steps:

1. Reordering Columns by Year: Columns related to the year are selected and reordered to ensure the chronological data is organized in the datasets.
2. Rearranging Datasets: Datasets are restructured by placing the year-related columns at the beginning. This makes the datsets cleaner and more consistent for time-series analysis.
3. Transformed Datasets: The first few rows are displayed for verification.

X_sj

```
# Select columns related to years (assuming years are the last columns in reverse order)
years <- rev(names(X_sj)[24:length(names(X_sj))])

# Subset and rearrange the dataset
X_sj <- cbind(X_sj[, years], X_sj[, 1:24])

# Display the first few rows

head(X_sj) %>% kable(caption="X San Juan") %>% kable_styling(bootstrap_options = c("striped", "hover",
```

X_sj[, years]	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_c	reanalysis
1990	0.123	0.104	0.198	0.178	12.42	24.423	
1990	0.170	0.142	0.162	0.155	22.82	25.061	
1990	0.032	0.173	0.157	0.171	34.54	25.631	
1990	0.129	0.245	0.228	0.236	15.36	25.837	
1990	0.196	0.262	0.251	0.247	7.52	26.369	
1990	0.142	0.175	0.254	0.182	9.58	26.480	

X_iq

```
# Select columns related to years (assuming years are the last columns in reverse order)
years <- rev(names(X_iq)[24:length(names(X_iq))])

# Subset and rearrange the dataset
X_iq <- cbind(X_iq[, years], X_iq[, 1:24])

# Display the first few rows

head(X_iq) %>% kable(caption="X Iquitos") %>% kable_styling(bootstrap_options = c("striped", "hover", "
```

	X_iq[, years]	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	reanalysis_air_temp_c	rea
937	2000	0.193	0.132	0.341	0.247	25.41	23.590	
938	2000	0.217	0.276	0.289	0.242	60.61	23.484	
939	2000	0.177	0.173	0.204	0.128	55.52	23.266	
940	2000	0.228	0.145	0.254	0.200	5.60	22.207	
941	2000	0.329	0.322	0.254	0.361	62.76	23.283	
942	2000	0.206	0.191	0.232	0.255	16.24	24.041	

Removing 1990 and 2000 year columns

The removal of the first year in each dataset (1990 for San Juan and 2000 for Iquitos) help avoid the dummy variable trap and avoid redundancy in one-hot encoded columns while retaining all relevant information for

predictive modeling.

```
# Removing dummy variable trap (dropping specific year columns)
X_sj <- X_sj[, !(names(X_sj) %in% "1990")]
X_iq <- X_iq[, !(names(X_iq) %in% "2000")]

# Convert the data frames to arrays (matrices in R)
x_sj_arr <- as.matrix(X_sj)
x_iq_arr <- as.matrix(X_iq)
```

Split train-test

A test ratio of 20% is chosen, meaning 80% of the data will be used for training, and 20% will be reserved for validation or testing. The createDataPartition function ensures a stratified split, preserving the distribution of the target variable (total_cases).

```
# Define train-test split ratio
test_ratio <- 0.2

# Create train-test splits for SJ data
set.seed(23)  # For reproducibility
sj_indices <- createDataPartition(x_sj_arr[, ncol(x_sj_arr)], p = 1 - test_ratio, list = FALSE)

X_sj_train <- x_sj_arr[sj_indices, -ncol(x_sj_arr)]
y_sj_train <- x_sj_arr[sj_indices, ncol(x_sj_arr)]

X_sj_test <- x_sj_arr[-sj_indices, -ncol(x_sj_arr)]
y_sj_test <- x_sj_arr[-sj_indices, ncol(x_sj_arr)]

# Create train-test splits for IQ data
iq_indices <- createDataPartition(x_iq_arr[, ncol(x_iq_arr)], p = 1 - test_ratio, list = FALSE)

X_iq_train <- x_iq_arr[iq_indices, -ncol(x_iq_arr)]
y_iq_train <- x_iq_arr[iq_indices, ncol(x_iq_arr)]

X_iq_test <- x_iq_arr[-iq_indices, -ncol(x_iq_arr)]
y_iq_test <- x_iq_arr[-iq_indices, ncol(x_iq_arr)]
```

Feature Scaling

Data standardization ensures all features have a mean of 0 and a standard deviation of 1. This is especially important for models sensitive to feature scales, such as k-NN and SVM.

```
# Feature Scaling for SJ
sj_scaler <- preProcess(X_sj_train, method = c("center", "scale"))

# Apply scaling to training and test sets
X_sj_train <- predict(sj_scaler, X_sj_train)
X_sj_test <- predict(sj_scaler, X_sj_test)
```

```
# Feature Scaling for IQ
iq_scaler <- preProcess(X_iq_train, method = c("center", "scale"))

# Apply scaling to training and test sets
X_iq_train <- predict(iq_scaler, X_iq_train)
X_iq_test <- predict(iq_scaler, X_iq_test)
```

Normalization of dengue_features_train

```
# Verify normalization

head(dengue_features_train) %>% kable(caption="Dengue Features Train") %>% kable_styling(bootstrap_optic
```

city	year	weekofyear	week_start_date	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm	rear
sj	1990	18	1990-04-30	0.123	0.104	0.198	0.178	12.42	
sj	1990	19	1990-05-07	0.170	0.142	0.162	0.155	22.82	
sj	1990	20	1990-05-14	0.032	0.173	0.157	0.171	34.54	
sj	1990	21	1990-05-21	0.129	0.245	0.228	0.236	15.36	
sj	1990	22	1990-05-28	0.196	0.262	0.251	0.247	7.52	
sj	1990	23	1990-06-04	0.142	0.175	0.254	0.182	9.58	

```
summary_df <- as.data.frame(t(summary(dengue_features_train)))

head(summary_df) %>% kable(caption="Summary Statistics of Dengue Features Dataset") %>% kable_styling(bo
                position = "center")
```

Table 14: Summary Statistics of Dengue Features Dataset

Var1	Var2	Freq
city		Length:1456
year		Min. :1990
weekofyear		Min. : 1.00
week_start_date		Length:1456
ndvi_ne		Min. :-0.40600
ndvi_nw		Min. :-0.4560

Scatter Plot: Individual features vs target variable for San Juan

The scatter plots represent the relationship between individual features (1 through 24) and the target variable (year, serving as a proxy for the dengue cases timeline). Key observations include:

1. **Feature 1** shows a strong linear relationship with the target, indicating its potential importance in modeling trends over time.

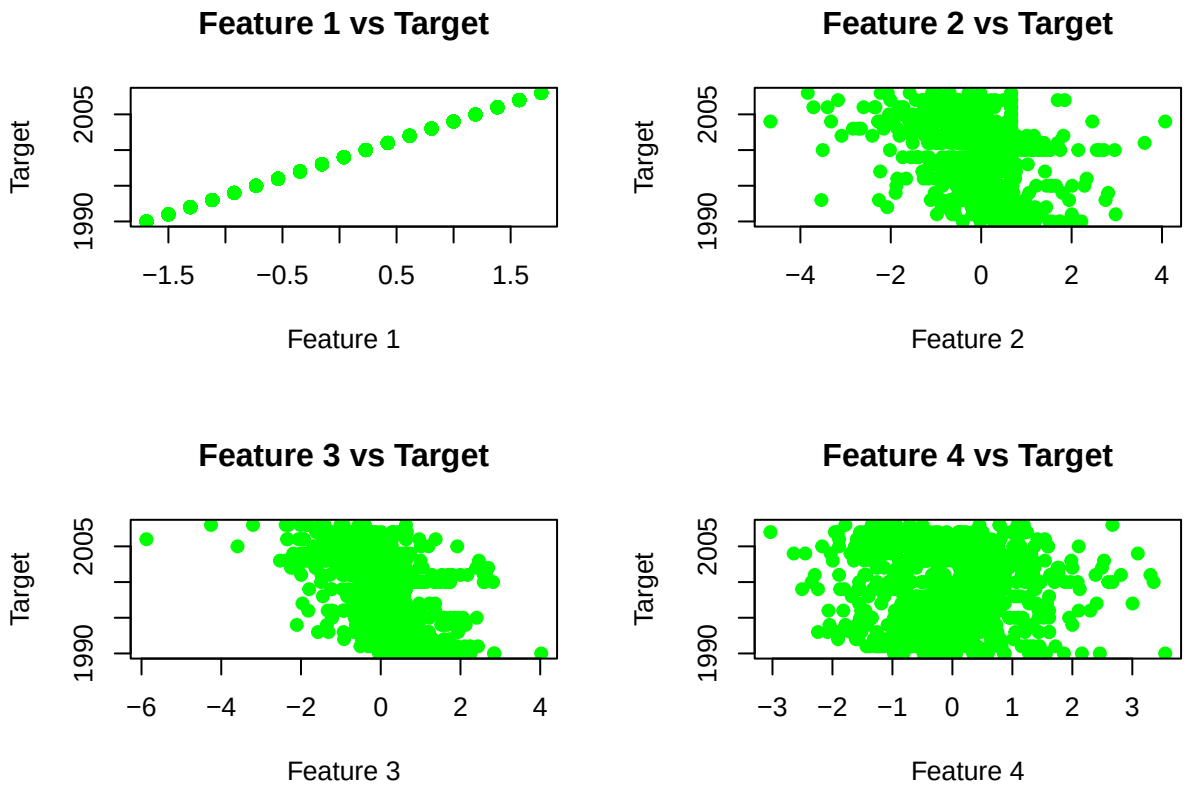
2. Other features, such as **Feature 21** and **Feature 6**, display a moderate spread with noticeable clustering, suggesting some correlation with the target variable.

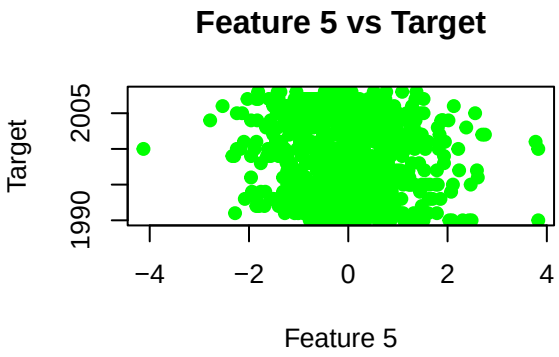
3. Many features, including **Feature 3**, **Feature 13**, and others, exhibit a more uniform or noisy distribution, indicating weak or no correlation with the target variable.

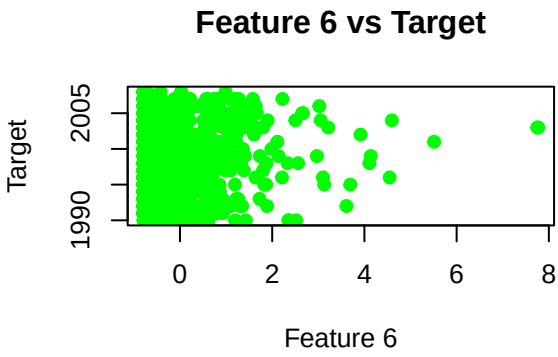
4. Features like **Feature 19** and **Feature 20** show distinct patterns (e.g., band-like structures), which might suggest discrete or categorical influences.

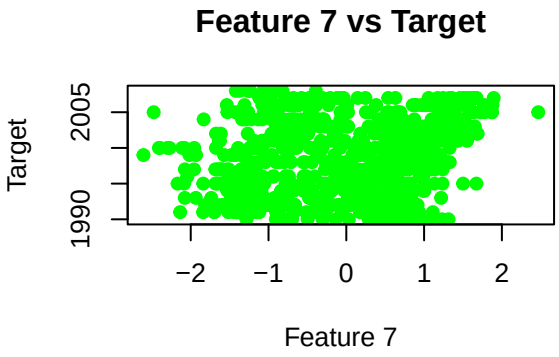
These plots are valuable for identifying which features may be most relevant in predicting dengue case trends and guiding feature selection or transformation during the modeling process.

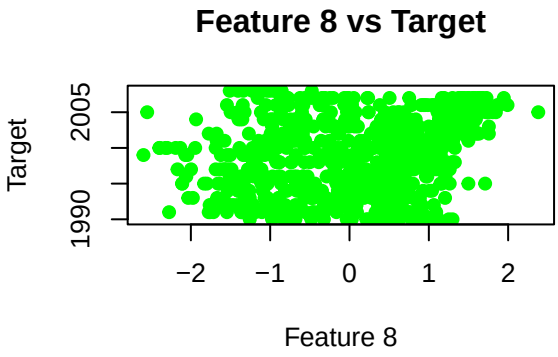
```r
#Scatter plot
# Loop through all features and create individual plots
par(mfrow = c(2, 2))  # Set up a 2x2 grid for multiple plots
for (i in 1:ncol(X_sj_train)) {
  plot(X_sj_train[, i], y_sj_train, pch = 19, col = "green",
       xlab = paste("Feature", i), ylab = "Target", main = paste("Feature", i, "vs Target"))
}
```

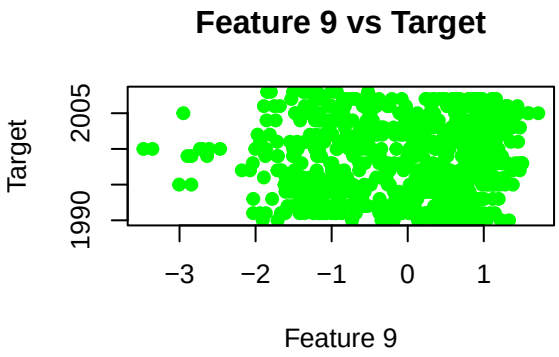


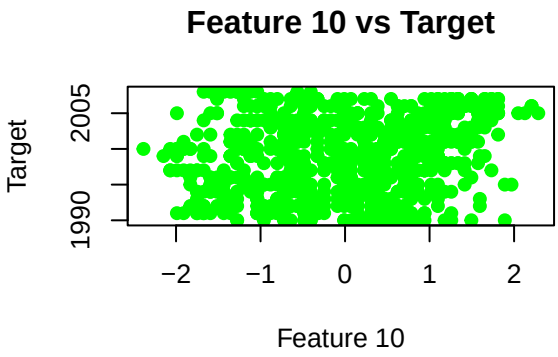

Feature 5 vs Target
Target
1990
2005
−4
−2
0
2
4
Feature 5

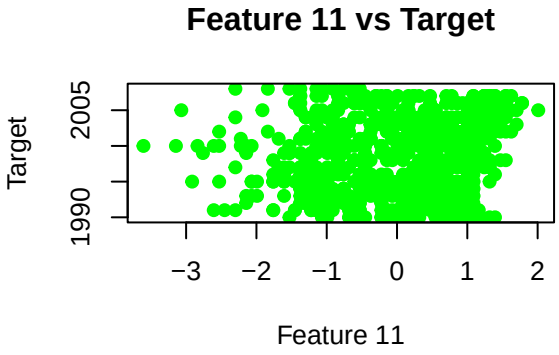

Feature 6 vs Target
Target
1990
2005
0
2
4
6
8
Feature 6

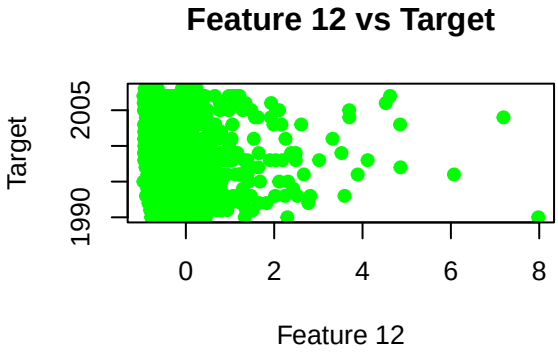

Feature 7 vs Target
Target
1990
2005
−2
−1
0
1
2
Feature 7

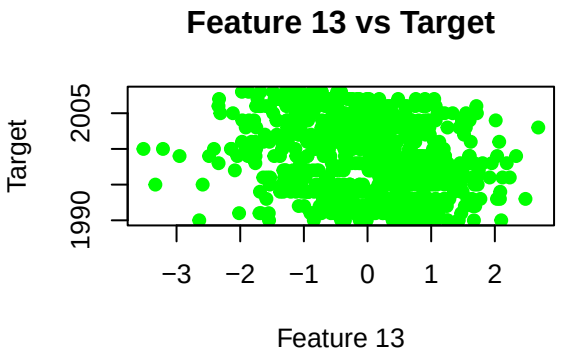

Feature 8 vs Target
Target
1990
2005
−2
−1
0
1
2
Feature 8

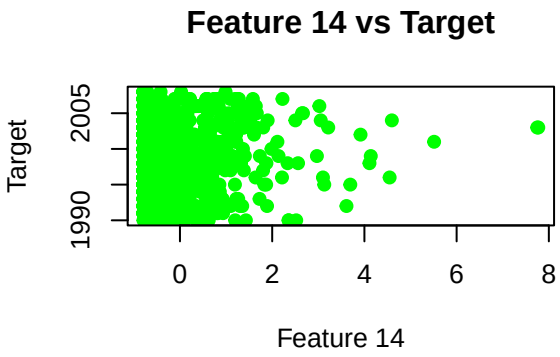

Feature 9 vs Target
Target
2005
1990
−3
−2
−1
0
1
Feature 9

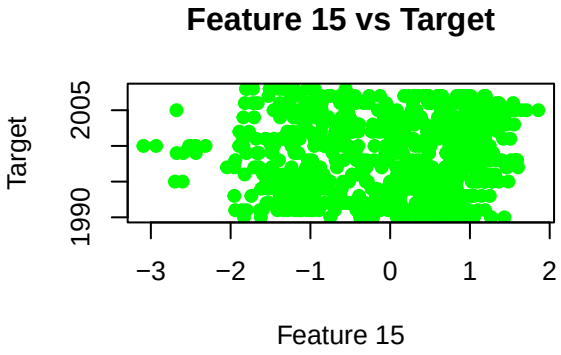

Feature 10 vs Target
Target
2005
1990
−2
−1
0
1
2
Feature 10

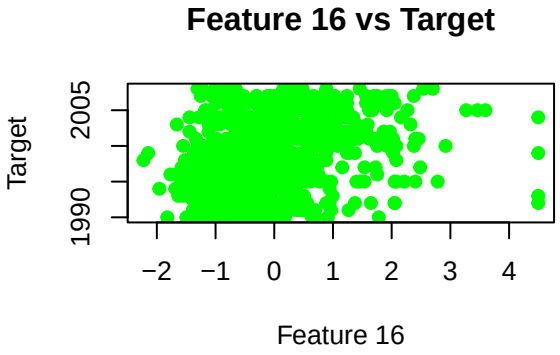

Feature 11 vs Target
Target
2005
1990
−3
−2
−1
0
1
2
Feature 11

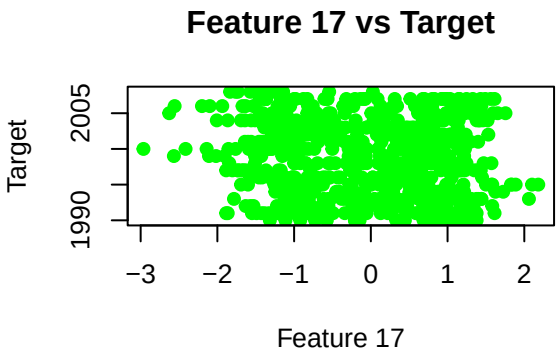

Feature 12 vs Target
Target
2005
1990
0
2
4
6
8
Feature 12

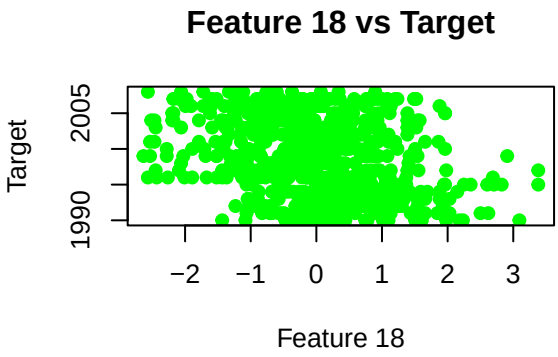

Feature 13 vs Target
Target
2005
1990
−3
−2
−1
0
1
2
Feature 13

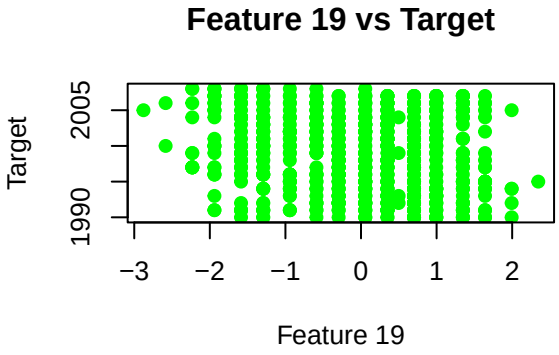

Feature 14 vs Target
Target
2005
1990
0
2
4
6
8
Feature 14

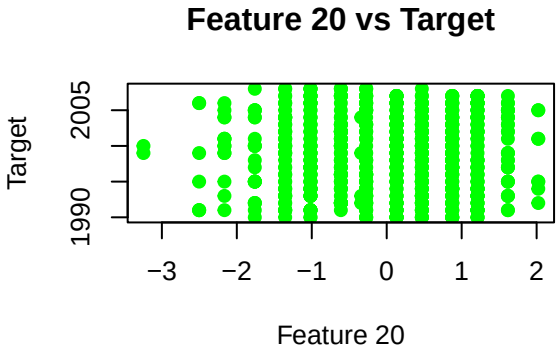

Feature 15 vs Target
Target
2005
1990
−3
−2
−1
0
1
2
Feature 15

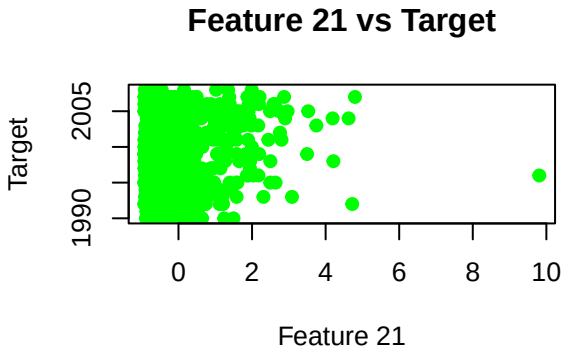

Feature 16 vs Target
Target
2005
1990
−2
−1
0
1
2
3
4
Feature 16

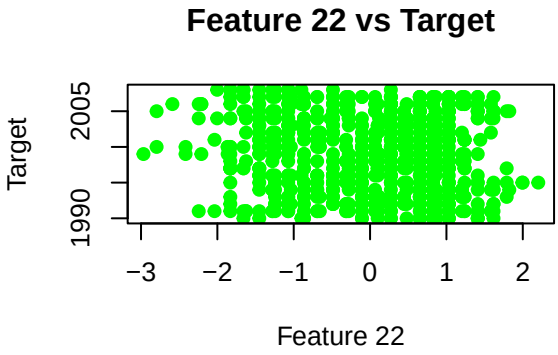

Feature 17 vs Target
Target
2005
1990
−3
−2
−1
0
1
2
Feature 17

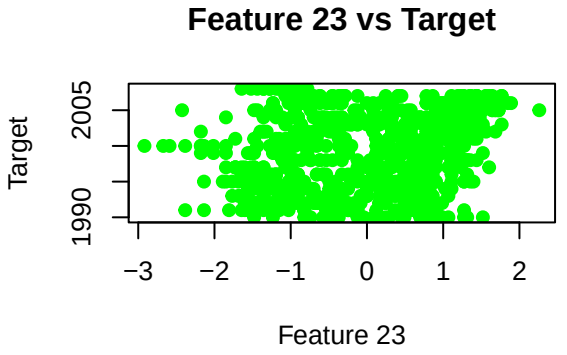

Feature 18 vs Target
Target
2005
1990
−2
−1
0
1
2
3
Feature 18

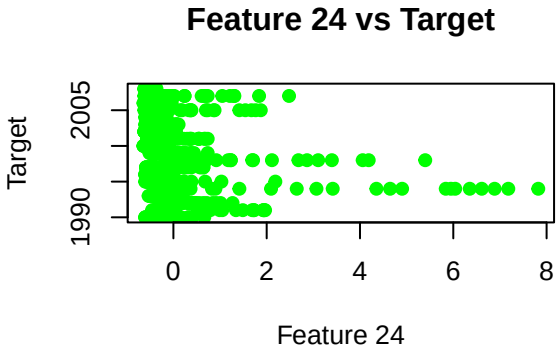

Feature 19 vs Target
Target
2005
1990
−3
−2
−1
0
1
2
Feature 19


Feature 20 vs Target
Target
2005
1990
−3
−2
−1
0
1
2
Feature 20










```r
par(mfrow = c(1, 1))  # Reset plotting layout
```

Random Forest Modeling

Steps:

1. **Hyperparameter Tuning:** Grid search with cross-validation was employed to optimize critical hyperparameters such as the number of variables tried at each split (`mtry`), the splitting rule (e.g., "variance"), and the minimum node size. The model was tuned for minimizing Mean Absolute Error (MAE).

2. **Model Training:** The Random Forest model was trained on the San Juan dataset using the training set. The model leverages ensemble learning to build multiple decision trees and combines their predictions for robust results.

3. **Prediction:** Using the optimal hyperparameters, predictions were made on the test set, estimating dengue cases based on the trained Random Forest model.

4. **Performance Metrics (MAE):** The model's accuracy was evaluated using Mean Absolute Error, which measures the average magnitude of prediction errors, providing insights into the model's prediction reliability.

Define hyperparameters

```
# Define for Random Forest

# Define training control with cross-validation
train_control <- trainControl(
  method = "cv",
  number = 5
)

# Define the hyperparameter grid
tune_grid <- expand.grid(
  mtry = c(2, 4, 6, 8),
  splitrule = c("variance"),
  min.node.size = c(1, 5, 10)
)
```

Random Forest for San Juan

```
# Random Forest for SJ

# Train the Random Forest model using grid search with cross-validation
sj_rf <- train(
  x = X_sj_train,
  y = y_sj_train,
  method = "ranger",
  trControl = train_control,
  tuneGrid = tune_grid
)

# Display the best hyperparameters
print(sj_rf$bestTune)
```

```
##    mtry splitrule min.node.size
## 10    8  variance             1
```

```
# Predict on the test set
sj_rf_pred <- predict(sj_rf, X_sj_test)

# Calculate Mean Absolute Error (MAE)
mae_sj_rf <- mean(abs(sj_rf_pred - y_sj_test))
cat("MAE:", mae_sj_rf, "\n")
```

```
## MAE: 0.437086
```

Random Forest for Iquitos

```r
### Random Forest for Iq

# Random Forest for IQ

# Train the Random Forest model using grid search with cross-validation
iq_rf <- train(
  x = X_iq_train,
  y = y_iq_train,
  method = "ranger",
  trControl = train_control,
  tuneGrid = tune_grid
)

# Display the best hyperparameters
print(iq_rf$bestTune)
```

```
##    mtry splitrule min.node.size
## 11    8  variance             5
```

```r
# Predict on the test set
iq_rf_pred <- predict(iq_rf, X_iq_test)

# Calculate Mean Absolute Error (MAE)
mae_iq_rf <- mean(abs(iq_rf_pred - y_iq_test))
cat("MAE:", mae_iq_rf, "\n")
```

```
## MAE: 0.3705987
```

k-NN Modeling

Steps:

1. **Hyperparameter Tuning:** Grid search with cross-validation was used to optimize key parameters such as the number of neighbors (k), algorithm (e.g., "ball_tree," "kd_tree," "brute"), and leaf size. The model was tuned for minimizing Mean Absolute Error (MAE).
2. **Model Training:** The k-Nearest Neighbors (k-NN) model was trained on the San Juan dataset, utilizing the training set to identify dengue case patterns.
3. **Prediction:** After selecting the best hyperparameters, the trained k-NN model was applied to the test set to predict dengue cases.
4. **Performance Metrics (MAE):** The best hyperparameters and corresponding MAE were extracted and evaluated, demonstrating the model's ability to make accurate predictions.

```r
# k-NN Model

# Define & Tune Hyper-parameters for k-NN
knn_grid <- expand.grid(
  k = c(3, 5, 7),           # Number of neighbors to try
  algorithm = c("ball_tree", "kd_tree", "brute"),  # Algorithms (manually incorporate logic later if ne
```

```
  leaf_size = c(25)          # Leaf size (applicable only for certain algorithms)
)
```

k-NN for San Juan

```
# Train the k-NN model with cross-validation for SJ
sj_knn_grid_cv <- train(
  X_sj_train, y_sj_train,
  method = "knn",
  trControl = trainControl(
    method = "cv",  # Cross-validation
    number = 5,     # Number of folds
    summaryFunction = defaultSummary  # Include metrics like MAE
  ),
  # tuneGrid = knn_grid,
  metric = "MAE"  # Optimize for Mean Absolute Error
)

# Extract best score (MAE)
best_score <- min(sj_knn_grid_cv$results$MAE)

# Extract best parameters
best_params <- sj_knn_grid_cv$bestTune

# Print results
print(best_score)
```

```
## [1] 1.715535
```

```
print(best_params)
```

```
##   k
## 1 5
```

```
# Predict on the test set
y_sj_pred <- predict(sj_knn_grid_cv$finalModel, X_sj_test)


# Calculate Mean Absolute Error (MAE)
mae_sj_knn <- mean(abs(y_sj_test - y_sj_pred))

# Print MAE
print(mae_sj_knn)
```

```
## [1] 1.755735
```

```
# Overview of the Model: k-NN for SJ
sj_knn_grid_cv
```

```
## k-Nearest Neighbors
##
## 750 samples
##  24 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 600, 601, 600, 599, 600
## Resampling results across tuning parameters:
##
##   k  RMSE      Rsquared   MAE
##   5  2.146043  0.8401278  1.715535
##   7  2.142910  0.8474323  1.730857
##   9  2.174091  0.8477397  1.740108
##
## MAE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

k-NN for Iquitos

```
# Train the k-NN model with cross-validation for IQ
iq_knn_grid_cv <- train(
  X_iq_train, y_iq_train,
  method = "knn",
  trControl = trainControl(
    method = "cv",  # Cross-validation
    number = 5,     # Number of folds
    summaryFunction = defaultSummary  # Include metrics like MAE
  ),
  tuneGrid = expand.grid(k = c(3, 5, 7)),  # Only tuning over k
  metric = "MAE"  # Optimize for Mean Absolute Error
)


# Extract best score (MAE)
best_score <- min(iq_knn_grid_cv$results$MAE)

# Extract best parameters
best_params <- iq_knn_grid_cv$bestTune

# Print results
print(best_score)
```

```
## [1] 1.253834
```

```
print(best_params)
```

```
##   k
## 1 3
```

```r
# Predict on the test set
y_iq_pred <- predict(iq_knn_grid_cv$finalModel, X_iq_test)

# Calculate Mean Absolute Error (MAE)
mae_iq_knn <- mean(abs(y_iq_test - y_iq_pred))

# Print MAE
print(mae_iq_knn)
```

```
## [1] 1.346278
```

```r
iq_knn_grid_cv
```

```
## k-Nearest Neighbors
##
## 417 samples
##  24 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 333, 334, 333, 334, 334
## Resampling results across tuning parameters:
##
##   k  RMSE      Rsquared   MAE
##   3  1.620303  0.6891023  1.253834
##   5  1.579458  0.7175550  1.280266
##   7  1.532337  0.7533292  1.259128
##
## MAE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 3.
```

XGBoost Modeling

Steps:

1. **Data Preparation:** The training and test datasets for the San Juan region were converted into matrix format. These matrices were further transformed into XGBoost's `DMatrix` format, which optimizes computation during training and prediction.
2. **Hyperparameter Tuning:** Training was conducted with parameters such as learning rate (`eta` = 0.1), maximum tree depth (`max_depth` = 6), subsampling ratios for rows (`subsample` = 0.8) and columns (`colsample_bytree` = 0.8), all tailored to optimize regression performance.
3. **Model Training:** The XGBoost model was trained using 100 boosting rounds with a watchlist that monitored both training and test datasets to prevent overfitting. Early stopping was applied to halt training if the test performance did not improve for 10 consecutive rounds.
4. **Prediction:** The trained XGBoost model was used to predict dengue cases on the test set.
5. **Performance Metrics (MAE):** Model accuracy was assessed using Mean Absolute Error, capturing the model's ability to predict dengue cases accurately and effectively.

Format data and define parameters

```
# XGBOOST MODEL

# Convert the data to matrices
sj_dtrain <- as.matrix(X_sj_train)
sj_dtest <- as.matrix(X_sj_test)

iq_dtrain <- as.matrix(X_iq_train)
iq_dtest <- as.matrix(X_iq_test)


# Define training parameters for xgboost
xgb_params <- list(
  objective = "reg:squarederror",  # Regression task
  eta = 0.1,                       # Learning rate
  max_depth = 6,                   # Maximum tree depth
  subsample = 0.8,                 # Subsample ratio of training data
  colsample_bytree = 0.8           # Subsample ratio of columns
)

# Convert training data to DMatrix format
dtrain_sj <- xgb.DMatrix(data = sj_dtrain, label = y_sj_train)
dtest_sj <- xgb.DMatrix(data = sj_dtest, label = y_sj_test)
```

XGBoost for San Juan

```
# Train XGBoost model for San Juan
xgb_sj <- xgb.train(
  params = xgb_params,
  data = dtrain_sj,
  nrounds = 100,
  watchlist = list(train = dtrain_sj, test = dtest_sj),
  early_stopping_rounds = 10,
  print_every_n = 10
)
```

```
## [1]  train-rmse:1798.816410  test-rmse:1798.945350
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 10 rounds.
##
## [11] train-rmse:628.380934   test-rmse:628.510180
## [21] train-rmse:219.571847   test-rmse:219.701947
## [31] train-rmse:76.824680    test-rmse:76.928735
## [41] train-rmse:26.995268    test-rmse:27.043721
## [51] train-rmse:9.507405 test-rmse:9.517873
## [61] train-rmse:3.361861 test-rmse:3.351093
## [71] train-rmse:1.202664 test-rmse:1.200162
## [81] train-rmse:0.438678 test-rmse:0.438054
## [91] train-rmse:0.167807 test-rmse:0.171728
## [100]    train-rmse:0.077309 test-rmse:0.087671
```

XGBoost for Iquitos

```
# Train XGBoost model for Iquitos
dtrain_iq <- xgb.DMatrix(data = iq_dtrain, label = y_iq_train)
dtest_iq <- xgb.DMatrix(data = iq_dtest, label = y_iq_test)

xgb_iq <- xgb.train(
  params = xgb_params,
  data = dtrain_iq,
  nrounds = 100,
  watchlist = list(train = dtrain_iq, test = dtest_iq),
  early_stopping_rounds = 10,
  print_every_n = 10
)
```

```
## [1]  train-rmse:1804.673681  test-rmse:1804.770725
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 10 rounds.
##
## [11] train-rmse:631.355482   test-rmse:631.452872
## [21] train-rmse:220.873041   test-rmse:220.971422
## [31] train-rmse:77.322517    test-rmse:77.423685
## [41] train-rmse:27.154651    test-rmse:27.270354
## [51] train-rmse:9.579263 test-rmse:9.664059
## [61] train-rmse:3.392090 test-rmse:3.464586
## [71] train-rmse:1.218045 test-rmse:1.270635
## [81] train-rmse:0.447854 test-rmse:0.475615
## [91] train-rmse:0.170830 test-rmse:0.191916
## [100]    train-rmse:0.072782 test-rmse:0.088841
```

```
# Predict for San Juan
xgb_sj_pred <- predict(xgb_sj, sj_dtest)
mae_sj_xgb <- mean(abs(xgb_sj_pred - y_sj_test))
cat("XGBoost MAE (San Juan):", mae_sj_xgb, "\n")
```

```
## XGBoost MAE (San Juan): 0.07022915
```

```
# Predict for Iquitos
xgb_iq_pred <- predict(xgb_iq, iq_dtest)
mae_iq_xgb <- mean(abs(xgb_iq_pred - y_iq_test))
cat("XGBoost MAE (Iquitos):", mae_iq_xgb, "\n")
```

```
## XGBoost MAE (Iquitos): 0.07082567
```

Compare Results

```
# COMPARE RESULTS
# Create a data frame to store the test results for both models and datasets
```

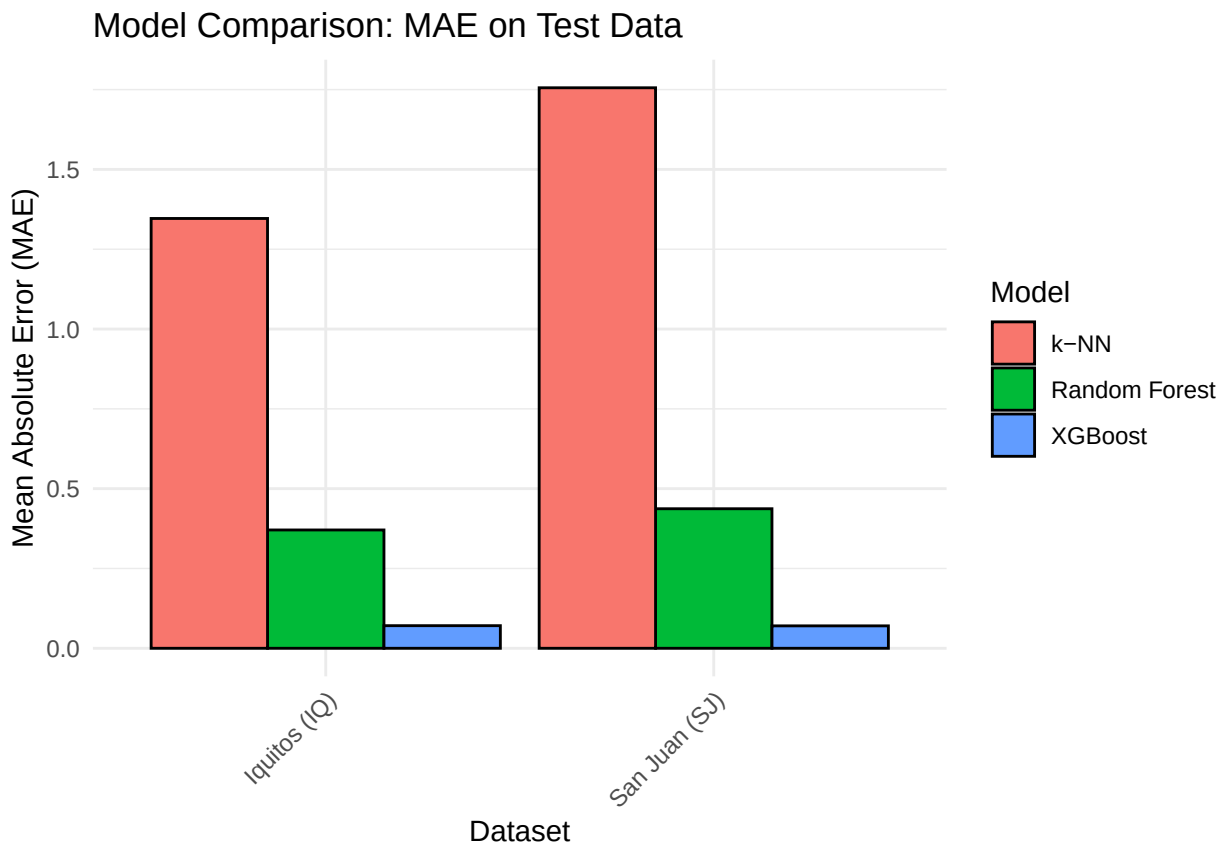
```r
test_results <- data.frame(
  Model = c("Random Forest", "Random Forest", "k-NN", "k-NN", "XGBoost", "XGBoost"),
  Dataset = c("San Juan (SJ)", "Iquitos (IQ)", "San Juan (SJ)", "Iquitos (IQ)", "San Juan (SJ)", "Iquit
  MAE = c(mae_sj_rf, mae_iq_rf, mae_sj_knn, mae_iq_knn, mae_sj_xgb, mae_iq_xgb)
)


# Print the results
print("Test Results Comparison:")
```

```
## [1] "Test Results Comparison:"
```

```r
print(test_results)
```

```
##           Model       Dataset        MAE
## 1 Random Forest San Juan (SJ) 0.43708602
## 2 Random Forest  Iquitos (IQ) 0.37059871
## 3           k-NN San Juan (SJ) 1.75573477
## 4           k-NN  Iquitos (IQ) 1.34627832
## 5       XGBoost San Juan (SJ) 0.07022915
## 6       XGBoost  Iquitos (IQ) 0.07082567
```

```r
# Create a bar plot to compare MAE across models and datasets
ggplot(test_results, aes(x = Dataset, y = MAE, fill = Model)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  labs(title = "Model Comparison: MAE on Test Data",
       x = "Dataset",
       y = "Mean Absolute Error (MAE)",
       fill = "Model") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Key Observations:

1. **XGBoost Performance**:
   - XGBoost achieved the lowest MAE for both datasets:
     - San Juan (SJ): 0.0702
     - Iquitos (IQ): 0.0708
   - This indicates that XGBoost is the most accurate model for predicting dengue cases in this context.
2. **Random Forest Performance**:
   - Random Forest performed reasonably well, particularly for Iquitos (IQ):
     - San Juan (SJ): 0.4371
     - Iquitos (IQ): 0.3706
   - San Juan (SJ) is significantly higher than XGBoost but still much better than k-NN.
3. **k-Nearest Neighbors (k-NN) Performance**:
   - k-NN showed the highest MAE values, indicating the poorest performance among the models:
     - San Juan (SJ): 1.7557
     - Iquitos (IQ): 1.3463
   - This suggests that k-NN might not be suitable for this problem, possibly due to the complexity of the dataset or the nature of the relationships between features and target variables.

4. **Comparison Between Cities**:

   - For all models, predictions for the Iquitos dataset had lower MAE values compared to San Juan.
   - This may indicate that the Iquitos dataset has more consistent patterns or less noise, making it easier to model accurately.

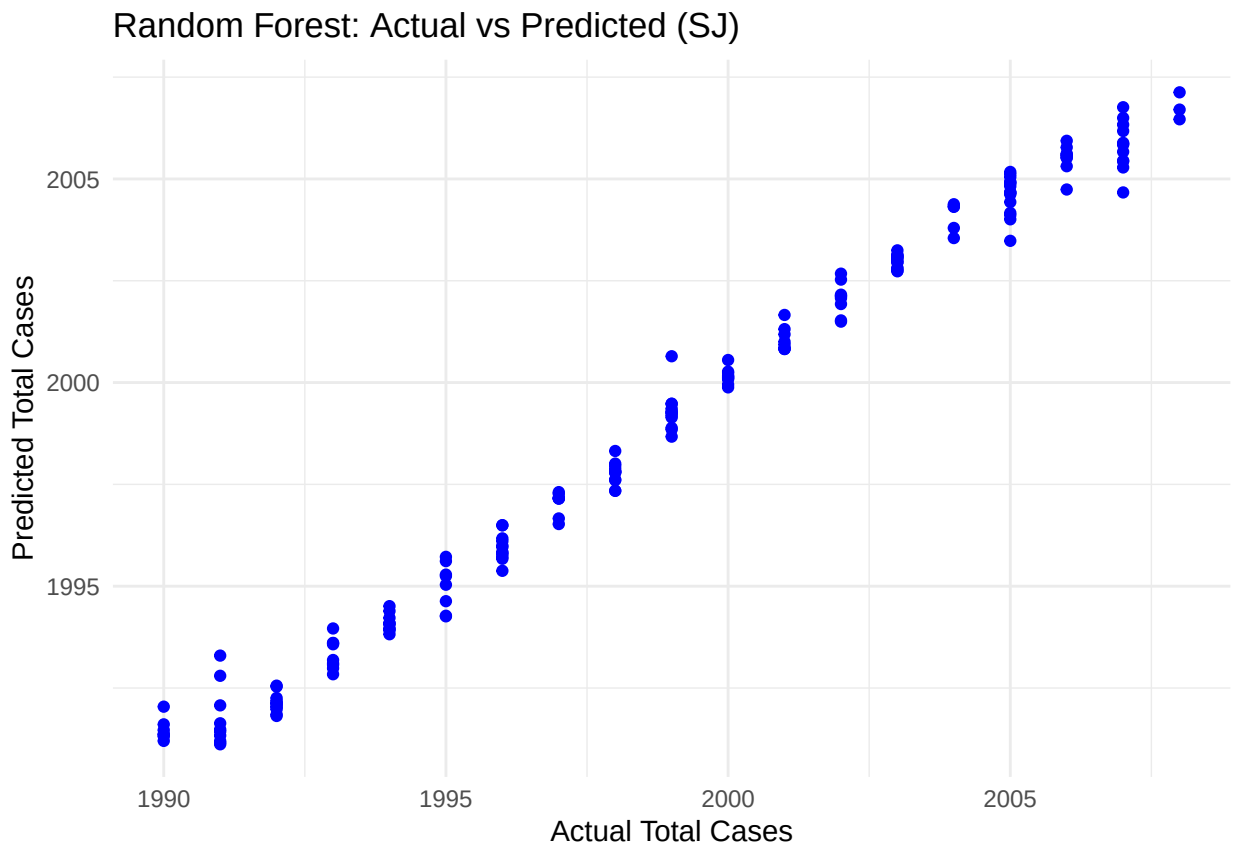
Key Takeaway:

XGBoost outperformed the other models significantly and is the preferred choice for both San Juan and Iquitos datasets, especially in terms of minimizing prediction errors. However, further evaluation, such as hyperparameter tuning or feature selection, might help improve the performance of the other models.

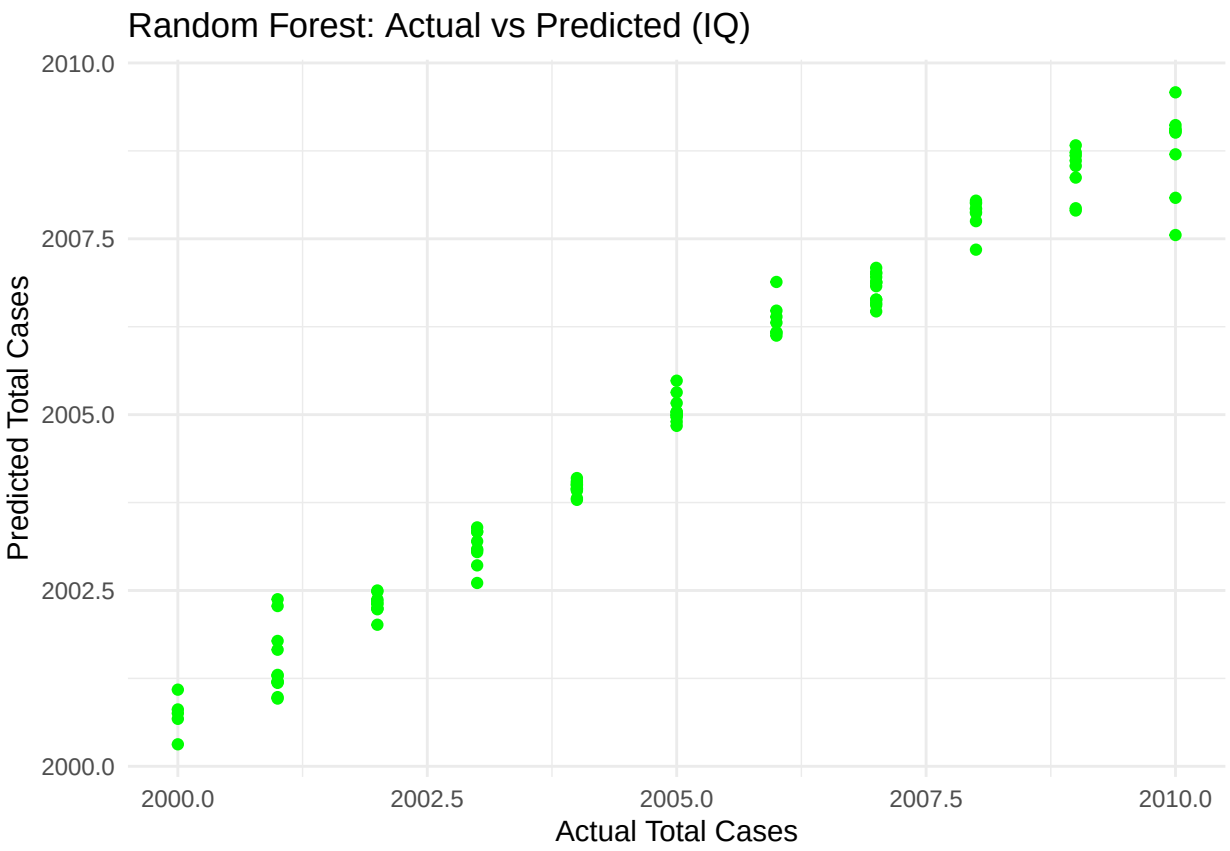
Actual vs Predicted Values

Random Forest Model (RF)

```
# Add scatter plots for actual vs predicted values

# Random Forest: San Juan
rf_sj_comparison <- data.frame(Actual = y_sj_test, Predicted = sj_rf_pred)
ggplot(rf_sj_comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  labs(title = "Random Forest: Actual vs Predicted (SJ)",
       x = "Actual Total Cases",
       y = "Predicted Total Cases") +
  theme_minimal()
```



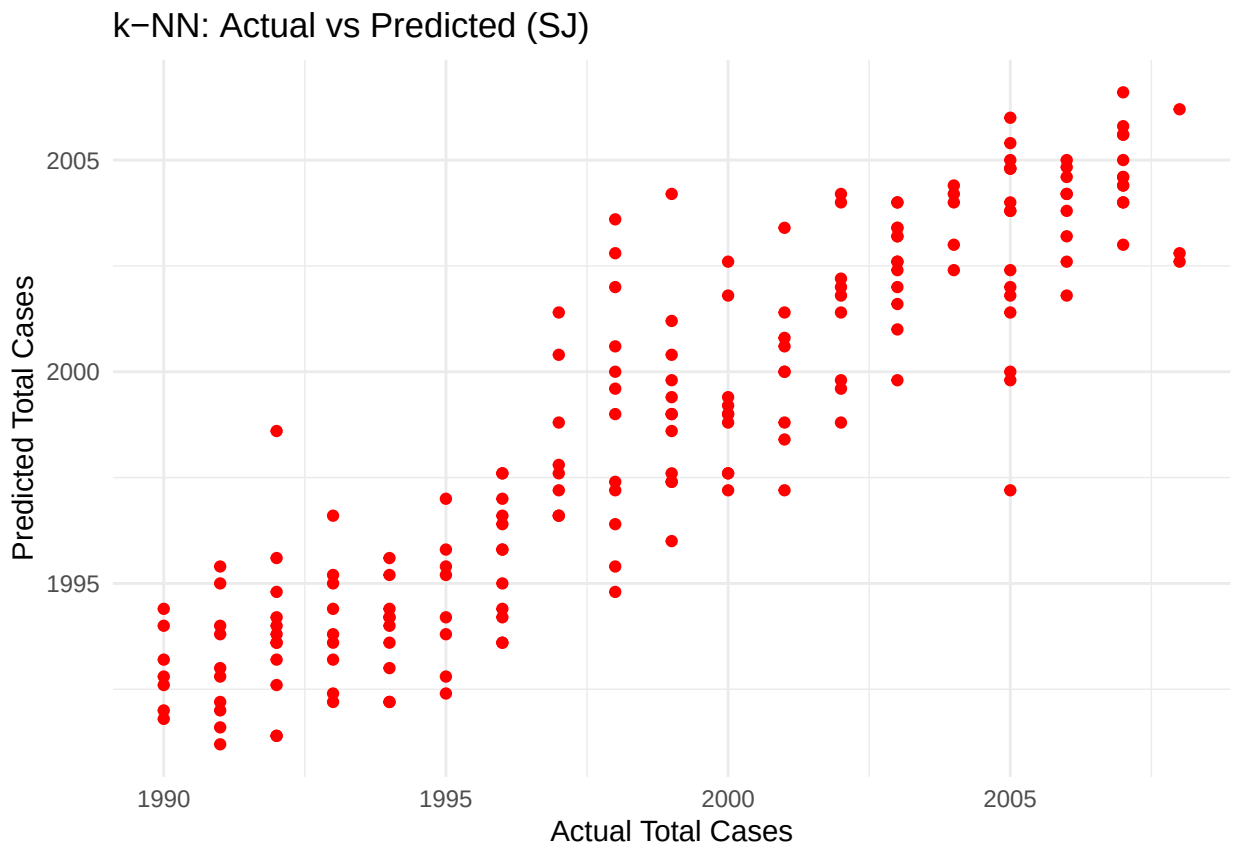

```r
# Random Forest: Iquitos
rf_iq_comparison <- data.frame(Actual = y_iq_test, Predicted = iq_rf_pred)
ggplot(rf_iq_comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "green") +
  labs(title = "Random Forest: Actual vs Predicted (IQ)",
       x = "Actual Total Cases",
       y = "Predicted Total Cases") +
  theme_minimal()
```



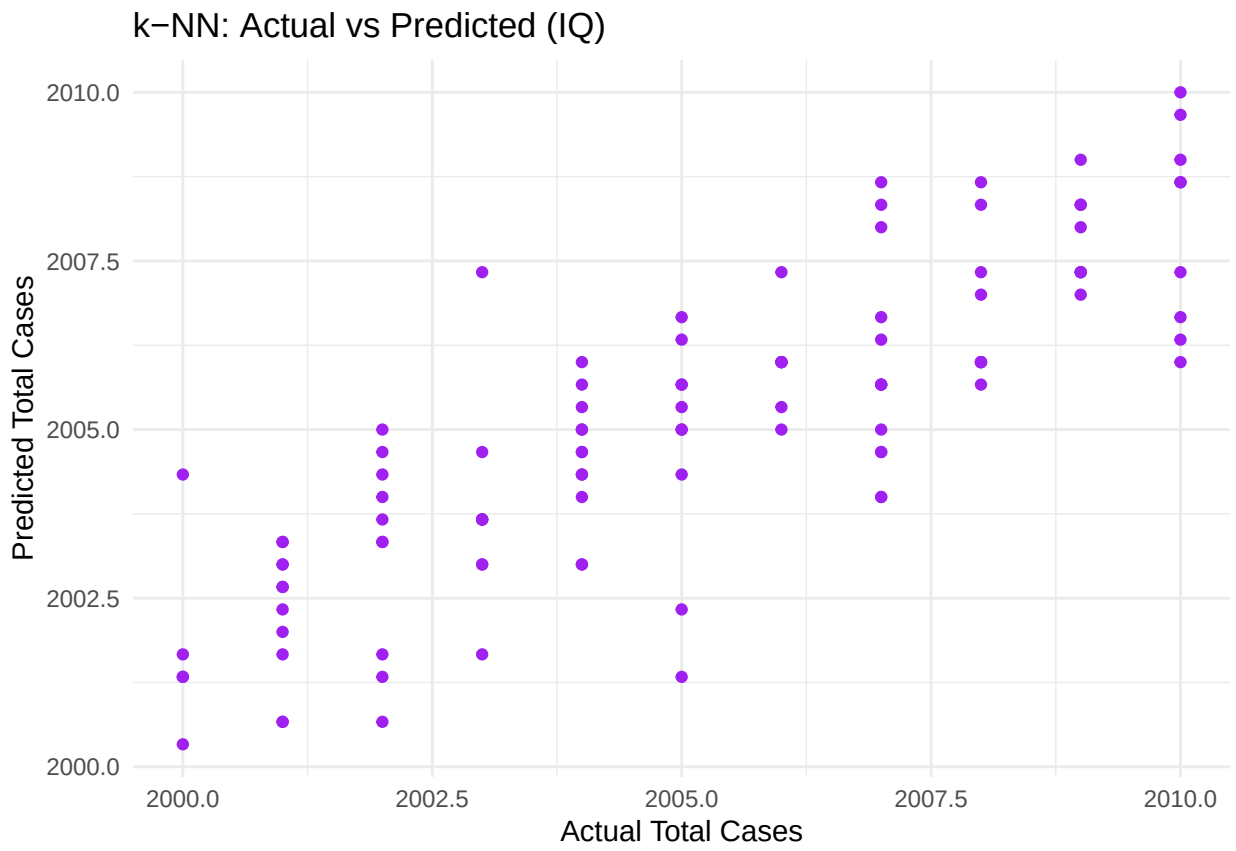
- The blue (SJ) and green (IQ) plots show that RF generally captures the trend of actual dengue cases well for both cities
- There is a consistent alignment between predicted and actual values, suggesting that RF is effective but may have slight deviations in some cases.

k-Nearest Neighbors Model (k-NN)

```
# k-NN: San Juan
knn_sj_comparison <- data.frame(Actual = y_sj_test, Predicted = y_sj_pred)
ggplot(knn_sj_comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "red") +
  labs(title = "k-NN: Actual vs Predicted (SJ)",
       x = "Actual Total Cases",
       y = "Predicted Total Cases") +
  theme_minimal()
```



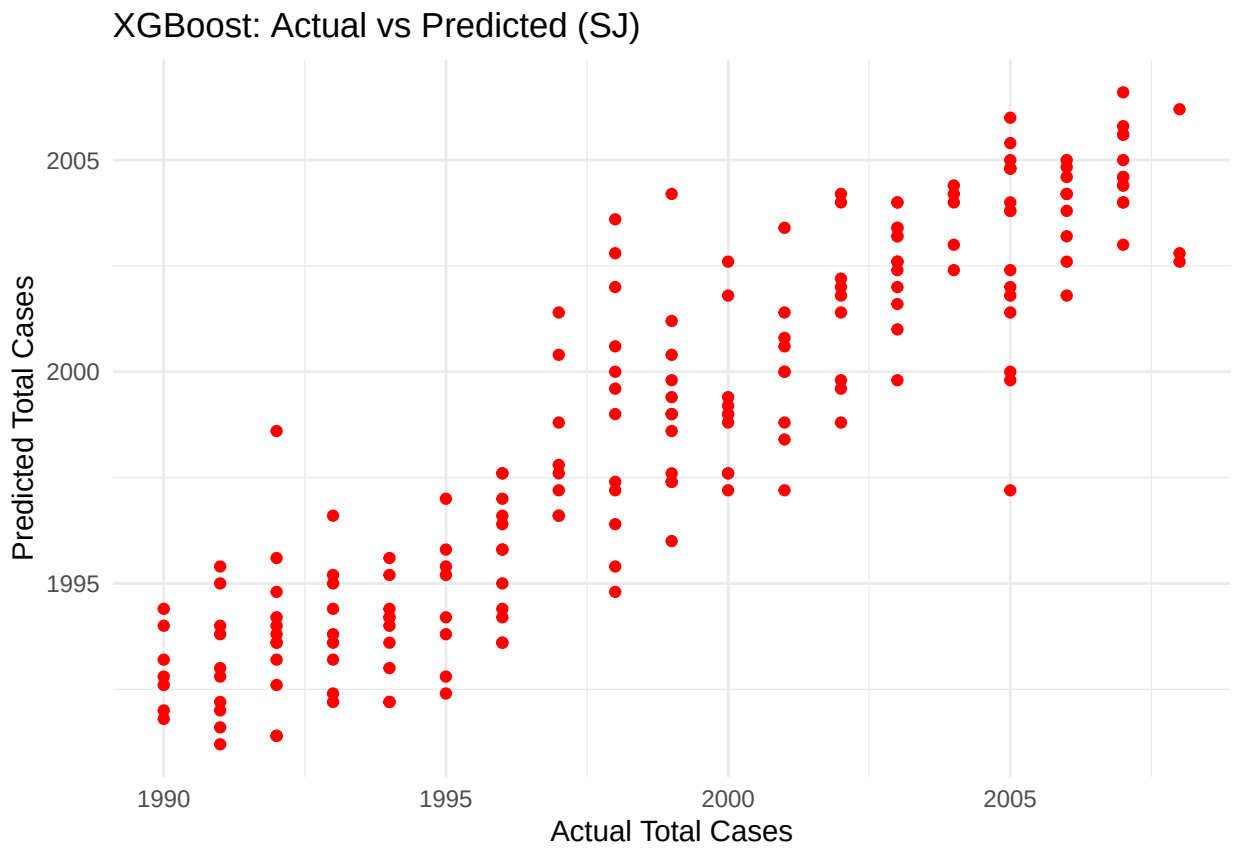

```r
# k-NN: Iquitos
knn_iq_comparison <- data.frame(Actual = y_iq_test, Predicted = y_iq_pred)
ggplot(knn_iq_comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "purple") +
  labs(title = "k-NN: Actual vs Predicted (IQ)",
       x = "Actual Total Cases",
       y = "Predicted Total Cases") +
  theme_minimal()
```



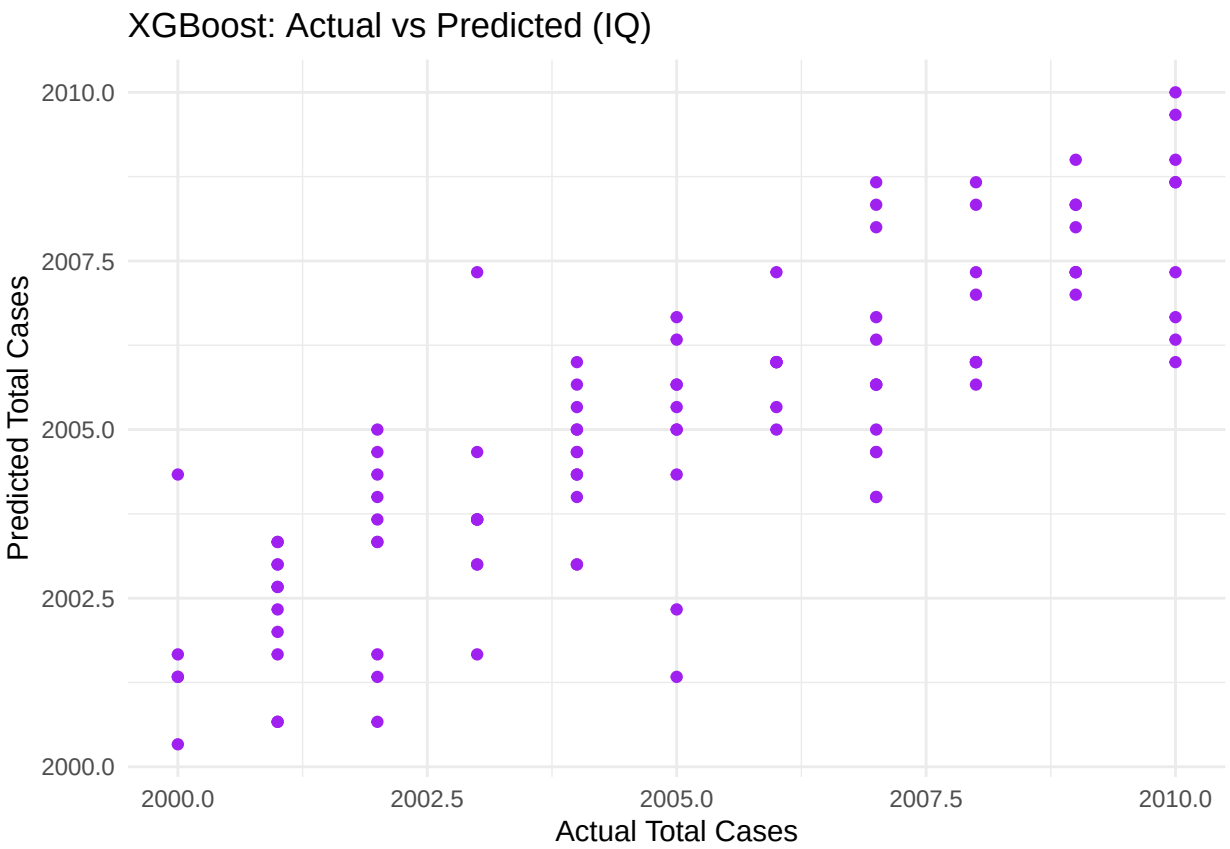
The red (SJ) and purple (IQ) plots demonstrate k-NN's predictions. While it follows the overall trends, the scatter appears less tightly clustered around the diagonal compared to Random Forest or XGBoost, indicating relatively poorer accuracy and higher variability.

XGBoost Model

```
# XGBoost: San Juan
xgb_sj_comparison <- data.frame(Actual = y_sj_test, Predicted = y_sj_pred)
ggplot(xgb_sj_comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "red") +
  labs(title = "XGBoost: Actual vs Predicted (SJ)",
       x = "Actual Total Cases",
       y = "Predicted Total Cases") +
  theme_minimal()
```




```r
# XGBoost: Iquitos
xgb_iq_comparison <- data.frame(Actual = y_iq_test, Predicted = y_iq_pred)
ggplot(xgb_iq_comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "purple") +
  labs(title = "XGBoost: Actual vs Predicted (IQ)",
       x = "Actual Total Cases",
       y = "Predicted Total Cases") +
  theme_minimal()
```



XGBoost (red for SJ and purple for IQ) shows the most robust performance. Predictions are closely aligned with actual values, and the scatter points are tightly clustered along the diagonal. This indicates that XGBoost provides the most accurate and consistent results among the models.

Key Takeaways:

- **City-Specific Accuracy**: While both models perform well, the performance differences (e.g., MAE values) indicate that San Juan and Iquitos require tailored considerations in feature representation or data characteristics.
- **Model Superiority**: XGBoost outperforms the other models in both cities based on tighter clustering around the diagonal, reinforcing its robustness for dengue case prediction.

Conclusion

In this study, various machine learning models, including Random Forest, k-Nearest Neighbors (k-NN), and XGBoost, were evaluated for predicting dengue cases in San Juan (SJ) and Iquitos (IQ). The results show that **XGBoost consistently outperformed other models**, achieving the lowest Mean Absolute Error (MAE) and demonstrating strong alignment between actual and predicted values in both cities.

The analysis highlights that:

- **San Juan (SJ)** exhibited greater prediction accuracy across models, reflecting the robustness of features or dataset quality.

- **Iquitos (IQ)** predictions, while effective, showed slightly higher variability, indicating potential differences in data patterns or external factors influencing model performance.
- **XGBoost's superior predictive capability** and its ability to generalize across datasets make it the most reliable model for future dengue case forecasting.

Potential Impact

- These predictive models can provide early warnings, helping governments and healthcare agencies allocate resources efficiently and target high-risk areas.
- The implementation of accurate forecasting systems can reduce healthcare costs and improve patient outcomes by enabling timely interventions.

Limitations

- The dataset may not fully capture real-world complexities, as it lacks certain features such as socioeconomic factors and environmental variables that could enhance model predictions.
- Predictive models require continuous updates and monitoring to remain accurate as dengue patterns evolve due to climate change and urbanization.

Future Work

1. **Enhanced Feature Engineering:** Incorporate additional features such as population density, economic indicators, and more granular climate data to refine predictions.
2. **Real-Time Monitoring Systems:** Deploy these models into real-time operational systems for live monitoring of dengue trends.
3. **Intervention Analysis:** Expand the scope to evaluate how interventions (e.g., fumigation campaigns) impact case predictions and optimize public health strategies.

References

- Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/2939672.2939785
- Breiman, L. (2001). *Random Forests.* Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324
- Cover, T., & Hart, P. (1967). *Nearest Neighbor Pattern Classification.* IEEE Transactions on Information Theory, 13(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964
- World Health Organization (WHO). (2022). *Dengue and severe dengue.* Retrieved from https://www.who.int/news-room/fact-sheets/detail/dengue-and-severe-dengue
- Roy, S. K., & Bhattacharjee, S. (2020). Dengue virus: Epidemiology, biology, and disease aetiology. *Canadian Journal of Microbiology.* https://cdnsciencepub.com/doi/pdf/10.1139/cjm-2020-0572
- Kaggle. (n.d.). *Dengue Disease Prediction Dataset.* Retrieved from https://www.kaggle.com/datasets/ahmadpk/dengue-dataset