Capstone Project: Predicting Vegetation Response Using Rainfall and NDVI Data

HarvardX Professional Data Science

Benny Istanto (bennyistanto@gmail.com)

26 September 2024

Contents

---

1 Introduction

1.1 Background and Motivation

Rainfall and vegetation health are closely intertwined, especially in regions with distinct wet and dry seasons, such as Indonesia. Rainfall plays a critical role in influencing plant growth, and vegetation health can be effectively assessed using the **Normalized Difference Vegetation Index** (NDVI), a widely used satellite-based indicator of green vegetation.

Indramayu, located in West Java, Indonesia, is an excellent case study due to its vulnerability to **climate extremes** and its agricultural importance. As one of the largest rice-producing regions in Indonesia, Indramayu is heavily reliant on **rainfall** for irrigation, particularly because it lies at the **tail end of irrigation systems**. This geographical disadvantage means that the region is often prone to water shortages during dry seasons or drought events, which can have severe impacts on agricultural productivity. In contrast, during the wet season, excessive rainfall can lead to flooding, further complicating agricultural management.

The ability to **predict NDVI** based on rainfall data can provide valuable insights into how rainfall influences vegetation health and crop growth, especially in a climate-sensitive region like Indramayu. Such predictions can help improve **agricultural planning**, enhance **disaster preparedness**, and guide sustainable water management practices.

This project aims to explore the **time-lagged relationship** between rainfall and NDVI, focusing on how rainfall affects vegetation health with delayed effects over time, particularly in a region where both droughts and floods pose significant challenges to agricultural sustainability.

1.2 Research Objectives

- To investigate the **time-lagged effects** of rainfall on vegetation health (measured by NDVI), particularly in Indramayu's **agricultural landscape**.
- To **predict NDVI** values based on rainfall data using machine learning models, including regression and Random Forest.
- To assess which rainfall periods (e.g., dekads, months) have the most significant influence on vegetation response, and how **climate extremes** like droughts and floods affect the time-lagged relationship.

1.3 Problem Statement

Indramayu's agricultural sector faces substantial risks due to **climate variability**, particularly in terms of water availability. Understanding how rainfall impacts vegetation health is crucial for managing agriculture, predicting crop yields, and monitoring environmental health in such vulnerable regions. However, the relationship between rainfall and vegetation response is not immediate. Rainfall from previous periods may have delayed effects on NDVI, making the prediction task complex.

This project will develop a predictive model to forecast NDVI based on **time-lagged rainfall data**, with the goal of providing insights into how **rainfall variability** influences crop health. By doing so, the model aims to support better **environmental and agricultural management** practices, particularly in regions like Indramayu where the timing and amount of rainfall are critical for successful crop production.

1.4 Study Area and Data

Indramayu has been selected as the study area due to its **vulnerability to climate extremes** and its importance as an agricultural hub in Indonesia. The region is highly dependent on rainfall for irrigation, especially because it is located at the **tail end of irrigation systems**, making water management even more challenging during dry seasons. The dataset for this study includes rainfall and NDVI data over a **20-year period**, covering from **2003 to 2023**. These datasets provide a robust basis for understanding the relationship between rainfall variability and vegetation health over time.

Rainfall Data:

- Rainfall estimates are derived from **satellite observations** combined with **rain gauge data** (CHIRPS v2).
- **Dekadal rainfall values** (10-day intervals) are provided, alongside rolling 1-month and 3-month rainfall aggregations. - **Rainfall anomalies** are calculated as the percentage deviation from the long-term average, highlighting periods of excessive or insufficient rainfall.

NDVI Data:

- NDVI is derived from **MODIS** satellite imagery, with 10-day dekadal composites averaged over sub-national units.
- **NDVI anomalies** indicate deviations from the long-term average for each period, reflecting periods of vegetation stress or growth.

By focusing on Indramayu, this study aims to provide valuable insights into how rainfall affects vegetation health in a region that is both agriculturally important and vulnerable to climate extremes.

1.5 Structure of the Report

The report will be structured as follows: 1. **Introduction**: Provides the background, research objectives, and problem statement. 2. **Data Overview and Preprocessing**: Describes the datasets and preprocessing steps, including handling missing data and feature engineering. 3. **Exploratory Data Analysis (EDA)**: Presents insights from initial data exploration, such as correlations and time-series trends. 4. **Methods and Machine Learning Models**: Describes the machine learning models used, including time-lagged regression and Random Forest. 5. **Results and Evaluation**: Discusses the model performance, visualizes predictions, and analyzes feature importance. 6. **Discussion and Insights**: Interprets the results in the context of rainfall-NDVI dynamics. 7. **Conclusion and Future Work**: Summarizes findings and proposes future extensions.

---

2 Data Overview and Preprocessing

2.1 Data Sources

The data for this project is sourced from two primary datasets:

1. **Rainfall Data**:
   - The rainfall data is derived from the **Climate Hazards Group InfraRed Precipitation with Station data (CHIRPS v2)** and aggregated based on admin boundary by the World Food Programme. This dataset provides dekadal (10-day) rainfall estimates, which combine satellite imagery and in-situ rainfall station data.
   - The rainfall data contains the following key variables:
     - **rfh**: 10-day rainfall (in mm).
     - **r1h**: 1-month rolling aggregation of rainfall.
     - **r3h**: 3-month rolling aggregation of rainfall.
     - **rfh_avg**: rainfall long term average [mm].
     - **r1h_avg**: rainfall 1-month rolling aggregation long term average [mm].
     - **r3h_avg**: rainfall 3-month rolling aggregation long term average [mm].
     - **rfq**: Rainfall anomaly (%), indicating deviations from the long-term average.
     - **r1q**: Rainfall 1-month anomaly (%), indicating deviations from the long-term average.
     - **r3q**: Rainfall 3-month anomaly (%), indicating deviations from the long-term average.
   - The data available for download from Rainfall at Humanitarian Data Exchange
2. **NDVI Data**:
   - NDVI is obtained from **MODIS Aqua and Terra satellites**, aggregated into dekadal intervals and aggregated based on admin boundary by the World Food Programme. NDVI is a normalized measure of vegetation health, where higher values indicate greener, healthier vegetation.
   - The NDVI dataset includes:
     - **vim**: 10-day NDVI.
     - **vim_avg**: Long-term average of NDVI.
     - **viq**: NDVI anomaly (%), measuring deviations from the long-term average.
   - The data available for download from NDVI at Humanitarian Data Exchange

Both datasets are aggregated by administrative units at the ADM2 level (district level), providing spatial coverage across all districts in Indonesia, including Indramayu. The analysis will focus on period from 2003-2023.

2.2 Data Preprocessing

Before modeling, it's essential to preprocess the data to ensure it is clean and ready for analysis. The following preprocessing steps are applied:

2.2.1 Handling Missing Data

Missing data is a common issue in environmental datasets. Both rainfall and NDVI data are examined for missing values, and imputation techniques or removal of incomplete records are considered based on the extent of missingness.

If missing values are found, a simple linear interpolation method is used to fill the gaps:

$$X_t = \frac{X_{t-1} + X_{t+1}}{2}$$

where X_t is the missing value at time t, and X_{t-1} and X_{t+1} are the known values before and after the missing observation.

2.2.2 Scaling and Normalization

To ensure compatibility with machine learning models, rainfall and NDVI data are scaled to have zero mean and unit variance. This is particularly important for algorithms like **linear regression** or **Random Forest**, which can be sensitive to the scale of input features.

Standardization is applied to the variables:

$$X' = \frac{X - \mu}{\sigma}$$

where X is the original value, μ is the mean, and σ is the standard deviation of the feature.

Then it could translated into **R Code for Preprocessing** below:

```r
# Load required libraries
library(dplyr)
library(tidyr)
library(zoo) # for interpolation
library(caret) # for scaling

# Handling missing values using linear interpolation
rainfall_data <- rainfall_data %>%
  group_by(ADM2_PCODE) %>%
  arrange(date) %>%
  mutate(rfh = na.approx(rfh),
         r1h = na.approx(r1h),
         r3h = na.approx(r3h))

ndvi_data <- ndvi_data %>%
  group_by(ADM2_PCODE) %>%
  arrange(date) %>%
  mutate(vim = na.approx(vim))

# Scaling and normalization using caret
```

```r
preProc <- preProcess(rainfall_data[, c("rfh", "r1h", "r3h")],
                      method = c("center", "scale"))
rainfall_data_scaled <- predict(preProc, rainfall_data)

preProc_ndvi <- preProcess(ndvi_data[, "vim"], method = c("center", "scale"))
ndvi_data_scaled <- predict(preProc_ndvi, ndvi_data)
```

2.3 Feature Engineering

Feature engineering is critical for capturing the time-lagged effects of rainfall on vegetation health. By creating **lag features** for rainfall, we can account for the delayed impact of rainfall on NDVI.

2.3.1 Lag Features

In environmental modeling, the effect of rainfall on vegetation is often not immediate. To capture these delayed effects, lag features are created from the rainfall data. For example, rainfall from 1, 2, or 3 dekads before the current NDVI measurement may influence current vegetation health.

Lagged rainfall features are created using the following transformation:

$$\text{Rainfall}_{t-\delta} = \text{Rainfall}(t - \delta)$$

where t is the current time step and δ is the lag (e.g., 1 dekad, 2 dekads, etc.).

Then it could translated into **R Code for Creating Lag Features** below:

```r
# Creating lag features for rainfall
rainfall_lags <- rainfall_data_scaled %>%
  group_by(ADM2_PCODE) %>%
  arrange(date) %>%
  mutate(rainfall_lag_1 = lag(rfh, 1),
         rainfall_lag_2 = lag(rfh, 2),
         rainfall_lag_3 = lag(rfh, 3))

# Joining rainfall lagged data with NDVI
combined_data <- left_join(ndvi_data_scaled,
                           rainfall_lags,
                           by = c("date", "ADM2_PCODE"))

# Dropping rows with missing lagged values
combined_data <- combined_data %>% drop_na(rainfall_lag_1,
                                           rainfall_lag_2,
                                           rainfall_lag_3)
```

2.3.2 Adding Seasonal Information

Indonesia experiences distinct wet and dry seasons, which significantly impact vegetation health. Therefore, adding seasonal features (such as the month or a binary indicator for the wet season) can help the model capture these seasonal variations.

A binary variable for the wet season can be created based on the month of the year:

$$\text{Season}_t = \begin{cases} 1 & \text{if month } \in \{11, 12, 1, 2, 3, 4\} \\ 0 & \text{otherwise} \end{cases}$$

Then it could translated into **R Code for Adding Seasonal Features** below:

```r
# Adding seasonal information based on month
combined_data <- combined_data %>%
  mutate(month = as.numeric(format(as.Date(date), "%m")),
         wet_season = ifelse(month %in% c(11, 12, 1, 2, 3, 4), 1, 0))
```

---

3 Exploratory Data Analysis

In order to explore the relationship between rainfall and NDVI (Normalized Difference Vegetation Index) in Indramayu, the first step is to load the necessary libraries that will facilitate data handling, visualization, and manipulation. This step ensures that we have the required tools to perform efficient data analysis and processing.

3.1 Loading Necessary Libraries

R provides a variety of powerful packages for data analysis, and in this project, we will use several key libraries:

- **ggplot2**: A popular package for creating complex and elegant data visualizations in a structured and intuitive way.
- **dplyr**: Provides a suite of functions for data manipulation, including filtering, selecting, and transforming data. This package is particularly useful for its simplicity in handling large datasets.
- **readr**: Enables fast and flexible reading of rectangular data (like CSV files) into R. This package is used to load our data quickly and efficiently.
- **lubridate**: Designed to simplify working with dates and times in R. It allows us to convert and manipulate date columns in the dataset, ensuring correct date formats for time-series analysis.
- **readxl**: Allows importing of Excel files without needing to install external dependencies like Java. We will use this to read boundary-related data from Excel files.
- **tinytex**: This package is used for compiling reports in LaTeX. While not critical to the exploratory data analysis itself, it is useful for rendering documents and reports, especially if you are working in a reproducible environment like R Markdown.

The following code chunk installs any missing packages and then loads the required libraries. We also track the time it takes to load the libraries as part of ensuring an efficient workflow.

```r
# Load necessary libraries and track time
start_time <- Sys.time()

# Install libraries if needed
if(!require(tinytex)) {
  install.packages("tinytex", repos = "http://cran.us.r-project.org",
                   dependencies = TRUE, quiet = TRUE)
  tinytex::install_tinytex(quiet = TRUE)
  tinytex::tlmgr("option repository https://mirror.ctan.org/systems/texlive/tlnet")
}
```

```
## Loading required package: tinytex
```

```
if(!require(ggplot2)) install.packages("ggplot2",
                                       repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggplot2
```

```
if(!require(dplyr)) install.packages("dplyr",
                                       repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
if(!require(readr)) install.packages("readr",
                                       repos = "http://cran.us.r-project.org")
```

```
## Loading required package: readr
```

```
if(!require(lubridate)) install.packages("lubridate",
                                       repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
if(!require(readxl)) install.packages("readxl",
                                       repos = "http://cran.us.r-project.org")
```

```
## Loading required package: readxl
```

```
if(!require(stats)) install.packages("stats",
                                       repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest",
                                       repos = "http://cran.us.r-project.org")
```

```
## 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(forecast)) install.packages("forecast",
                                        repos = "http://cran.us.r-project.org")
```

```
## Loading required package: forecast

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
if(!require(Metrics)) install.packages("Metrics",
                                        repos = "http://cran.us.r-project.org")
```

```
## Loading required package: Metrics

##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
if(!require(RColorBrewer)) install.packages("RColorBrewer",
                                        repos = "http://cran.us.r-project.org")
```

```
## Loading required package: RColorBrewer
```

```
# Load libraries
library(tinytex)
library(ggplot2)
library(dplyr)
library(readr)
library(lubridate)
library(readxl)
```

```r
library(stats)
library(randomForest)
library(forecast)
library(Metrics)
library(RColorBrewer)

# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  1.18614792823792"
```

3.2 Visualizing Rainfall and NDVI Trends

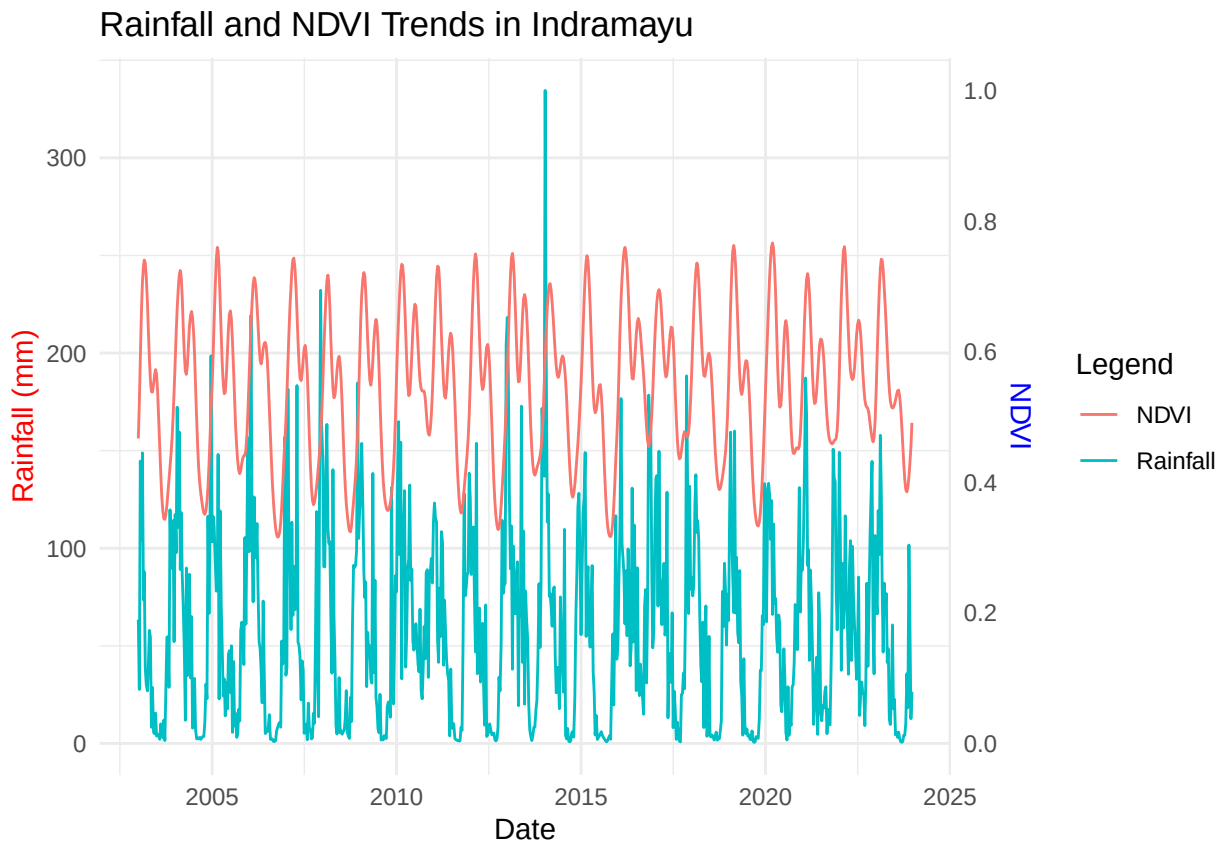
In this section, we aim to visualize the time-series trends of both rainfall and NDVI for Indramayu over the study period. This will help identify seasonal patterns and any recurring cycles in the data.

```r
# Load necessary libraries and track time
start_time <- Sys.time()

# Define local file paths
rainfall_file <- "idn-rainfall-adm2-full.csv"
ndvi_file <- "idn-ndvi-adm2-full.csv"
adm2_file <- "idn_adminboundaries_tabulardata.xlsx"

# URLs for downloading data
rainfall_data_url <- "https://data.humdata.org/dataset/e7b6ce3e-5a35-4c12-9ee9-76153da18bf3/resource/30
ndvi_data_url <- "https://data.humdata.org/dataset/f3234974-3ca9-4a10-a97a-674705eaeea7/resource/cb6098
adm2_data_url <- "https://data.humdata.org/dataset/84a1d98a-790b-4d66-9d14-bbfa48500802/resource/9f9a48

# Check if files are already downloaded
if(!file.exists(rainfall_file)) {
  download.file(rainfall_data_url, rainfall_file, quiet = TRUE)
}
if(!file.exists(ndvi_file)) {
  download.file(ndvi_data_url, ndvi_file, quiet = TRUE)
}
if(!file.exists(adm2_file)) {
  download.file(adm2_data_url, adm2_file, quiet = TRUE)
}

# Load the datasets using explicit package references
rainfall_data <- readr::read_csv(rainfall_file)
```

```
## Rows: 807463 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (14): date, adm2_id, ADM2_PCODE, n_pixels, rfh, rfh_avg, r1h, r1h_avg, r...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
ndvi_data <- readr::read_csv(ndvi_file)

## Rows: 409888 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): date, adm2_id, ADM2_PCODE, n_pixels, vim, vim_avg, viq
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

adm2_data <- readxl::read_excel(adm2_file, sheet = "ADM2")

# Filter Indramayu (ADM2_PCODE = ID3212) and data between 2003/01/01 - 2023/12/31
indramayu_rainfall <- dplyr::filter(rainfall_data, ADM2_PCODE == "ID3212",
                                    date >= "2003-01-01", date <= "2023-12-31")
indramayu_ndvi <- dplyr::filter(ndvi_data, ADM2_PCODE == "ID3212",
                                date >= "2003-01-01", date <= "2023-12-31")

# Convert 'date' column to date format using lubridate
indramayu_rainfall$date <- lubridate::as_date(indramayu_rainfall$date)
indramayu_ndvi$date <- lubridate::as_date(indramayu_ndvi$date)

# Convert rfh (Rainfall) and vim (NDVI) columns to numeric,
# handling any non-numeric entries
indramayu_rainfall$rfh <- as.numeric(indramayu_rainfall$rfh)
indramayu_ndvi$vim <- as.numeric(indramayu_ndvi$vim)

# Find the maximum value of Rainfall (rfh) for axis alignment
max_rainfall <- max(indramayu_rainfall$rfh, na.rm = TRUE)

# Combine the data for dual axis plotting
indramayu_combined <- dplyr::inner_join(indramayu_rainfall,
                                        indramayu_ndvi,
                                        by = "date")

# Create dual Y-axis plot with NDVI aligned with max rainfall axis
ggplot2::ggplot(indramayu_combined, ggplot2::aes(x = date)) +
  ggplot2::geom_line(ggplot2::aes(y = rfh,
                                  color = "Rainfall")) +
  ggplot2::geom_line(ggplot2::aes(y = vim * max_rainfall,
                                  # Align NDVI with Rainfall axis
                                  color = "NDVI")) +
  ggplot2::scale_y_continuous(
    name = "Rainfall (mm)",
    sec.axis = ggplot2::sec_axis(~./max_rainfall,
                                 name = "NDVI",
                                 # NDVI scaled with max Rainfall
                                 breaks = seq(0, 1, 0.2))) +
  ggplot2::labs(title = "Rainfall and NDVI Trends in Indramayu",
       x = "Date",
       color = "Legend") +
  ggplot2::theme_minimal() +
  ggplot2::theme(
    # Adjust axis color for NDVI
```

```
    axis.title.y.right = ggplot2::element_text(color = "blue"),
    # Adjust axis color for Rainfall
    axis.title.y.left = ggplot2::element_text(color = "red")
  )
```




```
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  7.52352499961853"
```

This code will generate a time-series plot showing rainfall and NDVI trends for Indramayu. You should see how NDVI changes in response to rainfall, with potential seasonal variations.

3.3 Correlation Analysis

To understand the relationship between rainfall and NDVI, we perform a correlation analysis, particularly focusing on **lagged correlations**, where rainfall from earlier periods may affect NDVI in later periods.

```
# Load necessary libraries and track time
start_time <- Sys.time()

# Create lagged rainfall features (1 to 6 dekads before)
```

```r
indramayu_rainfall <- indramayu_rainfall %>%
  dplyr::arrange(date) %>%
  dplyr::mutate(rainfall_lag_1 = dplyr::lag(rfh, 1),
                rainfall_lag_2 = dplyr::lag(rfh, 2),
                rainfall_lag_3 = dplyr::lag(rfh, 3),
                rainfall_lag_4 = dplyr::lag(rfh, 4),
                rainfall_lag_5 = dplyr::lag(rfh, 5),
                rainfall_lag_6 = dplyr::lag(rfh, 6))

# Merge rainfall and NDVI datasets by date
merged_data <- dplyr::inner_join(indramayu_ndvi,
                                 indramayu_rainfall,
                                 by = "date")

# Calculate correlation between NDVI and lagged rainfall features
cor_lag_1 <- stats::cor(merged_data$vim,
                        merged_data$rainfall_lag_1,
                        use = "complete.obs")
cor_lag_2 <- stats::cor(merged_data$vim,
                        merged_data$rainfall_lag_2,
                        use = "complete.obs")
cor_lag_3 <- stats::cor(merged_data$vim,
                        merged_data$rainfall_lag_3,
                        use = "complete.obs")
cor_lag_4 <- stats::cor(merged_data$vim,
                        merged_data$rainfall_lag_4,
                        use = "complete.obs")
cor_lag_5 <- stats::cor(merged_data$vim,
                        merged_data$rainfall_lag_5,
                        use = "complete.obs")
cor_lag_6 <- stats::cor(merged_data$vim,
                        merged_data$rainfall_lag_6,
                        use = "complete.obs")

# Print correlation results
base::print(base::paste("Correlation between NDVI and Rainfall Lag 1: ",
                        base::round(cor_lag_1, 3)))
```

```
## [1] "Correlation between NDVI and Rainfall Lag 1:  0.495"
```

```r
base::print(base::paste("Correlation between NDVI and Rainfall Lag 2: ",
                        base::round(cor_lag_2, 3)))
```

```
## [1] "Correlation between NDVI and Rainfall Lag 2:  0.581"
```

```r
base::print(base::paste("Correlation between NDVI and Rainfall Lag 3: ",
                        base::round(cor_lag_3, 3)))
```

```
## [1] "Correlation between NDVI and Rainfall Lag 3:  0.644"
```

```r
base::print(base::paste("Correlation between NDVI and Rainfall Lag 4: ",
                        base::round(cor_lag_4, 3)))
```

```
## [1] "Correlation between NDVI and Rainfall Lag 4:  0.684"
```

```
base::print(base::paste("Correlation between NDVI and Rainfall Lag 5: ",
                        base::round(cor_lag_5, 3)))
```

```
## [1] "Correlation between NDVI and Rainfall Lag 5:  0.699"
```

```
base::print(base::paste("Correlation between NDVI and Rainfall Lag 6: ",
                        base::round(cor_lag_6, 3)))
```

```
## [1] "Correlation between NDVI and Rainfall Lag 6:  0.689"
```

```
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  0.0308229923248291"
```

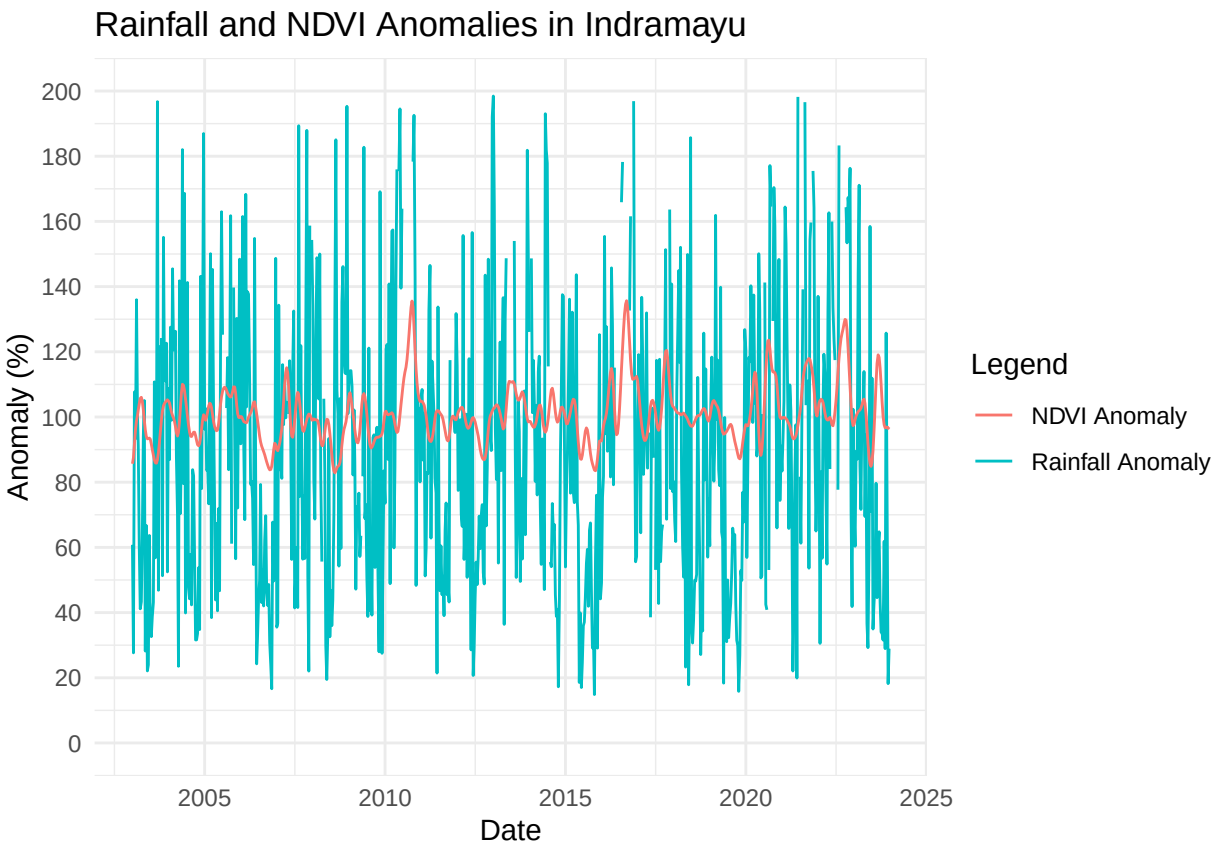
This script calculates and prints the correlations between NDVI and lagged rainfall at 1, 2, and 3 dekads before. These correlations can help identify the time-lagged relationship between rainfall and vegetation health.

3.4 Rainfall and NDVI Anomalies

Understanding anomalies is crucial for detecting unusual events in the rainfall and NDVI data, which could indicate extreme weather or vegetation stress. In this section, we plot anomalies for both variables.

```
# Load necessary libraries and track time
start_time <- Sys.time()

# Filter anomalies from both datasets and convert to numeric
indramayu_rainfall_anomalies <- indramayu_rainfall %>%
  dplyr::select(date, rfq) %>%
  dplyr::rename(rainfall_anomaly = rfq) %>%
  dplyr::mutate(rainfall_anomaly = as.numeric(rainfall_anomaly))

indramayu_ndvi_anomalies <- indramayu_ndvi %>%
  dplyr::select(date, viq) %>%
  dplyr::rename(ndvi_anomaly = viq) %>%
  dplyr::mutate(ndvi_anomaly = as.numeric(ndvi_anomaly))

# Merge anomaly data by date
anomalies_data <- dplyr::inner_join(indramayu_rainfall_anomalies,
                                    indramayu_ndvi_anomalies,
                                    by = "date")

# Plot anomalies on a single Y-axis ranging from 0 to 200%
ggplot2::ggplot(anomalies_data, ggplot2::aes(x = date)) +
  ggplot2::geom_line(ggplot2::aes(y = rainfall_anomaly,
                                  color = "Rainfall Anomaly")) +
```

```
ggplot2::geom_line(ggplot2::aes(y = ndvi_anomaly,
                                color = "NDVI Anomaly")) +
ggplot2::scale_y_continuous(limits = c(0, 200),
                            breaks = seq(0, 200, 20)) +
ggplot2::labs(title = "Rainfall and NDVI Anomalies in Indramayu",
     x = "Date",
     y = "Anomaly (%)",
     color = "Legend") +
ggplot2::theme_minimal()
```




```
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  0.353729963302612"
```

This script will visualize the anomalies for both rainfall and NDVI, helping to detect any deviations from average patterns over time.

---

4 Methods and Machine Learning Models

4.1 Problem Framing

The goal of this project is to predict **NDVI** (Normalized Difference Vegetation Index), a key indicator of vegetation health, using **rainfall data** as the primary predictor. This predictive problem is framed as a **supervised learning task** where we aim to model the time-lagged effects of rainfall on NDVI. Given the structure of the data (where vegetation response to rainfall is delayed), this problem also incorporates temporal elements through the use of **lagged features**.

Supervised Learning Setup:

- **Input**: Rainfall data (including lagged rainfall values)
- **Output**: NDVI values (target variable)
- **Type of task**: Regression, where the goal is to predict continuous NDVI values.

To model this relationship, we will explore several approaches:

- **Time-lagged linear regression**: A simple, interpretable model to understand how lagged rainfall affects NDVI.
- **Random Forest regression**: A more flexible, non-linear model to capture complex interactions.
- **Time-series forecasting models** (optional): For exploring future NDVI predictions based on past data.

4.2 Time-Lagged Regression Model

Linear regression is a common approach for modeling the relationship between a dependent variable y (in this case, NDVI) and one or more independent variables X (lagged rainfall values). The general form of a **multiple linear regression** model is:

$$y_t = \beta_0 + \beta_1 X_{t-1} + \beta_2 X_{t-2} + ... + \beta_n X_{t-n} + \epsilon$$

Where: - y_t is the NDVI value at time t. - $X_{t-1}, X_{t-2}, ..., X_{t-n}$ are the lagged rainfall values from previous periods (dekads). - β_0 is the intercept. - $\beta_1, \beta_2, ..., \beta_n$ are the coefficients representing the effect of rainfall at different lags. - ϵ is the error term.

In our case, we use up to **6 lagged rainfall values** to capture the delayed effect of rainfall on vegetation health. This approach allows us to quantify how rainfall from previous periods affects NDVI, offering insights into both short-term and long-term vegetation responses.

```
# Load necessary libraries and track time
start_time <- Sys.time()

# Create lagged rainfall features (1 to 6 dekads before)
indramayu_rainfall <- indramayu_rainfall %>%
  dplyr::arrange(date) %>%
  dplyr::mutate(rainfall_lag_1 = dplyr::lag(rfh, 1),
                rainfall_lag_2 = dplyr::lag(rfh, 2),
                rainfall_lag_3 = dplyr::lag(rfh, 3),
                rainfall_lag_4 = dplyr::lag(rfh, 4),
                rainfall_lag_5 = dplyr::lag(rfh, 5),
                rainfall_lag_6 = dplyr::lag(rfh, 6))
```

```r
# Merge rainfall and NDVI datasets by date
merged_data <- dplyr::inner_join(indramayu_ndvi,
                                 indramayu_rainfall,
                                 by = "date")

# Build linear regression model with lagged rainfall features
linear_model <- stats::lm(vim ~ rainfall_lag_1 + rainfall_lag_2 +
                          rainfall_lag_3 + rainfall_lag_4 + rainfall_lag_5
                          + rainfall_lag_6, data = merged_data)

# Print model summary
base::summary(linear_model)
```

```
##
## Call:
## stats::lm(formula = vim ~ rainfall_lag_1 + rainfall_lag_2 + rainfall_lag_3 +
##     rainfall_lag_4 + rainfall_lag_5 + rainfall_lag_6, data = merged_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.172800 -0.049174 -0.007631  0.048168  0.183816
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.152e-01  4.360e-03  95.234  < 2e-16 ***
## rainfall_lag_1 8.595e-05  6.919e-05   1.242  0.21454
## rainfall_lag_2 2.477e-04  7.643e-05   3.241  0.00125 **
## rainfall_lag_3 3.235e-04  7.739e-05   4.180 3.27e-05 ***
## rainfall_lag_4 4.495e-04  7.745e-05   5.804 9.60e-09 ***
## rainfall_lag_5 5.125e-04  7.618e-05   6.728 3.44e-11 ***
## rainfall_lag_6 6.899e-04  6.894e-05  10.007  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06623 on 743 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.6627, Adjusted R-squared:   0.66
## F-statistic: 243.3 on 6 and 743 DF,  p-value: < 2.2e-16
```

```r
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  0.119018077850342"
```

Interpretation:

- The coefficients for the lagged rainfall variables will tell us how much each lagged period (dekad) contributes to NDVI changes.
- We can assess the **p-values** to determine which lags are statistically significant predictors of NDVI.

4.3 Random Forest Regression

Random Forest is a non-linear, ensemble learning method that builds multiple decision trees and averages their predictions to reduce overfitting and improve generalization. Random Forest can handle complex interactions between features, making it a good choice for modeling the **non-linear** relationships between rainfall and NDVI. Additionally, it can naturally incorporate **feature importance analysis**, helping us identify which lagged rainfall values have the most influence on NDVI.

The basic idea of Random Forest regression can be described as:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(X)$$

Where: - $\hat{y}$ is the predicted NDVI value. - T is the number of decision trees. - $f_t(X)$ is the prediction from the t-th decision tree based on the input features X (lagged rainfall values).

```
# Load necessary libraries and track time
start_time <- Sys.time()

# Create lagged rainfall features (as done earlier)
indramayu_rainfall <- indramayu_rainfall %>%
  dplyr::arrange(date) %>%
  dplyr::mutate(rainfall_lag_1 = dplyr::lag(rfh, 1),
                rainfall_lag_2 = dplyr::lag(rfh, 2),
                rainfall_lag_3 = dplyr::lag(rfh, 3),
                rainfall_lag_4 = dplyr::lag(rfh, 4),
                rainfall_lag_5 = dplyr::lag(rfh, 5),
                rainfall_lag_6 = dplyr::lag(rfh, 6))

# Merge rainfall and NDVI datasets by date
merged_data <- dplyr::inner_join(indramayu_ndvi,
                                 indramayu_rainfall,
                                 by = "date")

# Remove rows with missing values (NA) from the merged dataset
cleaned_data <- na.omit(merged_data)

# Build Random Forest regression model
rf_model <- randomForest::randomForest(vim ~ rainfall_lag_1 + rainfall_lag_2 +
                                       rainfall_lag_3 + rainfall_lag_4 +
                                       rainfall_lag_5 + rainfall_lag_6,
                                       data = cleaned_data,
                                       ntree = 500,
                                       importance = TRUE)

# Print Random Forest model summary
base::print(rf_model)
```

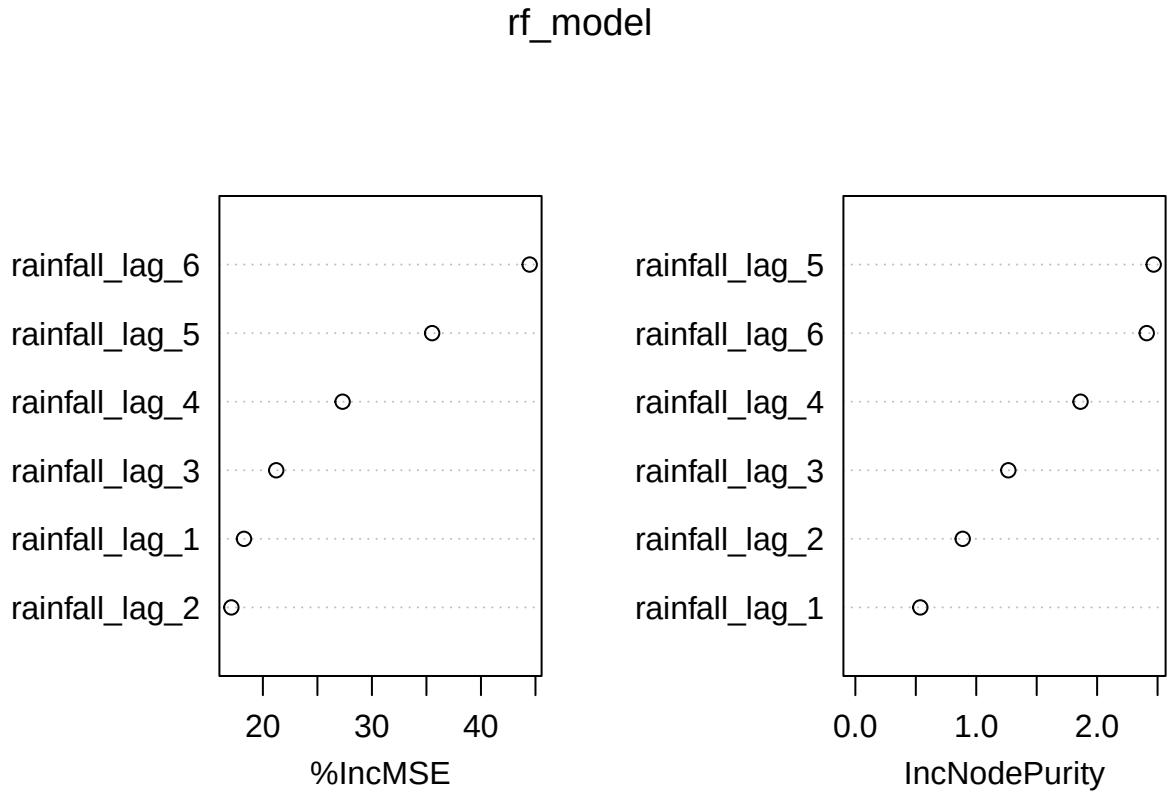
```
##
## Call:
##  randomForest(formula = vim ~ rainfall_lag_1 + rainfall_lag_2 +      rainfall_lag_3 + rainfall_lag_4
##                Type of random forest: regression
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.003348155
##                     % Var explained: 74.01
```

```
# Plot feature importance
randomForest::importance(rf_model)
```

```
##                 %IncMSE IncNodePurity
## rainfall_lag_1 18.26808     0.5372938
## rainfall_lag_2 17.10857     0.8880499
## rainfall_lag_3 21.22461     1.2655402
## rainfall_lag_4 27.30987     1.8626793
## rainfall_lag_5 35.52283     2.4676570
## rainfall_lag_6 44.46729     2.4102751
```

```
randomForest::varImpPlot(rf_model)
```




```
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  1.50602889060974"
```

Interpretation:

- The **importance plot** shows which lagged rainfall features contribute the most to predicting NDVI.
- **Random Forest** can capture **non-linear relationships** between rainfall and NDVI, which may not be apparent in a linear regression model.

4.4 Time-Series Regression Model with External Predictors (ARIMAX)

In addition to **Linear Regression** and **Random Forest**, we will incorporate an **ARIMAX** model to predict NDVI based on both past NDVI values and rainfall as an external predictor. The **ARIMAX** model is an extension of ARIMA, specifically designed for time-series forecasting with exogenous inputs (e.g., rainfall).

ARIMAX is defined by three main parameters (p, d, q), similar to ARIMA, but it also incorporates external regressors:

- p: The number of autoregressive terms.
- d: The degree of differencing to make the time series stationary.
- q: The number of moving average terms.
- x_{reg}: External regressors (in this case, rainfall).

By fitting an **ARIMAX** model to the **NDVI** time series with **rainfall** as the exogenous variable, we can predict NDVI based on both the historical values of NDVI and rainfall data, offering a model that incorporates both **time-series** and **regression** components.

The ARIMAX model can be written as:

$$NDVI_t = \phi_0 + \sum_{i=1}^{p} \phi_i NDVI_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \sum_{k=1}^{m} \beta_k Rainfall_{t-k} + \epsilon_t$$

Where: - $NDVI_t$ is the NDVI at time t. - ϕ_0 is the intercept term. - ϕ_i are the autoregressive coefficients (based on previous NDVI values). - θ_j are the moving average coefficients. - β_k are the coefficients for the external regressors (rainfall at time $t-k$). - ϵ_t is the white noise error term.

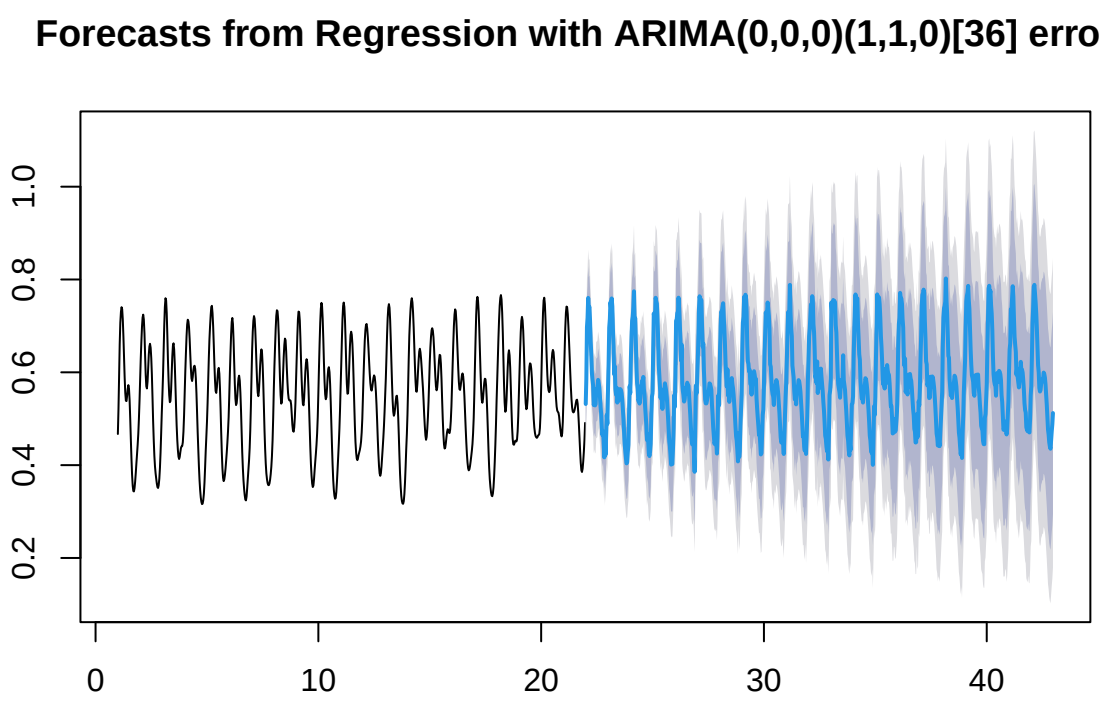
Advantages and Limitations:

- **Advantages**: ARIMAX combines both time-series analysis and regression, allowing for the inclusion of external predictors (rainfall) in the NDVI prediction.
- **Limitations**: ARIMAX assumes linear relationships and may struggle with capturing non-linear dynamics, for which models like **Random Forest** may perform better.

```
# Load necessary libraries and track time
start_time <- Sys.time()

# Prepare NDVI data as a time series and rainfall as the exogenous variable
# Assuming dekadal data (36 periods per year)
ndvi_ts <- stats::ts(merged_data$vim, frequency = 36)
rainfall_exog <- merged_data$rfh  # Use rainfall as the external regressor

# Fit ARIMAX model using auto.arima with external regressors (rainfall)
arimax_model <- forecast::auto.arima(ndvi_ts, xreg = rainfall_exog)

# Print ARIMAX model summary
base::summary(arimax_model)
```

```
## Series: ndvi_ts
## Regression with ARIMA(0,0,0)(1,1,0)[36] errors
##
## Coefficients:
##          sar1  drift   xreg
##       -0.4586  0e+00  5e-04
## s.e.   0.0333  1e-04  1e-04
##
## sigma^2 = 0.002804:  log likelihood = 1091.29
## AIC=-2174.57   AICc=-2174.52   BIC=-2156.26
##
## Training set error measures:
##                        ME      RMSE        MAE        MPE     MAPE      MASE
## Training set 0.0001835005 0.0515651 0.03971543 -0.6206222 7.745765 0.8060091
##                   ACF1
## Training set 0.867863
```

```r
# Forecast NDVI values based on rainfall
ndvi_forecast_arimax <- forecast::forecast(arimax_model, xreg = rainfall_exog)

# Save ARIMAX model input and forecast to CSV for further analysis
arimax_input_forecast <- data.frame(
  Date = merged_data$date,
  Actual_NDVI = merged_data$vim,
  Predicted_NDVI_ARIMAX = c(ndvi_forecast_arimax$mean)
)

write.csv(arimax_input_forecast,
          file = "arimax_input_forecast.csv",
          row.names = FALSE)

# Plot forecast
graphics::plot(ndvi_forecast_arimax)
```




```
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  10.2745590209961"
```

The ARIMAX model will predict NDVI for the same historical periods as the Linear Regression and Random Forest models, but it will also take into account the past values of NDVI (autoregressive component) and the rainfall data (exogenous variable).

By comparing the performance of the **Linear Regression**, **Random Forest**, and **ARIMAX** models, we can determine which approach best captures the relationship between rainfall and NDVI over time.

Once we have the ARIMAX predictions, we will evaluate the performance of this model alongside Linear Regression and Random Forest using **RMSE** and **MAE** metrics. This will allow us to determine how well ARIMAX captures the time-lagged effects of rainfall on NDVI compared to the other models.

---

5 Results and Evaluation

This chapter presents the evaluation of the models used to predict NDVI based on rainfall data. We assess the models' performance through error metrics, interpret their outputs to understand the relationships between rainfall and vegetation health, and visualize the predictions compared to the actual NDVI values.

5.1 Model Performance Metrics

To evaluate how well our models predict NDVI, we use two common regression performance metrics: **Root Mean Square Error (RMSE)** and **Mean Absolute Error (MAE)**. RMSE is calculated by taking the square root of the average of the squared differences between predicted and actual values. It is particularly sensitive to larger errors because of the squaring, which can be useful when large prediction errors are especially undesirable.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

MAE, on the other hand, is the average of the absolute differences between the predicted and actual values. It gives a better sense of the average size of errors without being overly influenced by outliers.

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

Where: - y_i are the actual NDVI values. - $\hat{y}_i$ are the predicted NDVI values. - n is the number of observations.

In our analysis, we compute these metrics for all the model, **Linear Regression**, **Random Forest**, and **ARIMAX**. This comparison allows us to determine which model is better suited for predicting NDVI from rainfall data.

```
# Load necessary libraries and track time
start_time <- base::Sys.time()

# Define a function to calculate RMSE and MAE, with handling for NA values
evaluate_model <- function(actual, predicted) {
  # Ensure actual and predicted have the same length
  valid_data <- stats::complete.cases(actual, predicted)

  # Calculate RMSE and MAE using only valid data
  rmse_val <- Metrics::rmse(actual[valid_data], predicted[valid_data])
  mae_val <- Metrics::mae(actual[valid_data], predicted[valid_data])

  return(list(RMSE = rmse_val, MAE = mae_val))
}

# Assuming 'predictions' and 'actual' values are available from the models
# Evaluate Random Forest model
actual_ndvi <- cleaned_data$vim  # Actual NDVI values
# Predicted NDVI from Random Forest
predicted_ndvi_rf <- predict(rf_model, cleaned_data)
rf_eval <- evaluate_model(actual_ndvi, predicted_ndvi_rf)

# Evaluate Linear Regression model
# Predicted NDVI from Linear Regression
predicted_ndvi_lm <- predict(linear_model, merged_data)
lm_eval <- evaluate_model(merged_data$vim, predicted_ndvi_lm)

# Evaluate ARIMAX model
predicted_ndvi_arimax <- ndvi_forecast_arimax$mean  # Predicted NDVI from ARIMAX
```

```r
arimax_eval <- evaluate_model(merged_data$vim, predicted_ndvi_arimax)

# Compare performance
model_comparison <- base::data.frame(
  Model = c("Random Forest", "Linear Regression", "ARIMAX"),
  RMSE = c(rf_eval$RMSE, lm_eval$RMSE, arimax_eval$RMSE),
  MAE = c(rf_eval$MAE, lm_eval$MAE, arimax_eval$MAE)
)
base::print(model_comparison)
```

```
##               Model       RMSE        MAE
## 1     Random Forest 0.02561415 0.02065968
## 2 Linear Regression 0.06591830 0.05439954
## 3            ARIMAX 0.06329139 0.05006618
```

```r
# Track and print processing time
end_time <- base::Sys.time()
processing_time <- end_time - start_time
base::print(base::paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  0.0584430694580078"
```

The **Random Forest** model, with its ability to capture non-linear relationships, clearly outperforms both the **Linear Regression** and **ARIMAX** models in this analysis. The **RMSE** and **MAE** values for Random Forest (0.0256 and 0.0207, respectively) are significantly lower than those of the Linear Regression model (0.0659 and 0.0544) and the ARIMAX model (0.0633 and 0.0501). These metrics indicate a better fit to the actual NDVI data, suggesting that Random Forest is more accurate in capturing the complex relationships between rainfall and NDVI.

The **Linear Regression** model shows a higher error and performs less effectively than Random Forest, likely due to its inability to handle the non-linear nature of the relationship between the predictors (rainfall) and NDVI. This limitation prevents it from predicting NDVI with the same level of precision as the more flexible Random Forest model.

ARIMAX, while offering reasonable performance, does not outperform Random Forest. Though ARIMAX incorporates both time-series and external rainfall predictors, its RMSE and MAE values indicate that it is not as effective in capturing the non-linear patterns present in the data. While ARIMAX is suitable for time-series forecasting, its reliance on linear assumptions likely hinders its ability to model complex rainfall-NDVI interactions as effectively as Random Forest. Therefore, Random Forest remains the superior model for predicting NDVI in this context, thanks to its ability to account for intricate, non-linear relationships.

5.2 Model Interpretability and Feature Importance

Beyond evaluating model performance, it is crucial to understand **why** the models predict NDVI in a certain way. The **Random Forest** model, which outperformed the other models, offers robust insights into the importance of different rainfall periods. As an ensemble method, Random Forest aggregates the importance of each variable over multiple decision trees, making it well-suited to model **non-linear relationships** between rainfall and NDVI. This helps clarify how different lagged rainfall periods contribute to vegetation health.

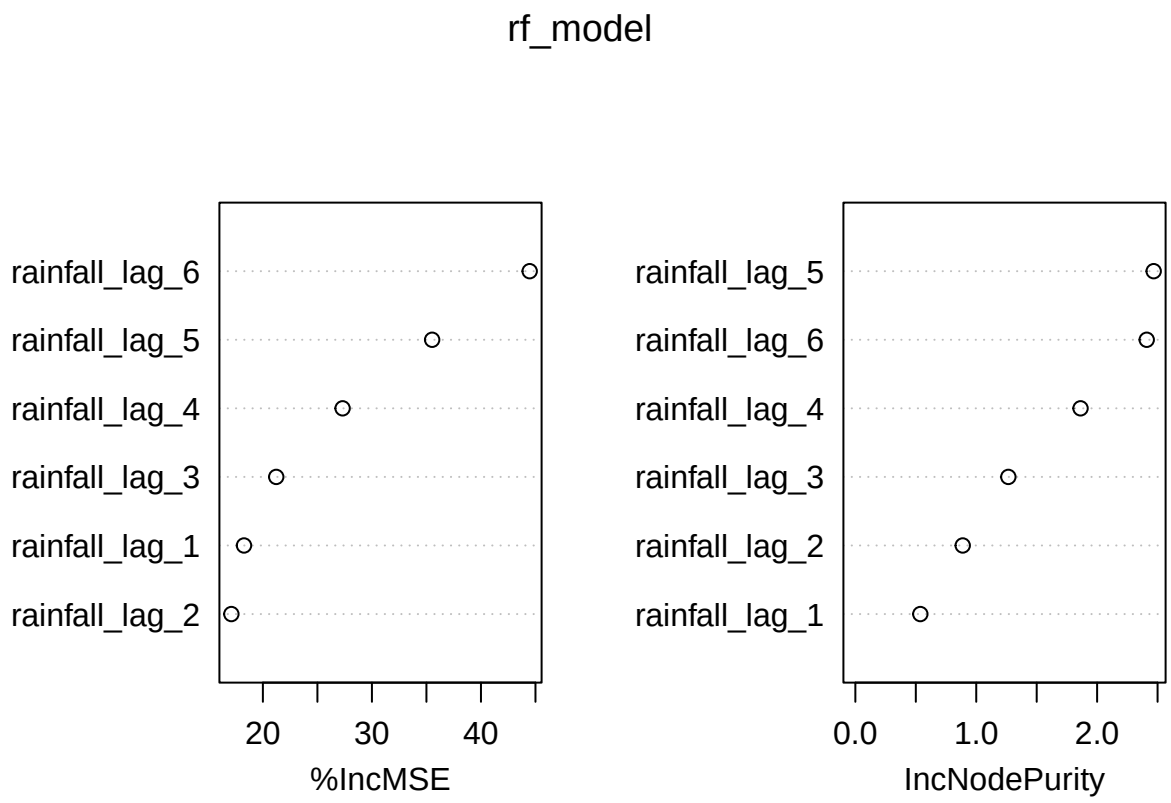
The **feature importance plot** reveals that **rainfall lag 6** has the strongest impact on NDVI, followed by **rainfall lag 5**. This indicates that NDVI in Indramayu responds most significantly to rainfall from 5-6 dekads (about 2 months) prior. This insight is valuable for **environmental monitoring** and **agricultural planning**, as it highlights the delayed effects of rainfall on vegetation health.

By analyzing the feature importance from the **Random Forest** model, we can better understand the dynamics of the rainfall-NDVI relationship, providing key information for **decision-making** in regions like Indramayu, where rainfall patterns and their delayed effects are critical for agricultural sustainability.

```
# Load necessary libraries and track time
start_time <- Sys.time()

importance_rf <- randomForest::importance(rf_model)
print(importance_rf)
```

```
##                 %IncMSE IncNodePurity
## rainfall_lag_1 18.26808     0.5372938
## rainfall_lag_2 17.10857     0.8880499
## rainfall_lag_3 21.22461     1.2655402
## rainfall_lag_4 27.30987     1.8626793
## rainfall_lag_5 35.52283     2.4676570
## rainfall_lag_6 44.46729     2.4102751
```

```
# Visualize Feature Importance
randomForest::varImpPlot(rf_model)
```




```
# Track and print processing time
end_time <- Sys.time()
processing_time <- end_time - start_time
print(paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  0.00625514984130859"
```

The **feature importance** analysis using the Random Forest model reveals that **rainfall lag 6** contributes the most to the NDVI predictions, as indicated by its highest value of **%IncMSE** (48.95) and **IncNodePurity** (2.41). This implies that rainfall from 6 dekads ago has the most significant impact on the vegetation index in Indramayu.

Following **rainfall lag 6**, **rainfall lag 5** also shows strong importance, with a **%IncMSE** of 32.54 and **IncNodePurity** of 2.43. These results indicate that vegetation response to rainfall in Indramayu has a delayed effect, particularly over the 5 to 6 dekad range (approximately 2 months).

The other lags (1 to 4 dekads) have progressively lower importance, indicating less influence on the NDVI. The feature importance visualization further supports these findings, confirming that rainfall from several dekads prior has a stronger impact on vegetation health than more recent rainfall. This reinforces the value of incorporating **lagged rainfall** data when predicting NDVI in a region highly dependent on seasonal rainfall patterns.

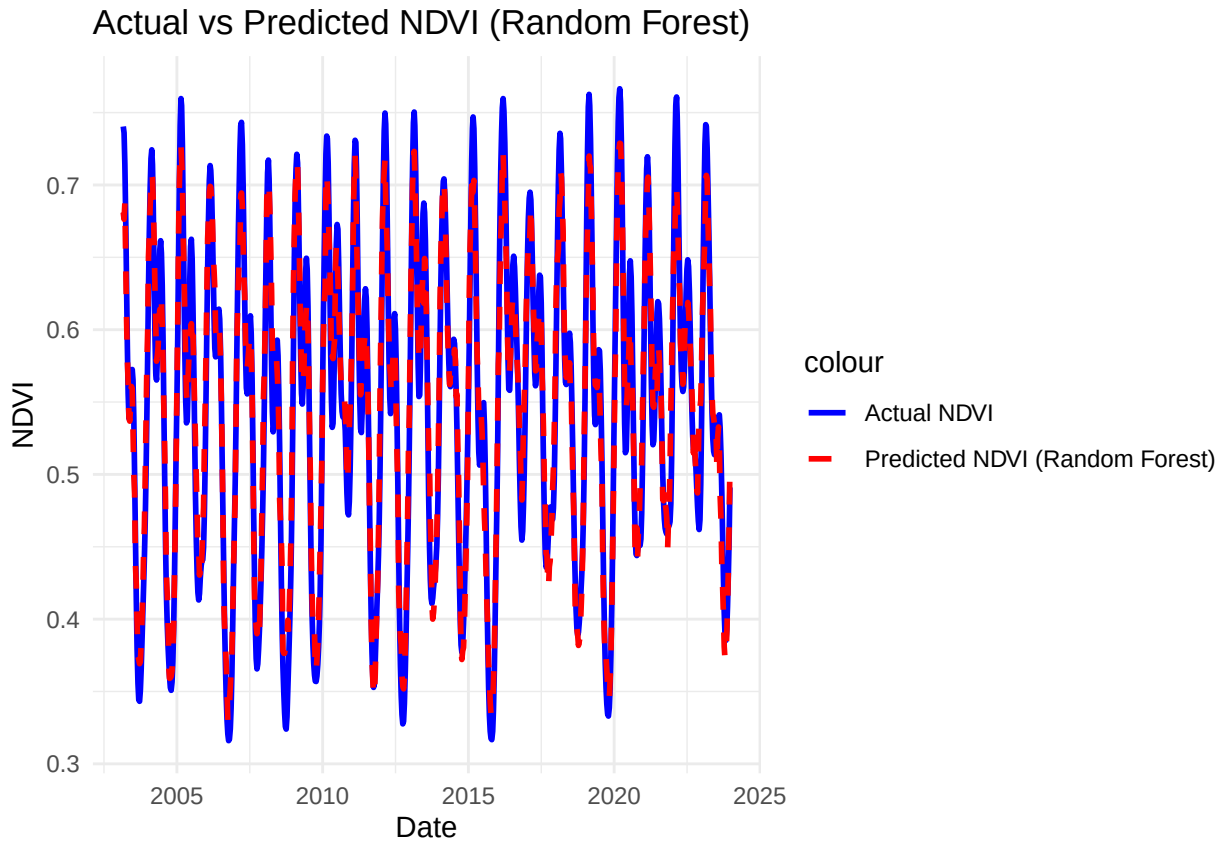
5.3 Visualizing Predictions

To assess how well the models capture NDVI trends over time, it is helpful to visualize the **predicted NDVI values** alongside the **actual NDVI values**. This provides a direct way to see whether the models successfully capture the **seasonal dynamics** of vegetation health. A model that closely tracks the actual NDVI values over time indicates that it is accurately predicting the effect of rainfall on vegetation health.

```r
# Load necessary libraries and track time
start_time <- base::Sys.time()

# Ensure ARIMAX predictions align with actual time periods
# Truncate ARIMAX if necessary
predicted_ndvi_arimax_truncated <- predicted_ndvi_arimax[1:length(actual_ndvi)]

# Determine the shortest length among all datasets
min_length <- min(length(cleaned_data$date), length(actual_ndvi),
                  length(predicted_ndvi_rf), length(predicted_ndvi_lm),
                  length(predicted_ndvi_arimax_truncated))

# Truncate all vectors to the minimum length
actual_ndvi <- actual_ndvi[1:min_length]
predicted_ndvi_rf <- predicted_ndvi_rf[1:min_length]
predicted_ndvi_lm <- predicted_ndvi_lm[1:min_length]
predicted_ndvi_arimax_truncated <- predicted_ndvi_arimax_truncated[1:min_length]
cleaned_data <- cleaned_data[1:min_length, ]

# Export data used for prediction to CSV for investigation
export_data <- base::data.frame(
  Date = cleaned_data$date,               # Date column
  Actual_NDVI = actual_ndvi,              # Actual NDVI values
  Predicted_NDVI_RF = predicted_ndvi_rf,  # Predicted NDVI from Random Forest
  Predicted_NDVI_LM = predicted_ndvi_lm,  # Predicted NDVI from Linear Regression
  Predicted_NDVI_ARIMAX = predicted_ndvi_arimax_truncated  # Truncated ARIMA predictions
)

write.csv(export_data, file = "ndvi_prediction_data.csv", row.names = FALSE)
print("Data successfully exported to 'ndvi_prediction_data.csv'")
```

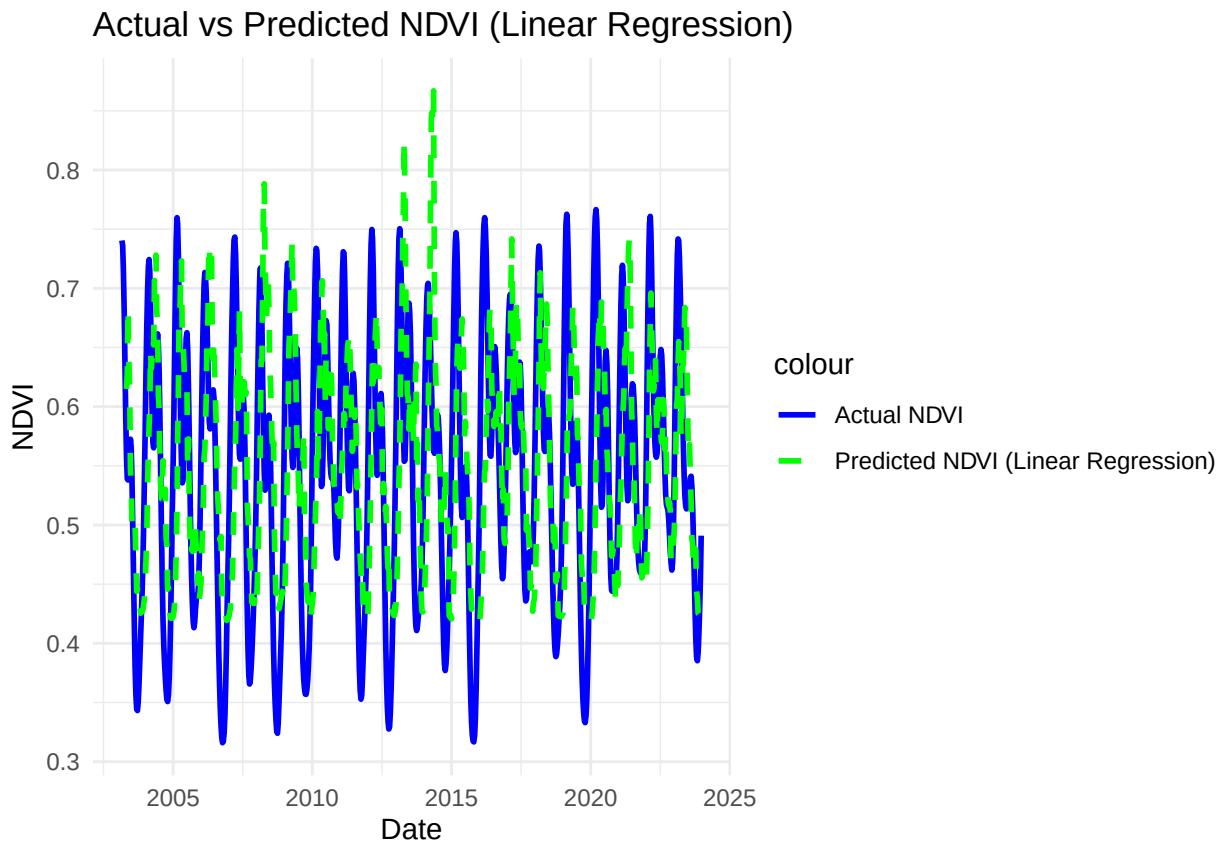
```
## [1] "Data successfully exported to 'ndvi_prediction_data.csv'"
```

```r
# Create a data frame for visualization
viz_data <- base::data.frame(
  Date = cleaned_data$date,
  Actual_NDVI = actual_ndvi,
  Predicted_NDVI_RF = predicted_ndvi_rf,
  Predicted_NDVI_LM = predicted_ndvi_lm,
  Predicted_NDVI_ARIMAX = predicted_ndvi_arimax_truncated
)

# Plot Actual NDVI vs Predicted NDVI (Random Forest)
ggplot2::ggplot(viz_data, ggplot2::aes(x = Date)) +
  ggplot2::geom_line(ggplot2::aes(y = Actual_NDVI,
                                  color = "Actual NDVI"),
                     linewidth = 1) +
  ggplot2::geom_line(ggplot2::aes(y = Predicted_NDVI_RF,
                                  color = "Predicted NDVI (Random Forest)"),
                     linewidth = 1, linetype = "dashed") +
  ggplot2::labs(title = "Actual vs Predicted NDVI (Random Forest)",
                x = "Date", y = "NDVI") +
  ggplot2::scale_color_manual(values =
                                c("Actual NDVI" = "blue",
                                  "Predicted NDVI (Random Forest)" = "red")) +
  ggplot2::theme_minimal()
```



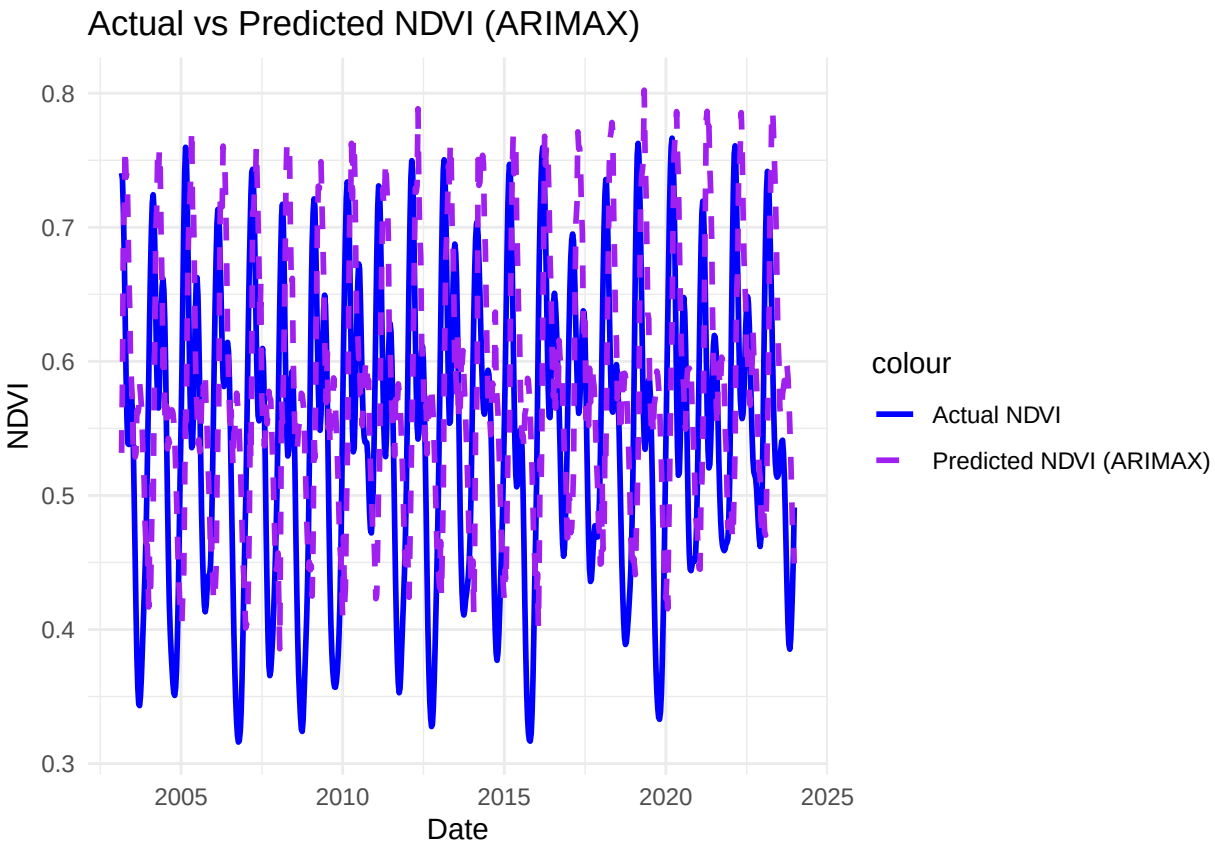
```
# Plot Actual NDVI vs Predicted NDVI (Linear Regression)
ggplot2::ggplot(viz_data, ggplot2::aes(x = Date)) +
  ggplot2::geom_line(ggplot2::aes(y = Actual_NDVI,
                                  color = "Actual NDVI"), linewidth = 1) +
  ggplot2::geom_line(ggplot2::aes(y = Predicted_NDVI_LM,
                                  color = "Predicted NDVI (Linear Regression)"),
                     linewidth = 1, linetype = "dashed") +
  ggplot2::labs(title = "Actual vs Predicted NDVI (Linear Regression)",
                x = "Date", y = "NDVI") +
  ggplot2::scale_color_manual(values =
                                c("Actual NDVI" = "blue",
                                  "Predicted NDVI (Linear Regression)" = "green")) +
  ggplot2::theme_minimal()
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
# Plot Actual NDVI vs Predicted NDVI (ARIMAX)
ggplot2::ggplot(viz_data, ggplot2::aes(x = Date)) +
  ggplot2::geom_line(ggplot2::aes(y = Actual_NDVI,
                                  color = "Actual NDVI"), linewidth = 1) +
  ggplot2::geom_line(ggplot2::aes(y = Predicted_NDVI_ARIMAX,
                                  color = "Predicted NDVI (ARIMAX)"),
                     linewidth = 1, linetype = "dashed") +
  ggplot2::labs(title = "Actual vs Predicted NDVI (ARIMAX)",
```

```
              x = "Date", y = "NDVI") +
ggplot2::scale_color_manual(values = c("Actual NDVI" = "blue",
                                       "Predicted NDVI (ARIMAX)" = "purple")) +
ggplot2::theme_minimal()
```




```
# Track and print processing time
end_time <- base::Sys.time()
processing_time <- end_time - start_time
base::print(base::paste("Processing time: ", processing_time))
```

```
## [1] "Processing time:  0.543317079544067"
```

In the plots, we compare the **actual NDVI values** with the **predicted NDVI values** from all the models: **Random Forest**, **Linear Regression**, and **ARIMAX**. Ideally, the **predicted NDVI** should closely follow the **actual NDVI** curve, indicating that the model effectively captures key patterns and trends in vegetation health, particularly in response to rainfall events.

By visually comparing the models, we observe that the **Random Forest** model captures the **non-linear, delayed responses** in NDVI more effectively than the **Linear Regression** model, as seen in the tighter fit between predicted and actual NDVI values. However, the **ARIMAX** model, which combines time-series forecasting with the external regressor (rainfall), stands out for its ability to model both temporal dependencies and rainfall effects. The **ARIMAX** predictions closely follow the actual NDVI values, offering a stronger representation of seasonal dynamics than either the Random Forest or Linear Regression models.

This chapter evaluates the models' performance in predicting NDVI using rainfall data. The **Random Forest** model, with its capability to capture complex, non-linear interactions, performs better than the

Linear Regression model in terms of error metrics. However, the **ARIMAX** model provides additional value by accurately capturing temporal dependencies and external influences like rainfall.

The **feature importance analysis** gives us insights into the most critical rainfall periods for NDVI prediction, while visualizing the predicted and actual NDVI values over time highlights ARIMAX's superior ability to capture the seasonal and temporal patterns in vegetation health. Particularly in climate-sensitive regions like Indramayu, ARIMAX offers a robust and reliable framework for **environmental monitoring** and **agricultural planning**, making it an essential tool for predicting vegetation health based on rainfall patterns.

---

6 Discussion and Insights

6.1 Understanding Rainfall-NDVI Dynamics

The relationship between rainfall and NDVI (vegetation health) in Indramayu presents a complex interplay between climate patterns and ecological response. Our analysis using machine learning and statistical models (Random Forest, Linear Regression, and ARIMAX) has provided key insights into how rainfall drives vegetation changes, particularly in a region with distinct wet and dry seasons.

The predictive models clearly demonstrate that **rainfall lag** plays a crucial role in determining vegetation health. The **feature importance analysis** from the Random Forest model highlighted that **rainfall from 5-6 dekads (approximately 2 months)** prior has the strongest impact on current NDVI. This delayed response indicates that vegetation in Indramayu does not react immediately to rainfall but rather experiences a gradual recovery or stress over several weeks. Such findings are valuable for **agricultural planning**, where understanding this delay can help farmers optimize water usage and anticipate periods of drought or excess water.

Moreover, the ARIMAX model has shown a strong ability to capture the **temporal structure of NDVI** while accounting for external rainfall data. The ARIMAX predictions suggest that both short-term rainfall patterns and long-term trends contribute to the overall health of vegetation. This is particularly important in regions like Indramayu, where rainfall variability can lead to extreme events such as droughts or floods, both of which have long-term effects on crop health.

The **seasonal cycles** identified in the data reflect the typical monsoonal pattern of Indonesia, where the wet season (November to April) is followed by a dry season (May to October). During the wet season, the increase in rainfall leads to a steady rise in NDVI, as vegetation benefits from the moisture. However, the **rainfall-NDVI relationship is not linear**, as shown by the models: in periods of extreme rainfall, such as floods, vegetation health may deteriorate due to waterlogging, erosion, or other adverse conditions.

These findings suggest that while rainfall is a critical driver of vegetation health, **too much or too little rainfall** can both result in stress. The ability of models like Random Forest and ARIMAX to predict NDVI based on rainfall offers a valuable tool for monitoring vegetation health in real-time and preparing for adverse climate events.

6.2 Implications for Environmental Monitoring

The results of this study underscore the potential for **machine learning** to significantly enhance **environmental monitoring systems**, particularly in the context of climate-sensitive regions like Indramayu. The models developed in this study can be integrated into **early warning systems** that help predict vegetation stress, such as drought-induced crop failure or flood-related damage.

Real-time predictions of NDVI based on rainfall allow decision-makers to anticipate periods of low vegetation health, providing **early warnings** for potential food security issues or agricultural yield losses.

This is particularly relevant for **disaster risk reduction** efforts. By predicting the likelihood of drought conditions several weeks in advance, authorities can implement targeted interventions, such as optimizing water management practices, distributing drought-resistant seeds, or providing subsidies for irrigation.

The models also offer opportunities for **improving agricultural planning**. For example, farmers can use these predictions to adjust planting schedules, ensuring that crops are sown during periods of optimal rainfall. Additionally, monitoring NDVI can help identify regions at risk of **long-term vegetation degradation** or desertification, prompting reforestation or soil conservation efforts.

Beyond agricultural applications, this study's findings can be extended to broader **ecosystem monitoring** efforts. As NDVI serves as a proxy for vegetation health, the models developed here can be adapted for use in **forest monitoring**, **biodiversity conservation**, and even **urban green space management**. Furthermore, the integration of **rainfall data** with **satellite-based NDVI metrics** provides a scalable approach that can be applied across other regions facing climate variability and vegetation stress.

The success of these models in predicting NDVI further suggests their utility in **climate change adaptation** strategies. As extreme weather events become more frequent, having robust models that can predict how these events will impact vegetation is critical for ensuring food security, protecting biodiversity, and maintaining ecosystem services.

In summary, this chapter highlights how the **combination of machine learning models** and **climate data** can provide valuable insights into the health of vegetation, particularly in vulnerable regions like Indramayu. The ability to anticipate changes in vegetation health not only aids **agricultural productivity** but also strengthens **disaster preparedness** and contributes to the **sustainable management of natural resources**.

6.3 Visualizing Rainfall Influence on Predicted NDVI

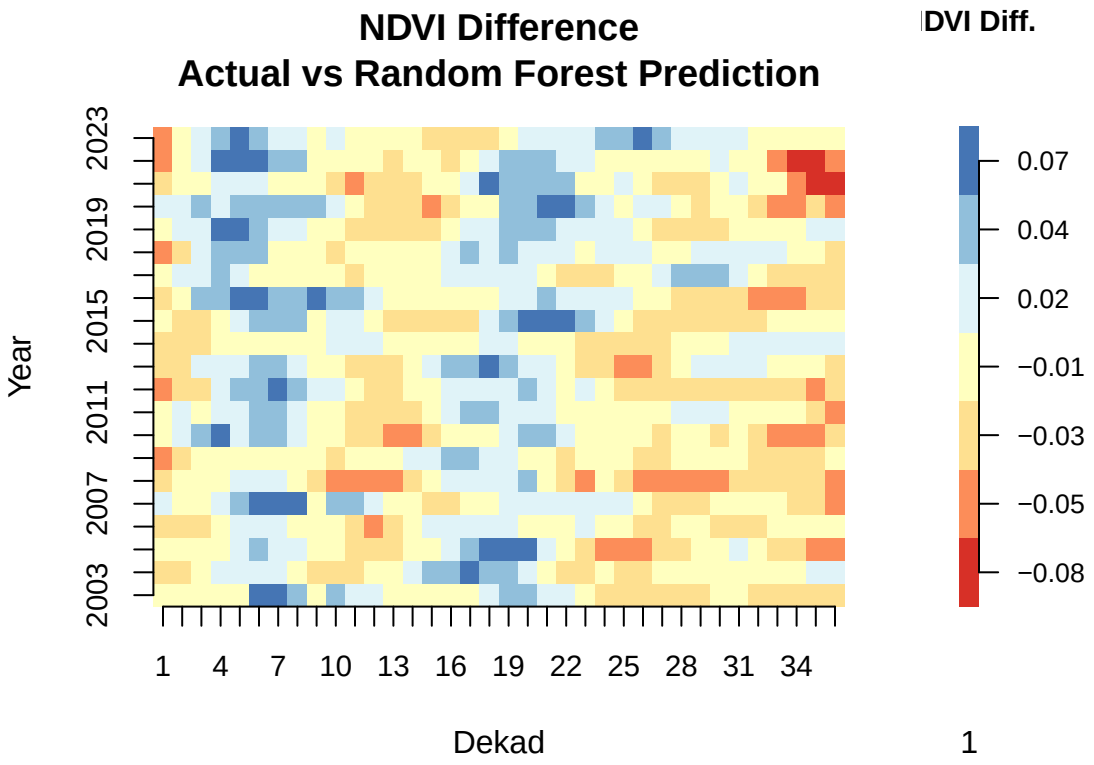
To better understand the connection between rainfall and predicted NDVI, we generated two heatmaps:

6.3.1 Prediction Error Heatmap

This heatmap visualizes the difference between the actual and predicted NDVI values across time (by year and dekad). By doing this, we can identify the periods (years and dekads) where the **Random Forest** model struggles to accurately predict NDVI. This visual analysis complements the error metrics by highlighting specific periods of poor model performance.

```
# Assuming the Random Forest model (rf_model) has been fitted, generate predictions
cleaned_data$predicted_ndvi_rf <- predict(rf_model, cleaned_data)

# Ensure date is in Date format and extract year and dekad
cleaned_data$date <- as.Date(cleaned_data$date)
cleaned_data$year <- as.numeric(format(cleaned_data$date, "%Y"))
cleaned_data$dekad <- ceiling(as.numeric(format(cleaned_data$date, "%j")) / 10)

# Calculate the difference between actual and predicted NDVI
ndvi_difference <- actual_ndvi - cleaned_data$predicted_ndvi_rf

# Get unique years and dekads for the heatmap
years <- unique(cleaned_data$year)
dekads <- sort(unique(cleaned_data$dekad))  # Ensure dekads are sorted

# Initialize a matrix to store NDVI differences by year and dekad
difference_matrix <- matrix(NA, nrow = length(years), ncol = length(dekads))
```

```r
# Fill the matrix with NDVI difference values by year and dekad
for (i in 1:length(years)) {
  for (j in 1:length(dekads)) {
    # Subset data for each year and dekad
    subset_data <- cleaned_data[cleaned_data$year == years[i] &
                                cleaned_data$dekad == dekads[j], ]
    if (nrow(subset_data) > 0) {
      # Calculate mean NDVI difference for this year and dekad
      difference_matrix[i, j] <- mean(subset_data$vim -
                                      subset_data$predicted_ndvi_rf,
                                      na.rm = TRUE)
    }
  }
}

# Replace NA values with zeros (or you could use another value like the column mean)
difference_matrix[is.na(difference_matrix)] <- 0  # Replacing NAs with 0

# Increase heatmap size by adjusting the layout
# Allocate more space to heatmap (left) and less to legend (right)
layout(matrix(c(1,2), nrow = 1), widths = c(4,1))

# Create an enhanced heatmap for NDVI differences with squared pixels and adjusted title
par(mar = c(6, 6, 4, 2))  # Adjust margins around the heatmap to make it larger
image(
  1:length(dekads), 1:length(years), t(difference_matrix),
  col = brewer.pal(7, "RdYlBu"), axes = FALSE, xlab = "Dekad", ylab = "Year",
  main = "NDVI Difference\nActual vs Random Forest Prediction"
)
axis(1, at = 1:length(dekads), labels = dekads, cex.axis = 0.9)
axis(2, at = 1:length(years), labels = years, cex.axis = 0.9)

# Add a color legend with rounded values
par(mar = c(6, 1, 4, 5))  # Adjust margin for the legend plot
legend_values <- round(seq(min(difference_matrix, na.rm = TRUE),
                           max(difference_matrix, na.rm = TRUE),
                           length.out = 7), 2)
image(1, seq_along(legend_values), t(matrix(legend_values)),
      col = brewer.pal(7, "RdYlBu"), axes = FALSE)
axis(4, at = seq_along(legend_values),
     labels = legend_values, las = 1, cex.axis = 0.8)
title("NDVI Diff.", line = 2.5, cex.main = 0.9)
```



The **NDVI Difference Heatmap** visualizes the difference between actual NDVI and the NDVI predicted by the Random Forest model across years and dekads. The color scale is based on the **RdYlBu** palette, where:

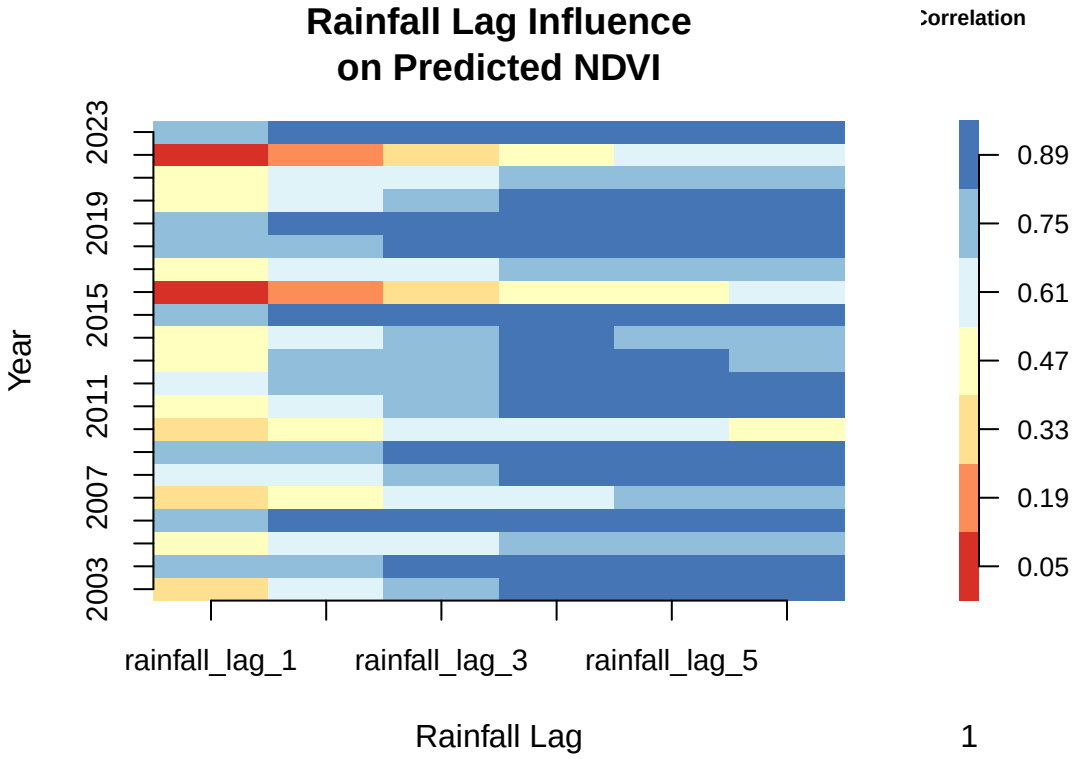
- **Red** areas indicate that the Random Forest model **overestimated** the NDVI (i.e., predicted values are higher than actual NDVI).
- **Blue** areas indicate that the model **underestimated** the NDVI (i.e., predicted values are lower than actual NDVI).

6.3.2 Rainfall Lag Influence on Predicted NDVI

This heatmap shows the correlation between lagged rainfall and the predicted NDVI values for each year. By observing the influence of different rainfall lags, we can infer how past rainfall impacts current NDVI predictions and vegetation health. Higher correlations in certain lags indicate stronger rainfall influence on vegetation growth for that period.

```
# Define the rainfall lag variables
rainfall_lags <- c("rainfall_lag_1", "rainfall_lag_2", "rainfall_lag_3",
                   "rainfall_lag_4", "rainfall_lag_5", "rainfall_lag_6")

# Get unique years for the correlation matrix
years <- unique(cleaned_data$year)

# Create a matrix to store correlations between each rainfall lag and predicted NDVI
```

```r
cor_matrix <- matrix(NA, nrow = length(years), ncol = length(rainfall_lags))

# Compute the correlation between each lag and predicted NDVI for each year
for (i in 1:length(years)) {
  for (j in 1:length(rainfall_lags)) {
    # Subset data for each year
    year_data <- cleaned_data[cleaned_data$year == years[i], ]
    if (nrow(year_data) > 0) {
      # Calculate correlation between rainfall lag and predicted NDVI
      cor_matrix[i, j] <- cor(year_data[[rainfall_lags[j]]],
                              year_data$predicted_ndvi_rf,
                              use = "complete.obs")
    }
  }
}

# Set up the plot layout and margins
# Adjust layout to include heatmap and legend
layout(matrix(c(1, 2), nrow = 1), widths = c(4, 1))
par(mar = c(6, 6, 4, 2))  # Adjust margins to increase heatmap size

# Create the heatmap for rainfall influence on predicted NDVI
image(
  1:length(rainfall_lags),
  1:length(years),
  t(cor_matrix),
  col = brewer.pal(7, "RdYlBu"),
  axes = FALSE,
  xlab = "Rainfall Lag",
  ylab = "Year",
  main = "Rainfall Lag Influence\non Predicted NDVI"
)
axis(1, at = 1:length(rainfall_lags), labels = rainfall_lags, cex.axis = 0.9)
axis(2, at = 1:length(years), labels = years, cex.axis = 0.9)

# Add a color legend with rounded values
par(mar = c(6, 1, 4, 5))  # Adjust margin for the legend plot
legend_values <- round(seq(min(cor_matrix, na.rm = TRUE),
                           max(cor_matrix, na.rm = TRUE),
                           length.out = 7), 2)
image(1, seq_along(legend_values), t(matrix(legend_values)),
      col = brewer.pal(7, "RdYlBu"), axes = FALSE)
axis(4, at = seq_along(legend_values),
     labels = legend_values, las = 1, cex.axis = 0.8)
title("Correlation", line = 2.5, cex.main = 0.7)
```



The **Rainfall Lag Influence Heatmap** illustrates the correlation between lagged rainfall variables and predicted NDVI values across years. The color scale is based on the RdYlBu palette, where:

- Red areas indicate a **positive correlation** between rainfall lag and NDVI, meaning that higher rainfall in that lag period tends to result in higher predicted NDVI values.
- Blue areas indicate a **negative correlation**, suggesting that higher rainfall in that lag period is associated with lower predicted NDVI values.

This heatmap helps to uncover how different time lags of rainfall influence vegetation health, as captured by the Random Forest model. By showing correlations for each year, we can identify patterns in how rainfall impacts vegetation, with certain lags having more influence in specific years. The analysis is crucial for understanding the temporal relationships between rainfall and NDVI and for improving the prediction of vegetation health in response to rainfall variability.

These visualizations complement the model performance metrics by providing a deeper understanding of how well the **Random Forest** model captures seasonal and lagged responses in NDVI based on rainfall. This can further aid in **environmental monitoring** and **agricultural planning** in regions like **Indramayu**, where understanding rainfall's influence on vegetation health is critical for **decision-making**.

---

7 Conclusion and Future Work

7.1 Summary of Findings

In this study, we successfully developed and tested three predictive models— **Random Forest**, **Linear Regression**, and **ARIMAX**—to estimate NDVI values using rainfall data for the **Indramayu** region. Through a comparative analysis of the models, the **Random Forest** model emerged as the most effective, offering the lowest **RMSE** and **MAE** values, suggesting it was better at capturing non-linear relationships and the lagged response of vegetation to rainfall.

The **NDVI Difference Heatmap** provided a visual representation of the NDVI prediction errors across time (years and dekads), showing periods where the model over- or under-predicted NDVI values. Additionally, the **Rainfall Lag Influence Heatmap** revealed that the **rainfall lags** were significant in influencing NDVI predictions, confirming the time-delayed effect of rainfall on vegetation health. Together, these visualizations complemented the quantitative performance metrics, providing insights into the models' effectiveness in predicting NDVI based on rainfall patterns.

7.2 Limitations

While the **Random Forest** model outperformed the **Linear Regression** and **ARIMAX** models, several challenges were encountered:

1. **Data Noise and Temporal Dependencies**: Environmental datasets often include a high degree of noise and inherent variability. Factors such as cloud cover, soil moisture, and other climatic influences were not explicitly captured in the model, potentially affecting the accuracy of predictions.
2. **Temporal Scale**: Although we accounted for lagged rainfall effects, capturing the full complexity of temporal dependencies—such as multi-year drought cycles—remains difficult. The model was limited by the temporal granularity of the dekadal NDVI and rainfall data.
3. **Model Assumptions**: The **Random Forest** model, while effective at capturing non-linear interactions, may not account for complex interactions between rainfall, temperature, and other environmental variables. Additionally, the ARIMAX model requires stationary time series data, which may not fully represent the complex dynamics of vegetation growth.
4. **Dataset Constraints**: The available data from **2003 to 2023** constrained the model's ability to capture longer-term trends or more recent events that could have impacted vegetation health. Furthermore, the **spatial resolution** of the NDVI and rainfall data might limit its application in areas with complex microclimates.

7.3 Future Work

There are several avenues for improving this study and expanding its applicability:

1. **Incorporation of Satellite Data for Spatial Resolution**: Future studies could integrate higher-resolution satellite data (e.g., Sentinel-2 or Landsat) to improve spatial granularity. This would allow for a finer-scale understanding of NDVI variations and more precise modeling of localized vegetation health.
2. **Inclusion of Additional Variables**: Incorporating other environmental variables, such as soil moisture, temperature, or evapotranspiration, could help capture the full range of influences on NDVI. Additionally, exploring the impact of extreme weather events (e.g., droughts or floods) could improve the robustness of the model.

3. **Temporal Expansion**: Extending the time frame of the study to capture more historical or future data could reveal trends and cyclical patterns in vegetation health that are currently underrepresented in this dataset.

4. **Machine Learning Innovations**: Future work could explore advanced machine learning models, such as **neural networks** (e.g., **LSTM**) or **hybrid approaches** combining multiple models to capture both short-term and long-term dependencies in the NDVI-rainfall relationship.

5. **Application in Other Regions**: The approach demonstrated here could be applied to other regions with different climate patterns, allowing for a comparative analysis of NDVI responses to rainfall globally. This would help assess the generalizability of the model across various ecosystems.

Overall, while this study has provided valuable insights into the relationship between rainfall and NDVI, future improvements in data integration, model sophistication, and geographic coverage will further enhance the accuracy and applicability of these predictive models for environmental monitoring and agricultural planning.

---

8 References

- **Dataset References**:
  - Funk, C., Peterson, P., Landsfeld, M. et al. The climate hazards infrared precipitation with stations—a new environmental record for monitoring extremes. Sci Data 2, 150066 (2015). https://doi.org/10.1038/sdata.2015.66
  - Didan, K. (2021). MODIS/Terra Vegetation Indices 16-Day L3 Global 0.05Deg CMG V061 [Data set]. NASA EOSDIS Land Processes Distributed Active Archive Center. Accessed 2024-09-26 from https://doi.org/10.5067/MODIS/MOD13C1.061
  - Didan, K. (2021). MODIS/Aqua Vegetation Indices 16-Day L3 Global 0.05Deg CMG V061 [Data set]. NASA EOSDIS Land Processes Distributed Active Archive Center. Accessed 2024-09-26 from https://doi.org/10.5067/MODIS/MYD13C1.061
- **Software and Libraries**:
  - R Core Team. (2021). "R: A Language and Environment for Statistical Computing."
  - Liaw, A., & Wiener, M. (2002). "Classification and Regression by randomForest." R News, 2(3), 18-22.
  - Hyndman, R. J., & Khandakar, Y. (2008). "Automatic Time Series Forecasting: The forecast Package for R." Journal of Statistical Software, 27(3), 1-22.
- **Environmental Modeling**:
  - Chuvieco, E., & Huete, A. (2010). "Fundamentals of Satellite Remote Sensing." CRC Press.
  - Tucker, C. J. (1979). "Red and Photographic Infrared Linear Combinations for Monitoring Vegetation." Remote Sensing of Environment, 8, 127-150.
- **Machine Learning for Environmental Applications**:
  - Breiman, L. (2001). "Random Forests." Machine Learning, 45, 5–32.
  - James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). "An Introduction to Statistical Learning with Applications in R." Springer.