Summary

The goal of this project is to determine whether machine learning (ML) can produce accurate short-term rainfall forecasts in tropical Africa, a region where storms are convective and highly localised, making prediction exceptionally challenging. Although they work well in mid-latitudes, traditional numerical weather prediction (NWP) models frequently perform poorly in the tropics due to their inability to resolve small-scale, rapidly evolving weather patterns. Therefore, the goal of the study was to determine whether observation-to-observation machine learning techniques that were trained solely on rainfall data could perform better than traditional baselines.

The study utilised the integrated multi-satellite retrievals for the GPM (IMERG) dataset, known for its high-resolution global rainfall records. The analysis concentrated on daily precipitation data across tropical Africa spanning from 1998 to 2024. To improve the model performance, Exploratory Data Analysis (EDA) and feature engineering techniques were applied. For classification purposes, rainfall was grouped into three categories: No Rain, Light Rain, and Heavy Rain.

Five predictive models were implemented in this study: Decision Trees, Logistic Regression, Random Forest, K-Nearest Neighbours (KNN), and a Long Short-Term Memory (LSTM) neural network. To reduce the pronounced imbalance among rainfall categories, the Synthetic Minority Oversampling Technique (SMOTE) was applied. Model performance was assessed using standard classification metrics, including accuracy, precision, recall, F1-score, and confusion matrices.

With an accuracy of 89%, the results showed that the most reliable and balanced performance was provided by simpler algorithms, most notably Logistic Regression enhanced with SMOTE. The LSTM model, on the other hand, showed early stagnation and only achieved 66.5% accuracy, highlighting the drawbacks of using precipitation data alone as input. These results highlight how important feature engineering and data balancing are to improve predictive results, even with relatively simple models.

In practical terms, the study shows that in environments with constrained computational resources, lightweight and interpretable methods can provide useful forecasting capabilities. For industries like agriculture, disaster risk reduction, and water resource management throughout tropical Africa, where accurate and timely rainfall forecasts are crucial for well-informed decision-making, this is highly significant.

Acknowledgement

I would first like to sincerely thank my supervisor, Professor Kristina Vuskovic, for her guidance and encouragement throughout this MSc project. I am especially grateful to Professor Richard Keane, who supported me in navigating a field that was initially unfamiliar to me and provided invaluable assistance whenever needed. I also extend my thanks to my assessor, Professor Haiko Müller, for his constructive feedback, which greatly improved the quality of my work.

Table of Contents

Chapter 1: Introduction

1.1 Introduction

Predicting rainfall in tropical regions, particularly in Africa, continues to be a significant scientific challenge. In tropical regions, convective storm clouds are primarily responsible for rainfall, in contrast to the steady weather systems found in places like Europe or North America. Due to their small size, unpredictability, and rapid development, these storms are extremely challenging to recognize and forecast with precision (Mathon et al., 2002; Nesbitt, Zipser, and Cecil, 2006).

Most of the forecasts today depend on Numerical weather prediction (NWP) models. These models use factors like wind, temperature, humidity, and pressure, and then apply physical equations to forecast weather. While this approach works reasonably well in the mind-latitudes, like Europe or North America, they struggle in tropical regions. One of the reasons is that NWP models break the atmosphere into big grid boxes, like looking at a low-resolution photograph. Small, fast-moving storms which drive tropical rainfall do not fit well in these grid/boxes. Instead, the models reply on simplification, known as “parameterisations”, these cannot fully capture the quick and localised behaviour of tropical storms (Bauer, Thorpe and Brunet, 2015; Marsham et al., 2011; Rasheeda Satheesh, Rajeevan and Pai, 2023).

In Africa, this limitation means that forecasts often perform marginally better than using the simple averages of past climate, and sometimes they perform worse when forecasting extreme rainfall events (Vogel et al., 2018). These drawbacks led to the study of different approaches. Machine Learning (ML) is one promising substitute. ML learns directly from the large datasets rather than relying on physical assumptions. By finding the hidden patterns in rainfall records, ML can spot statistical relationships that traditional models overlook (Ravuri et al., 2021; Walz et al., 2024).

Therefore, this study examines whether the machine learning models trained on satellite rainfall observations can improve short term rainfall forecasting in tropical regions of Africa. The focus is on two practical problems: predicting rainfall for the same day based on current conditions, and predicting rainfall for the following day, which represents the challenge in short range forecasting.

For this purpose, the study uses the IMERG dataset (Integrated Multi-satellite Retrievals for GPM), which provides consistent, high-resolution rainfall estimates worldwide since 2001. IMERG has been widely used in tropical rainfall studies because it offers detailed and long-term observations (Hou et al., 2014; Huffman et al., 2020). From this dataset, features such as lagged rainfall, rolling averages, and seasonal signals are extracted, following evidence from earlier studies of rainfall cycles in Africa (Vogel, Knippertz and Fink, 2018; Rasheeda Satheesh, Rajeevan and Pai, 2023).

The project tests a range of models, from simpler methods like decision trees and k-nearest neighbours to more advanced Long Short-Term Memory (LSTM) deep learning model. Their performance is compared using accuracy, precision, recall, F1-score, and confusion matrices, with results benchmarked against both climatology and NWP forecasts. In this way, the study builds directly on recent work showing that ML has strong potential to improve short-term rainfall forecasting in the tropics (Vogel et al., 2018; Walz et al., 2024).

1.2 Objectives

1.2.1 Data Acquisition & Preprocessing

- Collect daily precipitation values for tropical Africa from the IMERG (2001–2024) dataset.
- Validate data integrity through missing-value checks, date continuity verification, and outlier inspection.

1.2.2 Exploratory Data Analysis (EDA)

- Visualise long-term daily precipitation patterns to identify trends and irregularities.
- Analyse seasonal cycles via monthly distributions, highlighting wet and dry periods.
- Apply cyclical transformations (sine and cosine of day-of-year) to capture periodicity.
- Detect and analyse extreme rainfall events.
- Create rolling averages to smooth daily irregularity and reveal seasonal signals.
- Conduct lag correlation analysis to identify temporal dependencies in precipitation.

1.2.3 Feature Engineering

- Construct lagged precipitation features (e.g., t-1, t-2 days) for short-term dependency capture.
- Integrate rolling mean features for multi-day rainfall accumulation trends.
- Encode cyclical seasonal indicators for ML interpretability.

1.2.4 Model Development

- Implement baseline classification models (Decision Trees, Logistic Regression, K-Nearest Neighbours, Random Forest) to establish benchmark performance.
- Develop deep learning models (e.g., Long Short-Term Memory networks, LSTMs) for sequential pattern learning.

1.2.5 Model Evaluation & Comparison

- Assess models using accuracy, precision, recall, F1-score, and confusion matrices.
- Compare ML performance against baseline approaches, including climatology and traditional NWP forecasts where available.

1.2.6 Critical Analysis & Recommendations

- Analyse the strengths and weaknesses of each modelling approach.
- Provide recommendations for operational integration and future research directions.

1.3 Deliverables

1. A reproducible Python-based precipitation modelling pipeline, including all preprocessing, exploratory data analysis (EDA), feature engineering, model training, and evaluation scripts.
2. A complete MSc dissertation report documenting background research, methodology, results, and critical evaluation, with a specific focus on precipitation forecasting in tropical Africa using IMERG data.

1.4 Ethical, Legal, and Social Issues

This project uses precipitation data from the IMERG (Integrated Multi-satellite Retrievals for GPM) dataset. Since this is satellite-derived data provided openly by NASA and its partners for scientific purposes, the project presents minimal ethical or legal concerns:

- The dataset does not contain any personal, sensitive, or confidential information.
- The data is freely available for research and academic use under NASA's open data policy.
- No human participants are involved in the study, so there is no requirement for formal ethical approval.
- The analysis and outputs are used solely for academic purposes, ensuring that there are no commercial, political, or privacy-related implications.

Chapter 2: Background Research

2.1 Literature review

Rainfall prediction in tropical regions has always been a challenge for weather scientists. Unlike in the mid-latitudes, rain usually comes from large weather systems like cold fronts or cyclones. But in the tropics, rain often comes from short, sudden storms that appear and disappear quickly. This makes them very difficult to predict (Mathon et al., 2002; Nesbitt et al., 2006). In West Africa, many of these storms group into larger systems called mesoscale convective systems (MCSs), which produce a large portion of the region's rainfall. (Fink and Reiner, 2003; Maranan et al., 2018). Because their behaviour is local and changes very quickly, standard weather prediction models often fail to capture them.

Most forecasts today are based on Numerical Weather Prediction (NWP) models, which use physics laws combined with direct observations of temperature, winds, and pressures to forecast how the atmosphere is going to change. These are very successful in mid-latitude regions of the world but less in tropical regions (Bauer et al., 2015). This is mainly because NWP models don't handle the small, fast-moving processes that cause tropical storms very well. The models look at large areas and use simplified formulas to represent storms. These shortcuts aren't detailed enough to reflect how storms in Africa behave (Marsham et al., 2011). There is research which shows NWP models in tropical Africa only occasionally perform better than just using long-term average rainfall, and sometimes they even do worse when predicting heavy rainfall (Vogel et al., 2018).

Some of the larger-scale weather patterns, such as African easterly waves, cannot provide much help because they influence where the storms form and track (Lavaysse et al., 2006; Schlüter et al., 2019). These waves can only explain why some rainfall events are linked over several days. Even with this information, NWP models still don't clearly show how these waves connect with local storms (Marsham et al., 2011).

Because of these limitations, researchers have started using statistical methods and machine learning (ML) to improve predictions. Statistical models such as logistic regression and conditional probability methods rely on identifying relationships between rainfall today and rainfall in the previous few days. These methods make use of the "memory" in tropical rainfall, where storms are influenced by patterns propagating from the east (Rasheeda Satheesh et al., 2023). Vogel et al. (2018) demonstrated that even simple logistic regression models, trained on satellite rainfall data, were able to outperform NWP forecasts of rainfall occurrence.

The recent growth of machine learning has expanded these approaches significantly. Deep learning models are especially promising because they can find complicated patterns in data that change over time. For example, Ravuri et al. (2021) used generative neural networks to improve short-term rain forecasts using radar data. Other researchers (Espeholt et al., 2022; Lam et al., 2023) showed that these models can sometimes match or even beat traditional weather prediction systems. In the tropics, Walz et al. (2024) found that hybrid models combining machine learning and physics-based approaches achieved better performance than either method alone.

These improvements are possible because we now have big, reliable datasets that track rainfall over time. The Integrated Multi-Satellite Retrievals for GPM (IMERG) dataset (Hou et al., 2014; Huffman et al., 2020) has become a critical resource for rainfall prediction research. IMERG

provides consistent, gauge-calibrated rainfall estimates over multiple decades, which allows not only long-timescale seasonal cycles but short-timescale signals such as lag correlations to be resolved by the model, which is critical for prediction in tropical areas.

Taken together, the literature highlights several key insights. First, rainfall in tropical regions is tough to predict because it comes from sudden, local storms and it is not strongly connected to larger weather systems. Second, statistical and machine learning methods offer promising alternatives where NWP continues to struggle in this specific region. Finally, the availability of long-term observational datasets such as IMERG opens possibilities for observation-to-observation learning, where models are trained directly on past rainfall to predict future rainfall (Hou et al., 2014; Huffman et al., 2020). This project focuses solely on one variable, “precipitation” for prediction. This approach allows the study to test the pure potential of machine learning in identifying rainfall patterns without relying on additional atmospheric inputs.

2.2 Methods and techniques

Different approaches have been used to tackle the challenge of forecasting rainfall in the tropics. These methods can be grouped into three main categories: physics-based models, statistical approaches, and machine learning models. Each has strengths and weaknesses and understanding them helps explain the choices made in this project.

2.2.1 Physics-Based Models (NWP)

Numerical Weather Prediction (NWP) models are the traditional backbone of weather forecasting. These models use current weather data like wind, humidity and temperature and apply the equations of physics to figure out how the weather might change. These models are reliable in many parts of the world, but predicting tropical weather is still a challenge. The storms that bring most of the rainfall in Africa are too small and develop too quickly, making them hard for NWP models to capture (Bauer et al., 2015; Marsham et al., 2011). Even with improved parameterisation, NWP predictions in Africa still perform only slightly better than climatology and often fail to capture extreme events (Vogel et al., 2018).

2.2.2 Statistical Approaches

Statistical models look for patterns between today’s rain and what happened in previous days. These methods make use of the fact that rainfall sometimes shows “memory”: for example, rain on one day can increase the chance of rain on the next day if the weather continues to move through the region (Rasheeda Satheesh et al., 2023). Basic statistical methods like logistic regression have done better than global NWP models at predicting rain in West Africa (Vogel et al., 2018). However, while these models are easier to understand, they might miss complicated patterns in how rain behaves.

2.2.3 Machine Learning and Deep Learning Models

Machine learning learns patterns from previous data instead of using scientific equations like traditional models do. That’s why Machine Learning is helpful in tropical regions where regular models struggle to show how local storms behave. For example, ML can detect patterns in sudden rainstorms that form quickly and disappear. Machine learning can find complex patterns in how rain behaves (Walz et al., 2024). This project applies a set of baseline machine learning models

to provide benchmarks for performance and interpretability. Alongside these, more advanced deep learning methods are also tested

2.2.3.1 Decision Trees

A decision tree works by splitting up the data into branches according to simple rules. A first rule, for example, might test whether yesterday's rainfall was greater than 10 mm. If it were, it would predict rain; otherwise, it might take into account the season (wet or dry) before making a final prediction. That makes decision trees easy to interpret and helpful in identifying which features are most important, such as lagged rainfall or seasonal cycles. One tree does not always generalise sufficiently in its own right, though, because a tree is capable of overfitting information in a set: several trees are thus combined in an ensemble like a Random Forest for greater ability (Breiman et al., 1984; Vogel et al., 2018).

2.2.3.2 Logistic Regression

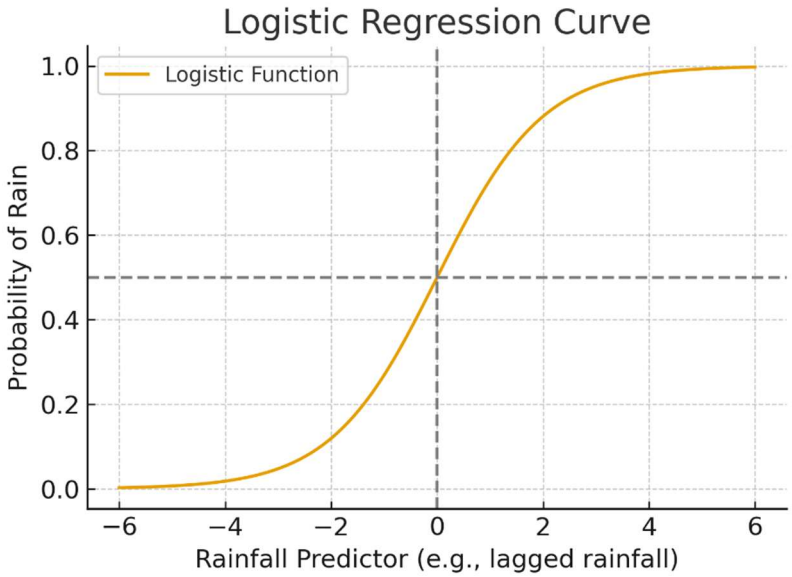

Figure 2.2.3.2.1: logistic regression

Logistic regression is a statistical model widely used for classification. It does so by fitting an S-shaped logistic curve to the information and returns a probability between 0 and 1. If a probability is greater than a threshold (for example, 0.5), it returns the positive class. Logistic regression is simple, rapid, and interpretable, making it heavily used in machine learning (Hosmer et al., 2013). In tropical Africa, logistic regression has been shown to predict rainfall occurrences more accurately than global NWP models, showing its effectiveness despite its simplicity (Vogel et al., 2018).

2.2.3.3 K-Nearest Neighbours (KNN)

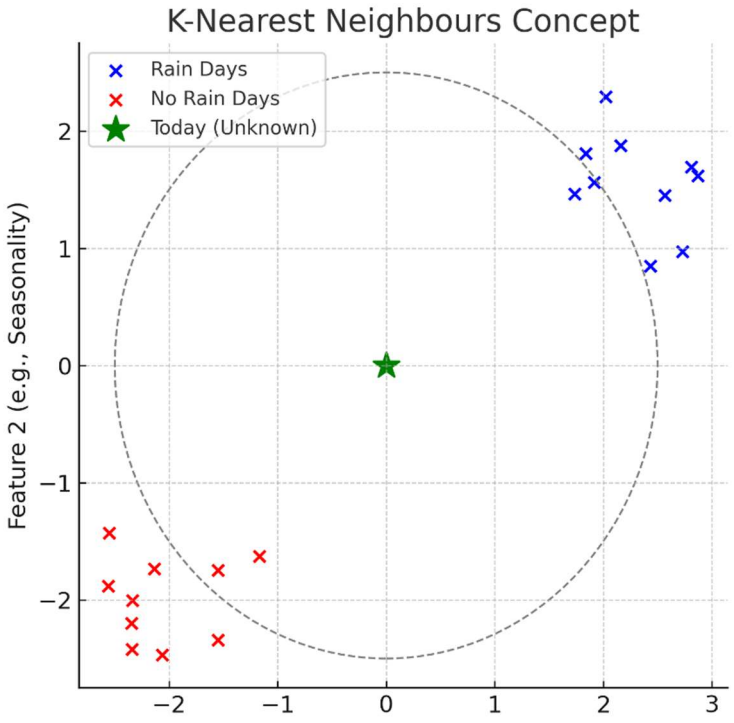

Figure 2.2.3.3.1: K-Nearest Neighbours

The KNN is a non-parametric prediction model which generates predictions for new points based on matching with k nearest points in the training dataset. Similarity is usually measured by distance measures such as Euclidean distance. Then it makes predictions for the most frequent class among neighbours. KNN is easy to implement but not flexible and computationally expensive when a large training set is involved (Cover and Hart, 1967). To make rainfall prediction, a KNN is useful in revealing patterns based on linking today's state with similar past days, such that it is a robust method when associations are not linear (Rasheeda Satheesh et al., 2023).

2.2.3.4 Long Short-Term Memory (LSTM) Network

LSTMs are a type of neural network made to work with data that comes in sequence, like time series. LSTMs use memory units and gates to decide what to remember, what to update, and what to forget. This helps LSTMs understand long-term patterns in data that changes over time (Hochreiter and Schmidhuber, 1997). LSTMs are very popular in areas like speech recognition and weather predictions because they handle time-based data well. These models can work well for predicting rain because they can learn patterns from past rainfall and make better short-term forecasts than simple models (Ravuri et al., 2021; Walz et al., 2024). For example, If it rains today, it might also rain two days later. LSTMs can learn these delayed patterns.

2.2.3.5 Random Forest Classification

Random Forest is one of the ensemble methods that generalises the decision tree method by pooling the output of many trees to compute a final prediction (Breiman, 2001). Each tree is learned from a bootstrapped copy of the data, and at each split, only a random subset of features is considered. For a classification task, the output is the majority vote of the trees. This

randomisation results in the trees being highly diverse, and it reduces the overfitting using a single decision tree.

The strength of Random Forest is its power and versatility. It can handle non-linear relationships between features, both continuous and categorical variables and provide feature importance estimates (Liaw and Wiener, 2002). It is yet less interpretable than a single tree and can tend to be more computationally expensive. Nonetheless, Random Forest is highly favoured when it comes to classification problems because of its accuracy and ability to generalise well when new data are encountered.

2.2.4 Feature Engineering

Feature engineering means creating new data inputs that help the model learn better, based on the original data. To improve the model's performance, extra features are created from the raw rainfall data. As there is only one variable available, rainfall data will turn into different types of signals to make it more useful for predictions. These extra features help both basic and advanced models learn how rainfall changes over time and across seasons.

2.2.5 Rationale for Model Selection

The selection of these models was made to strike a balance between simplicity, ease of understanding, and the ability to handle complex patterns. By using both simple and complex models, we can see how much machine learning helps in predicting rain.

a) Decision Tree and Logistic Regression

These were selected since they are simple, interpretable, and standard baselines for classification tasks. Decision trees also have inherent rules describing which variables (e.g., lagged rainfall or seasonal cycles) are significant (Breiman et al., 1984). Logistic regression is also unexpectedly successful at predicting tropical rainfall at times, even better than results for global NWP models (Vogel et al., 2018). We include these models to give a set of controls for comparison with more advanced approaches.

b) K-Nearest Neighbours (KNN)

KNN was included because it does not make strong assumptions about the nature of the data. Instead, it makes classifications based on comparing today's conditions with those most similar in the past. Because it is so flexible, it is employed in model testing for determining if similarity approaches are able to identify patterns that logistic regression or other linear models would not identify (Cover and Hart, 1967; Rasheeda Satheesh et al., 2023).

c) Long Short-Term Memory (LSTM) Networks

LSTMs were chosen as the best model because LSTMs are explicitly for sequence data. Using memory cells and gates in their design, LSTMs are capable of learning from past sequences of rainfall in addition to picking up on short- and long-term dependencies (Hochreiter and Schmidhuber, 1997). LSTMs were found to consistently accomplish tasks in weather and rainfall prediction like outperforming standard means when plenty of available data is in use (Ravuri et al., 2021; Walz et al., 2024).

The goal is to test both simple and complex models to see how much machine learning improves short-term rainfall forecasts in tropical regions.

d) Random Forest Classifier

Random Forest was included as a stronger non-linear baseline compared to a single decision tree. Through combining numerous trees, the approach enhances stability and accuracy and minimizes the possibility of overfitting (Breiman, 2001). Also, it has feature importance measures available, which can reveal which rainfall predictors play the greatest role in determining classification results. Since the rainfalls of the tropics are highly complex and erratic, the use of Random Forest makes the baseline models robust enough without resorting to heavy data preprocessing.

2.2.6 Environment setup

All analysis and modelling in this study were conducted using Python (version 3.9) within a Jupyter Notebook (.ipynb) environment. Jupyter was chosen because it allows code, results, and explanations to be combined in one document, improving both clarity and reproducibility (Kluyver et al., 2016). One of the key advantages is that it lets users execute code cell by cell, making it easier to test, debug, and understand each step of the analysis. This is useful in data science and machine learning, where iterative experimentation and visualisation are essential.

The required Python libraries for the project are briefly detailed as follows:

- Pandas: A data analysis toolkit constructed for handling structured, table-like data. It is very popular for data cleaning, data manipulation, and transforming operations (McKinney, 2010).
- NumPy: The fundamental Python package for numerical computation. It provides fast linear algebra and efficient array and matrix management (Harris et al., 2020).
- Matplotlib: It is a very flexible plot library that enables users to plot detailed and highly customizable data visualisations (Hunter, 2007).
- Seaborn: A statistical data visualisation library based on Matplotlib. It makes it easier to construct insightful charts and heatmaps (Waskom, 2021).
- Scikit-learn: A multidisciplinary machine learning library that offers routines for classification, regression, cluster analysis, and pre-processing and evaluation (Buitinck et al., 2013).
- PyTorch: A platform for deep learning that can be utilised for learning and building neural networks and that also offers support for GPU for quicker computation (Paszke et al., 2019).

They, together, form the heart of Python's data science stack and provide a stable foundation for tasks ranging from preprocessing and visualization to machine learning and deep learning.

Chapter 3: Research Design and Data

3.1 Research Design

The research design for the project follows a step-by-step structure to ensure the study is systematic, reproducible, and aligned with the objectives. The process includes exploratory data analysis, testing different models, and comparing their performance.

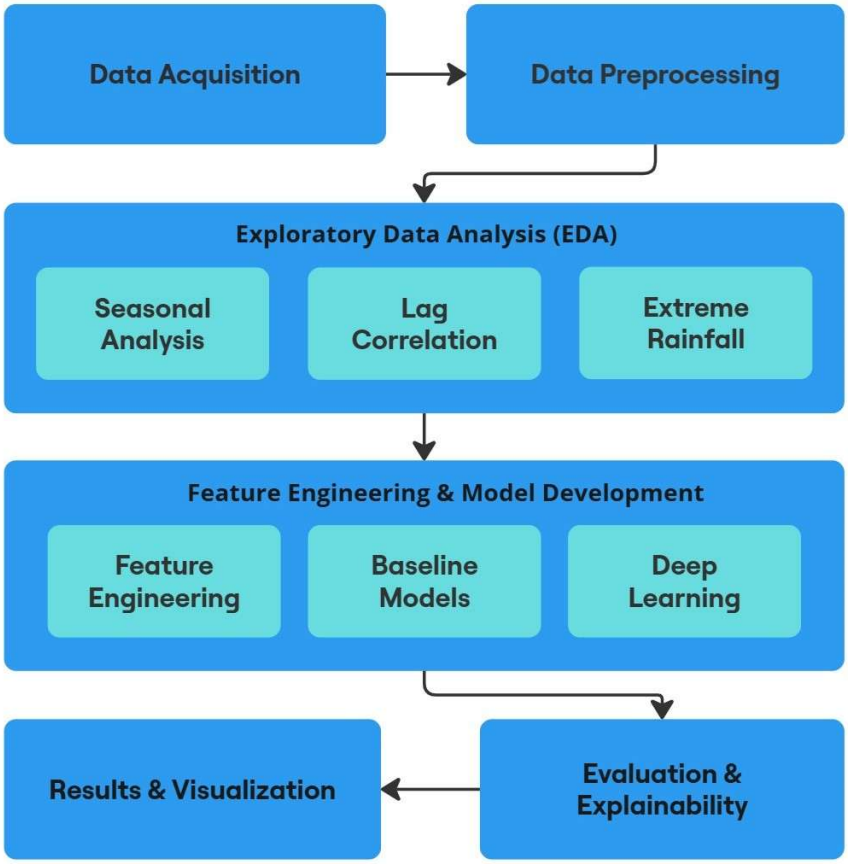

Figure 3.1.1: Research Design Flow

3.2 Data Acquisition and Description

The dataset used in the study comes from the multi-satellite retrievals for GPM (IMERG) product, which provides global estimates of precipitation. IMERG is widely used in meteorological and hydrological research due to its consistent, long-term, and high-resolution rainfall data (Hou et al., 2014; Huffman et al., 2020). This project focuses on daily precipitation measurements across tropical Africa from 1998 to 2024.

The IMERG dataset is in a HDF5 (Hierarchical Data Format) format, which is a commonly used format in climate and earth sciences since their datasets are so large and multidimensional, including their metadata in a single container (Folk et al., 2011). Even though it is so powerful, HDF5 files are typically very large in size and only readable using special tools or libraries. The

entire dataset was around 50–70 GB in size, which raised challenges not only for computation and storage but even for sharing and transferring the data. It is infeasible and difficult for machine learning workflows on general hardware to handle such large files.

CSV (Comma-Separated Values) files are less complex and much lighter in comparison. Data is stored in a table-like structure (columns and rows), easy to open in everyday software such as Excel or directly imported into programming environments such as R and Python (McKinney, 2010). CSVs are more open, shareable, and directly interoperable with conventional machine learning libraries such as Pandas and Scikit-learn.

For convenient manipulation of the dataset, a 1° × 1° gridded form in CSV format was provided by the project supervisor. It was a necessary conversion because:

- It reduced file size and computational expense so that training on regular hardware was feasible.
- It converted the dataset into a structured table format, which is usable in machine learning flows directly.
- It was consistent over the whole study period (1998–2024).

These were delivered as separate CSV files per year, which were simple to transfer, archive, as well as import into the analysis environment.

The use of CSV format also made direct compatibility with the most commonly used Python libraries, such as Pandas and Scikit-learn, which are essential for data analysis and machine learning. These tools were applied during the preprocessing and modelling stages of the project, and their role is explained in more detail in the following section on Data Preprocessing.

Although the IMERG dataset spans the years 1998–2024, only the recent past five years (2020–2024) were used in training and testing a model. This was a matter of computational feasibility: though converted to annual CSV files, the full dataset would mean exceedingly large numbers of records, making it difficult to train machine learning algorithms efficiently on standard computer hardware. Limiting the study to a five-year subset reduced this computational intensity and made it feasible to conduct experiments within the hardware limitations available.

At the same time, this chosen period still has sufficient variability between wet and dry seasons to provide informative training data. Furthermore, focusing on current years ensures that contemporary rainfall dynamics will be captured by models, which is information relevant to current rather than historical forecasting challenges (Vogel et al., 2018). This trade-off between tractability and representativeness achieved a balance between feasibility and scientific integrity.

3.3 Data Preprocessing

Before applying the machine learning models, the data was also pre-processed for uniformity and ease of use. The raw IMERG data had been supplied as separate CSV files for each year between 1998 and 2024. The individual files were integrated into a single Pandas DataFrame to enable consistent temporal analysis throughout the study period (McKinney, 2010). Combining the files in this manner reduced the complexity of future data operations and maintained a consistent temporal structure

```python
# Read and merge
all_data = []
for file in csv_files:
    file_path = os.path.join(data_folder, file)
    df = pd.read_csv(file_path)
    all_data.append(df)

# Combine all years into one DataFrame
df = pd.concat(all_data, ignore_index=True)
```

Figure 3.3.1: Merging of all the CSV files

The “time” column was converted from string format into a proper datetime object using Pandas, allowing for chronological ordering and time-based operations. The index was reset for a clean structure, and the dataframe was sorted by date.

```python
# Convert 'time' column to datetime
df['time'] = pd.to_datetime(df['time'])
# Sort by time
df = df.sort_values(by='time')
df.reset_index(drop=True, inplace=True)
df.info()
```

Figure 3.3.2: string to datetime conversion of column

To verify integrity, the first rows were printed, and descriptive summaries were validated (df.info() and df.head()) as shown in Figure 3.3.3 and Figure 3.3.4. This served to confirm that the dataset included the complete 1998–2024 period without any missing or duplicate time entries.

	latitude	longitude	time	precipitation
0	-9.5	-67.5	1998-01-01	1.309754
1	-9.5	-67.5	1998-01-02	7.817925
2	-9.5	-67.5	1998-01-03	1.078305
3	-9.5	-67.5	1998-01-04	17.203157
4	-9.5	-67.5	1998-01-05	7.456180

Figure 3.3.3: Dataset information

```
Data columns (total 4 columns):
 #   Column         Dtype
---  ------         -----
 0   latitude       float64
 1   longitude      float64
 2   time           datetime64[ns]
 3   precipitation  float64
dtypes: datetime64[ns](1), float64(3)
```

Figure 3.3.4: Datatypes of each column

Since the patterns for tropical African precipitation are extremely seasonal, a new column for the month was included from the time variable. This allows for the data to be aggregated and examined at a monthly level, enabling the identification of wet and dry seasons and also for future feature engineering.

This makes it easier to:

- Compare rainfall for different months of the year.
- Identify significant seasonal patterns (e.g., the July–September monsoon).
- Permit feature engineering, with month as a seasonal proxy.

These preprocessing steps provided a stable data set for future use. Higher-level steps such as feature engineering, handling class imbalance, and Exploratory Data Analysis (EDA) are explained in succeeding sections.

3.4 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a very important step involved in any data science project. It is comprised of exploration and data visualisation of the dataset for the purpose of understanding its salient features before calling for more advanced models. It has the possibility of uncovering data quality issues, transferring general statistical properties, and uncovering relationships that inform subsequent stages like feature building and model construction (Tukey, 1977; Shmueli and Lichtendahl, 2018). In this particular case of forecasting rainfall, it is particularly important because precipitation exhibits significant variations both in space and time, and precipitation phenomena occur both routinely over the seasons and intermittently based on weather conditions.

3.4.1 Data Integrity Check

The first step of carrying out EDA was checking for data integrity, i.e., correctness, completeness, and uniformity of the data set. In time series analysis, minute problems like missing days or null values will upset calculations and mislead prediction models (Little and Rubin, 2019).

```
# Checking for missing values in any column
print("Missing Values Summary:")
print(df.isnull().sum())

# Ensuring daily frequency is consistent
date_range = pd.date_range(start=df['time'].min(), end=df['time'].max(), freq='D')
actual_dates = df['time'].sort_values()

# Compare expected vs actual
missing_dates = set(date_range) - set(actual_dates)
print(f"\nMissing Dates in Time Series: {len(missing_dates)}")

# show a few missing dates if there are any
if missing_dates:
    print("Example missing dates:", sorted(list(missing_dates))[:10])
```

Figure 3.4.1.1: Checking missing and null Values

Two checks were conducted:

- Null values: Null values for each column were checked for. No nulls were discovered.
- Missing dates: A date variable was converted into a proper datetime variable and tested against a complete date range for every day from 1998 through 2024. No missing dates were found, affirming that the sequence was continuous.

```
Missing Values Summary:
latitude         0
longitude        0
time             0
precipitation    0
month            0
dtype: int64

Missing Dates in Time Series: 0
```

Figure 3.4.1.2: Summary of Missing values

According to the results, the dataset was complete and consistent, making it suitable for additional analysis.

3.4.2 Descriptive Statistics and Distributions

Descriptive statistics and distribution analysis are a large part of exploratory data analysis because they provide a summary of the rainfall dataset and extracted patterns which are not easily seen from raw data. Instead of looking at individual one-day values, descriptive analysis considers the behaviour of the rainfall at a monthly, seasonal, and long-term timescale. This makes it simple to see how rainfall is distributed throughout a year, what months are the biggest contributors towards yearly totals, and how rainfall patterns vary across the study period (Wilks, 2011).

Precipitation data is often highly irregular. Many days may experience no rain, while a few months or seasons experience localised heavy rainfall. This means that precipitation is not normal and instead has a positively skewed shape. In a positively skewed shape, there isn't a balanced spreading of the values around the average. In the case of rain, this means that a large number of days experience a low amount of or no rain, while a smaller number of days experience very heavy rain. This produces a lopsided shape: dry days dominate the dataset, while heavy rainy days reach into a long “tail” at the extreme right (Wilks, 2011; Vogel et al., 2018).

A positively skewed shape is not a balanced spreading of the values around the average. In the rain scenario, that means a lot of days experience a low amount or no rain, and not many have extremely heavy rain. This creates a lopsided shape: more days are dry in the data set, while heavy Rain days go into a long “tail” at the far-right (Wilks, 2011; Vogel et al., 2018).

3.4.2.1 Daily Precipitation Time Series (1998–2024)

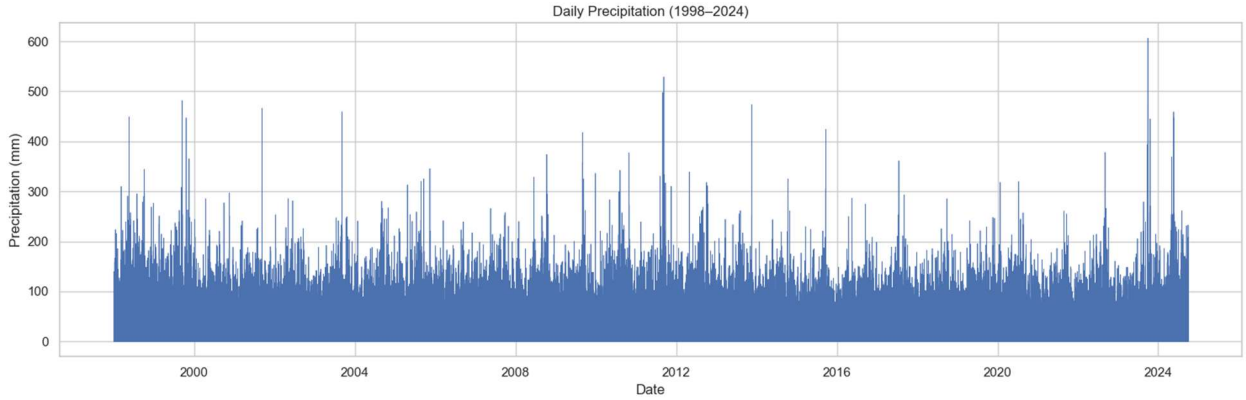

Figure 3.4.2.1.1: Daily precipitation over tropical Africa (1998–2024)

The initial step in descriptive analysis was to analyse the long-term yearly precipitation dataset for the period from 1998 to 2024 (Figure 3.4.2.1.1). This plot offers a general idea of the behaviour of rainfall over the 27 years. The shape and distribution of values look stable over time, and this gives the indication that there is no significant data drift or structural breaks in the series. This follows the previous integrity checks and verifies that the data are complete and can be trusted for further analysis.

The chart also shows the huge day-to-day variability of rain per day. Even though most days feature light or intermediate rain, there are some heavy events above 500–600 mm per day and that almost certainly are storms or heavy monsoon action. These exceptional events also must be considered in forecasting models because of their enormous potential for environmental and social harm (Vogel et al., 2018).

3.4.2.2 Monthly Precipitation Distribution (1998- 2024)

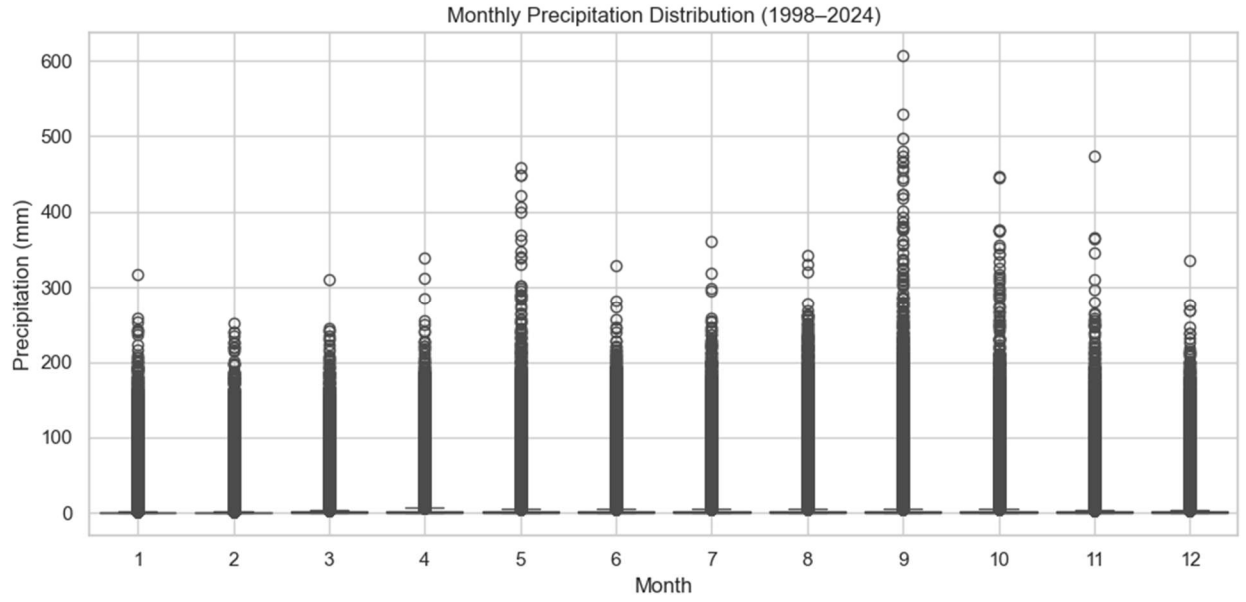

Figure 3.4.2.2.1: Monthly precipitation distribution (1998–2024)

Figure 3.4.2.2.1 shows the monthly precipitation seasonal distribution for the study period. Tropical Africa's seasonal rainfall patterns are shown in the plot. September receives the highest median rainfall and has a maximum high outlier, and thus corresponds to the peak of the rainy season. March-May also corresponds to relatively higher values of rainfall and thus potentially an early or a secondary wet season. January-February and July-August, however, are quite drier and have very low median values of rainfall.

This seasonal pattern agrees well with previous findings that highlight that July-September is the principal monsoon season for West Africa, and early March-May maxima are a result of pre-monsoon activity (Maranan et al., 2018; Rasheeda Satheesh et al., 2023).

3.4.2.3 Average Daily Precipitation by Day of Year (1998-2024)

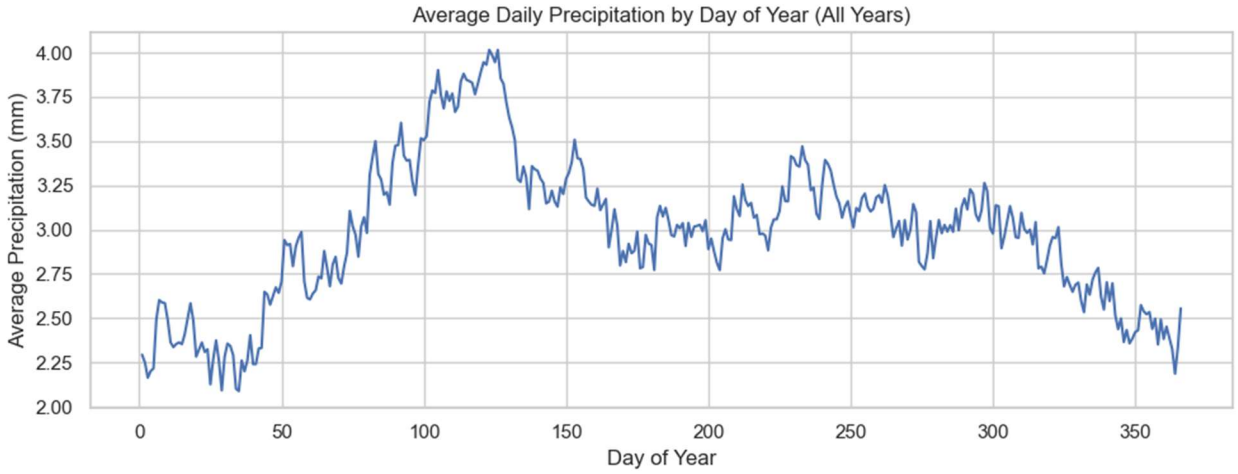

Figure 3.4.2.3.1: Average Daily Precipitation by Day of Year

Figure 3.4.2.3.1 shows the average daily precipitation quantities plotted by day of the year, averaged over the entire period of study. The figure exhibits a very pronounced cyclical seasonal pattern, with distinct rises, peaks, and falls in intensity of rain that repeat every year.

Key observations include:

- A sharp increase in rainfall from approximately day 50 until day 100, marking the onset of the main rainy season (likely February-April).
- Peak intensity of the rain during days 110-130, that is, at the middle of the wet season.
- There is a steady decline thereafter from day 150 onward, characteristic of a transition toward the dry phase.
- Mild fluctuations between day 200 and day 300, which may represent occasional rainfall bursts, indicating a transition phase between the wet and dry seasons.
- Minimum precipitation levels at the end of the year (around day 360), repeating the seasonal cycle, end in more regularly drier conditions.

These also confirm that tropical African rainfall is extremely seasonal and cyclical, as has been discovered from climatic analyses of the region (Maranan et al., 2018; Rasheeda Satheesh et al., 2023). The strong annual cycle also justifies the use of sine and cosine transformation of the day of year as important and extra features in machine learning modelling.

3.4.2.4 Mean Daily Precipitation during July-September (1998-2924)

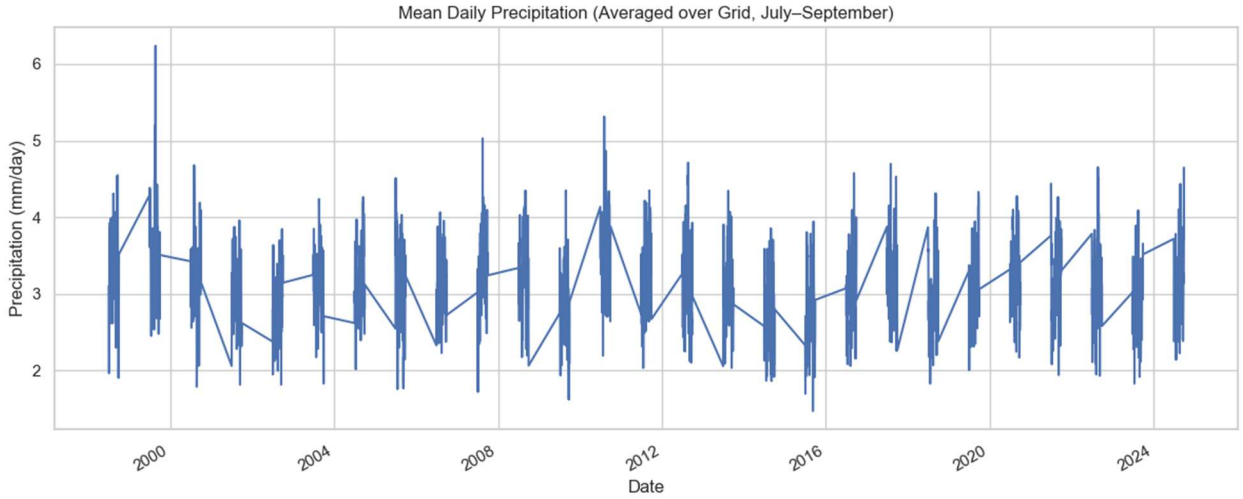

Figure 3.4.2.4.1: Mean Daily Precipitation during July-September (1998-2024)

Figure 3.4.2.4.1 shows the average precipitation per day, over grid boxes, for the July-September (JAS) season from the 27-year dataset. This season is climatologically defined as the prime rainy season for West Africa (Maranan et al., 2018; Rasheeda Satheesh et al., 2023), and it is therefore a season of particular interest for establishing rainfall predictability.

The chart highlights several key insights:

- Every year has sharp fluctuations in the JAS season, characteristic of the dynamic and convective nature of tropical rainfalls.
- Mean precipitation is generally within the range of 2 to 4 mm/day, although there are spikes above 5–6 mm/day in individual years that imply the occurrence of very heavy events.
- Interannual variability is evident, and there are some seasons (e.g., around 1999, 2012, 2021) that suggest more significant peaks in contrast to other years' drier JAS seasons.
- Despite those fluctuations, the seasonal pattern is steady in general, reinforcing the overriding control by JAS in determining the region's rainy cycle.

All things considered, the descriptive analysis confirms several aspects of tropical African rainfall behaviour. The monthly distribution highlighted September's dominance as the peak of the rainy season, and additional rainfall contributions during March-May that are indicators of a secondary wet phase. The day-of-year analysis confirmed the cyclical and repetitive nature of the rainfall, and sudden rises and declines that characterise the transitions of wet and dry phases. Finally, highlighting the July-September period, it showed that although this season is extremely variable within a single year and uneven across years, it remains the most reliable indication of the regional cycle of rain.

By combining these insights, the exploratory analysis validated that the analysis region's rainfall was complete, highly seasonal, and highly cyclical, and featured prominent wet and dry phases. These results are crucial for feature engineering, and this ensures that the machine learning models are trained with an accurate representation of the dynamics in the rainfall.

3.5 Feature Engineering

Feature engineering involves creating new variables, or “features”, from raw data in a manner that improves the effectiveness of machine learning models. While raw rainfall quantities can provide us with useful information, by themselves, raw values generally don't draw out the full patterns and interdependencies that exist within time series of precipitation. By creating additional features that convey seasonality, recent history of precipitation, or smooth trends, models can learn more informatively from the data (Wilks, 2011; Rasheeda Satheesh et al., 2023).

In precipitation forecasting, feature engineering is particularly important because precipitation is very variable, highly seasonal, and controlled by short-term persistence forces. To offer a very basic example, today's rain would increase the chance that it would be raining tomorrow more than a perfectly sunny day (Vogel et al., 2021). To express those relationships, one must extract from raw data features that encode not just short-term memory but long-term cycles.

3.5.1 Lag Features

Lag features were introduced for the models to learn whether today's rain depended on what happened in the previous few days. In tropical regions, it is not always a matter of raining or not raining at all: it will often rain for some days in a row or not at all. This is because storms and monsoon systems can last for some days or come in increments and not as isolated events (Vogel et al., 2018).

	time	precipitation	precip_lag1	precip_lag2
0	1998-01-01	1.309754	NaN	NaN
1	1998-01-01	0.030013	1.309754	NaN
2	1998-01-01	2.999300	0.030013	1.309754
3	1998-01-01	0.300424	2.999300	0.030013
4	1998-01-01	1.131319	0.300424	2.999300

Figure 3.5.1.1: Lag Features

For this short-term “memory” to be retained, new columns were also created for keeping rainfall for the previous 1-4 days (Figure 3.5.1.1). Last day’s precipitation (e.g., yesterday’s precipitation) is saved in precip_lag1, precipitation for two days ago is saved in precip_lag2, and so on. In the table’s early rows, NaNs appear in the lag columns, and that’s just because there isn’t any previous data for those days. The very first date in 1998, for instance, cannot have a “yesterday” because the data starts from that day. Those NaNs are a natural side effect of the construction of lag features and don't affect the main analysis.

The machine learning models also gain the advantage of history other than yesterday's rain through the inclusion of these lagged columns. They can now capture short-term trends, e.g., “if it rained yesterday and two days ago, there is a higher chance it might rain today.” This transformation is important for rainfall prediction, where clustering of wet days plays an important role in shaping rainfall behaviour.

a) Lag Correlation of Precipitation

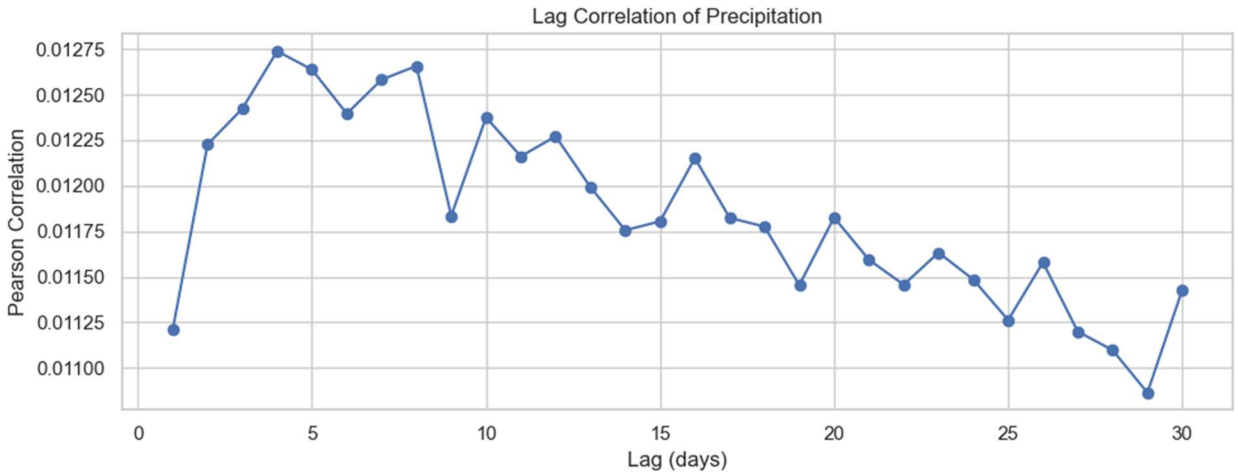

Figure 3.5.1.a: Lag Correlation of precipitation

To determine whether these lag values actually make a difference, a lag correlation analysis was performed for the whole dataset (Figure 3.5.1.a). It simply asks how today's rain correlated with rain of few days ago. The output showed that, averaged out, the correlation is quite weak. Today's rain isn't significantly controlled by conditions weeks in the past. But there were slight bumps at around 4–5 days and 7–8 days. These humps suggest that rain isn't truly stochastic: short weather cycles or groupings of storms may sometimes last for a few days before vanishing.

While the relationships were weak, this discussion illustrates that there is at least short-term persistence for rainfalls. This makes it sensible to implement lag features because a weak signal can support the models in detecting repeated patterns.

b) Average Lag Correlation of precipitation (July-September)

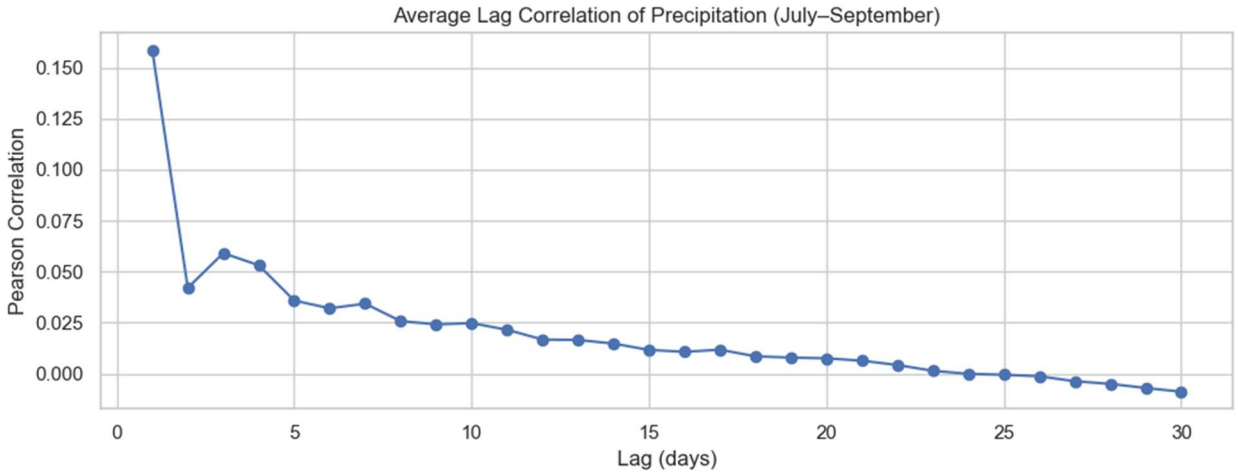

Figure 3.5.1.b: Average lag correlation of precipitation (JAS)

If we consider only the July–September (JAS) monsoon season, the structure of the lag correlation appears quite clear. The correlation is more pronounced for the recent days,

particularly at lag 1 (~0.15), and smoothly declines as the distance between days widens. That is, a day-previous rain is far more useful than a week-previous rain in predicting today's weather conditions.

This declining trend resembles that of tropical rain systems that are generally short-lived but intense. These monsoon outbursts and convective storms leave a lasting signal for several days thereafter, but at longer lags, the atmosphere recycles, and the signal of earlier rainfall is lost. In model building, this means that we should only keep the previous days of rain as lag features, rather than very long lags. This way, the models can capture the strong short-term persistence of JAS rainfalls while avoiding unnecessary noise from days a long distance away that contribute relatively little prediction power (Rasheeda Satheesh et al., 2023).

Although initially the early implementation focused only on lag 1 and lag 2, later higher lag parameters (lag 3 and lag 4) would also figure in the project's development. This extension would allow the models to consider slightly longer short-term relationships, testing whether it was feasible for rain from a maximum of four previous days to provide some additional forecasting.

3.5.2 Rolling Averages

In addition to lag features, rolling averages also created to smooth out day-to-day fluctuations in rain and highlight broader trends. Rainfall data is highly irregular: some days would have no rain at all, and some would have downpours. This "noisy" nature makes it hard for models to extract useful patterns. Rolling averages reduce the noise by calculating the average rainfall across a fixed window of days and moving this window forward in stages (Wilks, 2011).

For this, rolling averages of 3 days, 7 days, and 30 days were calculated (Figure 17).

Each has a unique timescale for rainfall variability:

- **3-day rolling average** shows a short-term behaviour which is effective for the selection of short wet spells.
- **7-day rolling average** gives weekly rainfall behaviour, making it easier to check whether a week is wet or dry.
- **30-day rolling average** establishes long-term trend, such as build-up to the monsoon or the gradual decline into the dry season.

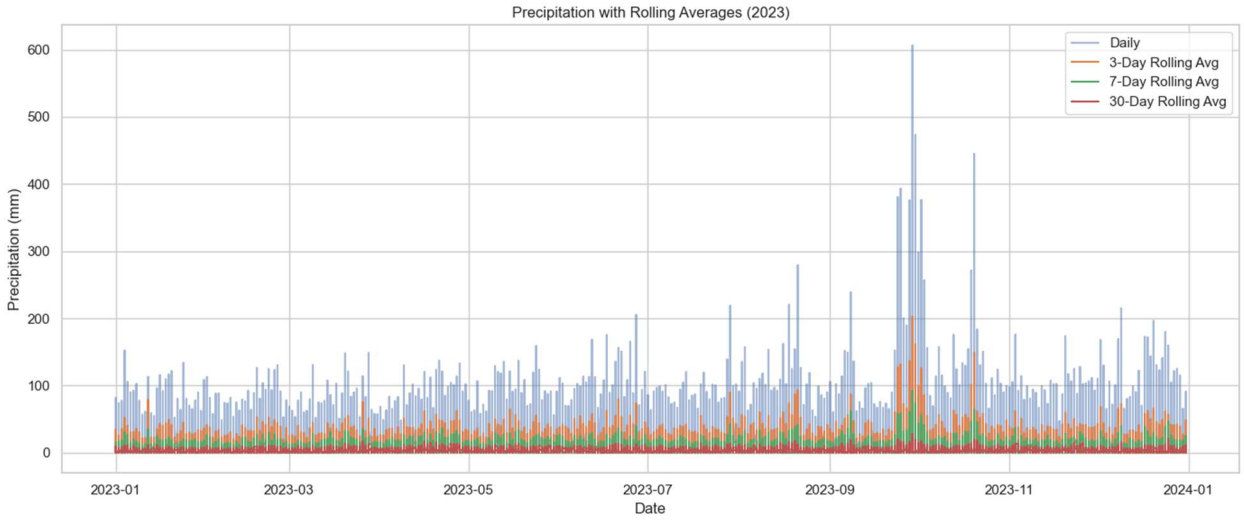

Figure 3.5.2.1: Precipitation with rolling averages for 2023 year

From the above figure 3.5.2.1 we can see that the raw daily precipitation (blue bars) is highly variable, with numerous spikes in September and October of 2023. In contrast, the rolling averages provide smoother curves: the 3-day average (orange) follows the daily values closely, catching clusters of storms; the 7-day average (green) smooths out the noise a bit further, bringing out long wet(rain) weeks; and the 30-day average (red) captures the seasonal cycle, in that the intensity of the precipitation increases in mid-year and falls at year-end.

This multi-layered approach makes sure that the model learns from both short-term behaviour (3- and 7-day averages) and long-term behaviour (30-day average). The use of rolling averages strengthens the data by balancing local weather indications against the seasonal trends (Hyndman and Athanasopoulos, 2018).

3.5.3 Seasonal Indicators

To account for seasonality in precipitation, the dataset was analysed by grouping the rainfall across the ay of year dimensions. As shown in Figure 12, we can see that rainfall in tropical Africa follows a distinct cyclic pattern, with steep increments and declines that align with wet and dry seasons. This confirmed that there is a strong influence of seasonality in rainfall, making it important to represent it within the features set for machine learning models.

However, the use of day of the year or of the month as a numerical variable is misleading, because there is an arbitrary break in the calendar: December (day of the year 365) follows January (day of the year 1). If passed directly into the model, those two days would be falsely regarded by it as very distant from one another, although consecutive in the cycle of seasons. To overcome this, the day-of-year values were transformed by using sine and cosine functions, which roll the year into a smooth circle. In this way, seasonal cycles are continuously represented, without breaks at the start or at the end of the year (Hyndman and Athanasopoulos, 2018).

Two transformations applied were:

- sin_doy = sin (2π * dayofyear / 365.25)
- cos_doy = cos (2π * dayofyear / 365.25)

The denominator was also changed from 365 to 365.25 for a more accurate record of leap years. A normal year is 365 days long, and every four years has 366 days. In long sequences, such as the 27-year dataset for this project (1998–2024), a steady usage of the 365 days would potentially impose a slowly increasing misalignment in the seasonal encoding. Using 365.25 keeps the encoding in sync with the actual length of a year and therefore always encodes the cycle of the rainfalls throughout the dataset.

Together, these transformations encode each day's location in the seasonal cycle. Days in the middle of we season (e.g., mid-August through September) cluster next to each other in the space of sines and cosines, and other days that are early in the dry season cluster at a different location in the cycle. In this way, this encoding allows the models to learn from seasonality continually, without being misled by the artificial calendar cut-off.

By adding sine and cosine transformations of the day of year, the data was enriched by features that encode seasonal change explicitly, in addition to the raw precipitation and lag features. This ensures that the models can not only account for short-term persistence, but also for the recurring yearly cycle of rain that is characteristic of tropical climates.

3.5.4 Categorical Binning

Aside from persistent rain features, a categorical representation of precipitation was also proposed for ease of prediction. Precipitation during each day was categorised into a total of three classes according to intensity:

- No – 0mm
- Light - ≤10 mm
- Strong - > 10 mm

	time	precip_bin	precip_lag1_bin	precip_lag2_bin
0	1998-01-01	Light	Strong	Strong
1	1998-01-01	Light	Light	Strong
2	1998-01-01	Light	Light	Light
3	1998-01-01	Light	Light	Light
4	1998-01-01	Light	Light	Light
5	1998-01-01	Light	Light	Light

Figure 3.5.4.1: Categorical binning

This binning was also performed for the lag features (precip_lag1 and precip_lag2). The resulting dataset thus contained not just the continuous level of rain (in millimetres), but also categorical bins that represented whether the previous few days had been dry, somewhat rainy, or actively rainy.

There are two reasons for categorial bins. First, a lot of classification models are more precise when forced to predict classes rather than hard numerical values. In the case of rainfall, this framing makes sense since we are often more concerned about whether a certain day will be dry, rainy, or have heavy rain, rather than the actual amount in millimetres (Wilks, 2011). Second, binning makes it easier to accommodate the extremely highly skewed shape of the distribution of

rainfalls, for which most days receive very little, or no rain and a minority receive heavy rain. By discretizing into bins, extreme values are handled more precisely, and the class balance is more required for modelling.

Through categorial binning, the models gained a complementary perspective on rainfall. The continuous features (e.g., lags, rolling means, seasonal indicators) capture the numerical patterns, and the categorical bins highlight the useful thresholds dividing no rain, light rain, and heavy rainfall events. A side effect of this binning is the creation of a class imbalance: by far most of the days are included in the “No rain” class, and “Light” and especially “Strong” days of rainfall are much less frequent. This imbalance can trick models into always predicting the majority class, and therefore requires special attention during model development, e.g., by using resampling protocols for rebalancing the training data.

3.5.4.1 Conditional Probability of today’s rain class given yesterday’s class

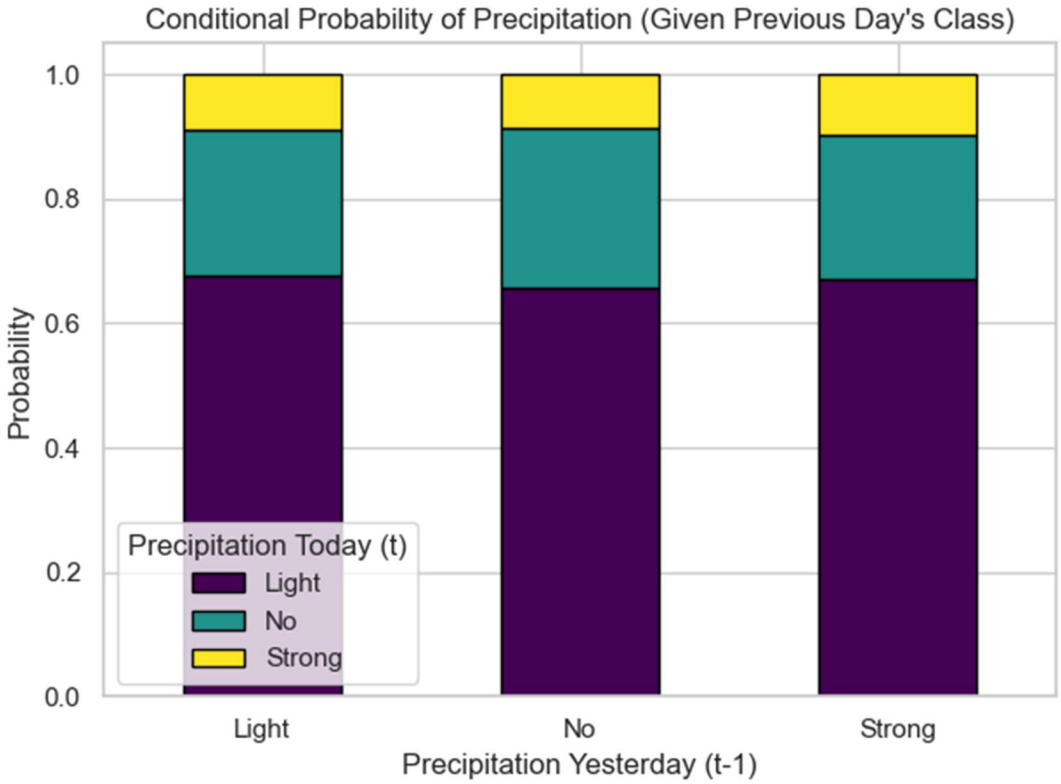

Figure 3.5.4.1.1: Conditional Probability Plot given the previous Day

Figure 3.5.4.1.1 shows the conditional probability of the precipitation category today given the category of the previous day (t−1), based on the bins listed in Section 3.5.4 (No, Light, Strong).

There are two clear visible patterns:

Light dominates all transitions. Regardless of whether before yesterday was No, Light, or Strong, Light rain is the probable result tomorrow. This is both a sign of the inherent class imbalance (Light is the prevailing class) and of the very highly skewed distribution of daily tropical

precipitation, where most of the events are light and only a few are strong (Wilks, 2011; Vogel et al., 2018).

Weak first-order dependence. Bars are nearly unchanged from one of the three “yesterday” groups to the next and therefore indicate that yesterday’s class has very little influence on the present day’s class. That is, the transition probabilities are largely equal whether the antecedent day was No, Light, or Strong. This is because our lag-correlation findings (Section 3.5.1) verify: persistence exists but is weak beyond the central monsoon region and drops off very quickly even at JAS (Rasheeda Satheesh et al., 2023).

These transitions suggest that general "yesterday’s class to today’s class" rules are not highly predictive on their own. Models therefore:

(i) Must include additional temporal features (e.g., numerical lags and moving averages) to detect short-range structure.

(ii) Must address the class imbalance (Section 3.5.4) so the frequent Light class doesn’t overwhelm rarer Strong events when training and testing. It is therefore advisable to use metrics other than complete accuracy (precision/recall/F1 by class) when evaluating skill fairly (Wilks, 2011; Vogel et al., 2018).

3.6 Handling Class Imbalance

As specified within Section 3.5.4, the rain values were put into three bins: No (0 mm), Light (≤10 mm), and Strong (>10 mm). While this binning provided an easier-to-interpret viewing of rain strength, it created a class imbalance problem. Most of the days within the data are of the Light class, then the next greatest is the category of no rain, and the Heavy rain events are rather rare. This skew was then further exaggerated within Section 3.5.4.1, where the findings of the conditional probability plot (Figure 19) were that the “Light” class dominates within all the transitions regardless of the previous day’s class.

Class imbalance is a significant issue when it comes to machine learning models. It can cause models to become biased towards the majority of the classes if not handled properly, resulting in very high overall accuracy but low performance on the minority of the classes. For a rain prediction task, this is especially undesirable because these rare phenomena, like Heavy rain, are of the highest societal impact, for example, when it comes to flooding, agriculture use cases, or disaster management. A model predicting always “Light” or always “No” rain may show statistical accuracy, but will prove useless in practice (He and Garcia, 2009).

This imbalance was addressed using the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002). Unlike simple oversampling, which duplicates minority class samples, SMOTE creates new examples synthetically by interpolating between minority class examples within feature space. It simply generates “new but realistic” examples of Strong rainfall by learning from points nearby rather than simply repeating existing ones. This expands the representation of minority classes so that they are more visible for the model at training time.

Application of SMOTE in this project was standard: first, the dataset was split into test and training sets, and then SMOTE was applied only to the training set. That is, the test set remained untouched and served as a reasonable baseline by which the quality of models could be assessed.

Overall, addressing class imbalance using SMOTE was a crucial step for model development. By reducing the dominance of the “Light” class and strengthening the rare “strong” rainfall events, the models were permitted to discern significant differences within all rainfall classes. In order to properly evaluate resampling, SMOTE was applied where appropriate. That is, the Decision Tree was learned on both raw imbalanced data and on a training set sampled using SMOTE; Random Forest and Logistic Regression were learned using SMOTE; K-Nearest Neighbours was learned using no SMOTE (to preserve neighbourhood relationships and ease of computation). Through this process, it was possible to conduct a direct comparison of performance and test not only their general performance, but also their capacity after SMOTE treatment. This aligns with the project’s objective: not only to predict common conditions of rainfall but further develop the ability to predict rarer high-impact examples (Fernández et al., 2018).

Chapter 4: Implementation

4.1 Overview

The previous chapters developed the theoretical foundation, reviewed relevant literature, and outlined how the dataset was prepared. This chapter now shifts into the implementation phase, where those earlier techniques are put into action. The focus here is to use those engineered features and preprocessing steps into predictive models that can predict rainfall categories (No, light or strong). following the approach outlined in Section 3.2 Data Acquisition and Description, all models were trained and tested on a sample of a five-year IMERG dataset (2020–2024). This constraint was selected to balance computational feasibility with representativeness of recent rainfall dynamics.

As discussed under Chapter 3, feature engineering methods were applied to the dataset. They were:

- **Lag features** - store short-range persistence of rainfall by retaining precipitation values from the past few days.
- **Rolling averages** - remove day-to-day fluctuations and reveal short, medium, and long-range rainfall trends.
- **Seasonality** - cyclical pattern of rain using sine and cosine transformations of day of year.
- **Categorical binning** - rainfall is binned into No, Light, and Strong bands to provide more sensible classification.

One of the main challenges was class imbalance. Most days in the dataset recorded Light or No rain, while Strong rainfall events were relatively rare. This skew can lead models to favor the more common classes, reducing their ability to detect rare but significant rainfall events. To address this, SMOTE (Synthetic Minority Oversampling Technique) was applied to selected models in the training set, helping to balance the data and improve learning on minority classes.

With the preparatory work complete, the next phase is to train the predictive models. This stage is broken into two parts:

(i) **Baseline Machine Learning Models:** These include simple yet proven machine learning algorithms like Decision Trees, Random Forests, Logistic Regression, and K-Nearest Neighbours. They serve as interpretable benchmarks and help test whether the engineered features alone are sufficient to capture rainfall behaviour.

(ii) **Deep Learning Model:** A Long Short-Term Memory (LSTM) network is used to evaluate whether more advanced neural networks can better capture temporal patterns than the simpler baseline models.

This systematic approach ensures that the implementation is both reproducible and clear. Using a sequence of steps from baseline models up to deep learning architectures, the project systematically takes account of the predictive capacity of different methods, while maintaining clear connections to the rationale and methodological choices outlined in earlier chapters.

4.2 Baseline Machine Learning Models

4.2.1 Decision Tree Classifier

The first baseline model implemented was the Decision Tree Classifier. A Decision Tree is a supervised learning algorithm that makes predictions by splitting the dataset into smaller sets based on a sequence of if–then rules (Breiman et al., 1984). At each split, the algorithm selects the feature and threshold that best partitions the classes (in this case, no, Light, and Strong rainfall) and repeats the process recursively until termination criteria such as maximum depth is achieved. This resulting model is easy to visualise and interpret and is consequently commonly used when facing environmental and classification problems.

In the context of this project, Decision Trees provide a simple baseline. They can capture non-linear thresholds within rainfall features such as persistence from lagged rainfall or accumulated effects, rolling averages and are easy to interpret. Unlike complex models, Decision Trees do not come with assumptions of linearity or scaling and can ingest numerical features (e.g., rolling averages, sine/cosine seasonals) and categorical inputs (e.g., rainfall bins) directly.

To implement the model, the first step was to prepare the input features and the target variable. The features included lagged rainfall values (lag-1 and lag-2), rolling averages over 3, 7, and 30 days, and seasonal indicators using sine and cosine transformations of the day of year. The target variable rainfall categories (No, Light, Strong) were converted into a numeric format using a label encoder. This allowed the classifier to interpret the categorical labels as numerical values without altering their original meaning.

The dataset was then divided into training and test sets using stratified sampling, a method used to ensure the ratios of each rainfall class are retained within both sets. For example, if 15% of the entire dataset is of type “Strong” rainfall days, then roughly 15% of both training and test sets are of type “Strong” days. This step is important because it prevents rare classes from being under-represented within the test set and therefore prevents biased verification of the models (Hastie, Tibshirani and Friedman, 2009).

We also addressed the problem of class imbalance. As described in Section 3.6, “Strong” rainfalls are significantly rarer than are “Light” or “No” rainfalls. Class weights were utilised in an attempt at rectifying bias toward majority classes. They penalised the system harder when it made the wrong conclusion on infrequent events and caused the Decision Tree to focus more on minority classes.

The model was regularised by limiting its maximum depth to 5, so it didn't get too deep and overfit the training data. It was then trained and tested on the test set. It was checked on performance by using a classification report (precision/recall/F1-score by class) and a confusion matrix, and both clearly indicated whether the model did a decent job of capturing rare “Strong” rainfall events by comparison with capturing the more frequent categories.

The essential part of this implementation is shown in this code snapshot below, where the Decision Tree is generated using a maximum depth limit and class weights:

```
# Training of the model
clf = DecisionTreeClassifier(max_depth=5, class_weight=class_weights, random_state=42)
clf.fit(X_train, y_train)

# Prediction
y_pred = clf.predict(X_test)
```

Figure 4.2.1.1: Decision Tree Classifier regularisation with maximum depth and class weights

In addition to training on original imbalanced data, the Decision Tree was trained on a SMOTE-balanced dataset. As discussed in Section 3.6, SMOTE generates synthetic examples of minority classes to reduce the imbalance. This ensured that "Strong" rain events were better represented at training, such that the classifier was better placed to identify patterns pertaining to these rare but crucial outcomes. By comparing the standard Decision Tree with its SMOTE-enhanced version, it was possible to check whether balancing the training data improved predictive performance across the rainfall categories.

The decision tree served as a valuable benchmark. It was explanatory enough so that we would know which of the rainfall features was primarily driving classification, but simple enough so that the results were not surprising. Meanwhile, it was not deep enough to capture the highly dynamic temporal structures of the tropical rainfall. This is where the comparison of Decision Trees with deeper models, such as Logistic Regression, K-Nearest Neighbours, and LSTMs, enters.

4.2.2 Logistic Regression

The second baseline model implemented was Logistic Regression, which provides a simple but effective method for multi-class classification. Unlike linear regression, which predicts continuous values, Logistic Regression estimates the probability of an observation belonging to each class. In this project, the multinomial version of Logistic Regression was applied, allowing the model to classify rainfall into the three categories: No, Light, and Strong (Hosmer, Lemeshow and Sturdivant, 2013).

Logistic Regression works by calculating a weighted sum of the input features and then applying a logistic function, which converts this sum into probabilities between 0 and 1. The predicted class is the one with the highest probability. For example, given yesterday's rainfall (lag-1), the 7-day rolling average, and the seasonal sine/cosine indicators, the model might calculate a 70% chance of Light rainfall, 20% chance of No rainfall, and 10% chance of Strong rainfall, leading to a prediction of Light.

The same engineered features were used with the Decision Tree: lagged rainfall (lag-1 and lag-2), rolling averages (3, 7, 30 days), and seasonal indicators (sine and cosine of the day of year). Because Logistic Regression is sensitive to feature scales, the features were standardised to ensure that no single variable dominated the model. The rainfall categories were label-encoded into numerical values, and the dataset was split into training and test sets using stratified sampling to preserve the proportions of each class.

The model was implemented using the scikit-learn multinomial solver. A snapshot of the core model setup is shown below:

```
log_reg = LogisticRegression(max_iter=1000, multi_class='multinomial', solver='lbfgs')
log_reg.fit(X_train_sm, y_train_sm)

y_pred_lr = log_reg.predict(X_test)
```

Figure 4.2.2.1: Logistic regression implementation using scikit-learn multinomial solver

Logistic Regression was trained on a SMOTE-balanced dataset such that the minority "Strong" rainfall instances were adequately represented at training time. The test dataset was kept untouched and used to provide a realistic performance measure. This allowed the system to learn from a balanced training distribution, but was tested under real-world actual class ratios.

```
log_reg.fit(X_train_sm, y_train_sm)
```

Figure 4.2.2.2: SMOTE- resampled training data for logistic regression

To achieve a more balanced training procedure at the class level, the model was trained using the SMOTE-resampled training data (X_train_sm, y_train_sm) as shown in Figure 4.2.2.1. These SMOTE-resampled data sets ensured that minority “Strong” rain events were properly represented.

Logistic Regression added a useful probabilistic baseline to this project. It is strong where it is fast and simple, and where coefficients can show the sign and relative influence of each feature on the outcome. Nevertheless, since Logistic Regression makes linear assumptions about associations between predictor and target, it may miss the more subtle, non-linear relationships that characterise tropical rain. This limitation invites our interest in non-linear classifiers such as K-Nearest Neighbours subsequently.

4.2.3 Random Forest Classifier

The third baseline used was the Random Forest Classifier, an ensemble method that builds on the decision tree approach by combining the output of many individual trees. Each tree is trained on a randomly selected subset of the training data, and at each split, only a random subset of features is considered (Breiman, 2001). By combining the predictions of many diverse trees through majority voting, Random Forest reduces the variance and instability often seen in individual trees, resulting in a more stable and accurate classifier.

Random Forests are especially well-suited for complex classification problems because they can capture non-linear relationships and interactions between features. Like decision trees, they handle both continuous features (like rolling averages or sine/cosine indicators) and categorical ones (such as rainfall bins). However, by using an ensemble of trees, they avoid the tendency to overfit. Another strength is their ability to generate feature importance scores, which reveal which variables, such as lagged rainfall or seasonal patterns, contribute most to predictions. This adds interpretability, even though the overall ensemble is less transparent than a single decision tree.

The features used for the Random Forest were consistent with those in earlier models: lag-1 and lag-2 rainfall values, rolling averages over 3, 7, and 30 days, and seasonal indicators based on sine and cosine transformations. Rainfall categories were label-encoded into numeric values, and the data was split into training and test sets using stratified sampling to ensure that the proportions of No, Light, and Strong rainfall events were maintained.

The following code snippet shows the core setup of the Random Forest model:

```
# Train Random Forest
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=20, random_state=42)
rf_clf.fit(X_train_sm, y_train_sm)

# Predictions
y_pred = rf_clf.predict(X_test)
```

Figure 4.2.3.1: Random Forest classifier core setup for training

Here, n_estimators (number of trees) was set to 100, the usual starting parameter that strikes a balance between accuracy and computation time. Class weights were also applied during training to give more attention to minority classes at each split. As with Logistic Regression, the Random Forest was trained on the SMOTE-balanced training set (X_train_sm, y_train_sm), ensuring that the relatively rare "Strong" rainfall events were well represented. The test set was left unchanged to reflect real-world class distributions.

In general, Random Forest provided a stronger non-linear baseline than a single decision tree. Averaging the results of many trees reduced overfitting and captured more complex patterns in the rainfall data. Its feature importance scores also offered insights into which engineered features were most useful for prediction. These strengths make Random Forest a valuable part of the baseline model suite, complementing the interpretability of Decision Trees, the simplicity of Logistic Regression, and the instance-based reasoning of KNN.

4.2.4 K-Nearest Neighbours (KNN)

The fourth baseline model implemented was the K-Nearest Neighbours (KNN) Classifier, a simple algorithm that predicts the class of a new observation based on its similarity to past cases. KNN simply finds the most similar k neighbours (closest) in the training data and takes the majority class among them (Cover and Hart, 1967). For example, if most of the closest past days similar in lagged rainfall and rolling average were the "Light" rainfall days, then the new day would itself be predicted as a "Light."

KNN is simple and flexible because it is low-maintenance and versatile. It makes no assumptions about a particular mathematical output/input relationship, like models such as Logistic Regression. It is entirely data-driven. what it outputs is entirely a function of the training data structure. This is a strength when dealing with complex or non-linear relationships, such as tropical rainfalls, where feature relationships cannot easily be represented using linear relationships. Even with this strength, KNN is scale sensitive because it relies on measures of distance. All input features were hence standardised before training so that variables such as rolling averages and rainfall contributed equally when computing distances.

Same engineering features were used as before: lag-1 and lag-2 precipitation, rolling averages (3, 7, and 30 days), and seasonal indicators (sine and cosine of day of year). The target was the categorical rainfall bins (No, Light, Strong), encoded numerically for classification. Again, the dataset was divided into training and test sets using stratified sampling based on the original class ratios.

The following code illustrates the core setup of the KNN Classifier:

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

# Predict
y_pred = knn.predict(X_test_scaled)
```

Figure 4.2.4.1: Setup of KNN Classifier

Here, the parameter n_neighbors was set at 5, so new observations were categorised based on the majority vote of their 5 neighbours. Even though this value of k was arbitrarily chosen after considering it a sensible starting point, k can be further calibrated in practice such that both the model's bias and variance can both come closer to being balanced.

While the Decision Tree, Logistic Regression, and Random Forest classifiers were trained on SMOTE-sampled data, the KNN classifier was not. This was because it was feared that oversampling would simply duplicate or artificially generate points and therefore warp the local neighbourhood structures KNN depends on. Instead, training was undertaken using the raw, imbalanced dataset, and performance was measured using the same metrics as the other classifiers.

Overall, KNN gave a useful non-linear baseline. Its instance-based approach represented a fundamentally different viewpoint from models that acquire global decision rules, but their sensitivity to imbalanced data and computational expense when handling very large datasets are significant disadvantages. The results of evaluations in subsequent sections are shown to reveal how KNN fared by comparison with the other baselines.

4.3 Long Short-term Memory (LSTM) Network

In addition to the baseline models, a Long Short-Term Memory (LSTM) network was implemented to evaluate whether deep learning can identify the temporal associations of rainfall more accurately. LSTMs are a special type of recurrent neural network (RNN) designed specifically to handle sequential data where observation order is significant. Unlike traditional networks, LSTMs possess memory cells and gate mechanisms that allow them to remember useful information of previous time steps and forget useless information (Hochreiter and Schmidhuber, 1997).

This architecture is particularly suitable for predicting precipitation, where rainfall now often depends on weather conditions of the past few days. For example, if the past few days were characterised by constant rain, then the network can recall this information and use it when it makes the next day’s prediction. This short-range dependency is worth modelling in the tropics because rainfall there is driven by quick-changing convection processes.

Input features to the LSTM were the same as baseline models: lagged rain values (lag-1 through lag-4), moving averages (3, 7, 30 days), and sine/cos seasonal dummies. All the features were scaled prior to training using the MinMax normalisation such that values were between 0 and 1, and network training was stabilised. Input data was transformed into a sequence type with a time dimension, and was the correct input format of the LSTM architecture.

Network architecture developed for this purpose was built using Long Short-Term Memory (LSTM) architecture because of its strength in handling sequential data like distributions of daily rainfall. It consisted of an LSTM layer of 64 units to capture temporal dependencies, a dropout layer to

reduce the possibility of overfitting after training. A dense (fully connected) hidden layer then ensued and finally an output layer with a softmax activation function.

We selected softmax because we were to classify into one out of three rainfall categories (No rain, Light rain, or Heavy rain). A basic classifier will give us a hard label but softmax makes our model's outputs a probability distribution that sums to 1. That is, the network doesn't merely predict the most likely class but gives us how certain it is about each. For example, a prediction will have 70% sure about Light rain, 20% about No rain, and 10% about Heavy rain. Our highest-scored category becomes our final output, but having the whole probability collection makes us able to interpret the prediction more meaningfully (Goodfellow, Bengio and Courville, 2016).

To train the model effectively, the Adam optimiser was used. Adam Optimiser is very popular since it can adaptively learn. It adjusts the learning rate during training to help fast convergence and minimise chances of converging to poor solutions (Kingma and Ba, 2015). Since this was a multi-class classification problem, training utilised a categorical cross-entropy loss function that is designed particularly for situations where only a single class is correct. It works such that it compares predicted probabilities made against actual class labels and imposes a stiffer penalty upon realising that a model is overly confident about a wrong prediction.

The following code illustrates the core architecture of the LSTM model:

```
# 6. Building the LSTM model
model = Sequential()
model.add(LSTM(64, input_shape=(1, X_scaled.shape[1]), return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(32, activation='relu'))
model.add(Dense(3, activation='softmax'))  # 3 categories

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 4.3.1: Core architecture of the LSTM model

It was trained on mini-batches with early stopping to prevent overfitting. It was not trained on SMOTE-resampled data. Oversampling could produce phantom copies of sequential structures and harm sequential dependencies that a network is designed to learn. Instead, class imbalance was addressed at testing time by reporting a measure corresponding to a broader set of rainfall categories, including these metrics: precision and recall and an F1-score across rainfall categories.

LSTM was overall the most advanced model we considered. Its ability to incorporate time information and non-linear relations provided a solid baseline against which to compare other methods. It was, however, computationally most costly and required a proportionally higher amount of training time and effort to tune parameters. Comparison results follow later in this report and assess how well its performance is compared to baseline models, and assess whether deep learning offers a meaningful advantage in short-term rainfall predictions.

Chapter 5: Model Evaluation and Explainability

5.1 Evaluation Metrics

For model performance evaluation in classification tasks, it requires more than checking how often a model is accurate. Rainfall events considered here were labelled into three categories (No, Light, and Strong), and the dataset was naturally imbalanced, such that Strong events were rather sparse. Given such an imbalance, a model might appear accurate if it guessed most frequently a majority class but still made zero calls on sparse but informative events. To get a fair and complete picture of model performance, a variety of evaluation metrics were utilised: accuracy, precision, recall, F1-score, and confusion matrices.

Accuracy determines correct predictions out of total predictions without respect to actual class distributions. Although this is the most typical measure, it can be misleading in imbalanced data. For example, if just 80% of days get Light rainfall, a model can reach 80% precision simply guessing “Light” every occasion but never accurately calling Strong occasions (Sokolova and Lapalme, 2009).

Precision determines how frequently cases predicted to be a particular class were classified accurately. An example would be precision for Strong rainfall: “If the model outputs Strong rainfall, how often is it correct?” A high level of precision reduces false alarms, desirable in operational forecasting, where over-prediction of extremes could lose people's confidence in the model.

Recall refers to how successful a model was in identifying the actual occurrences of a specific class. For Strong rainfall, recall answers: “Of all days on which Strong rainfall actually did occur, how many did the model detect?” High recall indicates fewer such events missed, and this is especially relevant for extreme rainfall because misses might have severe consequences.

F1-score is a harmonic mean between recall and precision (Sokolova and Lapalme, 2009). It provides a single number that's balanced, especially if a dataset is imbalanced. If a model has a high F1-score, then it's accurate whenever it's making a class decision (very precise) and successful at finding a class whenever it's present (High Recall).

Finally, confusion matrices were used to visualise outputs. A confusion matrix is a very simple table that compares predicted model outputs against actual outputs. It shows how many instances out of some rainfall categories were classified correctly and how many incorrectly within each rainfall category. It provides a very intuitive idea about strengths and weaknesses, such as if a model always confuses Strong rainfall and Light rainfall.

Confusion Matrices divide model predictions into four categories:

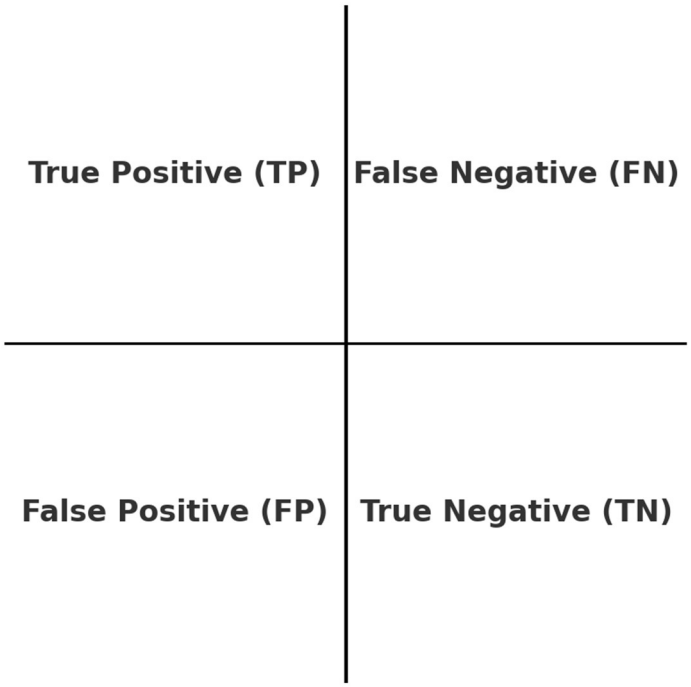

Figure 5.1.1: Confusion Matrix Explanation

- True Positives (TP): Situations when the model correctly classifies a rainfall class. For example, if the actual is “Strong” rainfall and the model similarly classifies “Strong” rainfall, then it is a TP.
- False Positives (FP): Situations in which a model inaccurately classifies a rainfall class. If a model classifies “Strong” but it was actually “Light,” that would be considered a FP of the Strong class.
- False Negatives (FN): Instances when the model doesn't recognise the true class. In our example, if the true rainfall was “Strong” but it was predicted as “Light” by our model, that is a FN for the Strong class.
- True Negatives (TN): All remaining cases where the model correctly predicts that an instance does not belong to the class in question.

Other than these usual evaluation metrics, the LSTM model was tested during training regarding its performance per epoch. A pass through the whole training dataset defines an epoch. Following the model's training and validation accuracy and loss between sequential epochs enabled a visual check if a network was learning appropriately, if it was overfitting or if it was underfitting (Goodfellow, Bengio and Courville, 2016). These learning curves provide complementary information: a progressively improving validation accuracy suggests that a model is generalising appropriately, while a difference between training performance and validation performance suggests overfitting. For this reason, epochs and training curves were also considered part of the evaluation process for the deep learning model.

5.2 Model Evaluation

5.2.1 Decision Tree Classifier

As introduced in Section 2.2.3.1 Decision Trees, we first implemented a Decision Tree classifier as a baseline model. Decision Trees classify a dataset stepwise into progressively smaller partitions along rules derived from predictor features, which makes it simple to interpret and useful for classification (Breiman et al., 1984). We used input features like lagged features, rolling averages, and season indicators engineered during feature engineering (see Sections 3.5.1–3.5.3) as features. Categorical rainfall bins (No, Light, Strong) engineered during Section 3.5.4 Categorical Binning served target classes for classification.

Classification Report:

	precision	recall	f1-score	support
Light	0.86	0.50	0.63	4640031
No	0.38	0.72	0.50	1656821
Strong	0.58	0.99	0.73	620303
accuracy			0.60	6917155
macro avg	0.60	0.74	0.62	6917155
weighted avg	0.72	0.60	0.61	6917155

Figure 5.2.1.1: Decision tree classifier Report

First, it was trained on the raw imbalanced dataset. Just like Section 3.6 Handling Class Imbalance indicated, rainfall data is highly imbalanced because most days lie within the Light or No rainfall class, but a negligible amount lies within the Strong class. The evaluation confirmed this imbalance (Figure 5.2.1.1 and Figure 5.2.1.2):

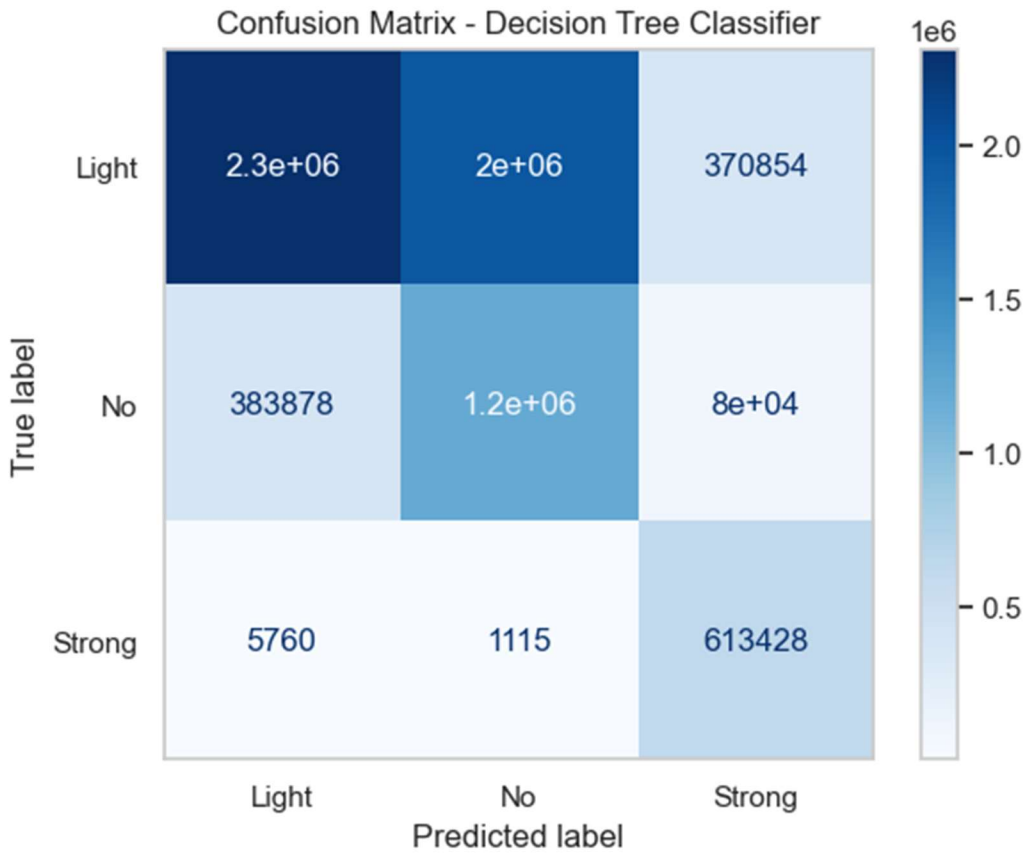

Figure 5.2.1.2: Decision Tree Confusion Matrix

- The overall accuracy was approximately 60%, but this masked significant differences between rainfall categories.
- High accuracy was achieved in Light rainfall (0.86), but low recall (0.50) was observed; that is, predictions were highly accurate, but most genuine Light rainfall days were not considered.
- No-rainfall was the least performing category, with a precision of 0.38 and a recall of 0.72, mainly due to a high number of false predictions
- Strong rainfall achieved a very high recall (0.99) but a moderate precision (0.58) because nearly all strong rainfall events were picked up, but many false alarms were made.

The confusion matrix further indicated large misclassification between Light and No rainfall, and fewer mistakes were witnessed in Strong rainfall.

To address these limitations, the SMOTE-balanced dataset (Chawla et al., 2002) was again used to train the model. Figure 5.2.1.3 and Figure 5.2.1.4 include this iteration's classification report and confusion matrix. Results demonstrated improvement across all categories:

	precision	recall	f1-score	support
Light	0.92	0.86	0.89	811196
No	0.68	0.80	0.74	308384
Strong	0.99	0.99	0.99	108800
accuracy			0.86	1228380
macro avg	0.86	0.88	0.87	1228380
weighted avg	0.87	0.86	0.86	1228380

Figure 5.2.1.3: Classification Report for Decision Tree with SMOTE

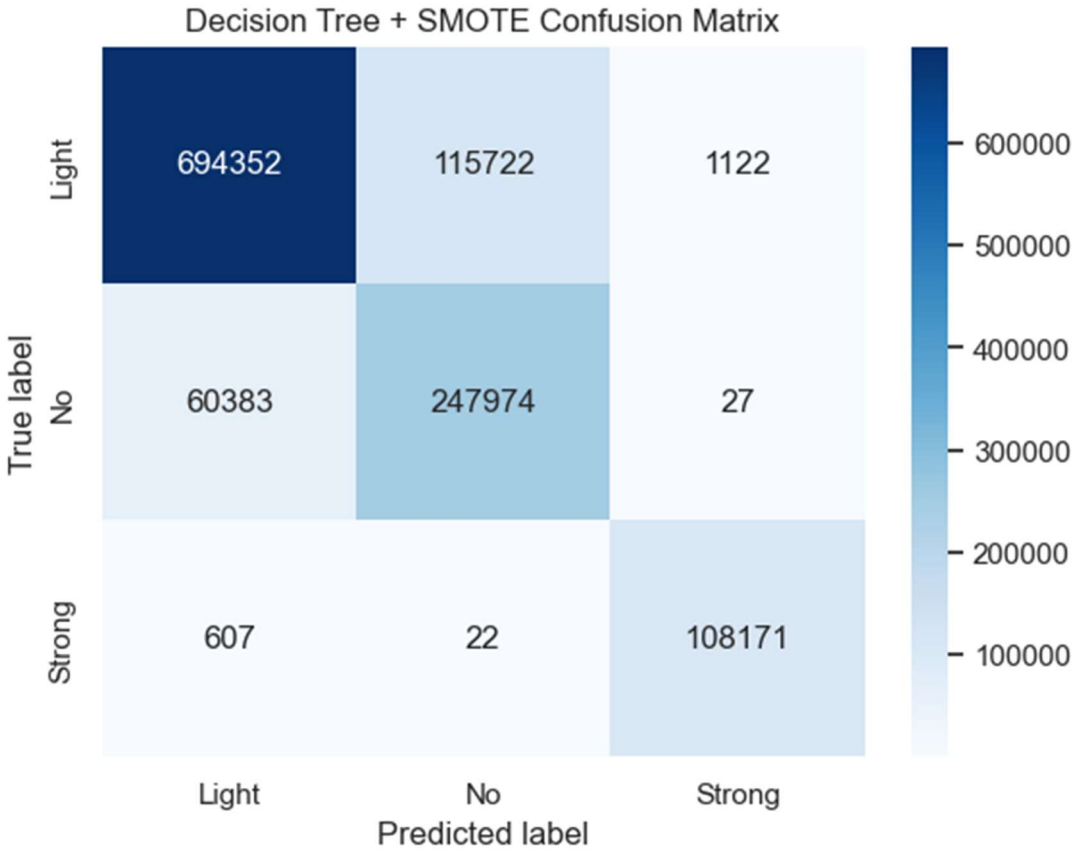

Figure 5.2.1.4: Confusion Matrix for Decision Tree with SMOTE

- Overall accuracy achieved 86%, and both macro F1-score (0.87) and weighted F1-score (0.86) suggested that performance was highly balanced across categories of rainfall.
- Precision in Light rainfall performance was higher, improving to 0.92 and recall to 0.86 while producing a highly balanced F1-score of 0.89.

- No rainfall also made notable enhancements with precision up to 0.68 and recall up to 0.81 while having an F1-score of 0.74 versus much weaker performance in the imbalanced model.
- Strong rainfall achieved near-perfect classification, as the precision, recall, and F1-score were approximately 0.99, reflecting that almost every instance of Strong rainfall was classified correctly.

The confusion matrix revealed that over 248,000 no-rainfall days were accurately classified, and misclassification between No and Light rainfall was reduced. Errors related to Strong rainfall were negligible, such that only a total of 609 cases were misclassified.

Overall, these findings suggest that while the initial Decision Tree was strongly affected by class imbalance, the application of SMOTE resulted in a more balanced and consistent model across all rainfall categories. This outcome underlines the importance of addressing class imbalance in rainfall prediction, particularly when rare but critical events must be detected (Chawla et al., 2002; Fernández et al., 2018).

5.2.2 Logistic Regression

As described in Section 2.2.3.2 Logistic Regression, this model is a typical baseline classifier that estimates the probability of each class through a logistic function (Cox, 1958; Hosmer, Lemeshow and Sturdivant, 2013). Unlike tree-based algorithms, Logistic Regression assumes a relationship between input features and log odds of the output and is therefore interpretable and computationally efficient. We took the same engineered features (lagged features, rolling averages, and season indicators) as inputs, but used categorical bins for rainfall (No, Light, Strong) as our response classes.

Since the dataset was highly imbalanced, we trained a Logistic Regression model against a SMOTE-balanced dataset, which is presented in Section 3.6 Handling Class Imbalance. Figure 30 presents the classification report, and Figure 5.2.2.1 shows the confusion matrix. These two provide a complete idea regarding how rainfall categories are represented by the model.

Logistic Regression Classification Report:

	precision	recall	f1-score	support
Light	1.00	0.83	0.91	817032
No	0.69	1.00	0.82	302620
Strong	1.00	1.00	1.00	108727
accuracy			0.89	1228379
macro avg	0.90	0.94	0.91	1228379
weighted avg	0.92	0.89	0.89	1228379

Figure 5.2.2.1: Classification Report for Logistic Regression

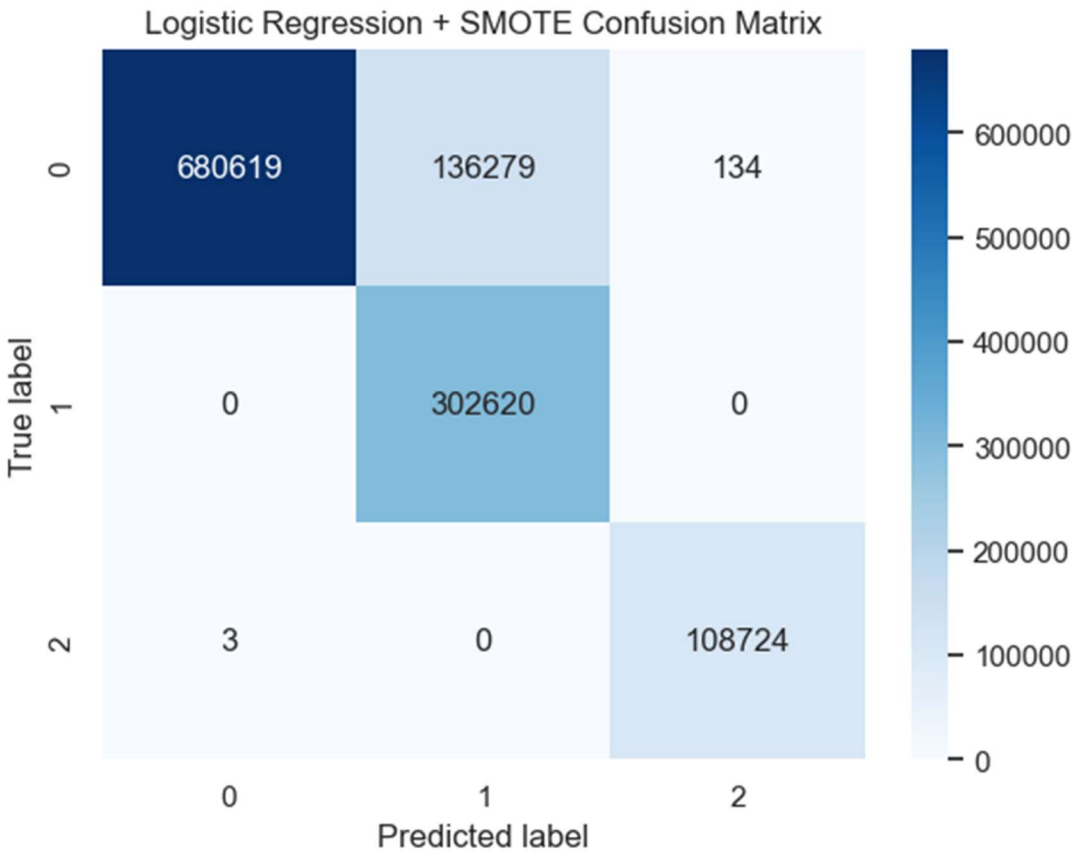

Figure 5.2.2.2: Confusion Matrix for Logistic Regression

- Overall accuracy was 89%, and macro and weighted average F1-scores were 0.91 and 0.89 correspondingly. That means two things: good predictive power and a consistent level of performance across this dataset.
- For Light rainfall, optimal precision was achieved (1.00), and recall was 0.83, creating an F1-score of 0.91. That is, almost all Light rainfall was accurately predicted, but some actual Light instances weren't predicted.
- For No rainfall, performance was considerably better compared to Decision Trees without balancing. Precision was 0.69 and recall was 1.00, which provided an F1-score of 0.82. This shows that nearly all no-rainfall days were captured, but some false alarms remained.
- For Strong rainfall, performance was practically perfect, with precision, recall, and F1-score all at 1.00. This is consistent with the ability of the model to readily discriminate. these are very rare but distinct events.

The confusion matrix (Figure 5.2.2.2) also shows these trends. While no rainfall and Strong rainfall were correctly classified confidently, some large misclassifications did still occur between Light and no rainfall. Specifically, around 136,000 Light rainfall days were misclassified as no rainfall days, but smaller numbers of no rainfall days were incorrectly predicted as Light. Most prominently, Strong rainfall was scarcely confused with other categories, demonstrating how robust the model was in characterising extreme events.

Logistic Regression demonstrated strong and balanced results when it was trained on SMOTE. It's a linear model, so it's less flexible than higher-order models, but the consistency of its performance across categories shows that even quite basic methods can achieve consistent short-term rainfall predictions if paired with correctly engineered features and appropriate handling of class imbalance (Hosmer, Lemeshow and Sturdivant, 2013).

5.2.3 Random Forest Classifier

Random Forest is an ensemble learning algorithm that aggregates many decision trees to produce steadier and more reliable predictions (Breiman, 2001). Instead of implementing a single tree, the algorithm creates a “forest” of trees that have been trained on different random samples of features and data. By producing a final prediction that is a vote-based combination of outputs of all trees, the problem of overfitting is reduced, and generalisation is boosted. For this task, a Random Forest model was trained on the same engineered features used in the other baseline models, with the rainfall categories (No, Light, Strong) as target classes.

Random Forest Classification Report:

	precision	recall	f1-score	support
Light	0.92	0.74	0.82	817032
No	0.54	0.83	0.66	302620
Strong	0.97	1.00	0.98	108727
accuracy			0.78	1228379
macro avg	0.81	0.85	0.82	1228379
weighted avg	0.83	0.78	0.79	1228379

Figure 5.2.3.1: Random Forest Classification Report

Since categories of rainfall were skewed, a SMOTE-balanced dataset was used to train a Random Forest (Chawla et al., 2002) corresponding to Section 3.6 Handling Class Imbalance. Figure 5.2.3.1 (classification report) and Figure 5.2.3.2 (confusion matrix) show evaluation results.

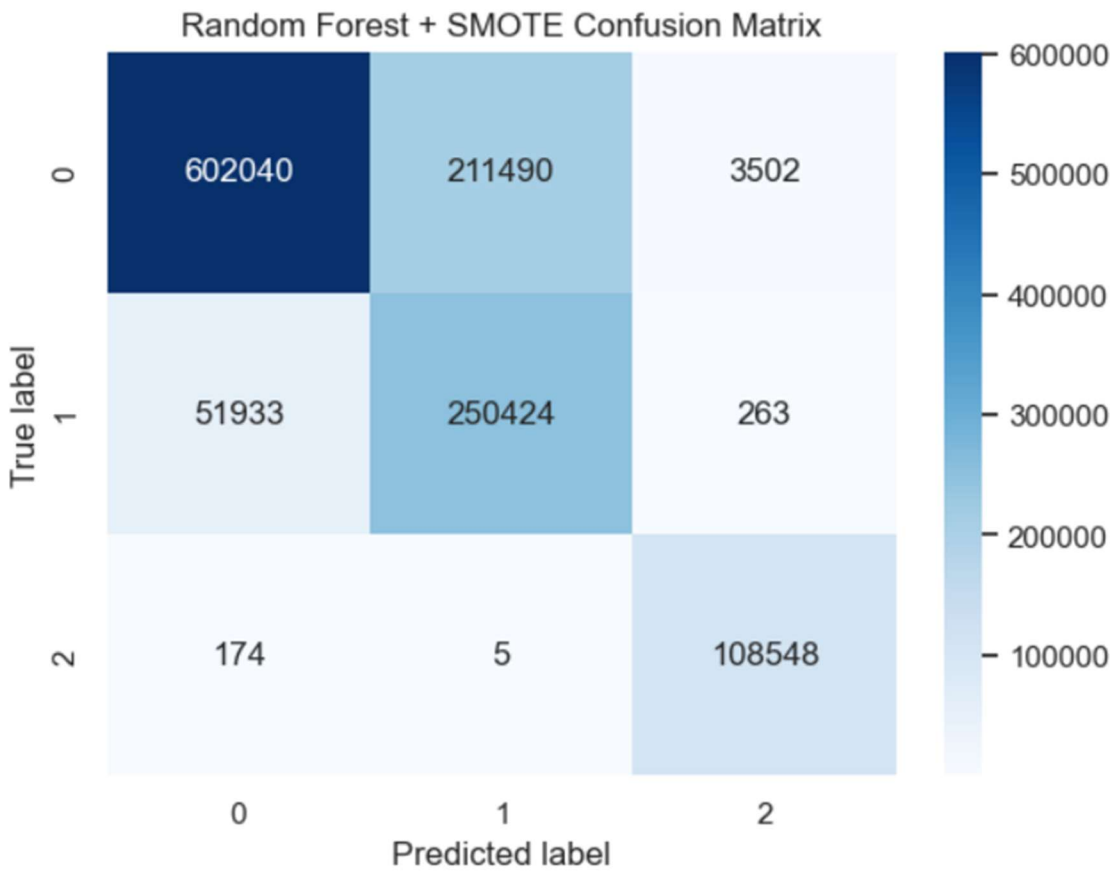

Figure 5.2.3.2: Random Forest Confusion Matrix

- Random Forest model achieved overall accuracy of 78%, a macro F1-score of 0.82 and a weighted F1-score of 0.79. These reflect respectable but poorer performance relative to Logistic Regression and Decision Tree using SMOTE.
- For Light rainfall, it obtained a precision of 0.92 and a recall of 0.74, thus an F1-score of 0.82. It indicates that predictions of Light rainfall were mostly accurate, though some Light cases were left out.
- For no rainfall, performance was worse, with a precision of 0.54 and a recall of 0.83 (F1-score 0.66). Though most of the no-rainfall instances were correctly classified, large false positive counts reduced precision.
- For Strong rainfall, the model excelled greatly, having a precision of 0.97, a recall of 1.00, and an F1-score of 0.98, showing high reliability in identifying extreme rainfall events.

Figure 5.2.3.2 highlights these patterns in a confusion matrix. The major issue was confusing Light and no rainfall. About 211,000 Light rainfall days were misclassified as no rainfall, and about 52,000 no rainfall days were misclassified as Light. Strong rainfall was correctly classified with a very high level of certainty, with only slight misclassification in other categories.

Overall, the Random Forest classifier provided good Strong rainfall performance and moderate Light rainfall performance, but was poor in discriminating between No and Light rainfall. Though the ensemble approach improved strength compared to a single Decision Tree, it did not act in this study in a balanced way compared to Logistic Regression with SMOTE. These results indicate

how model selection should be made cautiously in function of the balance between desired precision and recall between specific rainfall categories (Breiman, 2001; Fernández et al., 2018).

5.2.4 K-Nearest Neighbours (KNN) Classifier

K-Nearest Neighbours (KNN) is a non-parametric classifier that classifies based on a majority vote between an observation's nearest neighbours in feature space (Cover and Hart, 1967). It does not generate an explicit model at training, such as Logistic Regression or Random Forests; however, it stores the entire dataset and makes classifications by calculating a measure of distance between a new observation and the samples that have been stored, typically a Euclidean measure. Whilst this makes KNN simple to use and understand, it can be computationally expensive on large datasets and is sensitive to class imbalance (Peterson, 2009).

As seen in Section 3.5 (Feature Engineering), KNN was trained in all features that were engineered (i.e., lagged features, rolling averages, and seasonality). It was trained without SMOTE balancing, however (see Section 3.6 Handling Class Imbalance). This choice serves as a good baseline of comparison with the other models, as it highlights how much class imbalance can affect performance when left unaddressed.

Classification Report:

	precision	recall	f1-score	support
Light	0.75	0.84	0.80	811196
No	0.42	0.29	0.34	308384
Strong	0.97	0.96	0.97	108800
accuracy			0.71	1228380
macro avg	0.71	0.70	0.70	1228380
weighted avg	0.69	0.71	0.70	1228380

Figure 5.2.4.1: KNN Classification report

Figure 5.2.4.1 (classification report) and Figure 5.2.4.2 (confusion matrix) report the results of the evaluation. Overall accuracy was 71%, while both macro and weighted F1-scores were 0.70, indicating moderate predictive power.

- For Light rainfall, the model performed reasonably well, having a precision of 0.75, a recall level of 0.84, and an F1-score of 0.80. This shows that KNN could identify most instances of Light rainfall but still made many false alarms.
- For No rainfall, performance was significantly weaker, having a precision of 0.42, a recall of 0.29, and an F1-score of 0.34. That confirms that discriminating between No rainfall and Light rainfall was challenging for the model, a flaw likely made worse due to the class imbalance seen above.
- For Strong rainfall, performance was strong too, with a precision of 0.97, a recall of 0.96, and an F1-score of 0.97, reflecting KNN's ability to classify correctly extreme rainfall events.

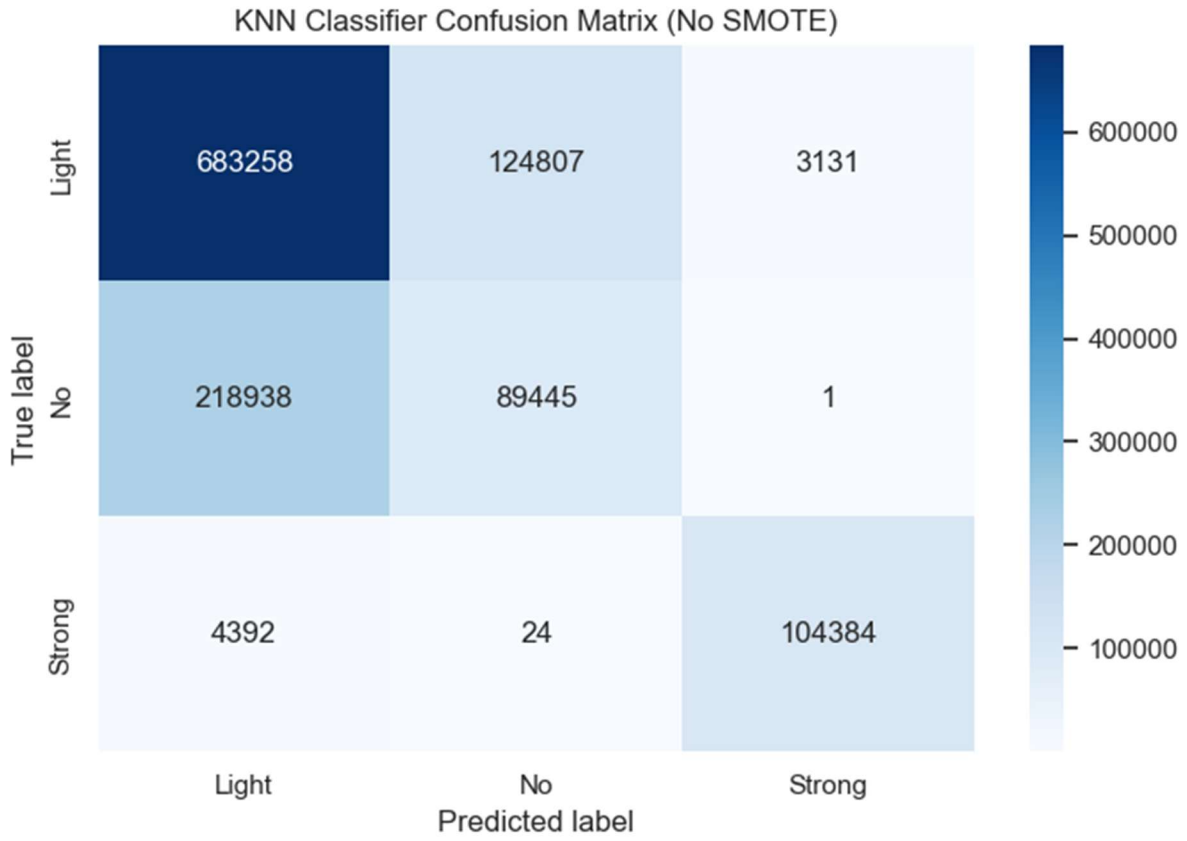

Figure 5.2.4.2: KNN confusion Matrix

The confusion matrix (Figure 5.2.4.2) further exposes these findings. A main weakness was between Light and No-rainfall misclassification. About 219,000 no-rain days were predicted as Light rainfall, and 124,000 Light rainfall days were classified as No-rainfall. Correct classification was made on Strong rainfall with minimal error instead, demonstrating KNN's ability to detect distinct rainfall extremes.

Overall, while KNN gave adequate performance for Light and Strong rainfall, low performance against the No rainfall class shows weaknesses in relying on this method without using balancing techniques such as SMOTE. This again stresses considering class imbalance while dealing with precipitation prediction using machine learning (Fernández et al., 2018).

5.2.5 Long Short-Term Memory (LSTM) Model Evaluation

Evaluation of the LSTM model was done within 10 epochs, during which both training and validation accuracy and loss were observed at each step. Figure 36 shows raw training logs that indicate model training began at approximately 67.7% during the first epoch but stabilised at around 66.5% in later epochs. Validation accuracy remained stable at around 66.6% throughout all epochs without any apparent sign of improvement. This is evidence that model convergence was quick but could not generalise beyond any performance threshold, a common issue when modelling highly complex temporal data such as tropical rainfall (Goodfellow, Bengio and Courville, 2016).

```
Epoch 1/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 99s 2ms/step - accuracy: 0.6771 - loss: 0.7799 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 2/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 102s 2ms/step - accuracy: 0.6650 - loss: 0.8310 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 3/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 111s 2ms/step - accuracy: 0.6651 - loss: 0.8308 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 4/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 105s 2ms/step - accuracy: 0.6649 - loss: 0.8310 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 5/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 106s 2ms/step - accuracy: 0.6651 - loss: 0.8311 - val_accuracy: 0.6660 - val_loss: 0.8300
Epoch 6/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 110s 2ms/step - accuracy: 0.6646 - loss: 0.8316 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 7/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 103s 2ms/step - accuracy: 0.6650 - loss: 0.8312 - val_accuracy: 0.6660 - val_loss: 0.8300
Epoch 8/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 106s 2ms/step - accuracy: 0.6655 - loss: 0.8304 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 9/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 104s 2ms/step - accuracy: 0.6655 - loss: 0.8305 - val_accuracy: 0.6660 - val_loss: 0.8299
Epoch 10/10
61419/61419 ━━━━━━━━━━━━━━━━━━━━ 104s 2ms/step - accuracy: 0.6647 - loss: 0.8314 - val_accuracy: 0.6660 - val_loss: 0.8299
38387/38387 ━━━━━━━━━━━━━━━━━━━━ 42s 1ms/step - accuracy: 0.6645 - loss: 0.8317
Test Loss: 0.8315, Test Accuracy: 0.6648
```

Figure 5.2.5.1: Raw training logs of LSTM

To further illustrate these results, Figure 5.2.5.2 graphically plots validation and training accuracy against epochs. The graph shows the early decrease during the first epoch and then levels out again, showing that though the model is acquiring information, it's not significantly improving beyond baseline levels. Figure 5.2.5.3 plots training and validation loss side by side, both levelling out again after the first epoch, which shows that maybe the model has hit capacity within this given setup.

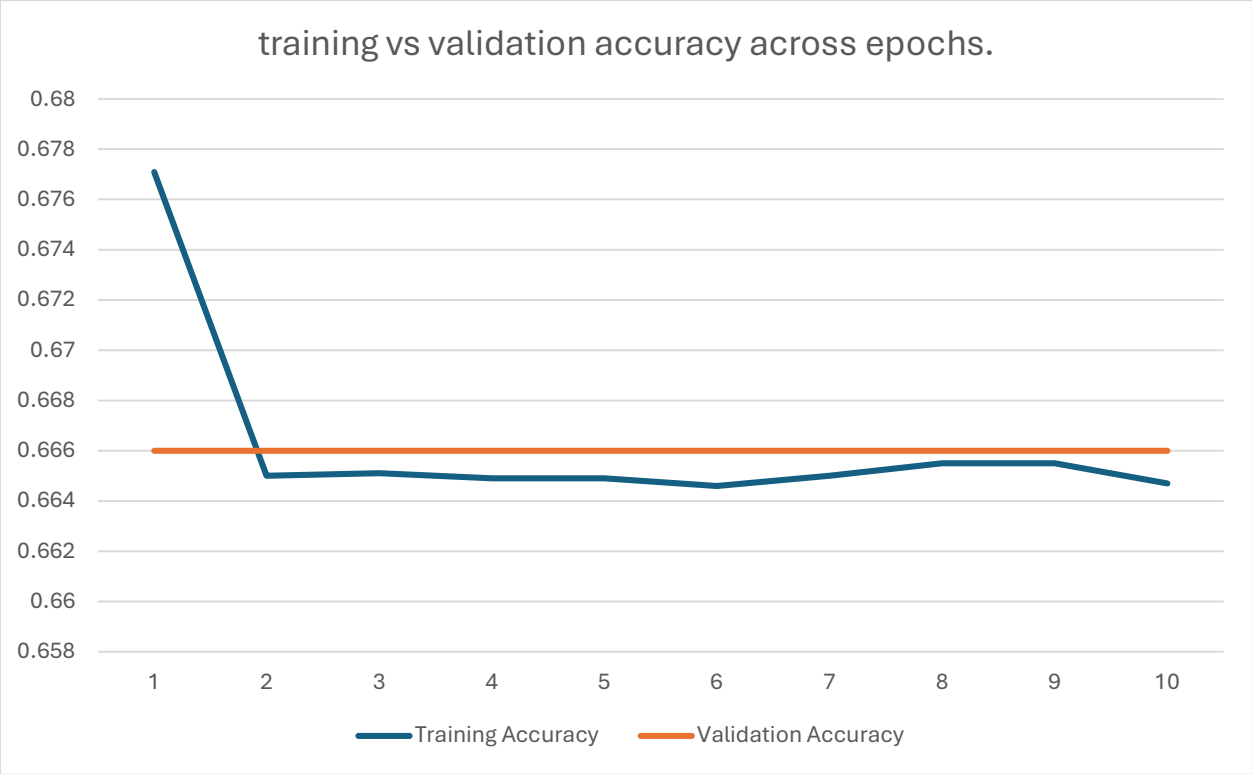

Figure 5.2.5.2: Training accuracy vs Validation Accuracy plot

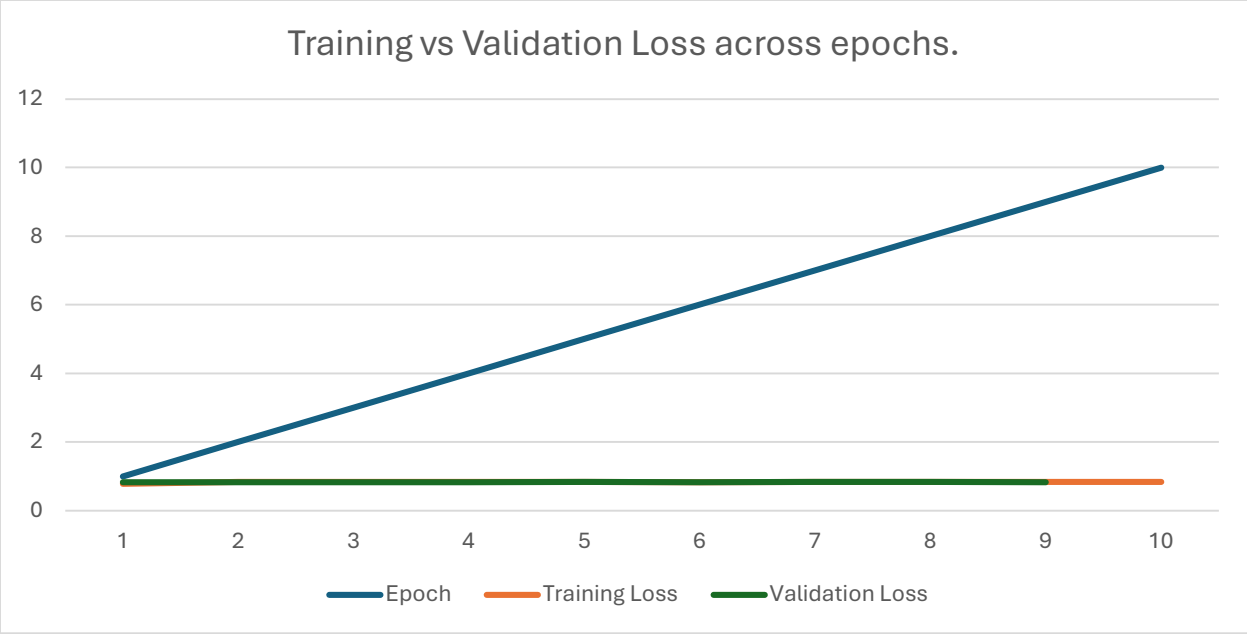

Figure 5.2.5.3: Training Loss vs Validation Loss plot

Overall, these outputs suggest that while some sequential information was maintained by LSTM, predictive capacity stabilised early in training. This could be a function of having minimal input features (only precipitation features were used) or that model depth was not sufficient to capture advanced rainfall dynamics. But consistency in validation loss and accuracy across epochs does solidify that the model did not overfit, but instead gave consistent performance on new data.

It is also noteworthy to observe the role of feature engineering constructed in Section 3.5. Though lagged features and seasonal indicators (sine-cosine transformations of the day-of-year) were added to bring in time structure, level performance throughout this component is an indication that these feature-engineered quantities were not exploited fully. This is an indication of both the difficulty in modelling tropical convective rainfall, which is known to be weakly autocorrelated (Vogel et al., 2018) or without a strong autocorrelation structure, and the requirement to exploit richer multi-variable inputs beyond precipitation alone. In short, evaluation substantiates that the LSTM ran to a consistent but constrained level of skill matching broader research that identifies tropical precipitation forecasting as a challenge even with sophisticated sequential models (Rasheeda Satheesh et al., 2023).

Chapter 6: Results and Discussion

6.1 Comparative performance of Models

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-score (Macro)	Notes
Decision Tree (Imbalanced)	60%	0.60	0.73	0.58	Struggled with 'No' rainfall.
Decision Tree (SMOTE)	86%	0.87	0.87	0.87	Strong balanced improvement.
Logistic Regression (SMOTE)	89%	0.91	0.91	0.91	Best balance across all classes.
Random Forest (SMOTE)	78%	0.82	0.82	0.82	Strong for 'Strong', weaker for 'No'.
KNN (No SMOTE)	71%	0.70	0.70	0.70	Reliable for 'Strong', weak for 'No'.
LSTM (2020–2024)	66.5%	-	-	-	Stabilised early, limited by features.

Table 6.1.1: Comparative Performance Table

The relative performance of the models assessed in this study is shown in Table 6.1.1. It displays overall accuracy along with metrics indicating how well each rainfall category was represented by the models. These results demonstrate the advantages and disadvantages of different methods for forecasting short-term rainfall in tropical Africa.

- With only 60% accuracy, the Decision Tree without balancing did not perform well. As demonstrated previously in Section 5.2.1, the model had trouble accurately identifying days with no rainfall, mistaking them for days with light rainfall. Performance dramatically improved to 86% accuracy when balancing was implemented using SMOTE, demonstrating the importance of resolving the uneven distribution of rainfall categories (Chawla et al., 2002).
- With an accuracy of 89%, logistic regression with SMOTE produced the best overall results. According to Hosmer, Lemeshow, and Sturdivant (2013), it was consistent across all rainfall categories, demonstrating that even a basic model can function admirably with well-designed features and a balanced dataset.
- Although the Random Forest detected Strong rainfall nearly flawlessly, it still had trouble distinguishing between No rainfall and Light rainfall, despite its respectable 78% accuracy (Breiman, 2001).

- The accuracy of the KNN model was 71%. Although it correctly classified "strong rainfall," it frequently confused "no rainfall" with "light rainfall." This demonstrates how sensitive models that depend on "closeness" between data points are to dataset imbalance (Cover and Hart, 1967).
- Ultimately, the LSTM deep learning model demonstrated minimal improvement during training, but it achieved 66.5% accuracy. This implies that the model was unable to discover more profound patterns when rainfall was the only input used. According to the results of other studies, deep learning may need additional input variables (like temperature, humidity, and wind) to outperform simpler models in tropical rainfall forecasting (Vogel et al., 2018; Rasheeda Satheesh et al., 2023).

In conclusion, while deep learning models are expected to be most powerful, more straightforward models, such as Decision Trees and Logistic Regression, performed better when backed by balancing strategies. This outcome shows how crucial it is to select models that are appropriate for the dataset and the computing resources.

Overall, it can be observed that the low-complexity baseline models, and Logistic Regression with SMOTE specifically, performed better than the high-complexity LSTM deep neural network model. Logistic Regression performed better in accuracy and balanced prediction across all of the rainfall classes, while, for all of its complexity, the LSTM topped out at a relatively low level of performance. It can be said that Logistic Regression handled its feature-engineered and balanced set of samples well, while the LSTM required more input variables and computational power to adequately capture the complex temporal dynamics of tropical rainfall. These findings show that in resource-scarce or data-sparse conditions, baseline models can produce better performance than deep network methods, and that is a theme that we explore further in the next section.

6.2 Interpretation of findings

The results of the research confirm that forecasting short-term rainfall in tropical Africa is still difficult. The convective nature of tropical rainfall, which is caused by small-scale storms that form and dissipate rapidly, is one of the main reasons for this problem (Rasheeda Satheesh et al., 2023). Because these storms frequently have ties with large-scale climate systems, predicting them is more challenging than predicting rainfall in other regions.

The evaluation of models in Chapter 5 indicated that simpler baseline techniques, like Logistic Regression and Decision Trees with SMOTE balance, outperformed the LSTM deep learning model. This result supports Vogel et al.'s (2018) argument that statistical and machine learning models can perform better than conventional forecasts in the tropics when appropriately designed. In our project, Logistic Regression's success can be attributed to two key factors in particular: careful use of features that were engineered (lagged features, rolling averages, and seasonality; Section 3.5), and correction of class imbalance (Section 3.6). Instead of being biased toward the most frequent results, these models could learn more consistently from all rainfall categories.

On the other hand, despite being an efficient technique, the LSTM model performed poorly. Its accuracy peaked at roughly 66.5% and slightly improved over epochs. This result reflects two important limitations. First, the model relied only on precipitation data and its derived features without additional climate variables such as temperature, humidity, or wind. Recent studies highlight that deep learning networks require more input data to capture the rainfall dynamics

(Rasheeda Satheesh et al., 2023). Second, as mentioned in section 3.2, training was limited to a 5-year subset (2020-2024) because of computational limitations. This provided feasibility but also restricted the diversity of conditions available for training, which may have limited the model's learning capacity.

When put together, the results show that although deep learning models are more complex and promising, simpler and more balanced models performed better in this scenario. This demonstrates the significance of matching model selection to available resources and the dataset.

These findings are broadly consistent with the work of Vogel et al. (2018), who showed that statistical forecasts could outperform physical models such as NWP in the tropics, particularly when forecasting rainfall intensity. Similarly, Rasheeda Satheesh et al. (2023) reported that synoptic-scale signals alone were not sufficient to capture tropical rainfall variability, reinforcing the need for data-driven approaches that focus on local and short-term predictors. The results of this project, therefore, strengthen the case for using simpler but well-balanced machine learning models in tropical contexts.

These results are consistent with the work of Vogel et al. (2018), which demonstrates that statistical forecasts can outperform physical models such as NWP in the tropical regions, especially when predicting rainfall intensity. The need for data-driven approaches that concentrate on local and short-term predictors was further highlighted by Rasheeda Satheesh et al.'s (2023) report, which found that synoptic-scale signals alone were not sufficient to capture tropical rainfall variability, reinforcing the need for data-driven approaches that focus on local and short-term predictors. The result of this project strengthens the case for using simpler but balanced machine learning models in tropical regions.

At the same time, the underperformance of the LSTM compared with expectations contrasts with studies where deep learning achieved clear improvements once richer datasets were available. This difference suggests that the potential of deep learning in tropical rainfall forecasting has not been fully realised in this project, largely due to the single-variable input and limited training period. Consequently, while the results align with the view that machine learning offers practical improvements over traditional methods, they also highlight that deep learning models may require larger and more complex datasets before their advantages can be seen.

6.3 Practical Implications

For tropical Africa's short-term rainfall forecasting, this study has several useful implications. In areas where economies rely significantly on climate-sensitive industries like agriculture, water management, and disaster preparedness, accurate rainfall prediction is not only a scientific challenge but also an urgent social necessity.

First, the findings demonstrate that more straightforward models can produce accurate forecasts if they are appropriately balanced. Forecasts from SMOTE-enabled Decision Trees and Logistic Regression were more balanced and accurate than those from the more complicated LSTM. This demonstrates that advanced deep learning architectures or large computational resources are not always necessary for forecasting tools to be effective. In practice, these lighter models could be implemented more affordably by local institutions, NGOs, and government agencies, making them appropriate for use in areas with poor technical infrastructure.

Second, the ability of these models to improve the classification of rare but high-impact events like heavy rainfall has important implications for disaster risk management. Extreme rainfall is often linked with floods and landslides, which pose serious risks to human life. By detecting these events, even with the small improvements in accuracy, these models can support early warning systems and help higher authorities allocate resources more effectively.

Third, the study demonstrated the value of recent rainfall data for operational forecasting. By using the 2020-2024 data subset, models were able to capture the patterns that are more representative of current climate variability, rather than relying on older long-term averages. This approach aligns with the increasing demand for smart climate decision-making, where up-to-date data is used to make good decisions in farming, water storage, and public safety.

In conclusion, this project's practical value exists in illustrating that feature engineering and balancing techniques, when combined with basic, resource-efficient models, can produce forecasting improvements that are both practical and significant for tropical regions.

6.4 Limitations of the study

While this study offers some valuable insights into rainfall prediction in tropical Africa, there are a few limitations that should be recognised. Acknowledging these helps clarify where the findings need to be interpreted carefully and where future research could improve on this work.

The first limitation lies in the choice of input variables. Here, precipitation was the only variable used. Although rainfall was the main outcome of interest, it is shaped by many other atmospheric factors such as temperature, humidity, and wind. By leaving these out, the models could not fully capture the complex processes that drive tropical rainfall. This limitation was particularly noticeable with the LSTM, which usually performs better when trained on multiple inputs and can pick up on more intricate relationships (Vogel et al., 2018; Rasheeda Satheesh et al., 2023).

A second limitation relates to the size of the dataset. The models were trained using only five years of data (2020-2024), mainly due to computational constraints (see Section 3.2). While this made the analysis manageable and focused attention on recent conditions, it reduced the range of scenarios the models could learn from. For example, the dataset may not have captured longer-term patterns, such as unusually wet or dry years, which could affect how well the models perform in practice.

The third issue also links back to computational resources. Deep learning models like LSTMs can often be improved by using larger datasets, additional layers, and more advanced hyperparameter tuning. These options were not feasible in the current environment, meaning the study could not fully explore what more advanced architectures might achieve (Goodfellow, Bengio and Courville, 2016).

A fourth limitation comes from the balancing technique used. SMOTE was effective in improving model performance, but oversampling methods can sometimes create synthetic patterns that do not perfectly mirror real-world variability. This is especially important in meteorology, where capturing the characteristics of rare but high-impact events, such as extreme rainfall, is crucial (Chawla et al., 2002; Fernández et al., 2018).

Finally, the study relied on the IMERG dataset alone. IMERG is widely recognised and validated, but like all satellite products, it carries some uncertainty, particularly in regions with fewer ground-

based measurements available for calibration. This means that part of the error observed may reflect limitations of the dataset itself rather than the models (Huffman et al., 2019).

Overall, these limitations suggest that while the results are promising, they should be seen as a first step rather than a final solution. Expanding future research to include additional climate variables, longer datasets, and greater computational capacity would help build a complete and more reliable picture of rainfall forecasting in tropical Africa.

6.5 Future Work

The limitations identified in this study also open several promising directions for future research. Addressing these areas would not only improve the accuracy of machine learning models but also make them more practical for real-world rainfall forecasting in tropical Africa.

One important step would be to include additional climate variables rather than relying solely on precipitation. Rainfall in the tropics is shaped by a range of factors such as temperature, humidity, and wind, which play a major role in driving convective storms (Rasheeda Satheesh et al., 2023). By incorporating these variables, future studies could help deep learning approaches like LSTMs capture more of the complex atmospheric processes that influence rainfall patterns.

A second direction is to make use of longer datasets. This study focused on the years 2020-2024 for practical reasons, but using the full IMERG record from 1998-2024 would give models exposure to a much broader range of conditions, including very wet and very dry years. This additional variability could improve generalisation and ensure that predictions remain robust when faced with unusual climate events (Vogel et al., 2018).

There is also scope to explore more advanced model designs. Combining convolutional neural networks (CNNs) with LSTMs, for example, would allow models to learn from both spatial and temporal features of rainfall data. Ensemble methods, which merge the strengths of different models, may also provide more reliable and less biased results compared with single-model approaches (Shi et al., 2015).

Another avenue for improvement is computational capacity. With access to high-performance computing, it would be possible to carry out more extensive hyperparameter tuning, train deeper networks, and run larger-scale experiments. These steps are often essential for getting the best out of modern deep learning models (Goodfellow, Bengio and Courville, 2016).

Finally, future studies should pay closer attention to data quality. Although IMERG is a well-established dataset, combining it with ground-based rainfall measurements where they exist could help improve calibration and reduce satellite-related biases (Huffman et al., 2019). This kind of hybrid approach would be particularly valuable in tropical Africa, where rainfall monitoring networks are limited but accurate forecasts are vital for agriculture, disaster risk management, and planning.

In short, future work should focus on expanding the range of input data, extending the training period, experimenting with more advanced model architectures, and improving validation through multiple data sources. Taken together, these steps would help move rainfall prediction from being largely experimental toward becoming a practical tool that supports decision-making in regions most affected by climate variability.

6.6 Final Reflection

In conclusion, this study demonstrates that observation-to-observation machine learning techniques offer a promising approach to predicting tropical precipitation. The results indicate that simpler models, when carefully balanced and engineered, can perform strongly and, in some cases, surpass more complex deep learning methods. These findings reinforce the potential of machine learning to contribute to rainfall forecasting in tropical Africa, while also underscoring the need for expanded datasets and advanced architectures to fully capture the complexity of convective precipitation.

Chapter 7: Conclusion

With an emphasis on tropical Africa, the goal of this dissertation was to investigate whether machine learning could provide reliable short-term rainfall forecasts in the tropics, with a focus on tropical Africa. The motivation stemmed from the persistent challenges of forecasting convective rainfall, which is often poorly represented in traditional numerical weather prediction (NWP) models (Vogel et al., 2018; Rasheeda Satheesh et al., 2023). A deep learning Long Short-Term Memory (LSTM) network and several baseline machine learning models were implemented and assessed using the IMERG dataset for the years 2020–2024.

The findings revealed that the LSTM was consistently outperformed by simpler baseline models, especially Logistic Regression and Decision Trees with SMOTE balancing. With an accuracy rate of 89% and results that were balanced across rainfall categories, logistic regression performed the best overall. The limitations of using precipitation alone as the input variable were highlighted by the LSTM, which plateaued early in training and only achieved 66.5% accuracy. These results show that in environments with limited resources, well-designed but computationally efficient models can offer useful predictive ability.

In practical terms, the study emphasises the value of data balancing strategies in addressing class imbalance and the potential of utilising recent rainfall data to generate insightful short-term projections. Agriculture, disaster risk reduction, and water management in tropical regions can all benefit from even small advancements in forecasting uncommon but extreme events, like heavy rainfall.

At the same time, several limitations were identified. These included limitations on computational resources, the use of only one input variable, and the use of only five years' worth of training data. These restrictions hindered a more thorough investigation of sophisticated model architectures and limited the LSTM's performance. Future research could fully utilise deep learning techniques for rainfall forecasting by filling in these gaps with multivariable input, longer training datasets, and hybrid models.

List of references

1. Bauer, P., Thorpe, A. and Brunet, G., 2015. The quiet revolution of numerical weather prediction. Nature, 525(7567), pp.47-55. doi: 10.1038/nature14956
2. Breiman, L., 2001. Random forests. Machine learning, 45(1), pp.5-32. doi: 10.1023/A:1010933404324
3. Breiman, L., Friedman, J., Olshen, R.A. and Stone, C.J., 2017. Classification and regression trees. Chapman and Hall/CRC.
4. Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, pp.321-357. doi: 10.1613/jair.953
5. Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), pp.21-27. doi: 10.1109/TIT.1967.1053964
6. Cox, D.R., 1958. The regression analysis of binary sequences. Journal of the Royal Statistical Society Series B: Statistical Methodology, 20(2), pp.215-232. https://www.jstor.org/stable/2983890
7. Espeholt, L., Agrawal, S., Sønderby, C., Kumar, M., Heek, J., Bromberg, C., Gazen, C., Carver, R., Andrychowicz, M., Hickey, J. and Bell, A., 2022. Deep learning for twelve hour precipitation forecasts. Nature communications, 13(1), p.5145. doi: 10.1038/s41467-022-32595-7
8. Fernández, A., Garcia, S., Herrera, F. and Chawla, N.V., 2018. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. Journal of artificial intelligence research, 61, pp.863-905. doi: 10.1613/jair.1.11192
9. Fink, A.H. and Reiner, A., 2003. Spatiotemporal variability of the relation between African easterly waves and West African squall lines in 1998 and 1999. Journal of Geophysical Research: Atmospheres, 108(D11). doi: 10.1029/2002JD002816
10. Folk, M., Heber, G., Koziol, Q., Pourmal, E. and Robinson, D., 2011, March. An overview of the HDF5 technology suite and its applications. In Proceedings of the EDBT/ICDT 2011 workshop on array databases (pp. 36-47). doi: 10.1145/1966895.1966900
11. Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., 2016. Deep learning (Vol. 1, No. 2). Cambridge: MIT press.
12. Harris, C.R., Millman, K.J., Van Der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J. and Kern, R., 2020. Array programming with NumPy. nature, 585(7825), pp.357-362. doi: 10.1038/s41586-020-2649-2
13. Hastie, T., Tibshirani, R., Friedman, J. and Franklin, J., 2005. The elements of statistical learning: data mining, inference and prediction. The Mathematical Intelligencer, 27(2), pp.83-85. doi: 10.1007/978-0-387-84858-7
14. Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780. doi: 10.1162/neco.1997.9.8.1735
15. Hosmer Jr, D.W., Lemeshow, S. and Sturdivant, R.X., 2013. Applied logistic regression. John Wiley & Sons. doi: 10.1002/9781118548387
16. Hou, A.Y., Kakar, R.K., Neeck, S., Azarbarzin, A.A., Kummerow, C.D., Kojima, M., Oki, R., Nakamura, K. and Iguchi, T., 2014. The global precipitation measurement mission. Bulletin of the American meteorological Society, 95(5), pp.701-722. doi: 10.1175/BAMS-D-13-00164.1

17. Huffman, G.J., Bolvin, D.T., Braithwaite, D., Hsu, K.L., Joyce, R.J., Kidd, C., Nelkin, E.J., Sorooshian, S., Stocker, E.F., Tan, J. and Wolff, D.B., 2020. Integrated multi-satellite retrievals for the global precipitation measurement (GPM) mission (IMERG). In Satellite precipitation measurement: Volume 1 (pp. 343-353). Cham: Springer International Publishing. doi: 10.1175/JHM-D-19-0287.1
18. Huffman, G.J., 2019. GPM IMERG final precipitation L3 half hourly 0.1 degree× 0.1 degree V06. (No Title). doi: 10.5067/GPM/IMERG/3B-HH/06
19. Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(03), pp.90-95. doi: 10.1109/MCSE.2007.55
20. Hyndman, R.J. and Athanasopoulos, G., 2018. Forecasting: principles and practice. OTexts. https://otexts.com/fpp2/
21. Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. https://arxiv.org/abs/1412.6980
22. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S. and Ivanov, P., 2016. Jupyter Notebooks–a publishing format for reproducible computational workflows. In Positioning and power in academic publishing: Players, agents and agendas (pp. 87-90). IOS press. doi: 10.3233/978-1-61499-649-1-87
23. Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W. and Merose, A., 2023. Learning skillful medium-range global weather forecasting. Science, 382(6677), pp.1416-1421. doi: 10.1126/science.adi2336
24. Lavaysse, C., Diedhiou, A., Laurent, H. and Lebel, T., 2006. African Easterly Waves and convective activity in wet and dry sequences of the West African Monsoon. Climate Dynamics, 27(2), pp.319-332. doi: 10.1007/s00382-006-0137-5
25. Liaw, A. and Wiener, M., 2002. Classification and regression by randomForest. R news, 2(3), pp.18-22.
26. Little, R.J. and Rubin, D.B., 2019. Statistical analysis with missing data. John Wiley & Sons. doi: 10.1002/9781119482260
27. Maranan, M., Fink, A.H. and Knippertz, P., 2018. Rainfall types over southern West Africa: Objective identification, climatology and synoptic environment. Quarterly Journal of the Royal Meteorological Society, 144(714), pp.1628-1648. doi: 10.1002/qj.3345
28. Marsham, J.H., Knippertz, P., Dixon, N.S., Parker, D.J. and Lister, G.M., 2011. The importance of the representation of deep convection for modeled dust-generating winds over West Africa during summer. Geophysical Research Letters, 38(16). doi: 10.1029/2011GL048207
29. Mathon, V., Laurent, H. and Lebel, T., 2002. Mesoscale convective system rainfall in the Sahel. Journal of applied meteorology, 41(11), pp.1081-1092. doi: 10.1175/1520-0450(2002)041<1081:MCSRIT>2.0.CO;2
30. McKinney, W., 2010. Data structures for statistical computing in Python. scipy, 445(1), pp.51-56. doi: 10.25080/Majora-92bf1922-00a
31. Nesbitt, S.W., Zipser, E.J. and Cecil, D.J., 2000. A census of precipitation features in the tropics using TRMM: Radar, ice scattering, and lightning observations. Journal of climate, 13(23), pp.4087-4106. doi: 10.1175/JCLI3738.1
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. Pytorch: An imperative style, high-

performance deep learning library. Advances in neural information processing systems, 32. https://arxiv.org/abs/1912.01703
33. Peterson, L.E., 2009. K-nearest neighbor. Scholarpedia, 4(2), p.1883. doi: 10.4249/scholarpedia.1883
34. Rasheeda Satheesh, A., Knippertz, P., Fink, A.H., Walz, E. and Gneiting, T. (2023). Sources of predictability of synoptic-scale rainfall during the West African summer monsoon. Quarterly Journal of the Royal Meteorological Society, 149(757), pp.3721–3737. doi: https://doi.org/10.1002/qj.4581.
35. Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S. and Prudden, R., 2021. Skilful precipitation nowcasting using deep generative models of radar. Nature, 597(7878), pp.672-677. doi: 10.1038/s41586-021-03854-z
36. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K. and Woo, W.C., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. Advances in neural information processing systems, 28. https://arxiv.org/abs/1506.04214
37. Shmueli, G. and Polak, J., 2024. Practical time series forecasting with r: A hands-on guide. Axelrod schnall publishers.
38. Sokolova, M. and Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Information processing & management, 45(4), pp.427-437. doi: 10.1016/j.ipm.2009.03.002
39. Tukey, J.W., 1977. Exploratory data analysis (Vol. 2, pp. 131-160). Reading, MA: Addison-wesley.
40. Vogel, P., Knippertz, P., Fink, A.H., Schlueter, A. and Gneiting, T., 2018. Skill of global raw and postprocessed ensemble predictions of rainfall over northern tropical Africa. Weather and Forecasting, 33(2), pp.369-388. doi: 10.1175/WAF-D-20-0161.1
41. Walz, E.-M., Knippertz, P., Fink, A.H., Gregor Köhler and Gneiting, T. (2024). Physics-Based vs Data-Driven 24-Hour Probabilistic Forecasts of Precipitation for Northern Tropical Africa. Monthly Weather Review, [online] 152(9), pp.2011–2031. doi: https://doi.org/10.1175/mwr-d-24-0005.1.
42. Waskom, M.L., 2021. Seaborn: statistical data visualization. Journal of open source software, 6(60), p.3021. doi: 10.21105/joss.03021
43. Wilks, D.S., 2011. Statistical methods in the atmospheric sciences (Vol. 100). Academic press.