CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter will elaborate more on the methodology used in this research which includes the research design framework, steps and techniques used for the research. The overall research workflow consists of four phases which will be discussed further in this chapter. Each step of the framework will be elaborated in detail on how it will be implemented to this research including the techniques that will be used.

3.2 Research Workflow

The research methodology framework has four main phases which are Literature Review, Problem Definition, Experiment & Evaluation and Result Documentation. The framework starts with Phase 1 which is the literature review of the study, it related to the process of gathering information related to microclimate monitoring and prediction and machine learning techniques as shown in Figure 3.1. The framework then continues with the second phase which is the problem definition phase of the thesis, the problem on microclimate monitoring & prediction. The next phase is the experiment and evaluation phase which is how to conduct the research and how to evaluate data of the results acquired. The last phase is the result documentation phase where end results will be reported.

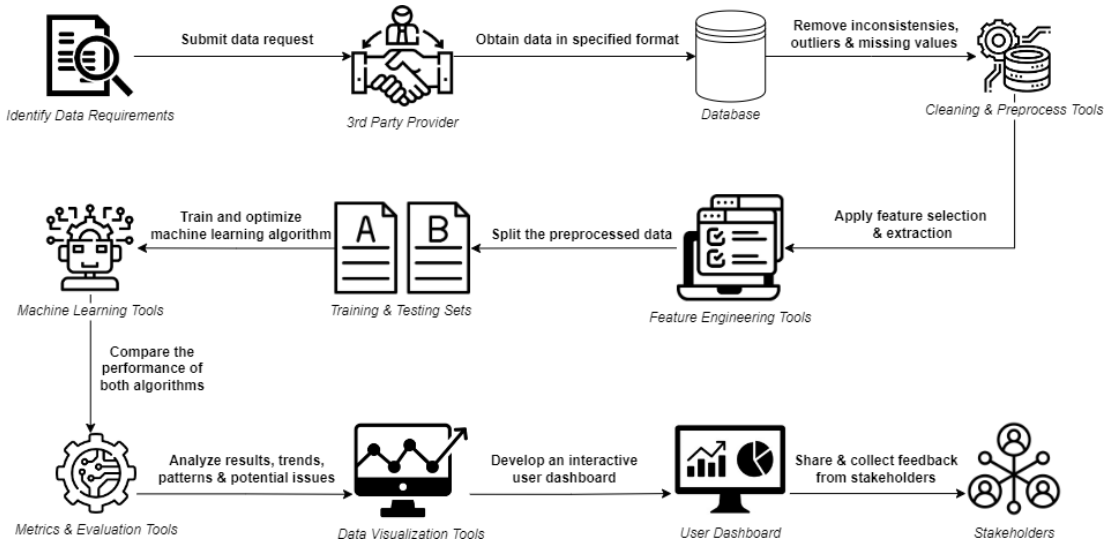


Figure 1: Research Workflow

3.2.1 Phase 1: Literature Review

In the first phase, a comprehensive literature review is conducted to gather relevant information on preserving cultural heritage sites through microclimate monitoring and prediction using Random Forest and XGBoost algorithms. Various scholarly sources, including journals, articles, and theses, are explored to understand the current state of research in this field. The literature review delves into topics such as data collection methods, pre-processing techniques, and the application of machine learning models. By examining existing studies, this phase helps identify gaps, challenges, and potential solutions for effectively monitoring and predicting microclimate conditions at heritage sites. The insights gained from the literature review form the foundation for the subsequent phases and guide the research towards developing a robust methodology.

3.2.2 Phase 2: Data Collection and Preprocessing

In the second phase, the focus shifts to obtaining microclimate data from the Copernicus Climate Change Service (C3S) for a specific heritage site in Johor Bahru. This data, encompassing temperature, relative humidity, precipitation, and wind speed

measurements, serves as the basis for subsequent analysis and modelling. To ensure data quality, a rigorous pre-processing stage is undertaken, which involves cleaning the raw data and addressing any missing values or outliers. Techniques such as interpolation, statistical analysis, and data imputation are employed to enhance the integrity and accuracy of the collected data. Additionally, feature engineering techniques are applied to extract meaningful features from the raw data, enabling the capturing of temporal dependencies and relationships between variables. This phase prepares the dataset for further analysis and model development in the subsequent phases.

3.2.3 Phase 3: Machine Learning Model Development

The third phase revolves around the development of machine learning models for microclimate monitoring and prediction. The pre-processed data is divided into training and testing sets, with the training set used to train and optimize the Random Forest and XGBoost algorithms. Various parameters and hyperparameters are fine-tuned using techniques like grid search and cross-validation to achieve optimal model performance. The trained models are then evaluated using appropriate assessment metrics, such as mean absolute error and mean squared error, to assess their predictive capabilities. This evaluation process helps determine the effectiveness and performance of the Random Forest and XGBoost algorithms in accurately predicting microclimate patterns for the designated heritage site. The models' performance and generalizability are crucial factors in ensuring the reliability and usefulness of the developed models for microclimate monitoring and prediction.

3.2.4 Phase 4: Dashboard Development

In the fourth phase, a user-friendly dashboard is designed and developed to visualize and present the microclimate data in a comprehensible manner. The dashboard provides real-time insights into the microclimate conditions of the heritage site, displaying key metrics such as temperature, humidity, and wind speed. The trained machine learning models are integrated into the dashboard to provide

recommendations for preventive maintenance actions based on the analysed data. Additionally, visualization tools and interactive features are incorporated to facilitate a better understanding of trends and patterns in the microclimate data. The dashboard serves as a valuable tool for local authorities and stakeholders involved in the preservation of cultural heritage sites, enabling them to make informed decisions and take proactive measures to mitigate potential issues that may impact the site's condition.

3.3 Justification of Tools, Techniques and Data

The chosen tools for this research include data collection from the Copernicus Climate Change Service (C3S), data pre-processing techniques, machine learning algorithms (Random Forest and XGBoost), and dashboard development. These tools have been selected based on their suitability for addressing the research objectives and providing valuable insights for preventive maintenance strategies at the designated heritage site in Johor Bahru.

Microclimate data from Copernicus is essential for understanding the environmental conditions at the heritage site. Temperature, precipitation, humidity, wind speed are crucial parameters that can affect the preservation of heritage structures. By obtaining this data, we can assess the impact of these environmental factors on the site's structural integrity and identify potential maintenance issues.

The research utilizes machine learning techniques to analyse the collected microclimate data and develop predictive models for preventive maintenance. Random Forest and XGBoost algorithms are chosen due to their proven effectiveness in handling complex relationships between variables and handling both regression and classification tasks. These algorithms can capture temporal dependencies in the data and provide accurate predictions for future microclimate conditions.

The preservation of heritage sites is of paramount importance to maintain cultural identity and historical significance. By utilizing advanced data analysis

techniques, this research aims to provide insights into the impact of microclimate conditions on the designated heritage site in Johor Bahru. The findings can help authorities develop targeted preventive maintenance strategies to ensure the site's long-term preservation.

By collecting and analysing microclimate data, this research enables data-driven decision making for preventive maintenance. Traditional approaches may not consider the dynamic nature of microclimate conditions and their impact on heritage sites. The use of machine learning algorithms allows for a more comprehensive understanding of the relationships between environmental factors and potential maintenance issues, leading to more informed decision making.

Implementing preventive maintenance strategies based on predictive models can result in cost savings and increased efficiency. By identifying trends, patterns, and potential issues, maintenance activities can be prioritized, scheduled, and targeted accordingly. This approach minimizes reactive maintenance efforts, reduces costs associated with emergency repairs, and optimizes resource allocation.

The development of a user-friendly dashboard integrating real-time microclimate data and machine learning models provides a powerful tool for monitoring and maintenance planning. The visualization tools within the dashboard help users understand trends and patterns in the data, facilitating proactive decision making. This allows local authorities to respond promptly to changing microclimate conditions and potential threats to the heritage site.

The research encourages collaboration between local authorities, heritage site management teams, and relevant stakeholders. By involving these parties in the evaluation and testing phases, their feedback can be gathered to refine the system and ensure its usability and effectiveness. Engaging stakeholders throughout the research process increases their ownership and facilitates the adoption of preventive maintenance strategies.

3.4 Chapter Summary

This chapter summarized the four phases of the research study. The literature review phase involved a comprehensive review of relevant literature, providing a solid knowledge base for the subsequent phases. The data collection and pre-processing phase focused on obtaining and cleaning microclimate data, while the machine learning model development phase involved training and evaluating Random Forest and XGBoost algorithms. The final phase focused on developing a user-friendly dashboard that visualizes the microclimate data and provides maintenance recommendations.

CHAPTER 4

RESEARCH DESIGN AND IMPLEMENTATION

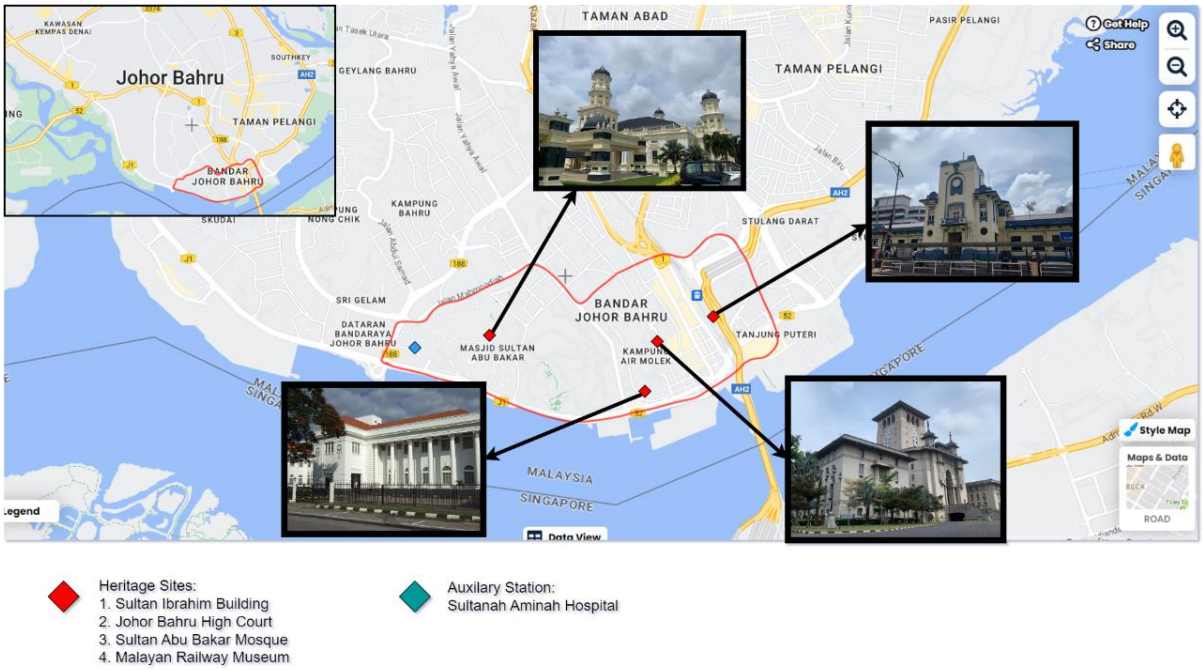
4.1 Introduction

This chapter discusses in depth the research design and implementation of the research methodology described in the previous chapter. The proposed solution will be broken down into several steps, including the data collection, pre-processing, feature extraction, training and testing and machine learning models development.

4.2 Proposed Solution

The proposed solution encompasses several steps to address the objectives outlined in the research scope. Initially, microclimate data will be acquired from the Copernicus Climate Change Service (C3S), focusing on parameters like temperature, humidity, precipitation, and wind speed. Subsequently, the collected data will undergo preprocessing to ensure quality and reliability, including handling missing values and outliers. Relevant features will be selected for model training, emphasizing factors such as temperature variations, humidity levels, and wind patterns. Two machine learning algorithms, Random Forest and XGBoost, will be implemented and compared for their performance in microclimate monitoring and prediction. Additionally, a dashboard will be designed and developed using data visualization tools like Matplotlib to display real-time microclimate data. Finally, the effectiveness of the developed algorithms and dashboard will be evaluated based on their ability to assist local authorities in planning preventive maintenance actions for the heritage site, considering metrics such as prediction accuracy and user feedback. Through these steps, the proposed solution aims to provide a comprehensive framework for enhancing microclimate monitoring and management for the preservation of the Johor Bahru High Court, Sultan Ibrahim Building, Sultan Abu Bakar Mosque and Malayan Railway Museum.

4.3 Research Area Zone Mapping



In the context of this research, a research area zone mapping was established to focus on the preservation of cultural heritage sites in Johor Bahru, Malaysia. The selected heritage sites for this study are:

Heritage Sites	Description
Sultan Ibrahim Building	This building holds historical and architectural significance as the state secretariat building of Johor. It represents the rich cultural heritage of the region.
Johor Bahru High Court	The Johor Bahru High Court is a notable judicial institution that plays a crucial role in the administration of justice. It possesses historical and legal importance.
Sultan Abu Bakar Mosque	Known for its exquisite Moorish-inspired architecture, the Sultan Abu Bakar Mosque is a revered religious site. It serves as a symbol of faith and cultural identity.

Malayan Railway Museum	The Malayan Railway Museum, located in Johor Bahru, showcases the historical development and significance of the railway system in Malaysia. It offers insights into the country's transportation heritage.

4.4 Experiment Design

4.4.1 Microclimate Data Collection Process

In this research study, the microclimate data was obtained from the ECMWF Reanalysis v5 (ERA5) dataset, provided by the Copernicus Climate Change Service (C3S) at the European Centre for Medium-Range Weather Forecasts (ECMWF). ERA5 is the fifth generation of ECMWF's atmospheric reanalysis, offering a comprehensive global climate analysis spanning from January 1940 to the present day.

The data was available in the NetCDF format, a widely used self-describing, machine-independent data format for array-oriented scientific data. To extract and process the data, we utilized Spyder, a powerful Integrated Development Environment (IDE) for Python, which facilitated efficient data handling and analysis.

For this research, we collected microclimate data spanning a substantial time range, from 1940 to 2023. This extended historical period allowed us to gather a significant amount of data, enabling more reliable and robust predictions related to the microclimate conditions at the designated heritage site.

Historical data plays a crucial role in understanding and predicting microclimate patterns in the context of heritage sites. By analyzing long-term data trends, we can gain valuable insights into the site's microclimate dynamics, including temperature, humidity, wind patterns, and precipitation levels. These insights are essential for developing effective strategies to preserve and protect the heritage site from potential environmental impacts.

Moreover, the comprehensive nature of the ERA5 dataset, encompassing various climate variables and spanning over eight decades, provides a holistic perspective on the microclimate conditions. This holistic view is invaluable for making accurate predictions and informing decision-making processes related to the conservation and management of the heritage site.

By leveraging the power of the ERA5 dataset and the capabilities of Python's data analysis tools, this research study aims to develop a robust predictive model and a user-friendly dashboard prototype. The ultimate goal is to equip stakeholders with valuable insights and tools to proactively address microclimate-related challenges and ensure the long-term preservation of the heritage site.

4.4.2 Data Collection and Extraction

```
import cdsapi
c = cdsapi.Client()
c.retrieve( 'reanalysis-era5-pressure-levels-monthly-means',
           { 'product_type': 'monthly_averaged_reanalysis',
            'variable': 'total_precipitation',
            'year': [ '1940', '1941', '1942', '1943', '1944', '1945', '1946',
                      '1947', '1948', '1949', '1950', '1951', '1952', '1953',
                      '1954', '1955', '1956', '1957', '1958', '1959', '1960',
                      '1961', '1962', '1963', '1964', '1965', '1966', '1967',
                      '1968', '1969', '1970', '1971', '1972', '1973', '1974',
                      '1975', '1976', '1977', '1978', '1979', '1980', '1981',
                      '1982', '1983', '1984', '1985', '1986', '1987', '1988',
                      '1989', '1990', '1991', '1992', '1993', '1994', '1995',
                      '1996', '1997', '1998', '1999', '2000', '2001', '2002',
                      '2003', '2004', '2005', '2006', '2007', '2008', '2009',
                      '2010', '2011', '2012', '2013', '2014', '2015', '2016',
                      '2017', '2018', '2019', '2020', '2021', '2022', '2023', ]
            'month': '12',
            'time': '00:00',
            'area': [ 10, 90, -10, 130, ],
            'format': 'netcdf', }, 'Monthly-Precip-Dec-1940-2023.nc')
```

Figure 2: Data Retrieval from CDSAPI

The script in Figure 2 above utilizes the Climate Data Store (CDS) Application Programming Interface (API) to retrieve monthly averaged temperature data from the ERA5 dataset, specifically focusing on pressure level 1000 hPa. It spans from July 1940 to July 2023 and covers a defined geographical area. The data is requested in NetCDF format, which is a

common format for storing multidimensional scientific data. The geographical area of interest is specified by the latitude and longitude coordinates [10, 90, -10, 130], representing the region from 10°S to 90°N latitude and from 130°E to 10°W longitude.

The script iterates over each year from 1940 to 2023, retrieving monthly temperature data for the month of July at the specified time (00:00 UTC). This process is essential for obtaining historical temperature records, which are crucial for understanding climate trends and patterns over time. The retrieved data is then stored in a NetCDF file named "Monthly-TEMP-Jul-1940-2023.nc" for further analysis and visualization.

```
13  data_raw='../Data-RAW-RAIN/'
14
15  # files_month=['Jun']
16  files_month=['Jan','Feb','Mac','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov'
17
18  for m in files_month:
19      filename='Monthly-Precip-'+m+'-1940-2023.nc'
20      f=nc.Dataset(data_raw+filename)
21      v=f.variables; keys=f.variables.keys(); data={}
22
23      for i in keys:
24          data[i]=np.squeeze(v[i][:])
25          print(i)
26
27      lat=data['latitude']
28      lon=data['longitude']
29      rain=data['tp'][:]
30
31      times = f.variables['time'][:]
32      units=f.variables['time'].units
33      ptime = num2date (times[:], units, calendar='gregorian')
34      print ('Available Time Observation to Plot : (index,pressure) ')
```

Figure 3: Data Extraction and Processing

The process followed by processing and extracting the the raw data files as shown in Figure 3. The script loops through a list of month names (e.g., 'Jan', 'Feb', 'Mar', etc.) and opens each corresponding NetCDF file containing temperature data for that month. It reads the latitude, longitude, and temperature variables from the file, as well as the time dimension and its associated units. For each file, the script converts the time values from the NetCDF file to Python datetime objects using the num2date function from the netCDF4 library. It then prints the available time observations, allowing the user to select a specific time index for further processing.

After selecting a time index, the script extracts the corresponding temperature data and converts it from Kelvin to Celsius. It then creates a grid of latitude and longitude values using np.meshgrid and combines the temperature data with the grid coordinates into a single 2D array. The script then creates a folder structure based on the year and saves the extracted temperature data, along with the corresponding latitude and longitude coordinates, into a text file named Extract-YYYYMM.dat. The file is saved in a subdirectory named after the year, within a directory called '20-Ekstrak' located in the current working directory. Overall, this script is designed to extract and process temperature data from a set of NetCDF files, convert the data to a more accessible format, and save it to text files organized by year and month. This process can be useful for further analysis, visualization, or integration with other data sources.

```
93      # Define colormap and contour levels
94      cmap = cm.s3pcpn_l
95      clevprecip = np.arange(0, 1200, 50)
96      norm1 = mpl.colors.BoundaryNorm(clevprecip, cmap.N)
97
98      # Create the contour plot
99      cf = m.contourf(X, Y, Z, clevprecip, cmap=cmap, norm=norm1, latlon=True)
100     cbar = m.colorbar(cf, location='bottom', pad="12%")
101     cbar.ax.tick_params(labelsize=10)
102     cbar.set_label('Monthly Accumulated Rainfall (mm/month)', fontsize=10)
103
104     # Add title to the plot
105     title1 = 'ERA5- Monthly Accumulated Rainfall for ' + nama_file
106     plt.title(title1)
107
108     # Save the plot as a PNG file
109     plt.savefig(path3 + nama_file + '.png', dpi=500, bbox_inches='tight')
```

Figure 4: Plotting Monthly Average Temperature Data

The script in Figure 4 is designed to visualize monthly average temperature data for a specific region, with Malaysia as an example. It starts by importing necessary libraries for data processing and plotting. After setting up the directory structure, it reads temperature data files, processes them to extract latitude, longitude, and temperature values, and creates a contour plot of the temperature data on a map. The map is centred on the region of interest which in this case, Malaysia and includes country and state boundaries, coastlines, parallels, and meridians for reference as shown in Figure 5 below.

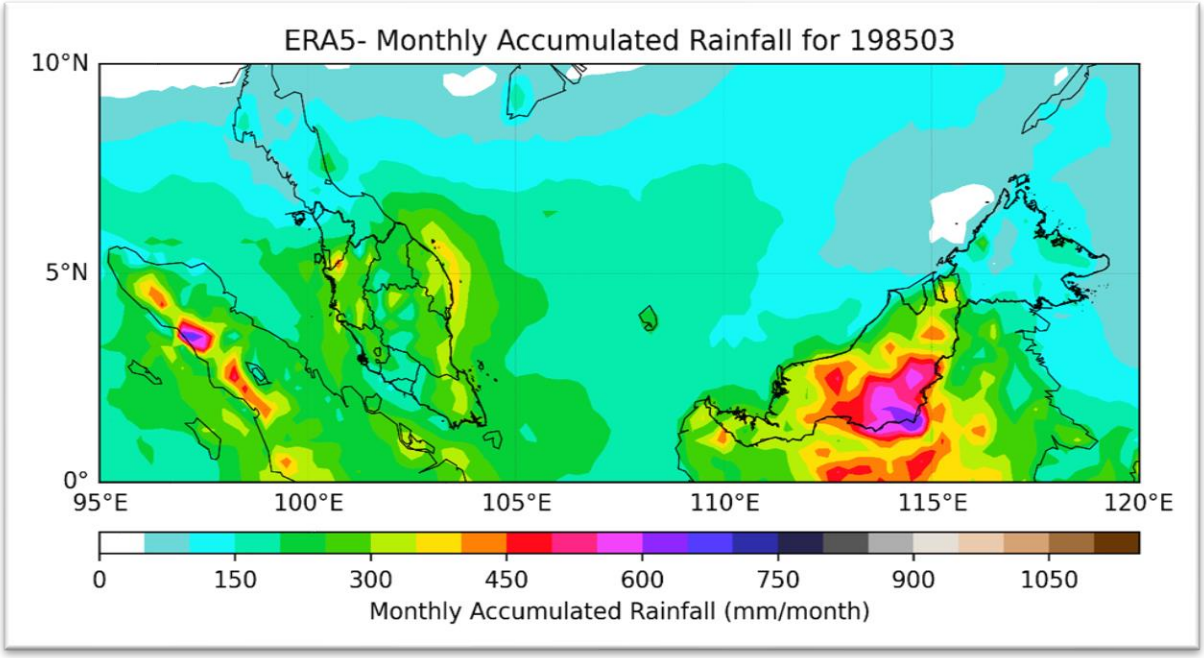


Figure 5: Monthly Accumulated Rainfall Sample Output

Additionally, it overlays administrative boundaries of Malaysian states obtained from a shapefile. The script utilizes a colormap to represent temperature variations and adds a colorbar for reference. Finally, it saves the generated plot images, each corresponding to a specific data file, for further analysis or visualization. Overall, this script facilitates the visualization of monthly temperature patterns for a chosen region, aiding in understanding climate variations over time.

```
63              # Extract latitude, longitude, and rainfall rate
64              lat = data[:, 0]
65              lon = data[:, 1]
66              rain_rate = data[:, 2]
67
68              # Define the target latitude and longitude (location to select data for)
69              in_lats1 = [q]
70              in_lons1 = [r]
71
72              # Find the closest data point to the target location
73              ind = []
74              for i in range(1):
75                  dist = (lat - in_lats1[i])**2 + (lon - in_lons1[i])**2
76                  ind.append(np.where(dist == np.min(dist))[0][0])
77
78                  lat2 = lat[ind]
79                  lon2 = lon[ind]
80                  rain_rate2 = rain_rate[ind]
81
82              # Combine the date, target location, and selected data into an array
83              data3 = [np.array([dates]), in_lats1, in_lons1, lat2, lon2, rain_rate2]
84              data3 = np.transpose(data3)
```

Figure 6: Extracting Location-Specific Data from Data Files

The script described in Figure 6 is designed to extract location-specific meteorological data from a collection of data files. It allows users to specify one or more locations of interest, along with optional criteria such as specific years or a range of years. After importing necessary libraries, the script prompts users to input the location(s) they are interested in, along with any desired years. It then creates directories for storing both the original data files and the extracted location-specific data.

For each combination of year and location, the script searches for matching data files and reads the data, typically containing latitude, longitude, and meteorological parameters like temperature or rainfall rates. It calculates the distance between each data point and the specified location to find the closest data point, from which it extracts relevant values. The extracted location-specific data is then saved to new files, named to indicate the location, year, and time period. This process streamlines the extraction of meteorological data tailored to specific locations, eliminating the need for manual data searching and filtering. This functionality is valuable for researchers, meteorologists, or anyone interested in analysing weather patterns or environmental conditions at specific locations.

```
        # Extract rainfall data from each month's data
        rain1 = data1[5]
        rain2 = data2[5]
        rain3 = data3[5]
        rain4 = data4[5]
        rain5 = data5[5]
        rain6 = data6[5]
        rain7 = data7[5]
        rain8 = data8[5]
        rain9 = data9[5]
        rain10 = data10[5]
        rain11 = data11[5]
        rain12 = data12[5]

        # Combine the year and monthly rainfall data into a list
        dataA = [year, rain1, rain2, rain3, rain4, rain5, rain6, rain7, rain8, rain9, rain10

        # Append the combined data to the accum list
        accum.append(dataA)
```

Figure 7: Merging and Consolidating Location-Specific Microclimate Data

The script described in Figure 7 is designed to merge and consolidate location-specific microclimate data from multiple files into a single file for each location of interest. This consolidation process helps in organizing and simplifying the data for easier analysis or visualization. After importing necessary libraries and defining the locations and years of interest, the script creates directories for storing both the extracted location-specific data files and the merged data files.

For each combination of year and location, the script reads the monthly data files containing information such as date, latitude, longitude, and meteorological parameters like temperature or rainfall rates. It then extracts the relevant temperature or rainfall rate values for each month and organizes them into a list, along with the corresponding year. These lists are then appended to a larger list, accumulating the data for all years and locations. Once all specified years and locations have been processed, the accumulated data list is saved to a new file in the '80-Merge-Data' directory. This file contains the merged and consolidated data for the specific location, with each row representing a year and the corresponding temperature or rainfall rate values for each month.

By running this script, users can efficiently merge and consolidate location-specific meteorological data from multiple files into a single file for each location of interest. This consolidated data can then be further analysed to identify trends, patterns, or anomalies in the meteorological data over time, or used for creating visualizations or reports.

4.4.3 Splitting of Data into Training and Testing Sets

In the below code snippet, the train_test_split function from scikit-learn is used to split the pre-processed data (X_new) and the corresponding target variable (y) into training and testing sets. The test_size parameter is set to 0.2, indicating that 20% of the data will be used for testing, while the remaining 80% will be used for training. The random_state parameter is set to 42 to ensure reproducibility of the split. Then, it will proceed with training and evaluating machine learning models using the X_train, y_train for training, and X_test, y_test for testing.

```
from sklearn.model_selection import train_test_split

# Split the preprocessed data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2, random_state=42)
```

Figure 8: Split the pre-processed data into training and testing sets with an 80-20 ratio

4.4.4 Training and Evaluation of Machine Learning Models

The next step after splitting the data into training and testing sets is to train and evaluate the machine learning models. The figure shows the code for training and evaluating the Random Forest and XGBoost algorithms: In this code snippet, the Random Forest and XGBoost models are initialized and trained using the training data (X_train and y_train). Then, predictions are made on the testing data (X_test) using the trained models. The mean absolute error (MAE) and mean squared error (MSE) are calculated to evaluate the performance of both models.

```
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Initialize and train the Random Forest model
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)

# Make predictions on the testing set using the trained Random Forest model
rf_predictions = rf_model.predict(X_test)

# Calculate evaluation metrics for the Random Forest model
rf_mae = mean_absolute_error(y_test, rf_predictions)
rf_mse = mean_squared_error(y_test, rf_predictions)

# Initialize and train the XGBoost model
xgb_model = XGBRegressor()
xgb_model.fit(X_train, y_train)

# Make predictions on the testing set using the trained XGBoost model
xgb_predictions = xgb_model.predict(X_test)

# Calculate evaluation metrics for the XGBoost model
xgb_mae = mean_absolute_error(y_test, xgb_predictions)
xgb_mse = mean_squared_error(y_test, xgb_predictions)

# Print the evaluation results
print("Random Forest - Mean Absolute Error:", rf_mae)
print("Random Forest - Mean Squared Error:", rf_mse)
print("XGBoost - Mean Absolute Error:", xgb_mae)
print("XGBoost - Mean Squared Error:", xgb_mse)
```

Figure 9: Train and evaluate the Random Forest and XGBoost algorithms

4.5 Parameter and Testing Methods

In order to evaluate the effectiveness of the proposed solution, several parameters are measured during the testing phase. These parameters provide valuable information about the performance and efficacy of the developed models. The parameters to be measured include:

4.5.1 Parameters to be Measured

1. Prediction Accuracy: The accuracy of the predictive models in capturing and predicting the microclimate conditions is measured. This indicates how well the models are able to estimate the actual values of temperature, humidity, wind speed, solar radiation, and rainfall.

2. Mean Absolute Error (MAE): MAE is measured to determine the average absolute difference between the predicted and actual values. It provides insights into the average prediction error across all the microclimate parameters.

3. Mean Squared Error (MSE): MSE is calculated to assess the average squared difference between the predicted and actual values. It measures the overall variance between the predicted and actual values.

4.5.2 Testing Procedure

1. Splitting Data: The pre-processed microclimate data is divided into training and testing sets, typically using an 80:20 split. The training set is used to train the machine learning models, while the testing set is used for evaluating the performance.

2. Model Evaluation: The trained Random Forest and XGBoost models are applied to the testing set to make predictions for the microclimate parameters. The predicted values are then compared with the actual values from the testing set.

3. Calculation of Evaluation Metrics: The evaluation metrics, such as MAE and MSE, are calculated based on the predicted and actual values. These metrics provide quantitative measures of the model's performance.

4. Analysis and Interpretation: The evaluation results are analysed to gain insights into the accuracy and effectiveness of the trained models. The performance of the models is assessed based on the evaluation metrics and compared to determine the superior algorithm for microclimate prediction.

4.6 Chapter Summary

In this chapter, the research experimental design and implementation of the proposed solution for preserving cultural heritage sites through microclimate monitoring and prediction are summarized. The steps for data collection, pre-processing, model development using

Random Forest and XGBoost, evaluation and testing are outlined. The experimental setup and parameter details are provided, along with the evaluation metrics used to assess the performance of the models. The next chapter will present the results and analysis of the experiments, highlighting the insights gained and the recommendations for preserving cultural heritage sites.