



**RAINFALL PREDICTION IN TAUNGGYI, MYANMAR USING ADDITIONAL
HYDROLOGICAL DATA WITH MACHINE LEARNING TECHNIQUES**

SAI NAING LYNN OO

**A thesis submitted in fulfilment of the
requirements for the award of the degree of
MASTER OF SCIENCE IN DATA SCIENCE AND BUSINESS ANALYTICS**

**ASIA PACIFIC UNIVERSITY OF TECHNOLOGY & INNOVATION (APU)
SCHOOL OF COMPUTING AND TECHNOLOGY**

APRIL 2023

ACKNOWLEDGEMENT

To begin with, I would like to express my sincere gratitude to my supervisor, Mr. Dhason Padma Kumar, for his unflagging support and guidance during the course of this capstone project. He provided valuable feedbacks, constructive comments, and excellent expertise along all the way, all of which contributed to the successful completion of this project. I am also grateful for my second marker Dr. Dewi Octaviani for her insights and contributions in shaping the final outcome of my project. Furthermore, I would like to express my deepest gratitude to all the lecturers who shared their invaluable knowledge with us and their experiences so that we could improve the skills required to carry out this project. In the meantime, I extend my sincerest appreciation to meteoblue® for their invaluable contribution of meteorological data to this project. Without their support, this project would not have been possible. My last but not the least, I would like to express my sincere thanks and appreciation to my beloved parents, family members and my classmates especially Mr. Ammar Hezam Ahmed Mufleh from Yemen and Ms. Fathmath Maya from Maldives who encouraged and supported me tremendously in every aspect during my Data Science journey at APU.

ABSTRACT

Rainfall is a natural phenomenon that occurs as a result of complex interactions between factors within the atmosphere that create a dynamic and complicated climate system on earth. There is no doubt that it plays an important role in regulating ecological balance across the globe. Since ancient times, the study of meteorology has been devoted to the prediction of rainfall, which has been a major theme within this discipline. In the modern world, many scientists are using various methods to forecast rainfall in order to obtain reliable predictions, which plays an important role in the development of many sectors. This project seeks to fill the gap by implementing a machine learning-based rainfall prediction model in Taunggyi city using both traditional weather data as well as hydrological data to improve the predictive model's performance. SEMMA methodology has been adopted for this project and Python programming language is used to implement the whole project. Three machine learning models: Random Forest, Support Vector Machine, and Artificial Neural Network are proposed in order to develop a rainfall prediction model with the best accuracy. The models were assessed using the evaluation metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Coefficient of determination (R Square). The Artificial Neural Network model which contains both traditional weather data and hydrological data performed the best in this study, and it was deployed on the Streamlit Cloud. It is also revealed that the inclusion of hydrological data in the model did boost the performance by two times. The project's deliverable is an estimation model based on machine learning for Taunggyi that provides reliable accuracy and the insights from the predictive models.

Keywords: Rainfall Estimation Model, Precipitation Prediction, Rainfall Prediction, Machine Learning Techniques, RF, SVR, ANN, Taunggyi, Hydrological data

TABLE OF CONTENTS

<i>ACKNOWLEDGEMENT</i>	2
<i>ABSTRACT</i>	3
<i>LIST OF TABLES</i>	9
<i>LIST OF FIGURES</i>	10
<i>LIST OF ABBREVIATIONS</i>	15
<i>CHAPTER 1 : INTRODUCTION</i>	17
1.1 Introduction	17
1.2 Research Background	19
1.3 Problem Statement	21
1.4 Research Questions	22
1.5 Aim of the Study	22
1.6 Objectives of the Study	23
1.7 Significance of the Study	23
1.8 Scope of the Study	23
1.9 Structure of the Report	25
<i>CHAPTER 2 : LITERATURE REVIEW</i>	26
2.1 Introduction	26
2.2 Method	26
2.3 Climatology of Taunggyi City	26
2.4 Rainfall Background	27
2.4.1 Rainfall Categories.....	28
2.4.1.1 Conventional rainfall	28
2.4.1.2 Orographic rainfall.....	29
2.4.1.3 Cyclonic or Frontal rainfall.....	29
2.4.2 Measurement of rainfall	30
2.4.2.1 Ordinary Rain Gauge	30
2.4.2.2 Self-recording (Automatic) Rain Gauge.....	31
2.4.3 Rainfall Forecasting Intervals	32
2.4.3.1 Now-casting	32
2.4.3.2 Short-range forecast	32
2.4.3.3 Medium-range forecast	32

2.4.3.4	Long-range (Extended range) forecast.....	32
2.4.4	Types of Rainfall Data	33
2.4.4.1	Measurement data	33
2.4.4.2	Simulation data	33
2.5	Methods of Rainfall Prediction.....	33
2.5.1	Conventional Dynamic Method	34
2.5.1.1	The Weather Research and Forecasting Model (WRF)	34
2.5.1.2	Seasonal Climate Forecasting Model.....	34
2.5.1.3	Global Data Forecasting System (GFS).....	34
2.5.1.4	Challenges of Numerical Weather Prediction Models.....	34
2.5.2	General Data Mining Method	35
2.5.3	Rainfall Prediction using Machine Learning Techniques.....	36
2.5.3.1	Rainfall Classification Models.....	36
2.5.3.2	Rainfall Estimation Models	37
2.5.3.3	Evaluation Metrics	40
2.6	Summary of related work	48
2.7	Implementation method	48
2.8	The application of selected models in different domains	49
2.8.1	Radom Forest Model.....	49
2.8.2	Support Vector Regressor (SVR).....	50
2.8.3	Artificial Neural Network (ANN).....	50
2.9	Summary of literature review.....	51
CHAPTER 3 : RESEARCH METHODOLOGY		52
3.1	Introduction.....	52
3.2	Research Approach and Methodology.....	52
3.2.1	Problem Understanding.....	54
3.2.2	Data Collection (SAMPLE)	55
3.2.2.1	Dataset Description	55
3.2.3	Data Exploration (EXPLORE).....	57
3.2.4	Data Pre-processing (MODIFY).....	57
3.2.4.1	Handling Missing Values.....	57
3.2.4.2	Handling Anomalies	58
3.2.4.3	Feature Engineering	59
3.2.4.4	Feature Scaling.....	60

3.2.4.5	Data Partition	60
3.2.5	Modelling (MODEL)	61
3.2.5.1	Random Forest	61
3.2.5.2	Support Vector Machine	62
3.2.5.3	Artificial Neural Network	63
3.2.6	Model Evaluation and Selection (ASSESS)	64
3.2.6.1	Mean Absolute Error (MAE)	65
3.2.6.2	Root Mean Square Error (RMSE).....	65
3.2.6.3	Determination of Coefficient (R Square)	65
3.2.6.4	Verifying Overfitting or Underfitting	66
3.2.7	Deployment	67
3.3	Summary.....	68
CHAPTER 4 : EXPERIMENTATION.....		69
4.0	Introduction.....	69
4.1	Data Exploration.....	69
4.1.2	Importing libraries and dataset.....	70
4.1.3	Exploring the dataset.....	70
4.1.4	Descriptive Statistics of the variables.....	71
4.1.5	Analysis of variables	72
4.1.5.1	Univariate Analysis	72
4.1.5.2	Bivariate Analysis	80
4.1.6	Anomalies detection	87
4.1.6.1	Exploring Missing Values.....	87
4.1.6.2	Identifying Duplicate Data.....	87
4.1.6.3	Identifying Zero-variance variable.....	88
4.1.6.4	Outlier detection.....	88
4.1.6.4.1	Outlier detection using boxplot.....	88
4.1.6.4.2	Outlier detection using Z-Score Analysis	90
4.1.6.5	Identifying Invalid Data	90
4.1.6.6	Examining deep down into potential outliers	90
4.1.6.6.1	Short-wave radiation Vs rainfall variable	91
4.1.6.6.2	Evapotranspiration Vs rainfall variable	91
4.1.7	Summary of data exploration	92
4.2	Data Pre-processing	92

4.2.1	Imputation for invalid data in wind direction variable	93
4.3	Feature engineering	93
4.3.1	Seasonal variables	93
4.3.2	Previous rainfall variable	94
4.3.3	Wind direction in sine and cosine values	95
4.3.4	Summary of Feature engineering	96
4.3.5	Explanatory Data Analysis (EDA) after Feature engineering	96
4.3.5.1	Seasonal variable Vs rainfall	96
4.3.5.2	Previous rainfall Vs today rainfall	97
4.4	Feature Selection.....	97
4.4.1	Feature selection using Correlation Analysis.....	97
4.4.2	Feature selection using Tree-based Method.....	104
4.4.3	Summary of Feature selection.....	107
4.5	Normalization using Z-score.....	108
4.6	Data partitioning.....	109
4.7	Modelling and Evaluation Metrics	110
4.7.1	Random Forest Model.....	110
4.7.1.1	RF model using rf_1 dataset	111
4.7.1.2	RF model using rf_1_fe dataset	112
4.7.1.3	RF model using rf_2 dataset	114
4.7.1.4	RF model using rf_2_fe dataset	115
4.7.2	Support Vector Regressor	117
4.7.2.1	SVR model using rf_1 dataset	117
4.7.2.2	SVR model using rf_1_fe dataset	119
4.7.2.3	SVR model using rf_2 dataset	121
4.7.2.4	SVR model using rf_2_fe dataset	122
4.7.3	Artificial Neural Network	124
4.7.3.1	ANN model using rf_1 dataset	125
4.7.3.2	ANN model using rf_1_fe dataset	128
4.7.3.3	ANN model using rf_2 dataset	130
4.7.3.4	ANN model using rf_2_fe dataset	132
4.7	Summary of experimentation	134
CHAPTER 5 : RESULT AND DISCUSSION.....		135
5.0	Introduction.....	135

5.1	Evaluation Metrics	135
5.2	Random Forest Model (RF)	135
5.2.1	Random Forest Model with traditional weather data.....	136
5.2.2	Random Forest Model with additional hydrological data.....	137
5.3	Support Vector Regressor (SVR)	139
5.3.1	Support Vector Regressor with traditional weather data	139
5.3.2	Support Vector Regressor with hydrological data	140
5.4	Artificial Neural Network	142
5.4.1	Artificial Neural Network with traditional weather data	143
5.4.2	Artificial Neural Network with traditional weather data	144
5.5	Models Evaluation and Discussion	145
5.6	Models comparison	146
5.6.1	Random Forest Model	146
5.6.2	Support Vector Regressor	147
5.6.3	Artificial Neural Network	147
5.7	Models validation with previous relevant studies	148
5.8	Deployment of the best model	149
5.8.1	Model Testing	152
5.9	Summary of result and discussion	157
CHAPTER 6 : CONCLUSION		158
6.0	Introduction	158
6.1	Addressing Project's Objectives and Research Questions	158
6.1.1	Objectives of the study	158
6.1.1	Research Questions of the study	160
6.2	Discussion and conclusion	163
6.3	Contributions	165
6.4	Limitations	165
6.5	Future work	165
6.6	Summary of conclusion	166
APPENDICES		172
Appendix 1: Research Plan		172
Research Plan		173

LIST OF TABLES

Table 1: The summary table for related work.....	41
Table 2: The metadata or descriptions of the dataset.....	56
Table 3: Features in the dataset after feature engineering	107
Table 4: Hyperparameters for Random Forest Model	110
Table 5: Hyperparameters for Support Vector Regressor.....	117
Table 6: Hyperparameters for Artificial Neural Network model.....	125
Table 7: Comparison of RF models built with traditional data	136
Table 8: Comparison of RF models built with traditional data and feature engineering.....	136
Table 9: Comparison of RF models built with hydrological data.....	137
Table 10: Comparison of RF models built with hydrological data and feature engineering .	137
Table 11: Comparison of RF Tuned models.....	138
Table 12: Comparison of SVR models built with traditional data.....	139
Table 13: Comparison of SVR models built with traditional data and feature engineering ..	140
Table 14: Comparison of SVR models built with hydrological data.....	140
Table 15: Comparison of SVR models built with hydrological data and feature engineering	141
Table 16: The comparison of SVR Tuned models.....	141
Table 17: Comparison of ANN models built with traditional data.....	143
Table 18: Comparison of ANN models built with traditional data and feature engineering .	143
Table 19: Comparison of ANN models built with hydrological data	144
Table 20: Comparison of ANN models built with hydrological data and feature engineering	144
Table 21: The comparison of ANN Tuned models.....	145
Table 22: The comparison of best performing models using RF, SVR and ANN	145
Table 23: Comparison of Random Forest model with and without Hydrological data	146
Table 24: Comparison of Support Vector Regressor model with and without Hydrological data.....	147
Table 25: Comparison of Artificial Neural Network model with and without Hydrological data.....	147
Table 26: Comparing the current study with previous related work	148
Table 27: The comparison of the model with and without additional Hydrological data	161
Table 28: Comparison of the model with and without feature engineering	162

LIST OF FIGURES

Figure 1: The location of Taunggyi township (NordNordWest, 2017)	24
Figure 2: Conventional rainfall formation (Allaway, 2022)	29
Figure 3: Orographic rainfall formation (Allaway, 2022)	29
Figure 4: Cyclonic / frontal rainfall formation (Allaway, 2022)	30
Figure 5: Ordinary or Non-automatic rain gauge (Pudyastuti, 2018)	31
Figure 6: Self-recording or Automatic rain gauge (Pudyastuti, 2018)	31
Figure 7: General Data Mining Model Flow Chart	35
Figure 8: SEMMA Methodology Flow Chart (Palacios et al., 2017)	52
Figure 9: Proposed Methodology for conducting this project	54
Figure 10: Illustration of Random Forest Model (Tharun et al., 2018)	62
Figure 11: Illustration of Support Vector Machine Hyperplane (Sureh et al., 2019)	63
Figure 12: Illustration of Artificial Neural Network Archicture (Pham et al., 2020)	64
Figure 13: Uploading dataset and Python Notebook into Google Drive	69
Figure 14: Importing libraries for Data Exploration	70
Figure 15: Dataset Exploration	70
Figure 16: Descriptive Statistics of the variables	71
Figure 17: Univariate Analysis of 'temperature'	72
Figure 18: Univariate Analysis of 'sun_dur'	73
Figure 19: Univariate Analysis of 'shortwave_rad'	73
Figure 20: Univariate Analysis of 'r_humdity'	74
Figure 21: Univariate Analysis of 'cloud_covr'	74
Figure 22: Univariate Analysis of 'pressure'	75
Figure 23: Univariate Analysis of 'evapotrans'	75
Figure 24: Univariate Analysis of 'soil_temp'	76
Figure 25: Univariate Analysis of 'soil_moisture'	76
Figure 26: Univariate Analysis of 'vapor_pressure'	77
Figure 27: Univariate Analysis of 'wind_speed'	77
Figure 28: Univariate Analysis of 'wind_dir'	78
Figure 29: Univariate Analysis of 'wind_gust'	78
Figure 30: Univariate Analysis of 'rainfall'	79
Figure 31: Bivariate Analysis between temperature and rainfall	80
Figure 32: Bivariate Analysis between sun_dir and rainfall	81

Figure 33: Bivariate Analysis between shortwave and rainfall	81
Figure 34: Bivariate Analysis between r_humidity and rainfall	82
Figure 35: Bivariate Analysis between cloud_covr and rainfall.....	82
Figure 36: Bivariate Analysis between pressure and rainfall	83
Figure 37: Bivariate Analysis between evapotrans and rainfall	83
Figure 38: Bivariate Analysis between soil_temp and rainfall	84
Figure 39: Bivariate Analysis between soil_moisture and rainfall	84
Figure 40: Bivariate Analysis between vapor_pressure and rainfall	85
Figure 41: Bivariate Analysis between wind_speed and rainfall.....	85
Figure 42: Bivariate Analysis between wind_dir and rainfall	86
Figure 43: Bivariate Analysis between wind_gust and rainfall	86
Figure 44: The amount of Non-Missing Values percentage chart.....	87
Figure 45: The number of total duplicate data in each variable	87
Figure 46: Identifying zero variance variable in the dataset.....	88
Figure 47: Box plots for each variable.....	89
Figure 48: Z-score Analysis for detecting outliers	90
Figure 49: The total number of invalid data point	90
Figure 50: Potential outliers between 3500 ~ 4500 value in short-wave radiation.....	91
Figure 51: The data points between 2500 ~ 4500 in short-wave radiation.....	91
Figure 52: Potential outliers between 3 ~ 4 value in evapotranspiration.....	91
Figure 53: The data points between 2.5 ~ 4 in evapotranspiration	92
Figure 54: Invalid or inconsistent data points in the wind direction variable.....	93
Figure 55: Replacing inconsistent data with appropriate value of 360.....	93
Figure 56: Backup the original dataset and converting the date variable in Year-Month-Date format.....	93
Figure 57: The creation of seasonal variable based on the date variable.....	94
Figure 58: One-hot encoding for seasonal variable	94
Figure 59: Features creation from wind direction variable.....	95
Figure 60: Comparison of existing features and newly created features	96
Figure 61: Seasonal variables Vs rainfall variable	96
Figure 62: The relationship between previous rainfall and today rainfall amount	97
Figure 63: Correlation matrix for independent variables without soil_moisture & evapotrans	98
Figure 64: List of highly correlated variable (cut off 0.8 and above).....	99

Figure 65: Correlation matrix after dropping highly correlated variables	99
Figure 66: Correlation matrix for independent variables without soil_moisture & evapotrans with feature engineered variables	100
Figure 67: List of highly correlated variable with feature engineered variables (cut off 0.8 and above).....	101
Figure 68: Correlation matrix after dropping highly correlated variables with feature engineered variables.....	101
Figure 69: Correlation matrix for independent variables with soil_moisture & evapotrans .	102
Figure 70: Correlation matrix for independent variables with soil_moisture & evapotrans with feature engineered variables	103
Figure 71: Feature importance scores for dataset without soil moisture and evapotranspiration	104
Figure 72: Feature importance scores for dataset without soil moisture and evapotranspiration with feature engineering	105
Figure 73: Feature importance scores for dataset with soil moisture and evapotranspiration	105
Figure 74: Feature importance scores for dataset with soil moisture and evapotranspiration with feature engineering	106
Figure 75: Normalization using Standard Scaler	108
Figure 76: Data partition using a 70:30 ratio	109
Figure 77: Random Forest Regressor (Base model) using rf_1 dataset.....	111
Figure 78: Hyperparameter tuning for RF model using rf_1 dataset.....	111
Figure 79: Random Forest Tuned model using rf_1 dataset.....	112
Figure 80: Generating Evaluation metrics for RF model using rf_1 dataset	112
Figure 81: Random Forest Regressor (Base model) using rf_1_fe dataset	112
Figure 82: Hyperparameter tuning for RF model using rf_1_fe dataset.....	113
Figure 83: Random Forest Tuned model using rf_1_fe dataset.....	113
Figure 84: Generating Evaluation metrics for RF model using rf_1_fe dataset	114
Figure 85: Random Forest Regressor (Base model) using rf_2 dataset.....	114
Figure 86: Hyperparameter tuning for RF model using rf_2 dataset.....	114
Figure 87: Random Forest Tuned model using rf_2 dataset.....	115
Figure 88: Generating Evaluation metrics for RF model using rf_2 dataset	115
Figure 89: Random Forest Regressor (Base model) using rf_2_fe dataset	115
Figure 90: Hyperparameter tuning for RF model using rf_2_fe dataset.....	116

Figure 91: Random Forest Tuned model using rf_2_fe dataset.....	116
Figure 92: Generating Evaluation metrics for RF model using rf_2_fe dataset	117
Figure 93: Support Vector Regressor (Base model) using rf_1 dataset.....	118
Figure 94: Hyperparameter tuning for SVR model using rf_1 dataset.....	118
Figure 95: Support Vector Regressor Tuned model using rf_1 dataset.....	118
Figure 96: Generating Evaluation metrics for SVR model using rf_1 dataset	119
Figure 97: Support Vector Regressor (Base model) using rf_1_fe dataset	119
Figure 98: Hyperparameter tuning for SVR model using rf_1_fe dataset.....	120
Figure 99: Support Vector Regressor Tuned model using rf_1_fe dataset.....	120
Figure 100: Generating Evaluation metrics for SVR model using rf_2 dataset	121
Figure 101: Support Vector Regressor (Base model) using rf_2 dataset.....	121
Figure 102: Hyperparameter tuning for SVR model using rf_1 dataset	121
Figure 103: Support Vector Regressor Tuned model using rf_2 dataset.....	122
Figure 104: Generating Evaluation metrics for SVR model using rf_2 dataset	122
Figure 105: Support Vector Regressor (Base model) using rf_2_fe dataset	122
Figure 106: Hyperparameter tuning for SVR model using rf_2_fe dataset.....	123
Figure 107: Support Vector Regressor Tuned model using rf_2_fe dataset.....	123
Figure 108: Generating Evaluation metrics for SVR model using rf_2_fe dataset	124
Figure 109: Artificial Neural Network (Base model) using rf_1 dataset.....	126
Figure 110: Hyperparameter tuning for ANN model using rf_1 dataset	126
Figure 111: Artificial Neural Network Tuned model using rf_1 dataset.....	127
Figure 112: Generating Evaluation metrics for ANN model using rf_1 dataset	127
Figure 113: Hyperparameter tuning for ANN model using rf_1_fe dataset.....	128
Figure 114: ANN Tuned model using rf_1_fe dataset.....	129
Figure 115: Generating Evaluation metrics for ANN model using rf_1_fe dataset	129
Figure 116: Artificial Neural Network (Base model) using rf_2 dataset.....	130
Figure 117: Hyperparameter tuning for ANN model using rf_2 dataset	130
Figure 118: Artificial Neural Network Tuned model using rf_2 dataset.....	131
Figure 119: Generating Evaluation metrics for ANN model using rf_2 dataset	131
Figure 120: Artificial Neural Network (Base model) using rf_2_fe dataset	132
Figure 121: Hyperparameter tuning for ANN model using rf_2_fe dataset.....	132
Figure 122: Artificial Neural Network Tuned model using rf_2_fe dataset.....	133
Figure 123: Generating Evaluation metrics for ANN model using rf_2_fe dataset	133

Figure 124: Feature importance on rainfall prediction by best performing Random Forest Model	138
Figure 125: Feature importance on rainfall prediction by best performing Support Vector Regressor Model	142
Figure 126: A simple web interface to access the model.....	152
Figure 127: Model Test case No. (1)	152
Figure 128: Model Test Case No. (2)	153
Figure 129: Feedback (1)	154
Figure 130: Feedback (2)	155
Figure 131: Feedback (3)	156
Figure 132: Feature importance on rainfall prediction by best performing Random Forest Model	161
Figure 133: Research Plan	173

LIST OF ABBREVIATIONS

ANFIS.....	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
BC.....	Before Christ
BP.....	Backpropagation
CLR.....	Combined Linear Regression
CNN.....	Convolutional Neural Network
DMH.....	Department of Meteorology and Hydrology
DT.....	Decision Trees
ELM.....	Extreme Learning Machine
F1.....	F-Score
GDP.....	Gross Domestic Product
GFS.....	Global data Forecasting System
GPCP.....	Global Precipitation Climatology Project
IEEE.....	Institute of Electrical and Electronics Engineers
KNN.....	K-Nearest Neighbor
LDA.....	Linear discriminant analysis
LR.....	Logistic regression
LRA.....	Linear Regression Analysis
LSTM.....	Long Short-Term Memory
MAE.....	Mean Absolute Error
MLP.....	Multilayer Perceptron
MLP.....	Multiple-Layer Perceptron
MLR.....	Multiple Linear Regression
MT-CNN...	Multitask Convolutional Neural Network
MT-RNN...	Multitask Recurrent Neural Network
NB.....	Naive Bayes
NEMS.....	NOAA Environment Monitoring System
NMM.....	Nonhydrostatic Meso-Scale Modeling
NN.....	Neural Network
NOAA.....	National Oceanic and Atmospheric Administration
PCA.....	Principle Component analysis
QDA.....	Quadratic discriminant analysis

RBF.....Radial Basis Function
RF.....Random Forest
RMS.....Root Mean Square Error
RNN.....Recurrent Neural Network
S2S.....Sub-seasonal to Seasonal Prediction Project
SEMMA...Sample, Explore, Modify, Model, Assess
SVM.....Support Vector Machines,
SVR.....Support Vector Regression
WRF.....Weather Research and Forecasting
XGBoost...Extreme Gradient Boosting

CHAPTER 1

INTRODUCTION

1.1 Introduction

Weather forecasting involves estimating the state of atmospheric variables over a specified area and over a specified period. Various types of prediction models have been used to forecast weather variables such as temperature, humidity, and rainfall. As a result of natural phenomena, such as rainfall, these phenomena play an essential role in maintaining ecological balance worldwide. This results from complex interactions between the meteorological system at play and is associated with its processes of vaporization (Aswin et al., 2018). Rainfall forecasting applies to determining the amount of rainfall for a particular region at a particular time, which is one of the major types of weather forecasting. Forecasting rainfall is a crucial part of agriculture, tourism, construction, transportation, and even life itself (Sureh et al., 2019). Predictions must therefore be reasonably accurate, have a positive economic impact, and be communicated well to users in order to be considered valuable.

The ability to foretell the weather is vital in contributing to a country's economic and social growth. There is no doubt that forecasting has been an influential topic of study since the Babylonians forecasted weather based on cloud observations (observed patterns) back in 650 BC, which was over 2,500 years ago. There were many different forecasting approaches proposed during that time by many philosophers. In the course of time, it became prominent that these theories did not meet the needs of prediction. It was therefore perceived that there was a requirement for a broader outlook on the weather as well. Atmospheric measurements were carried out with the assistance of new instruments. In order to enhance the quality of weather monitoring, radiosondes and telegraphs were designed (Dueben & Bauer, 2018). There are currently a number of mechanisms that are used to measure weather conditions today. After World War I, Lewis Fry Richardson utilised arithmetic equations in his attempts to predict the rain decades before computers were created (Gomathy et al., 2021). Several new methods for predicting rainfall were produced, along with the proliferation of technological advancements in recent years. Many scientists now use a variety of methods to forecast rainfall based on weather conditions. Some of the models that are being used by many weather forecasters today require supercomputers in order to provide reliable forecasts. Weather forecasting is one of the

most substantial aspects of life around the globe since it applies to both human needs in daily life (Bosher & Chmutina, 2017).

A variety of domains can benefit from predicting rainfall, from a decision-making standpoint to a resource planning perspective, all the way to risk management. With the use of predictions, profits can be maximized, and losses can be minimized, thus maximizing profitability. There is an apparent difference between the agricultural sector and other sectors in terms of the benefits it receives (Gunda et al., 2017). A comprehensive knowledge of the quantity of rainfall that can be predicted in a particular area would help increase the profitability of agricultural activities in that area. A large number of benefits can be derived from the ability to accurately predict the seasonal and regional rainfall circumstances in agricultural operations, such as planting crops, irrigation, fertilizing, and harvesting. These benefits result in tremendous yields, thus resulting in greater profits for those agricultural areas. It is also crucial that rainfall projections are used in order to handle electricity demand, the construction enterprise, tourism, and the mining sector (Gomathy et al., 2021). It may become necessary to alter work schedules in the event of wet weather conditions. It can be beneficial if stakeholders and managers are notified of rain information in advance.

In spite of this, to the best of my knowledge, there has not been any study performed on the prediction of rainfall utilizing machine learning approaches for the city of Taunggyi in Myanmar using both traditional weather data and hydrological data. Therefore, the primary goal of this project is to seal this gap. This study focuses on projecting rain estimations using the historical precipitation data containing daily observations of climate variables for (37) years. There are two stages to this project. It is first necessary to review the existing literature as it relates to the domain understanding of rain as well as the machine learning methods used for the prediction of rainfall. Secondly, the techniques that can predict rainfall more accurately from the works of literature will be chosen to be implemented in this project, and their performance will also be compared to determine which delivers the most accurate results. In the meantime, whether the inclusion of hydrological data could help the model's performance will be experimented as well. For rainfall prediction for Taunggyi city, the best-performing model among the implemented techniques will be chosen as the final model.

1.2 Research Background

It is widely believed that rainfall is one of the most important atmospheric phenomena that helps the environment as well as all life on planet Earth. The precipitation is also a factor that affects the local meteorological conditions as well as the atmospheric circulation. A large role is played by rainfall in maintaining the right balance between climbing temperatures and providing the necessary conditions to sustain life. The increasing global temperatures is having an adverse effect on the availability of water, one of the most valuable and essential resources on the planet (Bagirov et al., 2017). As well as this, precipitation also acts as a compensating factor for all of these resources, as well as promoting the productivity of crops and agricultural products in the area (Hartigan et al., 2020).

Rainfall is just the fall of water drop that result from the condensation of moisture in the atmosphere (Ahmed & Direkoglu, 2018). A rainfall or precipitation can be categorised into three different types. Conventional rainfall is the most common form that occurs in most parts of the world. This phenomenon is particularly observed in tropical and sub-tropical areas at higher latitudes, as well as during thunderstorms and lightning storms (Selase et al., 2015). Another type of rainfall is orographic rainfall. A key factor in the formation of those clouds is the accumulation of moist air in the vicinity of mountains which is raised upward. Mountains with a midlatitude location are usually the most common place to find this type of rain. Finally, cyclonic rainfall, which is also known as frontal rainfall, originates from frontal systems or cyclonic systems. Tropical regions are the most likely to form them, while temperate regions are the least likely. Frontal rainfalls bring excessive precipitation with rapid onset, and if they last for a long time. They could result in wet weather for days afterward (Ahmed & Direkoglu, 2018).

A millimeter (mm) is the unit of measurement for rainfall. The amount of rainfall can be measured with two different instruments. Regardless of the type, they are fundamentally funnels that collect rainwater. Their capacity is 25mm. Ordinary rain gauge instruments are conventional methods for measuring rainfall. This instrument is ineffective for gathering precipitation when heavy rain is caused by cyclones and air fronts (Porto de Carvalho et al., 2014). Comparatively, it is less accurate and error-prone than the other types. There is also a rain gauge that records its own measurements which can be used as a measuring instrument. As compared to ordinary rain gauges, this instrument can provide a higher level of accuracy and reliability (Agnihotri & Panda, 2014).

Predictions of rainfall are provided in four different intervals according to different purposes, i.e., the nowcasting corresponds to a forecast for the next few hours, whereas the

short-range forecasting focuses on the next three days. A forecast that extends from a few days to a week can be considered to be medium range, while a forecast for a few months or a season may be referred to as long-range or extended range (Vitart et al., 2017). Depending on the accumulation process, the rainfall data type can be divided into two types, each of which comes with its own advantages and disadvantages. The first type is the measurement data that can be obtained from a variety of rain gauge stations that record rainfall data. However, it does not provide complete coverage for all locations in the world. In order to solve this problem, simulation data is used. In this case, the data are generated from progressive climate monitoring systems, which is able to provide users with 100 % coverage of the world at all times (Meteoblue, 2022). A number of meteorologists, like Iryani et al., (2021), have asserted that the simulation data can be substituted with measurements for the purposes of climate research. In order to make long-range weather predictions, these data are appropriate. A simulation dataset for the Taunggyi city is being used for this project as a basis for its modeling.

In order to predict rainfall, two approaches are currently being used. The first type is a conventional dynamic model that utilizes numerical weather prediction systems such as the Weather Research and Forecasting Model (WRF), Seasonal Climate Model, and Global Forecast System (GFS). Due to the complexity of the earth's weather system, these models require a lot of data and processing power. As for the second prediction, it will be based on general data mining models. In the meteorological field, this is commonly referred to as meteorological data mining. In terms of data mining models, machine learning techniques are an integral part of them. There are some scientists who are questioning whether the deep learning techniques can perform better than the conventional numeric weather systems as a result of the limitations of the standard numeric weather system (Dueben & Bauer, 2018).

A machine learning approaches for rainfall prediction consists of rainfall classifications and rainfall estimation models. Using the classification models, it is possible both to accurately predict whether it will rain or not that is, whether it will be binary 'Yes' or 'No' plus the intensity of the rain which could be either 'Light', 'Medium' or 'Heavy' rainfall. Several rainfall classification models have been used in the literature by researchers in the past. As an example, Raval et al. (2021) compared the results of traditional classification methods including Naive Bayes, K-Nearest Neighbor, Quadratic discriminant analysis, Linear discriminant analysis, Decision tree, Logistic regression, Random Forest, and Neural Network models on rainfall datasets collected from 26 locations in Australia. The Neural networks performed better in their study.

Aside from predicting the category of the rainfall variable, there is also a model that calculates the rainfall amount according to the climate variables, which could be defined as a regression problem in the context of machine learning. Many scholars also use the regression model to predict rainfall. For example, Qiu et al. (2017) and Aswin et al. (2018) employed deep learning architectures such as Long Short-Term Memory, Recurrent Neural Network, and Convolutional Neural Network in order to predict the estimated rainfall amount. The similar study was also performed by Bagirov et al. (2017), several models were implemented, including Multiple Linear Regressions, Support Vector Machines, and Artificial Neural Networks. It has also been presented in Dash et al. (2018) that a study was carried out using K-nearest neighbor, Artificial Neural Networks, and Extreme Learning Machines to study seasonal monthly precipitation. In a study conducted by Basha et al. (2020), they evaluated Random Forest and Artificial Neural Networks, as well as Linear Regression Analysis as methods for forecasting average rainfall during each season. The research has shown that the Random Forest regression model provided the highest level of accuracy in comparison to other regression models.

1.3 Problem Statement

Since the advent of global warming in the past few decades, climate has been changing a lot as a result of the increase of greenhouse gases in the atmosphere; this has made it more difficult for the planet and living organisms to receive the amount of precipitation that they require. In consequence, in order to satisfy the needs of humans and to prevent the occurrence of natural disasters that may be triggered by unexpectedly heavy downpours, the assessment of rain patterns has become an essential part of informing human decisions. As far as environmental disasters are concerned, Myanmar was previously thought of as a relatively safe country. This perception gradually started to change after the event of cyclone Nargis taking place in May 2008, one of the greatest natural disasters ever experienced in the country. Recent years have seen numerous major disasters in Myanmar, including cyclones, droughts, landslides, and floods which have affected various areas of the country. It is therefore important to develop a system that can accurately predict rainfall for specific geographic area. Myanmar's Taunggyi township is also one of those regions affected by natural disasters in its eastern region.

The Asia Development Bank conducted a study on Myanmar's agricultural sector in 2015, and based on that report, Shan State, where Taunggyi city is located, could be one of the most vital farming systems of the country. According to Raitzer et al. (2015), this is an area

that can produce a variety of crops, vegetables, and fruits. Despite the significance of the area for the agricultural productivity in the country, there have been relatively few studies carried out on the area and no study has been completed on the prediction of rainfall using machine learning techniques, even though literature in the area is extensive. There are several scholars in Myanmar who have conducted research in other parts of the country such as Yangon (Mar & Naing, 2008), Chauk townships (Aung, 2019), and Pyin Oo Lwin (Thwe et al., 2019). Hence, there is a necessity for the development of new types of rainfall prediction for Taunggyi city.

In the meantime, although some researchers such as Kundu et al. (2017) and Wang et al. (2018) claimed that the inclusion of hydrological data such as soil moisture and evapotranspiration could improve in predicting the rainfall, there is no such study for rainfall prediction model using traditional weather data incorporation with hydrological data. Therefore, this project will seal the gap and experiment whether the inclusion of hydrological data will improve the performance and accuracy of the rainfall prediction model. Furthermore, the best model will offer benefits to the residents of the area and to local authorities by providing them with more reliable rainfall information, predicting the quantity of rainfall and the expected pattern of precipitation that is relevant specifically to this area.

1.4 Research Questions

- What machine learning techniques and evaluation metrics could be used to build and assess the performance of a rainfall prediction model?
- Does a model trained on traditional weather data or one that is additionally trained on hydrological data perform better in predicting rainfall amount?
- Is there any particular feature(s) from hydrological data that will play a more significant role in predicting the amount of rainfall?
- Is the feature engineering process going to provide the best results in terms of efficiency and performance for the rainfall prediction model?

1.5 Aim of the Study

To implement a machine learning model that can predict the quantity of rainfall with the highest degree of accuracy by incorporating the hydrological data and the traditional weather data, thereby providing informed decision-making for residents and local authorities of Taunggyi city.

1.6 Objectives of the Study

As specific objectives, this study aims to accomplish the followings.

- To identify appropriate machine learning techniques and evaluation metrics for the development and evaluation of a rainfall prediction model.
- To compare the performance of prediction models trained solely on traditional weather data versus those trained on both traditional weather data and hydrological data in predicting rainfall amount.
- To identify the specific feature(s) from the hydrological data that significantly influence the prediction of rainfall amount.
- To evaluate the effectiveness and efficiency of the feature engineering process in improving the performance of the prediction model.

1.7 Significance of the Study

Having the ability to predict rainfall has a positive effect on both macro and micro levels, as explained in above sections. Due to the fact that there has never been a study that has examined rainfall prediction for Taunggyi using machine learning techniques incorporating the hydrological, it seeks to construct a model of rainfall prediction that is reliable and accurate for the area. With the aim of providing great contributions to the agricultural sector, this study will be able to provide valuable information regarding rainfall for harvesting and planting crops, managing and allocating water resources, managing floods and predicting weather patterns, with the intention of keeping local authorities, businesses, and the community informed. In addition, the rainfall prediction has considerable importance for agricultural industries as it could help maintain the safety of crops and ensure the production of vegetables and fruits. Furthermore, the study will be beneficial to flood management authorities because it will serve as a more accurate forecast of monsoon rains. By taking preventative measures in advance, the damage caused by a flood could be underrated, and the management can stay alert for a foreseeable threat.

1.8 Scope of the Study

This capstone project focuses on Taunggyi Township, a township located in eastern Myanmar, as shown in Figure (1). The final deliverable of the project is a rainfall prediction model for Taunggyi, which has been constructed based on the rainfall simulations data that have been collected on a daily basis that have been obtained from meteoblue® weather. As far

as the rainfall dataset is concerned, it is only permitted to be used for academic research purposes. This study, however, does not investigate whether global warming or climatic change will impact the predicted results in any way, since this is a topic which is beyond the scope of the study. Moreover, the results of the rainfall prediction are only applicable to the focus region as the pattern of rainfall is different from area to area. To implement the entire project, the Python programming language will be used on both a local machine and the Google Colab platform in the cloud accordingly. A literature review will provide the basis for the decision regarding which models could provide the higher accuracy results, as well as the evaluation metrics to be for the for comparing the model performance.



Figure 1: The location of Taunggyi township (NordNordWest, 2017)

1.9 Structure of the Report

There are six chapters in the capstone project report. Below is a detailed description and summary of each chapter.

Chapter 1: This is the first chapter which presents an overview of the whole project. The purpose of this section is to provide a brief overview of what the researchers has been discovered in the past and the introductory description for the background of this project. In addition, the chapter discusses the problem that this study tries to address, the research questions that will be answered, the study's aim and objectives along with the significance of the study as well as its scope.

Chapter 2: This chapter provides a review of the available literature that has relevance to this project. There will be a critical analysis of the past studies regarding the dataset used, the number of features, the methods and techniques adopted to develop the rainfall prediction model as well as the evaluation metrics to assess the performance of the machine learning models.

Chapter 3: A detailed description of the methodology to be used in this project will be provided in this chapter. It includes the procedures for collecting data and choosing a dataset, a detailed description of how the data-processing will be conducted, the methods for implementing the selected techniques and their descriptions along with the metrics for evaluating the model which has been built.

Chapter 4: This chapter tries to provide a step-by-step explanation for the whole process of implementation which was proposed in Chapter (3), starting from data exploration till the comparison of the model.

Chapter 5: The purpose of this chapter is to elaborate on the findings, analysis and, discussions derived from the previous chapter. In this chapter, the models that have been created will be closely evaluated and compared in terms of their performance. There will be an in-depth analysis of the results, where the best performing model will be proposed and deploy the model using a simple web interface. The feedbacks for the use of model from the relevant users will be also included in this chapter.

Chapter 6: This provides a summary of the whole capstone project that was completed. A brief synopsis of the study, along with an overview of the contributions, limitations, and future recommendations resulting from the study will be presented in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

A major aim of this chapter is to provide a basic understanding of the techniques that are used to develop rainfall prediction models. Due to the complexity of the atmospheric cycles, rainfall prediction is considered one of the most challenging aspects of hydrology and meteorology. In order to tackle these problems, many researchers have joined the domain and contributed priceless information that could be utilized to predict rainfall in the future. In this chapter, a critical analysis and discussion of some of the past research related to this project will be offered as well. Furthermore, the findings from the researchers will be explored as well as the evaluation methods and the limitations of the study. There will be a literature review matrix at the end of this chapter which provides a summary of previous research.

2.2 Method

This study utilized several databases in order to identify previous research publications and research results relevant to the study such as IEEE, Google Scholar, SpringerLink, Science Direct, and ProQuest. For the sake of Machine learning, the recent journals and articles published after 2017 and 2022 will be selected. However, the available resources will be reviewed for the purpose of conducting a background and domain study. Firstly, an abstract of every article is examined and then the articles which closely relevant to this study are selected for further analysis.

2.3 Climatology of Taunggyi City

At an elevation of 4711 feet or 1436 meters, Taunggyi city could be found at a geographical latitude of 20°47'N and a longitude of 97°2'E. Its climate can be described as a humid subtropical climate, with a hot, humid summer and a mild, dry and moderate winter, but a moderate seasonality. It is classified as "Cwa" by Köppen-Geiger, which means that the weather is moderately humid year-round. A typical temperature in Taunggyi throughout the year is 19.4 °C. On a monthly basis, the temperature varies by an average of 7.8 degrees Celsius depending on the time of year. It is a hyper oceanic type of continentality. On average, April is one of the hottest months of the year, with 22.2 °C average temperature. January, with 14.4

degrees Celsius average temperature, making it the coldest month of the year in this region. The annual precipitation in Taunggyi is 1551.9 mm on average, with the majority falling during the rainy season. A month with the most precipitation is August, which has on average 330.2 mm of precipitation, which is more than any other month. As far as precipitation in general is concerned, the month of January has 2.5 average precipitation in millimeters, making it one of the least precipitous months in the year. According to the DMH, there are approximately 81.6 rainy days per year, the highest number being 16.7 rainy days occurring in August, and the lowest number being 0.2 rainy days occurring in January (DMH, 2022).

2.4 Rainfall Background

All living organisms on this planet benefit from rainfall, a major atmospheric phenomenon. It is claimed that climate change has had a direct or indirect impact on everything and that it is essential for humans to take into account how precipitation has changed in response to climate change over the past few decades (Bagirov et al., 2017). Moreover, it has an influence both on the global gauge of the atmospheric circulation as well as the influence of rainfall on local weather. Rainfall plays a crucial role in offsetting the rising temperatures, ensuring the survival of all life forms. Because of the global warming, the temperature of the earth is rising, and one of the scarcest and most essential resources on Earth, water, is evaporating from the reserves as a result of this temperature increase. It is also important to understand that the rainfall contributes to all these resources, and that it is crucial for agriculture and crop production in general (Hartigan et al., 2020). A rainfall phenomenon varies according to the latitude and longitude as well as to the planes of the area. It also differs according to mountains and plateaus within the region. Precipitation also has a tendency to be inconsistent, and it also changes in a certain way from year to year. Water evaporates from the surface of the earth when it is heated by high temperatures, which leads to evaporation from the surface and coming down as rainfall from the clouds as a result (Alpers & Melsheimer, 2004).

With the growing needs of mankind, rainfall has become a very significant phenomenon. Water cannot be controlled artificially by humans on this planet, and they cannot live without it. In part, this explains why most studies and research are carried out around the world (Dash et al., 2018). For the sake of offsetting excessive water use across the globe, meeting growing needs for agriculture, water resources, and preventing natural disasters such as floods and land sliding, scientists are collaborating to find the most effective method of estimating and predicting rainfall (Tharun et al., 2018).

Rainfall also affects the environment in many ways and helps clean the environment. There is a freshness and cleanliness to the atmosphere when it rains. Rainfall is also one of the most important factors in regulating pollution that has been caused by humans. Rainfall amounts can vary considerably according to the region and area of a country; therefore, it is possible for floods, tsunamis, and landslides to occur in some regions due to the effects of precipitation (Sureh et al., 2019). Consequently, scientists and researchers are being encouraged to study ways of accommodating the natural phenomenon of precipitation, and its effects on land, so that the resource could be managed more effectively and beneficially.

2.4.1 Rainfall Categories

The term rainfall simply could be referred as the liquid drops from the cloud which is the resultant of condensation of water vapor in the atmosphere (Selase et al., 2015). In general, there are three types of rainfall: conventional, orographic, and cyclonic or frontal rainfall.

2.4.1.1 Conventional rainfall

In terms of rainfall types, convectional rainfall is by far the most dominant. The condition occurs in subtropical and tropical areas having higher latitudes (Selase et al., 2015). These types of rainfall are influenced by the mountains. Due to the heat emanating from the sun, conventional rainfall occurs when air conditions above Earth's surface become severe. Clouds appear in the atmosphere when the hot air in the atmosphere evaporates from the surface of the ground and ascends to the upper atmosphere as it is lighter than the cooler air in the atmosphere. A further increase in the water vapors resulted in clouds that were dense and heavy as a result. Because the unstable clouds ascend higher, they descend as conventional rain on the earth. West Africa experiences the majority of conventional rainfall. As a result of unstable clouds rising and converging air in the atmosphere, the process usually results in heavy lightening as well as a thunderstorm (Blyth, Bennett, & Collier, 2015).

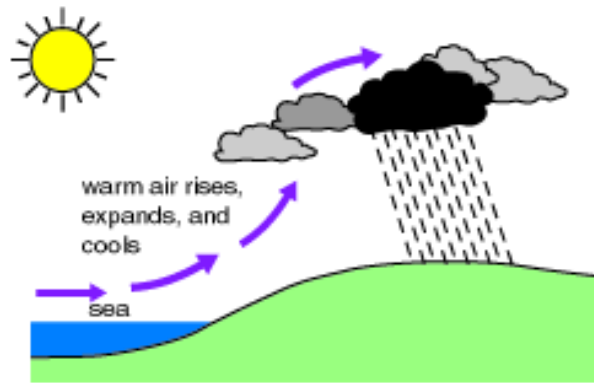


Figure 2: Conventional rainfall formation (Allaway, 2022)

2.4.1.2 Orographic rainfall

Orographic cloud formations are typically caused by humid air and can often be seen over the mountains. The damp air around the mountains is evaporated upwards as a result of the mountainous terrain. When humid air reaches a certain height, it cools off, creating orographic clouds. It is during this time that the clouds thicken and start to fall as precipitation. It has been found that orographic rainfall occurs more often at midlatitudes with large mountains than at lower latitudes (Ahmed & Direkoglu, 2018). During orographic rainfall, tiny drops of water become condensed and fall onto the ground (Selase et al., 2015).

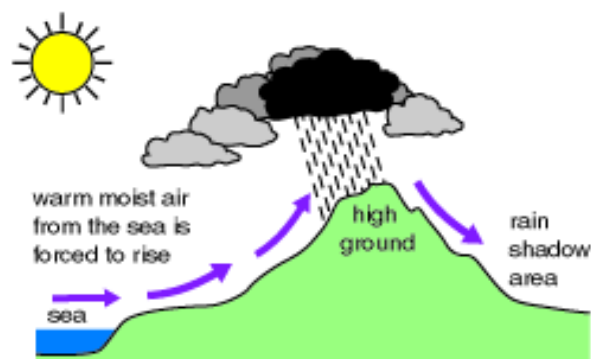


Figure 3: Orographic rainfall formation (Allaway, 2022)

2.4.1.3 Cyclonic or Frontal rainfall

A third type of rainfall is the rain that is caused by cyclonic systems or frontal systems. A cyclonic storm, as the name implies, is sometimes described as a tempest caused by a collision between various air masses which have highly diverse characteristics. There is a phenomenon whereby cool air collides with warm air, and since the cold air is heavier than the

warm air, it propels the warm air to rise. As rising air cools, water vapor is formed as a result. The condensation process then leads to the formation of clouds (Selase et al., 2015). A tropical area with a latitude of 23° north and 66° south has a higher likelihood of experiencing this type of rainfall than a temperate zone. The rain is also known as the frontal rainfall, or the rain that falls from the front. A cyclonic or frontal rain event may result in an extension of precipitation, which results in wet weather for many days (Ahmed & Direkoglu, 2018).

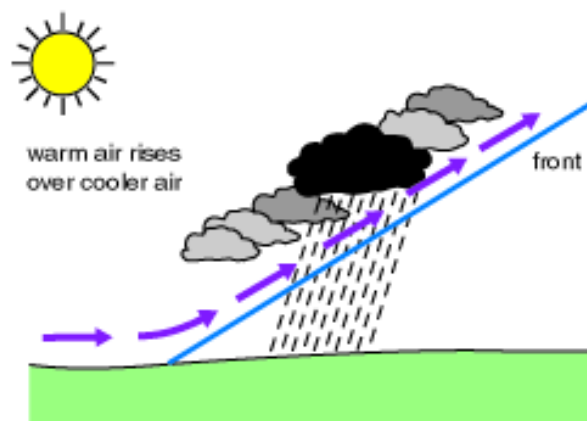


Figure 4: Cyclonic / frontal rainfall formation (Allaway, 2022)

2.4.2 Measurement of rainfall

Generally, there are two rain gauges: Standard rain gauges and self-recording (automatic) rain gauges. The standard measurement unit for rainfall is millimeters (mm). There is a measuring device designed to collect the rainfall with a diameter of 203mm consisting of a funnel to gather rainwater and an additional cylinder measuring up to 25 millimeters of rain.

2.4.2.1 Ordinary Rain Gauge

By collecting rainwater in a storage container and using a glass to measure the rainfall, the ordinary rain gauge is a manual device that measures rainfall regularly. During a measurement period, rainwater is compared and measured according to a rainfall record book. Because the measurements are manual, there is a higher chance of errors that cannot be minimized (Agnihotri & Panda, 2014). Heavy precipitation, such as those caused by cyclones and fronts, is ineffective and may result in less accurate data (Porto de Carvalho et al., 2014). In spite of the fact that this method is suitable for local-level and less precise measurements, it can also be used for more expansive measurements (Agnihotri & Panda, 2014).



Figure 5: Ordinary or Non-automatic rain gauge (Pudyastuti, 2018)

2.4.2.2 Self-recording (Automatic) Rain Gauge

A rain gauge that automatically or self-records rainfall is more accurate and efficient than a standard rain gauge. In order to achieve higher accuracy and reliability, these gauges utilize simple instruments and methods. During rain measurements, a lever balance and an indicator bucket are used to weight the rain. As a result, each movement level is recorded with a mark pen, and rainfall is automatically recorded by the gauge without any human assistance (Agnihotri & Panda, 2014).



Figure 6: Self-recording or Automatic rain gauge (Pudyastuti, 2018)

2.4.3 Rainfall Forecasting Intervals

The purpose of a rainfall forecast is to make predictions about what might happen in the future. These predictions are divided into a number of different intervals, namely:

2.4.3.1 Now-casting

As a result of evaluating present weather conditions and forecasting precipitation over the next few hours, the term "now-casting" is used. As a result of this weather forecast, commercial aviation, outdoor events, the construction industry, and electricity utilities can be benefited. The majority of now-casting methods are based upon extrapolation models derived from observations, commonly rainfall radar (Prudden et al., 2020).

2.4.3.2 Short-range forecast

It is possible to extend the forecast period up to three days by making short-range weather predictions for the next day or two. In addition to helping us determine when a cold front is going to arrive, we can also study how rainfall and thunderstorm systems develop and spread. As a general rule, short-range forecast models tend to be regional rather than global in nature, based on the domain (Chantry et al., 2021).

2.4.3.3 Medium-range forecast

In recent decades, the medium range weather forecast field, which covers a few days to a few weeks, has experienced significant advancements. With the help of modern numerical weather prediction models, rainfall prediction for medium-range timescales has become increasingly accurate. These models utilize grid spacings of around 10 km to produce satisfactory results (Bauer et al., 2015).

2.4.3.4 Long-range (Extended range) forecast

Long-range Predicting is a term used to refer to monthly or seasonal forecasts. The timeframe for sub-seasonal forecasting ranges from two weeks to a season. The Sub-seasonal to Seasonal Prediction Project (S2S) has been established to focus on sub-seasonal forecasting. S2S delivers access to large databases containing forecasts from exploration and operational models. These databases can be employed for machine learning purposes. Recently, there has been a lot of curiosity in sub-seasonal forecasting, and S2S has been instrumental in advancing research in this area (Vitart et al., 2017).

2.4.4 Types of Rainfall Data

The dual categories of precipitation data are determined by their collection method and features. The following section will discuss these categories of rainfall data, their potential uses, and any associated weaknesses or limitations.

2.4.4.1 Measurement data

Rainfall data are extensively used for various purposes, including weather and climate forecasting, irrigation planning, and flood forecasting. Multiple organizations, such as meteorology, climatology, and geophysics agencies, can supply these data at different intervals, including hourly, daily, monthly, and yearly. However, obtaining rainfall measurements from remote locations can be challenging. Consequently, the lack of adequate data collection has hindered the progress of meteorological research. Itryani et al. (2021) suggested that simulation data be used to replace actual measurement data to address this problem.

2.4.4.2 Simulation data

It is possible to obtain simulation rainfall data by utilizing modelling, which covers 100% of the globe and can be used to simulate a variety of global events. A number of topography and ground and shallow coverage parameters are incorporated into these models, which incorporate NOAA Environment Monitoring System (NEMS) or Nonhydrostatic Meso-Scale Modeling (NMM). It is possible to determine the accuracy of the information by analyzing the variables calculated and the resolution of the information. Meteoblue, 2022). A simulation model can be used as an alternative to actual measurements in most situations, as stated by Iryani et al. (2021). Because these models provide a similar level of spatial resolution, worldwide coverage, number of weather variables, number of years, intervals of time, completeness, and consistency to measurement data, they can be used as an alternative to actual measurements in most situations. In spite of the fact that simulation data can be helpful when assessing trends and conditions over extended periods of time, local weather conditions or extreme weather events are more accurate when measured. As a result, some insurers may not accept simulation data (Meteoblue, 2022).

2.5 Methods of Rainfall Prediction

Generally, two types of models are used to predict rainfall at the national and regional levels: traditional dynamical models and general data mining models.

2.5.1 Conventional Dynamic Method

The conventional dynamical approach to predicting rainfall involves using physical models that employ equations to forecast global climate changes from initial atmospheric circumstances. This technique uses numerical methods to implement the forecast.

2.5.1.1 The Weather Research and Forecasting Model (WRF)

A weather Research and Forecasting System (WRF) is an atmospheric simulation and forecasting technique utilized in systematic and operational applications. In order to improve predictions and fast-track the transfer of investigation advancements to operations, a collaboration among multiple agencies developed this forecasting system that uses data assimilation. Geogrids are employed in the system to specify the model domains and to interpolate geographic data onto the grids (Pu & Kalnay, 2018).

2.5.1.2 Seasonal Climate Forecasting Model

Seasonal climate prediction for climate-sensitive sectors in the tropical Pacific province is feasible because of the high predictability of the region's climates. The Australian Bureau of Meteorology has been managing the Pacific Island-Climate Prediction Project since 2004 to provide training, decision support software, and seasonal climate forecasts for the National Meteorological Services of Pacific Island countries. The software uses a statistical approach employing discriminant analysis, correlating variables with local predictions to produce seasonal predictions. (Vitart et al., 2017)

2.5.1.3 Global Data Forecasting System (GFS)

To predict the global climate, NOAA's Global Forecast System (GFS), operated by a global computer system and using variational analysis, is operated by the National Oceanic and Atmospheric Administration. It is widely accepted that the accuracy of weather predictions beyond seven days could be improved with the model, which is run four times per day. It produces generally accurate forecasts for up to sixteen days in advance. Providing cost-effective meteorological analysis and forecasting to its members is a crucial responsibility of GDPFS (Pu & Kalnay, 2018).

2.5.1.4 Challenges of Numerical Weather Prediction Models

Creating numerical weather forecasts is computationally intensive, and the accuracy of these forecasts decreases quickly, even with the best models. Limited computer processing

capacity means that high-resolution simulations of processes that cannot be explicitly resolved are impossible. Because most Earth system processes are nonlinear differential equations and involve interactions between different components, so approximations must be made while developing the weather prediction model. Additionally, collecting large amounts of data to establish initial weather conditions introduces uncertainty and errors. It is difficult to incorporate a large number of observations into weather prediction models due to their computational complexity and simplified assumptions. It is being explored by scientists whether machine learning techniques, including deep learning, can be used to train climate data for prediction or even to replace numerical weather models (Dueben & Bauer, 2018).

2.5.2 General Data Mining Method

There are a number of steps involved in the making of a weather forecast, including observations, collecting and transforming data, visualizing weather data, analyzing it, extrapolating it, and predicting specific weather variables. It is possible for meteorological data mining to be used to extract knowledge from large amounts of data, also called meteorological data mining. A data mining technique has been proven to be one of the best tools for discovering valuable, non-obvious knowledge within large data sets. It has been demonstrated that this novel technique can facilitate the analysis of data and the development of decisions (Shekana et al., 2020).

The availability of climate data has increased in recent years, including observations, satellite images, radar images, and proxy data. In order to analyze and extract knowledge from these data, it is imperative to locate practical tools (Aftab et al., 2018). In meteorology, the weather is a valuable source of information, and the extracted knowledge of the weather can be a valuable asset to understanding and predicting climate variability. Despite the fact that this understanding can be applied to several vital sectors, including agriculture, tourism, and water resources, one of the greatest challenges met by meteorologists worldwide is making accurate predictions of weather conditions (Shekana et al., 2020).



Figure 7: General Data Mining Model Flow Chart

2.5.3 Rainfall Prediction using Machine Learning Techniques

Machine learning is part of the broader data mining approach in meteorology. Machine learning refers to using algorithms, models, and programs to enable machines to learn from data similarly to humans, allowing for the generation of new knowledge and judgments. In order to improve performance, machine learning techniques rely heavily on high-quality data. In meteorology, there are two main types of rainfall forecast via machine learning: rainfall classification and rainfall estimation (Chu et al., 2022).

2.5.3.1 Rainfall Classification Models

Classification involves identifying and organizing objects and concepts into specific groups. Classification algorithms are utilized to predict whether new data fit into predetermined categories. In rainfall prediction, the goal is to determine whether it will rain, and the target variable is labelled accordingly as rain or no rain. Researchers in the field of rainfall classification have employed various machine-learning algorithms. The subsequent paragraphs outline some of the relevant studies in this area.

In the study of Aftab et al. (2018) examined several data mining techniques for predicting rainfall in Lahore, Pakistan, regardless of whether it is expected to rain. They examined the previous two years' weather data in Lahore. In addition to Naive Bayes and Decision Trees, Support Vector Machines, k-Nearest Neighbors (KNN), and Multilayer Perceptron (MLP), the researchers employed several algorithms. This study concluded that, out of all the models tested, the Decision Tree model was the most effective due to the normalization of the variable values and imputed missing values using the mean value.

Based on rainfall data collected from 26 locations in Australia, Raval et al. (2021) compared the performance of an optimized neural network prediction model with other existing techniques. Several traditional models were utilized, including Decision Trees, Logistic Regressions, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Random Forest, K-Nearest Neighbor, and Naive Bayes. Precision and F1 metrics were used to evaluate the model's performance. Based on the study's results, the Neural Network model performed the best, followed by the Logistic Regression model. Although the authors utilized tenfold cross-validation, they needed to balance the classes.

Sarasa-Cabezuelo (2022) used the same rainfall dataset from 26 locations in Australia and applied several widespread algorithms, including Neural Network, Random Forest, K-Nearest Neighbor, and Decision Tree, to foreshadow the amount of rainfall daily. The

researcher employed ten-fold cross-validation in their analysis and found that the Neural Network model outperformed the other approaches in forecasting rainfall.

Based on a daily rainfall dataset obtained from various regions in India, Ganesh et al. (2021) conducted a study that compared two popular supervised machine classifiers, Random Forest and Logistic Regression, to predict the rainfall status for the following day. Logistic Regression and Random Forest models both demonstrated impressive accuracy, although the Logistic Regression model outperformed the Random Forest model slightly. This study did not balance the classes in the target variable, and it contained only 966 observations in the dataset. In their work, missing values were imputed with the mean value, normalized, and rescaled but no feature selection methods were used.

According to Sasikala and colleagues (2020), a time series dataset covering a period of over 15 years was used to analyze the binary classification of 'rainy' and 'non-rainy' days using a Support Vector Machine and Artificial Neural Network. The data was used to create the target variable based on rainfall amounts on specific days by simply dropping missing values. Those rainfall amounts less than 2.5 mm were classified as 'no-rain', whereas those equal or greater were classified as 'rainy'. Before building the models, they normalized the data using min-max and optimized both algorithms using different settings, including polynomial and RBF for SVM. A polynomial function SVM model outperformed a non-class balancing or feature selection method. The investigation concluded that the polynomial function SVM model performed better.

2.5.3.2 Rainfall Estimation Models

Rainfall prediction estimation involves employing machine learning algorithms to predict the amount of rainfall, which can be hourly or daily, monthly averages, or annual estimations. Recent publications reveal that many researchers have endeavored to forecast the strength or magnitude of precipitation utilizing a ample range of machine-learning algorithms. As an example, Qiu et al. (2017) developed a neural network-based short-term rainfall prediction model using deep convolutional neural networks. The deep convolutional neural networks were used for automatic feature extraction from observed data from multiple sites, and a fully connected layer for prediction. To extract independent variables from a time series dataset, the researchers used a Convolutional Neural Network (CNN). Their comparison was based on the application of various hyperparameter settings of the proposed model as compared to other models, including Linear Regression (LR), Quantitative Precipitation Forecast (QPF), Multiple-Layer Perceptron (MLP), and Recurrent Neural Network (RNN). An end-to-end

multisite model was developed based on the knowledge gained from one site and was able to be applied to similar sites. It significantly outperformed other models.

Aswin et al. (2018) conducted an investigation using convolutional neural networks (CNNs) and three-layered long-short-term memories (LSTMs) to improve the accuracy of rainfall prediction models over time. The Global Precipitation Climatology Project (GPCP) time series datasets were used to estimate precipitation from 1979 to 2018. Feature extraction was performed using convolutional layers, dimension reduction was accomplished using max-pooling layers, and rainfall prediction was performed using fully interconnected layers. For the purpose of analyzing the dataset, the three-dimensional arrays of latitudes, longitudes, and months were transformed into two-dimensional arrays of locations and months. According to their findings, CNN was slightly more accurate than LSTM in their investigation and increasing the number of hidden layers reduced error significantly.

To determine the effectiveness of the hybrid method, Bagirov et al. (2017) analyzed data from eight different weather stations in Victoria, Australia to determine whether the hybrid approach combines regression and clustering algorithms. Furthermore, a comparison was made between the results of this algorithm and those of other algorithms, including Support Vector Machines (SVM), Multiple Linear Regression (MLR), and Artificial Neural Networks (ANN). Compared to the other algorithms, the proposed method performed better.

Tharun et al. (2018) employed regression techniques and statistical modelling to develop a precipitation intensity forecast model. As part of the data preprocessing, mean imputation, categorical encoding, one-hot encoding, feature scaling, and normalization was used. They used Support Vector Regression (SVR), Decision Tree (DT), and Random Forest (RF) with hyperparameter tuning. According to their findings, radial basis kernels outperformed polynomial kernels in SVR by outperforming. It has been shown that RF regression produced a more accurate and effective prediction model than SVR and DT models by altering evaluation criteria and increasing the number of estimators. In addition, they found that statistical models could not deliver sufficient forecast accuracy due to the complexity of the input variables used in developing the model.

Artificial intelligence techniques were employed by Das et al. (2018) to predict seasonal monthly precipitation. The dataset of the Indian state of Kerala was preprocessed using min-max normalization before training the neural networks, which included K-Nearest Neighbor (KNN), Artificial Neural Network (ANN), and Extreme Learning Machine (ELM). According to them, the accuracy of ANN predictions is highly dependent on the number of hidden layer nodes, and when using a small dataset, overfitting may occur. A single-layer feed-

forward architecture, the Levenberg-Marquardt algorithm, a radial basis activation function, an input, a hidden, and an output layer were used to construct the ANN model. They conclude that the ELM method outperformed the other two methods with fewer errors when comparing the final model, which contained eight input neurons, 20 hidden neurons, and one output neuron.

Using ANFIS, ANN, and SVM machine learning models, Pham and colleagues (2020) investigated the prediction of daily precipitation in the province of Hoa Binh, Vietnam. The models were developed using techniques such as c-means clustering, Gaussian membership function, and scaled conjugate gradient algorithms with ten hidden layers. By using statistical analysis and Monte Carlo simulations, they assessed the robustness of their models. Based on their results, the SVM model was found to be more accurate and robust than the other models for predicting rainfall in this region.

A study by Sureh et al. (2019) used Support Vector Regression (SVR) and K-Nearest Neighbor (KNN) to predict monthly precipitation using meteorological variables from the Chabahar station. The Relief Algorithm and correlation analysis feature selection techniques were used to identify significant precipitation-related variables. To address non-linearity and high dimension issues, a normalized polynomial kernel function was implemented in SVR. The model with a variable set of maximum mean, and wind speed performed better than the KNN model. This model effectively reduced the dimension in hydrological modelling due to its performance.

The study of temperature and precipitation patterns in the Sydney catchment area by Hartigan et al. (2020) was conducted using data gathered between 1957 and 2019. The scholars employed Random Forests, Multiple Linear Regressions (MLR), and Support Vector Regressions (SVR) to predict precipitation over the area. Furthermore, eight "climate driver" variables and local weather station data were incorporated into the analysis. According to the study, specific driver attributes influence precipitation predictions in autumn and winter; however, not all driver characteristics affect all seasons. To predict rainfall, mean values were used for imputation, followed by a ten-fold cross-validation of the models. Their results indicated that Multiple Linear Regression was the most effective method.

Several data-driven models were compared for predicting average rainfall during the monsoon season in Odisha, India, by Basha et al. (2020). In order to reduce computational costs, linear regression techniques were used to reduce dimensions before fitting the models. These techniques included Linear Regression Analysis (LRA), Random Forest (RF), and Artificial Neural Network (ANN). In addition to parameter optimization, the researchers

implemented the backpropagation method in the ANN model and determined the optimal parameter for RF as a thousand trees. Compared to the other models, the Random Forest model provided the most accurate predictions in the study.

The performance of three machine-learning techniques for predicting precipitation was examined by Gomathy et al. (2021). Using Multiple Linear Regression (MLR) and Lasso regression methods, as well as Support Vector Regression (SVR), the model was reduced in complexity and accuracy by applying Principal Component Analysis (PCA). Pre-processing was performed on the models by imputing missing values using mean values and feature scaling. This study demonstrated that the tuned SVR model outperformed the MLR model for predicting precipitation by using lasso regression in the MLR model and the RBF model in the SVR model to minimize the non-linear relationship.

Using Multiple Linear Regression (MLR) and Artificial Neural Network (ANN) models, Jinashree et al. (2021) conducted a comparison study to determine the amount of precipitation. In addition to missing value imputation and feature scaling, the researchers employed the same preprocessing methods as Gomathy et al. The researchers evaluated all proposed models and concluded that MLR using Lasso regression had the most significant degree of accuracy. The authors of Jinashree et al. (2021) did not perform feature selection before the model's development. They utilized the Lasso regression method for MLR and the backpropagation method for ANN, in contrast to Gomathy et al. (2021).

2.5.3.3 Evaluation Metrics

To assess the performance of predictive models, additional evaluation metrics are available, including RMSE, MAE, Correlation, and R Square, as well as precision, recall, F1 score, and ROC curve (Receiver Operating Characteristics). Binary classification problems generally involve precision and recall, and the F1 score is a harmonic mean of these two variables. For the regression predictive model, the metrics such as MAE (Mean Absolute Error), RMSE (Root Mean Square Error), R Square (Coefficient of determination), and R (Correlation) values are widely applied. The evaluation metrics should be chosen according to the specific problem being addressed, and the results should be interpreted carefully to avoid overfitting or underfitting the model.

Table 1: The summary table for related work

No.	Researcher	Objective	Dataset	ML Model	Result	Findings and Limitations
1.	Bagirov et al. (2017)	To develop CLR (Cluster-wise Linear Regression) algorithm and compare with a popular regression model	Monthly rainfall dataset (1889-2014) from Victoria Stations, (6) variables	ANN, SVM, CLR, MLR	ANN RMSE - 30.2, MAE - 22.4, MASE - 0.7, CE - 0.3 SVM RMSE - 31.0, MAE - 22.4, MASE - 0.7, CE - 0.2 CLR RMSE - 27.0, MAE - 19.4, MASE - 0.6, CE - 0.4 MLR RMSE - 29.4, MAE - 22.5, MASE - 0.7, CE - 0.3	The CLR outperforms than popular regression models. The study use minimum and maximum temperature, evaporation, vapor pressure and solar radiations. The inclusion of evaporation, vapor pressure and radiation helped slightly improve in some models.
2.	Qiu et al. (2017)	To propose a Multitask Convolutional Neural Network to extract features from time series data automatically and predict rainfall amount	Daily rainfall datasets from 2002 to 2015 form Manizales city and Guangdong, (17) variables	QPF, LR, MLR, MT-MLP, RNN, MT-RNN, CNN, MT-CNN	QPF MSE - 15.55, CSI - 0.866, Correlation: 0.399 LR MSE - 13.28, CSI - 0.875, Correlation: 0.469 MLR MSE - 11.966, CSI - 0.711, Correlation: 0.472 MT-MLP MSE - 11.894, CSI - 0.824, Correlation: 0.473 RNN MSE - 11.809, CSI - 0.881, Correlation: 0.480 MT-RNN MSE - 11.801, CSI - 0.884, Correlation: 0.488 CNN	The proposed model incorporation with CNN network can predict the raining case with precise accuracy. The study used the observations from sensors. Their future work would be the incorporation of weather variables with radar images, more useful data sources such as topological information and cloud graph data and other data that could improve the prediction.

					MSE - 15.55, CSI - 0.866, Correlation: 0.399 MT-CNN MSE - 11.253, CSI - 0.872, Correlation: 0.521	
3.	Aswin et al. (2018)	To obtain accurate monthly rainfall prediction models using Deep Learning Architectures	Monthly rainfall time-series dataset from 1979 to 2018 from The Global Precipitation Climatology Project (GPCP), (2) variables	LSTM, CNN	LSTM RMSE - 2.55, MAPE - 1.679 CNN RMSE - 2.44, MAPE - 1.728	The study observed that both CNN and LSTM yielded similar result and performed well for predicting the rainfall. The data used in this study is from several location around the world. Their study was not area specific and would be applied the similar techniques to an individual region for rainfall prediction.
4.	Tharun et al. (2018)	To predict the rainfall intensity using regression techniques and statistical modelling	Daily rainfall dataset from 2005 to 2014 from Coonoor city, India, (10) variables	SVR, DT, RF	SVR (Poly) RMSE - 8.290, R^2 - 0.649 SVR (RBF) RMSE- 7.422, R^2 - 0.814 DT (MAE) RMSE - 6.410, R^2 - 0.899 DT (MSE) RMSE - 6.915, R^2 - 0.904 RF (10 estimator) RMSE - 5.556, R^2 - 0.965 RF (100 estimator) RMSE- 5.228, R^2 - 0.979	The best model in this study is Random Forest. In SVR, the Radial Basic kernel performed better than other kernels. The data used in this study is the common traditional weather data. The authors wish to include more useful variables and suggested to implement non-linear machine learning algorithms such as neural networks for rainfall prediction.
5.	Dash et al. (2018)	To forecast seasonal rainfall using artificial intelligence approaches	Monthly rainfall dataset from 1871 to 2016 from Karala state, India, (8) variables	KNN, ANN, ELM	KNN RMSE - 17.079, MAE - 12.068, MASE - 0.248, PP - 0.653, R - 0.935 ANN	The study discovered that the accuracy of the model is greatly affected by the number nodes in hidden layers. The best technique in this study is ELM. The author recommended to

					RMSE - 11.542, MAE - 9.514, MASE - 0.195, PP - 0.753, R - 0.967 ELM RMSE - 6.000, MAE - 3.149, MASE - 0.053, PP - 0.912, R - 0.0.998	utilize different techniques for southern India and should consider other data sources to improve the variability of the rainfall prediction model.
6.	Aftab et al. (2018)	To predict rainfall today, either “Rain” or “No Rain”	Lahore’s 12 years rainfall dataset (2005-2017), (10) variables	SVM, NB, KNN, DT, MLP	SVM Precision - 0.919, Recall - 1, F1 - 0.958 NB Precision - 0.936, Recall - 0.954, F1 - 0.945 KNN Precision - 0.942, Recall - 0.943, F1 - 0.943 DT Precision - 0.939, Recall - 0.981, F1 - 0.960 MLP Precision - 0.94, Recall - .984, F1 - 0.96	The techniques used in this study performing generally very well in terms of accuracy. However, class balancing was not carried out during pre-processing. The author advised to explore other different algorithms and use more climatic attributes in the model to increase the performance of the classification model.
7.	Pham et al. (2020)	To develop and compare daily rainfall prediction models	Daily rainfall dataset from 2004 to 2013 from Hoa Binh province, Vietnam, (5) variables	ANFIS, ANN, SVM	ANFIS MAE - 3.281, R - 0.844 ANN MAE - 3.209, R - 0.829 SVM MAE - 2.728, R - 0.863	The author concluded that the algorithms used in this study especially SVM performed better than previous studies. The model can perform according to the seasonal variety of the region. However, the variables used in this study is not so relevant for prediction of 24-hour precipitation. It is required to identify more suitable variable set to improve the model performance.

8.	Sureh et al. (2019)	To estimate monthly precipitation using two different combinations of input variables	Monthly rainfall dataset form 1986 to 2017, (7) variables	SVR, KNN	<p>SVR(Poly) RMSE - 15.08, MAE - 7.02, R - 0.75</p> <p>SVR (Normalized Poly) RMSE - 14.58, MAE - 6.73, R - 0.78</p> <p>SVR (Pearson) RMSE - 14.98, MAE - 7.38, R - 0.76</p> <p>K-NN(Euclidean) RMSE - 36.83, MAE - 16.13, R - 0.24</p> <p>K-NN(Chebyshev) RMSE - 28.61, MAE - 13.85, R - 0.31</p> <p>K-NN(Manhattan) RMSE - 34.66, MAE - 14.76, R - 0.32</p>	The study concluded that the best result was obtained from the SVR using normalized poly kernel function. The algorithms are modelled using different combination of independent variables. The data used are traditional weather data. The author advocated to experiment with alternatives data to increase the accuracy of the prediction models.
9.	Hartigan et al. (2020)	To identify temperature and rainfall trends in Sydney	Time series dataset (1957-2019) from Sydney catchment area, (11) variables	MLR, SVM, RF	<p>MLR RMSE - 764.971, R² - 0.663</p> <p>SVR (RBF) RMSE - 651.487, R²- 0.579</p> <p>SVR (Poly) RMSE - 613.704, R²- 0.848</p> <p>RF RMSE - 780.455, R²- 0.387</p>	The model used in this study could highlight the effect of DMI, EMSO and SST anomalies on the annual rainfall prediction. However, the model did not perform well in autumn and winter. They just used the traditional weather variables, and the variance of the models are relatively lower in most of the models.
10.	Kumar Misra et al. (2020)	To forecast rainfall for an average rainy season in Odisha, India	Annual rainfall dataset from 1901 to 2018 from Indian Metrological	LRA, RF, ANN	<p>LRA MAE - 3.8, Acc - 87.28 %</p> <p>RF MAE - 2.3, Acc - 91 %</p> <p>ANN</p>	The researchers deduced that the Random Forest model is the most viable for the prediction of weather variables such as rainfall . The data used in this

			Department, (5) variables		MAE - 3.2, Acc - 89.64 %	study only focus on rainy season in the area. The variability of the model could be improved.
11.	Gomathy et al. (2021)	To predict precipitation using machine learning techniques	Monthly rainfall dataset from 1901 to 2015 from each state of India, (19) variables	MLR, SVR	MLR (Base) MAE - 11.02 MLR (LR) RMSE - 11.78 SVR (RBF) RMSE - 4.31	The study observed that the SVR is better than MLR due to the non-linear relationship between the traditional climate data and the rainfall variable. The Radial Basic Function provided the best result.
12.	Jinashree et al. (2021)	To compare regression machine learning algorithms to estimate rainfall	Monthly rainfall dataset from 1900 to 2020 from several regions of India	MLR, ANN	MLR(Base) ACC - 94.05, RMSE - 37.64 MLR(LR) ACC - 93.25, RMSE - 38.52 ANN(Base) ACC - 86.80, RMSE - 45.84 ANN(BP) ACC - 53.80, RMSE - 75.89	The result of the study demonstrated that the LASSO regression outperformed than ANN model. The dataset used in this study is from different location of India rather than area specific. The author offered to include more complexity and reduce the splitting ratio for training dataset.
13.	Raval et al. (2021)	To construct a neural network prediction model and compare the performance with existing models	Daily rainfall dataset from 49 cities in Australia (2007-2017), (24) variables	LR, LDA, QDA, KNN, DT, XGBoost, RF, NB, NN	LR Precision - 97.14, F1- 86.87 LDA Precision - 72.36, F1- 78.29 QDA Precision - 63.70, F1- 77.25 KNN Precision - 77.84, F1- 81.24	The study concluded that a simple deep learning model outperforms than a group of statical models such as logistic regression model. The reason is due to the non-linear nature of the rainfall pattern with the climate variable. The prediction accuracy could be further improved by using advanced algorithms and inclusion of alternatives data.

					DT Precision - 95.20, F1-86.22 XGBoost Precision - 84.95, F1-84.01 RF Precision - 97.06, F1-86.60 NB Precision - 47.72, F1-58.79 NN Precision - 98.26, F1-88.61	
14.	Ganesh et al. (2021)	To predict whether it will rain or not tomorrow	Daily rainfall dataset (2015-2018) from many cities in India, (10) variables	RF, LR	RF Accuracy - 94.4 % LR Accuracy - 95.9%	The scholar drew a final conclusion that although the accuracy seem higher in both algorithms, the model has lower performance in predicting no rain class. The possible reasons are that the dataset does not contain weather variable which could help in predicting the target variable and the data quality issues in the dataset. The class balancing was not conducted.
15.	Sarasa-Cabezuelo (2022)	To predict today rainfall	Daily rainfall dataset from 49 cities in Australia (2007-2017), (24) variables	KNN, DT, RF, NN	KNN Misclassification rate - 0.044, MSE - 0.042, Accuracy - 0.5 DT	The techniques used in this study provided the same misclassification rate for prediction of rainfall. Therefore, another metric was used that it was revealed that

					<p>Misclassification rate - 0.044, MSE - 0.043, Accuracy - 0.6</p> <p>RF</p> <p>Misclassification rate - 0.044, MSE - 0.041, Accuracy - 0.65</p> <p>NN</p> <p>Misclassification rate - 0.044, MSE - 0.039, Accuracy - 0.7</p>	<p>the Neural Network model perform the best because of the non-relationship nature of rainfall pattern. The data used are traditional weather data. The author recommended to use the data from other regions and try to include other variables to improve accuracy.</p>
16.	Hudnurkar & Rayavarapu (2022)	To classify 'rain' and 'non-rainy' day for summer monsoon season	Daily rainfall dataset (2000-2018) from Maharashtra, India, (6) variables	SVM, ANN	<p>SVM (Poly)</p> <p>Accuracy - 0.779, Precision - 0.779, F1-0.761, misclassification rate - 0.22</p> <p>SVM (RBF)</p> <p>Accuracy - 0.767, Precision - 0.758, F1-0.618, misclassification rate - 0.233</p> <p>ANN</p> <p>Accuracy - 0.755, Precision - 0.71, F1-0.62, misclassification rate - 0.24</p>	<p>The comparison of three models such as SVM with Radial Basic Function and Poly kernel and ANN model are quite comparable and Poly one slightly performs better than others. The variables used in this study are the traditional weather data. The accuracy could be further improved by incorporating other informative variables.</p>

2.6 Summary of related work

A summary of recent rainfall prediction literatures is presented in Table (1). The researchers are listed in the table, along with their objectives, the datasets and number of features, algorithms implemented, evaluation metrics and their finding and limitations.

Based on the literature, the following conclusions can be drawn.

- Machine learning algorithms commonly used to predict rainfall include Random Forest, Multiple Linear Regression, Artificial Neural Networks, and Support Vector Regressor models. When compared with other machine learning algorithms, all of these techniques mentioned produced promising results in many studies.
- There are no benchmark datasets used in rainfall forecasting; all datasets are based on the location; some are of aggregated, and some are specific to the region. A maximum of 24 variables are used in the studies.
- Predictive models are greatly influenced by the input features and optimization parameters used in training the model.
- Many researchers suggested that the accuracy of the prediction model could be improved using alternative data, however, most of the study use traditional weather data and no study has been conducted in the reviewed journals that use the incorporating of the hydrological data in their study.
- Model performance is typically measured by Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Correlation (R) and the Coefficient of determination or variance of the model (R^2).

2.7 Implementation method

In order to implement the whole project, Python programming language will be used. Python is chosen because of its simplicity, flexibility, and the availability of a wide range of frameworks and library when it comes to data science and machine learning compared to other popular languages like R programming. Below are some of the advantages of Python language.

Easy to learn and use: The Python is simple to use especially for the beginners because of its English-like syntax and easy to implement.

Versatile: It is the open-source and general-purpose language that can be used with free of charge in many cases and tasks such as data science, web development, machine learning and artificial intelligence.

Rich Ecosystem: Python offers a wide variety of packages and frameworks such as Pandas, NumPy, Matplotlib, Seaborn, PyTorch, Tensorflow, Keras, Scikit-learn which

supports for conducting data science and machine learning workflow. The web-based application such as Jupyter Notebook and Jupyter Lab also allow to organize the codes, visualize the data, share to other team members, and can be used for reproduce purposes, which makes beneficial for the data scientists.

Easy to deploy: Python is also very versatile in supporting different deployment options ranging from simple project to large scale deployment and production. It also offers cloud services, containers, web servers, making the machine learning model to deploy with easy and easily accessible by the end-users.

Limitation: The advantages outweigh the limitations of the Python in several ways. The only issue with the Python is the performance issued. As it is structured in interpreted language, it can be slower compared to other compiled languages such as Java and C++. Other than that, Python is the most popular programming language in the field of data Science.

Due to the above reasons, this project employs Python for starting with data exploration to the final process of deployment. The details use of libraries and the summary of the tools for experimentation stage will be provided in the next chapter.

2.8 The application of selected models in different domains

Based on the findings from the literature review, the three algorithms selected are Random Forest model, Support Vector Regressor and Artificial Neural Network. Therefore, the applications and usefulness of these model in the field of meteorology as well as in other domains are reviews in this section. The technical details and the implementation method for these models will be provided in detail in the modelling section of Chapter (3).

2.8.1 Radom Forest Model

A Random Forest is a well-known machine learning technique that is widely used in classification, regression, and feature selection fields. Using many decision trees together, it classifies or predicts data points by combining their decisions. Many benefits are associated with this method, such as robustness, accuracy, feature selection, scalability, and interpretability.

Random Forest is used in a number of fields. It is used in the financial industry for identifying fraudulent activity, rating credit, and approving loans. Healthcare professionals use Random Forest to diagnose diseases, predict patient outcomes, and develop new medications. In the retail industry, Random Forests are used for segmenting customers, recommending products, and forecasting demand. There are a number of marketing disciplines that benefit

from the use of Random Forests, including customer segmentation, response prediction, and campaign optimization. Agricultural applications of Random Forest include predicting crop yields, analyzing soil quality, and identifying pests. Meteorology uses it for a variety of purposes such as forecasting weather, modeling climate, predicting severe weather phenomena, and ensuring that meteorological data is of a high quality.

2.8.2 Support Vector Regressor (SVR)

The Support Vector Regressor is one of the most widely used machine learning techniques that can be applied to regression tasks in a variety of different fields. It locates a hyperplane that maximizes the margin between the training data and the projected values, by finding the best point on that hyperplane which maximizes the margin between the training data and projected values.

In the world of finance, the SVR algorithm is used to perform tasks such as forecasting stock prices, managing risks, and optimizing portfolios in order to increase profitability. It is used in the field of medicine for purposes, such as identifying individuals who are likely to acquire diseases, and in predicting the outcome of a particular treatment in relation to those individuals. Furthermore, it is capable of analyzing medical photos and detecting abnormalities in them. There are several ways in which SVR is used in the retail industry, including predicting the lifetime value of customers, optimizing pricing strategies, and forecasting demand for products and services. It is used in marketing to forecast the actions of customers, to identify those who are most likely to defect, and to put them in groups based on their preferences. It also analyzes the data to identify trends or patterns. There are several ways in which SVR can be used in the field of agriculture, including prediction of crop yield, identification of areas that need more irrigation or fertilizer, and analysis of soil data pertaining to soil fertility. As a field of meteorology, it involves the prediction of future weather conditions, the identification of areas at risk of experiencing extreme weather events, and the analysis of climate data to reveal patterns that have developed over time.

2.8.3 Artificial Neural Network (ANN)

Artificial Neural Networks are well-known techniques used for recognizing patterns, analyzing regressions, and classifying data in a wide variety of fields. Taking inspiration from the biological neural networks of the human brain, it mimics the behavior of neurons. One of the benefits of artificial neural networks is the ability to learn from large datasets with high

accuracy and robustness, as well as the ability to manage complex interactions between variables. Furthermore, ANNs can also handle nonlinear interactions between variables, which makes them suitable for a wide range of applications.

In finance, ANNs are used to predict stock prices, identify fraudulent activity, and rate creditworthiness. The use of artificial neural networks in the medical field is widely used for diagnosing diseases, predicting patient outcomes, and developing new drugs. In the retail sector, the ANN is utilized to segment clients, recommend products, and forecast demand. In the marketing industry, ANNs are used for segmenting consumers, predicting response rates, and optimizing campaigns. ANNs are used in the agricultural industry for predicting crop yields, analyzing soil quality, and detecting pests to ensure that crops are not harmed by pests. In the field of meteorology, artificial neural networks (ANNs) can be used to forecast weather, model climate conditions, predict future weather conditions based on data from previous patterns, and locate regions that are at risk of being affected by extreme weather events based on past patterns.

2.9 Summary of literature review

According to an analysis of the literature, rainfall prediction is widely used in a wide variety of areas. In order to obtain accurate rainfall predictions, a variety of algorithms, techniques, and concepts have been utilized. Research has led to a number of significant breakthroughs in the field of rainfall prediction, and the models proposed by researchers continue to show positive results. Many researchers recommended including more alternative data to help improve the performance of the model, nonetheless, none of them has been experimented with hydrological data in the rainfall prediction model. With the help of machine learning, this project aims to build a rainfall prediction model which could provide a higher degree of accuracy for rainfall prediction in Taunggyi city using both traditional data as well as hydrological data.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter aims to provide a brief overview of the workflow and methodology for completing the capstone project, with a brief description of each process. It will cover a wide range of topics, including the dataset's description, theoretical and conceptual details relevant to the models. In addition, it will present the approaches used to train the rainfall prediction model. This chapter will also include a subsection that focuses on data preprocessing, feature engineering including feature selection and feature creation, evaluation metrics to compare the model, and process of deployment for the best performing model. Furthermore, the detailed explanation of each step of the experimentation and implementation, as well as justifications for the selected approaches are also included as part of the chapter.

3.2 Research Approach and Methodology

This study adopted the SEMMA methodology (Sample, Explore, Modify, Model, and Assess) to conduct the entire capstone project as shown in Figure (8). SEMMA provides a straightforward, easy-to-follow process for developing projects relating to the exploitation of information. Depending on the project requirements, the SEMMA methodology allows iteration at each step (Palacios et al., 2017).

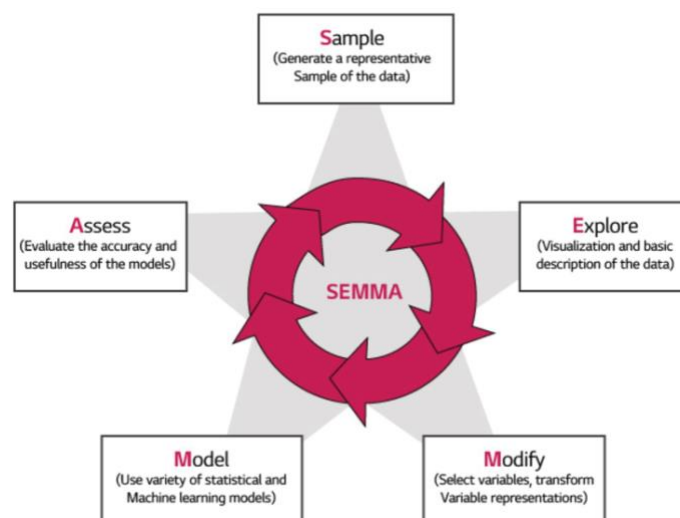


Figure 8: SEMMA Methodology Flow Chart (Palacios et al., 2017)

Below are the six phrases of the SEMMA methodology.

Sample: SEMMA methodology begins with sampling, which is creating a subset of data from a large data source to conduct preliminary analyses and it is optional to complete this step. The essence of this step is to define both independent variables and dependent variables to be use in the whole project.

Explore: This step involves a bivariate and univariate analysis of the variables to identify gaps and analyze the associations between the variables in the dataset. Bivariate analysis is used to see the relationship of two variables while univariate analysis explores each factor individually to comprehend more about the data using visualizations and plots.

Modify: In this stage, the data are pre-processed in accordance with the information discovered from the analyses of the variables, such as cleansing the data, transforming, handling missing values, and imposing data imputation, so that the data are ready for the next stage of model building.

Modeling: During the modeling stage, a variety of data mining techniques is used to develop the models based on the data that has been refined and cleaned in the modify stage in order to achieve the final desired result through the use of the final desired results from the data mining tools.

Assess: SEMMA methodology ends with this phrase. This phrase refers to the assessment of the model's usefulness and credibility in relation to the study's topic. Moreover, the quality of the data can also be observed at this stage (Palacios et al., 2017).

In order to complete this capstone project, the SEMMA methodology was chosen mainly due to the fact that this is a quantitative project that examines the rainfalls' patterns and information. This project is going to be conducted according to the proposed methodology shown in Figure (9). It is based on the SEMMA methodology and is composed of six main stages as described in blocks (highlighted in light green are from the SEMMA methodology). Using the purposed methodology, the problem is identified, the data will be collected (Sample), explored (Explore), prepared for further pre-processing (Modify), modeled (Model), evaluated and selected (Assess), and finally deployed the model. Following is a detailed description of each stage.

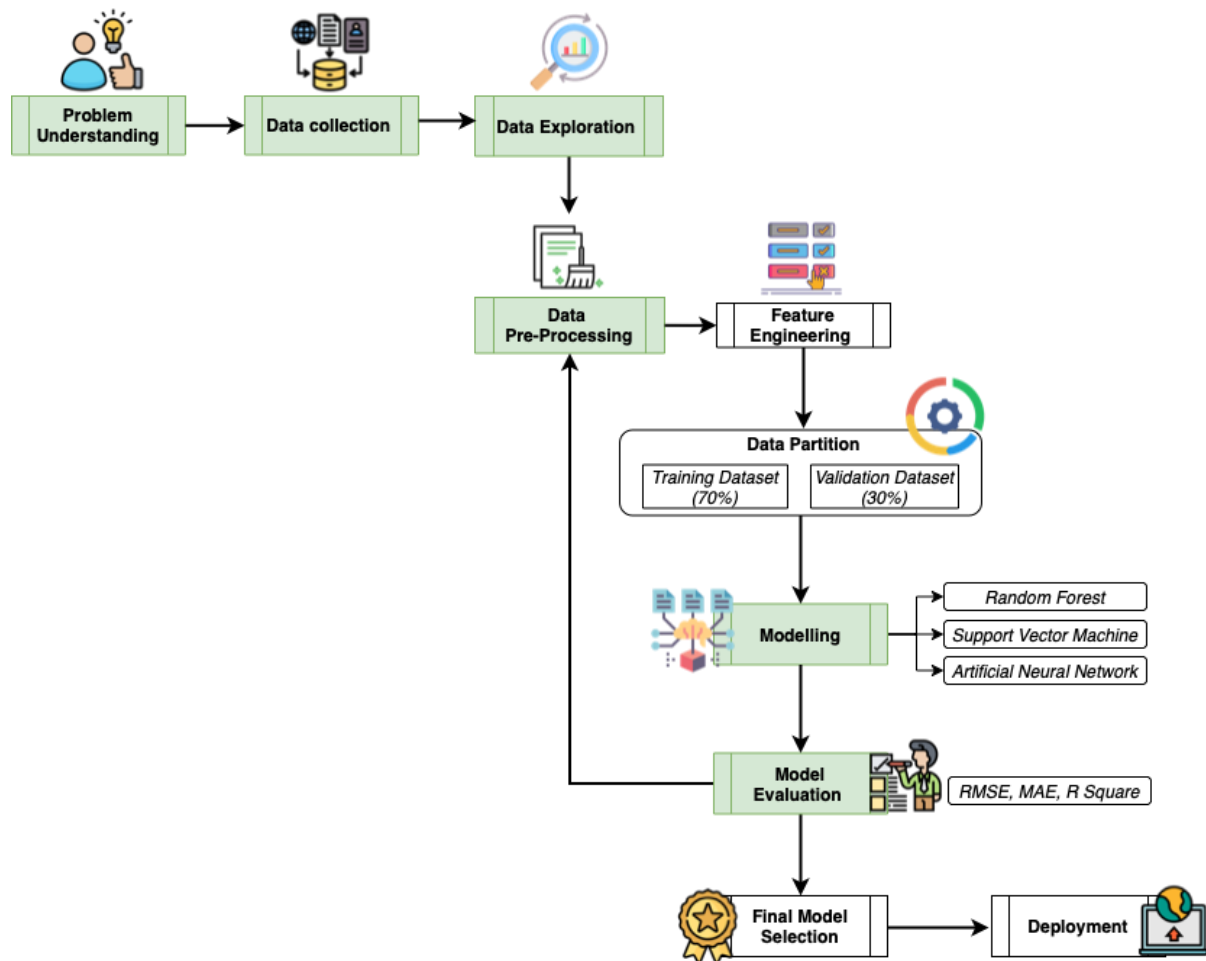


Figure 9: Proposed Methodology for conducting this project

3.2.1 Problem Understanding

It is at this stage that the emphasis is on establishing the field of research that needs to be addressed and the research problem that will need to be addressed, for the purpose of this case, it is to get prediction of rainfall for Taunggyi using both traditional weather data and hydrological data. At this stage, a problem statement, a set of research questions, an aim and objectives, and a scope of the research are all defined. A literature review is also conducted in order to gain a comprehensive understanding of the literature related to developing rainfall prediction models. It also explores the techniques and approaches to be used to develop the models, the type of dataset and features, the metrics for the evaluation purposes, and the suitable methodology to be used in the project.

3.2.2 Data Collection (SAMPLE)

This project uses the daily rainfall data of Taunggyi city. Obtaining the data was done with the kind approval of Meteoblue® weather for the purpose of academic studies and can be accessed from the following link: "<http://www.meteoblue.com/en/historyplus>". As far as the type of dataset is concerned, it is simulation rainfall data. There are 37 years of daily weather records in the dataset starting from the year 1985 till 2021. In total, the dataset contains 13,514 observations containing 15 features which are traditional weather variables such as wind speed and direction, temperature, relative humidity, cloud coverage, the hour of sunshine duration, along with hydrological data variables such as soil moisture and evapotranspiration. The climate data and hydrological data are the independent variables, and the dependent variable is a measure of rainfall in this project.

3.2.2.1 Dataset Description

In Table (2), the metadata of the dataset is presented with detail explanation including, the name of the variable in the dataset, their types, the unit of measurement for each variable, and the sample values of the features. There are 15 variables in the dataset and most of the variables are continuous data types except the date variable which contains the string values.

Table 2: The metadata or descriptions of the dataset

No.	Variable	Data Type	Unit	Description	Sample
1.	date	String	-	The observed local timestamp (UTC +6.5) in 24-hour format	19850101T0000 / 20211231T0000
2.	temperature	Continuous	Degree Celsius	The standard temperature measured at two-meter elevation with three variables of minimum, maximum and mean values	21.23 / 20.64
3.	sun_dur	Continuous	Minutes	Total sunshine duration during the whole day	581.98 / 576.65
4.	shortwave_rad	Continuous	Watts per Square Meter	The amount of total radiant energy generated by sun, which reaches the surface of the earth	1393.04 / 4733.91
5.	r_humidity	Continuous	Percentage	The percentage of water in the air relative to the amount water which is required to be saturated at the same temperature.	99 / 92
6.	cloud_covr	Continuous	Percentage	The percentage of cloud covered in the sky.	37.1 / 20.18
7.	pressure	Continuous	Hecto Pascals	The average pressure of the location at mean sea level	1017.3 / 1018.4
8.	evapotrans	Continuous	Millimeter	The total amount of water which involves in the process of water transferring from the land into the atmosphere by mean of evaporation .	2.64 / 2.93
9.	soil_temp	Continuous	Degree Celsius	The temperate of the soil.	18.86 / 11.78
10.	soil_moisture	Continuous	Cubic Meter	The amount of moisture content in the soil.	0.29 / 0.26
11.	vapor_pressure	Continuous	Hecto Pascals	The difference in pressure between air vapor and the saturation point of the air.	11.92 / 12.41
12.	wind_speed	Continuous	Meters per Second	The speed of the wind that is measured at (10) meters above the ground.	2.67 / 0.22
13.	wind_dir	Continuous	Degrees	The direction of the wind measured at (10) meters (Wind blows from 360 degree means from North, 90 degree means from East, etc.)	289.26 / 251.6
14.	wind_gust	Continuous	Meters per Second	The speed of the wind that increases very briefly, usually for less than twenty seconds.	4.8 / 0.5
15.	rainfall	Continuous	Millimeter	The total amount of daily rainfall in millimeter (mm).	0.40 / 8.50

3.2.3 Data Exploration (EXPLORE)

This stage involves the examination of all the data for exploratory purposes, and it is one of the most important stages as it helps us gain a deeper understanding of the data in hand. This stage begins with the shape of the dataset in terms of number of observations and number of variables, the types of the variable. The features in the dataset in terms of visualizations, descriptive statistics, distributions, anomaly detection, and a discussion and summary of the features will be provided in detail. The variables are analyzed using the univariate approach for analyzing a single variable and the bivariate approach for understanding the relationship between two variables. In bivariate analysis, Pearson's correlation and Spearman Correlation will be used to determine how the variables are correlated to each other either in linear or non-linear manner. The variables are also closely examined for missing values in the data, zero variance variables, their skewness, duplicate data, and outliers through boxplot and Z-score analysis. To explore all the features within the dataset, this project will use Python packages such as Pandas, NumPy, Seaborn, Matplotlib, and Missingno libraries for exploratory data analysis.

3.2.4 Data Pre-processing (MODIFY)

At this stage, the data is manipulated and preprocessed, including handling missing values and transforming the data based on the findings from the data exploration phase. In order to build effective models, one of the most essential steps that needs to be carried out before modelling is the preparation of the cleaned data due to the fact that high quality data will be able to provide more accurate results. Therefore, the model will be able to perform better as a result of having high quality data. As soon as the exploratory phase of the analysis has been completed and the missing values and anomalies have been mapped out, the next step is to address these issues before moving on to model development. The data preprocessing in this project is carried out using Pandas package in Google Colab environment.

3.2.4.1 Handling Missing Values

When there are missing values, it can result in a reduction of the accuracy of the model as well as its performance. In most cases, missing values are replaced with the mean, or median value based on the type of feature and the context. All variables in this project are continuous data types, and the dataset does not contain any categorical variables. With the help of the bar plot from the Missingno library, missing values will be identified for each variable. The variables which contain missing values in more than 30% of the data are regarded not suitable

for modelling and thus, they should be removed. Missing values can be dealt with by deleting the entire observation easily. This approach may be effective if there are very few missing values or if the dataset consists of a large number of observations. Further, removing missing values from independent variables could result in unreliable models, since information is lost in the data. A missing observation will, however, be removed if the rainfall (dependent variable) contains any of those values since this could bias the model result. However, zero values are not considered as missing values since zero in the rainfall variable mean no rain on that day and hence, they carry meaningful information and will be kept as they are.

3.2.4.2 Handling Anomalies

The anomalies in the dataset could be zero variance variables, duplicate data, outliers or extreme values, invalid data, and highly skewed variables. The description and the approaches to handle them are expressed in detail as below.

Zero variance variables: Zero variance variables or the constant variable means that the variable is having the same values for all the observations, in other words, it has non variability and thus it cannot provide any useful information for prediction model (Shi et al., 2018). The analysis gained from the descriptive statics and visualizing the distribution of the variables, the zero variance variables will be identified. Since these variables cannot provide any values and could potentially bring issues and instability to the prediction model, should these variables found in the dataset, these will be dropped.

Duplicate data: It is common to see in the dataset that have duplicate data because of error in data entry, integration of the data using several sources. As they are the same observation, they could bring inaccuracies and as a result could decrease the performance of the model. In this project this duplicate data will be detected using the conditional statement with Pandas library and they duplicate data will be removed from the dataset.

Outliers or extreme values: Another common type of anomaly found in most of the dataset is the existence of outliers or extreme values in the variables. They are basically data points which are significantly different from most of the data in the dataset. There are many reasons which might cause the outlier in the dataset. They can be due to data entry error, measurement or observational. The handling of the outliers is important since sometimes even though the data point is far away from the other values, based on the objectives of the analysis or research

they could be useful as well. In this project, the outlier will be identified using visualization method such as the boxplot and statistical method such as Z-score analysis. However, simply deleting the outliers can arise implications as well. As this project will predict the rainfall events including no rainfall amount and extreme rainfall amount to get early warning, the outlier will be closely analyzed and if they seem not useful for the model, they will be dropped. Otherwise, they will be kept in the dataset. They will be substituted by using the mean or median value of the feature.

Invalid data: This type of anomaly is sometimes overlooked as they are not easily detectable unless they are closely inspected. The invalid data are data points they are not conformed with expected data type or format. It is essential to check them as they could cause inaccurate result in model performance. If the invalid data are found in the data, they will be imputed with relevant values such as equivalent value or mean, or mode value based on the context.

Highly skewed data: This refers to the distribution of the data is highly concentrated in a range of area. The variable can be considered highly skewed if its skewness is over -2 and +2 (Hair et al., 2010). Skewness is the measurement of the asymmetry of the distribution, and it can affect the performance of the model if it is highly skewed. The skewness values can be easily generated using skew function in Padas library. If the independent variable is found to be highly skew, it will be transformed using log plus one transformation since this transformation can handle although the variable contains zero values. However, the transformation of dependent variable is not necessary if the data are in non-linear relationship because it does not require to comply with linearity assumption.

3.2.4.3 Feature Engineering

Feature engineering in this study will be employed feature selection and feature creation method. They are described as below.

Feature selection: This step is part of the data pre-processing phrase, which involves removing variables of low importance from the dataset. Through the use of this approach, dimensionality of features could be reduced and only the most significant features would be retained in the dataset. Two different methods of feature selection will be implemented in this project, namely, Correlation Analysis and a Tree-based method using Random Forest. The correlation matrices will be used to identify strongly correlated variables. The correlation values of 0.8 and above

are considered to be highly correlated and one of the variables will be dropped from the dataset. The reason is that since they are strongly associated, they are just redundant data and, in most cases, they could lead to instability of the model performance. Then followed by Tree-based method using Random Forest to map out the feature importance of each variable and the variables showing very low feature importance scores will be eliminated from the dataset.

Feature creation: It is also a phase in data pre-processing in which new features are created or extracted using the existing feature in the dataset to capture information efficiently and thereby improve the model's performance. The primary aim of feature creation is to obtain the features which would increase the predictive ability for the target variable. There are several methods to carry out feature creation such as mathematical transformation, one-hot encoding, label encoding, aggregation, interaction terms in accordance with the nature of the features. This project will carry out feature creation accordingly using suitable method mentioned before based on the data.

3.2.4.4 Feature Scaling

Before initiation of the training phase, it is necessary to scale the data as a pre-processing step to improve performance of the model while saving computational costs. Since this project involves deploying Artificial Neural Network and Support Vector Machine, it is required to implement feature scaling before building the model (Dash et al., 2018). In this project, the Standard Scaler from the Scikit-learn library will be used since it is less sensitive to outlier. The process of standard scalar is to scale the data into zero mean and unit variance based on the standard deviation. In addition, due to the fact that the data used in this project involved natural process which could have unusual event with extreme condition or outliers, the Standard Scaler is preferable than other feature scaling method.

3.2.4.5 Data Partition

In order to model the data, the last step is to separate the dataset into two subsets, the training dataset and the testing dataset. It is going to be partitioned based on a 70:30 ratio for the dataset. As such, 70% of the dataset is for training the rainfall prediction model, while 30% is for evaluating the model. This split is being carried out in order to be able to validate the accuracy as well as the performance of the model that has been built on the training dataset. All of these steps will be performed by utilizing the `train_test_split` function from the Sklearn package, which is implemented using Python.

3.2.5 Modelling (MODEL)

During this stage, different modelling techniques are used to develop models based on the cleaned data in order to get the optimal prediction. The project is at its most critical stage at this point. In this stage, the proposed models for the prediction of rainfall are developed. There are a number of machine learning algorithms that have shown to be most effective in solving rainfall estimation problems, as evidenced in the literature review. These algorithms will be utilized in the current study. The algorithms that have been selected for this study are Random Forest, Support Vector Machine, and Artificial Neural Networks. Furthermore, a nonlinear relationship exists in the rainfall patterns, which is one of the reasons why these techniques are implemented in order to predict rainfall. There has been considerable discussion about the suitability of Support Vector Regressor, and the use of Artificial Neural Networks for addressing this type of data pattern (Sureh et al., 2019). There is also sufficient data in this dataset which is suitable for training an artificial neural network. In this study, the base and tuned model for all techniques will be constructed. In addition, the model using only traditional weather data and the model with additional hydrological data are built separately and compare their performance. The techniques that will be used in this study are discussed in more detail as follows.

3.2.5.1 Random Forest

A random forest algorithm is one of the most famous machine learning algorithms, which is also called an ensemble model due to the fact that it can estimate the outcome of a model using more than one decision tree. In the industry, random forest is the most useful algorithm for solving both classification and regression problems because it performs well in both of these areas. It is based on a majority rule principle, which means that every decision tree will produce a prediction, and the one with the highest decision will be selected (Tharun et al., 2018). The figure (10), for instance, illustrates this.

During the training period of a Decision Tree, it is common for the model to become overfitted over time. In spite of this, multi-tree decision trees are able to predict the same outcome in a better way and can help prevent overfitting in the future. The model will be built using the RandomForestRegressor from the Sklearn library. The initial settings to build the base model are 10 for maximum depth, 5 for minimum sample leaf, 10 for minimum sample leaf split, 150 for number of estimators, and finally the random state is set to 42. Then the model will be tuned with Random Search method using RandomizedSearchCV incorporating

Ten-fold cross validation in the function to ensure the model performs consistently with unseen data and avoid overfitting.

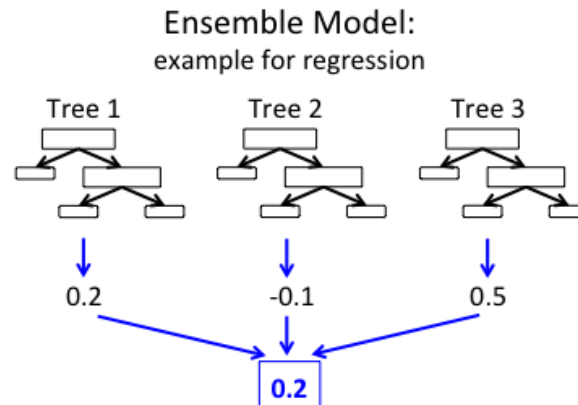


Figure 10: Illustration of Random Forest Model (Tharun et al., 2018)

3.2.5.2 Support Vector Machine

The Support Vector Machine was created by Cortes and Vapnik in 1995, incorporating the idea of statistical learning theory in order to analyze data for classification as well as regression problem in real life scenario. This algorithm is based on the N-dimensional data, and its objective is to come up with a hyper plane that makes a clear distinction between each pair of data points which yields a maximum distance to separate between the two data as shown in Figure (11). An SVM has an ability to determine both the position as well as the orientation of data based upon the support vectors. The support vectors are those vectors which are close to the hyperplane. A kernel K is rendered in a feature space as a dot product of a hyperplane. A cost function is used in maximizing the separation between a set of data points along with the hyperplane in order to tackle the regression problem (Sureh et al., 2019). The model will be built using the SVR from the Sklearn library. The initial settings to build the base model are used with default values and the random state is set to 42. Next, the model will be tuned using Random Search method utilizing RandomizedSearchCV function that incorporates a Ten-fold cross validation procedure to keep the model performing consistently even with unknown data sets and prevent the model for being overfit.

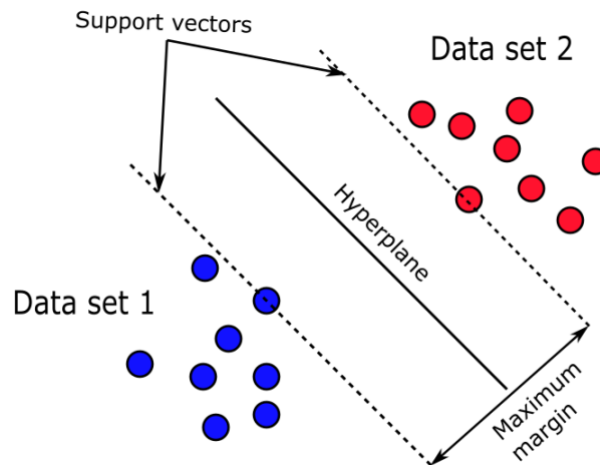


Figure 11: Illustration of Support Vector Machine Hyperplane (Sureh et al., 2019)

3.2.5.3 Artificial Neural Network

Based on the analogy of how humans process information, Artificial Neural Networks portrays a new computing technology, which is distributed and paralleled massively. As inputs are processed by altering the dynamic activity of neurons within a neural network, information is processed in response to those inputs. The behavior of these neurons is similar to that of biological neurons in human brains. As inputs are received, calculations are performed, and outputs are generated by neurons. The biological neurons typically have layers and are responsible for a particular subtask. The artificial neurons are also structured in several layers, with each layer carrying a specific function. A simple ANN model is fundamentally composed of three layers, the input layer to receive the values from external sources, a hidden layer to process and compute the values, and the output layer to generate the results or prediction, and they are interconnected in a serial manner. A model's hidden layers can have different numbers of additional layers, however, not more than two hidden layers. The ANN model having more than two hidden layers could be considered as deep neural architecture. The neurons in the layers have weights or specific values, which describe their relationship. The activation functions are applied to neurons to calculate outputs based on input values. Each neuron multiplies the weight when transferring the result to the output layer (Pham et al., 2020).

A significant issue that is observed during the construction of the artificial neural network is usually overfitting. Rather than learning to generalize, the networks memorize training data and poorly perform on unseen data. The problem may occur in a network that

contains a large number of hidden nodes, meaning having high complexity. In contrast, a model with very few hidden nodes may also be less robust to generalization (Pham et al., 2020).

The model will be built using the Sequential and Dense function from the Keras library. Two hidden layers will be implemented with 64 neurons in each layer with 'relu' activation function. In compiling stage, 'adam' optimizer will be utilized. When fitting the model, 100 for epochs and 32 for batch size will be passed into the function. The activation function in the hidden layer, and finally the random state is set to 42. After that, the model will be tweaked with the Random Search approach by utilizing RandomizedSearchCV, which will incorporate with Ten-fold cross validation in the function. This will ensure that the model is constantly performing well with data that has not been seen before and helps to steer away from overfitting.

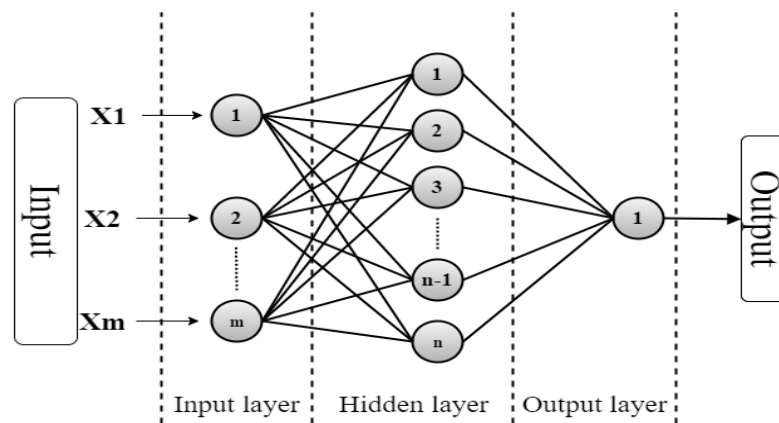


Figure 12: Illustration of Artificial Neural Network Architecture (Pham et al., 2020)

3.2.6 Model Evaluation and Selection (ASSESS)

It is at this stage that the models that were constructed during the modeling phase are assessed in terms of their reliability, robustness, and performance. It is imperative to note that there are several types of evaluation metrics to compare the models. Each of these metrics has its own unique advantages as well as shortcomings. Taking into account that the target variable of this project is the quantity of rainfall estimation, it can be regarded as a regression problem and thus, regression metrics will be applied to compare the models. The values of RMSE, MAE, and R Square will be computed after each model is developed successfully. In the final step, the most accurate model that can provide the highest accuracy will be chosen as the final model and the deployment process will proceed afterwards.

Also, at this stage, a report on the capstone project is produced. From the beginning to the end, the report will provide information regarding the experimentation process, including

analysis, results and findings. It will also address the research questions based on the results as well as how the objectives were completed, and finally the conclusions. It is also intended to conduct a retrospective review of the project as a way of determining what went well s, what could have been enhanced, and what will be needed in future projects.

3.2.6.1 Mean Absolute Error (MAE)

The MAE is useful for comparing the performance of regression model. It essentially computes the amount of error produced by the model in comparison with actual values which has been known. It does not take into account of the direction that exists in the error. The lower the MAE is generated by the model, the better the model performance is. The below formula is used in calculating the MAE value of the model.

$$MAE = \frac{1}{n} \sum_i^n |y'_i - y_i| \quad (1)$$

In the above equation (1), y is the known actual values and y'_i represents the value of prediction made by the model. The total observations are expressed by n where i ranges within $1 < i \leq n$.

3.2.6.2 Root Mean Square Error (RMSE)

This is also one of the metrics used to compare the performance of the regression model and widely used by many researchers. It also computes the amount of error produced by the model in comparison with actual values which has been known in similar way as MAE, except it has taken the square root and thus, the results are the actual values. The closer the RMSE value is to the zero, the better the performance of the model is. The below formula is used in calculating the RMSE value of the model.

$$RMSE = \sqrt{\frac{\sum_i^n (y'_i - y_i)^2}{n}} \quad (2)$$

In the above equation (2), y is the known actual values and y'_i represents the value of prediction made by the model. The total observations are expressed by n where i ranges within $1 < i \leq n$.

3.2.6.3 Determination of Coefficient (R Square)

The coefficient of determination or simply R Square values is the measurement of variance of the model, which is standardized between 1 and 0 value. It expresses the amount of variance of information of the dependent variable which has been explained by all the

independent combined. The R Square of zero mean there is no information is captured to predict the target variable while 1 means the model capture 100% information from the input variable to predict the dependent variable. The higher the value of R Square, the better the model is. The below formula is used in calculating the R Square value of the model.

$$R^2 = \left(\frac{n(\sum_i^n y' y) - (\sum_i^n y')(\sum_i^n y)}{\sqrt{(n \sum_i^n y^2 - (\sum_i^n y)^2) ((n \sum_i^n y'^2 - (\sum_i^n y')^2))}} \right)^{\frac{1}{2}} \quad (3)$$

In the above equation (3), y is the known actual values and y'_i represents the value of prediction made by the model. The total observations is expressed by n where i ranges within $1 < i \leq n$.

3.2.6.4 Verifying Overfitting or Underfitting

The overfitting and underfitting of the model are very common issues in machine learning. They simply mean the machine learning model is either learned too much or too little from the training data and it perform poorly on new or unseen data.

The overfitting issue can happen when the model learned the noises and tried to fit all the data from training dataset, resulting badly performs on testing data. It also could be when the model is very complex. The underfitting issue occurs when the model is not complex sufficiently and cannot capture the information and patterns from the data. As a result, it poorly performs on both training and testing datasets. Most of the time, overfitting issue is more typical than underfitting since the amount of data for training a machine learning model is highly available these days. In case the model is having either issue, it can be considered that the model is useless and cannot proceed to deployment process (Montesinos López et al., 2022).

There are ways to verify whether the model is underfit or overfit. In this project, to determine the model is overfit or underfit, the evaluation metrics of training and testing dataset will be examined. If the training and testing evaluation are quite comparable, the model is considered as having a good fit model and neither overfit nor underfit. For the sake of tackling, these issues, Ten-fold cross validation method is implemented in model tuning. Ten-fold cross validation is the process of obtaining the approximates of generalized performance by spiting the dataset into ten subsets in equal size and trained and tested ten times. Each time, the model is trained with nine subset of dataset and tested with remaining one subset of data. The process

is repeated until all the subsets are trained and tested. This could make sure, the model can generalize well and perform optimally on unseen data (Montesinos López et al., 2022).

3.2.7 Deployment

This is the final stage of the project where the best performing model which has been tuned and assessed using the evaluation metrics is deployed on the web page. After the model has been trained and selected the best model, it is not possible for other people to test and use the model directly with training environment. For that reason, the model is deployed using the simple web interface for easier use of model. In this project, the model will be deployed using the Streamlit Community Cloud. The simple web user interface will be developed in the Visual Studio Code on local machine first. Then, the application will be uploaded onto Streamlit Cloud. The interface will have data to input in the text box and a button to predict based on the put. The model will output the prediction result once the predict button is clicked. There are a few essential steps involved in deployment of the model, which are explained below.

Saving the model: The best model will be saved using model saving libraires such as dump and load function from Joblib library in case the best model is either Random Forest or Support Vector Regressor. If the best model is found to be Artificial Neural Network, save and load function from the Kera library will be used.

Saving the Standard Scaler: Owing to the fact that the input features are scaled using the Standard Scaler before modelling, it is necessary that the new input data are also scaled using the same method before passed into the model for prediction. Therefore, the Standard Scaler which is used for the best model will be saved using the dump function from Joblib library and will be loaded again using load function when deploying.

Pre-processing for input data: It is also possible that the input data are being pre-processed during the data pre-processing stage such as variable transformation or feature creation. The same approaches will be applied to the input data before the data are passed into the model in the process of deployment.

Accessible link: Once the model has been deployed successfully and tested satisfactorily, the access URL link will be provided to the users to access the model and predict the estimates of rainfall amount accordingly.

3.3 Summary

In this chapter, a detailed discussion of the methodologies and approaches to be employed in this entire project is presented. The proposed methodology is based upon SEMMA methodology, which is composed of six main stages: understanding the problem, obtaining a dataset (Sample), exploring the dataset (Explore), preprocessing the data (Modify), modeling rainfall prediction models (Model), evaluating the built models and selecting the model (Assess), and finally deploying it on the web interface. In order to facilitate a clear understanding of the process, each stage has been thoroughly explained. Furthermore, the methods and algorithms which will be constructed for rainfall prediction models in the subsequent chapters of this report are also justified and demonstrated in detail.

Summary of tools for implementation

Programming language	Python
Training and Computing environment	Jupyter Notebook in Google Colab: Organizing codes Google Colab: for model development and training Local machine: for developing simple web interface Streamlet Cloud : for deployment of the best model
Data Exploration	Pandas, NumPy, Matplotlib, Seaborn, Stats, Missingno
Data Pre-processing	Pandas, Scikit Learn, NumPy
Modelling	Scikit Learn, Keras
Model evaluation	Scikit Learn
Deployment	Streamlit, Virtual Studio Code

CHAPTER 4

EXPERIMENTATION

4.0 Introduction

This chapter documents the whole implementation process proposed in last chapter, starting from initial data exploration stage till the final stage of deployment of machine learning models for predicting the amount of rainfall for the area of Taunggyi. The chapter composes of data exploration using descriptive statistics, histograms, correlation matrix, identifying anomalies in the dataset such as missing values, outliers and duplicates data, data pre-processing such as handling anomalies, data pre-processing including feature engineering, normalization, model building and the deployment. All the stage experimented are performed using Python programming language in Google Colab. The data manipulation and data modelling libraires such as Pandas, Numpy, Matplotlib, Seaborn, Scikit-learn and Streamlit app framework will be employed.

4.1 Data Exploration

This section explores the variables in the dataset using exploratory data analysis. The analysis is conducted using Pandas, Seaborn, Matplotlib, Missingno libraries. This section will begin with the preliminary exploration of data and thereafter the further analysis of data will be carried out after data pre-processing and feature engineering steps have been completed.

4.1.1 Uploading the dataset into Google Drive

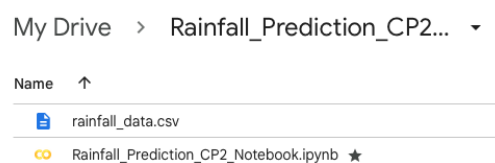


Figure 13: Uploading dataset and Python Notebook into Google Drive

Firstly, the dataset is uploaded, and a new python notebook file is created on Google Drive in a folder named "Rainfall_Prediction_CP2_202302" as in Figure (13).

4.1.2 Importing libraries and dataset

```
# Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt

# Set max row and col for pandas dataframe
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 200)

# Supress warnings
import warnings
warnings.filterwarnings('ignore')

# Importing dataset
rf = pd.read_csv('/content/drive/MyDrive/Rainfall_Prediction_CP2_202302/rainfall_data.csv')
```

Figure 14: Importing libraries for Data Exploration

The libraries required for data exploration and the dataset are imported as shown in the Figure (14).

4.1.3 Exploring the dataset

```
# Check Dataset's number of rows and columns
rf.shape

(13514, 15)

# Examine Types of variables in the dataset
rf.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13514 entries, 0 to 13513
Data columns (total 15 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   date                 13514 non-null  object
1   temperature          13514 non-null  float64
2   sun_dur              13514 non-null  float64
3   shortwave_rad        13514 non-null  float64
4   r_humidity           13514 non-null  float64
5   cloud_covr           13514 non-null  float64
6   pressure             13514 non-null  float64
7   evapotrans           13514 non-null  float64
8   soil_temp            13514 non-null  float64
9   soil_moisture        13514 non-null  float64
10  vapor_pressure       13514 non-null  float64
11  wind_speed           13514 non-null  float64
12  wind_dir             13514 non-null  float64
13  wind_gust            13514 non-null  float64
14  rainfall             13514 non-null  float64
dtypes: float64(14), object(1)
memory usage: 1.5+ MB

# View dataset
rf.head(10)
```

	date	temperature	sun_dur	shortwave_rad	r_humidity	cloud_covr	pressure	evapotrans	soil_temp	soil_moisture	vapor_pressure
0	19850101T0000	17.24	581.98	4393.04	74.79	37.10	1015.48	2.64	16.06	0.28	5.62
1	19850102T0000	16.79	576.65	4733.91	73.79	20.18	1015.70	2.93	16.27	0.28	5.86
2	19850103T0000	15.96	424.92	4748.15	74.04	60.13	1015.85	2.92	17.05	0.27	5.94
3	19850104T0000	15.33	419.72	4887.88	75.29	37.93	1016.10	2.85	15.80	0.27	5.35
4	19850105T0000	15.49	514.05	4951.96	72.13	34.67	1015.27	2.94	17.16	0.27	5.92
5	19850106T0000	14.96	658.83	5155.77	61.25	0.00	1014.96	3.01	13.42	0.27	7.73
6	19850107T0000	14.87	612.33	4981.33	62.71	4.42	1016.38	2.88	13.31	0.26	7.31
7	19850108T0000	15.51	437.56	4981.33	70.42	22.37	1015.56	2.83	14.62	0.26	6.29
8	19850109T0000	15.22	579.62	5077.45	75.38	17.09	1016.01	2.75	14.76	0.26	5.43
9	19850110T0000	15.19	464.33	4913.69	76.13	41.00	1014.89	2.72	16.27	0.26	5.14

Figure 15: Dataset Exploration

The dataset consists of (13,514) observations and (15) variables in total. All the variables except date variable are continuous data type, which is represented as "float64" in the Figure (15). A data exploration allows a more complete understanding of the dataset and determines what methods should be used during the data pre-processing stage. The target variable is 'rainfall' which is a numerical continuous data type. The data variable will be processed in the next stage of data pre-processing.

4.1.4 Descriptive Statistics of the variables

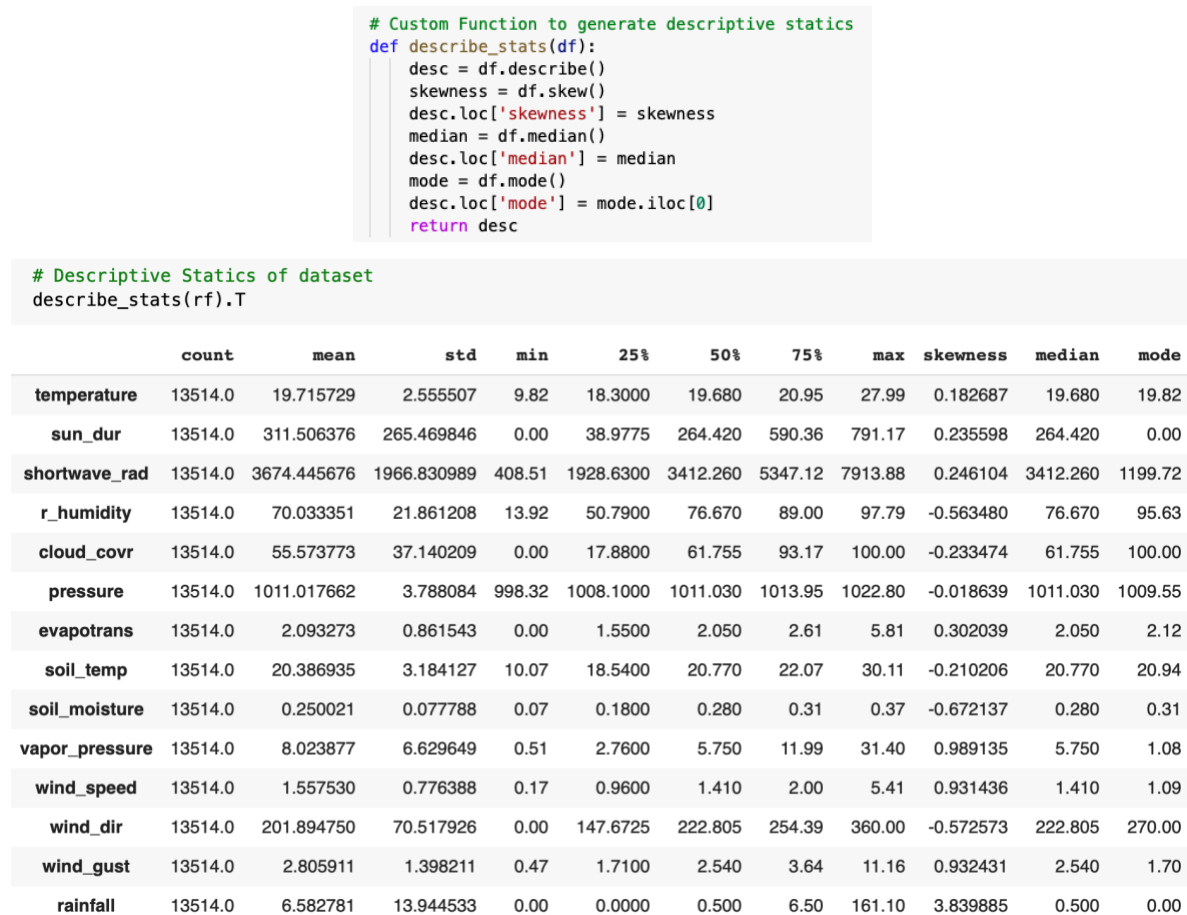


Figure 16: Descriptive Statistics of the variables

In order to generate the common descriptive statistics such as mean, median, mode and other important information such as skewness of the variables, a custom function is created. As seen in Figure (16), the dataset seems to be lacking missing data points and the range of the values of in the variables are vary with different unit measurements. In terms of numerical mean values, the highest value is found in sun wave radiation and the lowest value in soil moisture. There are also zero value in some of the variables such as sun duration, cloud coverage, evapotranspiration, wind direction and rainfall. The skewness of most variable are

within the Detail explorations of each variable will be analyzed in univariate analysis of the variables section.

4.1.5 Analysis of variables

4.1.5.1 Univariate Analysis

A single variable analysis is called a univariate analysis. Analyses based on univariate data can be used to describe data and identify patterns. The univariate analysis of the all the variables is conducted using the loop as in below code.

```
# Copy dataframe for data exploration without date variable
rf1 = rf.drop('date', axis =1)

# Loop and create a separate histogram for each variable
sns.set_palette("muted", color_codes=True)
for col in rf1.columns:
    # Create a new figure
    plt.figure(figsize=(10, 6))

    # Plot histogram of current column
    sns.histplot(data=rf1, x=col, kde=True)

    # Compute the descriptive statistics
    col_min = rf1[col].min()
    col_max = rf1[col].max()
    col_mean = rf1[col].mean()
    col_median = rf1[col].median()
    col_mode = rf1[col].mode().iloc[0]
    col_skewness = rf1[col].skew()
    col_std = rf1[col].std()

    # Add axis labels and title with descriptive statistics
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.title(f"Univariate Analysis of {col}\n(min={col_min:.2f}, max={col_max:.2f},
            mean={col_mean:.2f}, median={col_median:.2f}, mode={col_mode:.2f},
            skewness={col_skewness:.2f}, std={col_std:.2f})")

    # Display the plot
    plt.show()
```

4.1.5.1.1 Univariate Analysis of continuous variable - TEMPERATURE

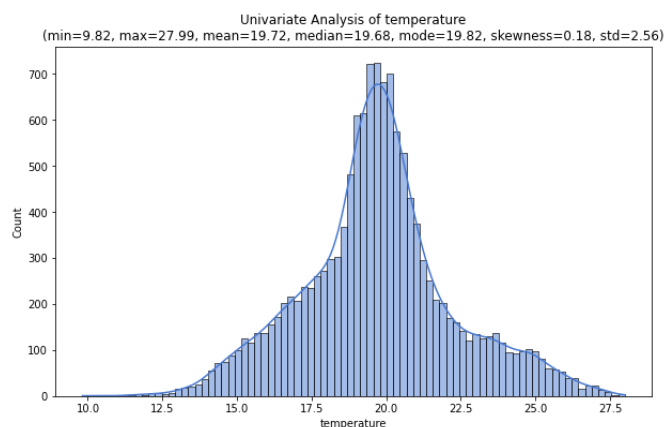


Figure 17: Univariate Analysis of 'temperature'

The average minimum temperature is 9.82 °C and the maximum temperature is 27.99 °C. The values are relatively narrow across the range, with no extreme values observed at either end. Considering the range of values, the standard deviation of 2.56 and the histogram

indicate that the data are well clustered around the mean. The mode is close to the mean, implying the distribution is smooth and does not contain any significant outliers. The skewness is +0.18, which means slightly right skewed.

4.1.5.1.2 Univariate Analysis of continuous variable - SUN_DUR

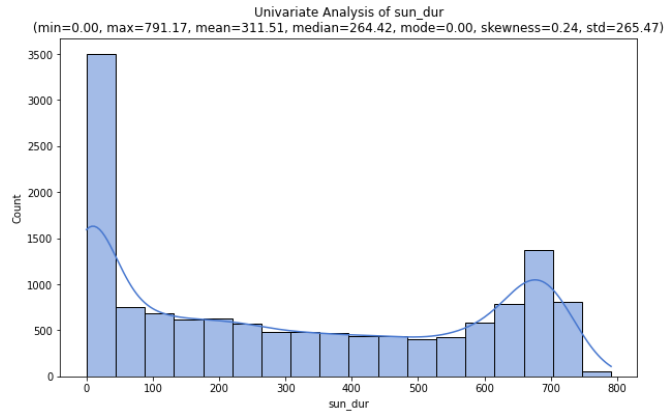


Figure 18: Univariate Analysis of 'sun_dur'

The average minimum and maximum sun duration per day are zero minute (no sun light at all) and 791.17 minutes (13.2 hours). The mean duration is 311.51 (5.2 hours). There are many days which does not see the sun light and thus the distribution is in right skewed with +0.24. The zero values are not regarded as null values since it carries a meaning. The larger standard deviation and the histogram imply that the variability is high. As the mode is much smaller than the mean and median may represent the possibility of outliers, which will be further investigated in outliers' detection section.

4.1.5.1.3 Univariate Analysis of continuous variable - SHORTWAVE_RAD

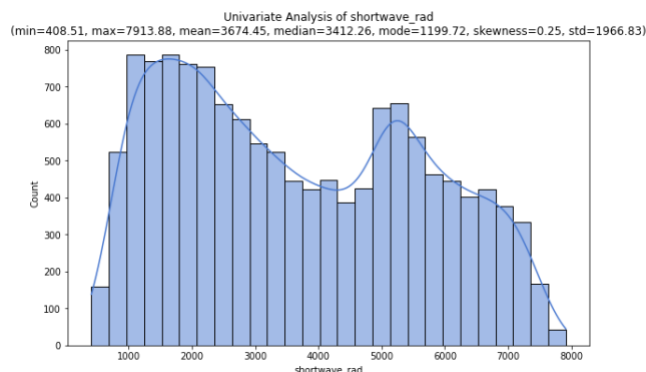


Figure 19: Univariate Analysis of 'shortwave_rad'

The values of short-wave radiation ranges from a minimum of 408.51 to a maximum of 7913.88 Wm⁻². It also has high variability which can be seen from the histogram and having

large standard deviation of 1966.83. The mean of the variable is 3674.45 and the median is 3412.26. The distribution can be considered left skewed with +0.25.

4.1.5.1.4 Univariate Analysis of continuous variable - R_HUMIDITY

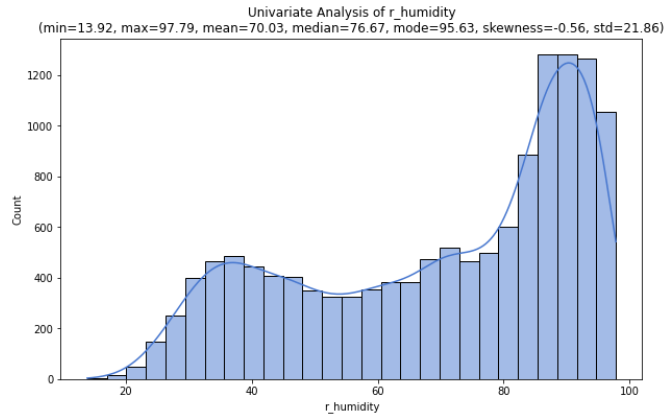


Figure 20: Univariate Analysis of 'r_humidity'

The mode value of relative humidity is 95.63 %, meaning the region has the high relative humidity most of the time. It ranges from the minimum value of 13.92 % and the maximum of 97.79 % with the mean having 70.03 %. The distribution is negatively skewed since most of the values lies on the right side with -0.56. No sign of extreme values seem to present on both sides of the distribution.

4.1.5.1.5 Univariate Analysis of continuous variable - CLOUD_COVR

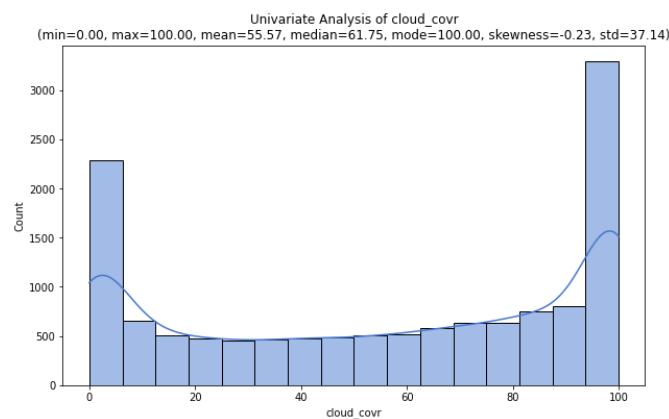


Figure 21: Univariate Analysis of 'cloud_covr'

The cloud coverage values range from 0% (completely blue sky) and 100% (covered with cloud all over the sky). Although the minimum value starts from zero, it makes sense to the context, and they are not null values. The distribution can be considered negative skewed with 0.23 although some frequent values are present on the left side. As observed from the

histogram and high standard deviation of 37.14, the variability of the variable is also high. The higher mode values represent, the region is covered with cloud in majority of the time.

4.1.5.1.6 Univariate Analysis of continuous variable - PRESSURE

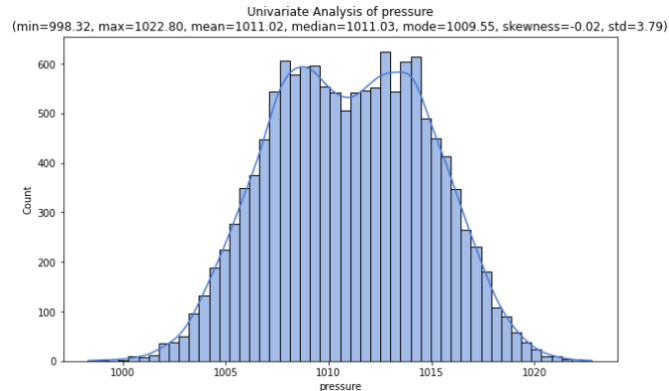


Figure 22: Univariate Analysis of 'pressure'

The pressure in the region has the lowest value of 998.32 and the highest pressure of 1022.8 hPa. The mean pressure is 1022.03, which suggest that the average pressure is slightly above the standard atmospheric pressure of 1013.25 hPa. Since the mean and median values are equivalent with 1011.02 hPa, indicating the distribution is almost symmetrical. The lower standard deviation of 3.79 and the histogram shows that the majority of the values are tightly clustered around the mean.

4.1.5.1.7 Univariate Analysis of continuous variable - EVAPOTRANS

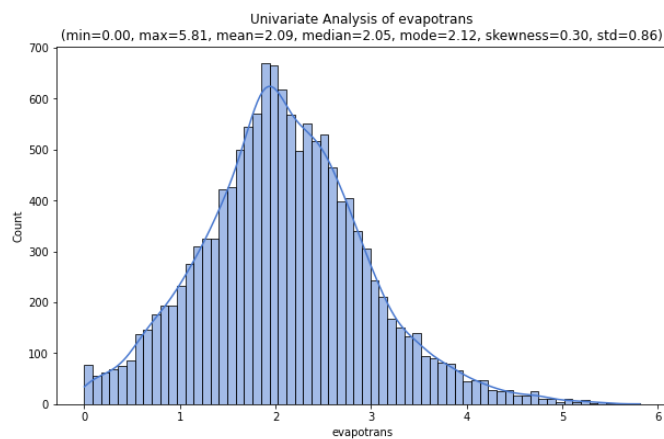


Figure 23: Univariate Analysis of 'evapotrans'

The distribution of the evapotranspiration (ET) seems roughly normal as observed from the histogram and the mean and the median and mode are closer. The values ranges from a minimum of zero and a maximum of 5.81 mm. The ET can be zero, meaning there are very

low or no moisture in the soil and less vegetation, which is very rare occasion in this region. Thus, these zero values are considered as invalid data and need to process before modelling. The mean of the variable is 2.09 and the standard deviation of 0.86. The data are widely spread, and it has higher variability as it can see in the histogram.

4.1.5.1.8 Univariate Analysis of continuous variable - SOIL_TEMP

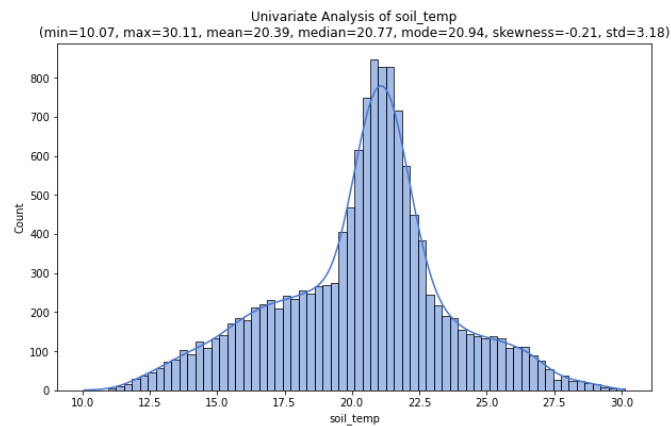


Figure 24: Univariate Analysis of 'soil_temp'

The soil temperature variable has the lowest temperature of 10.07°C and 30.11°C maximum soil temperature. A relatively narrow range of values is observed throughout the range. No extreme values are seen at either end of the range. According to the histogram and the standard deviation of 3.18, the data are well clustered around the mean. Mode is close to mean, indicating smooth distribution and absence of significant outliers. The skewness of the data is -0.2, which indicates that it is slightly left-skewed.

4.1.5.1.9 Univariate Analysis of continuous variable - SOIL_MOISTURE

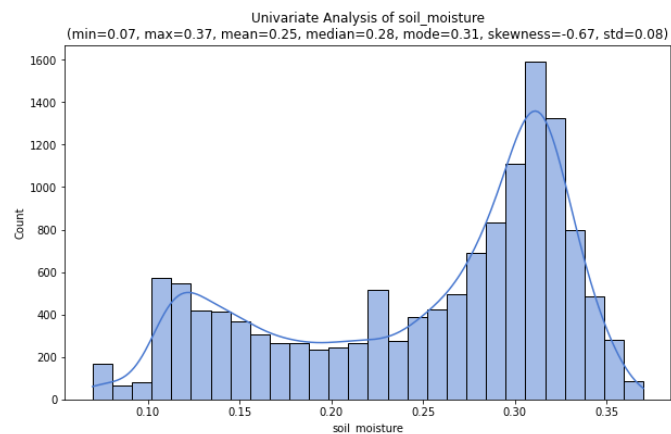


Figure 25: Univariate Analysis of 'soil_moisture'

The soil moisture varies with the minimum of 0.07 m³ and the maximum of 0.37 m³, representing very narrow range. The distribution is left-skewed with -0.67 as shown in the histogram. The standard deviation is as low as 0.08 compared to the mean and thus less variability in the values of the soil moisture variable. The mode values of 0.31 means that the region has high soil moisture most of the time.

4.1.5.1.10 Univariate Analysis of continuous variable - VAPOR_PRESSURE

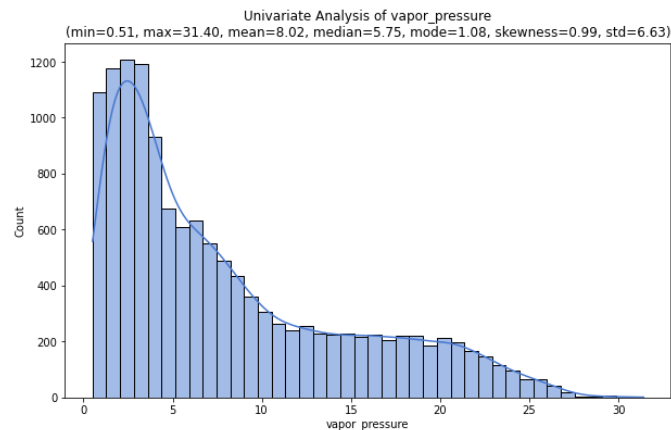


Figure 26: Univariate Analysis of 'vapor_pressure'

A relatively wider range of values are present in the vapor pressure deficit variable since the maximum and minimum values are 0.51 and 31.40 hPa. The mean value 8.02 is higher than median and mode values and the histogram also show that the distribution is positively skewed with a skewness of 0.99. The variable has a large amount of variability with higher standard deviation of 6.63, which is also reflected in the histogram.

4.1.5.1.11 Univariate Analysis of continuous variable - WIND_SPEED

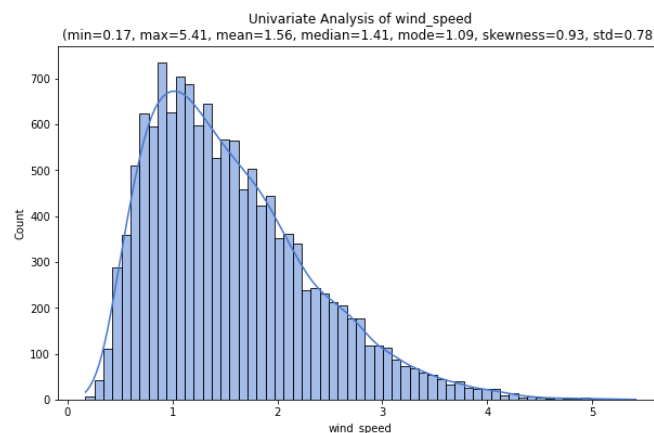


Figure 27: Univariate Analysis of 'wind_speed'

The wind speed blows of 0.17 m/s to the highest wind speed of 5.41 m/s in the region. The mode value is 1.41 represent the region has light wind most of the time. The mean wind speed is 1.56. The distribution is right skewed with 0.93. Since the standard deviation is 0.78, compared to the mean, it can be considered the data are moderately distributed. There are no sign of significant extreme values or outliers.

4.1.5.1.12 Univariate Analysis of continuous variable - WIND_DIR

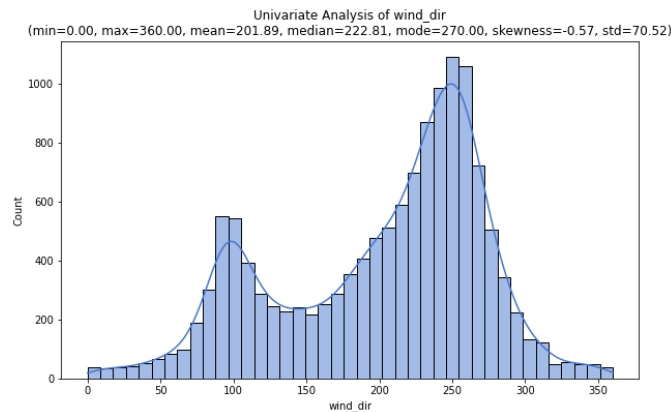


Figure 28: Univariate Analysis of 'wind_dir'

The wind direction variable has the range from 000 to 360 degrees. The distribution is negatively skewed with 0.57. The mode value is 270 which means most of the time the wind blows from West direction. In a compass, either wind direction from 000 or 360 carry the same interpretation as North. Therefore, the data are not consistent and need to convert to either 000 or 360 so as to have consistency across the values of the variable. This will be tackled in data pre-processing stage. In addition, the wind direction values are in circular nature, they need to be transformed to maintain the circular values. This process will be conducted in feature engineering section.

4.1.5.1.13 Univariate Analysis of continuous variable - WIND_GUST (Seasonally)

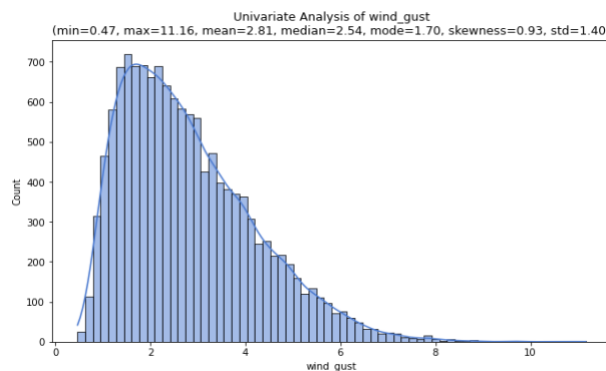


Figure 29: Univariate Analysis of 'wind_gust'

The wind gust variable seems to indicate that the gusts were relatively mild, with most of the data clustered around 1.7 to 2.5, but there were some stronger gusts that reached up to a maximum of 11.16 occasionally. The distribution is skewed to the right with 0.93. As observed from the histogram and the standard deviation of 1.4, it can be regarded as the variability of the wind gust is not that high.

4.1.5.1.14 Univariate Analysis of continuous variable - RAINFALL

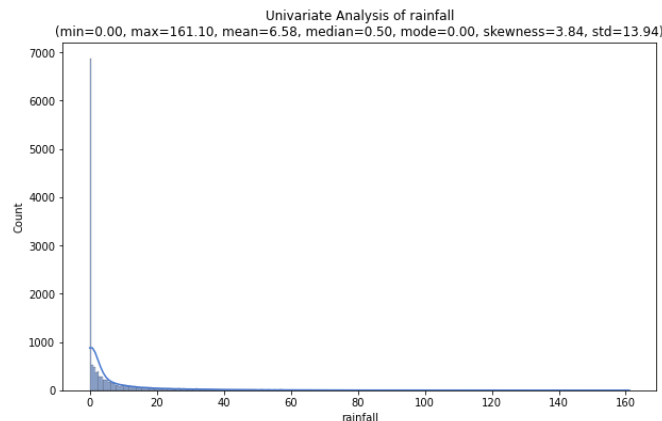


Figure 30: Univariate Analysis of 'rainfall'

The rainfall variable is the target variable for this project. The values vary from 0 (no rainfall) to 161.1 mm per day. The mean rainfall amount is 6.58 while the median is 0.5 and the mode is zero, representing the region receive less rainfall most of the time. The variable is considered positively high skewed with 3.84. The zero values are not considered null or missing values in the variable. It has meaning that there is no rain on the day. Since this project is also considering the amount of rainfall which includes whether it will rain based on the amount of rainfall and the effect of independent variable, these values will be treated as valid data. Furthermore, based on the literature review, many researchers claimed that the rainfall patterns with climate data are in non-linear and thus the dependent variable is not mandatory to follow the normality assumption of having a normal distribution. As the machine learning techniques chosen also are also suitable for non-linear data, no further treatment will be applied to the dependent variable.

4.1.5.2 Bivariate Analysis

It is the objective of a bivariate analysis to determine the statistical relationship between two variables and to determine if one variable is capable of predicting another based on the other. In the following section, the relationship between the independent variables such as temperate, relative humidity, wind speed, soil moisture against the dependent variable rainfall will be observed. For the reason that all the variables are numerical, scatter plot from Matplotlib library is used. In order to better understanding of their association of direction and strength, Pearson correlation and Spearman correlation are incorporated into the plot to help determine whether they are having linearly related or non-linearly correlated. However, an important point to keep in mind is that correlation is not causality, and there may be much more than one factor at play when it comes to the relationship between the two variables.

```
# Loop over each independent variable
for var in rf1.columns:
    plt.figure(figsize=(10, 6))
    # Create a scatter plot of the current variable against the dependent variable
    sns.scatterplot(x=rf1[var], y=rf1['rainfall'], color='brown')

    # Compute the Pearson and Spearman correlations
    pearson_corr = rf1[[var, 'rainfall']].corr(method='pearson').iloc[0, 1]
    spearman_corr = rf1[[var, 'rainfall']].corr(method='spearman').iloc[0, 1]

    # Add title with correlation labels
    plt.title(f'Bivariate Analysis of {var} vs rainfall\n(Pearson r = {pearson_corr:.2f},\nSpearman rho = {spearman_corr:.2f})')
    plt.show()
```

4.1.5.2.1 Bivariate Analysis between *TEMPERATURE* and *RAINFALL*

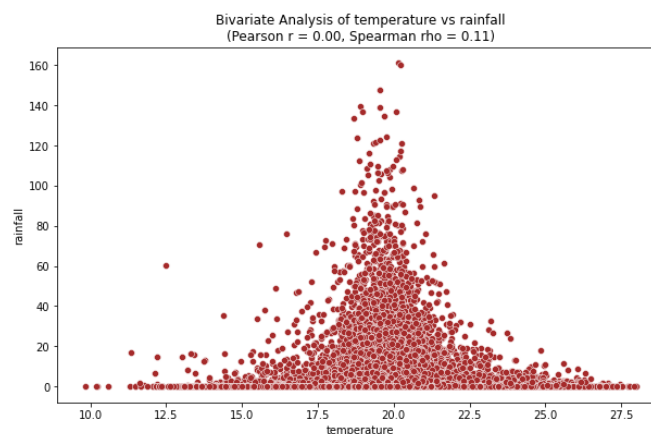


Figure 31: Bivariate Analysis between temperature and rainfall

The scatter plot from the figure show that the temperature and rainfall are not having linear relationship, this is also confirmed by Pearson correlation value of zero. It seems to have very weak positive non-linear relation, curve regression line rather than straight one, meaning an increase in temperature will increase the amount of rainfall non-linearly with Spearman

correlation of 0.11. Therefore, the two variables are having very weak positive relationship. The effect of the temperature on rainfall is not prominent.

4.1.5.2 Bivariate Analysis between *SUN_DUR* and *RAINFALL*

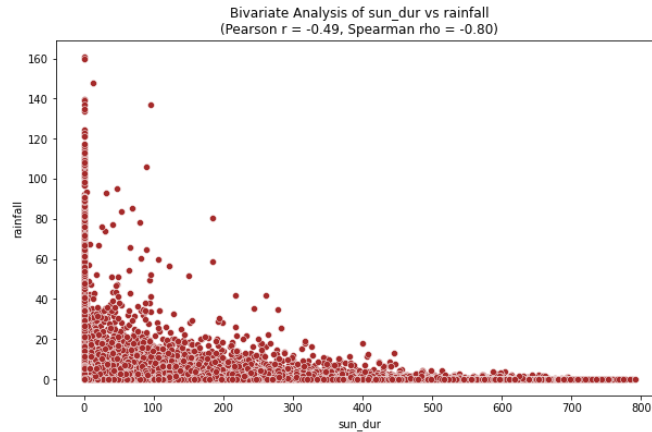


Figure 32: Bivariate Analysis between *sun_dir* and *rainfall*

Figure (32) reveals that the sun duration is negatively related with non-linear manner. Any increase in sun duration will have decrease in rainfall amount. Pearson correlation value of -0.49 and Spearman correlation of 0.8 confirm that the two variables has very strong non-linear relationship rather than liner association. The effect of sun duration on rainfall has can be considered strong.

4.1.5.2.3 Bivariate Analysis between *SHORTWAVE_RAD* and *RAINFALL*

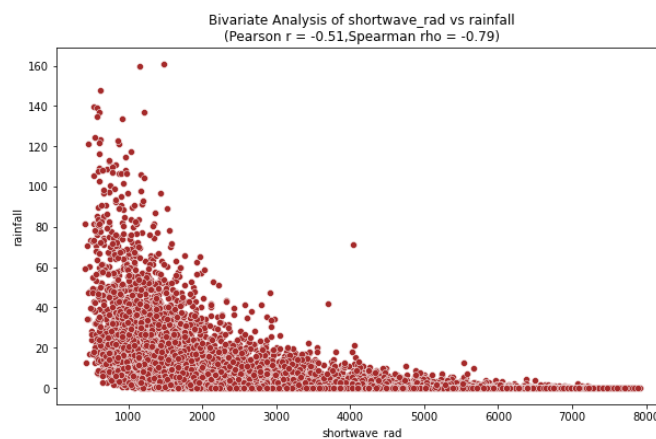


Figure 33: Bivariate Analysis between *shortwave* and *rainfall*

The scatter plot between shortwave radiation and rainfall variable illustrates that they have negative non-linear relation because the regression line seems to be a curve shape. Spearman correlation of -0.79 also supports that the two variables are strongly related in non-

linear manner. Any increase in shortwave radiation could cause lower amount of rainfall. Thus, the effect of shortwave radiation on rainfall can be considered strong.

4.1.5.2.4 Bivariate Analysis between *R_HUMIDITY* and *RAINFALL*

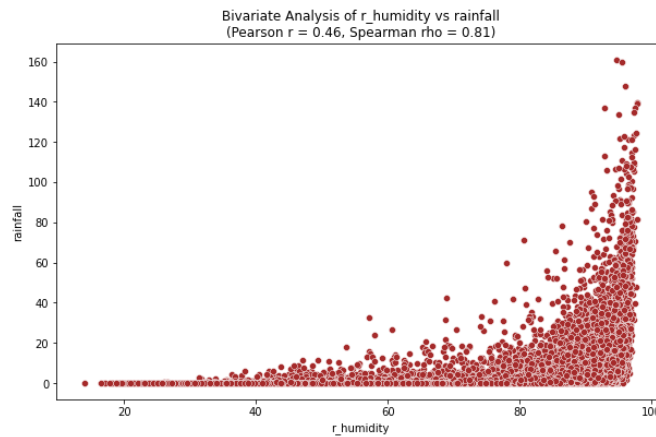


Figure 34: Bivariate Analysis between *r_humidity* and *rainfall*

The effect of relative humidity variable on rainfall can be stronger in non-linear relation since the Spearman correlation is 0.81 which is two times higher than Pearson correlation. The scatter plot also clearly shows that the two variables are having non-linear relationship. Since they have positively correlated, any increase in relative humidity will result in higher the amount of rainfall.

4.1.5.2.5 Bivariate Analysis between *CLOUD_COVR* and *RAINFALL*

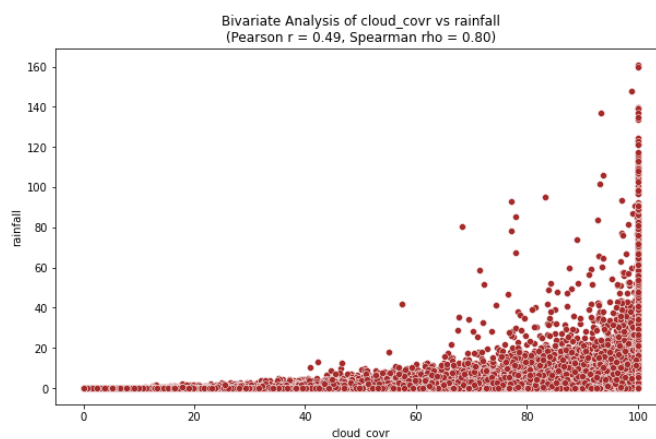


Figure 35: Bivariate Analysis between *cloud_covr* and *rainfall*

The cloud coverage variable and rainfall variable somehow positively related by observing the scatter plot, it seems to have non-linear relationship. However, the Pearson correlation value of 0.49 and the Spearman value of 0.80 represent that they are having positively strong relations. Any increase in cloud coverage could have higher rainfall.

4.1.5.2.6 Bivariate Analysis between *PRESSURE* and *RAINFALL*

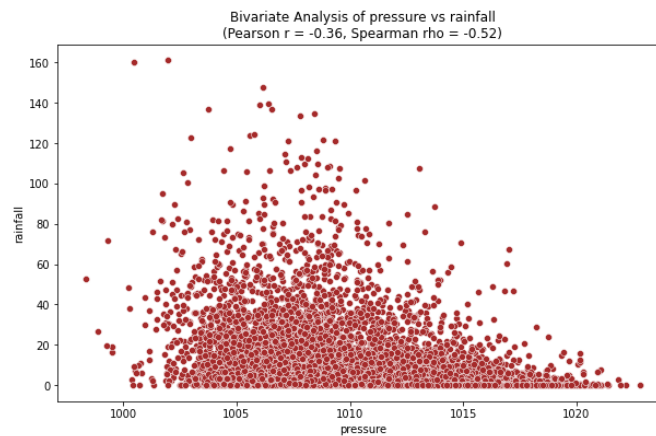


Figure 36: Bivariate Analysis between pressure and rainfall

Looking up the scatter plot, it is difficult to say there is a relationship between the pressure and rainfall variable. The plot rather seems to have random. Even so, the Pearson correlation of -0.36 means that they have weak linear and the Spearman correlation of -0.52 reveals that they are having negative moderate non-linear relationship. Any increase in the pressure would decrease the amount of rainfall, but the effect is just marginal.

4.1.5.2.7 Bivariate Analysis between *EVAPOTRANS* and *RAINFALL*

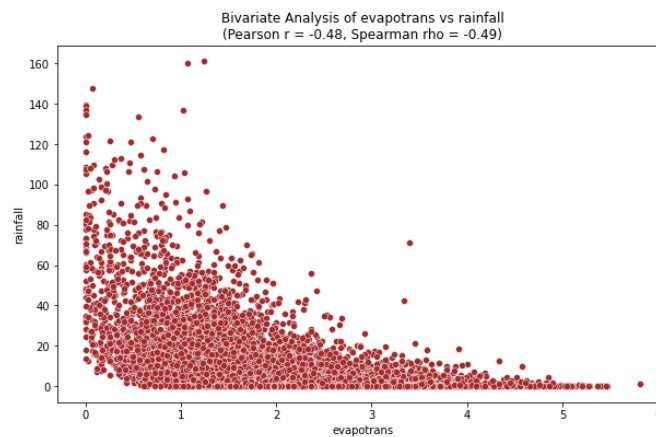


Figure 37: Bivariate Analysis between evapotrans and rainfall

The Pearson and Spearman correlation are almost equivalent with about -0.48 . The evapotranspiration variable and rainfall variable seem to have non-linear relationship rather than linear based on the scatter plot. Any increase in evapotranspiration will likely to decrease the amount of rainfall. However, the relationship is just moderate. Some potential outliers appear to exist based on the scatter plot.

4.1.5.2.8 Bivariate Analysis between SOIL_TEMP and RAINFALL

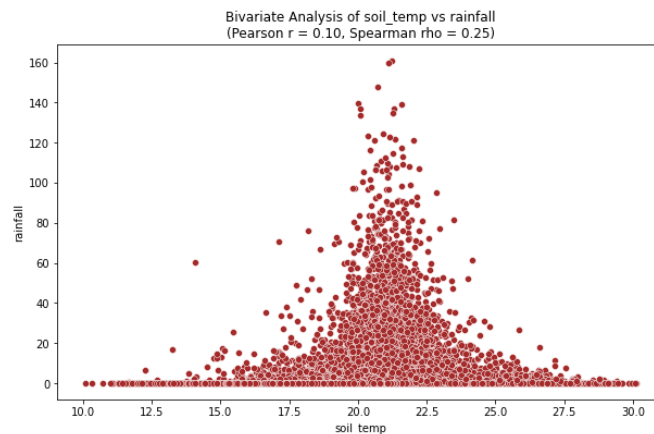


Figure 38: Bivariate Analysis between soil_temp and rainfall

According to the scatter plot, it can be considered that the soil temperature and rainfall variable are not having very low relationship or not at all as the data pattern are almost random. However, the Pearson correlation value of 0.1 and Spearman correlation of 0.25 shows that they are somehow more into non-linearly related. The effect of soil temperature has less effect on rainfall as the temperature variable.

4.1.5.2.9 Bivariate Analysis between SOIL_MOISTURE and RAINFALL

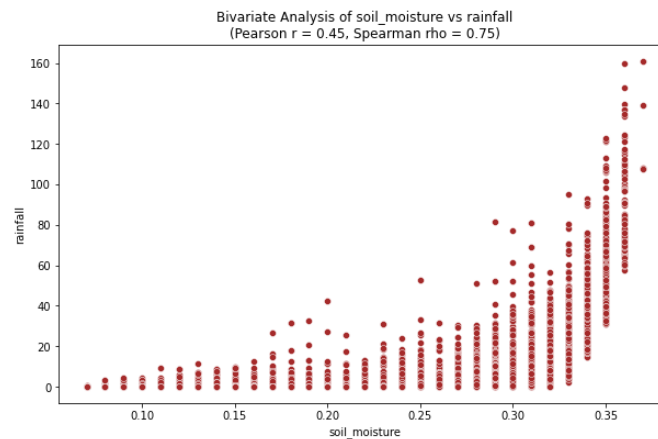


Figure 39: Bivariate Analysis between soil_moisture and rainfall

Based on the Figure (39) it is apparent that there is a strong and positive non-linear relationship between soil moisture and rainfall rather than a simple linear relationship. This information is also confirmed by the Pearson and Spearman correlations. Since the Spearman correlation of 0.75 is higher than Pearson correlation, the two variables are strongly correlated in non-linear manner. Any increase in soil moisture will result in increase of rainfall amount.

4.1.5.2.10 Bivariate Analysis between VAPOR_PRESSURE and RAINFALL

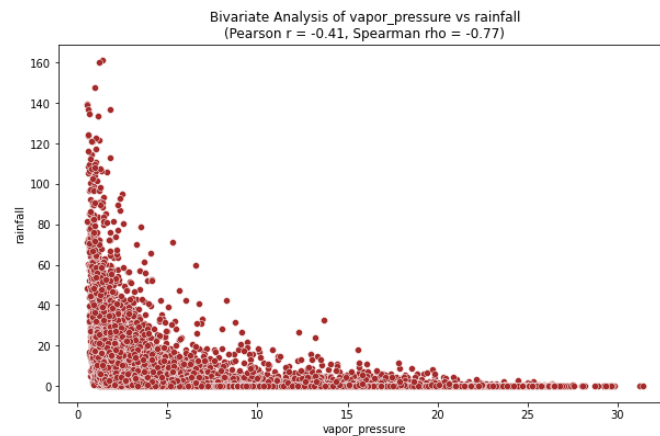


Figure 40: Bivariate Analysis between vapor_pressure and rainfall

It can be assumed that the vapor pressure is non-linearly related with rainfall as seen from the scatter plot. The correlation value from Pearson is just -0.41 while that of Spearman is -0.77, implying they are more into negatively strong non-linearity. The linear relationship is not prominent. Any increase in vapor pressure could decrease the amount of rainfall. The effect of vapor pressure is strong.

4.1.5.2.11 Bivariate Analysis between WIND_SPEED and RAINFALL

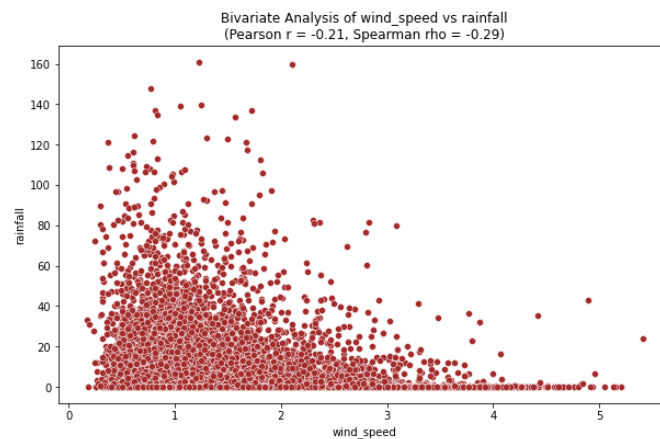


Figure 41: Bivariate Analysis between wind_speed and rainfall

Figure (41) portrays the relationship between the wind speed and rainfall variables. They look to have non-linear relationship but not strong since the data points are scattered. The effect of wind speed can be less. The Pearson value of -0.21 and Spearman correlation value of -0.29 convey that the two variable has negative association with non-linear manner.

4.1.5.2.12 Bivariate Analysis between WIND_DIR and RAINFALL

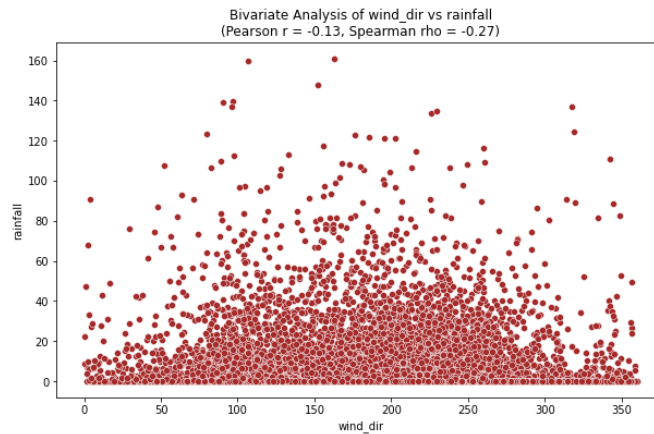


Figure 42: Bivariate Analysis between wind_dir and rainfall

In accordance with the scatter plot of wind direction and rainfall, it can be regarded that the two variables are merely having a relationship since the data pattern is almost random. However, looking up their correlation values from Pearson and Spearman represents that they have very weak negative correlation in non-linear fashion since the Spearman correlation value of -0.27 is higher than that of Pearson. The effect of wind direction could be very weak on predicting the rainfall.

4.1.5.2.13 Bivariate Analysis between WIND_GUST and RAINFALL

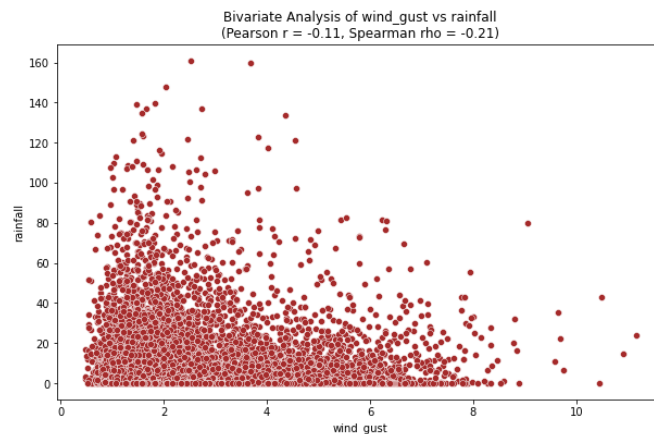


Figure 43: Bivariate Analysis between wind_gust and rainfall

The correlation of wind gust and rainfall can be assumed that they are having very weak negative relationship as seen from the plot. They are more into non-linearity rather than linear relationship. The Pearson correlation value of -0.11 and Spearman correlation of -0.21 also describe that the relationship between two variables is weak and negative non-linearity. The effect of wind gust can be low on the rainfall variable, which is similar to the association with wind speed variable.

4.1.6 Anomalies detection

In this section, the anomalies in the dataset such as missing values, duplicates data, zero variance variables, outliers, invalid data are explored using relevant libraries such as Pandas, Missingno and Scipy libraries in Python.

4.1.6.1 Exploring Missing Values

```
# Plot bar chart for non-missing values to spot the percentage of missing value in a variable
import missingno as msno
msno.bar(rf, color = 'lightgreen', figsize=(12, 6))
plt.title("The amount of Non-Missing value percentage for each variable")
plt.show()
```

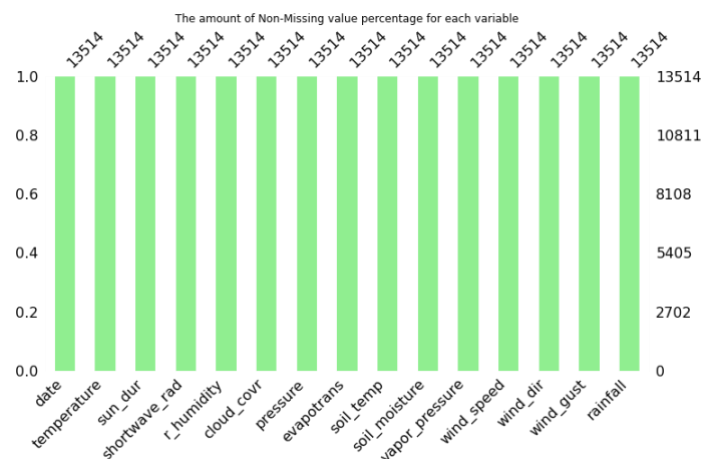


Figure 44: The amount of Non-Missing Values percentage chart

The missing values in the dataset are identified using the non-missing values bar chart from Missingno library. As seen in Figure (44), the percentage of non-missing values in each variable is 100%, meaning there is no missing values in every variable. Therefore, no further missing values imputation is required.

4.1.6.2 Identifying Duplicate Data

```
# Check duplicates
rf[rf.duplicated()== True].count()

date          0
temperature  0
sun_dur       0
shortwave_rad 0
r_humidity    0
cloud_covr    0
pressure      0
evapotrans    0
soil_temp     0
soil_moisture 0
vapor_pressure 0
wind_speed    0
wind_dir      0
wind_gust     0
rainfall      0
dtype: int64
```

Figure 45: The number of total duplicate data in each variable

The duplicate data are identified using the conditional statements and the total values are counted individually. The Figure (45) shows that there is no duplicate data in the dataset. No further treatment of duplicate data is needed.

4.1.6.3 Identifying Zero-variance variable

```
# List out the variables that are having zero variance
var_zero = list(rf.var()[rf.var() == 0].keys())
print(pd.DataFrame(var_zero))
```

```
Empty DataFrame
Columns: []
Index: []
```

Figure 46: Identifying zero variance variable in the dataset

As a method to identify variables that are zero variance in the dataset, the conditional statement is used, and it confirms that there is no zero-variance variable in the dataset since the generated output is an empty data frame as in Figure (46).

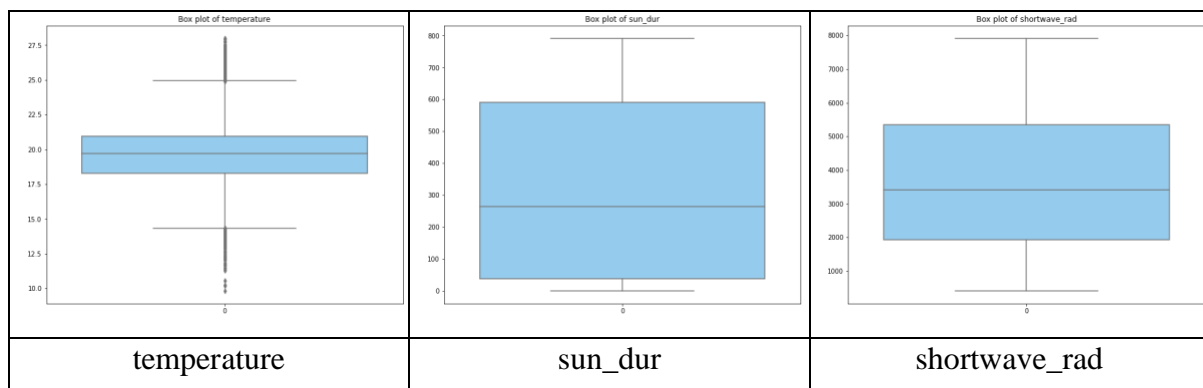
4.1.6.4 Outlier detection

The outliers are explored using two different methods, in particular, the use of boxplot and Z-score analysis. In the univariate analysis, some variables seem to have potential outliers. The variables are examined in detail in this section.

4.1.6.4.1 Outlier detection using boxplot

The following code generates the boxplot for each variable in the dataset except the date variable. The variables are investigated if there is any extreme values or outliers using box plot as below.

```
# Loop through each variable and create Box plots
for col in rf1.columns:
    fig, ax = plt.subplots(figsize=(10, 8))
    sns.boxplot(data=rf1[col], ax=ax, color = 'lightskyblue')
    ax.set_title('Box plot of ' + col)
    plt.show()
```



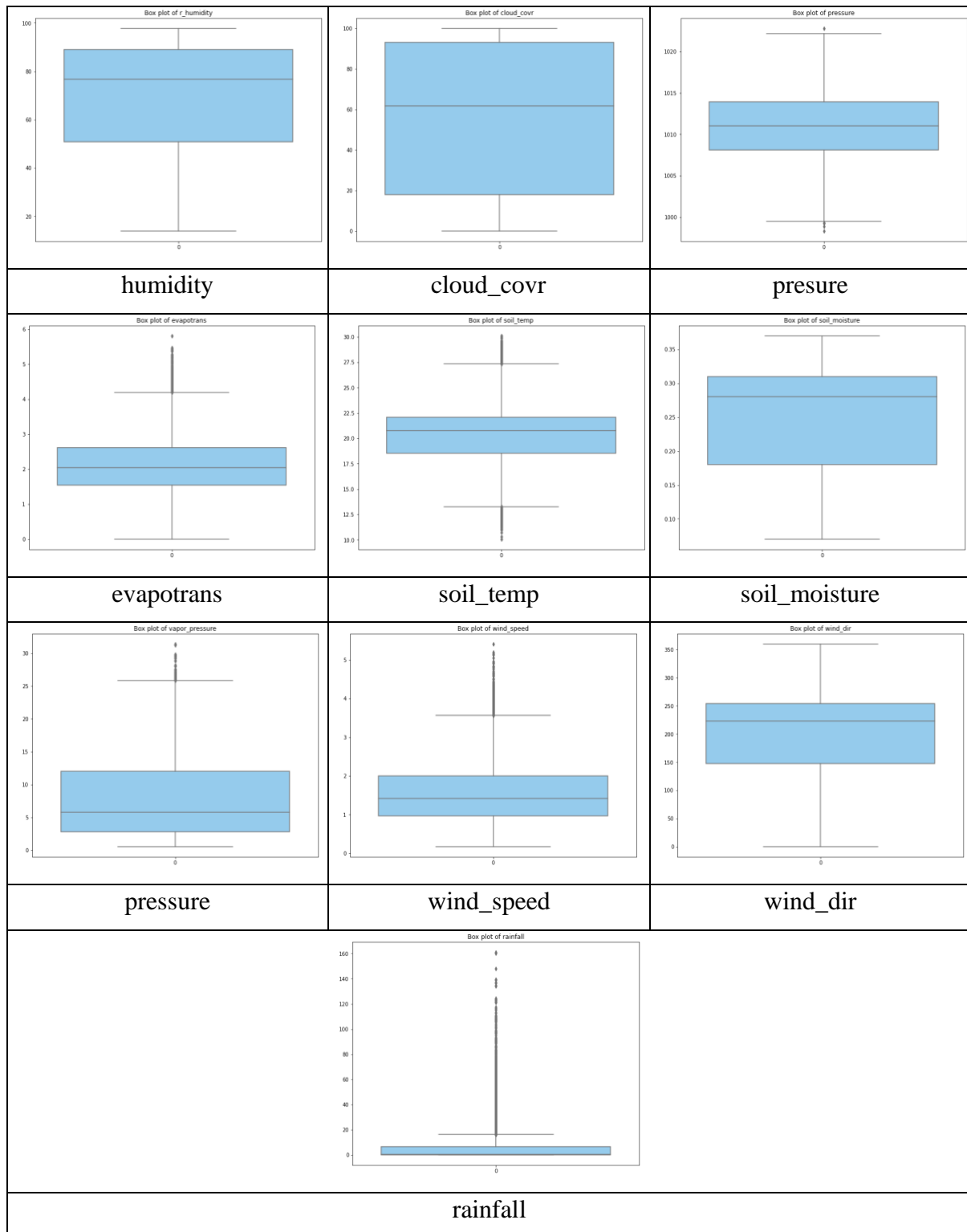


Figure 47: Box plots for each variable

According to the box plots, it is appeared to be no extremes values across all the variables, some variables contain values outside of the lower and upper ranges. However, they are not very far away from each other. Although many values are outside of the range in rainfall variable, they are not considered as outliers since this project aim is to reflect no raining days as well as the potential for heavy rainfall.

4.1.6.4.2 Outlier detection using Z-Score Analysis

The outliers are also analyzed using Z-score analysis that utilizes the distance between the mean and standard deviation to identify potential outliers. In this analysis, stats function from Scipy library and a threshold of three is used.

```
from scipy import stats

# Calculate the Z-score for the entire dataframe
z_scores = stats.zscore(rf1)

# Identify outliers using a Z-score threshold of 3
outliers = rf1.where(np.abs(z_scores) > 3)

# Print the outliers
print("Outliers: ", outliers.dropna())
```

```
Outliers: Empty DataFrame
Columns: [temperature, sun_dur, shortwave_rad, r_humidity, cloud_covr, pressure, evapotrans, soil_temp, soil_moisture, vapor_pressure, wind_speed, wind_dir, wind_gust, rainfall]
Index: []
```

Figure 48: Z-score Analysis for detecting outliers

The Z-score analysis is carried out for all the variables in the dataset, and it generated empty data frame as in Figure (48). Thus, it can also be concluded that all the variables do not contain outliers.

4.1.6.5 Identifying Invalid Data

In accordance with the univariate analysis of wind direction variable, it is noted that the invalid data points are present in the variable. The variable contains 000 degree and 360 degree which have the same meaning and, thus the conditional statement is used to see how many values of 000 and 360 are in the variable.

```
# Identify wind direction having 0 or 360 degree
rf[(rf['wind_dir'] == 0) | (rf['wind_dir'] == 360)][['wind_dir']]
```

	wind_dir
4386	360.0
4702	360.0
7546	0.0

Figure 49: The total number of invalid data point

The output illustrates that there is only one data point in the wind direction variable. It will be imputed with 360 values in data pre-processing section.

4.1.6.6 Examining deep down into potential outliers

There are two variables, namely, short-wave radiation and evapotranspiration that seem to have potential outliers in the variables based on the scatter plot in bivariate analysis. They will be reviewed in detail whether they are outliers or not.

4.1.6.6.1 Short-wave radiation Vs rainfall variable

```
# Query shortwave_rad values between 3500 to 4500 and rainfall values between 30 to 80
rf.query('shortwave_rad.between(3500, 4500) and rainfall.between(30, 80)')[['shortwave_rad', 'rainfall']]
```

	shortwave_rad	rainfall
4528	4036.15	71.2
8303	3693.50	42.1

Figure 50: Potential outliers between 3500 ~ 4500 value in short-wave radiation

The values between 3500 to 4500 in short-wave radiation is queried to see the potential outlier data points. The two data points are found as shown in Figure (50). For better understand the data points in relation to surrounding data, the range is increased between 2500 to 4500.

```
# Query shortwave_rad values between 2500 to 4500 and rainfall values between 30 to 80
rf.query('shortwave_rad.between(2500, 4500) and rainfall.between(30, 80)')[['shortwave_rad', 'rainfall']]
```

	shortwave_rad	rainfall
148	2667.33	37.8
2119	2875.59	35.3
3151	2568.54	41.5
4528	4036.15	71.2
4579	2969.93	34.1
4611	2930.77	33.5
5217	2917.42	47.2
8303	3693.50	42.1
8914	2796.38	41.6
9748	2928.99	30.4
12650	2610.37	35.0

Figure 51: The data points between 2500 ~ 4500 in short-wave radiation

As observed in Figure (51), these two values are not at the extreme end, and they are within the normal range. They appeared to be the unusual rainfall events which is the important information for this project and thus they will be included in the modelling process.

4.1.6.6.2 Evapotranspiration Vs rainfall variable

```
# Query evapotrans values between 3 to 4 and rainfall values between 30 to 80
rf.query('evapotrans.between(3, 4) and rainfall.between(30, 80)')[['evapotrans', 'rainfall']]
```

	evapotrans	rainfall
4528	3.40	71.2
8303	3.34	42.1

Figure 52: Potential outliers between 3 ~ 4 value in evapotranspiration

The values between 3 to 4 in evapotranspiration is queried to see the potential outlier data points. The two data points are found as shown in Figure (52). For better understand the data points in relation to surrounding data, the range is increased between 2.5 to 4.

```
# Query evapotrans values between 3 to 4 and rainfall values between 30 to 80
rf.query('evapotrans.between(2.5, 4) and rainfall.between(30, 80)')[['evapotrans', 'rainfall']]
```

	evapotrans	rainfall
2119	2.53	35.3
4528	3.40	71.2
4579	2.56	34.1
4611	2.68	33.5
8303	3.34	42.1
9748	2.66	30.4
12650	2.50	35.0

Figure 53: The data points between 2.5 ~ 4 in evapotranspiration

As seen in Figure (53), they are within the normal range and not located at the extreme points. They look to be the unusual rainfall occasion, the same case with short-wave radiation, which is the important information for this project and thus they will be included in the modelling process as well.

4.1.7 Summary of data exploration

The dataset used in this project is said to be very clean since it is received from the good weather data provider. There are no missing values, no zero-variance variable, no duplicate data, no outliers, and not required for transformation as the skewness of all the independent variable are within acceptable range. The only thing needs to tackle is the invalid or inconsistent data in wind direction variable.

4.2 Data Pre-processing

In this section, the anomaly found in the data exploration is addressed. As previously discussed in the data exploration section, the only issue to solve is the invalid or inconsistent data in the wind direction variable.

4.2.1 Imputation for invalid data in wind direction variable

```
# Identify wind direction having 0 or 360 degree
rf[(rf['wind_dir'] == 0) | (rf['wind_dir'] == 360)][['wind_dir']]
```

	wind_dir
4386	360.0
4702	360.0
7546	0.0

Figure 54: Invalid or inconsistent data points in the wind direction variable

Figure (54) display the inconsistent data in the wind direction variable, there is only one 0 data point in the variable.

```
# replace zero value to 360 in the wind_dir variable
rf['wind_dir'] = rf['wind_dir'].replace(0, 360)
```

```
# Identify whether all the zero value are replaced with 360
rf[(rf['wind_dir'] == 0) | (rf['wind_dir'] == 360)][['wind_dir']]
```

	wind_dir
4386	360.0
4702	360.0
7546	360.0

Figure 55: Replacing inconsistent data with appropriate value of 360

In Figure (55), the data point of 0 is replaced with 360 in the wind direction and then they are confirmed that the data is successfully replaced or imputed with 360.

4.3 Feature engineering

The process of feature engineering consists of transformation, extraction, and selection of the existing data to obtain more meaningful features, which could be used in training machine learning models to boost their performance. In this section, new features namely seasonal variables, previous rainfall and wind direction in sine and cos values will be created.

4.3.1 Seasonal variables

```
# Back up original dataset
rf_fe = rf.copy()
```

```
# Convert '19850101T0000' to '1985-01-01' format
rf_fe['date'] = pd.to_datetime(rf_fe['date'], format='%Y%m%dT%H%M').dt.date
```

Figure 56: Backup the original dataset and converting the date variable in Year-Month-Date format

First of all, the original dataset is back up in order to avoid any disruption in data handling process and a new data frame is created. The date variable is transformed into the data

format in YYYY-MM-DD to remove unnecessary string and to be easily readable by the machine.

```
# Define a function to assign season based on date
def get_season(date):
    year = date.year
    if (date >= pd.to_datetime(f'{year}-02-15')) and (date < pd.to_datetime(f'{year}-05-15')):
        return 'summer'
    elif (date >= pd.to_datetime(f'{year}-05-15')) and (date < pd.to_datetime(f'{year}-10-22')):
        return 'rainy'
    else:
        return 'winter'

# Apply the function to the date column and create a new season column
rf_fe['season'] = rf_fe['date'].apply(lambda x: get_season(pd.to_datetime(x)))
```

Figure 57: The creation of seasonal variable based on the date variable

Myanmar locates in the tropical zone, and it has three different seasons, particularly, summer, rainy and winter season. As per Myanmar Department of Meteorology, the summer starts from mid-February to first week of May, then followed by the rainy season till the late October and the winter season take over the remaining month of the year. Since the rainfall can have seasonal variation during the year, a new categorical variable named season is created based on the date variable in the dataset. In order to include the values into machine learning model, they must be transformed into numerical values. For the reason that the values of summer, rainy and winter has no ordinal or ranking, they are encoded using one-hot encoding method and generated three new features season_summer, season_rainy, and season_winter as in Figure (58).

```
# perform one-hot encoding
season_dummies = pd.get_dummies(rf_fe['season'], prefix='season')

# add the one-hot encoded variables to the original dataframe
rf_fe = pd.concat([rf_fe, season_dummies], axis=1)
```

Figure 58: One-hot encoding for seasonal variable

4.3.2 Previous rainfall variable

```
# (4) create previous day rainfall variable
rf_fe['previous_rainfall'] = rf_fe['rainfall'].shift(1)

# Drop the first observation which contains NA value
rf_fe.dropna(inplace=True)

rf_fe.shape

(13513, 20)
```

The rainfall amount of previous day could be a predictor for today rainfall amount and thus, the existing rainfall value is shifted into one row down and the variable will be served as independent variable for predicting today rainfall amount. Given that there is no previous

rainfall value for the first observation in the dataset, the first observation will be dropped, and the shape of the data frame becomes 13,513 observations with 20 variables.

```
# Check the skewness of previous rainfall
rf_fe['previous_rainfall'].skew()

3.839741219207025

# Import library for transformation
import numpy as np

# Add a constant of 1 to all values in the "previous_rainfall" variable
rf_fe['previous_rainfall_transformed'] = rf_fe['previous_rainfall'] + 1

# Apply a log transformation to the "previous_rainfall_transformed" variable
rf_fe['previous_rainfall_transformed'] = np.log(rf_fe['previous_rainfall_transformed'])

# Check the skewness of the transformed variable
print("Skewness of previous_rainfall_transformed:", rf_fe['previous_rainfall_transformed'].skew())

Skewness of previous_rainfall_transformed: 0.8979584951707537

# Replace previous rainfall with transformed variable
rf_fe['previous_rainfall'] = rf_fe['previous_rainfall_transformed']
rf_fe.drop('previous_rainfall_transformed', axis=1, inplace=True)

rf_fe['previous_rainfall'].skew()

0.8979584951707537
```

However, as it is noted from the data exploration stage that the rainfall variable is highly skewed. It has the skewness of about 3.8340 which is beyond the acceptable limit and thus, the variable should be transformed. One issue is that the variable also contains zero values, therefore, log +1 transformation will be applied. After the transformation, the variable became 0.8980 that falls under the normal range.

4.3.3 Wind direction in sine and cosine values

The wind direction variable is in 360 degrees, and which has the cyclical nature. With the aim of maintaining and better capture its cyclical nature, the wind variable will be transformed into two variable having sine and cosine values (wind_dir_sin and wind_dir_cos). This would make the machine learning model to learn better the relationship between the dependent and the wind direction variable instead of using the continuous value. The original wind direction variable will be removed from the dataset as in Figure (59).

```
# Circular Feature Engineering to wind direction variable
import numpy as np

# Compute sine and cosine of wind direction
rf_fe['wind_dir_sin'] = np.sin(np.radians(rf_fe['wind_dir']))
rf_fe['wind_dir_cos'] = np.cos(np.radians(rf_fe['wind_dir']))

# Drop original wind direction column
rf_fe.drop('wind_dir', axis=1, inplace=True)
```

Figure 59: Features creation from wind direction variable

4.3.4 Summary of Feature engineering

After feature engineering process, (6) new feature are added into the feature engineered data frame (rf_fe). The new feature are season, season_rainy, season_summer, season_winter, previous_rainfall, wind_dir_sin, and wind_dir_cos as presented in Figure (60).

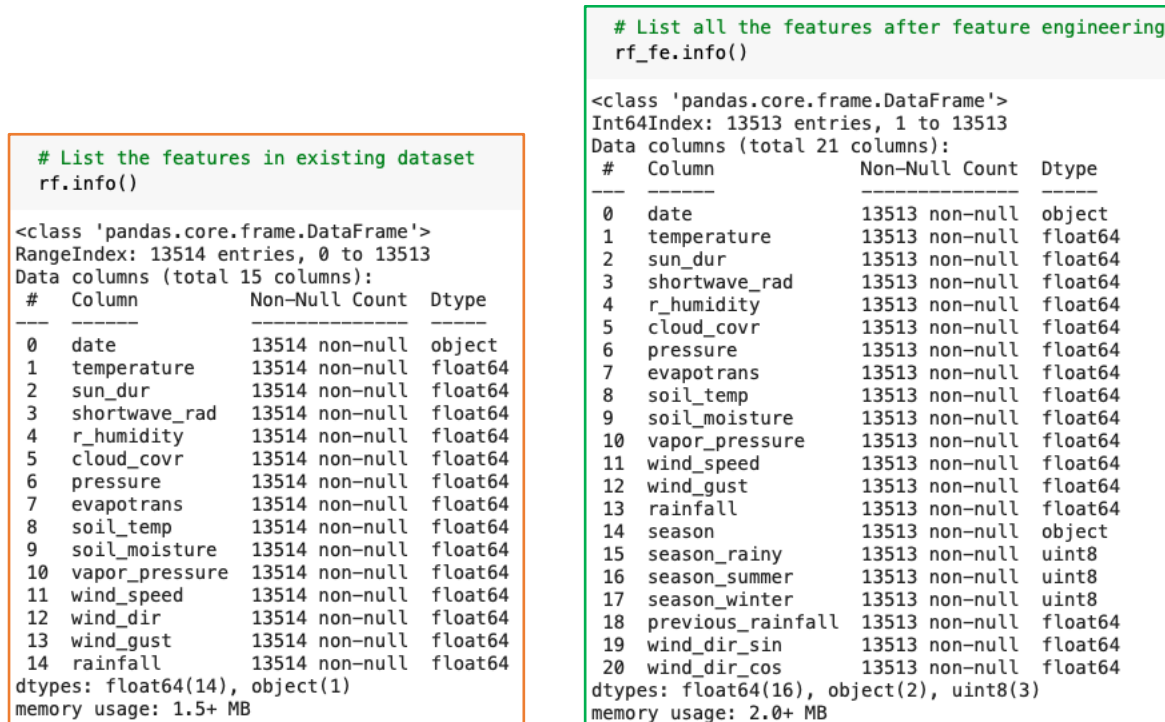


Figure 60: Comparison of existing features and newly created features

4.3.5 Explanatory Data Analysis (EDA) after Feature engineering

In this section, the EDA between some of the newly created variables and target variable will be conducted as followings.

4.3.5.1 Seasonal variable Vs rainfall

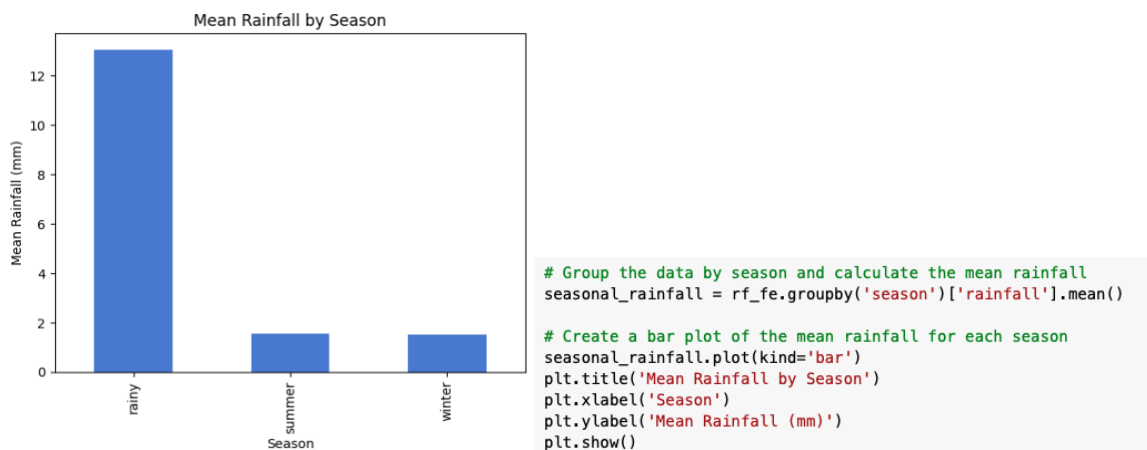


Figure 61: Seasonal variables Vs rainfall variable

Figure (61) analyzes the average amount of rainfall received in each season in Myanmar. It is clear that rainy season yield a significant amount of rainfall than other two season. The importance of these feature will be examined in feature selection process.

4.3.5.2 Previous rainfall Vs today rainfall

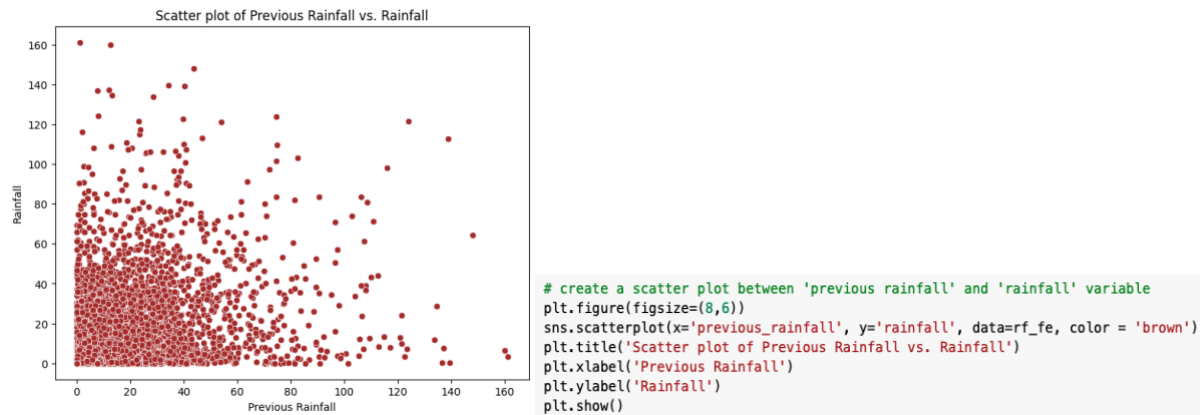


Figure 62: The relationship between previous rainfall and today rainfall amount

The scatter plot in Figure (62) portrays that the relationship between previous rainfall and today rainfall amount rarely correlated. It appears to have some kind non-linear relationship and difficult to determine. The importance of previous rainfall variable will be assessed in feature selection process.

4.4 Feature Selection

The primary aim of feature selection is to obtain the most important features suitable for a machine learning model, thereby improving its performance. The inclusion of redundant and irrelevant features would add up the noise or increase the complexity into the model unnecessarily, which could lower the model's performance while consuming high computational cost. Therefore, in this section, correlation analysis and tree-based feature selection using Random Forest model will be implemented to select the most appropriate features.

4.4.1 Feature selection using Correlation Analysis

One of the most common statistical techniques which could be useful in feature selection is the correlation analysis. The purpose of the analysis is to determine if there is a relationship between the two variables in both directions (negative or positive) and strength (ranging from -1 to +1). It helps find the highly correlated variables which are redundant for feeding a

machine learning model and even in worst scenario, the highly correlated variables could decrease the model's performance. The correlation is conducted among the independent variables. In case highly associated variable are present in the dataset, it is recommended to eliminate. In this project, the cut off values for correlation values of 0.8 and above will be dropped from the dataset. The Spearman correction will be utilized since the data are in non-linear relationship.

4.4.1.1 Correlation Analysis for dataset without soil moisture and evapotranspiration

```
# Back up and drop dependent variable to conduct correlation analysis among independent variables
rf_1 = rf.drop(['date', 'rainfall', 'soil_moisture', 'evapotrans'], axis=1)

# Correlation plots for original dataset
import matplotlib.pyplot as plt
import seaborn as sns
correlations = rf_1.corr('spearman')
mask = np.triu(np.ones_like(correlations, dtype=bool))
f, ax = plt.subplots(figsize = (10, 10))
sns.heatmap(correlations, mask=mask, annot = True, cmap='PiYG', center = 0, fmt = '.1f', square = True)
```

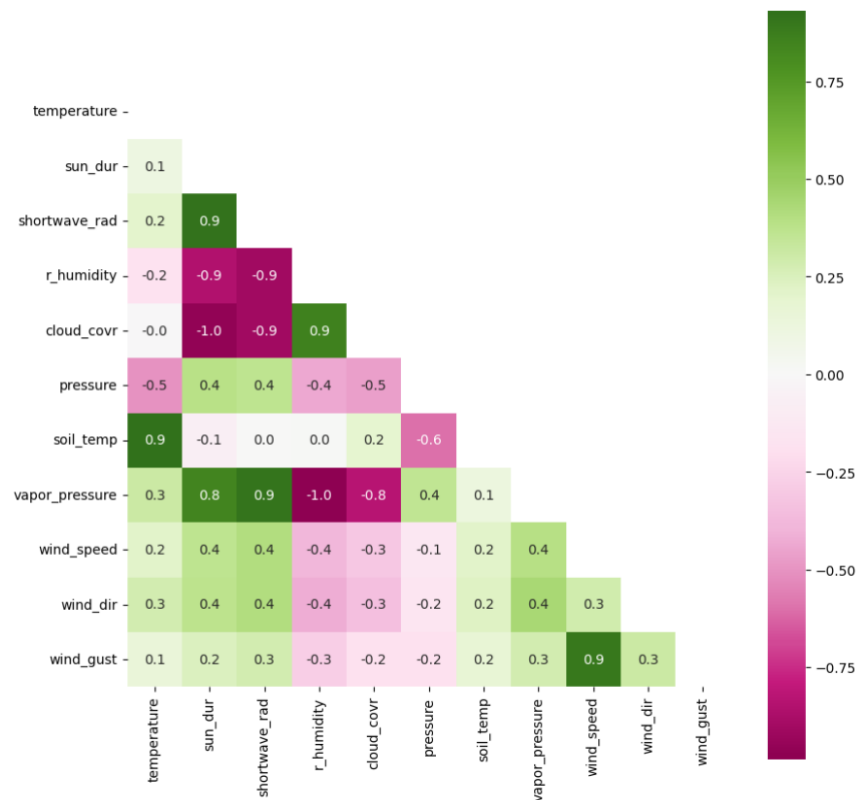


Figure 63: Correlation matrix for independent variables without soil_moisture & evapotrans

The correlation analysis in Figure (63) reveals that some of the variables are highly correlated and some of the variables are perfectly associated. For the purpose of seeing the correlated variable clearly, the list of variable pairs that are having the correlation values of 0.8 and above are generated.

```
# Generate a list of highly correlated variables with cut off >= 0.8
high_corr_vars = np.where(abs(correlations) >= 0.8)
high_corr_vars = [(correlations.columns[x], correlations.columns[y])
                  for x, y in zip(*high_corr_vars) if x != y and x < y]
df_high_corr_vars = pd.DataFrame({'Var_1': [x[0] for x in high_corr_vars],
                                 'Var_2': [x[1] for x in high_corr_vars]})
print(df_high_corr_vars)
```

	Var_1	Var_2
0	temperature	soil_temp
1	sun_dur	shortwave_rad
2	sun_dur	r_humidity
3	sun_dur	cloud_covr
4	sun_dur	vapor_pressure
5	shortwave_rad	r_humidity
6	shortwave_rad	cloud_covr
7	shortwave_rad	vapor_pressure
8	r_humidity	cloud_covr
9	r_humidity	vapor_pressure
10	cloud_covr	vapor_pressure
11	wind_speed	wind_gust

Figure 64: List of highly correlated variable (cut off 0.8 and above)

Based on the list generated in Figure (64), the highly correlated variables having the correlation value of 0.8 and more will be dropped as follow.

```
# List out variables to drop
highly_correlated_list = ['soil_temp', 'sun_dur', 'shortwave_rad',
                          'vapor_pressure', 'wind_gust', 'r_humidity']
# Dropping features
rf_1.drop(highly_correlated_list, axis = 1, inplace = True)
```

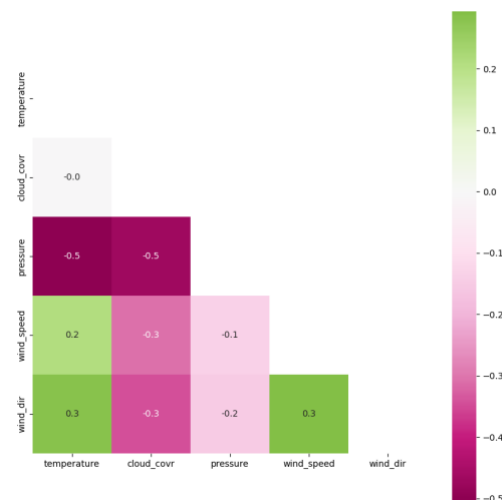


Figure 65: Correlation matrix after dropping highly correlated variables

As seen in Figure (65), the variables in the dataset are not highly correlated and they are less than 0.8.

4.4.1.2 Correlation Analysis for dataset without soil moisture and evapotranspiration with Feature engineered variables

```
# Drop dependent variable to conduct correlation analysis among independent variables
rf_1_fe = rf_fe.drop(['date', 'rainfall', 'season', 'soil_moisture', 'evapotrans'], axis=1)
```

```
# Correlation plots for dataset with new features
import matplotlib.pyplot as plt
import seaborn as sns
correlations_fe = rf_1_fe.corr('spearman')
mask = np.triu(np.ones_like(correlations_fe, dtype=bool))
f, ax = plt.subplots(figsize = (20, 20))
sns.heatmap(correlations_fe, mask=mask, annot = True, cmap='PiYG', center = 0, fmt = '.1f', square = True)
```

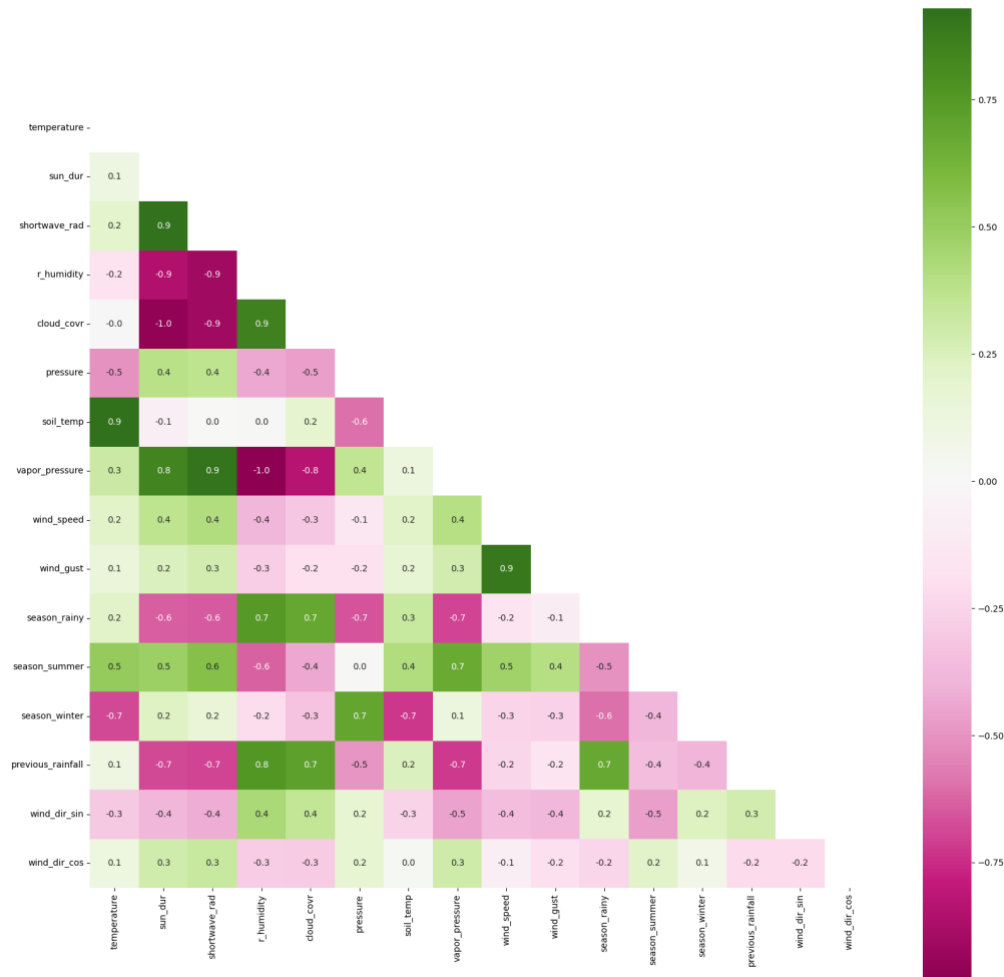


Figure 66: Correlation matrix for independent variables without soil_moisture & evapotrans with feature engineered variables

Figure (66) shows correlation analysis of the independent variables with feature engineered variables. As in previous case, some of the variables are highly correlated and some of the variables are perfectly associated. In order to see the correlated variable clearly, the list of variable pairs that are having the correlation values of 0.8 and above are generated.

```
# Generate a list of highly correlated variables with cut off >= 0.8
high_corr_vars_fe = np.where(abs(correlations_fe) >= 0.8)
high_corr_vars_fe = [(correlations_fe.columns[x], correlations_fe.columns[y])
                    for x, y in zip(*high_corr_vars_fe) if x != y and x < y]
df_high_corr_vars_fe = pd.DataFrame({'Var_1': [x[0] for x in high_corr_vars_fe],
                                     'Var_2': [x[1] for x in high_corr_vars_fe]})
print(df_high_corr_vars_fe)
```


	Var_1	Var_2
0	temperature	soil_temp
1	sun_dur	shortwave_rad
2	sun_dur	r_humidity
3	sun_dur	cloud_covr
4	sun_dur	vapor_pressure
5	shortwave_rad	r_humidity
6	shortwave_rad	cloud_covr
7	shortwave_rad	vapor_pressure
8	r_humidity	cloud_covr
9	r_humidity	vapor_pressure
10	cloud_covr	vapor_pressure
11	wind_speed	wind_gust

Figure 67: List of highly correlated variable with feature engineered variables (cut off 0.8 and above)

According to the list generated in Figure (67), the highly correlated variables having the correlation value of 0.8 and above will be dropped as following.

```
# List out variables to drop
highly_correlated_list_fe = ['soil_temp', 'sun_dur', 'shortwave_rad',
                             'vapor_pressure', 'wind_gust', 'r_humidity']
# Dropping features
rf_1_fe.drop(highly_correlated_list_fe, axis = 1, inplace = True)
```

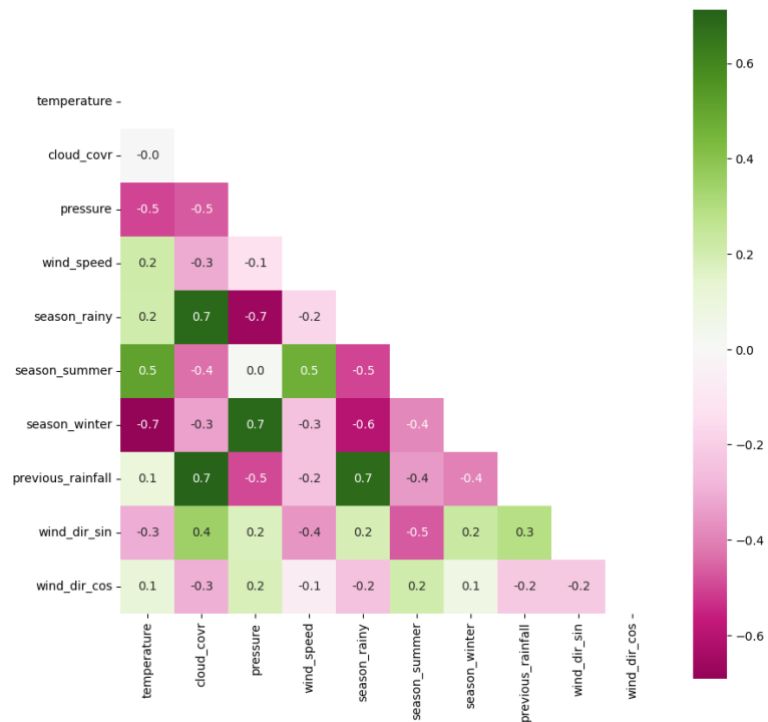


Figure 68: Correlation matrix after dropping highly correlated variables with feature engineered variables

It is clear in Figure (68) that the variables in the dataset are now not highly correlated, and they are less than 0.8.

4.4.1.3 Correlation Analysis for dataset with soil moisture and evapotranspiration

```
# Merge soil_moisture & evapotrans variables to rf_1
rf_2 = pd.concat([rf_1, rf[['soil_moisture', 'evapotrans']]], axis=1)
```

```
# Correlation plots for dataset with new features again
correlations_rf_2 = rf_2.corr('spearman')
mask = np.triu(np.ones_like(correlations_rf_2, dtype=bool))
f, ax = plt.subplots(figsize = (10, 10))
sns.heatmap(correlations_rf_2, mask=mask, annot = True, cmap='PiYG', center = 0, fmt = '.1f', square = True)
```

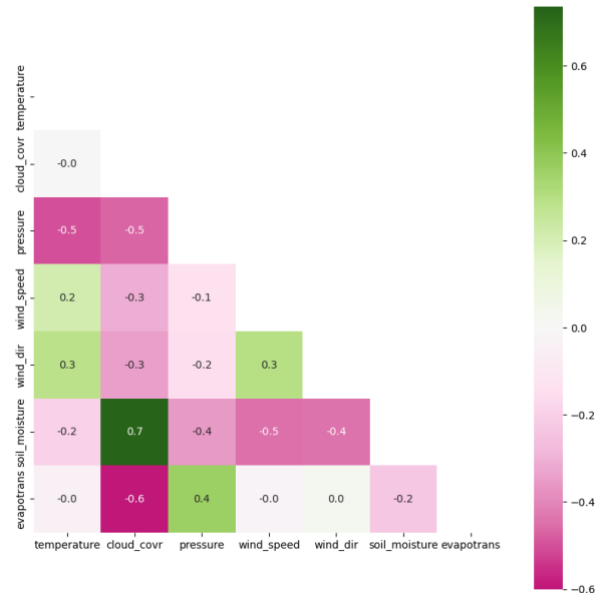


Figure 69: Correlation matrix for independent variables with soil_moisture & evapotrans

The correlation matrix in Figure (69) displays that the soil moisture and evapotranspiration are not highly associated, and their values are less than 0.8. Therefore, no further elimination of variable is required.

4.4.1.4 Correlation Analysis for dataset with soil moisture and evapotranspiration with

Feature engineered variables

```
# Merge soil_moisture & evapotrans variables to rf_1_fe
rf_2_fe = pd.concat([rf_1_fe, rf_fe[['soil_moisture', 'evapotrans']]], axis=1)
```

```
# Correlation plots for dataset with new features again
correlations_rf_2_fe = rf_2_fe.corr('spearman')
mask = np.triu(np.ones_like(correlations_rf_2_fe, dtype=bool))
f, ax = plt.subplots(figsize = (10, 10))
sns.heatmap(correlations_rf_2_fe, mask=mask, annot = True, cmap='PiYG', center = 0, fmt = '.1f', square = True)
```

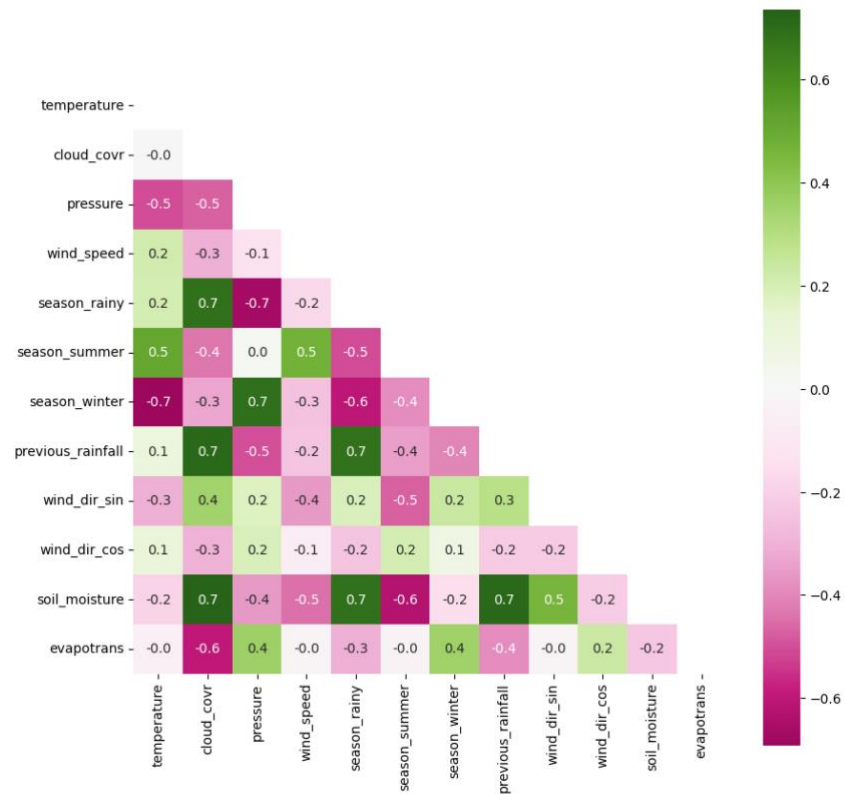


Figure 70: Correlation matrix for independent variables with soil_moisture & evapotrans with feature engineered variables

In Figure (70), the correlation analysis among the independent portrays that they correlation values are less than 0.8 and they are not considered as highly correlated and thus, no additional treatment is needed for these variables.

4.4.2 Feature selection using Tree-based Method

In this project, tree-based feature selection using Random Forest (RF) method is implemented to identify the most importance and relevant variables for predicting rainfall amount. Based on the final feature importance scores, the features will be selected to use for modelling purpose. The following code blocks are used to produce the feature importance scores and the bar chart to see visually. For the sake of convenience for documentation, only the codes used in the data without the soil moisture and evapotranspiration is illustrated.

```
# Import libraries for Tree-based feature selection
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler

# Define dependent and independent variables
X_rf_1 = rf_1
y_rf_1 = rf['rainfall']

# Scale the features using the StandardScaler class
scaler = StandardScaler()
X_rf_1_scaled = pd.DataFrame(scaler.fit_transform(X_rf_1), columns=X_rf_1.columns)

# Initialize a random forest regressor object
rfr_rf_1 = RandomForestRegressor(random_state=42)

# Fit the random forest regressor on the dataset
rfr_rf_1.fit(X_rf_1_scaled, y_rf_1)

# Get feature importance scores
importances_rf_1 = rfr_rf_1.feature_importances_

# Create a dataframe with feature importance scores
feature_importances_rf_1 = pd.DataFrame({'Feature': X_rf_1_scaled.columns, 'Importance': importances_rf_1})

# Sort the features by importance score
feature_importances_rf_1 = feature_importances_rf_1.sort_values('Importance', ascending=False)

# Print the sorted feature importances
print(feature_importances_rf_1)
```

```
# Plot Feature importance in rf_1 dataset
plt.figure(figsize=(8, 6))
plt.bar(feature_importances_rf_1['Feature'], feature_importances_rf_1['Importance'])
plt.xticks(rotation=90)
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances in rf_1 dataset')
plt.show()
```

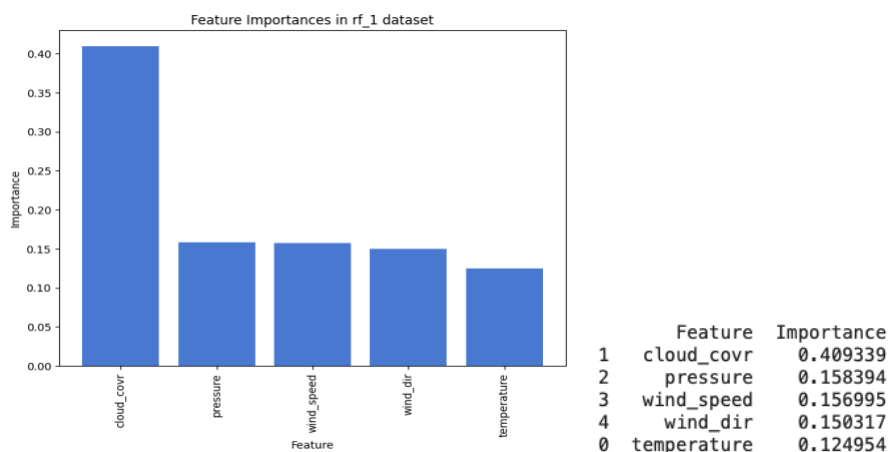


Figure 71: Feature importance scores for dataset without soil moisture and evapotranspiration

Figure (71) displays the feature importance of each variable in predicting the amount of rainfall for the dataset without soil moisture and evapotranspiration. As it can be seen, the cloud coverage variable has the highest feature importance score of 0.4093 while the temperature variable is having the lowest score of 0.1250. All features are seemed to be important and thus they will be used in building the model for this dataset.

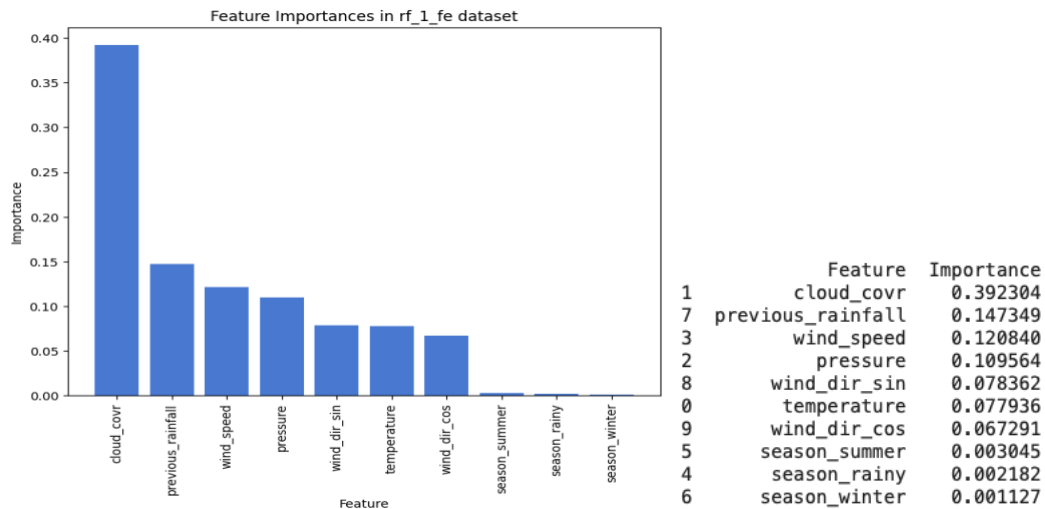


Figure 72: Feature importance scores for dataset without soil moisture and evapotranspiration with feature engineering

According to the information from Figure (72), the cloud coverage variable has the highest feature importance score of 0.3923 whereas the season_winter variable has the lowest score of 0.0011. The season_rainy and season_winter will be dropped since they show very less importance compared to other variables. Although the season_summer is having 0.0030, it will be used since the inclusion of one of the seasonal variables would improve the performance of the model.

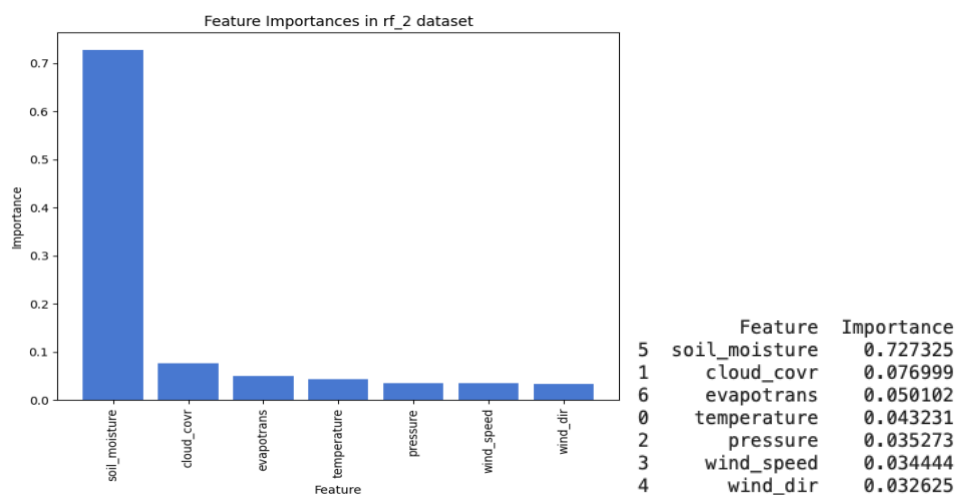


Figure 73: Feature importance scores for dataset with soil moisture and evapotranspiration

In Figure (73), it is clear that the soil moisture variable has the highest feature importance score of 0.7273 while the wind direction variable has the lowest score of 0.0326. Since all features are seemed to be important and therefore, they will be used in modelling.

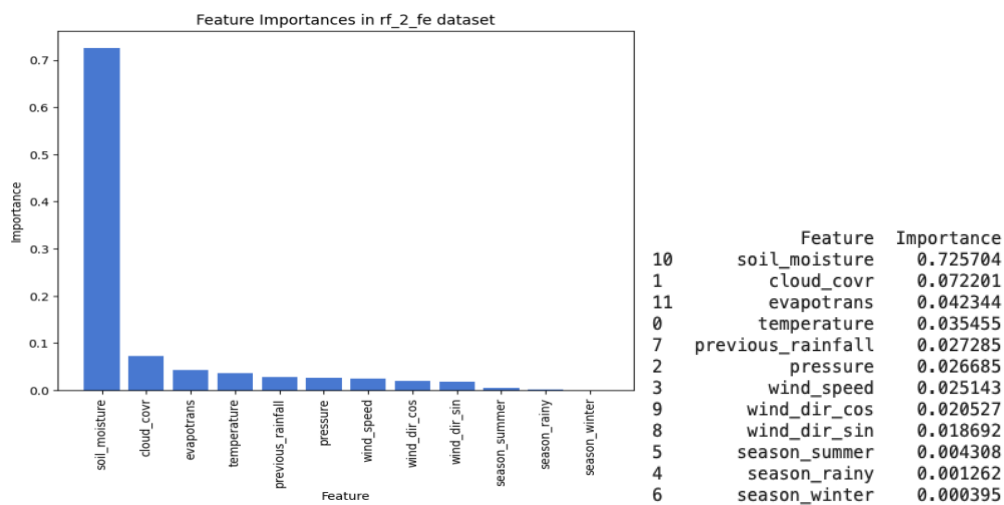


Figure 74: Feature importance scores for dataset with soil moisture and evapotranspiration with feature engineering

The feature importance score and the bar chart in Figure (74) reveal that the soil moisture variable has the largest feature importance score of 0.7257 while the season_winter variable has the lowest score of 0.0004. The season_rainy and season_winter will be dropped since they contribute less as compared to the rest variables. Even though the season_summer has feature importance score of 0.0043, it will be selected as the inclusion of one of the seasonal variables would increase the performance of the model.

4.4.3 Summary of Feature selection

Both correlation matrix and tree-based feature selection method using Random Forest are implemented to remove less important variables. From the correlation analysis, highly correlated variables such as soil temperature, sun duration, short-wave radiation, vapor pressure deficit, wind gust and relative humidity are dropped. Then, feature engineered variables such as season_rainy and season_winter are selected to remove based on tree-based method. The total number of (8) will be eliminated. The variable contains in each dataset after feature engineering are presented in the Table (3).

No.	Dataset name	Total number and name of independent variables in the dataset	Feature engineered (fe)
1	rf_1	cloud_covr, pressure, wind_speed, wind_dir, temperature (Total features = 5)	No
2	rf_1_fe	cloud_covr, pressure, temperature, wind_speed, wind_dir_sin, wind_dir_cos, season_summer, previous_rainfall (Total features = 8)	Yes
3	rf_2	cloud_covr, pressure, wind_speed, wind_dir, temperature, soil_moisture, evapotrans (Total features = 7)	No
4	rf_2_fe	cloud_covr, pressure, wind_speed, temperature, soil_moisture, evapotrans, previous_rainfall, wind_dir_sin, wind_dir_cos, season_summer (Total feature = 10)	Yes

Table 3: Features in the dataset after feature engineering

4.5 Normalization using Z-score

The standard scaler or Z-score normalization is used in this project to normalize the data before building the machine learning model. The technique of Z-score normalization transforms the variables to have a zero mean and the standard deviation value of one. The transformed variable will become a normal distribution while scaled the values which are in different units and scale. These results in making comparable data for the model and improve its performance. The dependent and independent variables are defined, and the four sets of data are scaled using StandardScaler from Sklearn library as in below code blocks in Figure (75). Lastly, the shape of the scaled data frames is then verified whether they are having the right number of variables in the respective scaled data frames.

```

from sklearn.preprocessing import StandardScaler

# Define dependent and independent variables
X_rf_1 = rf_1
y_rf_1 = rf['rainfall']

# Define dependent and independent variables & drop less importance features
X_rf_1_fe = rf_1_fe.drop(['season_rainy', 'season_winter'], axis=1)
y_rf_1_fe = rf_fe['rainfall']

# Define dependent and independent variables
X_rf_2 = rf_2
y_rf_2 = rf['rainfall']

# Define dependent and independent variables & drop less importance features
X_rf_2_fe = rf_2_fe.drop(['season_rainy', 'season_winter'], axis=1)
y_rf_2_fe = rf_fe['rainfall']

# Scale the features using the StandardScaler class (Z-Score Normalization)
scaler_1 = StandardScaler() # Initial the Scalar
scaler_2 = StandardScaler()
scaler_3 = StandardScaler()
scaler_4 = StandardScaler()

# Scaling the features
X_rf_1_scaled = pd.DataFrame(scaler_1.fit_transform(X_rf_1), columns=X_rf_1.columns)
X_rf_1_fe_scaled = pd.DataFrame(scaler_2.fit_transform(X_rf_1_fe), columns=X_rf_1_fe.columns)
X_rf_2_scaled = pd.DataFrame(scaler_3.fit_transform(X_rf_2), columns=X_rf_2.columns)
X_rf_2_fe_scaled = pd.DataFrame(scaler_4.fit_transform(X_rf_2_fe), columns=X_rf_2_fe.columns)

```

```

# Print shape of scaled dataframes
print("Scaled rf_1 dataframe shape :", X_rf_1_scaled.shape)
print("Dependet variable in rf_1 :", y_rf_1.shape)

print("Scaled rf_1_fe dataframe shape :", X_rf_1_fe_scaled.shape)
print("Dependet variable in rf_1_fe :", y_rf_1_fe.shape)

print("Scaled rf_2 dataframe shape :", X_rf_2_scaled.shape)
print("Dependet variable in rf_2 :", y_rf_2.shape)

print("Scaled rf_2_fe dataframe shape :", X_rf_2_fe_scaled.shape)
print("Dependet variable in rf_2_fe :", y_rf_2.shape)

Scaled rf_1 dataframe shape : (13514, 5)
Dependet variable in rf_1 : (13514,)
Scaled rf_1_fe dataframe shape : (13513, 8)
Dependet variable in rf_1_fe : (13513,)
Scaled rf_2 dataframe shape : (13514, 7)
Dependet variable in rf_2 : (13514,)
Scaled rf_2_fe dataframe shape : (13513, 10)
Dependet variable in rf_2_fe : (13514,)

```

Figure 75: Normalization using Standard Scaler

4.6 Data partitioning

Before any algorithms are utilized to construct the model, the four datasets are partitioned using a ratio of 70:30. The 70 % of the dataset will be used as a training data to build the model using the selected models and the remaining 30% will be employed for validating the built models' performance. The testing dataset functions as an unseen dataset for the model and useful for accessing underfitting and overfitting.

```
# Data partitioning using 70:30 ratio
# Data partition for rf_1 dataset
from sklearn.model_selection import train_test_split
X_train_rf_1, X_test_rf_1, y_train_rf_1, y_test_rf_1 = train_test_split(X_rf_1_scaled, y_rf_1,
                                                                    test_size = 0.3, random_state = 42)
print("Training set :", X_train_rf_1.shape, y_train_rf_1.shape)
print("Testing set :", X_test_rf_1.shape, y_test_rf_1.shape)

Training set : (9459, 5) (9459,)
Testing set : (4055, 5) (4055,)
```

```
# Data partition for rf_1_fe dataset
X_train_rf_1_fe, X_test_rf_1_fe, y_train_rf_1_fe, y_test_rf_1_fe = train_test_split(X_rf_1_fe_scaled,
                                                                    y_rf_1_fe, test_size = 0.3, random_state = 42)
print("Training set :", X_train_rf_1_fe.shape, y_train_rf_1_fe.shape)
print("Testing set :", X_test_rf_1_fe.shape, y_test_rf_1_fe.shape)

Training set : (9459, 8) (9459,)
Testing set : (4054, 8) (4054,)
```

```
# Data partition for rf_2 dataset
X_train_rf_2, X_test_rf_2, y_train_rf_2, y_test_rf_2 = train_test_split(X_rf_2_scaled, y_rf_2,
                                                                    test_size = 0.3, random_state = 42)
print("Training set :", X_train_rf_2.shape, y_train_rf_2.shape)
print("Testing set :", X_test_rf_2.shape, y_test_rf_2.shape)

Training set : (9459, 7) (9459,)
Testing set : (4055, 7) (4055,)
```

```
# Data partition for rf_2_fe dataset
X_train_rf_2_fe, X_test_rf_2_fe, y_train_rf_2_fe, y_test_rf_2_fe = train_test_split(X_rf_2_fe_scaled,
                                                                    y_rf_2_fe, test_size = 0.3, random_state = 42)
print("Training set :", X_train_rf_2_fe.shape, y_train_rf_2_fe.shape)
print("Testing set :", X_test_rf_2_fe.shape, y_test_rf_2_fe.shape)

Training set : (9459, 10) (9459,)
Testing set : (4054, 10) (4054,)
```

Figure 76: Data partition using a 70:30 ratio

The above codes in Figure (76) are executed to confirm all the training and testing dataset has the relevant features and ready for modelling purposes. As it can be seen in the output, the training data for rf_1 dataset has (5) independent variables, rf_1 dataset contains (8) predictors while the rf_2 and rf_2_fe dataset have (7) and (10) independent, respectively.

4.7 Modelling and Evaluation Metrics

This stage is the most importance of the project whereby the proposed models adapted from the literature review are implemented using the cleaned datasets. As discovered from the literatures, Random Forest (RF), Support Vector Regressor (SVR), and Artificial Neural Network (ANN) models will be construct in this section. Firstly, the base models for each dataset are modelled and then the model are tuned using the Random Search CV with Ten-fold cross validation is incorporated in the function from Sklearn library to get the most suitable hyperparameter values for the dataset. The evaluation metrics are also generated once the models have been built. The detail discussion of model result and the comparison of the model using the evaluation metrics will be presented in Chapter (5) Results and Discussion.

4.7.1 Random Forest Model

The Random Forest model is built by using the Random Forest Regressor function from Sklearn library with the training set having 70% of the data. The base models are based on four set of data using the common hyper parameters such as number of estimators, max depth, minimum samples split with the random state of 42. The initial setting as are shown in the Table (4).

Hyper parameter	Description	Initial setting
n_estimators	The total number of decision trees to be included in Random Forest. Default value = 100.	150
max_features	The number of features to be set at each split in decision tree. Default value = auto.	-
max_depth	The depth of individual decision tree in maximum. Default = None	10
min_samples_split	The number of samples to split the trees in minimum. Default value = 2	10
min_samples_leaf	The number of samples to be in the leaf node. Default value = 1	5

Table 4: Hyperparameters for Random Forest Model

4.7.1.1 RF model using rf_1 dataset

First of all, the Random Forest Regressor function is imported from the Sklearn library since the target variable is a continuous variable and it is a regression problem. The initial settings are passed into the function when building the model and the RF regressor has been constructed in Figure (77). The remaining datasets are implemented the same method as rf_1 dataset.

```
# Import libraries for building Random Forest model
from sklearn.ensemble import RandomForestRegressor

# Building Random Forest base model using rf_1 dataset
rf_reg_rf_1 = RandomForestRegressor(n_estimators=150, max_depth=10,
                                  min_samples_split=10, min_samples_leaf=5, random_state=42)
rf_reg_rf_1.fit(X_train_rf_1, y_train_rf_1)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=10, min_samples_leaf=5, min_samples_split=10,
                      n_estimators=150, random_state=42)
```

Figure 77: Random Forest Regressor (Base model) using rf_1 dataset

```
# Import library for model tuning with Random Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter distribution to sample from
param_dist = {
    'n_estimators': [50, 100, 150],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [1, 2, 5, 7, 10],
    'min_samples_leaf': [1, 2, 5, 7, 10],
}

# Create a RandomForestRegressor object
rf_reg_rf_1 = RandomForestRegressor(random_state=42)

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(estimator=rf_reg_rf_1, param_distributions=param_dist,
                                  cv=10, n_iter=100, random_state=42)

# Fit the RandomizedSearchCV object to the training data
random_search.fit(X_train_rf_1, y_train_rf_1)

# Get the best parameters and score
best_params = random_search.best_params_
best_score = random_search.best_score_

# Create a new RandomForestRegressor object with the best parameters
rf_reg_best_rf_1 = RandomForestRegressor(**best_params, random_state=42)

# Fit the best model to the training data
rf_reg_best_rf_1.fit(X_train_rf_1, y_train_rf_1)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=10, max_features='log2', min_samples_leaf=7,
                      n_estimators=50, random_state=42)
```

Figure 78: Hyperparameter tuning for RF model using rf_1 dataset

Once the base model has been built, the hyperparameter tuning is initiated with a range of 50, 100 to 150 for the number of estimators, the auto, square root and log2 for maximum features, the values starting from 10 till 30 including none for max depth, the values from 1 to

10 for minimum sample split and minimum sample split, respectively. The optimal parameters provided by the random search are 10 for max depth, log2 for maximum features, 7 for minimum samples leaf and 50 for the number of estimators. The tuned model is built using optimal hyperparameters as in Figure (79) after the evaluation metrics has been generated.

```
# Building Random Forest tuned model using rf_1 dataset
rf_reg_rf_1 = RandomForestRegressor(n_estimators=50, max_depth=10,
                                   min_samples_leaf=7, max_features='log2', random_state=42)
rf_reg_rf_1.fit(X_train_rf_1, y_train_rf_1)
```

RandomForestRegressor

RandomForestRegressor(max_depth=10, max_features='log2', min_samples_leaf=7, n_estimators=50, random_state=42)

Figure 79: Random Forest Tuned model using rf_1 dataset

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Predict on the training set
y_train_pred_rf_1 = rf_reg_rf_1.predict(X_train_rf_1)
# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_1 = mean_squared_error(y_train_rf_1, y_train_pred_rf_1, squared=False)
mae_train_rf_1 = mean_absolute_error(y_train_rf_1, y_train_pred_rf_1)
r2_train_rf_1 = r2_score(y_train_rf_1, y_train_pred_rf_1)

# Predict on the test set
y_test_pred_rf_1 = rf_reg_rf_1.predict(X_test_rf_1)
# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_1 = mean_squared_error(y_test_rf_1, y_test_pred_rf_1, squared=False)
mae_test_rf_1 = mean_absolute_error(y_test_rf_1, y_test_pred_rf_1)
r2_test_rf_1 = r2_score(y_test_rf_1, y_test_pred_rf_1)

# Print the evaluation metrics
print("Evaluation Metrics for RF base model \nusing rf_1 dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_1, 4))
print("Training MAE:", round(mae_train_rf_1, 4))
print("Training R square:", round(r2_train_rf_1, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_1, 4))
print("Testing MAE:", round(mae_test_rf_1, 4))
print("Testing R square:", round(r2_test_rf_1, 4))
```

Figure 80: Generating Evaluation metrics for RF model using rf_1 dataset

4.7.1.2 RF model using rf_1_fe dataset

```
# Building Random Forest base model using rf_1_fe dataset
rf_reg_rf_1_fe = RandomForestRegressor(n_estimators=150, max_depth=10,
                                       min_samples_split=10, min_samples_leaf=5, random_state=42)
rf_reg_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe)
```

RandomForestRegressor

RandomForestRegressor(max_depth=10, min_samples_leaf=5, min_samples_split=10, n_estimators=150, random_state=42)

Figure 81: Random Forest Regressor (Base model) using rf_1_fe dataset

```

# Import library for RF model tuning with Randomo Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter distribution to sample from
param_dist = {
    'n_estimators': [50, 100, 150],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [1, 2, 5, 7, 10],
    'min_samples_leaf': [1, 2, 5, 7, 10],
}

# Create a RandomForestRegressor object
rf_reg_rf_1_fe_t = RandomForestRegressor(random_state=42)

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(estimator=rf_reg_rf_1_fe_t, param_distributions=param_dist,
                                   cv=10, n_iter=100, random_state=42)

# Fit the RandomizedSearchCV object to the training data
random_search.fit(X_train_rf_1_fe, y_train_rf_1_fe)

# Get the best parameters and score
best_params = random_search.best_params_
best_score = random_search.best_score_

# Create a new RandomForestRegressor object with the best parameters
rf_reg_best_rf_1_fe = RandomForestRegressor(**best_params, random_state=42)

# Fit the best model to the training data
rf_reg_best_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe)

```

RandomForestRegressor

RandomForestRegressor(max_depth=10, max_features='log2', min_samples_leaf=10, n_estimators=50, random_state=42)

Figure 82: Hyperparameter tuning for RF model using rf_1_fe dataset

After the base model has been constructed, the hyperparameter tuning is conducted with a range of 50, 100 to 150 for the number of estimators, the auto, square root and log2 for maximum features, the values starting from 10 till 30 including none for max depth, the values from 1 to 10 for minimum sample split and minimum sample split, respectively. The optimal parameters provided by the random search are 10 for max depth, log2 for maximum features, 10 for minimum samples leaf and 50 for the number of estimators. The tuned model is built using optimal hyperparameters in seen below Figure (83) after the evaluation metrics has been generated.

```

# Building Random Forest tuned model using rf_1_fe dataset
rf_reg_rf_1_fe = RandomForestRegressor(max_depth=10, max_features='log2',
                                     min_samples_leaf=10, n_estimators=50, random_state=42)
rf_reg_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe)

```

RandomForestRegressor

RandomForestRegressor(max_depth=10, max_features='log2', min_samples_leaf=10, n_estimators=50, random_state=42)

Figure 83: Random Forest Tuned model using rf_1_fe dataset

```

# Predict on the training set
y_train_pred_rf_1_fe = rf_reg_rf_1_fe.predict(X_train_rf_1_fe)
# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_1_fe = mean_squared_error(y_train_rf_1_fe, y_train_pred_rf_1_fe, squared=False)
mae_train_rf_1_fe = mean_absolute_error(y_train_rf_1_fe, y_train_pred_rf_1_fe)
r2_train_rf_1_fe = r2_score(y_train_rf_1_fe, y_train_pred_rf_1_fe)

# Predict on the test set
y_test_pred_rf_1_fe = rf_reg_rf_1_fe.predict(X_test_rf_1_fe)
# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_1_fe = mean_squared_error(y_test_rf_1_fe, y_test_pred_rf_1_fe, squared=False)
mae_test_rf_1_fe = mean_absolute_error(y_test_rf_1_fe, y_test_pred_rf_1_fe)
r2_test_rf_1_fe = r2_score(y_test_rf_1_fe, y_test_pred_rf_1_fe)

# Print the evaluation metrics
print("Evaluation Metrics for RF base model \nusing rf_1_fe dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_1_fe, 4))
print("Training MAE:", round(mae_train_rf_1_fe, 4))
print("Training R square:", round(r2_train_rf_1_fe, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_1_fe, 4))
print("Testing MAE:", round(mae_test_rf_1_fe, 4))
print("Testing R square:", round(r2_test_rf_1_fe, 4))

```

Figure 84: Generating Evaluation metrics for RF model using rf_1_fe dataset

4.7.1.3 RF model using rf_2 dataset

```

# Buliding Random Forest model using rf_2 dataset
rf_reg_rf_2 = RandomForestRegressor(n_estimators=150, max_depth=10,
                                  min_samples_split=10, min_samples_leaf=5, random_state=42)
rf_reg_rf_2.fit(X_train_rf_2, y_train_rf_2)

```

```

RandomForestRegressor
RandomForestRegressor(max_depth=10, min_samples_leaf=5, min_samples_split=10,
                      n_estimators=150, random_state=42)

```

Figure 85: Random Forest Regressor (Base model) using rf_2 dataset

```

# Import library for RF model tuning with Randomo Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter distribution to sample from
param_dist = {
    'n_estimators': [50, 100, 150],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [1, 2, 5, 7, 10],
    'min_samples_leaf': [1, 2, 5, 7, 10]
}

# Create a RandomForestRegressor object
rf_reg_rf_2_t = RandomForestRegressor(random_state=42)

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(estimator=rf_reg_rf_2_t, param_distributions=param_dist,
                                   cv=10, n_iter=100, random_state=42)

# Fit the RandomizedSearchCV object to the training data
random_search.fit(X_train_rf_2, y_train_rf_2)

# Get the best parameters and score
best_params = random_search.best_params_
best_score = random_search.best_score_

# Create a new RandomForestRegressor object with the best parameters
rf_reg_best_rf_2 = RandomForestRegressor(**best_params, random_state=42)

# Fit the best model to the training data
rf_reg_best_rf_2.fit(X_train_rf_2, y_train_rf_2)

```

```

RandomForestRegressor
RandomForestRegressor(max_features='sqrt', min_samples_leaf=2,
                      min_samples_split=10, random_state=42)

```

Figure 86: Hyperparameter tuning for RF model using rf_2 dataset

Upon the base model has been constructed, the hyperparameter tuning is performed with a range of 50, 100 to 150 for the number of estimators, the auto, square root and log2 for maximum features, the values starting from 10 till 30 including none for max depth, the values from 1 to 10 for minimum sample split and minimum sample split, respectively. The optimal parameters provided by the random search are square root for maximum features, 2 for minimum samples leaf, and 10 for minimum samples split. The tuned model is built again using optimal hyperparameters as in Figure (87) after the evaluation metrics has been obtained.

```
# Buliding Random Forest tuned model using rf_2 dataset
rf_reg_rf_2 = RandomForestRegressor(max_features='sqrt', min_samples_leaf=2,
                                   min_samples_split=10, random_state=42)
rf_reg_rf_2.fit(X_train_rf_2, y_train_rf_2)
```

```
RandomForestRegressor
RandomForestRegressor(max_features='sqrt', min_samples_leaf=2,
                      min_samples_split=10, random_state=42)
```

Figure 87: Random Forest Tuned model using rf_2 dataset

```
# Predict on the training set
y_train_pred_rf_2 = rf_reg_rf_2.predict(X_train_rf_2)
# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_2 = mean_squared_error(y_train_rf_2, y_train_pred_rf_2, squared=False)
mae_train_rf_2 = mean_absolute_error(y_train_rf_2, y_train_pred_rf_2)
r2_train_rf_2 = r2_score(y_train_rf_2, y_train_pred_rf_2)

# Predict on the test set
y_test_pred_rf_2 = rf_reg_rf_2.predict(X_test_rf_2)
# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_2 = mean_squared_error(y_test_rf_2, y_test_pred_rf_2, squared=False)
mae_test_rf_2 = mean_absolute_error(y_test_rf_2, y_test_pred_rf_2)
r2_test_rf_2 = r2_score(y_test_rf_2, y_test_pred_rf_2)

# Print the evaluation metrics
print("Evaluation Metrics for RF Base model \nusing rf_2 dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_2, 4))
print("Training MAE:", round(mae_train_rf_2, 4))
print("Training R square:", round(r2_train_rf_2, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_2, 4))
print("Testing MAE:", round(mae_test_rf_2, 4))
print("Testing R square:", round(r2_test_rf_2, 4))
```

Figure 88: Generating Evaluation metrics for RF model using rf_2 dataset

4.7.1.4 RF model using rf_2_fe dataset

```
# Buliding Random Forest model using rf_2_fe dataset
rf_reg_rf_2_fe = RandomForestRegressor(n_estimators=150, max_depth=10,
                                      min_samples_split=10, min_samples_leaf=5, random_state=42)
rf_reg_rf_2_fe.fit(X_train_rf_2_fe, y_train_rf_2_fe)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=10, min_samples_leaf=5, min_samples_split=10,
                      n_estimators=150, random_state=42)
```

Figure 89: Random Forest Regressor (Base model) using rf_2_fe dataset


```

# Predict on the training set
y_train_pred_rf_2_fe = rf_reg_rf_2_fe.predict(X_train_rf_2_fe)
# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_2_fe = mean_squared_error(y_train_rf_2_fe, y_train_pred_rf_2_fe, squared=False)
mae_train_rf_2_fe = mean_absolute_error(y_train_rf_2_fe, y_train_pred_rf_2_fe)
r2_train_rf_2_fe = r2_score(y_train_rf_2_fe, y_train_pred_rf_2_fe)

# Predict on the test set
y_test_pred_rf_2_fe = rf_reg_rf_2_fe.predict(X_test_rf_2_fe)
# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_2_fe = mean_squared_error(y_test_rf_2_fe, y_test_pred_rf_2_fe, squared=False)
mae_test_rf_2_fe = mean_absolute_error(y_test_rf_2_fe, y_test_pred_rf_2_fe)
r2_test_rf_2_fe = r2_score(y_test_rf_2_fe, y_test_pred_rf_2_fe)

# Print the evaluation metrics
print("Evaluation Metrics for RF base model \nusing rf_2_fe dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_2_fe, 4))
print("Training MAE:", round(mae_train_rf_2_fe, 4))
print("Training R square:", round(r2_train_rf_2_fe, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_2_fe, 4))
print("Testing MAE:", round(mae_test_rf_2_fe, 4))
print("Testing R square:", round(r2_test_rf_2_fe, 4))

```

Figure 92: Generating Evaluation metrics for RF model using rf_2_fe dataset

4.7.2 Support Vector Regressor

The Support Vector Regressor (SVR) model using the kernel of Radial Basic Function (rbf) is implemented by the use of SVR function from Sklearn library with the training dataset having 70% of the data. The 'rbf' kernel is recommended by many researchers in the literature review that the 'rbf' works well with non-linear data such as the weather data in this project and therefore, it is selected to use. The base models are constructed on the four set of data using the common hyper parameters such as the regularization control parameter and the kernel coefficient. The initial setting as are shown in the Table (5).

Hyper parameter	Description	Initial setting
kernel	The kernel is used to separate the data into higher dimensional space. Default value = linear	rbf
C	The regularization parameter to prevent overfitting. Default value = 1.0	1.0
gamma	The coefficient for the kernel. Default value = scale	scale

Table 5: Hyperparameters for Support Vector Regressor

4.7.2.1 SVR model using rf_1 dataset

In order to build the Support Vector Regressor model, the SVR function is imported from the Sklearn library. The initial settings are set in the function when modelling as in Figure (93). The remaining datasets are also implemented the same method as rf_1 dataset.

```
# Import libraries for building Support Vector Regressor
from sklearn.svm import SVR
import numpy as np

# Set random_state = 42
np.random.seed(42)
```

```
# Building SVR base model with RBF kernel using rf_1 dataset
svr_rf_1 = SVR(kernel='rbf')
svr_rf_1.fit(X_train_rf_1, y_train_rf_1)
```

```
▼ SVR
SVR()
```

Figure 93: Support Vector Regressor (Base model) using rf_1 dataset

```
# Import libraries for tuning SVR using Random Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter
param_grid = {'C': [0.1, 1, 10],
              'gamma': [0.01, 0.1, 1, 'scale', 'auto']}

# SVR class with RBF kernel
svr_rbf_rf_1_t = SVR(kernel='rbf')

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(svr_rbf_rf_1_t, param_grid=param_grid, cv=10)

# Fit the grid search object to the training data
random_search.fit(X_train_rf_1, y_train_rf_1)

# Get the best model and its hyperparameters
best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Print the best hyperparameters
print('Best hyperparameters:', best_params)
```

```
Best hyperparameters: {'C': 10, 'gamma': 1}
```

Figure 94: Hyperparameter tuning for SVR model using rf_1 dataset

The hyperparameters are tuned after the base model is created. For SVR, the most common hyperparameters for regularization, which is to prevent the model from overfitting, such as C and 'gamma' are calibrated using Random Search with Ten-fold cross validation as in Figure (75). The param set includes 0.1, 1, 10 for C values and 0.01, 0.1, 1, scale, and auto values for gamma parameter. The optimal hyperparameters obtained after tuning are C = 10 and gamma = 1.

```
# Building SVR tuned model with RBF kernel using rf_1 dataset
svr_rf_1 = SVR(kernel='rbf', C=10, gamma=1)
svr_rf_1.fit(X_train_rf_1, y_train_rf_1)
```

```
▼ SVR
SVR(C=10, gamma=1)
```

Figure 95: Support Vector Regressor Tuned model using rf_1 dataset

The tuned model is built again with the hyperparameters provided by the Random Search as in Figure (95).

```

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Predict on the training set
y_train_pred_rf_1 = svr_rf_1.predict(X_train_rf_1)

# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_1 = mean_squared_error(y_train_rf_1, y_train_pred_rf_1, squared=False)
mae_train_rf_1 = mean_absolute_error(y_train_rf_1, y_train_pred_rf_1)
r2_train_rf_1 = r2_score(y_train_rf_1, y_train_pred_rf_1)

# Predict on the test set
y_test_pred_rf_1 = svr_rf_1.predict(X_test_rf_1)

# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_1 = mean_squared_error(y_test_rf_1, y_test_pred_rf_1, squared=False)
mae_test_rf_1 = mean_absolute_error(y_test_rf_1, y_test_pred_rf_1)
r2_test_rf_1 = r2_score(y_test_rf_1, y_test_pred_rf_1)

# Print the evaluation metrics
print("Evaluation Metrics for SVR base model \using rf_1 dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_1, 4))
print("Training MAE:", round(mae_train_rf_1, 4))
print("Training R square:", round(r2_train_rf_1, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_1, 4))
print("Testing MAE:", round(mae_test_rf_1, 4))
print("Testing R square:", round(r2_test_rf_1, 4))

```

Figure 96: Generating Evaluation metrics for SVR model using rf_1 dataset

4.7.2.2 SVR model using rf_1_fe dataset

```

# Building SVR base model with RBF kernel using rf_1_fe dataset
svr_rf_1_fe = SVR(kernel='rbf')
svr_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe)

```

▼ SVR
SVR()

Figure 97: Support Vector Regressor (Base model) using rf_1_fe dataset

```

# Import libraries for tuning SVR using Random Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter
param_grid = {'C': [0.1, 1, 10],
              'gamma': [0.01, 0.1, 1, 'scale', 'auto']}

# SVR class with RBF kernel
svr_rf_1_fe_t = SVR(kernel='rbf')

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(svr_rf_1_fe_t, param_grid=param_grid, cv=10)

# Fit the grid search object to the training data
random_search.fit(X_train_rf_1_fe, y_train_rf_1_fe)

# Get the best model and its hyperparameters
best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Print the best hyperparameters
print('Best hyperparameters:', best_params)

```

Best hyperparameters: {'C': 10, 'gamma': 0.1}

Figure 98: Hyperparameter tuning for SVR model using rf_1_fe dataset

Once the base model has built with rf_1_fe training dataset, the hyperparameters are tuned using Random Search with Ten-fold cross validation as in Figure (98). The tuning param set includes 0.1, 1, 10 for C values and 0.01, 0.1, 1, scale, and auto values for gamma parameter. The optimal hyperparameters obtained after tuning are $C = 10$ and $\gamma = 0.1$.

```

# Building SVR tuned model with RBF kernel using rf_1_fe dataset
svr_rf_1_fe = SVR(kernel='rbf', C=10, gamma=0.1)
svr_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe)

```

SVR
SVR(C=10, gamma=0.1)

Figure 99: Support Vector Regressor Tuned model using rf_1_fe dataset

Along with the hyperparameters provided by the Random Search, the tuned model is constructed again as in Figure (99).

```

# Predict on the training set
y_train_pred_rf_1_fe = svr_rf_1_fe.predict(X_train_rf_1_fe)

# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_1_fe = mean_squared_error(y_train_rf_1_fe, y_train_pred_rf_1_fe, squared=False)
mae_train_rf_1_fe = mean_absolute_error(y_train_rf_1_fe, y_train_pred_rf_1_fe)
r2_train_rf_1_fe = r2_score(y_train_rf_1_fe, y_train_pred_rf_1_fe)

# Predict on the test set
y_test_pred_rf_1_fe = svr_rf_1_fe.predict(X_test_rf_1_fe)

# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_1_fe = mean_squared_error(y_test_rf_1_fe, y_test_pred_rf_1_fe, squared=False)
mae_test_rf_1_fe = mean_absolute_error(y_test_rf_1_fe, y_test_pred_rf_1_fe)
r2_test_rf_1_fe = r2_score(y_test_rf_1_fe, y_test_pred_rf_1_fe)

# Print the evaluation metrics
print("Evaluation Metrics for SVR base model \nusing rf_1_fe dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_1_fe, 4))
print("Training MAE:", round(mae_train_rf_1_fe, 4))
print("Training R square:", round(r2_train_rf_1_fe, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_1_fe, 4))
print("Testing MAE:", round(mae_test_rf_1_fe, 4))
print("Testing R square:", round(r2_test_rf_1_fe, 4))

```

Figure 100: Generating Evaluation metrics for SVR model using rf_2 dataset

4.7.2.3 SVR model using rf_2 dataset

```

# Building SVR base model with RBF kernel using rf_2 dataset
svr_rf_2 = SVR(kernel='rbf')
svr_rf_2.fit(X_train_rf_2, y_train_rf_2)

```

▼ SVR
SVR()

Figure 101: Support Vector Regressor (Base model) using rf_2 dataset

```

# Import libraries for tuning SVR using Random Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter
param_grid = {'C': [0.1, 1, 10],
              'gamma': [0.01, 0.1, 1, 'scale', 'auto']}

# SVR class with RBF kernel
svr_rf_2_t = SVR(kernel='rbf')

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(svr_rf_2_t, param_grid=param_grid, cv=10)

# Fit the grid search object to the training data
random_search.fit(X_train_rf_2, y_train_rf_2)

# Get the best model and its hyperparameters
best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Print the best hyperparameters
print('Best hyperparameters:', best_params)

Best hyperparameters: {'C': 10, 'gamma': 'scale'}

```

Figure 102: Hyperparameter tuning for SVR model using rf_1 dataset

As soon as the base SVR model has constructed with rf_2 training dataset, the hyperparameters are tuned using Random Search with Ten-fold cross validation as in Figure (102). The tuning param set includes 0.1, 1, 10 for C values and 0.01, 0.1, 1, scale, and auto values for gamma parameter. The optimal hyperparameters obtained after tuning are C = 10 and gamma = 'scale'.

```
# Building SVR tuned model with RBF kernel using rf_2 dataset
svr_rf_2 = SVR(kernel='rbf', C=10, gamma='scale')
svr_rf_2.fit(X_train_rf_2, y_train_rf_2)
```

▼ SVR
SVR(C=10)

Figure 103: Support Vector Regressor Tuned model using rf_2 dataset

The tuned SVR model using rf_2 training dataset is built again using hyperparameters generated by the Random Search as in Figure (103).

```
# Predict on the training set
y_train_pred_rf_2 = svr_rf_2.predict(X_train_rf_2)
# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_2 = mean_squared_error(y_train_rf_2, y_train_pred_rf_2, squared=False)
mae_train_rf_2 = mean_absolute_error(y_train_rf_2, y_train_pred_rf_2)
r2_train_rf_2 = r2_score(y_train_rf_2, y_train_pred_rf_2)

# Predict on the test set
y_test_pred_rf_2 = svr_rf_2.predict(X_test_rf_2)
# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_2 = mean_squared_error(y_test_rf_2, y_test_pred_rf_2, squared=False)
mae_test_rf_2 = mean_absolute_error(y_test_rf_2, y_test_pred_rf_2)
r2_test_rf_2 = r2_score(y_test_rf_2, y_test_pred_rf_2)

# Print the evaluation metrics
print("Evaluation Metrics for SVR base model \nusing rf_2 dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_2, 4))
print("Training MAE:", round(mae_train_rf_2, 4))
print("Training R square:", round(r2_train_rf_2, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_2, 4))
print("Testing MAE:", round(mae_test_rf_2, 4))
print("Testing R square:", round(r2_test_rf_2, 4))
```

Figure 104: Generating Evaluation metrics for SVR model using rf_2 dataset

4.7.2.4 SVR model using rf_2_fe dataset

```
# Building SVR base model with RBF kernel using rf_2_fe dataset
svr_rf_2_fe = SVR(kernel='rbf')
svr_rf_2_fe.fit(X_train_rf_2_fe, y_train_rf_2_fe)
```

▼ SVR
SVR()

Figure 105: Support Vector Regressor (Base model) using rf_2_fe dataset

```

# Import libraries for tuning SVR using Random Search CV
from sklearn.model_selection import RandomizedSearchCV

# Define the parameter
param_grid = {'C': [0.1, 1, 10],
              'gamma': [0.01, 0.1, 1, 'scale', 'auto']}

# SVR class with RBF kernel
svr_rf_2_fe_t = SVR(kernel='rbf')

# Create a RandomizedSearchCV object
random_search = RandomizedSearchCV(svr_rf_2_fe_t, param_grid=param_grid, cv=10)

# Fit the grid search object to the training data
random_search.fit(X_train_rf_2_fe, y_train_rf_2_fe)

# Get the best model and its hyperparameters
best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Print the best hyperparameters
print('Best hyperparameters:', best_params)

```

Best hyperparameters: {'C': 10, 'gamma': 0.1}

Figure 106: Hyperparameter tuning for SVR model using rf_2_fe dataset

The base SVR model with rf_2_fe training dataset is constructed as in Figure (84), then followed by the process of the hyperparameters tuning using Random Search with Ten-fold cross validation as in Figure (106). The tuning param set includes 0.1, 1, 10 for C values and 0.01, 0.1, 1, scale, and auto values for gamma parameter. The optimal hyperparameters received after tuning are $C = 10$ and $\gamma = 0.1$.

```

# Building SVR tuned model with RBF kernel using rf_2_fe dataset
svr_rf_2_fe = SVR(kernel='rbf', C=10, gamma=0.1)
svr_rf_2_fe.fit(X_train_rf_2_fe, y_train_rf_2_fe)

```

▼ SVR

SVR(C=10, gamma=0.1)

Figure 107: Support Vector Regressor Tuned model using rf_2_fe dataset

The tuned model is built again by including the best hyperparameters generated by the Random Search as in Figure (107).

```

# Predict on the training set
y_train_pred_rf_2_fe = svr_rf_2_fe.predict(X_train_rf_2_fe)

# Calculate RMSE, MAE, and R square for training set
rmse_train_rf_2_fe = mean_squared_error(y_train_rf_2_fe, y_train_pred_rf_2_fe, squared=False)
mae_train_rf_2_fe = mean_absolute_error(y_train_rf_2_fe, y_train_pred_rf_2_fe)
r2_train_rf_2_fe = r2_score(y_train_rf_2_fe, y_train_pred_rf_2_fe)

# Predict on the test set
y_test_pred_rf_2_fe = svr_rf_2_fe.predict(X_test_rf_2_fe)

# Calculate RMSE, MAE, and R square for test set
rmse_test_rf_2_fe = mean_squared_error(y_test_rf_2_fe, y_test_pred_rf_2_fe, squared=False)
mae_test_rf_2_fe = mean_absolute_error(y_test_rf_2_fe, y_test_pred_rf_2_fe)
r2_test_rf_2_fe = r2_score(y_test_rf_2_fe, y_test_pred_rf_2_fe)

# Print the evaluation metrics
print("Evaluation Metrics for SVR base model \nusing rf_2_fe dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(rmse_train_rf_2_fe, 4))
print("Training MAE:", round(mae_train_rf_2_fe, 4))
print("Training R square:", round(r2_train_rf_2_fe, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(rmse_test_rf_2_fe, 4))
print("Testing MAE:", round(mae_test_rf_2_fe, 4))
print("Testing R square:", round(r2_test_rf_2_fe, 4))

```

Figure 108: Generating Evaluation metrics for SVR model using rf_2_fe dataset

4.7.3 Artificial Neural Network

The final model selected to implement in this project is the Artificial Neural Network model which has been employed in rainfall prediction in many studies. The model consists of three layers, particularly, the input, output layer and the additional layers in the middle which are hidden layers. The neural network containing more than three layers can be regarded as deep neural network which is not the scope of this project. ANN is well known for its ability to detect the complex relationship and patterns that are impossible to be discovered by human beings. In this project, an ANN model with two hidden layers will be employed as a base model with common initial setting of 64 neurons in input and hidden layers. Since the prediction of rainfall amount is a regression problem, the output layer of activation function is set to linear function, which deals with continuous data prediction. The activation function used in input and hidden layer is 'relu' which stands for Rectified Linear Unit, and it has been chosen because of its ability to efficiently handle the computational capability and it help improves the training speed and the model performance, as suggested in the literature review. The 'adam' optimizer which stands for Adaptive Moment Estimation that is computational efficient and well suited for problems with complex data and high-dimensional such as rainfall data will be used in the ANN models. The below table show the initial setting of hyperparameters for building base model. The base models will be tuned using Random Search with Ten-fold cross validation to avoid overfitting and improve the model's performance.

Hyper parameter	Description	Initial setting
neurons	The number of neurons in each hidden layer. Default value = 1	64
activation	The mathematical function to insert the non-linearity into the output or the layer of neurons. Default value = 'relu'	relu
loss function	The function to evaluate the model's performance. Default value = MSE (Mean Square Error)	mean_squared_error
optimizer	The algorithms for adjustment of the weights and biases to minimize the loss function. Default value = 'adam'	adam
epochs	The number of iterations to update the weights and bias using the entire training data. Default value = '1'	100
batch size	The number of samples used at each iteration to update the weights during the training phase. Default value = 32	32

Table 6: Hyperparameters for Artificial Neural Network model

4.7.3.1 ANN model using rf_1 dataset

The ANN model is built using Keras library. Initially, the model is designed with the help of Sequential function. The number of neurons, hidden layers and the activation function used in each layer are defined using add method. The model is then compiled by inserting loss function and optimizer. Finally, the ANN model is fitted on training dataset as in Figure (109). The hyperparameter settings mentioned in above table is utilized in modelling ANN base models. The same approach is implemented for each four datasets.

```

# Import libraies for buliding ANN models
import keras
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

# Set random state = 42
np.random.seed(42)

# Building ANN base model using rf_1 dataset

# Define the model
ANN_rf_1 = Sequential()
ANN_rf_1.add(Dense(64, input_dim=5, activation='relu'))
ANN_rf_1.add(Dense(64, activation='relu'))
ANN_rf_1.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_1.compile(loss='mean_squared_error', optimizer='adam')

# Train the model
ANN_rf_1.fit(X_train_rf_1, y_train_rf_1, epochs=100, batch_size=32, verbose=0)

# Evaluate the model on the test data
test_loss = ANN_rf_1.evaluate(X_test_rf_1, y_test_rf_1, verbose=0)

# Print the test loss
print('Test loss:', test_loss)

Test loss: 91.83589172363281

```

Figure 109: Artificial Neural Network (Base model) using rf_1 dataset

```

# Import libraries for tuning ANN model using Random Search CV
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import RandomizedSearchCV

# Define the Keras model
def create_model(neurons=1):
    model = Sequential()
    model.add(Dense(neurons, input_dim=5, activation='relu'))
    model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Create the KerasRegressor object
keras_reg = KerasRegressor(build_fn=create_model, verbose=0)

# Define the hyperparameter grid
param_dist = {'neurons': np.arange(16, 128, 16),
              'batch_size': [16, 32, 64, 128],
              'epochs': [50, 100, 200],
              'validation_split': [0.1, 0.2, 0.3]}

# Create the RandomizedSearchCV object
search = RandomizedSearchCV(keras_reg, param_distributions=param_dist, n_iter=10, cv=10)

# Fit the RandomizedSearchCV object to the training data
search.fit(X_train_rf_1, y_train_rf_1)

# Print the best parameters and score
print(search.best_params_)
print(search.best_score_)

{'validation_split': 0.1, 'neurons': 64, 'epochs': 50, 'batch_size': 16}
-102.99927597045898

```

Figure 110: Hyperparameter tuning for ANN model using rf_1 dataset

After the ANN base model has been built and the evaluation metrics are generated, the model is tuned using the Random Search by incorporating Ten-fold cross validation. The pram set include neurons in 16, 128, 16 using np.arrange function, 16, 32, 64, 128 for the batch size,

50, 100, 200 for epochs, 0.1, 0.2, 0.3 for validation split. The best parameters obtained after tuning are 'validation_split' = 0.1, 'neurons' = 64, 'epochs' = 50, 'batch_size' = 16 as in below Figure (110).

```
# Building ANN tuned model with the best hyperparameters using rf_1 dataset

# Set the tuned parameters
params = {'validation_split': 0.1, 'neurons': 64, 'epochs': 50, 'batch_size': 16}

# Define the model
ANN_rf_1 = Sequential()
ANN_rf_1.add(Dense(params['neurons'], input_dim=8, activation='relu'))
ANN_rf_1.add(Dense(params['neurons'], activation='relu'))
ANN_rf_1.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_1.compile(loss='mean_squared_error', optimizer='adam')

# Train the model with the tuned parameters
ANN_rf_1.fit(X_train_rf_1, y_train_rf_1, epochs=params['epochs'], batch_size=params['batch_size'],
            verbose=0, validation_split=params['validation_split'])

# Evaluate the model on the test data
test_loss = ANN_rf_1.evaluate(X_test_rf_1, y_test_rf_1, verbose=0)

# Print the test loss
print('Test loss:', test_loss)

Test loss: 91.86727142333984
```

Figure 111: Artificial Neural Network Tuned model using rf_1 dataset

The tuned model is constructed again with the best hyperparameters provided by the Random Search with Ten-fold cross validation as in Figure (111).

```
# Generate predictions for training and testing set
y_train_pred_rf_1 = ANN_rf_1.predict(X_train_rf_1)
y_test_pred_rf_1 = ANN_rf_1.predict(X_test_rf_1)

# Calculate RMSE for training and testing set
train_rmse_rf_1 = np.sqrt(mean_squared_error(y_train_rf_1, y_train_pred_rf_1))
test_rmse_rf_1 = np.sqrt(mean_squared_error(y_test_rf_1, y_test_pred_rf_1))

# Calculate MAE for training and testing set
train_mae_rf_1 = mean_absolute_error(y_train_rf_1, y_train_pred_rf_1)
test_mae_rf_1 = mean_absolute_error(y_test_rf_1, y_test_pred_rf_1)

# Calculate R-squared for training and testing set
train_r2_rf_1 = r2_score(y_train_rf_1, y_train_pred_rf_1)
test_r2_rf_1 = r2_score(y_test_rf_1, y_test_pred_rf_1)

# Print the evaluation metrics
print("Evaluation Metrics for ANN base model \nusing rf_1 dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(train_rmse_rf_1, 4))
print("Training MAE:", round(train_mae_rf_1, 4))
print("Training R square:", round(train_r2_rf_1, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(test_rmse_rf_1, 4))
print("Testing MAE:", round(test_mae_rf_1, 4))
print("Testing R square:", round(test_r2_rf_1, 4))
```

Figure 112: Generating Evaluation metrics for ANN model using rf_1 dataset

4.7.3.2 ANN model using rf_1_fe dataset

```
# Building ANN base model using rf_1_fe dataset

# Define the model
ANN_rf_1_fe = Sequential()
ANN_rf_1_fe.add(Dense(64, input_dim=8, activation='relu'))
ANN_rf_1_fe.add(Dense(64, activation='relu'))
ANN_rf_1_fe.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_1_fe.compile(loss='mean_squared_error', optimizer='adam')

# Train the model
ANN_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe, epochs=100, batch_size=32, verbose=0)

# Evaluate the model on the test data
test_loss = ANN_rf_1_fe.evaluate(X_test_rf_1_fe, y_test_rf_1_fe, verbose=0)

# Print the test loss
print('Test loss:', test_loss)
```

Test loss: 98.15483093261719

Artificial Neural Network (Base model) using rf_1_fe dataset

```
# Import libraries for tuning ANN model using Random Search CV
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import RandomizedSearchCV

# Define the Keras model
def create_model(neurons=1):
    model = Sequential()
    model.add(Dense(neurons, input_dim=8, activation='relu'))
    model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Create the KerasRegressor object
keras_reg = KerasRegressor(build_fn=create_model, verbose=0)

# Define the hyperparameter grid
param_dist = {'neurons': np.arange(16, 128, 16),
              'batch_size': [16, 32, 64, 128],
              'epochs': [50, 100, 200],
              'validation_split': [0.1, 0.2, 0.3]}

# Create the RandomizedSearchCV object
search = RandomizedSearchCV(keras_reg, param_distributions=param_dist, n_iter=10, cv=10)

# Fit the RandomizedSearchCV object to the training data
search.fit(X_train_rf_1_fe, y_train_rf_1_fe)

# Print the best parameters and score
print(search.best_params_)
print(search.best_score_)
```

```
{'validation_split': 0.1, 'neurons': 48, 'epochs': 50, 'batch_size': 16}
-92.6961441040039
```

Figure 113: Hyperparameter tuning for ANN model using rf_1_fe dataset

Once the base ANN model has been constructed and the evaluation metrics are generated, the model is then tuned using the Random Search with Ten-fold cross validation. The parameter set includes neurons in 16, 128, 16 using np.arange function, 16, 32, 64, 128 for the batch size, 50, 100, 200 for epochs, 0.1, 0.2, 0.3 for validation split. The best parameters received after tuning are 'validation_split' = 0.1, 'neurons' = 48, 'epochs' = 50, 'batch_size' = 16 as in below Figure (113).

```

# Building ANN tuned model with the best hyperparameters using rf_1_fe dataset

# Set the tuned parameters
params = {'validation_split': 0.1, 'neurons': 48, 'epochs': 50, 'batch_size': 16}

# Define the model
ANN_rf_1_fe = Sequential()
ANN_rf_1_fe.add(Dense(params['neurons'], input_dim=8, activation='relu'))
ANN_rf_1_fe.add(Dense(params['neurons'], activation='relu'))
ANN_rf_1_fe.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_1_fe.compile(loss='mean_squared_error', optimizer='adam')

# Train the model with the tuned parameters
ANN_rf_1_fe.fit(X_train_rf_1_fe, y_train_rf_1_fe, epochs=params['epochs'], batch_size=params['batch_size'],
                verbose=0, validation_split=params['validation_split'])

# Evaluate the model on the test data
test_loss = ANN_rf_1_fe.evaluate(X_test_rf_1_fe, y_test_rf_1_fe, verbose=0)

# Print the test loss
print('Test loss:', test_loss)

```

Test loss: 96.13118743896484

Figure 114: ANN Tuned model using rf_1_fe dataset

The tuned ANN model using rf_1_fe dataset is modelled again with the best hyperparameters given by the Random Search with Ten-fold cross validation as in Figure (114).

```

# Generate predictions for training and testing set
y_train_pred_rf_1_fe = ANN_rf_1_fe.predict(X_train_rf_1_fe)
y_test_pred_rf_1_fe = ANN_rf_1_fe.predict(X_test_rf_1_fe)

# Calculate RMSE for training and testing set
train_rmse_rf_1_fe = np.sqrt(mean_squared_error(y_train_rf_1_fe, y_train_pred_rf_1_fe))
test_rmse_rf_1_fe = np.sqrt(mean_squared_error(y_test_rf_1_fe, y_test_pred_rf_1_fe))

# Calculate MAE for training and testing set
train_mae_rf_1_fe = mean_absolute_error(y_train_rf_1_fe, y_train_pred_rf_1_fe)
test_mae_rf_1_fe = mean_absolute_error(y_test_rf_1_fe, y_test_pred_rf_1_fe)

# Calculate R-squared for training and testing set
train_r2_rf_1_fe = r2_score(y_train_rf_1_fe, y_train_pred_rf_1_fe)
test_r2_rf_1_fe = r2_score(y_test_rf_1_fe, y_test_pred_rf_1_fe)

# Print the evaluation metrics
print("Evaluation Metrics for ANN base model \nusing rf_1_fe dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(train_rmse_rf_1_fe, 4))
print("Training MAE:", round(train_mae_rf_1_fe, 4))
print("Training R square:", round(train_r2_rf_1_fe, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(test_rmse_rf_1_fe, 4))
print("Testing MAE:", round(test_mae_rf_1_fe, 4))
print("Testing R square:", round(test_r2_rf_1_fe, 4))

```

Figure 115: Generating Evaluation metrics for ANN model using rf_1_fe dataset

4.7.3.3 ANN model using rf_2 dataset

```
# Building ANN base model using rf_2 dataset

# Define the model
ANN_rf_2 = Sequential()
ANN_rf_2.add(Dense(64, input_dim=7, activation='relu'))
ANN_rf_2.add(Dense(64, activation='relu'))
ANN_rf_2.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_2.compile(loss='mean_squared_error', optimizer='adam')

# Train the model
ANN_rf_2.fit(X_train_rf_2, y_train_rf_2, epochs=100, batch_size=32, verbose=0)

# Evaluate the model on the test data
test_loss = ANN_rf_2.evaluate(X_test_rf_2, y_test_rf_2, verbose=0)

# Print the test loss
print('Test loss:', test_loss)
```

Test loss: 32.10271072387695

Figure 116: Artificial Neural Network (Base model) using rf_2 dataset

```
# Import libraries for tuning ANN model using Random Search CV
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import RandomizedSearchCV

# Define the Keras model
def create_model(neurons=1):
    model = Sequential()
    model.add(Dense(neurons, input_dim=7, activation='relu'))
    model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Create the KerasRegressor object
keras_reg = KerasRegressor(build_fn=create_model, verbose=0)

# Define the hyperparameter grid
param_dist = {'neurons': np.arange(16, 128, 16),
              'batch_size': [16, 32, 64, 128],
              'epochs': [50, 100, 200],
              'validation_split': [0.1, 0.2, 0.3]}

# Create the RandomizedSearchCV object
search = RandomizedSearchCV(keras_reg, param_distributions=param_dist, n_iter=10, cv=10)

# Fit the RandomizedSearchCV object to the training data
search.fit(X_train_rf_2, y_train_rf_2)

# Print the best parameters and score
print(search.best_params_)
print(search.best_score_)

{'validation_split': 0.2, 'neurons': 32, 'epochs': 200, 'batch_size': 128}
-35.861480712890625
```

Figure 117: Hyperparameter tuning for ANN model using rf_2 dataset

Later the base ANN model using rf_2 dataset has been created and the evaluation metrics are generated, the model is then tuned using the Random Search by incorporating Ten-fold cross validation. The param set include neurons in 16, 128, 16 using np.arrange function, 16, 32, 64, 128 for the batch size, 50, 100, 200 for epochs, 0.1, 0.2, 0.3 for validation split. The best parameters obtained after tuning are 'validation_split' = 0.2, 'neurons' = 32, 'epochs' = 200, 'batch_size' = 128 as in below Figure (117).

```

# Building ANN Tuned model with the best parameters using rf_2 dataset

# Set the tuned parameters
params = {'validation_split': 0.2, 'neurons': 32, 'epochs': 200, 'batch_size': 128}

# Define the model
ANN_rf_2 = Sequential()
ANN_rf_2.add(Dense(params['neurons'], input_dim=7, activation='relu'))
ANN_rf_2.add(Dense(params['neurons'], activation='relu'))
ANN_rf_2.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_2.compile(loss='mean_squared_error', optimizer='adam')

# Train the model with the tuned parameters
ANN_rf_2.fit(X_train_rf_2, y_train_rf_2, epochs=params['epochs'], batch_size=params['batch_size'],
            verbose=0, validation_split=params['validation_split'])

# Evaluate the model on the test data
test_loss = ANN_rf_2.evaluate(X_test_rf_2, y_test_rf_2, verbose=0)

# Print the test loss
print('Test loss:', test_loss)

```

Test loss: 33.18529510498047

Figure 118: Artificial Neural Network Tuned model using rf_2 dataset

The tuned ANN model using rf_2 dataset is built again with the best hyperparameters gained from Random Search with Ten-fold cross validation as in Figure (118).

```

# Generate predictions for training and testing set
y_train_pred_rf_2 = ANN_rf_2.predict(X_train_rf_2)
y_test_pred_rf_2 = ANN_rf_2.predict(X_test_rf_2)

# Calculate RMSE for training and testing set
train_rmse_rf_2 = np.sqrt(mean_squared_error(y_train_rf_2, y_train_pred_rf_2))
test_rmse_rf_2 = np.sqrt(mean_squared_error(y_test_rf_2, y_test_pred_rf_2))

# Calculate MAE for training and testing set
train_mae_rf_2 = mean_absolute_error(y_train_rf_2, y_train_pred_rf_2)
test_mae_rf_2 = mean_absolute_error(y_test_rf_2, y_test_pred_rf_2)

# Calculate R-squared for training and testing set
train_r2_rf_2 = r2_score(y_train_rf_1, y_train_pred_rf_2)
test_r2_rf_2 = r2_score(y_test_rf_1, y_test_pred_rf_2)

# Print the evaluation metrics
print("Evaluation Metrics for ANN base model \nusing rf_2 dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:" , round(train_rmse_rf_2, 4))
print("Training MAE:" , round(train_mae_rf_2, 4))
print("Training R square:", round(train_r2_rf_2, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:" , round(test_rmse_rf_2, 4))
print("Testing MAE:" , round(test_mae_rf_2, 4))
print("Testing R square:", round(test_r2_rf_2, 4))

```

Figure 119: Generating Evaluation metrics for ANN model using rf_2 dataset

4.7.3.4 ANN model using rf_2_fe dataset

```
# Building ANN base model using rf_2_fe dataset

# Define the model
ANN_rf_2_fe = Sequential()
ANN_rf_2_fe.add(Dense(64, input_dim=10, activation='relu'))
ANN_rf_2_fe.add(Dense(64, activation='relu'))
ANN_rf_2_fe.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_2_fe.compile(loss='mean_squared_error', optimizer='adam')

# Train the model
ANN_rf_2_fe.fit(X_train_rf_2_fe, y_train_rf_2_fe, epochs=100, batch_size=32, verbose=0)

# Evaluate the model on the test data
test_loss = ANN_rf_2_fe.evaluate(X_test_rf_2_fe, y_test_rf_2_fe, verbose=0)

# Print the test loss
print('Test loss:', test_loss)
```

Test loss: 34.38838195800781

Figure 120: Artificial Neural Network (Base model) using rf_2_fe dataset

```
# Import libraries for tuning ANN model using Random Search CV
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import RandomizedSearchCV

# Define the Keras model
def create_model(neurons=1):
    model = Sequential()
    model.add(Dense(neurons, input_dim=10, activation='relu'))
    model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# Create the KerasRegressor object
keras_reg = KerasRegressor(build_fn=create_model, verbose=0)

# Define the hyperparameter grid
param_dist = {'neurons': np.arange(16, 128, 16),
              'batch_size': [16, 32, 64, 128],
              'epochs': [50, 100, 200],
              'validation_split': [0.1, 0.2, 0.3]}

# Create the RandomizedSearchCV object
search = RandomizedSearchCV(keras_reg, param_distributions=param_dist, n_iter=10, cv=10)

# Fit the RandomizedSearchCV object to the training data
search.fit(X_train_rf_2_fe, y_train_rf_2_fe)

# Print the best parameters and score
print(search.best_params_)
print(search.best_score_)
```

```
{'validation_split': 0.1, 'neurons': 96, 'epochs': 50, 'batch_size': 32}
-34.52014675140381
```

Figure 121: Hyperparameter tuning for ANN model using rf_2_fe dataset

Following the ANN base model is created, and the evaluation metrics are produced, the model is tuned using the Random Search along with Ten-fold cross validation. The parameter set include neurons in 16, 128, 16 using np.arrange function, 16, 32, 64, 128 for the batch size, 50, 100, 200 for epochs, 0.1, 0.2, 0.3 for validation split. The best parameters gained after tuning are 'validation_split' = 0.1, 'neurons' = 96, 'epochs' = 50, 'batch_size' = 32 as in below Figure (121).


```

# Building ANN Tuned model using rf_2_fe dataset

# Set the tuned parameters
params = {'validation_split': 0.1, 'neurons': 96, 'epochs': 50, 'batch_size': 32}

# Define the model
ANN_rf_2_fe = Sequential()
ANN_rf_2_fe.add(Dense(params['neurons'], input_dim=10, activation='relu'))
ANN_rf_2_fe.add(Dense(params['neurons'], activation='relu'))
ANN_rf_2_fe.add(Dense(1, activation='linear'))

# Compile the model
ANN_rf_2_fe.compile(loss='mean_squared_error', optimizer='adam')

# Train the model with the tuned parameters
ANN_rf_2_fe.fit(X_train_rf_2_fe, y_train_rf_2_fe, epochs=params['epochs'], batch_size=params['batch_size'],
                verbose=0, validation_split=params['validation_split'])

# Evaluate the model on the test data
test_loss = ANN_rf_2_fe.evaluate(X_test_rf_2_fe, y_test_rf_2_fe, verbose=0)

# Print the test loss
print('Test loss:', test_loss)

Test loss: 33.6565055847168

```

Figure 122: Artificial Neural Network Tuned model using rf_2_fe dataset

The tuned model is then created again using the best hyperparameters yielded by the Random Search with Ten-fold cross validation as in Figure (122).

```

# Generate predictions for training and testing set
y_train_pred_rf_2_fe = ANN_rf_2_fe.predict(X_train_rf_2_fe)
y_test_pred_rf_2_fe = ANN_rf_2_fe.predict(X_test_rf_2_fe)

# Calculate RMSE for training and testing set
train_rmse_rf_2_fe = np.sqrt(mean_squared_error(y_train_rf_2_fe, y_train_pred_rf_2_fe))
test_rmse_rf_2_fe = np.sqrt(mean_squared_error(y_test_rf_2_fe, y_test_pred_rf_2_fe))

# Calculate MAE for training and testing set
train_mae_rf_2_fe = mean_absolute_error(y_train_rf_2_fe, y_train_pred_rf_2_fe)
test_mae_rf_2_fe = mean_absolute_error(y_test_rf_2_fe, y_test_pred_rf_2_fe)

# Calculate R-squared for training and testing set
train_r2_rf_2_fe = r2_score(y_train_rf_2_fe, y_train_pred_rf_2_fe)
test_r2_rf_2_fe = r2_score(y_test_rf_2_fe, y_test_pred_rf_2_fe)

# Print the evaluation metrics
print("Evaluation Metrics for ANN base model \nusing rf_2_fe dataset\n")
print("=====TRAINING SET=====")
print("Training RMSE:", round(train_rmse_rf_2_fe, 4))
print("Training MAE:", round(train_mae_rf_2_fe, 4))
print("Training R square:", round(train_r2_rf_2_fe, 4))
print("")
print("=====TESTING SET=====")
print("Testing RMSE:", round(test_rmse_rf_2_fe, 4))
print("Testing MAE:", round(test_mae_rf_2_fe, 4))
print("Testing R square:", round(test_r2_rf_2_fe, 4))

```

Figure 123: Generating Evaluation metrics for ANN model using rf_2_fe dataset

4.7 Summary of experimentation

In this chapter, the step-by-step explanation of every stage in experimentation and implementation starting from data exploration phase till the modelling and generating of evaluation metrics are conveyed. The features are thoroughly explored to understanding them better and the issues related with data such as missing values and anomalies are identified. Then, the data are pre-processed based on the finding in the previous stage and they are properly cleaned. The featuring engineering is also conducted, the data are normalized using z-score method and partitioned into train and test data set using a 70:30 ratio before building the models. The models are then constructed using different datasets which contains only the traditional weather data and with additional hydrological data. The base models are trained primarily and then the model are tuned and their evaluation metrics are generated. The implemented Python codes as well as the explanations are provided in detail in this chapter.

CHAPTER 5

RESULT AND DISCUSSION

5.0 Introduction

This chapter validate and discuss the results attained by comparing the models built with traditional weather data and the models that contains additional hydrological data. All the models are assessed using the proposed evaluation metrics in the methodology. Three different type of machine learning algorithms are utilized in this project, namely, Random Forest, Support Vector Machine and Artificial Neural Network. The best performing model will be selected for deployment by comparing the evaluation metrics.

5.1 Evaluation Metrics

The performance of the models are assessed based on the evaluation metrics such as RMSE, MAE, and R Square. Since RMSE and MAE are the error produced by the model, the lower values in RMSE and MAE means the model is performing well. However, for R Square values, it ranges from 0 to 1 and because the R Square allow to see the total variance in the model, the model having higher R Square values performs better than other models. All the models will also be verified for overfitting issues using the training and testing metrics. On the occasion that the training and testing metrics are comparable, they are not considered as overfitted model.

5.2 Random Forest Model (RF)

The first four models built with Random Forest Regressor using the traditional weather data with and without feature engineering are shown in below tables. The dataset that has the name with 'fe' contains feature engineered variables which have created in the experimentation stage.

5.2.1 Random Forest Model with traditional weather data

<i>Base model</i>	<i>Tuned model</i>
Evaluation Metrics for RF base model using rf_1 dataset =====TRAINING SET===== Training RMSE: 8.0236 Training MAE: 3.6335 Training R square: 0.6845 =====TESTING SET===== Testing RMSE: 9.5565 Testing MAE: 4.4058 Testing R square: 0.4683	Evaluation Metrics for RF Tuned model using rf_1 dataset =====TRAINING SET===== Training RMSE: 8.666 Training MAE: 3.9308 Training R square: 0.6319 =====TESTING SET===== Testing RMSE: 9.5212 Testing MAE: 4.4244 Testing R square: 0.4722

Table 7: Comparison of RF models built with traditional data

The Table (7) shows the evaluation metrics of the base and tuned models of RF using traditional weather data without feature engineering. When the base model was built with initial hyperparameter setting as seen on the left, the RMSE values of testing set is 9.5565 while that of training set is 8.0236. The difference between the two are quite comparable, but, looking up the R-square values, the testing set just have 0.4683 while the training set has 0.6845, their difference is a bit large. Therefore, the model can be considered slightly overfit. After it has been tuned as seen on the right side, the RMSE, MAE and R Square values became closer, and the model performance is increased.

<i>Base model with FE</i>	<i>Tuned model with FE</i>
Evaluation Metrics for RF base model using rf_1_fe dataset =====TRAINING SET===== Training RMSE: 7.3494 Training MAE: 3.2775 Training R square: 0.7202 =====TESTING SET===== Testing RMSE: 9.8633 Testing MAE: 4.2514 Testing R square: 0.5079	Evaluation Metrics for RF Tuned model using rf_1_fe dataset =====TRAINING SET===== Training RMSE: 8.2368 Training MAE: 3.684 Training R square: 0.6486 =====TESTING SET===== Testing RMSE: 9.8702 Testing MAE: 4.2686 Testing R square: 0.5072

Table 8: Comparison of RF models built with traditional data and feature engineering

The Table (8) represents the comparison of the evaluation metrics between the base and tuned model using traditional weather data with feature engineering. Looking up the base model with feature engineering on the left, the RMSE and MAE values of testing set became 9.8633 and 4.2514. Comparing with the base model in Table (7), the errors are not much improved, but the R Square became subtly increased from 0.4683 to 0.5079 since more variables are added into the model. Although the model is tuned as seen on the right, the metrics are not that better than the base model with feature engineering.

By comparing the Table (7) and (8), it is clear that feature engineering does not bring much improvement in terms of accuracy compared to models without feature engineered variables.

5.2.2 Random Forest Model with additional hydrological data

<i>Base model</i>	<i>Tuned model</i>
Evaluation Metrics for RF Base model using rf_2 dataset	Evaluation Metrics for RF Tuned model using rf_2 dataset
=====TRAINING SET=====	=====TRAINING SET=====
Training RMSE: 4.6334	Training RMSE: 4.2762
Training MAE: 2.2756	Training MAE: 1.9749
Training R square: 0.8948	Training R square: 0.9104
=====TESTING SET=====	=====TESTING SET=====
Testing RMSE: 5.8092	Testing RMSE: 5.8607
Testing MAE: 2.6933	Testing MAE: 2.7516
Testing R square: 0.8035	Testing R square: 0.8

Table 9: Comparison of RF models built with hydrological data

The evaluation metrics of the base and models which contains additional hydrological data are shown in Table (9). The base model has RMSE and MAE values of 5.8092 and 2.6933 for the testing dataset as seen on left side and the RMSE values of training dataset is 4.63354, having a difference of around 1. Meanwhile, the MAE and R Square values are quite similar to each other and thus the model can be considered a good fit. Unfortunately, the model performance does not significantly optimize after the model is tuned as seen on the right side of Table (9).

<i>Base model with FE</i>	<i>Tuned model with FE</i>
Evaluation Metrics for RF base model using rf_2_fe dataset	Evaluation Metrics for RF Tuned model using rf_2_fe dataset
=====TRAINING SET=====	=====TRAINING SET=====
Training RMSE: 4.445	Training RMSE: 4.702
Training MAE: 2.1285	Training MAE: 2.2192
Training R square: 0.8977	Training R square: 0.8855
=====TESTING SET=====	=====TESTING SET=====
Testing RMSE: 5.9551	Testing RMSE: 5.9923
Testing MAE: 2.7508	Testing MAE: 2.7534
Testing R square: 0.8206	Testing R square: 0.8184

Table 10: Comparison of RF models built with hydrological data and feature engineering

The Random Forest base and tuned models which incorporate with hydrological data and feature engineering are illustrated in Table (10). The RMSE value of the testing set has 5.9551 and that of training set is 4.445, the difference being around 1.5. At the same time, the MAE values and R Square values are very close and thus the model is not considered overfit.

The R Square values is boosted from 0.5079 to 0.8206 compared to the base model in Table (8). After tuning, the model is somewhat improved as seen on the right side of Table (10).

No.	Model	Dataset	Train/Test	RMSE	MAE	R ²
1.	RF - Tuned model	rf_1	Test	9.5212	4.4244	0.4722
2.	RF - Tuned model	rf_1_fe	Test	9.8702	4.2686	0.5072
3.	RF - Tuned model	rf_2	Test	5.8607	2.7516	0.8000
4.	RF - Tuned model	rf_2_fe	Test	5.9923	2.7534	0.8184

Table 11: Comparison of RF Tuned models.

To summarize, the best performing Random Forest model is the model which has additional hydrological data without feature engineering. As observed in the Table (11), the performance of the model is the best in terms of R Square, RMSE and MAE values and the models also has a good fit.

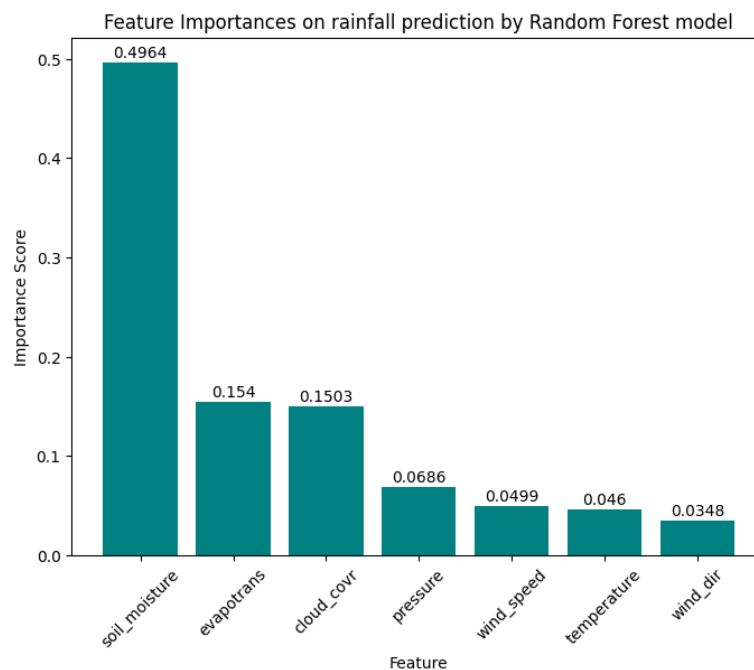


Figure 124: Feature importance on rainfall prediction by best performing Random Forest Model

According to the Figure (124), the feature importance on prediction of rainfall revealed by the Random Forest model is greatly rely upon hydrological data such as the soil moisture. It has the feature importance scores of 0.4964. In the bivariate analysis, a positive correlation was found between soil moisture and rainfall, meaning that any additional increase in the soil moisture could result in the higher amount of rainfall. The evapotranspiration follows at the

second importance and since the relationship between evapotranspiration and rainfall are negatively associated, any increase in the evapotranspiration, would bring the decrease in rainfall amount. Then, the traditional weather data are followed, and the least feature importance is scored by wind direction.

5.3 Support Vector Regressor (SVR)

The Support Vector Regressor models are built with using traditional weather data and with hydrological data using with and without feature engineering. The SVR models are compared whether their performance is improved.

5.3.1 Support Vector Regressor with traditional weather data

<i>Base model</i>	<i>Tuned model</i>
Evaluation Metrics for SVR base model using rf_1 dataset =====TRAINING SET===== Training RMSE: 11.4541 Training MAE: 4.673 Training R square: 0.3569 =====TESTING SET===== Testing RMSE: 10.4504 Testing MAE: 4.2689 Testing R square: 0.3642	Evaluation Metrics for SVR Tuned model using rf_1 dataset =====TRAINING SET===== Training RMSE: 10.3194 Training MAE: 4.0908 Training R square: 0.478 =====TESTING SET===== Testing RMSE: 9.9274 Testing MAE: 4.1683 Testing R square: 0.4262

Table 12: Comparison of SVR models built with traditional data

Table (12) represents the comparison of the base and tuned models of SVR using only with traditional weather data. The RMSE, MAE and R Square values of testing dataset have 10.4504, 4.2689, and 0.3642, respectively. When they are compared with the training dataset metrics, they are comparable, and the difference is not significant. Therefore, the base model does not show any sign of overfit. After the model is tuned, the performance of the model vastly improved since the RMSE, MAE values became lower, and the R square values has increased from 0.3642 to 0.4262. The evaluation metrics of training and testing dataset are very close to each other and thus the tuned model is having a good fit.

<i>Base model with FE</i>	<i>Tuned model with FE</i>
Evaluation Metrics for SVR base model using rf_1_fe dataset =====TRAINING SET===== Training RMSE: 10.828 Training MAE: 4.3593 Training R square: 0.3927 =====TESTING SET===== Testing RMSE: 11.1621 Testing MAE: 4.3579 Testing R square: 0.3697	Evaluation Metrics for SVR Tuned model using rf_1_fe dataset =====TRAINING SET===== Training RMSE: 10.2345 Training MAE: 4.1407 Training R square: 0.4574 =====TESTING SET===== Testing RMSE: 10.6665 Testing MAE: 4.1921 Testing R square: 0.4245

Table 13: Comparison of SVR models built with traditional data and feature engineering

The Support Vector Regressor base and tuned model using traditional weather data with feature engineering are compared as in Table (13). The base model's RMSE, MAE and R square values of testing set yields 11.1621, 4.3579, and 0.3697, respectively. These metrics are comparable with training dataset and thus, the model is a good fit. The model after tuning demonstrates that the performance of the model including all the metrics has improved as seen on the right side of Table (13). However, when in contrast with these models to the models without feature engineering in Table (12), there is not much improvement in terms of accuracy and the variance of the model. Even, the models without feature engineering performs better.

5.3.2 Support Vector Regressor with hydrological data

<i>Base model</i>	<i>Tuned model</i>
Evaluation Metrics for SVR base model using rf_2 dataset =====TRAINING SET===== Training RMSE: 8.9493 Training MAE: 3.7154 Training R square: 0.6074 =====TESTING SET===== Testing RMSE: 8.2212 Testing MAE: 3.4168 Testing R square: 0.6065	Evaluation Metrics for SVR Tuned model using rf_2 dataset =====TRAINING SET===== Training RMSE: 7.2697 Training MAE: 3.1155 Training R square: 0.741 =====TESTING SET===== Testing RMSE: 6.9129 Testing MAE: 2.9929 Testing R square: 0.7218

Table 14: Comparison of SVR models built with hydrological data

In Table (14), the base and tuned model of SVR using both traditional data and hydrological data are evaluated. The testing set has the RMSE, MAE and R Square values of 8.2212, 3.4168 and 0.605. The values are quite closer with the training set RMSE, MAE and R Square values. Thus, the model can be assumed having a good fit. The tuned model on the right side of Table (14) displays that the performance of the model became better in terms of accuracy and variance of the model. When these models are compared with the models in Table (13) and (14) which just used the traditional data, they demonstrate almost two times boosted in model performance either in base or tuned model.

<i>Base model with FE</i>	<i>Tuned model with FE</i>
Evaluation Metrics for SVR base model using rf_2_fe dataset =====TRAINING SET===== Training RMSE: 8.9049 Training MAE: 3.6502 Training R square: 0.5893 =====TESTING SET===== Testing RMSE: 9.3184 Testing MAE: 3.6644 Testing R square: 0.5607	Evaluation Metrics for SVR Tuned model using rf_2_fe dataset =====TRAINING SET===== Training RMSE: 7.1756 Training MAE: 3.0411 Training R square: 0.7333 =====TESTING SET===== Testing RMSE: 7.7034 Testing MAE: 3.1761 Testing R square: 0.6998

Table 15: Comparison of SVR models built with hydrological data and feature engineering

Table (15) exhibits the comparison of the base and tuned models of SVR which use the traditional weather data by incorporation with hydrological data. The testing set has the RMSE, MAE and R Square values of 9.3184, 3.6644, and 0.5607, respectively. The evaluation metrics values of the training set are not so different from the testing set and thus the model is said to be having a good fit. After the model is tuned as seen on the right side of Table (15), the model's performance has significantly improved than the base model.

No.	Model	Dataset	Train/Test	RMSE	MAE	R ²
1.	SVR - Tuned model	rf_1	Test	9.9274	4.1683	0.4262
2.	SVR - Tuned model	rf_1_fe	Test	10.6665	4.1921	0.4245
3.	SVR - Tuned model	rf_2	Test	6.9129	2.9929	0.7218
4.	SVR - Tuned model	rf_2_fe	Test	7.7034	3.1761	0.6998

Table 16: The comparison of SVR Tuned models

To summarize, the best performing Support Vector Regressor model is the model which has additional hydrological data without feature engineering. As observed in the Table (16), the performance of the model is the best in terms of R Square, RMSE and MAE values and the models also has a good fit.

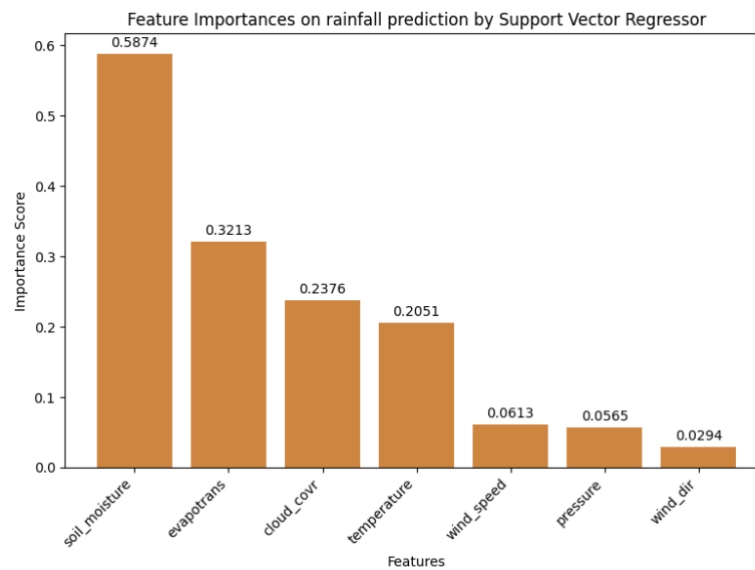


Figure 125: Feature importance on rainfall prediction by best performing Support Vector Regressor Model

Figure (125) illustrates the feature importance scores for predicting rainfall in descending order by the best performing Support Vector Regressor. It is clear from the bar chart that the soil moisture has the highest feature importance score of 0.5874, meaning about 50 % of the soil moisture can effect on the predictability of the rainfall. In the bivariate analysis it was noted that the soil moisture and rainfall were positively correlated, which means that the higher the soil moisture, the chance of having higher rainfall is increased. The evapotranspiration can be ranked as second with the score of 0.3213. The variable has a negative correlation and that any increase in evapotranspiration would result in lower rainfall. The traditional weather data such as cloud coverage, temperature, wind speed, pressure and wind direction are followed after the hydrological data.

5.4 Artificial Neural Network

The Artificial Neural Network models are also built with using traditional weather data and with hydrological data using with and without feature engineering. The ANN models are compared whether their performance is improved.

5.4.1 Artificial Neural Network with traditional weather data

<i>Base model</i>	<i>Tuned model</i>
Evaluation Metrics for ANN base model using rf_1 dataset =====TRAINING SET===== Training RMSE: 9.6479 Training MAE: 4.5833 Training R square: 0.5438 =====TESTING SET===== Testing RMSE: 9.5831 Testing MAE: 4.574 Testing R square: 0.4653	Evaluation Metrics for ANN Tuned model using rf_1 dataset =====TRAINING SET===== Training RMSE: 9.9455 Training MAE: 4.3993 Training R square: 0.5152 =====TESTING SET===== Testing RMSE: 9.5847 Testing MAE: 4.2529 Testing R square: 0.4652

Table 17: Comparison of ANN models built with traditional data

In Table (17), the evaluation metrics of the base and tuned ANN model using only traditional data is compared. As it can be seen the RMSE, MAE and R Square of the testing dataset in base model has 9.5831, 4.574, and 0.4653 and they are similar to the values of training set and thus the model is deduced as a good fit model. After the model has tuned, except the MAE values, the performance of the tuned model is almost the same with the base model.

<i>Base model with FE</i>	<i>Tuned model with FE</i>
Evaluation Metrics for ANN base model using rf_1_fe dataset =====TRAINING SET===== Training RMSE: 8.8575 Training MAE: 3.9939 Training R square: 0.5936 =====TESTING SET===== Testing RMSE: 9.9073 Testing MAE: 4.2285 Testing R square: 0.5035	Evaluation Metrics for ANN Tuned model using rf_1_fe dataset =====TRAINING SET===== Training RMSE: 9.1469 Training MAE: 4.1621 Training R square: 0.5666 =====TESTING SET===== Testing RMSE: 9.8047 Testing MAE: 4.2529 Testing R square: 0.5137

Table 18: Comparison of ANN models built with traditional data and feature engineering

The evaluation metrics of the base and tuned ANN models using traditional weather data with feature engineering are shown in Table (18). The R Square, RMSE and MAE values of base model have 0.5035, 9.9073, and 4.2285. These values are close to the training dataset and thus, the base model is having a good fit. After the model is tuned as seen on the right side of Table (18), the performance of the model is improved in terms of RMSE and R Square and the tuned model is neither under nor overfit.

The comparison of the ANN models only using the traditional weather data with and without feature engineering reveals that the model without feature engineered variables perform better either in base or tuned model.

5.4.2 Artificial Neural Network with traditional weather data

<i>Base model</i>	<i>Tuned model</i>
Evaluation Metrics for ANN base model using rf_2 dataset	Evaluation Metrics for ANN Tuned model using rf_2 dataset
===== Training RMSE: 5.3583 Training MAE: 2.8372 Training R square: 0.8593	===== Training RMSE: 5.6507 Training MAE: 2.8477 Training R square: 0.8435
===== Testing RMSE: 5.6659 Testing MAE: 2.877 Testing R square: 0.8131	===== Testing RMSE: 5.7607 Testing MAE: 2.7804 Testing R square: 0.8068

Table 19: Comparison of ANN models built with hydrological data

The base and tuned ANN models built with traditional data and hydrological data are displayed in Table (19). The inclusion of hydrological data provides the RMSE, MAE and R Square values of 5.6659, 2.877, and 0.8131 in the testing dataset. The training dataset also get the similar result with testing set in base model that the model is not overfit. In the meantime, the model with hydrological data yields 0.8131 and that about 30% increase in the variance of the model, just by adding two variables. The tune models performance also almost the same with the base model and both models are having a good fit.

<i>Base model with FE</i>	<i>Tuned model with FE</i>
Evaluation Metrics for ANN base model using rf_2_fe dataset	Evaluation Metrics for ANN Tuned model using rf_2_fe dataset
===== Training RMSE: 5.1182 Training MAE: 2.5406 Training R square: 0.8643	===== Training RMSE: 5.2192 Training MAE: 2.6501 Training R square: 0.8589
===== Testing RMSE: 5.8642 Testing MAE: 2.7724 Testing R square: 0.826	===== Testing RMSE: 5.7337 Testing MAE: 2.7829 Testing R square: 0.8337

Table 20: Comparison of ANN models built with hydrological data and feature engineering

The ANN models using both traditional data and hydrological data with featuring engineering are illustrated in the Table (20). The base model on the left demonstrates the RMSE, MAE and R Square values of 5.8642, 2.7724 and 0.26, respectively, in the testing dataset. The evaluation metrics of the training as well as testing dataset are close to each other and therefore the base model is considered not overfit. After the model is tuned on right, the performance has improved in terms of RMSE and R Square values. Since the training and testing set's evaluation are very close to each other. The tuned model is also considered a good fit.

No.	Model	Dataset	Train/Test	RMSE	MAE	R ²
1.	ANN - Tuned model	rf_1	Test	9.5847	4.2529	0.4652
2.	ANN - Tuned model	rf_1_fe	Test	9.8047	4.2529	0.5137
3.	ANN - Tuned model	rf_2	Test	5.7607	2.7804	0.8068
4.	ANN - Tuned model	rf_2_fe	Test	5.7337	2.7829	0.8337

Table 21: The comparison of ANN Tuned models

To summarize, it can be seen from the Table (21) that the ANN model trained with both traditional data and hydrological data, containing feature engineered variables is performing the best in terms of RMSE and R Square although it has slightly higher MAE values of 2.7829 compared to the model without feature engineered variables.

5.5 Models Evaluation and Discussion

No.	Model	Dataset	Train/Test	RMSE	MAE	R ²
1.	ANN - Tuned model	rf_2_fe	Test	5.7337	2.7829	0.8337
2.	RF - Tuned model	rf_2	Test	5.8607	2.7516	0.8000
3.	SVR - Tuned model	rf_2	Test	6.9129	2.9929	0.7218

Table 22: The comparison of best performing models using RF, SVR and ANN

In Table (22), the models' result built with three different algorithms are shown. Each model has already tuned and conducted Ten-fold cross validation, meaning the models has been tested on subsets of multiple datasets and thus the model performance is more reliable than testing on single dataset. This method helps to minimize the risk overfitting issues and at the same time, it can provide more accurate evaluation.

The first model, a tuned ANN, has achieved an MAE of 5.7337, a RMSE of 2.7829, and a test coefficient of determination (R-squared) of 0.8337. In terms of MAE and RMSE, these figures indicate the average magnitude of errors in the predictions provided by the model. The R-squared value, on the other hand, gives a measure of how well the model is performing. A R-squared value of 0.8337 indicates that the model fits the data well in this instance.

This second model, a RF with tuning, is reported to have achieved a test MAE of 5.8607, a test RMSE of 2.7516, and a test R-squared value of 0.8000. The results are slightly worse than those obtained by the ANN model, with a slightly higher MAE and RMSE and a lower R-squared value. As a result, there is still a reasonable level of performance for this model,

suggesting that it can capture patterns in the data to a certain extent. According to the third model, an SVR with tuning, the MAE has been accomplished at 6.7129, the RMSE at 2.9929, and the R-squared has been achieved at 0.7218.1 Among the three models, this is the worst results and thus, the SVR model may not be an appropriate choice for this dataset.

It is the ANN model which performs the best among the models. It shows the lowest RMSE values of 5.7337 and the highest values of 0.8337. Even though the ANN model has slightly higher MAE value of 2.7829 compared to the Random Forest model's MAE value of 2.7516, it is regarded as the best performing model because the other two metrics are performing better than other models. The R Square value is 0.8337, meaning the variance of the dependent variable, which is the amount of rainfall, can be explained by 83.37 % of the independent variables including both traditional and hydrological.

5.6 Models comparison

In this section, the models trained only with traditional data and those trained incorporation with hydrological data will be compared and evaluated whether the inclusion of hydrological data contributes to the better performance of the rainfall prediction models. The dataset named rf_1 dataset only contains the traditional weather variables such as wind speed and direction, pressure, and temperature whereas the rf_2 dataset consists of both the traditional and hydrological additional variables such as soil moisture and evapotranspiration. Their comparisons are as below. In experimentation stage, the base and tuned model are built and here, only the tuned models with better accuracy and variance will be assessed.

5.6.1 Random Forest Model

Model	Dataset	Train/Test	RMSE	MAE	R ²
RF Model without Hydrological data	rf_1	Test	9.5212	4.4244	0.4722
RF Model with Hydrological data	rf_2	Test	5.8607	2.7516	0.8000

Table 23: Comparison of Random Forest model with and without Hydrological data

It can be seen from the above Table (23) that the first model that is built only with traditional data and it achieved 9.5212 for RMSE, 4.4244 for MAE and 0.4722 for R Square values. In contrast, the second model which include both traditional weather data and hydrological data provided 5.8607 for RMSE, 2.7516 for MAE and 0.80 for R Square value.

Based on the result, it appears that the inclusion of hydrological data significantly improved the performance of rainfall predicting. In addition, the R Square became significantly higher, demonstrating that a larger variance in the rainfall variable could be explained by the model which has additional hydrological data.

5.6.2 Support Vector Regressor

Model	Dataset	Train/Test	RMSE	MAE	R ²
SVR Model without Hydrological data	rf_1	Test	9.9274	4.1683	0.4262
SVR Model with Hydrological data	rf_2	Test	6.9129	2.9929	0.7218

Table 24: Comparison of Support Vector Regressor model with and without Hydrological data

A look at Table (24), the first model which only uses traditional data, had a RMSE of 9.9274, MAE of 4.1683, and R-Square of 0.4262. However, the second model, which included both traditional weather data and hydrological data, produced RMSE of 6.9129, MAE of 2.9929, and R-Square of 0.7218. A significant improvement in rainfall prediction performance was demonstrated by these results by the inclusion of hydrological data. Furthermore, the higher R-Square value of the second model suggests that the hydrological model can explain a greater portion of the variation in rainfall variable.

5.6.3 Artificial Neural Network

Model	Dataset	Train/Test	RMSE	MAE	R ²
ANN Model without Hydrological data	rf_1	Test	9.5847	4.2529	0.4652
ANN Model with Hydrological data	rf_2	Test	5.7607	2.7804	0.8068

Table 25: Comparison of Artificial Neural Network model with and without Hydrological data

Table (25) shows that the first model, which used only traditional data as input, obtained a RMSE of 9.5847, a MAE of 4.2529, and a R Square of 0.4652. Comparatively, the second model, which incorporated both traditional weather data and hydrological data, provided 5.7607 RMSE, 2.7804 MAE, and 0.8068 R Square value. Considering the results, it appears that the inclusion of hydrological information improved rainfall prediction performance

significantly. Furthermore, the R Square increased significantly, indicating that the model with hydrological data accounts for a larger proportion of the variance in rainfall.

5.7 Models validation with previous relevant studies

Researcher	Variable used	ML Model	Result
Dash et al. (2018)	8	Extreme Learning Machines (ELM)	$R^2 - 0.99$, RMSE - 6.000, MAE -3.149,
Tharun et al. (2018)	10	Random Forest (RF)	$R^2 - 0.979$, RMSE- 5.228
Hartigan et al. (2020)	11	Support Vector Regressor (SVR)	$R^2 - 0.85$, RMSE - 613.704
Current study	10	Artificial Neural Network (ANN)	$R^2 - 0.83$, RMSE - 5.73, MAE - 2.78,
Pham et al. (2020)	5	Support Vector Regressor (SVR)	$R^2 - 0.74$, MAE - 2.728,
Sureh et al. (2019)	7	Support Vector Regressor (SVR)	$R^2 - 0.37$, RMSE - 14.58, MAE - 6.73

Table 26: Comparing the current study with previous related work

Table (26) illustrates the comparison of previous related work with this current study. Although many articles in the literature, the only studies that use similar kind of evaluation metrics are selected. The RMSE and MAE values could be varied according to the range of target variable and hence, R Square is utilized to validate this study with other relevant studies. According to the table, the more variables are in the model, the higher the variance and the lower the error are. In the other studies, generally, Support Vector Regressor is commonly used, however, the ELM is the best and captured up to 90% of the information only with eight variables in the study of (Dash et al., 2018). Tharun et al. (2018) and Hartigan et al. (2020) also obtained 0.97 R Square with then variable and 0.85 R Square with 11 variables, respectively. They used RF and SVR in their studies. This project achieved the best model with 0.83 R Square with ten variables, which is acceptable and less variable were used compared to (Hartigan et al., 2020). The other two studies used fewer variable and their results are not satisfactory. Overall, it could be included that the current study result is comparable to other relevant work and having the 0.83 R Square, the model performance could be considered acceptable.

5.8 Deployment of the best model

In accordance with the result and analysis in above section, the best model which yields the highest accuracy is the Artificial Neural Network model. The model uses both the traditional weather data and hydrological data with feature engineered variable. As it is proposed in methodology, the best model will be deployed using the Streamlit Community Cloud. Before the model is deployed on the Streamlit, the model is required to save as a 'h5' model file. The best ANN model which uses the rf_2_fe data will be saved using the save method from the Keras library as below.

```
# Save model for deployment
ANN_rf_2_fe.save('ANN_rf_2_fe_t_5.7337.h5')

# For saving scalar for deployment
from sklearn.preprocessing import StandardScaler
from joblib import dump, load

# save the scaler object using joblib
dump(scaler_4, 'scaler_4.joblib')
```

In addition to saving the model, it is imperative to scale and pre-process the input data before feeding into the model because if this is not done properly, the model will output incorrect result. Therefore, firstly the Standard Scaler used for building the ANN best model is also saved with dump function from Joblib library. The model file and scalar file are downloaded from the Google Colab environment into the local machine.

Next, the process of the model deployment is performed using Visual Studio code to develop the web interface. The model will be running at the backend and output the result if the user is requested by the use of "Predict" button. The stages in deploying the model are explained in detail below.

```
1| # Import required libraries for deployment
2| import streamlit as st
3| import pandas as pd
4| from sklearn.preprocessing import StandardScaler
5| from joblib import dump, load
6| import numpy as np
7| from keras.models import load_model
8| import tensorflow
9|
10| # Setting custom title and BG color
11| st.set_page_config(
12|     page_title = "Rainfall Prediction",
13|     page_icon = '🌧️'
14| )
15| st.markdown(
16|     """
17|     <style>
18|     .main {
19|     background-color: #a4f2 ;
20|     }
21|     </style>
22|     """, unsafe_allow_html=True)
23|
24| # Display banner image
25| st.image('resources/banner.png')
```

Firstly, a new python scrip file is created, and the required libraries are imported. The libraires used in building the model is required to be imported such as Keras, Pandas, NumPy. The tensorflow library is also imported because the load function from Kears libray is dependent on the library of tensorflow. Then, the background color and banner images are added to the interface.

```

27 # load the saved scaler object
28 scaler = load('model/scaler_4.joblib')
29
30 # Load pre-built ANN model
31 ANN_model = load_model('model/ANN_rf_2_fe_t.h5')
32
33 # Collect data to predict
34 st.write("Fill in the input features to predict the rainfall.")
35 with st.form(key="values"):
36     col1, col2, col3 = st.columns(3)
37     with col1:
38         input_temp = st.number_input("Temperature", min_value=0.0, max_value=30.0, step=0.1)
39         input_season_summer = st.selectbox("Season", ["Summer", "Rainy", "Winter"])
40         if input_season_summer == "Summer":
41             input_season_summer = int(1)
42         else:
43             input_season_summer = int(0)
44         input_pressure = st.slider("Pressure", min_value=995.0, max_value=1030.0, step=0.1)
45     with col2:
46         input_wind_sp = st.number_input("Wind Speed", min_value=0.0, max_value=12.0, step=0.1)
47         input_soil_moisture = st.number_input("Soil Moisture", min_value=0.00, max_value=0.5, step=0.01)
48         input_wind_dir = st.slider("Wind Direction", min_value=1.0, max_value=360.0, step=0.1)
49     with col3:
50         input_evapotrans = st.number_input("Evapotranspiration", min_value=0.0, max_value=10.0, step=0.1)
51         input_previous_rain = st.number_input("Previous Day Rainfall", min_value=0.0, max_value=180.0, step=0.1)
52         input_cloud = st.slider("Cloud Coverage", min_value=0.0, max_value=100.0, step=0.1)
53
54     input_data = pd.DataFrame({'temperature' : [input_temp],
55                               'cloud_covr' : [input_cloud],
56                               'pressure': [input_pressure],
57                               'wind_speed' : [input_wind_sp],
58                               'season_summer' : [input_season_summer],
59                               'previous_rainfall' : [input_previous_rain],
60                               'wind_dir_sin' : [input_wind_dir],
61                               'wind_dir_cos' : [input_wind_dir],
62                               'soil_moisture' : [input_soil_moisture],
63                               'evapotrans': [input_evapotrans]})

```


As seen in above code, the scalar and the model are loaded. The form container from Streamlit library is created to receive the input from the users. There will be three columns to fill in the inputs for the model. The input area is restricted from entering the abnormal values so as to maintain the reliable prediction result from the model. The pressure, wind direction and cloud coverage variables are received using the slide bar since these values can be range within a certain limit. All the new input values are combined into a data frame to pass into the model. However, the ANN used here contains the feature engineered variables and thus the same pre-processing must be applied to properly feed into the model. For that reason, the highly skewed previous rainfall variable is transformed using $\log + 1$ and the wind direction variable is converted into sine and cosine values. Since there is one seasonal variable, the drop-down

input box is used to collect the seasonal data which has Summer, Rainy, and Winter. The process of pre-processing the new input data is shown in below code.

```

65 # Data pre-processing for input data
66 # Apply log +1 transformaion to previous rainfall data
67 input_data['previous_rainfall'] = input_data['previous_rainfall'] + 1
68 input_data['previous_rainfall'] = np.log(input_data['previous_rainfall'])
69
70 # Compute sine and cosine of wind direction
71 input_data['wind_dir_sin'] = np.sin(np.radians(input_data['wind_dir_sin']))
72 input_data['wind_dir_cos'] = np.cos(np.radians(input_data['wind_dir_cos']))
73
74 # use the loaded scaler object to transform new data
75 scaled_data = pd.DataFrame(scaler.transform(input_data), columns=input_data.columns)
76
77 predict_data = st.form_submit_button(label = 'Predict')
78 if predict_data:
79     rainfall = ANN_model.predict(scaled_data)
80     rainfall = round(float(rainfall), 2)
81     # Show 0 if the estimate is less than zero
82     if rainfall < 0:
83         rainfall = 0
84     st.info(f"The estimated rainfall amount is :")
85     st.subheader(f":green[{rainfall}]")
86
87 # Display disclamier
88 st.write('Disclaimer: The rainfall prediction model presented here is still in the experimental
89 stage and is provided for informational purposes only. Please note that the developer of
90 this model is not liable for any direct, indirect, incidental, consequential, or
91 punitive damages arising from the use of the model or its results.
92 Version 1.0, Last updated: April-2023.')
```

Once all the input values are collected, they are then scaled using the Standard Scalar which has been loaded and the predict function is used to generate the estimated rainfall prediction result from the model. There is the possibility that the model could give the negative result which is discovered during the developing phase that the negative values will be generated as zero which mean no rainfall for the day. For the reason that the model is still in experimental phase, and it is intended to use for academic purposes, the disclaimer is also added at the bottom to warn the users not to over relying on the model results. The deployment was taken place in local machine primarily and then the python scrip is uploaded into Streamlit Community Cloud to be accessible for other users and test its performance. The deployed app is available to use via the link: <https://tgi-rainfall-prediction.streamlit.app/>. The simple web interface for prediction of rainfall estimates using the ANN models is show as below.



Rainfall Estimation model for Taunggyi

Fill in the input features to predict the rainfall.

Temperature <input type="text" value="0.00"/> - +	Wind Speed <input type="text" value="0.00"/> - +	Evapotranspiration <input type="text" value="0.00"/> - +
Season Summer ▾	Soil Moisture <input type="text" value="0.00"/> - +	Previous Day Rainfall <input type="text" value="0.00"/> - +
Pressure <input type="range" value="995.00"/> 995.00 1030.00	Wind Direction <input type="range" value="1.00"/> 1.00 360.00	Cloud Coverage <input type="range" value="0.00"/> 0.00 100.00

Disclaimer: The rainfall prediction model presented here is still in the experimental stage and is provided for informational purposes only. Please note that the developer of this model is not liable for any direct, indirect, incidental, consequential, or punitive damages arising from the use of the model or its results. Version 1.0, Last updated: April-2023. Developed by Mr.Sai Naing Lynn Oo (TP068393).

Figure 126: A simple web interface to access the model

5.8.1 Model Testing

The deployed model is tested with the know actual values to determine how the model is performing as below.

Index	date	temperature	cloud_covr	pressure	evapotrans	soil_moisture	wind_speed	wind_dir	rainfall
0	19850101T0000	17.24	37.1	1015.48	2.64	0.28	0.93	289.26	0.0
1	19850102T0000	16.79	20.18	1015.7	2.93	0.28	1.18	308.82	0.0
2	19850103T0000	15.96	60.13	1015.85	2.92	0.27	0.92	265.89	0.0
3	19850104T0000	15.33	37.93	1016.1	2.85	0.27	0.88	298.26	0.0
4	19850105T0000	15.49	34.67	1015.27	2.94	0.27	1.04	286.7	0.0

Temperature <input type="text" value="16.79"/> - +	Wind Speed <input type="text" value="1.18"/> - +	Evapotranspiration <input type="text" value="2.93"/> - +
Season Winter ▾	Soil Moisture <input type="text" value="0.28"/> - +	Previous Day Rainfall <input type="text" value="0.00"/> - +
Pressure <input type="range" value="1015.79"/> 995.00 1030.00	Wind Direction <input type="range" value="308.20"/> 1.00 360.00	Cloud Coverage <input type="range" value="20.20"/> 0.00 100.00

The estimated rainfall amount is :

0.24

Figure 127: Model Test case No. (1)

The Figure (124) demonstrates the model test case No. (1). The actual value of all the independent variable from the data is entered into the interface and the actual rainfall result is zero. The model estimated that the rainfall amount will be 0.24 based on the input values. It can predict well in this case.

index	date	temperature	cloud_covr	pressure	evapotrans	soil_moisture	wind_speed	wind_dir	rainfall
110	19850421T0000	23.57	89.79	1008.32	1.7	0.16	1.05	277.82	7.0
117	19850428T0000	22.93	76.17	1007.8	2.56	0.24	0.86	92.29	8.5
121	19850502T0000	19.55	80.42	1006.08	2.22	0.23	1.35	272.65	6.6
144	19850525T0000	19.57	90.42	1003.84	1.39	0.31	2.57	173.8	8.0
147	19850528T0000	20.96	79.16	1001.97	2.25	0.3	1.59	193.44	9.3
154	19850604T0000	19.28	100.0	1005.04	0.78	0.31	1.9	165.93	6.9
156	19850606T0000	18.82	100.0	1006.91	0.49	0.31	2.96	184.06	7.2
157	19850607T0000	18.58	100.0	1006.1	1.21	0.31	3.46	195.61	7.0
172	19850622T0000	20.43	77.71	1000.52	1.71	0.32	1.74	103.97	7.1
173	19850623T0000	19.42	100.0	1002.15	1.13	0.32	1.87	201.3	7.8

The screenshot shows a user interface for a model. It features several input fields with numerical values and a 'Predict' button. Below the inputs, a blue box displays the estimated rainfall amount as 11.29.

Variable	Value
Temperature	19.42
Wind Speed	1.87
Evapotranspiration	1.13
Season	Rainy
Soil Moisture	0.32
Previous Day Rainfall	7.10
Pressure	1002.20
Wind Direction	291.30
Cloud Coverage	100.00
Estimated Rainfall	11.29

Figure 128: Model Test Case No. (2)

In Figure (125), the result of test case No. (2) is presented. The actual values of rainfall amount on that day are 7.8 and the model predicted as 11.29, the error of 3.49 is produced. The RMSE value of this model is 5.73 and thus, on average the model can have an average error of 5.73. Therefore, the predictions could be considered acceptable.

The model is also tested by the right users from the area of Taunggyi city including the personnel from Taunggyi Metrological Office, the local agriculture business owner and a resident. Since the feedbacks are given in Myanmar language, they are translated and presented below. Their feedbacks are also noted for future works as well.

အားလုံး သိကျတဲ့အခါတိုင်း တစ်ခုခုတစ်ခုခုကို မှီ၊ စွမ်းစွမ်းကို သိဖို့က
 အရေးကြီးပါတယ်။ ဒါမှလည်း ဒီနေ့အင်အား အလုပ်အကိုင် အခါအခါကို ချိန်လျှောက်ပေါ့။
 လိုက်အောင်ထွက်ချင်တဲ့ ချိန် မှီ၊ စွမ်းစွမ်းကို ဖြိုဖော်ထားရင် အောင်မြင်မှုပေးပေ။
 နဲ့မှန်လည်း အမှန်တော်က နှစ်ဖြင့် မှီ၊ စွမ်းစွမ်းကို သိဖို့ မှီ၊ စေပသ အမှန်၊ ချက်တွေကို
 မှီ၊ ကန် ဖြစ်ဖြစ်ပါတယ်။ တစ်ခါတစ်ရံလည်း အမှန်၊ ချက်တွေက မှားတာပေါ့ပေ။
 ဒါ့အပြင် စက် ချက်က အမှန်၊ ချက်တွေကို နိုင်တာကို သေ အလုပ်ပေါ့ပေ။ လိုက်သိတဲ့
 အချက်အလက်တွေကို စိုက်ထည့်ပေးလို့လဲတော့၊ သူက မှီ၊ အလုပ်အကိုင် စွမ်းစွမ်း
 ဖြစ်ပေးပေး အောင်မြင်တာပေါ့။ အောင်မြင်ပေးပေး မှီ၊ စေပပါတယ်။ အမှန်၊ ချက်တွေလည်း
 အတိအကျ ဖြစ်ပေး မှီ၊ စေပပါတယ်။ ထပ်ပြီး ဒီ အင်အား မှီ၊ စေပတဲ့ အခါ နိုင်
 ပါပေါ့။ နေ့တိုင်း စေပရင်။

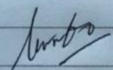

 မျိုးစွမ်းစွမ်း
 မှီ၊ စေပတဲ့ အင်အား နိုင်
 စေပရင် မှီ။

Figure 131: Feedback (3)

English Translation

"As everybody knows it is important to know whether today will rain or not so that we can arrange our daily task. If don't know there is rainfall in advance, it would be troublesome. I usually check the forecast of rainfall using my mobile apps. Sure, they predict wrongly sometimes since they are just predictions. I really like the website you provided for predicting the rainfall. It is great that I can get the predictions by typing in the data I know on the website. I understood that it is not possible to be the exact result, but the results are close. Wishing you to be able to develop more accurate and advanced system in the future."

(Signature)

Mr. Myo Min Oo
 Mobile Tower Engineer
 Taunggyi city.

5.9 Summary of result and discussion

In this chapter, the implemented models are compared and analyzed critically. The models are evaluated using the evaluation metrics mentioned in the methodology such as MAE, RMSE and R Square values. This chapter also extensively analyze and discuss the models which have been built with serval approaches and their result are also compared with some of the previous studies from the related work. The models are also verified whether they are overfit or underfit using the test and train evaluation metrics. Some of the base models are slightly overfit initially, however, after the models are tuned, all the models demonstrated having a good fit. Among them, the ANN model which includes both traditional weather data as well as hydrological data with feature engineering provided the best result and the model is deployed on Streamlit Cloud. The details process for deploying the best model and the model testing, including the feedbacks are also included in this chapter.

CHAPTER 6

CONCLUSION

6.0 Introduction

In this chapter, all aspects of the study are summarized in a conclusive manner in order to answer the capstone project questions and demonstrate a clear understanding of how the project met its objectives. There will be documentations of how the problem statement were addressed, the objectives were achieved, and the research questions were answered in this chapter. Additionally, this chapter summarizes the overall conclusion of the capstone project, followed by its contributions, limitations, and future work.

6.1 Addressing Project's Objectives and Research Questions

This section intends to address the provisions of the project objectives and the research questions formulated in Chapter 1 using the results and finding obtained from experimentations.

6.1.1 Objectives of the study

- *To identify appropriate machine learning techniques and evaluation metrics for the development and evaluation of a rainfall prediction model.*

This objective is achieved by conducting the literature review on the rainfall prediction using machine learning techniques. The journals and articles which are published recently are selected to identify what algorithms are suitable for predicting the estimation of rainfall. As noted from the literatures, the most appropriate machine learning algorithms for rainfall prediction are Random Forest, Support Vector Machine, and Artificial Neural Network. These techniques demonstrated promising result in most of the reviewed articles. Since the estimation of rainfall is the regression problem in machine learning, the evaluation metrics used for regressor model are found to be Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and the Coefficient of determination (R Square). These machine learning techniques and evaluation metrics are used in the development of rainfall prediction models using only with the traditional weather data and also incorporation with hydrological data. The models are then compared, and the best performing model is selected using the studied evaluation metrics.

- ***To compare the performance of prediction models trained solely on traditional weather data versus those trained on both traditional weather data and hydrological data in predicting rainfall amount.***

This objective is accomplished by building the models only with the traditional weather data and using both traditional weather data and hydrological data. The data sets are divided into two types, the one only with traditional weather variables such as wind speed and direction, pressure, and temperature while the other datasets contain additional hydrological data such as soil moisture and evapotranspiration. The proposed three different algorithms are implemented on both types of datasets and the model results are assessed. Based on the results of the study, it was found that rainfall prediction models that were trained and used both traditional weather data and hydrological data were two times more effective in terms of their model performance as compared to models that were used only with traditional weather data.

- ***To identify the specific feature(s) from the hydrological data that significantly influence the prediction of rainfall amount.***

This objective is fulfilled by assessing the variables from hydrological data using the feature importance scores from the Random Forest and Support Vector Regressor. It is discovered that both algorithms reveal the soil moisture and evapotranspiration contributes the greatest prediction power with the soil moisture being the highest with over 40% and then followed by the evapotranspiration with around 20% among other variables in the model. The inclusion of just these two variables significantly improved the model's performance. This fact is also supported by the R Square value of the model. After these variables are incorporated into the model, R Square values boosted from about 40% to 80%, explaining the larger variance is achieved in the model.

- ***To evaluate the effectiveness and efficiency of the feature engineering process in improving the performance of the prediction model.***

This objective is met by constructing the model with and without feature engineering in the model. The performance of the model with and without feature engineered variable are

then compared and evaluated. The new variables such as seasonal variable from the original date variable, wind direction in sine and cosine from the existing wind variables are created and these variables are included in the modelling. The result reveals that the effectiveness and efficiency seem also dependent upon the machine learning techniques used. For instance, in the Random Forest model and Support Vector model, the model shows not much improvement in terms of RMSE, MAE and R Square values after conducting the feature engineering whereas in the Artificial Neural Network model, it slightly improved in terms of the R Square value. This might be due to that more variables are entered into the model.

6.1.1 Research Questions of the study

- *What machine learning techniques and evaluation metrics could be used to build and assess the performance of a rainfall prediction model?*

The first step in performing this project is to study the most suitable machine learning techniques to be used for the prediction of rainfall estimation and to assess the performance of model after they have been built. For this reason, the domain knowledge, the methods currently using for rainfall prediction models are thoroughly and critically evaluated in Chapter (2) Literature review of this project report. The answer to this question would be that the machine learning techniques that could be used to build the rainfall prediction models are the Random Forest model, Support Vector Regressor, and Artificial Neural Network model. The evaluation to assess the performance of the models are Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and the Coefficient of determination (R Square) values.

- *Does a model trained on traditional weather data or one that is additionally trained on hydrological data perform better in predicting rainfall amount?*

This project studies both models that is only built with traditional weather data and the model with addition of hydrological data such as soil moisture and evapotranspiration using three different algorithms. It is revealed from the study that the model trained with both traditional weather variables and hydrological data are performing better than those trained only with traditional weather data in all algorithms. The detail of this analysis is carried out in section 5.6 of Chapter (5) Result and Discussion. In summary, the inclusion of hydrological

data in the model greatly contributes the significant improvement in the model's performance. Among them, the Artificial Neural Network model yielded the best accuracy and provided reliable prediction. Therefore, it could be concluded that the model that is additionally trained on hydrological data perform better in prediction the rainfall amount.

<i>Model</i>	<i>Data used in training model</i>	<i>RMSE</i>	<i>MAE</i>	<i>R²</i>
ANN Model	Traditional weather data	9.5847	4.2529	0.4652
ANN Model	Additional Hydrological data	5.7607	2.7804	0.8068
RF Model	Traditional weather data	9.5212	4.4244	0.4722
RF Model	Additional Hydrological data	5.8607	2.7516	0.8000
SVR Model	Traditional weather data	9.9274	4.1683	0.4262
SVR Model	Additional Hydrological data	6.9129	2.9929	0.7218

Table 27: The comparison of the model with and without additional Hydrological data

- *Is there any particular feature(s) from hydrological data that will play a more significant role in predicting the amount of rainfall?*

By using the feature importance scores of the predictors, this research question can be addressed by using the second-best performing model, referred to as Random Forest. Because the Random Forest model employs the major voting techniques by utilizing decision trees as part of the model, it is able to provide the influence of the climate and hydrological variables on rainfall by examining the feature importance scores.

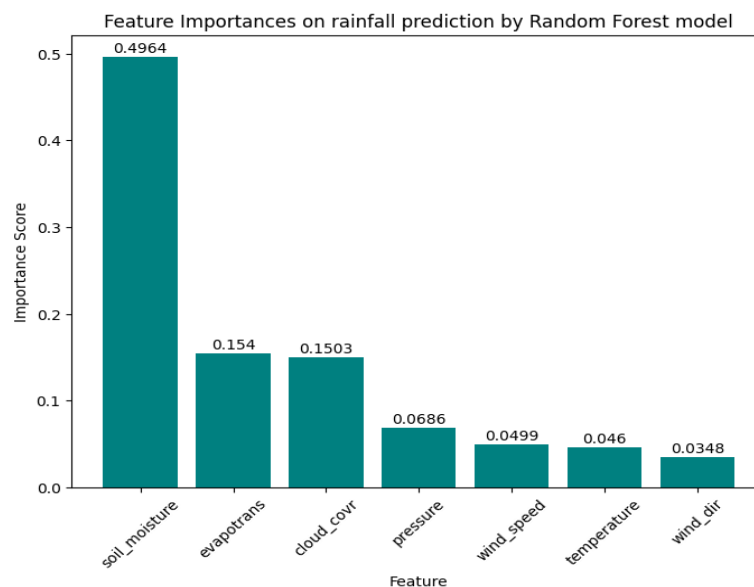


Figure 132: Feature importance on rainfall prediction by best performing Random Forest Model

As it can be seen from the Figure (111), the most influencing variable on the amount of rainfall is soil moisture with a feature importance score with 0.4964 and then followed by the evapotranspiration variable with 0.1540. Both variables are under the category of hydrological data. Thus, it could be deduced that the soil moisture feature will play a more significant role in predicting the amount of rainfall.

- *Is the feature engineering process going to provide the best results in terms of efficiency and performance for the rainfall prediction model?*

This study used correlation analysis as well as tree-based feature selection as a part of the feature engineering techniques to select the most informative and relevant features that can then be retained in the dataset to predict the target variable in a meaningful manner. Then, feature creation method is used to create new features to better capture the valuable information to improve the efficiency of the predictive model. As part of the evaluation of the effectiveness of feature engineering, two versions of the model, one with and one without feature engineering, were created and their performance was compared.

<i>Model</i>	<i>Feature Engineering</i>	<i>RMSE</i>	<i>MAE</i>	<i>R²</i>
ANN Model	No	5.7607	2.7804	0.8068
ANN Model	Yes	5.7337	2.7829	0.8337
RF Model	No	5.8607	2.7516	0.8000
RF Model	Yes	5.9923	2.7534	0.8184
SVR Model	No	6.9129	2.9929	0.7218
SVR Model	Yes	7.7034	3.1761	0.6998

Table 28: Comparison of the model with and without feature engineering

Table (28) represents the comparison of the model which includes both the traditional weather data and hydrological data with and without feature engineering. The better performing models are highlighted in bold. As noted from the table that, in most of the models such as Random Forest and Support Vector Regressor did not yield better performance after the model is built with feature engineered variables. It is only the Artificial Neural Network model that achieved slightly better performance after feature engineering. Hence, it could be concluded that the feature engineering in this project did not provide significant improvement in the model performance.

6.2 Discussion and conclusion

Throughout the history, the topic of rainfall prediction has been the most demanding theme since it is closely related to the nature and the lives of living beings on the earth. The meteorologists are attempting to predict the rainfall using a wide range of techniques. In modern world, the prediction of rainfall is currently operated by many meteorological and hydrological agencies all around the world using numerical weather prediction system and meteorological data mining approaches. Since the traditional prediction system are computationally expensive and hence, the scientists are trying to use the artificial intelligence such as deep neural network along with machine learning algorithms to accurately predict the rainfall. Due to the effect of ongoing Global warming due to increase of greenhouse gases caused by human activities, the atmosphere system has been shifting from its normal circulation. As a result, every part of the world is experiencing the adverse weather condition frequently such as droughts, floods, land sliding, hurricanes, extremes heat and cold, wildfires. Among them, unusual rainfall activities are also contributing to occur these types of events. Therefore, it is essential to have a rainfall prediction system which helps to mitigate the effect of these occasions.

The prediction of rainfall either classification or regression problem is exceptionally popular in the field of meteorology according to the literature review. Many researchers and scholars are attempting to solve this problem using several approaches. They used the dataset from different locations, employ many algorithms using available features to predict the rainfall. However, the accuracy is still the issue to be addressed. In many studies, the researchers used well known algorithms such as Random Forest, Support vector Machine, Deep Learning architecture including Artificial Neural Network, Logistic and Linear Regression. Most studies used the traditional weather data such as wind speed and direction, pressure, cloud coverage, temperature. They also achieved acceptable result in their model performance. However, the study of incorporation with hydrological data such as soil moisture and evapotranspiration are still lacking and hence this study experiment modelling with both data whether the model's performance is improved.

In this capstone project, three different algorithms such as Random Forest, Support Vector Regressor and Artificial Neural Network are proposed to conduct this study. To compare their performance and reliability, three different evaluation metrics used for regression problem namely MAE, RMSE, and R Square values are utilized. The models are built with different dataset containing only the traditional weather and additional hydrological data. The feature engineering techniques are also applied to better capture the information by

the model. Based on the critical analysis of the models, the ANN model which includes both types of data with feature engineering yielded the best in this project. The result of the Random Forest model is also comparable to the ANN model. However, since the selection of the best model is in accordance with evaluation metrics, the model with better metrics was chosen. The model is also deployed on Streamlit Community cloud to be accessible by end-users.

The study also examines the impact of including the hydrological data into the model built with traditional weather data. The findings from three different models proved that the inclusion of hydrological data demonstrated almost two times improvement in model performance. It is also discovered that the effect of hydrological data such as soil moisture and evapotranspiration have the highest influence among the variable. These two features have two opposite effects in prediction the rainfall amount. The soil moisture has the positive association while the evapotranspiration is negatively correlated to the amount of rainfall. Their feature importance scores are higher than the traditional weather variables. The feature engineering techniques are also applied to the variable, and it was observed that the feature engineering techniques does not provide significant improvement in terms of model performance.

The main challenges faced during the implementation stage is that the features in the dataset are highly correlated and the decision which variable to drop required domain knowledge. It is also applied in feature engineering process as well. The domain knowledge is essential for performing the feature extraction and transformation. However, this task is accomplished by researching which variable might be useful in predicting the rainfall. Another challenge is tuning the models. Although the models are tuned using Random Search method, the time required for tuning the models even including the help of GPU from Google Colab is tremendous. The Random Forest and Support Vector models required around 30 to 40 minutes for each model whereas the Artificial Neural Network model took from 2 to 2.5 hours for tuning the model with selected param sets. However, much new knowledges on techniques and approaches used in machine learning area are gained by conducting this capstone project as well.

The aim and objectives are met completely, and the research questions are also addressed by this capstone project. The detail discussions are provided in the Chapter (5) and thus the capstone project for rainfall prediction using machine learning approaches with additional hydrological data can be considered successfully achieved.

6.3 Contributions

There might have several factors required in predicting the rainfall with highest accuracy. This project contributes to the field of rainfall prediction with new insights on how the inclusion of hydrological data with traditional weather data could potentially improve the performance and accuracy of the rainfall predictive models. It also demonstrated that how the hydrological data have the higher feature importance compared to other common climate variables. In addition, the model achieved from the study is specific to the region of Taunggyi area and it could be helpful in prediction the future rainfall events for this area.

6.4 Limitations

The study included most of the recent literature related to the domain of interests and thoroughly reviewed the articles. Nonetheless, the important previous studies might have been overlooked. The results presented in their articles are produced by the respective authors and that the accurateness of the finding might affect the analysis of the current study. The dataset used is the area specific and thus the results and model obtained are applied just for the interested regions. The data used in this study is intended to use only for academic propose only with the permission from meteoblue® weather company and therefore, the data could not be exposed to other parties. All the data will be deleted once the project is completed.

6.5 Future work

Although this capstone project is completed successfully, there are still rooms for improvement for building the model which could provide better performance. The feature engineering techniques is known to improve the performance of the model in many machine learning projects, it did not assist to have significant improvement in the model in the current study. Therefore, in future projects, different feature engineering techniques could be applied to yield better result. Moreover, the inclusion of other alternative data such as radar images and climate indexes to improve the accuracy. The simple web interface provided in the deployment could also be improved its design and functionalities and the integration of input data from weather resources to get either the live or future predictions could be added as well.

6.6 Summary of conclusion

This chapter summarize all the insights and results gained from the findings and analysis during the whole capstone project. It also offered how the aim and objectives are met, the answers for the research questions are discussed in detail. In the discussion and conclusion section, the summary of how the rainfall prediction is essential and the approaches have been used to predict future rainfall event are explained. The achievement of the best model and the results and findings, the challenges faced during conducting this project. Finally, the contributions to the rainfall prediction area, limitations imposed by this project, the recommend future works are well delivered.

REFERENCES

- Aftab, S., Ahmad, M., Hameed, N., Salman, M., Ali, I., & Nawaz, Z. (2018). Rainfall Prediction using Data Mining Techniques: A Systematic Literature Review. *International Journal of Advanced Computer Science and Applications*, 9(5). <https://doi.org/10.14569/ijacsa.2018.090518>
- Agnihotri, G., & Panda, J. (2014). Comparison of rainfall from ordinary and automatic rain gauges in Karnataka. *MAUSAM*, 65(4). <https://doi.org/10.54302/mausam.v65i4.1205>
- Ahmed, R., & Direkoglu, M. S. (2018). *Rainfall Prediction Using Machine Learning Techniques*. www.semanticscholar.org.
- Allaway, R. (2022). *Why does it rain?* www.geographyalltheway.com. https://www.geographyalltheway.com/ks3_geography/weather_climate/weather.htm
- Aswin, S., Geetha, P., & Vinayakumar, R. (2018, April 1). *Deep Learning Models for the Prediction of Rainfall*. IEEE Xplore. <https://doi.org/10.1109/ICCSP.2018.8523829>
- Aung, T. (2019). Forecasting of rainfall in central dry zone in Myanmar using SARIMA model (Doctoral dissertation, MERAL Portal).
- Bagirov, A. M., Mahmood, A., & Barton, A. (2017). Prediction of monthly rainfall in Victoria, Australia: Clusterwise linear regression approach. *Atmospheric Research*, 188, 20–29. <https://doi.org/10.1016/j.atmosres.2017.01.003>
- Bauer, P., Thorpe, A., & Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, 525(7567), 47–55. <https://doi.org/10.1038/nature14956>
- Bosher, L., & Chmutina, K. (2017). *Disaster Risk Reduction for the Built Environment*. In *Google Books*. John Wiley & Sons.
- Chantry, M., Christensen, H., Dueben, P., & Palmer, T. (2021). Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft AI. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), 20200083. <https://doi.org/10.1098/rsta.2020.0083>
- Chu, K.-S., Oh, C.-H., Choi, J.-R., & Kim, B.-S. (2022). Estimation of Threshold Rainfall in Ungauged Areas Using Machine Learning. *Water*, 14(6), 859. <https://doi.org/10.3390/w14060859>
- Dash, Y., Mishra, S. K., & Panigrahi, B. K. (2018). Rainfall prediction for the Kerala state of India using artificial intelligence approaches. *Computers & Electrical Engineering*, 70, 66–73. <https://doi.org/10.1016/j.compeleceng.2018.06.004>

- DMH, M. (2022). *Department of Meteorology and Hydrology*. www.moezala.gov.mm.
<https://www.moezala.gov.mm>
- Dueben, P. D., & Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, *11*(10), 3999–4009. <https://doi.org/10.5194/gmd-11-3999-2018>
- Ganesh, Y. S., Ali, M. M., & M, G. S. (2021). Efficient Rainfall Prediction and Analysis using Machine Learning Techniques. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(6), 3467–3474.
<https://www.turcomat.org/index.php/turkbilmat/article/view/7135>
- Gomathy, C., Bala, A., Reddy, N., Aravapalli, P., Kumar, A., Lokesh, S., Chandrasekharendra, S., Mahavidyalaya, & Kanchipuram. (2021). A study on Rainfall Prediction Techniques. *International Journal of Scientific Research in Engineering and Management (IJSREM)*, *5*(10).
- Hair, J., Black, W. C., Babin, B. J., & Anderson, R. E. (2010). *Multivariate data analysis : a global perspective* (7th ed.). Pearson Education, Cop.
- Hartigan, J., MacNamara, S., Leslie, L. M., & Speer, M. (2020). Attribution and Prediction of Precipitation and Temperature Trends within the Sydney Catchment Using Machine Learning. *Climate*, *8*(10), 120. <https://doi.org/10.3390/cli8100120>
- Hudnurkar, S., & Rayavarapu, N. (2022). Binary classification of rainfall time-series using machine learning algorithms. *International Journal of Electrical and Computer Engineering (IJECE)*, *12*(2), 1945. <https://doi.org/10.11591/ijece.v12i2.pp1945-1954>
- International Trade Administration. (2021). Burma - Agriculture. U.S. Department of Commerce. <https://www.trade.gov/country-commercial-guides/burma-agriculture>
- Iryani, S. Y., Al Amin, M. B., & Muhtarom, A. (2021). Simulation of Rainfall Data by The GPM Satellite (Case Study at Sriwijaya University, Indralaya). *IOP Conference Series: Earth and Environmental Science*, *832*(1), 012051.
<https://doi.org/10.1088/1755-1315/832/1/012051>
- Jinashree, P., Pooja, D. R., Meghana, M. V., Aishwarya, G. P., & Siddharth, B. K. (2021). Rainfall Prediction Using LASSO Regression. *International Journal of Emerging Technologies and Innovative Research*, *8*(7). 205-215.
<https://www.jetir.org/view?paper=JETIR2107156>
- Kumar Misra, R., Kumar Panda, P., Behera, D., Kumar Sahu, A., & Sahoo, S. (2020). Rainfall Prediction using Machine Learning Approach: A Case Study for the State of Odisha. *Indian Journal of Natural Sciences*, *10*(60).

- Kundu, S., Khare, D., & Mondal, A. (2017). Future changes in rainfall, temperature and reference evapotranspiration in the central India by least square support vector machine. *Geoscience Frontiers*, 8(3), 583–596.
<https://doi.org/10.1016/j.gsf.2016.06.002>
- Mar, K. W., & Naing, T. T. (2008). Optimum neural network architecture for precipitation prediction of Myanmar. *International Journal of Environmental and Ecological Engineering*, 2(12), 154-158.
- Meteoblue. (2022). *Simulation data*. Meteoblue. <https://content.meteoblue.com/en/research-education/specifications/data-sources/weather-simulation-data>
- Montesinos López, O. A., Montesinos López, A., & Crossa, J. (2022). Overfitting, Model Tuning, and Evaluation of Prediction Performance. *Multivariate Statistical Machine Learning Methods for Genomic Prediction*, 109–139. https://doi.org/10.1007/978-3-030-89010-0_4
- NordNordWest. (2017, October 16). *Deutsch: Positionskarte von Myanmar*. Wikimedia Commons.
https://commons.wikimedia.org/wiki/File:Myanmar_adm_location_map.svg
- Palacios, H. J. G., Toledo, R. A. J., Pantoja, G. A. H., & Navarro, Á. A. M. (2017). A comparative between CRISP-DM and SEMMA through the construction of a MODIS repository for studies of land use and cover change. *Advances in Science, Technology and Engineering Systems Journal*, 2(3), 598–604. <https://doi.org/10.25046/aj020376>
- Pham, B. T., Le, L. M., Le, T.-T., Bui, K.-T. T., Le, V. M., Ly, H.-B., & Prakash, I. (2020). Development of advanced artificial intelligence models for daily rainfall prediction. *Atmospheric Research*, 237, 104845. <https://doi.org/10.1016/j.atmosres.2020.104845>
- Porto de Carvalho, J. R., Assad, E. D., de Oliveira, A. F., & Silveira Pinto, H. (2014). Annual maximum daily rainfall trends in the Midwest, southeast and southern Brazil in the last 71 years. *Weather and Climate Extremes*, 5-6, 7–15.
<https://doi.org/10.1016/j.wace.2014.10.001>
- Prudden, R., Adams, S., Kangin, D., Robinson, N., Ravuri, S., Mohamed, S., & Arribas, A. (2020). A review of radar-based nowcasting of precipitation and applicable machine learning techniques. *ArXiv:2005.04988 [Physics, Stat]*.
<https://arxiv.org/abs/2005.04988>
- Pu, Z., & Kalnay, E. (2018). Numerical Weather Prediction Basics: Models, Numerical Methods, and Data Assimilation. *Handbook of Hydrometeorological Ensemble Forecasting*, 1–31. https://doi.org/10.1007/978-3-642-40457-3_11-1

- Pudyastuti, P. S. (2018). *Hydrology Engineering IR HERMONO S. BUDINETRO, M. ENG.* slideplayer.com. <https://slideplayer.com/slide/13020545/>
- Qiu, M., Zhao, P., Zhang, K., Huang, J., Shi, X., Wang, X., & Chu, W. (2017). A Short-Term Rainfall Prediction Model Using Multi-task Convolutional Neural Networks. *2017 IEEE International Conference on Data Mining (ICDM)*. <https://doi.org/10.1109/icdm.2017.49>
- Raitzer, D., Wong, L. C. Y., & Samson, J. N. (2015). Myanmar's Agriculture Sector: Unlocking the Potential for Inclusive Growth. ADB Economics Working Paper Series, 470 (Dec, 2015). <https://doi.org/10.2139/ssrn.2709353>
- Raval, M., Sivashanmugam, P., Pham, V., Gohel, H., Kaushik, A., & Wan, Y. (2021). Automated predictive analytics tool for rainfall forecasting. *Scientific Reports, 11*(1), 17704. <https://doi.org/10.1038/s41598-021-95735-8>
- Sarasa-Cabezuelo, A. (2022). Prediction of Rainfall in Australia Using Machine Learning. *Information, 13*(4), 163. <https://doi.org/10.3390/info13040163>
- Selase, A. E., Agyimpomaa, D. E. E., Selasi, D. D., & Hakii, D. M. N. (2015). Precipitation and Rainfall Types with Their Characteristic Features. *Journal of Natural Sciences Research, 5*(20), 89. <https://www.iiste.org/Journals/index.php/JNSR/article/view/26509>
- Shekana, S., Mulugeta, A., & Prasad, D. (2020). Weather Variability Forecasting Model through Data Mining Techniques. *International Journal of Advanced Computer Science and Applications, 11*(9). <https://doi.org/10.14569/ijacsa.2020.0110905>
- Shi, L., Westerhuis, J. A., Rosén, J., Landberg, R., & Brunius, C. (2018). Variable selection and validation in multivariate modelling. *Bioinformatics, 35*(6), 972–980. <https://doi.org/10.1093/bioinformatics/bty710>
- Sureh, F. S., SattariM. T., & İrvem, A. (2019). Estimation of monthly precipitation based on machine learning methods by using meteorological variables. *Mustafa Kemal Üniversitesi Tarım Bilimleri Dergisi, 24*, 149–154. <https://dergipark.org.tr/en/pub/mkutbd/issue/51091/651866>
- Tharun, V. P., Prakash, R., & Devi, S. R. (2018). Prediction of Rainfall Using Data Mining Techniques. *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. <https://doi.org/10.1109/icicct.2018.8473177>
- Thwe, P., Win, E. K., & Wai, H. P. M. (2019). A Markov Chain Approach on Daily Rainfall Occurrence. *International Journal of Trend in Scientific Research and Development (IJTSRD), 3*(6).

- Vitart, F., Ardilouze, C., Bonet, A., Brookshaw, A., Chen, M., Codorean, C., Déqué, M., Ferranti, L., Fucile, E., Fuentes, M., Hendon, H., Hodgson, J., Kang, H.-S. ., Kumar, A., Lin, H., Liu, G., Liu, X., Malguzzi, P., Mallas, I., & Manoussakis, M. (2017). The Subseasonal to Seasonal (S2S) Prediction Project Database. *Bulletin of the American Meteorological Society*, 98(1), 163–173. <https://doi.org/10.1175/bams-d-16-0017.1>
- Wang, Y., Yang, J., Chen, Y., De Maeyer, P., Li, Z., & Duan, W. (2018). Detecting the Causal Effect of Soil Moisture on Precipitation Using Convergent Cross Mapping. *Scientific Reports*, 8(1). <https://doi.org/10.1038/s41598-018-30669-2>

APPENDICES

Appendix 1: Research Plan

In accordance with Figure (13), this appendix provides an overview of how the proposed research plan will be carried out, the tasks that need to be completed, including the start and end date, including the duration (in days) are all included in the plan. A research Gantt chart clearly outlines the steps involved in the entire capstone project, beginning with idea generation, followed by literature review, proposed methodology and the submission of the capstone project. Three milestones are included in the project plan, which include approval of the project proposal, submission of the first phase of the project CP1, and submission of the second phase CP2.

Research Plan

Task / Month	Start Date	End Date	Duration (Days)	Aug	Sep - 2022					Oct					Nov				Dec				Jan - 2023					Feb				Mar				Apr	
				29	5	12	19	26	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30	6	13	20	27	6	13	20	27	3	10	
Project Idea Formulation																																					
Exploratory for Project Ideas	29/08/2022	4/9/2022	14																																		
Problem Statement	12/9/2022	18/09/2022	7																																		
Literatures Search	19/09/2022	2/10/2022	14																																		
Writing Project Proposal	26/09/2022	16/10/2022	21																																		
Analysis																																					
Literture Review	17/10/2022	30/10/2022	14																																		
Data Collection	24/10/2022	6/11/2022	14																																		
Methodology	31/10/2022	27/11/2022	28																																		
CP1 Report Writing	24/10/2022	8/12/2022	46																																		
Implementation and Evaluation																																					
Data Pre-Processing and Data Exploration																																					
Data Exploration	12/12/2022	18/12/2022	7																																		
Handling Missing Values	12/12/2022	18/12/2022	7																																		
Handling Anomalies	19/12/2022	25/12/2022	7																																		
Data Transformation	19/12/2022	25/12/2022	7																																		
Feature Selection	26/12/2022	1/1/2023	7																																		
Normalization	26/12/2022	1/1/2023	7																																		
Data Partitioning	26/12/2022	1/1/2023	7																																		
Modelling																																					
Random Forest	2/1/2023	15/1/2023	14																																		
Support Vector Machine	16/1/2023	29/1/2023	14																																		
Artificial Neural Network	30/1/2023	12/2/2023	14																																		
Model Evaluation																																					
MAE, RMSE, R Square	13/2/2023	19/2/2023	7																																		
Comparison and Selection	20/2/2023	28/2/2023	7																																		
Documentation																																					
CP2 Report Writing	20/2/2023	5/4/2023	45																																		
Final Report Submission	3/4/2023	7/4/2023	5																																		
Presentaion	3/4/2023	9/4/2023	7																																		




- Mile Stone (1)**  Approval of project proposal (14th Oct, 2022)
- Mile Stone (2)**  Submission of Capstone Project 1 (9th Dec, 2022)
- Mile Stone (3)**  Submission of Capstone Project 2 (07th Apr, 2023)

Figure 133: Research Plan