# Time-series and ML models in Rainfall Prediction

End-Semester Report
BITS F421T

By
Soumadip De
ID No. 2017B1A21757H

Under the supervision of
Dr. Anmala Jagadeesh
------------------------------------------

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**HYDERABAD CAMPUS**
**(October, 2022)**

## Acknowledgement

I would like to express my gratitude to Dr Anmala Jagadeesh, who gave me the opportunity to work on this thesis under his supervision. Due to his guidance, I have learnt a lot over the duration of my work on this interesting topic.
I would also like to thank the Academic and Undergraduate Studies Division (AUGSD), BITS Hyderabad, for letting me take up this project on short notice before the beginning of my final semester on campus.
I am immensely thankful for this experience which has provided me with both knowledge and experience in the field of data science and its application on an important topic that is relevant in today's times.

-----------------------------------------------------------------------------------------------

## Certificate from the Supervisor

This is to certify that the thesis entitled
"___Time Series and ML Models in Rainfall Prediction____________________"
submitted
by _____Soumadip De_____________ ID No _2017B1A21757H__ in partial fulfilment of the
requirement of BITS F421T/422T Thesis embodies the original work done by him/her under my
supervision.

[illegible]

Signature of the Supervisor

**Name & Designation:** Dr. Jagadeesh Anmala, Associate Professor, Department of Civil Engineering, BITS Pilani, Hyderabad Campus
Date: 22.12.2022

-----------------------------------------------------------------------------------------------

## Contents

## Thesis Abstract

----------------------------------------------------------------------------------------------------
Thesis
Title: Time-series and ML models in Rainfall Prediction
Supervisor: Dr. Anmala Jagadeesh
Semester: First Semester Academic Year: 2022-23
Name of Student: Soumadip De ID No: 2017B1A21757H
Abstract
----------------------------------------------------------------------------------------------------

Predicting and modelling climate variables like rainfall requires sophisticated computing due to their highly complicated and non-linear nature. In recent years, there has been an increasing interest in using machine learning for weather forecasting. This is primarily to find an accurate but cheap and computationally light alternative to traditional statistical forecasting and NWP(numerical weather prediction) models that are often used by weather departments. In this thesis, we attempt to model rainfall in the Hyderabad region of Telangana using time series analysis and machine learning techniques. We have used monthly rainfall data from IMD (Indian Meteorological Department) from 1901-2002. Four correlated weather variables namely vapour pressure, wet day frequency, diurnal temperature range and cloud cover for the same location and time period have been used as exogenous variables for our models to help increase the prediction accuracy. The models were trained using the first 97 years (1901 to 1997) of monthly rainfall data, and the remaining 5 years (1998 to 2002) have been used as testing data.

----------------------------------------------------------------------------------------------------

## Introduction

Rainfall forecasting is essential for metro cities, where it is necessary not just for flood forecasting but also for preventing damage to infrastructure. Unprecedented rainfall can also cause disruptions in the transport network, tourism, and event management. In recent studies, the effect of weather conditions on pollutant concentrations has also been highlighted (Czarnecka, Nidzgorska-Lencewicz, et al., 2011). The complexity and inaccuracy of traditional statistical forecasting models of rainfall predictions have increased interest in exploring time series forecasting and machine learning models to predict rainfall.

ARIMA(autoregressive integrated moving average) is the most popular model used for time series analysis. A slightly modified version of ARIMA called Seasonal-ARIMA or SARIMA is used when dealing with seasonal time series data like the weather. It has been widely used for studying rainfall patterns and making predictions (Rahman, M.A., Yunsheng et al., 2017, A. Geetha and G.M. Nasira, 2017). Several studies have also used it alongside more advanced models for comparison and found simple SARIMA models to give more accurate results (Shardoor and Rao, 2019).

As for more advanced models, Artificial Neural Networks(ANNs) have recently been increasingly used for rainfall forecasting (Animas, Oyedele, Bilal et al., 2021). In some cases, different machine learning models have also been used as an ensemble, which has sometimes been shown to provide more accurate results than using each model individually (Jitendra Shreemali, Praveen Galav, et al., 2020). LSTM (Long short-term memory) models have been one of the most popularly used RNN models for time series forecasting, and in most cases show a high reduction in error when compared to traditional time series models like ARIMA (Sima Siami-Namini, Neda Tavakoli, et al., 2021). ML models have also been shown to work better than deep learning models when the dataset is smaller, but as the dataset increases in size, deep learning models outperform ML models (Sarker I.H, 2021).

In this project, we aim to build models for predicting Rainfall in Hyderabad, Telangana. We investigated the correlation of rainfall with several climatic conditions and tried to use these correlated variables to enhance the accuracy of our models further. Using data from 1901-2002, we calculated the correlation of some commonly used monthly weather variables with precipitation. These variables were: Avg. Temp, Cloud Cover, Diurnal Temp. Range, Ground Frost, Max. Temp., Min. Temp, Potential Evapotranspiration, Precipitation, Crop Evapotranspiration, Vapour Pressure, Wet day Freq.

Among these *Cloud Cover, Diurnal Temp. Range, Vapour Pressure,* and *Wet day Freq.* showed a significant correlation with monthly precipitation. (Table 1)

**Table 1: Variables showing significant correlation with Precipitation**

| **Variable** | **Correlation with Precipitation** |
|---|---|
| Wet day Frequency | 0.93 |
| Cloud Cover | 0.81 |
| Diurnal Temperature Range | -0.74 |
| Vapour Pressure | 0.73 |

The reasons why these variables have a high correlation with precipitation can be speculated. Wet Day frequency refers to the number of days in a month that witnessed rainfall. The higher the number of wet days, the greater is the likelihood of the month having a higher total precipitation. Cloud cover similarly implies more availability of water in the form of clouds that can eventually condense to cause rainfall. Diurnal temperature range, which refers to the difference between the highest and lowest temperature in a day, has a negative correlation with precipitation, which can be due to the fact that rain cools down a place and prevents the heating up of atmosphere due to the sun and it leads to a relatively constant temperature throughout the day.

We shall use time series data: a kind of data where the data points are indexed in time, e.g. monthly rainfall, yearly revenue, daily stock price, etc. The data is called stationary data where mean and variance do not vary across time significantly. The correlation of a time series with a lagged version of itself is called its autocorrelation.

We shall compare the results obtained using a primary time series model, SARIMA, with more advanced deep learning or machine learning models. Data for these weather variables is obtained from Indian Meteorological Department(IMD), for a 100-year time period (1901-2002).

# Literature Review

SARIMA is one of the simplest and most commonly used models for time series prediction. Certain studies showed that it outperforms more advanced ML or deep learning models, depending on the type and size of the data. Shardoor and Rao (2019) used a SARIMA model (parameters (0, 1, 1)x(1, 1, 1, 12)) for Rainfall forecasting in the Kerala region. The model was validated wrt ANN, SVM, linear regression, and ARIMA and provided maximum accuracy.

As for using neural networks for weather-related data, Mislana and Haviluddin (2015) employed a back propagation neural network (BPNN) with three different epochs and two hidden layer combinations. Mean squared error(MSE) was used to determine model performance. The experiment was done over the region of Tenggarong, East Kalimantan - in Indonesia. The best result gave an MSE value of $9.63 \times 10^4$ and was obtained with the architecture <2-50-20-1, epoch 1000>.

Namini, Tavakoli, et al. (2018) compared the performance of ARIMA and LSTM to ascertain if the newly developed deep learning approaches are more powerful than the traditional time series forecasting methods. The two models were used on financial data, and they found LSTM to provide an 85% better prediction than ARIMA. The paper also found that changing the number of epochs doesn't improve performance.

Liyew and Melese (2021) compared three different machine learning algorithms, namely Random Forest, Extreme Gradient Boosting, and Multivariate linear regression, to predict daily rainfall intensity over Bahir Dar City, Ethiopia. Extreme Gradient Boosting (XGBoost) performed the best among these three. MAE (mean absolute error) and RMSE(root mean squared error) were used to determine and compare the model performances.

Shreemali and Galav (2020) used an ensemble of 6 different models, namely, (i) Linear; (ii) Neural Net; (iii) Regression; (iv) Generalized Linear; (v) CHAID (Chi-square automatic interaction detection); and (vi) Random Forest, which seemed to give more accurate results than using individual models. However, the duration of the data used was less than a decade. The authors mention that using a more extended range of data and application to multiple geographic locations would be necessary to check its reliability.

Liu et al. (2019) studied the performances of several deep-learning models for hourly rainfall forecasting and compared them to an ML algorithm (XGBoost) as the baseline. It was found that LSTM models with a more significant number of hidden layers seem to perform poorly while predicting weather-related data. Bidirectional LSTM and a Stacked LSTM containing 2-hidden layers showed better metrics than other deep learning models. However, all the tested deep learning models showed one general disadvantage: the inability to generalize correctly. All LSTM-based models are prone to overfit the data and are incapable of providing similarly accurate predictions on test and validation sets.

Sarker I.H. (2021) discussed how deep learning could model at a very high level from enormous amounts of data, which makes it an apt solution to many real-world problems. However, it is also seen that for smaller datasets, other machine learning algorithms usually outperform deep learning.

# Methodology

## SARIMA

Seasonal ARIMA is composed of 2 main components: autoregression (AR) and moving average (MA). AR (autoregressive) models regress the values of a given variable against past values of itself. In equation (i), $y_t$ is the variable value at the current timestep, and $y_{t-1}$ is the value of the same variable, one timestep ago. $\varepsilon_t$ is the shock term (= residuals of the regression line with the observed values), and $a$ is the slope of the regression. This is called an AR(1) model, since only a lag of one timestep is being regressed against.

$$y_t = a_1 y_{t-1} + \epsilon_t \qquad \text{- (i)}$$

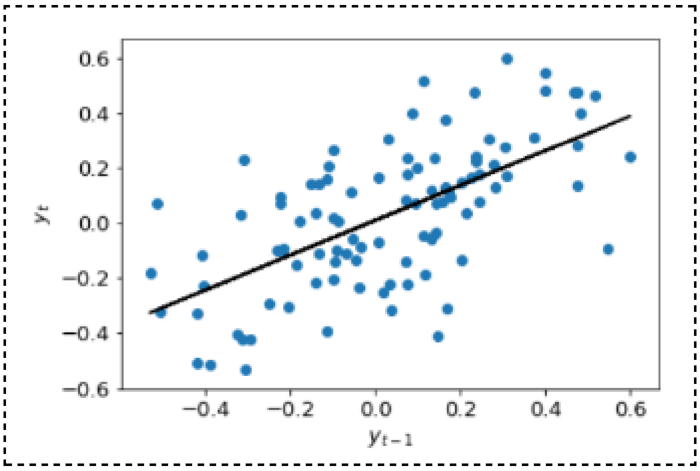


Fig. 1: an AR model of order 1, which means only values at lag 1 are regressed

Similarly, we can have models with higher orders which contain added "slope * lagged $y$" terms - one for each lag. (eqn. ii and iii)

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \epsilon_t \qquad \text{- (ii)}$$

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} \dots + a_p y_{t-p} + \epsilon_t \qquad \text{- (iii)}$$

In MA (moving average) models, the values of the variable of interest are estimated by using the previous "shock"(or error) values of the same time series. In eqn. (iv), $y_t$ is the variable of interest, $\varepsilon_{t-1}$ is the error term one timestep ago, and $m$ is the slope. $\varepsilon_t$ is the error term of the current timestep.

$$y_t = m_1 \epsilon_{t-1} + \epsilon_t \qquad \text{- (iv)}$$

$$y_t = m_1 \epsilon_{t-1} + m_2 \epsilon_{t-2} \dots + m_q \epsilon_{t-q} + \epsilon_t \qquad \text{- (v)}$$

ARMA (autoregressive moving average) models combine AR and MA models, containing both lagged variable and lagged error terms in its equation. (Eqn. vi)

$$y_t = a_1 y_{t-1} + m_1 \epsilon_{t-1} + \epsilon_t \qquad \text{- (vi)}$$

ARIMA(Autoregressive *Integrated* Moving Average) models are used with non-stationary data, e.g. the yearly GDP value of a country that shows an upward *trend*. To make the data stationary, we usually need to take a difference in the time series and then implement our ARMA model on that differenced time series. But since we want a forecast of the original data, not the differenced data, we have to *integrate* the values back into the original dataset. For cumulative data, this is usually done by taking the cumulative sum of the differences.

Seasonal ARIMA is a modified version of ARIMA that is used to model seasonal data, where we take another difference to account for the seasonality of data as well. In python's statsmodels library, we have the SARIMAX function, which can be used for this purpose.

The SARIMAX function takes model orders in the format SARIMAX (p,d,q) (P,D,Q)s where:
p: non-seasonal autoregressive order
d: non-seasonal differencing order
q: non-seasonal moving average order
P: seasonal autoregressive order per season
D: seasonal differencing order
Q: seasonal moving average order
s: no of time steps

## LSTM

An artificial neural network is a fundamental concept behind deep learning algorithms. The working of the human brain inspires them. A neural network contains nodes or "neurons", each of which is somewhat equivalent to a regression equation containing weighted parameters. The neurons are arranged in layers: 1 input layer, several hidden layers, and 1 output layer. (Fig. 2)

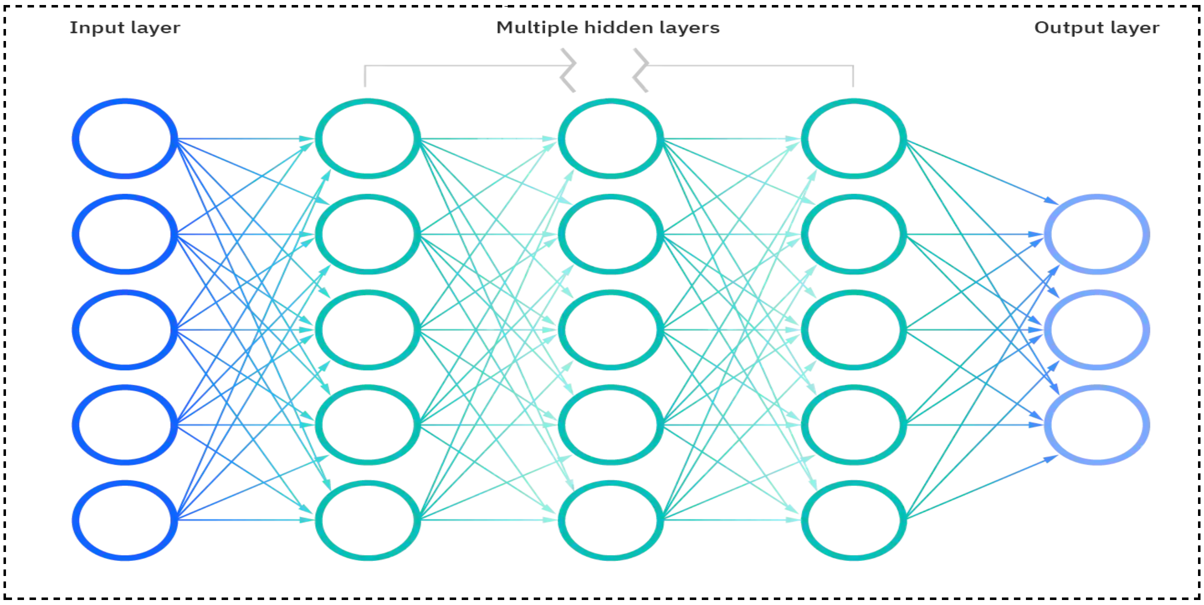


Fig. 2: A Deep Neural Network [Source: IBM Cloud Learn Hub]

A Recurrent Neural Network (RNN) is a special kind of neural network suitable for modeling sequential data. In an RNN, the data is fed into the model and goes through the layers of neurons in the forward direction — from the input layer, through the hidden layers, to the output layer. Then a method called “backpropagation” is employed.The weights of the parameters in the model are adjusted to decrease the error value, and then the data is fed again into the model. In every iteration, the weights are adjusted to give smaller and smaller errors in the next iteration. This is how an RNN trains itself.

LSTM (long short-term memory network) is a type of RNN unit cell that can model data with considerable time lags and remember inputs over a long period of time. This is because it has a more significant memory than normal RNNs, making it suitable for modeling time series data.

## CART (Decision Trees)

CART (Classification And Regression Tree) is a simple Decision Tree model. It uses a flowchart-like structure to compute all possible outcomes (y-values) given specific inputs (x-values). The tree contains points called “nodes”, which are of 2 types: decision nodes and end nodes. The former splits the data into two depending on a condition and the latter gives the end result of a specific series of splits.

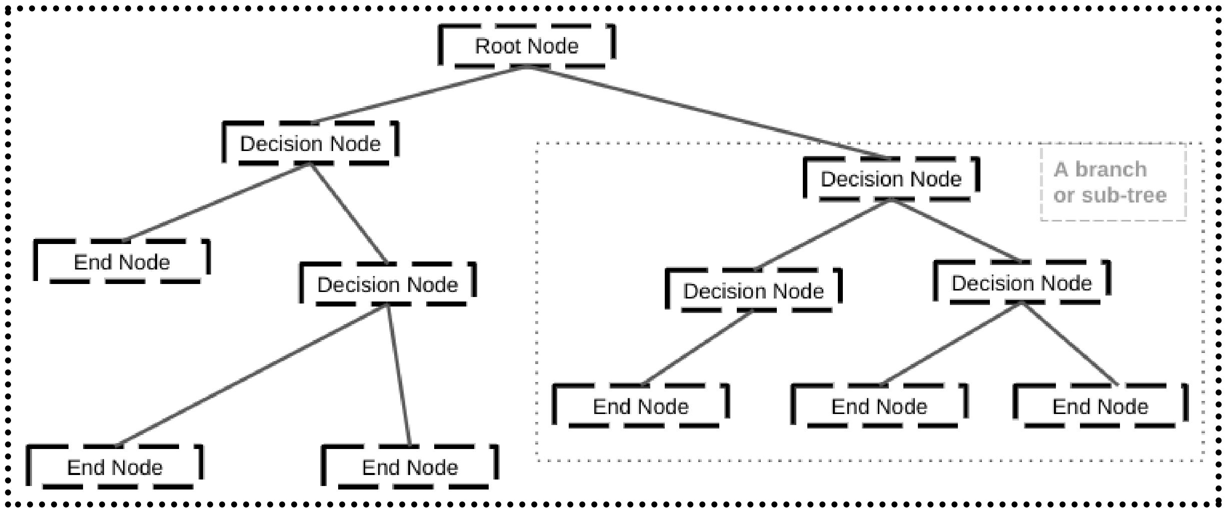


Fig. 3: A Decision tree

For regression, the model trains on a set of features in the structure of a tree and returns a continuous output of the target variable. The splitting takes place till a stopping criterion is met. To prevent clustering, the tree’s complexity is to be kept low, and for that. the effect of deletion of every end node is evaluated during the construction of the model, a process called “pruning”.

## Ensemble Models

In machine learning, ensemble modeling is an approach to combining multiple models for a single prediction process. This is done for three reasons: reducing the variance, increasing the accuracy, and eliminating noise/feature bias by aggregating the predictions from the combined models. Another prime advantage of ensemble models is decreasing the likelihood of overfitting, which can often occur with individual ML models or deep learning models when the dataset is not very large.

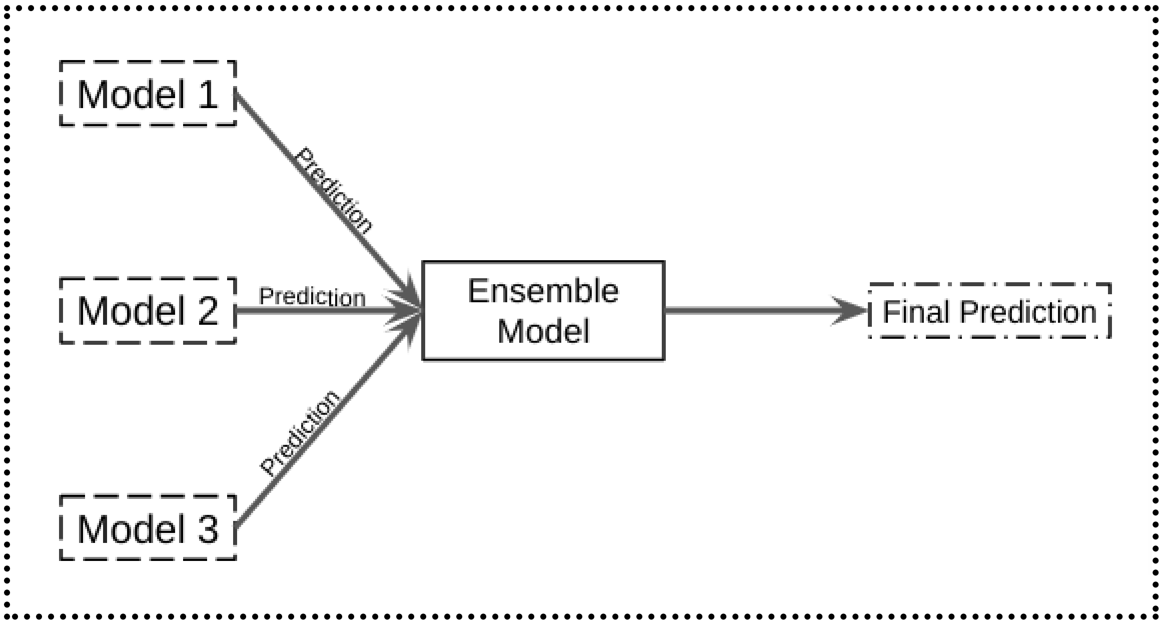


Fig. 4: An Ensemble model

We used five ensemble models. They can be divided into two broad ensembling principles:

## Boosting Regressors

Boosting is a technique that involves ensembling base models in series. At first, one model is built from the training data, followed by another model that tries to reduce the errors present in the first model, then another model that corrects the errors in the second model, and so on. This is done till the complete dataset is predicted correctly, or till we have used the total number of models, we planned to use.
We used the following boosting models-

AdaBoost(Adaptive Boosting): this uses a decision tree with one split (also known as a “stump”), as the base learner. Training set data is fed into a sequence of stumps, and depending on the performance of each stump, the next stump is constructed to reduce the error of incorrectly predicted data. Each stump also has a weight attached, so the lesser the error produced by a stump, the greater its significance in the ensembled model.

XGBoost (Extreme Gradient Boosting): this uses decision trees additively intending to reduce the value of the loss function in each successive tree. A regularization term is present to reduce the possibility of over-fitting.

## Bagging Regressors

Bagging is a technique that involves ensembling base models in parallel upon random subsets of the training dataset. The training set for each base model is independent of each other.

Random Forest: this model creates a “forest” of uncorrelated decision trees by generating a random subset of features, which leads creating uncorrelated forest of decision trees.

Extra trees: this is like a Random forest, but the splits of the decision trees are randomized in Extra Trees, whereas they are deterministic in a Random forest. Also, sampling is not bootstrapped, unlike Random forest; hence there is replacement during sampling.

## SVR

Support Vector Regressor tries to build a hyperplane as a "best-fit line" containing the maximum number of points. Instead of reducing the error between predicted and observed values, the SVR tries to find the best fit line between two decision boundaries.

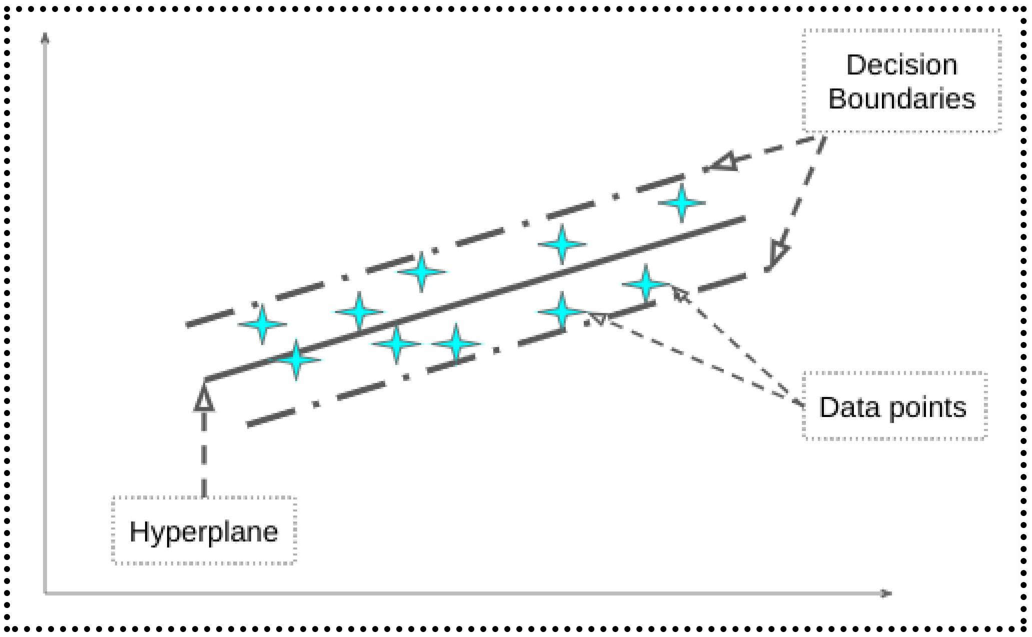


Fig. 5: A Support Vector Regressor

## Data preprocessing and Modelling

### Metrics and Tests

RMSE (root mean squared error): mean squared error (MSE) is the average of squared differences between observed and predicted values. RMSE is the square root of MSE.

$R^2$ value: the amount of variance in the output variable that is explainable by the input variable. It is a calculation of {total variance explained by model} / total variance. So an $R^2$ value of 1 means that the two variables are perfectly correlated. It is also known as the coefficient of correlation.

The Augmented Dickey-Fuller test is a statistical significance test that tells us whether a given dataset is stationary. It works with the null hypothesis that the dataset is non-stationary. The test returns a p-value. If this p-value is <= 0.05, we reject the null hypothesis and can say that the dataset is, in fact, stationary.

For selecting the model orders for SARIMA (i.e., finding P,Q,D,p,q,d), we used pmdarima() function in python, which shows the best model orders based on AIC or BIC.

Akaike Information Criterion(AIC): is a probabilistic method for scoring and selecting a model. It is calculated as

$$AIC = -2/N * LL + 2 * k/N \quad \text{- (vii)}$$

where N = no of samples in the training set, LL = log-likelihood of the model on the training set, and k = no of parameters in the model

Bayesian Information Criterion(BIC): is another method for selecting a model. It is calculated as

$$BIC = -2 * LL + \log(N) * k \quad \text{- (viii)}$$

(The Elements of Statistical Learning, Trevor Hastie)

For both these methods, the score has to be minimised, i.e. the model with the lowest AIC or BIC is selected.

## SARIMA

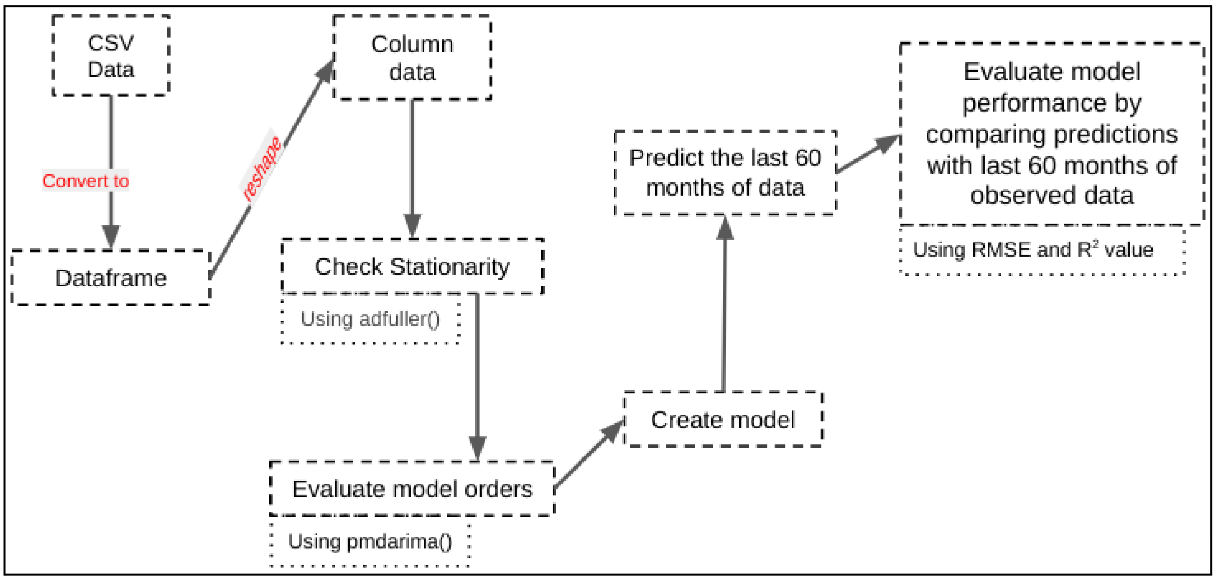
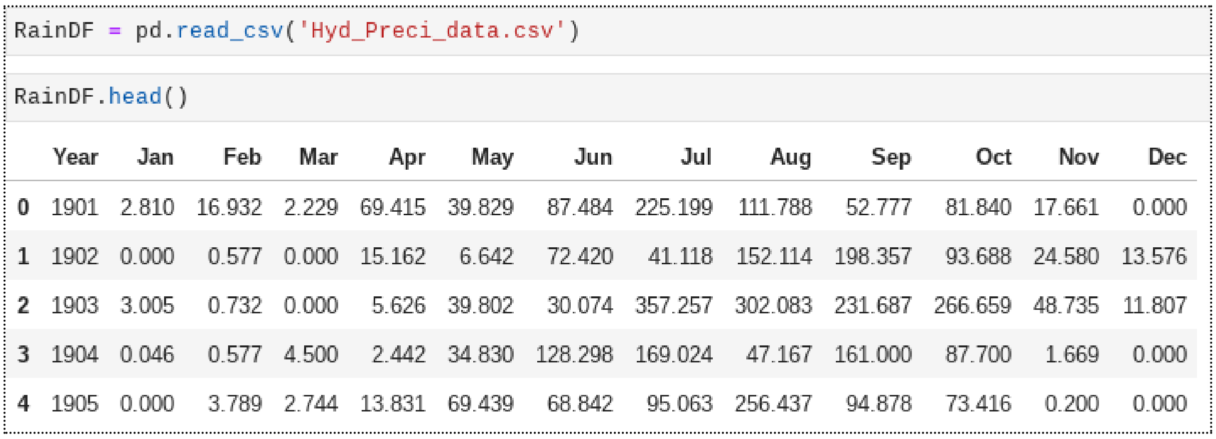


Fig. 6: Steps used for processing the data and creating the model using SARIMA

CSV data is first converted to a pandas dataframe, a tabular data format used in python.

Initially, we have the precipitation data in “wide data” format as shown in Table 2.

**Table 2: Dataframe in wide data format**



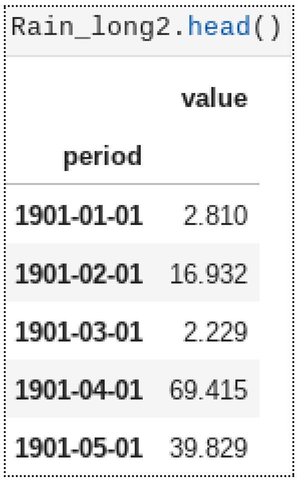
```
RainDF = pd.read_csv('Hyd_Preci_data.csv')
```

```
RainDF.head()
```

| | Year | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1901 | 2.810 | 16.932 | 2.229 | 69.415 | 39.829 | 87.484 | 225.199 | 111.788 | 52.777 | 81.840 | 17.661 | 0.000 |
| **1** | 1902 | 0.000 | 0.577 | 0.000 | 15.162 | 6.642 | 72.420 | 41.118 | 152.114 | 198.357 | 93.688 | 24.580 | 13.576 |
| **2** | 1903 | 3.005 | 0.732 | 0.000 | 5.626 | 39.802 | 30.074 | 357.257 | 302.083 | 231.687 | 266.659 | 48.735 | 11.807 |
| **3** | 1904 | 0.046 | 0.577 | 4.500 | 2.442 | 34.830 | 128.298 | 169.024 | 47.167 | 161.000 | 87.700 | 1.669 | 0.000 |
| **4** | 1905 | 0.000 | 3.789 | 2.744 | 13.831 | 69.439 | 68.842 | 95.063 | 256.437 | 94.878 | 73.416 | 0.200 | 0.000 |

First, we convert it to “long data”, i.e., reshape the dataframe into a single column (Table 3)

**Table 3: Dataframe in long data format**



```
Rain_long2.head()
```

| | value |
|---|---|
| **period** | |
| **1901-01-01** | 2.810 |
| **1901-02-01** | 16.932 |
| **1901-03-01** | 2.229 |
| **1901-04-01** | 69.415 |
| **1901-05-01** | 39.829 |

Next, we checked if the data was stationary with the adfuller() function, which gave us a p-value of 1.77x10^(-7), which satisfies the condition necessary for the data to be stationary. Next, we used the pmdarima() function to determine model orders. The best order was SARIMAX(0,0,0)(0,1,1)12. We took the last five years (= 60 months) of our data as the test set to check the accuracy of our model.

## DL and ML models

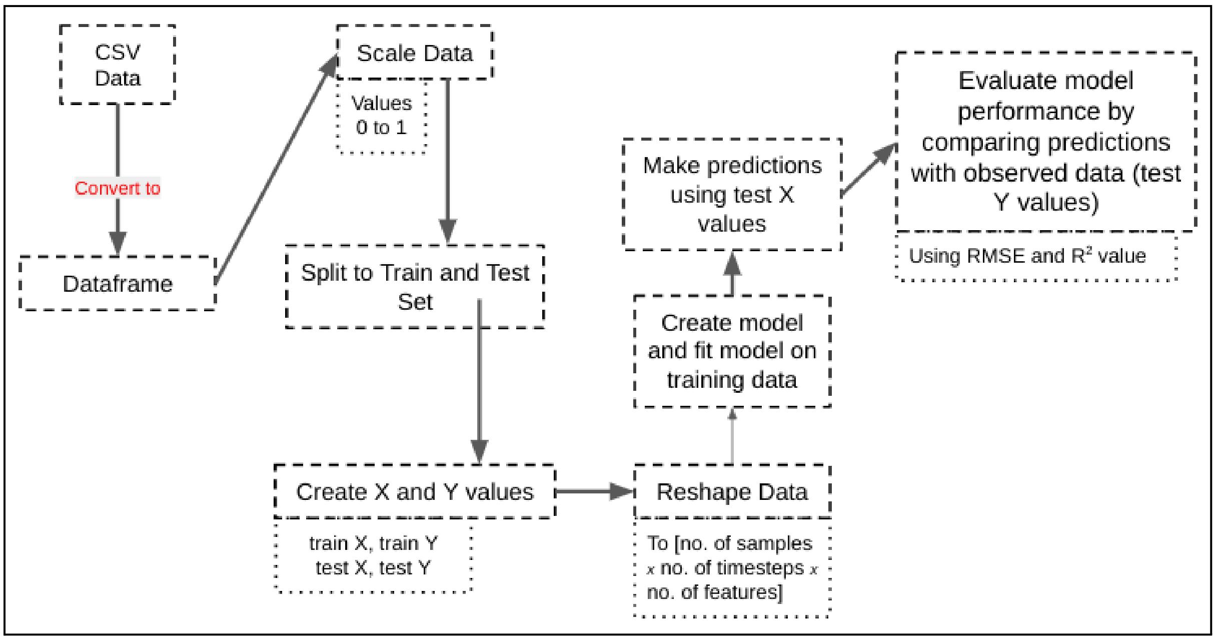

CSV Data
Convert to
Dataframe
Scale Data
Values 0 to 1
Split to Train and Test Set
Create X and Y values
train X, train Y test X, test Y
Reshape Data
To [no. of samples x no. of timesteps x no. of features]
Create model and fit model on training data
Make predictions using test X values
Evaluate model performance by comparing predictions with observed data (test Y values)
Using RMSE and R² value

Fig. 7: Steps used for processing the data and creating Deep Learning and Machine Learning models

First, we **scale** the data, i.e., we convert all the values of our dataset to the range 0-1. A smaller range of data makes it easier for the model to train and predict.

Next, we need to **split** the data into train and test sets. In our case, we took the first 80% of the sample as the training set and the rest 20% as the test set.

**Timesteps** are the number of data points the model looks back at to predict the next data point. So if we build a model with a timestep of 11 for our data, the model will look back at the last 11 months of rainfall to predict the 12th month of rainfall, which is what we did. For LSTM models, we need to specify the number of timesteps.

Based on our preferred timesteps, we create the **X and Y values** from training and test sets [trainX, trainY, testX, testY]. The trainX and trainY represent a matrix of data points on which the model is trained, where trainY represents the values that the model should predict when it takes trainX as input. So basically, **model (trainX) = trainY**.

The test set evaluates how well the model has been trained. testX values are fed into the model to get predictions, and these predictions are compared with testY values(i.e., observed values) to evaluate $R^2$ and RMSE.

Next, we **reshape** our training data. This is only done for the LSTM model because for building an LSTM model, we use the Keras library in python. The Keras LSTM model takes in data in a specific shape: *[no. of samples x no. of timesteps x no. of features].* So our model needs to be a 2D array of shape 1x3, which is easily done using the reshape() method of python. For other ML models, this reshaping step is not required.

For our data, the no of samples was 967, the no. of timesteps was taken as 11, and the no of features was taken as 1 for the univariate model (we only take the precipitation values for autocorrelation) and 5 for the multivariate model (we take the values of precipitation, cloud cover, vapour pressure, diurnal temperature, wet day frequency).

## Results

### Model 1. SARIMA Without using exogenous variables

Using a simple SARIMA model for autocorrelation, we got an RMSE (root mean squared error) of 65.73 and an r2-score of 0.59.

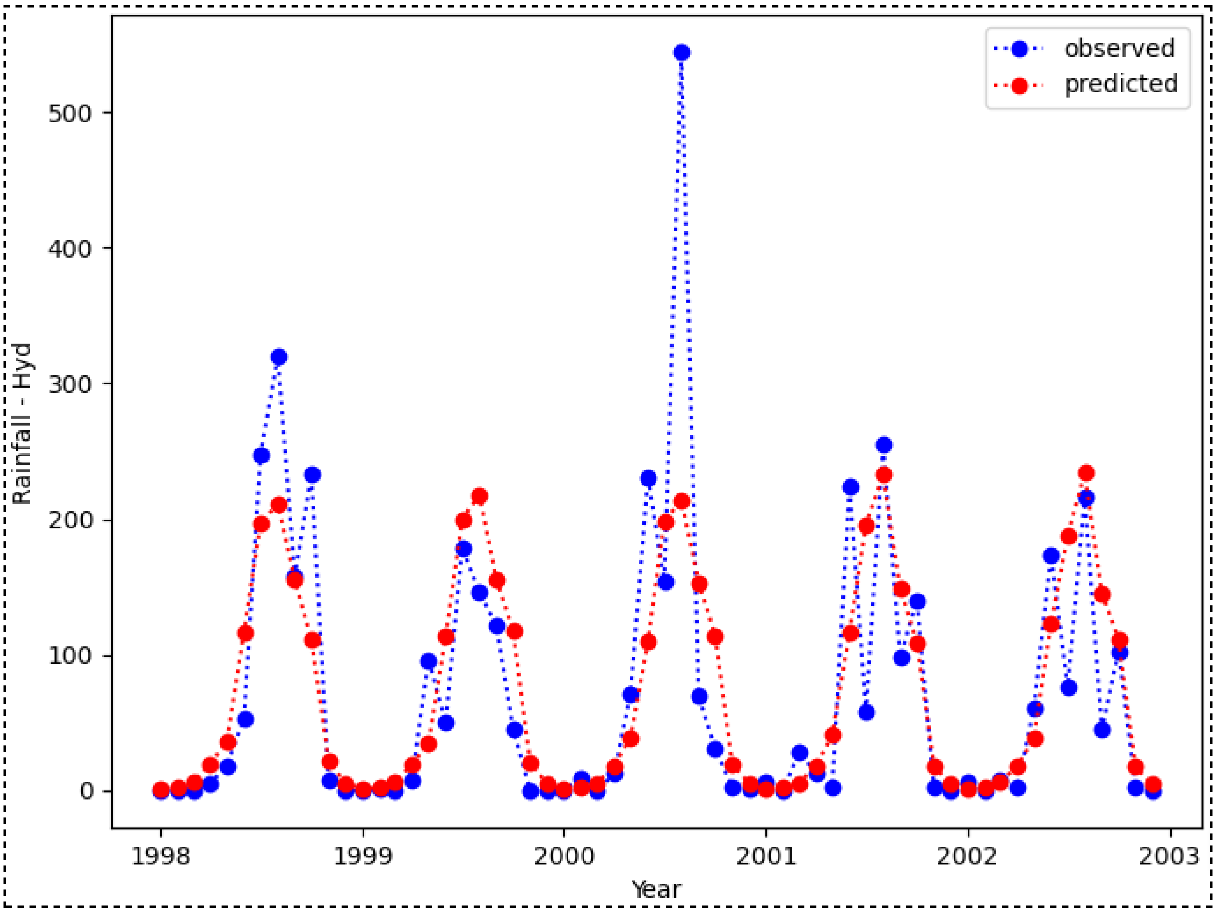


Fig. 8: Observed and predicted values for the last 5 years of the dataset - Model 1

### Model 2. SARIMA Using exogenous variables

We used the following exogenous variables: Wet Day frequency, Vapour pressure, Cloud cover, and Diurnal Temperature range. First, we used them individually, then used them taking one or more at a time. The model order was kept the same as before SARIMAX(0,0,0)(0,1,1)12

Wet day frequency, used individually, gave the best results with an $R^2$-value of 0.894 and RMSE of 33.76, which are pretty good values.

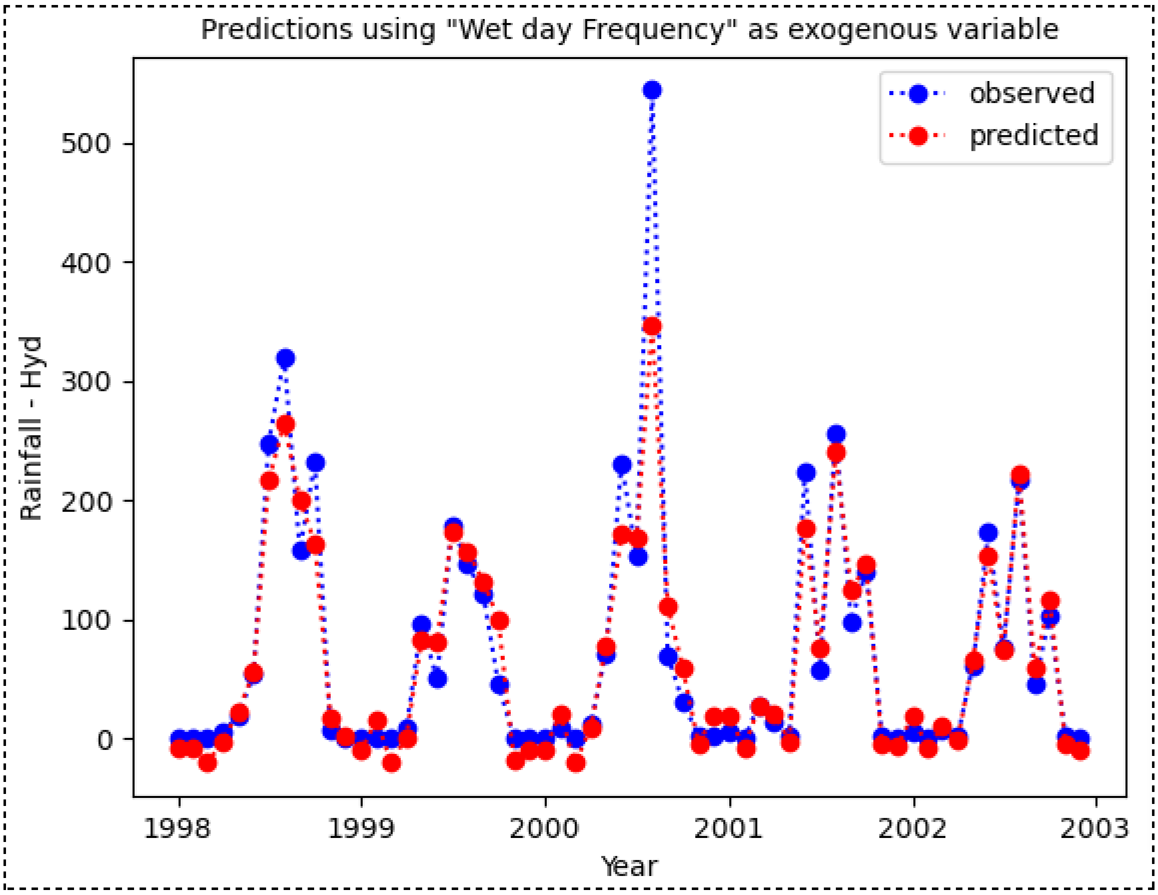


Fig. 9: Comparison of observed and predicted values when using Wet Day Frequency for the last 5 years - Model 2

**Table 4: Performance of the SARIMA model using different Exogenous variables**

| Exogenous Variable used | R2-value | RMSE |
|---|---|---|
| Wet Day frequency | 0.89 | 33.77 |
| Cloud Cover | 0.67 | 59.34 |
| Vapour Pressure | 0.62 | 63.96 |
| Diurnal Range | 0.59 | 65.75 |
| 4 together(Wet Day freq., Cloud Cover, Vapour Pressure, Diurnal Range) | 0.89 | 34.39 |

Other variables could have performed better. Results are shown in Table 4. Taking all four together (Wet Day freq., Cloud Cover, Vapour Pressure, Diurnal Range) into the model, we get an $R^2$-value of 0.890, which is still slightly less than the $R^2$-value of 0.894 that is obtained using only Wet Day Frequency.

**Table 5: Out-of-sample results obtained using the two SARIMAX models**

| ***SARIMAX Model*** | $R^2$ | RMSE |
|---|---|---|
| Without using exogenous variables | 0.57 | 68.23 |
| Using exogenous variables [Wet Day Frequency] | 0.89 | 33.77 |

**Table 6: In-sample results obtained using the two SARIMAX models**

| ***SARIMAX Model*** | $R^2$ | RMSE |
|---|---|---|
| Without using exogenous variables | 0.59 | 65.73 |
| Using exogenous variables [Wet Day Frequency] | 0.90 | 32.57 |

## Model 3. Univariate LSTM model results

Our model had 16 units in the first layer, 8 units in the second layer and 1 unit as the last layer. The loss function was taken as a mean squared error. It was trained for 200 epochs.

```
model1 = Sequential()
model1.add(LSTM(16, input_shape=(11, 1)))
model1.add(Dense(8))
model1.add(Dense(1))
```

Our model gave an r2 score of 0.55 and an RMSE of 69.65.

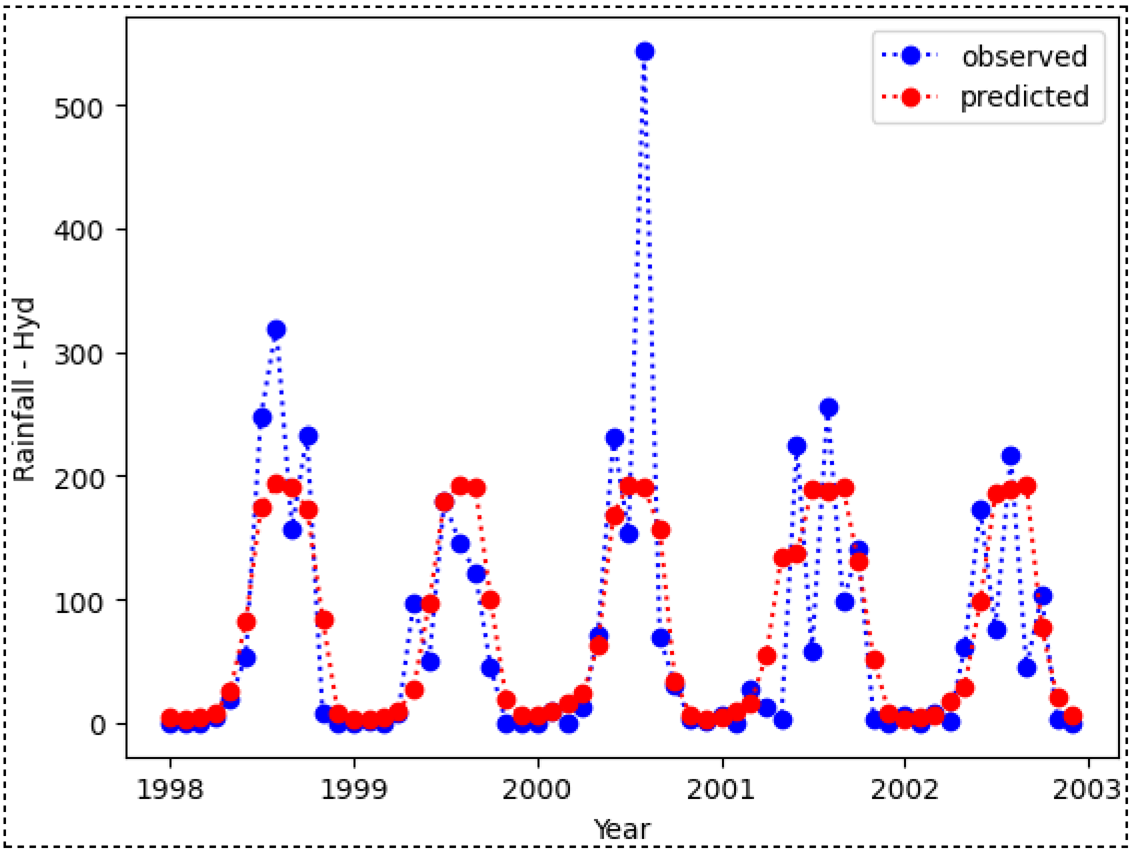


Fig. 10: Graphical representation of the predicted and observed (values for the last 5 years - Model 3

## Model 4. Multivariate LSTM model results: training and testing

Our model had 16 units in the first layer, 8 units in the second layer and 1 unit in the last layer. The loss function was taken as mean-squared error. It was trained for 100 epochs.

```
model4 = Sequential()
model4.add(LSTM(16, input_shape=(11, 5)))
model4.add(Dense(8, 'relu'))
model4.add(Dense(1, 'linear'))
```

The model returned an r2 score of 0.47, which is relatively low.

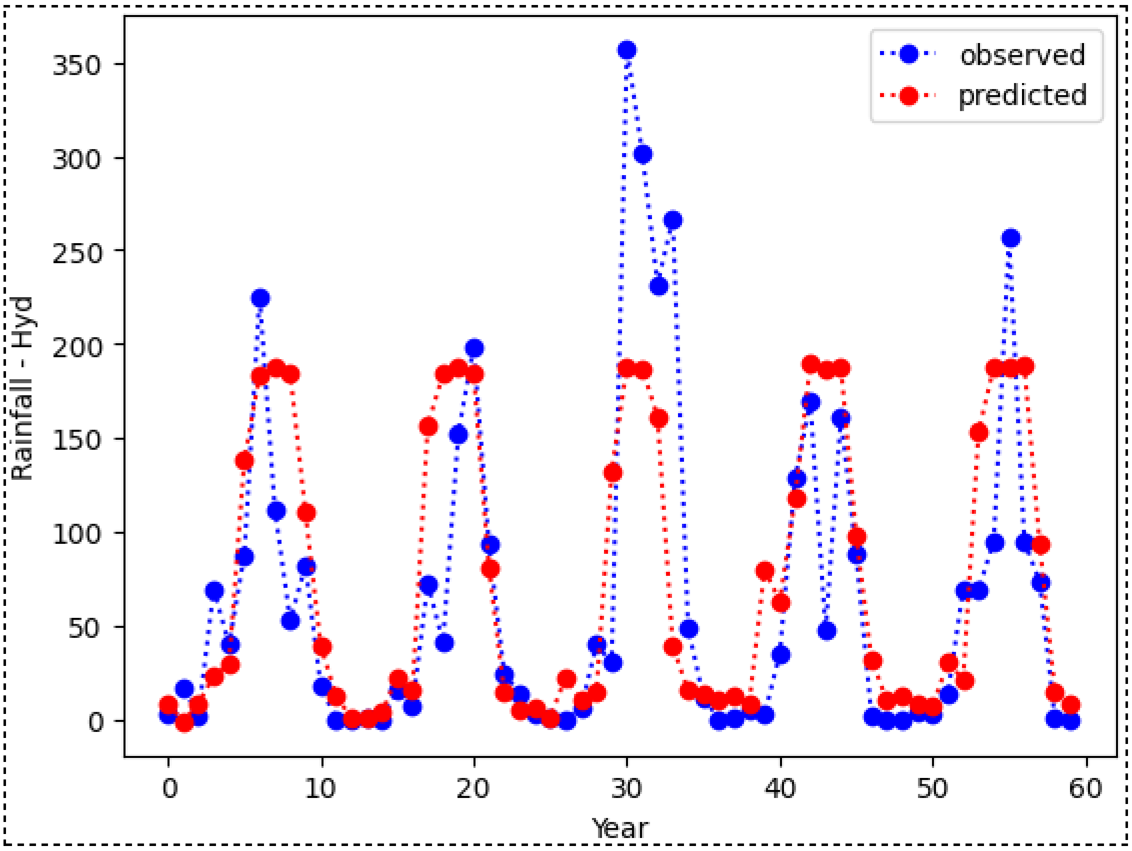


Fig. 11: Graphical representation of the predicted and observed values - Model 4

Consistent with the findings of Liu et al. (2019), the LSTM model seems to overfit on the training set very easily; also, when we try to prevent overfitting, the prediction accuracy is significantly reduced.

**Table 7: Results obtained using the two LSTM models (using Test set)**

| ***LSTM Model*** | $R^2$ | RMSE |
|---|---|---|
| Univariate(no exogenous variables) | 0.55 | 69.65 |
| Multivariate (using 4 exogenous variables) | 0.47 | 62.45 |

**Table 8: Training set performance of the two models**

| ***Training set of LSTM Model*** | $R^2$ | RMSE |
|---|---|---|
| Univariate(no exogenous variables) | 0.66 | 51.38 |
| Multivariate (using 4 exogenous variables) | 0.43 | 66.60 |

## Model 5. XGBoost

Our model had 2000 estimators and a maximum tree depth of 4. The model automatically assigned feature importances to our input variables as shown in Fig. 12. In our exogenous SARIMAX model we had seen that Wet Day Frequency provided the best predictions and the XGBoost model seems to automatically recognise it and adjust the weights accordingly.

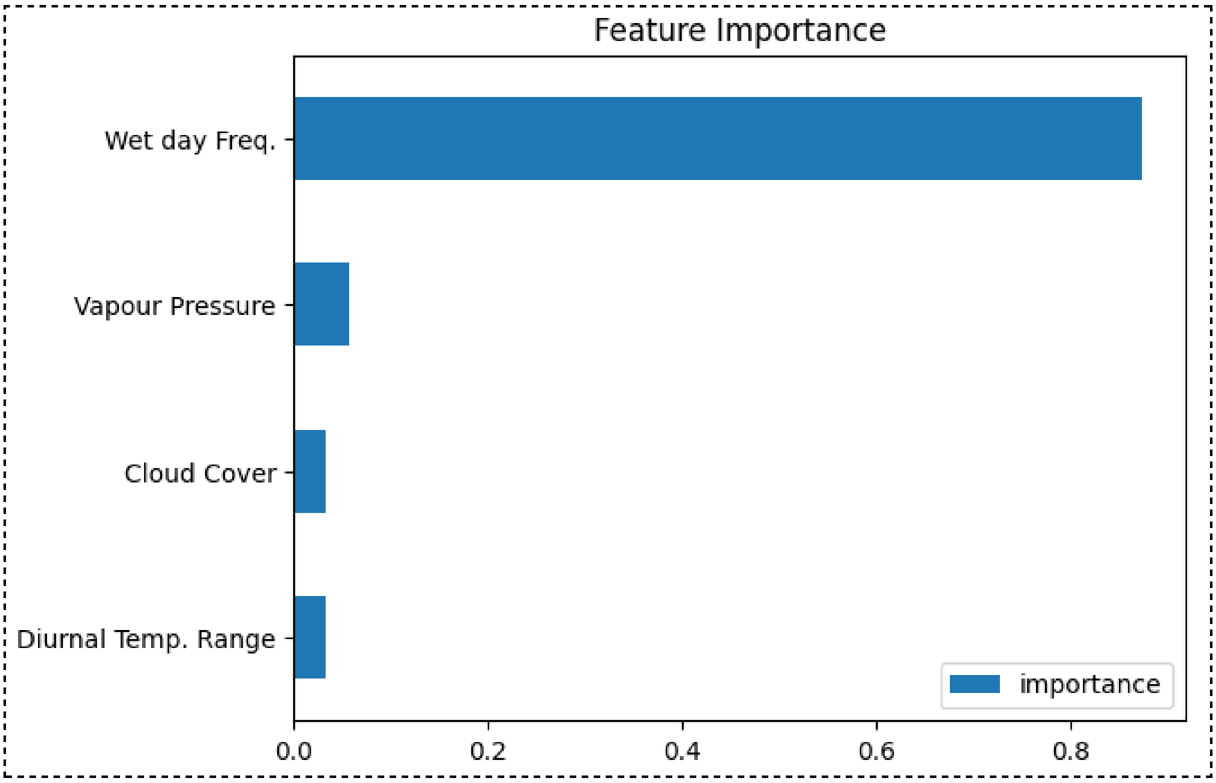


Fig. 12: Model assigned feature importances - Model 5

The R-square value obtained was 0.931, which is quite good, and the RMSE was 27.17. Thus this was our best-performing model.

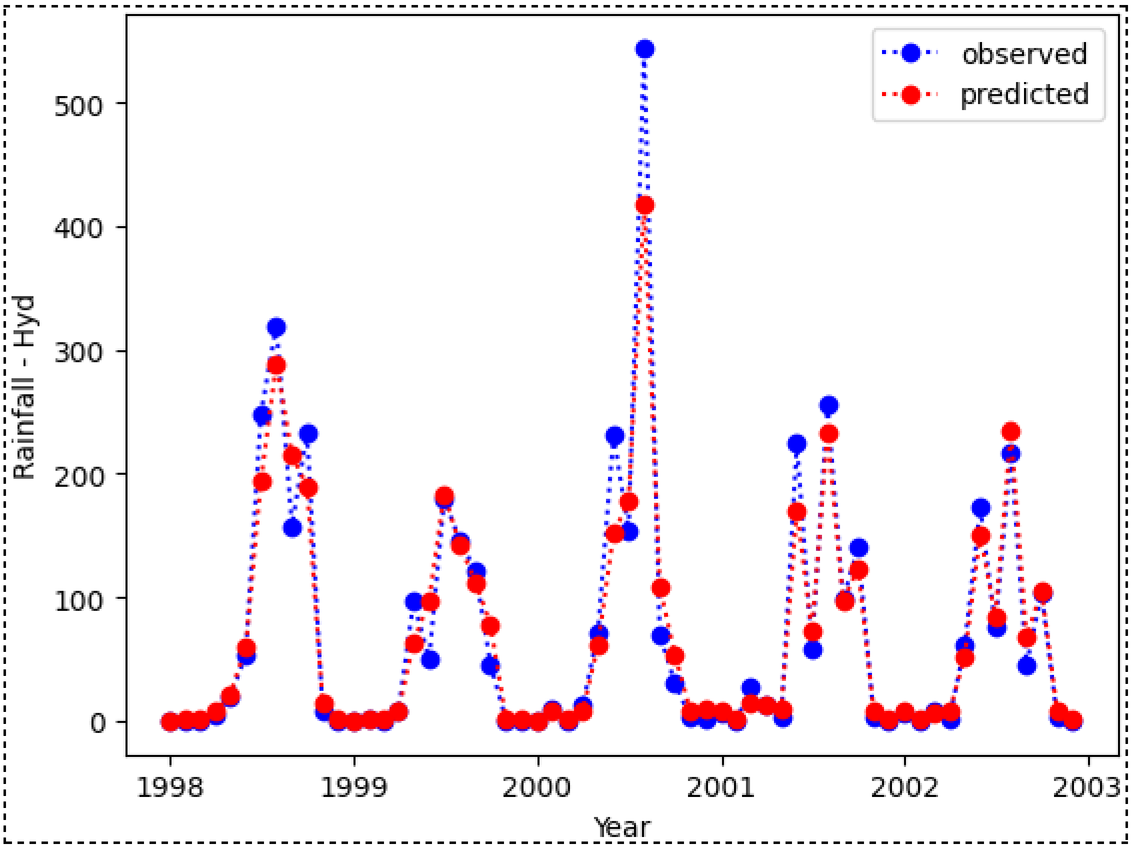


Fig. 13: Graphical representation of the predicted and observed values - Model 5

**Table 9: XGBoost results**

| ***Model 5. XGBoost*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.96 | 16.98 |
| Test Set | 0.93 | 27.17 |

## Model 6. Adaboost

This model was made with 2000 estimators, using python's AdaBoostRegressor() function from the sci-kit learn library. Here too, we see that the model has assigned feature importances that match our previous observation of Wet Day Frequency being the best exogenous predictor for our target variable.

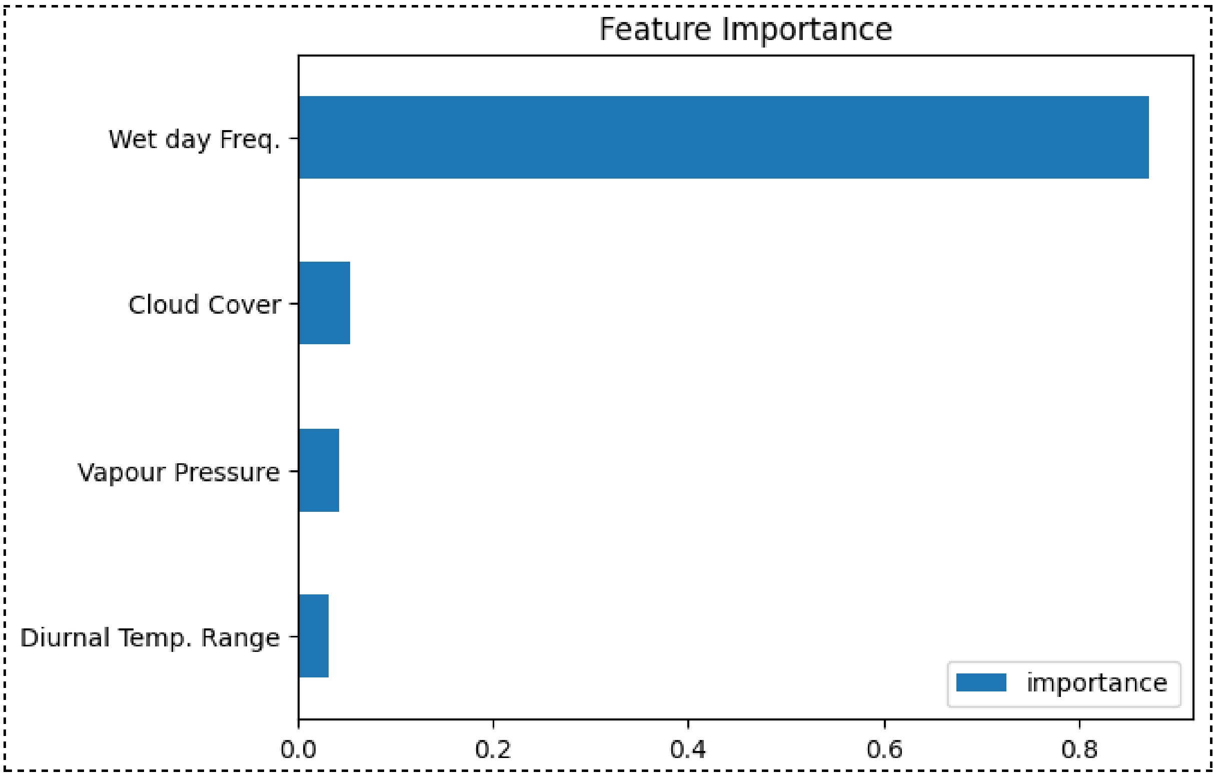


Fig. 14: Model-assigned feature importances - Model 6

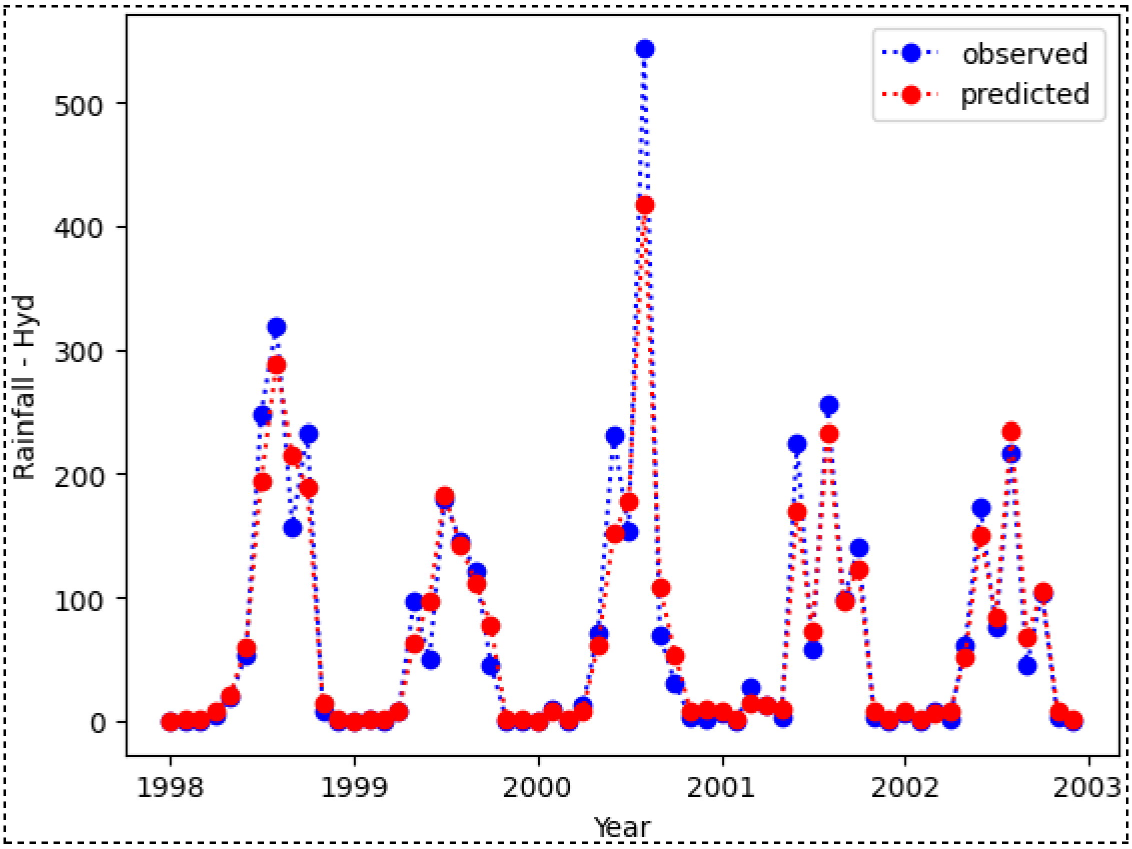


Fig. 15: Graphical representation of the predicted and observed values - Model 6

**Table 10: AdaBoost results**

| ***Model 6. Adaboost*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.91 | 27.23 |
| Test Set | 0.89 | 33.95 |

## Model 7. SVR

The model was implemented with python's SVR() function from sklearn.svm library. This model gave the poorest results, with an R-square value of 0.33 and an RMSE of 85.29. Unlike Model 4, overfitting was not the reason behind the poor performance of this model.

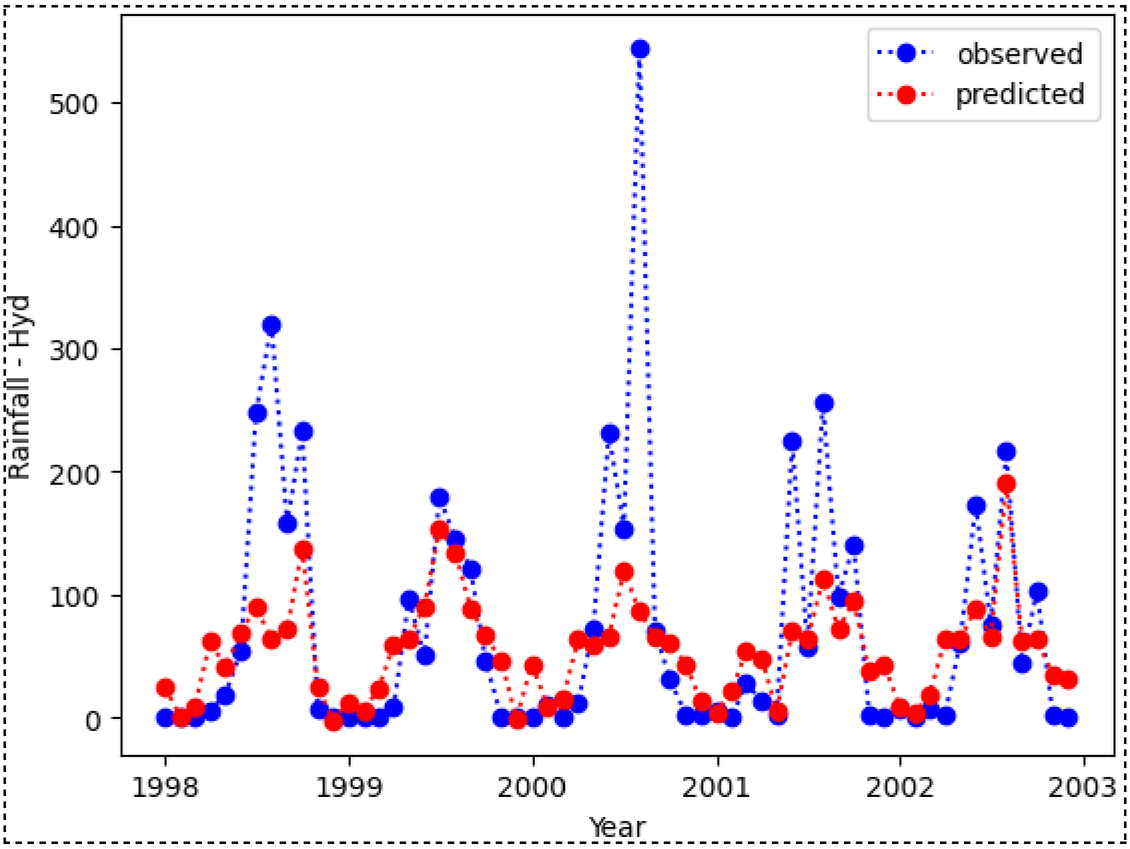


Fig. 16: Graphical representation of the predicted and observed values - Model 7

**Table 11: SVR Results**

| ***Model 7. SVR*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.66 | 51.98 |
| Test Set | 0.33 | 85.29 |

## Model 8. CART

The model was made using python’s DecisionTreeRegressor() function from the sci-kit learn library, with a maximum tree depth of 10. Again, this model assigned feature importances similar to the two boosting models (5 and 6).

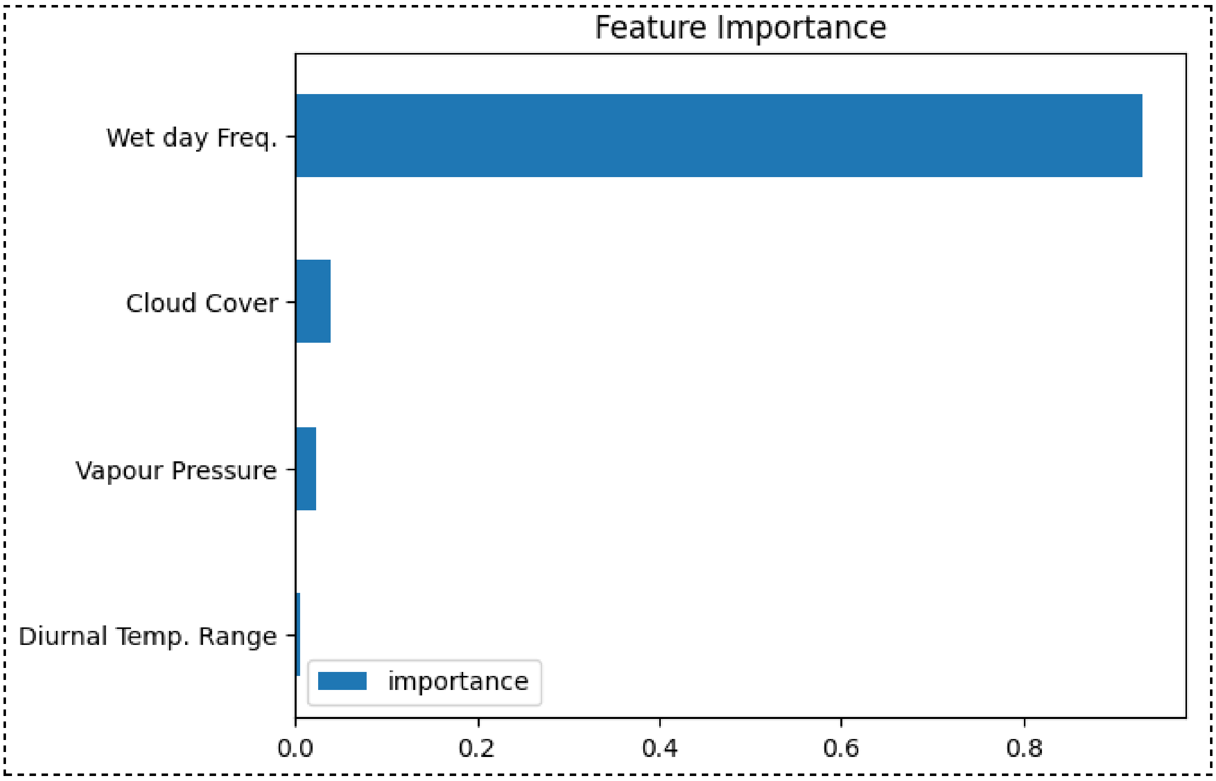


Fig. 17: Model-assigned feature importances - Model 8

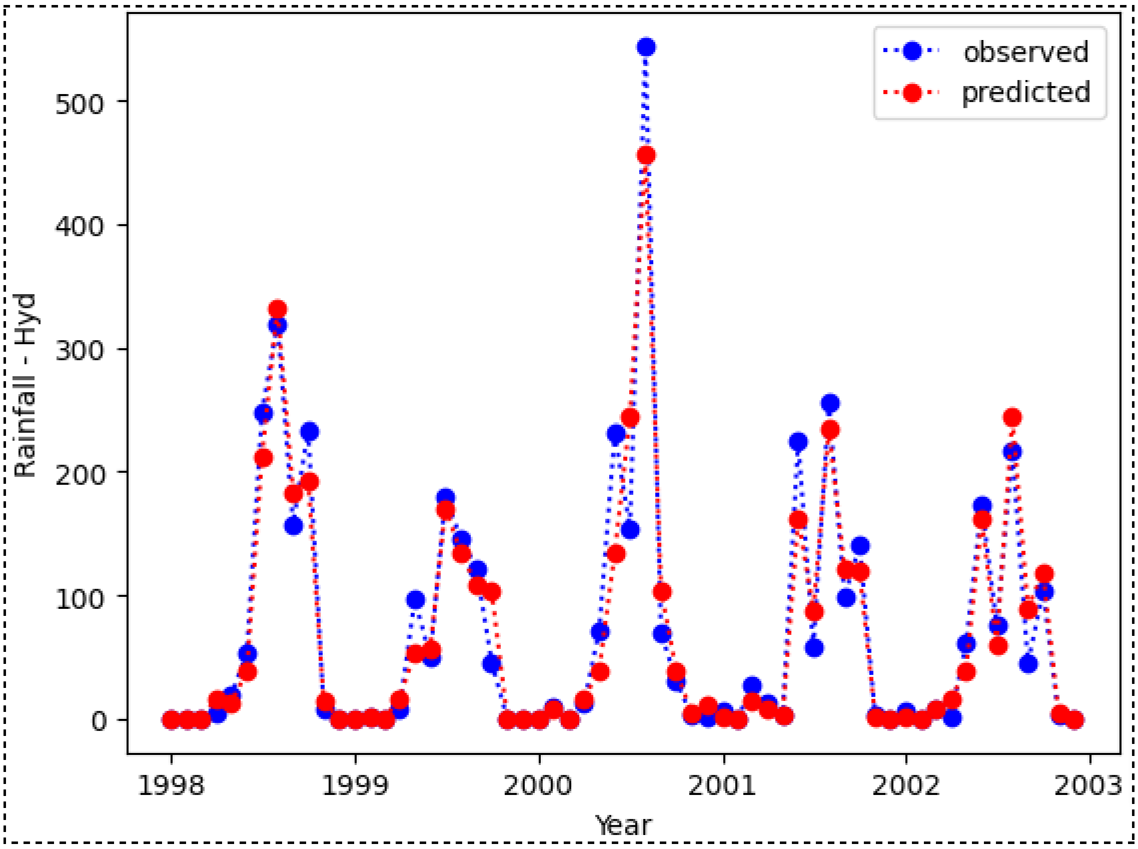


Fig. 18: Graphical representation of the predicted and observed values - Model 8

CART gave an R-square value of 0.927, which is almost as good as the R-square value of XGBoost (0.931).

**Table 12: CART results**

| ***Model 8. CART*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.98 | 13.97 |
| Test Set | 0.93 | 28.21 |

## Model 9. Extra Trees

The model was made using python's ExtraTreesRegressor() function from the sci-kit learn library. The number of estimators (total number of trees) used in the model was 2000.

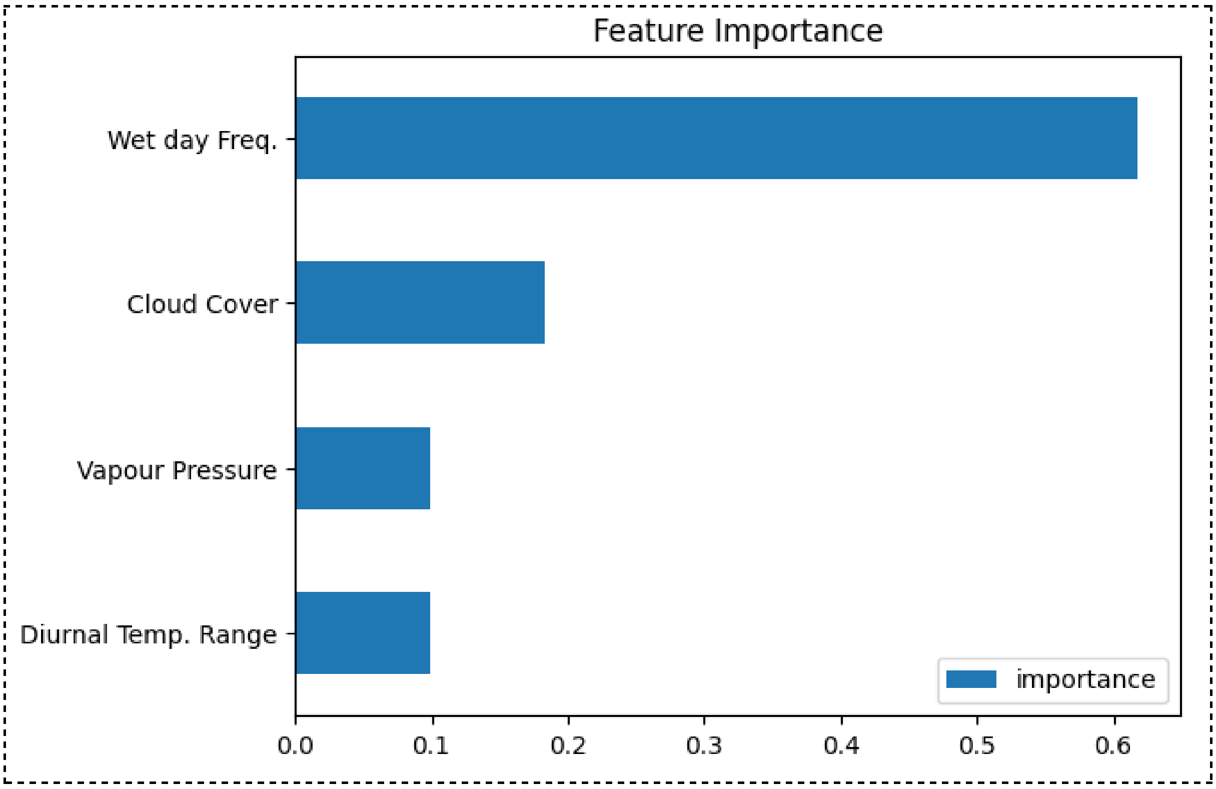


Fig. 19: Model assigned feature importances - Model 9

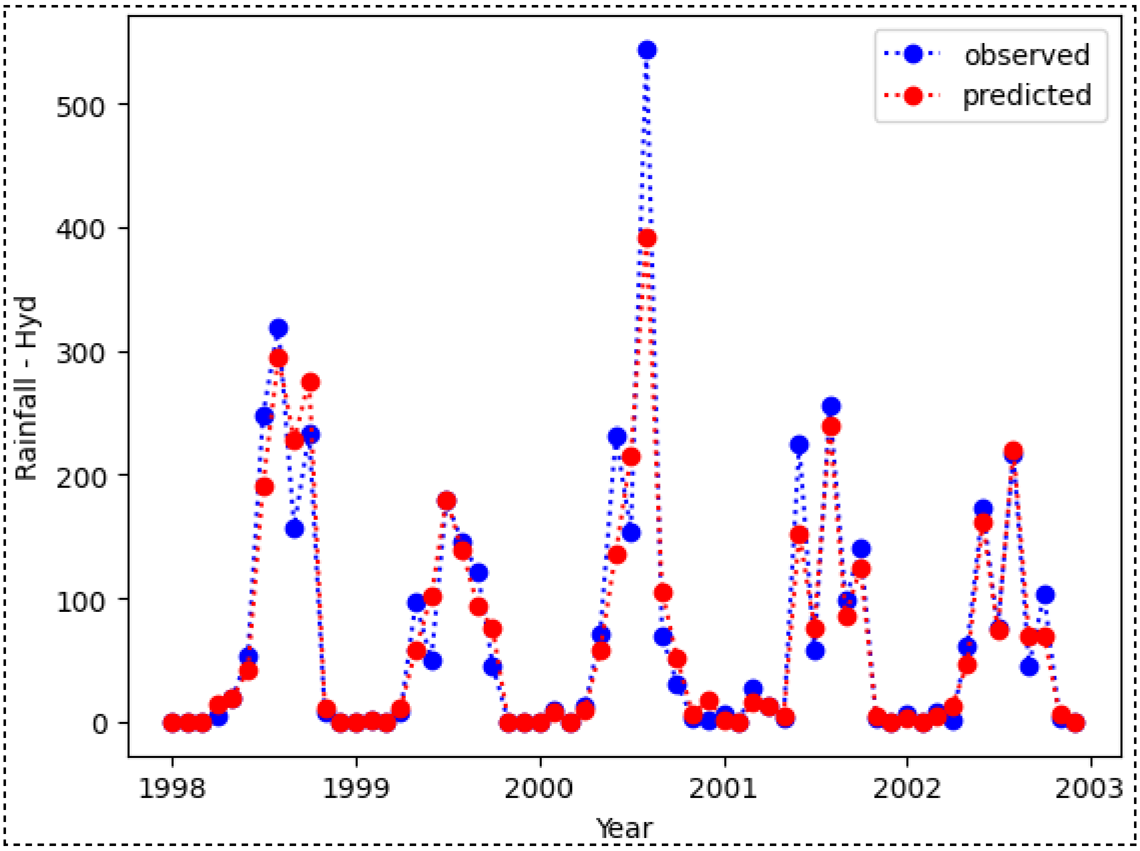


Fig. 20: Graphical representation of the predicted and observed values - Model 9

**Table 13: Extra Trees Results**

| ***Model 9. Extra Trees*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.99 | 0.42 |
| Test Set | 0.90 | 32.46 |

## Model 10. Bagging Regressor

The model was made using python’s BaggingRegressor() function from the sci-kit learn library. The number of estimators (base learners) used was 2000.

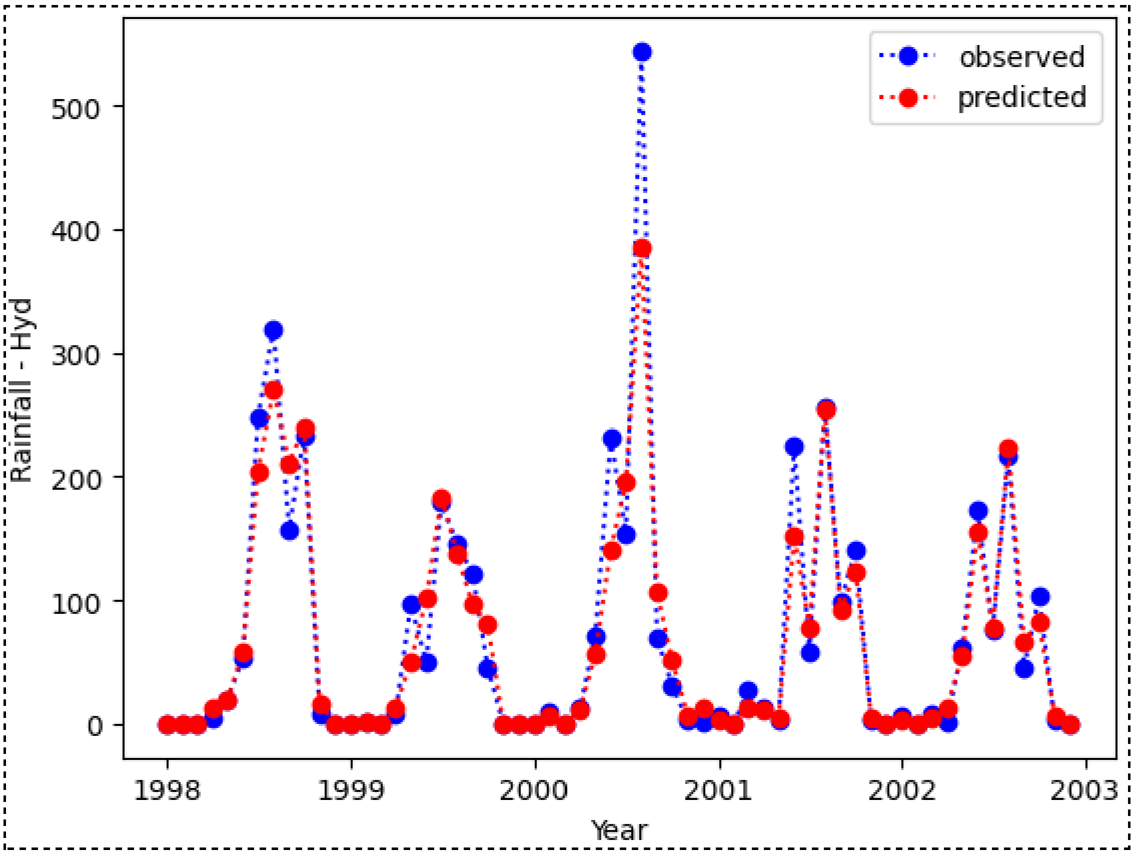


Fig. 21: Graphical representation of the predicted and observed values - Model 10

**Table 14: Bagging Regressor Results**

| ***Model 10. Bagging Regressor*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.98 | 11.01 |
| Test Set | 0.91 | 31.36 |

## Model 11. Random Forest

The model was made using python's RandomForestRegressor() function from the sci-kit learn library. The number of estimators used was 1000.

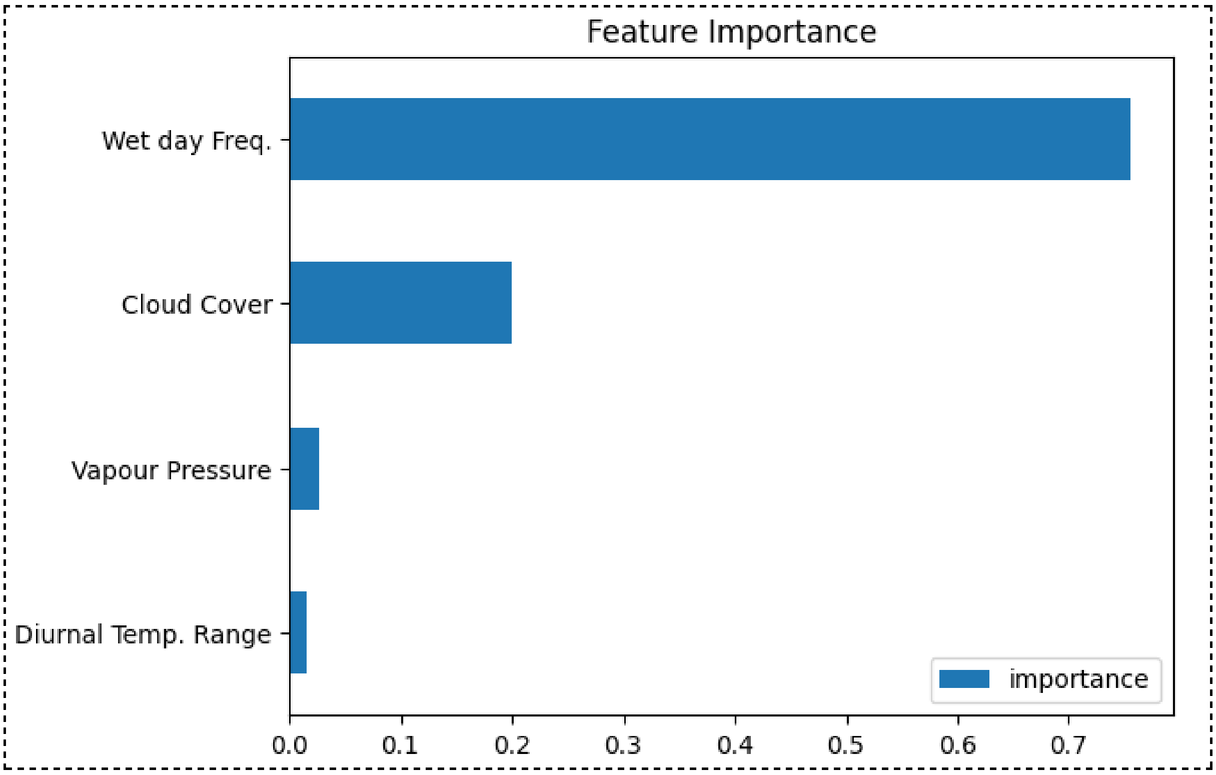


Fig. 22: Model assigned feature importances - Model 11

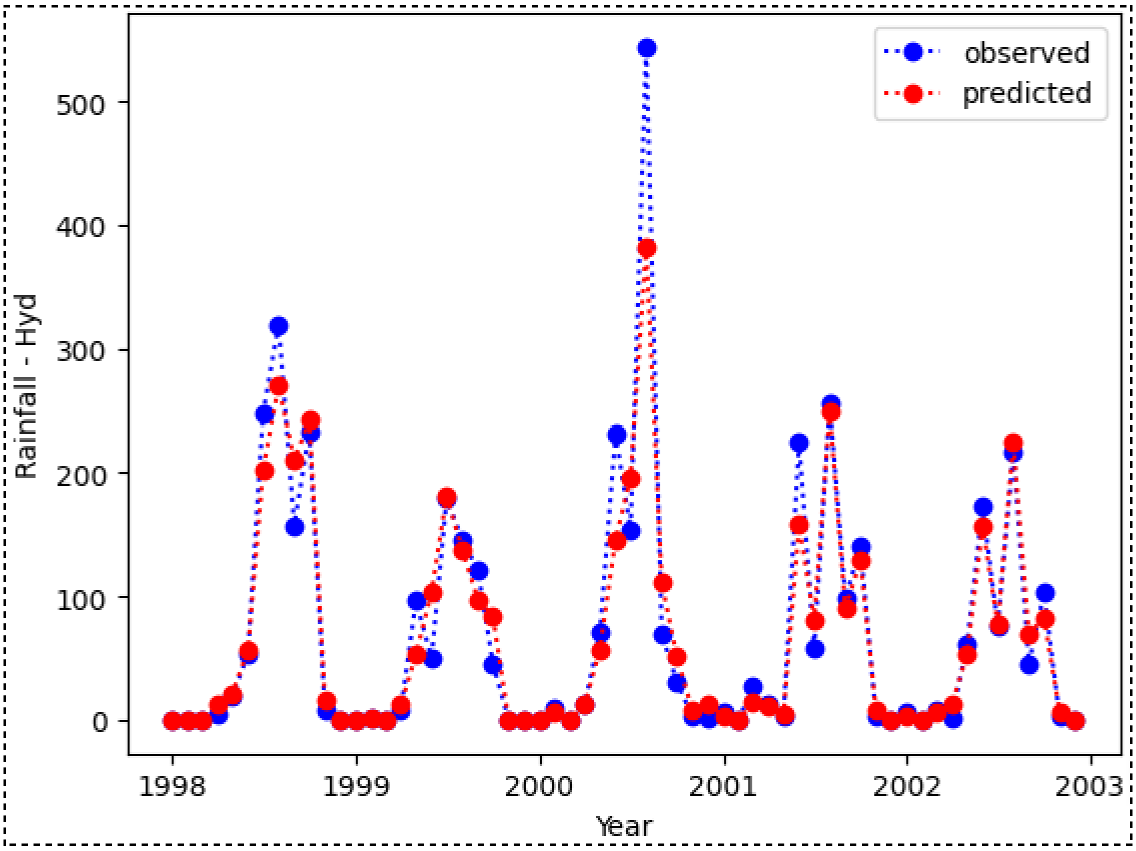


Fig. 23: Graphical representation of the predicted and observed values - Model 11

**Table 15: Random Forest results**

| ***Model 11. Random Forest*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.98 | 10.89 |
| Test Set | 0.91 | 31.38 |

## Implementation of Model 5 on Bengaluru and Chennai’s Data

To check for spatial transferability, we tested our best-performing model on Bangalore and Chennai rainfall datasets within the same 100-year time period and taking the same train-test split.

The model assigned different feature importances for the two cities, as shown in Figures 24 and 25. An interesting observation is how the same model sets slightly different feature importances for the same exogenous variables when applied to an other city. It makes sense since weather variables correlate differently in other regions. Yet at the same time, all three cities lie roughly in the same geo-climatic area, so the difference is not very heavy.

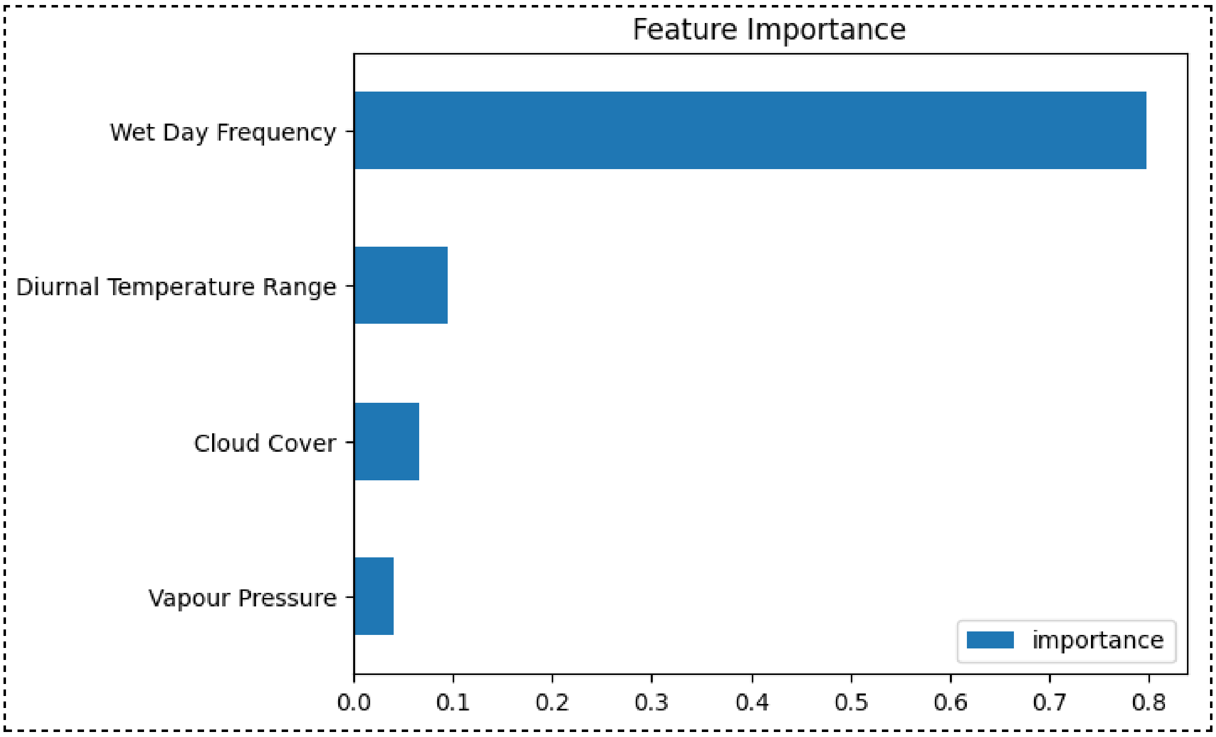


Fig. 24: Model assigned Feature importances for Bengaluru

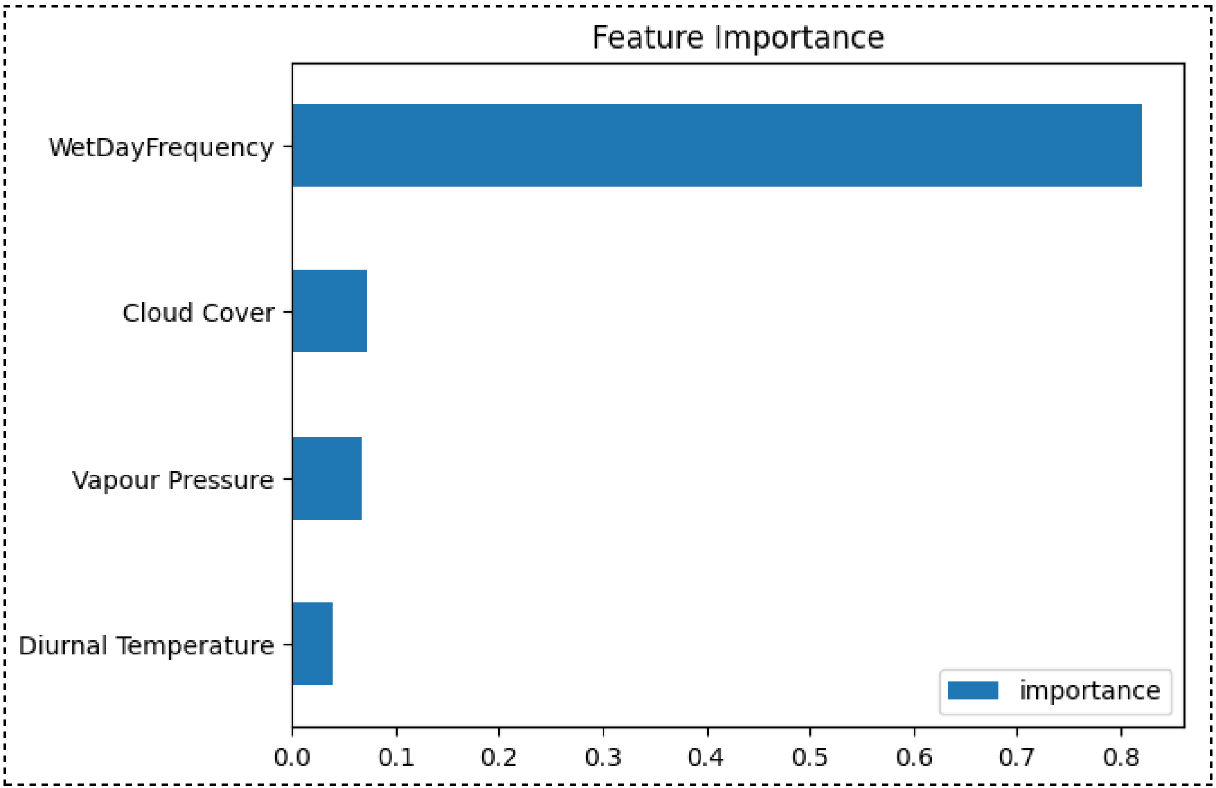


Fig. 25: Model assigned Feature importances for Chennai

The graphical representation of the predicted and observed values are shown in Fig.26 (Bengaluru) and Fig. 27 (Chennai).

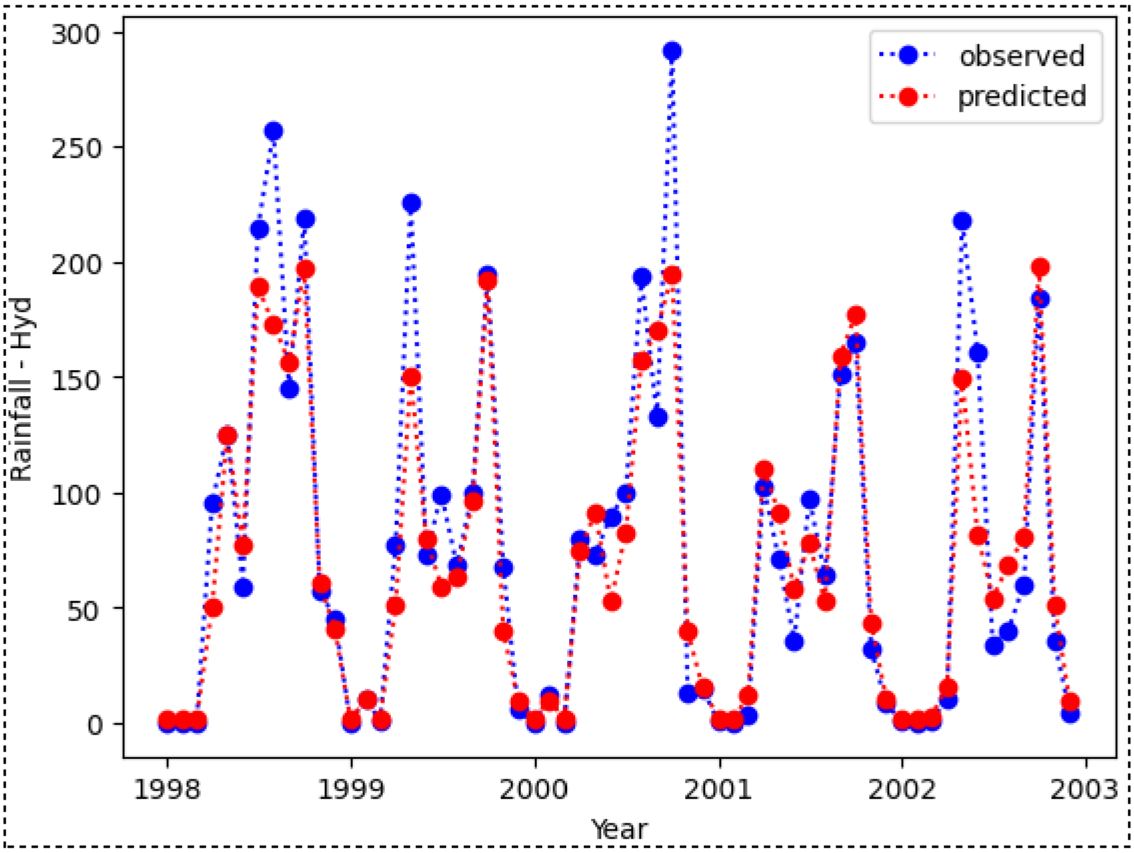


Fig. 26: Model 5 on Bengaluru: Predicted vs Observed values (test set)

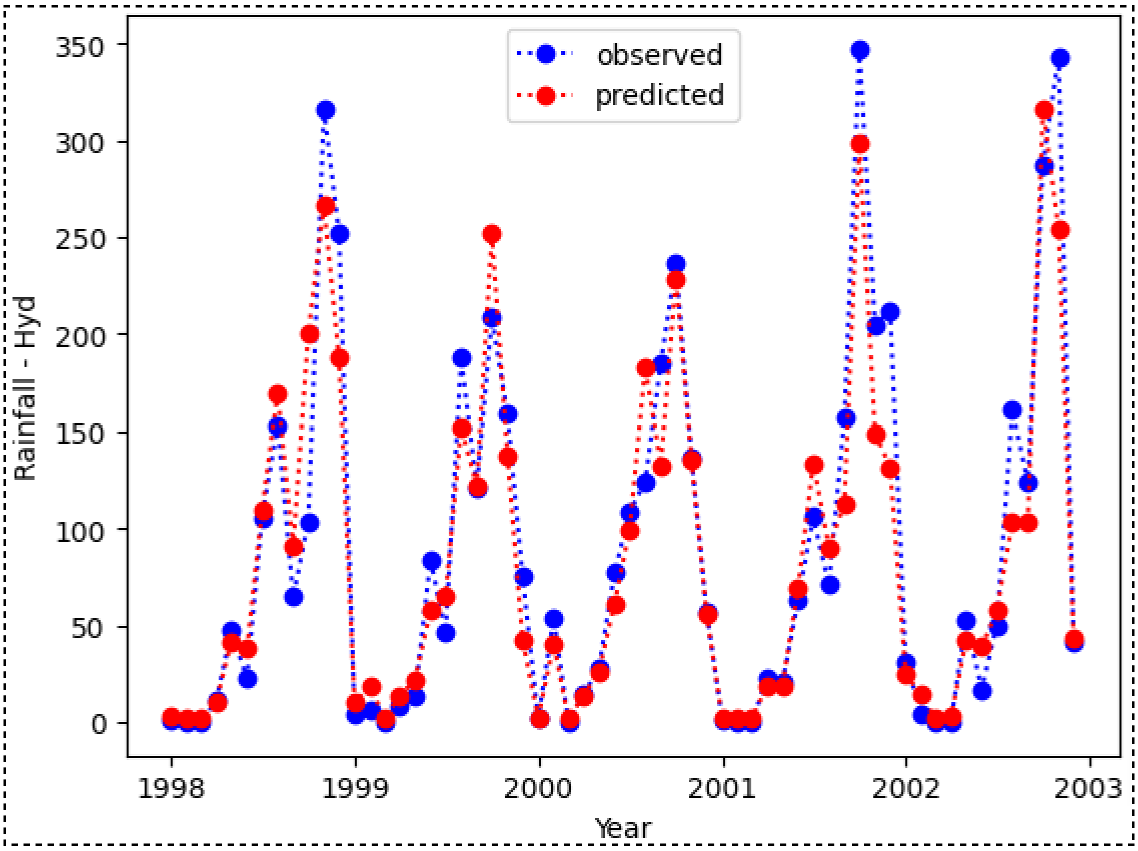


Fig. 27: Model 5 on Chennai: Predicted vs Observed values (test set)

The model returned an R-square value of 0.86 and 0.89 for Bengaluru and Chennai, respectively, which are pretty good.

**Table 16: Model 5 on Bengaluru Results**

| ***Model 5: Bengaluru*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.87 | 24.42 |
| Test Set | 0.86 | 28.73 |

**Table 17: Model 5 on Chennai Results**

| ***Model 5: Chennai*** | R-square value | RMSE |
|---|---|---|
| Training Set | 0.92 | 34.28 |
| Test Set | 0.89 | 31.29 |

## Results Table

The results of all the models

(test set) are compiled in Table 18, sorted from best to worst by performance. All the ML models performed better than LSTM, except for SVR. All the ML models, barring AdaBoost, outperformed SARIMA and returned R-square values greater than 0.9. CART performed better than all the ensemble models except for XGBoost. The boosting and bagging models seem to perform equally well on average among ensemble models. However, no bagging ensemble model outperformed the simpler CART model.

**Table 18: Model results, sorted from worst to best**

| Model No. | Model | Parameters / Architecture | $R^2$ | RMSE |
|---|---|---|---|---|
| 7 | SVR | ε = 0.05 | 0.33 | 85.29 |
| 4 | LSTM - Multivariate | (16-8-1) | 0.47 | 62.45 |
| 3 | LSTM - Univariate | (16-8-1) | 0.55 | 69.65 |
| 1 | SARIMA(without Exogenous variables) | $(0,0,0)(0,1,1)_{12}$ | 0.57 | 68.23 |
| 6 | Adaboost | 2000 estimators | 0.89 | 33.95 |
| 2 | SARIMA with Exogenous variables | $(0,0,0)(0,1,1)_{12}$ | 0.89 | 33.77 |
| 9 | Extra Trees | 2000 estimators | 0.90 | 32.46 |
| 11 | Random Forest | 1000 estimators | 0.91 | 31.38 |
| 10 | Bagging Regressor | 2000 estimators | 0.91 | 31.36 |
| 8 | CART | Max depth = 10 | 0.93 | 28.21 |
| 5 | XGBoost | 2000 estimators, depth = 4 | 0.93 | 27.17 |

# Conclusion

This study set out to build a model for predicting rainfall over the Hyderabad region of Telangana. We have used SARIMA, LSTM, XGBoost, AdaBoost, SVR, CART, Extra Trees, Bagging Regressor, and Random Forest models, using RMSE and $R^2$ values as metrics for model performance. For SARIMA, we have found that using exogenous variables (Model 2) gives better results than only precipitation data (Model 1). For LSTM, the univariate Model 3, which used only precipitation data, showed better results than Model 1. However, the multivariate LSTM Model 4, which uses exogenous correlated variables, gave a significantly lower prediction accuracy than the exogenous SARIMA. This is consistent with our literature review of how LSTM models tend to overfit the training set if our data is not very large.

ML models significantly outperformed LSTM. CART gave better results than the ensemble models, except for XGBoost, which was the best-performing model.

The proficiency of tree-based models for regression problems is seen from the results of ML models since all of them outperformed SVR. Since the size of our data is not very large, simpler ML models gave us much better results than the deep learning model

LSTM; other than AdaBoost, all the ML models outperformed SARIMA. We also found that among the ML models, CART performed better than all the ensemble models except for XGBoost. This again shows us how simpler models often generalize data better and have a lesser tendency to overfit the training set. Most of these models auto-assign feature importance to the various exogenous factors depending on how much the value of the given variable affects our target variable (precipitation).

To check its validity, we also applied our best-performing model (Model 5) to Bangalore and Chennai weather datasets since these cities lie in roughly the same geo-climatic region as Hyderabad, and they returned reasonably good results.

## References

Liyew, Chalachew Muluken, and Haileyesus Amsaya Melese. "Machine learning techniques to predict daily rainfall amount." *Journal of Big Data 8*.1 (2021): 1-11.

Hardwinarto, Sigit, and Marlon Aipassa. "Rainfall monthly prediction based on artificial neural network: a case study in Tenggarong Station, East Kalimantan-Indonesia." *Procedia Computer Science 59* (2015): 142-151.

Shreemali, Jitendra, et al. "Rainfall Prediction for Udaipur Rajasthan Using Machine Learning Models Based on Temperature Vapour Pressure and Relative Humidity." *International Journal of Recent Technology and Engineerin*g 8.

Shardoor, Nikhilkumar B., and Mandapati Venkateshwar Rao. "Rainfall Prediction and Forecasting using Time Series Analysis." *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 6, no. 4, 2019, pp. 72-83.

Sarker, Iqbal H. "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions." SN Computer Science 2.6 (2021): 1-20.

Siami-Namini, Sima, Neda Tavakoli, and Akbar Siami Namin. "A comparison of ARIMA and LSTM in forecasting time series." 2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE, 2018.

Barrera-Animas, Ari Yair, et al. "Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting." Machine Learning with Applications 7 (2022): 100204.

Rahman, Mohammad Atiqur, Lou Yunsheng, and Nahid Sultana. "Analysis and prediction of rainfall trends over Bangladesh using Mann–Kendall, Spearman's rho tests and ARIMA model." Meteorology and Atmospheric Physics 129.4 (2017): 409-424.

Geetha, A., and G. M. Nasira. "Time-series modelling and forecasting: Modelling of rainfall prediction using ARIMA model." International Journal of Society Systems Science 8.4 (2016): 361-372.

Somvanshi, V. K., et al. "Modeling and prediction of rainfall using artificial neural network and ARIMA techniques." J. Ind. Geophys. Union 10.2 (2006): 141-151.

Barrera-Animas, Ari Yair, et al. "Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting." Machine Learning with Applications 7 (2022): 100204.