FOSSEE Summer Fellowship Report

on

FLOSS - R

submitted by

Amish Sharma (C. G. Patel Institute of Technology)

under the guidance of

Prof. Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

Prof. Pennan Chinnasamy
Centre for Technology Alternatives
for Rural Areas (CTARA)
IIT Bombay

June 15, 2020

Acknowledgement

I want to express my sincere gratitude to Prof. Kannan M. Moudgalya, Department of Chemical Engineering, IIT Bombay, for creating the FOSSEE Fellowship programme and providing students from all over India an opportunity to participate in it. I would equally like to thank my project mentor, Prof. Pennan Chinnasamy, Centre for Technology Alternatives for Rural Areas (CTARA), IIT Bombay, for his immense support, patience, motivation, knowledge & influence throughout this fellowship and for helping me on various statistical models. I would also like to express my gratitude to other members of the R FLOSS team, namely Mrs. Smita Wangikar and Mr. Digvijay Singh for their guidance and valuable inputs throughout the fellowship and also for providing me with an overview on data analysis and LaTeX. I would also like to thank the other fellows who got selected along with me, namely M. Sai Anand, Sakshee Phade and Ashwin Guptha for their support, intellectual discussions and enthusiasm.

Contents

Chapter 1

Introduction

In this report, I mention my contributions to open-source software, made in the duration of the FOSSEE Fellowship, starting from 20th April 2020 to 15th June 2020. Contributions were made using a Free-Libre/Open Source Software (FLOSS) known as "R" as a part of the FOSSEE project by IIT Bombay and MHRD, Government of India. The FOSSEE project is a part of the National Mission on Education through ICT. The thrust area is the adaptation and deployment of open-source simulation packages equivalent to proprietary software, funded by MHRD, based at the Indian Institute of Technology Bombay (IITB). My contributions involved making Spoken Tutorial scripts and developing a river level and discharge prediction system using ANN and CNN.

Chapter 2

Spoken Tutorial

The Spoken Tutorial project aims to make video tutorials on Free and Open Source Software (FOSS) available in several Indian languages. The goal is to enable the use of spoken tutorials to teach in any Indian language to learners of all levels of expertise - Beginner, Intermediate or Advanced. Every tutorial has to go through a series of stages to ensure that it is perfect for its audience, which is crucial for achieving the goal of this project. I contributed to the creation of the "Logistic Regression" and "Support Vector Machine" tutorial scripts.

2.1 Logistic Regression

Logistic regression is a classification algorithm which predicts the class of a variable based on multiple predictor variables. It uses a sigmoid function to convert the output (i.e. probabilities) into the class of a target variable. I used "PimaIndianDiabetes" [1] data set for classifying the test for diabetes of patients. The pre-defined packages used were "mlbench" [2], "pROC" [3] and "caret" [4].

2.2 Support Vector Machine

Support Vector Machine (SVM) being a supervised machine learning algorithm is used for both regression and classification. The main idea of SVM is to find an N-dimensional hyperplane, where N is the number of features, that distinctly classifies the data points. I used "framingham" data set to classify whether a patient "died," "experienced hypertension" or "left the study without experiencing either event." The pre-defined packages used were "caret" [4], "LocalControl" [5] and "e1071" [6].

Chapter 3

River Level and Discharge Prediction System

Fellows were supposed to work on a research project along with the Spoken Tutorial scriptwriting. For me, the project involved collecting and analyzing the past 19 years (2000 - 2019) of data associated with the Periyar river, Kerala. The process of analysis consisted of exploring the data, cleaning it and then applying Artificial Neural Network and Convolutional Neural Network to predict the river level and discharge at downstream based on the parameters of upstream. I used R for data analysis and LaTeX for report writing.

3.1 Abstract

Floods drastically affect a significant portion of India's population, making it the worst flood-affected country in the world. The recent disaster in Kerala during August 2018 is a typical example of the problem India faces each year due to floods [7]. Hence, flood forecasting in India is a necessity for both water resource management and disaster management. I selected the Periyar river as the target river for my project because it is the longest river in Kerala with a length of 244 km. It also covers the largest area of 5029.03 square km as compared to the other rivers in the state. It also has the highest number of sub-basins, i.e. 183 and micro watersheds, i.e. 448 in Kerala [8]. I collected rainfall (mm), river level (m) and discharge (m^3/s) data associated with two manual stations of the river, namely Vandiperiyar (upstream) and Neeleeswaram (downstream) for my analysis. Rainfall data was the same for both stations. I made use of nineteen years of data (2000 - 2019) for river level and discharge forecasting at Neeleeswaram station with one-day, two-days, and three-days lag. I individually predicted river level (m) and discharge (m^3/s). Rainfall (mm), discharge of upstream (m^3/s) and discharge of downstream (m^3/s) were the inputs for predicting the discharge of downstream (m^3/s). Similarly, for the prediction of downstream level (m), inputs were rainfall (mm), level of the upstream (m), and level of downstream (m). ANN provided 0.000802, 0.0011 & 0.00152 as the Mean Squared Error (MSE) values for downstream discharge (m^3/s) and 0.000184, 0.00277 & 0.00317 for downstream level (m) whereas CNN gave 0.000872, 0.00124 & 0.00138 as the MSE values for downstream discharge (m^3/s) and 0.00194, 0.00270

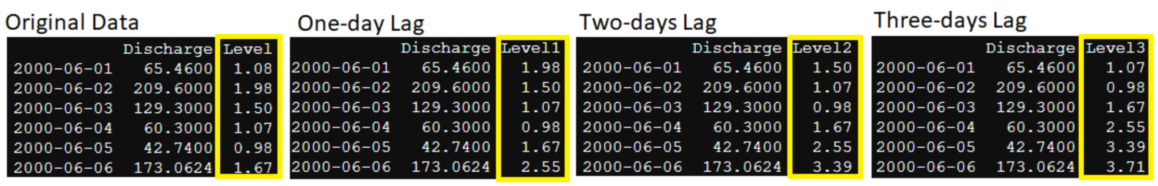
& 0.0032 for downstream level (m) corresponding to the three forecasting cases of one-day, two-days and three-days lag.

3.2 Introduction

Floods cause social and economic damage, as well as the loss of human life. We can reduce its destruction by accurate prediction of river level and discharge which are the critical components of a river. Flood prediction also benefits the planning and managing of water resource system. It turned out that precise forecasting of river level and discharge is a complex process and depends on many dynamic factors such as rainfall, riverbed terrain and other climatic conditions.

Currently, we use two approaches for hydrological forecasting. The basis of the first approach is mathematical modelling of physical dynamics between components of the hydrological system. These models need massive data and extra care for estimating parameters. The second approach is to use data-driven methods like ANN, based on the statistical relationship between the hydrological input and output variables.

In this project, ANN and CNN were used to forecast river level and discharge with one-day, two-days and three-days lag at Neeleeswaram station (downstream) using the data of rainfall, Vandiperiyar station (upstream) and Neeleeswaram station (downstream).



Original Data

```
             Discharge Level
2000-06-01     65.4600  1.08
2000-06-02    209.6000  1.98
2000-06-03    129.3000  1.50
2000-06-04     60.3000  1.07
2000-06-05     42.7400  0.98
2000-06-06    173.0624  1.67
```

One-day Lag

```
             Discharge Level1
2000-06-01     65.4600   1.98
2000-06-02    209.6000   1.50
2000-06-03    129.3000   1.07
2000-06-04     60.3000   0.98
2000-06-05     42.7400   1.67
2000-06-06    173.0624   2.55
```

Two-days Lag

```
             Discharge Level2
2000-06-01     65.4600   1.50
2000-06-02    209.6000   1.07
2000-06-03    129.3000   0.98
2000-06-04     60.3000   1.67
2000-06-05     42.7400   2.55
2000-06-06    173.0624   3.39
```

Three-days Lag

```
             Discharge Level3
2000-06-01     65.4600   1.07
2000-06-02    209.6000   0.98
2000-06-03    129.3000   1.67
2000-06-04     60.3000   2.55
2000-06-05     42.7400   3.39
2000-06-06    173.0624   3.71
```

Figure 3.1: Lag applied to data

The prediction ability of obtained models was examined using two parameters, namely Mean Squared Error (MSE) and Mean Absolute Error (MAE).

3.3 Methodology

3.3.1 Data Collection

Nineteen years of data related to Periyar river and rainfall in its region, starting from 1st June 2000 to 13th August 2019, was initially collected from Water Resources Information System (WRIS) [9]. WRIS is a public domain initiative of India-WRIS project, which was launched by the Central Water Commission (CWC) and Indian Space Research Organization (ISRO).

Following steps were followed to collect the required data from India-WRIS -

1. Rainfall: WRIS Tools -> WRIS Data online -> Applications -> Rainfall (Basin) -> WEST FLOWING RIVERS FROM TADRI TO KANYAKUMARI -> PERIYAR AND OTHERS

2. River level and discharge: WRIS Tools -> WRIS Data online -> Applications -> River point (River) -> Periyar

The data can be accessed only after registration on India-WRIS.

3.3.2 Data Exploration

3.3.2.1 Rainfall Data

Rainfall data consisted of three columns with the name of "Dates," "Normal Rainfall (in *mm*)," and "Actual Rainfall (in *mm*)." "Normal Rainfall (in *mm*)" is the amount of expected precipitation per year. The following code snippet shows the first six rows and a summary of the rainfall data.

```
> head(df)

     Dates Normal Rainfall Actual Rainfall
1 6/1/2000           13.75            9.27
2 6/2/2000           15.84           17.62
3 6/3/2000           16.17           10.01
4 6/4/2000           15.51            4.61
5 6/5/2000           16.91           10.37
6 6/6/2000           16.89           46.25

> summary(df)

    Dates              Normal Rainfall  Actual Rainfall
 Length:7153         Min.   : 0.000   Min.   :  0.000
 Class :character    1st Qu.: 1.780   1st Qu.:  0.020
 Mode  :character    Median : 6.810   Median :  1.890
                     Mean   : 7.917   Mean   :  5.953
                     3rd Qu.:12.140   3rd Qu.:  7.730
                     Max.   :25.530   Max.   :148.380
```

3.3.2.2 Vandiperiyar Station (Upstream) Data

Vandiperiyar station is an upstream station of the Periyar river. Its data set consisted of five columns named as "Dates," "Last.10.Year.Average (in m^3/s)," "Last.Year (in m^3/s)," "Discharge (in m^3/s)," and "Level (in m)."

```
> head(df1)

     Dates Last.10.Year.Average Last.Year Discharge  Level
1 6/1/2000                0.226         0     0.226 791.12
2 6/2/2000                0.433         0     0.433 791.18
3 6/3/2000                0.791         0     0.791 791.25
4 6/4/2000                0.988         0     0.988 791.28
5 6/5/2000                0.822         0     0.822 791.25
6 6/6/2000                9.564         0     9.564 791.78
> summary(df1)

    Dates             Last.10.Year.Average   Last.Year
 Length:6784        Min.   :  0.0000     Min.   :  0.000
 Class :character   1st Qu.:  0.1873     1st Qu.:  0.000
 Mode  :character   Median :  2.4875     Median :  1.125
                    Mean   :  5.3876     Mean   :  4.804
                    3rd Qu.:  8.5908     3rd Qu.:  4.911
                    Max.   :146.5286     Max.   :206.082

  Discharge              Level
 Length:6784        Min.   :  2.09
 Class :character   1st Qu.:791.05
 Mode  :character   Median :791.32
                    Mean   :773.40
                    3rd Qu.:791.63
                    Max.   :793.96
```

The above summary shows that the "Discharge" column contained data in character format as it was comprised of hyphens for 369 missing data points. Also, it included 1691 zero values, which was invalid as discharge values cannot be zero if the corresponding level values are non-zero. Therefore, I considered all the hyphens and zeros as missing values.

3.3.2.3 Neeleeswaram Station (Downstream) Data

Neeleeswaram station is a downstream station of the Periyar river. Its data set consisted of five columns, namely "Dates," "Last.10.Year.Average (in m^3/s)," "Last.Year (in m^3/s)," "Discharge (in m^3/s)," and "Level (in m)."

```
> head(df2)

     Dates Last.10.Year.Average Last.Year Discharge Level
1 6/1/2000             141.1510     362.0     65.46  1.08
2 6/2/2000             154.0550     281.4     209.6  1.98
3 6/3/2000             237.8930     265.7     129.3   1.5
4 6/4/2000             188.1360     237.3      60.3  1.07
```

```
5 6/5/2000              226.4560     187.3     42.74  0.98
6 6/6/2000              286.5602     398.0 173.06236  1.67

> summary(df2)
    Dates            Last.10.Year.Average   Last.Year
 Length:6737         Min.   :   8.472     Min.   :   0.00
 Class :character    1st Qu.:  55.794     1st Qu.:  41.69
 Mode  :character    Median : 154.007     Median : 124.10
                     Mean   : 221.720     Mean   : 212.93
                     3rd Qu.: 354.111     3rd Qu.: 284.84
                     Max.   :1138.416     Max.   :6323.55

  Discharge             Level
 Length:6737         Length:6737
 Class :character    Class :character
 Mode  :character    Mode  :character
```

The above summary concludes that both "Discharge" and "Level" columns were containing data in character format as they both had hyphens for 176 and 2 missing data points respectively. Also, there were 299 zero values in the "Discharge" column. As stated earlier, this situation is invalid because discharge values cannot be zero if the corresponding level values are non-zero. Thus, I considered all the hyphens and zeros as missing values.

3.3.3 Data Cleaning

Data exploration indicated that the raw data came up with many challenges. Hence, data cleaning was required, and it turned out to be one of the essential parts of this project.

In both of the stations' data sets, level (in m) and discharge (in m^3/s) were highly correlated. As a result, one variable could be used to predict another due to their strong correlation. Different techniques applied to clean the data of Vandiperiyar and Neeleeswaram stations made use of this strong correlation. Details regarding the complete data cleaning process are available in the following sections.

3.3.3.1 Rainfall Data

The "Normal Rainfall" was an irrelevant variable because it contained the expected rainfall calculated by taking an average of at least the last 30 years of data. It may not contribute well to the accurate prediction of both level and discharge and hence removed from the dataset. Also, the dates were initially in character format but later converted to "ymd (year/month/date)" format. Zeros are valid in rainfall data. Figure 3.2 displays cleaned rainfall data.

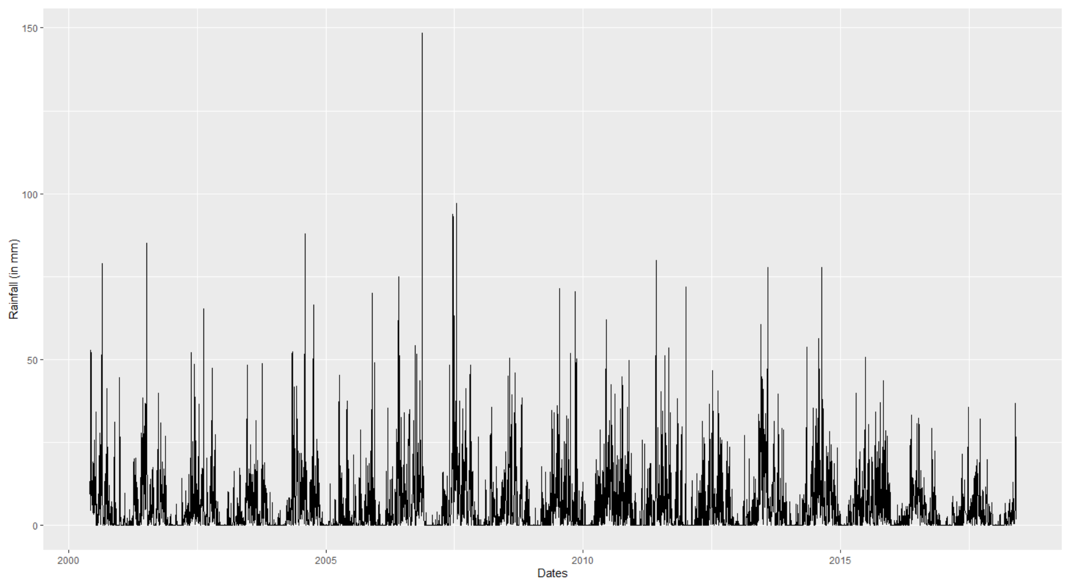


Figure 3.2: Rainfall Data (2000 - 2018)

3.3.3.2 Vandiperiyar Station (Upstream) Data

Following steps were performed to clean the Vandiperiyar station data -

1. Remove "Last.Year" and "Last.10.Year.Average" variables from the data set due to their irrelevance to the prediction.
2. Create a copy of the data set with the name "Temp." In "Temp" eliminate all rows with missing values as per section 3.3.2.2.
3. Remove all rows corresponding to the outliers of the "Level" column in "Temp" data set.
4. Create a scatter-plot of "Discharge" column values against "Level" column values of "Temp" data set. Fit a polynomial of degree three over the scatter-plot. Use the fitted model to predict missing "Discharge" column values by inputting corresponding "Level" column values from the original data set.
5. Remove the "Temp" data set.
6. Replace all missing "Discharge" column values in the original data set with the values predicted using the generated polynomial model.

Figures 3.3 and 3.4 display the cleaned data of Vandiperiyar station obtained from the steps mentioned above.

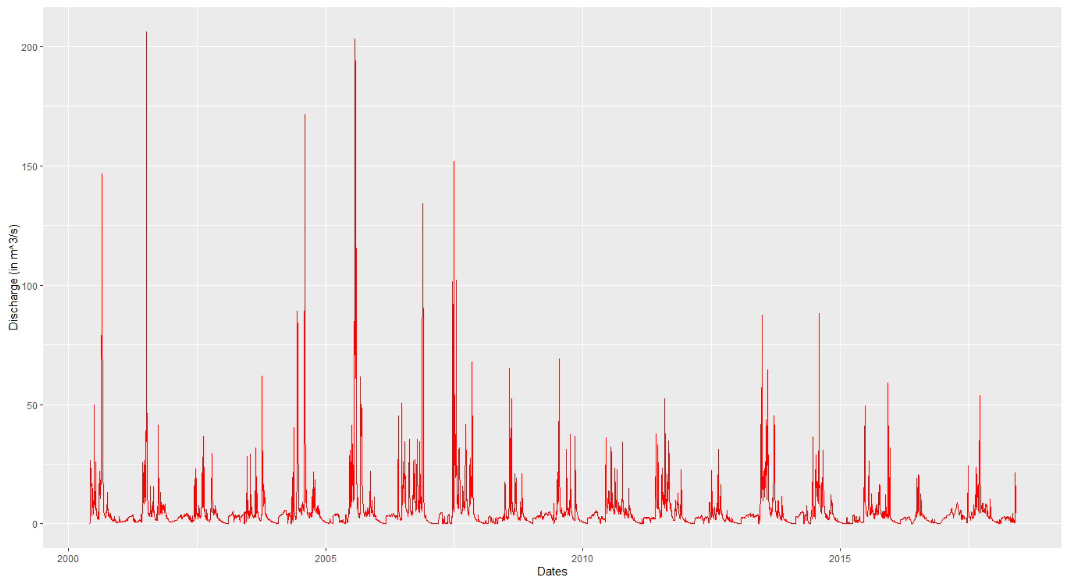


Figure 3.3: Vandiperiyar station discharge (2000 - 2018)

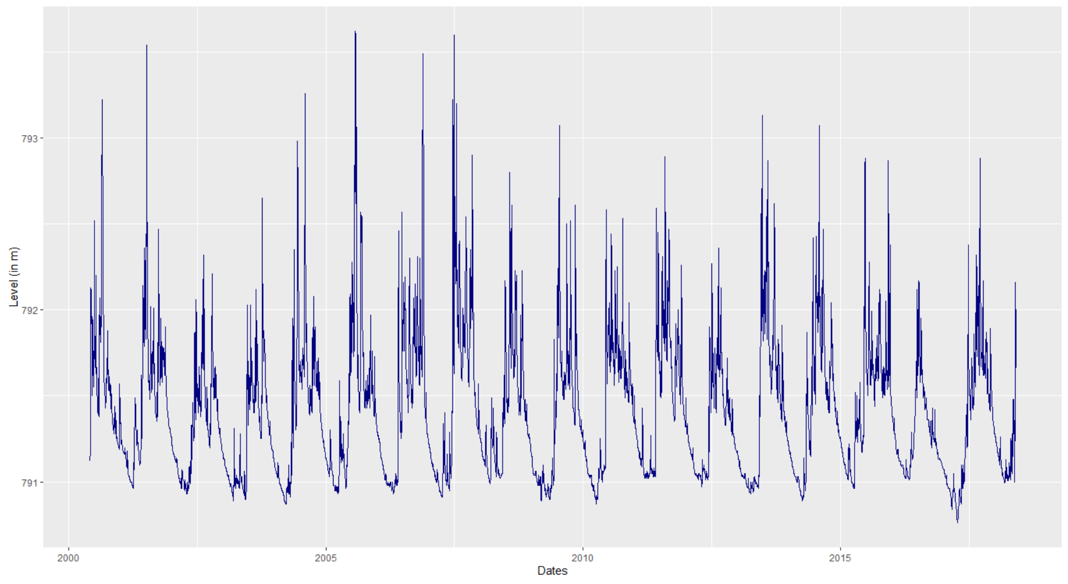


Figure 3.4: Vandiperiyar station level (2000 - 2018)

3.3.3.3 Neeleeswaram Station (Downstream) Data

Following steps were performed to clean Neeleeswaram station data -

1. Remove the two variables "Last.Year" and "Last.10.Year.Average", due to their irrelevance to the prediction.

2. If the data type of any remaining column in the data set is "factor" instead of "numeric," then make use of "unfactor" function of "varhandle" package for

data type conversion.

3. Create a copy of the data set with the name "Temp." In "Temp" eliminate all rows with missing values as per section 3.3.2.3.

4. Predict missing "Discharge" column values from "Level" column values by fitting an exponential curve using "nlsfit" function of "easynls" package over the "Temp" data set by keeping "Level" as the independent variable and "Discharge" as the dependent variable.

5. Create a scatter-plot of "Level" column values against "Discharge" column values of "Temp" data set. Fit a polynomial of degree five over the scatter-plot.

6. Use the previously fitted exponential model to predict missing "Discharge" column values by inputting "Level" column values from the original data set.

7. Use the previously fitted polynomial model to predict missing "Level" column values by inputting "Discharge" column values from the original data set.

Figures 3.5 and 3.6 display the cleaned data of Neeleeswaram station obtained from the steps mentioned above.

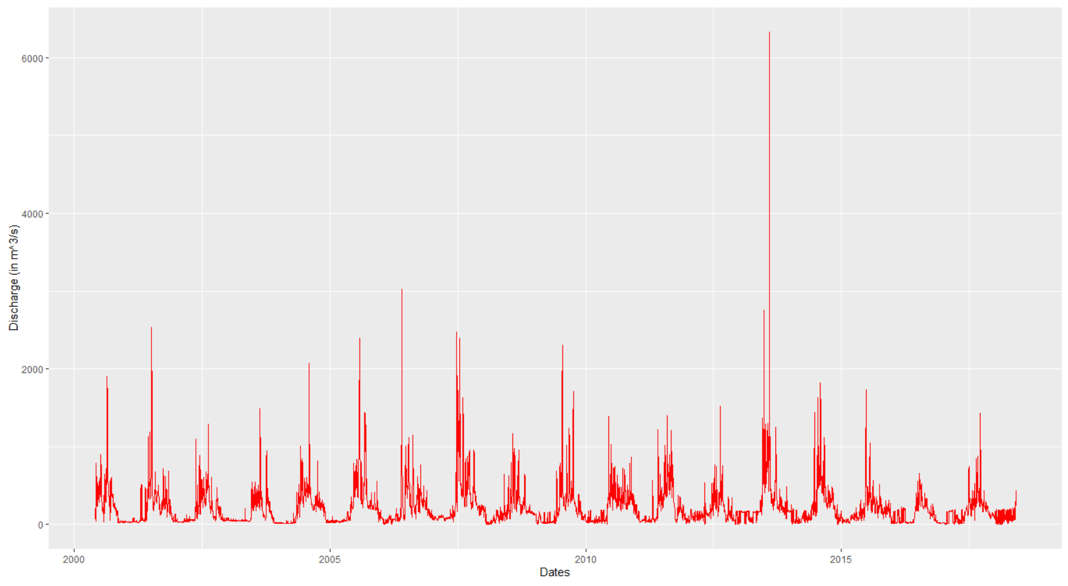


Figure 3.5: Neeleeswaram station discharge (2000 - 2018)

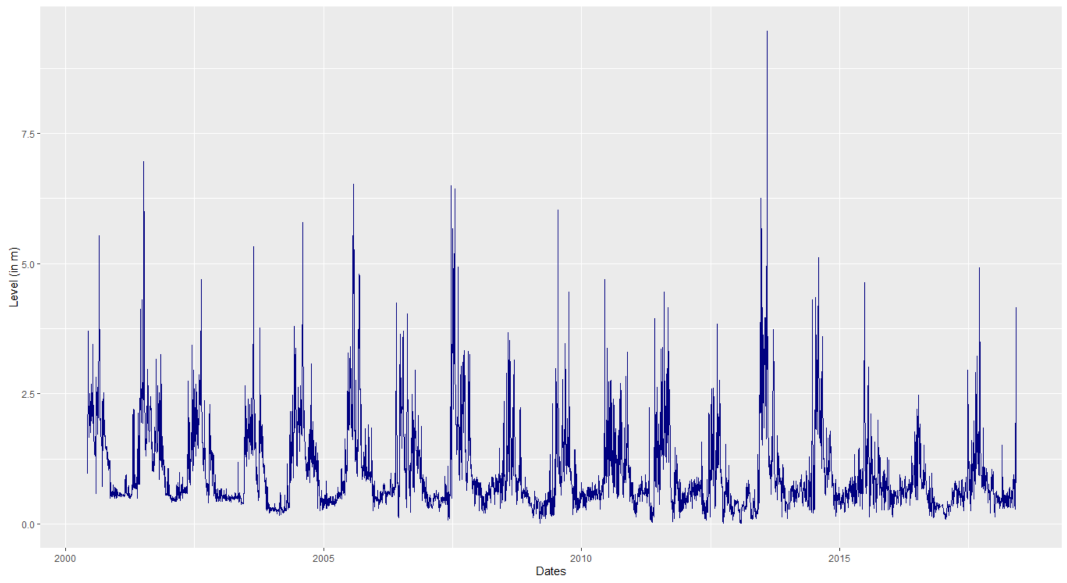


Figure 3.6: Neeleeswaram station level (2000 - 2018)

The cleaned data set contained values ranging from "2000-Jun-01" to "2018-Jun-01" as some data points were removed in the cleaning process. Also, the data was continuous only until "2016-May-31."

3.3.4 Data Analysis

3.3.4.1 Rating Curves

Rating curves are graphs depicting the relationship between discharge and level of a river.

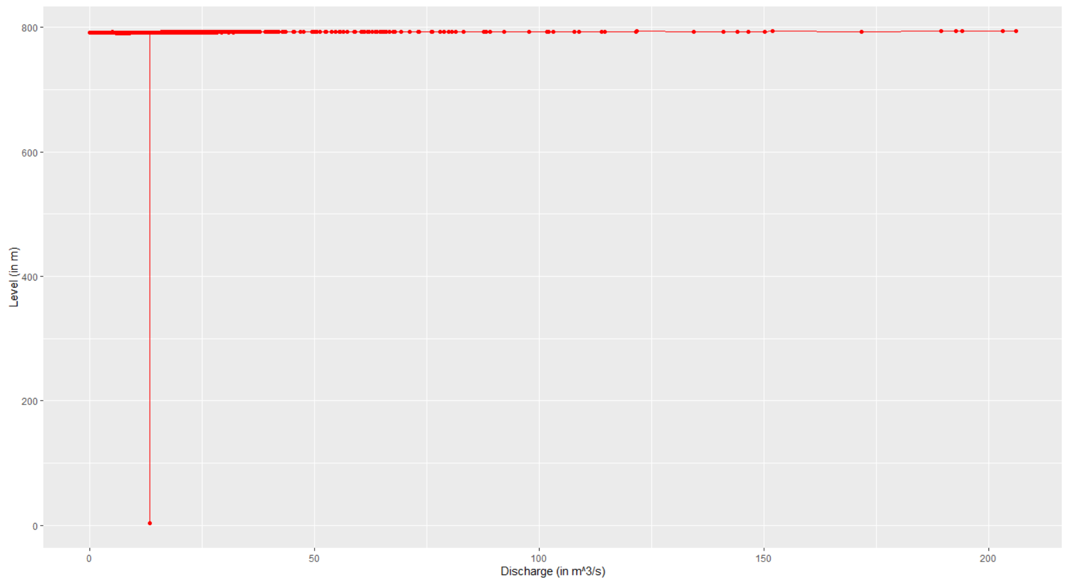


Figure 3.7: Rating curve - Vandiperiyar station

An outlier can be observed in the graph. Hence, it is plotted again without the outlier to visualize the remaining data better.

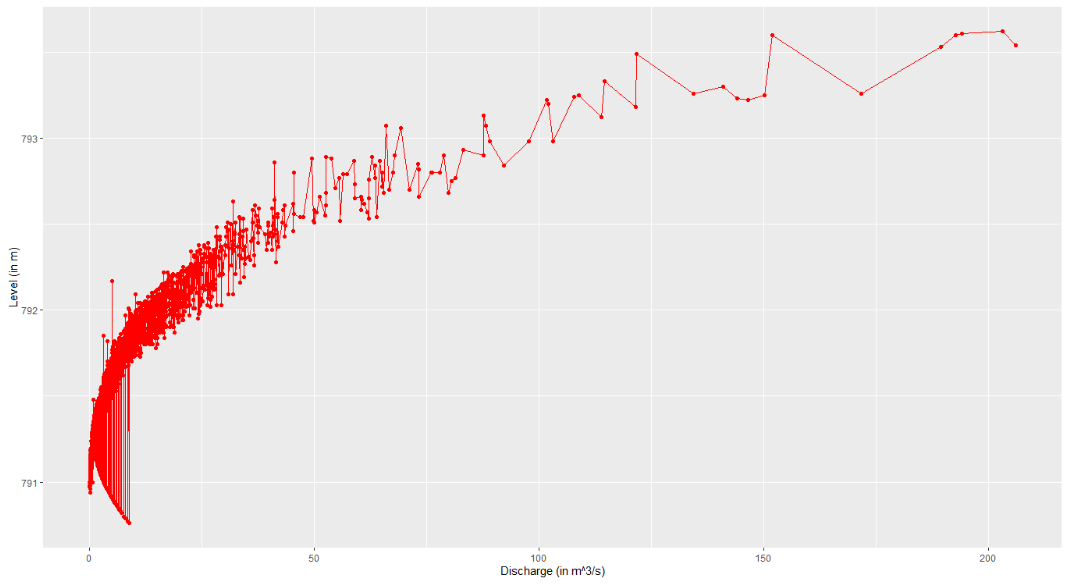


Figure 3.8: Rating curve - Vandiperiyar station

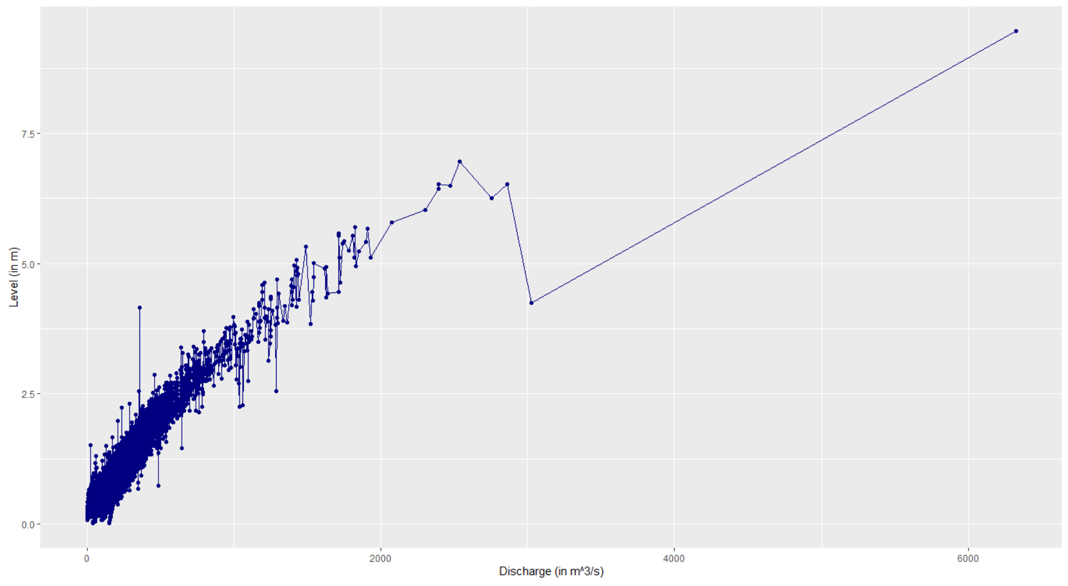


Figure 3.9: Rating curve - Neeleeswaram station

A direct relationship between the "Level" and "Discharge" variables can be observed from figures 3.8 and 3.9.

3.3.4.2 Artificial Neural Network

Artificial Neural Network (ANN) is a data-driven model inspired by the architecture of the biological brain. ANN was used in this project because of its capability to recognize complex nonlinear relationships between the data inputs and outputs [10]. The relation among the inputs and outputs is illustrated as follows -

$$Y = f(X^n) \tag{3.1}$$

where X^n is an n-dimensional input vector consisting of variables $X_1, X_2, ..., X_n$. Network parameters represent the function form of $f(.)$ as shown in equation 3.1.

ANN commonly consists of 3 layers, namely an input layer, a hidden layer, and an output layer. The complexity of ANN is closely related to the hidden layers. The size of the input data decides the number of hidden layers. In general, there is no specific way of finalizing an appropriate amount of hidden layers. One commonly used method to choose it is by training the model multiple times with a varying number of hidden layers and select the case where errors are least.

A stochastic gradient descent optimization algorithm is utilized for training of an ANN, and its weights get updated via a backpropagation algorithm which depends on the errors calculated from the loss function. The loss function gives the difference between actual output and the predicted output [11]. These errors are then propagated to the previous layer by calculating gradients of the loss function.

A neural network can be applied in R using "layer_dense" function of "keras" package. It requires the following essential parameters -

1. units - A positive integer which denotes the dimensionality of the output space
2. activation - Type of activation function to choose with default set to "linear"

Data from "2000-Jun-01" to "2016-May-31" was selected for training. The rest was for testing.

Model structure -

For predicting river discharge (m^3/s) -

1. Input parameters - Rainfall (mm), discharge at Vandiperiyar station (m^3/s) and discharge at Neeleeswaram station (m^3/s)
2. Output parameters - Discharge at Neeleeswaram station (m^3/s) with one-day, two-days and three-days lag

For predicting river level (m) -

1. Input parameters - Rainfall (mm), level at Vandiperiyar station (m) and level at Neeleeswaram station (m)
2. Output parameters - Level at Neeleeswaram station (m) with one-day, two-days and three-days lag

Training parameters -

1. Number of dense layers = 3
2. Units used = (64, 32, 1)
3. Activation function = relu
4. Dropout rate = 0.2
5. Optimizer = adam
6. Epochs = 100
7. Batch size = 10

```r
library(keras)         # For ANN
library(DataCombine)   # For "slide" function
library(lubridate)     # For changing dates to YMD format
library(ggplot2)       # For plotting

################################## ANN ####################################

# Reading cleaned data
df <- read.csv("Rainfall_clean.csv")
```

```r
df1 <- read.csv("Vandiperiyar_clean.csv")
df2 <- read.csv("Neeleswaram_clean.csv")

# Changing column names
colnames(df)[2] <- "Rainfall"
colnames(df1)[2] <- "Discharge"; colnames(df1)[3] <- "Level"
colnames(df2)[2] <- "Discharge"; colnames(df2)[3] <- "Level"

# Making dates same in all three data sets
dates <- setdiff(df1$Dates, df2$Dates)
dates <- c(dates, setdiff(df2$Dates, df1$Dates))
dates <- unique(dates)
rej <- match(dates, df1$Dates); rej <- sort(rej)
df1 <- df1[-rej, ]
rej <- match(dates, df2$Dates); rej <- sort(rej)
df2 <- df2[-rej, ]
dates <- setdiff(df$Dates, df1$Dates)
rej <- match(dates, df$Dates); rej <- sort(rej)
df <- df[-rej, ]

# Converting dates into YMD format
df$Dates <- ymd(df$Dates)
df1$Dates <- ymd(df1$Dates)
df2$Dates <- ymd(df2$Dates)

# Creating master data frame
temp <- df$Dates
data <- data.frame(df$Rainfall, df1$Discharge, df1$Level, df2$Discharge, df2$Level)
data <- data[-c(2191, 4814, 6186, 6184), ] # Removing outliers
temp <- temp[-c(2191, 4814, 6186, 6184)]

# Creating model
model <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = "linear")

# Compiling model
model %>% compile(
  optimizer = "adam",
  loss = "mse",
  metric = "mae")

# Summary
model %>% summary()

################## One-day lag prediction ###################

data1 <- data

# "slide" function can be used to take leads
data1 <- slide(data1, Var = "df2.Discharge", slideBy = 1)
data1 <- slide(data1, Var = "df2.Level", slideBy = 1)
data1 <- na.omit(data1)

# 1:5842 contains data from 1st June, 2000 to 31st May, 2016
index <- 1:5842
datatrain = data1[index, ]
datatest = data1[-index, ]

# Scaling data between 0 and 1
max = apply(data1 , 2 , max)
min = apply(data1, 2 , min)
scaled = as.data.frame(scale(data1, center = min, scale = max - min))

# Splitting data for training and testing
train = scaled[index , ]
test = scaled[-index , ]
```

```r

#------------------------- DISCHARGE ----------------------#

# Converting data frame into matrix
xtrain <- as.matrix(train[, c(1, 2, 4)])
ytrain <- as.matrix(train[, c(6)])
xtest = as.matrix(test[, c(1, 2, 4)])
ytest = as.matrix(test[, c(6)])

# Fitting model
history1 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.20)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

# MAE and MSE (loss)
print(scores)

# Predictions
predict_Discharge1 = model %>% predict(xtest)

# Un-scaling for converting back to the original range
predict_Discharge1 = (predict_Discharge1 * (max(data1$df2.Discharge1) - min(data1$
    df2.Discharge1))) + min(data1$df2.Discharge1)

# Plotting actual observations against predictions
plot(datatest$df2.Discharge1, predict_Discharge1, col='blue', pch=16, ylab = "
    Predicted Discharge (in m^3/s)", xlab = "Actual Discharge (in m^3/s)")
title("Predicted discharge v/s Actual discharge: One-day lag")
abline(0, 1, col = "black")

# Plotting "history1"
plot(history1)

#-------------------------- LEVEL -----------------------------------#

xtrain <- as.matrix(train[, c(1, 3, 5)])
xtest = as.matrix(test[, c(1, 3, 5)])
ytrain <- as.matrix(train[c(7)])
ytest = as.matrix(test[c(7)])

history2 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

print(scores)

plot(history2)

predict_Level1 = model %>% predict(xtest)
predict_Level1 = (predict_Level1 * (max(data1$df2.Level1) - min(data1$df2.Level1)))
     + min(data1$df2.Level1)

plot(datatest$df2.Level1, predict_Level1, col='blue', pch=16, ylab = "Predicted
    Level (in m)", xlab = "Actual Level (in m)")
title("Predicted level v/s Actual level: One-day lag")
abline(0, 1, col = "black")


# Visualizing actual observations against predictions for test data
dt <- data.frame(temp[5843:6395], datatest$df2.Discharge1, datatest$df2.Level1,
    predict_Discharge1, predict_Level1)

colnames(dt)[1] <- "Dates"
colnames(dt)[2] <- "Discharge1"
colnames(dt)[3] <- "Level1"

colors <- c("Actual discharge" = "red", "Predicted discharge" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Discharge1, color = "Actual discharge")) +
  geom_line(aes(y = predict_Discharge1, color = "Predicted discharge")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Discharge (in m^3/s)", color = "Legend") +
  ggtitle("Discharge: One-day lag")

colors <- c("Actual level" = "red", "Predicted level" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Level1, color = "Actual level")) +
  geom_line(aes(y = predict_Level1, color = "Predicted level")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Level (in m)", color = "Legend") +
  ggtitle("Level: One-day lag")


################## Two-days lag prediction ##################

data2 <- data
data2 <- slide(data2, Var = "df2.Discharge", slideBy = 2)
data2 <- slide(data2, Var = "df2.Level", slideBy = 2)
data2 <- na.omit(data2)


# 1:5842 contains data from 1st June, 2000 to 31st May, 2016

index <- 1:5842
datatrain = data2[index, ]
datatest = data2[-index, ]

max = apply(data2 , 2 , max)
min = apply(data2, 2 , min)
scaled = as.data.frame(scale(data2, center = min, scale = max - min))

train = scaled[index , ]
test = scaled[-index , ]

#---------------------------- DISCHARGE ---------------------------#

xtrain <- as.matrix(train[, c(1, 2, 4)])
ytrain <- as.matrix(train[, 6])
xtest = as.matrix(test[, c(1, 2, 4)])
ytest = as.matrix(test[, 6])

history3 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

print(scores)

plot(history3)

ypred = model %>% predict(xtest)
predict_Discharge2 = ypred
predict_Discharge2 = (predict_Discharge2 * (max(data2$df2.Discharge2) - min(data2$
    df2.Discharge2))) + min(data2$df2.Discharge2)

plot(datatest$df2.Discharge2, predict_Discharge2, col='blue', pch=16, ylab = "
    Predicted Discharge (in m^3/s)", xlab = "Actual Discharge (in m^3/s)")
title("Predicted discharge v/s Actual discharge: Two-days lag")
abline(0, 1, col = "black")

#-------------------------- LEVEL -----------------------------------#

xtrain <- as.matrix(train[, c(1, 3, 5)])
xtest = as.matrix(test[, c(1, 3, 5)])
ytrain <- as.matrix(train[, c(7)])
ytest = as.matrix(test[, 7])


history4 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtest, ytest, verbose = 0)
```

```r

print(scores)

plot(history4)

ypred = model %>% predict(xtest)
predict_Level2 = ypred
predict_Level2 = (predict_Level2 * (max(data2$df2.Level2) - min(data2$df2.Level2)))
     + min(data2$df2.Level2)

plot(datatest$df2.Level2, predict_Level2, col='blue', pch=16, ylab = "Predicted
    Level (in m)", xlab = "Actual Level (in m)")
title("Predicted level v/s Actual level: Two-days lag")
abline(0, 1, col = "black")


dt <- data.frame(temp[5843:6394], datatest$df2.Discharge2, datatest$df2.Level2,
    predict_Discharge2, predict_Level2)

colnames(dt)[1] <- "Dates"
colnames(dt)[2] <- "Discharge2"
colnames(dt)[3] <- "Level2"

colors <- c("Actual discharge" = "red", "Predicted discharge" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Discharge2, color = "Actual discharge")) +
  geom_line(aes(y = predict_Discharge2, color = "Predicted discharge")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Discharge (in m^3/s)", color = "Legend") +
  ggtitle("Discharge: Two-days lag")

colors <- c("Actual level" = "red", "Predicted level" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Level2, color = "Actual level")) +
  geom_line(aes(y = predict_Level2, color = "Predicted level")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Level (in m)", color = "Legend") +
  ggtitle("Level: Two-days lag")


################## Three-days lag prediction ###################

data3 <- data
data3 <- slide(data3, Var = "df2.Discharge", slideBy = 3)
data3 <- slide(data3, Var = "df2.Level", slideBy = 3)
data3 <- na.omit(data3)

index <- 1:5842
datatrain = data3[index, ]
datatest = data3[-index, ]

max = apply(data3 , 2 , max)
min = apply(data3, 2 , min)
scaled = as.data.frame(scale(data3, center = min, scale = max - min))

train = scaled[index , ]
test = scaled[-index , ]

#---------------------------- DISCHARGE ---------------------------#

xtrain <- as.matrix(train[, c(1, 2, 4)])
ytrain <- as.matrix(train[, c(6)])
xtest = as.matrix(test[, c(1, 2, 4)])
ytest = as.matrix(test[, 6])

histroy5 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

print(scores)
```

```r

plot(histroy5)

ypred = model %>% predict(xtest)
predict_Discharge3 = ypred
predict_Discharge3 = (predict_Discharge3 * (max(data3$df2.Discharge3) - min(data3$
    df2.Discharge3))) + min(data3$df2.Discharge3)

plot(datatest$df2.Discharge3, predict_Discharge3, col='blue', pch=16, ylab = "
    Predicted Discharge (in m^3/s)", xlab = "Actual Discharge (in m^3/s)")
title("Predicted discharge v/s Actual discharge: Three-days lag")
abline(0, 1, col = "black")


#-------------------------- LEVEL -----------------------------------#

xtrain <- as.matrix(train[, c(1, 3, 5)])
xtest = as.matrix(test[, c(1, 3, 5)])
ytrain <- as.matrix(train[, c(7)])
ytest = as.matrix(test[, 7])

history6 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

print(scores)

plot(history6)

ypred = model %>% predict(xtest)
predict_Level3 = ypred
predict_Level3 = (predict_Level3 * (max(data3$df2.Level3) - min(data3$df2.Level3)))
     + min(data3$df2.Level3)

plot(datatest$df2.Level3, predict_Level3, col='blue', pch=16, ylab = "Predicted
    Level (in m)", xlab = "Actual Level (in m)")
title("Predcited level v/s Actual level: Three-days lag")
abline(0, 1, col = "black")

dt <- data.frame(temp[5843:6393], datatest$df2.Discharge3, datatest$df2.Level3,
    predict_Discharge3, predict_Level3)

colnames(dt)[1] <- "Dates"
colnames(dt)[2] <- "Discharge3"
colnames(dt)[3] <- "Level3"

colors <- c("Actual discharge" = "red", "Predicted discharge" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Discharge3, color = "Actual discharge")) +
  geom_line(aes(y = predict_Discharge3, color = "Predicted discharge")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Discharge (in m^3/s)", color = "Legend") +
  ggtitle("Discharge: Three-days lag")

colors <- c("Actual level" = "red", "Predicted level" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Level3, color = "Actual level")) +
  geom_line(aes(y = predict_Level3, color = "Predicted level")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Level (in m)", color = "Legend") +
  ggtitle("Level: Three-days lag")

plot(history6)
```

Code Snippet 3.1: Predicting river level and discharge for one-day, two-days and three-days lag using ANN

3.3.4.3 Convolutional Neural Network

Convolutional Neural Network or CNN comprises of multiple layers. The first one is the convolution layer which extracts high-level features. Next is the pooling layer, which reduces the spatial size of the convolved element with the aim of dimensionality reduction. The convolution and pooling layers are followed by a dense, fully connected layer used for interpreting the extracted features. A flattened layer is used between the dense layer and convolution layer for reducing feature maps to a one-dimensional vector [12].

A one-dimensional or 1D CNN is a CNN model that operates over a one-dimensional sequence. 1D CNN can offer a fast alternative to RNN for time series forecasting [13]. It can be implemented using "layer_conv_1d" function of the "keras" package in R. It inputs three-dimensional or 3D (samples, time, features) tensors and returns similar 3D tensors. Data from "2000-Jun-01" to "2016-May-31" was selected for training. The rest was for testing.

Model structure -

For predicting river discharge (m^3/s) -

1. Input parameters - Rainfall (mm), discharge at Vandiperiyar station (m^3/s) and discharge at Neeleeswaram station (m^3/s)
2. Output parameters - Discharge at Neeleeswaram station (m^3/s) with one-day, two-days and three-days lag

For predicting river level (m) -

1. Input parameters - Rainfall (mm), level at Vandiperiyar station (m) and level at Neeleeswaram station (m)
2. Output parameters - Level at Neeleeswaram station (m) with one-day, two-days and three-days lag

Training parameters -

1. Number of 1D convolutional layers = 1
2. Number of filters = 64
3. Kernel size = 2
4. Input shape = (3, 1)
5. Number of dense layers = 3
6. Units used = (64, 32, 1)
7. Activation function = relu

8. Dropout rate = 0.2
9. Optimizer = adam
10. Pool size = 2
11. Epochs = 100
12. Batch size = 10

```r
library(keras)        # For CNN
library(DataCombine)  # For "slide" function
library(lubridate)    # For changing dates to YMD format
library(ggplot2)      # For plotting

################################## CNN ####################################

# Reading cleaned data
df <- read.csv("Rainfall_clean.csv")
df1 <- read.csv("Vandiperiyar_clean.csv")
df2 <- read.csv("Neeleswaram_clean.csv")

# Changing column names
colnames(df)[2] <- "Actual"
colnames(df1)[2] <- "Discharge"; colnames(df1)[3] <- "Level"
colnames(df2)[2] <- "Discharge"; colnames(df2)[3] <- "Level"

# Making dates same in all three data sets
dates <- setdiff(df1$Dates, df2$Dates)
dates <- c(dates, setdiff(df2$Dates, df1$Dates))
dates <- unique(dates)
rej <- match(dates, df1$Dates); rej <- sort(rej)
df1 <- df1[-rej, ]
rej <- match(dates, df2$Dates); rej <- sort(rej)
df2 <- df2[-rej, ]
dates <- setdiff(df$Dates, df1$Dates)
rej <- match(dates, df$Dates); rej <- sort(rej)
df <- df[-rej, ]

# Converting dates into YMD format
df$Dates <- ymd(df$Dates)
df1$Dates <- ymd(df1$Dates)
df2$Dates <- ymd(df2$Dates)

# Creating master data frame
temp <- df$Dates
data <- data.frame(df$Actual, df1$Discharge, df1$Level, df2$Discharge, df2$Level)
data <- data[-c(2191, 4814, 6186, 6184), ] # Removing outliers
temp <- temp[-c(2191, 4814, 6186, 6184)]

# Creating model
model <- keras_model_sequential() %>%
  layer_conv_1d(filters = 64, kernel_size = 2, input_shape = c(3, 1), activation =
      "relu") %>%
  layer_max_pooling_1d(pool_size = 2) %>%
  layer_flatten() %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = "linear")

# Compiling model
model %>% compile(
  optimizer = "adam",
  loss = "mse",
  metric = "mae")

# Summary
model %>% summary()

################## One-day lag prediction ###################

data1 <- data

# "slide" function can be used to take leads
data1 <- slide(data1, Var = "df2.Discharge", slideBy = 1)
data1 <- slide(data1, Var = "df2.Level", slideBy = 1)
data1 <- na.omit(data1)


# 1:5844 contains data from 1st June, 2000 to 31st May, 2016
index <- 1:5842
datatrain = data1[index, ]
datatest = data1[-index, ]

# Scaling data between 0 and 1
max = apply(data1 , 2 , max)
min = apply(data1, 2 , min)
scaled = as.data.frame(scale(data1, center = min, scale = max - min))

# Splitting data for training and testing
train = scaled[index , ]
test = scaled[-index , ]

#------------------------- DISCHARGE -----------------------#

# Converting data frame into matrix
xtrain <- as.matrix(train[, c(1, 2, 4)])
ytrain <- as.matrix(train[, c(6)])
xtest = as.matrix(test[, c(1, 2, 4)])
ytest = as.matrix(test[, c(6)])

# Transforming 2D matrix into 3D matrix
xtrain = array(xtrain, dim = c(nrow(xtrain), 3, 1))
xtest = array(xtest, dim = c(nrow(xtest), 3, 1))

# Fitting model
history1 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.20)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

# MAE and MSE (loss)
print(scores)

# Plotting "history1"
plot(history1)

# Predictions
predict_Discharge1 = model %>% predict(xtest)

# Un-scaling for converting back to the original range
predict_Discharge1 = (predict_Discharge1 * (max(data1$df2.Discharge1) - min(data1$
    df2.Discharge1))) + min(data1$df2.Discharge1)

# Plotting actual observations against predictions
plot(datatest$df2.Discharge1, predict_Discharge1, col='blue', pch=16, ylab = "
    Predicted Discharge (in m^3/s)", xlab = "Actual Discharge (in m^3/s)")
title("Predicted discharge v/s Actual discharge: One-day lag")
abline(0, 1, col = "black")


#------------------------- LEVEL ----------------------------------#

xtrain <- as.matrix(train[, c(1, 3, 5)])
xtest = as.matrix(test[, c(1, 3, 5)])
```

```r
ytrain <- as.matrix(train[c(7)])
ytest = as.matrix(test[c(7)])

xtrain = array(xtrain, dim = c(nrow(xtrain), 3, 1))
xtest = array(xtest, dim = c(nrow(xtest), 3, 1))

history2 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtest, ytest, verbose = 0)

print(scores)

plot(history2)

predict_Level1 = model %>% predict(xtest)
predict_Level1 = (predict_Level1 * (max(data1$df2.Level1) - min(data1$df2.Level1)))
     + min(data1$df2.Level1)

plot(datatest$df2.Level1, predict_Level1, col='blue', pch=16, ylab = "Predicted
    Level (in m)", xlab = "Actual Level (in m)")
title("Predicted level v/s Actual level: One-day lag")
abline(0, 1, col = "black")

# Visualizing actual observations against predictions for test data
dt <- data.frame(temp[5843:6395], datatest$df2.Discharge1, datatest$df2.Level1,
    predict_Discharge1, predict_Level1)

colnames(dt)[1] <- "Dates"
colnames(dt)[2] <- "Discharge1"
colnames(dt)[3] <- "Level1"

colors <- c("Actual discharge" = "red", "Predicted discharge" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Discharge1, color = "Actual discharge")) +
  geom_line(aes(y = predict_Discharge1, color = "Predicted discharge")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Discharge (in m^3/s)", color = "Legend") +
  ggtitle("Discharge: One-day lag")

colors <- c("Actual level" = "red", "Predicted level" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Level1, color = "Actual level")) +
  geom_line(aes(y = predict_Level1, color = "Predicted level")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Level (in m)", color = "Legend") +
  ggtitle("Level: One-day lag")


################## Two-days lag prediction ##################

data2 <- data
data2 <- slide(data2, Var = "df2.Discharge", slideBy = 2)
data2 <- slide(data2, Var = "df2.Level", slideBy = 2)
data2 <- na.omit(data2)

index <- 1:5842
datatrain = data2[index, ]
datatest = data2[-index, ]

max = apply(data2 , 2 , max)
min = apply(data2, 2 , min)
scaled = as.data.frame(scale(data2, center = min, scale = max - min))

train = scaled[index , ]
test = scaled[-index , ]

#---------------------------- DISCHARGE ---------------------------#

xtrain <- as.matrix(train[, c(1, 2, 4)])
ytrain <- as.matrix(train[, 6])
```

```r
xtest = as.matrix(test[, c(1, 2, 4)])
ytest = as.matrix(test[, 6])

xtrain = array(xtrain, dim = c(nrow(xtrain), 3, 1))
xtest = array(xtest, dim = c(nrow(xtest), 3, 1))

history3 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtrain, ytrain, verbose = 0)

print(scores)

plot(history3)

ypred = model %>% predict(xtest)
predict_Discharge2 = ypred
predict_Discharge2 = (predict_Discharge2 * (max(data2$df2.Discharge2) - min(data2$
    df2.Discharge2))) + min(data2$df2.Discharge2)

plot(datatest$df2.Discharge2, predict_Discharge2, col='blue', pch=16, ylab = "
    Predicted Discharge (in m^3/s)", xlab = "Actual Discharge (in m^3/s)")
title("Predicted discharge v/s Actual discharge: Two-days lag")
abline(0, 1, col = "black")

#-------------------------- LEVEL -----------------------------------#

xtrain <- as.matrix(train[, c(1, 3, 5)])
xtest = as.matrix(test[, c(1, 3, 5)])
ytrain <- as.matrix(train[, c(7)])
ytest = as.matrix(test[, 7])

xtrain = array(xtrain, dim = c(nrow(xtrain), 3, 1))
xtest = array(xtest, dim = c(nrow(xtest), 3, 1))

history4 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtrain, ytrain, verbose = 0)

print(scores)

plot(history4)

ypred = model %>% predict(xtest)
predict_Level2 = ypred
predict_Level2 = (predict_Level2 * (max(data2$df2.Level2) - min(data2$df2.Level2)))
     + min(data2$df2.Level2)

plot(datatest$df2.Level2, predict_Level2, col='blue', pch=16, ylab = "Predicted
    Level (in m)", xlab = "Actual Level (in m)")
title("Predicted level v/s Actual level: Two-days lag")
abline(0, 1, col = "black")

dt <- data.frame(temp[5843:6394], datatest$df2.Discharge2, datatest$df2.Level2,
    predict_Discharge2, predict_Level2)

colnames(dt)[1] <- "Dates"
colnames(dt)[2] <- "Discharge2"
colnames(dt)[3] <- "Level2"

colors <- c("Actual discharge" = "red", "Predicted discharge" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Discharge2, color = "Actual discharge")) +
  geom_line(aes(y = predict_Discharge2, color = "Predicted discharge")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Discharge (in m^3/s)", color = "Legend") +
  ggtitle("Discharge: Two-days lag")

colors <- c("Actual level" = "red", "Predicted level" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Level2, color = "Actual level")) +
  geom_line(aes(y = predict_Level2, color = "Predicted level")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Level (in m)", color = "Legend") +
  ggtitle("Level: Two-days lag")


################## Three-days lag prediction ###################

data3 <- data
data3 <- slide(data3, Var = "df2.Discharge", slideBy = 3)
data3 <- slide(data3, Var = "df2.Level", slideBy = 3)
data3 <- na.omit(data3)

index <- 1:5842
datatrain = data3[index, ]
datatest = data3[-index, ]

max = apply(data3 , 2 , max)
min = apply(data3, 2 , min)
scaled = as.data.frame(scale(data3, center = min, scale = max - min))

train = scaled[index , ]
test = scaled[-index , ]

#---------------------------- DISCHARGE ---------------------------#

xtrain <- as.matrix(train[, c(1, 2, 4)])
ytrain <- as.matrix(train[, c(6)])
xtest = as.matrix(test[, c(1, 2, 4)])
ytest = as.matrix(test[, 6])

xtrain = array(xtrain, dim = c(nrow(xtrain), 3, 1))
xtest = array(xtest, dim = c(nrow(xtest), 3, 1))

history5 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtrain, ytrain, verbose = 0)

print(scores)

plot(history5)

ypred = model %>% predict(xtest)
predict_Discharge3 = ypred
predict_Discharge3 = (predict_Discharge3 * (max(data3$df2.Discharge3) - min(data3$
    df2.Discharge3))) + min(data3$df2.Discharge3)

plot(datatest$df2.Discharge3, predict_Discharge3, col='blue', pch=16, ylab = "
    Predicted Discharge (in m^3/s)", xlab = "Actual Discharge (in m^3/s)")
title("Predicted discharge v/s Actual discharge: Three-days lag")
abline(0, 1, col = "black")

#-------------------------- LEVEL -----------------------------------#

xtrain <- as.matrix(train[, c(1, 3, 5)])
xtest = as.matrix(test[, c(1, 3, 5)])
ytrain <- as.matrix(train[, c(7)])
ytest = as.matrix(test[, 7])

xtrain = array(xtrain, dim = c(nrow(xtrain), 3, 1))
xtest = array(xtest, dim = c(nrow(xtest), 3, 1))

history6 <- model %>% fit(xtrain, ytrain, epochs = 100, batch_size = 10, verbose =
    1, validation_split = 0.2)
scores = model %>% evaluate(xtrain, ytrain, verbose = 0)

print(scores)

plot(history6)

```

```r
ypred = model %>% predict(xtest)
predict_Level3 = ypred
predict_Level3 = (predict_Level3 * (max(data3$df2.Level3) - min(data3$df2.Level3)))
     + min(data3$df2.Level3)

plot(datatest$df2.Level3, predict_Level3, col='blue', pch=16, ylab = "Predicted
    Level (in m)", xlab = "Actual Level (in m)")
title("Predcited level v/s Actual level: Three-days lag")
abline(0, 1, col = "black")

dt <- data.frame(temp[5843:6393], datatest$df2.Discharge3, datatest$df2.Level3,
    predict_Discharge3, predict_Level3)

colnames(dt)[1] <- "Dates"
colnames(dt)[2] <- "Discharge3"
colnames(dt)[3] <- "Level3"

colors <- c("Actual discharge" = "red", "Predicted discharge" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Discharge3, color = "Actual discharge")) +
  geom_line(aes(y = predict_Discharge3, color = "Predicted discharge")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Discharge (in m^3/s)", color = "Legend") +
  ggtitle("Discharge: Three-days lag")

colors <- c("Actual level" = "red", "Predicted level" = "navy")
ggplot(dt, aes(x = Dates)) +
  geom_line(aes(y = Level3, color = "Actual level")) +
  geom_line(aes(y = predict_Level3, color = "Predicted level")) +
  scale_color_manual(values = colors) +
  labs(x = "Dates", y = "Level (in m)", color = "Legend") +
  ggtitle("Level: Three-days lag")
```

Code Snippet 3.2: Predicting river level and discharge for one-day, two-days and three-days lag using 1D CNN

3.3.5 Results

Model performance evaluation was based on two parameters, namely MSE (Mean Squared Error) and Mean Absolute Error (MAE).

MSE is defined as follows -

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(P_i - O_i)^2 \tag{3.2}$$

where $\boldsymbol{P_i}$ and $\boldsymbol{O_i}$ are the predicted and observed values, respectively.

MAE is defined as follows -

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|P_i - O_i| \tag{3.3}$$

where $\boldsymbol{P_i}$ and $\boldsymbol{O_i}$ are the predicted and observed values, respectively.

3.3.5.1 Artificial Neural Network

Table 3.1 contains the evaluation parameters and their corresponding values for river discharge and level.

	Discharge		Level	
	MSE	MAE	MSE	MAE
One-day lag	0.000802	0.01931	0.000184	0.02456
Two-day lag	0.0011	0.0228	0.00277	0.03251
Three-day lag	0.00152	0.0284	0.00317	0.0362

Table 3.1: Evaluation metrics for river discharge and level

Following sections contain six plots corresponding to each forecasting case. Among them, the first two are scatter plots between predicted and actual values of river discharge and level in which the data points near to (or on) the 45-degree line correspond to the most accurate predictions. The next two plots represent the actual and predicted time series data. Here, the section of the plot with the most overlapping corresponds to the most accurate predictions. The last two graphically display the change in values of MSE and MAE with epoch.

One day lag prediction

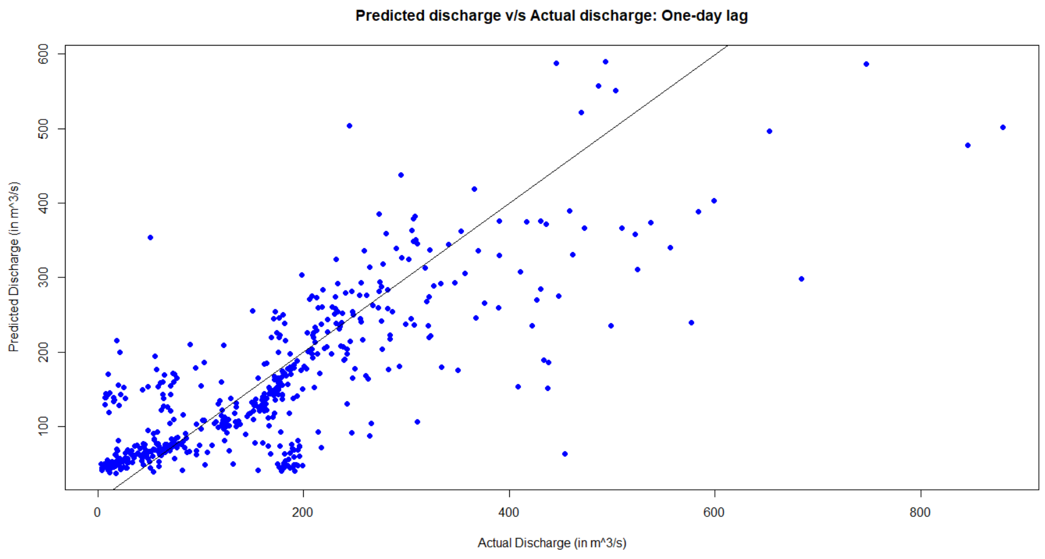


Figure 3.10: Predicted discharge versus Actual discharge

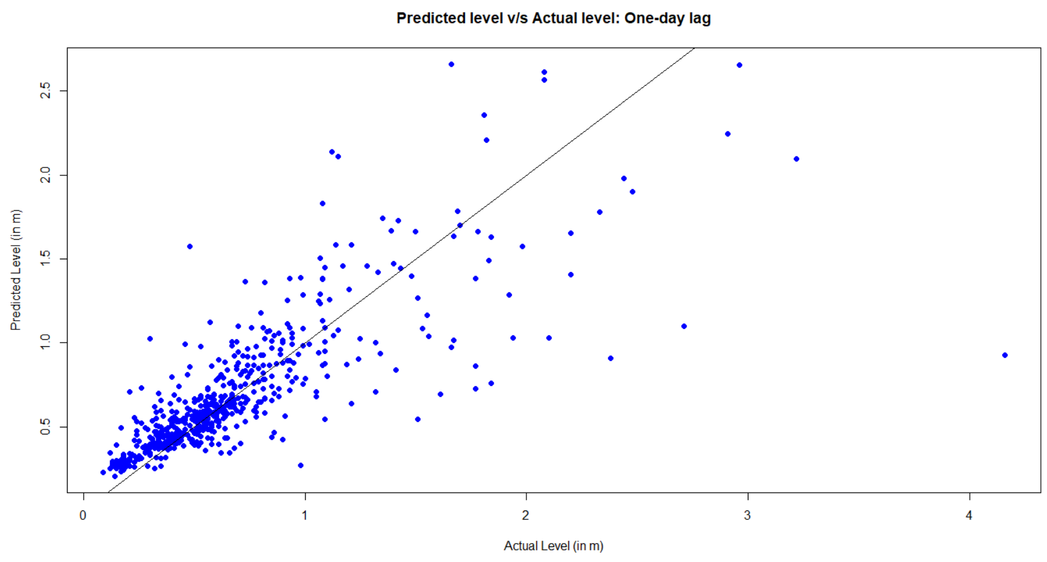


Figure 3.11: Predicted level versus actual level

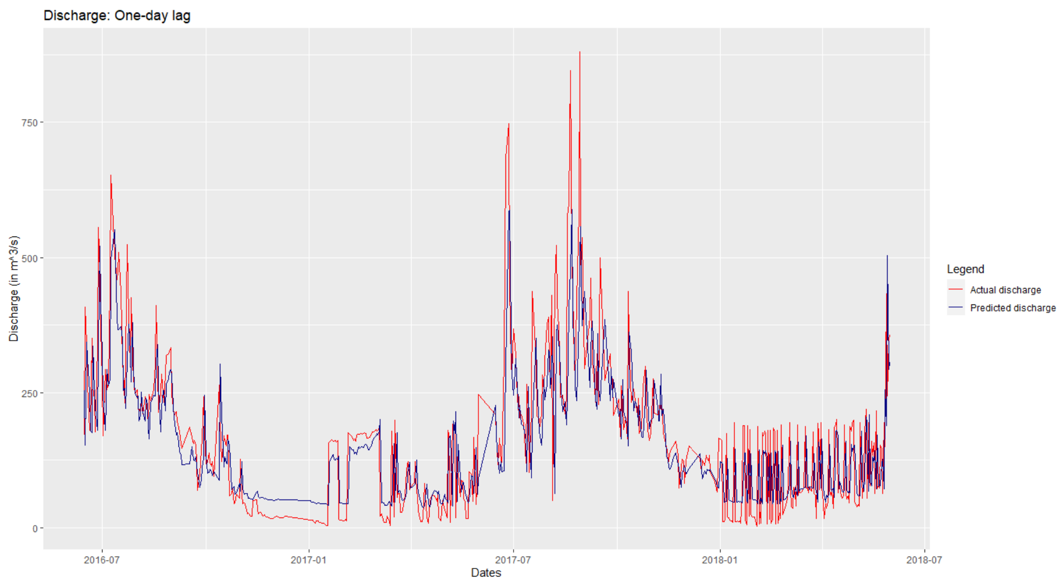


Figure 3.12: Test data - Discharge one-day lag

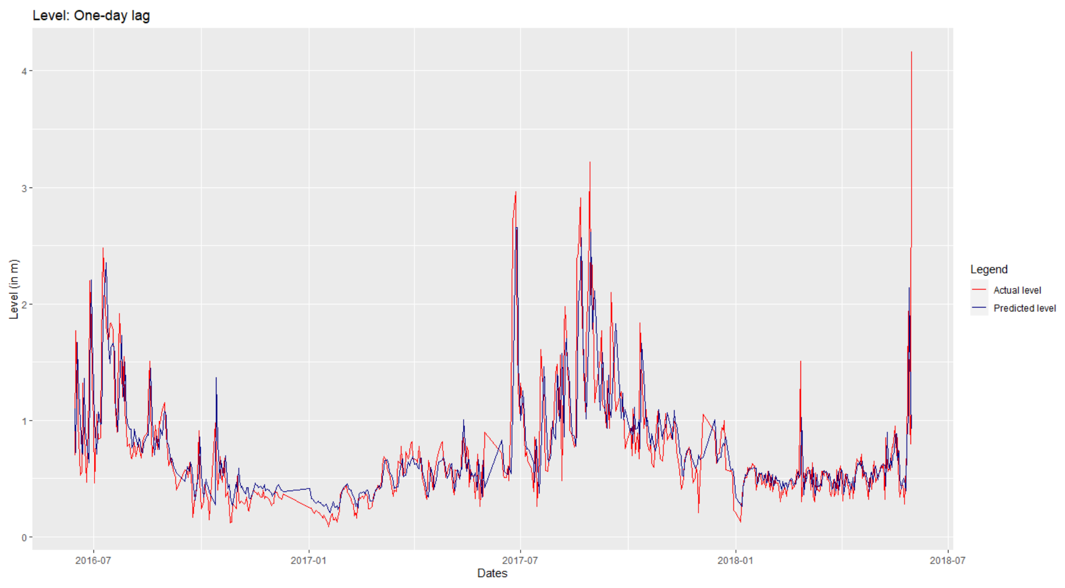


Figure 3.13: Test data - Level one-day lag

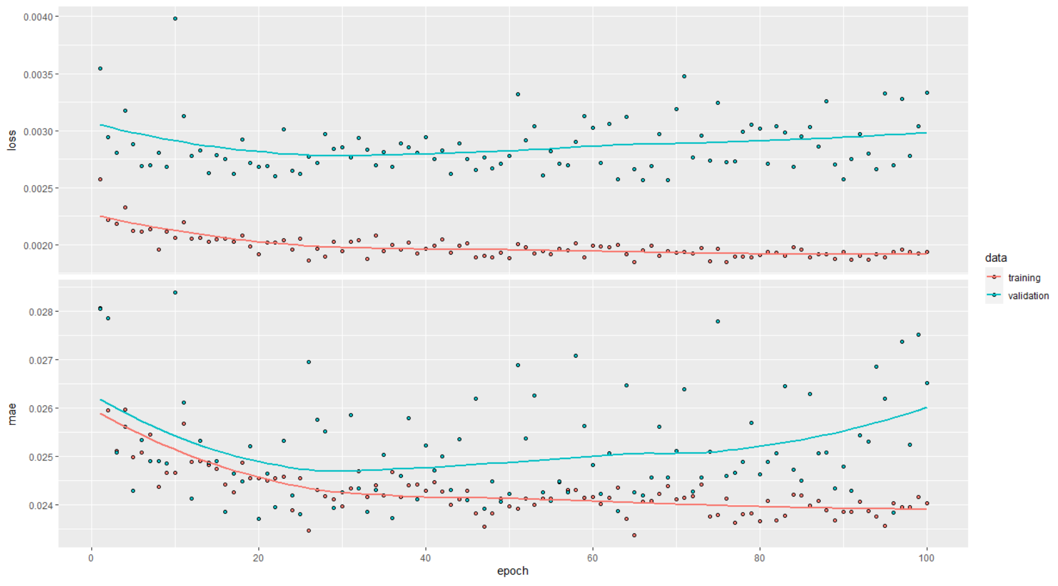


Figure 3.14: Evaluation metric versus epoch - Discharge one-day lag

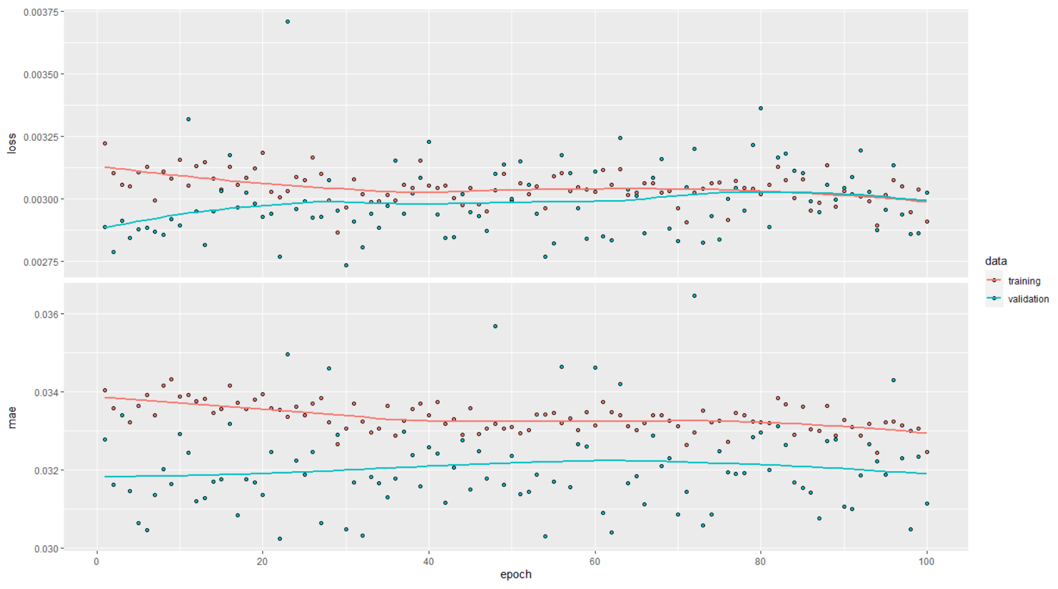


Figure 3.15: Evaluation metric versus epoch - Level one-day lag

Two days lag prediction

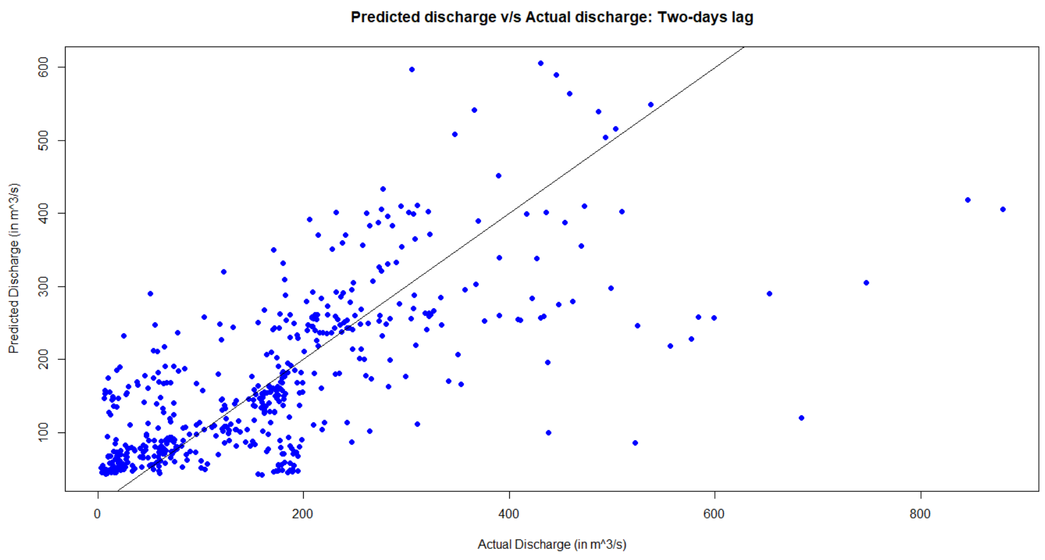


Figure 3.16: Predicted discharge versus actual discharge

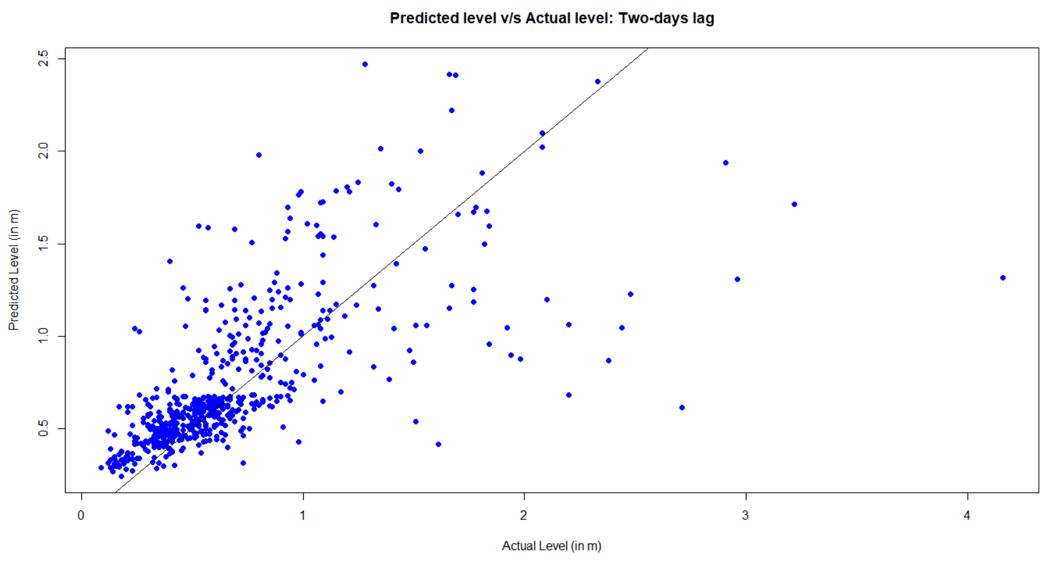


Figure 3.17: Predicted level versus actual level

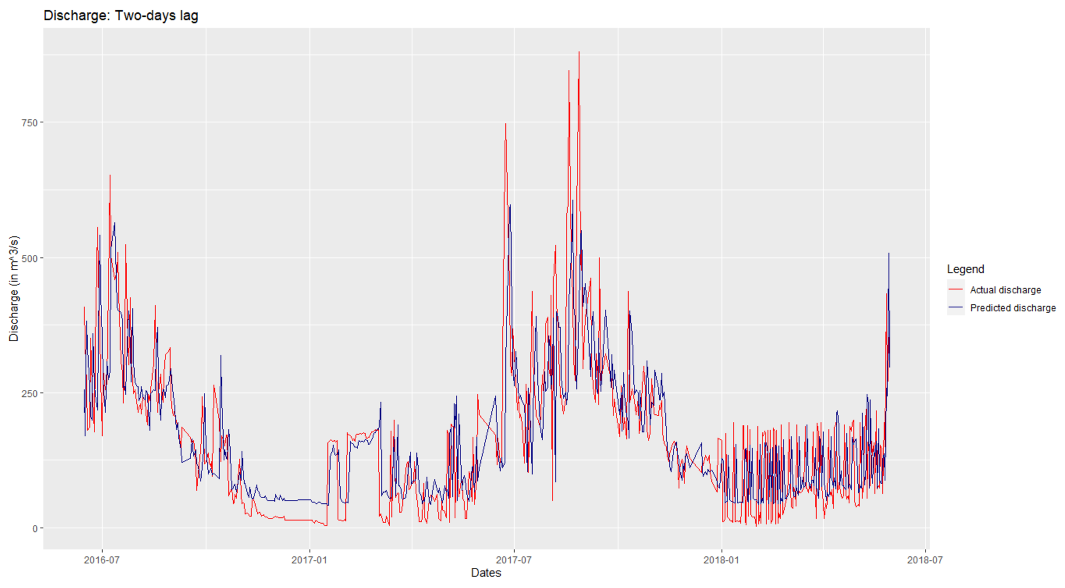


Figure 3.18: Test data - Discharge two-days lag

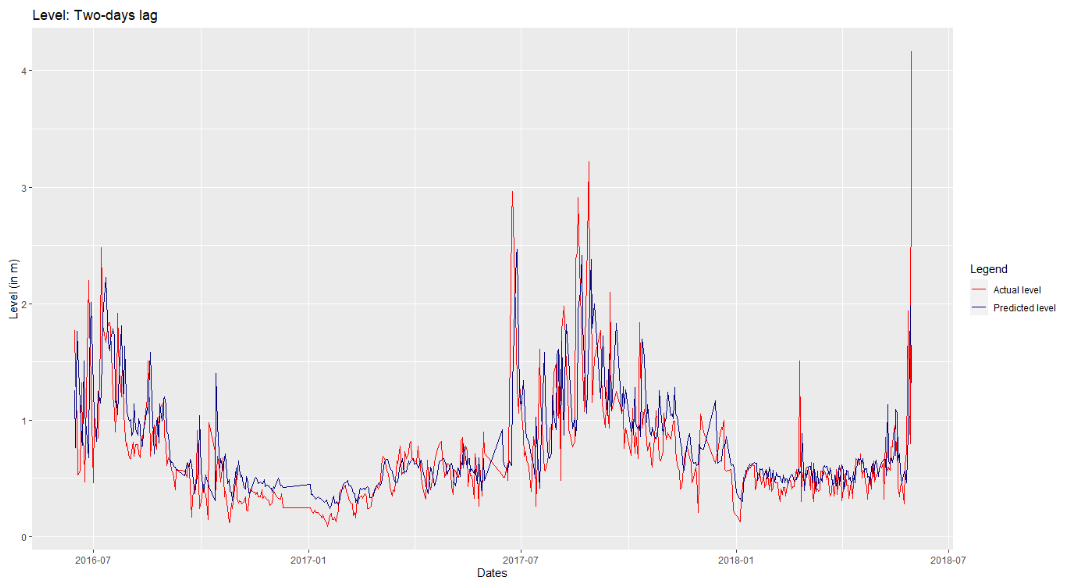


Figure 3.19: Test data - Level two-days lag

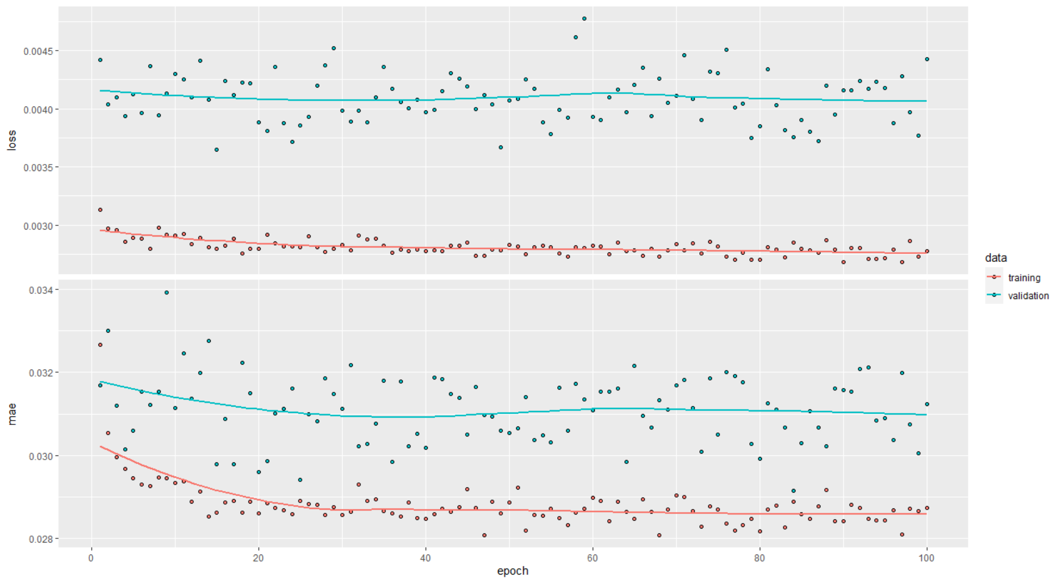


Figure 3.20: Evaluation metric versus epoch - Discharge two-days lag

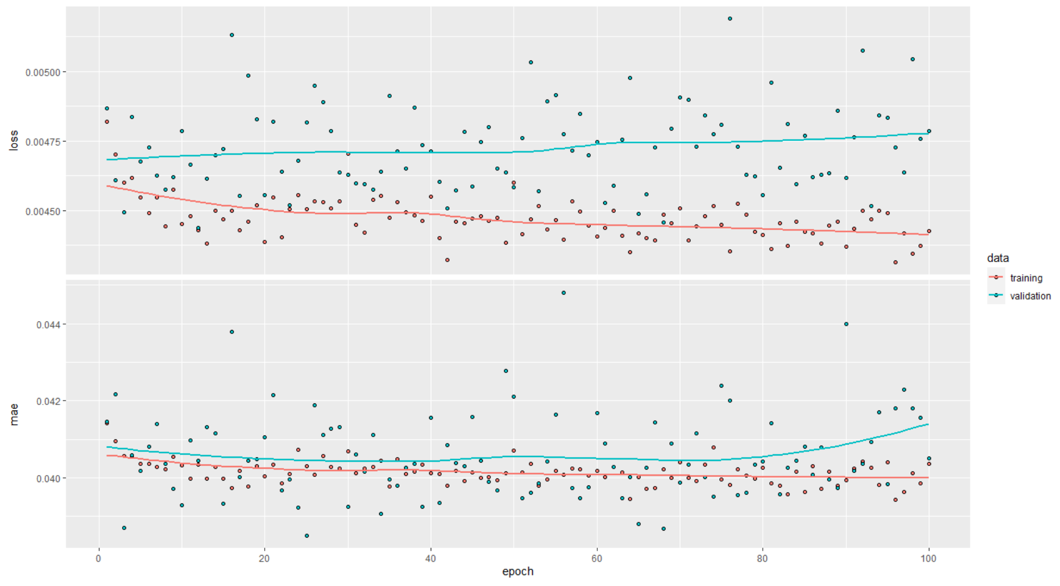


Figure 3.21: Evaluation metric versus epoch - Level two-days lag

Three days lag prediction

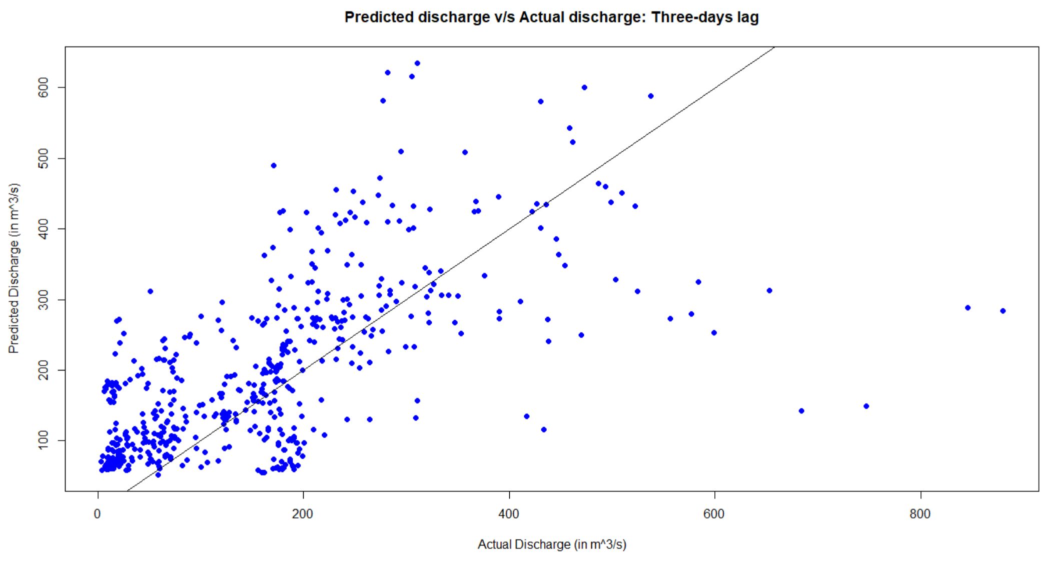


Figure 3.22: Predicted discharge versus actual discharge

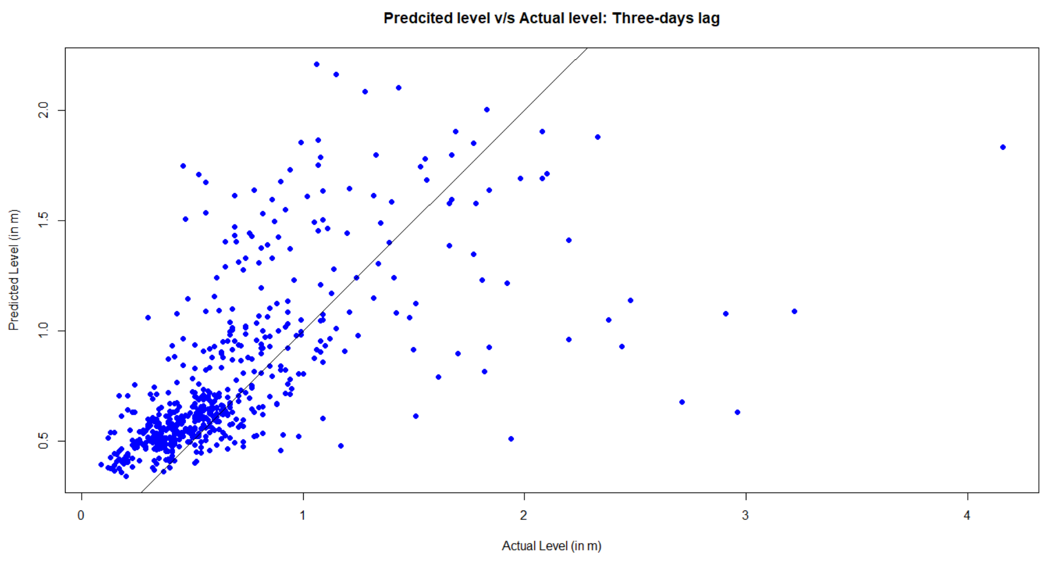


Figure 3.23: Predicted level versus actual level

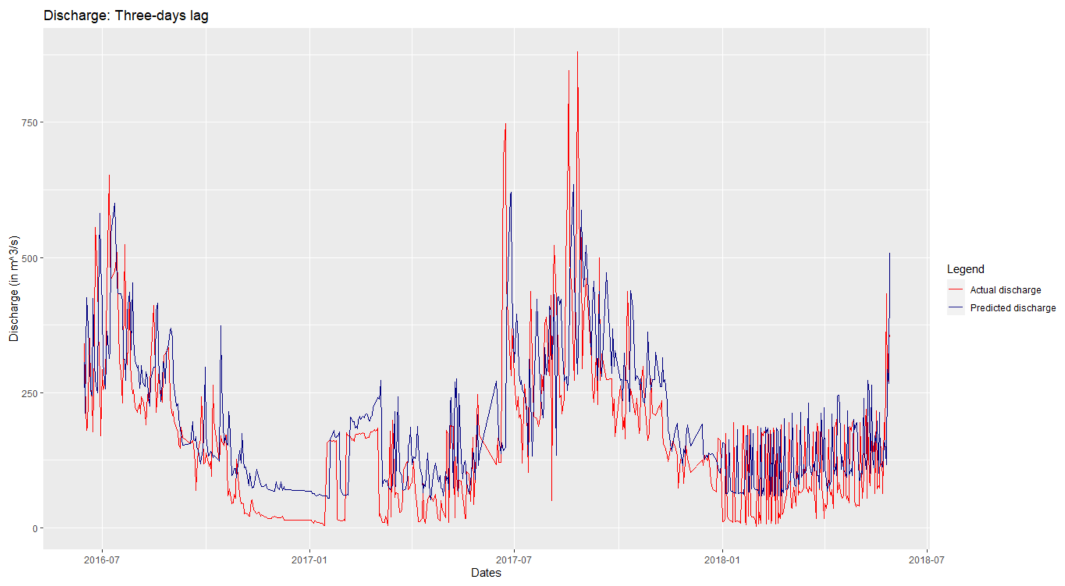


Figure 3.24: Test data - Discharge three-days lag

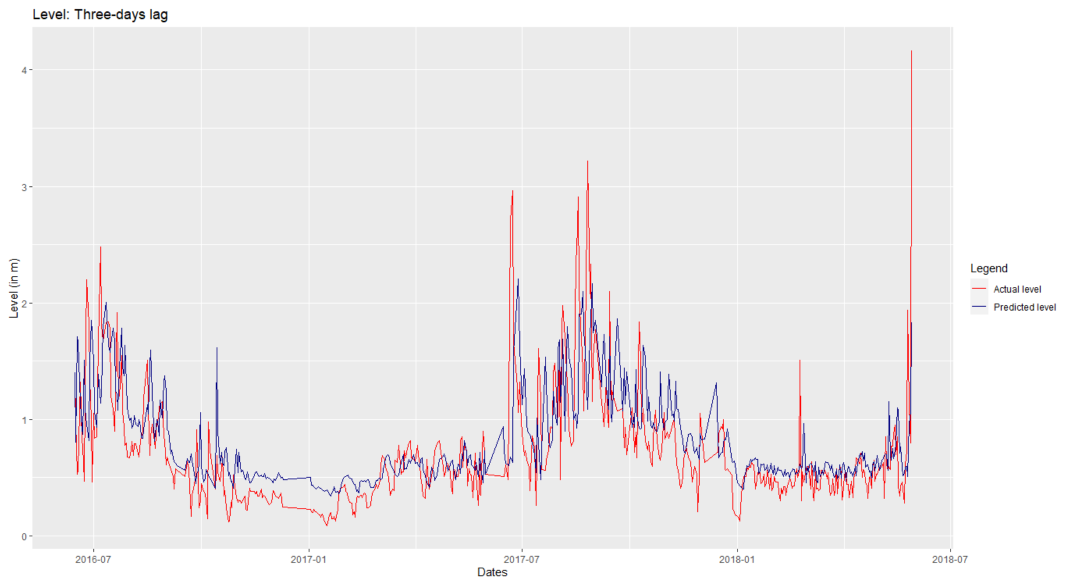


Figure 3.25: Test data - Level three-days lag

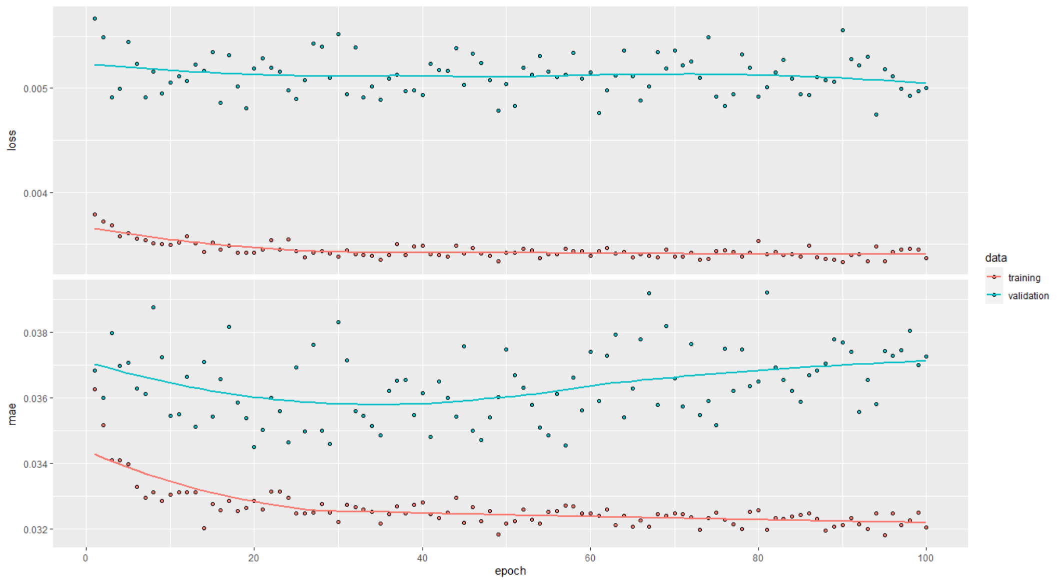


Figure 3.26: Evaluation metric versus epoch - Discharge three-days lag

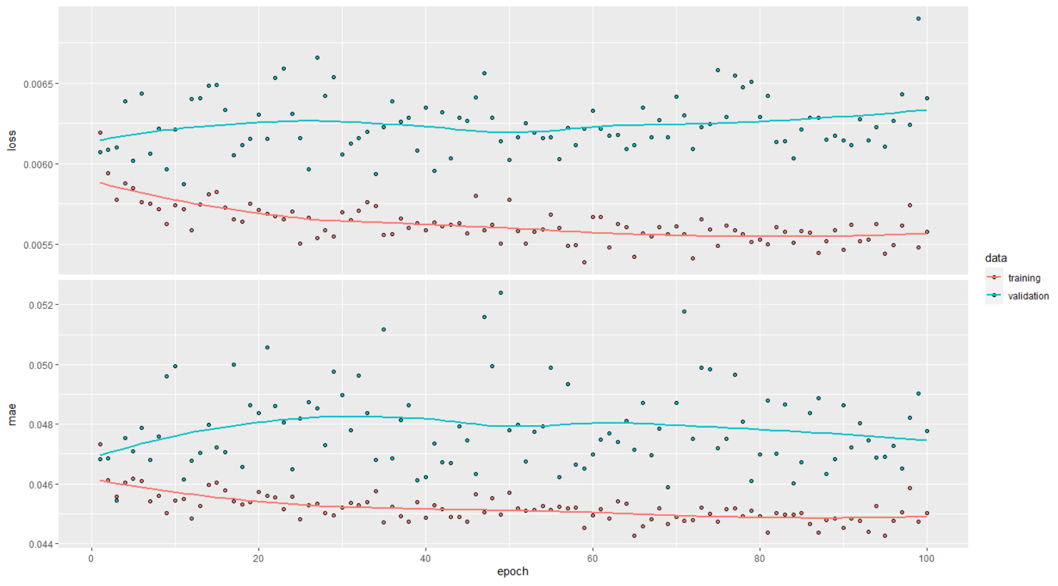


Figure 3.27: Evaluation metric versus epoch - Level three-days lag

3.3.5.2 Convolutional Neural Network

Table 3.2 contains the evaluation parameters and their corresponding values for river discharge and level.

	Discharge		Level	
	MSE	MAE	MSE	MAE
One-day lag	0.000872	0.02179	0.00194	0.0268
Two-day lag	0.00124	0.02581	0.00270	0.03135
Three-day lag	0.00138	0.0264	0.0032	0.0370

Table 3.2: Evaluation metrics for one-day lagged river discharge and level

Similar to ANN, here also the following sections contain six plots corresponding to each forecasting case. Among them, the first two are scatter plots between predicted and actual values of river discharge and level in which the data points near to (or on) the 45-degree line correspond to the most accurate predictions. The next two plots represent the actual and predicted time series data. Here, the section of the plot with the most overlapping corresponds to the most accurate predictions. The last two graphically display the change in values of MSE and MAE with epoch.

One day lag prediction

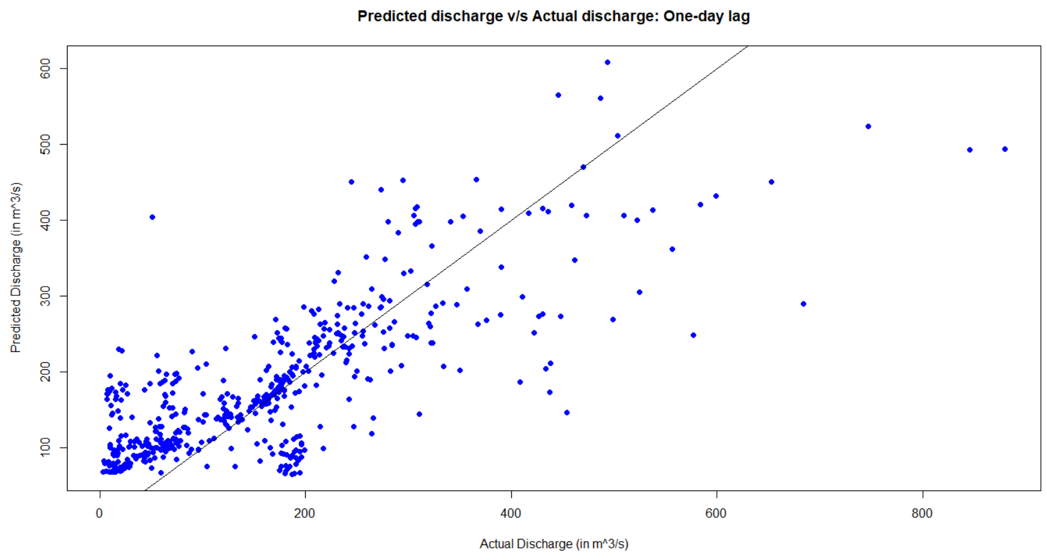


Figure 3.28: Predicted discharge versus actual discharge

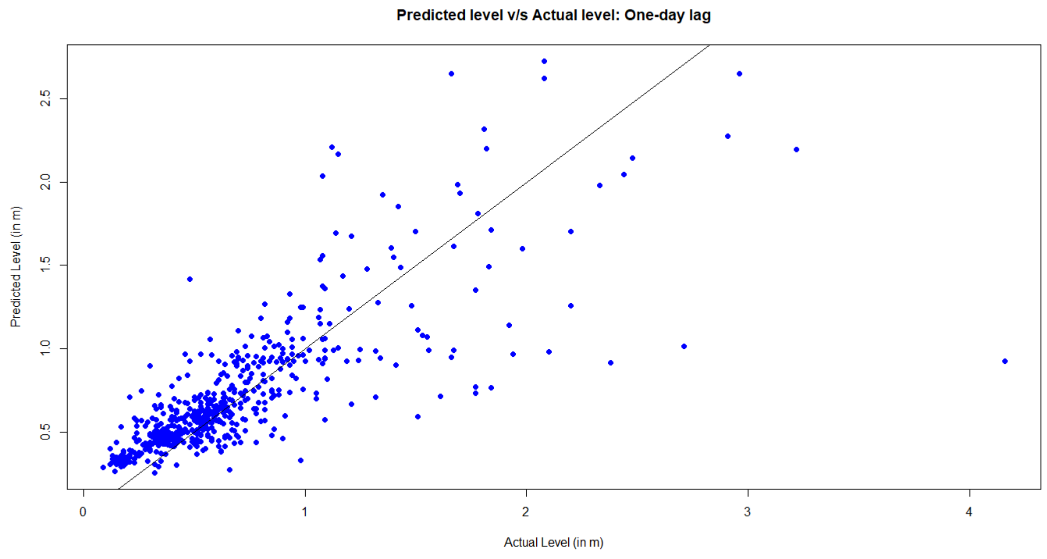


Figure 3.29: Predicted level versus actual level

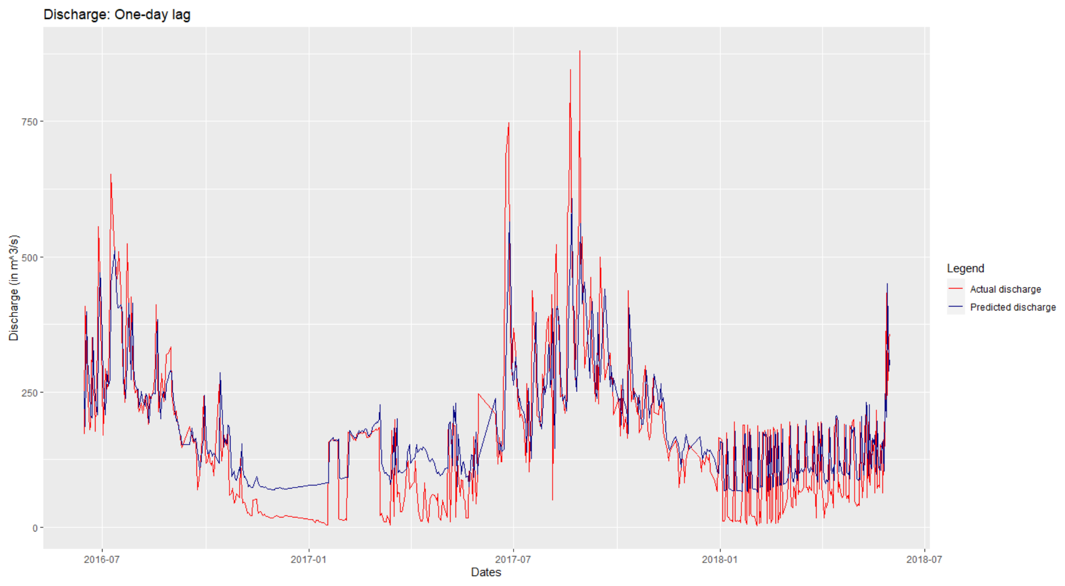


Figure 3.30: Test data - Discharge one-day lag

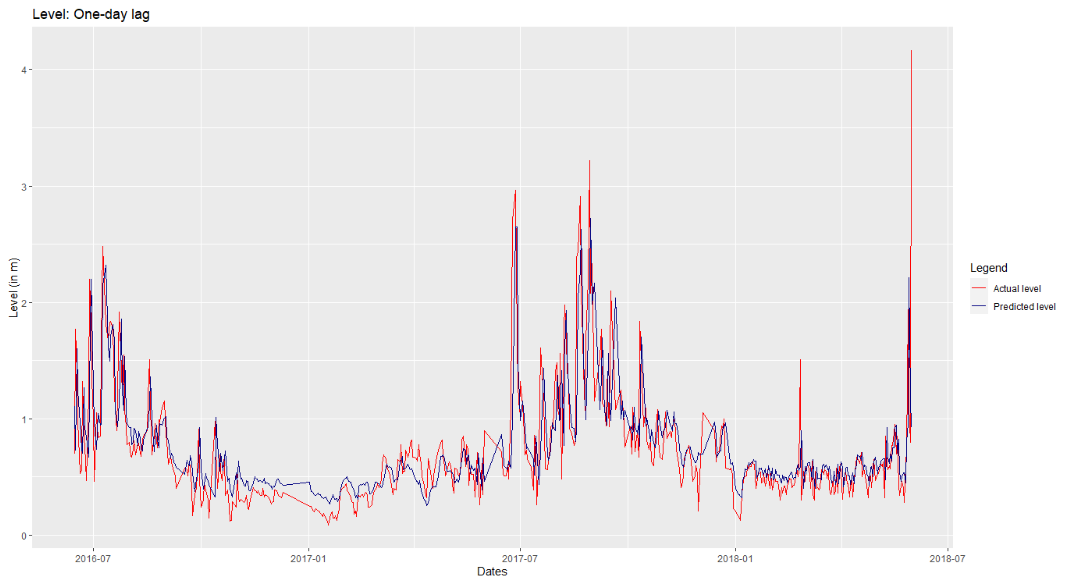


Figure 3.31: Test data - Level one-day lag

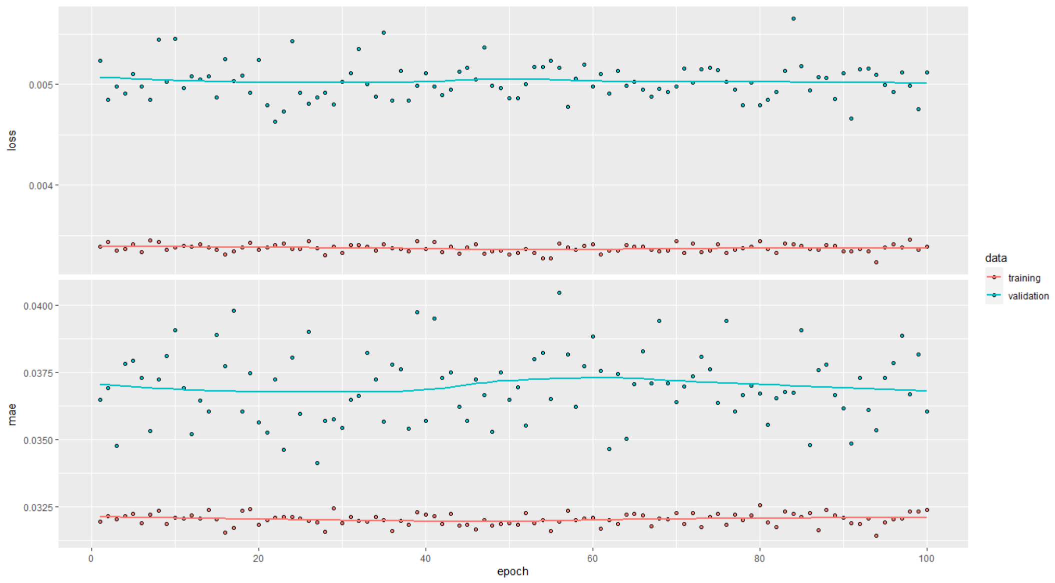


Figure 3.32: Evaluation metric versus epoch - Discharge one-day lag

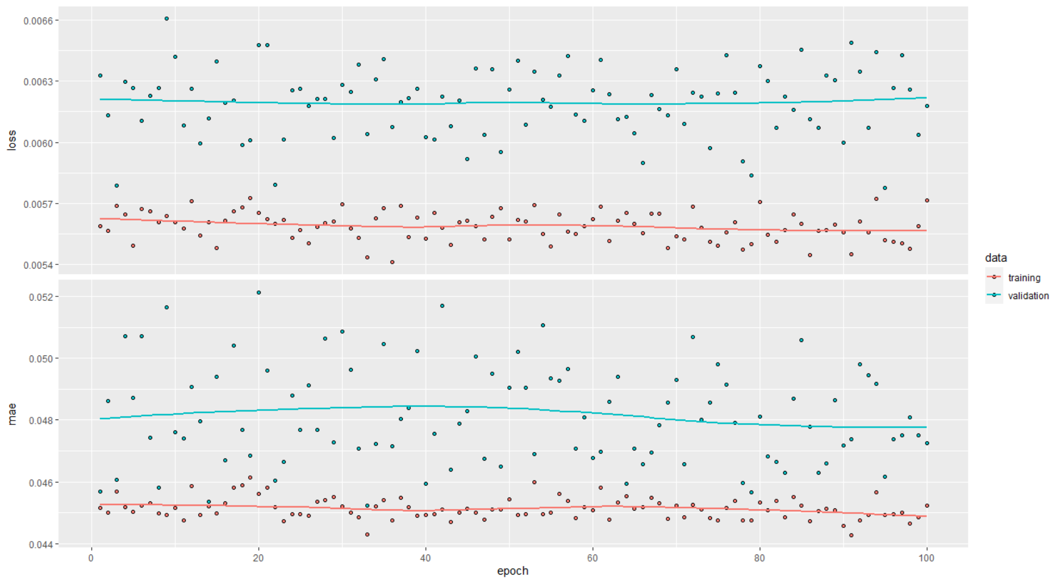


Figure 3.33: Evaluation metric versus epoch - Level one-day lag

Two days lag prediction

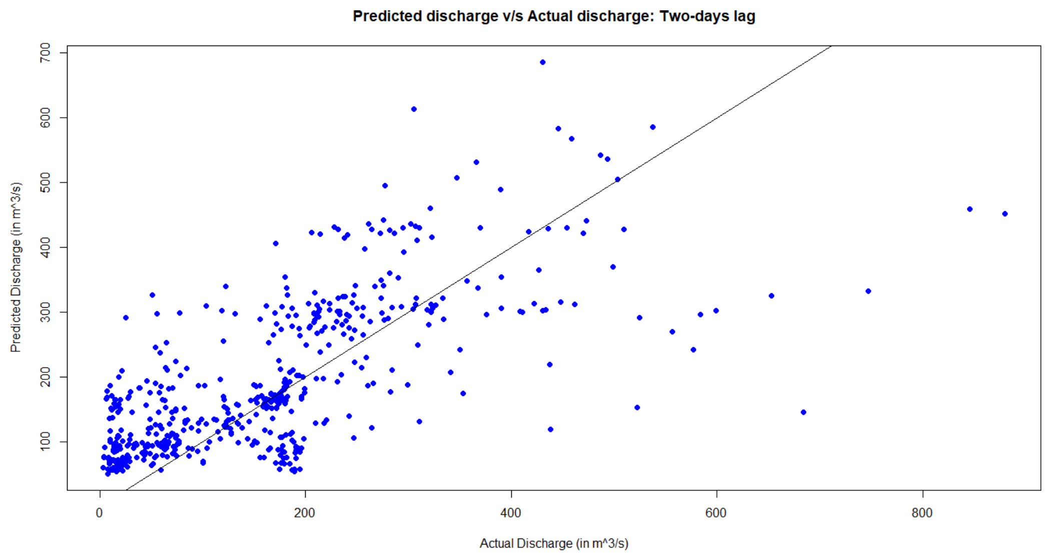


Figure 3.34: Predicted discharge versus actual discharge

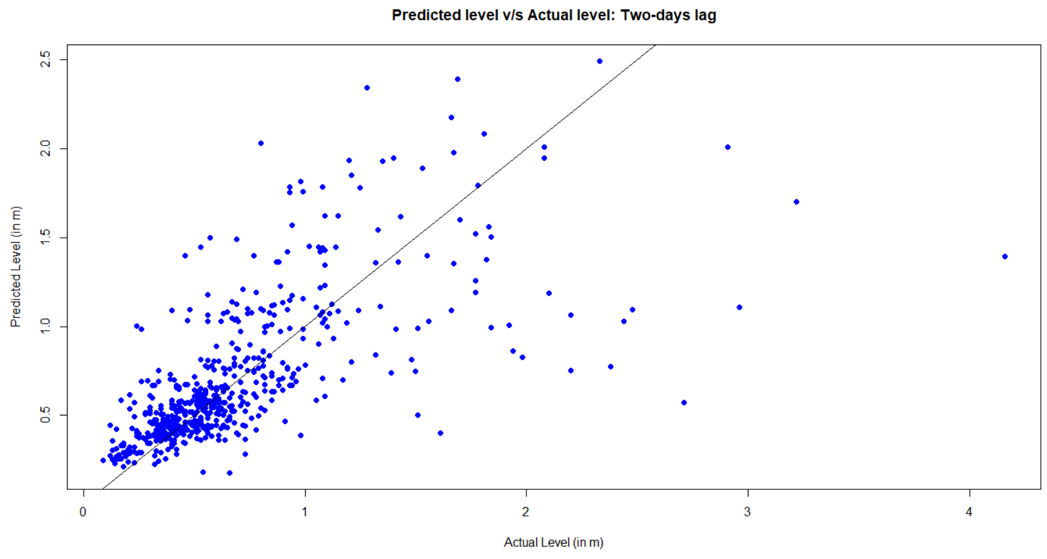


Figure 3.35: Predicted level versus actual level

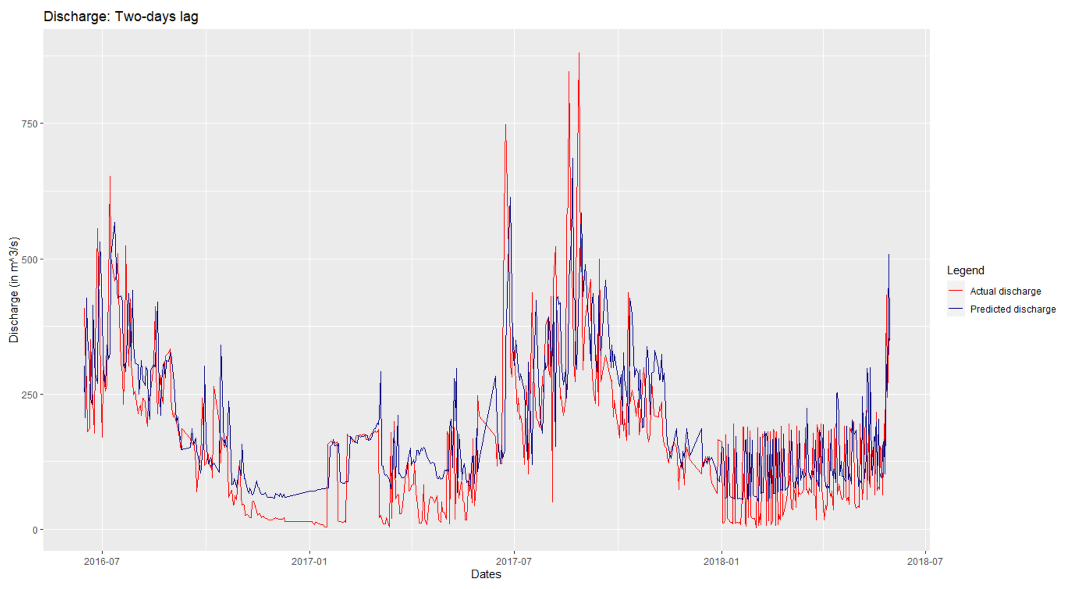


Figure 3.36: Test data - Discharge two-days lag

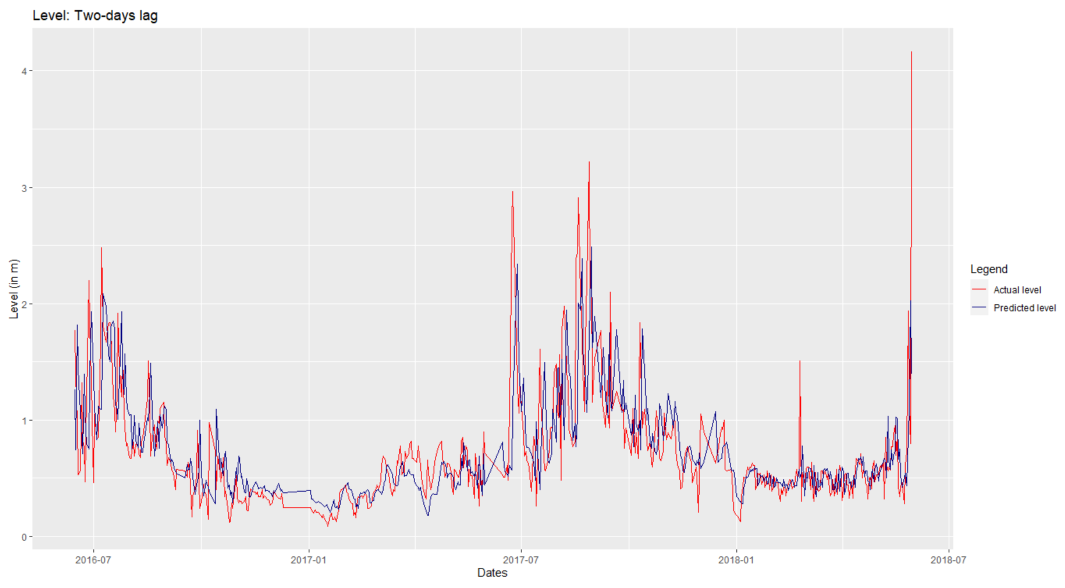


Figure 3.37: Test data - Level two-days lag

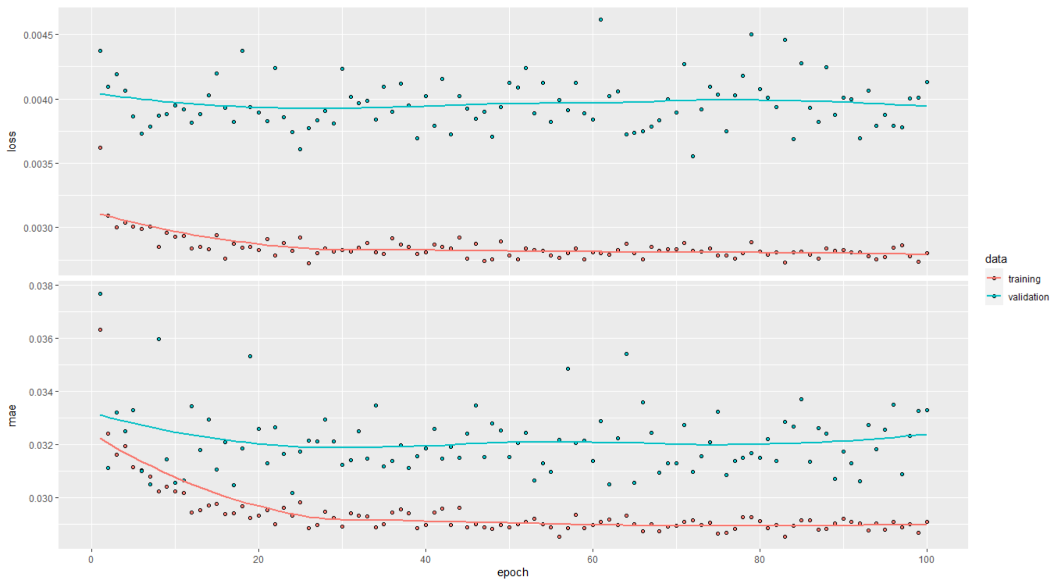


Figure 3.38: Evaluation metric versus epoch - Discharge two-days lag

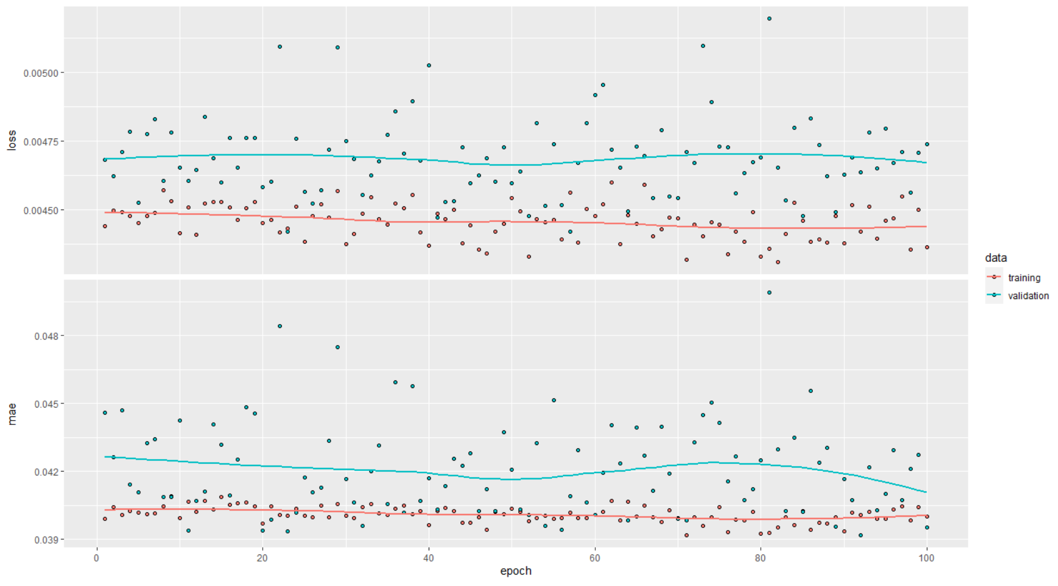


Figure 3.39: Evaluation metric versus epoch - Level two-days lag

Three days lag prediction

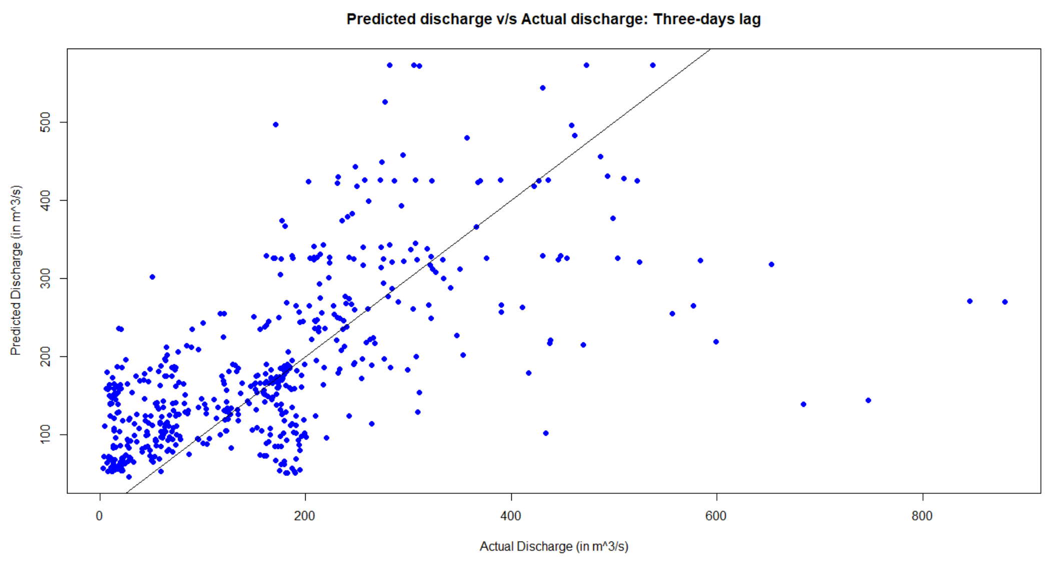


Figure 3.40: Predicted discharge versus actual discharge

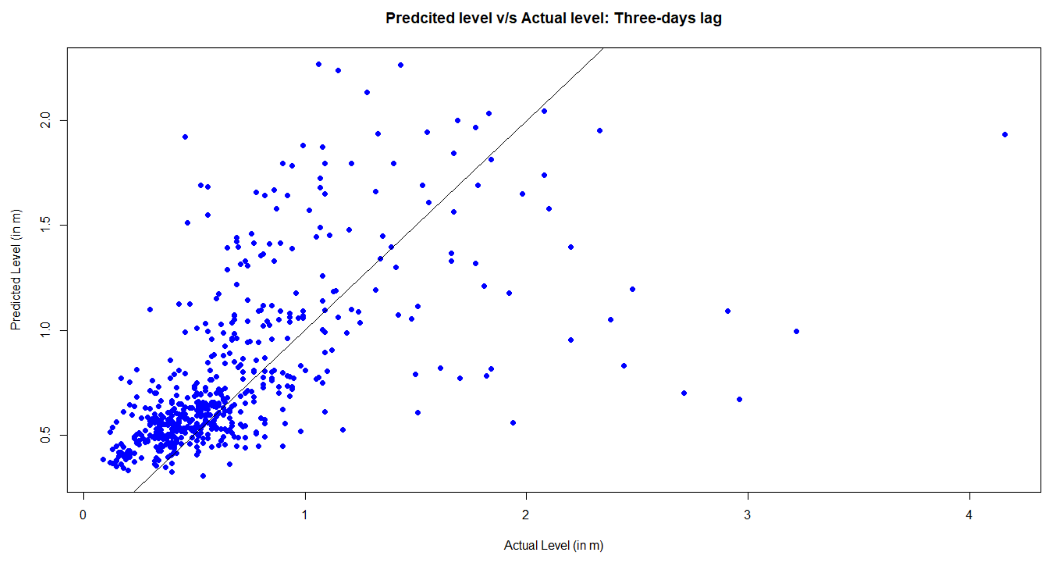


Figure 3.41: Predicted level versus actual level

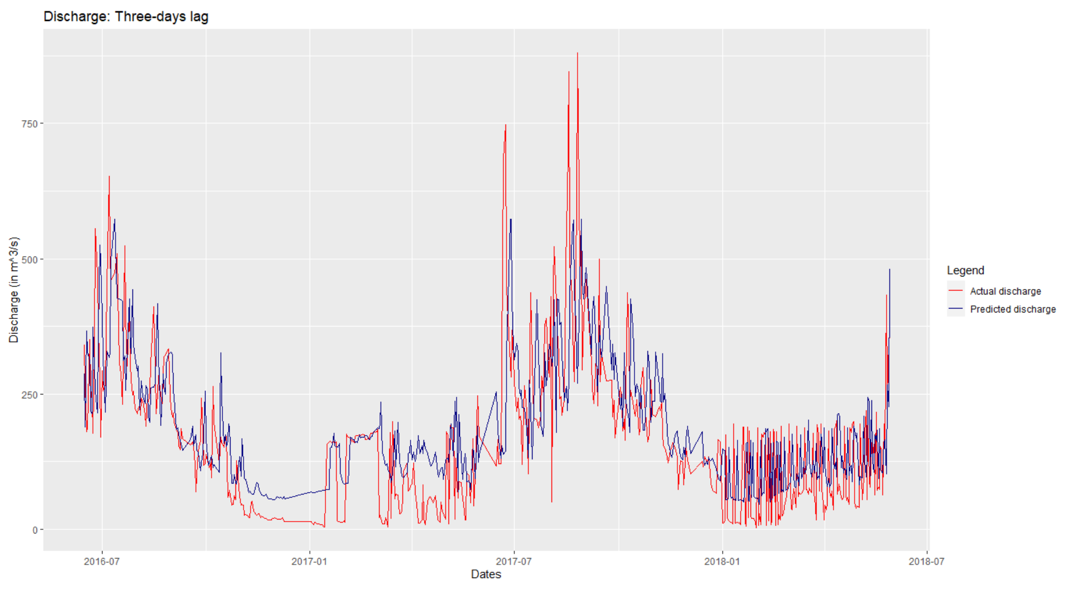


Figure 3.42: Test data - Discharge three-days lag

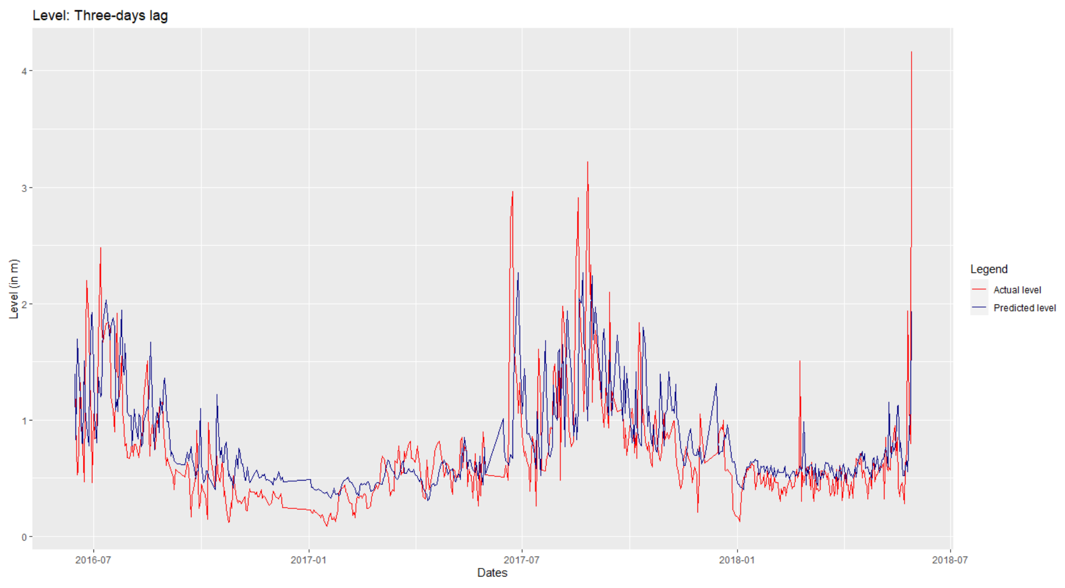


Figure 3.43: Test data - Level three-days lag

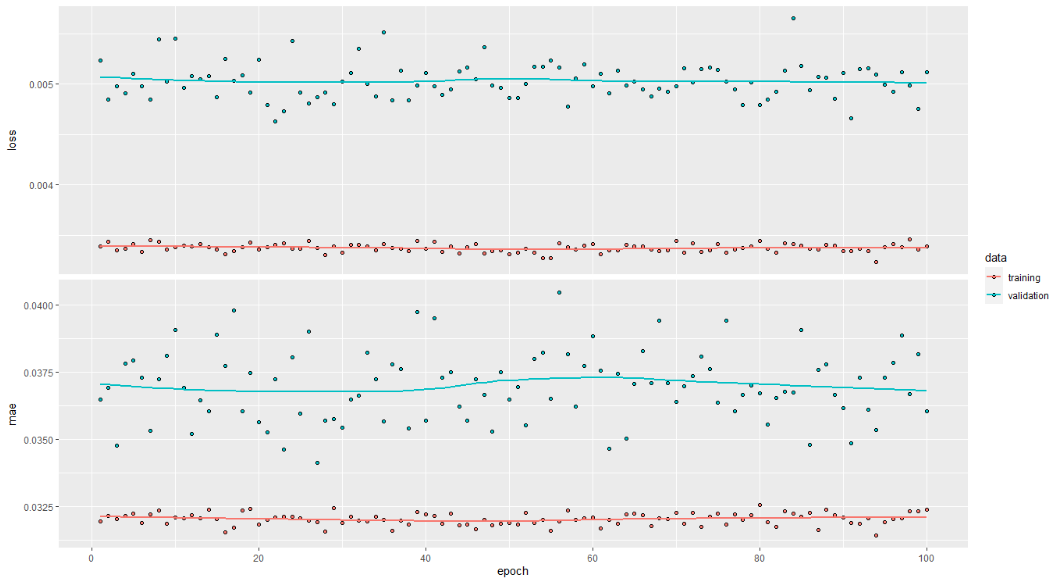


Figure 3.44: Evaluation metric versus epoch - Discharge three-days lag

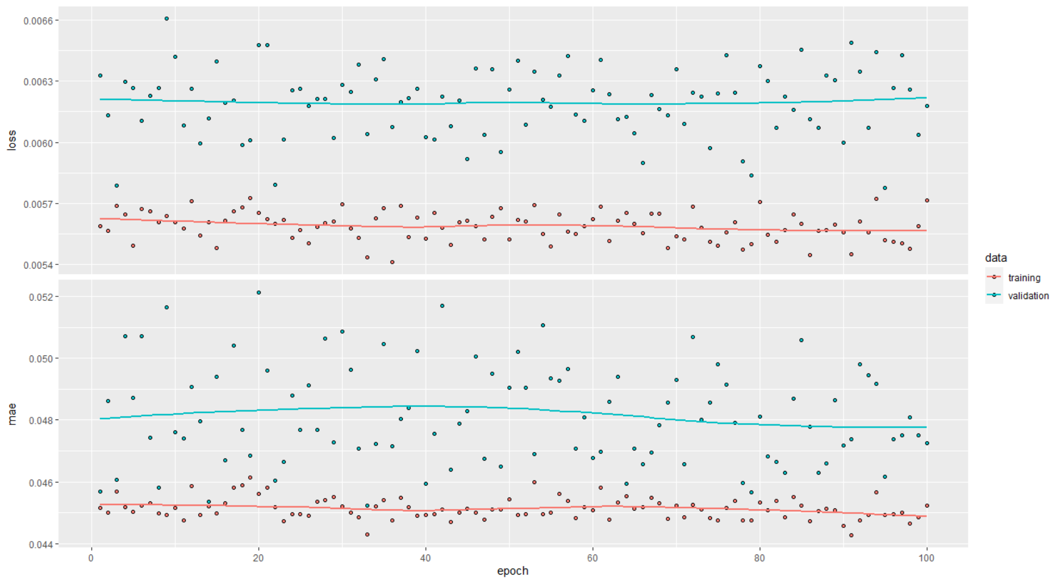


Figure 3.45: Evaluation metric versus epoch - Level three-days lag

Chapter 4

Conclusion

The projects completed during the fellowship have contributed towards the promotion of open source software, i.e. R. The tutorial scripts developed by each fellow will help to understand various R programming concepts with examples via audiovisual media. The river level and discharge prediction research project demonstrated the importance of data-driven models in predicting natural phenomenon. ANN performed slightly better than CNN in all three forecasting cases. Accurate forecasts may help people living in flood-prone areas by providing them with more time to evacuate during an emergency.

The FOSSEE fellowship was a great learning experience. Every fellow gained new skills and knowledge. The tasks performed gave an insight into a professional work environment. Fellows also learned the different facets of working within an organization. Also, the objective of contributing back to society via open source was a big motivator. In a nutshell, the fellowship taught each fellow work ethics, commitment, and the importance of contributing back to society besides technical skills.

References

[1] D. Newman, S. Hettich, C. Blake, and C. Merz, "Uci repository of machine learning databases," 1998.

[2] F. Leisch and E. Dimitriadou, "mlbench: Machine learning benchmark problems," 2010. R package version 2.1-1.

[3] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller, "proc: an open-source package for r and s+ to analyze and compare roc curves," *BMC Bioinformatics*, vol. 12, p. 77, 2011.

[4] M. Kuhn, "caret: Classification and regression training," 2020. R package version 6.0-86.

[5] N. R. Lauve, S. J. Nelson, S. S. Young, R. L. Obenchain, and C. G. Lambert, "Localcontrol: Nonparametric methods for generating high quality comparative effectiveness evidence," 2020. R package version 1.1.2.1.

[6] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, "e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071), tu wien," 2019. R package version 1.7-3.

[7] N. E. S. D. Program, "Kerala flood (2018)."

[8] Kerala Irrigation Department, "Periyar."

[9] D. of Water Resources, "Periyar river data."

[10] S. H. Elsafi, "Artificial neural networks for flood forecasting at dongola station in the river nile, sudan," *Alexandria Engineering Journal*, vol. 53, no. 3, pp. 655 – 662, 2014.

[11] X.-H. Le, H. V. Ho, G. Lee, and S. Jung, "Application of long short-term memory neural network for flood forecasting," *Water*, vol. 11, no. 7, p. 1387, 2019.

[12] Brownlee, J., "Deep learning for time series forecasting: Predict the future with mlps, cnns and lstms in python," 2018.

[13] Beysolow, Taweh, "Introduction to deep learning using r: A step-by-step guide to learning and implementing deep learning models using r," 2017.